



Faculty of Computer Science and Information Technology

***BLOCKCHAIN-BASED TRAFFIC VIOLATION LOGGING AND
PENALTY MANAGEMENT SYSTEM***

Richard Jong Wei Ze

Bachelor of Computer Science with Honours (Network Computing)

2025

**BLOCKCHAIN-BASED TRAFFIC VIOLATION LOGGING AND PENALTY
MANAGEMENT SYSTEM**

RICHARD JONG WEI ZE

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 PENGURUSAN PELANGGARAN TRAFIK DAN PENALTI
BERASASKAN BLOCKCHAIN**

RICHARD JONG WEI ZE

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 hereby declare that this project report titled “**Blockchain-based Traffic Violation Logging and Penalty Management System**” is my original work in partial fulfilment of the requirements for the degree of Bachelor of Computer Science with Honours (Network Computing). This work has been carried out under the supervision of Associate Professor Dr. Johari bin Abdullah. All information derived from other sources has been duly acknowledged in this report.



.....

RICHARD JONG WEI ZE (81049)

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

UNIVERSITI MALAYSIA SARAWAK

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my supervisor, Associate Professor Dr. Johari bin Abdullah, for his invaluable guidance, continuous support, and informative feedback through this final year project. His expertise and insights have been crucial in helping me complete this project successfully. I would also like to thank Associate Professor Ts. Dr. Adnan Shahid Khan, who is the examiner for this project, for providing valuable feedback that helped improve the quality of my work. In addition, I am grateful to the program coordinator of final year project, Professor Dr. Wang Yin Chai, for preparing the information and guidance during the class. I extend my appreciation to the Faculty of Computer Science and Information Technology (FCSIT) at Universiti Malaysia Sarawak (UNIMAS) for the support and resources that made this project possible. I am also thankful for the respondents, who provided their feedback in my questionnaire. Furthermore, special appreciation goes to my family for their continuous support, encouragement, and understanding. Their constant motivation has been a driving force behind my journey in completing this project. Lastly, I would like to thank my friends and coursemates who have supported me through this journey, sharing knowledge, and providing moral support when needed.

ABSTRACT

Traffic violations have been a persistent challenge in developing countries, posing public threats such as road accidents, traffic congestion, and decreased quality of life for citizens. Current traffic management systems suffer from critical limitations including data manipulation, lack of transparency, inefficient penalty tracking, and corruption in enforcement. Traditional centralized systems are vulnerable to unauthorized alterations, which has led to significant revenue losses and decreased public trust. This project proposes a Blockchain-Based Traffic Violation Logging and Penalty Management System to address these challenges by leveraging the transparency, security, and decentralization offered by blockchain technology. The system ensures all traffic violations are logged in an immutable blockchain ledger, which prevents data tampering and maintains record integrity, while smart contracts automate penalty issuance and tracking to reduce human intervention and corruption risks. Through a unified web-based platform, the system provides real-time access to violation data for both drivers and authorities, ensuring seamless user engagement. Detailed testing validated the system's effectiveness through functional testing including unit testing of individual components, and non-functional testing consisting reliability, compatibility, and user acceptance testing, demonstrating that the system meets stability requirements and end-user expectations. Future enhancements include implementing persistent wallet and certificate storage for production-ready deployment, and enhancing blockchain network scalability to support higher transaction throughput for real-world traffic management scenarios. By implementing these technological innovations and addressing inefficiencies in traditional enforcement systems, this solution fosters accountability and trust among stakeholders. In the long run, it contributes to safer roads, reduced congestion, and improved quality of life for citizens in developing countries.

ABSTRAK

Pelanggaran lalu lintas telah menjadi cabaran berterusan di negara-negara yang sedang membangun, menimbulkan ancaman awam seperti kemalangan jalan raya, kesesakan lalu lintas, dan penurunan kualiti hidup bagi warganegara. Sistem pengurusan traffic semasa mengalami had kritikal termasuk manipulasi data, kekurangan ketelusan, penjejakan penalti yang tidak cekap dan rasuah dalam penguatkuasaan. Sistem berpusat tradisional terdedah kepada perubahan yang tidak dibenarkan, yang telah menyebabkan kerugian hasil yang ketara dan mengurangkan kepercayaan orang ramai. Projek ini mencadangkan “Sistem Pengurusan Pelanggaran Trafik dan Penalti Berasaskan Blockchain” untuk menangani cabaran ini dengan memanfaatkan ketelusan, keselamatan, dan desentralisasi yang ditawarkan oleh teknologi rantai blok. Sistem ini memastikan semua pelanggaran lalu lintas dilog masuk ke dalam lejar blockchain yang tidak boleh diubah, yang menghalang gangguan data dan mengekalkan integriti rekod, manakala kontrak pintar mengautomasikan pengeluaran dan penjejakan penalti untuk mengurangkan campur tangan manusia dan risiko rasuah. Melalui platform berasaskan web bersatu, sistem menyediakan akses masa nyata kepada data pelanggaran untuk kedua-dua pemandu dan pihak berkuasa, memastikan penglibatan pengguna yang lancar. Ujian terperinci mengesahkan keberkesanan sistem melalui pengujian fungsian termasuk ujian unit komponen individu, dan pengujian bukan fungsian yang terdiri daripada ujian kebolehpercayaan, keserasian dan penerimaan pengguna, menunjukkan bahawa sistem memenuhi keperluan kestabilan dan jangkaan pengguna akhir. Penambahbaikan pada masa hadapan termasuk melaksanakan storan dompet dan sijil yang berterusan untuk penggunaan sedia pengeluaran, dan meningkatkan kebolehskalaan rangkaian rantai blok untuk menyokong pemprosesan transaksi yang lebih tinggi untuk senario pengurusan trafik dunia sebenar. Dengan melaksanakan inovasi teknologi ini dan menangani ketidakcekapan dalam sistem penguatkuasaan tradisional, penyelesaian ini memupuk akauntabiliti dan kepercayaan di kalangan pihak berkepentingan. Dalam jangka masa panjang, sistem tersebut akan menyumbang kepada jalan raya yang lebih selamat, mengurangkan kesesakan, dan meningkatkan kualiti hidup bagi warganegara di negara-negara yang sedang membangun.

TABLE OF CONTENTS

DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT.....	iii
ABSTRAK.....	iv
LIST OF FIGURES	x
LIST OF TABLES.....	xiii
LIST OF ABBREVIATION	xiv
CHAPTER 1: INTRODUCTION.....	1
1.1. Background.....	1
1.2. Problem Statement.....	3
1.3. Objectives	6
1.4. Brief Methodology.....	6
1.5. Scopes	8
1.6. Significance of Project.....	9
1.7. Project Schedule.....	9
1.8. Expected Outcome	10
1.9. Project Outline	11
CHAPTER 2: LITERATURE REVIEW	13
2.1. Introduction.....	13
2.2. Review of Similar Existing Systems.....	13
2.2.1. Traffic Rules Violation Detection System	14
2.2.2. JPJ Digital Platform mySIKAP Online Transaction	21
2.2.3. MyBayar PDRM	32
2.2.4. Features Comparison of the Reviewed Systems	40
2.3. Review on Tools and Technology	43
2.3.1. Software	43
2.3.1.1. Ubuntu Desktop 24.04 LTS Noble Numbat.....	43
2.3.1.2. Visual Studio (VS) Code.....	45
2.3.1.3. Docker.....	46
2.3.1.4. Blockchain Framework - Hyperledger Fabric.....	47
2.3.1.5. Web Development.....	52
2.3.1.5.1. HTML	52

2.3.1.5.2. CSS	52
2.3.1.5.3. JavaScript.....	53
2.3.1.5.4. PHP	53
2.3.1.5.5. MongoDB	53
2.3.1.6. Application Programming Interface (API) Layer	54
2.3.1.6.1. Node.js	54
2.3.1.6.2. Node Package Manager (npm).....	54
2.3.1.6.2. Express.js	55
2.3.2. Hardware.....	56
2.3.2.1. Asus TUF FX505 Gaming Laptop.....	56
2.4. Summary	57
CHAPTER3: REQUIREMENT ANALYSIS AND DESIGN.....	58
3.1. Introduction.....	58
3.2. Scrum Methodology.....	58
3.3. Requirements	62
3.3.1. Analysis of Similar Existing Systems	62
3.3.2. Analysis of Proposed System.....	63
3.3.3. Questionnaire	64
3.3.3.1. Demographic Information.....	65
3.3.3.2. Current System Assessment.....	68
3.3.3.3. Digital System Features and Improvements	69
3.3.3.4. Privacy and Public Acceptance	71
3.3.4. Software Requirements	73
3.3.5. Hardware Requirements.....	74
3.4. Design	76
3.4.1. Scenario Illustration	76
3.4.2. Web Application and Software Architecture	78
3.4.3. Flow Chart Diagram.....	80
3.4.4. Context Diagram	84
3.4.5. Data Flow Diagram (DFD)	85
3.4.5.1. Level 1 Data Flow Diagram.....	85
3.4.5.2. Level 2 Data Flow Diagram.....	89
3.4.5.2.1. Level 2 Data Flow Diagram for Process 1.0	89
3.4.5.2.2. Level 2 Data Flow Diagram for Process 2.0	90

3.4.5.2.3. Level 2 Data Flow Diagram for Process 3.0	91
3.4.5.2.4. Level 2 Data Flow Diagram for Process 4.0	92
3.4.5.2.5. Level 2 Data Flow Diagram for Process 5.0	93
3.4.5.2.6. Level 2 Data Flow Diagram for Process 6.0	94
3.4.6. Entity Relationship Diagram (ERD)	95
3.4.7. Data Dictionary	96
3.4.7.1. Data Dictionary for Driver Table	96
3.4.7.2. Data Dictionary for Traffic Authority Table	99
3.4.7.3. Data Dictionary for Violation Table	100
3.4.7.4. Data Dictionary for Appeal Table	103
3.4.8. High-fidelity Wireframe Design	105
3.4.8.1. Web Application Wireframe for Multiple Users	105
3.4.8.1. Web Application Wireframe for Drivers	106
3.4.8.2. Web Application Wireframe for Traffic Authorities	112
3.5. Development	117
3.6. Testing	117
3.7. Deployment	118
3.8. Review	118
3.9. Summary	119
CHAPTER 4: IMPLEMENTATION	120
4.1. Introduction	120
4.2. Installation and Configuration of ChainGuard	120
4.2.1. Hyperledger Fabric Network	121
4.2.1.1. Prerequisites Installation	121
4.2.1.2. Hyperledger Fabric Blockchain Network Installation	125
4.2.1.3. Network Monitoring – Prometheus and Grafana	130
4.2.1.4. Chaincode Deployment	134
4.2.1.4. Hyperledger Explorer	135
4.2.2. Node.js Hyperledger Fabric Software Development Kit (SDK)	137
4.2.2.1. Setting Up Node.js Express Server (API)	137
4.2.2.2. Dockerizing the API for Deployment	140
4.2.3. Web Application	143
4.2.3.1. Setting Up the Web Application Environment	143
4.2.3.2. Setting Up the MongoDB Database	144
4.2.3.3. PHP Dependencies with Composer	146

4.2.3.4. Apache Web Server Configuration	147
4.2.4. Automation For System Deployment.....	150
4.3. System Architecture and Network Design.....	153
4.3.1. Hyperledger Fabric Blockchain Network Configuration	153
4.3.2. System Flow and Component Integration.....	154
4.4. Proposed System Implementation.....	156
4.4.1. Login.....	156
4.4.2. Driver Registration.....	157
4.4.3. Pending Violation	158
4.4.4. Driver Payment Process	159
4.4.5. Violation History.....	160
4.4.6. Appeal Submission	161
4.4.7. Driver Profile	164
4.4.8. Edit Driver Profile.....	164
4.4.9. Admin Registration	166
4.4.10. Admin Dashboard	167
4.4.11. Violation Logging.....	167
4.4.12. Violation Record Management	170
4.4.13. Appeal Management	171
4.4.14. User Account Management.....	172
4.5. Summary	174
CHAPTER 5: TESTING	175
5.1. Introduction.....	175
5.2. Functional Testing	176
5.2.1. Unit Testing	176
5.2.1.1. Test Cases for Shared Function.....	176
5.2.1.2. Test Cases for Drivers' Application.....	177
5.2.1.3. Test Cases for Traffic Authorities' Application.....	186
5.3. Non-Functional Testing	195
5.3.1. Reliability and Availability Testing	196
5.3.2. Compatibility Testing	197
5.3.3. User Acceptance Testing	198
5.4. Discussion on Results	201
5.5. Summary	203
CHAPTER 6: CONCLUSION AND FUTURE WORKS	204

6.1. Introduction.....	204
6.2. Objectives and Achievements.....	204
6.3. Project Limitations and Constraints.....	206
6.4. Future Works	208
6.5. Conclusion	210
Appendix A: Gantt Chart.....	217
Appendix B: Sample Questionnaire.....	218
Appendix C: Sample Feedback Form	223

LIST OF FIGURES

Figure 1.1: Statistic of Malaysia Road Accident from 2010 to 2019.	2
Figure 1.2.: Scrum Methodology (What is Scrum?, n.d.).....	7
Figure 1.3: Gantt Chart.	10
Figure 2.1: Main Dashboard (Reze, 2019).....	16
Figure 2.2: Adding Camera into System (Reze, 2019).....	17
Figure 2.3: List of Violation Logs (Reze, 2019).....	17
Figure 2.4: Management Interfaces (Reze, 2019).....	18
Figure 2.5: Violation Log Interface (Reze, 2019).....	19
Figure 2.6: Search Violation Log Interface (Reze, 2019).....	20
Figure 2.7: mySIKAP Login Page	23
Figure 2.8: mySIKAP Main Dashboard.....	24
Figure 2.9: Vehicle Information Section and List of Services.....	24
Figure 2.10: Driving License Information Section.....	25
Figure 2.11: List of Services Related to Driving License.....	26
Figure 2.12: Traffic Violation Penalty Section.....	27
Figure 2.13: Search Penalty Record via Identity Card Number.	27
Figure 2.14: Transaction Page	28
Figure 2.15: Select Payment Method.....	28
Figure 2.16: Transaction Completion Notification.....	28
Figure 2.17: List of Vehicle Operation Service.	29
Figure 2.18: View Transaction History.	30
Figure 2.19: Mobile Web View for mySIKAP.....	31
Figure 2.20: MyBayar PDRM Login Page.	34
Figure 2.21: MyBayar PDRM Main Dashboard.....	35
Figure 2.22: User Profile Management.....	35
Figure 2.23: List of Traffic Violation Recorded.....	36
Figure 2.24: Transaction Details.....	36
Figure 2.25: Payment Methods.....	37
Figure 2.26: Completion of Payment.....	37
Figure 2.27: Traffic Accident Management Page.....	38
Figure 2.28: Transaction History.....	38
Figure 2.29: Main Interface of Ubuntu 24.04 Noble Numbat.	45
Figure 2.30: User Interface of VS Code (Visual Studio Code User Interface, 2021).....	46
Figure 3.1. Three Pillars of Scrum (Hoffmann, n.d.).....	59
Figure 3.2: Product Development Flow with Scrum (Juillet, 2022).....	60
Figure 3.3. Age Group	66
Figure 3.4. States of Residence.....	66
Figure 3.5. Driving Experience.....	67
Figure 3.6. Vehicle Type	68
Figure 3.7. Analysis Summary of Section B.....	69
Figure 3.8. Analysis Summary for Section C	71
Figure 3.9. Analysis Summary for Section D	72
Figure 3.10. Manual Logging Scenario	76
Figure 3.11. Penalty Payment Scenario	77
Figure 3.12. Traffic Authority Analytics and Reporting	77

Figure 3.13. Web Application and Software Architecture.....	78
Figure 3.14. Flowchart of Proposed System	80
Figure 3.15. Context Diagram for Proposed System	84
Figure 3.16. Level 1 DFD for proposed system.....	86
Figure 3.17. Level 2 DFD for Process 1.0	89
Figure 3.18. Level 2 DFD for Process 2.0	90
Figure 3.19. Level 2 DFD for Process 3.0	91
Figure 3.20. Level 2 DFD for Process 4.0	92
Figure 3.21. Level 2 DFD for Process 5.0	93
Figure 3.22. Level 2 DFD for Process 6.0	94
Figure 3.23. Entity Relationship Diagram (ERD) for Proposed System	95
Figure 3.24. User Login Page	105
Figure 3.25. Registration Page.....	106
Figure 3.26. Pending Violations Page.....	107
Figure 3.27. Payment Page	108
Figure 3.28. Violation History Page	109
Figure 3.29. Appeal Submission Page	110
Figure 3.30. Edit User Profile Page	111
Figure 3.31. Admin Dashboard Page	112
Figure 3.32. Manual Violation Logging Page	113
Figure 3.33. Violation Record Page.....	114
Figure 3.34. Appeal Management Page	115
Figure 3.35. User Account Management Page	116
Figure 4.1. Go Language Website	123
Figure 4.2. Copy link address	124
Figure 4.3. Prerequisites Installation Validation Output.	125
Figure 4.4. Login Interface of Portainer	126
Figure 4.5. List of Containers Running.	127
Figure 4.6. GitHub Repository Page.....	127
Figure 4.7. Grafana Dashboard.....	133
Figure 4.8. Prometheus Dashboard.....	133
Figure 4.9. CouchDB Interface.....	134
Figure 4.10. Output of API Connection Testing.....	139
Figure 4.11. MongoDB Compass Download Page.....	145
Figure 4.12. MongoDB Connection Setup.	146
Figure 4.13. Hyperledger Fabric Network Configuration	154
Figure 4.14. ChainGuard System Architecture.....	155
Figure 4.15. ChainGuard Login Page.....	156
Figure 4.16. Driver Registration Page.	157
Figure 4.17. Pending Violation Page.....	158
Figure 4.18. Payment Loading Screen.....	159
Figure 4.19. Stripe Payment Gateway Page.....	159
Figure 4.20. Driver's Violation Record Page.	160
Figure 4.21. Driver's Violation Record Details.....	161
Figure 4.22. Driver's Appeal Submission Page.....	162

Figure 4.23. Driver's Appeal Submission Successful Notification.	162
Figure 4.24. Driver's Appeal History Page.	163
Figure 4.25. Driver's Appeal Details.	163
Figure 4.26. Driver's Profile Page.	164
Figure 4.27. Edit Driver's Profile Page.	165
Figure 4.28. Admin Registration Page.....	166
Figure 4.29. Admin Dashboard.....	167
Figure 4.30. Violation Logging Page.....	168
Figure 4.31. Hyperledger Explorer Dashboard.....	169
Figure 4.32. Driver's Email Notification.....	169
Figure 4.33. Violation Record Page.....	170
Figure 4.34. Details of Selected Violation.....	170
Figure 4.35. Appeal List Page.....	171
Figure 4.36. Details of Selected Appeal	172
Figure 4.37. User Account Management Page.	172
Figure 4.38. User Account Details.....	173
Figure 4.39. Edit User Account Page.....	173
Figure 5.1. Number of Participants and Role Assigned.....	199
Figure 5.2. Admin Task Completion Feedback Analysis.....	200
Figure 5.3. User Task Completion Feedback Analysis.....	200
Figure 5.4. Overall System Usability Analysis.....	201

LIST OF TABLES

Table 2.1. Features and functionalities of Traffic Rules Violation Detection System.	15
Table 2.2. Features and Functionalities of mySIKAP.	21
Table 2.3. Features and Functionalities of MyBayar PDRM.....	32
Table 2.4. Comparison of Features between Existing Systems and Proposed System.	40
Table 2.5. Features Comparison between Ethereum and Hyperledger Fabric.	48
Table 3.1. Functionalities of Proposed System.....	64
Table 3.2. Software Requirements for Proposed System.	73
Table 3.3. Hardware Requirements for Proposed System.	75
Table 3.4. Data Flow between System and Driver.	84
Table 3.5. Data Flow between System and Traffic Authority.	85
Table 3.6. Data Dictionary for Driver Table.....	96
Table 3.7. Data Dictionary for Traffic Authority Table	99
Table 3.8. Data Dictionary for Violation Table	100
Table 3.9. Data Dictionary for Appeal Table	103
Table 5.1. Unit Test Case for Log In.	176
Table 5.2. Unit Test Case for Driver Registration.	178
Table 5.3. Unit Test Case for Pending Violation Viewing and Payment Processing.	181
Table 5.4. Unit Test Case for Viewing Violation History.	182
Table 5.5. Unit Test Case for Viewing Appeal.....	183
Table 5.6. Unit Test Case for Appeal Submission.....	183
Table 5.7. Unit Test Case for Viewing Driver Profile and Update Profile Information.....	185
Table 5.8. Unit Test Case for Driver Sign Out.	185
Table 5.9. Unit Test Case for Admin Registration.	187
Table 5.10. Unit Test Case for Admin Registration.	189
Table 5.11. Unit Test Case for Logging Violation Record.....	190
Table 5.12. Unit Test Case for Admin Violation Record Management.	191
Table 5.13. Unit Test Case for Appeal Management.....	193
Table 5.14. Unit Test Case for User Management.....	194
Table 5.15. Unit Test Case for Admin Sign Out.....	195
Table 5.16. Test Case for Reliability and Availability Testing.	196
Table 5.17. Test Case for Compatibility Testing.....	198
Table 6.1. Analysis of Achievements.	205

LIST OF ABBREVIATION

AES	Automated Enforcement System
AI	Artificial Intelligence
API	Application Programming Interface
BFT	Byzantine Fault Tolerant
CFT	Crash Fault Tolerant
CSS	Cascading Style Sheets
DApps	Decentralized Applications
DFD	Data Flow Diagram
EIP	Ethereum Improvement Proposal
ETH	Ether
ERD	Entity Relationship Diagram
FPX	Financial Process Exchange
HTML	HyperText Markup Language
IBM	International Business Machines Corporation
JPJ	Road Transport Department (Jabatan Pengangkutan Jalan)
JS	JavaScript
LTS	Long Term Support
PHP	Hypertext Preprocessor
PoS	Proof of Stake
PDRM	Royal Malaysia Police (Polis Diraja Malaysia)
RAM	Random Access Memory
ROI	Region of Interest
SDK	Software Development Kit
URL	Uniform Resource Locator
VRAM	Video Random Access Memory
W3C	World Wide Web Consortium
WWW	World Wide Web

CHAPTER 1: INTRODUCTION

1.1. Background

Traffic violations represent a persistent and growing threat in developing countries. This is because these nations have been experiencing rapid urbanization and economic growth, resulting in a significant increase in vehicles on the road. Such a surge in traffic often leads to a rise in traffic violations, such as speeding, reckless driving, and red-light jumping. According to the Road Transport Act 1987, there are several traffic violations in Malaysia, ranging from behaviours that pose significant risks to public safety and urban mobility (Road Transport Department, n.d.). These common violations include speeding, illegal parking, illegal maneuvers, operating vehicles with expired or absent licenses and road tax, and various vehicle technical issues (Hariz Ikhwan, 2021). Despite their varying nature, each of these traffic violations can have a significant impact on several aspects of society, the environment, and the economy.

In terms of social factors, traffic violations can cause road accidents which can lead to the increase of mortality of the society. Additionally, violations can lead to traffic congestion which potentially decreases the quality of life and causes social disruption such as hindering access to the emergency lane for essential services like ambulances. On the environmental aspect, traffic violations can contribute to air and noise pollution, as well as resource depletion. Economically, they can also increase healthcare costs, reduce productivity, and eventually lead to economic loss.

Furthermore, the impact of these traffic violations is illustrated by recent road accidents statistics from the Ministry of Transport. The statistics shows an alarming increase in road accidents in the whole Malaysia, where approximately 600,000 road accidents cases were recorded throughout Malaysia which resulted in 6443 fatalities. Notably, 65% of these fatalities

involved motorcyclists (Chan, 2024). While in 2022, which had 545,585 cases road accidents were recorded. The statistic has showed a nearly 10% increase within a year (Benjamin et al., 2024). Figure 1.1 illustrated the increment of number of road accidents in Malaysia from 2010 to 2019 (*Road Accidents and Fatalities in Malaysia*, n.d.).

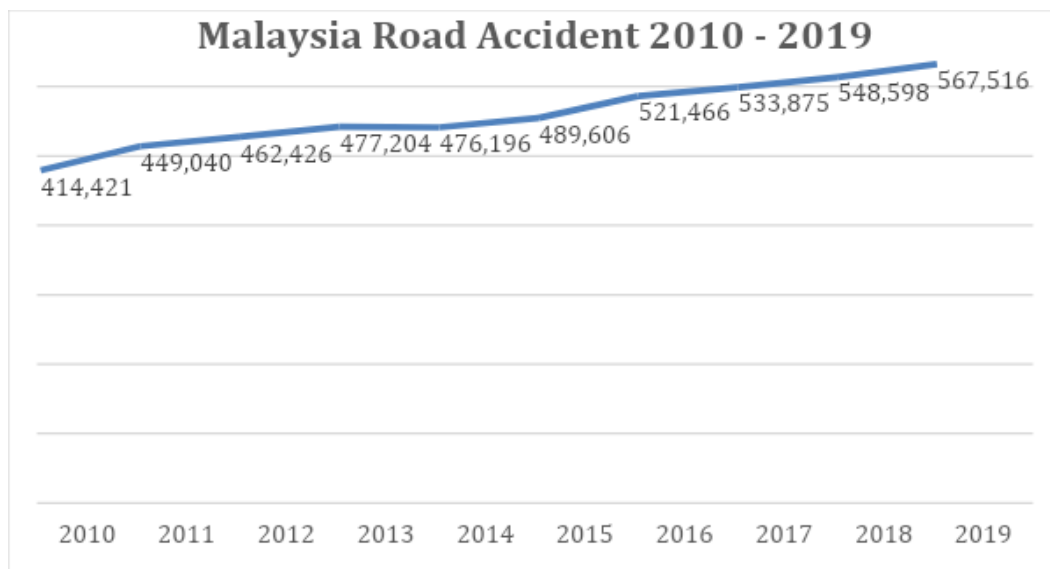


Figure 1.1: Statistic of Malaysia Road Accident from 2010 to 2019.

In addition, according to the Road Transport Department (JPJ), the number of fatal road accidents in Johor Baru rose slightly from 6,080 in 2022 to 6,344 in 2023. The Road Transport Department (JPJ) also conducted a nationwide operation during holiday season which involving 768,276 individuals. This operation revealed that a substantial portion of offenses, 43,505 cases were related to driving or riding without a license (Devi, 2024). The severity of the traffic violation situation is also underscored by recent incidents, such as a tour bus crash in Genting Highlands resulting in two deaths and 13 injuries, where the driver had no licence and 27 outstanding traffic summon for various offences (Genting accident, 2024).

The traditional traffic violation control and management systems in Malaysia have proven inadequate, inefficient, and lacking in transparency and accountability. This ineffectiveness has contributed to public frustration and a rise in aggressive driving behaviour. A survey has estimated that up to 2.4 million of the 13.3 million registered motorists in

Malaysia exhibit furious behaviour while driving, often due to traffic congestion and the perceived ineffectiveness of existing systems (Morhan, 2024).

Furthermore, some individuals take advantage of the ineffective penalty tracking system by exploiting the system to evade penalties, which causes significant financial losses and public discomfort. The total unpaid traffic fines have reached a staggering RM 4 billion from 1990 until 2024, highlighting the urgent need for reform in current system (Lau, 2023; *Unpaid traffic fines*, 2024). These issues collectively emphasize the critical need for an innovative, transparent, and efficient system to manage traffic violations in Malaysia which can effectively track and enforce penalties, deter evasion, and promote public trust.

Blockchain technology, a decentralized digital ledger system, which offers a promising solution to the limitations of traditional traffic violation management systems in Malaysia (Zheng et al., 2017). By providing a secure, transparent, and immutable platform for recording and managing violations, this technology is capable to address the issues of inefficiency, lack of accountability, and penalty evasion. For instance, the decentralized nature of blockchain can make the record difficult for individuals to manipulate the system, while the smart contracts can automate the penalty enforcement.

1.2. Problem Statement

Current traffic violation management systems show several critical issues that hinder their effectiveness and public trust. These issues manifest primarily through operational inefficiencies, data integrity concerns, and systemic vulnerabilities in enforcement mechanisms. The manual handling of violation logging and penalty issuance in traditional traffic enforcement system is time-consuming and prone to human error, such as mistyping wrong license plate numbers or identity card numbers. This can lead to delay in processing violations,

causing ineffective traffic management processes. In addition, the physical storage and retrieval of paper records can be troublesome and inefficient for record tracking. These inefficiencies will eventually cause an increase in administrative costs and reduced public trust in dealing with traffic management.

Besides, current traffic violation management systems are using the centralized framework, which means the systems are susceptible to manipulation and tampering. Unauthorized individuals might alter or delete violation records to avoid penalties, or errors in data entry that can lead to incorrect fines being issued to citizens. Moreover, bribery is one of the concerns with the traditional traffic management system due to the nature of centralized system. Individuals with authorized power may take advantage of the inefficient traffic violation logging process to solicit bribes (Kaur, 2024). These vulnerabilities raise concerns about the accuracy and reliability of violation records, as any inconsistencies and inaccurate records may lead to unfair penalties, disputes, and a loss of public trust in the system.

