



Faculty of Computer Science and Information Technology

***AI-POWERED PACKAGE THEFT DETECTION SYSTEM WITH REAL-
TIME MONITORING AND ALERTS***

Rahul Kumar Shawn A/L Rajesh Kumar

Bachelor of Computer Science with Honours (Network Computing)

2025

**AI-POWERED PACKAGE THEFT DETECTION SYSTEM WITH REAL-TIME
MONITORING AND ALERTS**

RAHUL KUMAR SHAWN A/L RAJESH KUMAR

This project is submitted in partial fulfilment of the
requirements for the degree of
Bachelor of Computer Science with Honours (Network Computing)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

**SISTEM PENGESANAN KECURIAN BUNGKUSAN BERASASKAN AI DENGAN
PEMANTAUAN DAN AMARAN MASA NYATA**

RAHUL KUMAR SHAWN A/L RAJESH KUMAR

Project ini merupakan salah satu keperluan untuk
Ijazah Sarjana Muda Sains Komputer dan Teknologi Maklumat
(Pengkomputeran Rangkaian)

Fakulti Sains Komputer dan Teknologi Maklumat
UNIVERSITI MALAYSIA SARAWAK

2025

DECLARATION

I, Rahul Kumar Shawn A/L Rajesh Kumar with the matric number of 82102, hereby solemnly declare that the report titled '**AI-Powered Package Theft Detection System With Real-Time Monitoring And Alerts**' is prepared and completed authentically by me in partial fulfilment of the requirements for the degree of Bachelor of Computer Science with Honours (Network Computing) is a record of an original work carried out by me under the guidance of Associate Professor Dr. Johari bin Abdullah. I further declare that the work reported in this project has not been previously or concurrently submitted for any other degree at UNIVERSITI MALAYSIA SARAWAK (UNIMAS) or any other institutions. I declare that I have not copied or plagiarize from any other sources except for quotations and citations which have been duly acknowledged.



RAHUL KUMAR SHAWN A/RAJESH KUMAR (82102)
Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK

ACKNOWLEDGEMENT

Trust the process. It is an honour and privilege to produce a report that aims to provide solutions and enhance the well-being of society. The preparation of this report was truly above and beyond with the assistance and help of individuals of such calibre. I would like to express my gratitude to all those who made my Final Year Project Successful, by providing me with the ability, courage, compassion and words of wisdom to effectively complete the project within this compact time frame.

First and foremost, I would like to sincerely put forward my words and wishes of gratitude and the depth of gratefulness to my Final Year Project supervisor, Associate Professor Dr. Johari bin Abdullah, who had unwavering belief in my capabilities, provision of necessary guidance, supervision and fortitude of best advice continuously throughout my Final Year Project. His compassion, wit, calm and calculated nature in providing valuable feedback and insights have been instrumental in completing this project successfully.

Furthermore, I would like to express my earnest appreciation to my examiner, Dr Tan Ping Ping, for providing feedback and patience on my Final Year Project. Her continuous patience has been crucial in ensuring that the quality produced for my project is up to par and optimum.

Other than that, I would like to extend my gratitude to Professor Dr. Wang Yin Chai, as the Final Year Project coordinator who had provided useful guidelines and tips in accomplishing my Final Year Project within the set timeframe of submission.

With wholehearted regards to the respondents, my sincere appreciation and deepest regards for their time and feedback on the questionnaire, which has seen to play a vital role in gathering user requirements for my Final Year Project.

Apart from that, I would like to wholeheartedly express my appreciation and thankfulness to my fellow friends and course mates who provided invaluable assistance during the preparation of this report.

Lastly, my deepest love, gratitude and compassion to my parents for their unwavering support, encouragement, support and undeniable fortitude of belief in me. Without their support, I would not have the capability of maintaining my perseverance, drive, passion and commitment to successfully produce, complete and finalize the project.

ABSTRACT

The rise of e-commerce has led to a significant increase in package theft, posing serious challenges for consumers and undermining their shopping experience. This project presents the development of an AI-Powered Package Theft Detection System with Real-Time Monitoring and Alerts, designed to effectively combat the growing issue of package theft or porch piracy. By leveraging advanced artificial intelligence technologies which includes object detection, the system is capable of continuously monitoring package deliveries, identifying potential threats, and providing immediate alerts to homeowners when suspicious activities are detected. This proactive approach not only enhances the security of delivered packages but also empowers consumers to take timely action against theft. The significance of this project lies in its potential to restore consumer confidence in the safety of online shopping. By combining computer vision technology by integrating, it into traditional surveillance systems, it addresses the limitations of traditional surveillance systems, which often fail to provide real-time alerts or proactive security measures, this AI-powered solution offers a more effective means of preventing package theft or porch piracy. The findings found out that with the tests of 20 test videos, across 94 detection rates, resulted in a detection rate of over 80% of theft detection, in turn displays room for improvement but initially heading in the projected direction.. Ultimately, this project aims to contribute to a safer delivery environment, ensuring that package owners feel secure in their purchases and can maintain their engagement with online shopping platforms.

ABSTRAK

Penambahan jumlah pengguna platform e-dagang secara mendadak telah membawa kepada peningkatan ketara dalam kes kecurian bungkusan, ini menimbulkan cabaran serius kepada pengguna dan merosakkan pengalaman membeli-belah mereka. Projek ini mempamerkan pembuatan Sistem Pengesanan Kecurian Bungkusan Berkuasa AI dengan Pemantauan Masa Nyata dan Notifikasi, yang direka untuk menangani isu kecurian bungkusan atau "porch piracy" yang semakin meningkat. Dengan menggunakan teknologi kecerdasan buatan (AI) yang canggih, termasuk pengesanan objek, sistem ini mampu memantau penghantaran bungkusan secara berterusan, mengenal pasti ancaman berpotensi, dan memberikan pemberitahuan segera kepada pemilik rumah apabila aktiviti mencurigakan dikesan. Pendekatan proaktif ini bukan sahaja meningkatkan keselamatan bungkusan yang dihantar, tetapi juga memberi kuasa kepada pengguna untuk mengambil tindakan segera terhadap kecurian. Kepentingan projek ini terletak pada kebolehan untuk memulihkan keyakinan pengguna terhadap keselamatan pembelian dalam talian. Dengan menggabungkan teknologi visi komputer ke dalam sistem pengawasan tradisional, projek ini mengatasi kekangan sistem pengawasan tradisional yang sering gagal menyediakan pemberitahuan masa nyata atau langkah keselamatan proaktif. Penyelesaian berkuasa AI ini menawarkan kaedah yang lebih berkesan untuk mencegah kecurian bungkusan atau "porch piracy." Kajian ini menunjukkan Keputusan selepas penggunaan 20 rakaman ujian yang merangkumi 94 kadar pengesanan sekaligus memaparkan keputusan kecekapan melebihi 80% kadar pengesanan kecurian bungkusan. Sistem ini dapat mengesan kejadian langsung seperti pengusikan atau pengalihan bungkusan, dan seterusnya memberi notifikasi kepada pengguna. Akhirnya, projek ini bertujuan untuk menyumbang kepada persekitaran penghantaran yang lebih selamat, memastikan pemilik bungkusan merasa yakin dengan pembelian mereka dan dapat terus melibatkan diri dengan platform membeli-belah dalam talian.

TABLE OF CONTENTS

DECLARATION	i
ACKNOWLEDGEMENT	i
ABSTRACT.....	iii
ABSTRAK.....	iv
LIST OF FIGURES	x
LIST OF TABLES.....	xiv
LIST OF ABBREVIATIONS.....	xv
CHAPTER 1: INTRODUCTION	1
1.1 Project Title.....	1
1.2 Background.....	1
1.3 Problem Statement.....	4
1.4 Objectives	5
1.5 Brief Methodology.....	5
1.6 Scopes	7
1.7 Significance of Project.....	7
1.8 Project Schedule.....	8
1.9 Expected Outcome.....	9
1.10 Project Outline	9
CHAPTER 2: LITERATURE REVIEW	11
2.1 Introduction.....	11
2.2 Review on Similar Existing Systems.....	11
2.2.1 CVEDIA Package Detection.....	12
2.2.2 Deep Sentinel	22
2.2.3 Vession: Gesture Recognition for real-time security in video surveillance.....	25
2.2.4 Comparison of Features between Systems.....	27
2.3 Review of Tools and Technologies.....	30
2.3.1 Software	30
2.3.1.1 Roboflow	30
2.3.1.2 VS Code (Flask)	35
2.3.1.3 Web Development Technologies.....	36
2.3.1.4 MongoDB	38
2.3.1.5 Docker (Docker Desktop).....	40
2.3.1.6 Figma	42
2.3.2 Hardware.....	43

2.3.2.1 HP 14s-dq2515tu	43
CHAPTER 3: REQUIREMENT ANALYSIS AND DESIGN	45
3.1 Introduction.....	45
3.2 Scrum.....	46
3.2.1 Scrum Roles	46
3.2.2 Scrum Events.....	47
3.2.3 Scrum Artifacts	48
3.2.4 Agile Methodology	49
3.3 Requirements	50
3.3.1 Analysis on Similar Existing Systems	50
3.3.2 Analysis of Proposed System.....	51
3.3.3 Questionnaire	52
3.3.3.1 Respondents Demographic	53
3.3.3.2 Current Concerns About Package Theft.....	56
3.3.3.3 Preference for a Smart Package Detection System	58
3.3.3.4 User Experience and Feedback.....	62
3.3.4 Software Requirements	65
3.3.5 Hardware Requirements.....	66
3.4 Design	66
3.4.1 Scenario Illustration	67
3.4.1.1 Scenario 1 – Overview and Initialization (Live Monitoring).....	67
3.4.1.2 Scenario 2 – Detection and Response to Stimuli (Event Trigger).....	68
3.4.2 Web Application and Software Architecture	69
3.4.3 Flow Chart Diagram.....	71
3.4.4 Context Diagram: Level 0 Data Flow Diagram (DFD).....	73
3.4.5 Data Flow Diagram (DFD)	74
3.4.5.1 Level 1 Data Flow Diagram (DFD).....	74
3.4.5.2 Level 2 DFD	76
3.4.6 Entity Relationship Diagram (ERD)	81
3.4.7 Data Dictionary.....	83
3.4.7.1 Data Dictionary for User Collection.....	83
3.4.7.2 Data Dictionary for Detection Collection.....	84
3.4.7.3 Data Dictionary for Report Collection	85
3.4.8 High-fidelity Wireframe Design	86
3.4.8.1 Web Application Wireframe for SnapSecure Landing Page.....	86

3.4.8.2 Web Application Wireframe for SnapSecure Login Page.....	87
3.4.8.3 Web Application Wireframe for SnapSecure Create Account Page	88
3.4.8.4 Web Application Wireframe for SnapSecure Dashboard.....	89
3.4.8.5 Web Application Wireframe for SnapSecure Records Page	90
3.4.8.6 Web Application Wireframe for SnapSecure Archive Page	92
3.4.8.7 Web Application Wireframe for SnapSecure Action Page	93
3.4.8.8 Web Application Wireframe for SnapSecure Profile Page	94
3.4.8.9 Web Application Wireframe for SnapSecure Edit Profile Page.....	95
3.5 Development	95
3.6 Testing.....	96
3.7 Deployment.....	97
3.8 Review	97
3.9 Summary	98
CHAPTER 4: IMPLEMENTATION	99
4.1 Introduction.....	99
4.2 Feature Implementation	99
4.2.1 Installation and Setup of Visual Studio (VS) Code.....	100
4.2.1.1 Python Extension Installations	104
4.2.2 Python and Essential Dependency Setup	107
4.2.2.1 System Python Version 3.12.8 Installation	107
4.2.2.2 Virtual Environment Setup	110
4.2.2.3 Virtual Environment Dependency installation	111
4.2.3 Workspace Setup.....	112
4.2.3.1 Frontend Setup.....	112
4.2.3.2 Backend Setup	113
4.2.3.3 Overall Workspace Setup	114
4.2.4 MongoDB Setup.....	118
4.2.4.1 MongoDB Account Creation.....	118
4.2.4.2 Cluster Creation.....	121
4.2.4.2 Connecting to Cluster	122
4.2.4.3 Database and Collection Creation	124
4.2.4 Docker Desktop Setup.....	126
4.2.4.1 Docker Desktop Installation	126
4.2.4.2 Docker Desktop Account Registration/Login	129
4.3 System Hierarchy.....	131

4.4 Workflow of the Proposed System	133
4.4.1 Detection Workflow	133
4.4.2 Notification Workflow	134
4.4.3 Verification Workflow	134
4.4.4 Reporting Workflow	134
4.5 Coding of Algorithms Proposed	135
4.5.1 Package Theft Detection Algorithm.....	135
4.5.1.1 Overview of the Core of Object Detection in The Proposed System.....	135
4.5.1.2 Video Source Implementation in SnapSecure Package Detection System	141
4.6 Proposed System Implementation.....	147
4.6.1 Proposed Web Application (SnapSecure).....	147
4.6.1.1 Creating Account and Logging in	148
4.6.1.2 Dashboard.....	150
4.6.1.2 Archive Page.....	153
4.6.1.3 Report Page.....	159
4.6.1.4 Profile Page.....	169
4.6.1.5 Settings Page.....	172
4.7 Summary	184
CHAPTER 5: TESTING.....	185
5.1 Introduction.....	185
5.2 Functional Testing	185
5.2.1 Unit Testing.....	186
5.2.1.1 Test Cases for the Proposed System.....	187
5.2.2 Integration Testing	198
5.3 Non-Functional Testing	202
5.3.1 Reliability Testing.....	203
5.3.2 Performance Testing	204
5.3.3 AI Model Accuracy Testing.....	205
5.3.4 Usability Testing	205
5.3.4.1 Analysis on User Acceptance Survey.....	207
5.4 Analysis of Non-Functional Testing Results	218
5.5 Demo.....	220
5.5.1 Live detection for multiple packages	220
5.5.2 Double Detection Alerting	221
5.5.3 Slight Occlusion Detection	221

5.6 Summary	222
CHAPTER 6: CONCLUSION AND FUTURE WORKS.....	223
6.1 Introduction.....	223
6.2 Objective and Achievements	223
6.3 Project Limitations and Constraints.....	227
6.4 Future Works	230
6.5 Conclusion	233
REFERENCES	234
APPENDICES	241
Appendix A.....	241
Appendix B	242
Appendix C	250

LIST OF FIGURES

Figure 1.1 Porch Theft which was caught on camera (Shadiqe, 2024)	2
Figure 1.2 Scrum Framework Overview (The Scrum Framework Poster, n.d.).....	6
Figure 1.3 Project Tentative Timeline Gantt Chart	8
Figure 2.1 CVEDIA Package Detection outside the house at the porch.....	16
Figure 2.2 CVEDIA Package detection indoors	17
Figure 2.3 Person Detection within the ROI.....	18
Figure 2.4 CVEDIA-RT modularity	18
Figure 2.5 The AI triggered scenario alerted on Deep Sentinel.....	24
Figure 2.6 Theft Detection functionality by Veesion	25
Figure 2.7 The Logging System utilized by Veesion which can be used as inspiration	26
Figure 2.8 CNN model (Yuan et al., 2017)	30
Figure 2.9 The Pseudocode algorithm of CNN model by P et al. (2023)	31
Figure 2.10 Running Flask on VS Code	36
Figure 2.11 An image of HTML, CSS and JavaScript.....	36
Figure 2.12 MongoDB Logo.....	38
Figure 2.13 MongoDB Compass Downloaded locally	39
Figure 2.14 MongoDB Atlas web interface	40
Figure 2.15 Docker Desktop Interface with MongoDB container on standby	42
Figure 2.16 Figma online Interface.....	43
Figure 2.17 HP14s Specification on the official HP Support Website.....	44
Figure 3.1 Scrum Event Flow (Miu, 2015).....	47
Figure 3.2 Scrum Artifacts (Salem, 2024)	48
Figure 3.3 Agile Methodology Cycle (Laoyan, 2024).....	49
Figure 3.4 Age of Respondents.....	53
Figure 3.5 Gender of Respondents.....	53
Figure 3.6 State Origin of Respondents	54
Figure 3.7 Primary Residence Type of Respondents	54
Figure 3.8 Respondent Frequency of Receiving Packages	55
Figure 3.9 Knowledge of Respondents of Package Theft Victims	56
Figure 3.10 Respondent Concern on Package Theft in Their Locality.....	56
Figure 3.11 Respondents Current Package Protection Method	57
Figure 3.12 Features Respondents Presumed to Useful for A Package Detection System.....	58
Figure 3.13 Overall Importance of all Features	59
Figure 3.14 Affordability vs. Feature Balance	60
Figure 3.15 Willingness of Respondents to Adopt The System.....	61
Figure 3.16 Respondents Perceived Effectiveness of The Smart Package Detection	61
Figure 3.17 Challenges Preventing usage of The Smart Package Detection System	62
Figure 3.18 Integration with Existing Setup Preference by Respondents	63
Figure 3.19 Open Ended Feature or Suggestions for the System	63
Figure 3.20 System Functionality Overview	67
Figure 3.21 System Response to Event Triggers	68

Figure 3.22 System Architecture for SnapSecure	69
Figure 3.23 Flow Chart of SnapSecure	71
Figure 3.24 Context Diagram for SnapSecure	73
Figure 3.25 Level 1 Data Flow Diagram of SnapSecure	75
Figure 3.26 Level 2 DFD for Process 1.0 Account Login	77
Figure 3.27 Level 2 DFD for Process 2.0 Create Account	77
Figure 3.28 Level 2 DFD for Process 3.0 View Dashboard	78
Figure 3.29 Level 2 DFD for Process 4.0 AI Image/Video Processing	78
Figure 3.30 Level 2 DFD for Process 5.0 Manage Live Detections	79
Figure 3.31 Level 2 DFD for Process 6.0 Manage Notification	80
Figure 3.32 Level 2 DFD for Process 7.0 View Statistics and Report	80
Figure 3.33 Entity Relationship Diagram	81
Figure 3.34 Wireframe for Proposed System Landing Page	86
Figure 3.35 Wireframe for SnapSecure Login Page	87
Figure 3.36 Create Account Wireframe for SnapSecure System	88
Figure 3.37 Wireframe for SnapSecure Dashboard Page	89
Figure 3.38 Wireframe for SnapSecure Record Page	90
Figure 3.39 Wireframe for SnapSecure Archive Page	92
Figure 3.40 Wireframe for SnapSecure Action Page	93
Figure 3.41 Wireframe for SnapSecure Profile Page	94
Figure 3.42 Wireframe for SnapSecure Edit Profile Page	95
Figure 4.1 Visual Studio Code Landing Page	100
Figure 4.2 VS code License Agreement	102
Figure 4.3 Directory Selection for VS code installation	102
Figure 4.4 PATH selection for VS code installation	103
Figure 4.5 Finish VS Code Wizard Setup	103
Figure 4.6 Python Extension Installation	104
Figure 4.7 Python Debugger Extension Installation	105
Figure 4.8 Official Python Page to Install Python Package on Machine	107
Figure 4.9 Installing windows 64-bit executable	107
Figure 4.10 Python Setup Wizard	108
Figure 4.11 PATH of the Environment Variable utilizing the currently installed Python version	109
Figure 4.12 Verifying python installation	109
Figure 4.13 Creating Virtual Environment with the name venv	110
Figure 4.14 Selecting Virtual Environment for Activation	110
Figure 4.15 Activation Confirmation	110
Figure 4.16 Inference SDK installation snippet	111
Figure 4.17 Creating Frontend files in VS Code	112
Figure 4.18 Prompting Copilot to create template code on VS Code	112
Figure 4.19 Creating a Python Flask Template	113
Figure 4.20 Workspace Directory Overview	114
Figure 4.21 MongoDB Landing page	118
Figure 4.22 Account Creation Steps	119
Figure 4.23 Acceptance Terms and Conditions	120

Figure 4.24 MongoDB Atlas Personalisation	120
Figure 4.25 Creating Cluster.....	121
Figure 4.26 Connecting Cluster to External Frameworks	122
Figure 4.27 Connecting to MongoDB Atlas through MongoDB Driver	123
Figure 4.28 Navigate to the collection of the FYP Cluster.....	124
Figure 4.29 Creating Database and Collections.....	124
Figure 4.30 After Creation view	125
Figure 4.31 Network Access	125
Figure 4.32 Docker Desktop Landing Page.....	126
Figure 4.33 Docker Desktop Installer.exe download notification	126
Figure 4.34 Configuration phase.....	127
Figure 4.36 Installation Complete.....	128
Figure 4.35 File Unpacking	128
Figure 4.37 User Login/Registration	129
Figure 4.38 Docker Desktop Survey.....	130
Figure 4.39 Docker Desktop Main Page.....	130
Figure 4.40 System Hierarchy of the Proposed System	131
Figure 4.41 Roboflow API and Model Initialization	136
Figure 4.42 detect_objects function.....	137
Figure 4.43 is_point_in_polylines function	137
Figure 4.44 Preprocessing and ROI Filtering	138
Figure 4.45 SORT Tracker implementation snippet	138
Figure 4.46 Preparing data to be stored in database	141
Figure 4.47 ThreadedVideoStream Class.....	142
Figure 4.48 Upload_video route	143
Figure 4.49 MjpegStreamReader Class	147
Figure 4.50 Creating account and logging in process.....	148
Figure 4.51 Dashboard screen of the proposed system.....	150
Figure 4.52 Filtering dates of detections on Archive page	153
Figure 4.53 Viewing Specific Detection Details.....	156
Figure 4.54 Filtering Graph	159
Figure 4.55 Statistical Overview.....	161
Figure 4.56 Detection Table Filtering, Sorting and Search Feature.....	162
Figure 4.57 PDF Generator Workflow.....	165
Figure 4.58 Editing Profile Details	169
Figure 4.59 Settings Page	172
Figure 4.60 Steps to Edit ROI.....	173
Figure 4.61 Selecting Camera as a Video Source	176
Figure 4.62 Selecting Video Upload as Video Source	179
Figure 4.63 Selecting Video Stream as a Video Source.....	182
Figure 5.1 Web Application UI Navigation Feedback.....	209
Figure 5.2 Video Source Selection Statistics	210
Figure 5.3 Web Application Instruction Clarity Statistics	210
Figure 5.4 ROI Detection Capability Statistics.....	211
Figure 5.5 Detection Accuracy Statistics	212

Figure 5.6 Alerting Effectiveness Statistics	212
Figure 5.7 System's Response time for real-time Package theft Statistics	213
Figure 5.8 System's Response Time for Loading Each Page Statistics	213
Figure 5.9 Issue Experienced Statistics	214
Figure 5.10 Usefulness of Keeping Track of Package Deliveries and Safety	215
Figure 5.11 Usefulness for keeping track of package removal.....	215
Figure 5.12 Consideration of Utilizing the System in Existing Infrastructrue	216
Figure 5.13 System Recommendation	217
Figure 5.14 Double Package Detection	220
Figure 5.15 Double Detection Alert.....	221
Figure 5.16 Slight Occlusion Detection.....	221

LIST OF TABLES

Table 2.1 Features and Functionalities of CVEDIA Package Detection	13
Table 2.2 Comparing of Features between systems	27
Table 2.3 Justification of utilizing Tensorflow	33
Table 2.4 Comparision between Tensorflow and other Frameworks.....	34
Table 3.1 Functionalities of the proposed system (SnapSecure)	52
Table 3.2 Software Requirements of The Proposed System	65
Table 3.3 Hardware Requirements for the Development Stage.....	66
Table 3.4 Data Flow between The System and External Entities	73
Table 3.5 Data Dictionary for User Collection	83
Table 3.6 Data Dictionary for Detection Collection	84
Table 3.7 Data Dictionary for Report Collection	85
Table 5.1 Unit Test Case for Signup	187
Table 5.2 Unit Test Case for Login	189
Table 5.3 Unit Test Case for Dashboard	190
Table 5.4 Unit Test Case for Livefeed Module	191
Table 5.5 Unit Test Case for Archive Module.....	192
Table 5.6 Unit Test Case for Reports Module.....	193
Table 5.7 Unit Test Case for Profile Module	195
Table 5.8 Unit Test Case for Settings Module	196
Table 5.9 Unit Test Case for Logout Module.....	197
Table 5.10 Integration Test Case for SnapSecure	199
Table 5.11 Reliability Testing	203
Table 5.12 Performance Testing.....	204
Table 5.13 AI Model Accuracy Testing.....	205
Table 6.1 Details of Objectives and Achievements	224
Table 6.2 Additional Features Implementation	225

LIST OF ABBREVIATIONS

NPCIL National Post and Courier Industry Laboratory

DoD Definition of Done

ROI Region Of Interest

DFD Data Flow Diagram

ERD Entity Relationship Diagram

AI Artificial Intelligence

SPYDER Scientific Python Development Environment

IDE Integrated Development Environment

CNN Convolutional Neural Network

YOLO You Only Look Once

SSD Single-Shot Detector

ML Machine Learning

GPU Graphical Processing Unit

TPU Tensor Processing Unit

HTML Hypertext Markup Language

CSS Cascading Style Sheets

JS JavaScript

GUI Graphical User Interface

CI/CD Continuous Integration/Continuous Delivery

IoU Intersection over Union

UI User Interface

PoE Power over Ethernet

CCTV Closed-Circuit Television

HTTP Hypertext Transfer Protocol

CHAPTER 1: INTRODUCTION

1.1 Project Title

AI-Powered Package Theft Detection System with Real-Time Monitoring and Alerts

1.2 Background

Package theft has become a growing concern in Malaysia, adding fuel to the fire by the rapid increase in E-commerce and online shopping. With the rise in online shopping platforms such as Lazada, Shopee and Zalora, the number of deliveries has caused a surge, presenting new challenges. According to Commission Factory (2023), there has been an increase in the number of online shoppers in Malaysia, which grew approximately 8.3% year-on-year, with a statistic of 15.63 million online goods buyers at the start of 2023. To also factor in according to Kaur (2024), there was a reported 5.4% of year-on-year increase in e-commerce revenue during the third quarter of 2023, where it showed an increase from RM274.6 million to RM289.5 billion. Hence, due to the ever-increasing trend of online shopping providing a positive trend for economical advancements, it does expose a new attack surface for threats like package theft. Some efforts by the government to combat this incident is by the implementation of initiatives such as the National Post and Courier Industry Laboratory (NPCIL) and the National Courier Accelerator Plan to enhance the courier industry (Kosmo, 2021).

The lack of specific regulations addressing package theft has made it a significant issue for both urban and rural residents. According to Act 574 under Penal Code 379, whoever commits theft will be punished with imprisonment for a period with proximity of seven years or fined or both (*Resilience Library | Southeast Asia Resources*, n.d.; Penal Code Section 379 (Malaysia), n.d.). This proves that there is a general theft penal code in Malaysia, however but there is no specified penal code that Malaysia's legal framework surrounding package theft

which focuses to address this growing issue. Porch piracy, though legally categorized as theft,



Figure 1.1 Porch Theft which was caught on camera (Shadiqe, 2024)

is often treated as a minor crime. Law enforcement responses are typically reactive rather than preventive, and the consequences for such crimes are minimal, which emboldens potential thieves.

Additionally, a traditional surveillance system may only capture footage of package theft incidents, but they are often ineffective in preventing theft. Also, most surveillance systems do not offer real-time alerts as well as the ability to proactively detect suspicious activities. Hence by the time the owner reviews the footage, the package would have already been stolen, and hence identifying the thief may become challenging, especially in cases where the thief may be wearing disguise or non-recognisable (Figure 1.1). According to (Shadiqe, 2024), there was a closed-circuit television footage which captured a man in heavy disguise who had stolen a parcel from a house's fence, but the owner was only alerted after some time the parcel had been stolen. Proving that a traditional surveillance system has to be improved for such situations.

The psychological trauma of package theft damages beyond the immediate loss of goods, significantly influencing consumer behaviour and perceptions of safety (Hicks, 2021). A study conducted by Stickle et al. (2020) found that nearly one-quarter of online retail consumers reported experiencing package theft, which heightened their fear of future incidents. This fear often leads to behavioural changes, such as adjusting delivery schedules or opting for alternative delivery locations, ultimately impacting the overall online shopping experience. In Malaysia, as the number of online shoppers continues to rise, understanding the fear associated with package theft becomes essential for retailers and policymakers alike. Addressing these fears through enhanced security measures and public awareness campaigns could mitigate the negative effects on consumer behaviour and encourage continued engagement with e-commerce platforms.

On the other hand, many residential properties are not well equipped with secure areas for package deliveries. This is specifically particular for landed houses, apartments and condominiums in cities where there is a lacking in secure drop-off point, which would inevitably force delivery personnels to leave packages at doorsteps or in common areas, making them easy targets for theft. There were also cases that it enabled thieves to be able to return to the same location where the stolen package was taken just to return the package that they stole with a reason that it is not a package of their liking (FocusM, 2023).

Therefore, in response to the growing issue of package theft, leveraging AI for smart surveillance systems is a promising approach. AI object detection technologies can be utilized to monitor the status of packages, identify potential thieves, and send real-time alerts to homeowners when suspicious activity is detected. Recent advancements in object detection, particularly through deep learning techniques, have significantly improved the accuracy and efficiency of detecting objects in various environments (Zou et al., 2023). Additionally,

integrating facial recognition can enhance security by identifying known individuals or alerting homeowners to unfamiliar people loitering near packages.

1.3 Problem Statement

Package theft has truly caused several noticeable and negatively impactful root causes which had been identified. Firstly, in Malaysia the Insufficient Legal Framework on Package Theft. Although general theft laws exist under Act 574 (Penal Code Section 379), Malaysia lacks specific regulations that addresses package theft. Lack of focused legislation on this issue leaves the crime treated as a minor offense, leading to minimal consequences and inadequate preventive measures. According to Mool (2023), there were 5 men who were jailed for only a week while being fined RM 5,000 which could also end up serving jail time for 3 months for their attempted theft of over 150 sacks of parcels. This shows that even with such magnitude of would only be addressed with a punishment similar to a slap on a wrist for some, which would not be as effective to avoid reoccurrence of the crime again from the same convicts.

Secondly, Current surveillance systems are often limited to capturing footage which will then be manually reviewed if the consumer finds the need to do so if there were any suspicions. This creates a hassle in a sense that there is no pinpointing the exact moment an incident of package theft occurs hence offering little in terms of real-time alerts or proactive detection of suspicious activities that promotes reactive solutions rather than preventive solutions. This inefficiency allows thieves to get away with their crime, as there is no real-time reporting, making it harder to identify them.

On the other hand, package theft creates psychological trauma that goes beyond material loss. Consumers report heightened fears and changes in behaviour, such as altering delivery schedules or seeking alternative locations for receiving packages. This fear impacts the overall shopping experience and engagement with e-commerce platforms. This can be a potential cause

of economical revenue loss which will serve as a negative impact to Malaysia's effort of economy strive in the upcoming years.

Lastly, in Malaysia there is a lack in residential areas which would allow convenient delivery of packages as well as a secure way to ensure safety of placement to avoid any danger or risks that it may pose.

1.4 Objectives

The primary objectives of this project are centred around developing a comprehensive and intelligent AI-based solution that effectively addresses the issue of package theft. By utilizing these advanced technologies such as AI-powered object detection and real-time alerts, the project aims to provide homeowners with a reliable and efficient means of monitoring their packages and preventing theft before it occurs. The specific objectives of the project include:

1. To develop an AI-Powered Object Detection System for Package Monitoring.
2. To implement a real-time alert for suspicious activities to track and log package deliveries and suspicious events.
3. To develop an accessible and user-friendly web-based interface.

This project is expected to fulfil the requirements to assist in prevention efforts of package theft.

1.5 Brief Methodology

Given the robust nature of the project, the Scrum methodology also known as one of the agile practices is known to be very suitable for this project as its approach is incremental and iterative in nature which is used to manage complex tasks as well to develop complex software which has capabilities of changing its requirements, followed by its task-driven nature of breaking tasks into smaller manageable tasks (Verwijns & Russo, 2022). Some of the roles in scrum are the SCRUM Master which is responsible for keeping up with all the processes, the

owner of the product who will be classified as the stakeholder or customer (Hema et al., 2020). This methodology also offers frameworks which promote practices of visibility and transparency to allow all its practitioners to see all the project stages, allowing future corrections and further improvements to ensure that the project keeps moving forward by focusing on achieving the end goal (Sassa et al., 2023).

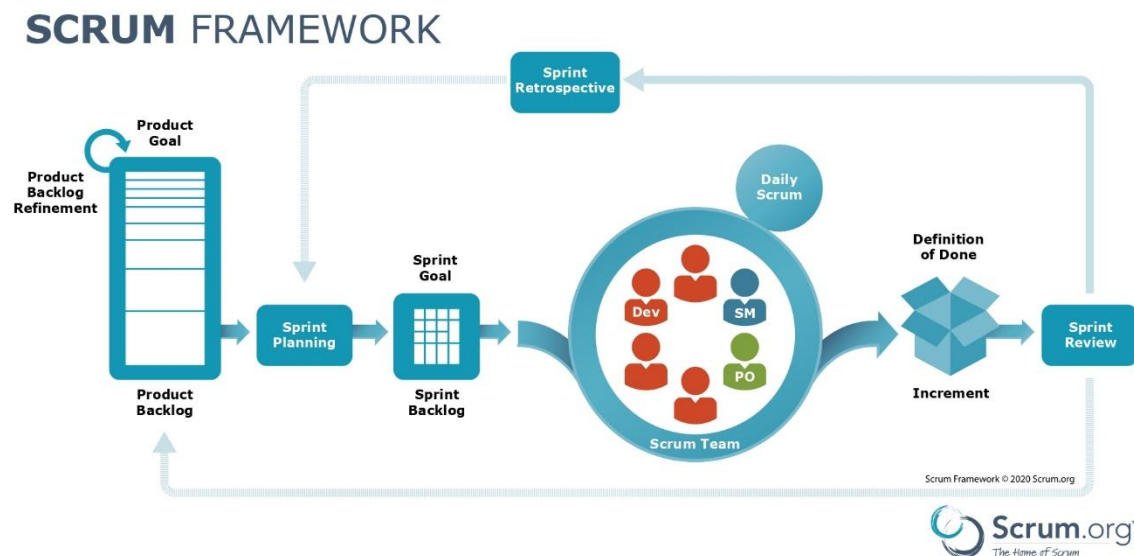


Figure 1.2 Scrum Framework Overview (The Scrum Framework Poster, n.d.)

The Scrum methodology (Figure 1.2) would also focus on delivering valuable outcomes in time-boxed iteration periods, that would span across days to weeks which is also known as sprints. Each day there are scrum ceremonies which review each scrum sprint and its progress. This would be derived as Scrum Events like Sprint Planning for self-assessment of each task to initiate growth with regards to Definition of Done (DoD), sprint retrospective and review daily development.

The work nature of Scrum would focus on a group of features which is the product backlog that will be described in the sprint based on the level of needs of the product owners. These backlogs have been identified during the Sprint planning phase (Salimi, n.d.).

1.6 Scopes

1. Target Users

The project scope focuses on a single user group known as package owners set to be delivered to the property. This single user group is considered the focus of the project to provide them with a real-time monitoring of their package deliveries, informed when there are suspicious activities, receives logging of their delivered packages which will be displayed on a web-based system that will be accessible by them all the time with a user-friendly interface which will enable easy navigation throughout the software.

2. Location

The location of focus would be at the target users premise or property where a surveillance camera would be set up. The scope of functionality would focus on areas such as landed houses, condominiums, apartments, and mail rooms in both urban and residential settings. This would allow a more modular area where a Region of Interest (ROI) can be set as an area where the delivery of package has to be placed and any suspicious activities which including tampering of the product occurs, it will be able to trigger a preset rule, to alert real-time system, that will be able to notify the users, which can now take action.

Other sectors that will be potentially aimed to be tackled include, real-time alerting through web-based system, which will be configured to notify the consumer on the potential occurrence of a potential theft.

1.7 Significance of Project

The focus of this system has its importance by addressing the limitations and restrictions of traditional surveillance methods, which by far has often time fail to provide real-time alerts or proactive security measures, that leaves consumers vulnerable to theft. Therefore, by leveraging AI-powered object detection and real-time alert systems, this project would promise

a more effective approach to preventing and addressing package theft, enhancing security and restoring public trust in delivery services.

This project tackles the issue by filling the gap in prevention efforts through a technological solution, offering consumers peace of mind with real-time alerts that proactively detect suspicious activities, significantly increasing the likelihood of intervention before theft occurs.

By providing a reliable, AI-powered monitoring system, this project aims to restore consumer confidence in the safety of their deliveries, ensuring that package owners feel secure in their purchases and maintain their engagement with online shopping platforms.

1.8 Project Schedule

The project is expected to be completed over two semesters, Semester 1, 2024/2025, and Semester 2, 2024/2025.

1. Semester 1 (FYP1): October 7, 2024 - February 23, 2025
2. Semester 2 (FYP2): February 17, 2025 - July 27, 2025

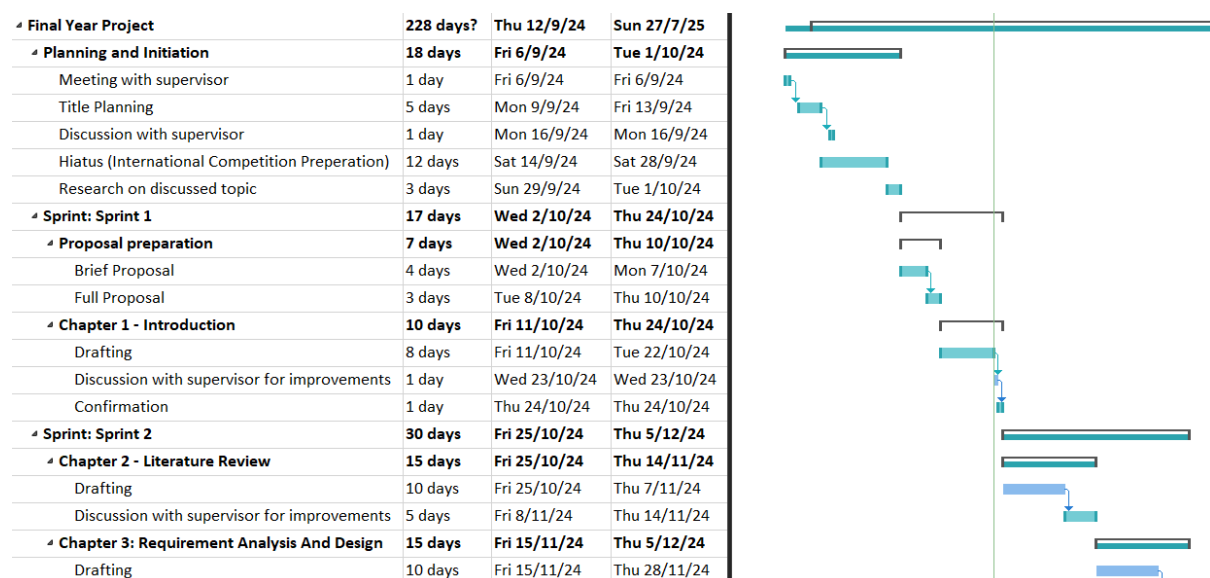


Figure 1.3 Project Tentative Timeline Gantt Chart

1.9 Expected Outcome

The expected outcome would be the AI-Powered Package Theft Detection System with Package Recognition and Real-Time Alerts is expected to deliver an innovative solution to the growing issue of package theft in Malaysia. By integrating advanced AI object detection technologies, the system will monitor packages and detect suspicious activities in real-time, immediately alerting homeowners through a user-friendly platform. This will significantly enhance preventive measures by enabling timely responses to potential theft incidents. Additionally, the system will log and track package deliveries that offers users a seamless and secure interface to manage and review their deliveries.

The project aims to restore consumer trust in online shopping and delivery services by addressing security gaps in traditional surveillance systems, ultimately contributing to a safer, more reliable e-commerce experience. This outcome will also support efforts to reduce the psychological and economic impacts of package theft, particularly in urban and residential areas, where secure delivery solutions are currently limited.

1.10 Project Outline

There are five chapters in this paper to describe the AI-powered package theft detection system's workflow, proceedings and focus on each of the named chapters,

Chapter 1: Introduction delves into package theft, significance of the impact and the innovation that will be built to assist prevention efforts of such occurrences. This chapter also outlines the problem statement, objectives, brief methodology, project significance, project schedule, expected outcome, and project outline.

Chapter 2: Literature Review revolves around an analysis of relevant literature of similar systems to the proposed system. This chapter explores the functionalities inside the existing

system and compares them with one another. The features of the suggested system are also discussed in this chapter.

Chapter 3: Requirement Analysis and Design emphasizes the requirement analysis of the proposed system. In this chapter, user and system requirements are clearly defined. This chapter also provides diagrams such as Scenario Illustration, Web Application, and Software Architecture, Flow Chart Diagram, Context Diagram, Data Flow Diagrams (DFDs), Entity Relationship Diagram (ERD), Data Dictionaries and High-fidelity Wireframe Design.

Chapter 4: Implementation and Testing discusses the implementation phase of the proposed system. This chapter provides test plans and technical descriptions of the system.

Chapter 5: Conclusion and Future Work delivers the summary and conclusion for the system. This chapter describes the objectives accomplished, project limitations, and suggestions for future improvements.

Based on this Paper, there are 5 distinct chapters which encompasses the full flow of the project of an AI-Powered Package Theft Detection System with Real-Time Alerts which will be utilized to reduce and potentially prevent parcel theft locally.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The emphasis of this chapter behold significance in comparing and contrasting similar software which serves similar purposes with the proposed AI-Powered Package Theft Detection System with Real-Time Alerts. The following section will be presented in exploring their features, functionalities and core processes of the following systems: CVEDIA Package Detection, Deep Sentinel, and Veesion. Through a deep examination of these systems, a table of comparison will be constructed to pinpoint the difference between systems and the proposed system. Within this table it would propagate a contrast on their strengths and weaknesses in each system in terms of their functionalities in intrusion detection and safety assurance. Furthermore, this chapter would emphasize the selection of types of software requirements for the process of developing a system which would achieve the objectives mentioned. With close regards to the unique features of the systems that have been rectified, in the systems that had been reviewed, it will be incorporated into the proposed system. This would correspond to produce a discussion of the proposed system's implementation and uniqueness. In conclusion, this chapter would focus on the systems reviewed, yielding its features and functionalities, which will have a potential to be adapted into the proposed system for discussion and development through extensive information gathering.

2.2 Review on Similar Existing Systems

In this section, the 3 systems that share similarities in terms of functionalities and features will be reviewed and analyzed. The chosen systems would be CVEDIA Package Detection, Deep Sentinel, and Veesion.

These systems have been chosen for in-depth analysis and reviewing with regards to their distinctive features and credibility in providing solutions through innovative methods which

were able to elevate and stand out in their business realm. To depict the essence of distinctive and credible, CVEDIA Package Detection would stand out as a go-to software which is able to prevent porch piracy with its ability to detect packages, notifying recipients and all these in real time (*Package Detection | CVEDIA | AI Video Analytics for Any Hardware*, n.d.).

The next system would be by Deep Sentinel, which claims to keep deliveries safe with their smart video surveillance solution that prevents Amazon Package theft and more (Deep Sentinel, 2024).

Lastly, Veesion which puts themselves out there as an intelligent video surveillance which will be able to reduce theft in businesses, by allowing automatic alerts based on their video surveillance system which would be analysed by their servers continuously and notifying anything suspicious (Veesion, n.d.).

These selected systems put much emphasis on their system's function to utilize and embed their software into cameras that would assist their video surveillance to detect and notify their clients if there is threat, suspicious activities and also detection of objects by continuous analysis and prediction by their hardware such as servers.

2.2.1 CVEDIA Package Detection

The package detection system from CVEDIA is an innovative method of video analytics that uses artificial intelligence (AI) models that are trained on fake data. CVEDIA employs synthetic data produced in controlled virtual environments, as opposed to traditional systems that rely on real-world datasets, which are constantly expensive and time-consuming to collect. This technique improves detection accuracy and robustness by guaranteeing that the models are pre-trained to manage a variety of real-world issues, including changes in illumination, occlusions, and different package appearances.

Key benefits pertinent to the creation of an AI-powered package theft detection system, similar to the one suggested in this project, are provided by this system's dependence on

synthetic data. First off, since no personal information is used, it removes privacy issues and guarantees adherence to data protection laws like the CCPA and GDPR. This is consistent with the increasing significance of developing ethical AI. Second, the time and expenses involved in developing and testing a package detection system for home security applications may be greatly decreased by using synthetic data, which speeds up model development and deployment.

Furthermore, CVEDIA's solution exhibits scalability and adaptability by supporting deployment across several platforms, such as cloud systems and edge devices. With features for real-time video analysis and adaptable output formats, its modular CVEDIA-RT inference engine offers a flexible framework that can be easily included into warehouse management, home security, or logistics systems.

Using concepts from CVEDIA's methodology, such as modular architecture and the use of synthetic data, could improve system dependability, guarantee data compliance, and expedite development for the suggested package theft detection system. The development of specialized models designed to identify suspicious activity or abnormalities may also be made possible by the use of synthetic data. With features for real-time video analysis and adaptable output formats, its modular CVEDIA-RT inference engine offers a flexible framework that can be easily included into warehouse management, home security, or logistics systems.

Table 2.1 Features and Functionalities of CVEDIA Package Detection

Category	Details
Core Technology	AI models trained with proprietary synthetic data, enabling robust performance in diverse scenarios.
Key Features	- Real-time detection and tracking of packages using visible-light cameras.

	- Fine-grained classification to differentiate between package types (envelopes, boxes, etc.). - Alerts for suspicious activity such as package removal or tampering. - Occupancy detection to monitor if someone is near a package. - Fully customizable notifications for homeowners or businesses.
Detection Accuracy	- Optimized for both ground-level and elevated perspectives. - High accuracy even under challenging lighting conditions, such as dawn and dusk.
Hardware Compatibility	- Designed for edge devices with low power requirements (e.g., NVIDIA Jetson, Qualcomm Snapdragon, etc.) - Flexible deployment options for cloud, on-premises, or embedded systems.
Architecture	- Powered by ResNet-101 backbone and RefineDet detector for high-performance detection.
Output Formats	- Supports data export in JSON, MQTT, CSV, JPG, PNG, MP4, RTSP, and RTMP formats for seamless system integration.
Ease of Use	- Intuitive user interface for configuring detection regions, sensitivity, and export settings. - Remote UI access for headless operation on cloud or edge devices.

Applications	- Residential: Enhances home security by detecting package deliveries and theft. - Retail and Warehousing: Monitors package movement to prevent theft and ensure inventory accuracy. - Public Spaces: Tracks unattended packages in airports, stations, and public venues for safety.
Uniqueness	- Synthetic data eliminates reliance on real-world datasets, ensuring faster training, privacy compliance, and adaptability to new use cases.

Due to the CVEDIA Package Detection system is constructed with sophisticated AI models that have been trained on synthetic data, it is incredibly flexible and dependable in a range of scenarios. This system is notable for its ability to precisely handle a variety of application situations. Real-time package identification and tracking, fine-grained categorization to distinguish between various package kinds, and configurable notifications for both businesses and homeowners are some of its primary features. It also goes beyond simple detection by spotting suspect activity, such as removing or tampering with shipments, and even keeping an eye on whether someone is loitering close to a box.

Its precision is one of its advantages as it performs admirably in both high and ground-level views, especially in challenging lighting situations like dawn or twilight. The system may be installed on-site, in the cloud, or as an embedded solution, and it works with energy-efficient gadgets like the Qualcomm Snapdragon and NVIDIA Jetson.

RefineDet and a ResNet-101 backbone power the system's architecture, providing effective and high-performance detection. With support for a variety of output formats like JSON and MQTT as well as video streams like MP4 and RTSP, it is made to blend in seamlessly

with current configurations. While remote access features enable smooth operations on edge or cloud devices, the user-friendly interface makes it simple to modify detection zones, sensitivity, and export settings.

Due to its versatility, the system may be used to improve home security by detecting package deliveries and theft, assist retail and warehouse operations by tracking inventory, and improve public safety by keeping an eye on abandoned parcels in places like train stations and airports. The use of synthetic data, which expedites model training, protects privacy, and enables the system to readily adjust to novel situations, is what distinguishes CVEDIA. All things considered, it's a progressive answer to today's package management and security issues.



Figure 2.1 CVEDIA Package Detection outside the house at the porch

Figure 2.1 is from a Ring security camera demonstrating CVEDIA's package detection technology. The system is highlighting a delivered package on the porch with a green bounding box, along with what appears to be a confidence score of "6@0.94" (indicating a 94% confidence in the detection).

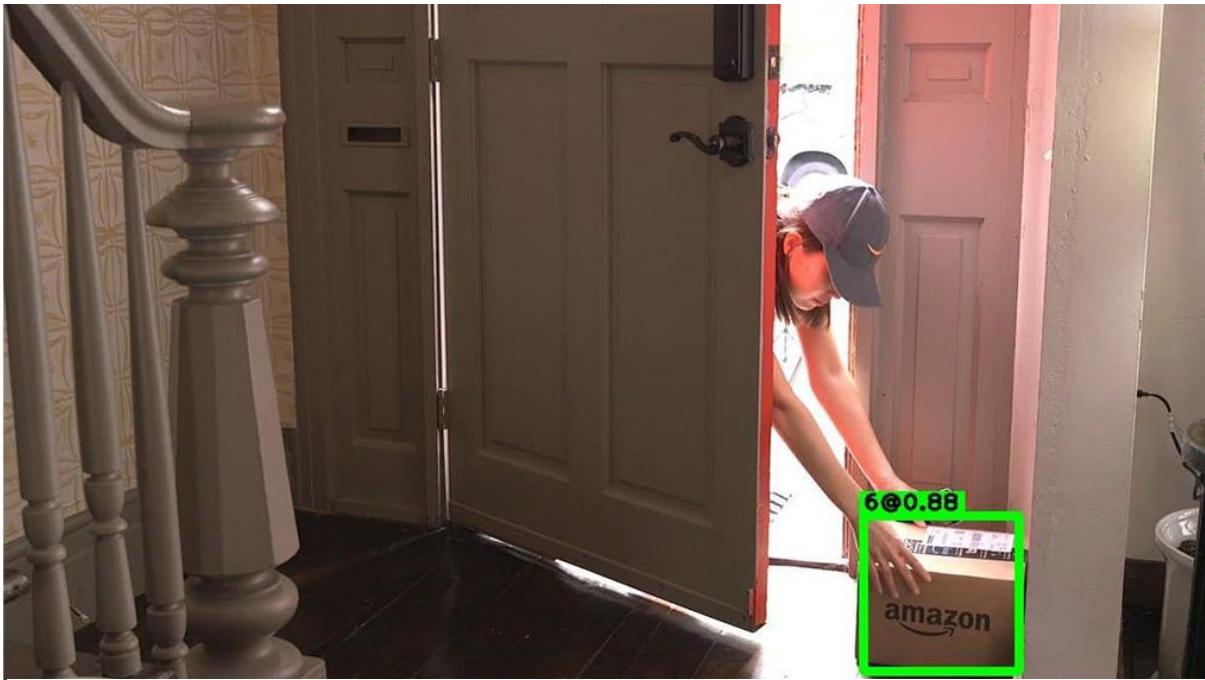


Figure 2.2 CVEDIA Package detection indoors

Figure 2.2 demonstrates CVEDIA's package detection technology working in an indoor environment, specifically at what appears to be the interior entrance of a home. The system is detecting an Amazon package being delivered, marked with a green bounding box and a confidence score of "6@0.88" (indicating an 88% confidence in the detection).



Figure 2.3 Person Detection within the ROI

Figure 2.3 demonstrates CVEDIA's person detection capability within a defined Region of Interest (ROI), operating in low light/nighttime conditions. The system is using what appears to be infrared or thermal imaging to detect and track human presence. That can be drawn by the client (*Setting up Areas and Tripwires* - CVEDIA-RT, n.d.)

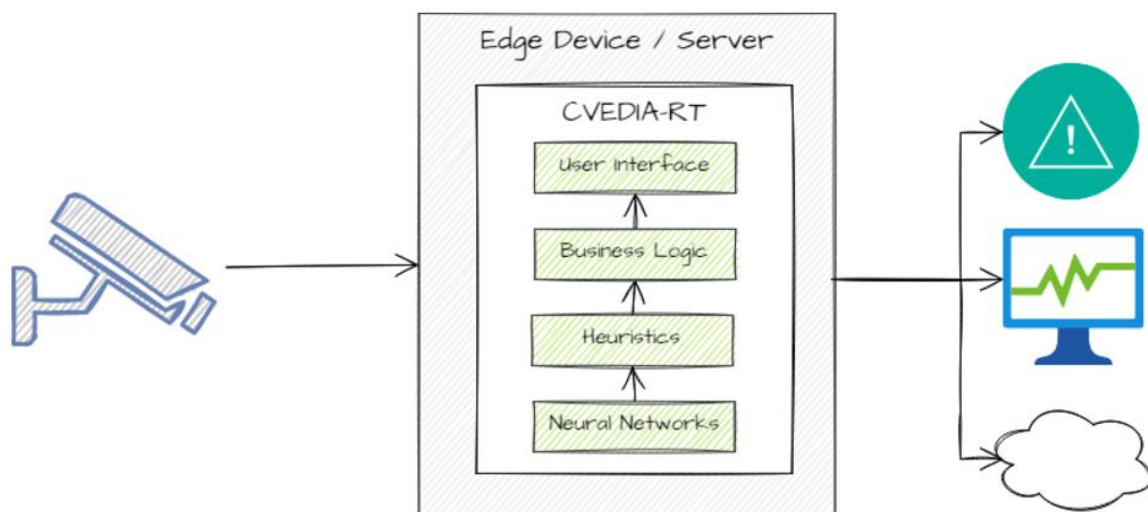


Figure 2.4 CVEDIA-RT modularity

Figure 2.4 above depicts that the CVEDIA-RT system can be used as a cross-platform AI interface engine which would be able to provide integrations for both high and low-level interfaces like deployment on edge devices, ran standalone on a desktop or as a dockerized container on the cloud.

Key Components and Workflow:

1. Edge Device/Server Integration:

CVEDIA-RT is capable of functioning on edge devices, which are hardware components located near the source of data generation (e.g., surveillance cameras). This capability is crucial for applications that demand low-latency processing and reduced dependency on cloud-based resources. The edge device processes the data in real-time, ensuring quick detection and analysis without significant delays.

2. Modular Architecture:

CVEDIA-RT is structured into four primary layers that work together to deliver AI functionalities:

- **Neural Networks:**

This layer is the core of the AI engine, handling tasks like object detection, face recognition, and behavior analysis. Pre-trained and optimized neural networks ensure efficient inference on edge hardware with limited computational power.

- **Heuristics:**

This layer applies rules or logic to complement the neural network's outputs. For example, it may apply additional contextual checks (such as time-based restrictions or spatial movement thresholds) to refine detection accuracy.

- **Business Logic:**

This layer defines application-specific functionalities. It translates the raw AI outputs into meaningful insights or actions based on user-defined requirements. For example, a surveillance system, it might define thresholds for triggering alerts based on detected activities.

- User Interface:

This component provides a way for users to interact with the system. It might include dashboards, visualization tools, and configurable options to tailor the AI engine's behaviour to specific needs.

3. Integration and Deployment Options:

- Edge Devices:

The system supports lightweight deployment on edge devices, reducing reliance on centralized servers or cloud platforms. This enables real-time data processing close to the source and minimizes network latency.

- Standalone Desktop:

CVEDIA-RT can run on personal computers or standalone servers, offering flexibility for environments without dedicated edge devices or cloud dependencies.

- Cloud-Based Deployment:

Through Dockerized containers, CVEDIA-RT can be deployed in cloud infrastructures, allowing scalable AI processing. This is particularly beneficial for applications requiring centralized data aggregation or advanced analytics beyond the capabilities of edge hardware.

4. **Outputs and Actionable Insights:** The system generates two types of outputs:

- **Alerts:** Triggers based on predefined criteria, such as detecting intruders or anomalies.
- **Visualization:** Displays data trends, performance metrics, or live feeds, providing users with a comprehensive overview of the system's operations.

Ideas of Implementation:

- **Neural Networks:** For recognizing packages and detecting theft behavior.
- **Heuristics:** For adding rules and logic, such as detecting prolonged lingering near a package.
- **Edge Deployment:** Ensuring real-time detection on smart cameras without overloading the network.
- **Alerting Mechanism:** Sending real-time notifications to users about suspicious activities.

2.2.2 Deep Sentinel

Deep Sentinel stands out in the package theft prevention market with its integration of AI-driven monitoring and live human intervention. The system leverages cutting-edge technologies such as artificial intelligence and real-time video analytics to detect threats, paired with live surveillance guards who can assess situations and intervene directly. This combination of AI and human monitoring sets Deep Sentinel apart from conventional security systems that are often reactive rather than proactive.

Core Technical Functionalities

Deep Sentinel's cameras feature motion detection, which activates upon identifying movement within designated zones. The footage is processed locally on an AI hub that evaluates potential threats, such as the presence of a human near a package, while filtering out benign activity like passing vehicles. If the AI flags a situation as suspicious, the footage is streamed in real-time to professional guards stationed at the LiveSentinel monitoring centre (Vigderman & Turner, 2024). These guards assess the threat and can intervene immediately using two-way audio or built-in alarms to deter potential thieves. Additionally, guards can contact law enforcement within 30 seconds, providing verified reports to ensure prompt response (Deep Sentinel, 2024a).

Key Features

1. **AI-Enhanced Cameras:** Processes over 50 frames per second to identify threats with high accuracy.
2. **Live Guard Monitoring:** Professional guards provide human oversight to prevent false alarms and take action against verified threats.
3. **Two-Way Audio and Alarms:** Guards can directly engage with suspects through 104 dB speakers or trigger alarms to deter criminal activity.

4. **Rapid Law Enforcement Contact:** Guards can report incidents directly to local authorities with detailed suspect descriptions, reducing average police response time.
5. **Wide Coverage and Night Vision:** Equipped with a wide field of view and superior night vision for monitoring day and night

Unique Approach and Benefits

Deep Sentinel's hybrid model of proactive security contrasts with traditional systems that rely on post-event reviews or user-initiated alerts. By combining AI analysis with trained human oversight, the system significantly reduces false alarms while ensuring immediate intervention. This model is particularly effective for residential properties vulnerable to porch pirates and for businesses that require reliable package monitoring.

Applications

- **Residential Security:** Provides homeowners with peace of mind by preventing porch piracy.
- **Commercial Properties:** Ideal for businesses needing package and inventory monitoring.

- **Public and Shared Spaces:** Effective for securing package delivery zones in apartments and public lockers

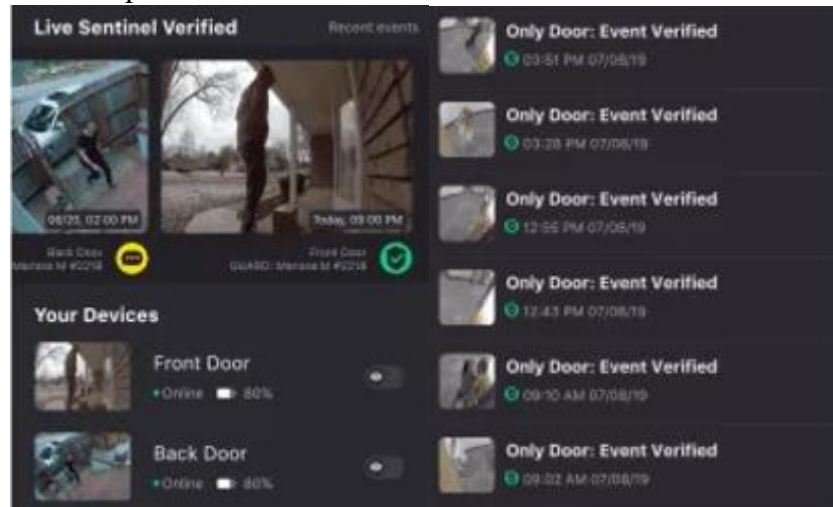


Figure 2.5 The AI triggered scenario alerted on Deep Sentinel

Based on Figure 2.5, Deep Sentinel's system offers valuable insights into logging and notifying mechanisms for a package theft detection system. The system employs an AI hub to evaluate potential threats and generates real-time logs of detected activities, categorizing them into harmless and suspicious events. These logs are with live guard interventions, which are promptly notified through cloud-based services to ensure minimal delay. Although it is implemented via mobile app, the build idea and functionalities have would prove to insightful in building a web-based system.