Variations in penalty applications across different jurisdictions or even within the same jurisdiction can result in perceived unfairness and reduced compliance. For instance, drivers in one area might receive heavier penalties for the same violation compared to drivers in another area. This inconsistency undermines the credibility of the traffic management system and can discourage drivers from adhering to traffic laws, which can lead to them defying fines and regulations (*Unpaid traffic fines*, 2024).

Additionally, the inconsistent enforcement of traffic fines such as granting discounts to certain drivers while others face full penalties, has eroded public trust in the traditional traffic management system. The inconsistency will result in confusion and undermine the seriousness of traffic violations, which potentially allows drivers to engage in risky behaviours. For instance, drivers who believe that discounted fines are an affordable consequence of their

actions may be more likely to speed, red-light jumping, or other dangerous driving practices (Hassan, 2023).

To address these critical drawbacks of current traffic violation management systems, this project proposes a blockchain-based traffic violation logging and penalty management system. This innovative approach leverages blockchain technology to provide a secure, transparent, and immutable platform for managing traffic violations. The system's decentralized architecture ensures that violation records cannot be manipulated, while its implementation of smart contracts automates penalty enforcement and reduces human intervention, thereby minimizing opportunities for corruption. Through a user-friendly web-based application interface, the system facilitates seamless interaction between drivers and traffic authorities, streamlining the violation management process. The automated penalty management capabilities effectively increase efficiency by reducing administrative overhead, while the immutable nature of blockchain records promotes accountability and transparency. By establishing a trustworthy and efficient framework for traffic violation management, the system enhances enforcement effectiveness and builds public confidence in traffic authorities. This solution represents an advancement over traditional systems, offering a more efficient, transparent, and decentralized approach that benefits all stakeholders.

1.3. Objectives

The primary objective of this project is to develop a decentralized system for securely logging and storing traffic violation data by utilizing blockchain technology. Other specific objectives of this project include:

1. To implement smart contracts for automated and consistent penalty issuance.
2. To develop a transparent and accessible platform for all the stakeholders, including traffic authorities, and drivers.

This project aims to achieve these objectives by developing a web-based application that leverages blockchain technology for traffic violation logging and penalty management system.

1.4. Brief Methodology

The project methodology is strategic-level planning for project management. It is a model that is often employed by project managers for designing, planning, and implementing their project goals within a defined period and budget (Sherman, 2015). In this project, Scrum, an agile framework that promotes flexibility, transparency, and iterative development, will be employed. In Scrum methodology, there are three crucial roles in the project development lifecycle: Scrum master, product owner, and development team. The Scrum master is the project manager, the product owner is the stakeholder of the product, and the production team is the individual responsible for developing and delivering the product within the expected duration (Marchewka, 2016).

Additionally, the Scrum project team's success hinges on five core values: Commitment, Focus, Openness, Respect, and Courage. The first core value, Commitment, drives the team's dedication to time-based task completion and sprint goals. Focus keeps the team centered on current sprint objectives. Openness fosters receptivity to changes, feedback, and challenges.

Respect promotes mutual understanding and collaboration within the team. Courage allows the team to speak out about uncertain areas and challenges, establishing transparent communication and effective problem-solving (*What is Scrum?*, n.d.; Schwaber & Sutherland, 2020). These values shape a productive and collaborative Scrum project team.

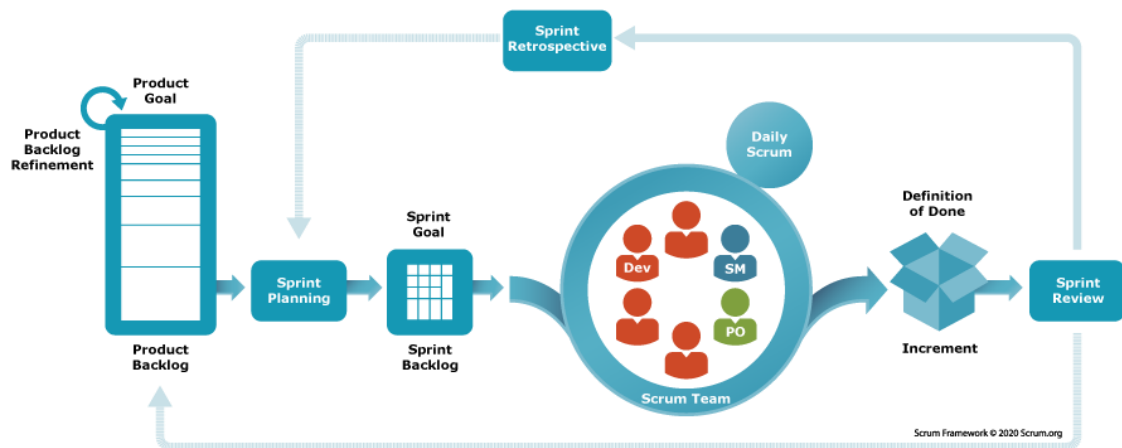


Figure 1.2.: Scrum Methodology (*What is Scrum?*, n.d.).

Figure 1.2 illustrates the Scrum methodology workflow. This approach focuses on delivering valuable product features in time-boxed iterations, which can range from days to weeks. Each iteration, also known as a sprint, will start with planning, where the product owner defines all the features of the product such as blockchain integration, violation logging, and penalty issuance. A product backlog, a dynamic list used to keep track of the product features, requirements, and changes, will be created based on the defined goals. Then, a sprint backlog is determined based on the product backlog which focuses on the aim of the current sprint (AWS, n.d.). These features stated will be prioritized as the sprint goal.

Besides, daily scrum meetings will be held to facilitate team communication, allowing the members to share their progress, planning, and problems encountered. Such communication link ensures continuous alignment with the sprint goal. After each sprint has reached the end,

reviews will be conducted to demonstrate completed features to the product owner. This phase allows the development team and scrum master to gather crucial feedback for future iterations. After each sprint is concluded, sprint retrospectives will be held. This phase provides opportunities for the team to reflect on the concluded sprint, allowing improvement and continuous enhancement. Throughout the project, the product backlog will be maintained to reflect the goal achieved, any new requirements that emerge, and the remaining parts that need to be done (Schwaber & Sutherland, 2020). It allows the team to have a clear understanding of the current project progression and deliver the final product with all the essential features at the project conclusion.

Furthermore, this project will not fully follow the structure of this methodology due to the project's single-person nature. However, the beneficial characteristics of this methodology will be utilized throughout the project such as the time-boxed task sprint, allowing the project to be completed in a modular and systematic manner.

1.5. Scopes

The project scope covers two main user groups: drivers and traffic authorities. For drivers, the system will provide access to personal violation history. This allows them to view and pay penalties issued, as well as receive real-time notifications of violations. On the other hand, traffic authorities will have the administrative privilege to utilize tools for logging and managing traffic violations, monitor violation trends, and oversee penalty issuance and collection processes.

In terms of the system's operational scope, it is extended to various road networks. This includes urban areas, highways, intersections with traffic signals, and designated parking zones. Such comprehensive coverage ensures that the blockchain-based solution can address traffic

violations across various environments and conditions, providing a unified approach to traffic management and enforcement.

1.6. Significance of Project

First and foremost, this blockchain-based traffic violation logging and penalty management system can significantly enhance the traffic management and enforcement. This project ensures data transparency and integrity by creating tamper-proof traffic violation records that substantially reduce the likelihood of disputes, bribery, and fraud by leveraging blockchain technology. Such advancement not only streamlines the enforcement process but also establishes trust between traffic authorities and the public.

Besides, the automated penalty management capabilities can dramatically increase efficiency by reducing administrative overhead. It also potentially accelerating the violation handling process and minimize external errors while handling violations. The automation of the system combined with the transparency provided by blockchain technology can provides all stakeholders with access to an immutable ledger of violations and penalties, which foster a culture of accountability and fairness in traffic management.

By addressing the ineffectiveness and inefficiencies of current traffic management system, this project has the potential to revolutionize traffic management system by providing a more efficient, transparent, and decentralized system that benefits both traffic authorities and the public.

1.7. Project Schedule

A project schedule is a project planning timetable developed based on the objectives, resources, and duration of the project (Gurnov, 2024). It allows the developer to have an

overview of the project lifecycle and it is tightly linked with the project methodology chosen. Besides, the Gantt Chart is utilized for scheduling and planning the project. Furthermore, this project is expected to be completed over the course of two semesters, which are Semester 1, 2024/2025, and Semester 2, 2024/2025. The duration of these two semesters is the following:

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

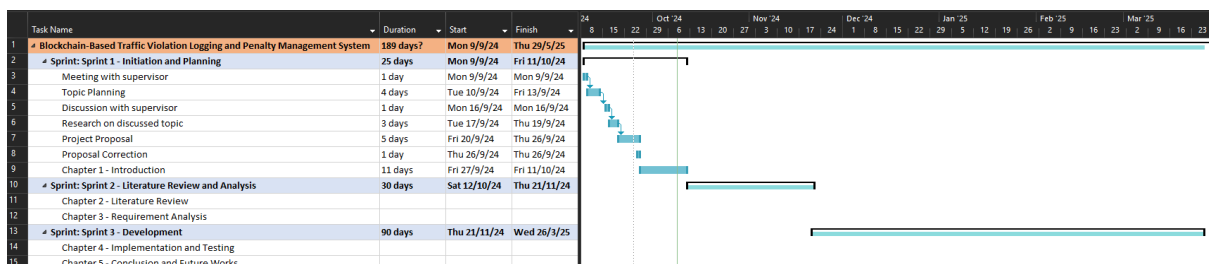


Figure 1.3: Gantt Chart.

The overall duration of both semesters is estimated at 293 days, excluding certain weeks that need to be subtracted for more accurate planning. Figure 1.3 illustrates the Gantt Chart for the project. A complete version of the Gantt Chart can be found in **Appendix A: Gantt Chart** of the report.

1.8. Expected Outcome

The expected goal of this project is the development of a traffic violation logging and penalty management system that leverages blockchain technology to create an unalterable, tamper-proof violation records handler. Such a system can enhance traffic management and enforcement. This system will ensure immutable and secure violation logs as all the traffic violations will be stored securely on the blockchain network which promotes transparency and prevents unauthorized tampering. Besides, automated fine management will be implemented through smart contracts, streamlining the process and minimizing human error. Additionally,

the system will provide transparency and traceability to all users of the system, including traffic authorities and drivers by giving access to an auditable trail of violations to ensure accountability. Furthermore, the immutable nature of blockchain records will aid in settling disputes about traffic violations and minimizing conflicts. By allowing digital payments and real-time notifications, the system will enhance the fine collection process and reduce administrative overhead, leading to a more efficient management process.

1.9. Project Outline

This paper details the development and implementation of the blockchain-based traffic violation logging and penalty management system in five chapters: Chapter 1: Introduction, Chapter 2: Literature Review, Chapter 3: Requirement Analysis and Design, Chapter 4: Implementation and Testing, and Chapter 5: Conclusion and Future Work.

Chapter 1: Introduction provides an in-depth overview of the project. This chapter consists of a background, problem statement, objectives, brief methodology, project scopes, significance of the project, project schedule, expected outcome, and project outline. These sections give a detailed explanation of the blockchain-based traffic violation logging and penalty management system.

Chapter 2: Literature Review covers the background study of the proposed project and area of investigation, such as literature review. This chapter reviews and evaluates the existing and similar applications to identify the key functionalities. The functionalities of the proposed system will also be discussed in this chapter.

Chapter 3: Requirement Analysis and Design highlights the detailed requirements specification and design for the proposed system. Visual representations include scenario illustration, system architecture, Data Flow Diagram (DFD), Entity Relationship Diagram (ERD), Activity Diagram, Data Dictionary, and proposed system wireframe.

Chapter 4: Implementation and Testing aims at the proposed system development and implementation. This chapter explains the implementation phases and provides test plans and technical descriptions of the proposed system.

Chapter 5: Conclusion and Future Work delivers the summary of the proposed system. The chapter focuses on the objectives accomplished, and project limitations, and outlines suggestions for future improvements.

CHAPTER 2: LITERATURE REVIEW

2.1. Introduction

This chapter delves into two main sections. The first section focuses a comprehensive analysis of three similar existing systems related to traffic violation logging and penalty management. The analysis explores each system's features, functionalities, advantages, and disadvantages for traffic violation logging and penalty management. For instance, these systems will aid in traffic violation logging via an automated approach with artificial intelligence (AI), enhancing the logging process and reducing administrative overhead. After the analysis, features comparison between chosen systems and the proposed system are made, highlighting the importance of the proposed system to achieve the targeted objectives. Additionally, the second section of this chapter focuses on the tools and technologies review for the proposed system development. The chapter also provides a conclusion to summarize the findings.

2.2. Review of Similar Existing Systems

Within this section, three similar systems related to the proposed system are comprehensively analysed and reviewed. The selected systems for in-depth analysis are Traffic Rules Violation Detection System, JPJ Digital Platform mySIKAP Online Transaction, and MyBayar PDRM.

These systems are chosen for in-depth review due to their unique and innovative functionalities and characteristics in the field of traffic violation logging and penalty management. For example, the Traffic Rules Violation Detection System utilized computer vision artificial intelligence for automating the traffic violation logging processes, potentially reducing enforcement task overhead. Besides, JPJ Digital Platform mySIKAP Online

Transaction offered drivers to have easy access to their traffic information and services via the internet, reducing petrol consumption and extra human resources. Lastly, the MyBayar PDRM, is a similar system to the second system which offers specific supports such as checking and paying fines as well as accident documentation services.

2.2.1. Traffic Rules Violation Detection System

Traffic Rules Violation Detection System is a desktop-based application that designed specifically for traffic authorities to bring ease in traffic enforcement and management by automating the identification and logging of traffic violations. It is an open-source project developed by a student in Department of Computer Science and Engineering from Khulna University of Engineering and Technology in Khulna (Reze, 2019). It is an open-source project licensed under GPL-3.0 license. By making the project open-source, the author aims to encourage the community to contribute to its improvement and enhance its impact on society. As an overview, the system is designed to enhance road safety and compliance by automatically identifying and flagging traffic violation instances. With computer vision artificial intelligence and media processing, the system also capable of processing live video feeds to detect violations such as red light jumping, illegal parking, and illegal maneuvers. The features and functionalities of Traffic Rules Violation Detection System are outlined in Table 2.1 below (Reze, 2019).

Table 2.1. Features and functionalities of Traffic Rules Violation Detection System.

Features	Functionalities
Detection Configuration	• Camera feed integration • Multiple camera support • Define capture areas for different violations ○ Signal violation crossing lines ○ No-parking zone boundaries ○ Direction monitoring zones
Automated Violation Detection	• Real-time video processing • Vehicle detection and tracking
Administrative Management	• Real-time violation notifications • Violation review and validation • Vehicle data entry processing such as vehicle information and violation details • Violation status tracking such as historical data access, search and filter capabilities, and archive management • Ticket generation

Upon launching the system, users, specifically the traffic authorities are greeted with the main interface as shown in Figure 2.1. The main interface is designed to provide a comprehensive overview of the live traffic footage and logged violation list. As shown in Figure 2.1, the main dashboard includes the option for selecting camera, location, search function for existing violation records, and overview for the live camera feed on selected location and violations flagged, providing easy navigation for users.

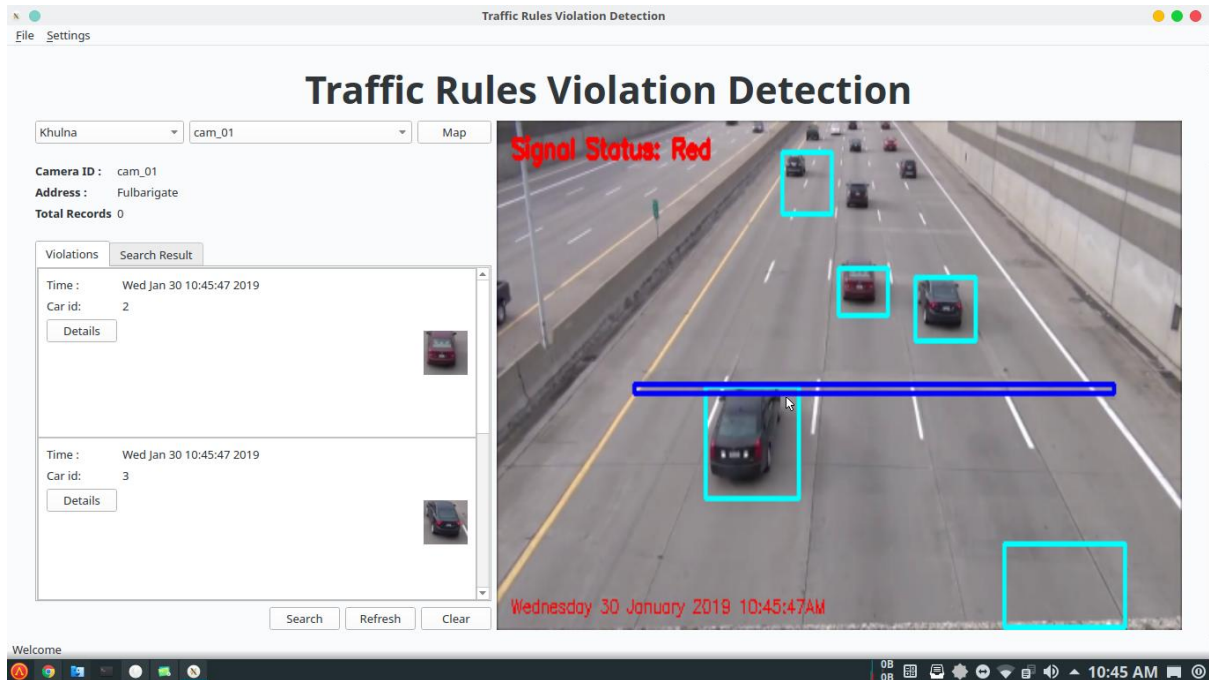
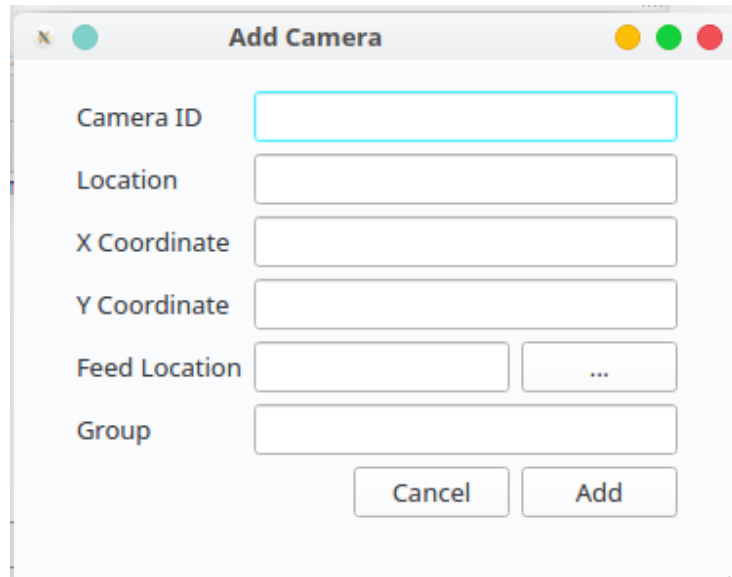


Figure 2.1: Main Dashboard (Reze, 2019).

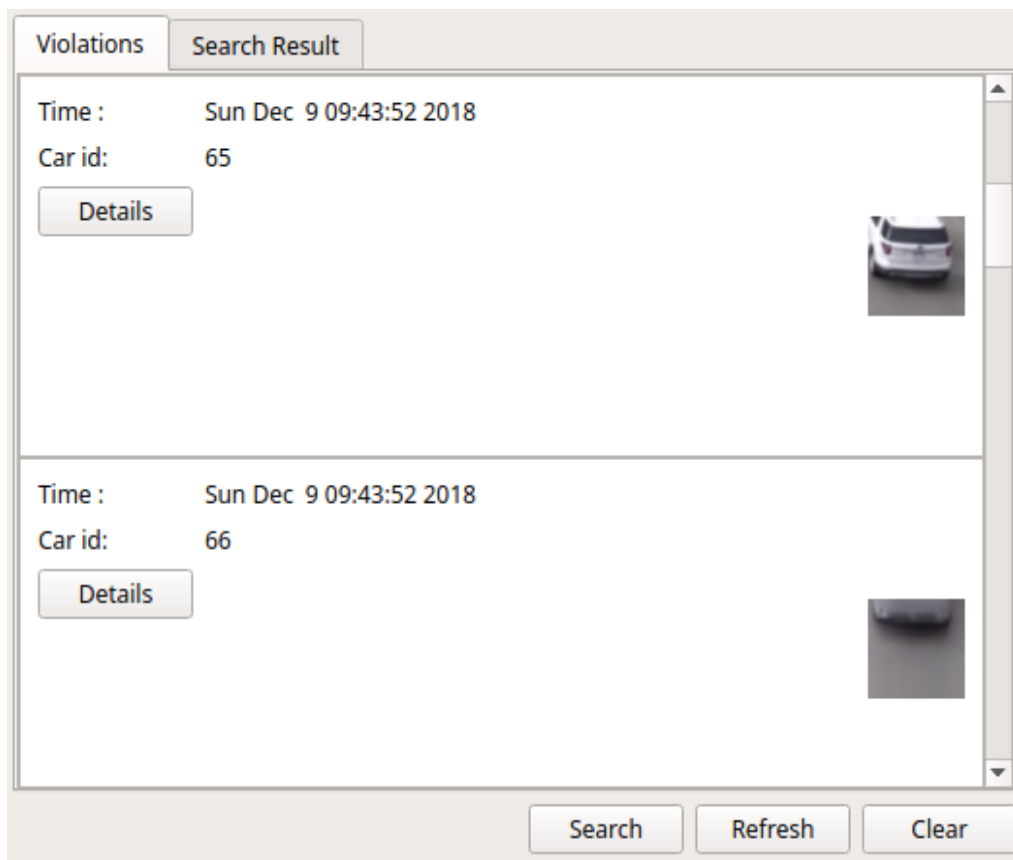
Furthermore, on the detection configuration features, the system provides flexibility to the user in configuring the camera entities to the system. As shown in Figure 2.2, users can seamlessly integrate new cameras into the system by specifying few parameters, namely unique camera identifiers, installation locations, detection boundaries, and custom Region of Interest (ROI). For the ROI, it allows user to define specific monitoring zones for various traffic violations, specifically signal violation crossing lines, no-parking zone boundaries and direction monitoring zones. The system also provides camera grouping feature to enhance the operational efficiency for multiple cameras. In Figure 2.3, the system offers real-time traffic violation log in the main interface, allowing the user to have a comprehensive view and understanding of the current traffic situation.



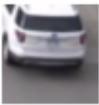
The 'Add Camera' dialog box contains the following fields and controls:

- Camera ID:
- Location:
- X Coordinate:
- Y Coordinate:
- Feed Location:
- Group:
-

Figure 2.2: Adding Camera into System (Reze, 2019).



The 'Violations Search Result' window displays a list of violation logs with the following details:

Time	Car id	Details	Image
Sun Dec 9 09:43:52 2018	65	Details	
Sun Dec 9 09:43:52 2018	66	Details	

At the bottom of the window are three buttons:

Figure 2.3: List of Violation Logs (Reze, 2019).

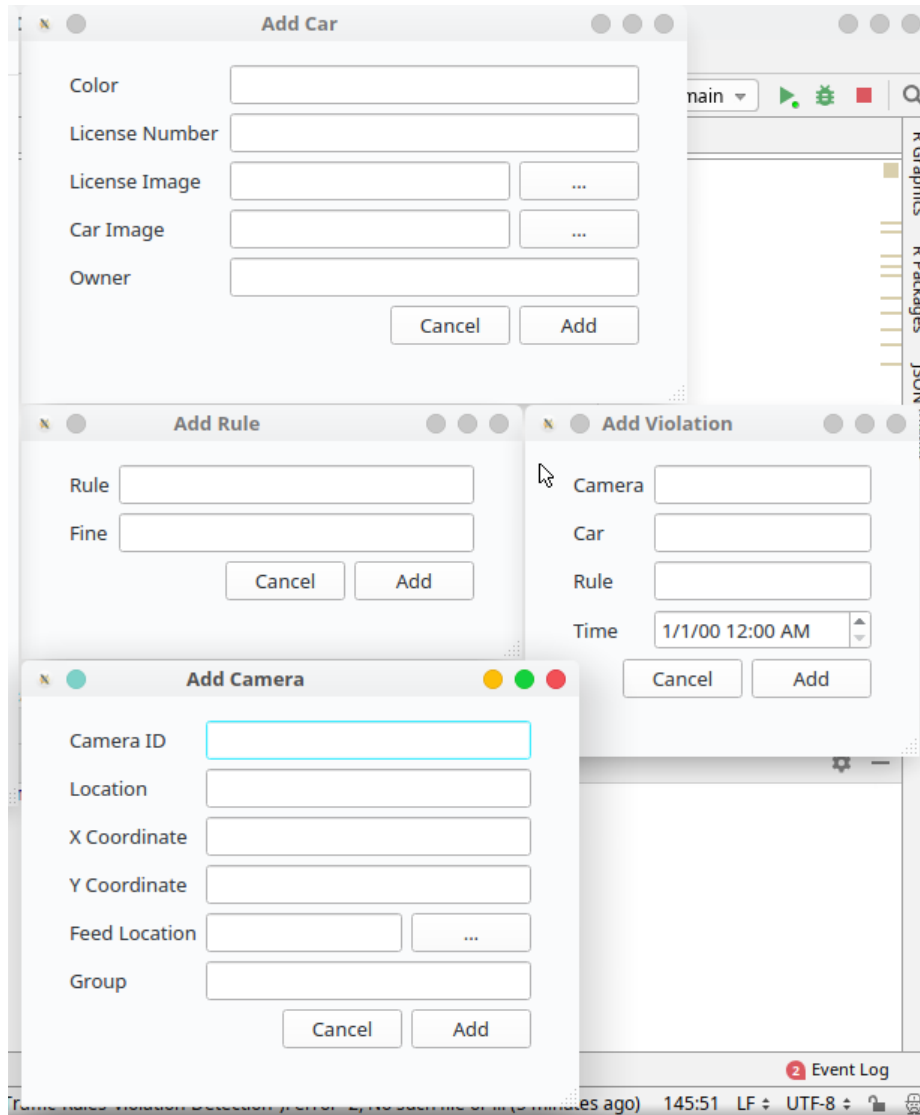


Figure 2.4: Management Interfaces (Reze, 2019).

To facilitate the administrative features of the system, there are four key management interfaces as shown in Figure 2.4. Each interface plays a crucial role in managing the traffic violation logs. Firstly, the Add Car interface allows administrators to enter vehicle information into the system database, such as color of the vehicle, license plate number and image, vehicle image, and vehicle owner information. Next, the Add Rule interface provides administrators to define traffic rules in the system with its corresponding fine amount. For the Add Violation interface, it allows administrators to operate the violation logging manually by inputting unique

camera identifiers, vehicle identifier, violated rule and timestamp. Last interface is Add Camera interface which allows administrator to seamlessly integrate new camera entity into the system.

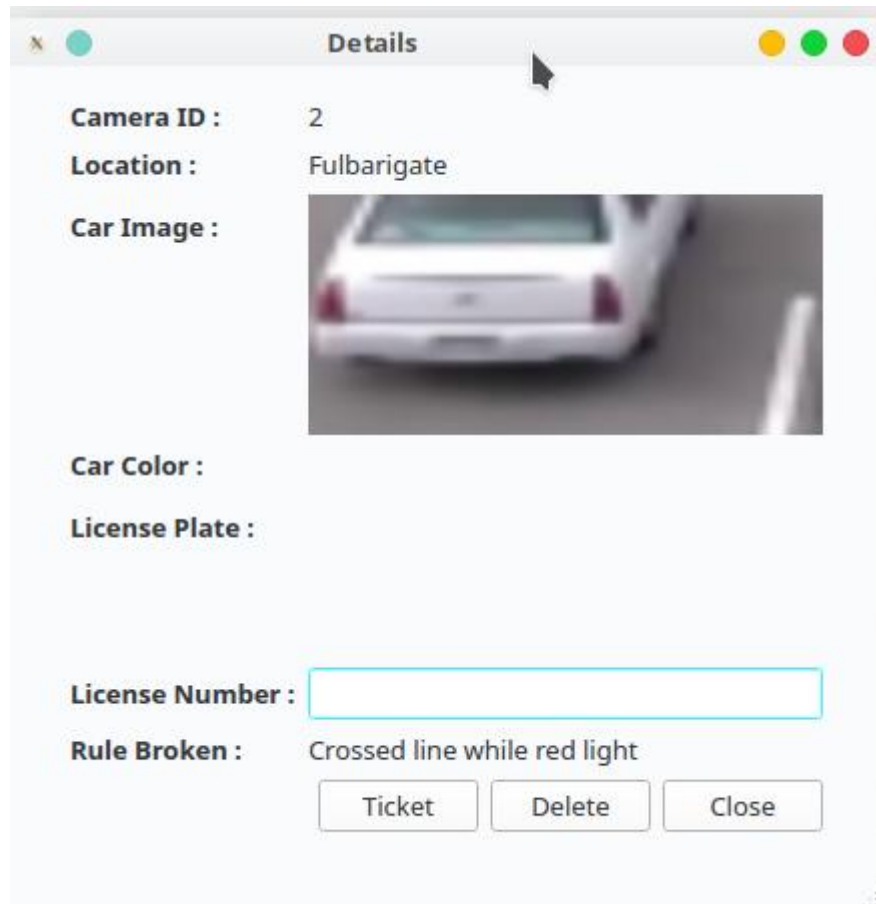


Figure 2.5: Violation Log Interface (Reze, 2019).

Furthermore, the system allows users to view and verify the violation recorded by interacting with the list of violation logs. Violation logs are automatically generated whenever the system's cameras detect and capture vehicles breaking pre-defined rules within the monitored ROIs. Figure 2.5 illustrates the violation log details, which displays the captured vehicle image and enables users to perform several actions: entering the vehicle's license plate number, issuing tickets to vehicle owners, and archiving any incorrectly recorded violations.

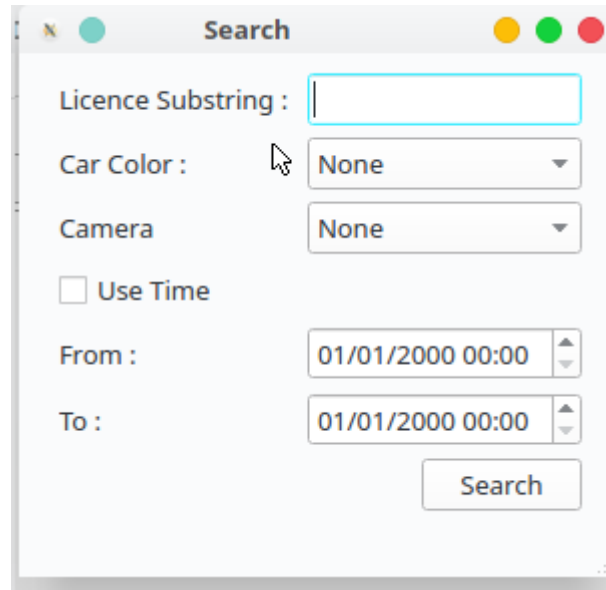


Figure 2.6: Search Violation Log Interface (Reze, 2019).

Another functionality that the system provides is the ability to search for historical violation records as illustrated in Figure 2.6. Users can retrieve past violations by filtering through multiple parameters including license plate numbers, vehicle colors, camera locations, and specific timestamps. The Traffic Rules Violation Detection System has presented notable advantages and disadvantages after comprehensive system analysis and review. On the bright side, it provides real-time detection and notification to users, which allows users to receive immediate identification of violations and take prompt actions against traffic violations. Besides, the automatic traffic violation detection can increase efficiency in traffic management and reduce the need for manual enforcement and associated costs.

However, there are some limitations in this system. Firstly, it is a partially automated system, which requires administrators to manage the traffic enforcement, such as inputting specific violation details manually like vehicle information and location. In addition, the system lacks a payment method implementation. It only generates tickets through human intervention, mainly administrators, and it does not offer real-time notification to the vehicle owners, relying heavily on manual ticket issuance by administrators.

2.2.2. JPJ Digital Platform mySIKAP Online Transaction

JPJ Digital Platform mySIKAP Online Transaction, also known as mySIKAP is a web-based portal that provided by the Road Transport Department (JPJ) of Malaysia (*JPJ MySIKAP*, n.d.). It is designed to streamline vehicle-related administrative services. This digital portal modernized the delivery of essential transportation services by enabling Malaysian citizens and residents to conduct various transactions remotely such as vehicle registration, road tax renewal, driver's license renewal, traffic violation penalty payments, and vehicle insurance management. The implementation of mySIKAP represents JPJ's commitment to Malaysia digital transformation through improving service efficiency while reducing wait times and administrative overhead at physical service counters. This approach is ultimately making essential transportation services more accessible and convenient to the public (*Online Vehicle Registration*, n.d.).

In general, mySIKAP provides essential transportation-related services through online portal. The system analysis will emphasize its features and functionalities, as detailed in Table 2.2.

Table 2.2. Features and Functionalities of mySIKAP.

Features	Functionalities
User Profile Management	• Enables changes to account credentials, such as updating passwords.
Vehicle Information	• Facilitates access to vehicle-related services, including license renewal, registration number purchase, and vehicle listings.
Driving License Information	• Provides access to driving license details and owner information, along with related services and transactions.

	• Provides direct access other services or transactions related to driving license.
Traffic Violation Penalty Management	• Display the traffic violation recorded by manual or automated enforcement system (AES). • Support online payment for recorded traffic violation via online banking, credit or debit cards. • Enables user to view their online transaction and reprint for completed transactions.
Vehicle Operation Services	• Offers various services for vehicles operating on roads such as vehicle structure change due to accidents, private vehicle type approval, beacon light installation, chassis and engine number restamping, vehicle modification requests, and tracking application status.

The mySIKAP portal features a secure login system that accommodates different user categories. First-time users must complete a one-time registration process using their respective identification documents where Malaysian citizens can register with their Malaysian Identity Card (MyKad), permanent residents with their Malaysian Permanent Residential Identity Card (MyPR), foreigners with their passport, and organizations with their valid agency registration number, including businesses, companies, and government entities. Upon accessing the mySIKAP URL, users are directed to a login page where they can either sign in with existing credentials or proceed with new account registration as illustrated in Figure 2.7.

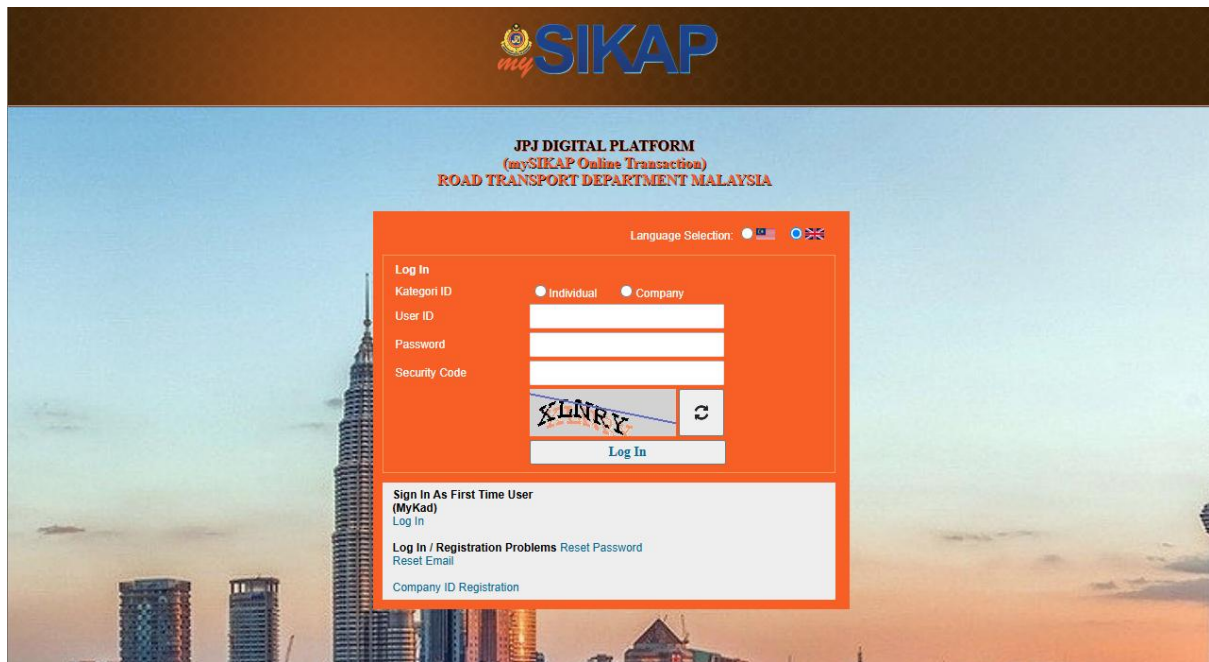


Figure 2.7: mySIKAP Login Page

Once the login is completed, users are directed to the main page, which provides an overview of the available services. These include managing personal profiles, handling vehicle-related transactions, accessing driving license services, managing offences and penalties, and exploring frequently asked questions about the platform as shown in Figure 2.8. Besides, the vehicle tab provides wide-range of vehicle-related services as illustrated in Figure 2.9. These services include renewing vehicle licenses, purchasing registration numbers at a fixed price, and viewing detailed vehicle listings along with digital road tax (eLKM) information.



Figure 2.8: mySIKAP Main Dashboard.

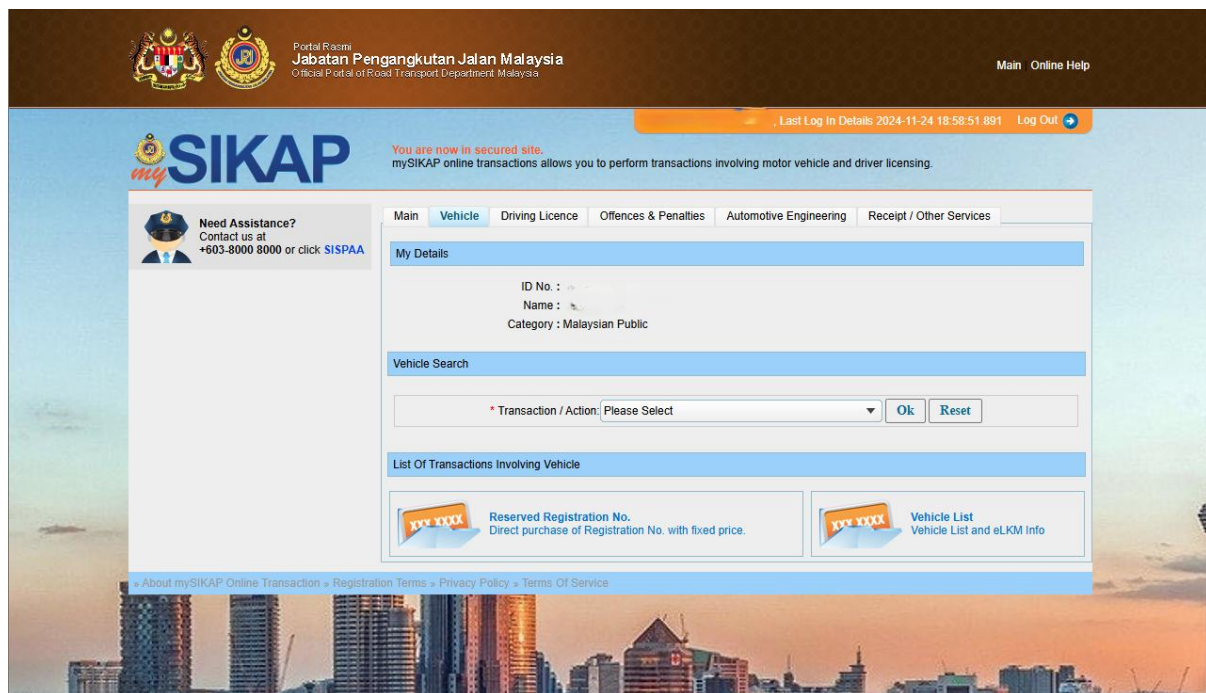


Figure 2.9: Vehicle Information Section and List of Services.

Additionally, Figure 2.10 provides an overview of the user's information, offering a detailed view of personal details, driving license records, physical restriction data, and driving instructor certification status. Apart from user information, the system also offers a comprehensive list of driving license-related services as shown in Figure 2.11. These services streamline access to key functionalities, such as applying for or renewing a driving license, checking driving test results, managing international driving permits, and converting foreign licenses, which ensure a seamless user experience.

The screenshot shows the mySIKAP web portal interface. At the top, there is a header with the Malaysian coat of arms, the text 'Portal Rasmi Jabatan Pengangkutan Jalan Malaysia', and 'Official Portal of Road Transport Department Malaysia'. A navigation bar includes 'Main' and 'Online Help'. A status bar indicates 'Last Log In Details 2024-11-24 18:58:51.891' and a 'Log Out' button.

The main content area features the 'mySIKAP' logo and a message: 'You are now in secured site. mySIKAP online transactions allows you to perform transactions involving motor vehicle and driver licensing.' Below this is a navigation menu with tabs: 'Main', 'Vehicle', 'Driving Licence' (selected), 'Offences & Penalties', 'Automotive Engineering', and 'Receipt / Other Services'.

The 'Driving Licence' section is titled 'My Details' and shows the following information:

- ID No. : 00000000000000000000
- Name : MUHAMMAD MUHAMMAD
- Category : Malaysian Public

Below this, there are four tabs: 'Owner Information' (selected), 'Licence Info', 'Physical Restrictions', and 'SPIM Info'. The 'Owner Information' tab displays a form with the following fields:

- Name :
- Address :
- City :
- State :
- Sex :
- Telephone :
- Notification :
- Passport Expiry Date :
- Postcode :
- Country :
- Date of Birth :
- E-mail :
- Owner Status :

Figure 2.10: Driving License Information Section.

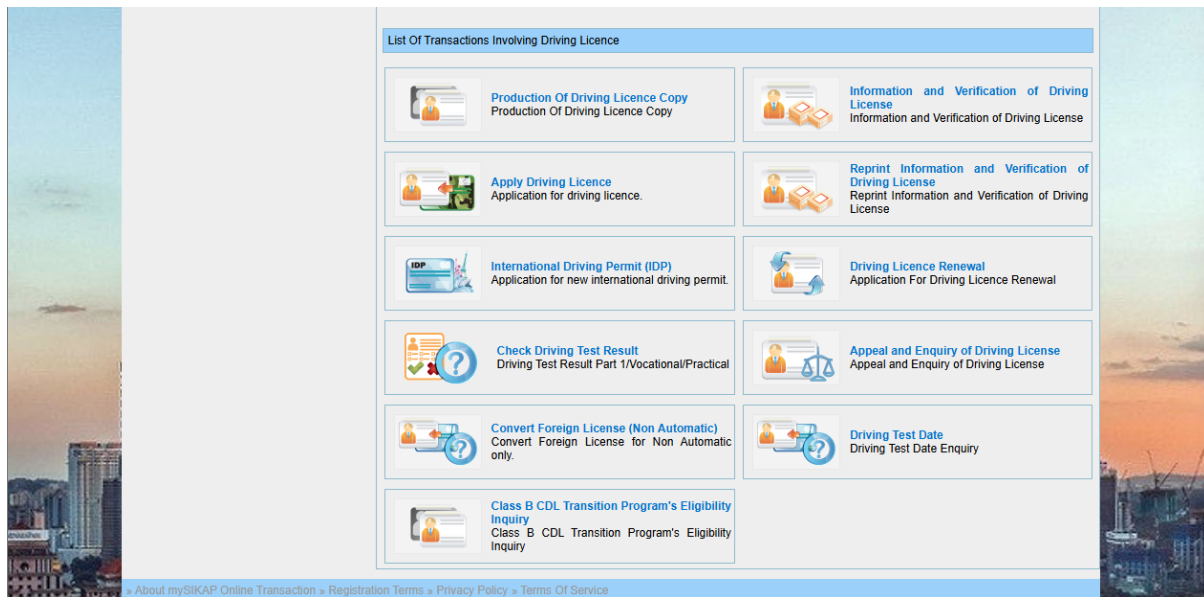


Figure 2.11: List of Services Related to Driving License.

Furthermore, the mySIKAP portal accelerate the process of managing and paying traffic violation penalties through its digital platform. As shown in Figure 2.12, users can access the penalty management interface through the “Offences & Penalties” section, which displays the user’s basic information, such as ID number, name, and category. The system enables users to search for specific penalty records through the “Carian Maklumat” section. If there are records of traffic violation, the system will display a comprehensive list of violations with details such as summons number, vehicle registration number, status, types of violation, and the amount as illustrated in Figure 2.13. In Figure 2.14, users can proceed to the payment page by selecting the specific penalty they wish to pay. In the transaction page, user can review the transaction details, including the violation description and penalty amount. The payment process is shown in Figure 2.15, where it is facilitated through Financial Process Exchange (FPX) and credit card payment. Users can also choose between individual or corporate account options for bank transfer. Upon successful payment completion, as demonstrated in Figure 2.16, the system generates a transaction notification containing essential details such as the

transaction date, payment amount, reference numbers, and banking information. The system also offers option for user to print the receipt for record-keeping purposes.

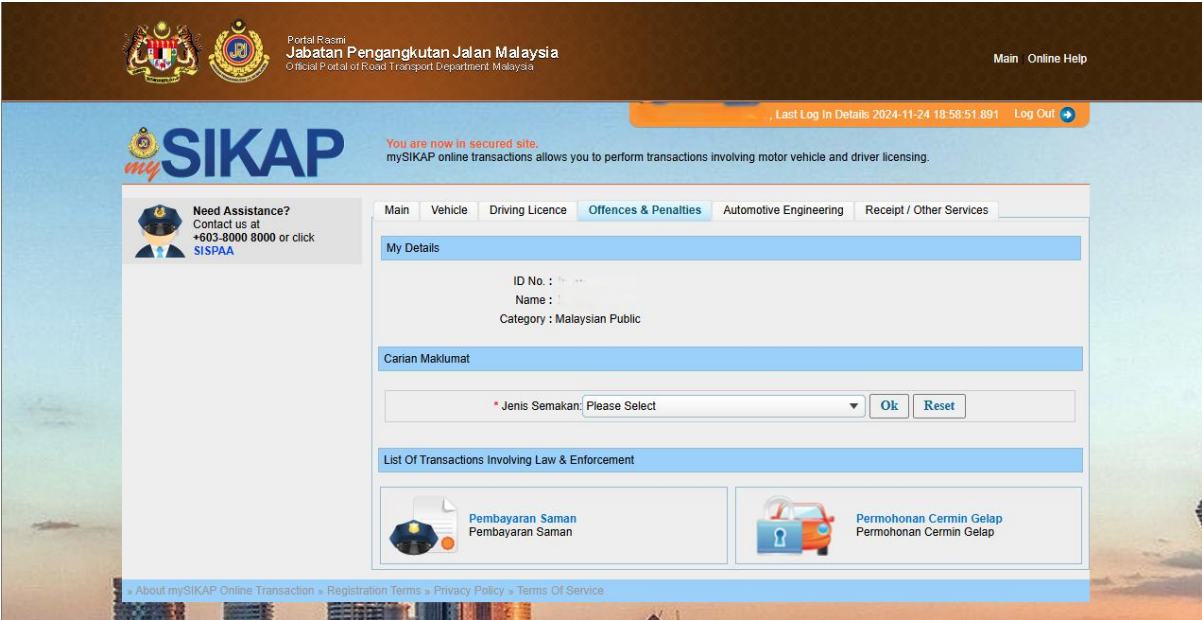


Figure 2.12: Traffic Violation Penalty Section.

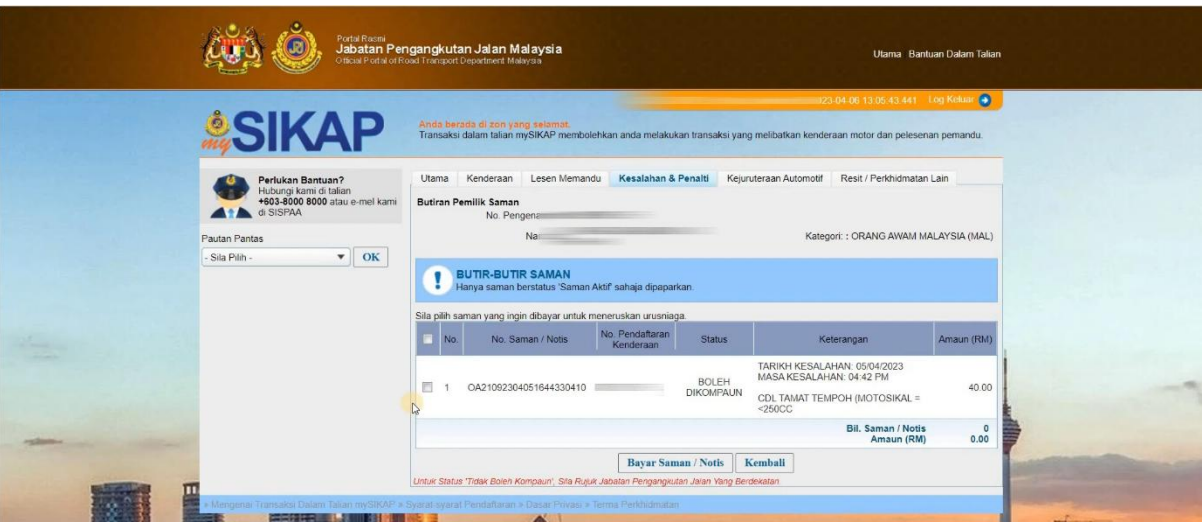


Figure 2.13: Search Penalty Record via Identity Card Number.

Portal Rasmi
Jabatan Pengangkutan Jalan Malaysia
Official Portal of Road Transport Department Malaysia

Utama Bantuan Dalam Talian

0A21092304051644330410 CDL TAMAT TEMPOH (MOTOSIKAL =<250CC) 40.00
Amaun Yang Perlu Dibayar (RM) : 40.00

2. PILIH KAEDAH PEMBAYARAN

☐ Kad Kredit

☒ Perbankan Internet **FPX**

Amaun Diterima (RM) : 40.00

☒ Individu **Maybank**

Jenis Akaun ☐ Korporat -- Sila Pilih --

Dengan menekan butang 'Bayar' di bawah, anda bersetuju dengan [Terma dan Syarat FPX](#). Bayaran minimum transaksi bagi Akaun Individu adalah RM 1.00 dan amaun maksimum adalah RM 30,000.00

3. BAYAR

Kepada pelanggan yang dihormati,

Sila patuhi arahan berikut sebelum memulakan transaksi :-

1. Sila pastikan penyekat popup dimatikan.

2. Jika terdapat gangguan semasa transaksi, Sila tunggu selama 15 minit untuk transaksi dikemaskini.

Bayar Kembali Batal

Mengenal Transaksi Dalam Talian mySIKAP • Syarat-syarat Pendaftaran • Dasar Privasi • Terma Perkhidmatan

Figure 2.14: Transaction Page

Portal Rasmi
Jabatan Pengangkutan Jalan Malaysia
Official Portal of Road Transport Department Malaysia

Utama Bantuan Dalam Talian

Perlukan Bantuan? Hubungi kami di talian +603-8000 8000 atau e-mel kami di SISFAA

Pautan Pantas
- Sila Pilih - **OK**

Utama Kenderaan Lesen Memandu **Kesalahan & Penalti** Kejuruteraan Automotif Resit / Perkhidmatan Lain

1. KETERANGAN TRANSAKSI

Butiran Pemilik
No. ID Pemilik
Nama Pemilik
Kategori Pemilik : ORANG AWAM MALAYSIA (MAL)

No. Transaksi	Keterangan	Amaun (RM)
0A21092304051644330410	CDL TAMAT TEMPOH (MOTOSIKAL =<250CC)	40.00

Amaun Yang Perlu Dibayar (RM) : 40.00

2. PILIH KAEDAH PEMBAYARAN

☐ Kad Kredit

☒ Perbankan Internet **FPX**

Amaun Diterima (RM) : 40.00

☒ Individu -- Sila Pilih --

Jenis Akaun ☐ Korporat -- Sila Pilih --

Dengan menekan butang 'Bayar' di bawah, anda bersetuju dengan [Terma dan Syarat FPX](#). Bayaran minimum transaksi bagi Akaun Individu adalah RM 1.00 dan amaun maksimum adalah RM 30,000.00

Figure 2.15: Select Payment Method.

Portal Rasmi
Jabatan Pengangkutan Jalan Malaysia
Official Portal of Road Transport Department Malaysia

Utama Bantuan Dalam Talian

SIKAP

Anda berada di zon yang selamat.
Transaksi dalam talian mySIKAP membolehkan anda melakukan transaksi yang melibatkan kenderaan motor dan pelesenan pemandu.

Utama Kenderaan Lesen Memandu **Kesalahan & Penalti** Kejuruteraan Automotif Resit / Perkhidmatan Lain

TRANSAKSI ANDA TELAH BERJAYA
Transaksi telah berjaya. Sila cetak email sebagai naskah.

Tarikh Pembayaran : 05 Apr 2023 1:07:29 PM
Jumlah Amaun(RM) : 40.00
Jenis Transaksi : ENF04005 - PEMBAYARAN SAMAN/NOTIS
No. Rujukan : 23040601161010064801
ID Transaksi Pembayaran : J01PRD017023040601101
ID Transaksi FPX : J04061307290253
Nama Bank : MALAYAN BANKING BERHAD (M2U)
Status Pembayaran : Berjaya

Cetak

Mengenal Transaksi Dalam Talian mySIKAP • Syarat-syarat Pendaftaran • Dasar Privasi • Terma Perkhidmatan

Figure 2.16: Transaction Completion Notification.

In addition to the traffic violation management features offered by mySIKAP, user can perform various vehicle operation transactions as shown in Figure 2.17. Users can submit applications for vehicle structure panel changes due to accidents, apply for vehicle type approval for private vehicles, beacon light installation permissions, chassis and engine number restamping through dropdown menus, general modification requests, and monitor all application statuses through the system's tracking feature. In Figure 2.18, users have access to additional services including the ability to view their online transaction and reprint for completed transactions. The system also offers Pos Laju Malaysia's tracking system which allows users to track their postal deliveries related to vehicle documentation.

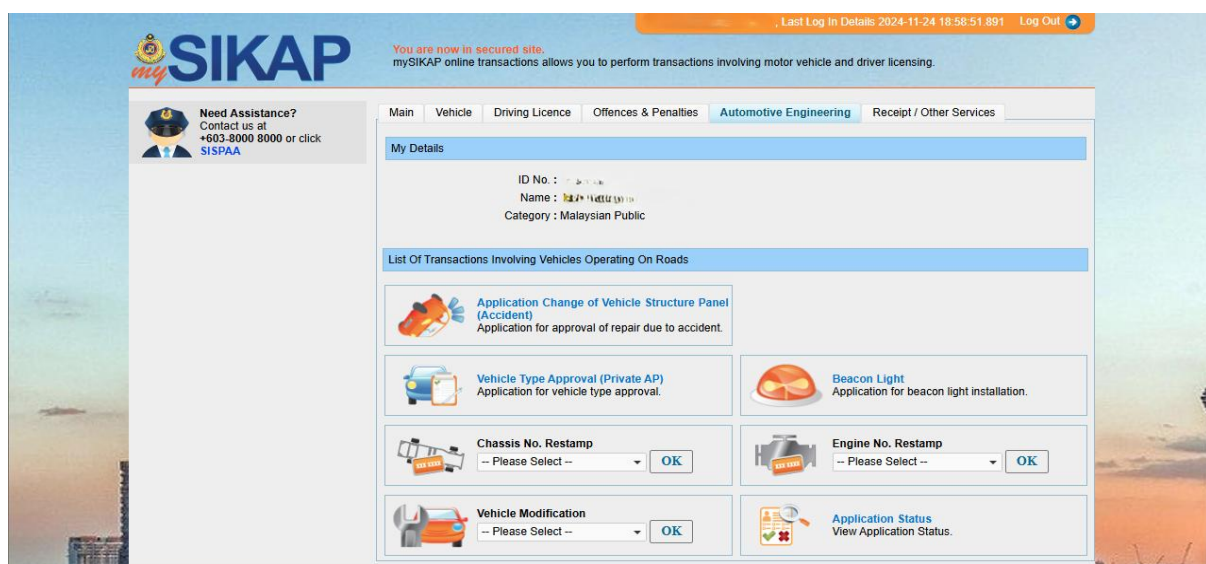


Figure 2.17: List of Vehicle Operation Service.