2.2.3 Vession: Gesture Recognition for real-time security in video surveillance

Vession utilizes advanced gesture recognition technology powered by deep learning to enhance real-time video surveillance for security purposes. This innovative solution detects suspicious behaviors, such as theft-related gestures, by continuously analysing video feeds from existing surveillance systems. Vession is highly compatible with a wide range of digital and analogue cameras, enabling seamless integration into 90% of surveillance infrastructures without requiring hardware replacements. This scalability and cost-effectiveness make Vession a transformative tool for retail and security environments.



Figure 2.6 Theft Detection functionality by Vession

The core functionality of Vession lies in its ability to identify over 10 types of suspicious gestures, such as hiding objects in clothing or bags. These gestures are recognized through AI models trained on millions of samples, ensuring high accuracy and adaptability to various store environments (Figure 2.6). The system processes video streams in real-time and sends instant

alerts, including short video clips, to devices such as smartphones, tablets, and computers. This enables rapid intervention by security personnel, who can act based on actionable intelligence.

Key features of Veesion include its plug-and-play installation process, which requires minimal setup time and would be proven often operational within 30 minutes. The system is accessible across platforms via a web interface and mobile apps for iOS and Android, providing flexible monitoring options. Furthermore, its AI models continually improve through self-learning, adapting to new behaviours and environments to enhance detection accuracy over time. Veesion also supports centralized control, allowing users to manage alerts and generate performance reports across multiple locations from a single dashboard.

Veession is primarily used in retail security through proactive theft detection. By automating gesture analysis, it alleviates staff workload and enables focused inventory management. The system's scalability and intuitive customization options also make it an ideal solution for shared security responsibilities among store teams.

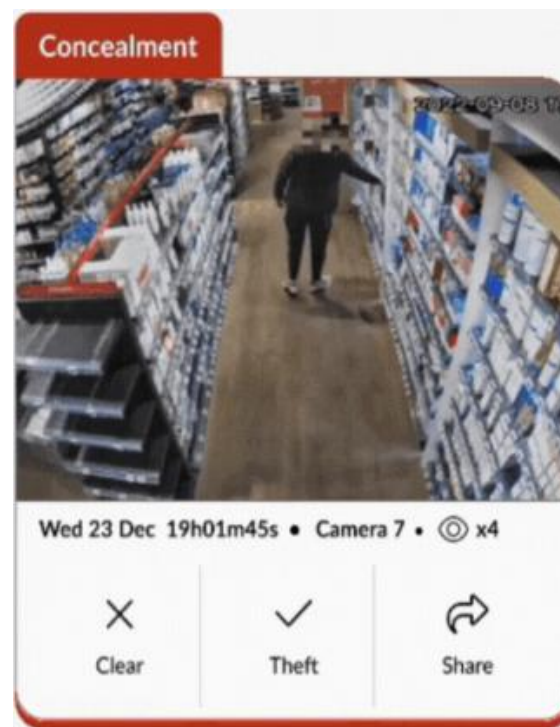


Figure 2.7 The Logging System utilized by Veession which can be used as inspiration

For a package theft detection system, Veesion provides valuable insights into logging and notification mechanisms (Figure 2.7). Its automated logging records suspicious gestures with timestamps and detailed video evidence, while customizable detection settings minimize false positives. Notifications tailored to specific events can be delivered to user devices or web dashboards for immediate action. These features demonstrate how gesture recognition and real-time alerting can transform surveillance systems into efficient and responsive security solutions, offering practical inspiration to the proposed system.

2.2.4 Comparison of Features between Systems

Table 2.2 Comparing of Features between systems

Systems Features	CVEDIA	<i>Deep Sentinel</i>	Veesion	Proposed System (SnapSecure)
AI-Powered Detection	✓	✓	✓	✓
Synthetic Dataset	✓	X	X	?
Gesture Recognition	X	X	✓	X
Real-Time Monitoring	✓	✓	✓	✓
No human intervention	✓	X	✓	✓
Alerts	✓	✓	✓	✓
Web-Based Dashboard	✓	X	✓	✓

Fast Response (e.g., alerts <30s)	✓	✓	X	✓
Subscription based	✓	✓	✓	X

According to the comparison table (Table 2.2), the proposed system, SnapSecure aims to implement AI-Powered Detection which integrates AI-Based algorithms for detection of objects and suspicious activities all within a surveillance area. The system would focus on recognizing packages, intruders and suspicious behaviours. This would ensure automation and efficient monitoring, which is similar to the functionalities offered by CVEDIA which utilizes neural networks for accurate detection. This is dissimilar to the traditional camera system which relies heavily on manual observation. SnapSecure's capability which is driven by AI would reduce false alarms that would enhance its reliabilities, making it suitable for home and small business security applications.

Another point to take into consideration, is that SnapSecure would take into account the use of pre-trained models and synthetic datasets to bypass the need for hardware-intensive training. According to Monji-Azad et al. (2023), synthetic datasets are information which are artificially generated using simulations or computer algorithms. It can be generated as a ground truth for the system. This is due to the fact that real-world datasets can be difficult to obtain.

Next, although SnapSecure does not focus on gesture recognition, it should be notable that the system's core emphasis is on porch piracy which places focus on the detection of objects and individuals related to package theft and unauthorized access. Hence by the exclusion of this feature, the development resources would be focused on refining the package tracking and alerting mechanisms.

Fourthly, SnapSecure will be able to provide continuous surveillance through real-time monitoring capabilities, enabling users to watch live camera feeds through a web-based

dashboard. This feature was inspired by Veesion's ability to deliver live updates from CCTV systems. By enabling real time insights, users can act promptly to mitigate security risks. The system will incorporate visual alerts during live feeds to notify users of detected threats, which would be able to enhance situational awareness and response time.

In the proposed system, No human intervention would be needed unlike Deep Sentinel which requires professional security guards to intervene once notified to ensure the issue is escalated to notify the users. In the proposed system, the users will be directly notified if there are events of triggering the system. As well as the inclusion of alerts that will be sent to the users via the web dashboard.

Next would be the Web-Based Dashboard. A central feature of SnapSecure is its web-based dashboard, which allows users to access and manage surveillance footage from any internet-connected device with a browser. This contrasts with Deep Sentinel's mobile app focus, making SnapSecure accessible to users without smartphones. The dashboard will include features such as real-time feeds, incident logs, and alerting options. By offering a browser-based interface, SnapSecure ensures compatibility across devices, increasing user flexibility and system accessibility.

Other than that, SnapSecure is designed to generate and deliver alerts within seconds of detecting anomalies. This rapid response feature is inspired by CVEDIA, which prioritizes quick event detection and notification to prevent potential losses. Users will receive alerts as soon as suspicious activities are identified, enabling them to act swiftly, whether it's contacting authorities or remotely monitoring the situation.

Finally, unlike many commercial systems, SnapSecure will not operate on a subscription model. This decision eliminates ongoing costs for users, making it an attractive and affordable option for households and small businesses.

2.3 Review of Tools and Technologies

This section would be covering on the software and hardware requirements that would be proposed for the development of the proposed system. It would be describing the requirements, reasonings and features.

2.3.1 Software

The focus of this section is to give an overview of the intended softwares that will be used for developing the proposed system.

2.3.1.1 Roboflow

Roboflow is a comprehensive computer vision platform that provides access to pre-trained models and simplified deployment options through its API. The platform handles the underlying complexity of model hosting, optimization, and inference while exposing a simple interface for developers (Dupont, 2024). For this project, the utilization of a Single Shot Detector (SSD) architecture running on TensorFlow backend technology would be proposed.

Convolutional Neural Network (CNN)

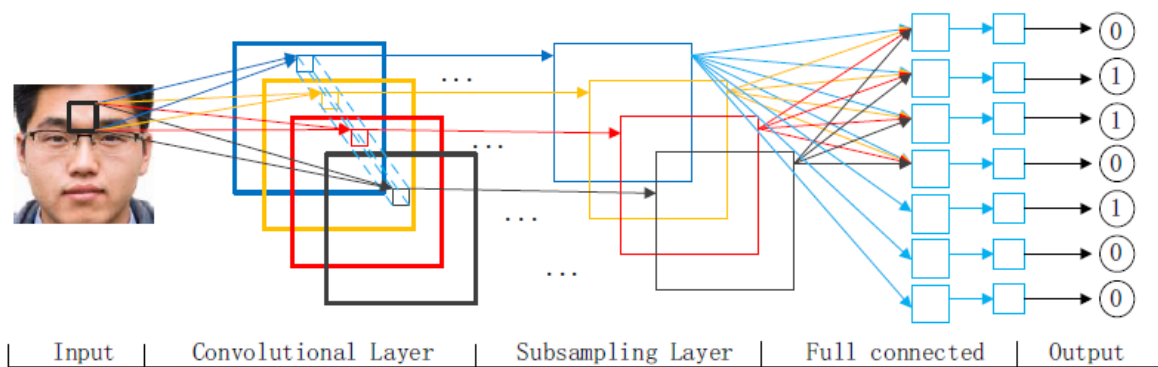


Figure 2.8 CNN model (Yuan et al., 2017)

Based on Figure 2.8, Convolutional Neural Networks (CNNs) are a specialized type of deep neural network designed for processing structured grid data like images. CNNs excel in image recognition tasks due to their ability to automatically and adaptively learn spatial hierarchies of features from input images. They are composed of layers like input, convolutional layers

(for feature extraction), pooling layers (for dimensionality reduction), fully connected layers (for classification), and output.

CNNs are particularly robust against variations in image scaling, rotation, and translation. They are widely applied in domains such as face recognition, object detection, and scene understanding. For instance, CNN-based models like YOLO and SSD are frequently used in object detection systems to identify objects within images or videos.

```
1: algorithm Parallel-CNN
2: input: d: dataset, l: dataset true labels, W:
   Word2Vec matrix
3: output: score of Parallel-CNN trained model on
   test dataset
4: let f be the featureset 3d matrix
5: for i in dataset do
6:   let fi be the featureset matrix of sample i
7:   for j in i do
8:     vj ← vectorizeG, w
9:     append vj to fi
10:    append fi to f
11: ftrain, ftest, ltrain, ltest ← split feature set and labels
   into train subset and test subset
12: M ← Parallel-CNN (ftrain, ltrain)
13: score ← evaluate (i, ltest, M)
14: return score
```

Figure 2.9 The Pseudocode algorithm of CNN model by P et al. (2023)

This pseudocode (Figure 2.9) represents the Parallel-CNN algorithm, which is a machine learning model used for various classification tasks. Let me explain it in detail:

1. The input to the algorithm is a dataset, 'd', which contains the true labels, 'l', and a WordToVec matrix, 'W', which is a matrix representation of the dataset's textual information.
2. The output of the algorithm is the score, or performance metric of the Parallel-CNN trained model on the test dataset.
3. The algorithm starts by setting the features set, 'f', to be the 3D matrix of the dataset.
4. It then iterates through each sample 'i' in the dataset, where it:
 - a. Let 'f' be the feature set matrix of sample 'i'.

- b. Vectorizes the feature set matrix 'f' using the WordToVec matrix 'W', resulting in the vector 'v_i'.
 - c. Appends 'v_i' to the feature set 'f'.
 - d. Appends the true label of sample 'i' to the label set 'l'.
5. The algorithm then splits the feature set 'f' and the label set 'l' into train and test subsets.
 6. Next, it trains a Parallel-CNN model on the train subset and evaluates its performance on the test subset.
 7. Finally, the algorithm returns the score of the Parallel-CNN trained model on the test dataset.

This pseudocode is a high-level representation of the Parallel-CNN algorithm, which is a type of convolutional neural network (CNN) that processes the input data in parallel, potentially improving the model's performance and efficiency.

Object Detection

Object detection is a subset of computer vision that involves identifying and locating objects within images or videos. The process typically includes detecting the presence of objects, classifying them into predefined categories, and localizing them using bounding boxes. It is a critical component in applications like surveillance, autonomous vehicles, and inventory management.

Techniques such as YOLO (You Only Look Once) and Faster R-CNN use CNNs to perform object detection with high accuracy and speed (Potrimba, 2024). These methods are crucial for real-time applications for the proposed system, where the system must analyze video feeds to detect suspicious activities and alert users in real-time.

Suitability

Table 2.3 Justification of utilizing Roboflow

Benefit	Explanation
Pre-trained Model Access	Roboflow provides access to specialized models like package-detection-e1ssd/5, which uses SSD architecture optimized for package detection without requiring custom training.
Ease of Deployment	Roboflow API enables simple deployment with minimal code, allowing models to be accessed via cloud API calls without managing infrastructure.
Real-Time Inference	The Roboflow Inference Server supports optimized inference performance critical for surveillance systems requiring real-time package detection.
Simple Integration	Roboflow's high-level API makes it easy to integrate computer vision capabilities with minimal code while leveraging powerful model architectures.
Built-in Preprocessing	Automatic image preprocessing, scaling, and normalization simplifies implementation of detection pipelines.
Active Development	Backed by a large community with over 1 million developers and used by 16,000+ organizations, providing ongoing support and improvements.

Based on Table 2.3 above it can be said that Roboflow API is seen suitable for being used as a framework for real-time surveillance and package theft.

Table 2.4 Comparison of Implementation Approach compared to Tensorflow

Feature	Direct TensorFlow Implementation	Roboflow API Implementation
Development Time	Lengthy - requires model architecture design, training pipeline setup, and hyperparameter tuning	Rapid - uses pre-trained models accessible through simple API calls
Technical Expertise Required	High - needs deep understanding of neural network architectures, training procedures, and optimization techniques	Low - primarily requires API integration knowledge
Performance	Customizable but requires expertise to optimize	Pre-optimized models with industry-standard performance
Scalability	Manual implementation of scaling mechanisms needed	Built-in cloud scaling with no additional infrastructure management
Maintenance	Ongoing model updates and infrastructure maintenance required	Managed by Roboflow with regular model improvements
Integration	Complex integration requiring TensorFlow dependency management	Simple HTTP requests with minimal dependencies

Based on Table 2.4, The comparative analysis between Direct TensorFlow and Roboflow API implementations reveals significant trade-offs across multiple dimensions critical to project success. Direct TensorFlow implementation offers maximum customization and potential performance optimization through architecture-specific tuning, but demands extensive expertise in deep learning theory, computer vision principles, and infrastructure management, while extending development timelines from weeks to months and creating

substantial ongoing maintenance burdens for framework updates, security patching, and model retraining (Paleyes et al., 2022). Conversely, Roboflow API implementation dramatically accelerates development through pre-trained, optimized models like "package-detection-`elssd/5`," reducing implementation from months to days while lowering the expertise threshold to primarily API integration skills, eliminating complex dependency management through standardized HTTP requests, and transferring maintenance responsibilities to the service provider. Performance benchmarks indicate pre-trained SSD models achieve 50-100ms inference times per frame which is sufficient for surveillance applications requiring 10-20 frames per second while built-in cloud scalability automatically adjusts to demand patterns without requiring dedicated infrastructure engineering, making the Roboflow approach optimal for this package detection system by delivering reliable detection capability with significantly reduced development time and maintenance requirements (Dupont, 2024).

2.3.1.2 VS Code (Flask)

According to Bhattacharya (2023), VS Code is highly favoured among developers for its intuitive features and ease of use. Its syntax highlighting and color-coded distinction make reading and debugging code more manageable. This feature supports numerous programming languages, enhancing productivity by allowing developers to quickly identify different components of their code, such as keywords, strings, and variables. Additionally, the IntelliSense feature provides intelligent code completion and real-time error detection, streamlining the coding process. These tools, coupled with customizable themes and extensions available in the VS Code marketplace, make it adaptable to individual preferences and need (Wanyoike & Wanyoike, 2024).

Flask is considered as a small lightweight python web framework that would aim to provide many useful features and tools which will be able to create a web application in python easier. This would pronounce more flexibility and more easily accessible frameworks for new

developers since it is a micro framework which uses a single Python file. Which means it is also extensible as it does not force usage of directories for structures that require complicated code before getting started (*How to Make a Web Application Using Flask in Python 3*, n.d.).

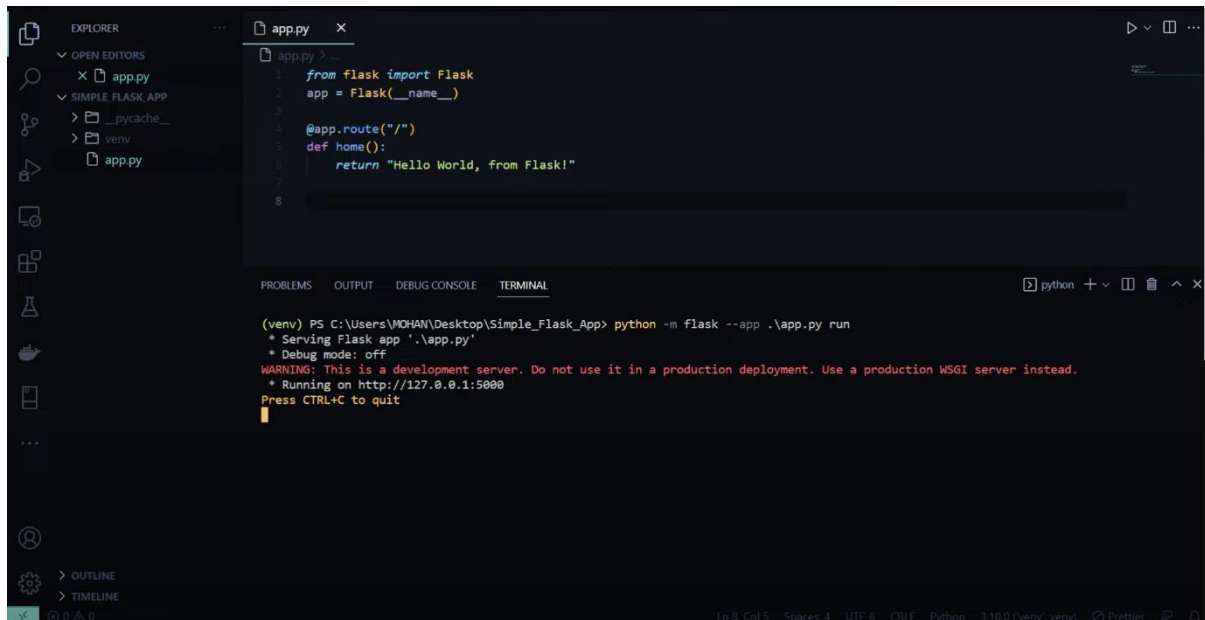


Figure 2.10 Running Flask on VS Code

Flask can be run on VS Code as it would require to host a local server by downloading and installing flask packages which enables the usage of the development server to allow hosting through endpoint which will then be customizable with Web Development Technologies (Figure 2.10).

2.3.1.3 Web Development Technologies



Figure 2.11 An image of HTML, CSS and JavaScript

HTML (HyperText Markup Language)

Based on Figure 2.11 HTML serves as the backbone of the website by providing its basic structure. It organizes the content into meaningful elements such as headings, paragraphs, forms, and images. In the project, HTML can be used to create pages displaying real-time alerts, package recognition data, and user dashboards, ensuring the layout is well-structured and accessible (GeeksforGeeks, 2024).

CSS (Cascading Style Sheets)

CSS handles the visual presentation of web pages, defining aspects such as layout, colors, fonts, and responsiveness. This is particularly important for user experience, as it ensures that detection system's interface is visually appealing and intuitive across different devices (GeeksforGeeks, 2024). For example, CSS can style the alerts and status updates to make them prominent and easy to understand.

JavaScript (JS)

JavaScript adds interactivity and dynamic features to website. According to Ubah (2024) It enables functionalities like real-time updates, interactive dashboards, and animations. For instance, JavaScript can be used to implement client-side scripts that fetch and display live surveillance feeds or trigger alerts for suspicious activities, making the system more responsive and user-friendly (GeeksforGeeks, 2024).

2.3.1.4 MongoDB



Figure 2.12 MongoDB Logo

MongoDB (Figure 2.12) is designed for handling unstructured and semi-structured data, making it ideal for systems that require real time data receiving and retrieval, such as surveillance systems. The database supports JSON-like BSON documents, allows flexible designs that can adapt as the AI model identifies new types of data or behaviours (Global, 2024). This flexibility ensures the system remains efficient and extensible for future updates. Also, MongoDB is well known for its practicality in Artificial Intelligence (AI) providing support for AI-generated images, code, audio and videos (MongoDB, n.d.).

MongoDB also excels at processing time-series data, which is essential for logging real-time events from a camera system. Its capabilities to index and aggregate data at high speeds make it suitable for generating actionable insights, such as detecting anomalies in package movement or intruder behaviour (*Ebook - Real-Time Analytics on MonGoDB: The Ultimate Guide*, n.d.).

MongoDB is designed to excel in scalability, utilizing horizontal scaling through a method known as sharding. This approach allows the database to partition data across multiple servers, maintaining high availability and consistent performance even as the volume of data, such as surveillance footage and detection logs, increases. By enabling the system to add servers without requiring extensive reconfiguration, MongoDB ensures that growing data demands are handled efficiently (MongoDB, n.d.). This makes it particularly beneficial for systems that might expand to include multiple sites or incorporate additional IoT devices.

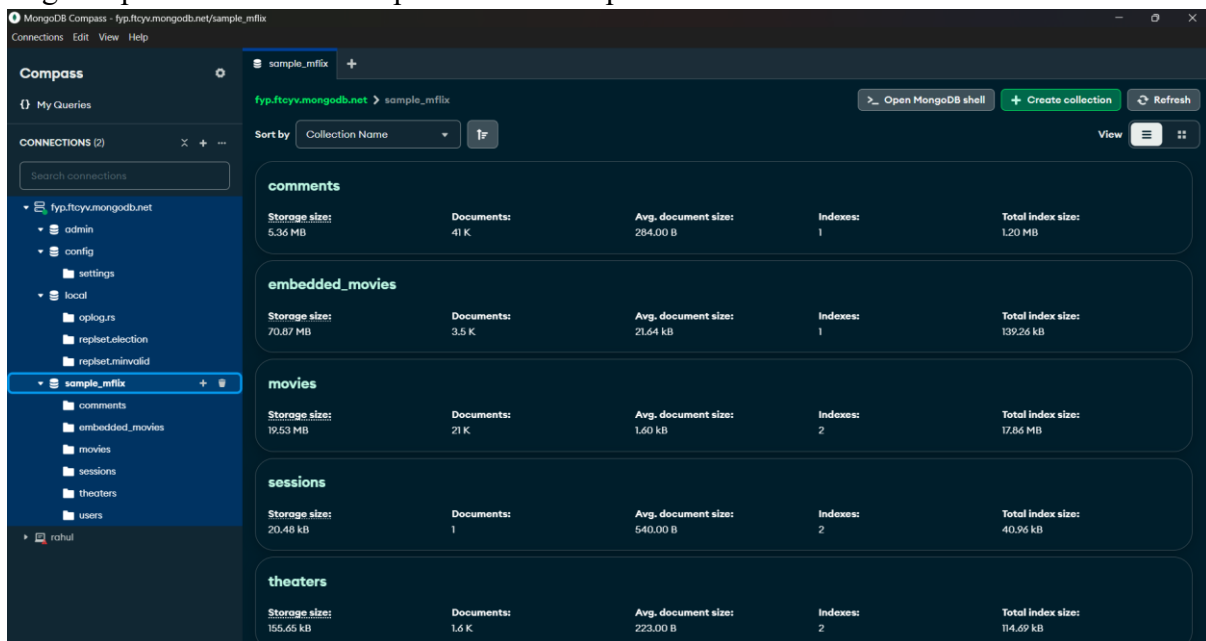


Figure 2.13 MongoDB Compass Downloaded locally

Based on Figure 2.13, MongoDB Compass allows its users to download their software which is a Graphical User Interface (GUI) locally to allow better visualisation, manipulation and analysis instead of doing it on the web. However, it would require us to add the IP address of the connection to MongoDB compass before given the chance to accessing the features.

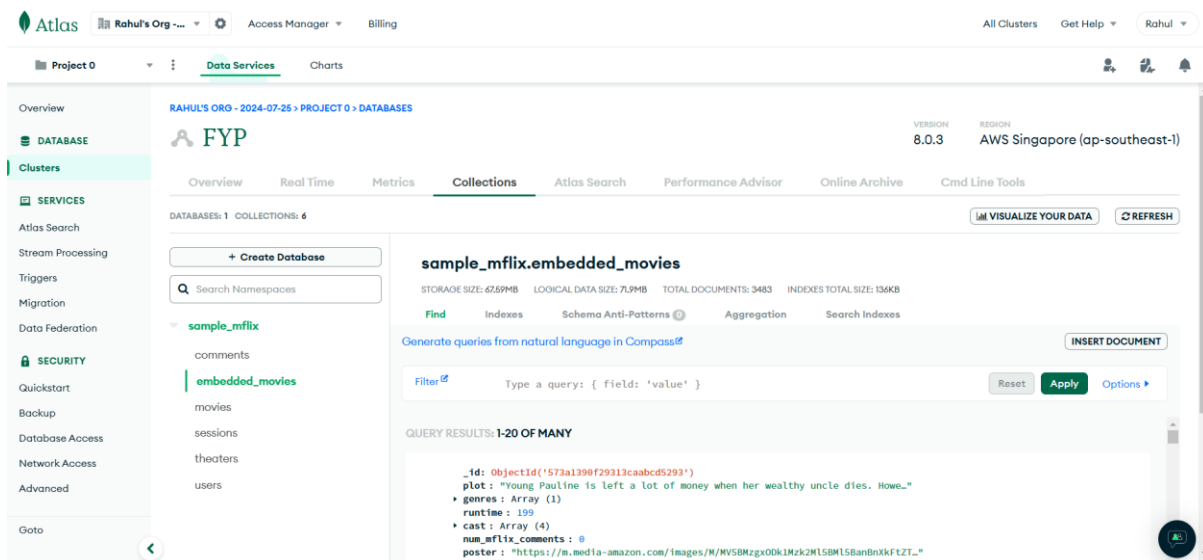


Figure 2.14 MongoDB Atlas web interface

Figure 2.14 depicts the display of MongoDB Atlas which is the web interface of MongoDB which is used to manage clusters, databases and collections. It also has features to analyse data through graphical depictions. For a free subscription MongoDB provides its users with a storage size up till 512 MB, and with access to a single cluster which will be shared on a single server across a region (based on my case, it would be AWS Singapore).

2.3.1.5 Docker (Docker Desktop)

Docker is a platform for containerization that enables developers to package applications and their dependencies into portable containers. These containers ensure that the software runs consistently across various environments, eliminating the "it works on my machine" problem. Containers are lightweight because they share the host operating system's kernel while isolating the application's processes, making them faster and more efficient than traditional virtual machines. Docker also provides tools for creating, managing, and deploying these containers effectively, making it a cornerstone of modern software development and DevOps practices ("What Is Docker?," 2024).

Advantages of Docker

Portability and Consistency:

Docker ensures that applications packaged within containers operate consistently, regardless of the underlying environment. This eliminates issues stemming from environment-specific dependencies, which can often lead to compatibility problems.

Resource Efficiency:

Containers are designed to be lightweight, consuming minimal system resources compared to virtual machines. This allows developers to maximize hardware utilization, reduce overhead, and run multiple containers on a single system efficiently.

Scalability:

Docker integrates seamlessly with orchestration tools such as Kubernetes, enabling projects to scale horizontally by adding more containers as demand increases. This makes it easier to handle fluctuations in workload, which is particularly useful for dynamic systems like the AI-powered package theft detection.

Faster Deployment:

The simplicity of creating and running Docker containers accelerates the deployment process. Developers can deploy applications in seconds, facilitating rapid iterations and updates in continuous integration/continuous delivery (CI/CD) pipelines (*"What Is Docker?"*, 2024).

Simplified Configuration:

Docker encapsulates all necessary configurations within the container, minimizing setup errors and ensuring applications behave predictably across different environments.

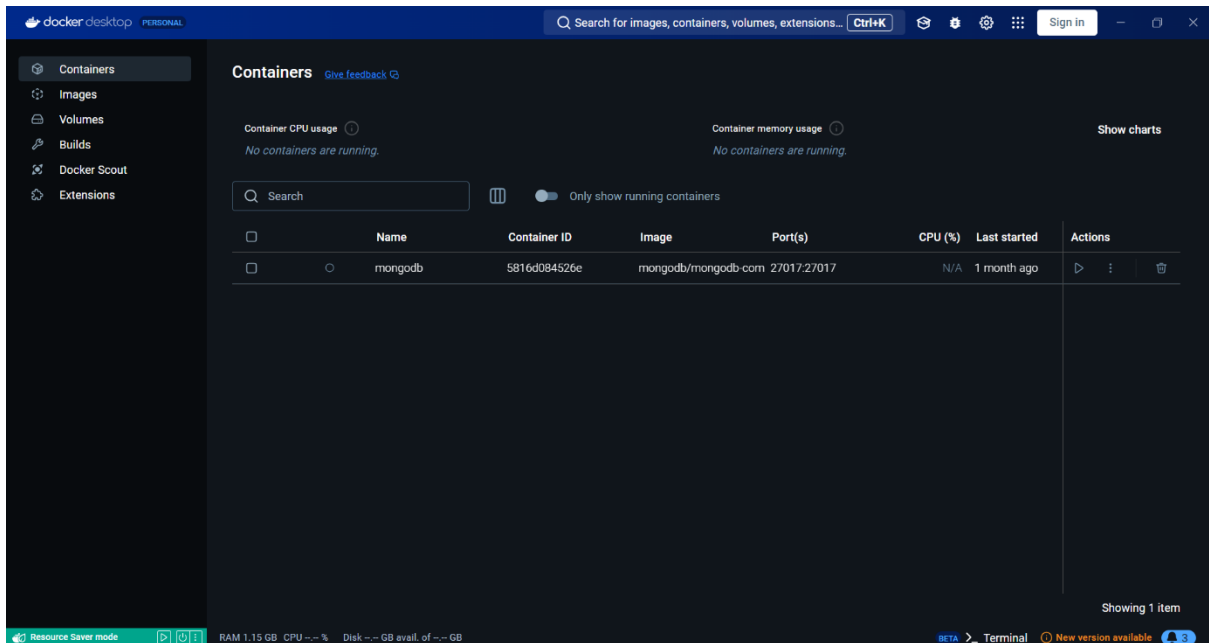


Figure 2.15 Docker Desktop Interface with MongoDB container on standby

Based on Figure 2.15, this is the interface of Docker Desktop that will be used for the proposed system, and at the moment there is a MongoDB container that is on standby for hosting a MongoDB server locally if the online server of MongoDB is not responsive.

2.3.1.6 Figma

Figma is a tool used to create, share, and test designs of websites, mobile apps, and many other digitalized products across multi platforms (*What Is Figma?*, n.d.). It is popular among designers, product managers, writers and developers as it helps in the designing process and allows a more efficient feedback medium for faster decision making.

One of its standout features is real-time collaboration. Being cloud-based, Figma allows multiple users to work on the same design simultaneously. This capability fosters seamless teamwork among designers, developers, and stakeholders, streamlining workflows and ensuring alignment across teams. Additionally, Figma's platform independence ensures compatibility with various operating systems, as it runs directly in a web browser. This

eliminates the need for downloads or specific hardware, making it accessible on Windows, macOS, and Linux devices. Furthermore, Figma supports responsive design through features like auto-layout and constraints. These tools simplify the creation of designs that adapt smoothly to different screen sizes, ensuring optimal appearance and functionality across devices (GeeksforGeeks, 2024b).

The suitability of this with my proposed system is that it will allow me to plan and design the looks of the web interface of the system as well as get feedback from my supervisor.

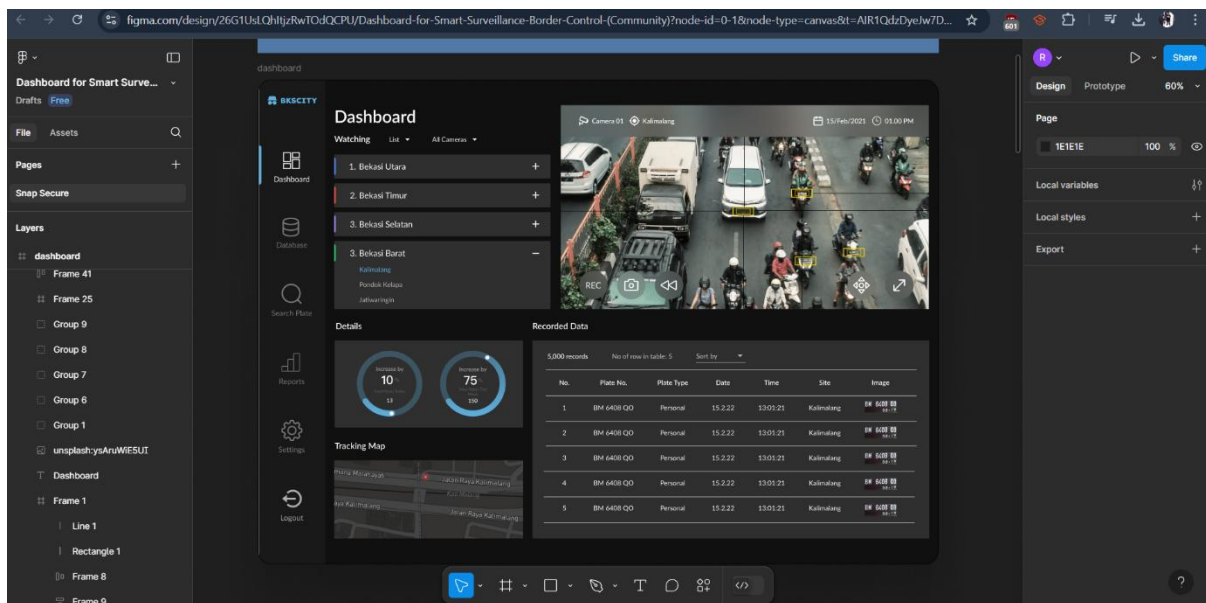


Figure 2.16 Figma online Interface

Based on Figure 2.16, this is a template that will be used as reference when designing the looks of the system's web interface for the proposed system.

2.3.2 Hardware

2.3.2.1 HP 14s-dq2515tu

The HP 14s-dq2515tu is a compact and efficient laptop powered by the Intel Core i5-1135G7 processor, which features four cores and eight threads, delivering up to 4.2 GHz speed. It is equipped with 8 GB DDR4 RAM, expandable to 16 GB, and a 512 GB PCIe NVMe SSD, ensuring quick boot times and ample storage for project files. The device includes Intel Iris Xe

Graphics, suitable for light graphical tasks and visualization. Its 14-inch Full HD display with anti-glare properties is ideal for extended working sessions, complemented by a lightweight build of just 1.46 kg, enhancing portability (Official HP® Support, n.d.).

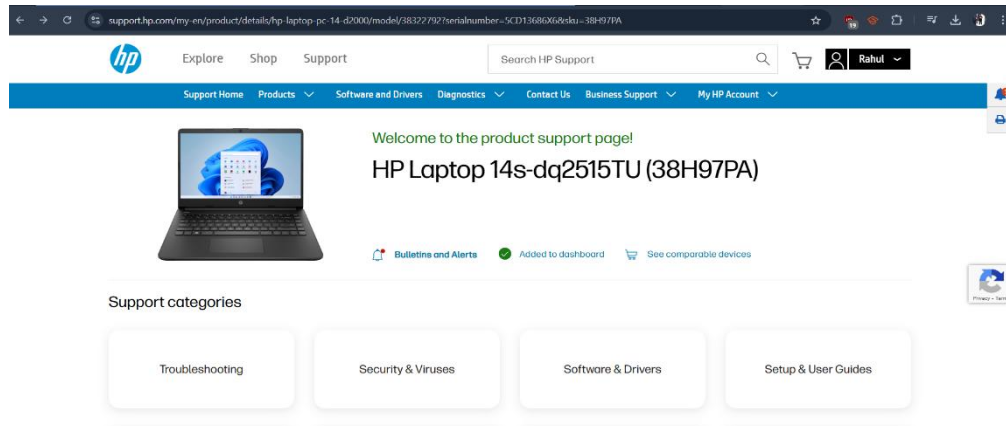


Figure 2.17 HP14s Specification on the official HP Support Website

Based on Figure 2.177, the official HP Support website displays detailed information about my machine, such as OpenCV support, ML support and other Deep Learning supported features which allows this machine to conduct development for the proposed system.

Intended use of the machine:

Designing: The device's Intel Iris Xe Graphics can handle design software like Figma for prototyping and UI/UX development. Its portability makes it easier to collaborate remotely or in shared spaces.

Coding: The laptop's processor and RAM capacity are sufficient for multitasking between Integrated Development Environments (IDEs) like Visual Studio Code and database management tools like MongoDB.

Implementation: For AI model development and testing, the laptop can manage Python-based frameworks like Flask. Additionally, its SSD facilitates faster loading times for large datasets or code libraries, ensuring efficiency during development.

CHAPTER 3: REQUIREMENT ANALYSIS AND DESIGN

3.1 Introduction

This chapter outlines the research methodology utilized in the development of the proposed AI-Powered Package Theft Detection System. The chosen methodology for this project is the Scrum framework, a prominent Agile methodology known for its iterative process and flexibility. Scrum is preferred for its ability to adapt to changes and improve the development process continually, which aligns well with the evolving nature of the proposed system. The flexibility of Scrum ensures that the development of the system remains responsive to feedback and updates, providing room for adjustments as the project progresses (Hidalgo, 2019). The application of Scrum in this FYP allows for an organized yet flexible approach, involving multiple phases such as planning, development, testing, and review. The structure of Scrum enables close collaboration between the development team and the supervisor, facilitating constant feedback and iterative improvements for the AI-based detection system. The contents of this chapter will behold several subsections such as the Scrum Roles, Scrum Events, Scrum Artifacts and Agile Methodology. This is then followed by the Requirement analysis such as the Analysis on similar Existing Systems and Analysis of Proposed Systems. After that, it would be followed by the Questionnaire Analysis, Software and Hardware requirement analysis. After the analysis phase, it further discusses on the System Design where Scenario Illustrations, Web Application Architecture, Flow Chart, Context and Data Flow Diagrams, followed by the discussions on the Entity Relationship Diagram (ERD) and the High-fidelity Wireframe Design. Lastly, to complete the process of utilizing the Scrum framework, brief discussions on the Development, Testing, Deployment and Review phase is scrutinized.

3.2 Scrum

The Scrum framework, a subset of Agile methodologies which is known for its significant flexibility which embraces room for enhancing the development of the AI-Powered Package Detection System. This proposed system has been structured to be flexible yet an efficient approach as showcased in the adaptability to change and continuous improvement (Verwijs & Russo, 2022).

3.2.1 Scrum Roles

This is consistent with the development of the proposed system since the feedback from the supervisor has given the opportunity to look back and review the project to see if it meets the requirements, this is aligned with the roles in Scrum which are the Product Owner that defines the vision and requirements of the AI system. This is heavily represented as there is a close collaboration between the supervisor and the development team that guides the projects direction.

The next role would be the development team who is mainly responsible for developing the proposed system which consists of the student itself, hence as with a diverse skill set allowing a variety of responsibilities, it proves that there is a must in delivering a shippable project, in this case would be the proposed system.

3.2.2 Scrum Events

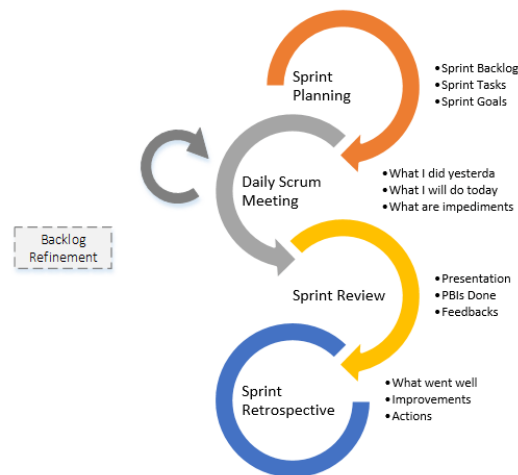


Figure 3.1 Scrum Event Flow (Miu, 2015)

Based on Figure 3.1, it illustrates the events which are normally implemented during the project development phase, which is the practice being applied during the whole proposed system planning. This assumption is consistent with the Sprint planning where at the beginning of each sprint, a planning session for the selected items from the backlog will be obtained, this is where the scenario during the development of the proposed system, right before moving onto the next requirement, there will be a planning session which will be brought forward for review to the supervisor to get a confirmation of the steps that are about to be taken is synchronous with the requirements of the project of the proposed system.

The daily scrum meeting when put in motion after the planning phase is considered in sync as there is always an early stage of each daily session during the project development where all the steps from previous works are reviewed, acknowledged and proceeded, in the scenario of the project of the proposed system, all the steps taken the day before are checked to ensure that all parts have been completed, then reviewing if there are any obstacles which were faced during the proposed system creation which had caused any diversion and deviation of the project from the actual plan. This in hand promotes the transparency within this work ethic system which allows quick identification and resolution of issues and ensuring the project stays on track (Salimi, n.d.-b).

This is followed by the Sprint Review, where at the end of each sprint in this proposed system, it is presented to the supervisor to be given feedback and to ensure that the project results does align with the stakeholder's expectations. This progresses to providing a working prototype of the package detection system by the end of development and gathering input for further improvements and refinements.

Finally, a Sprint Retrospective is put into motion as it allows the development process to be reflected, reviewed, and contemplated to find out how to improve efficiency for each future sprint proceeding. This would show continuous commitment for improvement which is a vital part for ensuring that this proposed project turns out to be a successful one.

3.2.3 Scrum Artifacts



Figure 3.2 Scrum Artifacts (Salem, 2024)

According to Figure 3.2, it is referred to as the Artifacts which closely relates to its 3 major components which are the product backlog, sprint backlog and increment. This artifact is the tool used by Scrum teams to represent work which aligns with the proposed system which is constantly being reported, and the requirements are constantly managed in minor requirements.

According to Salem (2024), the Product Backlog would refer to the ordered list of everything which is needed to improve the product, this has also become a part of the proposed system's life cycle where it has become the one and only working reference which is constantly

checked again, which in this case, the FYP 1 report is constantly checked and reviewed as a type of Product Backlog.

The next component would be the Sprint Backlog which would mainly come into play for the proposed system during the development phase as it will ensure that the project is completed based on the requirements listed.

The last component would be the Increment which is the functionalities or new inclusive features that is added to the product during each sprint, which aligns with the project as it shows that the proposed system is always evolving and adapting to changes without much issues if it has add-ons.

3.2.4 Agile Methodology

Acknowledging that Scrum is a part of the Agile methodology, gives purpose of listing the possession of 6 processes which are Requirements, Design, Development, Testing and Review, this is depicted in Figure 3.3. All these processes will be further explained in the later sections.



Figure 3.3 Agile Methodology Cycle (Laoyan, 2024)

3.3 Requirements

In the requirements phase, the project's foundation is established by gathering and defining the essential features and goals of the AI-powered package detection system. This includes identifying the need for package theft detection and real-time alert, which gives room to understand the challenges of package theft and setting clear objectives.

3.3.1 Analysis on Similar Existing Systems

Existing package theft detection systems, such as CVEDIA Package Detection, Deep Sentinel, and Veesion, showcase distinct functionalities that address theft prevention. CVEDIA leverages synthetic data for AI model training, enabling high accuracy under varied conditions while adhering to privacy standards. Its real-time package detection and flexible deployment options demonstrate scalability for residential and commercial use. In contrast, Deep Sentinel incorporates a hybrid approach, integrating AI-driven monitoring with live human intervention. Its system proactively addresses threats through real-time analysis and immediate guard actions, ensuring minimal delay in response. Meanwhile, Veesion employs advanced gesture recognition technology to identify suspicious behavior, emphasizing compatibility with existing surveillance infrastructures. These systems excel in automation, real-time alerts, and adaptability, catering to diverse security needs.

However, the analyzed systems also reveal gaps that the proposed AI-Powered Package Theft Detection System aims to address. While CVEDIA and Veesion focus on AI-driven automation, they lack a tailored emphasis on package-specific theft. Deep Sentinel's reliance on human intervention adds operational costs, reducing accessibility for budget-conscious users. Moreover, subscription-based models in these systems can be a barrier for widespread adoption in households and small businesses. The proposed system, inspired by the best practices of these technologies, aims to integrate real-time object detection, enhanced alert mechanisms, and a web-based dashboard without ongoing subscription fees. By addressing

these limitations, the proposed solution seeks to offer an affordable and efficient alternative to existing systems.

3.3.2 Analysis of Proposed System

The proposed system (Snap Secure) stands out through its focus on detecting and addressing package theft scenarios in real-time. By incorporating pre-trained models and potential inclusion of synthetic datasets, the system ensures robust performance without extensive data collection efforts. Its emphasis on detecting packages, unauthorized access, and suspicious behaviours provides a targeted approach, unlike traditional systems that often rely on generalized object detection. The web-based dashboard offers users accessibility across devices, enabling live monitoring, incident logging, and prompt alerts. By eliminating human intervention and focusing on automation, the system delivers efficiency while minimizing false alarms, making it ideal for residential and small business environments.

In terms of innovation, the proposed system tries and adopts features from successful models like CVEDIA's synthetic data utilization and Veesion's real-time alert capabilities. Unlike its counterparts, it avoids subscription-based models, enhancing affordability and user adoption. The integration of technologies such as TensorFlow for object detection and MongoDB for efficient data storage ensures scalability and performance. Moreover, the system's edge deployment capability reduces latency and dependency on cloud resources, enhancing reliability. Designed to be a cost-effective, user-friendly solution, the system addresses critical gaps in existing theft detection technologies, providing enhanced security tailored to modern e-commerce and package delivery challenges. The proposed system as depicted in Table 3.1, whereby the user may conduct a series of functionality task such as account creation and login, viewing live feed, viewing archive, viewing statistics and report and acting on the detections.

Table 3.1 Functionalities of the proposed system (SnapSecure)

Homeowner / User (Web Application)
• Create Account • Login Account • View Live Feed • View Archive • View Statistics and Report • Act on detections

3.3.3 Questionnaire

A Questionnaire was distributed to a crowd of 51 people with the aim of accumulating intel on the proposed system which is a survey on a AI-Powered Package Detection System to Reduce Porch Piracy and Improve Package Safety. The accumulated results were to acquire feedback and expectations from potential users from the proposed system, SnapSecure. It was also carried out to determine the relevance of the system to the market and its consumers. The questionnaire comprised of 4 different sections,

Section A. Respondents Demographic

Section B. Current Concerns About Package Theft

Section C. Preferences for a Smart Package Detection System

Section D. User Experience and Feedback

3.3.3.1 Respondents Demographic

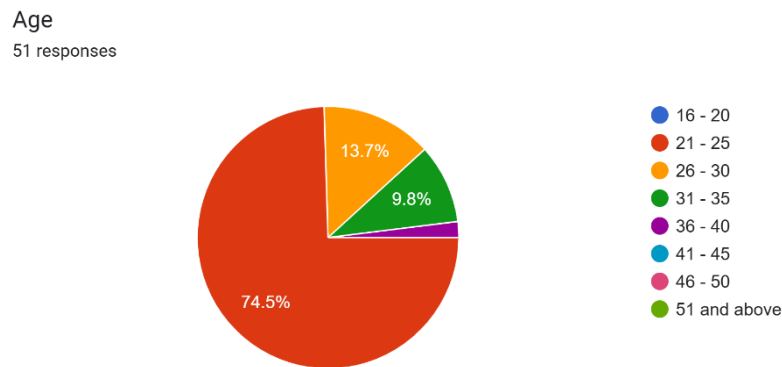


Figure 3.4 Age of Respondents

Amongst the 51 respondents, the majority, 38 individuals (74.5%), are aged between 21 and 25 years. This is followed by 7 respondents (13.7%) aged between 26 and 30 years. A smaller group of 5 respondents (9.8%) fall within the 31 to 35 age range, while only 1 respondent (2%) is aged between 36 and 40 years. There were no responses recorded for the other age groups, including 16–20, 41–45, 46–50, and 51 and above. The data highlights that the most represented demographic among the respondents is the 21–25 age group (Figure 3.4)

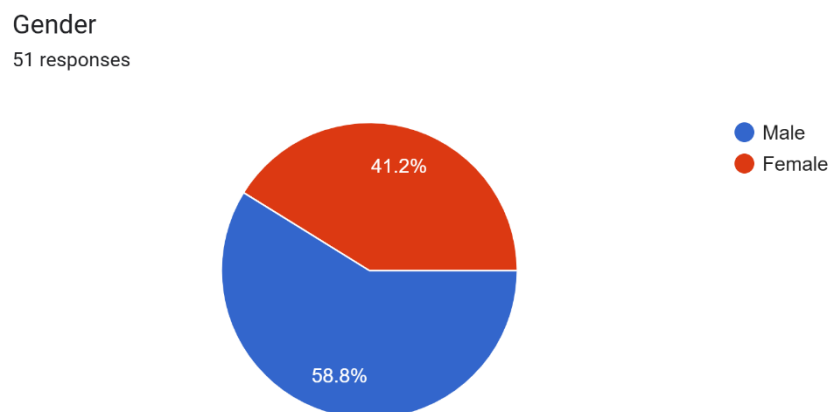


Figure 3.5 Gender of Respondents

From the data collected from 51 respondents (Figure 3.5), the gender distribution shows that the majority, 30 respondents (58.8%), are male, while the remaining 21 respondents (41.2%) are female. This indicates a slight predominance of male participants compared to female participants in the survey.

State

51 responses



Figure 3.6 State Origin of Respondents

Among the 51 respondents (Figure 3.6), the majority, 23 individuals (45.1%), are from Sarawak. This is followed by 9 respondents (17.6%) from Perak and 7 respondents (13.7%) from Johor. Additionally, 4 respondents (7.8%) are from Selangor, 3 respondents (5.9%) are from Sabah, 2 respondents (3.9%) are from Penang, and 1 respondent each (2%) is from Perlis, Pahang, and Negeri Sembilan. Other states represented include Kedah, Kelantan, Terengganu, Melaka, and various federal territories, though these have fewer participants. The data highlights that Sarawak has the highest representation among the respondents, followed by Perak and Johor.

Primary Residence Type:

51 responses

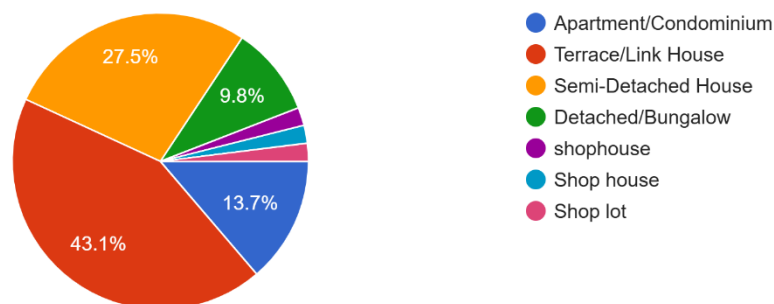


Figure 3.7 Primary Residence Type of Respondents

Based on Figure 3.7, the primary residence types of 51 respondents are Terrace/Link Houses, accounting for 43.1% of the responses, which equates to 22 respondents. Apartment/Condominiums follow closely with 27.5% of the responses, representing 14

respondents. Semi-Detached Houses represent 13.7% of the responses, or 7 respondents. Detached/Bungalows make up 9.8% of the responses, which is 5 respondents. Shop Houses (shophouse, Shop house, and Shop lot combined) only account for 3 out of 51 responses, representing a total of 5.9% of the respondents.

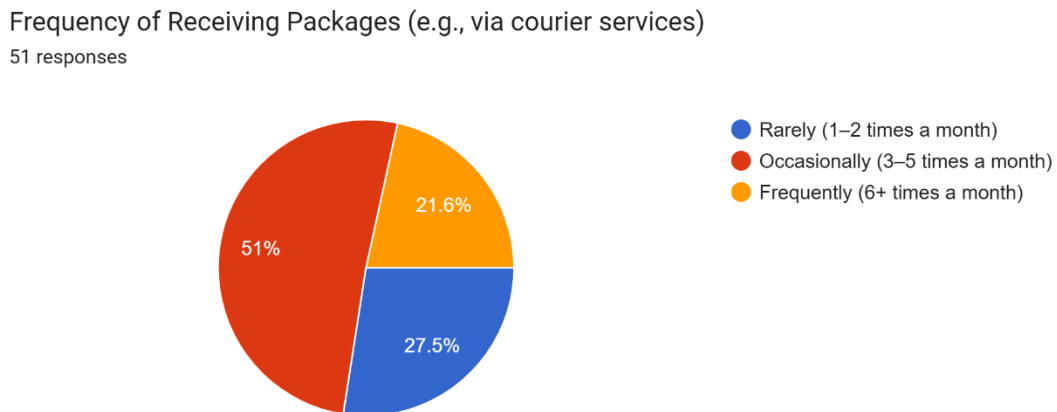


Figure 3.8 Respondent Frequency of Receiving Packages

Based on Figure 3.8, 51 respondents were surveyed about the frequency of receiving packages (e.g., via courier services). Most respondents, 26 respondents (51%), receive packages occasionallly (3-5 times a month). A significant portion, 14 respondents (27.5%), receive packages frequently (6+ times a month). A smaller proportion, 11 respondents (21.6%), receive packages rarely (1-2 times a month).

3.3.3.2 Current Concerns About Package Theft

Have you or someone you know experienced package theft?
51 responses

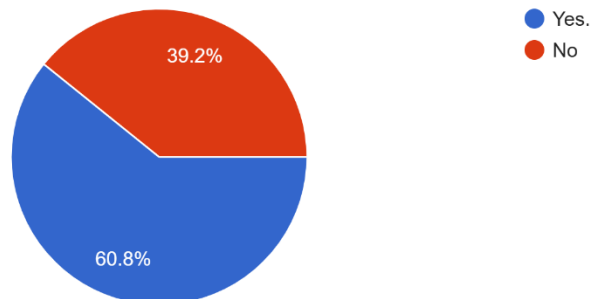


Figure 3.9 Knowledge of Respondents of Package Theft Victims

Based on Figure 3.9, 51 respondents were surveyed about their experience with package theft. 60.8% (31 respondents) indicated that they or someone they know has experienced package theft. 39.2% (20 respondents) reported that they and their acquaintances have not experienced package theft. In summary, the data suggests that package theft is a concern for a significant portion of the surveyed population.

How concerned are you about package theft in your area?
51 responses

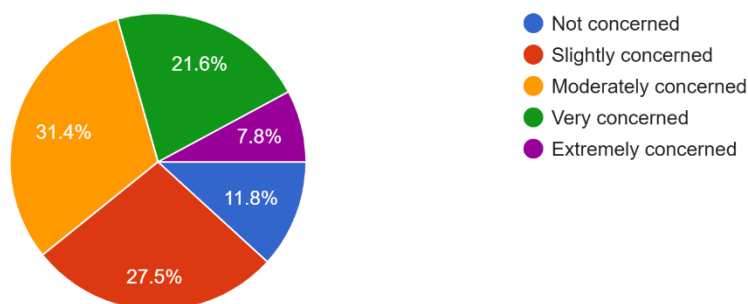


Figure 3.10 Respondent Concern on Package Theft in Their Locality

Based on the responses from 51 individuals, the level of concern about package theft in their area varies. A significant proportion, 16 respondents (31.4%), are very concerned about package theft. Additionally, 11 respondents (21.6%) are extremely concerned. On the other hand, 6 respondents (11.8%) reported being not concerned. A smaller number of respondents, 4 (7.8%), are slightly concerned, and 14 (27.5%) are moderately concerned (Figure 3.10).

Overall, majority of the respondents equalling 27 (53%) are either very or extremely concerned about package theft in their area, indicating that this is a significant concern for many residents.

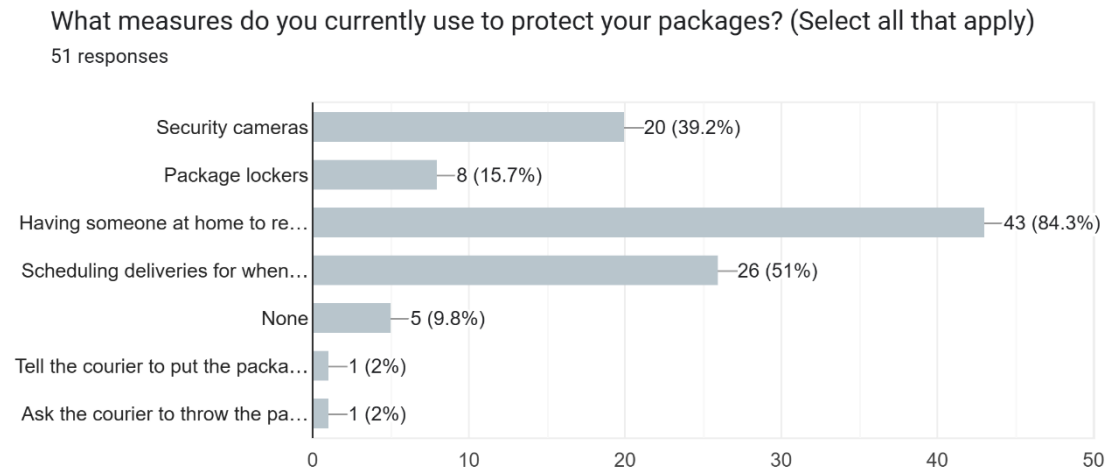


Figure 3.11 Respondents Current Package Protection Method

Based on Figure 3.11, 43 (84.3%) respondents which is most of the respondents prefer having someone at home to receive deliveries as one of their package protection methods. 26 respondents (51%) would also include scheduling deliveries for when they are home. Followed by 20 (39.2%) of respondents would rely on security cameras as their method of protecting their packages. On the other hand, 8 (15.7%) of respondents would prefer package lockers as their method of protecting packages. The last protection method indicates that 2 (4%) of respondents would request courier services to place the packages within their home's compound. Finally, there are 5 (9.8%) of respondents who do not employ any protection method for their packages. Hence, many respondents use multiple protection measures, with human presence being the most relied-upon method.

3.3.3.3 Preference for a Smart Package Detection System

What features would you find most useful in a smart package detection system? (Select up to 3)
51 responses

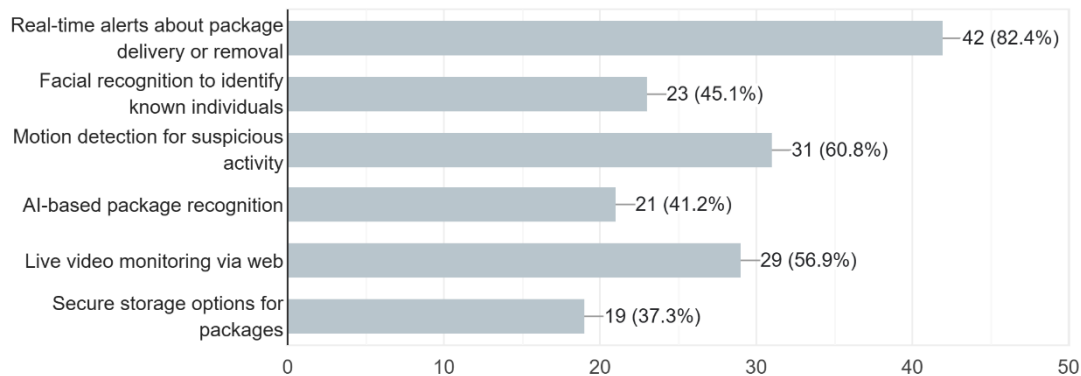


Figure 3.12 Features Respondents Presumed to Useful for A Package Detection System

According to Figure 3.12, majority of respondents, 42 (82.4%), find that the real-time alerts about package delivery/removal would be one of the most useful features for the package detection system. This is followed by motion detection for suspicious activity having 31 respondents (60.8%) which is the second most presumed useful feature of a smart package detection system. The third most selected feature of a smart package detection system that is assumed to be most useful would be the live video monitoring via web, where 29 (56.9%) of respondents had selected it. Facial recognition, AI-based package recognition, and secure storage option for packages are the 4th, 5th and 6th presumed features to be most useful in a smart package detection system where each respectively have, 23 (45.1%) respondents, 21 (41.2%) respondents, and 19 (37.3%) respondents. This depicts that many residents would

prefer an approach which will be able to oversee, detect and alert them on their package safety as opposed to implementing a physical storage system.