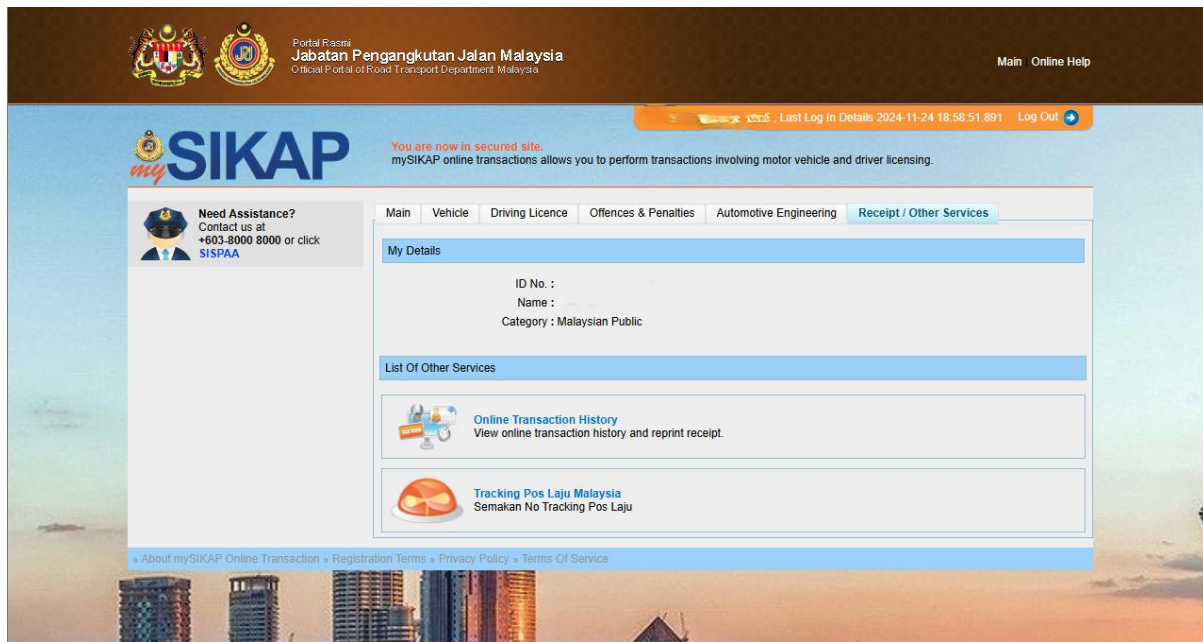


Figure 2.18: View Transaction History.

Based on the system analysis and review for mySIKAP, it offers several advantages, where users can perform digital payments through various methods, including online banking and credit cards, which effectively reduces administrative management tasks. Besides, users can conveniently check their violation records and view their transaction history through the system, while violation logging is handled either through manual entry or automated electronic enforcement (AES) (AES - JPJ Portal, n.d.).

On the other hand, mySIKAP portal does portrait some disadvantages. the primary limitation is the lack of real-time notification functionality, where users cannot directly receive notifications about newly recorded traffic violations and must rely on manually checking the system to discover any summons. Additionally, the system is designed specifically for desktop usage which make it less user-friendly for mobile phone users due to the web application is not optimized for mobile phone, as shown in Figure 2.19.

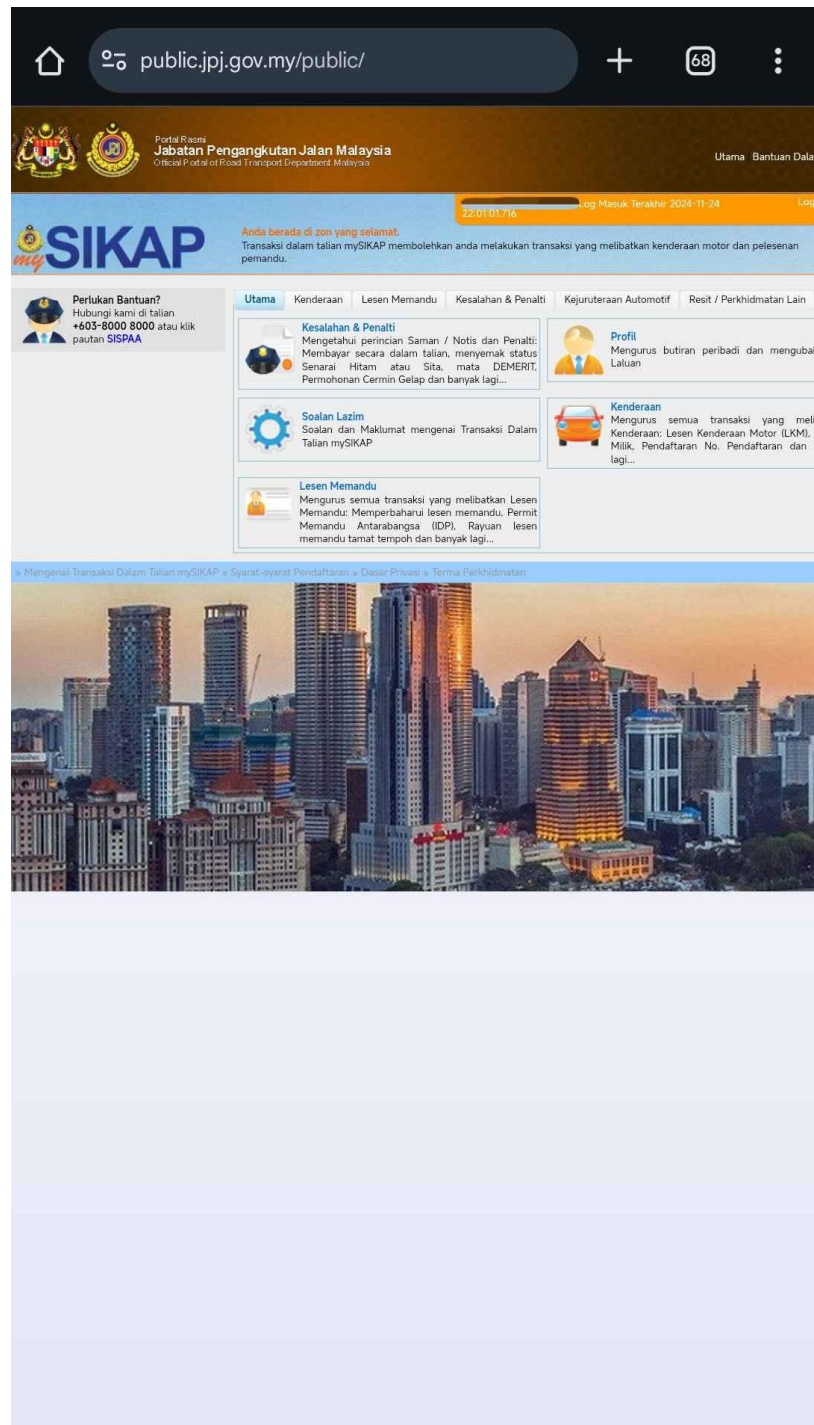


Figure 2.19: Mobile Web View for mySIKAP.

2.2.3. MyBayar PDRM

MyBayar PDRM shows a significant milestone in Malaysia's digital transformation journey, launched in March 2021 by the Traffic Investigation and Enforcement Department (JSPT) of the Royal Malaysia Police (PDRM). This innovative platform serves as an inclusive online portal specifically to streamline the management and payment of traffic violation penalties, which address the growing public preference for digital transactions in the modern era (Tan, 2022; Yin, 2021). In term of system overview, MyBayar PDRM operates through two distinct yet complementary platforms. The primary platform is a web-based application, which provides users with full access to the system through standard web browsers (MyBayar PDRM, n.d.). It enables easy access from any electronic devices with internet connectivity. Besides, MyBayar PDRM also launched with a mobile application, which has been developed to cater the increasing reliance on smartphones and on-the-go purposes, allowing users to manage and settle their traffic violations directly from their mobile devices (*MyBayar PDRM*, n.d.). The dual-platform approach offers maximum accessibility and flexibility for all users, regarding of their preferred device or location. As general, MyBayar PDRM serves as an online portal offering essential transportation-related services by PDRM. The system analysis focuses on its features and functionalities, as outlined in Table 2.3.

Table 2.3. Features and Functionalities of MyBayar PDRM.

Features	Functionalities
User Profile Management	• Enables changes to account credentials, such as updating identity card number, email, and address.
Traffic Violation Penalty Management	• Display the traffic violation recorded by manual or automated enforcement system (AES).

	• Support online payment for recorded traffic violation via online banking, and credit card. • Provides transaction history for completed payment.
Traffic Accident Management	• Provides services related to traffic accident such as preview and purchase accident documentation.

The system features and functionalities analysis will focus solely on web-based application of MyBayar PDRM due to mobile application has restrictions on taking screenshot. When users navigate to MyBayar PDRM's official URL, they are presented with a secure login page, as illustrated in Figure 2.20. The system provides two methods of accessing the system, which are existing users can directly access using their account credentials, and first-time users have the option to create a new account through the registration process. For new users, the system offers a step-by-step guidance for setting up their account securely. Upon successful authentication, whether through existing user login or completion of the registration process, the system automatically directs users to their personalized main dashboard, as shown in Figure 2.21. This dashboard acts as the central hub for accessing various features and functionalities offered by MyBayar PDRM.

In term of user account management functionalities, users can access their profile settings by navigating to the profile icon located in the top-right corner of the interface and selecting the "My Profile" option from the dropdown menu. Upon selection, the system redirects users to their dedicated account management page, where they can view and modify their personal information, including the user's full name, email address, and current residential address, as shown in Figure 2.22.

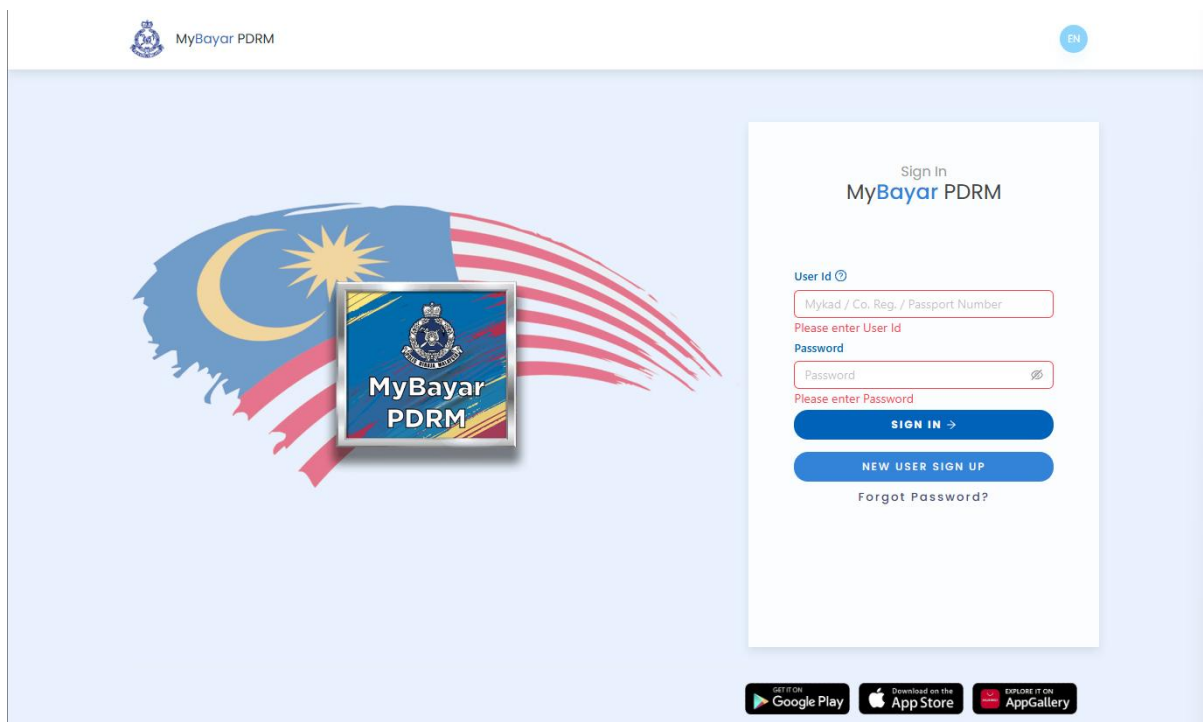


Figure 2.20: MyBayar PDRM Login Page.

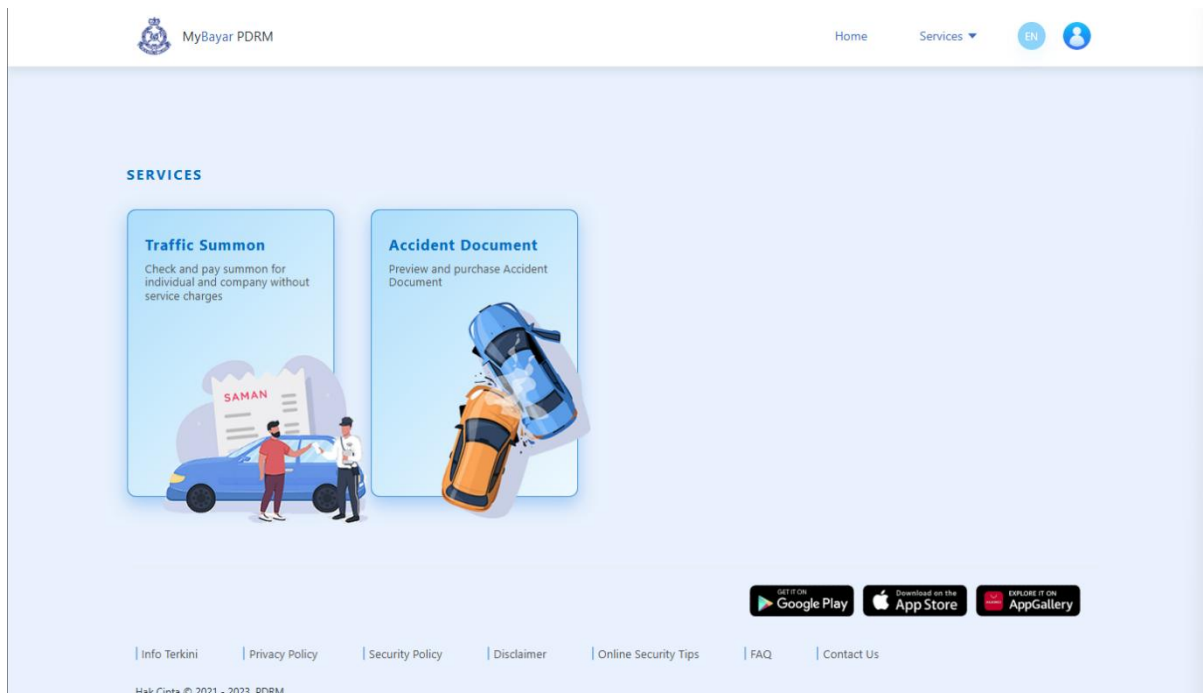


Figure 2.21: MyBayar PDRM Main Dashboard.

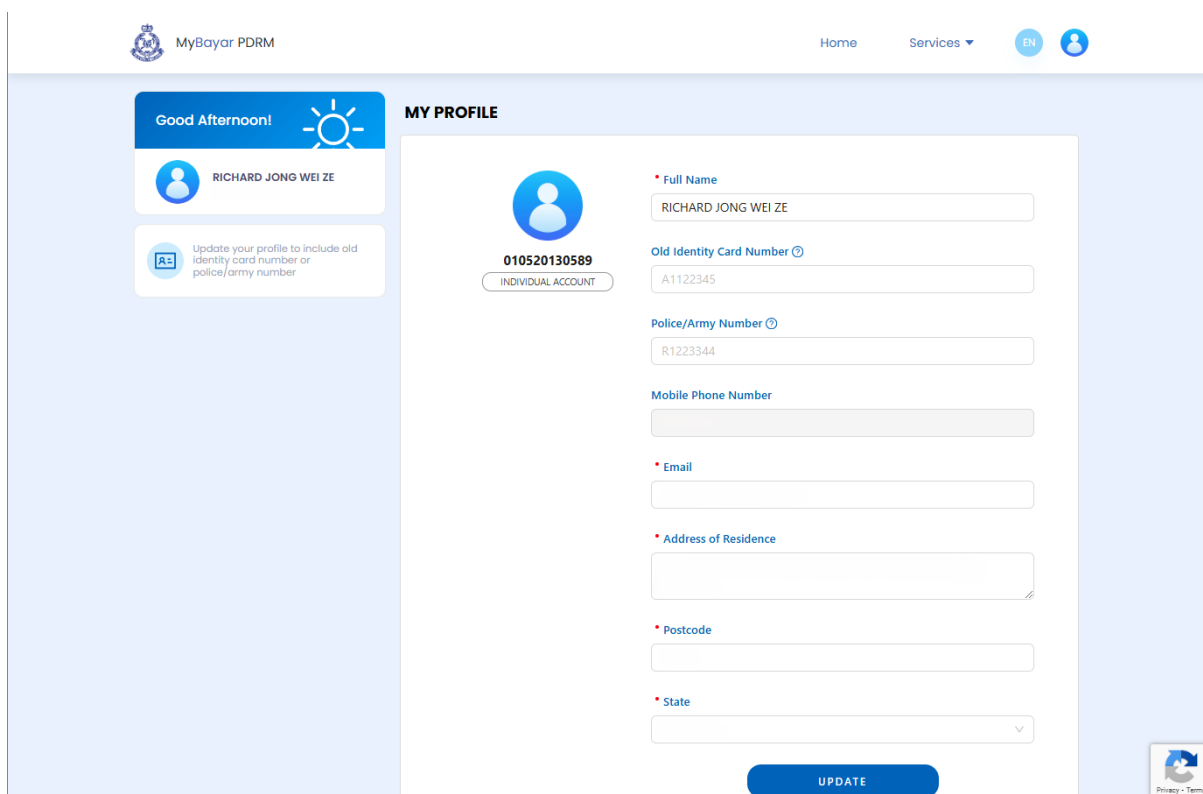


Figure 2.22: User Profile Management.

For traffic violation penalty management service, user can directly assess it through the main dashboard as highlighted in Figure 2.20. When accessing this service, as illustrated in Figure 2.23, the system displays a comprehensive list of traffic violations recorded by PDRM, allowing users to view their outstanding summons. To initiate payment, users can select the specific violation they wish to settle and click the "Pay" button. As shown in Figure 2.24, the system then directs users to a transaction details page, where they can verify important information including the vehicle number, violation date, and compound amount. For payment processing, Figure 2.25 demonstrates that the system offers two convenient payment methods to accommodate different user preferences: online banking through FPX or credit card payment.

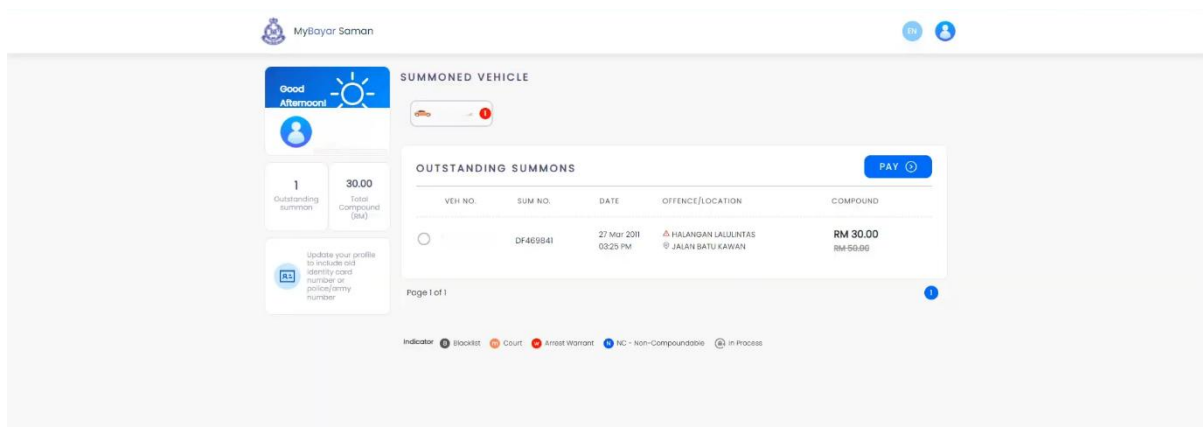


Figure 2.23: List of Traffic Violation Recorded.

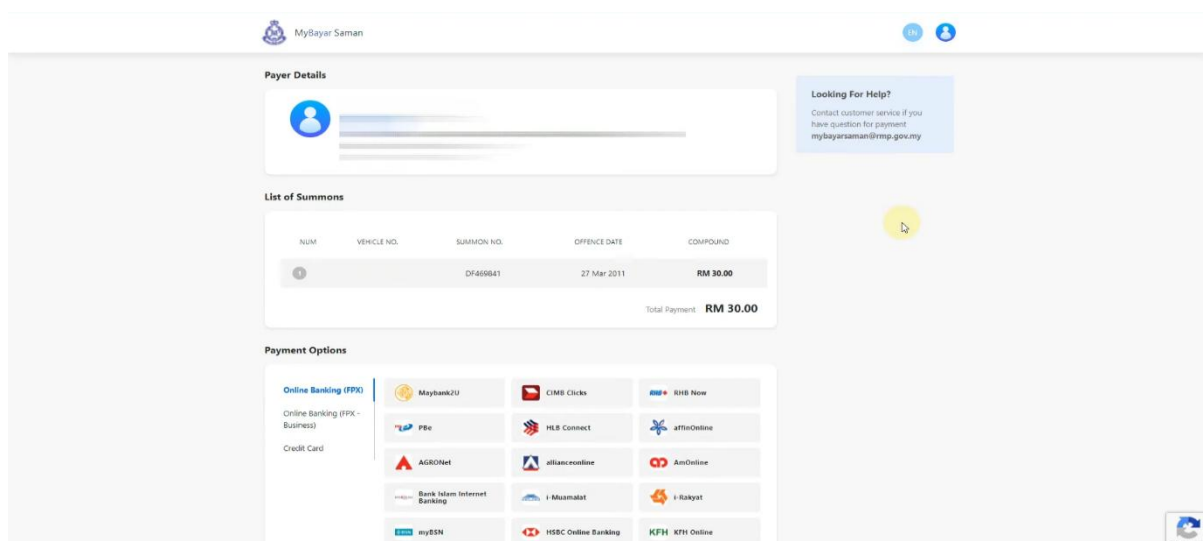


Figure 2.24: Transaction Details.

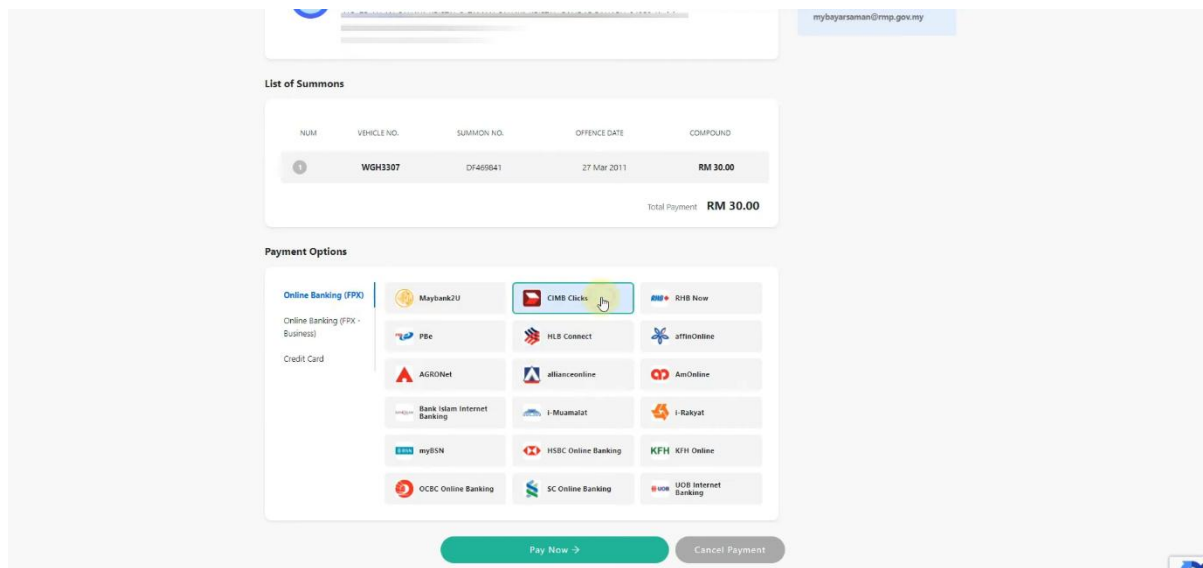


Figure 2.25: Payment Methods.

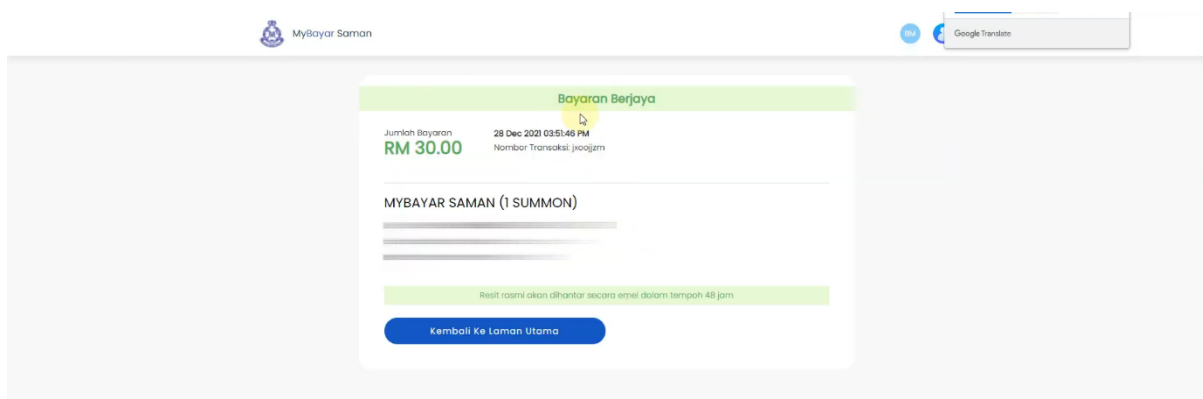


Figure 2.26: Completion of Payment.

In addition, as depicted in Figure 2.26, the system automatically generates a notification to confirm the transaction upon successful completion of the payment. This streamlined process ensures efficient management of traffic violation penalties while providing users with a secure and user-friendly payment experience. Furthermore, Figure 2.27 shows the traffic accident management page, where users can search and obtain accident investigation documents. Users can search for specific accident reports by entering their vehicle number and report number. MyBayar PDRM also provides transaction history preview features to users, illustrated in

Figure 2.28. This feature allows users to view a comprehensive record of their prior transactions.

Good Afternoon!

RICHARD JONG WEI ZE
010520130589

Purchase History
View purchased documents

Need Help?
Contact district police office

ACCIDENT DOCUMENT

Important Notes:

1. The online supply of Accident Investigation Documents begins on January 1, 2023.
2. Accident Investigation Documents will only be provided after 10 working days from the date of reporting the accident.
3. The online purchase of documents is limited to Damaged Accidents only.
4. The purchased documents are applicable for insurance claims only.
5. Please refer to Investigating Officer or the relevant district police headquarters if you have any issues.

Search For Accident Report

Vehicle Number **Report Number** **GUIDE**

VBE2234 TRAFIK KUALA LUMPUR / 000008 / 23

Start your search with vehicle registration number and police report number

Figure 2.27: Traffic Accident Management Page.

Good Afternoon!

RICHARD JONG WEI ZE
010520130589

PAYMENT HISTORY

No payment record

Figure 2.28: Transaction History.

MyBayar PDRM offers a notable advantage through its dual-platform availability, which provides users with both mobile and web-based applications. Such multi-platform approach ensures maximum accessibility and flexibility, offering users to access services provided by PDRM directly from their preferred devices, whether on-the-go through their smartphones or via desktop computers. On the other hand, the system does show limitations in transaction history and lacks of integration with the existing penalty management system offered by JPJ, mySIKAP. In terms of transaction history, users can only view their transactions

for the past 60 days, which may pose challenges for those require referencing from older records. Besides, the system's lack of integration with mySIKAP poses another disadvantage. Users must navigate between separate systems and account to access vehicle and driver-related services. This separation could potentially create inconvenience for users who need to manage both Royal Malaysia Police (PDRM) and Road Transport Department (JPJ) services simultaneously.

2.2.4. Features Comparison of the Reviewed Systems

Table 2.4. Comparison of Features between Existing Systems and Proposed System.

Features	Existing Systems			Proposed System
	Traffic Rules Violation Detection System	mySIKAP	MyBayar PDRM	
Blockchain Technology for Secure Logging	✗	✗	✗	✓
Manual Logging System	✓	✓	✓	✓
Automated Logging System	✓	✓	✓	✓
Real-time Traffic Violation Notification	✗	✗	✗	✓
Digital Payment Integration	✗	✓	✓	✓
Administrative Tools	✓	✗	✗	✓
Traffic Violation History Access	✓	✓	✓	✓

Based on Table 2.4, a comparative features analysis is done to examine the proposed blockchain-based traffic violation logging and penalty management system with existing systems: Traffic Rules Violation Detection System, mySIKAP, and MyBayar PDRM. The proposed system employs blockchain technology to create an immutable and tamper-proof

traffic violation logging mechanism. The existing systems rely on traditional centralized databases which may have systemic vulnerabilities in traffic management, such as potential data manipulation and lack of accountability. On the other hand, the proposed system utilizes a blockchain approach to ensure data integrity, preventing unauthorized modifications, and providing a transparent, verifiable traffic violation log.

Besides, all the compared systems do have manual logging and automated logging capabilities. Manual logging allows administrative personnel to manually enter or verify traffic violation details when automated systems might not capture the complete information. It serves as a critical backup and validation mechanism for traffic management and enforcement. Automated logging provides continuous traffic monitoring and traffic violation log generation, streamlining the traffic enforcement process and reducing administrative workload overhead.

Additionally, the existing systems do not have real-time traffic violation notifications for drivers, while the proposed system does provide this feature. This feature provides immediate alerts to drivers about their violations which promote transparency and instance response, allowing drivers to address violations promptly. It also can potentially reduce repeated offenses and improve overall road safety. In terms of digital payment integration, mySIKAP and MyBayar PDRM already support digital payment, and the proposed system has this feature to enable seamless online penalty payments. The digital payment integration offers simplicity over the fine settlement process, providing a user-friendly experience and potentially increasing penalty collection rates. In addition to that, blockchain technology will provide an additional layer of security and transparency to these transactions.

Furthermore, the proposed system and the first existing system mentioned in Table 2.4 do offer administrative tools that enable traffic authorities to seamlessly manage traffic-related tasks, such as managing violation logs, generating reports, and tracking violation trends. Lastly,

across all the systems, users can access their traffic violation history. The proposed blockchain-based system will provide a more secure and transparent method of accessing this history, which can be crucial for record-keeping and legal purposes.

In summary, the proposed system is the only system that stands out from other existing systems based on the features compared. By leveraging its key functionalities, it provides a blockchain-powered, secure, and transparent approach to traffic violation management and enforcement by offering real-time notification, manual and automated logging, digital payment integration, administrative tools, and traffic violation history access that address current systems' inefficiency and ineffectiveness.

2.3. Review on Tools and Technology

This section delves into the tools and technologies that are required for the development of the proposed system. It categorizes them into two primary components: software and hardware, providing a detailed explanation of the system requirements for the proposed solution.

2.3.1. Software

This section provides a comprehensive software requirement analysis of the proposed system.

2.3.1.1. Ubuntu Desktop 24.04 LTS Noble Numbat

Ubuntu is a widely recognized open-source Linux-based operating system developed by Canonical Ltd., a company founded by Mark Shuttleworth and a team of Debian developers in 2004. The name “Ubuntu” derives from an ancient African word, which represent the meaning of “humanity to others”, highlighting its aim of creating a community-driven platform for collaboration and development (*About the Ubuntu Project*, n.d.). There are four versions of Ubuntu, which are Ubuntu Desktop, Ubuntu Server, Ubuntu Core, and Ubuntu Cloud. Each version of Ubuntu has its unique use-case that fits the community requirements.

Furthermore, the one of the key features of Ubuntu Desktop is its consumer-oriented design. It gained popularity by prioritising user experience and addressing challenges associated with early Debian systems, which were complex, fragmented and difficult to install. Hence, Canonical launched Ubuntu project to offer intuitive, easy-to-use Linux desktop system based on Debian distribution. Besides, the open-source community’s regular checks and reviews provides Ubuntu operating system with high security level as vulnerabilities are identified and resolved immediately. Same as other Linux operating system, Ubuntu also benefits from robust security practices such as user privilege separation, flexible system

configuration and secure file system, creating a secure computing environment for users (Noviantika, 2024).

Another notable aspect is its commitment to enhance user privacy. Unlike other operating systems like Windows, Ubuntu does not require or store users' personal information and provides tools for managing privacy preferences. This ensures user rights are protected from third parties. (Noviantika, 2024). Furthermore, the open-source nature of Ubuntu allows users access to vast range of software and source code, with simplified installation process compared to others operating system. It is also highly reliable, especially for its Long Term Support (LTS) versions. For instance, Ubuntu 24.04 LTS Noble Numbat launched in April 2024, will receive security patches and maintenance support until 2029, making it a dependable choice for various applications (*About the Ubuntu Project*, n.d.).

Lastly, Ubuntu's lightweight performance and vast device compatibility set it apart from other operating systems. Unlike resource-intensive operating systems like Windows, Ubuntu can operate efficiently on entry-level devices with as low as 2GB of RAM and 5GB of storage. Additionally, it supports multiple architectures, including amd64, arm64, ppc64el, and s390x, ensuring compatibility across a wide range of devices, from single-board computers to enterprise-grade servers (*Basic installation*, n.d.). The main interface of Ubuntu 24.04 Noble Numbat is illustrated in Figure 2.29, highlighting the changes compared to previous generation.

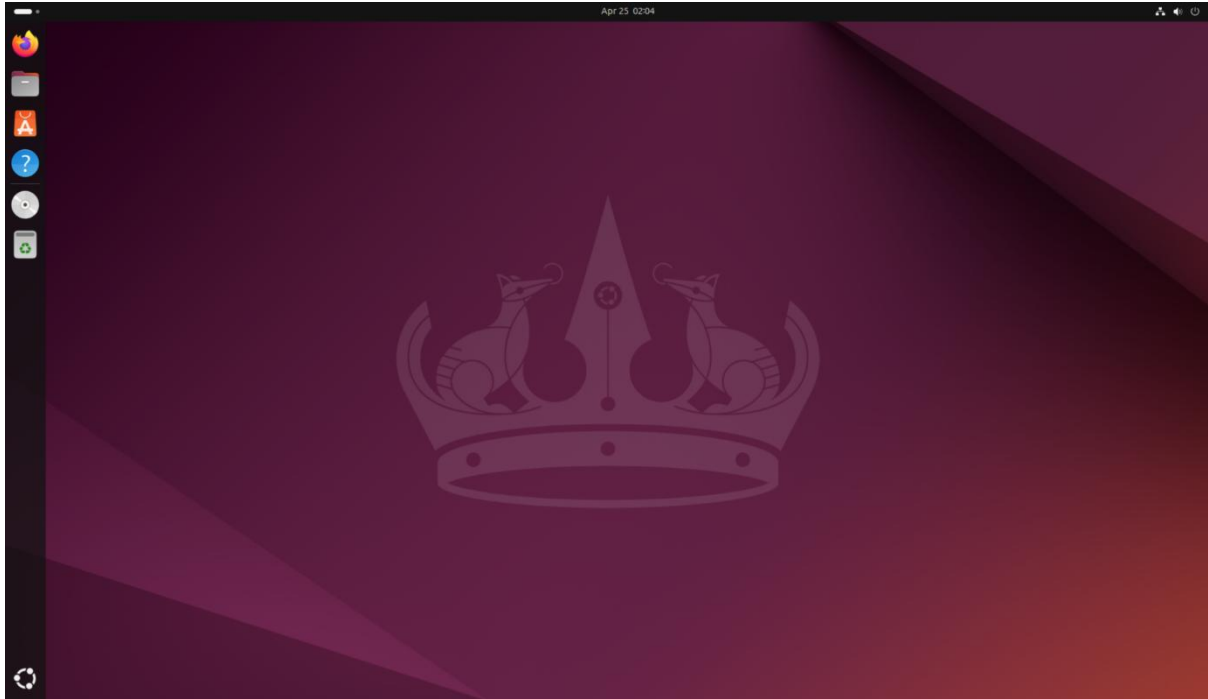


Figure 2.29: Main Interface of Ubuntu 24.04 Noble Numbat.

2.3.1.2. Visual Studio (VS) Code

Visual Studio Code, also known as VS Code, is a popular and open-source text editor developed by Microsoft. It was first release in 2015 and has since gained widespread adoption due to its versatility and extensive feature set (Johnson, 2022). The primary aim behind the creation of VS Code was to provide developers with a lightweight yet powerful source code editor that suitable for day-to-day usage (*Why Visual Studio Code?*, 2021). Figure 2.30 shows the user interface of VS Code, a popular IDE for writing, testing, and debugging code, with key components including the Activity Bar, Primary Side Bar, Editor Groups, Panel, and Status Bar.

One of the key features of VS Code is its extensive support for various programming languages and frameworks. Upon installation, it is packed with built in support for JavaScript, TypeScript, and Node.js. Users can expand their workspace with extension to make their own

VS Code to support for other programming languages, frameworks, and environments such as Python, Java, and Docker (Heller, 2022).

In addition, VS Code now integrates with Microsoft Copilot. This AI-powered code assistant aids in programming and debugging, which provides real-time suggestions and support during the development process. It also offers broad operating system compatibility, including Windows, macOS, and Linux, offering accessibility and flexibility to a wide user base. Furthermore, its user-friendly features such as syntax highlighting, which simplifies the code view for better readability, a built-in debugger that helps developers gain a solid understanding of their program while efficiently identifying and resolving errors, and code completion which enhances developers' code writing process (Iqbal, 2024).

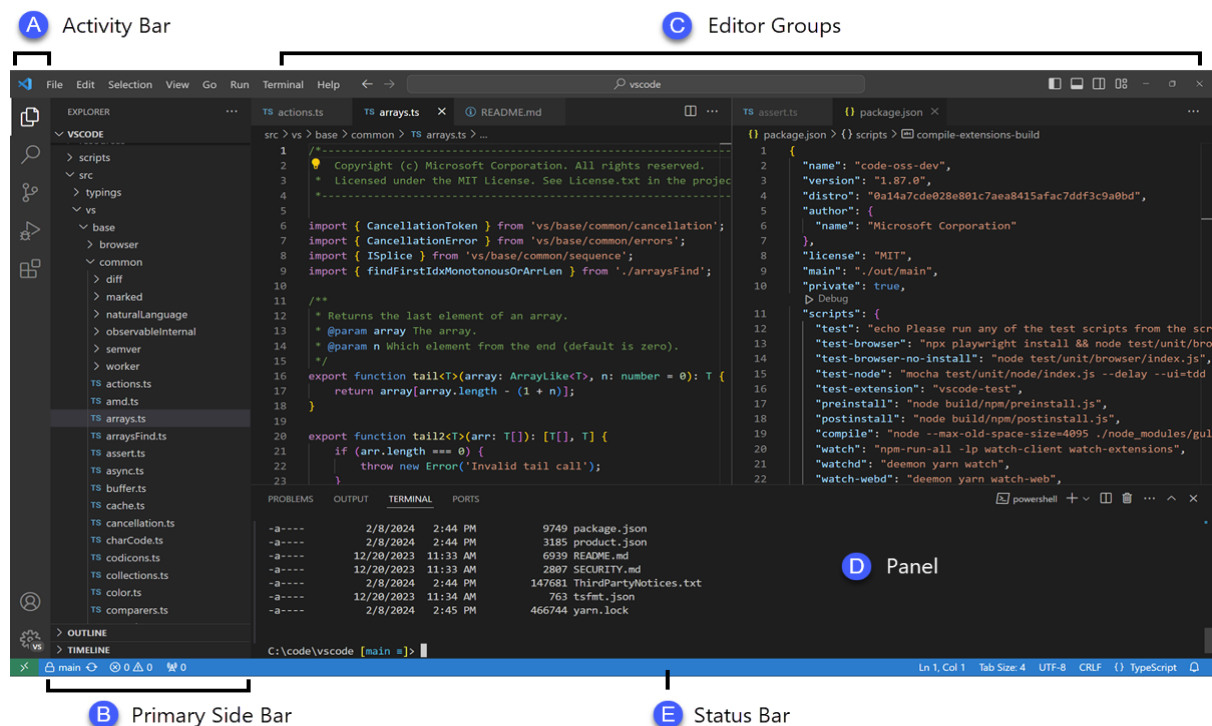


Figure 2.30: User Interface of VS Code (Visual Studio Code User Interface, 2021).

2.3.1.3. Docker

Docker is an open-source platform designed to automate the deployment, scaling, and management of applications using containerization. It allows developers to package application

and their dependencies into lightweight, portable containers that can run consistently across various devices and environments (*What is Docker?*, 2024). Docker was developed by Solomon Hykes and launched at PyCon 2013, which made an incredible transformation in software development. (Johnston, 2024). The primary aim of Docker is to simplify the application packaging, distribution, and deployment across diverse computing environments, focusing on the stability of the application. Through isolating applications and their dependencies in containers, it eliminates compatibility issues between development and production environment, enhancing portability and reliability (Susnjara & Smalley, 2024).

2.3.1.4. Blockchain Framework - Hyperledger Fabric

Ethereum blockchain is a decentralized, open-source blockchain platform for decentralized applications (DApps) development and deployment (Spydra, 2024). It was initially conceptualized by Vitalik Buterin, a Russian-Canadian programmer, and developed with a team of programmers. Ethereum was formally proposed in late 2013, with its development beginning in 2014, making the official Ethereum platform launched in 2015 (Sergeenkov, 2021; Marr, 2021). The primary aim of Ethereum is to expand the potential of blockchain technology beyond cryptocurrency by providing a flexible platform for developers to build decentralized applications. It offers the deployment of smart contracts, a self-execution algorithm, to operate without the need for intermediaries, which further enhance transparency, security, and trust in various use cases (*What Is Ethereum?*, n.d.)

Hyperledger Fabric blockchain is a popular and open-source blockchain framework designed specifically to suit enterprise use cases due to its modular architecture. It is part of the Hyperledger Umbrella project, an initiation by International Business Machines Corporation (IBM) and later contributed to the Linux Foundation in 2015. The first version was globally

launched in 2015, highlighting its appearance in the blockchain technology advancement (Chen, n.d.; Spydra, 2024). The aim of Hyperledger Fabric is to provide a secure and scalable blockchain platform that can integrate to enterprise needs. Unlike public blockchains, it offers private and permissioned networks, ensuring that only authorized parties can access or modify ledgers. Besides, the modularity nature of Hyperledger Fabric allows organization to customize the blockchain to suit specific requirements such as governance, confidentiality, and transaction throughput (*Hyperledger Fabric in Blockchain*, 2023).

Blockchain technology has revolutionized various industries with constant advancement in recent years. This proposed system will focus on reviewing two primary blockchain framework: Ethereum and Hyperledger Fabric, to identify the most suitable platform for its implementation. Table 2.5 provides an overview comparison between these two blockchain framework, highlighting key differences in their features.

Table 2.5. Features Comparison between Ethereum and Hyperledger Fabric.

Features	Ethereum	Hyperledger Fabric
Type of Network	• Permissionless (Public) network: Allows anyone to participate in the blockchain network.	• Permissioned (Private) network: Only allow authorised parties to access the blockchain network.
Consensus Mechanism	• Offers various support of consensus algorithms, such as crash fault tolerant (CFT) and byzantine fault tolerant (BFT) consensus protocols.	• Offers Proof of Stake (PoS) consensus protocol for facilitates agreement among parties in the Ethereum blockchain network.

Smart Contracts	• Defined with the term “chaincode” • Offers multiple programming languages support such as Go, Java and Node.js.	• Written in Solidity, a high-level programming language developed specifically for smart contracts. • Offers wide range of smart contracts development environment (Truffle Suite, HardHat, Foundry, and Remix).
Native Cryptocurrency	• Does not have native cryptocurrency.	• Do have native cryptocurrency, known as Ether (ETH), which can be used for transaction fees and member participation.
Governance	• Monitored by Hyperledger community and managed by the Linux Foundation.	• Managed by Ethereum community through community proposals (EIPs)

First and foremost, one key difference of Ethereum and Hyperledger Fabric is the type of network they support. Ethereum operates as a permissionless network, also known as public network, which allow anyone to join and participate without membership restrictions. Other the other hand, Hyperledger Fabric has a permissioned network. This indicates the blockchain network is private and only allow authorized users to participate, making it an ideal choice for enterprise use cases that prioritize privacy and access control (Spydra, 2024; *Hyperledger Vs Ethereum in Blockchain*, 2024).

The consensus mechanisms used by these frameworks also differ. Ethereum utilizes the Proof of Stake (PoS) consensus protocol, which facilitates agreement among parties in a decentralized manner, enhancing the synchronization, validation and data processing with low computational work (Team, 2024). Hyperledger Fabric supports various consensus algorithms, including Crash Fault Tolerant (CFT) and Byzantine Fault Tolerant (BFT) protocols. This key feature of Hyperledger Fabric provides flexibility and reliability to developers for choosing ideal approach in fitting diverse enterprise applications (*Introduction — Hyperledger Fabric*, n.d.)

In addition, smart contracts play a crucial role in blockchain management. It is a self-executing script that automate the transaction in the blockchain network according to the pre-determined terms and conditions (*Smart contracts on blockchain*, 2024). Ethereum provides a robust environment for smart contract development using Solidity, a high-level programming language designed specifically for writing smart contracts. Due to its popularity, there are wide-range of smart contract development tools such as Truffle Suite, HardHat, Foundry, and Remix, providing developers flexibility in development process. In contrast, Hyperledger Fabric refers smart contracts as “chaincode” and supports multiple programming languages, such as Go, Java, and Node.js, offering options for developer to choose based on their programming language preferences.

Furthermore, in terms of native cryptocurrency, Ethereum comes with built-in cryptocurrency, known as Ether (ETH), which serves multiple purposes, such as transaction fees and member participation. With primary focus on diverse enterprise use case, Hyperledger Fabric does not include a native cryptocurrency, focusing instead on permissioned data exchange and governance (*Hyperledger Vs Ethereum in Blockchain*, 2024).

Lastly, these two blockchain framework have their unique governance structures. Ethereum is managed by the Ethereum community through a decentralized process that involve community proposals known as Ethereum Improvement Proposals (EIPs). This allows for a more open and inclusive development and maintenance for the framework. In contrast, Hyperledger Fabric is governed by Hyperledger community and is managed under the Linux Foundation. This approach ensures a collaborative and modular approach to foster a robust enterprise level blockchain development (*Hyperledger Vs Ethereum in Blockchain*, 2024).

In conclusion, Ethereum and Hyperledger Fabric have their uniqueness in diverse use case. With permissionless network, native cryptocurrency, and robust smart contract ecosystem, Ethereum is an ideal option for public blockchain applications that requires openness and decentralization. On the other hand, Hyperledger Fabric's features such as permissioned network, modular architecture, and enterprise-focus features, making it a suitable approach in developing private, controlled environments, where confidentiality and governance are priorities. For the proposed system, the choice between these frameworks will highly depends on the specific requirements. For the current stage, Hyperledger Fabric will be primary option due to its controlled and modular architecture that can fulfil the objectives of the proposed system.

2.3.1.5. Web Development

The proposed system will leverage the accessibility and scalability of web applications for development. This section delves into each component that will be utilized for the frontend and backend development of the system.

2.3.1.5.1. HTML

HyperText Markup Language (HTML) is a standardized markup language used to develop the structure and content of web pages. It was developed by Tim Berners-Lee in 1990 after he invented the World Wide Web (WWW) in 1989 to interact over the internet (<https://www.geeksforgeeks.org/who-developed-html/>). HTML provides the basic structure of a web page using elements and tags, such as headings, paragraphs, and links. The simplicity and standardization of HTML make it the key to web development, ensuring compatibility across browsers and devices (*Who developed HTML?*, 2024).

2.3.1.5.2. CSS

Cascading Style Sheets (CSS) is a scripting language used to manage the presentation and layout of web pages. It was proposed by Hakon Wium Lie in 1994 and developed by the World Wide Web Consortium (W3C) to address the presentation of web pages that solely rely on HTML. In addition, CSS provides flexibility to developers to separate content from design, allowing for consistent styling, and responsive layouts throughout the web application and enhancing user experience across various devices (*CSS history and versions*, 2024).

2.3.1.5.3. JavaScript

JavaScript (JS) is a versatile, high-level programming language used to create interactive and dynamic web content. It was developed by Brendan Eich in 1995 and first implemented in Netscape Navigator, a popular web browser of that era. The initial naming for JavaScript was Mocha, then after ten days, it was changed to LiveScript, then finally became known as JavaScript (*History of JavaScript*, 2024). Besides, JavaScript's extensive features enable developers to create a wide range of interactive web experiences, from simple form validation to complex, asynchronous applications using AJAX. It is also supported by numerous frameworks and libraries such as React, Angular, and Vue.js.

2.3.1.5.4. PHP

Hypertext Preprocessor (PHP) is a server-side scripting language mainly used for web development which was developed by Rasmus Lerdorf in 1994. It is an interpreter language, meaning it does not require compilation. It enables developers to create and place server-side logic directly into HTML, making it ideal for tasks like form handling, database interaction, and session management (*PHP Introduction*, 2024). As of 2024, PHP is still a popular server-side scripting language for building dynamic web applications and seamlessly integrates with frontend web components like HTML, CSS, and JavaScript.

2.3.1.5.5. MongoDB

MongoDB is a non-relational (NoSQL) database designed for storing and managing semi-structured and unstructured data. It was developed by Dwight Merriman, Eliot Horowitz, and Kevin Ryan in 2007 (MongoDB, n.d.). Unlike relational databases like MySQL, MongoDB offers a document-oriented approach to data storage and retrieval. Documents store data in a

flexible format with key-value pairs, often containing nested structures and arrays. This makes MongoDB a popular choice for handling large datasets, real-time analytics, and applications with dynamic data models that evolve (Happy, 2018).

2.3.1.6. Application Programming Interface (API) Layer

The proposed system has an API layer that leverages a JavaScript-based stack optimized for blockchain services. Node.js provides a high-performance, event-driven runtime built on Chrome's V8 engine, ideal for handling many concurrent connections in a non-blocking fashion. npm (Node Package Manager) serves as the world's largest registry for JavaScript packages, enabling dependency management, reproducible builds via package.json, and seamless integration of blockchain SDKs and utilities. Finally, Express.js offers a minimal, unopinionated web framework on top of Node.js, delivering routing, middleware, and HTTP utilities for rapidly building and scaling RESTful blockchain APIs.

2.3.1.6.1. Node.js

Node.js is a free, open-source, cross-platform JavaScript runtime built on Chrome's V8 engine that executes code outside the browser with high throughput and low latency. It uses an asynchronous, event-driven architecture which is powered by a single-threaded event loop that enables non-blocking I/O operations and the ability to handle thousands of concurrent connections, making it ideal for real-time blockchain node communication and transaction broadcasting (Chintanonweb, 2024).

2.3.1.6.2. Node Package Manager (npm)

To manage dependencies and share code, Node.js projects rely on Node Package Manager (npm). It is the world's largest software registry, hosting over two million packages

that span utilities, frameworks, and blockchain software development kits (SDKs). It uses a package.json manifest to declaratively define project metadata, such as name, version, and description, list and lock dependencies with semantic versioning rules, and specify custom scripts for development and deployment workflows. By committing both package.json and the accompanying package-lock.json, developers ensure reproducible builds, consistent environments across teams, and the ability to automate tasks via npm CLI commands. This manifest serves as the blueprint for npm to install and manage packages, enabling seamless integration of blockchain SDKs, middleware, and utilities in a maintainable, version-controlled manner (T & Villa, 2022).

2.3.1.6.2. Express.js

Express.js is a fast, unopinionated, minimalist web framework for Node.js that supplies a robust set of HTTP utility methods and middleware for building RESTful APIs and web services. Its flexible middleware architecture enables modular composition of JSON parsing, CORS, authentication, logging, and other cross-cutting concerns, keeping route handlers concise and maintainable. With support for parameterized paths, route chaining, and sub-routers, Express.js facilitates clean organization of blockchain endpoints while preserving direct access to native Node.js features for fine-grained performance tuning (*Express.js Tutorial*, 2025).

2.3.2. Hardware

This section provides a comprehensive hardware requirement analysis of the proposed system.

2.3.2.1. Asus TUF FX505 Gaming Laptop

The Asus TUF FX505 Gaming Laptop is an entry-level gaming laptop produced by Asus. It was introduced in 2018 as an upgraded successor to the FX504 model. Known for its The Ultimate Force (TUF) branding, which emphasizes affordability and durability, the FX505 has offered multiple configurations to cater wide range of users, including options like AMD processor with GTX 1650/GTX 1660 Ti graphics cards and Intel Core H-series processor with GTX 1060 graphics card (Diew, 2019). The laptop that will be utilized in this proposed system development has the following configuration. Starting with the processor, which is an Intel Core i7-8750H, an 8th generation 6 cores and 12 threads processor from Intel that offers a base frequency of 2.20GHz, and up to 4.10GHz boost frequency. This laptop also came with a GTX1050 Ti Mobile graphics card which is packed with 4GB GDDR5 VRAM, 16GB of DDR4 RAM running at 2666MHz in dual channel configuration, and a 500GB NVME Gen3x4 M.2 SSD. It also features a 15.6-inch IPS panel with a 1920x1080 resolution, while maintaining a relatively lightweight design of 2.2kg compared to others in this range (Girbea, 2018). These specifications make the FX505 capable of handling most of the tasks, ranging from daily browsing to resource-intensive tasks such as networking and software development. While its older-generation CPU and GPU may lead to longer processing times compared to modern devices, it is still capable of handling heavy computational tasks, which highlights its performance and reliability, making it a suitable choice for various computing needs.

2.4. Summary

In conclusion, Chapter 2: Literature Review focused on the in-depth system analysis of three existing systems with similar features to the proposed system. Discussion on the features, advantages, and disadvantages of the existing systems were stated clearly, and comparisons between the existing systems and the proposed system were then made as tabulated in Table 2.4. Furthermore, Chapter 2 also presented an overview of the tools and technologies that will be utilized to develop the proposed system. In this section, the in-depth reviews were categorized into two primary sections: software and hardware, enabling better readability. With that, the reviews have justified that the proposed system, a blockchain-based traffic violation logging and penalty management system, is better in providing secure logging through a decentralized approach to eliminate potential vulnerabilities.

CHAPTER3: REQUIREMENT ANALYSIS AND DESIGN

3.1. Introduction

Chapter 3 will focus on the explanation of the chosen methodology for the development of proposed system. According to Marchewka (2016), project methodology is a template that provides a systematic approach in planning, managing, and executing the tasks by outlining specific phases, processes, tools, and techniques to be followed. The selection of methodology is highly depending to the nature of project, such as frequent changes, require communication, and incremental progress. Hence, this proposed system selected Scrum methodology throughout the development process. The subsequence sections will further explain the phases involved in Scrum, including requirement planning, design, development, testing, and review, demonstrating how each phase contributes to the iterative and incremental progress of the project.

3.2. Scrum Methodology

Scrum is one of the most well-known agile methodologies for project management and has gained widespread popularity in recent years. The concept of Scrum initially introduced by Takeuchi and Nonaka in 1986 on their Harvard Business Review article, where they compared high-performing, cross-functional team to a rugby team, highlighting the importance of teamwork in product development. Building on this concept, Jeff Sutherland and Ken Schwaber formalized the Scrum framework in the early 1990s, and showcased it at the OOPSLA conference in 1995 (Bultmann, 2024). Their work laid the foundation for Scrum's widespread adoption in software development and beyond.

As a project management framework, it guides teams and organizations in creating and delivering products through an iterative and incremental approach. While agile methodology

emphasizes human creativity in the development process rather than focusing on rigid procedures (Moe & Dingsøyr, 2008), Scrum builds upon these agile principles by specifically adapting them for small teams and organizations. In addition, it is founded on the three main pillars, which are transparency, inspection, and adaption as shown in Figure 3.1. Transparency plays the crucial role in scrum due to it is the foundation of trust and openness by ensuring that all the aspects of the process are visible to those responsible for the outcome. Regular inspection helps examine the product and processes to detect any potential issues early, allowing for timely adjustments during the development process. Besides, adaption enable the teams to response swiftly by making necessary adjustments to processes or products based on inspection findings, which ensure effective responses to changes and continuous improvement (Ben Salem, 2023; Hoffmann, n.d.).

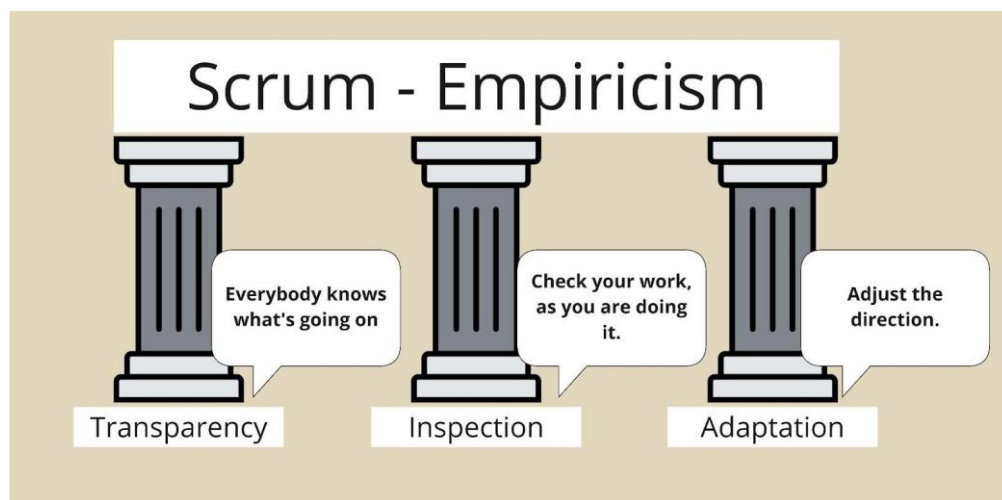


Figure 3.1. Three Pillars of Scrum (Hoffmann, n.d.).

Furthermore, the scrum framework comprises specific roles, events, and artifacts designed specifically to promote effective collaboration and deliver high-quality products. In terms of key roles, it consists of Product Owner, who is the stakeholders and manages the product backlog, Scrum Master, the leader who facilitates the process and removes impediments, and the Development Team, a cross-functional group of professionals who

collaborate to deliver product increments. In Figure 3.2, it shows the structure of the development process with scrum methodology. The sprint which typically last around two to four weeks is the central event during which a usable product increment is developed. Sprint planning meetings is the event that define the goals and select work items, while the daily stand up provides quick synchronization of activities among each other. Sprint reviews is the event where the finalized product handover to the stakeholders. Upon handover, sprint retrospectives allow the teams to reflect and improve their processes throughout the product development phase (Juillet, 2022).