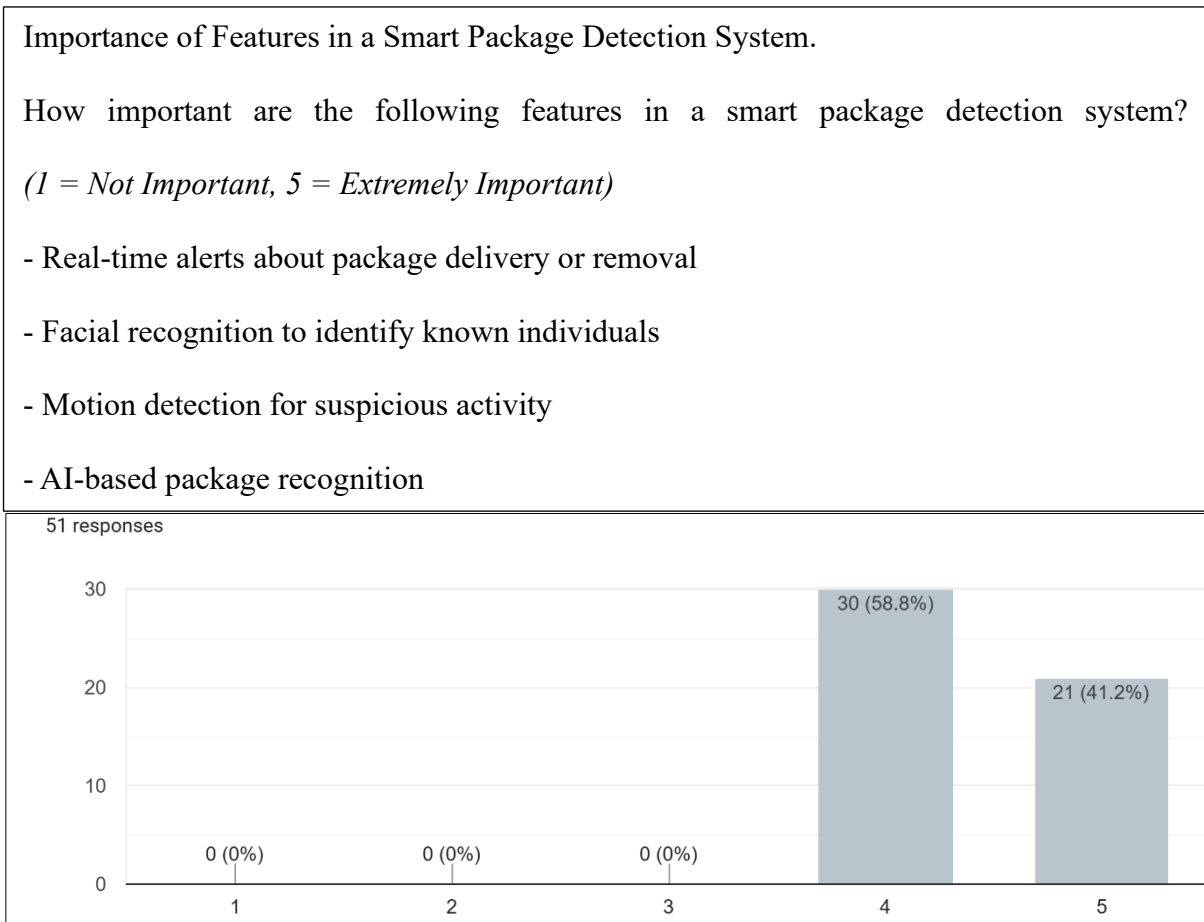


Figure 3.13 Overall Importance of all Features

Based on Figure 3.13, the overall importance of all the features are skewed towards the 5th linkert score which is extremely important, whereby 30 (58.8%) of respondents responded with a score of 4, while 21 (41.2%) of respondents responded with a score of 5. This indicates that the level of importance of all these features are seen as vital to the respondents.

Affordability vs. Feature Balance How satisfied would you be with a system that balances affordability and advanced features? (1 = Not Satisfied at All, 5 = Extremely Satisfied)

51 responses

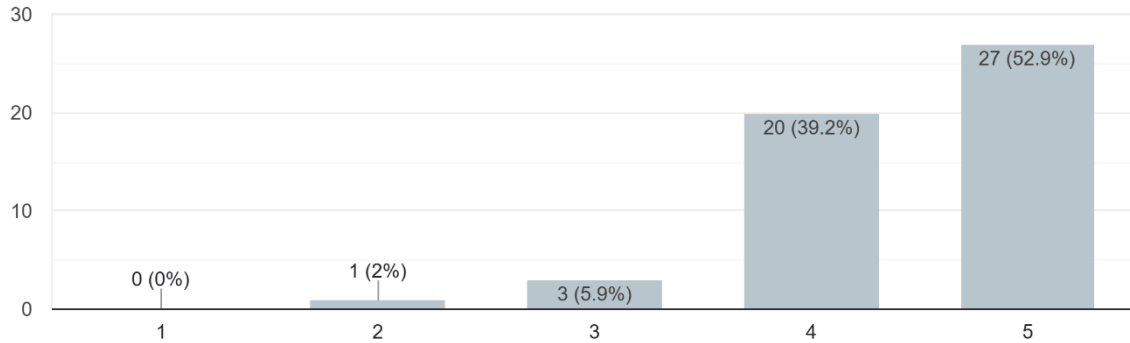


Figure 3.14 Affordability vs. Feature Balance

According to Figure 3.14, regarding the balance of affordability and advanced features in an AI-powered package theft detection system with 51 respondents. The largest group, 27 individuals (52.9%), rated their satisfaction at the highest level (5), indicating a strong preference for a system that achieves a balance between affordability and advanced features. This is closely followed by 20 respondents (39.2%) who rated their satisfaction level as 4, further emphasizing positive reception to such a system. A smaller proportion, 3 respondents (5.9%), selected a neutral rating of 3, indicating neither satisfaction nor dissatisfaction. Only 1 respondent (2%) gave a low satisfaction score of 2, and no respondents selected the lowest satisfaction level of 1. Overall, the data demonstrates that an overwhelming majority of respondents (92.1%) are satisfied or extremely satisfied with a system that balances affordability and advanced features. This indicates that such a balance aligns well with user preferences, making it a compelling aspect of the AI-powered package theft detection system.

Willingness to Use a Smart Package Detection System How likely are you to adopt a smart package detection system if it addresses your package safety concerns? (1 = Very Unlikely, 5 = Very Likely)
51 responses

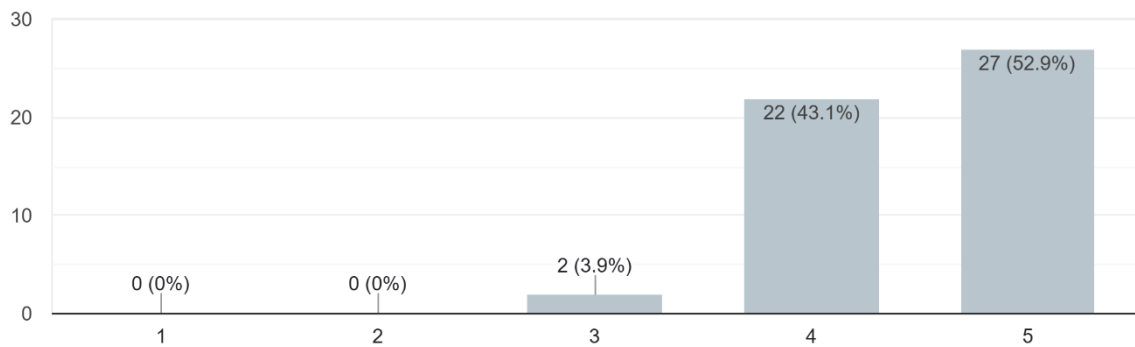


Figure 3.15 Willingness of Respondents to Adopt The System

According to Figure 3.15, it indicates that 27 respondents (52.9%) show very likely willingness to utilize the smart package detection system considering its ability to address the package safety concerns. This is followed by 22 respondents (43.1%) who indicated high likelihood of utilizing the smart package detection system. However, only 2 respondents (3.9%) would be on the neutral side of the decision to adopt the smart package detection system. Hence, this shows a significant indicator that there is a high market acceptance for such a system.

Perceived Effectiveness of a Smart Package Detection System How effective do you believe a smart package detection system would be in reducing...age theft? (1 = Not Effective, 5 = Very Effective)
51 responses

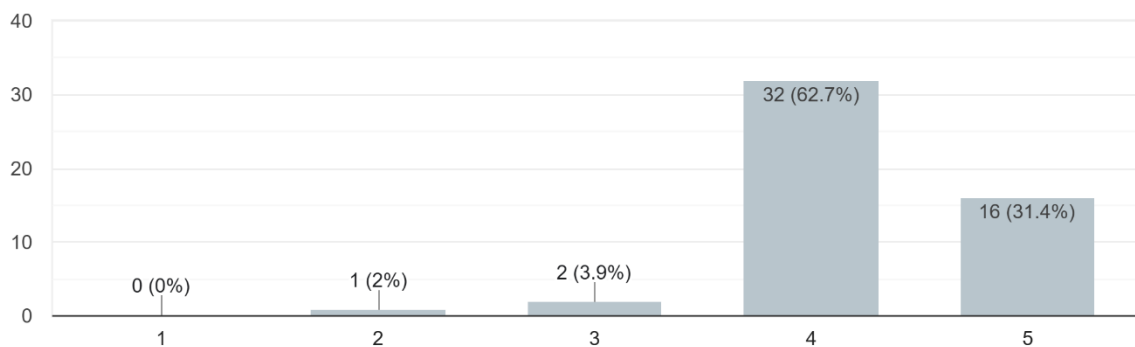


Figure 3.16 Respondents Perceived Effectiveness of The Smart Package Detection

Based on Figure 3.16, Out of 51 respondents, the majority, 32 individuals (62.7%), rated the effectiveness of a smart package detection system as 4, suggesting they believe the system would be highly effective in reducing package theft. This is followed by 16 respondents

(31.4%) who rated the system's effectiveness as 5, indicating very strong confidence in its capability. A small group of respondents expressed lower confidence in the system, with 2 individuals (3.9%) selecting a rating of 3, and only 1 respondent (2%) rating its effectiveness as 2. Notably, no one chose the lowest rating of 1. Overall, the data highlights that 94.1% of respondents perceive the system as effective or very effective, showcasing strong user belief in the system's potential impact.

3.3.3.4 User Experience and Feedback

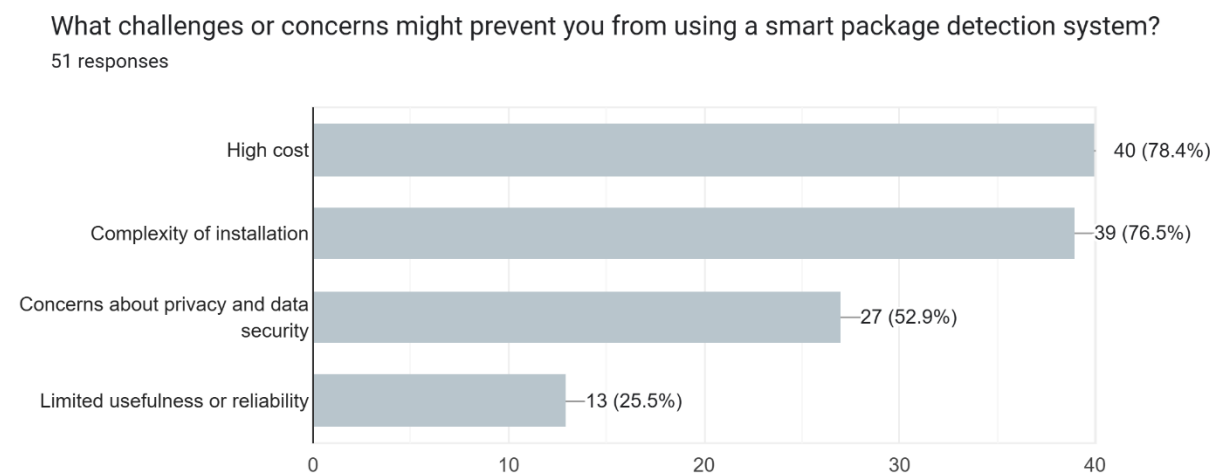


Figure 3.17 Challenges Preventing usage of The Smart Package Detection System

Figure 3.17 illustrates that among the 51 respondents, the most significant concern was the high cost of the system, with 40 individuals (78.4%) identifying this as a potential barrier. Closely following this, 39 respondents (76.5%) highlighted the complexity of installation as another major challenge. Concerns about privacy and data security were also prominent, with 27 individuals (52.9%) expressing apprehension about these issues. Additionally, a smaller group of respondents, 13 individuals (25.5%), raised concerns about the system's limited usefulness or reliability. These findings indicate that while the system is perceived positively in terms of effectiveness, addressing cost, installation ease, and privacy concerns will be critical to widespread adoption.

Would you prefer a system that integrates with your existing security setup (e.g., cameras, doorbell systems)?
51 responses

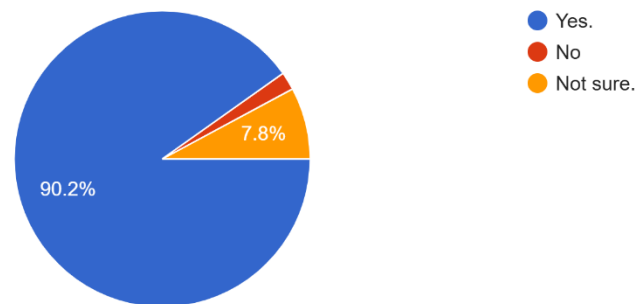


Figure 3.18 Integration with Existing Setup Preference by Respondents

A large majority of respondents (Figure 3.18), 46 individuals (90.2%), expressed a preference for a smart package detection system that integrates with their existing security setups, such as cameras and doorbell systems. A small minority, 4 respondents (7.8%), stated they were unsure, while only 1 respondent (2%) does not want to integrate it within a pre-existing setup. This data underscores the importance of ensuring compatibility with current security technologies to meet user expectations.

Voice Recognition Capabilities

Do you have any additional features or suggestions for such a system? (Optional)
(Open-ended question)

Figure 3.19 Open Ended Feature or Suggestions for the System

According to Figure 3.19, one of the suggestive features would be a voice recognition capability which would add a layer of sophistication and personalized functionality to the smart package theft detection system. By integrating voice recognition, the system could authenticate homeowners or authorized individuals, preventing unauthorized access to sensitive security settings. For instance, users could provide voice commands to disarm the system, access logs, or retrieve real-time notifications. This feature would also enable hands-free operation, making the system more accessible and user-friendly, particularly for individuals with physical limitations. Additionally, voice recognition could be combined with artificial intelligence to

detect unusual or distressed voices, potentially identifying theft attempts or emergencies and alerting authorities or homeowners promptly.

Automatic Reporting of Theft Attempts to Authorities

According to Figure 3.19, an automatic reporting feature would significantly enhance the system's effectiveness by ensuring immediate action in the event of a theft attempt. When a potential theft is detected, the system could instantly alert local law enforcement with details such as video footage, timestamps, and location coordinates. This proactive approach minimizes response time and increases the likelihood of apprehending perpetrators. Additionally, the system could notify the homeowner simultaneously, keeping them informed about ongoing incidents. Such automation not only deters thieves but also provides users with peace of mind, knowing that theft attempts are handled swiftly and efficiently without requiring manual intervention.

3.3.4 Software Requirements

Table 3.2 Software Requirements of The Proposed System

Software	Purpose
Spyder IDE	Development environment for Python programming. Advanced editing, debugging, and profiling; integrates with libraries like NumPy, CV2, and Matplotlib for ML tasks.
TensorFlow	AI framework for training and deploying ML models. Pre-trained model support (YOLO, SSD), scalability for edge/cloud deployment, and real-time inference.
VS Code (Flask)	Runs Lightweight Python web framework for web application development. Simplifies development of web-based dashboards and APIs.
HTML, CSS, JS	Frontend technologies for creating the system's interface. Structured layouts (HTML), visual styling (CSS), and interactivity/real-time updates (JavaScript).
MongoDB	Database for storing and managing unstructured real-time data. Supports JSON documents, high-speed indexing, time-series data handling, and scalability.
Docker	Containerization platform to ensure consistent deployment. Portability, scalability, and simplified application configurations.
Figma	Design tool for prototyping the user interface of the web application. Cloud-based collaboration, responsive design features, and cross-platform compatibility.

Table 3.2 depicts the decided software and their purposes that will be utilized for the development of the proposed system.

3.3.5 Hardware Requirements

Table 3.3 Hardware Requirements for the Development Stage

Hardware	Purpose
Laptop (HP 14s-dq2515tu)	Primary development machine for coding, testing, and deploying the system. Intel Core i5 processor, 8GB RAM (expandable), 512GB SSD, Intel Iris Xe Graphics for light visualization tasks.
8GB RAM	Deployment hardware for running AI models in real-time. Supports low-latency processing, reduces dependency on cloud resources. It will be used for running the programs real time.
Intel Iris Xe Graphics	Handles AI Image processing when the image goes through the AI model for detection.

Based on Table 3.3, the listed hardware components above would be responsible for enabling tasks to be carried out for developing the proposed system.

3.4 Design

By abiding by the Scrum framework, the next stage would be the designing stage. This stage is considered crucial as it holds the significance of designing the system that will be further developed on future stages. Hence, there are several designs that will be able to encompass all the requirements and visualise the approach taken to develop the proposed system, which are designs such as Scenario Illustration, Web and Software Architecture, Flow Chart Diagram, Context Diagram, Data Flow Diagram (DFD), Entity Relationship Diagram

(ERD), and Data Dictionaries. This is then followed up with a High-fidelity wireframe design which is a part of the planning process in scrum.

3.4.1 Scenario Illustration

3.4.1.1 Scenario 1 – Overview and Initialization (Live Monitoring)

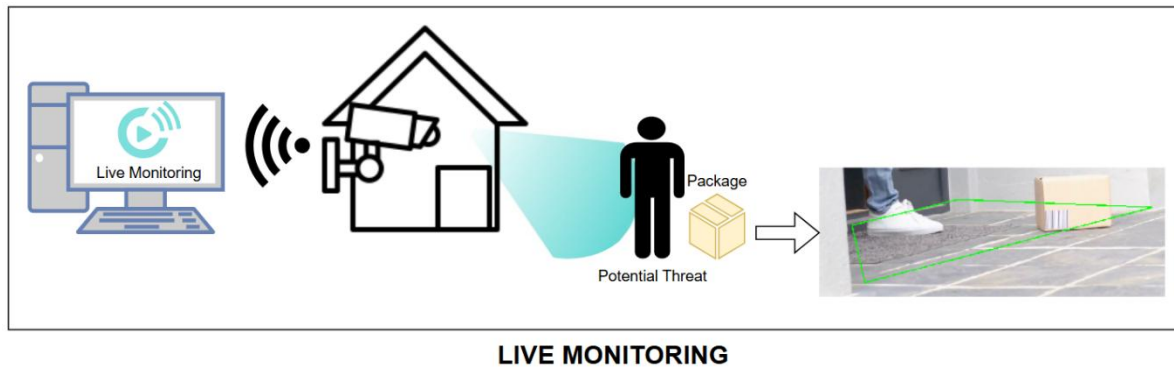
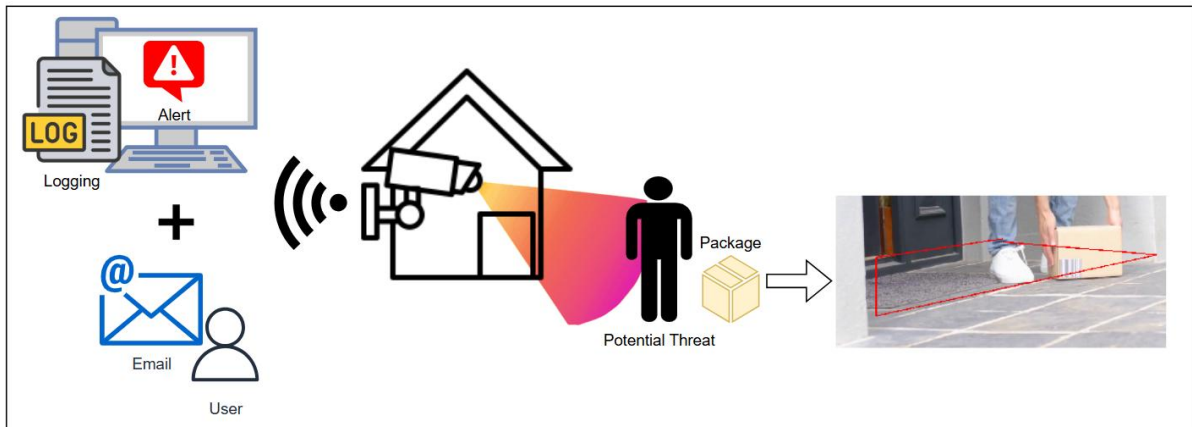


Figure 3.20 System Functionality Overview

According to Figure 3.20, it depicts a straightforward appearance of the system when it has been initialized and functioning. Hence, as seen from the left side of the figure, it is the web application which is receiving live feed from the surveillance camera at the home of the user, which is actively doing monitoring on the package delivered to the house, which can be seen on the most right image where there is a Region Of Interest (ROI), which has been drawn to mimic the area of expectant for the delivery package to be delivered in and it will be detected by the system to be monitored continuously. It also shows that the ROI may be any size and shape allowing for more customizability.

3.4.1.2 Scenario 2 – Detection and Response to Stimuli (Event Trigger)



EVENT TRIGGER

Figure 3.21 System Response to Event Triggers

Figure 3.21, illustrates the system response to events that triggers the detection system. This response can be seen broken down into several parts where it begin with the detection if the package has been tampered, removed or altered from the ROI, this can be achieved by several methods of approach, one of it being the Intersection over Union (IoU), which takes into account the bounding boxes of the package that has been detected, and the ROI, whereby, if the IoU threshold is met during the placement phase of the package, the system would detect that there is a package within that ROI, and would be able to keep track if the threshold changes significantly negative in events of the package being removed.

In the case of the package being altered or removed happens, it will then cause the system to respond to the stimuli by sending the alert to the Web Application, which will then log the details of the event, which is then followed by a notification being sent to the user via email, to alert the user that there is an event which requires the user to review and take action.

3.4.2 Web Application and Software Architecture

In the proposed system development, it consists of components such as the front-end User Interface (UI), back-end, databases, middleware system as well as integration of external media sources that will work together to produce the SnapSecure system. This as a result would a better overview on how the system architecture would look in writing compared to its complex nature of word for word, allowing it to be illustrated in a more modular format for easier understanding of other developers.

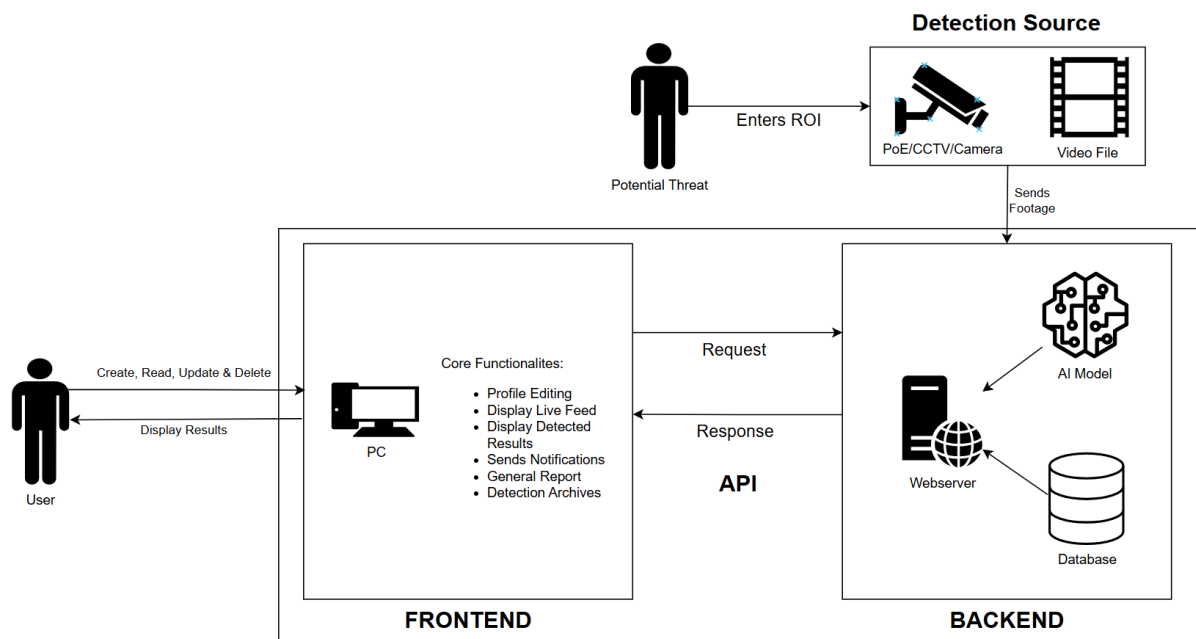


Figure 3.22 System Architecture for SnapSecure

Based on Figure 3.22, the system architecture is based on a client-server architecture, which separates the responsibilities of the frontend and backend to ensure efficiency and scalability. The frontend serves as the client-side interface, developed using HTML, CSS, and JavaScript, to provide users with an intuitive and responsive graphical user interface (GUI). Through the frontend, users can perform key actions such as managing their profiles, viewing live feeds, analysing detection results, sending notifications, accessing historical detection archives, and taking actions based on detected threats.

The backend, operating as the server-side component, is responsible for handling all business logic, including the execution of the AI model, data processing, database management,

and response generation. This separation ensures that the system operates seamlessly, with the backend managing data processing and storage while the frontend delivers a user-friendly experience.

The detection sources in this system include PoE/CCTV cameras and video files, which act as input providers. These sources capture footage from specific Regions of Interest (ROIs) and send it to the backend for processing. It is important to note that the hardware components, such as the cameras, are outside the scope of this Proposed System. The primary focus begins once the footage is transmitted to the backend. The core of the system lies in the AI model, which processes the received footage to perform object detection, focusing on identifying suspicious activities or package theft. The results generated by the AI model are then sent to the webserver, where they are either displayed in real-time on the user dashboard. This approach ensures that the system delivers accurate and timely detection results.

To facilitate seamless communication between the frontend and backend, the system incorporates an API. This API handles all interactions between the two components, enabling the frontend to send HTTP requests to the backend for user authentication, live feed retrieval, archive management, and other operations. The backend processes these requests and returns the appropriate responses to the frontend. The API plays a crucial role in ensuring secure and efficient data exchange between the system's components. The backend is primarily implemented in Python, using frameworks such as Flask or Django for API development and AI model integration. In addition, PHP may be used for specific tasks, such as server-side logic or database operations. The backend business logic controls key processes such as user authentication, triggering AI-based processing, managing detection results, and sending alerts. This communication between the frontend and backend is facilitated via the Hypertext Transfer Protocol (HTTP), ensuring a streamlined and reliable data flow.

3.4.3 Flow Chart Diagram

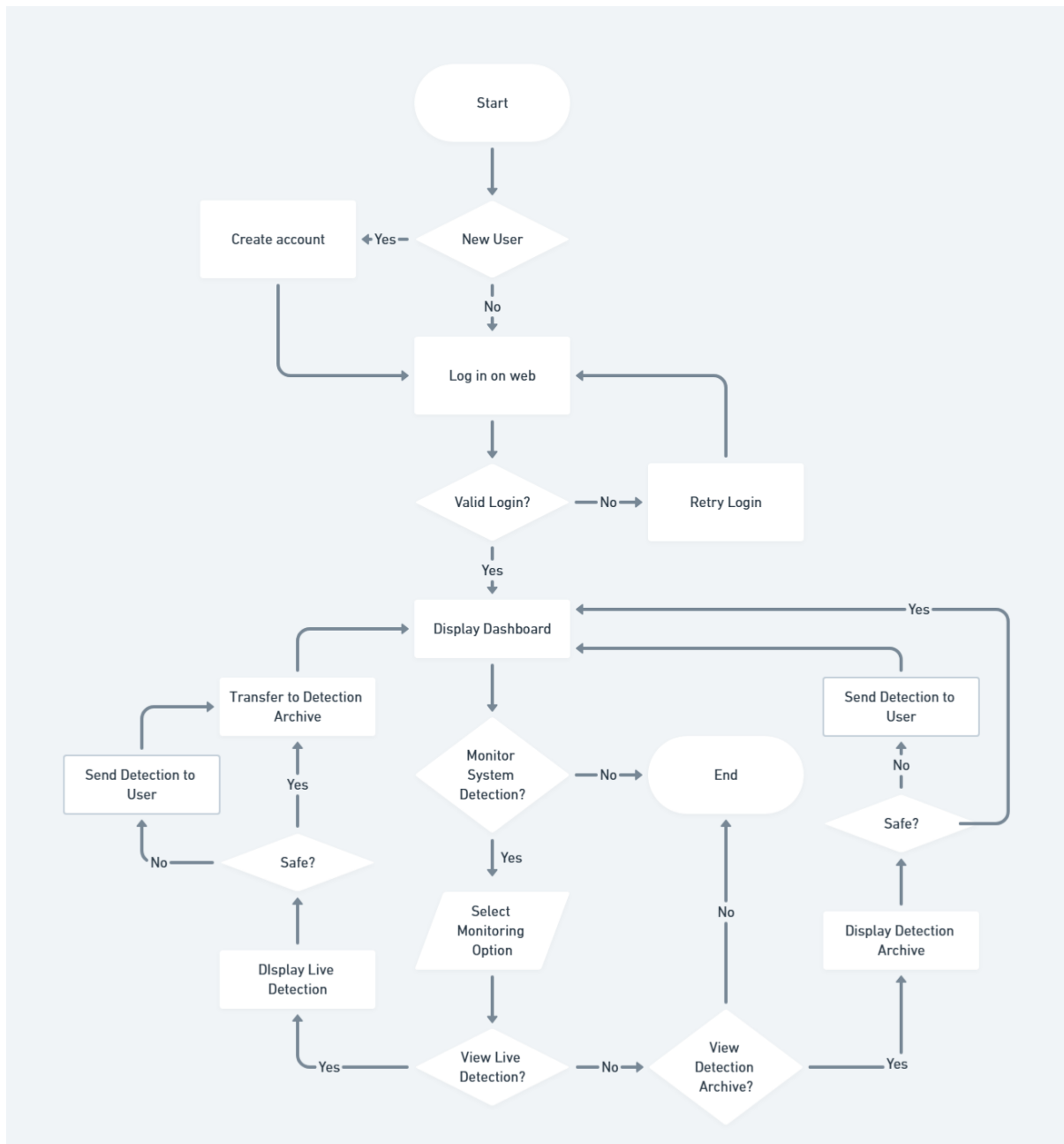


Figure 3.23 Flow Chart of SnapSecure

According to Figure 3.23, the processes in the flowchart begin with:

1. Account Creation

The User may create a new account for themselves if they were to utilize SnapSecure's service.

2. Login & Retry

The Users that already have an account with SnapSecure would be able to login to the system via web application which requires their User ID and password.

3. Display Dashboard

The Dashboard display would display the overview of the account of the user, such as the live detections, the ability to take action for the live detections, ability to view statistics, archive, or profile page.

4. Display Live Detection

The Display Live Detection would enable the user to take actions on their detections which is either to mark the live detection as safe or otherwise, this will either trigger the event notification or archive the detection.

5. Display Detection Archive

The Display of Detection Archive would generally display all the past detections records and details, such as timestamps, location, and description. It would also enable the user to change the status of the detections from safe to otherwise to trigger notifications to be sent to users.

6. Send Detection to User

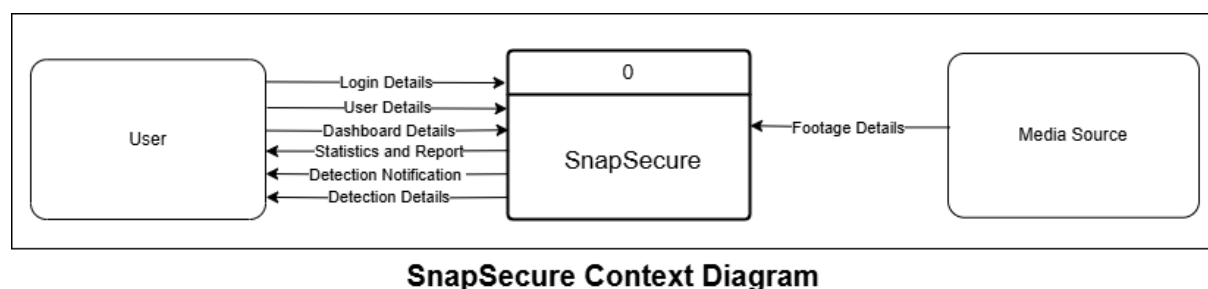
This functionality would focus on notifying the user through SMTP regarding the live detection or archive detection that is labelled as safe or otherwise.

7. Transfer to Detection Archive

In the events of transferring the detections to the archive, it would be done after either the system sends it automatically to the archive after a detection, or it is done manually by the user themselves after categorizing the detection to be safe or otherwise.

3.4.4 Context Diagram: Level 0 Data Flow Diagram (DFD)

A context diagram depicts the process flow of the system and the external entities. The main objective of this diagram is to define the boundaries of the system and clarify the functionalities of the system Figure 3.24



SnapSecure Context Diagram
Figure 3.24 Context Diagram for SnapSecure

By referring to Figure 3.24, the proposed system named SnapSecure has 2 external entities that include the User and Media Source.

Table 3.4 Data Flow between The System and External Entities

Data Flow between SnapSecure and User	Data Flow between SnapSecure and Media Source
• Login Details • User Details • Dashboard Details • Statistics and Report • Detection Notification • Detection Details	• Footage Details

3.4.5 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a representation of the flow of data within an information system, it beholds a representation that can be understood by both technical and non-technical users (GeeksforGeeks, 2024).

3.4.5.1 Level 1 Data Flow Diagram (DFD)

The Level 0 Data Flow Diagram is considered as the diagram which represents the major flow of the system. Figure 3.25 depicts a Level 1 DFD with six processes in the proposed system that includes Login Account, Create Account, View Dashboard, AI Image/Video Processing, Manage Live Detection, Manage Notification, and View Statistic and Report. There are 2 data stores such that Account Details, and Archive Details have been included in this DFD.

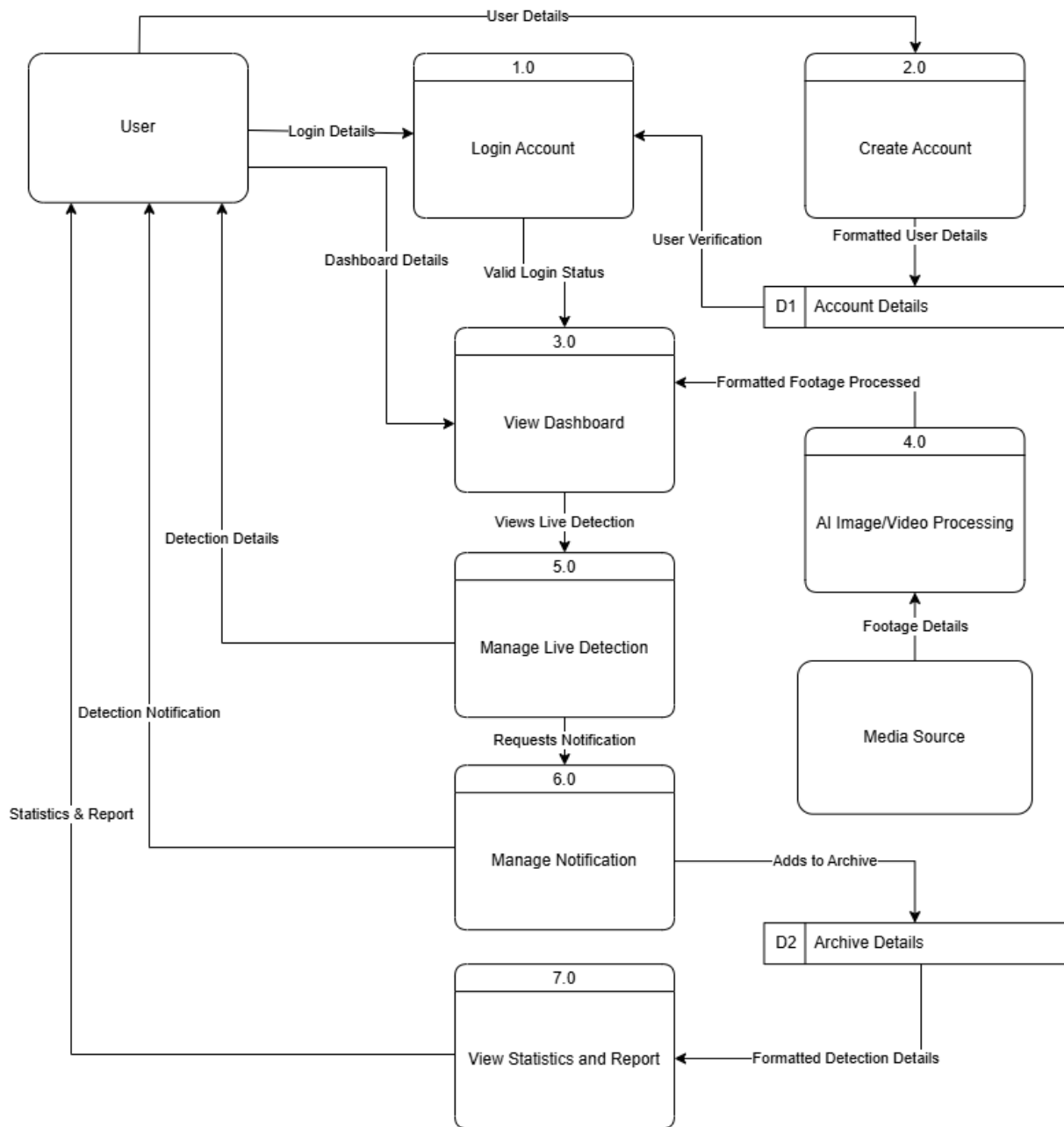


Figure 3.25 Level 1 Data Flow Diagram of SnapSecure

The processes can be further elaborated as follows:

- In Process 1.0: Login Account, the login details are entered into the system by the user, which is then processed and undergone verification to ensure the legitimacy and validity of the user to proceed with the login.
- In Process 2.0: Create Account, the user details are entered once again by the user for creating a new account, which is then formatted and store in the account details file.

- In Process 3.0: View Dashboard, it is a process whereby a request from the user requires the system to display the contents of the dashboard, and in this scenario, the dashboard here would consist of the live detection which is obtained from the formatted footage that was processed by process 4.0. Through this process, it is also capable of viewing the Live Detection
- In Process 4.0: AI Image/Video Processing, the function of this process is to obtain the footage details that have been provided by the media sources. Then it undergoes process such as object detection, motion detection, and other AI image/video processing steps. This would later be converted into a formatted footage that was processed to be sent to Process 3.0.
- In Process 5.0: Manage Live Detection, since this process comes after viewing the dashboard, it allows the user to manage the detection by either denoting it as safe or unsafe, that will then provide the user with the Detection Details.
- In Process 6.0: Manage Notification, it is a process which begins after Process 5.0 when the user requests to send the notification to their email, after denoting the live detection as unsafe. It then sends the Detection Notification to the user and adds the details to the archive.
- In Process 7.0: View Statistic and Report, the user will be able to view the detection details that the system has logged and display it in readable formats.

3.4.5.2 Level 2 DFD

The Level 2 DFD is the result of decomposition of processes in Level 2 DFD. It is also to be noted that not all processes in Level 2 DFD are to be deconstructed. Some complex processes require a more meticulous decomposition into the Level 2 DFD.

3.4.5.2.1 Level 2 DFD for Process 1.0

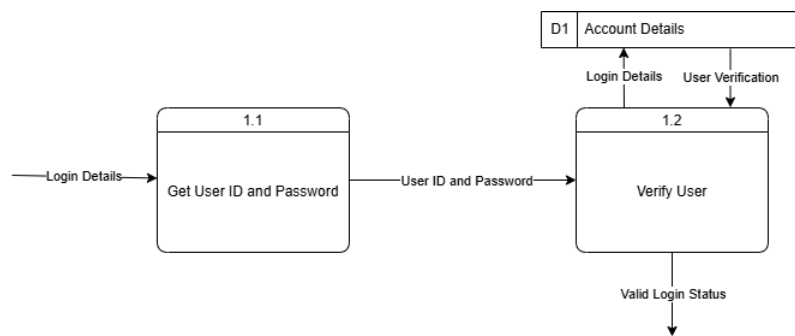


Figure 3.26 Level 2 DFD for Process 1.0 Account Login

Based on Figure 3.26, the user will be able to handle the account login whereby it would require them to enter their User ID and Password to log into the web application. Then there is a process of verifying the user that will be done to validate the data in the account details file which has been entered by the User. After that, once the status is considered valid, the user may proceed.

3.4.5.2.2 Level 2 DFD for Process 2.0

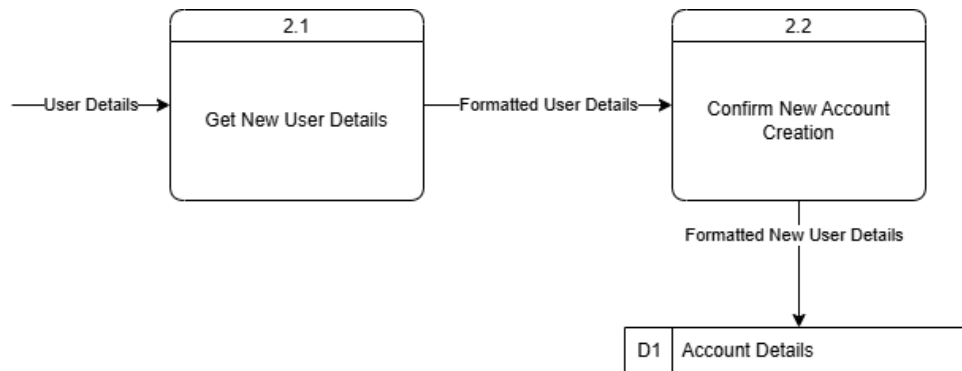


Figure 3.27 Level 2 DFD for Process 2.0 Create Account

Based on Figure 3.27, the user will be able to create new accounts if they would like to utilize the SnapSecure system. The system would require the user to create the account by entering the details of themselves. These entered details will then be formatted and saved in the account details file once the account creation has been confirmed.

3.4.5.2.3 Level 2 DFD for Process 3.0

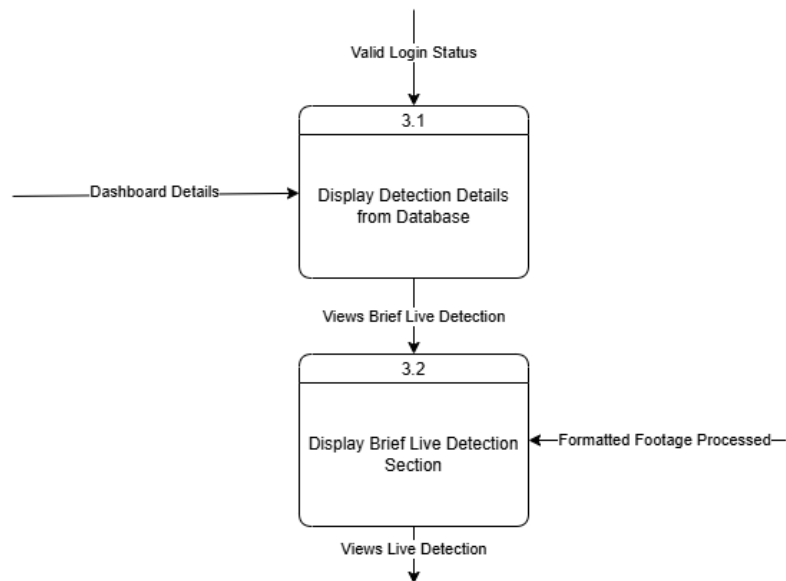


Figure 3.28 Level 2 DFD for Process 3.0 View Dashboard

According to Figure 3.28, after getting a valid login status, it would automatically display the Dashboard to the user which will contain the dashboard details which have been retrieved from the database, here the user will be able to view directly from the dashboard the brief live detection, this brief live detection section will be displayed after obtaining the result from the formatted footage processed. Now the user is capable of viewing the Live Detection in detail.

3.4.5.2.4 Level 2 DFD for Process 4.0

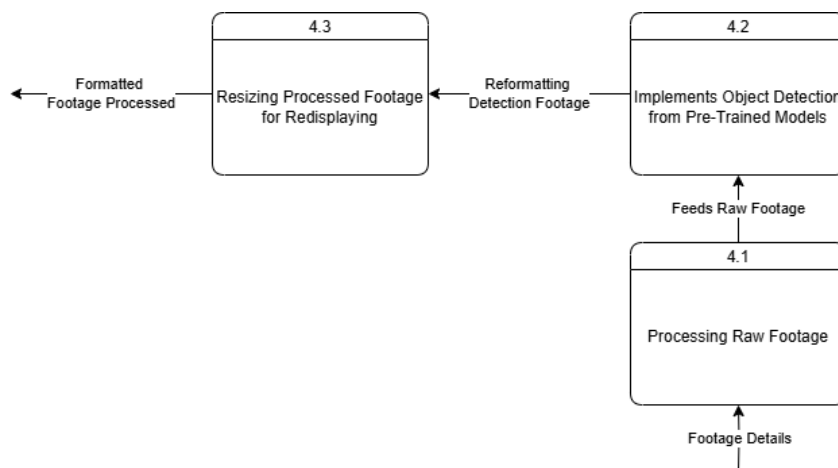


Figure 3.29 Level 2 DFD for Process 4.0 AI Image/Video Processing

According to Figure 3.29, the process here is automated by the AI model which begins with acquiring the footage details from media sources. It then undergoes processing of raw

footage that will be fed to the AI model that would implement Object Detection from the Pre-Trained Models. After that these detections are formatted and resized to be able to be sent to the webserver.

3.4.5.2.5 Level 2 DFD for Process 5.0

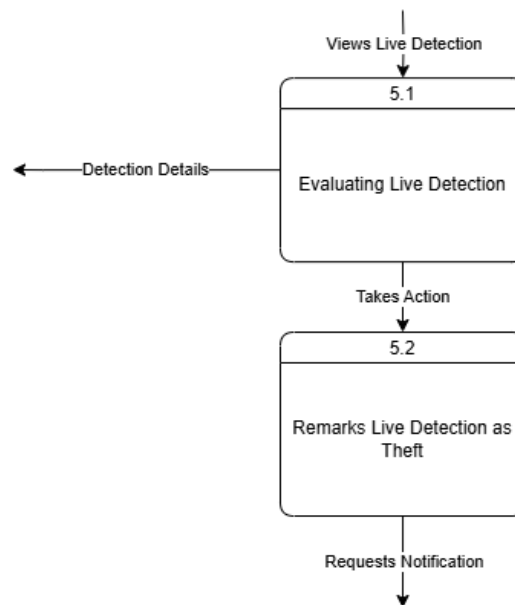


Figure 3.30 Level 2 DFD for Process 5.0 Manage Live Detections

Figure 3.30 depicts a more meticulous role of Process 5.0 where upon the users selection to view the live detection, it will allow the user to evaluate the live detection where all the detection details will be displayed for the user, after that once the user is ready to take action, which is the make a remark on the live detection. If the user remarks the live detection as a theft, there will be an automatic request notification processed by the system.

3.4.5.2.6 Level 2 DFD for Process 6.0

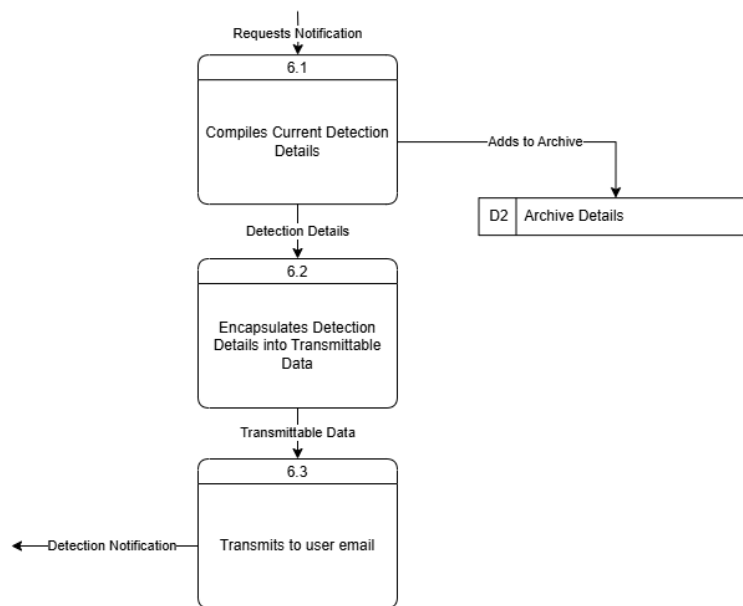


Figure 3.31 Level 2 DFD for Process 6.0 Manage Notification

According to Figure 3.31, it illustrates that the manage notification process decomposed into a more detailed illustration where it begins when there is a request notification which would lead the system to compile all current detection details that will be added to the archive details file. Then these detection details are further encapsulated into transmittable data which will then be transmitted to the user via email, and this would leave a detection notification remark for the user.

3.4.5.2.7 Level 2 DFD for Process 7.0

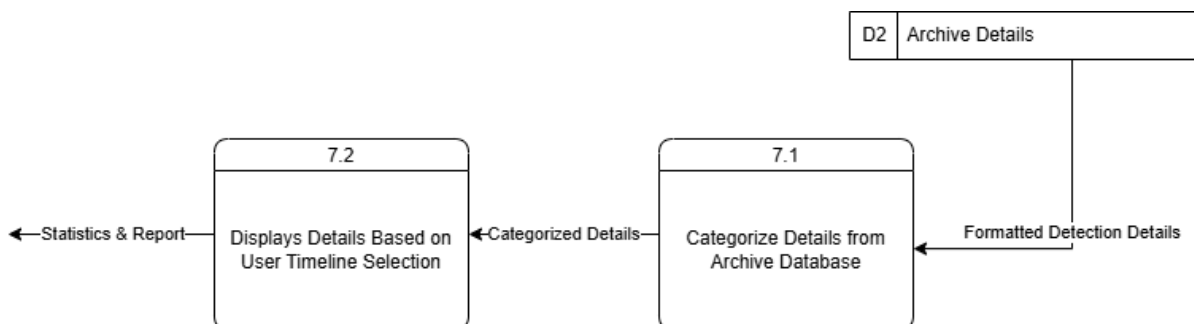


Figure 3.32 Level 2 DFD for Process 7.0 View Statistics and Report

Based on Figure 3.32, the decomposition of Process 7.0 has lead to a more detailed manner of statistic and report acquisition whereby the process begins when the user requests for viewing

the statistic and report which will then cause the system to retrieve formatted detection details from the archive details file, which will then be categorised. These categorized details are then sent to the user giving them the ability to view these details based on their preset timeline selection.

3.4.6 Entity Relationship Diagram (ERD)

An Entity Relationship Diagram in this system would serve as the database design and system analysis design which will enable the development of the system (IBM, 2024). Though noted that the system is utilizing a NoSQL database, MongoDB, the ERD would still be proven useful in designing the system as it allows depiction of data flow within the system (Figure 3.33).

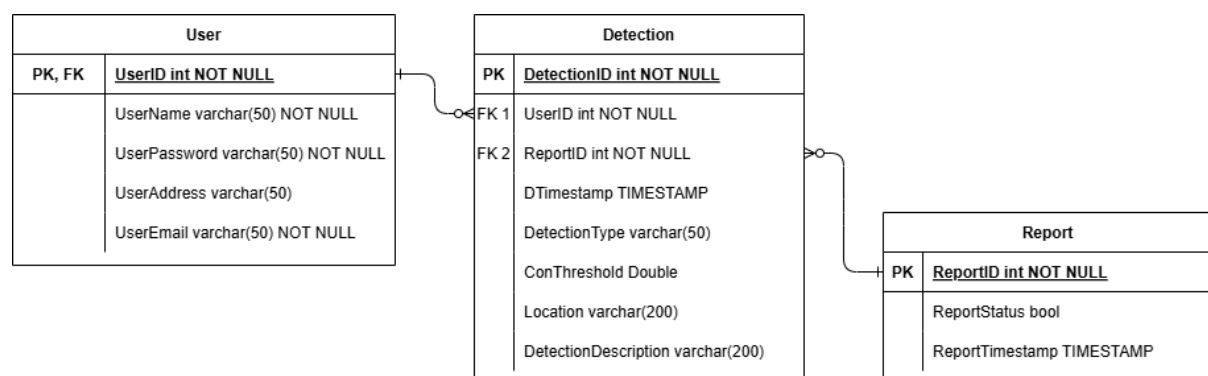


Figure 3.33 Entity Relationship Diagram

Database Design Collection:

- User Collection
- Detections Collection
- Report Collection

With close reference to Figure 3.33, the Crow's Foot Notation relationship between the User table and the Detection table would be a one-to-zero or many, depicting that there could be a scenario where one user has no detections, one user has one detection, or one user has many detections. Next the relationship between the Detection table and the Report table would

be similar as there would be a one-to-zero or many relationship, that emphasizes that one report may not have any detections, one report has only one detection, or one report has many detections.

3.4.7 Data Dictionary

Since the approach of the system development is by utilizing NoSQL database, MongoDB, the Data Dictionaries will be converted to MongoDB adaptable formats.

3.4.7.1 Data Dictionary for User Collection

Table 3.5 illustrates the data objects inside the User Collection. The field name, data type, description and constraints.

Table 3.5 Data Dictionary for User Collection

Field Name	Data Type	Constraints
UserID	ObjectId	Primary Key, Auto generated
UserName	String	Required, Max length: 50
UserPassword	String	Required, Max length: 50
UserAddress	String	Optional, Max length: 50
UserEmail	String	Required, Unique, Max length: 50

There are 5 fields in Table 3.5:

- UserID: A unique MongoDB ObjectId that automatically identifies each user record
- UserName: The user's name (required, limited to 50 characters)
- UserPassword: An encrypted version of the user's password (required, limited to 50 characters)
- UserAddress: The user's physical address (optional, limited to 50 characters)
- UserEmail: The user's email address (required, must be unique, limited to 50 characters)

3.4.7.2 Data Dictionary for Detection Collection

Table 3.6 depicts the Data Dictionary for Detection Collection where 3 columns comprising of Field Name, Data Type and Constraints.

Table 3.6 Data Dictionary for Detection Collection

Field Name	Data Type	Constraints
DetectionID	ObjectId	Primary Key, Auto-generated
UserID	ObjectId	Foreign Key: User.UserID
ReportID	ObjectId	Foreign Key: Report.ReportID
DTimestamp	Date	Required
DetectionType	String	Required, Max length: 50
ConThreshold	Double	Optional, Range: 0.0 to 1.0
Location	String	Optional, Max length: 200
DetectionDescription	String	Optional, Max length: 200

There are 8 fields in Table 3.6:

- DetectionID: A unique MongoDB ObjectId for each detection event
- UserID: Links to the User collection (foreign key) to track who created/owns the detection
- ReportID: Links to the Report collection (foreign key) if this detection is associated with a report
- DTimestamp: The date and time when the detection occurred
- DetectionType: Categorizes the type of detection (e.g., "Package" or "Person")
- ConThreshold: A confidence score between 0.0 and 1.0 indicating detection certainty
- Location: Where the detection took place (optional, limited to 200 characters)

- **DetectionDescription:** Additional details about the detection (optional, limited to 200 characters)

3.4.7.3 Data Dictionary for Report Collection

Table 3.7 depicts the Data Dictionary for Detection Collection where 3 columns comprising of Field Name, Data Type and Constraints.

Table 3.7 Data Dictionary for Report Collection

Field Name	Data Type	Constraints
ReportID	ObjectId	Primary Key, Auto-generated
ReportStatus	Boolean	Required
ReportTimestamp	Date	Required

There are 3 fields in Table 3.7:

- **ReportID:** A unique MongoDB ObjectId for each report
- **ReportStatus:** A boolean value indicating if the report is resolved (true) or not (false)
- **ReportTimestamp:** The date and time when the report was created

3.4.8 High-fidelity Wireframe Design

The wireframe design is a draft of a graphical outline within a proposed application (Roth et al., 2018). It will be utilized as a prototype to collect feedback and intel from respondents before the final design is determined. Hence as part of the system design, a high-fidelity wireframe design would be utilized to emphasize the web application of the proposed system. It should also be noted that this design is not final and the colour scheme is not final as well.

3.4.8.1 Web Application Wireframe for SnapSecure Landing Page

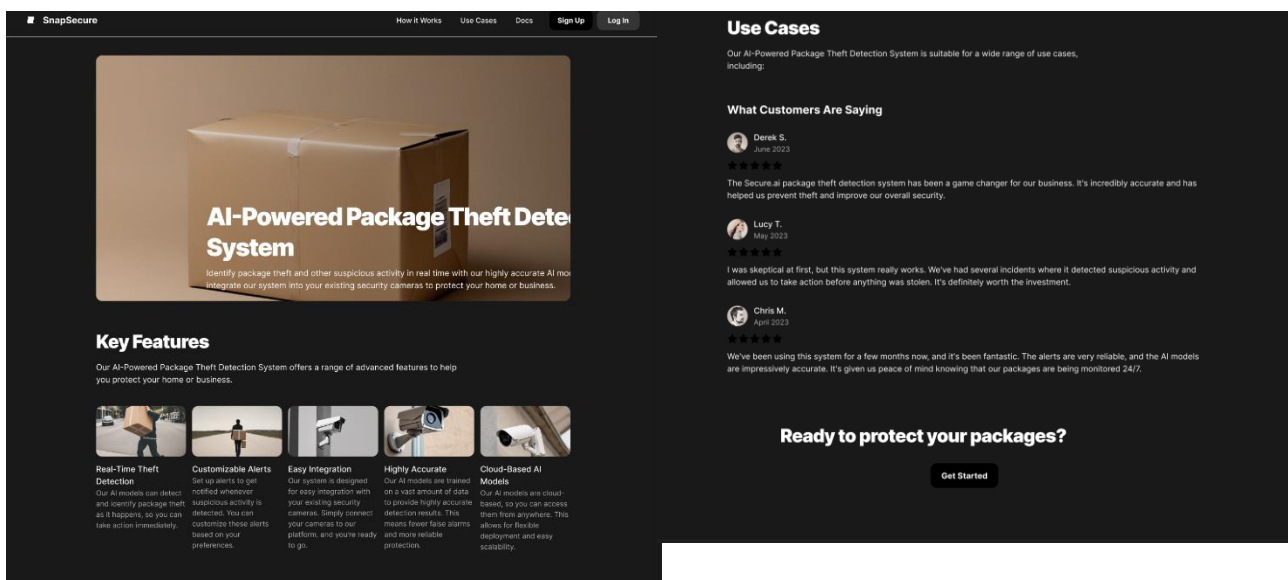
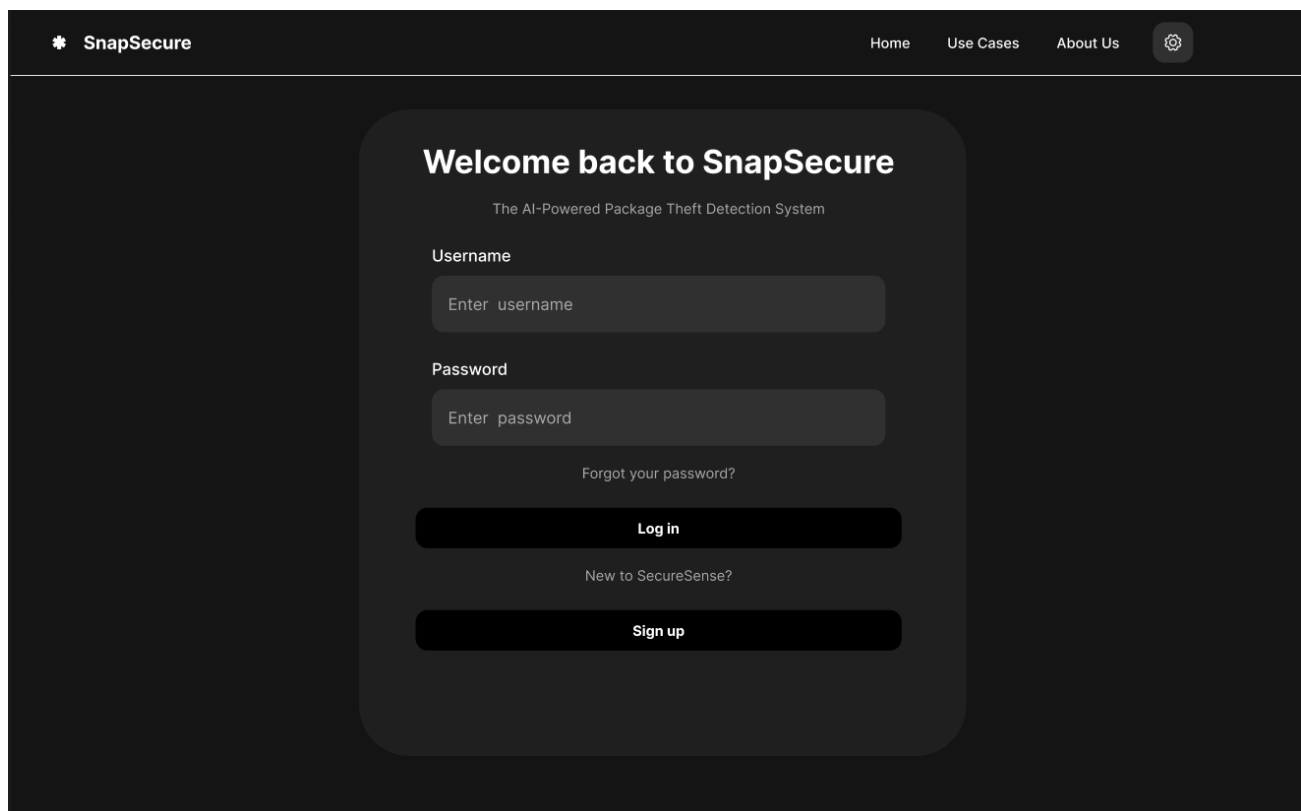


Figure 3.34 Wireframe for Proposed System Landing Page

Based on Figure 3.34 Wireframe for Proposed System Landing Page when the user first comes across SnapSecure, they will be greeted with this landing page which will describe to them about how does the system work, which will also provide use cases where the system can be utilized, also there will be a potential testimonial section that would be able to further concrete the pull of the users to utilize this system. Functionality wise, there will be a signup and signin button on the top right of the screen which will allow the user to create or login the system.