Figure 3.2: Product Development Flow with Scrum (Juillet, 2022).

The scrum framework also offers three primary artifacts, which are the product backlog, sprint backlog, and increment. Product backlog is an ordered list of all desired work for the improvement of the product, while the sprint backlog is a list of selected works for the current sprint. The increment represents the completed work that meets the team's definition of completion. These artifacts provide transparency and help the team to keep track of the progress throughout the development process (Schwaber & Sutherland, 2020).

In short, Scrum methodology, which follows the Agile development phases of Requirements, Design, Development, Testing, Deployment, and Review, is chosen as the product development framework for the proposed system due to its iterative and engaging nature in the product development phase. It allows continuous assessment and improvement throughout the development lifecycle, enabling quick adaptations to changing requirements and stakeholder feedback. This approach ensures that the development team can deliver working software incrementally while maintaining quality and meeting project objectives through regular sprints and collaborative team interactions, all while adhering to the systematic flow of Agile development processes.

3.3. Requirements

In Scrum methodology, requirement planning is one of the crucial phases for the product. During this phase, it defines what the proposed system, a blockchain-based traffic violation logging and penalty management system, needs to accomplish and how it will be delivered. Requirements are identified, prioritized, and broken down into manageable tasks, which are added to the product backlog. This process ensures a clear understanding of the product goals, allowing for efficient progress toward meeting project objectives. To investigate the demand of the proposed system, a survey is conducted to collect insights from users regarding the current system and the proposed system. Additionally, this phase also involves outlining the software and hardware requirements.

3.3.1. Analysis of Similar Existing Systems

Based on the review on similar existing systems in the previous chapter, there are few key points/functionalities that need to be taken into consideration. Firstly, the ability to track traffic violation history is crucial for an advanced traffic violation logging system. It allows users to keep track of previous violations, which can be particularly useful in case of any disputes. Therefore, the system should include comprehensive history tracking functionality. This includes real-time status tracking, search and filter capabilities for historical data, and archive management for past violations and transactions. Additionally, the ability to review and reprint transaction records is highly beneficial for users. Ensuring a robust system for managing and reporting violations, along with related payments, will significantly enhance the system's reliability and transparency.

Furthermore, real-time notifications for system users, especially drivers, are essential. As the automated system integrated with traffic enforcement captures a violation, real-time

notifications should be sent to drivers, immediately alerting them that their action has violated traffic rules. This timely notification can potentially encourage drivers to correct their behaviour promptly, which can reduce the likelihood of repeated offenses. Additionally, through providing real-time updates ensures transparency and gives drivers the opportunity to address the violation as soon as it occurs, fostering greater accountability and compliance with traffic regulations.

In addition, administrative features such as the review and approval of violations, ticket generations, and detailed historical data access, should be integrated. It enhances system management and provide the authorities with tools for efficient decision-making and enforcement.

3.3.2. Analysis of Proposed System

The design and development of the proposed system is motivated by the need for an innovation solution that overcomes the drawbacks of existing systems. It is designed based on in-depth analysis and evaluation of the features present in the existing systems. Primarily, the aim of this project is to develop a transparent and accessible traffic violation logging management system by using blockchain technology, ensuring secure, tamper-proof records, and providing an automated process for issuing penalties. The proposed system consists of a web-based application used by two primary user groups, driver and traffic authorities. The main function of the web application is to allow the drivers to access and manage their traffic violation records, handle penalty payments, and manage account information. While for the administrative side, the web application acts as the control panel for the traffic authorities for managing the violation records, handle penalty administration, and monitor and analyse the violation data. The in-depth functionalities of the proposed system are tabulated in Table 3.1.

Table 3.1. Functionalities of Proposed System.

Driver	Traffic Authorities (Administrator)
• Login an account • Register an account • View violation records • View outstanding penalties • Pay violation records • View user profile • Edit user profile • Receive real-time notification • Submit appeals	• Login an administrator account • Access and review violation logs • Manual traffic violation logging • Verify and validate violation data • Manage user accounts • Process appeals

3.3.3. Questionnaire

A survey was distributed randomly to 50 respondents, primarily general drivers, to gather insights and feedback for the requirements analysis of the proposed system. The questionnaire aimed to assess the respondents' perspectives on current traffic violation management systems, the proposed system features and improvements, as well as their considerations regarding privacy and public acceptance. The questionnaire was conducted using Google Forms to ensure easy accessibility and convenience for respondents. In addition, there are 4 sections in the questionnaire, which are Section A: Demographic Information, Section B: Current System Assessment, Section C: Digital System Features and Improvements, and Section D: Privacy and Public Acceptance. A sample questionnaire is attached in **Appendix B: Sample Questionnaire** for reference. The responses collected provided valuable input for refining the system's design, addressing user needs, and identifying potential

challenges. The survey was designed to maintain the anonymity and confidentiality of all participants, ensuring unbiased and honest feedback. Analysis has been conducted based on the responses collected, and summarized in the following sub-sections.

3.3.3.1. Demographic Information

Section A focuses on the collection of basic demographic information such as age group, state of residence, driving experience, and vehicle type. It helps in understanding the background of the respondents, ensuring the questionnaire captures diverse perspectives that relevant to the target population for the proposed system.

First and foremost, the respondents were categorized into five age groups: 16–25, 26–35, 36–45, 46–55, and 56 and above. As shown in Figure 3.3. Age Group Figure 3.3, a pie chart summarized the total number of responses based on the age group. The majority of respondents fall within the 16-25 age group, accounting for 62% of the total responses. Following by 26-35 age group, which has around 24%. The remaining age groups - 36-45, 46-55, and 56 and above constitute smaller portions, each accounting for 6%, 6%, and 2%, respectively. This distribution is due to the questionnaire is primarily distributed among university students, reflecting a demographic more familiar with digital technologies and potentially more open to the adoption of digital solutions for traffic violation management.

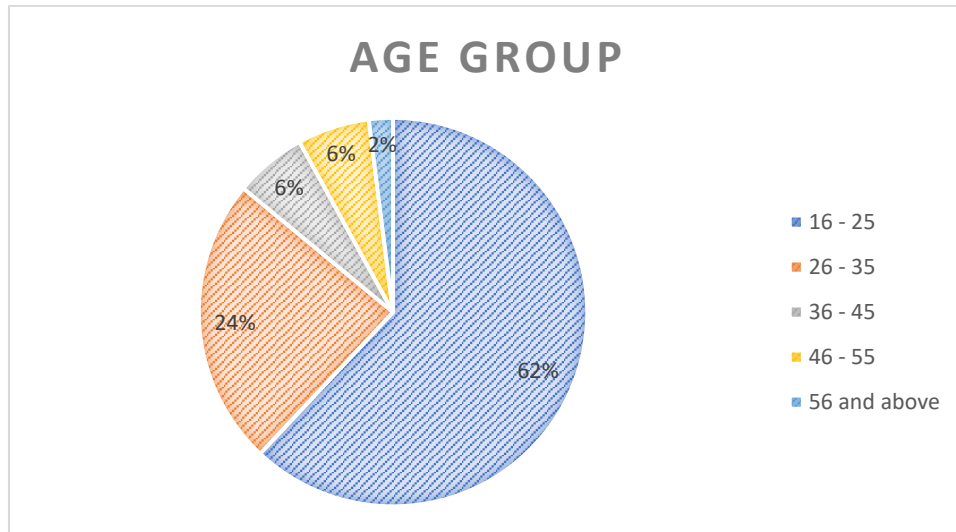


Figure 3.3. Age Group

Besides, the respondents chosen in this questionnaire came from various states across Malaysia, with the majority residing in Sarawak, which accounted for 80% of the total respondents as illustrated in Figure 3.4 . This representation from Sarawak can be attributed to the fact that the questionnaire was distributed within local networks, influencing the high response rate from this state. Followed by Perak with 10% while Selangor and Johor each contribute 4% of the total responses. The smallest percentage of respondents, which is 2% came from Sabah.

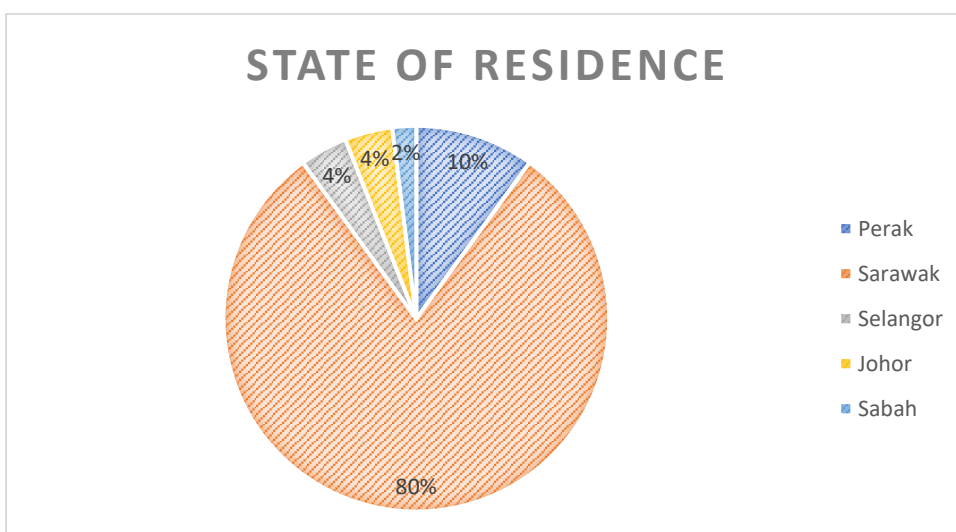


Figure 3.4. States of Residence.

In terms of driving experience illustrated in Figure 3.5, the majority of respondents which around 76% has been driving for 1 to 5 years, indicating the most participants are relatively new drivers, likely within the younger demographic reflected in the age group section. This is followed by 12% who have 6 to 10 years of driving experience. A small section of the respondents has either less than 1 year or more than 10 years of driving experience which is around 4% each. The range of driving experience provides the insights into how familiarity with road usage may influence views on traffic violation logging and penalty management.

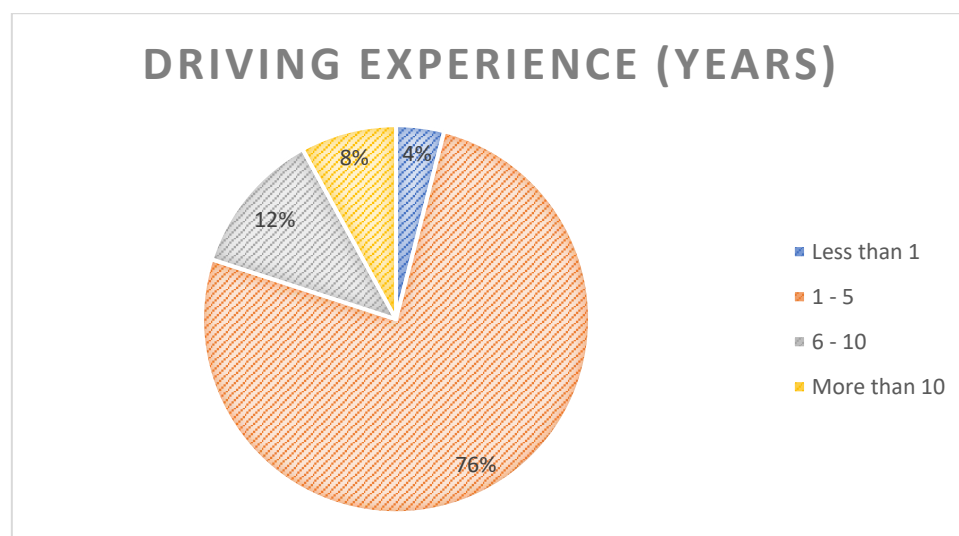


Figure 3.5. Driving Experience

Additionally, based on Figure 3.6 majority of the respondents indicated that they drive a car in daily basis, accounting 90% out of the 50 respondents. The remaining 10% are respondents who drive a motorcycle. This sub-section is to understand the type of vehicle most respondents use, which can aid in tailoring the proposed traffic violation management system to better suit the need of road users.

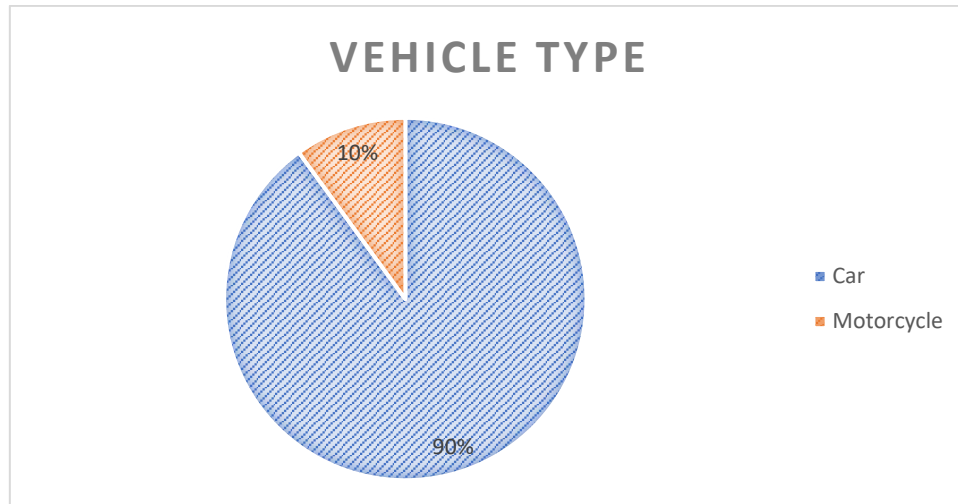


Figure 3.6. Vehicle Type

3.3.3.2. Current System Assessment

Section B evaluates the respondent's experience towards the existing traffic violation management systems. It assesses the usability level, accuracy, accessibility, and convenience. The insights gained from this section will highlights the areas that the current systems may need improvement.

Figure 3.7 depicted the analysis summary of Section B responses collected from 50 respondents. It reveals significant challenges in the existing traffic violation management system across multiple aspects. As an overview, the data shows mostly negative user experience, with the majority of respondents expressing dissatisfaction. For instance, 23 to 29 respondents indicated they were not satisfied with various aspects of the current system, particularly regarding the accessibility of violation history and payment process convenience. The ease of checking traffic summons also one of the major negative points, with 23 out of 50 respondents stating it "Not at all" easy and another 20 finding it only "Slightly" easy. The data further indicates minimal positive feedback, with almost no respondent rated any aspects as "Extremely" good and very few rating aspects as "Very" good.

Besides, the moderate satisfaction levels were relatively low, which only accounted 5 to 8 respondents across different aspects. These findings indicate that the current traffic violation management system requires improvements across all evaluated areas, with particular attention needed in making violation history more accessible, enhancing payment process convenience, and improving the overall user experience in checking traffic summons and maintaining the accuracy of the violation records.

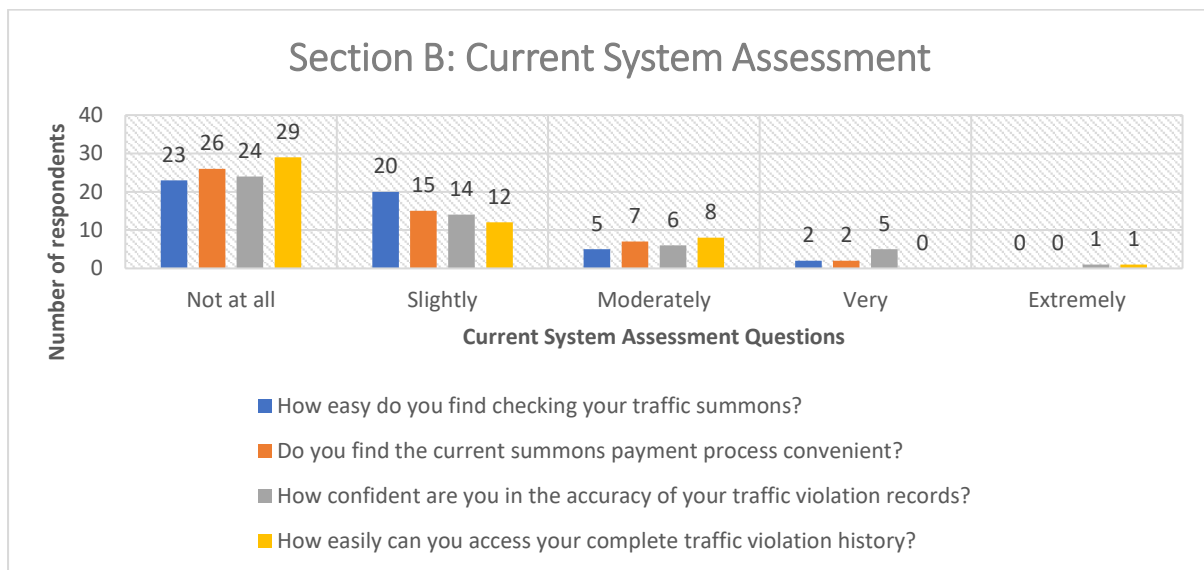


Figure 3.7. Analysis Summary of Section B

3.3.3.3. Digital System Features and Improvements

Section C focuses on the respondents' preferences and expectations for the features in a blockchain-based traffic violation management system. It explores the areas like real-time notifications, efficiency, tamper-proof records, and payment convenience to gather feedback on the potential features and usability of the proposed system.

Figure 3.8 illustrates the analysis summary for Section C: Digital System Features and Improvements. It shows a strong positive rating towards proposed system features and improvements for traffic violation management. In general, the data reveals that there are overwhelming supports across all the proposed features, with the majority of responses

concentrated in the “Strongly Agree” and “Agree” categories. The most notable statistic is 31 to 32 respondents selected “Strongly Agree” with receiving instant notifications about traffic summons, having online payment options through web application, and accessing complete traffic violation history. The automated summons issuance features also received significant support, with 31 respondents strongly agree on its implementation.

Additionally, the concept of unchangeable proof for reducing summons disputes accounted strong agreement from 25 respondents, while payment tracking feature was highly valued by 32 respondents. There was also no opposition to these proposed improvements, with zero responses in the “Strongly Disagree” category and minimal responses in the “Disagree” category. The neutral responses were also relatively low, which range from 2 to 4 responses across all the proposed features. In summary, the high positive responses suggest that implementing these digital features would significantly address the downsides identified in the current system and fulfil user expectations for a more efficient, accessible, and user-friendly traffic violation management system.

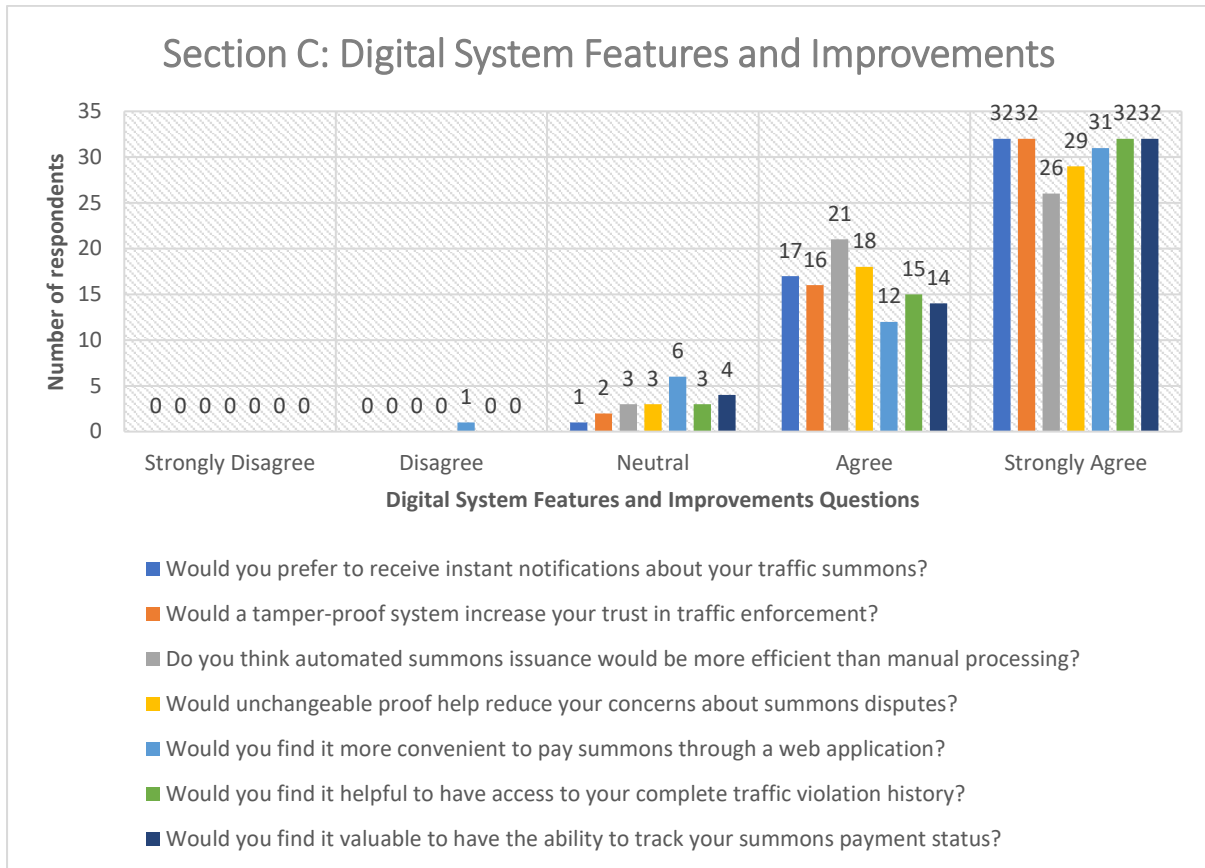


Figure 3.8. Analysis Summary for Section C

3.3.3.4. Privacy and Public Acceptance

Section D, last section of the questionnaire surveys the respondents' concerns and attitudes regarding privacy and the public acceptance of a new blockchain-based traffic violation management system. It also explores their trust in automation, willingness in the adoption of new technology, and perceived effectiveness of a digital approach in promoting traffic rule compliance.

Figure 3.9 analysed the privacy and public acceptance of the proposed traffic violation management system. It focuses on the insights about the user attitudes and concerns towards the proposed system. The data reveals a notably high level of concern about personal information security, with 31 respondents being "Extremely" concerned and 15 being "Very" concerned. However, this high concern rating in the personal information security is balanced

by a strong positive attitude towards digital system adoption, with 25 responses indicated they would find a new digital system “Extremely” trustworthy for traffic enforcement, while 17 responses rated it as “Very” trustworthy.

In addition, the comfort level with automatic summons issuance through traffic cameras also received positive feedback, with 29 respondents being “Extremely” comfortable and 15 “Very” comfortable. There is also a strong willingness for the adoption to new digital traffic summons system, accounting at 30 responses with “Extremely” willing and 18 responses with “Very” willing to embrace the changes. The potential for the new proposed system to motivate better traffic rule compliance was also reviewed positively, with 33 respondents rated with “Extremely” motivate and 11 respondents rated with “Very” motivate. Furthermore, there were no responses recorded in the “Not at all” and “Slightly” categories across all the questions in this sub-section, which suggest an overall positive recognition to the digital transformation despite privacy concerns.

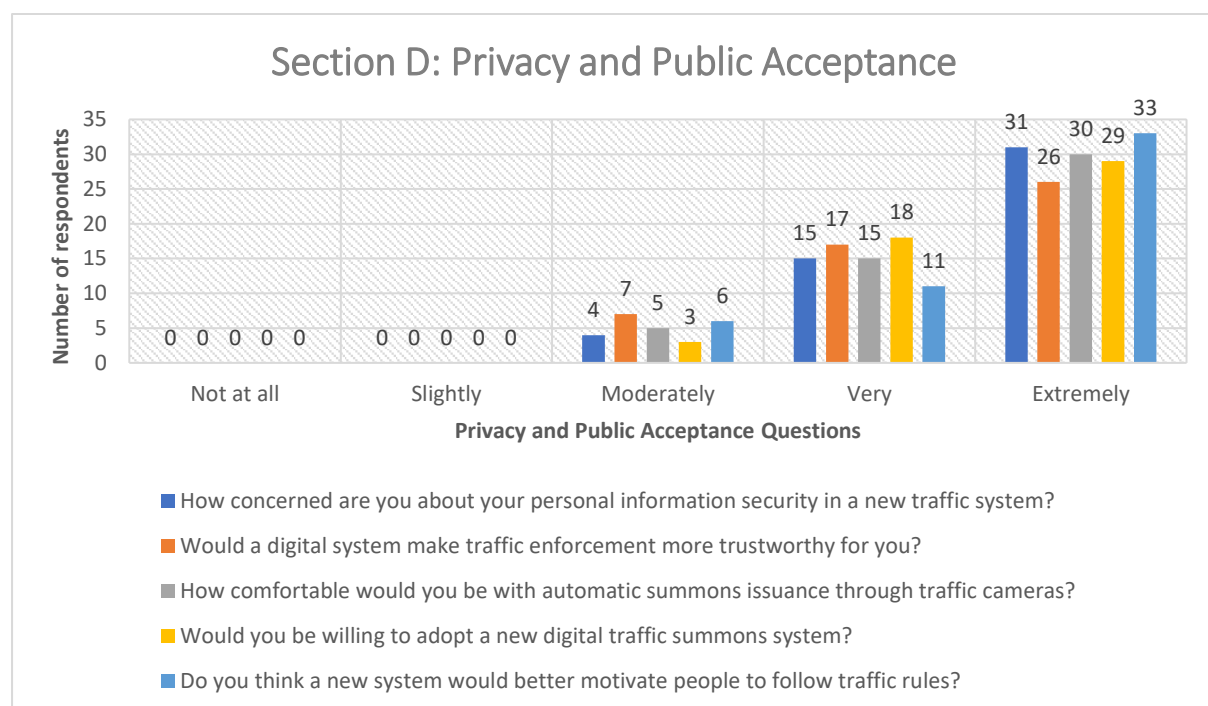


Figure 3.9. Analysis Summary for Section D

3.3.4. Software Requirements

The software requirements for the proposed blockchain-based traffic violation logging and penalty management system have been discussed thoroughly in Section 2.3.1. Software Analysis. An overview of the software requirements is tabulated in Table 3.2.

Table 3.2. Software Requirements for Proposed System.

Software	Descriptions
Ubuntu Desktop 24.04 LTS Noble Numbat	A Linux-based operating system that serves as the development and deployment platform, which offers long-term stability and security support.
Visual Studio Code	A lightweight but powerful source code editor with extensive plugin support for multiple programming languages and development tools.
Docker	A containerization platform that enables consistent development, shipping, and deployment of applications across different environments.
Blockchain Framework - Hyperledger Fabric	An enterprise-grade, permissioned (private) blockchain platform for developing secure and scalable applications with smart contract support.
Web Development	Utilize HTML, CSS, and JavaScript for creating the user interface and interactive web components of the system.
MongoDB	A NoSQL database for storing non-sensitive data and system configurations with high scalability and flexibility.
Node.js	An open-source, cross-platform JavaScript runtime built on Chrome's V8 engine that run JavaScript on the server with non-

	blocking I/O and high concurrency which is ideal for handling blockchain node communications and transaction broadcasts in real time
Node Package Manager (npm)	Package manager that installs and manages project dependencies via a manifest.
Express.js	A lightweight web framework for routing and middleware in API servers

3.3.5. Hardware Requirements

The hardware requirements for the proposed system have been discussed thoroughly in Section

2.3.2. **Hardware** Analysis. An overview of the hardware requirements is tabulated in Table 3.3.

Table 3.3. Hardware Requirements for Proposed System.

Hardware	Descriptions
Intel® Core™ i7-8750H Mobile Processor	A 6-core mobile processor with base clock speed of 2.2GHz and boost up to 4.1GHz, making it suitable for handling intensive computational tasks during the development of proposed system.
Nvidia GeForce GTX1050 Ti Mobile GPU	A dedicated graphics card with 4GB GDDR5 memory, capable of accelerating heavy computing tasks.
16GB DDR4 2666MHz RAM	Dual channel high-speed system memory providing sufficient capacity for running multiple applications, and handling blockchain operations simultaneously.

3.4. Design

After the requirement planning, the next phase in the Scrum methodology is the design stage, where the proposed system's scenario, web and software architecture, flowchart diagram, context diagram, data flow diagrams (DFDs), entity relationship diagram (ERD), data dictionaries, and wireframes of the proposed system are designed and showed. It provides the in-depth overview of the proposed system in achieving the targeted objectives.

3.4.1. Scenario Illustration

This sub-section illustrates the scenario for the blockchain-based traffic violation logging and penalty management system. The scenarios presented demonstrate the system's key functionalities and interactions between different user groups, primarily the drivers and traffic authorities. These scenarios are designed to showcase how the proposed system addresses the current challenges in traffic violation management through blockchain technology and automation.

First scenario will focus on the manual violation logging by traffic authorities as shown in Figure 3.10. When the traffic authorities notice vehicle that violate any traffic rules such as illegal parking, speeding, and illegal modification, they can use the proposed system to manually record the vehicle's information. This data is then transmitted to the blockchain network, creating a decentralized data block associated with the involved parties, namely the traffic authorities and the driver. Once the violation record is created, the system will notify the driver using the contact information linked in the database.

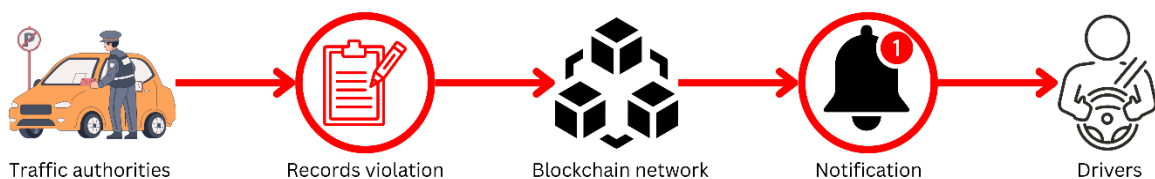


Figure 3.10. Manual Logging Scenario

Furthermore, the second scenario shown in Figure 3.11 is penalty payment and record management. Drivers can log in to the system to view their traffic violation records, including both resolved and pending violations. They can select the pending records and proceed with the payment. Once the payment is completed, the blockchain network will automatically update the payment status and generate a confirmation block, ensuring that the payment record is securely and immutably stored within the decentralized network, preventing any tampering or loss of data.



Figure 3.11. Penalty Payment Scenario

The last scenario for the proposed system can be seen in Figure 3.11., involves traffic authorities accessing the administrative page. Through this interface, traffic authorities can view detailed violation records and monitor real-time data. The system also enables authorities to generate comprehensive reports, which can be used for enforcement planning and decision-making. This functionality ensures that authorities can effectively manage and respond to traffic violations, enhancing overall traffic law enforcement.

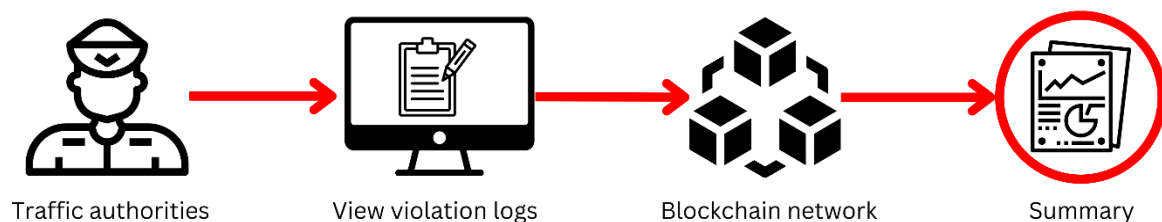


Figure 3.12. Traffic Authority Analytics and Reporting

3.4.2. Web Application and Software Architecture

The proposed system, blockchain-based traffic violation logging and penalty management system utilized web application as the development framework. The web application and software architecture for the proposed system is designed to provide a scalable, secure, and efficient platform for both use groups, namely drivers and traffic authorities.

Figure 3.13 illustrates the three main components of the system architecture: Frontend, Backend, and External Services.

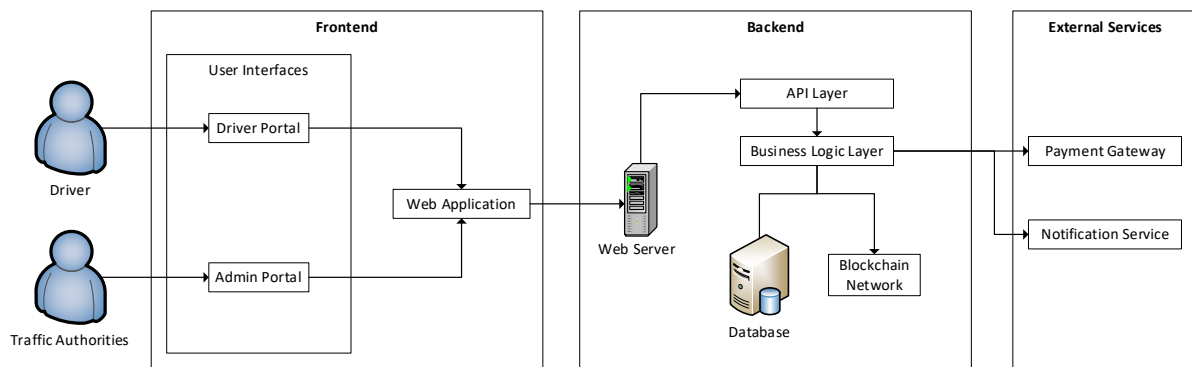


Figure 3.13. Web Application and Software Architecture.

In terms of the frontend component, it consists of two main user interfaces integrated within a single web application. The Driver Portal provides a dedicated interface for drivers to access their violation records, and make penalty payments. On the other hand, the Admin Portal offers specialized functionality for traffic authorities, which enable them to manage violations, manual logging, and generate reports. The web application servers as the central frontend component that manage user interactions, handling data presentation, and ensure responsive design across various devices and screen sizes.

For the backend, it plays a crucial role as the core of the system, which hold several critical components that work together to deliver the system's functionality. At the entry point, the web server handles incoming HTTP requests from the frontend, managing the session handling and request routing while ensuring secure communication between client and server

components. The API layer implements RESTful endpoints that facilitate communication with the frontend. It also handles request validation, authentication, and manages data formatting for consistent system interaction.

Furthermore, the business logic layer within the backend implements the core system rules and validation logic, which process traffic violation data and managing penalties while handling user authentication and authorization. The blockchain network serves as an essential component for storing immutable records of traffic violations, ensuring data integrity and transparency throughout the system. A traditional database is also used in managing user profiles, authentication data, and system configuration, which provide efficient data access and relationship management.

In addition, the system integrates with two essential external services to provide additional functionalities to the users. The payment gateway facilitates secure online payment processing, supporting various payment methods and ensuring secure transaction handling. The notification service manages real-time communications with users, providing real-time alert for violation updates to the drivers.

This architectural design ensures that the proposed system can efficiently handle traffic violation management while maintaining security, scalability, and user-friendly interaction for all the user groups. With the combination of modern web technologies, and blockchain technology creates a reliable platform for managing traffic violations in a transparent and efficient manner.

3.4.3. Flow Chart Diagram

Figure 3.14 illustrates the flow of the proposed system. It involves two key user groups: Drivers and Traffic Authorities (Admin). The system begins with a login and account creation process, followed by specific functionalities depending on the user's role.

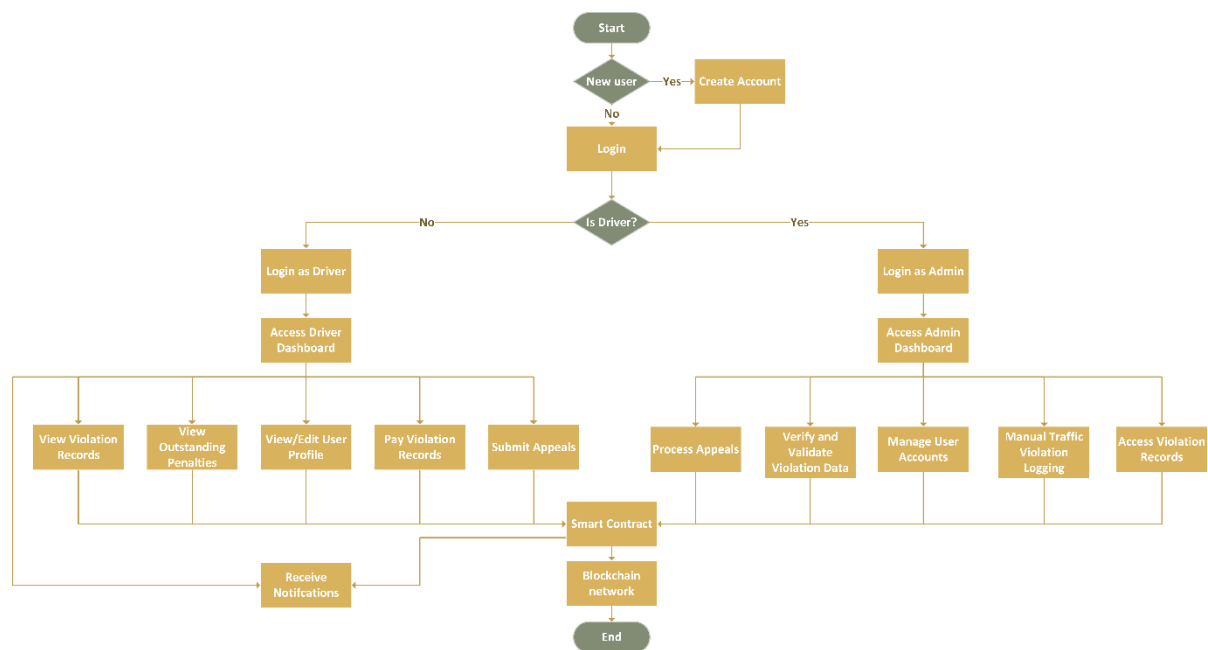


Figure 3.14. Flowchart of Proposed System

The in-depth explanation in the flowchart above are explained as follows:

1. Create account

For new drivers who just use the system for the first time, they register their account through the web application of the system.

2. Login

For existing users, they can proceed with the sign in with the following user groups: Driver and Admin.

3. Login as Driver

This login is specifically for the drivers. They can sign in to their account by inputting their account credential and will be navigated to the driver dashboard.

4. Login as Admin

This login is for the traffic authorities. They can sign in to their account using their credential, and will be navigated to the admin dashboard.

5. Access Driver Dashboard

This dashboard is designed specifically for the driver, where they can access other features of the system. The in-depth features explanation will be in the following section.

6. View Violation Records

Drivers can access a complete and detailed history of all their traffic violations. This feature displays both settled and pending records, providing a comprehensive overview of their driving behaviour and any associated infractions.

7. View Outstanding Penalties

Drivers can also view their pending summons that they have missed out, ensuring they are aware of any pending penalties. This feature provides a convenient way to stay informed about potential fines and avoid further complications.

8. View/Edit User Profile

Drivers can access their user profile for viewing and updating. This feature provides the drivers the flexibilities in managing their personal account, such as mistyping phone number, or updating new phone number.

9. Submit appeals

If drivers find out certain traffic violation record is not accurate, they can submit appeal to request for validation. This feature ensures a fair process, allowing drivers to request a review of disputed records and seek resolution.

10. Receive notification

Drivers receive automated alerts about important updates in the system, including new violation records, payment reminders, appeal status changes, and verification results. These notifications help keep both drivers and administrators informed of any actions requiring their attention.

11. Access Admin Dashboard

This dashboard is designed specifically for the traffic authorities for administrative purposes. In the dashboard, several key features can be found for traffic enforcement.

12. Manual Traffic Violation Logging

For traffic authorities, they can utilize the web application to do manual traffic violation logging if they encountered any illegal behaviours on the road, providing the flexibility and accessibility to the traffic authorities during the enforcement.

13. Access Violation Records

Traffic authorities can access all violation records logged in the system, providing real-time insights into traffic situations. This feature enables authorities to view comprehensive violation details, generate reports, and track the status of all infractions, helping them make informed decisions for traffic management and enforcement.

14. Verify and Validate Violation Data

For automated violation records, traffic authorities can review and verify the records to ensure no fault detection. This involves examining evidence, validating violation details, and confirming the accuracy of automated detections before final approval, maintaining the integrity of the traffic enforcement system.

15. Manage User Accounts

Traffic authorities can manage all the user accounts, which can be beneficial if users encounter errors in updating account or other user account issues.

16. Process Appeals

Once there is appeal received from the drivers, traffic authorities can process the appeal by verifying and validating the traffic violation record.

17. Smart Contract

An automated program that executes and records all traffic violation transactions on the blockchain, ensuring transparent and tamper-proof processing of violations, payments, and appeals.

18. Blockchain Network

A secure, decentralized database that permanently stores all system data including violation records, transactions, and user information while maintaining data integrity through its immutable ledger system.

3.4.4. Context Diagram

The context diagram, also known as level 0 data flow diagram (DFD), provides a high-level overview of the interaction between external entities and the Blockchain-Based Traffic Violation Logging and Penalty Management System.

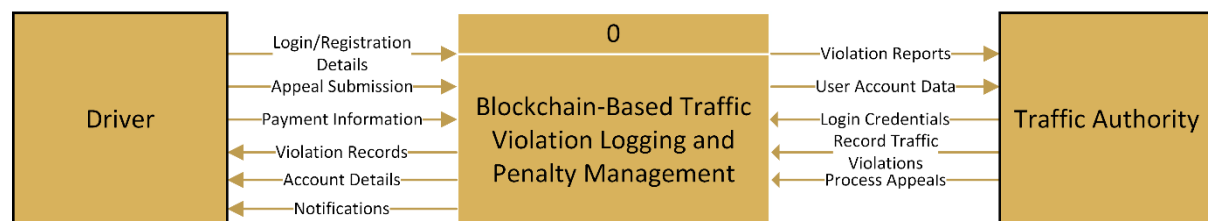


Figure 3.15. Context Diagram for Proposed System

Figure 3.15 focuses on the flow of data between the system and its two primary external entities, who are Driver and Traffic Authority. The system also acts a centralized platform for managing the traffic violations, issue penalties, and facilitate appeals. Table 3.4 and Table 3.5 shows the data flow between the system and external entities.

Table 3.4. Data Flow between System and Driver.

Data Flow	Explanation
Login/Registration Details	Drivers submit their credentials for authentication or registration.
Appeal Submission	Drivers can contest logged violations by submitting an appeal.
Payment Information	Payment details are provided for settling penalties.
Violation Records	Drivers access their logged violations.
Account Details	Drivers can view or update their account information.
Notifications	The system sends violation alerts and updates to drivers.

Table 3.5. Data flow between System and Traffic Authority.

Data Flow	Explanation
Violation Reports	The system generates detailed violation reports for monitoring and enforcement.
User Account Data	Relevant user information is provided for management or enforcement actions.
Login Credentials	Traffic authorities provide their credentials for authentication.
Record Traffic Violations	Traffic authorities log violations against drivers, such as speeding and signal jumping.
Process Appeals	Traffic authorities review and process appeals submitted by drivers.

3.4.5. Data Flow Diagram (DFD)

The data flow diagram (DFD) provides a detailed visual representation of the flow of data within the proposed system. It shows how the information is processed, stored, and exchanged between the system and the drivers and traffic authorities. This section will delve into level 1 and level 2 DFDs to highlight the system's functionalities.

3.4.5.1. Level 1 Data Flow Diagram

Figure 3.16 illustrates the level 1 DFD for the proposed system. There are 6 main processes, which are Register Account, Login Account, Manage User Account, Access Violation, Process Appeal, and Manual Violation Logging. In addition, the system incorporates two data stores: Account Details, which stores user-related account information, and the Blockchain Ledger, which securely records traffic violation data within the blockchain network.

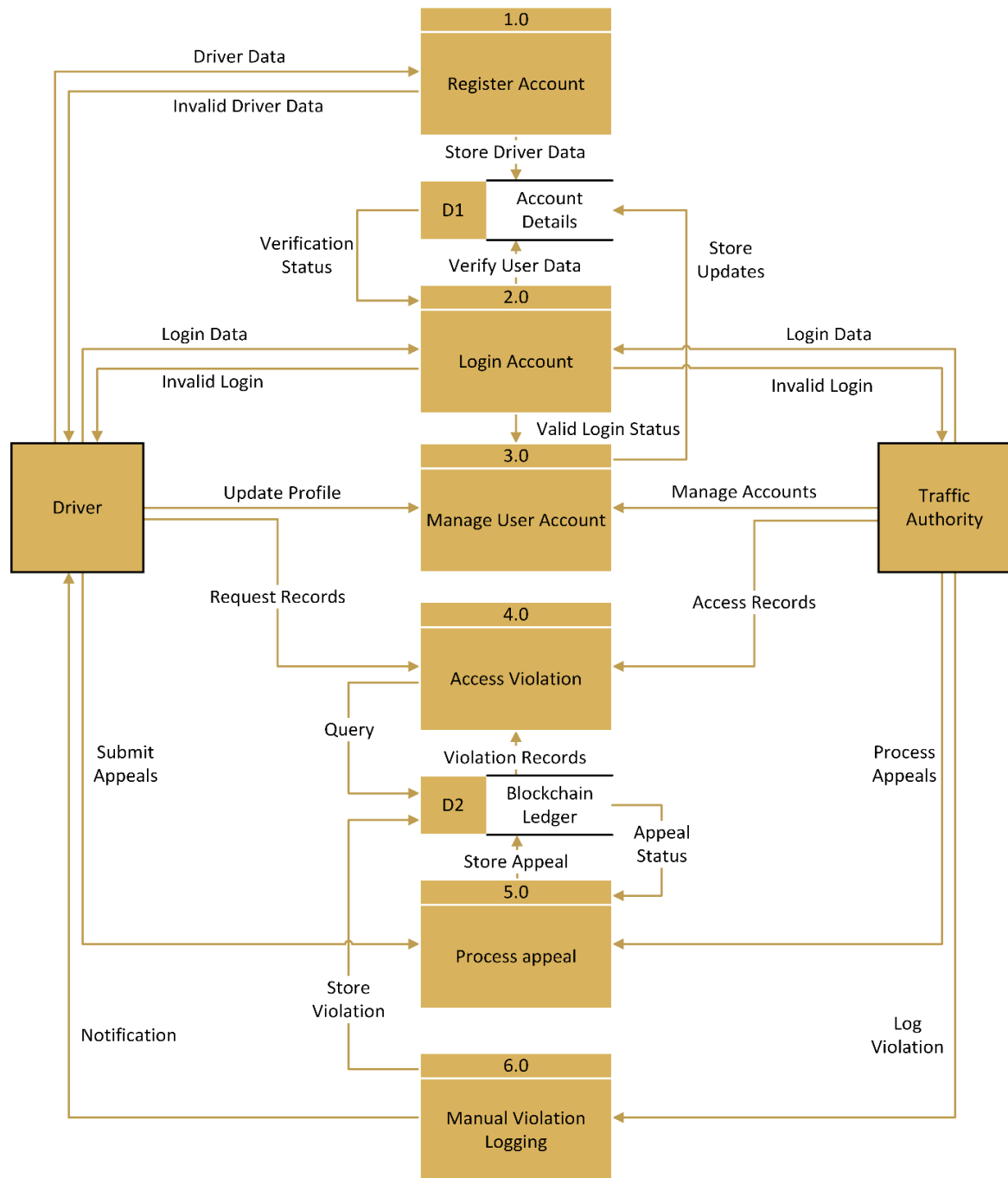


Figure 3.16. Level 1 DFD for proposed system

The following sections will delve into each process and the interactions in the proposed system:

- Process 1.0 - Register Account

This process serves as the starting point for new users, mainly drivers into the system. When they submit their personal information, it validates the data and if there is any issue, it will return an error message to the driver. Upon successful validation, the driver's information is stored in the first data store, Account Details, creating their system profile.

- Process 2.0 - Login Account

Once the driver has completed the registration, they can access the system through this process. It acts as the security gateway which handling login attempts from both drivers and traffic authorities. It verifies credentials against the Account Details data store and manages the authentication flow. Successful logins will grant the system access, while failed attempts trigger error messages to respective users.

- Process 3.0 - Manage User Account

After logging into the system, users can interact with this process for account management. It allows drivers to maintain their profile information and enables traffic authorities to manage all the user accounts in the system. Any changes made through this process are stored in the Account Details data store. In addition, this process requires users to have a valid login status from the previous process before allowing any modifications.

- Process 4.0 - Access Violation

This process handles all violation record retrieval for both drivers and traffic authorities. It provides drivers with access to their personal violation history while giving traffic authorities full access to all the violation records. It queries the second data store,

Blockchain Ledger to retrieve the requested information, ensuring all the violation data comes from an immutable and secure source.

- Process 5.0 - Process Appeal

Driver can submit appeals against the recorded violations through this process. Then, the appeals are stored in the Blockchain Ledger. Traffic authorities can review and process these appeals with their decisions also being recorded in the blockchain, creating a transparent and immutable record of the entire appeals process, from submission to resolution.

- Process 6.0 - Manual Violation Logging

This last process is only for traffic authorities to record new violations manually. When a violation is logged, it is stored in the Blockchain Ledger, creating a permanent, tamper-proof record. It also includes an automatic notification system that alerts drivers when violations are recorded against them.

3.4.5.2. Level 2 Data Flow Diagram

The level 2 data flow diagram provides a more detailed breakdown of the processes identified in the level 1 DFD. It decomposes the primary processes into sub-processes, further illustrates the detailed data flows and interactions within the system. This level of detail helps to better understand the relationships between the component of the system functions and the data processing at in-depth level.

3.4.5.2.1. Level 2 Data Flow Diagram for Process 1.0

Figure 3.17 shows the level 2 DFD for Process 1.0 - Register Account. It consists of 4 sub-processes which are Input Driver Details, Validate Driver Data, Generate Account and Store Account Details.

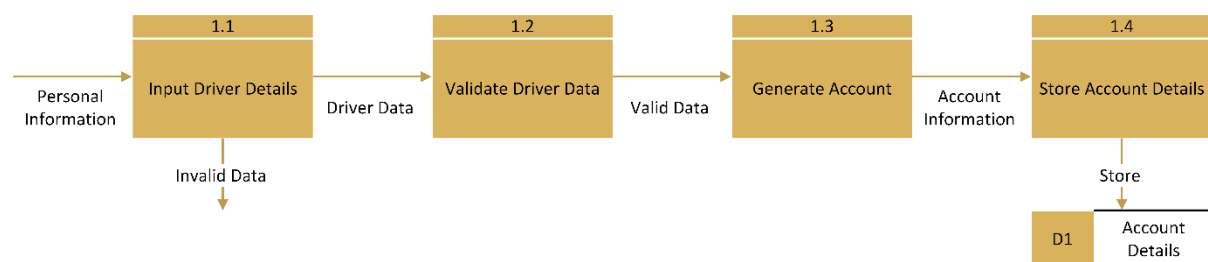


Figure 3.17. Level 2 DFD for Process 1.0

In Process 1.1 - Input Driver Details receives personal information from the drivers, which includes their identification, contact details, and license information. This raw data is then moves to Process 1.2 - Validate Driver Data, which performs validation check on the submitted information. If the validation fails, the system returns error messages to the driver. For valid data, Process 1.3 - Generate Account creates a new user account with appropriate access levels and security credentials. Lastly, Process 1.4 - Store Account Details securely saves the new account information in the Account Details data store.

3.4.5.2.2. Level 2 Data Flow Diagram for Process 2.0

Figure 3.18 illustrates the level 2 DFD for Process 2.0 - Login Account. It has 4 sub-processes for managing the system log in function for both entities, which are drivers and traffic authorities. The 4 sub-processes are Accept Login Credentials, Validate Credentials, Generate Session, and Log Login Activity.

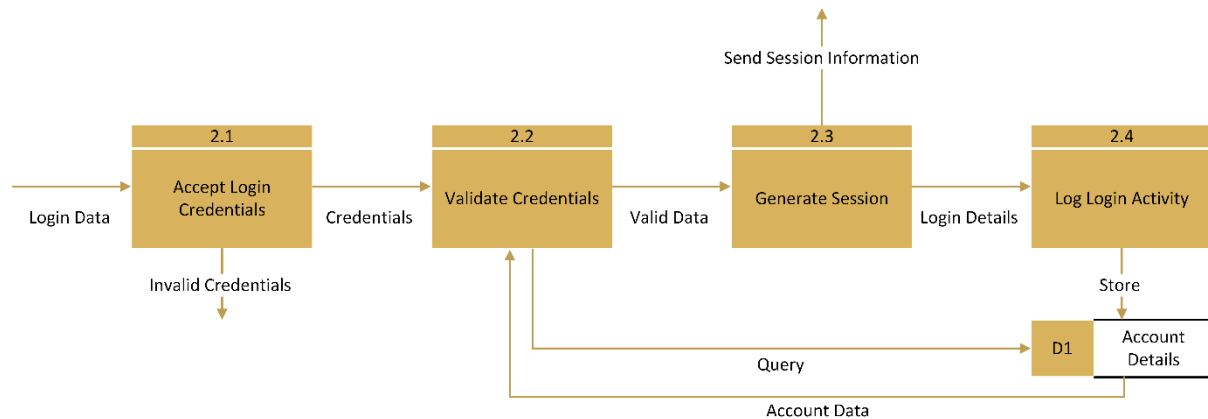


Figure 3.18. Level 2 DFD for Process 2.0

The process begins with Process 2.1 - Accept Login Credentials, which collects the login data such as username and password from both drivers and traffic authorities. These credentials flow to Process 2.2 - Validate Credentials, which queries the Account Details data store to verify the user's identity. Upon successful validation, Process 2.3 - Generate Session creates a secure session for the authenticated user. Process 2.4 - Log Login Activity records all the login attempts and stored in the Account Details data store for security monitoring and audit purposes.

3.4.5.2.3. Level 2 Data Flow Diagram for Process 3.0

In Figure 3.19, the level 2 DFD for Process 3.0 Manage User Account is shown. It consists of 4 sub-processes, which are Accept Update Request, Validate Changes, Update Profile, and Notify Changes.

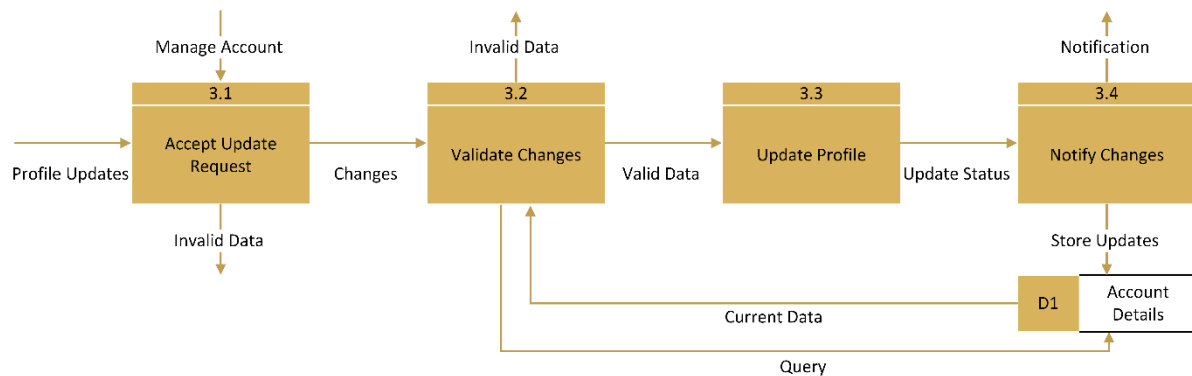


Figure 3.19. Level 2 DFD for Process 3.0

It starts with the first sub-process, Process 3.1 - Accept Update Request, where it handles incoming modification requests from both user groups. These requests move to Process 3.2 - Validate Changes, which ensures the proposed updates meet the system requirements, preserve authenticity and maintain data integrity. The valid changes then proceed to Process 3.3 - Update Profile, which modifies the stored information in the Account Details data store. Finally, Process 3.4 - Notify Changes sends confirmation messages to relevant users about the successful updates.

3.4.5.2.4. Level 2 Data Flow Diagram for Process 4.0

Figure 3.20 illustrates the level 2 DFD for Process 4.0, which is Access Violation. This process can breakdown into 4 sub-processes, which are Accept Query Request, Verify Access Rights, Retrieve Records, and Format Results.

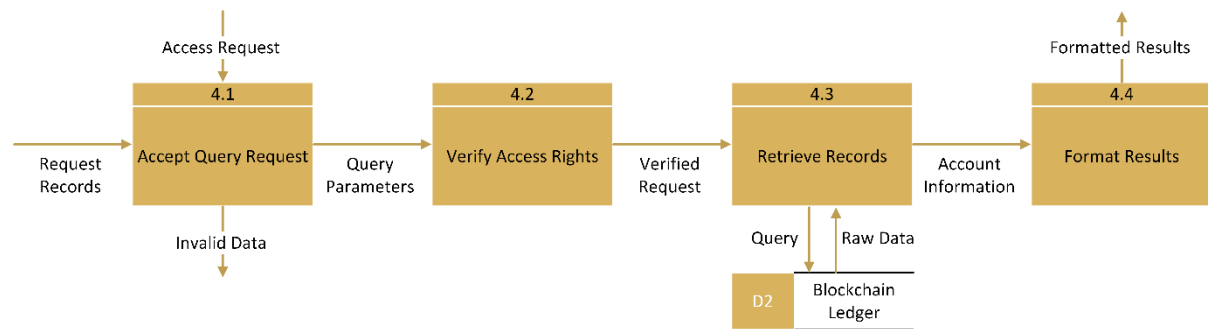


Figure 3.20. Level 2 DFD for Process 4.0

The flow begins with Process 4.1 - Accept Query Request, which collect record requests from both drivers and traffic authorities, and handles initial data validation. If the data is invalid, it will return to the user. Valid query parameters are passed to Process 4.2 - Verify Access Rights, which checks if the user has proper permissions to access the requested records. Once verified, Process 4.3 - Retrieve Records queries the Blockchain Ledger data store to fetch the relevant violation records. The raw data from the data store is then passed to Process 4.4 - Format Results, which organized and structures the data into a user-friendly format before presenting the formatted results to the users.