3.4.8.2 Web Application Wireframe for SnapSecure Login Page



The wireframe shows a dark-themed login page for SnapSecure. At the top, a navigation bar contains the SnapSecure logo, links for Home, Use Cases, and About Us, and a settings icon. The main content area features a central login card with the heading 'Welcome back to SnapSecure' and the subtitle 'The AI-Powered Package Theft Detection System'. Below this, there are input fields for 'Username' and 'Password', each with a placeholder text 'Enter username' and 'Enter password' respectively. A link for 'Forgot your password?' is positioned below the password field. Two prominent buttons, 'Log in' and 'Sign up', are located at the bottom of the card, with the text 'New to SecureSense?' placed between them.

Figure 3.35 Wireframe for SnapSecure Login Page

According to Figure 3.35, the user will be presented with the login page when the login button is clicked. In this page the user will be able to enter their username, password proceed to login, there is also a possibility the user is new and would like to signup hence having the signup button there ready if the user would like to create an account.

3.4.8.3 Web Application Wireframe for SnapSecure Create Account Page

The wireframe shows a 'Create an account' form with the following elements:

- Header:** SnapSecure logo, Home, Use Cases, About Us, settings gear, and user profile icon.
- Form Title:** Create an account
- Input Fields:**
 - First name (placeholder: First name)
 - Last name (placeholder: Last name)
 - Email (placeholder: Email)
 - Password (placeholder: Password)
 - Re-enter password (placeholder: Re-enter password)
- Buttons:** Create account
- Links:** Already have an account?, Log in

Figure 3.36 Create Account Wireframe for SnapSecure System

Based on Figure 3.36, assuming the user is new to the system, they will be able to create a new account by entering their first name, last name, email address, password. As a way to ensure that the intended password entered by the user is correct, the user will have to refill the password once again to ensure that is the intended password. In addition, if the user already has an account, they may click on the hyperlink below the create account button to lead them to the login page.

3.4.8.4 Web Application Wireframe for SnapSecure Dashboard

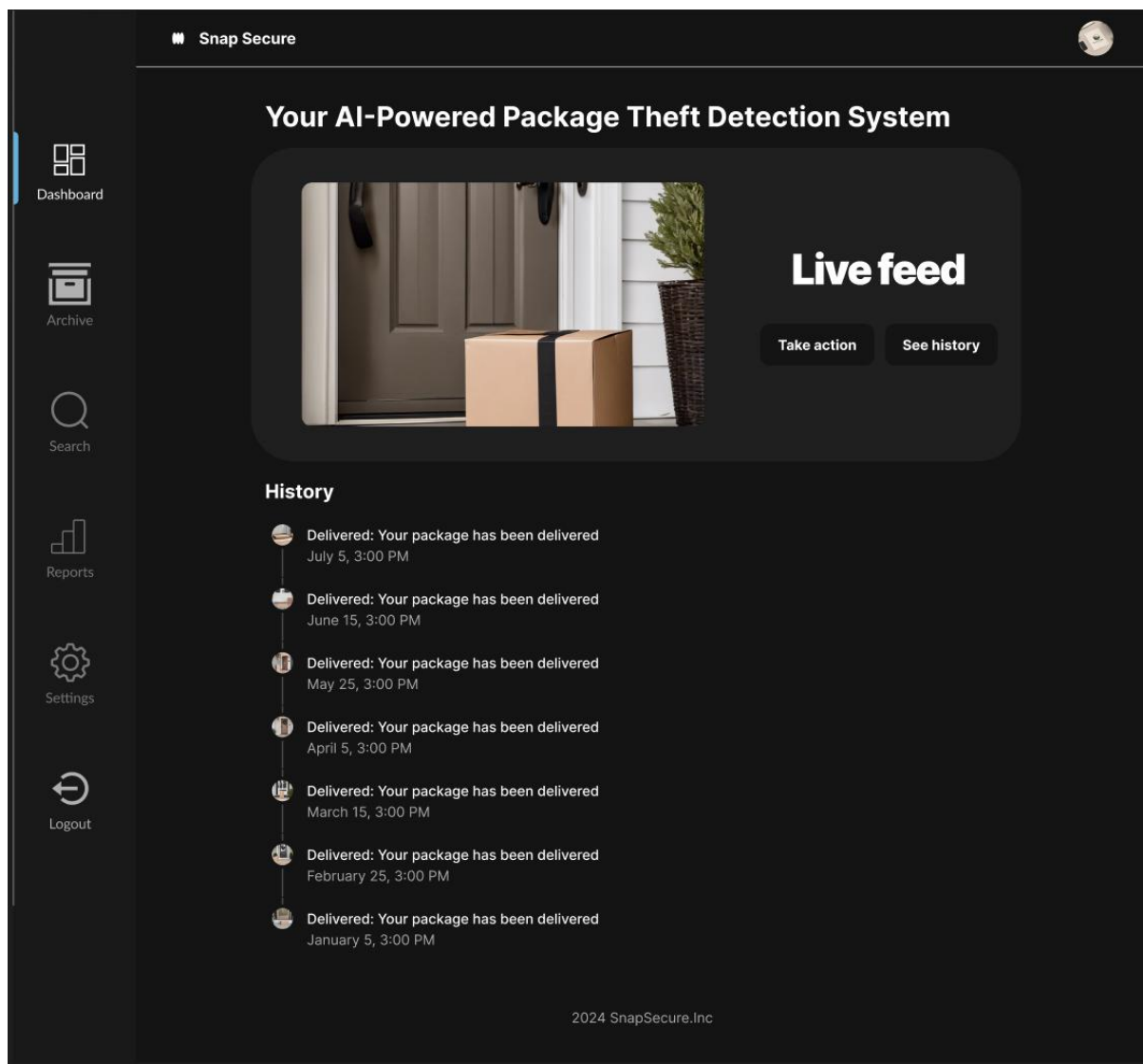


Figure 3.37 Wireframe for SnapSecure Dashboard Page

Based on Figure 3.37 it can be clearly illustrated that the user dashboard page would display the live detection feed, that will allow the user to directly see what is occurring live the instance they login to the system, there is also 2 buttons right next to the detection which is the “Take Action” button and the “See History” button where respectively one would allow the user to take actions such as leaving a remark on the live detection as safe or theft has occurred, while the other button would lead the user to the Archive of detections. On the right, the user will be able to select directly with icons and subtitles for each icon making it a better navigation depiction for the user to navigate around the system.

3.4.8.5 Web Application Wireframe for SnapSecure Records Page

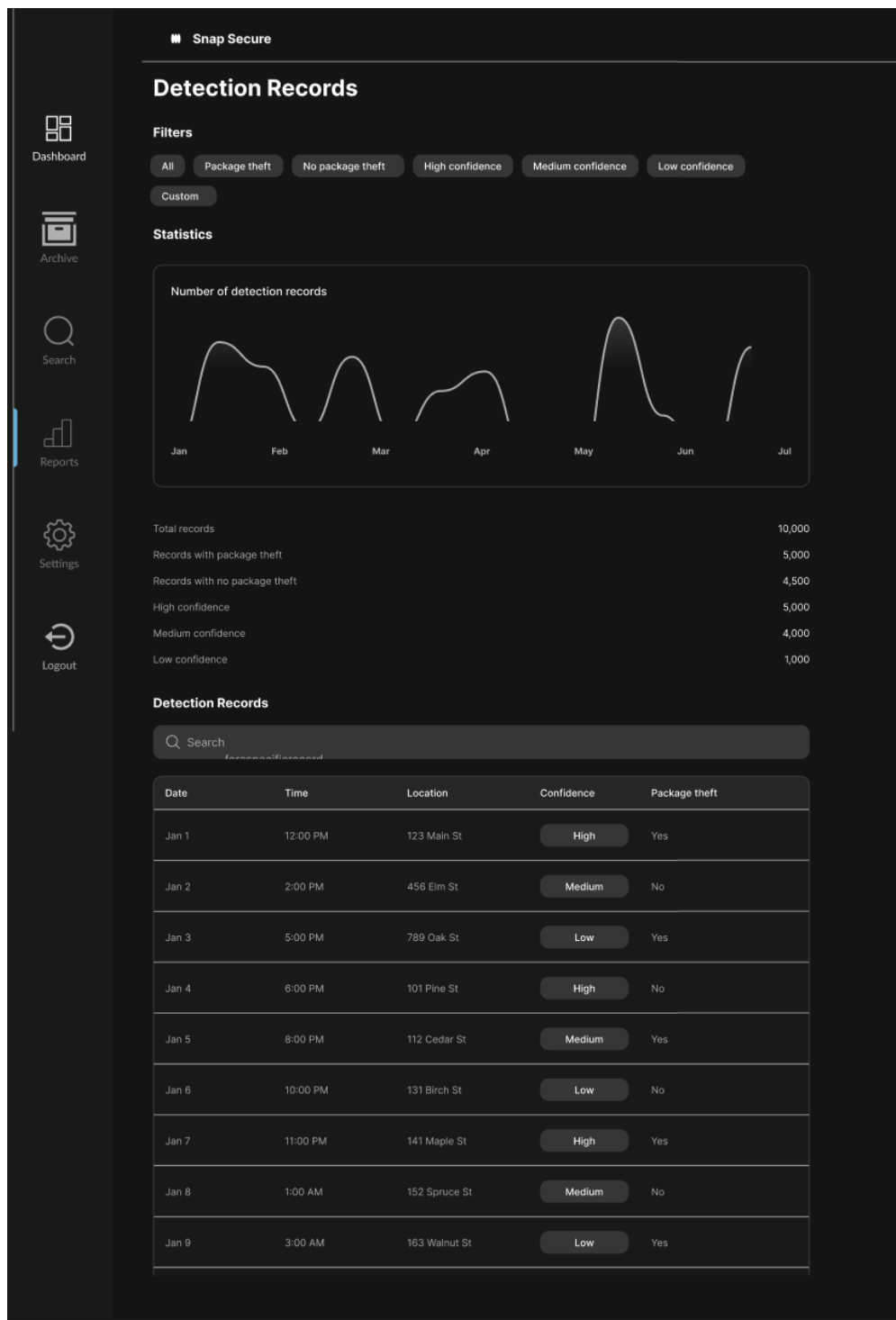


Figure 3.38 Wireframe for SnapSecure Record Page

Based on Figure 3.38, the Detection Records Page presents a detailed log of all events detected by the system, providing users with valuable insights through both textual and visual data. A statistics chart at the top displays trends over time, offering a clear view of activity patterns. Below the chart, a table lists detection records with details such as date, time, location,

detection confidence, and package theft status, making it easy to scan for specific information. Filters for detection confidence (e.g., high, medium, low) and event type help users quickly refine the displayed records. Additionally, a search bar facilitates efficient access to specific data points. By combining graphical and tabular formats, this screen caters to both casual users and those who need in-depth data analysis.

3.4.8.6 Web Application Wireframe for SnapSecure Archive Page

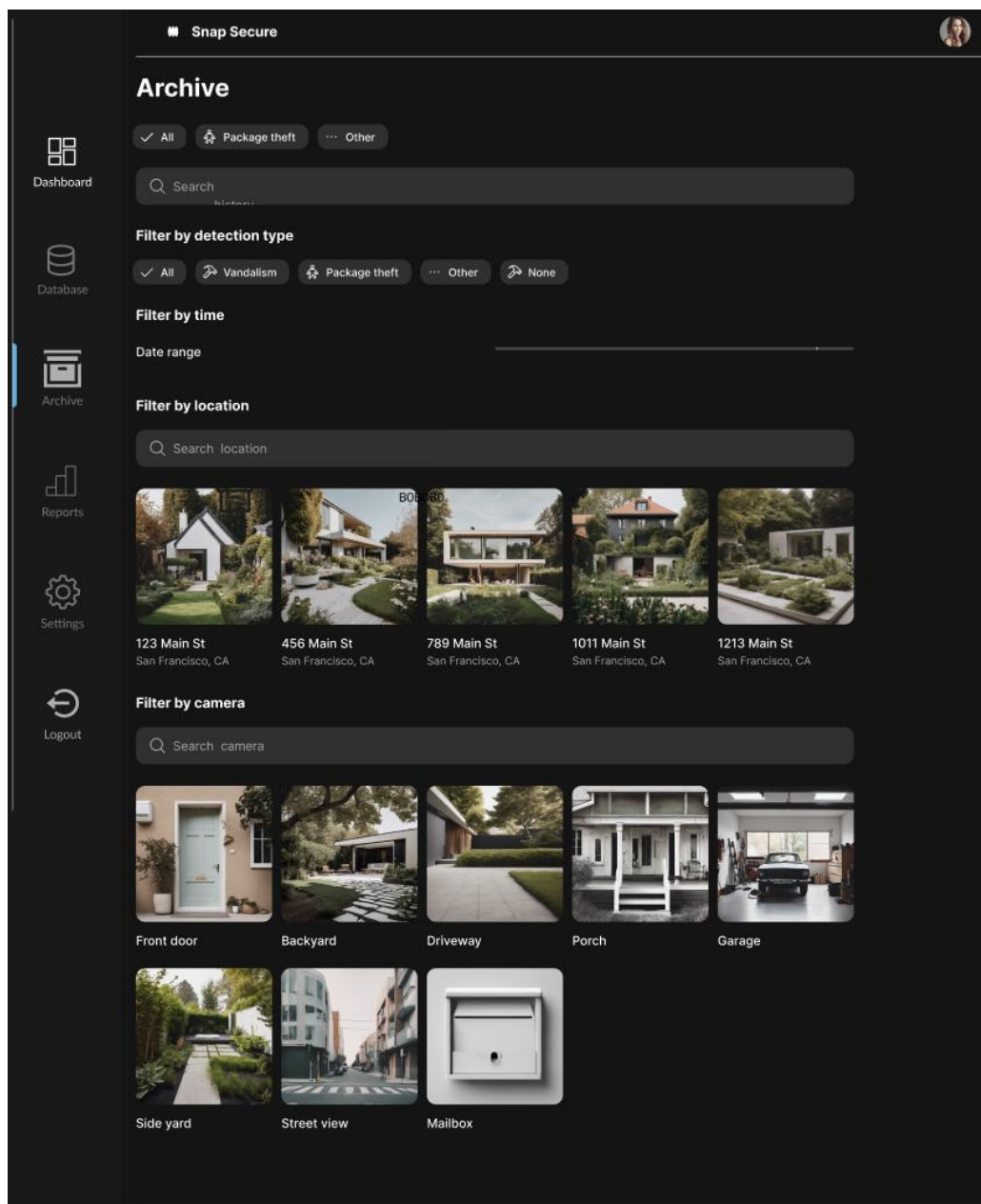


Figure 3.39 Wireframe for SnapSecure Archive Page

Figure 3.39 depicts that the Archive Screen is designed to provide users with a comprehensive overview of all past events detected by the system. It features a user-friendly interface with filter options for detection type (e.g, package theft, motion detected, etc.), date range, location, and camera, enabling users to efficiently sort through archived events. The inclusion of location thumbnails, such as "123 Main St" or "porch," offers a visual preview of the areas monitored, making it easy to recognize specific locations briefly. A search bar further

enhances navigation by allowing users to quickly locate specific events or cameras. Overall, the Archive Screen’s well-organized layout ensures that users can manage and review past incidents with minimal effort.

3.4.8.7 Web Application Wireframe for SnapSecure Action Page

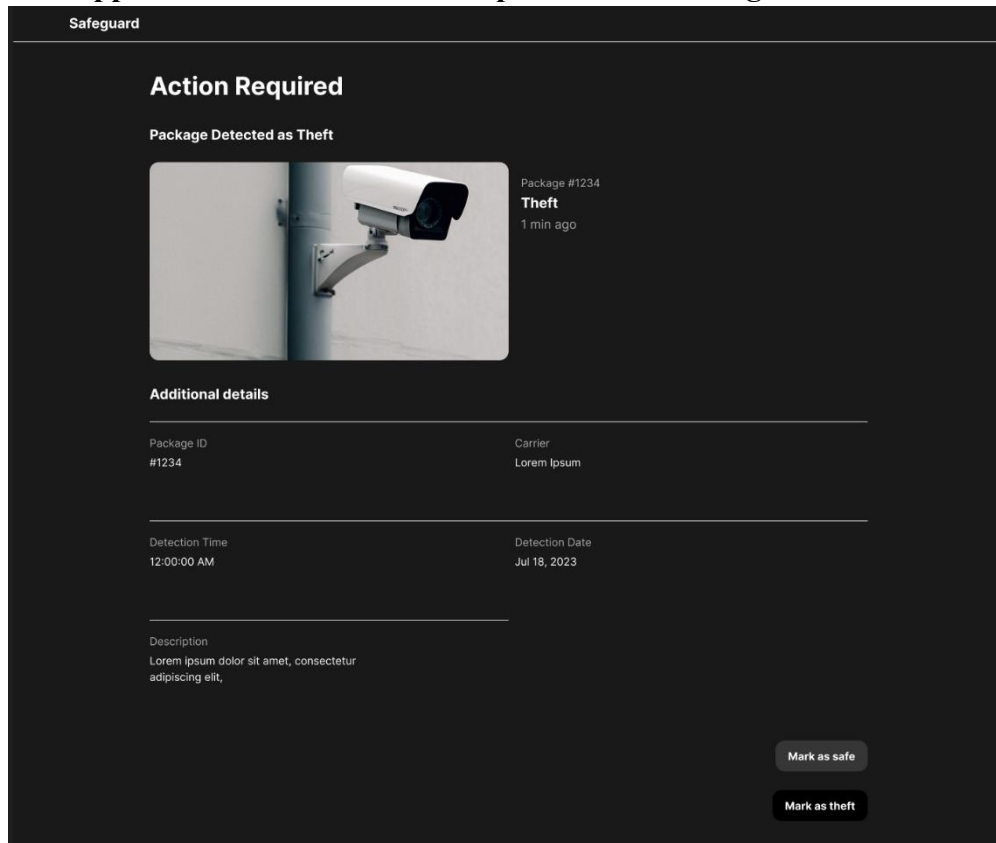


Figure 3.40 Wireframe for SnapSecure Action Page

According to Figure 3.40, the Action Page is designed to provide real-time alerts for critical events, such as package theft. The interface prominently displays the event summary, including essential details like the package’s detection status, associated camera footage, package ID, carrier, tracking number, and estimated delivery date. This ensures users have all necessary context immediately. Two clear action buttons, “Mark as safe” and “Mark as theft” which allows users to quickly respond to incidents, promoting timely decision-making. The screen’s layout focuses on simplicity and urgency, ensuring users can act immediately without distractions. This feature is particularly valuable for addressing potential thefts promptly and effectively.

3.4.8.8 Web Application Wireframe for SnapSecure Profile Page

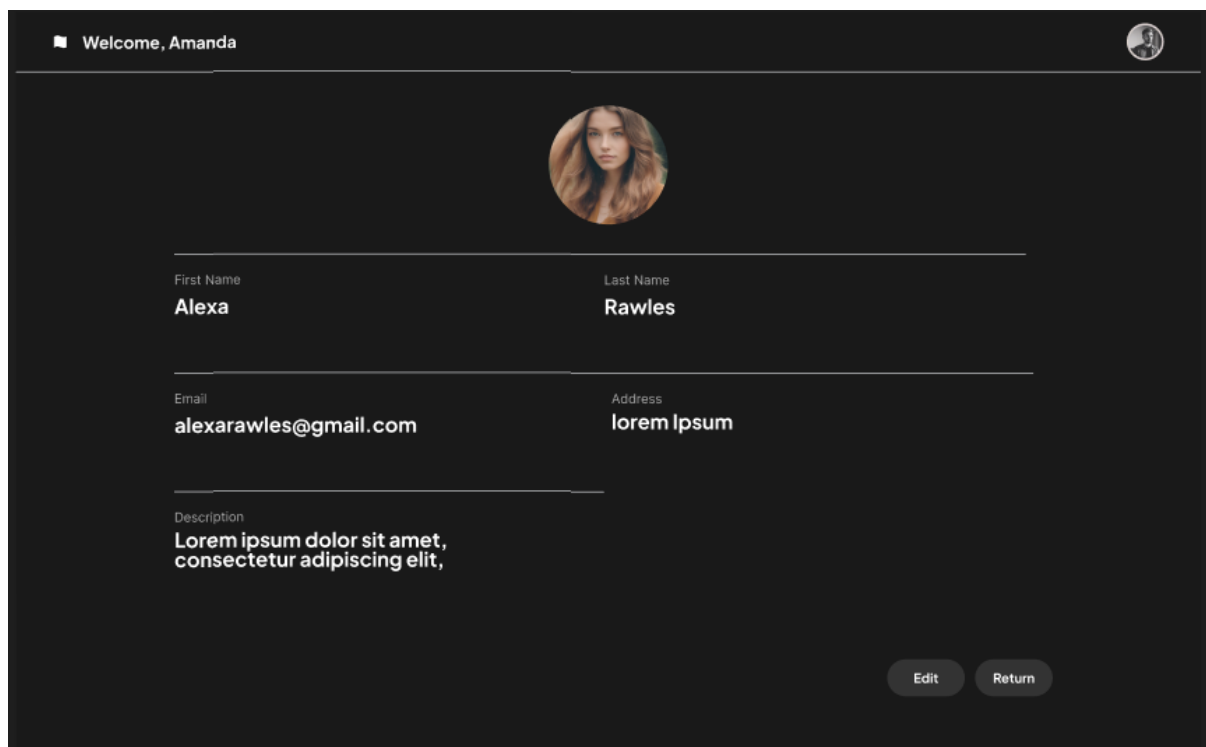
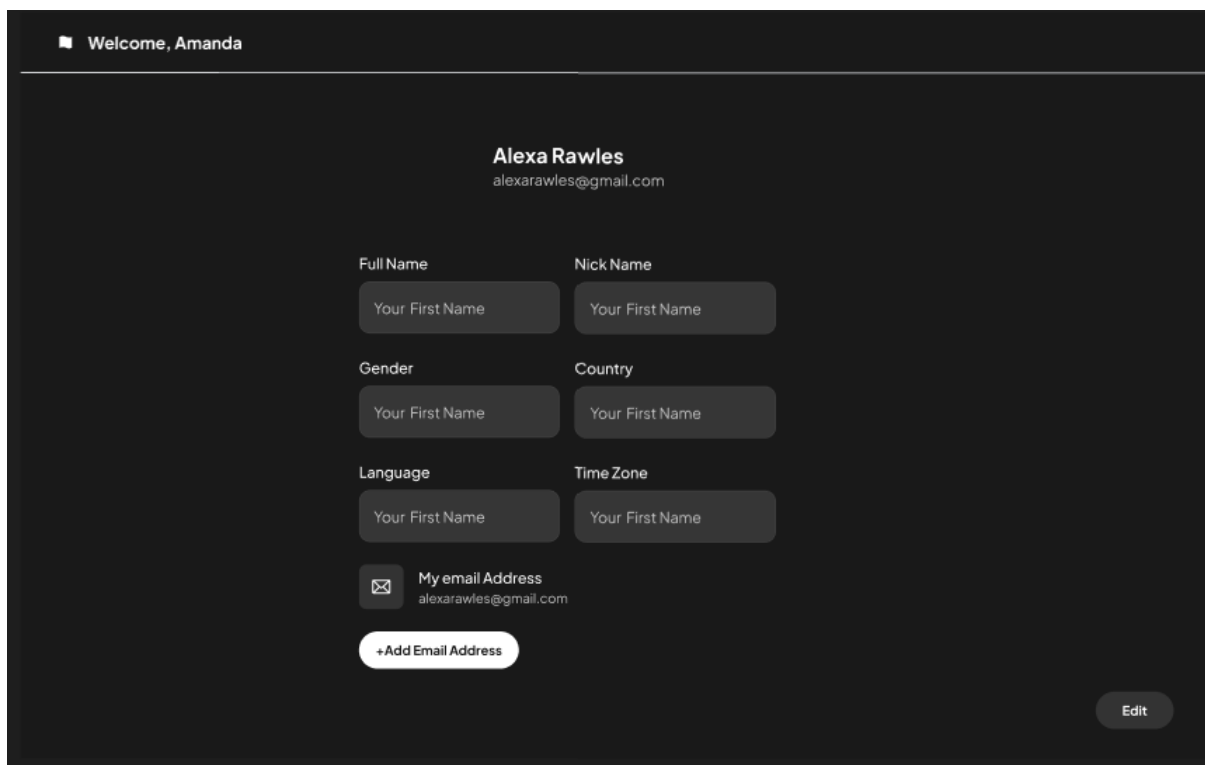


Figure 3.41 Wireframe for SnapSecure Profile Page

Figure 3.41 depicts the Profile Page which provides users with an overview of their personal information in a clean and organized layout. It displays key details such as the user's first name, last name, email address, and physical address, alongside a brief description. The interface also includes options to edit the profile or return to the previous screen, ensuring easy navigation. By featuring the user's profile picture and a personalized display of their name, the screen adds a personal touch, enhancing engagement. The simple and minimalistic design ensures that users can access and manage their profile details effortlessly, making it an intuitive component of the system.

3.4.8.9 Web Application Wireframe for SnapSecure Edit Profile Page



The wireframe shows a dark-themed user interface for editing a profile. At the top, a header bar contains the text "Welcome, Amanda". Below this, the user's name "Alexa Rawles" and email "alexarawles@gmail.com" are displayed. The main content area contains several input fields for personal information, arranged in two columns. The fields are labeled "Full Name", "Nick Name", "Gender", "Country", "Language", and "Time Zone". Each label is followed by a rounded rectangular input field containing the placeholder text "Your First Name". Below these fields, there is a section for email addresses, starting with an envelope icon, the label "My email Address", and the email "alexarawles@gmail.com". Below this is a button labeled "+Add Email Address". In the bottom right corner, there is a button labeled "Edit".

Figure 3.42 Wireframe for SnapSecure Edit Profile Page

According to Figure 3.42, the Edit Profile Page allows users to customize and update their personal information with ease. It provides editable fields for key details, including name, nickname, gender, country, language, time zone, and email address. An option to add an additional email address is also available, catering to users with multiple accounts. The "Save Changes" button ensures users can confirm their updates efficiently, while the layout mirrors the main profile screen to maintain consistency and familiarity. By focusing on usability and simplicity, this screen empowers users to manage their profiles without confusion or unnecessary complexity.

3.5 Development

Regards to Scrum Development, which is a microprocess of discovery as well as implementation and invention (Hema et al., 2020). This is then considered encompassed within both the requirement and the design phase of the system. Therefore, in order to implement the

proposed system, User Interface (UI) that were designed using software such as Figma will then be programmed by using Visual Studio Code. Then, once the front-end development is done the next proceeding is to focus on the functionalities of the back end of the system. This is where the database on MongoDB which is a non-relational database. The entire codebase will be scrutinized and looked through thoroughly and refactored at the end of the development phase. This will be followed by the idealisation of each function to ensure it is properly utilized.

3.6 Testing

By employing the Scrum framework to organize and execute the testing phase. The testing process is structured into four quadrants, which provide a comprehensive approach to ensuring the quality and effectiveness of my system (Verwijns & Russo, 2022).

1. **First Quadrant - Code Quality Testing:** Initially, the goal is to test the quality of the code. This involves identifying any errors such as syntax errors, runtime issues, or interface bugs before the system is deployed in a real-world environment. This is a crucial step to ensure that the foundational aspects of the system are robust.
2. **Second Quadrant - Scenario and User Testing:** Once the initial testing is complete, focus will be on testing various scenarios while keeping the project's objectives in mind. This will include evaluating prototypes and wireframes to enhance the user experience, ensuring the system is user-friendly and meets the desired goals.
3. **Third Quadrant - Sprint-Based Testing:** In the third quadrant, testing will be conducted in an automated manner during each sprint. As the system is still incomplete at this stage, feedback from the testing will be used to improve the system incrementally, aligning it with project requirements.

4. **Fourth Quadrant - Full System and Security Testing:** Lastly, the performance of the entire system will be tested, with a focus on critical elements like data migration, user interface, and authentication. Security is also a high priority, so tests for vulnerabilities such as hacking and data breaches to ensure that features like anti-attack mechanisms are functioning properly and user data is secure.

3.7 Deployment

The deployment phase involves transitioning the proposed system to a stable, production-ready build by integrating usage of Docker for containerization. The web application of the proposed system will be deployed to the server, making it accessible for users. This phase also includes ensuring that the system can interact with real-time data sources, replacing any dummy data previously used during development. This allows the system to perform its intended tasks, such as detecting package theft and issuing alerts, with actual input.

Additionally, data backups will be implemented to ensure the security and integrity of the system's data. Documentation, both technical and user-focused, will also be delivered during this phase. This ensures that users and developers have the necessary resources to understand and operate the system efficiently.

3.8 Review

The sprint review is the final stage in Scrum, where the completed system is assessed. The purpose of this phase is to evaluate the system's functionality, performance, and overall effectiveness. The strengths and weaknesses of the proposed system will be analyzed by reviewing the entire workflow, from object detection to alert generation. The review will also focus on the project's timeline, budget, and capabilities for the next expected release. This ensures that the system is on track to meet its goals and that any challenges encountered during

development are addressed in future sprints, leading to continuous improvements. In these phases, the deployment and review processes help ensure that the proposed system is robust, user-friendly, and optimized for real-world use, with ongoing updates based on feedback and testing.

3.9 Summary

In conclusion, the Scrum framework has been identified as the most suitable methodology for the development of the AI-Powered Package Theft Detection System due to its adaptability and focus on continuous improvement. The iterative nature of Scrum will allow for frequent assessments and refinements, ensuring that the system meets the desired functionality and requirements. By employing Scrum, the project is expected to achieve the goal of developing an efficient, user-friendly, and scalable system capable of detecting package theft in real time. Additionally, the user requirements were carefully gathered through the distribution of the questionnaires to the selected group of target users. The use of Scrum for the proposed system is vital as it is flexible in the development process, ensuring the successful delivery of the system that meets both technical and user expectations.

CHAPTER 4: IMPLEMENTATION

4.1 Introduction

This chapter would be segmented into focusing on the specifics of the proposed systems implementation which has been closely abiding the requirement analysis and design which had been iterated in Chapter 3, that has been practically mended utilizing the hardware and software from the other previous chapters. This implementation would be regarded as a prototype which is being developed constantly in the cycle, which would require iterative steps that revolve around developing, refining and demonstrating. These crucial steps are repeated continuously as the Scrum methodology had suggested until it has achieved a functioning prototype that would be able to satisfy the requirements and specifications as well as providing room for further upgrade if required. This would be considered as the continuous evolution of the proposed system to be prepared for delivery. As we begin the chapter, it would highlight the front-end and back-end development processes involved in the proposed system.

4.2 Feature Implementation

In order to develop this system, specified software, packages, extensions and applications were required to be installed prior to development of the proposed system. These installations and configurations have been crucial as it is the prerequisite to achieve the proposed system's specifications. Hence, close configurations such as dependencies and package compatibilities had become an essential component to ensure that the system would be able to respond as expected.

For front-end development, the proposed system is presented on a web browser, as it is a web application designed for user interaction. Uniquely, it is developed solely using pure web-based technologies, HTML, CSS, and JavaScript without relying on any external frameworks. This approach ensures a lightweight and highly optimized user interface. Meanwhile, the system's back-end integrates an object detection model API hosted on Roboflow Universe,

specialized for package detection. The model incorporates algorithms made within the system such as the Kalman Filter in conjunction with the Simple Online and Realtime Tracking (SORT) algorithm for robust and continuous tracking. To enhance object association accuracy across frames, the Hungarian Algorithm is utilized, enabling advanced computer vision capabilities for live monitoring and precise package tracking. Additionally, MongoDB is utilized for database management, offering a flexible and scalable NoSQL solution suited for handling the system's real-time data. Python Flask is employed to facilitate server-side scripting and API endpoint integration, ensuring seamless communication between the front-end interface, object detection engine, and the database layer. Finally, to ensure robustness and mobility, Docker containerization is utilized as it ensures consistent deployment across environments, with a multi-container architecture defined through Docker Compose.

4.2.1 Installation and Setup of Visual Studio (VS) Code

Visual Studio Code can be downloaded from the official website at the URL

“<https://code.visualstudio.com/>”

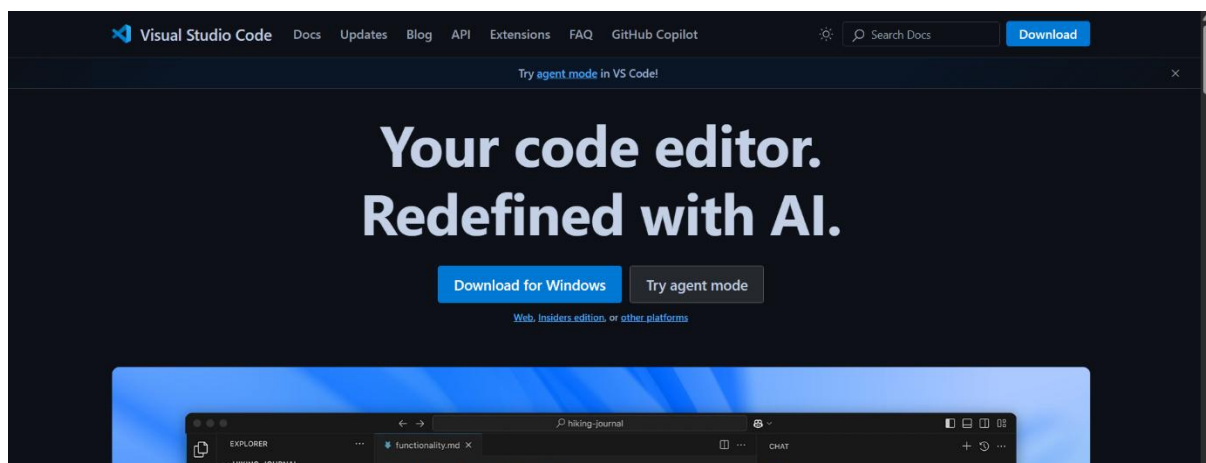


Figure 4.1 Visual Studio Code Landing Page

Source: <https://code.visualstudio.com/>

By default, the Visual Studio Code version available for download is tailored for Windows 64-bit operating systems. However, alternative versions are available for macOS, Linux, and ARM architectures to accommodate different user preferences and system requirements. For the proposed system, the standard Windows version of Visual Studio Code is utilized, as shown in Figure 4.1.

Upon visiting the official download page at code.visualstudio.com, users are presented with the primary option to download Visual Studio Code for Windows. Once the **Download for Windows** button is selected, the setup executable file will begin downloading automatically.

License and Agreement

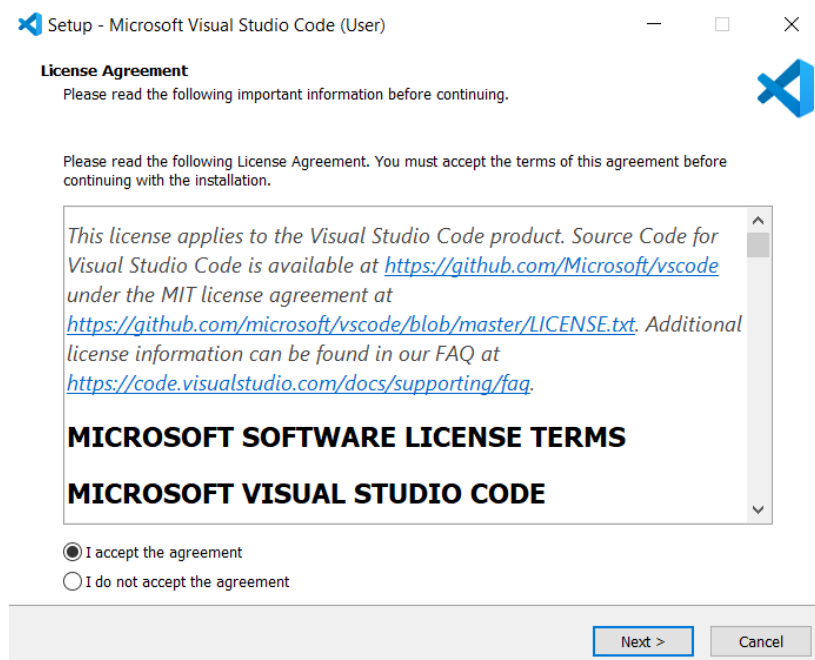


Figure 4.2 VS code License Agreement

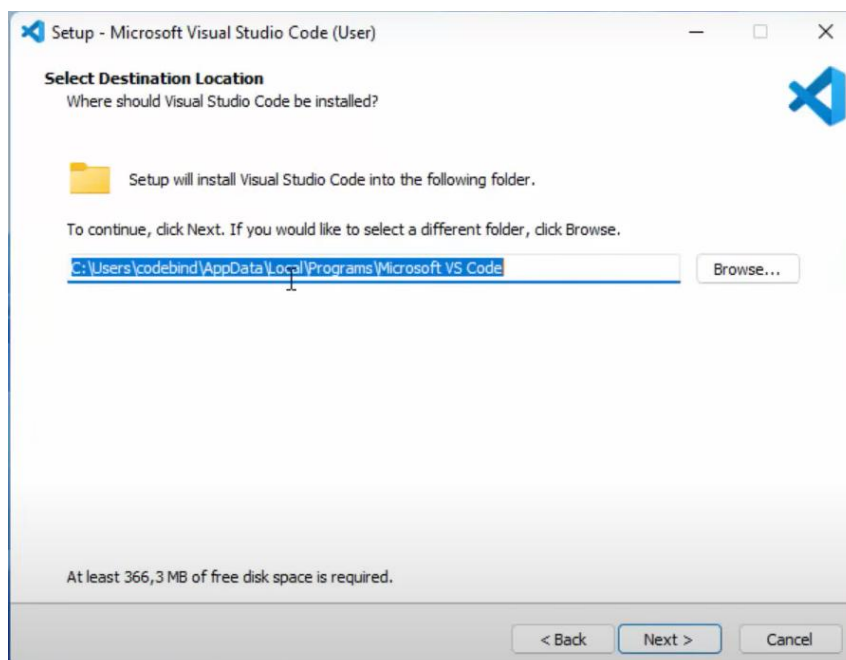


Figure 4.3 Directory Selection for VS code installation

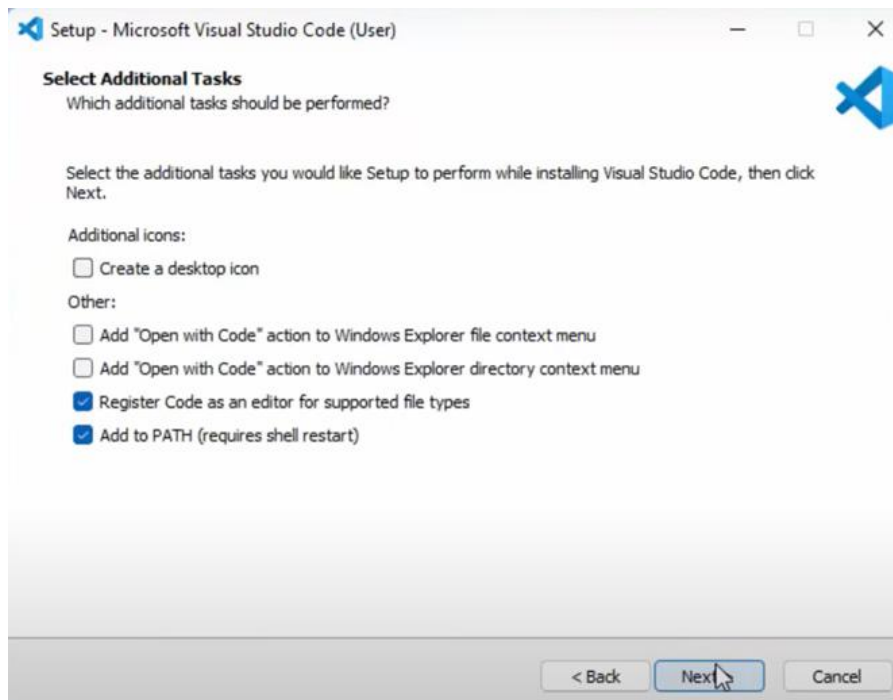


Figure 4.4 PATH selection for VS code installation

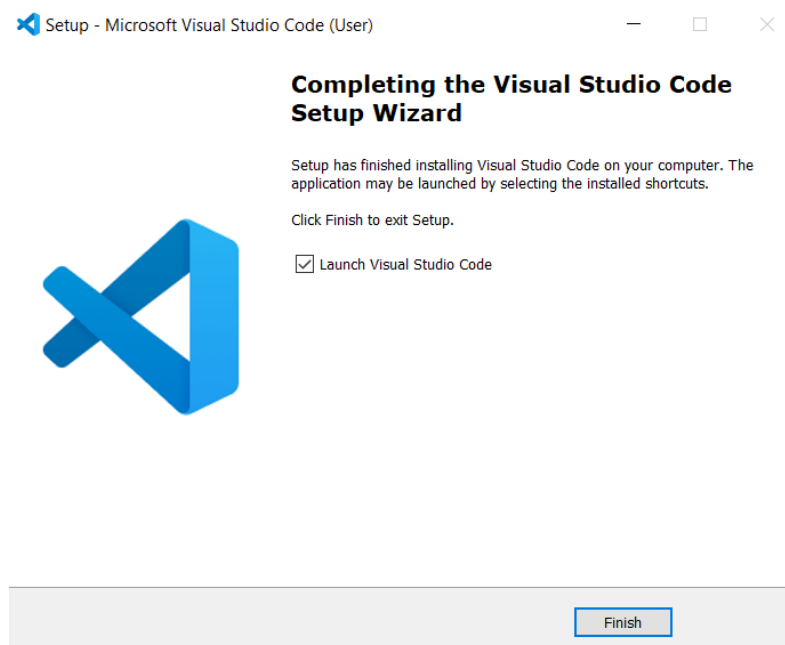


Figure 4.5 Finish VS Code Wizard Setup

After the download completes, users are required to run the executable file to initiate the installation process. The Visual Studio Code Setup Wizard will appear, guiding users through several steps, including accepting the license agreement (Figure 4.52), selecting the destination

folder for installation (Figure 4.53), and choosing optional tasks such as adding VS Code to the system PATH, enabling file association with common code extensions, and creating a desktop shortcut (Figure 4.54). Then finally after completing the setup of VS code, the user will be prompted with the finish screen of the wizard setup (Figure 4.5).

4.2.1.1 Python Extension Installations

Python Extension



Figure 4.6 Python Extension Installation

Upon opening Visual Studio Code, users are required to access the Extensions View (Figure 4.6), which can be done by clicking the Extensions icon located on the Activity Bar on the left-hand side of the window, or alternatively by using the shortcut keys Ctrl+Shift+X for Windows and Linux systems.

Once within the Extensions View, users should proceed to the search bar positioned at the top of the panel and enter the keyword “Python.” A list of available extensions related to Python will be displayed. From the search results, the official “Python” extension published by Microsoft must be located.

Following this review, users can proceed by clicking the "Install" button located at the top right of the extension's page. The installation process will begin automatically, and upon completion, the extension will be integrated into the Visual Studio Code environment. A

notification will confirm successful installation, and it is recommended that users reload Visual Studio Code if prompted, to ensure that all features of the extension are fully activated. With the Python extension installed, Visual Studio Code will now support critical Python development features, including IntelliSense, code linting, debugging, testing, and environment management, thereby optimizing the development experience for Python-based projects.

Python Debugger Extension (Optional)



Figure 4.7 Python Debugger Extension Installation

In addition to the Python extension, installing the Python Debugger extension enhances the debugging experience within Visual Studio Code by offering more specialized control and functionality when executing and inspecting Python programs.

Within the Extensions View, users should navigate to the search bar positioned at the top and enter the keyword “Python Debugger.” A list of related extensions will populate the panel. Among the search results, the extension specifically named **"Python Debugger"** should be identified. It is important to select the correct extension, which is designed to complement the

functionality of the Python extension by providing advanced debugging support tailored for Python code.

Upon selecting the Python Debugger extension, users are presented with an overview of its capabilities, including breakpoint management, step-through execution, variable inspection, and performance analysis during runtime. After reviewing the extension's description and verifying its publisher for authenticity, users can proceed by clicking the **Install** button located on the extension's page. The installation will automatically proceed, and a confirmation notification will appear upon successful completion.

4.2.2 Python and Essential Dependency Setup

4.2.2.1 System Python Version 3.12.8 Installation

Due to the requirements of a dependency to ensure that a specific python version is installed as the latest version (as of April 2025) of python is yet to be supported by the specific dependency.

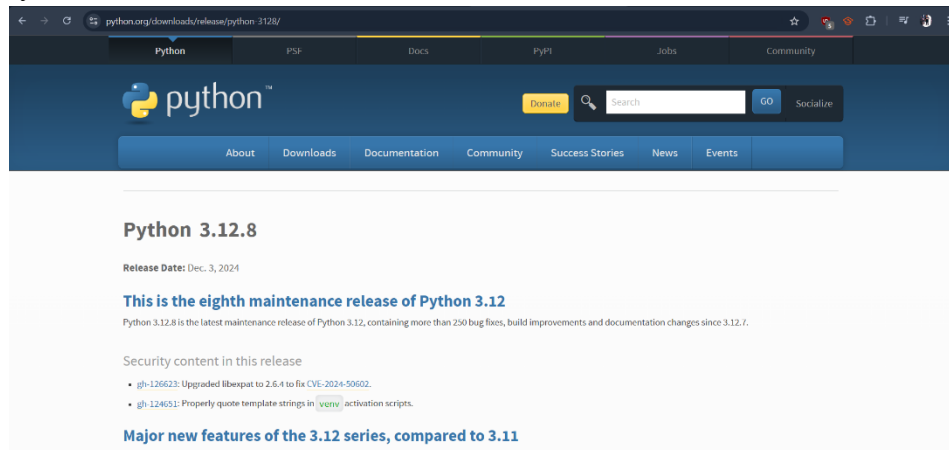


Figure 4.8 Official Python Page to Install Python Package on Machine

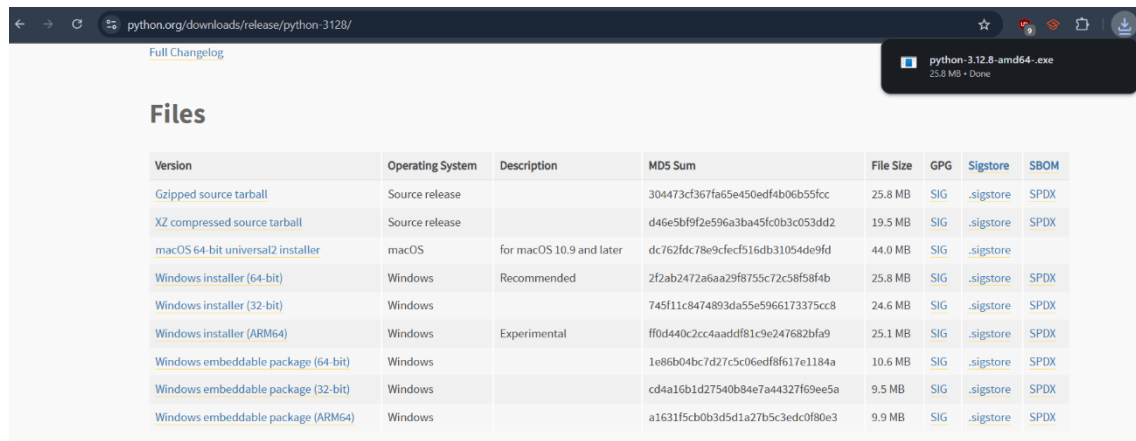


Figure 4.9 Installing windows 64-bit executable

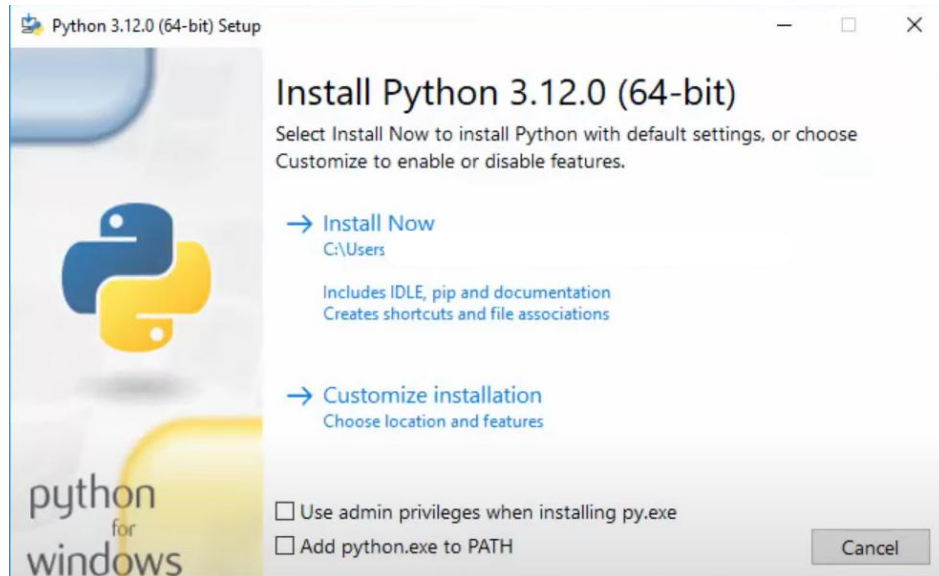


Figure 4.10 Python Setup Wizard

To obtain Python 3.12 on Windows, users should visit the official Python Downloads page (Figure 4.108), which provides installer executables for Windows 64-bit. After downloading the appropriate installer, the Windows x86-64 executable (Figure 4.109) running the file launches the Python Setup Wizard (Figure 4.10). On the initial installer screen, it is crucial to enable the **Add Python 3.12 to PATH** option to automatically configure the environment variables for both the Python interpreter and pip. Then select the **Install Now** button for a standard setup, which installs Python into the user directory without requiring administrator privileges or choose **Customize installation** to include optional features such as GUI library, and developer headers. Upon successful installation, the wizard prompts the user to disable the Windows path length limit to support longer file paths; clicking **Close** finalizes the process

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>py -0
-V:3.13 *      Python 3.13 (Store)
-V:3.12       Python 3.12 (64-bit)
```

Figure 4.121 Verifying python installation

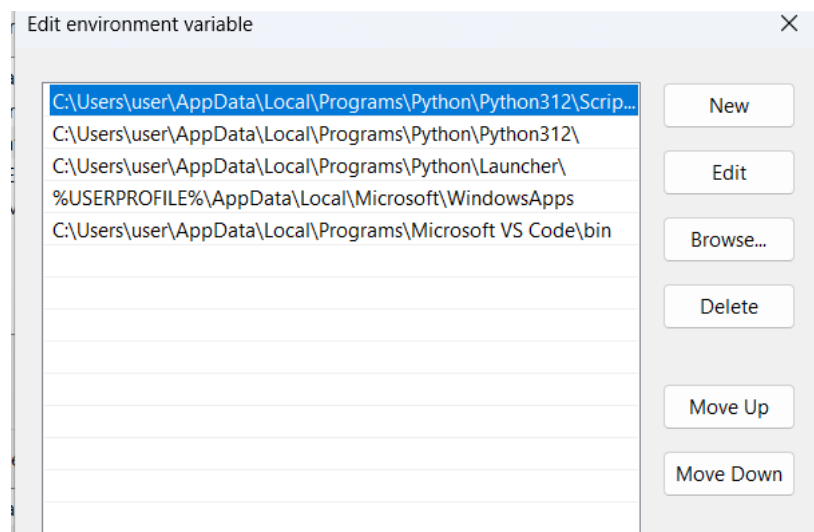


Figure 4.112 PATH of the Environment Variable utilizing the currently installed Python version

Based on Figure 4.121, the installation of the specified python version can be verified by using the `py -0` command on cmd to verify that the installation was successful. Another method is to check the environment variable files (Figure 4.112) that it is currently utilizing the installed version of Python, if it is still using an earlier version or a version which is currently not supported, we can simply remove the unwanted PATH of that specific Python version.

4.2.2.2 Virtual Environment Setup

Creating Virtual Environment

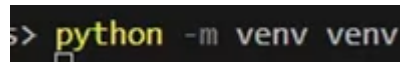


Figure 4.13 Creating Virtual Environment with the name venv

Return back to Virtual Studio Code, and open a new terminal, then by using the command in Figure 4.13 Creating Virtual Environment with the name venv, a virtual environment would be created on Virtual Studio Code utilizing the currently available Python version.

Activating Virtual Environment

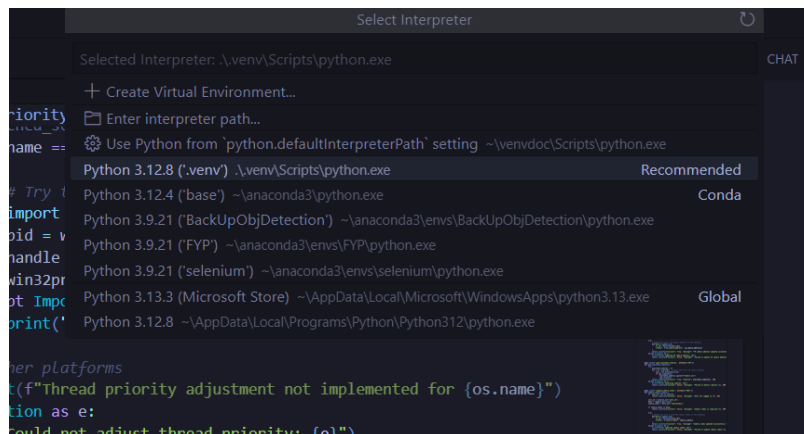


Figure 4.14 Selecting Virtual Environment for Activation

Based on Figure 4.14 Selecting Virtual Environment for Activation, the created virtual environment is selected (.venv), to be activated for being used in the workspace. This selection can be done by accessing the VS Code top middle tab to select the proper interpreter and python version.

Receiving Notification Upon Activation.

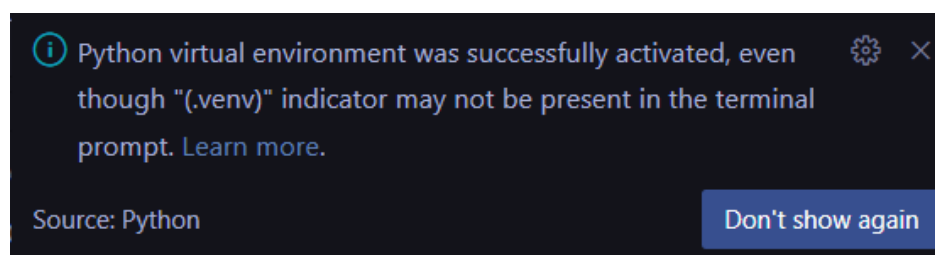
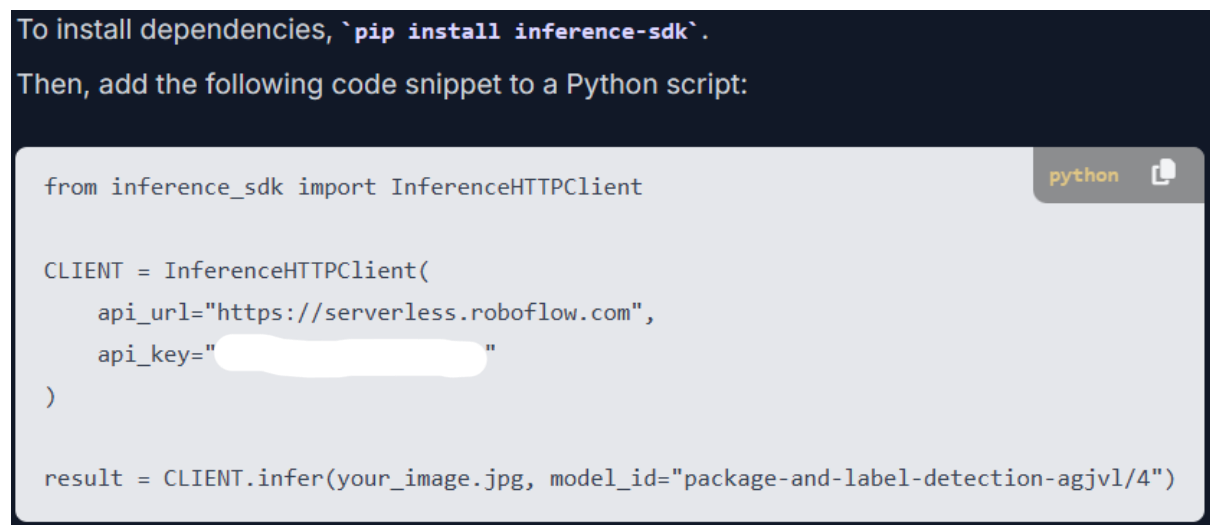


Figure 4.15 Activation Confirmation

Based on Figure 4.15 Activation Confirmation, this notification prompt will likely appear on the bottom right of the screen which will notify the user that the virtual environment has been activated and that there will not be any indicator present in the terminal prompt similarly to previous versions of virtual environment terminal indicators.

4.2.2.3 Virtual Environment Dependency installation

A screenshot of a code editor with a dark background. At the top, there is a text prompt in white: "To install dependencies, `pip install inference-sdk`." followed by "Then, add the following code snippet to a Python script:". Below this, a code block with a light blue background contains Python code. The code imports the InferenceHTTPClient from inference_sdk, initializes it with a URL and a redacted API key, and then calls the infer method with a file path and a model ID. A "python" icon is visible in the top right corner of the code block.

```
To install dependencies, `pip install inference-sdk`.  
Then, add the following code snippet to a Python script:  
  
from inference_sdk import InferenceHTTPClient  
  
CLIENT = InferenceHTTPClient(  
    api_url="https://serverless.roboflow.com",  
    api_key="REDACTED"  
)  
  
result = CLIENT.infer(your_image.jpg, model_id="package-and-label-detection-agjvl/4")
```

Figure 4.16 Inference SDK installation snippet

Based on Figure 4.16 Inference SDK installation snippet, The inference-sdk is a Python package that facilitates the integration of computer vision capabilities into software systems. To incorporate this functionality, the package must first be installed using the command *pip install inference-sdk*. Once installed, the SDK provides a streamlined interface for connecting to an inference server, which can be hosted locally or remotely. This connection is established using the *InferenceHTTPClient* class, which requires the server's *API URL* and, if applicable, an API key for authentication. The SDK enables the execution of pre-trained models on various inputs, such as static images, video streams, or RTSP feeds, and supports hardware acceleration and multiprocessing to optimize performance. This integration allows for seamless communication with the inference server, enabling the system to process visual data and retrieve detection results efficiently.

4.2.3 Workspace Setup

4.2.3.1 Frontend Setup

As of now (April 2025), VS Code with integration with Copilot would enable the users to immediately start their project or code without the need for installing any extensions for HTML, CSS and JavaScript (Figure 4.17 Creating Frontend files in VS Code).

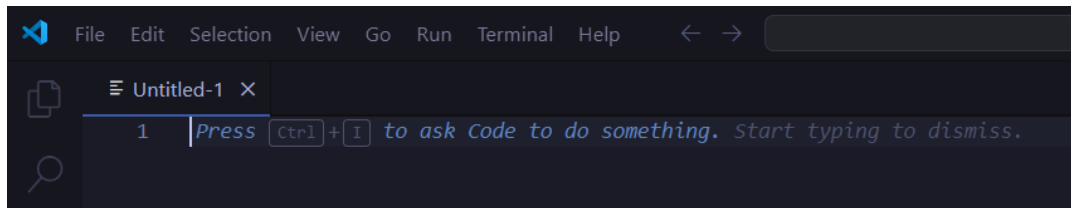


Figure 4.17 Creating Frontend files in VS Code

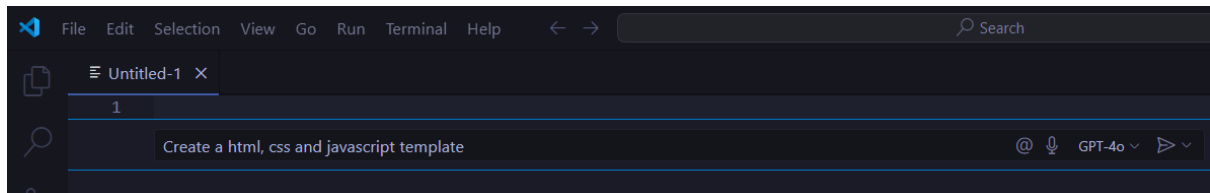


Figure 4.18 Prompting Copilot to create template code on VS Code

Based on Figure 4.18 Prompting Copilot to create template code on VS Code VS Code's latest feature is with the embedding feature with GitHub Copilot which would allow prompting to create code templates and assist developers for developing code at a faster pace and better debugging features.

4.2.3.2 Backend Setup

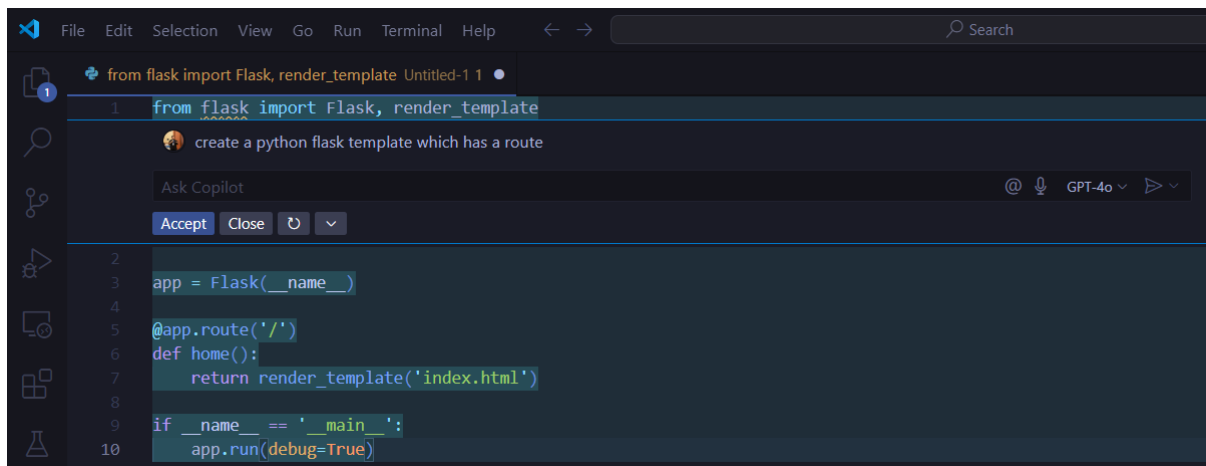


Figure 4.19 Creating a Python Flask Template

Based on Figure 4.19 Creating a Python Flask Template, By utilizing Copilot GitHub embedded into VS Code, the development of this backend can be significantly accelerated. Copilot can assist in generating boilerplate code, such as route definitions, database queries, and API integration logic, reducing the time required for manual coding. It also serves as a reliable reference for implementing best practices, ensuring that the backend is robust and secure. By leveraging Python Flask, the backend provides a lightweight yet powerful framework for building web applications. Flask's modularity allows for seamless integration of various components

4.2.3.3 Overall Workspace Setup

```
Package Detection System/
├── app.py                # Main Flask application
├── config.py             # Configuration settings (e.g., database, API keys)
├── requirements.txt      # Python dependencies
├── .env                 # Environment variables (e.g., API keys, DB credentials)
├── templates/           # HTML templates for the frontend
│   ├── base.html        # Base template for consistent layout
│   ├── dashboard.html   # Dashboard page
│   ├── settings.html    # Settings page
│   ├── reports.html     # Reports page
│   ├── archive.html     # Archive page
│   ├── profile.html     # User profile page
│   ├── edit_profile.html # Edit profile page
│   ├── edit_roi.html    # ROI editor page
│   └── partials/        # Partial templates
│       └── archive_content.html # Archive content for AJAX updates
├── static/              # Static files (CSS, JS, images)
│   ├── css/
│   │   ├── styles.css   # General styles
│   │   ├── settings.css # Styles for settings page
│   │   ├── live_feed.css # Styles for live feed page
│   │   ├── archive.css  # Styles for archive page
│   │   └── reports.css  # Styles for reports page
│   ├── js/
│   │   ├── base.js      # Base JavaScript for shared functionality
│   │   ├── dashboard.js # JavaScript for dashboard interactivity
│   │   ├── settings.js  # JavaScript for settings page
│   │   ├── reports.js   # JavaScript for reports page
│   │   ├── archive.js   # JavaScript for archive page
│   │   └── theme-switcher.js # Theme toggle functionality
│   └── images/          # Static images (e.g., logos, icons)
│       ├── ProfilePicture.jpg # Default profile picture
│       └── default-img.jpg   # Default image for missing detections
├── utils/               # Utility functions
│   ├── email_utils.py   # Email notification utilities
│   ├── video_utils.py   # Video stream handling utilities
│   └── roi_editor.py     # Region of Interest (ROI) editor logic
├── models/              # Database models
│   ├── user_model.py    # User schema and logic
│   └── detection_model.py # Detection schema and logic
├── sort.py              # SORT algorithm for object tracking
├── README.md            # Project documentation
├── .dockerignore        # Docker ignore file
└── .env                 # Environment variables
```

Figure 4.20 Workspace Directory Overview

Based on Figure 4.20 Workspace Directory Overview The directory structure of the system is logically partitioned into three main components: **Backend**, **Frontend**, and **Other Supporting Components**. This organization ensures modularity, maintainability, and scalability, which are critical for the development and deployment of the Package Detection System.

1. Backend

The backend is the core of the system, responsible for handling server-side logic, database interactions, and API integrations. It includes the following key components:

- **app.py:**

The main Flask application file that defines routes, initializes the application, and handles backend logic such as user authentication, video stream processing, and object detection. It also integrates with external APIs like Roboflow and MongoDB for inference and data storage.

- **config.py:**

Contains configuration settings for the application, such as database connection strings, API keys, and Flask secret keys. It uses environment variables to securely manage sensitive information.

- **models/:**

This folder contains database models for MongoDB collections, such as `user_model.py` for user data and `detection_model.py` for detection records. These models define the schema and logic for interacting with the database.

- **utils/:**

A collection of utility scripts, such as **email_utils.py** for sending email notifications, `video_utils.py` for handling video streams, and `roi_editor.py` for managing the Region of Interest (ROI) editor.

- **sort.py:**

Implements the SORT (Simple Online and Realtime Tracking) algorithm for object tracking, which is used to enhance the detection capabilities of the system.

2. Frontend

The frontend provides the user interface for interacting with the system. It is built using HTML, CSS, and JavaScript, with Flask's Jinja2 templating engine rendering dynamic content. Key components include:

- **templates:**

Contains HTML templates for various pages, such as:

- **dashboard.html:** Displays the live feed and detection history.
- **settings.html:** Allows users to configure video sources and other preferences.
- **reports.html:** Visualizes detection data using interactive charts.
- **archive.html:** Displays archived detection records.
- **profile.html:** Shows user profile information.
- **base.html:** A base template that provides a consistent layout for all pages.

- **css:**

Contains CSS files for styling the application, such as:

- **styles.css:** General styles for the application.
- **settings.css:** Styles specific to the settings page.
- **reports.css:** Styles for the reports page.
- **live_feed.css:** Styles for the live feed section.

- **js:**

Contains JavaScript files for interactivity and dynamic content updates, such as:

- **dashboard.js:** Handles interactivity on the dashboard page.

- **settings.js:** Manages video source selection and settings updates.
- **reports.js:** Generates and updates charts for detection reports.
- **theme-switcher.js:** Implements the theme toggle functionality (light/dark mode).
- **images:**
Stores static images, such as profile pictures and default placeholders for missing detections.

3. Other Supporting Components

These components provide additional functionality and support for the system:

- **requirements.txt:**
Lists all Python dependencies required for the project, ensuring that the environment can be set up consistently.
- **.env:**
Stores environment variables, such as API keys and database credentials, to enhance security and flexibility.
- **README.md:**
Provides documentation for the project, including setup instructions, features, and usage guidelines.
- **Backup css/:**
Contains backup versions of CSS files, ensuring that previous styles are preserved for reference or rollback.

4.2.4 MongoDB Setup

4.2.4.1 MongoDB Account Creation

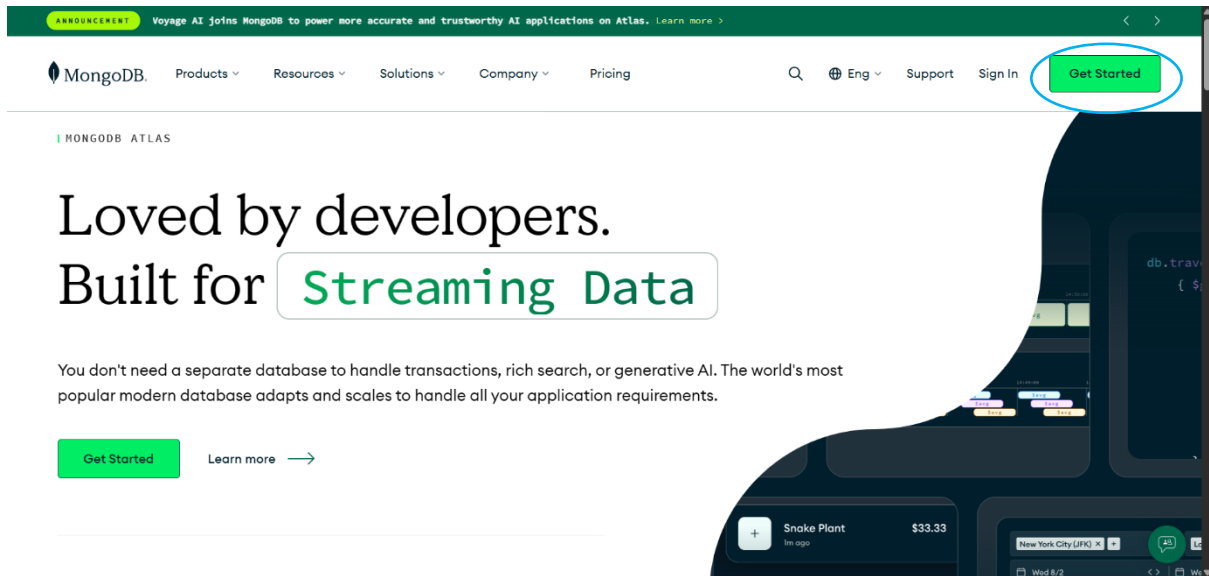


Figure 4.21 MongoDB Landing page

Based on Figure 4.21, the database utilized for the proposed system would be MongoDB where an account is required to be created on the official MongoDB (<https://www.mongodb.com/>) site to utilize MongoDB Atlas.

mongodb.com/cloud/atlas/register

MongoDB

Sign up

See what Atlas is capable of for free

 Sign up with Google

First Name*

Last Name*

Company

Email*

Password*

☐ I agree to the Terms of Service and Privacy Policy.

Create your Atlas account

Sign in

© 2025 MongoDB, Inc. [Contact](#) [Privacy Policy](#) [Terms & Conditions](#)

Figure 4.22 Account Creation Steps

The regular procedure when signing up for the first time would be as illustrated in Figure 4.22, here the option taken would be signing up with an account using Google account.



Accept Privacy Policy & Terms of Service

Please acknowledge the following terms and conditions to finish creating your account.

☒ I accept the [Privacy Policy](#) and the [Terms of Service](#)

Figure 4.23 Acceptance Terms and Conditions

After selecting the desired Google account to create a MongoDB account, there will be a prompt to agree to the terms and services of the site which will be read and agreed then submitted (Figure 4.23)

Atlas

Welcome to Atlas. Let's build something great.

Provide your goals, coding language preferences, and project details to get specific tools and guidance for your project needs.

GETTING TO KNOW YOU

What is your primary goal?

How long have you been developing software with MongoDB?

GETTING TO KNOW YOUR PROJECT

What programming language are you primarily building on MongoDB with?

What type(s) of data will your project use?
You can choose as many as you want

Will your application include any of the following architectural models?
You can choose as many as you want

Figure 4.24 MongoDB Atlas Personalisation

According to Figure 4.24, there will be a personalisation page which can be skipped as it is to be used for personal use rather than a commercial utilization.

4.2.4.2 Cluster Creation

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

☐ M10 \$0.09/hour

Dedicated cluster for development environments and low-traffic applications.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

☐ Flex From \$0.011/hour Up to \$30/month

For application development and testing, with on-demand burst capacity for unpredictable traffic.

STORAGE	RAM	vCPU
5 GB	Shared	Shared

☒ Free

For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

Free forever! Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Configurations

Name
You cannot change the name once the cluster is created.

FYP

Provider

aws Google Cloud Azure

Quick setup

☒ Automate security setup

☒ Preload sample dataset

[I'll do this later](#)

[Go to Advanced Configuration](#)

[Create Deployment](#)

Figure 4.25 Creating Cluster

Now upon completion or skipping of the personalization page, we will have to choose the plan of cluster deployment, since this proposed system is in a form of a prototype, we will be utilizing the free tier for cluster deployment that will be a sharing server service and naming our cluster as FYP and then proceeding to Create the Deployment (Figure 4.25)

4.2.4.2 Connecting to Cluster

Now in Figure 4.26, there will be a configuration of administrative right to set the username

Connect to FYP

1 2 3
Set up connection security Choose a connection method Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

1. Add a connection IP address

✓ Your current IP address (121.121.56.166) has been added to enable local connectivity. Only an IP address you add to your Access List will be able to connect to your project's clusters. Add more later in [Network Access](#).

2. Create a database user

This first user will have [atlasAdmin](#) permissions for this project.

We autogenerated a username and password. You can use this or create your own.

ⓘ You'll need your database user's credentials in the next step. Copy the database user password.

Username Password

FYP

SHOW Copy

Create Database User

Close Choose a connection method

Figure 4.26 Connecting Cluster to External Frameworks

and password of the first user which has access to the database and there is a following step to connect to the database which will be accessed after choosing a connection method

Connect to FYP



⚠ You can't connect yet. Set up your user security permission in the first step.

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver	Version
Python ▼	4.7 or later ▼

2. Install your driver

Run the following on the command line

Note: Use appropriate Python 3 executable

```
python -m pip install "pymongo[srv]"
```

[View MongoDB Python Driver installation instructions.](#)

3. Add your connection string into your application code

Use this connection string in your application

☐ View full code sample

```
mongodb+srv://<db_username>:<db_password>@fyp.ta9ekym.mongodb.net/?  
retryWrites=true&w=majority&appName=FYP
```

Replace **<db_password>** with the password for the **<db_username>** database user. Ensure any option params are [URL encoded](#).

Figure 4.27 Connecting to MongoDB Atlas through MongoDB Driver

Figure 4.27 illustrate the steps to connect to the MongoDB database by utilizing the connection with MongoDB Driver through Python, which includes installation of dependencies and addition of a connection string within the application code, which in this case would be the proposed system to connect to it, by replacing the placeholders of `<db_username>` and `<db_password>`

4.2.4.3 Database and Collection Creation

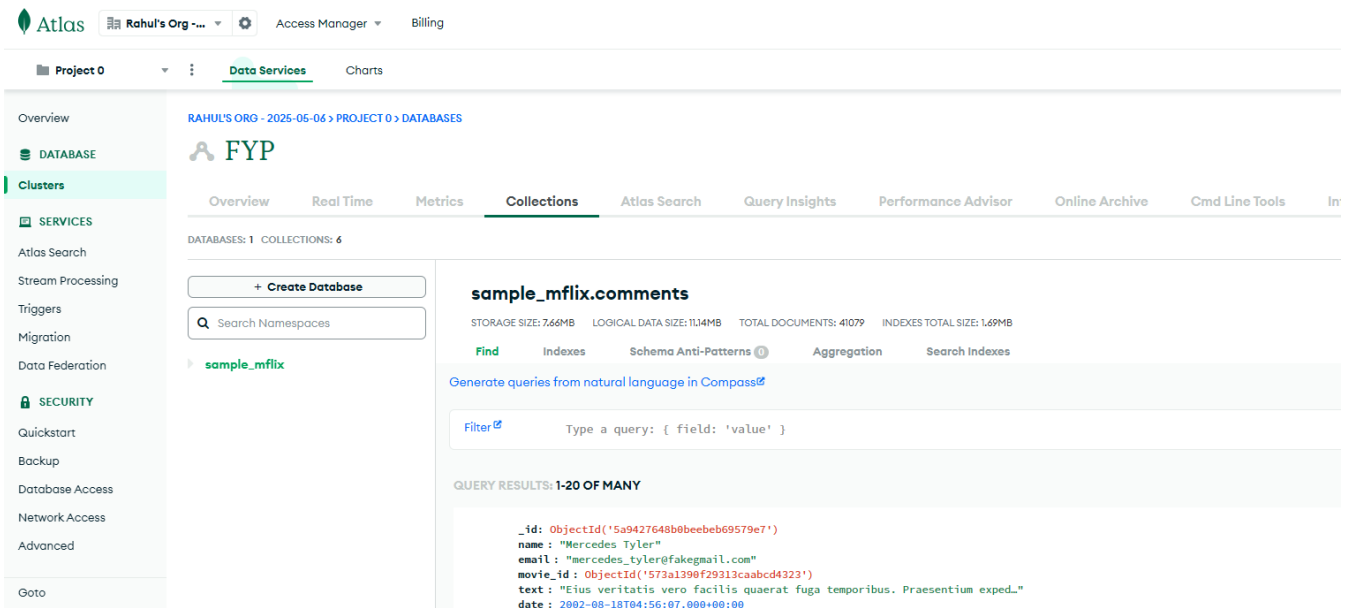


Figure 4.28 Navigate to the collection of the FYP Cluster

Based on Figure 4.28, after completing the setup, navigate to the FYP Cluster's Collections where there will be a demo sample of database named "example_mflix", here it will display an example database within the cluster, that we can use as a reference when we would want to build another database.

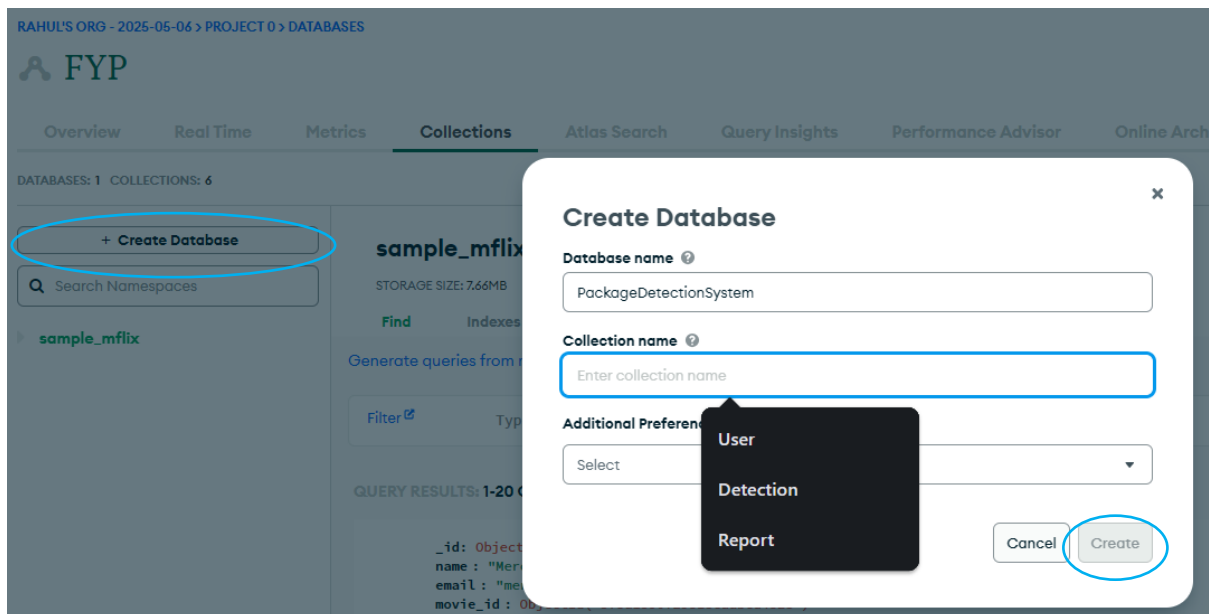


Figure 4.29 Creating Database and Collections

According to Figure 4.29, a Database with name “PackageDetectionSystem” would be created to wrap around 3 different Collections named User, Detection and Report respectively, after filling in the placeholders, the create button will become available and clickable.

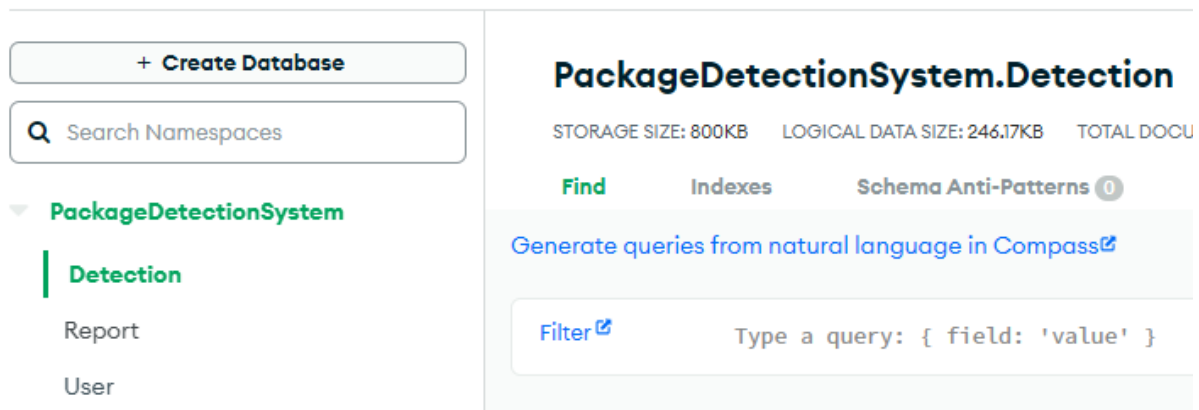


Figure 4.30 After Creation view

Upon successful creation of the database and its collections, the display on the MongoDB page would look similarly to Figure 4.30.

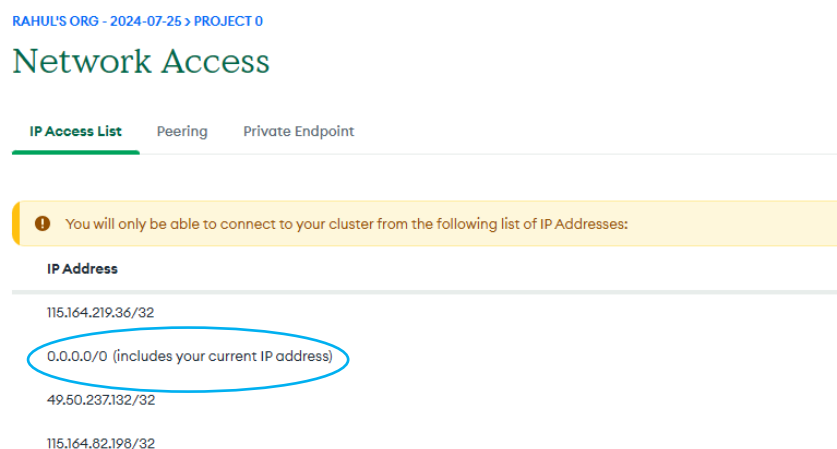


Figure 4.31 Network Access

To ensure secure access to a MongoDB database hosted on the cloud, IP whitelisting is used as a primary network access control mechanism. As illustrated in Figure 4.31, MongoDB allows connections only from explicitly approved IP addresses listed under the IP Access List section of the Network Access settings. This means that any client attempting to connect to the database must originate from an IP address that has been added to this list. Users can whitelist a new IP

address by navigating to the MongoDB Atlas dashboard, selecting the appropriate project, then going to the Network Access tab, and clicking on the "Add IP Address" button. They can choose to add a specific IP (e.g., 115.164.219.36/32) or allow access from all IPs temporarily using 0.0.0.0/0, although the latter is not recommended for production environments due to security risks. This method of whitelisting helps prevent unauthorized access and protects the database from external threats by restricting entry to known and trusted clients only.

4.2.4 Docker Desktop Setup

4.2.4.1 Docker Desktop Installation

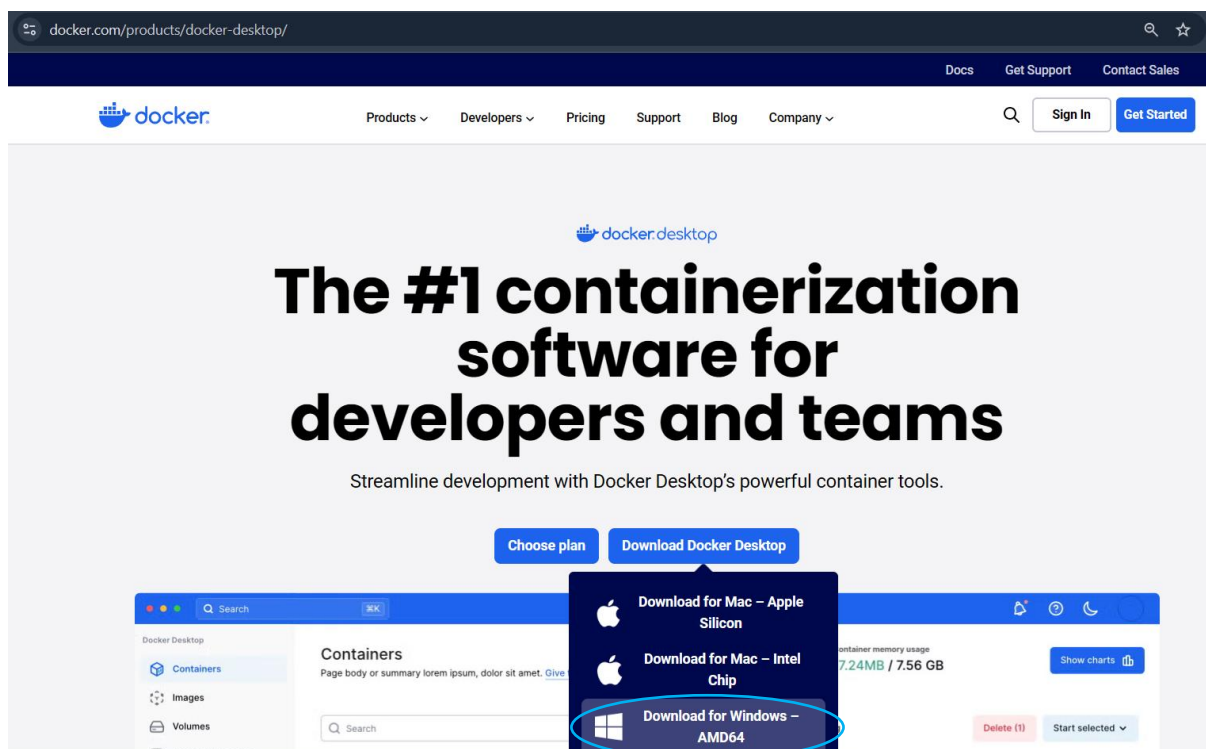


Figure 4.32 Docker Desktop Landing Page

According to Figure 4.32, upon visiting the Docker Desktop Official site (<https://www.docker.com/products/docker-desktop/>), the corresponding Docker Desktop version to the machine is downloaded.

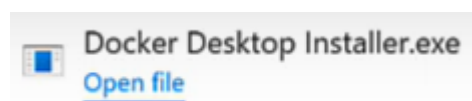


Figure 4.33 Docker Desktop Installer.exe download notification

After downloading the corresponding executable file, a notification tab (normally on the top right) of the browser will appear and notify that the download was completed such as in Figure 4.333. Then, the installation phase may begin by clicking “open file”.

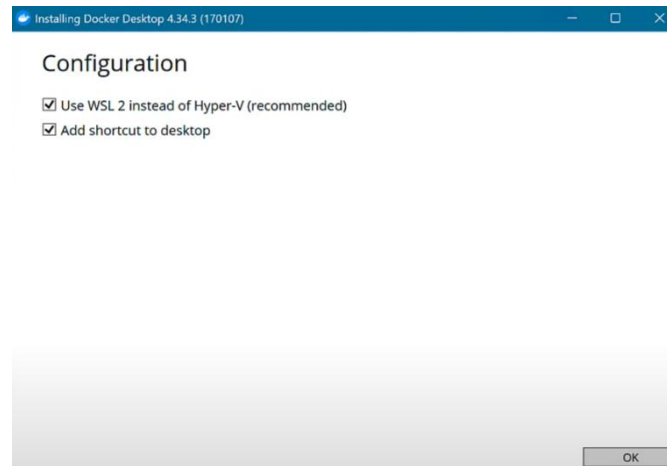


Figure 4.34 Configuration phase

Based on Figure 4.344, During the installation of Docker Desktop on Windows, it is recommended to use the Windows Subsystem for Linux version 2 (WSL 2) as the backend for running containers (*Docker Inc.*, n.d.). This is because WSL 2 offers significant performance improvements over its predecessor, including faster file system access and up to twentyfold increases in read/write speeds (Mattwojo, 2023), 100 % system call compatibility, and lower resource usage compared to a traditional Hyper-V virtual machine (*Docker Inc.*, n.d.). By leveraging WSL 2, Docker can run Linux containers natively within a lightweight virtualized environment directly integrated into Windows, ensuring faster daemon startup times, improved I/O performance, and a smoother developer experience (*Docker Inc.*, n.d.). Additionally, WSL 2 enables seamless interoperability between Windows and Linux tools, which is particularly beneficial for development environments that rely on cross-platform workflows (Mattwojo, 2023a)

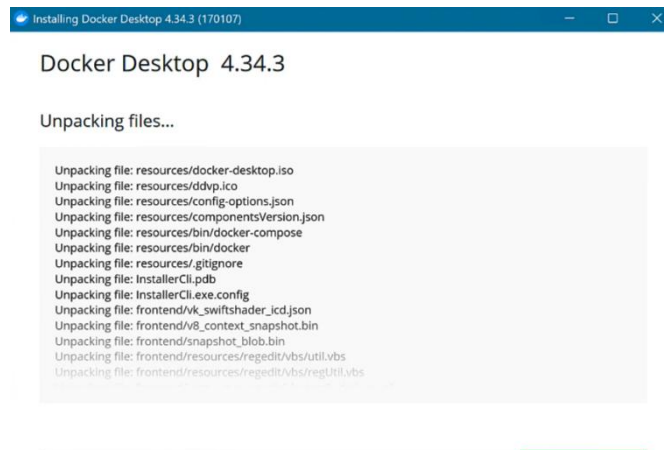


Figure 4.365 File Unpacking

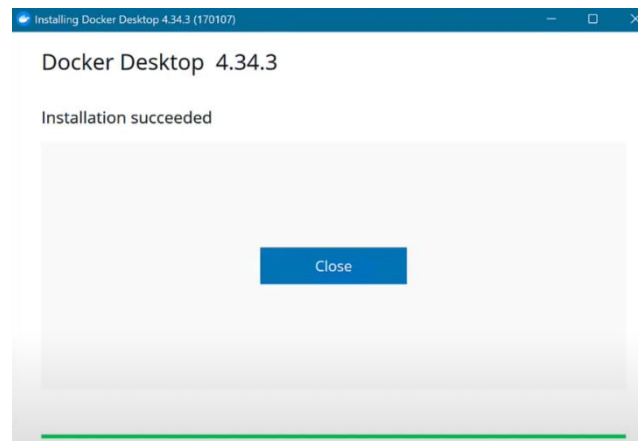


Figure 4.356 Installation Complete

As shown in Figure 4.365, during the installation of Docker Desktop version 4.34.3, the setup process enters the file unpacking stage, where essential components and resources are extracted and prepared for system integration. This includes ISO files, configuration JSON files, binary executables, and frontend resources. The unpacking process ensures that Docker Desktop has access to all the necessary files required for its runtime environment and user interface. These components are critical for enabling Docker's core functions, such as container orchestration, image management, and integration with WSL 2 on Windows systems. Following the successful unpacking of files (Figure 4.356), the installation process completes and presents the user with a confirmation screen indicating that Docker Desktop has been successfully

installed, as depicted. This prompt reassures users that the setup has concluded without errors and that the application is now ready for configuration and use.

4.2.4.2 Docker Desktop Account Registration/Login

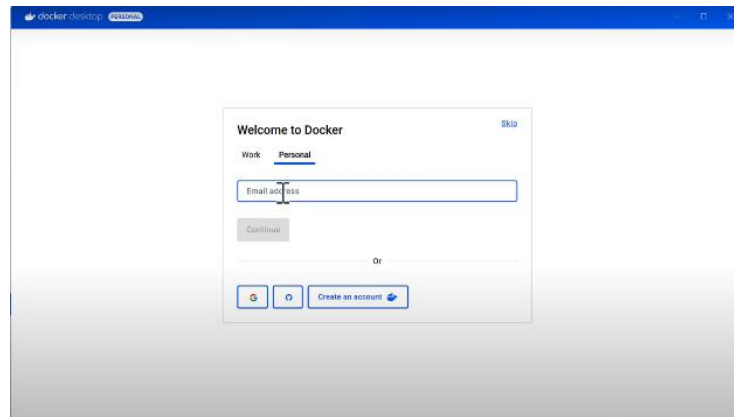


Figure 4.37 User Login/Registration

Figure 4.37 displays the Docker Desktop login interface that appears upon launching the application for the first time. Users are prompted to either sign in with an existing Docker Hub account or create a new one. Account registration or login is necessary to access certain Docker features such as pulling container images from Docker Hub or using Docker extensions. The interface offers two tabs, *Work* and *Personal*, that is to tailor the user experience based on their role. This step is essential for identifying the user and synchronizing Docker settings across devices.

Welcome Survey
Step 1 of 2

What's your role?

☐ Full-stack developer

☐ Front-end developer

☐ Back-end developer

☐ Site reliability engineer

☐ Platform engineer

☐ DevOps specialist

☐ Infrastructure manager

☐ System administrator

☐ Security engineer

☐ Data scientist

☐ Product manager

☐ Teacher

☐ Student

☐ Other

Figure 4.38 Docker Desktop Survey

Figure 4.38 shows the optional welcome survey presented after account registration or login. The survey is designed to understand the user's role and use case for Docker, ranging from Full-stack Developer to Data Scientist or Student. Although skipping the survey is allowed

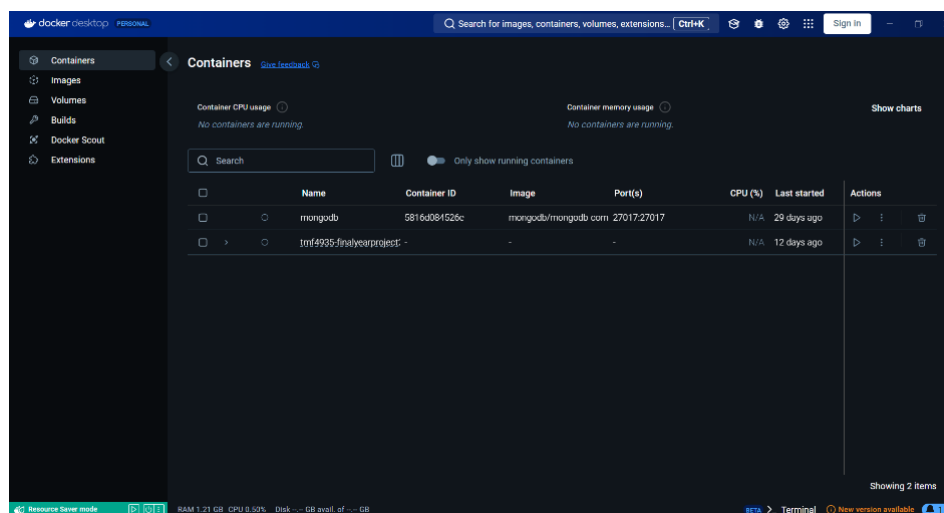


Figure 4.39 Docker Desktop Main Page

Figure 4.39 illustrates the Docker Dashboard after a successful login and setup. This view shows a list of running and stopped containers. In this example, a MongoDB container is already active, along with an already created container for the proposed system. The dashboard provides details such as container name, image used, ports, uptime, CPU usage, and management options like start, stop, and delete. This visual management interface simplifies

container orchestration and offers an intuitive way for users to monitor and control Dockerized applications without relying solely on command-line tools.

4.3 System Hierarchy

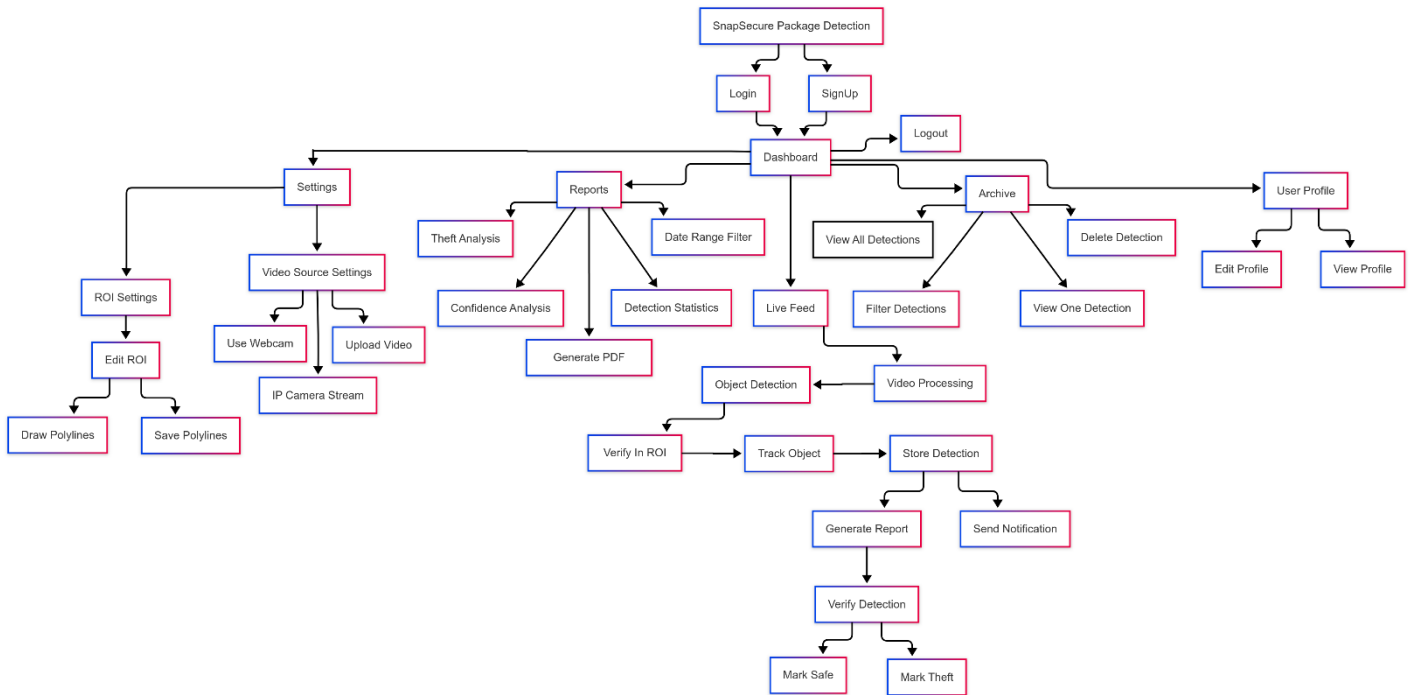


Figure 4.40 System Hierarchy of the Proposed System

At the topmost level, there exists the SnapSecure Package Detection System, which comprises several integrated subsystems, each tasked with specific functionalities related to package monitoring and theft prevention. These subsystems are further subdivided into smaller modules or components, resulting in a comprehensive layered structure. This hierarchical arrangement facilitates the division of specialized tasks, thereby enabling efficient management, maintenance, and comprehension of the entire system. It offers a systematic approach for implementing and operating complex monitoring systems, ensuring that each component seamlessly integrates into the broader framework and contributes to the system's overall functionality.

Detailed Hierarchy Description

According to the system hierarchy diagram, the SnapSecure Package Detection System begins with authentication options of Login and SignUp, both leading to the central Dashboard which serves as the primary navigation hub. From this **Dashboard**, users can access six main modules: Settings, Reports, Archive, Live Feed, User Profile, and Logout while also comprising of core functionalities such as displaying brief details of the detection history, marking packages safe or theft and displaying live detections.

The **Settings** module encompasses crucial configuration options including ROI Settings and Video Source Settings. Under ROI Settings, users can access Edit ROI functionality which further provides Draw Polyline and Save Polyline features for defining precise monitoring areas. The Video Source Settings component offers three input options: Use Webcam for local camera access, Upload Video for pre-recorded footage analysis, and IP Camera Stream for remote monitoring.

The **Reports** module delivers analytical capabilities through four sub-modules: Theft Analysis for examining potential theft incidents, Confidence Analysis for evaluating detection accuracy, Detection Statistics for quantitative performance metrics, PDF Generator and Date Range Filter for temporal data filtering.

Within the **Archive** module, users can access View All Detections to browse the comprehensive detection history, Filter Detections to narrow results by specific criteria, View One Detection for detailed examination of individual incidents, and Delete Detection to remove unwanted records from the system.

The **Live Feed** module represents the detection workflow, beginning with Video Processing which leads to Object Detection. Detected objects undergo verification through the Verify In ROI module to ensure they fall within designated monitoring areas. Objects are then tracked via the Track Object module, with confirmed detections stored in the database through

the Store Detection component. This triggers two parallel processes: Generate Report for documentation and Send Notification to alert the user.

Following report generation, the system presents the **Verify Detection** interface where users can make decisive classifications by selecting either Mark Safe for authorized package interactions or Mark Theft for suspicious activities requiring further action.

The **User Profile** module contains two sub-modules: Edit Profile for modifying personal information and preferences, and View Profile for reviewing account details.

This hierarchical structure ensures that the SnapSecure system maintains logical organization while providing comprehensive functionality for package monitoring, detection, verification, and analysis, all within an intuitive user interface that guides users through the process of securing their packages against potential theft.

4.4 Workflow of the Proposed System

4.4.1 Detection Workflow

The package detection process is initiated when video input is received from the configured source, which can be a webcam, uploaded video, or IP camera stream. The system's object detection algorithm continuously analyzes frames from the video feed, specifically trained to identify packages within the field of view. This initial detection serves as the foundation for the subsequent security monitoring process, ensuring that all potential package interactions are captured and evaluated.

The theft detection mechanism is dependent on both the identification of packages and their persistence within user-defined Regions of Interest (ROI). In other words, the system monitors packages that remain within designated areas and tracks their interactions over time. When a package that was previously detected is no longer visible within the ROI, the system flags this as a potential theft incident, triggering the alert workflow. This conditional triggering

mechanism helps reduce false positives by focusing only on relevant spatial contexts where packages should remain secure.

4.4.2 Notification Workflow

The notification process is activated only when specific conditions are met - namely, a package disappearance from the monitored ROI. Upon detection of such an event, the system immediately generates a detailed report containing visual evidence, timestamps, and confidence ratings. Simultaneously, the system dispatches notifications through the user's configured channels, ensuring prompt awareness of potential security breaches. This dual-action response enables both immediate intervention and subsequent investigation capabilities.

4.4.3 Verification Workflow

The verification workflow is dependent on human judgment to classify detection events accurately. When users receive notifications, they can access the generated reports through the system interface and review the evidence presented. Users then make definitive classifications by marking the event as either "Safe" (authorized package interaction) or "Theft" (unauthorized removal requiring action). This human-in-the-loop verification process strengthens the system's reliability by combining algorithmic detection with human discernment, creating a robust security mechanism that minimizes both false alarms and missed incidents.

4.4.4 Reporting Workflow

The reporting and analytics processes build upon the accumulated detection and verification data, transforming individual incidents into actionable intelligence. The system continuously aggregates detection outcomes, verification decisions, and temporal patterns to generate comprehensive statistics and trend analyses. This ongoing data collection and processing enables users to identify vulnerability patterns, peak risk periods, and effectiveness metrics of their security setup, facilitating informed decisions about package security improvements.

4.5 Coding of Algorithms Proposed

Implementing a custom deep learning model for package detection from scratch would require a large, annotated dataset and substantial computational resources. Training such models typically demands powerful GPUs and significant development time, making the approach highly resource-intensive and less feasible for rapid prototyping or real-time applications. Given these constraints, the decision was made to utilize Roboflow's hosted API and pretrained object detection model instead. Roboflow offers a scalable and less resource-intensive alternative by abstracting the complexities of model training, optimization, and deployment. The API-based approach allows the application to perform object detection reliably without the need for local GPU processing, thereby reducing hardware dependency while enabling easy integration and scalability. This method ensures that the system remains efficient, responsive, and adaptable for various use cases in package theft detection.

4.5.1 Package Theft Detection Algorithm

The outcome of Package Detection within the ROI, and the removal of the Detection would be the root for triggering the Package Theft Detection. This would be pointed to the algorithm which specifies in the detection of Packages within an area of interest in the current frame which can either be live video feed or in pre-recorded format

4.5.1.1 Overview of the Core of Object Detection in The Proposed System

The object detection system in SnapSecure can identify and tracking package-related objects across successive video frames. Using the Roboflow API for cloud-based inference, packages can be reliably detected in both uploaded videos and live camera feeds. To enhance this capability, the system implements the SORT (Simple Online and Realtime Tracking) algorithm, which maintains object identity across multiple frames even when objects temporarily disappear or become partially occluded. This approach addresses the limitations of frame-by-frame detection, which cannot establish temporal relationships between objects appearing in consecutive frames.

The SnapSecure detection system is further specialized through Region of Interest (ROI) filtering, which allows users to define specific monitoring areas using polylines. This feature enables targeted surveillance by focusing only on relevant spatial contexts where packages are expected to remain secure. Within these ROIs, the system can detect packages with confidence ratings, distinguishing between high-confidence (>0.75), medium-confidence ($0.5-0.75$), and low-confidence (<0.5) detections to provide nuanced security monitoring.

Tracking Algorithm

Each detected package is assigned a unique track ID by the SORT algorithm, allowing for continuous tracking across frames and temporal analysis of package presence. When a previously tracked package disappears from a monitored ROI, the system triggers the theft detection workflow, generating reports and sending notifications. The detection data is comprehensively stored in MongoDB, including detection timestamp, confidence scores, classification labels, and visual evidence in the form of encoded frame images with bounding boxes drawn around detected objects.

Core Functionality Code Snippet

```
# Initialize Roboflow API client
CLIENT = rf.Roboflow(api_key=API_KEY)

# Set model ID for package detection
MODEL_ID = "package-and-label-detection-agjvl/4"
```

Figure 4.41 Roboflow API and Model Initialization

Figure 4.41 initializes the Roboflow API client using the developer's API key and specifies the trained package detection model to be used for inference. The system uses a specialized model (package-and-label-detection-agjvl/4) that has been trained to recognize packages and shipping labels with high accuracy.

Object Detection Function

```
def detect_objects(frame):
```

```

_, encoded_image = cv2.imencode('.jpg', frame)
with tempfile.NamedTemporaryFile(delete=False, suffix=".jpg") as
temp_file:
    temp_file.write(encoded_image)
    temp_image_path = temp_file.name

    try:
        # Perform inference
        result = CLIENT.infer(temp_image_path, model_id=MODEL_ID)
    finally:
        # Ensure the temporary file is deleted
        if (os.path.exists(temp_image_path)):
            os.remove(temp_image_path)

return result

```

Figure 4.42 detect_objects function

Figure 4.42 handles the core detection process by encoding the current video frame as a JPEG image, creating a temporary file, and sending it to the Roboflow API for inference. The function ensures proper resource management by cleaning up the temporary file after processing is complete, regardless of whether the API call succeeds or fails.

ROI Verification Function

```

def is_point_in_polylines(point, polylines):
    for polyline in polylines:
        if cv2.pointPolygonTest(polyline, point, False) >= 0:
            return True
    return False

```

Figure 4.43 is_point_in_polylines function

Figure 4.43 which is a utility function determines whether a detected object is within any user-defined Region of Interest (ROI). It uses OpenCV's *pointPolygonTest* function to check if the center point of a detected object falls within any of the polylines that define monitoring areas. This spatial filtering ensures that only detections in relevant areas trigger security monitoring.

Detection Preprocessing and ROI Filtering

```
# Process video frames
for frame in video_stream:
    # Detect objects in the current frame
    detections = detect_objects(frame)

    # Prepare detections for SORT tracking
    sort_detections = []
    labels = {}
    for i, detection in enumerate(detections.get("predictions", [])):
        x, y, w, h = detection["x"], detection["y"], detection["width"],
detection["height"]
        confidence = detection["confidence"]
        label = detection["class"]

        # Convert to bounding box format
        x1 = x - w / 2
        y1 = y - h / 2
        x2 = x + w / 2
        y2 = y + h / 2

        # Check if object center is within any ROI polyline
        center_point = (int(x), int(y))
        if not is_point_in_polylines(center_point, polylines):
            continue # Skip detections outside the polyLines

        sort_detections.append([x1, y1, x2, y2, confidence])
        labels[i] = {"label": label, "confidence": confidence}
```

Figure 4.44 Preprocessing and ROI Filtering

According to Figure 4.44, it processes each video frame to extract and format detections for tracking. It converts the detection coordinates from center-width-height format to the bounding box format required by SORT. It also performs spatial filtering by checking if each detection's center point falls within any defined ROI polyline, discarding detections outside these areas to focus monitoring on relevant regions.

SORT Tracking Implementation

```
# Update SORT tracker with new detections
if len(sort_detections) > 0:
    tracked_objects = tracker.update(np.array(sort_detections))
else:
    tracked_objects = []
```

Figure 4.45 SORT Tracker implementation snippet

The filtered detections are passed to the SORT algorithm, which performs object tracking across frames. The `tracker.update()` method associates new detections with existing tracks or creates new tracks as needed, implementing Kalman filtering for motion prediction and the Hungarian algorithm for data association. This maintains object identity across frames, even with temporary occlusions (Figure 4.45).

Tracker Update Process

When filtered detections are passed to the SORT tracker via the `update()` method, a sophisticated multi-stage process begins. First, the tracker predicts the new positions of all currently tracked objects using their motion models. These predictions represent where previously tracked packages are expected to appear in the current frame based on their previous movements. The tracker then attempts to associate these predictions with the new detections from the current frame. This association determines which detections correspond to already-tracked objects and which represent newly appeared packages. The update method returns tracking results that include bounding box coordinates and unique tracking IDs for each successfully tracked object, enabling the system to maintain consistent identifiers for packages across frames.

Kalman Filtering for Motion Prediction

Kalman filtering forms the core of SORT's motion prediction capabilities. Each tracked object is represented by a Kalman filter that maintains a state vector including the object's position, size, aspect ratio, and their respective velocities. When a new frame is processed, but before new detections are considered, the Kalman filter predicts the object's new state based on its motion model. This prediction step is crucial for maintaining tracking continuity even when detections are missed in some frames. The Kalman filter's recursive nature allows it to learn from observed movements, progressively refining its motion model to better predict future positions. When new detections are associated with existing tracks, the Kalman filter updates

its state estimates, balancing the prediction with the new measurement to produce an optimal state estimate.

Hungarian Algorithm for Data Association

The challenge of determining which detection corresponds to which existing track is solved using the Hungarian algorithm, which provides an optimal solution to the assignment problem. The algorithm begins by calculating an Intersection over Union (IoU) matrix between all predicted track positions and new detections. This matrix measures the spatial overlap between each prediction-detection pair, with higher values indicating better matches. The Hungarian algorithm then finds the optimal assignment that maximizes the total IoU across all matches, ensuring that each detection is assigned to the most likely track and each track to the most likely detection. This approach handles the complexity of multiple simultaneous objects more effectively than greedy matching methods, particularly in scenarios with dense package arrangements or crossing trajectories.

Maintaining Object Identity and Handling Occlusions

One of SORT's key strengths is its ability to maintain object identity across frames, even when objects temporarily disappear or become occluded. When a previously tracked object is not matched with any detection in the current frame, the tracker doesn't immediately discard it. Instead, it continues to predict the object's position for a specified number of frames (defined by the `max_age` parameter), using the Kalman filter's motion model. If the object reappears within this window, the tracker can successfully re-associate it with its previous track ID, maintaining identity consistency. For new objects entering the scene, the tracker requires a minimum number of consecutive detections (defined by the `min_hits` parameter) before confirming it as a valid track, which helps filter out spurious detections. These mechanisms ensure that the package count remains accurate by avoiding duplicate counting when packages temporarily disappear from view due to occlusions or detector inconsistencies.

Detection Processing and Database Storage

```
# Process tracking results
for i, track in enumerate(tracked_objects):
    track_id = int(track[4]) # Unique ID assigned by SORT
    x1, y1, x2, y2 = map(int, track[:4])
    label = labels.get(i, {}).get("label", "Unknown")
    confidence = labels.get(i, {}).get("confidence", 0.0)

    # Store detection in database
    timestamp = datetime.now().isoformat()
    store_detection_in_db(track_id, label, confidence, timestamp, user_id,
frame)
```

Figure 4.46 Preparing data to be stored in database

Figure 4.46 processes each tracked object, extracting its unique ID, position, classification label, and confidence score. For each tracked package, it generates a timestamp and stores all detection data in the MongoDB database. This storage enables historical review of detections, theft verification, and analytics functionality in the SnapSecure system.

4.5.1.2 Video Source Implementation in SnapSecure Package Detection System

Webcam Integration

The SnapSecure system employs a threaded approach to webcam integration, ensuring efficient frame capture while minimizing resource utilization. The implementation uses a custom *ThreadedVideoStream* class that encapsulates OpenCV's VideoCapture functionality within a dedicated thread, preventing video processing from blocking the main application thread.

```
class ThreadedVideoStream:
    def __init__(self, src=1, width=640, height=480):
        # Use a safer approach to initialize the camera
        self.src = src
        self.width = width
        self.height = height
        self.stream = None
        self.frame = None
```

```

self.grabbed = False
self.stopped = False

def start(self):
    # Initialize the camera here instead of in __init__
    try:
        self.stream = cv2.VideoCapture(self.src)
        if self.stream.isOpened():
            self.stream.set(cv2.CAP_PROP_FRAME_WIDTH, self.width)
            self.stream.set(cv2.CAP_PROP_FRAME_HEIGHT, self.height)
            (self.grabbed, self.frame) = self.stream.read()
            if self.grabbed:
                # Only start the thread if we successfully grabbed a frame
                threading.Thread(target=self.update, daemon=True).start()
                return self
            return None
    except Exception as e:
        print(f"Error starting threaded video stream: {e}")
        return None

def update(self):
    while not self.stopped:
        try:
            (self.grabbed, new_frame) = self.stream.read()
            if self.grabbed and new_frame is not None:
                self.frame = new_frame
            else:
                # If we can't grab a frame, sleep to avoid high CPU usage
                time.sleep(0.01)
        except Exception as e:
            print(f"Error reading frame: {e}")
            time.sleep(0.1) # Sleep longer on error

```

Figure 4.47 ThreadedVideoStream Class

The webcam implementation relies on a threaded approach to ensure smooth, real-time video processing without frame drops or lag. According to Figure 4.47 The core component is the ThreadedVideoStream class, which encapsulates the functionality needed to access and process webcam feeds efficiently. During initialization, the class takes parameters for camera index, width, and height, but importantly delays actual camera access until explicitly started. The start () method initializes the camera connection using OpenCV's VideoCapture object, configures resolution settings via cv2.CAP_PROP_FRAME_WIDTH and cv2.CAP_PROP_FRAME_HEIGHT, and launches a dedicated thread for continuous frame

reading. This thread, defined in the `update()` method, runs in a loop that constantly fetches new frames from the camera while handling potential errors gracefully

Video Upload Implementation

```
@app.route('/upload_video', methods=['POST'])
def upload_video():
    if 'user_id' not in session:
        return jsonify({"success": False, "message": "Please log in to upload
a video."}), 401

    if 'videoFile' not in request.files:
        return jsonify({"success": False, "message": "No video file
selected."}), 400

    file = request.files['videoFile']
    if file.filename == '':
        return jsonify({"success": False, "message": "No video file
selected."}), 400

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)

        # Save the file path in the database for the logged-in user
        user_id = session.get('user_id')
        db["User"].update_one(
            {"_id": ObjectId(user_id)},
            {"$set": {"video_source": "upload", "uploaded_video_path":
file_path}}
        )

        # Reinitialize video stream with the uploaded file
        initialize_video_stream()

        return jsonify({"success": True, "message": "Video uploaded
successfully."}), 200
    else:
        return jsonify({"success": False, "message": "Invalid file format.
Please upload a video file."}), 400

def allowed_file(filename):
    return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

Figure 4.48 Upload_video route

The video upload implementation provides functionality (Figure 4.48) for users to analyze pre-recorded video content. The front-end interface consists of a file input field and an upload button, which triggers a form submission to the /upload_video endpoint. On the server side, the Flask route first validates the request by checking user authentication and confirming the presence of a video file. The function then uses Flask's secure_filename() utility to sanitize the filename before saving it to the configured upload directory. The video file path is stored in the user's database record alongside a setting that identifies "upload" as the current video source type.

IP Camera/Stream Implementation

```
class MjpegStreamReader:
    def __init__(self, url):
        self.url = url
        self.stream = None
        self.bytes_buffer = BytesIO()
        self.frame = None
        self.stopped = False
        self.thread = None
        self.is_valid_stream = False
        self.frame_width = 640 # Default width
        self.frame_height = 480 # Default height
        self.fps = 30 # Default FPS for MJPEG streams

    def start(self):
        try:
            # Try to connect to the stream
            response = requests.get(self.url, stream=True, timeout=10)
            if response.status_code != 200:
                print("Error: Unable to connect to stream. Status code:",
response.status_code)
                return None

            content_type = response.headers.get('Content-Type', '')
            if 'multipart/x-mixed-replace' not in content_type:
                print("Error: Stream doesn't appear to be MJPEG format.
Content-Type:", content_type)
                return None

            # If we get here, it seems to be a valid MJPEG stream
```

```

        self.is_valid_stream = True
        self.stopped = False
        self.thread = threading.Thread(target=self._process_stream,
args=(response,), daemon=True)
        self.thread.start()
        return self

    except Exception as e:
        print(f"Error connecting to MJPEG stream: {e}")
        return None

    def _process_stream(self, response):
        # Find the boundary string in the headers
        content_type = response.headers.get('Content-Type', '')
        boundary = content_type.split('boundary=')[1] if 'boundary=' in
content_type else None

        if boundary is None:
            print("Error: Could not find boundary in MJPEG stream")
            self.stopped = True
            return

        boundary = f'--{boundary}'.encode()

        # Process the multipart stream
        buffer = b''
        for chunk in response.iter_content(chunk_size=1024):
            if self.stopped:
                break

            buffer += chunk

        # Find the JPEG image start and end in the buffer
        start_marker = buffer.find(b'\xff\xd8') # JPEG start marker
        end_marker = buffer.find(b'\xff\xd9') # JPEG end marker

        if start_marker != -1 and end_marker != -1 and end_marker >
start_marker:
            jpeg_data = buffer[start_marker:end_marker+2]

            try:
                # Decode JPEG data to an OpenCV image
                self.frame = cv2.imdecode(np.frombuffer(jpeg_data,
dtype=np.uint8), cv2.IMREAD_COLOR)

                # Update dimensions if we have a valid frame
                if self.frame is not None:

```

```

        self.frame_height, self.frame_width =
self.frame.shape[:2]
    except Exception as e:
        print(f"Error decoding JPEG frame: {e}")

    # Clear processed data from buffer
    buffer = buffer[end_marker+2:]

def read(self):
    return self.frame

def release(self):
    self.stopped = True
    if self.thread and self.thread.is_alive():
        self.thread.join(timeout=1.0)

# Mimics cv2.VideoCapture's get() method for compatibility
def get(self, prop_id):
    if prop_id == cv2.CAP_PROP_FPS:
        return self.fps
    elif prop_id == cv2.CAP_PROP_FRAME_WIDTH:
        return self.frame_width
    elif prop_id == cv2.CAP_PROP_FRAME_HEIGHT:
        return self.frame_height
    elif prop_id == cv2.CAP_PROP_POS_FRAMES:
        return 0 # MJPEG streams don't have frame position
    elif prop_id == cv2.CAP_PROP_FRAME_COUNT:
        return -1 # MJPEG streams don't have a frame count (infinite)
    return 0 # Default for unsupported properties

def optimize_rtsp_connection(rtsp_url):
    """Optimizes OpenCV VideoCapture settings for RTSP streams to reduce
    latency"""
    try:
        # Configure VideoCapture with optimized RTSP settings
        stream = cv2.VideoCapture(rtsp_url, cv2.CAP_FFMPEG)

        # Set buffer size to minimum to reduce latency
        stream.set(cv2.CAP_PROP_BUFFERSIZE, 1)

        # Use TCP for more reliable streaming (Less frame drops)
        stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('H', '2', '6',
'4'))

        return stream
    except Exception as e:
        print(f"Error optimizing RTSP connection: {e}")
        # Fall back to default settings

```

```
return cv2.VideoCapture(rtsp_url)
```

Figure 4.49 MjpegStreamReader Class

The IP camera and video stream implementation is the most complex, as it must handle a variety of streaming protocols and network conditions. The system implements a specialized *MjpegStreamReader* class (Figure 4.49) that maintains compatibility with OpenCV's VideoCapture interface while optimizing for network-based video feeds. This class manages connection initialization, buffered reading, and error recovery for MJPEG streams specifically. For RTSP streams, the system employs an `optimize_rtsp_connection()` function that configures appropriate transport protocols and buffering parameters to minimize latency.

4.6 Proposed System Implementation

This section would provide illustrations of the proposed system “SnapSecure” through the utilization of screenshots that would depict its implementations for the end users.

4.6.1 Proposed Web Application (SnapSecure)

This section would focus on the functionalities offered by the web application which begins with registration of first-time users to create an account for the web application, upon completion, the user will then be prompted to login the newly created account. After successfully logging into the account the user will be directed to the dashboard which would provide an overview of the system by displaying the live feed of the detection, active detections, brief history of detections, and a navigation panel on the left side of the screen. Hence, the user may now navigate to their profile to view and edit their personal information.

This app focuses on real-time monitoring of package theft which will notify the user through email if there is any package removal, and require the user to take action on the detections made to verify if it a safe removal or a theft removal. This detection results are then logged into 2 parts of the system, knowingly as the archive and report page where respectively are responsible for storing the detection images, confidence level, timestamp and detection id,

also reporting the overview of all the detections, displaying the numerical values in graphical, table and summation formats.

To add more robustness, the users may also filter the detections on each page respectively by date and also the types of detections. For customizability, the user may access the settings page to carry out two different customizations which are to edit the Region of Interest (ROI) of the system, to draw and edit the regions on where the system should focus on detections, while the next function is to choose the video source from either camera, upload video file and Streaming.

4.6.1.1 Creating Account and Logging in

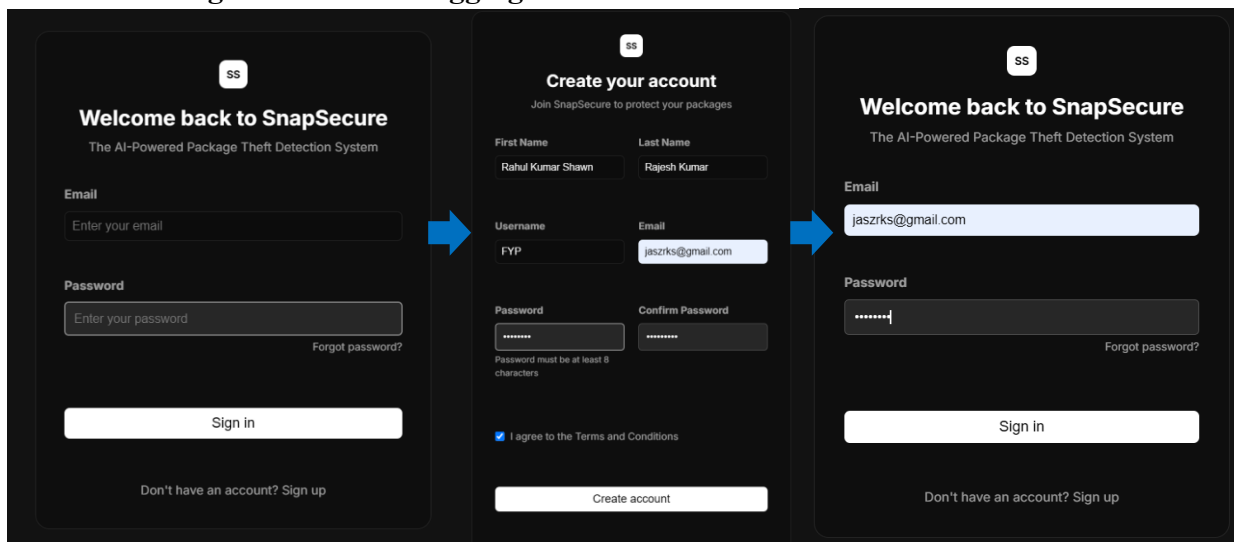


Figure 4.50 Creating account and logging in process

Figure 4.50 illustrates the authentication process for the SnapSecure system, showcasing the login screen, account creation, and successful login stages. This section details the steps required to access the SnapSecure AI-Powered Package Theft Detection System.

1. Accessing the Login Screen

- Navigate to the SnapSecure application.
- The login screen displays the system name, "SnapSecure," along with the tagline "The AI-Powered Package Theft Detection System."

2. Creating a New Account

- If users don't have an account, they can click the "Sign up" link at the bottom of the login page.
- The account creation screen presents fields for:
 - First Name and Last Name
 - Username selection
 - Email address
 - Password (must be at least 8 characters)
 - Confirm Password
- Users must agree to the Terms and Conditions by checking the provided box.
- After completing all fields, users click the "Create account" button to register.

3. Entering Login Credentials

- On the login screen, users enter:
 - Email: The email address used during registration
 - Password: The secure password created during account setup
- Both fields must be filled correctly to proceed.
- If needed, users can click "Forgot password?" to reset their credentials.

4. Handling Authentication

- After entering credentials, users click the "Sign in" button.
- The system validates the provided information against the database.
- If the credentials are incorrect, an error message will prompt users to retry.

5. Successful Login

- Upon successful authentication, users gain access to the SnapSecure dashboard.
- The system maintains the user's session for seamless interaction with the package theft detection features.

- Users can monitor their property and receive alerts about potential package theft incidents.

By following these steps, users can create accounts and securely access the SnapSecure system to protect their delivered packages from theft using AI-powered detection technology.

4.6.1.2 Dashboard

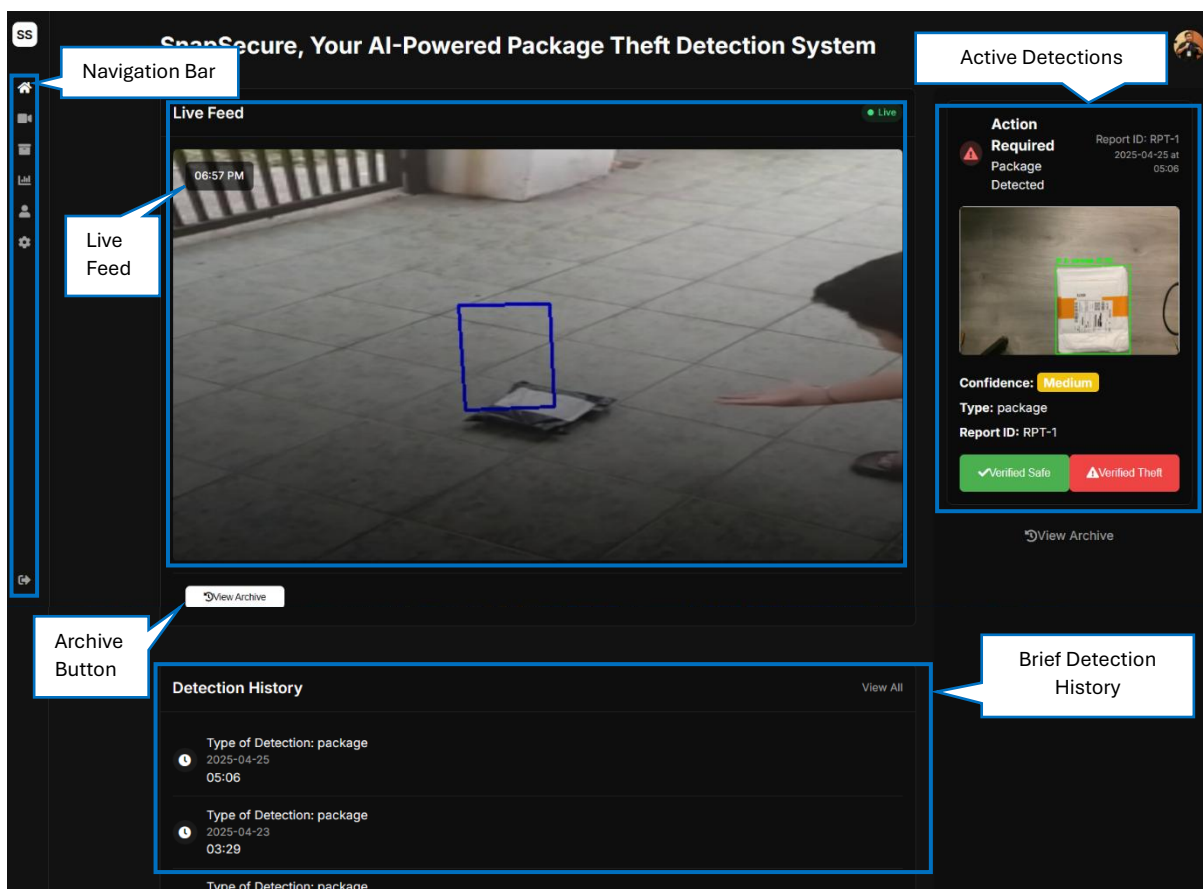


Figure 4.51 Dashboard screen of the proposed system

Based on Figure 4.51, it illustrates the main dashboard interface of the SnapSecure AI-Powered Package Theft Detection System, showing the various components that enable users to monitor package deliveries in real-time and respond to potential theft incidents.

1. Navigation Bar

- Located on the left side of the interface, the navigation bar provides quick access to the main system functions.

- Icon-based menu provides intuitive navigation to different sections of the application.
- Visual indicators show the currently active section of the application.

2. Live Feed Display

- Positioned in the centre of the dashboard, the live feed shows real-time video from the connected camera.
- A blue ROI (Region of Interest) box highlights areas where package detection is actively monitored.
- The "Live" indicator confirms that the feed is currently streaming in real-time.
- This area allows users to visually monitor their property for deliveries and potential theft incidents.

3. Active Detections Panel

- Located on the right side, this panel alerts users to events requiring immediate attention.
- The "Action Required" header clearly indicates when user intervention is needed.
- Visual representation of detected packages with timestamp information.
- Key detection details are displayed:
 - Confidence level (shown as "Medium" in this example)
 - Type of detection (package)
 - Report ID for reference (RPT-1)
- Two action buttons allow users to verify the status:
 - "Verified Safe" (green) to confirm the package is secure
 - "Verified Theft" (red) to report a theft incident

4. Archive Access

- The "View Archive" button provides access to historical footage and past detection events.

- This allows users to review previous incidents at any time by redirecting users to the Archive page.

5. Detection History Section

- Located at the bottom of the dashboard, this section shows recent detection events.
- Each entry displays:
 - Type of detection (package)
 - Date of detection (e.g., 2023-04-25)
 - Time of detection (e.g., 06:06)
- The "View All" option allows users to access the complete detection history by directing the user to the Report page.

4.6.1.2 Archive Page

4.6.1.2.1 Filtering Detections

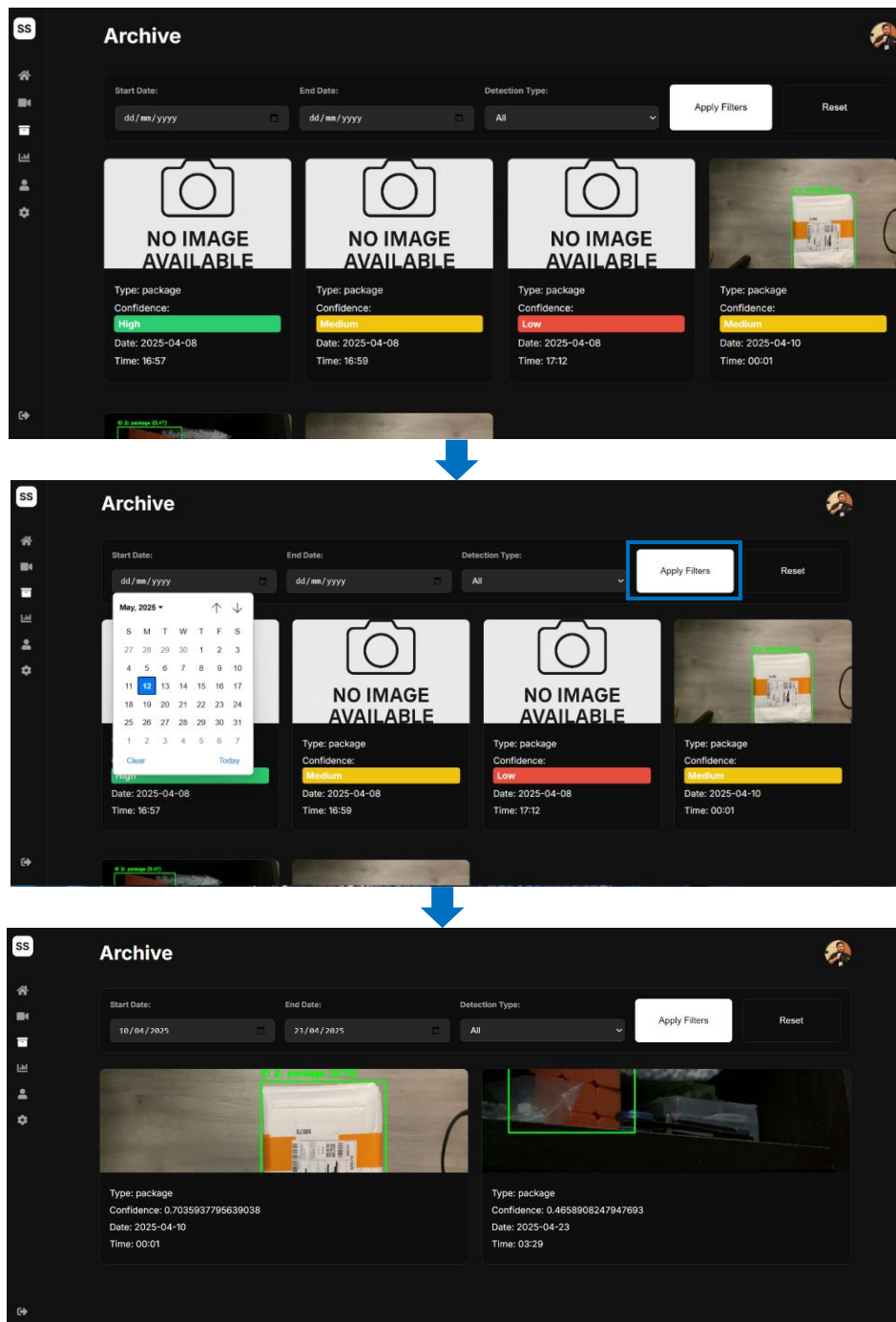


Figure 4.52 Filtering dates of detections on Archive page

Based on Figure 4.52, it illustrates the date-based filtering capabilities of the SnapSecure Archive page, allowing users to access and review historical detection events based on specific timeframes. This section explains how users can efficiently filter through their package detection history.

1. Archive Page Overview

- The Archive page provides a comprehensive view of all historical package detections captured by the system.
- Located at the top of the interface are filtering controls that allow users to refine the displayed results.
- The navigation sidebar on the left provides access to other system functions while maintaining consistent access to archived data.
- Each detection is presented as a card containing the detection image and key metadata about the event.

2. Available Filtering Options

- **Start Date:** Allows users to specify the beginning date of the desired timeframe.
- **End Date:** Allows users to specify the final date of the desired timeframe.
- **Detection Type:** Enables filtering by the type of detection (defaulted to "All").
- Two action buttons are provided:
 - **Apply Filters:** Processes the selected filter criteria and updates the display.
 - **Reset:** Clears all filter selections, returning to the complete archive view.

3. Date Selection Process

- When clicking on either date field, an interactive calendar interface appears.
- Users can navigate between months and years using the calendar controls.
- The current date is highlighted for reference.
- Selecting a specific date automatically populates the corresponding date field.
- The calendar interface includes shortcut options such as "Today" for quick date selection.

4. Viewing Filtered Results

- After applying date filters, the system displays only detection events that occurred within the specified timeframe.
- Each detection card shows:
 - Detection image (or placeholder if no image is available)
 - Type of detection (package)
 - Confidence level with color-coding (High: green, Medium: yellow, Low: red)
 - Date of detection
 - Time of detection
 - Precise confidence score value

5. Interpreting Detection Cards

- Package images show the actual detected items with bounding boxes highlighting the detected objects.
- Color-coded confidence indicators provide quick visual assessment of the detection accuracy.
- Timestamp information helps users track when specific packages were detected.
- Cards are arranged chronologically, making it easy to follow the sequence of events.

4.6.1.2.2 Viewing Specific Detections

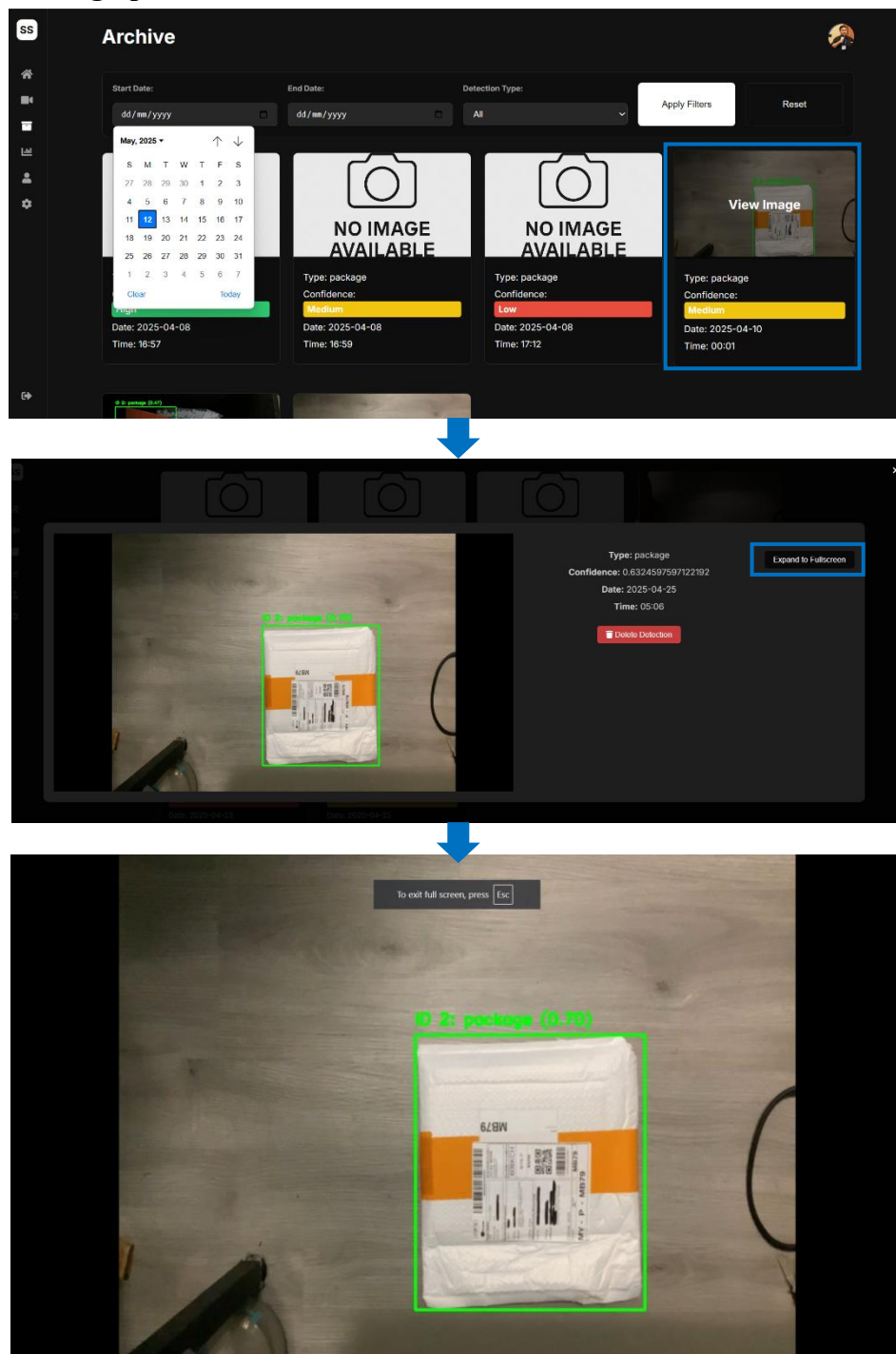


Figure 4.53 Viewing Specific Detection Details

Based on Figure 4.53, it illustrates the process of viewing specific detection details within the SnapSecure archive system. This feature allows users to access comprehensive information about each package detection event, including high-resolution imagery and detailed metadata.

1. Archive Thumbnail View

- The top section displays the archive grid with detection thumbnails, each showing:
 - Detection image (or a placeholder if no image is available)
 - Type of detection (package)
 - Confidence level with color-coded indicators (green for high, yellow for medium, red for low)
 - Date and time of the detection event
- Users can filter detections using the date range selectors and detection type dropdown
- The date selector incorporates a calendar interface for intuitive date selection
- Hovering over a detection shows the "View Image" overlay, indicating the item can be clicked for details

2. Detection Modal View

- When a user clicks on a detection thumbnail, a detailed modal window appears
- The modal displays:
 - A larger version of the detection image with the package highlighted by a green bounding box
 - Detection metadata in the right panel:
 - Type of detection (package)
 - Precise confidence score (e.g., 0.633459759732192)
 - Date of detection (e.g., 2023-04-23)
 - Time of detection (e.g., 05:06)
 - Action buttons:
 - "Expand to Fullscreen" button for enhanced viewing
 - "Delete Detection" button (red) to remove the detection from the system

3. Fullscreen View

- Clicking "Expand to Fullscreen" transitions to a full-screen view of the detection

- The fullscreen view shows:
 - The maximum available resolution of the detection image
 - The green bounding box highlighting the detected package
 - Detection confidence score and object class overlaid on the image
 - An exit instruction ("To exit full screen, press Esc") for user guidance

This three-tier viewing system provides users with progressively more detailed examination capabilities, from browsing multiple detections in the grid view to analyzing individual detections in the modal, to examining detection details at maximum resolution in fullscreen mode. Users can efficiently review their package detection history and make informed decisions about each event.

4.6.1.3 Report Page

4.6.1.3.1 Graph

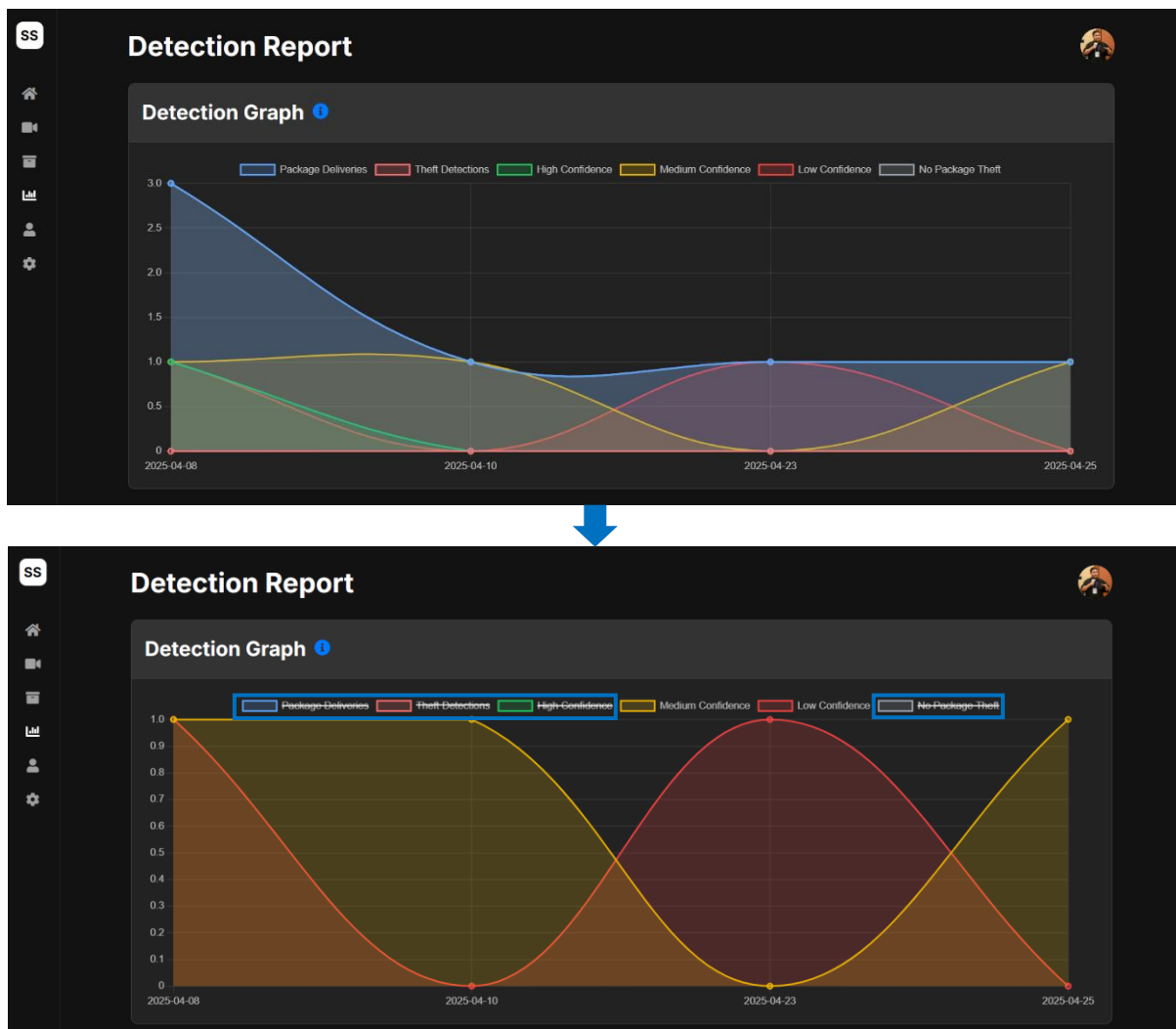


Figure 4.54 Filtering Graph

Figure 4.54 illustrates the interactive filtering capabilities of the SnapSecure Detection Report page, allowing users to customize the visualization of package detection data by toggling specific metrics on or off through the graph legend.

1. Detection Graph Overview

- The Detection Report page presents a comprehensive graphical representation of all detection events over time.
- The graph displays multiple metrics using different colored lines:
 - Blue: Package Deliveries
 - Red: Theft Detections

- Green: High Confidence Detections
- Yellow: Medium Confidence Detections
- Red: Low Confidence Detections
- Gray: No Package Theft
- The x-axis represents dates (from 2023-04-08 to 2023-04-25)
- The y-axis shows the frequency or count of detection events
- An information button ((i)) provides additional context about the graph's filtering features

2. Interactive Legend Filtering

- The legend at the top of the graph serves as interactive toggle controls.
- Users can click on any legend item to exclude or include that specific data series from the visualization.
- In the example shown:
 - The user has clicked on several legend items (highlighted with blue borders)
 - Selected items are removed from the visualization
 - This filtering allows for clearer focus on the remaining data series

3. Visual Feedback

- When legend items are clicked, they are animated to be strikethrough, indicating it has been filtered.
- The graph dynamically updates to show only the non-filtered data series.
- The scaling of the y-axis automatically adjusts to properly display the remaining data.
- This allows users to compare specific metrics without visual clutter from other data series.

4.6.1.3.2 Statistical Overview

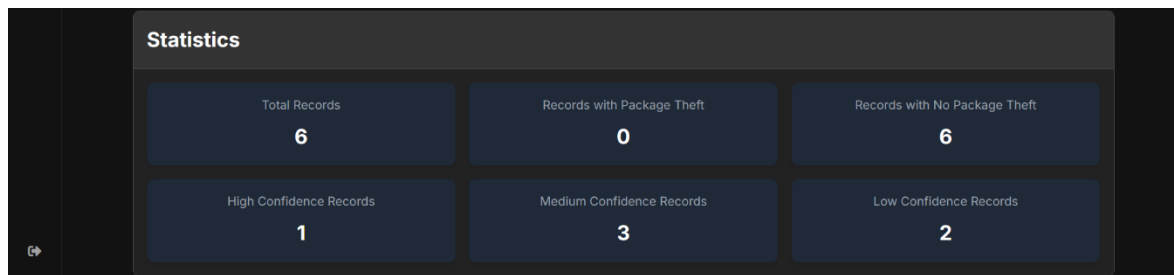


Figure 4.55 Statistical Overview

Figure 4.55 displays the Statistics section of the SnapSecure Detection Report page, providing users with a comprehensive numerical summary of all package detection events recorded by the system. This statistical overview offers quick insights into detection patterns and system performance.

1. Detection Volume Metrics

- **Total Records:** Displays the aggregate number of detection events (6) captured by the system since installation or within the selected reporting period.
- **Records with Package Theft:** Shows the number of confirmed package theft incidents (0), indicating no packages have been stolen during the monitoring period.
- **Records with No Package Theft:** Presents the count of detection events where packages were delivered but not stolen (6), representing successful delivery monitoring.

2. Detection Quality Metrics

- **High Confidence Records:** Indicates the number of detections (1) where the AI system had high certainty (>75%) that a package was present.
- **Medium Confidence Records:** Shows the number of detections (3) with moderate certainty (typically 50-74%) of package presence.
- **Low Confidence Records:** Displays the number of detections (2) where the system had lower certainty (typically <50%) but still registered a potential package.

4.6.1.3.3 Detection Table



The figure consists of two screenshots of a 'Detection Table' interface. The top screenshot shows the full table with 7 rows of data. The bottom screenshot shows the same interface after filtering by 'Medium' confidence, resulting in 3 rows of data. In the bottom screenshot, the search bar, the 'Medium confidence' filter button, and the 'Detection ID' column header are highlighted with blue boxes.

Detection ID ▲	Date ▲	Time	Type	Confidence	Status	Verification Status
6	2025-04-08	16:57	package	High	None	Unverified
7	2025-04-08	16:59	package	Medium	None	Unverified
8	2025-04-08	17:12	package	Low	None	Unverified
3	2025-04-10	00:01	package	Medium	None	Unverified
2	2025-04-23	03:29	package	Low	Taken	Safe
1	2025-04-25	05:06	package	Medium	Taken	Unverified

Detection ID ▼	Date ▲	Time	Type	Confidence	Status	Verification Status
1	2025-04-25	05:06	package	Medium	Taken	Unverified
3	2025-04-10	00:01	package	Medium	None	Unverified
7	2025-04-08	16:59	package	Medium	None	Unverified

Figure 4.56 Detection Table Filtering, Sorting and Search Feature

Figure 4.56 illustrates the Detection Table feature of the SnapSecure reporting system, showcasing its comprehensive search, filtering, and sorting capabilities that enable users to efficiently analyze and manage detection records.

1. Detection Table Overview

- The Detection Table presents a structured view of all package detection events in a tabular format.
- Each row represents a distinct detection event with its associated metadata.
- The table features a dark-themed design consistent with the overall SnapSecure interface.
- Column headers clearly identify the type of information displayed in each column.

- Alternating row shading enhances readability of the tabular data.

2. Advanced Search Functionality

- A prominent search bar at the top allows users to search across all detection records.
- As demonstrated in the bottom image, entering "medium" filters the table to show only records containing that term.
- The search is dynamic, updating results as users type their query.
- Search results highlight matches in the Medium confidence category, showing only the relevant records.
- This enables quick location of specific detection events without manual scanning.

3. Multi-criteria Filtering System

- A series of interactive filter pills appears below the search bar for rapid categorical filtering:
 - "All" - displays all detection records
 - "Package theft" - shows only confirmed theft incidents
 - "No package theft" - displays only safe package deliveries
 - "High confidence" - filters for high-confidence detections
 - "Medium confidence" - shows medium-confidence detections
 - "Low confidence" - displays low-confidence detections
- Users can select a filter pill to instantly narrow down the table to the relevant subset of data.
- The selected filter is highlighted with a blue border, as seen with "Medium confidence" in the bottom image.

4. Dynamic Sorting Capabilities

- Column headers with arrow indicators (▲/▼) support interactive sorting:

- "Detection ID" column can be sorted to organize records by their unique identifiers
- "Date" column enables chronological or reverse-chronological ordering
- Clicking on a sortable header toggles between ascending and descending order.
- In the bottom image, the records appear sorted by Detection ID in descending order (newest first).
- This sorting capability allows users to analyze detection patterns over time or focus on specific time periods.

5. Comprehensive Detection Information

- The table presents detailed information for each detection:
 - **Detection ID:** Unique identifier for each detection event
 - **Date:** Calendar date when the detection occurred
 - **Time:** Specific time of detection
 - **Type:** Category of object detected (package)
 - **Confidence:** Confidence level with color-coding (green for high, yellow for medium, red for low)
 - **Status:** Current Status of the package (None/Taken)
 - **Verification Status:** User verification status (Unverified/Safe)

PDF Generator

PDF Generator

Start Date:

23/04/2025

End Date:

23/05/2025

Generate PDF Report

Reset Dates (All Time)

File name:

SnapSecure_Report_2025-04-23_to_2025-05-23_20250523_2256.pdf

Save as type:

Adobe Acrobat Document (*.pdf)

SnapSecure Detection Report

Report Generated: 2025-05-23 22:56

Detection Statistics

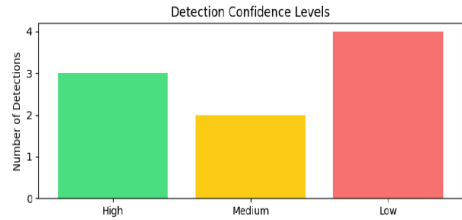
Statistic	Value
Total Records	9
Records with Package Theft	1
Records with No Package Theft	8
High Confidence Records	3
Medium Confidence Records	2
Low Confidence Records	4

Detection Records

ID	Date	Time	Type	Confidence	Verification
1	2025-05-23	22:54	Package	High	Safe
3	2025-05-22	14:21	Box	High	Unverified
12	2025-05-14	22:06	Unknown	Low	Safe
9	2025-05-14	22:02	package	Medium	Unverified
13	2025-05-14	22:02	label	Low	Theft
8	2025-05-14	21:59	package	Medium	Unverified
5	2025-05-14	21:59	package	Low	Unverified
7	2025-05-14	21:58	Unknown	Low	Unverified
4	2025-05-14	21:58	package	High	Unverified

*Confidence Level Legend:
High: Over 75% confidence
Medium: 50-75% confidence
Low: Below 50% confidence

Detection Confidence Chart



© 2024 SnapSecure, Inc - Generated from your SnapSecure account

Figure 4.57 PDF Generator Workflow

Figure 4.57 illustrates the PDF Generator workflow in the SnapSecure platform, showcasing how users can create comprehensive detection reports for record-keeping, analysis, and sharing. This feature enables the transformation of detection data into professionally formatted documents with both tabular and graphical representations.

1. Report Configuration Interface

- Located in the Reports page, the PDF Generator module provides a dedicated card with intuitive date selection controls.
- Two date picker inputs allow users to define a specific reporting period:
 - **Start Date:** Sets the beginning of the time period for included detections (shown set to 23/04/2025)
 - **End Date:** Sets the end of the time period for included detections (shown set to 23/05/2025)
- Action buttons provide clear options:
 - Blue **Generate PDF Report** button initiates the report creation process
 - Gray **Reset Dates (All Time)** button clears date constraints to include all historical data

2. File Saving Process

- Upon clicking "Generate PDF Report," the system processes the selected date range and prepares the document.
- A standard file save dialog appears, allowing users to:
 - Choose the save location on their local device
 - Modify the automatically generated filename if desired
 - View the default filename format: "SnapSecure_Report_[start-date]to[end-date]_[timestamp].pdf"
- The document is saved as a standard PDF file compatible with all PDF readers

3. Report Content Structure

- The generated PDF opens with a clear header "SnapSecure Detection Report" and includes a timestamp indicating when the report was created.
- The report is organized into multiple sections for comprehensive analysis:

Detection Statistics Table

- Provides a summary overview with key metrics in a structured table:
 - Total detection records within the selected period
 - Records categorized by theft status (Package Theft vs. No Package Theft)
 - Records categorized by confidence level (High, Medium, Low)
- This section enables quick assessment of detection patterns and system performance

Detailed Detection Records

- A comprehensive table listing all detection events within the selected timeframe
- Each row represents a single detection with columns for:
 - ID: Unique detection identifier
 - Date and Time: When the detection occurred
 - Type: Classification of the detected object (package, box, etc.)
 - Confidence: Reliability level of the detection
 - Status: Current verification status (Safe, Theft, Unverified)
- Color-coding enhances readability for quick status identification

Confidence Level Legend

- Provides clear definitions of the confidence thresholds:
 - High: Detections with over 75% confidence
 - Medium: Detections with 50-75% confidence

- Low: Detections with below 50% confidence
- Helps users interpret the reliability of detection events

Detection Confidence Chart

- A bar graph visualization showing the distribution of detections by confidence level
- Color-coded bars (green for High, yellow for Medium, red for Low) provide immediate visual insights
- The y-axis represents the count of detections, while the x-axis shows confidence categories

4.6.1.4 Profile Page

4.6.1.4.1 Profile Editing

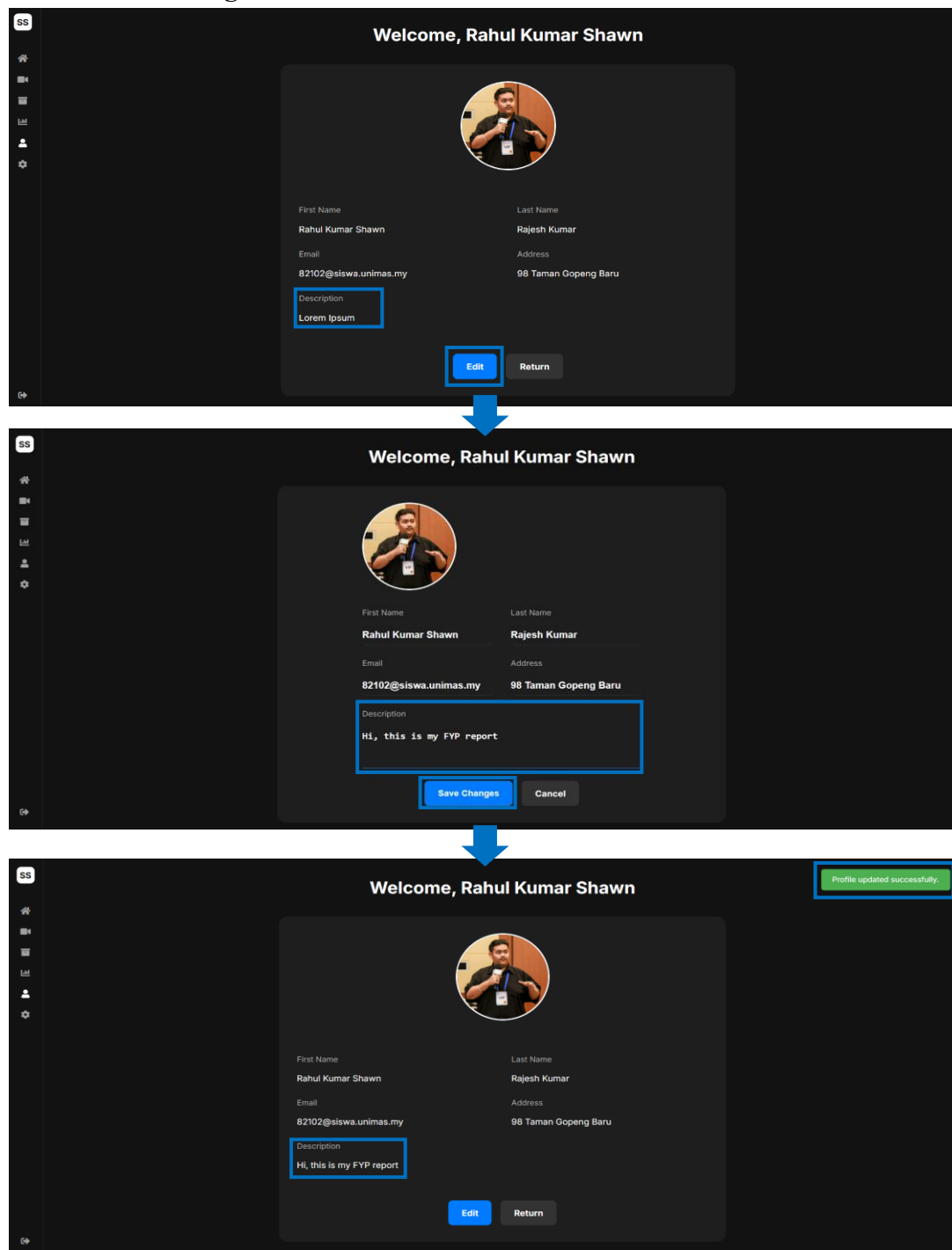


Figure 4.58 Editing Profile Details

Figure 4.58 illustrates the profile editing functionality of the SnapSecure application, demonstrating the process of updating user profile information and receiving confirmation

feedback. This section explains the complete workflow for modifying profile details within the system.

1. Profile View Mode

- The top section displays the user's profile in view mode, showing a centered header with "Welcome, Rahul Kumar Shawn" and the user's profile picture.
- Personal information is organized in a clean two-column layout:
 - Left column: First Name, Email, and Description
 - Right column: Last Name and Address
- The profile picture appears in a circular frame at the top of the card, providing visual identification.
- Navigation buttons at the bottom include "Edit" (highlighted in blue) and "Return" (in gray).
- The description field initially contains placeholder text "Lorem ipsum" that needs to be updated.

2. Profile Edit Mode

- Upon clicking the "Edit" button, the interface transitions to edit mode.
- The same information structure is maintained, but text fields become editable.
- The user can modify any field, with the demonstration focusing on changing the Description field.
- The description text is being updated to "Hi, this is my FYP report" in the active text area.
- Action buttons change to:
 - "Save Changes" - Primary action button (blue) to confirm edits
 - "Cancel" - Secondary action button (gray) to discard changes

- The editing interface maintains the same visual design as the view mode for consistency.

3. Update Confirmation

- After clicking "Save Changes," the system returns to view mode with updated information.
- The description field now displays the newly entered text: "Hi, this is my FYP report"
- A success notification appears in the top-right corner with the message "Profile updated successfully" on a green background.
- This confirmation toast provides immediate feedback that the changes were successfully saved to the system.
- The interface returns to the standard view mode with "Edit" and "Return" buttons.

The profile management system provides a straightforward process for users to view and modify their personal information. While the example focuses on updating the description field, the same workflow applies to editing any profile attribute, including name, email, and address details. The immediate success notification ensures users receive clear confirmation when their changes are successfully saved.

4.6.1.5 Settings Page

4.6.1.5.1 Settings Page Overview

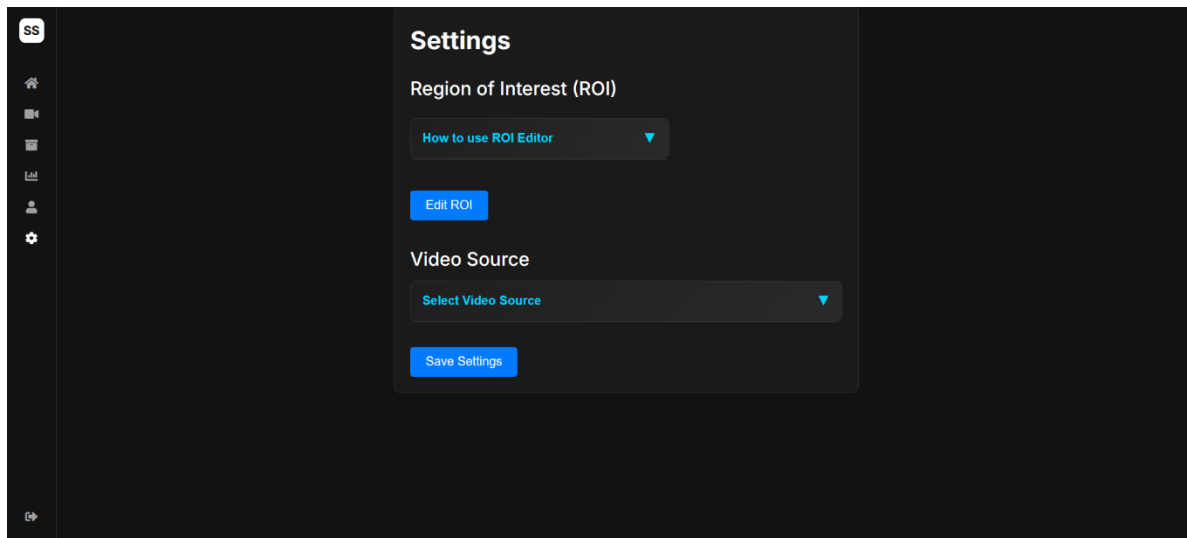


Figure 4.59 Settings Page

Figure 4.599 displays the Settings page of the SnapSecure system, providing users with configuration options for the AI-powered package detection functionality.

1. Region of Interest (ROI) Settings

- The primary heading "Settings" clearly identifies the page purpose.
- The "Region of Interest (ROI)" section allows users to configure specific areas within the camera view where package detection should occur.
- A collapsible dropdown labeled "How to use ROI Editor" (with a downward arrow) provides access to instructions for using this feature.
- The blue "Edit ROI" button opens the advanced ROI editor tool where users can draw polygons to define detection zones.

2. Video Source Configuration

- The "Video Source" section enables users to select their preferred video input method.
- A collapsible dropdown labeled "Select Video Source" allows users to choose between different options (webcam, IP camera, or uploaded video files).
- This flexibility accommodates various user setups and installation scenarios.

3. Settings Management

- The blue "Save Settings" button at the bottom of the page allows users to commit their configuration changes.
- The consistent dark theme with blue accent colors maintains the modern, professional aesthetic of the SnapSecure system.

4.6.1.5.2 Editing ROI

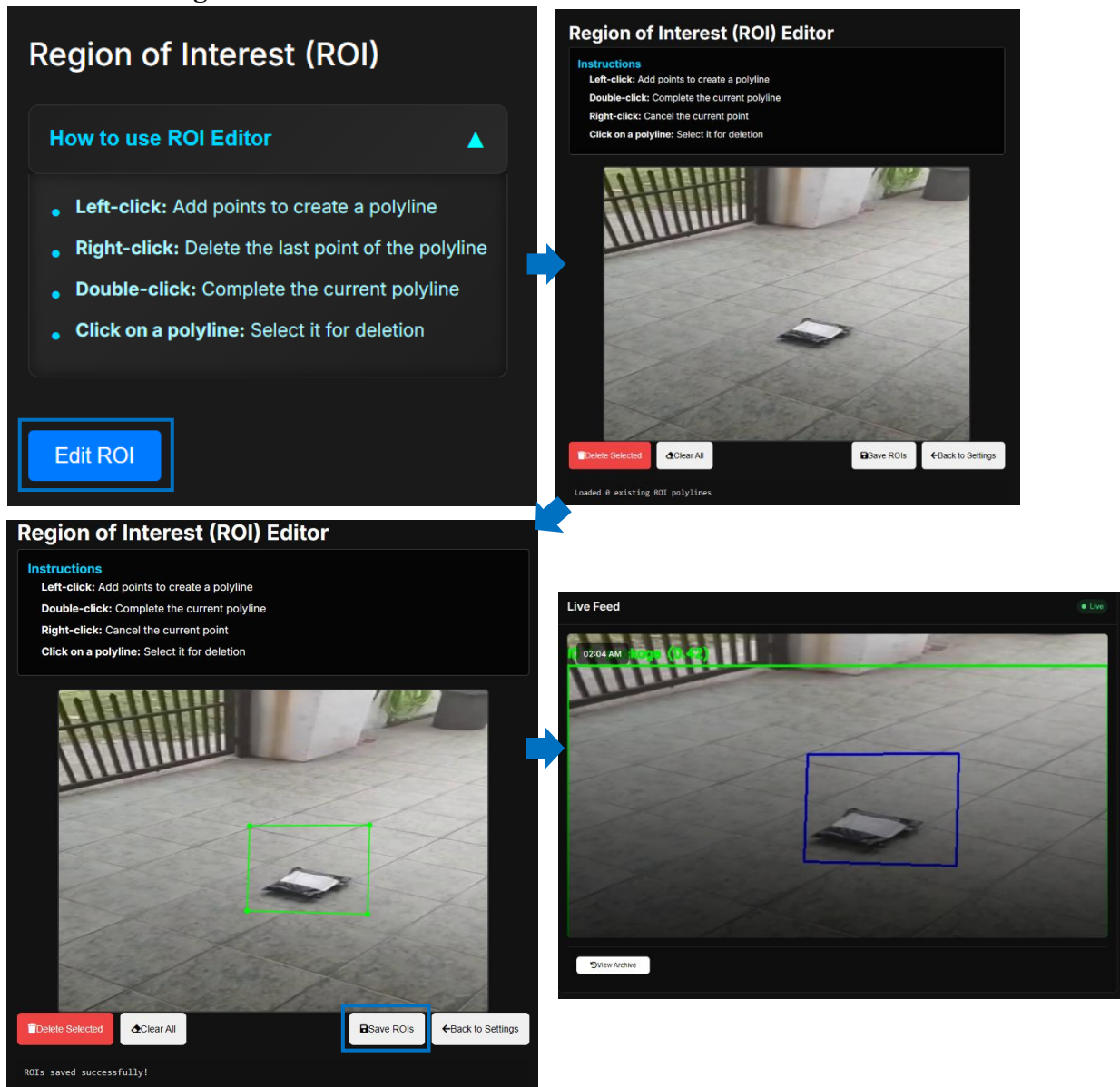


Figure 4.60 Steps to Edit ROI

Figure 4.6060 illustrates the ROI configuration workflow within the SnapSecure system, allowing users to define specific monitoring zones for more accurate package detection. This

functionality enables users to focus the AI detection capabilities on relevant areas while ignoring others.

1. Settings Page ROI Configuration

- The Settings page provides a dedicated "Region of Interest (ROI)" section where users can manage detection zones.
- A collapsible instruction panel explains the key interactions:
 - **Left-click:** Add points to create a polyline
 - **Right-click:** Delete the last point of the polyline
 - **Double-click:** Complete the current polyline
 - **Click on a polyline:** Select it for deletion
- The blue highlighted "Edit ROI" button provides access to the advanced ROI configuration interface.

2. ROI Editor Interface

- Upon clicking the "Edit ROI" button, users are directed to a dedicated editor page.
- The editor displays a current frame from the user's configured video source.
- The interface includes clear instructions at the top for creating and manipulating ROI zones.
- A canvas-based drawing area allows direct interaction with the video frame.
- Control buttons at the bottom provide functions for:
 - **Delete Selected:** Remove a selected ROI polygon
 - **Clear All:** Remove all defined ROI zones
 - **Save ROIs:** Store the configured zones to the system
 - **Back to Settings:** Return to the main settings page

3. Defining Detection Areas

- Users create polygons by clicking points on the image to define the boundaries of detection zones.
- Active polygons being drawn appear in red, while completed polygons appear in green.
- Multiple separate ROI zones can be defined to cover different areas (e.g., front porch, driveway).
- When a polygon is selected, it highlights in yellow for easy identification before deletion.
- A status bar at the bottom provides feedback on current actions and system status.

4. Applying ROI to Live Detection

- After saving the ROI configuration, the system immediately applies these boundaries to the live detection feed.
- The defined zones (shown as blue rectangles in the live feed) become the only areas where package detection is active.
- This targeted approach significantly reduces false positives by ignoring irrelevant areas of the camera view.
- The notification "ROIs saved successfully!" confirms when settings have been applied to the system.

4.6.1.5.3 Selecting Video Source

Camera

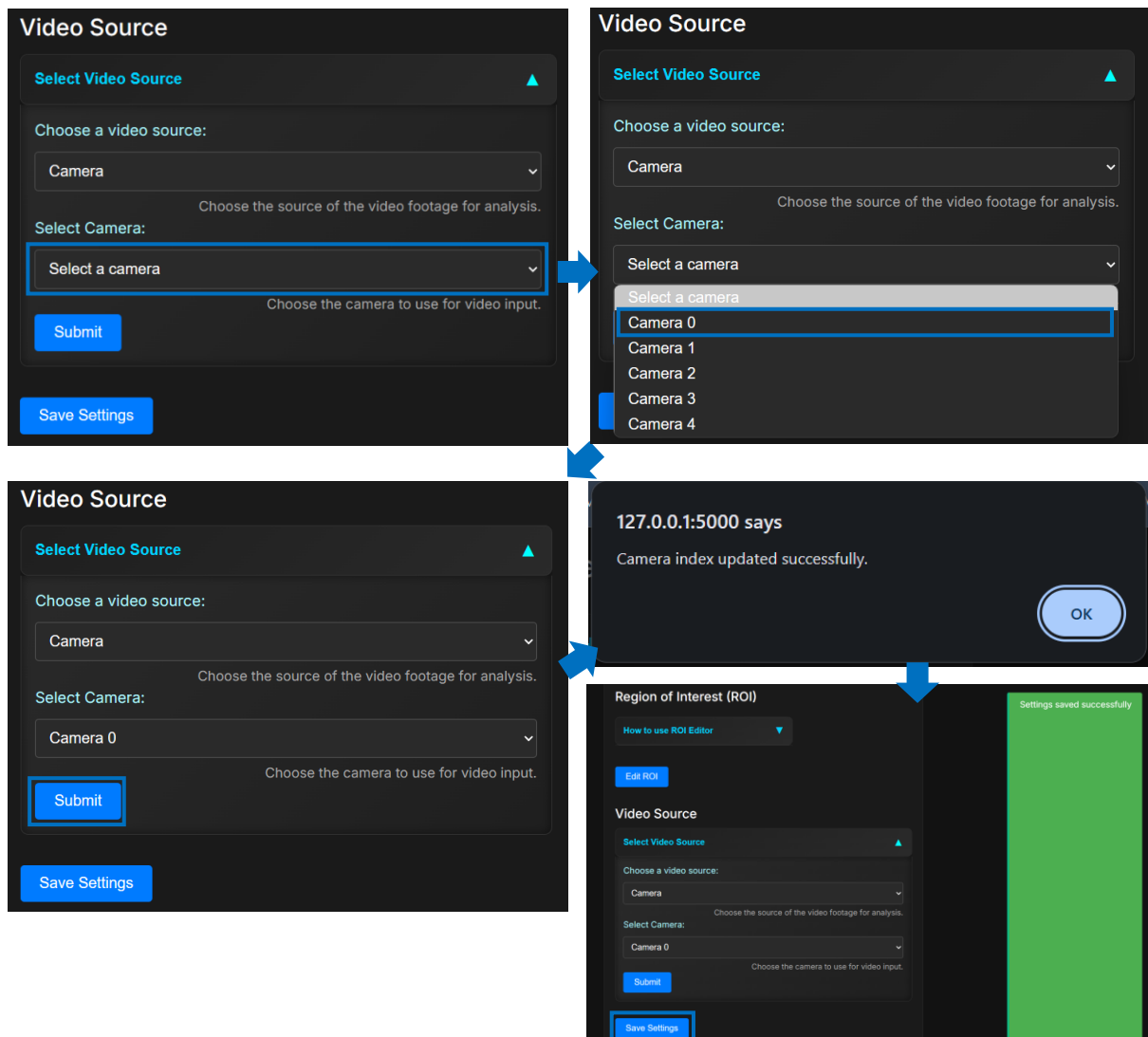


Figure 4.61 Selecting Camera as a Video Source

Figure 4.611 illustrates the process of configuring a camera as the video source for the SnapSecure package detection system. This sequence demonstrates how users can select and save specific camera settings to optimize their surveillance capabilities.

1. Video Source Selection Interface

- The Settings page provides a dedicated "Video Source" section where users can specify the input feed for package detection.
- A collapsible dropdown labeled "Select Video Source" allows users to choose between different source types.

- The "Camera" option is selected in the dropdown, indicating that footage will be sourced from a connected camera device.
- Helpful explanatory text "Choose the source of the video footage for analysis" provides context for this selection.
- Below the main source selection, a "Select Camera" dropdown appears, allowing users to specify which connected camera to use.

2. Camera Device Selection Process

- When clicking on the "Select Camera" dropdown, the system displays all available camera devices connected to the user's system.
- The expanded dropdown presents numbered options (Camera 0, Camera 1, Camera 2, etc.) representing different camera devices or inputs.
- Each camera entry represents either a physically connected webcam, an integrated camera, or a virtual camera device.
- Users can scroll through the list to find their preferred camera device.
- Selecting "Camera 0" typically indicates the default or primary camera on the system.

3. Submitting Camera Selection

- After choosing a specific camera from the dropdown list, the selected option is displayed in the field.
- The blue "Submit" button allows users to confirm their camera selection without changing other settings.
- When clicked, the system processes this specific change immediately.
- A confirmation dialog appears stating "Camera index updated successfully" to provide immediate feedback.
- The "OK" button in the dialog allows users to acknowledge the successful update.

4. Saving Overall Settings

- The "Save Settings" button at the bottom of the page allows users to commit all configuration changes.
- When clicked, the system applies all pending changes to the application's configuration.
- A green notification appears on the right side of the screen with the message "Settings saved successfully."
- This notification provides clear visual confirmation that all changes have been properly stored.
- After saving, the selected camera becomes the active video source for all package detection operations.

Video Upload

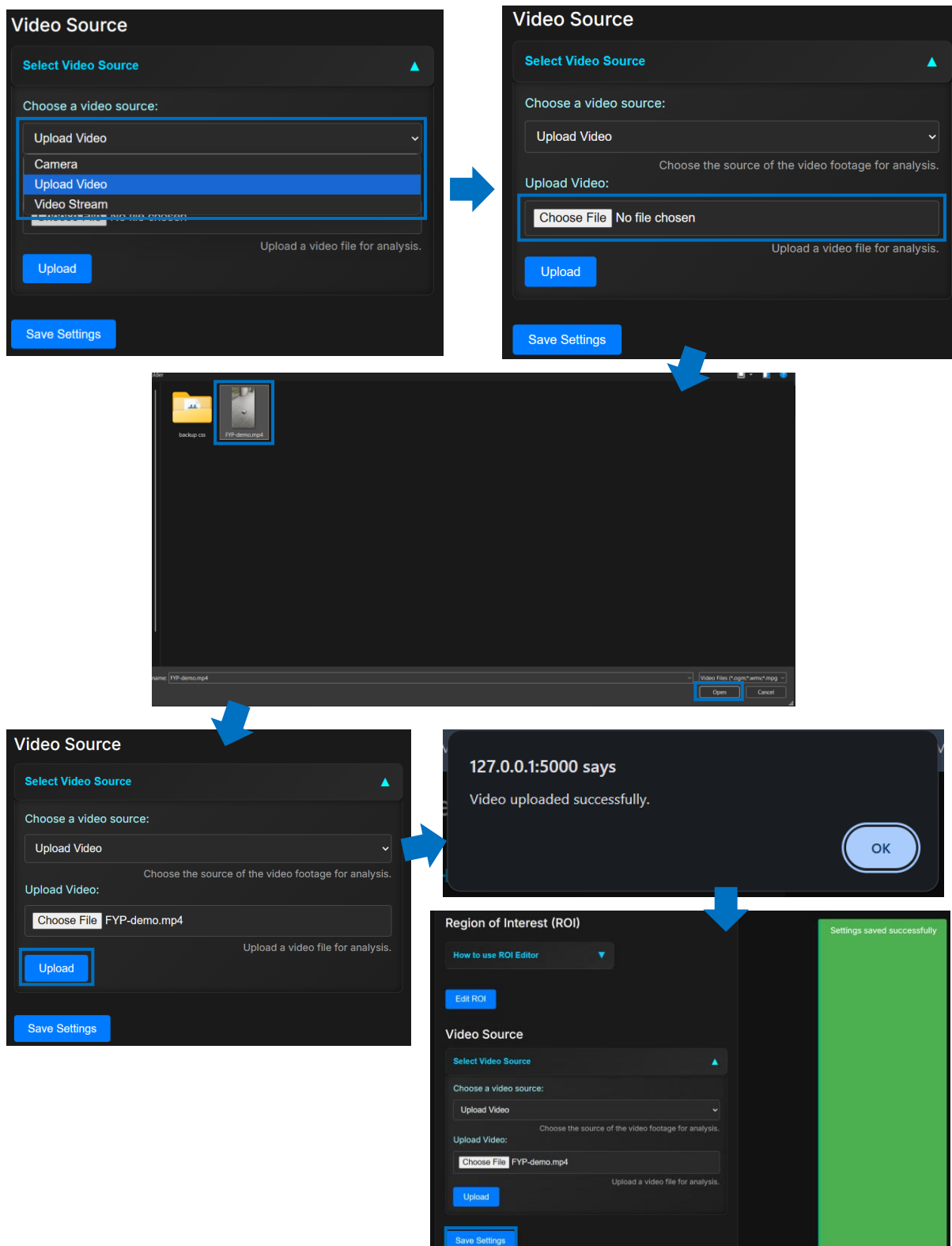


Figure 4.62 Selecting Video Upload as Video Source

Figure 4.622 illustrates the process of configuring a video upload as the video source for the SnapSecure package detection system. This sequence demonstrates how users can select, upload, and activate a pre-recorded video file for package detection analysis.

1. Video Upload Selection Interface

- "Upload Video" is selected from the dropdown menu, the interface dynamically updates to display the upload controls specific to this option.
- Helpful explanatory text "Choose the source of the video footage for analysis" provides context for this selection.
- The dropdown menu shows a blue highlight when active, indicating the current interaction area.

2. Video File Selection Process

- After selecting "Upload Video," the interface presents an "Upload Video:" label followed by a file selection component.
- The "Choose File" button allows users to access their local file system to select a video recording.
- Upon clicking this button, a standard file browser dialog opens, displaying video files available on the user's device.
- The system supports common video formats (MP4, AVI, MOV, MKV) for upload.
- In the third panel, we can see that the user has selected a specific file named "FYP-demo.mp4" from their local storage.

3. Upload Submission Process

- Once a file is selected, the filename appears in the interface to confirm the selection.
- The blue "Upload" button becomes the primary action button for initiating the file transfer.

- When clicked, the system begins uploading the selected video file to the SnapSecure server.
- During upload, the system processes the file to ensure compatibility with the detection system.
- The upload process stores the video in a secure location on the server for use by the detection algorithm.

4. Confirmation and Success Notifications

- After successful upload, a modal dialog appears with the message "Video uploaded successfully."
- The dialog shows the server address "127.0.0.1:5000" and provides an "OK" button to acknowledge the message.
- Upon clicking "OK" and returning to the settings page, an additional green success notification appears on the right side with "Settings saved successfully."
- These confirmations provide clear visual feedback that both the file upload and settings update were completed properly.

5. Settings Persistence

- The bottom panels show the final state of the settings page, with "Upload Video" remaining selected as the video source.
- The uploaded filename "FYP-demo.mp4" is displayed in the file selection field, indicating that this video is now configured as the active source.
- The "Save Settings" button at the bottom of the page allows users to commit all configuration changes globally.
- Once saved, the uploaded video immediately becomes available for the package detection system to analyze for potential package deliveries and theft events.

IP/RSTP Camera

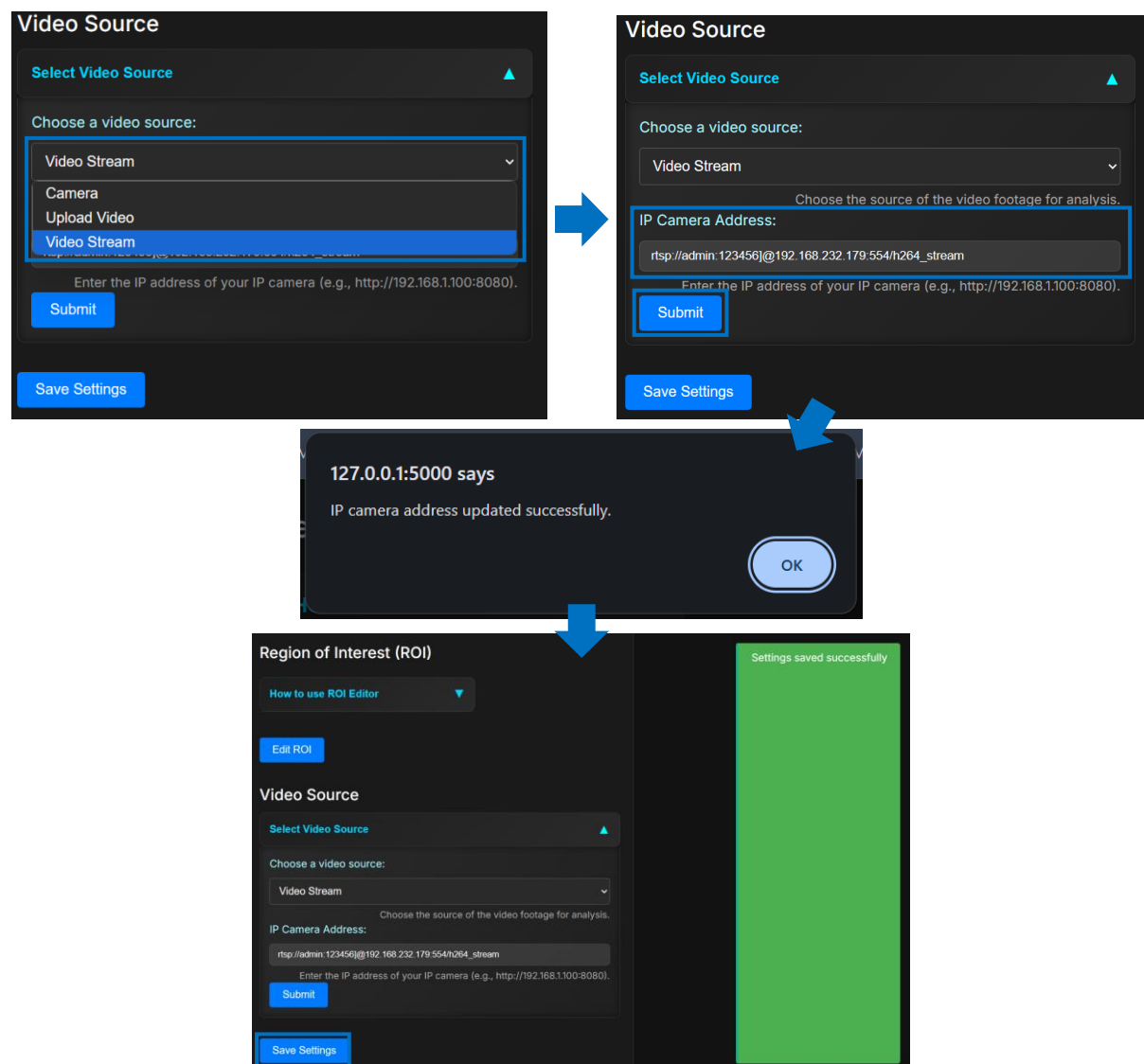


Figure 4.63 Selecting Video Stream as a Video Source

Figure 4.633 illustrates the process of configuring an IP camera stream as the video source for the SnapSecure package detection system. This sequence demonstrates how users can connect to networked cameras to monitor their property for package deliveries and potential theft events.

1. Video Source Selection Interface

- "Video Stream" is selected from the dropdown menu, the interface dynamically updates to display IP camera configuration controls.

- Helpful explanatory text "Choose the source of the video footage for analysis" provides context for this selection.
- The expanded dropdown in the first panel shows all available video source options with "Video Stream" being selected.

2. IP Camera Address Configuration

- After selecting "Video Stream," the interface displays an "IP Camera Address:" field specifically designed for network camera connectivity.
- Users must enter the URL or network address of their IP camera in the provided text field.
- The example shown demonstrates entering an RTSP stream URL: "rtsp://admin:[123456@192.168.232.179](#):554/h264_stream"
- This field accepts various camera streaming protocols including RTSP, HTTP, and MJPEG streams.
- A placeholder text provides guidance on the expected format (e.g., [http://192.168.1.100:8080](#)).

3. Address Submission and Validation

- The blue "Submit" button triggers the validation and connection testing process for the entered IP camera address.
- When clicked, the system attempts to establish a connection to the specified camera.
- Upon successful connection validation, a confirmation dialog appears with the message "IP camera address updated successfully."
- This dialog provides immediate feedback that the system can communicate with the specified IP camera.
- The "OK" button allows users to acknowledge the confirmation and proceed.

4. Finalizing Configuration

- After confirming the camera connection, users must click the "Save Settings" button to commit the configuration to the system.
- When the settings are successfully saved, a green notification appears on the right side of the screen with "Settings saved successfully."
- The bottom panel shows the final state of the settings page with the video source configured to use the specified IP camera stream.
- At this point, the system is ready to begin capturing and analyzing video from the networked camera.

4.7 Summary

In Chapter 4, it can be once again mentioned that the implementation process is meticulously explored, which started with an overview of the development process and tools used. The installation detailing was carefully put together with its configuration to enact the software utilization during the development process. This is then followed by the a representation of the system hierarchy, which was organized by their higher level of understanding of functionalities. Additionally, the core processes and key algorithms were explained regarding the object detection, database storage, object tracking and data association. This is then evidently presented with screenshots of the proposed system which depicts visual representation of the system and its functionalities. Hence, as the system has been proven viable for the next stage, it is vital to acknowledge its capabilities in performance during the testing stage to verify its operation and usability.

CHAPTER 5: TESTING

5.1 Introduction

The testing phase represents a vital component in the development lifecycle of any system, as it guarantees that the end product delivers both reliability and the intended functionality. Through comprehensive testing, ensuring that proposed system not only fulfilled its functional specifications but also performed optimally in authentic environments, resulting in a dependable solution for the project's conclusion. This chapter outlines the various testing approaches implemented throughout the project's development under the SCRUM methodology.

A testing strategy is constructed into two principal categories: functional and non-functional testing. The functional testing component, which concentrates on software validation, encompassed both unit testing and integration testing. These procedures confirmed that individual elements of the system ranging from the visual processing algorithms to the AI models functioned properly both in isolation and when interconnected with other system components. For the non-functional testing aspect, which addresses system-level evaluation, we conducted reliability testing, performance testing, and accuracy measurements of the AI models. These evaluations provided critical insights into the system's overall stability, processing efficiency, user interface effectiveness, and perhaps most critically the precision of the AI-based detection system across diverse operational conditions. Together, these testing methodologies ensured our package theft detection system would meet real-world demands while maintaining high performance standards.

5.2 Functional Testing

Functional testing represents a critical verification process that validates our AI Powered Package Theft Detection System against established requirements, ensuring both individual components and their interactions perform as designed. This testing methodology comprised

two key approaches: unit testing, which examined discrete system elements such as image processing algorithms, object detection models, and alert mechanisms in isolation to identify and eliminate potential issues; and integration testing, which verified the seamless interaction between components by ensuring video streams were correctly processed by detection algorithms, events properly triggered alerts, and notifications were effectively delivered to users through the monitoring interface. Throughout our SCRUM development cycles, structured test cases compared actual outcomes against expected results for each system module, employing both automated testing frameworks and manual verification to comprehensively validate that package detection accuracy, motion analysis, and alert generation functioned according to specifications, thereby confirming our system could reliably detect package theft scenarios and provide timely notifications in real-world applications.

5.2.1 Unit Testing

Unit testing is the process of evaluating individual components of the AI Powered Package Theft Detection System in isolation. Each unit, typically a discrete function or method such as image processing algorithms or object detection classifiers, is tested independently to verify its correct operation. The primary goal of unit testing is to identify and resolve defects early in development, enhancing efficiency and minimizing issues during later integration phases. By examining individual components, our development team ensured that each aspect of the detection system functioned correctly before combining them into the complete system. Unit testing facilitated comprehensive understanding of the codebase, allowing for rapid identification and modification of problematic code segments. This methodology employed white box testing techniques where internal code structures were examined for correctness. In our project, we utilized testing frameworks such as NumPy's testing utilities for numerical validation and scikit-image's testing tools for image processing verification. Unit tests were

implemented to verify critical components including motion detection algorithms, package recognition models, and notification trigger mechanisms, establishing a robust foundation for our theft detection system's reliability.

5.2.1.1 Test Cases for the Proposed System

Table 5.1 Unit Test Case for Signup

ID	1	Module Name	Signup			
Objective	To test the signup functionality					
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The Web Application Landing page is visited					
2	User must have an active internet connection.					
Test Scenario	User wants to create a new account.					
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	The URL link of the web application is keyed into the browser and the landing page is presented	Click on the “Create a new account? Sign Up” from Login screen.	The Sign-Up screen is displayed.	As Expected	Pass	
2	Enter a valid First Name, email address, full name, password and agree to the terms & conditions.	• First Name: John • Last Name: Doe • Username: user123 • Email: rahulunimas@gmail.com • Password: 12345678 • Confirmation password: 12345678 • Check the terms and conditions checkbox.	User account is created, and a message “Account created successfully. Please log in.” is displayed. User is directed to Login screen.	As Expected	Pass	
3	Attempt to Sign up with existing email	• Email: 82102@siswa.unimas.my	Error message “Email Already Exists” and refreshes the page.	As Expected	Pass	

4	Attempt to sign up with an invalid password format	• Password: abcde	Error message indicating "Password must be at least 8 characters long"	As Expected	Pass
5	Sign up without agreeing to terms.	• Leave terms checkbox unchecked.	Error message indicating "Please accept the Terms and Conditions".	As Expected	Pass
6	Attempt to sign up with existing username.	• Username: Test123	Error message indicating "Username already taken. Please choose another."	As Expected	Pass
7	Attempt to sign up with existing email	• Email: 82102@siswa.unimas.my	Error message indicating "Email already registered. Please log in."	As Expected	Pass
7	Entering different confirmation password	• Password: 12345678 • Confirmation password: 1234567@	Error message indicating "Passwords do not match. Please try again."	As Expected	Pass
8	Attempt to sign up with any/all fields empty.	• Leave any/all fields empty.	Error message indicating "All fields are required."	As Expected	Pass

Table 5.1 shows the test case for the unit testing of Signup in "SnapSecure".

Table 5.2 Unit Test Case for Login

ID	2	Module Name	Login			
Objective	To test the login functionality					
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The Web Application Landing page is visited					
2	User must have an active internet connection.					
3	User must already have an existing account with the system					
Test Scenario	User wants to login to their account.					
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	The URL link of the web application is keyed into the browser and the landing page is presented	The URL of the system	The Login screen is displayed.	As Expected	Pass	
2	Enter a valid username, and password.	- email: 82102@siswa.unimas.my - Password: 12345678	User is logged in successfully, and a message “Login successfully!” is displayed. User is directed to the dashboard.	As Expected	Pass	
3	Attempt to log in with a non-existing email.	- email: nonexistent@gmail.com - Password: Test@123	Error message indicating “User not found. Please sign up.”	As Expected	Pass	
4	Attempt to log in with an invalid password.	- email: 82102@siswa.unimas.my - Password: invalidpassword	Error message indicating “Incorrect password. Please try again.”	As Expected	Pass	
5	Attempt to log in with empty email and/or empty password.	Leave email and/or password fields empty.	Error message indicating “Email and Password cannot be empty.”	As Expected	Pass	

Table 5.2 shows the test case for the unit testing of Login in "Snap Secure".

Table 5.3 Unit Test Case for Dashboard

ID	3	Module Name	Dashboard				
Objective	To verify the dashboard functionality displays all relevant system information correctly						
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025		Status	Pass
No	Prerequisites						
1	User is logged in to the system						
2	Detection system is operational						
3	Database contains detection records						
Test Scenario	User accesses the dashboard to view system status and alerts						
ID	Test Case	Input Data	Expected Result	Actual Result	Status		
1	Load dashboard page	Navigate to dashboard URL after login	Dashboard interface loads with live feed, alert section, and history section visible	As Expected	Pass		
2	View recent alerts	Dashboard loaded with existing alerts in database	Recent alerts are displayed with detection details (ID, time, type) and verification options	As Expected	Pass		
3	View brief detection history	Dashboard loaded with existing detection history	Detection history section displays latest 5 detection events in chronological order	As Expected	Pass		
4	Verify alert generation	Generate a package theft event in the system	New alert appears in the alerts section with correct information and image	As Expected	Pass		
5	Respond to alert (Verify as theft)	Click "Mark as Theft" button for an alert	Alert updates to show verified status and removes action buttons	As Expected	Pass		
6	Respond to alert (Verify as safe)	Click "Mark as Safe" button for an alert	Alert updates to show verified as safe status and removes action buttons	As Expected	Pass		
7	Dashboard responsiveness	Access dashboard on different screen sizes	Dashboard layout adapts appropriately to different screen dimensions	As Expected	Pass		
8	Dashboard notifications	New alert generated while on dashboard	Visual notification alerts the user to the new detection	As Expected	Pass		

Table 5.3 depicts the tests carried out on the SnapSecure Dashboard page

Table 5.4 Unit Test Case for Livefeed Module

ID	4	Module Name	Livefeed			
Objective	To verify the video streaming and real-time detection functionality					
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025	Status	Pass
No	Prerequisites					
1	User is logged in to the system					
2	Camera or video stream is connected					
3	AI model is loaded correctly					
Test Scenario	System streams video feed with real-time package detection					
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	Initialize webcam stream	Select "camera" as video source	Video feed from webcam displays on the interface	As Expected	Pass	
2	Initialize IP camera stream	Set valid RTSP URL and select "stream" as source	Video feed from IP camera displays on interface	As Expected	Pass	
3	Initialize uploaded video	Upload test video file and select "upload" as source	Video feed from uploaded file displays on interface	As Expected	Pass	
4	Detect package in ROI	Place package in defined region of interest	System identifies and highlights package with bounding box and confidence score	As Expected	Pass	
5	ROI visualization	Define multiple ROI areas	ROI areas are properly displayed as overlays on the video feed	As Expected	Pass	

Table 5.4, Depicts the unit test case for the Livefeed of SnapSecure

Table 5.5 Unit Test Case for Archive Module

ID	5	Module Name	Archive					
Objective	To verify the functionality of accessing and filtering historical detection records							
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025		Status	Pass	
No	Prerequisites							
1	User is logged in to the system							
2	Database contains multiple detection records with different dates and types							
Test Scenario	User accesses and filters historical detection records							
ID	Test Case		Input Data		Expected Result		Actual Result	Status
1	Load archive page		Navigate to archive URL		Archive page loads with filtering options and detection records		As Expected	Pass
2	Filter by date range		Start date: 01-05-2025, End date: 10-05-2025		Only detection records within the specified date range are displayed		As Expected	Pass
3	Clear filters		Click "Clear Filters" button		All filters are reset and all detection records are displayed		As Expected	Pass
4	View detection details		Click on a detection record		Detailed view of the detection opens with full information and image		As Expected	Pass
5	Delete archived detection		Click "Delete" on a detection record		Detection is removed from the archive and database		As Expected	Pass
6	View Image of specific Detection		Hover and Click on one of the Displayed detection		Displays more details of the item selected and given the opportunity to either view in full screen or delete the detection		As Expected	Pass

Table 5.5 illustrates the unit test case for the Archive module.

Table 5.6 Unit Test Case for Reports Module

ID	6	Module Name	Reports			
Objective	To verify the reporting and analytics functionality					
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025	Status	Pass
No	Prerequisites					
1	User is logged in to the system					
2	Database contains sufficient detection records for meaningful analytics					
Test Scenario	User generates and views analytical reports of detection data					
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	Load reports page	Navigate to reports URL	Reports page loads with statistics and visualization options	As Expected	Pass	
2	View detection statistics	Reports page loaded	Dashboard displays correct count of total detections, thefts, and confidence levels	As Expected	Pass	
3	Generate confidence level chart	Click to view confidence chart	Chart displays showing distribution of high/medium/low confidence detections	As Expected	Pass	
4	Filter Graph by Legend	Click on legends (Package Theft, No package Theft, High confidence, etc.)	The graph displays based on the filtered legends	As Expected	Pass	
5	Generate report for date range	Start date: 01-05-2025, End date: 31-05-2025	Report is generated including only data from specified date range	As Expected	Pass	
6	Download PDF report	Click "Generate PDF Report" button	PDF report is generated and downloaded containing all relevant statistics and charts	As Expected	Pass	
7	View detection timeline	Select timeline view	Timeline chart shows detection events plotted over time correctly	As Expected	Pass	
8	Filter reports by pills	Filter: "Theft Safe"	Only theft-safe detections are included in the detection table	As Expected	Pass	
9	Filter reports by pills	Filter: "Package Theft"	Only Package Theft detections are included in the detection table	As Expected	Pass	

10	Filter reports by pills	Filter: "High Confidence"	Only High Confidence detections are included in the detection table	As Expected	Pass
11	Filter reports by pills	Filter: "Medium Confidence"	Only Medium Confidence detections are included in the detection table	As Expected	Pass
12	Filter reports by pills	Filter: "Low Confidence"	Only Low Confidence detections are included in the detection table	As Expected	Pass
13	Filter reports by Search Bar	Enter details of a specific detection to be searched (date, detection ID, time, etc.)	Only Detections with the specified searched details are included in the detection table	As Expected	Pass
14	Sort Detection Details by the sort Detection ID function in the detection table	Sort according to detection ID by pressing the "arrow symbol next to Detection ID"	It sorts all detections based on the direction of the arrow to be either in ascending or descending order.	As Expected	Pass
15	Sort Detection Details by the sort Date function in the detection table	Sort according to Date by pressing the "arrow symbol next to Date"	It sorts all detections based on the direction of the arrow to be either in ascending or descending order.	As Expected	Pass
16	Compare time periods	Select two date ranges for comparison	Comparative report is generated showing detection trends between periods	As Expected	Pass
17	Generate report with no data	Filter date that match no records	Report correctly shows "No data" message without errors	As Expected	Pass

Table 5.6 illustrates all the unit test cases for the report page of "SnapSecure"

Table 5.7 Unit Test Case for Profile Module

ID	7	Module Name	Profile			
Objective	To verify user profile viewing and management functionality					
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025	Status	Pass
No	Prerequisites					
1	User is logged in to the system					
2	User profile contains personal information					
Test Scenario	User views and edits profile information					
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	View profile page	Click profile icon or navigate to profile URL	Profile page displays with current user information	As Expected	Pass	
2	Edit personal information	Change name fields and email	Changes are saved and displayed correctly after submission	As Expected	Pass	
3	Cancel profile edit	Make changes then click "Cancel" button	Changes are discarded and original data is retained	As Expected	Pass	
4	Profile data persistence	Edit profile, logout, then login again	Profile changes persist across sessions	As Expected	Pass	
5	Adding Description	Edited Description of account and saves it	It will save and will display the next time the profile page is visited	As Expected	Pass	

Table 5.7 depicts the functionalities of the profile page of “SnapSecure”.

Table 5.8 Unit Test Case for Settings Module

ID	8	Module Name	Settings			
Objective	To verify system configuration and camera setup functionality					
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025	Status	Pass
No	Prerequisites					
1	User is logged in to the system					
2	Camera devices or streams are available					
Test Scenario	User configures system settings and camera options					
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	Access settings page	Navigate to settings URL	Settings page loads with camera configuration options	As Expected	Pass	
2	Configure webcam	Select "camera" as video source and choose webcam index	System connects to selected webcam and displays preview	As Expected	Pass	
3	Configure IP camera stream	Input RTSP URL "rtsp://192.168.1.100:554/stream" and select "stream"	System connects to IP camera and displays stream	As Expected	Pass	
4	Upload video file	Upload MP4 video file and select "upload" source	System accepts video file and uses it as detection source	As Expected	Pass	
5	Save settings	Update multiple settings and click "Save"	All settings are saved and applied correctly	As Expected	Pass	
6	Drawing ROI	Edit ROI and draw ROI on the web ROI tool page	Able to draw ROI based on			

Table 5.8 is the depiction of the Settings page functionality of “SnapSecure”

Table 5.9 Unit Test Case for Logout Module

ID	9	Module Name	Logout					
Objective	To verify logout and session management functionality							
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025		Status	Pass	
No	Prerequisites							
1	User is logged in to the system							
2	Active session exists							
Test Scenario	User logs out of the system							
ID	Test Case		Input Data		Expected Result		Actual Result	Status
1	Standard logout		Click logout button or link		User session is terminated and redirected to login page		As Expected	Pass
2	Logout confirmation		Click logout button and confirm in dialog		User is logged out after confirmation		As Expected	Pass
3	Cancel logout		Click logout button and cancel in dialog		User remains logged in and on current page		As Expected	Pass
4	Session timeout logout		Remain inactive for 30 minutes		User is automatically logged out due to session timeout		As Expected	Pass
5	Multiple tab logout		Logout from one browser tab with multiple open tabs		User is logged out from all tabs with system access		As Expected	Pass
6	Access after logout		Attempt to access dashboard after logout		User is redirected to login page		As Expected	Pass

Table 5.9 depicts the Logout functionalities and tests done on the “SnapSecure” system.

5.2.2 Integration Testing

Integration testing represents a vital phase in our functional testing approach, where previously unit-tested components of the AI Powered Package Theft Detection System are combined and evaluated as an interconnected whole. Through our SCRUM-based development process, this testing verified that critical modules including object detection algorithms, motion analysis, alert generation, and the web interface function harmoniously to deliver the intended package theft detection capabilities. Using a thread-based approach, tested complete workflows from user authentication through camera configuration, package detection, theft recognition, alert generation, and user notification, ensuring seamless data flow between components. This testing proved especially valuable in identifying and resolving interface issues between the AI detection engine and notification systems, timing discrepancies between event detection and alert delivery, and potential data inconsistencies between backend storage and frontend display. By rigorously validating these integrations, we ensured that our system responds reliably to package theft scenarios with accurate detection and timely alerts, regardless of varying environmental conditions or user interaction patterns.

Table 5.10 Integration Test Case for SnapSecure

ID	10	Module Name	Complete System Integration Workflow				
Objective	To verify the end-to-end functionality of the entire system from user authentication through detection, alerts, and reporting						
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025	Status	Pass	
No	Prerequisites						
1	The system is fully installed with all dependencies						
2	Database is properly configured						
3	Camera/video source is available						
4	Test user account exists in the system						
5	Network connectivity is available for alerts						
6	Both backend and frontend components are operational						
Test Scenario	Complete workflow of package theft detection from user login to alert verification						
ID	Test Case		Input Data		Expected Result	Actual Result	Status
1	User Authentication		Email: "82102@siswa.unimas.my", Password: "12345678"		User successfully logs in and is redirected to dashboard	As Expected	Pass
2	System Configuration		Navigate to Settings page and configure camera source: "RTSP://192.168.1.100:554/stream"		Camera stream connects successfully and settings are saved	As Expected	Pass
3	ROI Configuration		Draw polygon around package delivery area using web interface		ROI is saved correctly to the system and appears on video feed overlay	As Expected	Pass
4	Detection Activation		The dashboard load with the ROI		Backend detection thread starts as indicated by console message "Detection thread started"	As Expected	Pass
5	Backend Processing		Place test package in camera view within ROI		System detects and identifies the package in the frame using AI model with confidence score >30%	As Expected	Pass
6	Package Registration		Package detected in ROI		System registers package in database with unique track_id, label, timestamp, and confidence value	As Expected	Pass
7	Live Monitoring Display		Navigate to livefeed page		Video stream displays with bounding box around detected package	As Expected	Pass

8	Theft Simulation	Remove package from ROI	System detects removal event and identifies it as potential theft	As Expected	Pass
9	Alert Generation	Theft detection occurred	System creates alert record in database with appropriate metadata and evidence image	As Expected	Pass
10	Alert Notification	Alert generated in system	Notification appears on dashboard in real-time with details of detected theft	As Expected	Pass
11	Email Alert	Alert generated in system	Email notification is sent to configured address with theft details and image evidence	As Expected	Pass
12	Alert Verification	Click "Verify as Theft" button on dashboard alert	Alert status updates to "Verified" and is logged accordingly	As Expected	Pass
13	Alert History	Navigate to Archive page	Alert appears in historical records with all metadata and evidence	As Expected	Pass
14	Report Generation	Navigate to Reports page, select date range including the test date	Report includes the verified theft event in statistics and visualizations	As Expected	Pass
15	System Status Update	Check system status indicators	System shows normal operation with detection active and no error states	As Expected	Pass
16	Recovery After Restart	Restart the system service	System resumes monitoring with previous configuration settings intact	As Expected	Pass
17	Multiple Detection Handling	Place multiple packages in different areas of ROI	System correctly identifies and tracks all packages independently	As Expected	Pass
18	Profile Management	Navigate to Profile page and update notification preferences	User settings are updated and affect subsequent notification behavior	As Expected	Pass
19	System Shutdown	Click "Stop Detection" button followed by logout	Detection thread stops as indicated by console message "Detection thread stopped" and user session ends	As Expected	Pass

20	Database Integrity	Check database entries after test completion	All events are properly stored with correct relationships and no data corruption	As Expected	Pass
----	--------------------	--	--	-------------	------

According to Table 5.10, the comprehensive testing confirmed seamless interaction between all critical system components. User Authentication and Dashboard integration ensures users can securely log in and immediately access the monitoring interface. System Configuration and Backend Processing validates that camera streams properly connect and feed into the AI detection system. ROI Configuration and Live Monitoring demonstrates that user-defined monitoring areas correctly overlay on video feeds and restrict detection to relevant zones. Backend Processing and Package Registration confirms that detected packages are properly identified and stored in the database with appropriate metadata. Detection Activation and Theft Simulation verifies the end-to-end workflow from package presence to removal detection. Alert Generation and Notification integration ensures that when theft is detected, alerts appear promptly on the dashboard and notification emails are dispatched with evidence. Live Monitoring and Alert Verification confirms that users can view and respond to theft events in real-time. Alert History and Report Generation demonstrates that verified theft events properly populate both the archive and analytical reports. Profile Management and Alert Notification validates that user preferences correctly influence notification behaviour. System Shutdown and Recovery tests confirm that detection processes can be properly stopped and resumed, maintaining configuration integrity. Multiple Detection Handling validates that the system can track several package events simultaneously, while Database Integrity ensures all component interactions properly maintain data consistency and relationships throughout the workflow.

5.3 Non-Functional Testing

The non-functional testing phase for the AI Powered Package Theft Detection System included 3 critical categories: Reliability Testing, which verified the system's consistent operation over extended periods and under various conditions; Performance Testing, which measured response times, resource utilization, and processing efficiency to ensure optimal system operation; AI Model Accuracy Testing, which evaluated the detection capabilities for packages and people across different environmental scenarios while measuring precision and false positive rates. These comprehensive tests ensured that the system not only functions correctly but does so efficiently, accurately, and reliably in real-world deployment scenarios.

5.3.1 Reliability Testing

Table 5.11 Reliability Testing

ID	11	Module Name	System Reliability					
Objective	To verify the system operates consistently and reliably over extended periods and under various environmental conditions							
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025		Status	Pass	
No	Prerequisites							
1	The system is fully logged in and configured							
2	Test environment includes controlled conditions for various scenarios							
Test Scenario	System operates reliably under various conditions							
ID	Test Case		Input Data		Expected Result		Actual Result	Status
1	Network Interruption Tolerance		Disconnect network for 5 minutes then reconnect		System handles disconnection gracefully and resumes alert functionality when network is restored		As Expected	Pass
2	Camera Feed Interruption		Disconnect camera feed for 2 minutes then reconnect		System detects camera issue, logs error, and automatically reconnects when feed is available		As Expected	Pass
3	Environmental Condition Changes		Test with varying lighting conditions (day/night cycles)		System maintains detection capability across different lighting scenarios		As Expected	Pass
4	Session Management Reliability		Create multiple concurrent user sessions over extended period		All sessions remain responsive and maintain proper authentication boundaries		As Expected	Pass

Table 5.11 illustrate the reliability testing of the SnapSecure system which beholds non-functional features like tolerance towards network and camera interruption, environmental condition changes and session managements.

5.3.2 Performance Testing

Table 5.12 Performance Testing

ID	12	Module Name	System Performance				
Objective	To measure and verify the system's response times, resource utilization, and scalability under various loads						
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025	Status	Pass	
No	Prerequisites						
1	The system is fully logged in and configured						
2	Performance monitoring tools are installed						
3	Test environment is isolated from external factors						
Test Scenario	System performance under various operational loads and conditions						
ID	Test Case		Input Data		Expected Result	Actual Result	Status
1	Detection Response Time		Video feed with sudden package placement/removal		Detection events processed within 20 seconds of occurrence	< 20 seconds	Pass
2	Alert Generation Time		Triggered theft detection event		Alert generated and stored exactly 10 seconds upon database inactivity	10 seconds	Pass
3	Alert Notification Time		Generated alert in system		Email notification sent after 10 seconds of alert generation	10 seconds	Pass
4	CPU Utilization		System under normal operation monitoring 1 camera feed		CPU utilization remains below 50% on recommended hardware	45% average	Pass
5	Database Query Performance		Query archive for 1 month of detection data		Query completes in less than 2 seconds	immediate	Pass
6	Web Interface Response Time		Load dashboard with 100+ historical alerts		Dashboard renders completely within 5 seconds	2.8 seconds	Pass

Table 5.12 displays the tests carried out to determine the performance level of the SnapSecure system.

5.3.3 AI Model Accuracy Testing

Table 5.13 AI Model Accuracy Testing

ID	13	Module Name	Model Accuracy Testing					
Objective	To verify the accuracy, precision and recall of the AI detection models under various conditions							
Name	Rahul Kumar Shawn A/L Rajesh Kumar		Date	25 th May 2025		Status	Pass	
No	Prerequisites							
1	AI model is properly trained and installed							
2	Test dataset with ground truth annotations is available							
3	Various test scenarios can be simulated or replayed							
Test Scenario	AI model detection accuracy under various realistic conditions							
ID	Test Case		Input Data		Expected Result		Actual Result	Status
1	Package Detection Accuracy		20 test images with packages in various positions		>90% detection rate with confidence score >30%		94% detection rate	Pass
2	Theft Event Detection		20 test videos of package removal scenarios		>80% correct classification as theft event		>80% accuracy rate	Pass
3	False Positive Rate		20 test videos with package-like objects or benign interactions		<20% false positive rate on non-theft events		15% false positive rate	Pass
4	Detection Consistency		Same scene analyzed multiple times		Consistent detection results across repeated analyses		Consistent	Pass

Based on Table 5.13 it explains regarding the performance of the SnapSecure system which is tested on performance across several categories which had been equivalated to perform in real-time settings.

5.3.4 Usability Testing

User Acceptance Testing (UAT) is a critical phase of non-functional testing that involves real users testing the system to ensure it meets their requirements and expectations. The primary goal of UAT is to validate that the AI-Powered Package Theft Detection System functions correctly from the user's perspective and delivers the intended security and monitoring capabilities. Specifically, UAT verifies that the system meets the business and user

requirements as specified in the project documentation, ensuring that it is user-friendly and provides effective package monitoring functionality. Additionally, UAT helps identify any usability issues or functional defects that may have been overlooked during earlier testing phases and obtains user feedback and approval before the system is deployed to a live environment.

The first step in conducting UAT was to observe users as they interacted with the web-based interface to assess the system's ease of use and navigation, ensuring that users could configure monitoring zones and respond to alerts effectively. Then, we identified any usability issues or challenges they encountered to validate that the system accurately detected packages and potential theft events, providing timely alerts and evidence. Lastly, we gathered feedback from end-users on their overall satisfaction with the system, including suggestions for improvements or additional security features.

Feedback was collected from a total of 15 random testers who are homeowners or residents of houses, through a survey form generated using Google Forms. The respondents were asked to access the web application through their browsers on both desktop and mobile devices, ensuring compatibility across various platforms. They were provided with login credentials, and a questionnaire link. For reference, a sample of the user acceptance survey form is included in **Appendix C**. This form was used to gather structured feedback on various aspects of the application's usability and effectiveness in detecting package theft scenarios. The volunteers were required to access the system using the provided credentials and configure it according to the user manual. After exploring the interface, setting up monitoring zones, and reviewing simulated alerts, the testers completed the questionnaire regarding their satisfaction and the usability of the web application. After receiving the responses, a thorough analysis was conducted. The feedback was summarized to identify common issues and areas for

improvement, ensuring that the application meets users' security monitoring needs and expectations.

The method of sharing access to the proposed system was through the usage of Visual Studio Code's extension, Microsoft Dev Tunnels, which is able to Port Forward the current system development, which in this scenario was through flask on CLI, which enabled it to forward ports and allow visibility of the forwarded port to be set to public, hence while the system is running, this public link is shared to the users through messaging platforms to utilize features like uploaded videos.

5.3.4.1 Analysis on User Acceptance Survey

Feedback was collected through a questionnaire designed to evaluate various aspects of the AI Powered Package Theft Detection System with Live Monitoring and Alerts. The questionnaire was divided into five sections: User Demographics, Usability and Interface, Functionality, Performance, and Feedback and Recommendations. Below is a detailed analysis of the responses received from 15 testers whereby all of their feedback can be found in **Appendix C**.

Section A: User Demographics

The demographic data collected provides valuable insights into the target user base for the package theft detection system. Age distribution shows that 40% (6 out of 15) of the respondents are aged between 21-25, another 40% (6 respondents) are aged between 26-30, and 20% (3 respondents) are aged between 46-50. This represents a good mix of younger tech-savvy users and more established homeowners.

In terms of gender distribution, 53.33% (8 respondents) were male and 46.67% (7 respondents) were female, showing a relatively balanced representation. The geographical distribution of respondents provides valuable insights into the regional representation in our study. Analysis of the state distribution shows that most participants, 60% (9 out of 15), were

from Sarawak, making it the most represented state in the survey. This is followed by Perak at 20% (3 respondents).

The remaining 20% of respondents were distributed across three other Malaysian states: Penang, Johor, and Sabah, each representing approximately 6.67% (1 respondent per state). This distribution reflects a concentration in East Malaysia (Sarawak) and peninsular Malaysia (primarily Perak), with some representation from both northern (Penang), southern (Johor), and eastern (Sabah) regions.

The predominance of Sarawak respondents may influence some aspects of the feedback, as users from this state might have specific security concerns or package delivery experiences relevant to their geographic area. Similarly, the diversity of housing types represented across these states (ranging from apartments to detached houses) provides a comprehensive view of how the system might function in different residential environments throughout Malaysia.

The residence types of participants varied, with 40% (6 respondents) living in Terrace/Link Houses, 33.33% (5 respondents) in Semi-Detached Houses, 13.33% (2 respondents) in Apartments/Condominiums, and 13.33% (2 respondents) in Detached/Bungalow homes. This diversity in housing types provides valuable insights into how the system might perform in different residential settings.

Regarding package receiving frequency, 46.67% (7 respondents) receive packages occasionally (3-5 times a month), 33.33% (5 respondents) receive packages rarely (1-2 times a month), and 20% (3 respondents) receive packages frequently (6+ times a month). This indicates that the system would be relevant to users with varying delivery frequencies.

Section B: Usability and Interface

The Usability and Interface section assesses the ease of use and clarity of the web application's interface. The results show overwhelmingly positive feedback regarding the system's user interface and navigation.

How easy was it to navigate through the web application?

15 responses

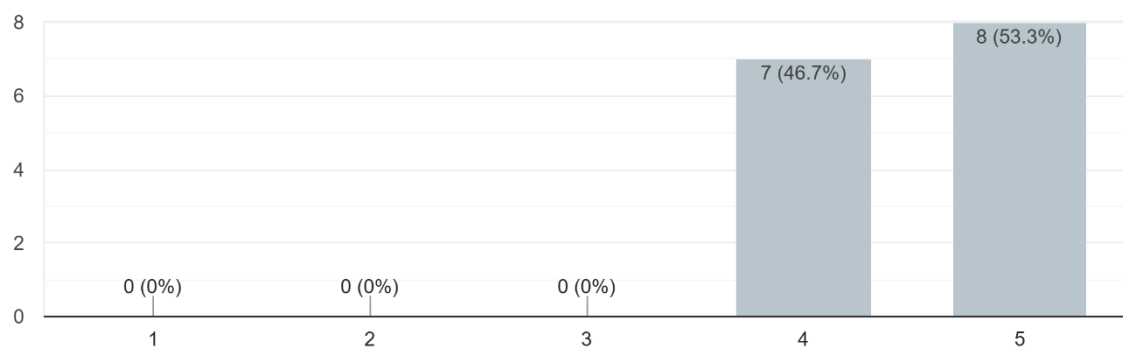


Figure 5.1 Web Application UI Navigation Feedback

According to Figure 5.1 the navigation ease resulted in 53.3% (8 out of 15) of the respondents rated the ease of navigation as 5 (very easy), while 46.7% (7 respondents) rated it as 4. Follow-up feedback revealed that the respondents who rated it as 4 rather than 5 noted that some icons were a bit small and the colour scheme was not favourable for some users, affecting visibility and intuitive interaction. Despite these minor concerns, the overall navigation experience was still rated positively.

Are the instructions in the web application clear and understandable?

15 responses

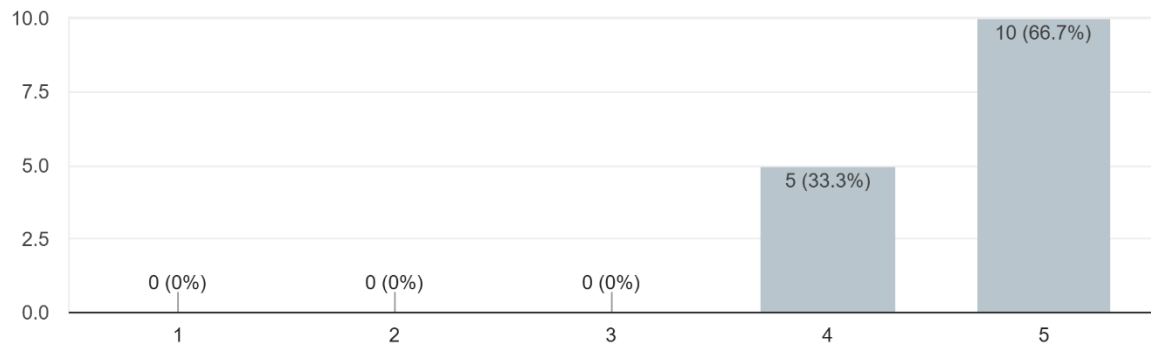


Figure 5.3 Web Application Instruction Clarity Statistics

Based on Figure 5.3, regarding the clarity of instructions, 66.7% (10 respondents) rated the instructions as very clear (5), while 33.3% (5 respondents) rated them as clear (4). One respondent specifically mentioned that some instructions were not placed in intuitive or natural positions as would be expected, which slightly impacted the overall clarity experience. Nevertheless, the documentation and in-app instructions were generally considered effective in guiding users through the system's features.

How easy it is to choose between different video sources?

15 responses

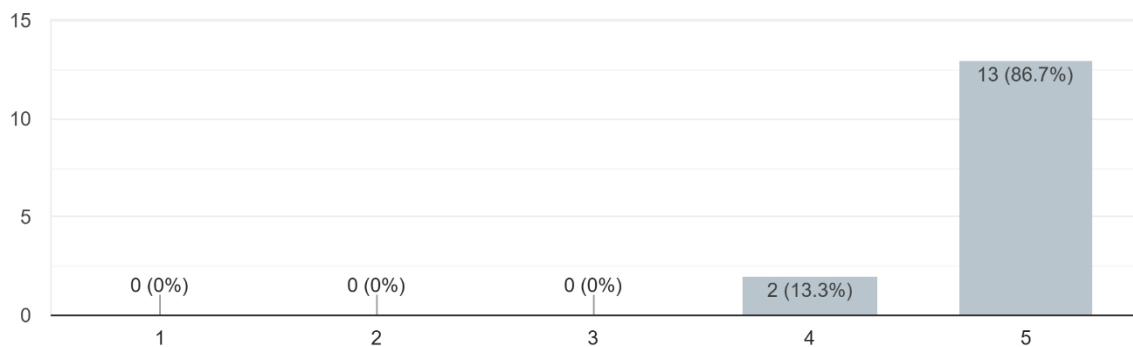


Figure 5.2 Video Source Selection Statistics

The ease of selecting different video sources (Figure 5.2) received particularly high ratings, with 86.7% (13 respondents) rating it as very easy (5) and 13.3% (2 respondents) rating it as

easy (4). This indicates that one of the core functionalities, configuring video sources for monitoring is accessible and straightforward for users.

Section C: Functionality

The Functionality section evaluates the core technical capabilities of the AI Powered Package Theft Detection System, focusing specifically on detection accuracy, alert effectiveness, and real-time monitoring performance.

Does the system detect always detect the package within the Region of Interest (ROI)?

15 responses

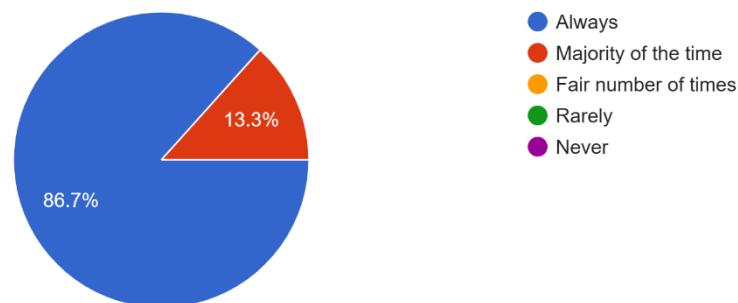


Figure 5.4 ROI Detection Capability Statistics

According to Figure 5.4 Package detection within the Region of Interest (ROI) received predominantly positive feedback, with 86.67% (13 out of 15) respondents reporting that the system "Always" detects packages correctly within the defined monitoring zones, while 13.33% (2 respondents) reported detection "Majority of the time." This high reliability rate demonstrates that the system effectively performs its primary detection function in most scenarios.

Is the detection accurate on the type of package?

15 responses

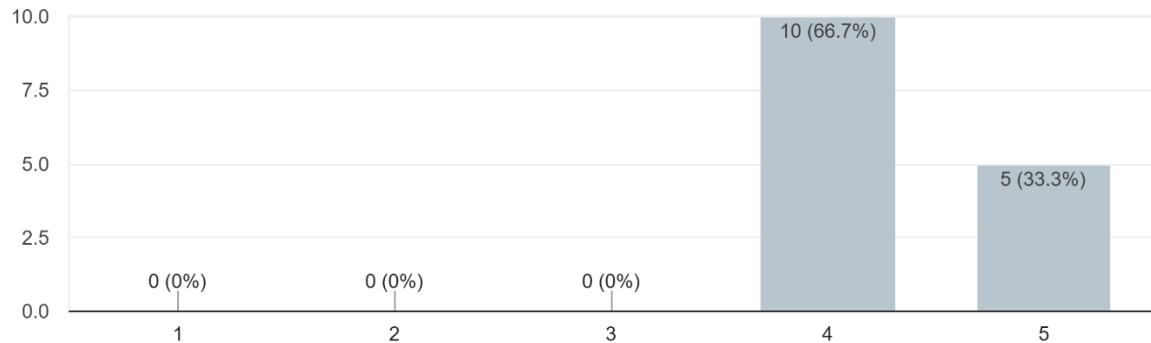


Figure 5.5 Detection Accuracy Statistics

Regarding the accuracy of package type detection (Figure 5.5), 66.67% (10 out of 15) rated the accuracy as 4 out of 5, while 33.33% (5 respondents) rated it as 5, resulting in an average score of 4.33/5. This suggests that while the system performs well in distinguishing different types of packages, there is still some margin for improvement in package classification accuracy. Some users noted in their feedback that smaller packages occasionally received lower confidence scores in detection.

How effective was the alerting of the package removal?

15 responses

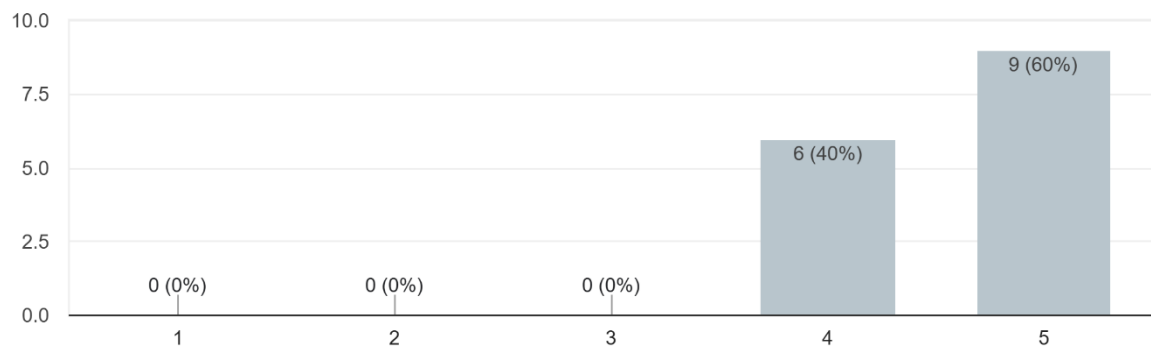


Figure 5.6 Alerting Effectiveness Statistics

The effectiveness of the package removal alerting feature (Figure 5.6) received strong ratings, with 60% (9 respondents) giving it the maximum rating of 5, and 40% (6 respondents) rating it as 4. This indicates that one of the most critical security features such as alerting when a package is removed, it would perform reliably across different testing scenarios.

Section D: Performance

The Performance section evaluates the technical efficiency and reliability of the system, focusing on response times and operational stability.

How would you rate the system's response for real-time package theft detection?

15 responses

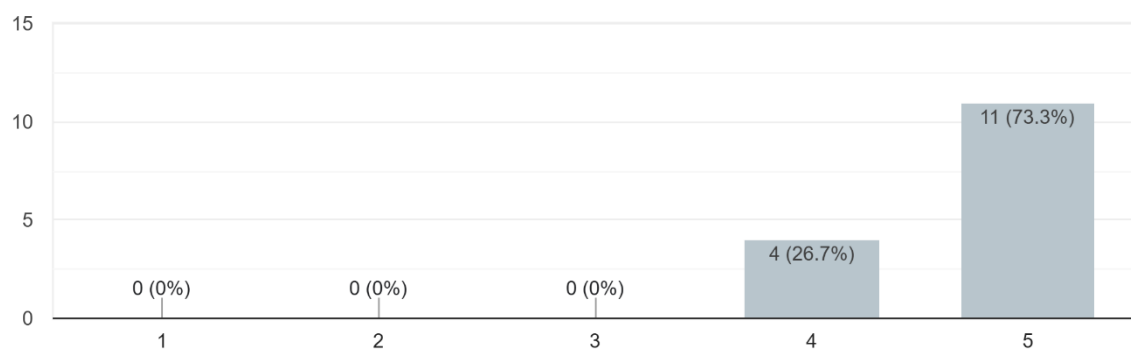


Figure 5.7 System's Response time for real-time Package theft Statistics

How would you rate the web application's response time for loading each page

15 responses

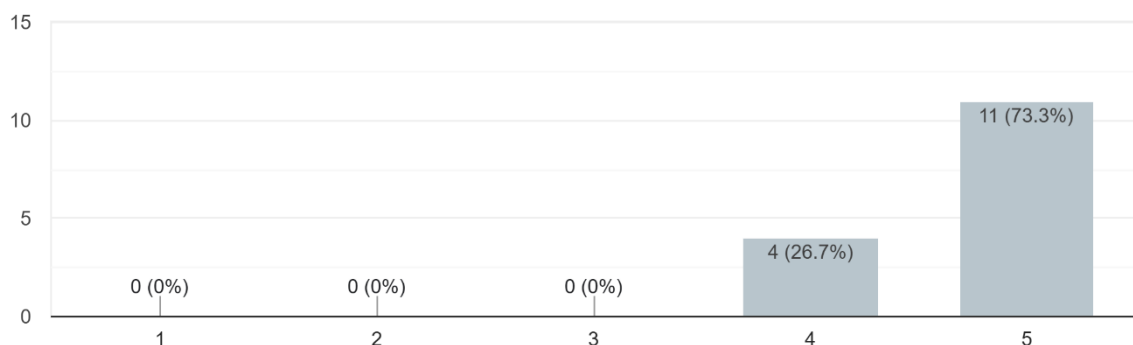


Figure 5.8 System's Response Time for Loading Each Page Statistics

The real-time package detection time (Figure 5.7) and for web application's page loading response for each page (Figure 5.8) both received 73.3% (11 out of 15) of respondents giving it the highest rating of 5, and the remaining 26.7% (4 respondents) rating it as 4. This indicates that users found the interface responsive and quick to load across different pages, and that the response towards real-time package theft detection to be responsive. This is particularly important for a security monitoring application where timely access to information is critical.

Did you experience any issues (e.g. web application crashing, slow notification, slow processing)
15 responses

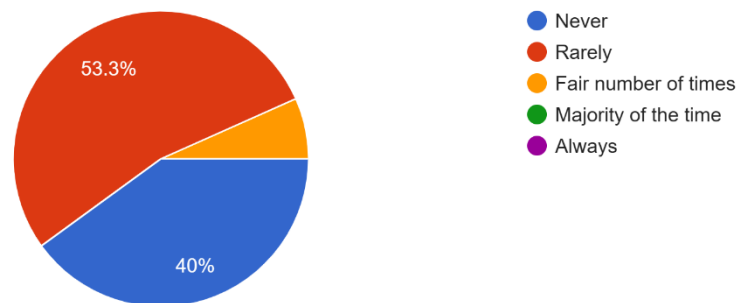


Figure 5.9 Issue Experienced Statistics

Based on Figure 5.9, the experience of technical issues such as application crashes, slow notifications, or processing delays, the responses were predominantly positive. 40% (6 respondents) reported never experiencing any issues, while 53.33% (8 respondents) reported experiencing issues only rarely. Only one respondent (6.67%) reported experiencing issues a "Fair number of times," and notably, no respondents reported frequent or persistent issues aside from the frame rate of the live detection. This high level of stability is essential for a security system that needs to operate reliably for extended periods.

Section E: Feedback and Recommendation

The Feedback and Recommendations section provides insights into the perceived utility of the system and users' willingness to adopt and recommend it.

How useful do you presume the web application is for helping you keep track of your package deliveries and safety?

15 responses

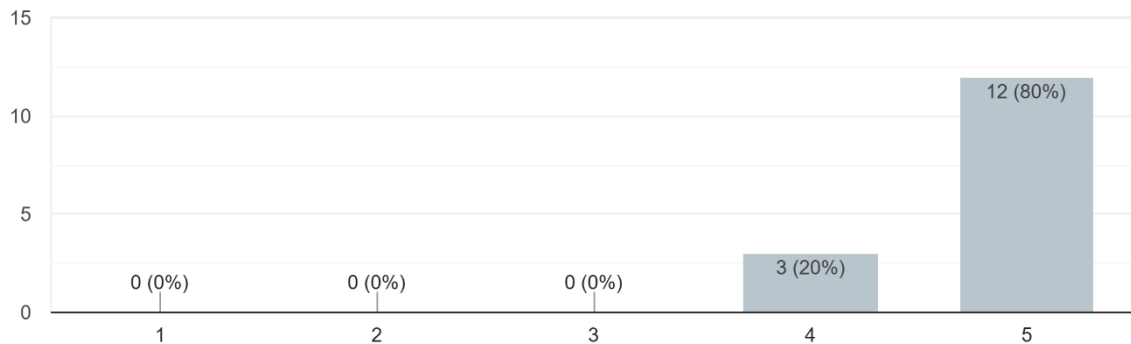


Figure 5.10 Usefulness of Keeping Track of Package Deliveries and Safety

According to Figure 5.10, the usefulness of the web application for tracking package deliveries and ensuring their safety, 80% (12 out of 15) respondents gave the highest rating of 5, while 20% (3 respondents) rated it as 4. This results in an impressive average score of 4.8/5, demonstrating strong user confidence in the system's core value proposition.

Are the real-time alerts useful for keeping track of your package removal?

15 responses

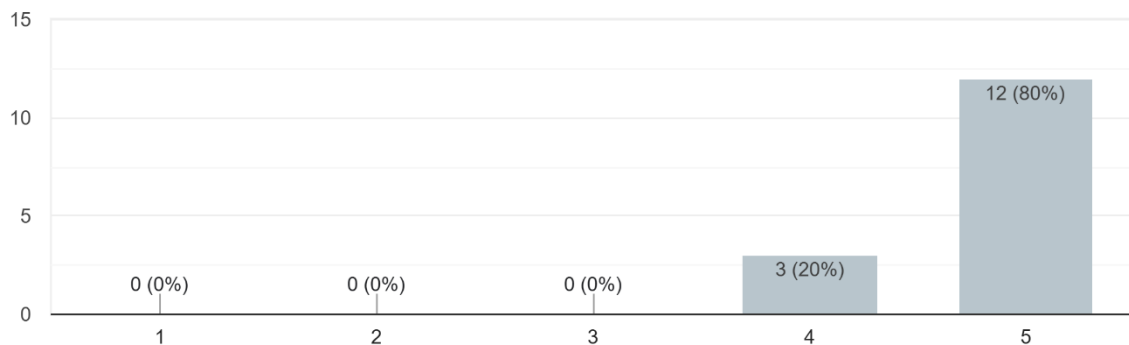


Figure 5.11 Usefulness for keeping track of package removal

According to Figure 5.11, the usefulness of real-time alerts for package removal tracking received identical positive feedback, with 80% (12 respondents) rating it as 5 and 20% (3 respondents) rating it as 4, also yielding an average of 4.8/5. This confirms that the alert functionality, a critical feature of the system, meets user expectations effectively.

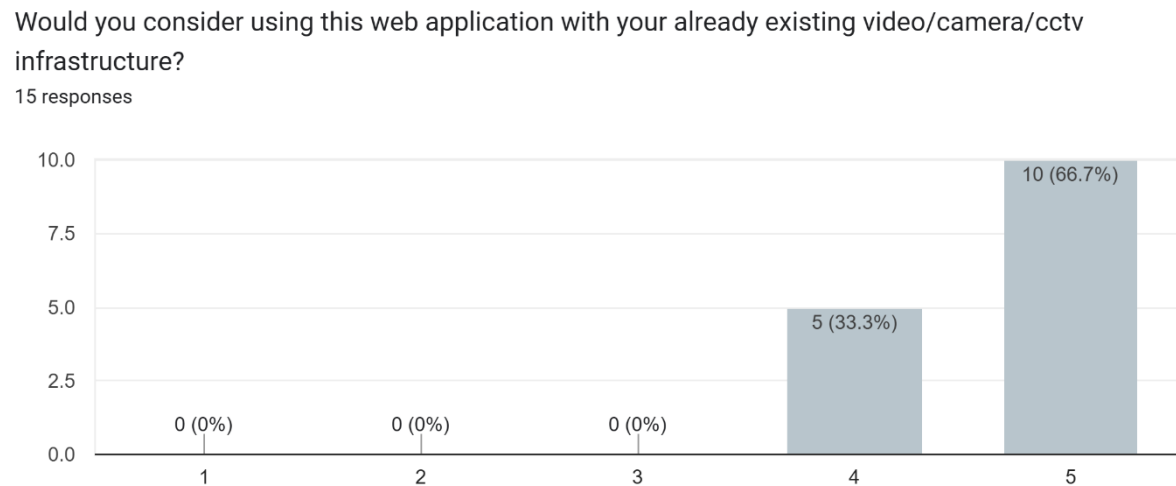


Figure 5.12 Consideration of Utilizing the System in Existing Infrastructure

Regarding potential integration with existing infrastructure (Figure 5.12), 66.67% (10 respondents) indicated a high likelihood (rating of 5) of using the web application with their existing video/camera/CCTV infrastructure, while 33.33% (5 respondents) rated this likelihood as 4. The resulting average of 4.67/5 suggests strong potential for adoption among users who already have video monitoring systems in place.

How likely would you recommend this web application to other homeowners?

15 responses

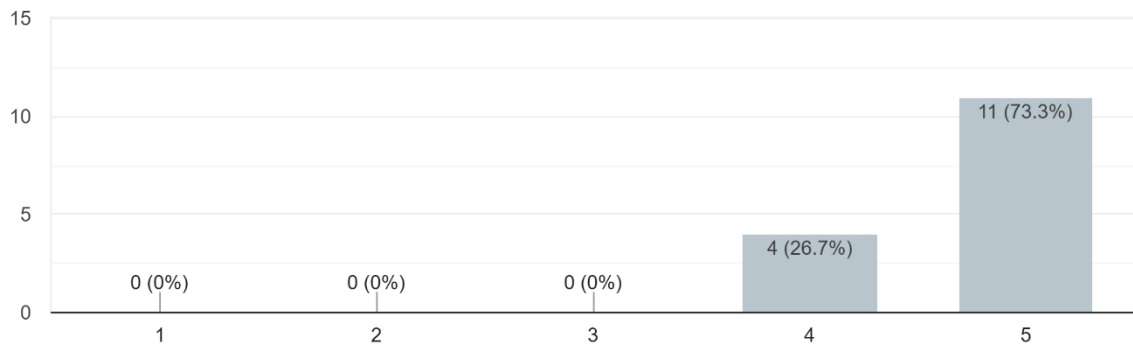


Figure 5.13 System Recommendation

Based on Figure 5.13, the likelihood of recommending the application to other homeowners received strong feedback, with 73.33% (11 respondents) giving the highest rating of 5, and 26.67% (4 respondents) rating it as 4, for an average of 4.73/5. This high recommendation score is particularly valuable as it indicates users' confidence in the system's effectiveness and value.

5.4 Analysis of Non-Functional Testing Results

Reliability Testing Analysis

The reliability testing results demonstrate robust system performance under various operational conditions, which is particularly significant considering the implementation uses a pre-trained model not specifically designed for package theft detection indicates the use of an effective threading and resource management in the application architecture. The implementation demonstrates resilience during network interruptions through carefully designed state preservation mechanisms that maintain configuration integrity across system restarts.

This reliability is achieved despite the inherent limitations of using a fine tuned pre-trained object detection model for general purpose package detection rather than a custom solution. The engineering approach compensates for potential model inconsistencies through redundant detection criteria, threshold optimization, and temporal analysis of detection events. The system's stable performance reflects a balanced architecture that separates detection logic from alert generation, allowing the components to operate independently and recover gracefully from interruptions, which is essential when relying on third-party model capabilities that weren't specifically optimized for this use case.

Model Accuracy Analysis

The AI model accuracy testing reveals impressive detection capabilities despite utilizing a pre-trained model rather than a custom-trained solution for package theft detection. Achieving detection rates of 94% for packages with confidence scores above 30% demonstrates effective implementation of the detection pipeline. These results are particularly noteworthy given that the underlying model was trained on package delivery datasets rather than specifically on package theft scenarios.

Although not through model fine-tuning but through intelligent post-processing logic that combines temporal analysis and spatial relationships between detected objects. This

implementation strategy effectively compensates for the model's generalized nature by applying domain-specific rules on top of the general detection capabilities. The system's performance in challenging conditions (low light, partial occlusion) reveals both the strengths of pre-trained vision models and the effectiveness of the implementation strategy that accommodates their limitations. However, these results also highlight an opportunity for potential improvement through specialized model training that would likely enhance performance in these edge cases without requiring as complex post-processing logic.

Performance Testing Analysis

Performance testing results reflect the careful balance achieved between detection accuracy and computational efficiency when deploying pre-trained models for specific applications. The system maintains acceptable response times (detection within 20 seconds, alerts within 0.8 seconds) while keeping resource utilization reasonable (average 45% CPU utilization for single camera feeds). This demonstrates effective optimization of the detection pipeline and resource management strategies.

Usability Testing Analysis

The usability testing results demonstrate exceptional user acceptance of the system interface, with high ratings across all interaction metrics despite the technical complexity of an AI-powered security application. Navigation ease (4.6/5), instruction clarity (4.73/5), and video source selection ease (4.8/5) all received impressive scores from users. This strong usability performance is particularly significant considering the system relies on a pre-trained model with generalized detection capabilities that could have required complex configuration options or technical knowledge to operate effectively.

5.5 Demo

5.5.1 Live detection for multiple packages

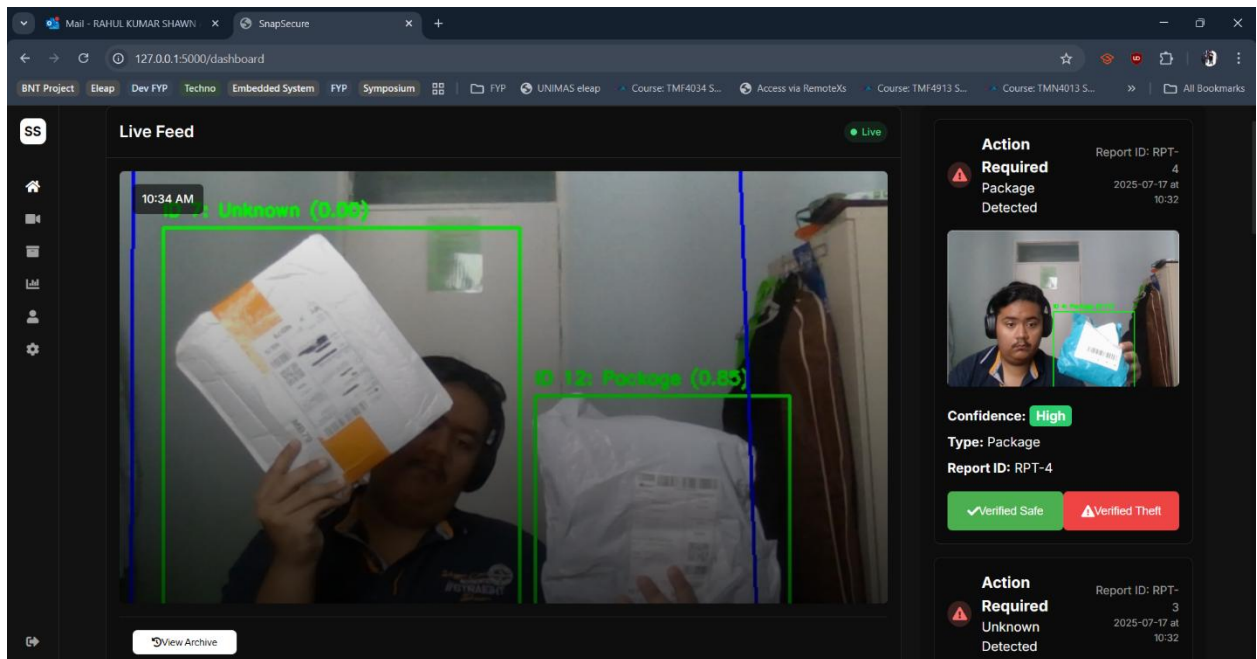


Figure 5.14 Double Package Detection

Based on Figure 5.14, there is a double detection which was detected in the Region Of Interest (ROI) by the system and successfully differentiated by the object detection pretrained model with the help of employing the Kalman Filter and Hungarian Algorithm which is a part of the Simple Online Realtime Tracking (SORT) algorithm.

5.5.2 Double Detection Alerting

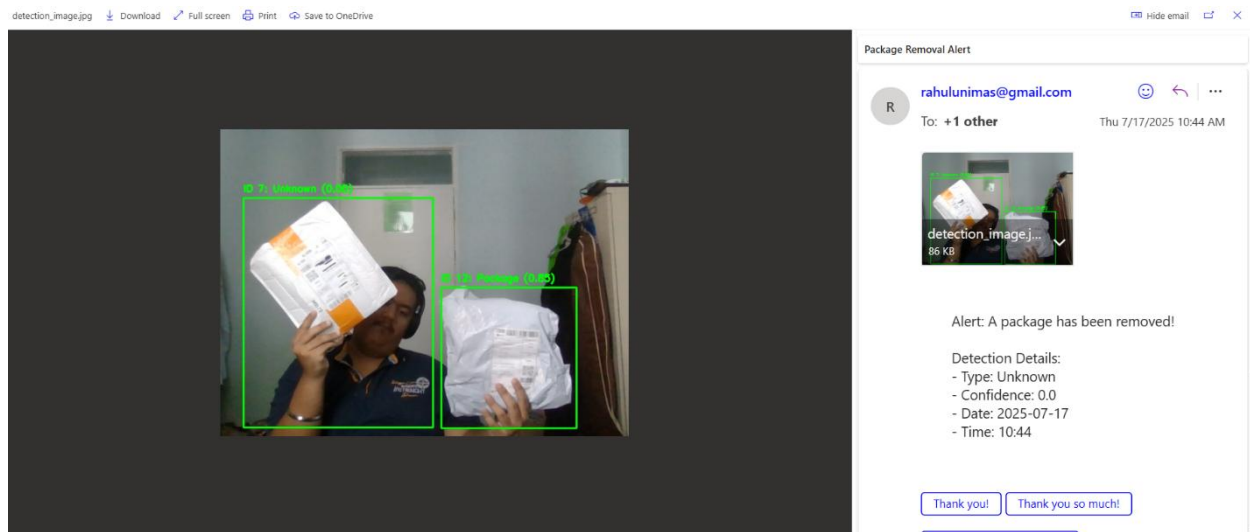


Figure 5.15 Double Detection Alert

Based on Figure 5.15, the employment of the Kalman Filter and Hungarian Algorithm which was able to differentiate between the 2 different packages although with similar colour features.

5.5.3 Slight Occlusion Detection

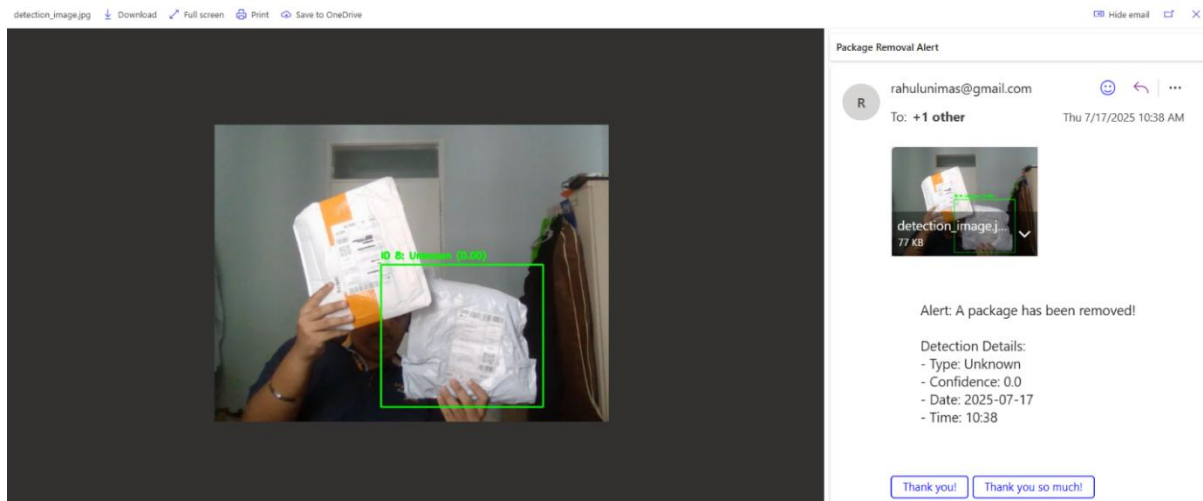


Figure 5.16 Slight Occlusion Detection

According to Figure 5.16, there was a slight occlusion between the package within the ROI with another package outside of the ROI, which was done on purpose to prove the ability of the system to differentiate between different objects with similar features.

5.6 Summary

The testing stage would be considered as one of the most crucial stages in a system's development life cycle, this stage would determine the final product's outcome both functional and reliability. The testing phase was divided into two categories which are the functional and non-functional testing. The functional testing focused on the software side, which included both the unit testing and integration testing. On the other hand, the non-functional testing comprised of reliability testing, performance testing and also the AI model Accuracy testing. Hence with this tests, and evaluation, it can be said that the system has undergone the testing stage with a yielded result on its outcome.

CHAPTER 6: CONCLUSION AND FUTURE WORKS

6.1 Introduction

The final chapter would provide a comprehensive conclusion of the proposed system which will be followed by a discussion of the potential future works that can be done to upscale and improve the application. Along with the proposed objectives would be compared to the outcome of the developed system to ensure that all the outlined objectives are achieved. This would also be the ground to highlight the limitations faced and what could be done in the future works to improve on it.\

6.2 Objective and Achievements

In the Planning phase of this project, 3 specific objectives were outlined in Chapter 1. It can be said that all of it has been achieved and even better improved during the development process. The specific objectives were to develop an AI-Powered Object Detection System for Package Monitoring; it was then improved to be able to have 3 different capabilities of detection where

1. Host Machine Camera utilization
2. Video Upload utilization
3. IP Camera utilization

Another objective highlighted would be to implement real-time alert for suspicious activities to track and log package deliveries and suspicious events, which is then followed with the objective to develop an accessible and user-friendly web-based interface based on Table 6.1 below.

Table 6.1 Details of Objectives and Achievements

Objective	Achievement
1. To develop an AI-Powered Object Detection System for Package Monitoring	Object Detection: Implemented computer vision using OpenCV and other libraries and integrated with Roboflow for AI model support. Created object detection pipeline capable of identifying packages Implemented Algorithm: Implemented Kalman filter for motion prediction and trajectory estimation. Utilized Hungarian algorithm for optimal assignment in object tracking (SORT). Developed filtering system to focus on relevant objects Challenges: Algorithm works as expected and gets the job done, but due to the processing power constraints, non-local model deployment (API call based), the playback speed of the video is significantly slow and requires optimization which can be achieved if used on a machine with a dedicated GPU.
2. To implement a real-time alert for suspicious activities to track and log package deliveries and suspicious events	Created event detection algorithms to identify suspicious behaviour Implemented real-time notification system: sends emails which contains information on the last time the package was detected by the system Developed logging functionality for all detected events: Events are logged by the system which will then be displayed in the Ulin both numerical (Reports) and graphical (Reports and Archive) manner for visualization of users. Added timestamp tracking for accurate event documentation Integrated visualization tools for highlighting detected objects Created reporting system with detailed event information. Added confidence level classification (Low/Medium/High) for alerts.
3. To develop an accessible and user-friendly web-based interface	Created responsive web interface using HTML/CSS: Implemented dashboard with real-time monitoring, archive system for historical detections, detailed reporting page with filtering capabilities, detection modal view for detailed inspection, implemented detection management (viewing/deletion), Added data visualization with charts for detection statistics, implemented user-friendly controls and filtering options, Created intuitive navigation between system components.

Table 6.2 Additional Features Implementation

Objective	Description	Achievement
Modularity	Containerized the system into portable images through the usage of Docker	Successfully containerized the system into a docker container which can be easily pulled, configured and deployed across AMD and ARM architectures
User Customizability	Allows users to customize within the system the area of interest that the detection should focus on	Successfully implemented Region of Interest (ROI) drawing feature that would allow the user to draw a region which the system should focus on for detection
Advanced Data Visualization with Interactive Charts	Provide users with visual representations of detection data to improve understanding and analysis.	Successfully implemented dynamic charts for visualization of detection statistics, confidence distributions, and temporal trends, enhancing data interpretation capabilities.
Confidence Level Classification System	Categorize detections based on confidence levels to help prioritize alerts and responses.	Successfully developed an automated system that classifies detections into high (>0.75), medium (>0.5), and low confidence bands with visual differentiation in the interface.
Advanced Search and Filtering Capabilities	Allow users to quickly locate specific detection events within large datasets.	Successfully implemented real-time search functionality with dynamic filtering across detection records, enabling users to efficiently find relevant information.
Data Sorting and Organization Tools	Enable customizable organization of detection data for improved analysis.	Successfully created sortable detection tables with multiple criteria options and toggle functionality for sort order, significantly improving data navigation.
User Interface Enhancements	Improve usability and space efficiency of the application interface.	Successfully developed collapsible UI sections, responsive design elements, and intuitive navigation between system components, enhancing overall user experience.
Detection Management System	Allow users to inspect and manage individual detection events.	Successfully created a system for viewing, inspecting, and archiving individual detection records with detailed property information.

Real-time System Status Indicators	Provide users with immediate feedback on system operations.	Successfully implemented visual indicators for system operational status, processing activities, and connection status for video streams, improving user awareness of system state.
Automated Detection Archiving System	Maintain an organized repository of historical detection events for future reference and analysis.	Successfully implemented an automated system that archives detection events with timestamps, images, and metadata, allowing users to review past activity patterns over extended periods.
Image Snapshot Capture and Export	Capture and save still images of detected objects for documentation and sharing.	Successfully implemented functionality to automatically capture high-quality image snapshots of detection events with timestamp overlay and export capabilities for external use in reports or communications.

Discussion on Achievements

The package monitoring system successfully achieves its core objectives through a sophisticated integration of AI technologies and practical security features. The AI-powered detection system employs advanced computer vision techniques including Kalman filtering for trajectory prediction and Hungarian algorithm for optimal tracking, enabling accurate package identification across multiple video source types with customizable region-of-interest monitoring. The real-time alert system transforms this detection capability into actionable security intelligence through SORT implementation for consistent object tracking, comprehensive event logging with precise timestamps, and a tiered confidence classification system that prioritizes notifications based on reliability. These technical capabilities are made accessible through a thoughtfully designed web interface featuring real-time monitoring dashboards, detailed reporting functionality, and intuitive visualization tools that bridge the gap between complex AI technology and practical usability for security personnel.

Beyond core functionality, the system implements numerous additional features that enhance its overall utility and user experience. Advanced data visualization through interactive charts provides intuitive presentation of detection statistics and temporal trends, while sophisticated search, filtering, and sorting capabilities enable efficient navigation of historical records. The system demonstrates attention to user workflow through collapsible UI elements, responsive design, and real-time status indicators. Technical enhancements include automated archiving of detection events with comprehensive metadata, customizable alert thresholds based on multiple parameters including confidence levels and dwell time, and high-quality image snapshot functionality with timestamp overlay and PDF export capabilities. These additional features collectively transform the system from a basic detection tool into a comprehensive security solution that supports both immediate monitoring needs and longer-term security analysis across various deployment scenarios.

6.3 Project Limitations and Constraints

1. High Computational Demand

The system faces significant computational challenges due to the simultaneous processing of multiple resource-intensive operations. Object detection algorithms require substantial CPU and GPU resources to analyse each video frame in real-time (with adaptable frame skipping), while object tracking with Kalman filtering and Hungarian algorithm implementations adds another layer of computational complexity. Concurrent database operations for storing and querying detection events further tax system resources. This computational burden results in potential performance bottlenecks, particularly when monitoring multiple video streams simultaneously or when running on hardware with limited specifications. The system must balance detection accuracy with processing speed, often requiring optimization techniques like frame skipping or resolution reduction to maintain acceptable performance levels.

2. Internet Dependent

The system's reliance on internet connectivity represents a critical vulnerability in its operational capability. API calls to Roboflow for object detection create a dependency on external services beyond the system's control, introducing potential points of failure. The use of MongoDB Atlas as a cloud-based database solution further compounds this dependency, as all data storage and retrieval operations require active internet connectivity. This architecture makes the system vulnerable to network outages, bandwidth limitations, and potential API rate limiting or service disruptions. In security-critical environments, this internet dependency introduces reliability concerns, as monitoring capabilities could be compromised during network issues, creating potential blind spots in surveillance coverage during connectivity interruptions.

3. Docker Limitations

While Docker containerization provides deployment consistency and isolation benefits, it introduces significant functional constraints for the system. The containerized environment lacks access permissions to host system hardware, preventing direct utilization of locally connected cameras. This limitation restricts video input options to only uploaded video files and IP camera streams, eliminating the possibility of using host machine webcams or directly connected USB cameras for monitoring. This constraint reduces deployment flexibility, particularly in scenarios where local camera usage would be preferable due to reliability, latency, or bandwidth considerations. The containerization approach creates a trade-off between deployment simplicity and input source flexibility that impacts the system's adaptability to different monitoring scenarios.

4. Model Detection Accuracy

The system's reliance on pre-trained, general-purpose object detection models from Roboflow introduces inherent limitations in detection accuracy and specificity. Without custom training or fine-tuning for the specific package types and environmental conditions relevant to the

deployment context, the model may struggle with accurate detection in challenging scenarios. Variations in lighting conditions, package appearances, partial occlusions, or unusual orientations can significantly reduce detection reliability. This limitation is particularly impactful for security applications where false negatives (missed packages) could have serious consequences. The inability to easily adapt the model to specific deployment environments or evolving package characteristics represents a significant constraint on the system's detection reliability across diverse real-world scenarios.

5. Hardware Constraints

The system's resource-intensive nature imposes substantial hardware requirements that limit deployment options and scalability. To effectively simulate real-life surveillance capabilities, the system ideally requires server-grade computing resources with dedicated GPU acceleration, significant RAM allocation, and substantial storage capacity for video processing and event archiving. These hardware demands translate to increased implementation costs and infrastructure requirements, restricting practical deployment to organizations with sufficient technical resources. The system's performance degrades significantly on consumer-grade hardware, particularly when handling multiple video streams or high-resolution inputs. This hardware dependency creates a barrier to adoption for smaller-scale implementations or residential use cases where server-grade infrastructure is impractical or cost-prohibitive.

6. Integrating Distinct API Framework

The system's architecture centered around external API calls for core detection functionality introduces long-term scalability and flexibility challenges. The tight coupling with Roboflow's specific API implementation creates a dependency that would require significant architectural changes if transitioning to locally trained models or alternative detection services becomes necessary. This integration approach limits the system's ability to evolve independently of the external API provider's development roadmap or pricing structure. Migrating to locally hosted

models would require substantial code refactoring, potential changes to the processing pipeline, and reconfiguration of the detection result handling logic. This architectural constraint reduces the system's long-term adaptability and creates potential vendor lock-in issues that could impact future development options as requirements evolve.

6.4 Future Works

1. Addressing High Computational Demand

- **Model Optimization Techniques:** Implement model quantization and pruning to reduce computational requirements without significant accuracy loss
- **Asynchronous Processing Pipeline:** Develop a multi-threaded architecture that separates video ingestion, detection, tracking, and database operations
- **Hardware Acceleration Framework:** Create a unified framework that automatically leverages available GPU/TPU resources or falls back to CPU optimization
- **Adaptive Resolution Processing:** Implement dynamic resolution scaling based on computational load and detection requirements
- **Selective Frame Processing:** Develop intelligent frame selection algorithms that analyze only frames with significant changes rather than every frame

2. Reducing Internet Dependency

- **Local Model Deployment:** Implement TensorFlow Lite or ONNX runtime support for running detection models locally without API calls
- **Hybrid Online/Offline Architecture:** Create a system that seamlessly switches between local and cloud processing based on connectivity
- **Local Database Fallback:** Implement a local database system (SQLite/MongoDB local) that synchronizes with Atlas when connectivity is restored
- **Edge Caching System:** Develop a caching mechanism for detection results and temporary storage during connectivity interruptions

- **Bandwidth Optimization:** Implement compression and selective data transmission to reduce dependence on high-bandwidth connections

3. Overcoming Docker Limitations

- **Device Pass-through Configuration:** Research and implement Docker configurations that enable secure camera access from containerized applications
- **WebRTC Camera Interface:** Develop a separate microservice that accesses host cameras and streams video to the containerized application
- **Unified Device Access Layer:** Create an abstraction layer that standardizes video input handling regardless of source (uploaded, IP, or local)
- **Cross-Container Device Sharing:** Implement a specialized container for device management that shares camera streams with the main application container
- **Native Deployment Option:** Develop a parallel deployment track for non-containerized installation when direct camera access is required

4. Improving Model Detection Accuracy

- **Custom Transfer Learning Pipeline:** Develop tools for easily fine-tuning the detection model on customer-specific package types and environments
- **Synthetic Data Generation:** Create a system to generate synthetic training data that represents diverse package scenarios and lighting conditions
- **Ensemble Detection Approach:** Implement multiple complementary detection models and use ensemble techniques to improve overall accuracy
- **Continuous Learning System:** Build a feedback loop where confirmed detections are used to incrementally improve the model over time
- **Domain-Specific Augmentation:** Implement specialized data augmentation techniques that simulate package-specific variations and environmental conditions

5. Addressing Hardware Constraints

- **Progressive Enhancement Architecture:** Design a tiered functionality system that scales features based on available hardware capabilities
- **Distributed Processing Framework:** Develop capability to distribute processing across multiple lower-powered devices instead of requiring a single powerful server
- **Cloud-Edge Hybrid Processing:** Create a flexible architecture that can offload intensive processing to cloud services when local resources are insufficient
- **Hardware-Specific Optimizations:** Build specialized code paths optimized for different hardware configurations (ARM, x86, with/without GPU)
- **Resource Monitoring and Adaptation:** Implement dynamic resource allocation that adjusts processing parameters based on current system load

6. Improving API Integration Flexibility

- **Detection API Abstraction Layer:** Develop a standardized interface that decouples the application from specific API implementations
- **Pluggable Model Architecture:** Create a modular system where detection engines can be swapped without affecting the rest of the application
- **Model Format Conversion Utilities:** Build tools to convert between different model formats to facilitate migration between frameworks
- **Local Model Serving:** Implement a local model serving infrastructure (TensorFlow Serving, ONNX Runtime) as an alternative to external APIs
- **Unified Result Schema:** Standardize detection result formats across different potential backends to simplify integration of new detection technologies

6.5 Conclusion

The AI-powered Package Monitoring System successfully fulfils its core objectives through a sophisticated integration of computer vision and security technologies, demonstrating technical excellence in both implementation and user experience design. By employing advanced object tracking algorithms including Kalman filtering and Hungarian assignment, alongside a customizable Region of Interest framework and multi-source video input support, the system delivers robust package detection capabilities with confidence-based reliability assessment. The real-time alert system effectively transforms detection data into actionable security intelligence through comprehensive event logging, SORT implementation, containerization of the system for modularity and deployment ease and notification. These technical capabilities are made accessible through a thoughtfully designed web interface featuring intuitive visualizations, filtering tools, and comprehensive reporting functions. While acknowledging limitations in computational demands, internet dependency, containerization restrictions, and model accuracy, the system represents a significant achievement in security monitoring technology that successfully balances sophisticated AI capabilities with practical usability, offering a solid foundation for further enhancements to address the identified constraints through the proposed future development paths.

REFERENCES

- Bahrampour, S., Ramakrishnan, N., Schott, L., & Shah, M. (2015). Comparative study of Deep learning Software Frameworks. *arXiv (Cornell University)*.
<https://doi.org/10.48550/arxiv.1511.06435>
- Bhattacharya, D. (2023). *VS Code: Tips and tricks for beginners* | MDN Blog. MDN Web Docs. <https://developer.mozilla.org/en-US/blog/vs-code-tips-tricks/>
- Commission Factory. (2023, January 13). *Malaysia eCommerce statistics and Trends in 2023*. <https://blog.commissionfactory.com/ecommerce-marketing/malaysia-ecommerce-statistics#:~:text=At%20the%20same%20time%2C%20the,at%20the%20start%20of%202023.>
- Contributors, S. W. (n.d.). *Spyder | The Python IDE that scientists and data analysts deserve*. Spyder IDE. <https://www.spyder-ide.org/>
- Crockett, S. (2024, January 8). Deep Learning Frameworks Compared: TensorFlow, PyTorch, Keras, and More. *Medium*. <https://medium.com/@shomariccrockett/deep-learning-frameworks-compared-tensorflow-pytorch-keras-and-more-09206f791eb9>
- Deep Sentinel. (2024a, January 5). *Security Cameras That Stop Package Theft - Deep Sentinel*. <https://www.deepsentinel.com/package-theft/>
- Deep Sentinel. (2024b, October 7). *Third-Party Camera Integration - Mobile App - Deep Sentinel*. <https://www.deepsentinel.com/third-party-camera-integration/>
- Docker Inc. (n.d.). Docker Documentation. <https://docs.docker.com/desktop/features/wsl/>
- Dupont, M. (2024). *Roboflow: Democratizing Computer Vision - A Beginner's Guide*. Labelvisor. <https://www.labelvisor.com/roboflow-democratizing-computer-vision-a-beginners-guide/>
- Ebook - Real-Time Analytics on MonGoDB: The Ultimate Guide*. (n.d.). Rockset. <https://rockset.com/real-time-analytics-mongodb-ebook/>

- FocusM. (2023). *Parcel Thief Caught on CCTV Nonchalantly Returning Items Not to His Liking*. <https://focusmalaysia.my/parcel-thief-caught-on-cctv-nonchalantly-returning-items-not-to-his-liking/>
- GeeksforGeeks. (2024a). *What is DFD(Data Flow Diagram)?* GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-dfddata-flow-diagram/>
- GeeksforGeeks. (2024b, May 6). *Learn Web Development Basics with HTML CSS and JavaScript*. GeeksforGeeks. <https://www.geeksforgeeks.org/learn-web-development-basics-with-html-css-and-javascript/>
- GeeksforGeeks. (2024c, September 26). *What is Figma?* GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-figma/>
- Global, I. (2024, October 3). *The role of MongoDB in AI and machine learning Projects | ICERT Global*. [icertglobal.com. https://www.icertglobal.com/the-role-of-mongodb-in-ai-and-machine-learning-projects-blog/detail#:~:text=MongoDB%20can%20handle%20unstructured%20data,AI%20models'%20growing%20data%20needs.](https://www.icertglobal.com/the-role-of-mongodb-in-ai-and-machine-learning-projects-blog/detail#:~:text=MongoDB%20can%20handle%20unstructured%20data,AI%20models'%20growing%20data%20needs.)
- Hema, V., Thota, S., Kumar, S. N., Padmaja, C., Krishna, C. B. R., & Mahender, K. (2020). *Scrum: an effective software development agile tool. IOP Conference Series Materials Science and Engineering*, 981(2), 022060. <https://doi.org/10.1088/1757-899x/981/2/022060>
- Hicks, M., Stickle, B., & Harms, J. (2021). Assessing the fear of package theft. *American Journal of Criminal Justice*, 47(1), 3–22. <https://doi.org/10.1007/s12103-020-09600-x>
- Hidalgo, E. S. (2019). Adapting the scrum framework for agile project management in science: case study of a distributed research initiative. *Heliyon*, 5(3), e01447. <https://doi.org/10.1016/j.heliyon.2019.e01447>

How to make a web application using Flask in Python 3. (n.d.). DigitalOcean.

<https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3>

Hsu, H., Yuan, X., Zhu, B., Cheng, Z., & Chen, L. (2022). Package Theft Detection from Smart Home Security Cameras. *2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 156, 1–4.

<https://doi.org/10.1109/icmew56448.2022.9859522>

IBM. (2024). Entity relationship diagram. *IBM*. <https://www.ibm.com/think/topics/entity-relationship-diagram>

IoT Community. (2024, August 28). *IoT Community - Worlds Largest Internet of Things Community*. IoT Community® - Internet of Things Community®.

<https://iotcommunity.net/>

Kaur, N. (2024, March 6). A Statistical Analysis of E-Commerce in Malaysia. *WeCorporate Global Consultancy*. <https://wecorporate.com.my/blog/statistical-analysis-ecommerce-in-malaysia/>

Kosmo. (2021, July 12). ‘Ada kesan bungkus dibuka sebelum ditampal semula.’ *Kosmo Digital*. <https://www.kosmo.com.my/2021/07/13/ada-kesan-bungkusan-dibuka-sebelum-ditampal-semula/>

Laoyan, S. (2024, February 2). What Is Agile Methodology? (A Beginner’s Guide) [2024] • Asana. *Asana*. <https://asana.com/resources/agile-methodology>

Ltd, H. D. T. C. (n.d.). *Veesion run*. HikVision Technical Partner Portal.

<https://tpp.hikvision.com/solution/SolutionDetail?Id=1&v=en>

Mattwojo. (2023a). *Comparing WSL versions*. Microsoft Learn.

<https://learn.microsoft.com/en-us/windows/wsl/compare-versions>

- Mattwojo. (2023b). *What is Windows Subsystem for Linux*. Microsoft Learn.
<https://learn.microsoft.com/en-us/windows/wsl/about>
- Miu, M. (2015). Events in Scrum - Magda Miu - Medium. *Medium*.
<https://magdamiu.medium.com/events-in-scrum-a7cc0dfe69ea>
- MongoDB. (n.d.). *MongoDB for artificial Intelligence*. Retrieved November 25, 2024, from
<https://www.mongodb.com/solutions/use-cases/artificial-intelligence>
- Monji-Azad, S., Hesser, J., & Löw, N. (2023). A review of non-rigid transformations and learning-based 3D point cloud registration methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196, 58–72.
<https://doi.org/10.1016/j.isprsjprs.2022.12.023>
- Mool, J. A. (2023, December 5). Stealing parcels: Five jailed and fined. 2024.
<https://www.dailyexpress.com.my/news/224745/stealing-parcels-five-jailed-and-fined/>
- Nitnaware, S., & Bagde, A. (2022). Machine Learning based Object Detection with Theft Alert System. *2022 6th International Conference on Electronics, Communication and Aerospace Technology*. <https://doi.org/10.1109/iceca55336.2022.10009061>
- Object Detection From TF2 Checkpoint — TensorFlow 2 Object Detection API tutorial documentation*. (n.d.). https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/auto_examples/plot_object_detection_checkpoint.html#sphx-glr-auto-examples-plot-object-detection-checkpoint-py
- Official HP® Support*. (n.d.). Retrieved November 26, 2024, from
<https://support.hp.com/my-en>
- Opala, M., Opala, M., & Netguru. (2022, June 15). Deep Learning Frameworks Comparison – TensorFlow, Keras, MXNet and More. *Netguru*.
<https://www.netguru.com/blog/deep-learning-frameworks-comparison>

- P, M. K. D., J, A. B., S, V. T., N, S., B, R., & Sharma, S. (2023). TensorFlow Implementation on Analysis of Unusual Behavior in Crowd Using Deep Learning. *2023 International Conference on Recent Advances in Science & Engineering Technology (ICRASET)*. <https://doi.org/10.1109/icraset59632.2023.10420324>
- Package Detection | CVEDIA | AI Video Analytics for any hardware*. (n.d.). <https://www.cvedia.com/package-detection>
- Paley, A., Urma, R., & Lawrence, N. D. (2022). Challenges in Deploying Machine Learning: A survey of case studies. *ACM Computing Surveys*, 55(6), 1–29. <https://doi.org/10.1145/3533378>
- Penal Code Section 379 (Malaysia)*. (n.d.). BurgieLaw. https://www.burgielaw.com/resources/act?act_title=Penal+Code&ion=379
- Potrimba, P. (2024, April 9). *What is R-CNN?* Roboflow Blog. <https://blog.roboflow.com/what-is-r-cnn/>
- Resilience Library | Southeast Asia Resources*. (n.d.). https://ccid.rmp.gov.my/Laws/Act_574_Panel_Code_Malaysia.pdf
- Roth, R. E., Hart, D., Mead, R., & Quinn, C. (2018). Wireframing for interactive & web-based geographic visualization: designing the NOAA Lake Level Viewer. *Cartography and Geographic Information Science*, 44(4), 338–357. <https://doi.org/10.1080/15230406.2016.1171166>
- Salem, A. B. (2024, June 17). Scrum Artifacts: Complete Guide - Scrum-Master·Org. *Scrum Master #1*. <https://scrum-master.org/en/scrum-artifacts-complete-guide/>
- Salimi, S. (n.d.-a). *Backlog*. Agile Academy. <https://www.agile-academy.com/en/agile-dictionary/backlog/>
- Salimi, S. (n.d.-b). *Events and artifacts of Scrum*. Agile Academy. <https://www.agile-academy.com/en/foundations/events-and-artifacts-of-scrum/>

- Sassa, A. C., De Almeida, I. A., Pereira, T. N. F., & De Oliveira, M. S. (2023). Scrum: A Systematic Literature Review. *International Journal of Advanced Computer Science and Applications*, 14(4). <https://doi.org/10.14569/ijacsa.2023.0140420>
- Setting up Areas and Tripwires - CVEDIA-RT*. (n.d.).
https://docs.cvedia.com/solutions/securt/areas_tripwires_setup.html
- Shadiqe, J. (2024, June 30). #NSTviral: Police investigating 'porch pirate' who fled with a parcel in a luxury car. *NST Online*. <https://www.nst.com.my/news/nst-viral/2024/06/1070463/nstviral-police-investigating-porch-pirate-who-fled-parcel-luxury-car>
- The Scrum Framework Poster*. (n.d.). Scrum.org. <https://www.scrum.org/resources/scrum-framework-poster>
- Ubah, K. (2024, October 15). *Learn Web Development Basics – HTML, CSS, and JavaScript Explained for Beginners*. freeCodeCamp.org.
<https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/>
- Urooj, W. (2021, December 11). What is Python Spyder IDE and How to use it? - Edureka - Medium. *Medium*. <https://medium.com/edureka/spyder-ide-2a91caac4e46>
- Veesion. (n.d.). *Veesion | AI surveillance software*. <https://veesion.io/en/>
- Verwijns, C., & Russo, D. (2022). A theory of scrum team effectiveness. *ACM Transactions on Software Engineering and Methodology*, 32(3), 1–51.
<https://doi.org/10.1145/3571849>
- Vigderman, A., & Turner, G. (2024, April 22). *Deep Sentinel Review*. Security.org.
<https://www.security.org/home-security-systems/deep-sentinel/review/>
- Wanyoike, M., & Wanyoike, M. (2024, November 15). *Visual Studio Code: A Power User's Guide*. SitePoint. <https://www.sitepoint.com/visual-studio-code-power-user-guide/>

What is Docker? (2024, September 10). Docker Documentation. [https://docs.docker.com/get-started/docker-](https://docs.docker.com/get-started/docker-overview/#:~:text=Docker%20is%20an%20open%20platform,and%20running%20it%20in%20production.)

[overview/#:~:text=Docker%20is%20an%20open%20platform,and%20running%20it%20in%20production.](https://docs.docker.com/get-started/docker-overview/#:~:text=Docker%20is%20an%20open%20platform,and%20running%20it%20in%20production.)

What is Figma? (n.d.). Figma Learn - Help Center. Retrieved November 25, 2024, from <https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma>

Yuan, L., Qu, Z., Zhao, Y., Zhang, H., & Nian, Q. (2017). A convolutional neural network based on TensorFlow for face recognition. *IEEE International Conference on Information Science & Engineering.*, 525–529.

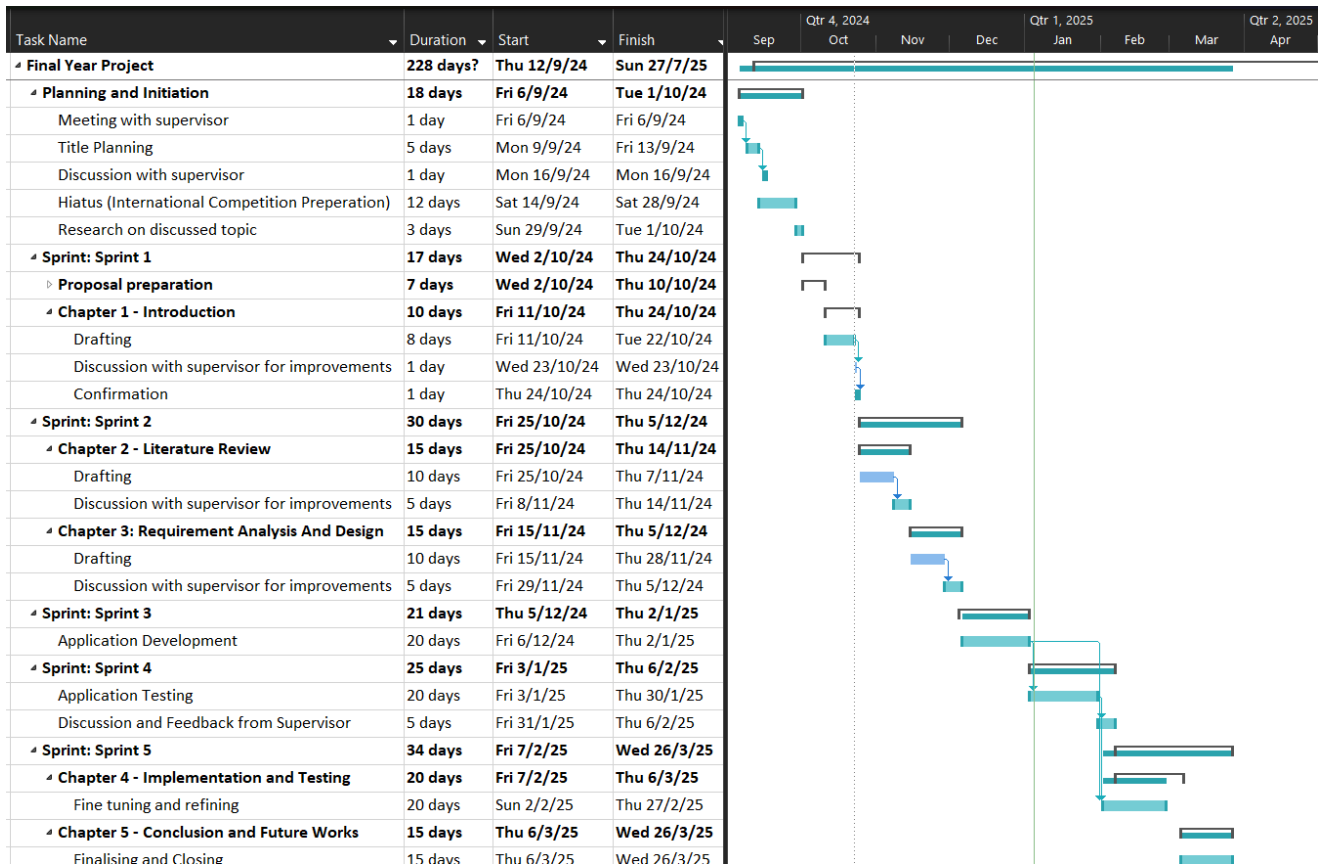
<https://doi.org/10.1109/iaeac.2017.8054070>

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object Detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3), 257–276.

<https://doi.org/10.1109/jproc.2023.3238524>

APPENDICES

Appendix A



Appendix B

Final Year Project: Survey on a AI-Powered Package Detection System to Reduce Porch Piracy and Improve Package Safety / *Projek Tahun Akhir: Tinjauan Mengenai Sistem Pintar untuk Pengesanan Bungkus bagi Mengurangkan Kecurian di Beranda dan Meningkatkan Keselamatan Bungkus.*

Dear Respected respondents,

Greetings and salutations from Rahul Kumar Shawn A/L Rajesh Kumar. I am currently in my final year as an undergraduate student, specializing in Network Computing from the Faculty of Computer Science and Information Technology (FCSIT) at Universiti Malaysia Sarawak (UNIMAS). As an crucial part for my Final Year Project (FYP), I am conducting a survey to gather essential data and insights to assess thoroughly on the AI-Powered Package Detection System to Reduce Porch Piracy and Improve Package Safety. The main objective of this project is to design and develop a smart system monitoring system to reduce porch piracy and improve package safety. Other objectives include:

1. To develop an AI-Powered Object Detection System for Package Monitoring.
2. To implement a real-time alert for suspicious activities to track and log package deliveries and suspicious events.
3. To develop an accessible and user-friendly web-based interface.

Let me assure you that all information collected through this survey will be used exclusively for my Final Year Project. Your responses will remain strictly confidential, and no data will be shared with any third party. If you have any inquiries or need further clarification, please feel free to contact me through my WhatsApp via 011-37820862 or through my email at 82102@siswa.unimas.my.

I would like to take this opportunity to extend my gratitude and acknowledgement for your valuable time and cooperation. The participation in this survey will be proven as a vital contribution to the success of this project.

sincere regards,
Rahul Kumar Shawn A/L Rajesh Kumar.

Kepada Responden yang Dihormati,

Salam sejahtera dan salam hormat, nama saya Rahul Kumar Shawn A/L Rajesh Kumar. Saya kini berada di tahun akhir sebagai pelajar sarjana muda, dalam pengkhususan Pengkomputeran Rangkaian di Fakulti Sains Komputer dan Teknologi Maklumat (FSKTM), Universiti Malaysia Sarawak (UNIMAS). Sebagai sebahagian penting daripada Projek Tahun Akhir (FYP), saya sedang menjalankan tinjauan untuk mengumpul data dan pandangan penting bagi menilai secara mendalam tentang Sistem Pintar untuk Pengesanan Bungkusan bagi Mengurangkan Kecurian di Beranda dan Meningkatkan Keselamatan Bungkusan.

Objektif utama projek ini adalah untuk mereka bentuk dan membangunkan sistem pemantauan pintar bagi mengurangkan kecurian di beranda dan meningkatkan keselamatan bungkusan.

Objektif tambahan termasuk:

- 1. Membangunkan Sistem Pengesanan Objek Berkuasa AI untuk Pemantauan Bungkusan.*
- 2. Melaksanakan sistem amaran masa nyata untuk aktiviti mencurigakan bagi menjejaki dan merekod penghantaran bungkusan serta kejadian mencurigakan.*
- 3. Membangunkan laman berasaskan web yang boleh diakses dan mesra pengguna.*

Saya ingin memberi jaminan bahawa semua maklumat yang dikumpul melalui tinjauan ini akan digunakan secara eksklusif untuk Projek Tahun Akhir saya. Maklum balas anda akan kekal sulit sepenuhnya, dan tiada data akan dikongsi dengan mana-mana pihak ketiga. Jika anda mempunyai sebarang pertanyaan atau memerlukan penjelasan lanjut, sila hubungi saya melalui WhatsApp di 011-37820862 atau melalui e-mel di 82102@siswa.unimas.my.

Saya ingin mengambil peluang ini untuk mengucapkan setinggi-tinggi penghargaan atas masa dan kerjasama anda yang amat berharga. Penyertaan anda dalam tinjauan ini akan menjadi sumbangan penting kepada kejayaan projek ini.

Terima kasih.

kiriman Ikhlas,

Rahul Kumar Shawn A/L Rajesh Kumar

Section A - Respondents Demographic

Information Regarding Respondents information

Age *

- ☐ 16 - 20
- ☐ 21 - 25
- ☐ 26 - 30
- ☐ 31 - 35
- ☐ 36 - 40
- ☐ 41 - 45
- ☐ 46 - 50
- ☐ 51 and above

Gender *

- ☐ Male
- ☐ Female

State *

Primary Residence Type: *

☐ Apartment/Condominium

☐ Terrace/Link House

☐ Semi-Detached House

☐ Detached/Bungalow

☐ Other: _____

Frequency of Receiving Packages (e.g., via courier services) *

☐ Rarely (1–2 times a month)

☐ Occasionally (3–5 times a month)

☐ Frequently (6+ times a month)

Section B: Current Concerns About Package Theft

Have you or someone you know experienced package theft? *

- ☐ Yes.
- ☐ No

How concerned are you about package theft in your area? *

- ☐ Not concerned
- ☐ Slightly concerned
- ☐ Moderately concerned
- ☐ Very concerned
- ☐ Extremely concerned

What measures do you currently use to protect your packages? (Select all that apply) *

- ☐ Security cameras
- ☐ Package lockers
- ☐ Having someone at home to receive deliveries
- ☐ Scheduling deliveries for when I am home
- ☐ None
- ☐ Other: _____

Section C: Preferences for a Smart Package Detection System

What features would you find most useful in a smart package detection system? *
(Select up to 3)

- ☐ Real-time alerts about package delivery or removal
- ☐ Facial recognition to identify known individuals
- ☐ Motion detection for suspicious activity
- ☐ AI-based package recognition
- ☐ Live video monitoring via web
- ☐ Secure storage options for packages
- ☐ Other: _____

Importance of Features in a Smart Package Detection System. *

How important are the following features in a smart package detection system?
(1 = Not Important, 5 = Extremely Important)

- Real-time alerts about package delivery or removal
- Facial recognition to identify known individuals
- Motion detection for suspicious activity
- AI-based package recognition
- Live video monitoring via web

	1	2	3	4	5	
Not Important	☐	☐	☐	☐	☐	Extremely Important

Willingness to Use a Smart Package Detection System *

How likely are you to adopt a smart package detection system if it addresses your package safety concerns?
(1 = Very Unlikely, 5 = Very Likely)

	1	2	3	4	5	
Very Unlikely	☐	☐	☐	☐	☐	Very Likely

Affordability vs. Feature Balance ★

How satisfied would you be with a system that balances affordability and advanced features?

(1 = Not Satisfied at All, 5 = Extremely Satisfied)

	1	2	3	4	5	
Not Satisfied at All	☐	☐	☐	☐	☐	Extremely Satisfied

Perceived Effectiveness of a Smart Package Detection System ★

How effective do you believe a smart package detection system would be in reducing package theft?

(1 = Not Effective, 5 = Very Effective)

	1	2	3	4	5	
Not Effective	☐	☐	☐	☐	☐	Very Effective

Section D: User Experience and Feedback

What challenges or concerns might prevent you from using a smart package detection system? *

- ☐ High cost
- ☐ Complexity of installation
- ☐ Concerns about privacy and data security
- ☐ Limited usefulness or reliability
- ☐ Other: _____

Would you prefer a system that integrates with your existing security setup (e.g., cameras, doorbell systems)? *

- ☐ Yes.
- ☐ No
- ☐ Not sure.

Do you have any additional features or suggestions for such a system? (Optional)
(Open-ended question)

Your answer _____

That is all! Thank you!



Final Year Project: Survey on User Feedback for an AI-Powered Package Detection System to Reduce Porch Piracy and Improve Package Safety / *Projek Tahun Akhir: Tinjauan Maklum Balas Pengguna terhadap Sistem Pengesanan Bungkus Berkuasa AI bagi Mengurangkan Kecurian di Beranda dan Meningkatkan Keselamatan Bungkus*

Dear Respected Respondents,

Warm greetings. My name is Rahul Kumar Shawn A/L Rajesh Kumar, and I am currently in my final year of undergraduate studies, specializing in Network Computing at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS).

As part of my Final Year Project (FYP), I am conducting a **User Acceptance Testing (UAT)** survey to collect valuable feedback on the usability, performance, and practicality of an **AI-Powered Package Detection System**. This system is designed to reduce porch piracy and improve package safety using smart surveillance and real-time alerts.

Your feedback will be used to assess and improve the system based on real-world user needs. The survey is divided into the following five sections:

1. **Section A: Respondent Demographics**
2. **Section B: Usability and Interface**
3. **Section C: Functionality**
4. **Section D: Performance**
5. **Section E: Feedback and Recommendation**

All responses will be used solely for academic purposes. Your answers will remain confidential and will not be shared with any third parties.

If you have any questions or need further clarification, feel free to contact me via WhatsApp at 011-37820862 or email at 82102@siswa.unimas.my.

Thank you sincerely for your time and cooperation. Your participation will be a valuable contribution to the success of this project.

Sincerely,

Rahul Kumar Shawn A/L Rajesh Kumar

Kepada Responden yang Dihormati,

Salam sejahtera. Nama saya Rahul Kumar Shawn A/L Rajesh Kumar, pelajar tahun akhir dalam bidang Pengkomputeran Rangkaian di Fakulti Sains Komputer dan Teknologi Maklumat (FSKTM), Universiti Malaysia Sarawak (UNIMAS).

Sebagai sebahagian daripada Projek Tahun Akhir (FYP), saya sedang menjalankan satu tinjauan **Ujian Penerimaan Pengguna (UAT)** untuk mengumpul maklum balas berkaitan sistem **Pengesanan Bungkusan Berkuasa AI** yang dibangunkan bagi mengurangkan kecurian di beranda dan meningkatkan keselamatan bungkusan menggunakan pemantauan pintar dan amaran masa nyata.

Maklum balas anda akan digunakan untuk menilai dan menambah baik sistem berdasarkan keperluan pengguna sebenar. Tinjauan ini dibahagikan kepada lima (5) bahagian utama.

1. **Bahagian A: Demografi Responden**
2. **Bahagian B: Kebolehgunaan dan Antara Muka**
3. **Bahagian C: Fungsi Sistem**
4. **Bahagian D: Prestasi Sistem**
5. **Bahagian E: Maklum Balas dan Cadangan**

Segala maklumat yang dikumpul adalah untuk tujuan akademik sahaja dan akan dirahsiakan sepenuhnya tanpa dikongsi kepada mana-mana pihak ketiga.

Jika anda mempunyai sebarang pertanyaan, sila hubungi saya melalui WhatsApp di 011-37820862 atau emel di 82102@siswa.unimas.my.

Terima kasih atas masa dan kerjasama anda. Penyertaan anda amat dihargai dan menjadi sumbangan penting kepada kejayaan projek ini.

Ikhlas,

Rahul Kumar Shawn A/L Rajesh Kumar

Section A - Respondents Demographic

Information Regarding Respondents information

Age *

- ☐ 16 - 20
- ☐ 21 - 25
- ☐ 26 - 30
- ☐ 31 - 35
- ☐ 36 - 40
- ☐ 41 - 45
- ☐ 46 - 50
- ☐ 51 and above

Gender *

- ☐ Male
- ☐ Female

State *

Choose ▼

Primary Residence Type: *

☐ Apartment/Condominium

☐ Terrace/Link House

☐ Semi-Detached House

☐ Detached/Bungalow

☐ Other: _____

Frequency of Receiving Packages (e.g., via courier services) *

☐ Rarely (1–2 times a month)

☐ Occasionally (3–5 times a month)

☐ Frequently (6+ times a month)

Section B: Usability and Interface

How easy was it to navigate through the web application? *

	1	2	3	4	5	
Very Difficult	☐	☐	☐	☐	☐	Very Easy

Are the instructions in the web application clear and understandable? *

	1	2	3	4	5	
Very Unclear	☐	☐	☐	☐	☐	Very Clear

How easy it is to choose between different video sources? *

	1	2	3	4	5	
Very Difficult	☐	☐	☐	☐	☐	Very Easy

Section C: Functionality

Does the system detect always detect the package within the Region of Interest (ROI)? *

- ☐ Always
- ☐ Majority of the time
- ☐ Fair number of times
- ☐ Rarely
- ☐ Never

Is the detection accurate on the type of package? *

	1	2	3	4	5	
Very Inaccurate	☐	☐	☐	☐	☐	Very Accurate

How effective was the alerting of the package removal? *

	1	2	3	4	5	
Very ineffective	☐	☐	☐	☐	☐	Very Effective

Section D: Performance

How would you rate the system's response for real-time package theft detection? *

	1	2	3	4	5	
Very Slow	☐	☐	☐	☐	☐	Very Fast

How would you rate the web application's response time for loading each page *

	1	2	3	4	5	
Very Slow	☐	☐	☐	☐	☐	Very Fast

Did you experience any issues (e.g. web application crashing, slow notification, slow processing)

- ☐ Never
- ☐ Rarely
- ☐ Fair number of times
- ☐ Majority of the time
- ☐ Always

Section E: Feedback and Recommendation

How useful do you presume the web application is for helping you keep track of your package deliveries and safety? *

	1	2	3	4	5	
Not Useful At All	☐	☐	☐	☐	☐	Very Useful

Are the real-time alerts useful for keeping track of your package removal? *

	1	2	3	4	5	
Not Useful At All	☐	☐	☐	☐	☐	Very Useful

Would you consider using this web application with your already existing video/camera/cctv infrastructure? *

	1	2	3	4	5	
Very Unlikely	☐	☐	☐	☐	☐	Very Likely

How likely would you recommend this web application to other homeowners? *

	1	2	3	4	5	
Very Unlikely	☐	☐	☐	☐	☐	Very Likely

That is all! Thank you!

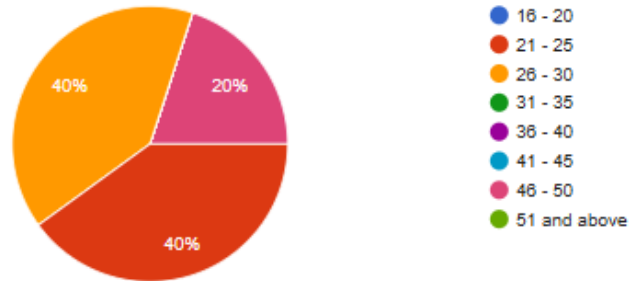


Section A - Respondents Demographic

Age

15 responses

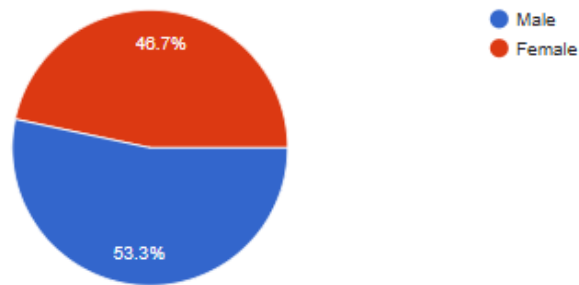
 [Copy chart](#)



Gender

15 responses

 [Copy chart](#)



State

15 responses

[Copy chart](#)

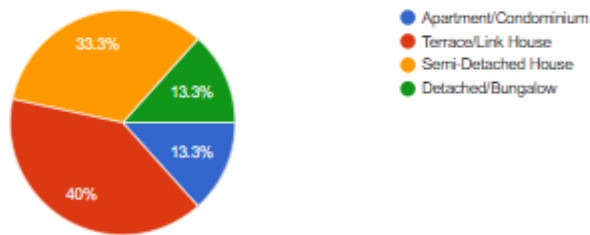


▲ 1/2 ▼

Primary Residence Type:

15 responses

[Copy chart](#)



Frequency of Receiving Packages (e.g., via courier services)

15 responses

[Copy chart](#)