3.4.5.2.5. Level 2 Data Flow Diagram for Process 5.0

Figure 3.21 shows the level 2 DFD for Process 5.0 - Process Appeal, which consists of 5 sub-processes.

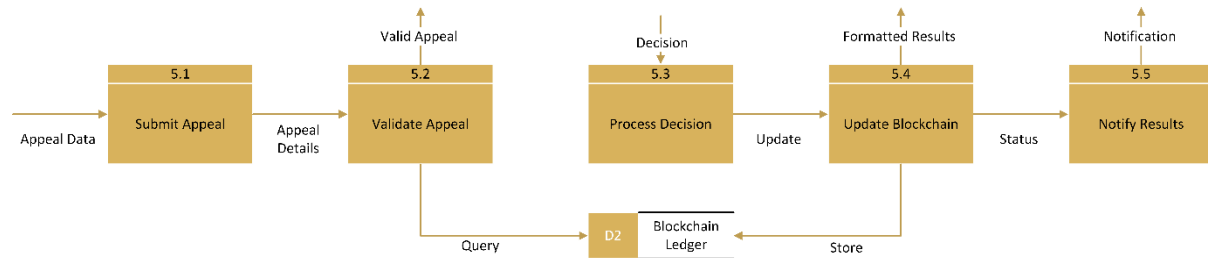


Figure 3.21. Level 2 DFD for Process 5.0

The appeal process starts with Process 5.1 - Submit Appeal, where the system accepts appeal data from drivers. The appeal details are passed to Process 5.2 - Validate Appeal, which verifies the appeal's validity by querying the Blockchain Ledger data store for the original violation. Valid appeals move to Process 5.3 - Process Decision, where traffic authorities review and make determinations on the appeal. The decision flows to Process 5.4 - Update Blockchain, where the appeal outcome is recorded in the Blockchain Ledger data store. Lastly, Process 5.5 - Notify Results informs relevant users about the appeal status through sending notification.

3.4.5.2.6. Level 2 Data Flow Diagram for Process 6.0

Figure 3.22 illustrates the level 2 DFD for Process 6.0, which is Manual Violation Logging.

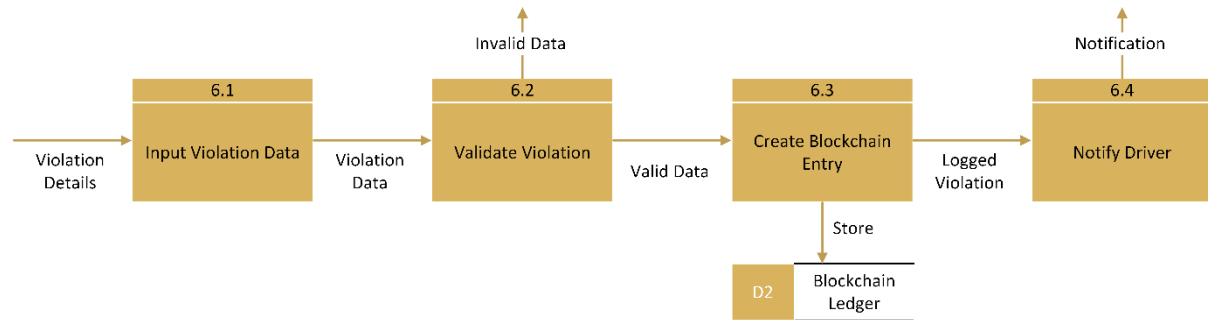


Figure 3.22. Level 2 DFD for Process 6.0

The violation logging begins with Process 6.1 - Input Violation Data, where traffic authorities enter violation details. This data flows to Process 6.2 - Validate Violation for verification. If the data is invalid, it will return for correction. After that, valid violation data proceeds to Process 6.3 - Create Blockchain Entry, where it generates and stores the violation record in the Blockchain Ledger data store. After successful storage, Process 6.4 - Notify Driver sends automated notifications to the affected drivers about their recorded violations.

3.4.6. Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) is a graphical representation that shows the relationships between entities in a database system. For the proposed system, the ERD is used to show how different components of the traffic violation system interact with each other through their relationships and attributes.

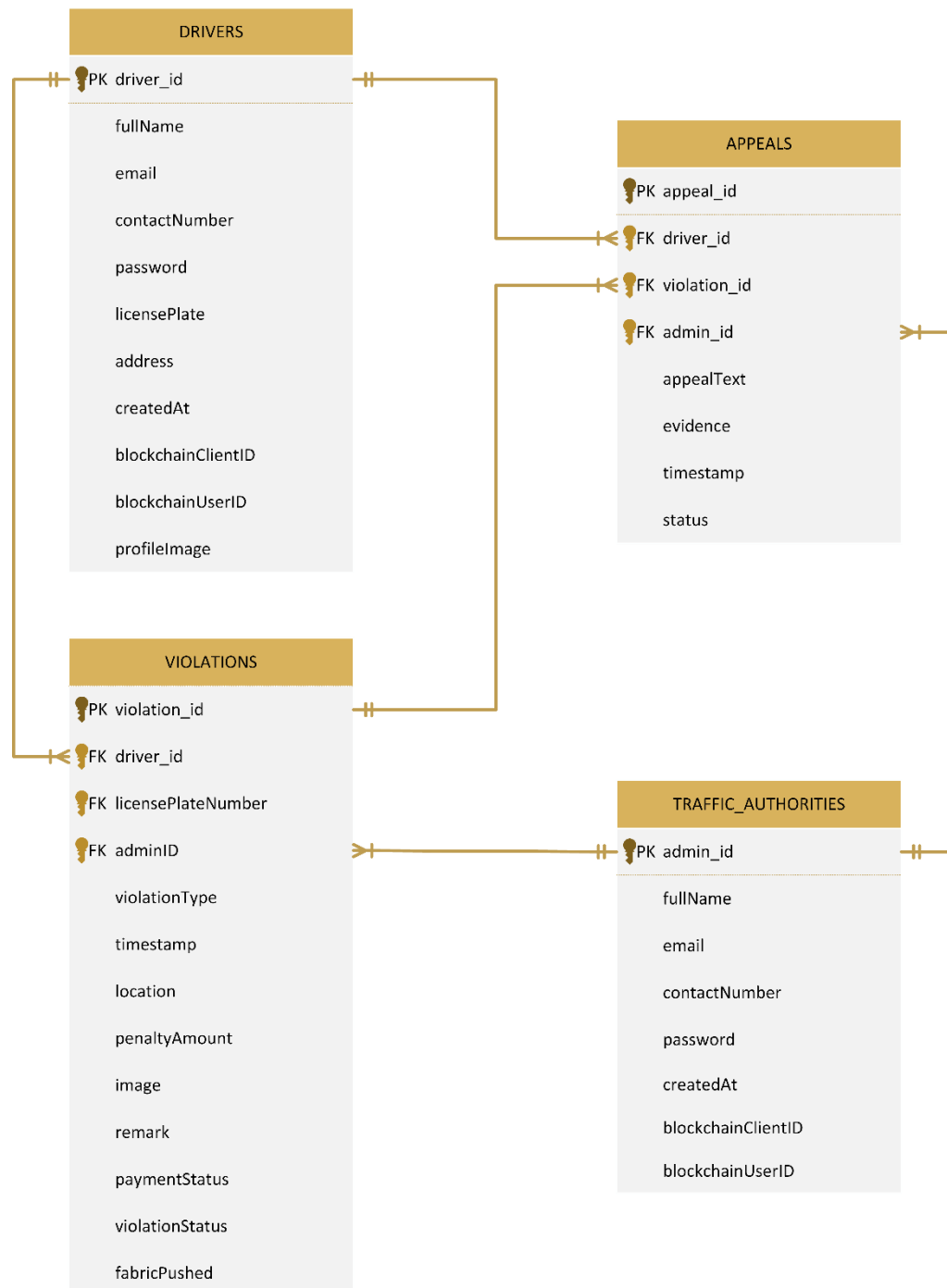


Figure 3.23. Entity Relationship Diagram (ERD) for Proposed System

Figure 3.23 shows the ERD for the proposed system. There are two main entities involved in this system, which are driver and traffic authority. There are also four tables: DRIVERS, TRAFFIC_AUTHORITIES, VIOLATIONS, and APPEALS. In addition, these tables are interconnected through one-to-many relationships, which are:

- A driver can have multiple violations.
- A driver can have multiple appeals.
- A traffic authority can manage multiple violations.
- A traffic authority can manage multiple appeals.
- A violation can have multiple appeals.

3.4.7. Data Dictionary

The data dictionary acts as a comprehensive repository that defines and describes the data element utilized within the system. For the proposed system, it outlines the structure, attributes, and constraints of the system's primary data tables. The following sub-sections will delve into the data dictionary for each table in the proposed system.

3.4.7.1. Data Dictionary for Driver Table

Table 3.6. Data Dictionary for Driver Table

Field Name	Data Type	Length	Constraints	Description
driverID	ObjectId	-	Primary Key Not Null	Unique identifier for driver
licensePlate	String	-	Not Null	Driver's vehicle license plate number
fullName	String	-	Not Null	Driver's full name
email	String	-	Not Null	Driver's email address

contactNumber	String	-	Not Null	Driver's contact number
address	String	-	Not Null	Driver's current address
password	String	-	Not Null	Driver's account password
createdAt	Date	-	Not Null	Date of account creation
blockchainClientID	String	-	Not Null	Driver's Blockchain Client ID
blockchainUserID	String	-	Null	Driver's Wallet ID
profileImage	String	-	Null	Driver's Profile Image in Base64 string

Table 3.6 shows the data objects in the Driver table. It stores the essential driver information which can be crucial for authentication, violation logging, and penalty management. There are 11 field in total:

1. *driverID* is the unique identifier for each driver and serves as the primary key for the Driver table. It is of type *ObjectId*, ensuring uniqueness and is marked as NOT NULL.
2. *licensePlate* stores the vehicle's license plate number associated with the driver. It is of type *String* and is required (NOT NULL).
3. *fullName* captures the driver's full name. This is a *String* field and cannot be null.
4. *email* contains the driver's email address. It is a *String* field and is required for account communication and verification purposes.
5. *contactNumber* stores the driver's mobile or contact number. It is a *String* field and is marked as NOT NULL.
6. *address* contains the current residential address of the driver. This is a *String* field and is required.

7. *password* holds the driver's password, typically stored in an encrypted format. It is a String field and must not be null.
8. *createdAt* stores the timestamp indicating when the driver's account was created. This field is of type Date and is required (NOT NULL).
9. *blockchainClientID* is a unique identifier that maps the driver to their Blockchain Client account. It is a String field and is mandatory.
10. *blockchainUserID* refers to the driver's Blockchain Wallet ID. This is a String field and may be null if not yet registered.
11. *profileImage* stores the driver's profile photo in Base64-encoded string format. It is a String field and is optional.

3.4.7.2. Data Dictionary for Traffic Authority Table

Table 3.7. Data Dictionary for Traffic Authority Table

Field Name	Data Type	Length	Constraints	Description
adminID	String	-	Primary Key Not Null	Unique identifier for the traffic authority
fullName	String	-	Not Null	Traffic authority's full name
email	String	-	Not Null	Traffic authority's email address
contactNumber	String	-	Not Null	Traffic authority's contact number
password	String	-	Not Null	Traffic authority's account password
createdAt	Date	-	Not Null	Date of account creation
blockchainClientID	String	-	Not Null	Admin's Blockchain Client ID
blockchainUserID	String	-	Not Null	Admin's Wallet ID

Table 3.7 depicts the data objects stored in the Traffic Authority table for administrative purposes. There are 8 fields in total:

1. *adminID* serves as the unique identifier for each traffic authority and acts as the primary key for the table. It is a String field and must be unique and NOT NULL.
2. *fullName* captures the full name of the traffic authority. This is a String field and is required.
3. *email* records the email address of the traffic authority. It is a String field marked as NOT NULL for communication and authentication purposes.

4. *contactNumber* stores the traffic authority's contact number. It is a String field and cannot be null.
5. *password* holds the traffic authority's account password, typically stored in an encrypted format. It is a String field and is required.
6. *createdAt* represents the date and time when the traffic authority's account was created. It is a Date field and is mandatory.
7. *blockchainClientID* is a unique identifier used to associate the traffic authority with their Blockchain Client account. This is a String field and must not be null.
8. *blockchainUserID* refers to the Wallet ID assigned to the traffic authority within the blockchain system. This is a String field and is also required.

3.4.7.3. Data Dictionary for Violation Table

Table 3.8. Data Dictionary for Violation Table

Field Name	Data Type	Length	Constraints	Description
violationID	String	-	Primary Key Not Null	Unique identifier for the violation
driver_id	String	-	Foreign Key Not Null	Reference to the DRIVER table
licensePlate	String	-	Foreign Key Not Null	Reference to the driver's license plate
admin_id	String	-	Foreign Key Not Null	Reference to the TRAFFIC_AUTHORITIES table
violationType	String	-	Not Null	Type of traffic violation

timestamp	String	-	Not Null	Date and time of the violation
location	String	-	Not Null	Location where the violation occurred
penaltyAmount	String	-	Not Null	Amount of fine imposed
image	String	-	Not Null	Violation's proof
remark	String	-	Not Null	Violation's description
paymentStatus	Boolean	-	Not Null	Violation's payment status
violationStatus	Boolean	-	Not Null	Violation's status
fabricPushed	Boolean	-	Not Null	Flag for validating the existence of violation records on the ledger.

Table 3.8 outlines the data objects in the Violation table. It is responsible for the secure violation record management. There are 13 fields in total:

1. *violationID* is the unique identifier for each violation and serves as the primary key. It is a String field, marked as NOT NULL and unique.
2. *driver_id* is a foreign key referencing the Driver table. It links the violation to the associated driver. This field is of type String and is required (NOT NULL).
3. *licensePlate* acts as a foreign key referencing the driver's vehicle license plate. It is a String field and must not be null.
4. *admin_id* is a foreign key referencing the Traffic Authority table, indicating the official who logged or validated the violation. It is a String field and is mandatory.
5. *violationType* specifies the category or type of traffic violation committed (e.g., speeding, illegal parking). This is a String field and is marked as NOT NULL.

6. *timestamp* stores the exact date and time when the violation occurred. It is a String field and is required.
7. *location* indicates the place where the violation was recorded. This is a String field and cannot be null.
8. *penaltyAmount* defines the amount of fine imposed for the violation. It is a String field (typically representing currency) and is marked as NOT NULL.
9. *image* stores visual evidence of the violation, typically in a Base64-encoded string or image file path. It is a String field and must not be null.
10. *remark* provides additional information or a description of the violation. This is a String field and is mandatory.
11. *paymentStatus* is a Boolean field indicating whether the penalty has been paid (true) or not (false). It is marked as NOT NULL.
12. *violationStatus* is a Boolean field reflecting the current status of the violation record. This field is required.
13. *fabricPushed* is a Boolean flag that indicates whether the violation record has been successfully pushed to and validated on the blockchain ledger. It is a required field (NOT NULL).

3.4.7.4. Data Dictionary for Appeal Table

Table 3.9. Data Dictionary for Appeal Table

Field Name	Data Type	Length	Constraints	Description
appealID	String	-	Primary Key Not Null	Unique identifier for the appeal
driverID	String	-	Foreign Key Not Null	Reference to the DRIVERS table
violationID	String	-	Foreign Key Not Null	Reference to the VIOLATIONS table
timestamp	String	-	Not Null	Date and time of the appeal submission
appealText	String	-	Not Null	Detailed reason for the appeal
status	String	-	Not Null	Current status of the appeal
evidence	String	-	Not Null	Appeal's Supporting Proof

Table 3.9 shows the data objects stored in the Appeal table, which handles the management of appeals submitted by drivers. It consists of 7 fields:

1. *appealID* is the unique identifier for each appeal and serves as the primary key for the table. It is a String field, marked as NOT NULL and unique.
2. *driverID* acts as a foreign key referencing the Drivers table. It links the appeal to the respective driver and is a required String field.
3. *violationID* is a foreign key referencing the Violations table. It establishes the connection between the appeal and the related traffic violation. This is a String field and is marked as NOT NULL.

4. *timestamp* stores the date and time when the appeal was submitted. It is a String field and is required.
5. *appealText* contains the detailed reason for the appeal provided by the driver. This is a String field (or Text, depending on your database system) and must not be null.
6. *status* indicates the current status of the appeal, such as "Pending", "Approved", or "Rejected". This is a String field and is required.
7. *evidence* holds supporting proof or documentation submitted by the driver to support the appeal. It is a String field, often encoded as a file path or Base64 string, and is marked as NOT NULL.

3.4.8. High-fidelity Wireframe Design

Wireframes are visual representations of a user interface, which outline the structure, layout, and functionality of a system. In this section, high-fidelity wireframe for the proposed system will be shown, providing a comprehensive visualization of the user interfaces for the web application. These wireframes serve a blueprint for the system's user experience, ensuring an intuitive and user-friendly interface for all the target user groups.

3.4.8.1. Web Application Wireframe for Multiple Users

This sub-section will focus on the web application wireframe for both user groups, which are drivers and traffic authorities. When the user enters the system, they will be greeted with the login page as shown in Figure 3.24. User can enter their credentials such as username and password to sign into the system.

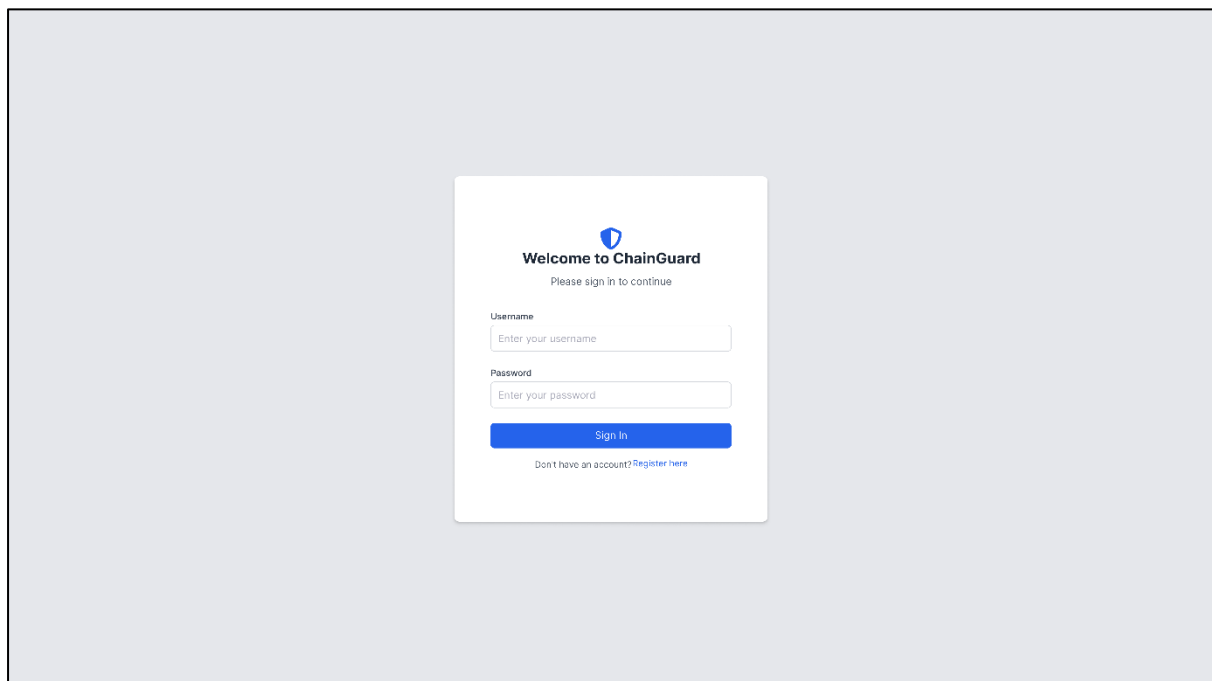
A high-fidelity wireframe of a user login page. The page has a light gray background. In the center, there is a white rectangular card with rounded corners. At the top of the card is a blue shield icon with a white checkmark inside. Below the icon, the text "Welcome to ChainGuard" is displayed in bold, followed by "Please sign in to continue" in a smaller font. Below this, there are two input fields: "Username" with the placeholder text "Enter your username" and "Password" with the placeholder text "Enter your password". Below the input fields is a blue button with the text "Sign In" in white. At the bottom of the card, there is a link that says "Don't have an account? Register here".

Figure 3.24. User Login Page

3.4.8.1. Web Application Wireframe for Drivers

This sub-section will focus on the web application wireframe for drivers. For new user to the system, driver will be navigated from the login page to this interface as shown in Figure 3.25. This interface allows the driver to register their account and uses the system. Driver's personal information such as full name, email, contact number, and vehicle license plate number is required for this process. Once completed, driver can interact with the "Register" button for validating and creating a new account.

The wireframe shows a registration form titled "Driver Registration". It contains the following fields:

- Full Name
- Email
- Contact Number
- Driver ID
- Vehicle License Plate
- Password
- Address

A blue "Register" button is located at the bottom of the form.

Figure 3.25. Registration Page

Figure 3.26 shows the main dashboard that appears after a driver successfully logs into their account. The interface features any pending traffic violations, allowing drivers to quickly view their unsettled infractions. From here, users can select individual violations to proceed to the payment page, where they can settle their fines.

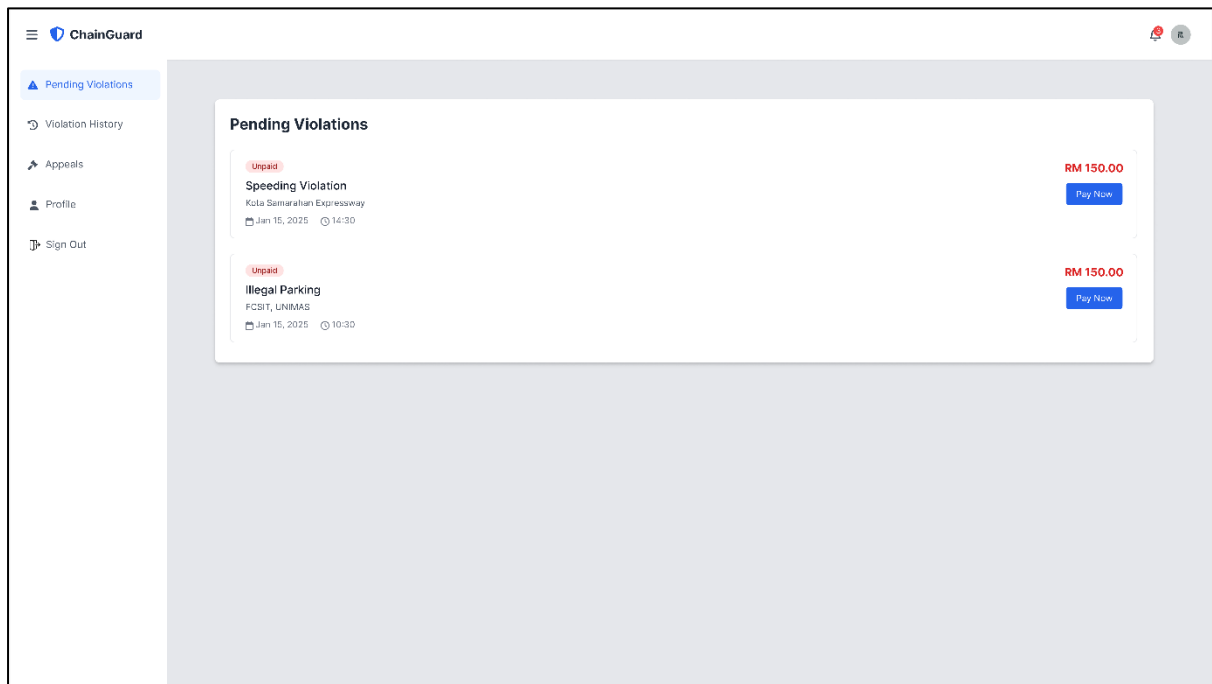
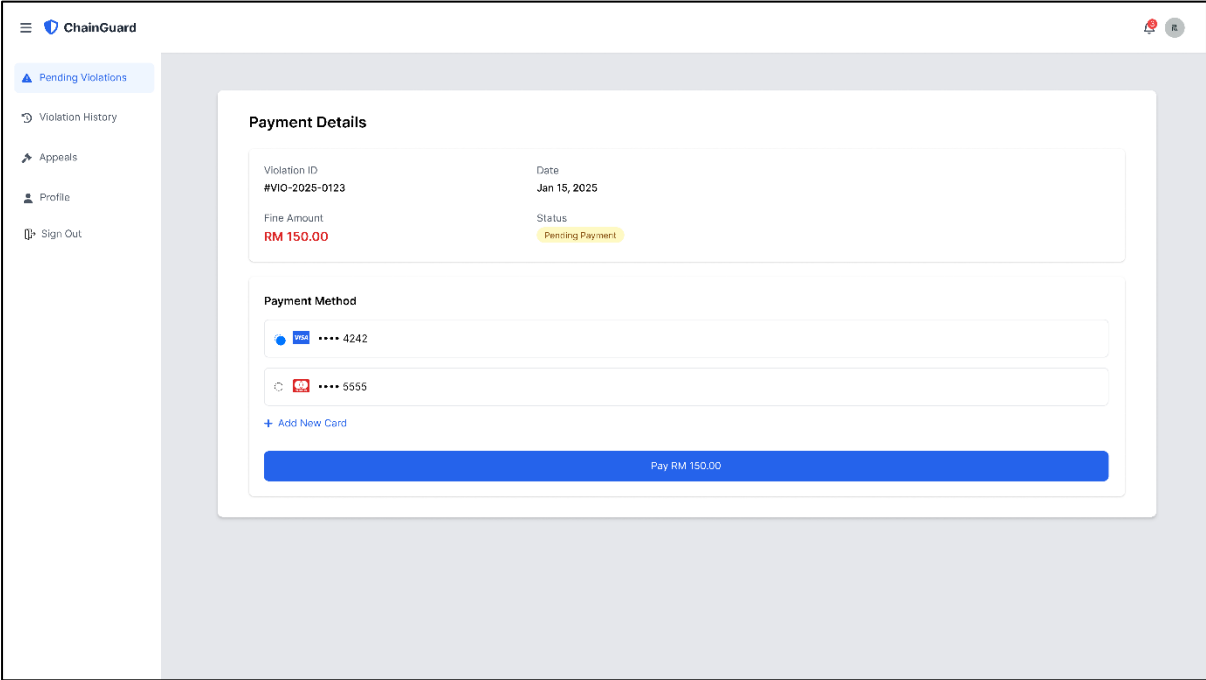


Figure 3.26. Pending Violations Page

Figure 3.27 showcases the payment interface wireframe, where users can settle their outstanding violation fines. The system provides multiple popular payment methods for convenience. After entering their payment details, users are redirected to a secure external payment gateway to complete the transaction. Upon successful payment, the system automatically updates the pending violations list and confirms the payment completion with a notification to the user.



The wireframe shows a web application interface for ChainGuard. On the left is a sidebar with a hamburger menu icon and the ChainGuard logo. The sidebar contains the following links: 'Pending Violations' (highlighted with a blue background and a triangle icon), 'Violation History' (with a magnifying glass icon), 'Appeals' (with a star icon), 'Profile' (with a person icon), and 'Sign Out' (with a door icon). The main content area is titled 'Payment Details' and contains a form with the following sections:

- Payment Details:** A table with two columns. The first column contains 'Violation ID' with the value '#VIO-2025-0123' and 'Fine Amount' with the value 'RM 150.00'. The second column contains 'Date' with the value 'Jan 15, 2025' and 'Status' with the value 'Pending Payment' (highlighted in yellow).
- Payment Method:** Two input fields for card numbers. The first field has a Visa logo and the number '**** 4242'. The second field has a Mastercard logo and the number '**** 5555'. Below these fields is a link '+ Add New Card'.
- Payment Button:** A large blue button at the bottom of the form with the text 'Pay RM 150.00'.

Figure 3.27. Payment Page

Figure 3.28 illustrates the violation history dashboard, which provides users with a detailed view of their past infractions. The interface features a searchable database of previous violations, enhanced with filtering capabilities for efficient record retrieval. Each violation entry includes essential details such as the date of occurrence, type of violation, location, fine amount, and status. This organized layout ensures users can easily access and review their complete violation history at a glance.

Date	Violation Type	Location	Fine	Status
Jan 15, 2025	Speeding	FSKTM, UNIMAS	\$150.00	Unpaid
Dec 20, 2024	Parking	CTF1, UNIMAS	\$75.00	Paid

Figure 3.28. Violation History Page

Figure 3.29 shows the submit appeal dashboard, where users can appeal incorrectly recorded violations. The dashboard features a comprehensive form where users can provide their violation ID, incident date, and select from standardized appeal reasons. Users can also include a detailed explanation of their case and upload supporting evidence to strengthen their appeal. Once all required fields are completed, the appeal is submitted for review by traffic authorities, who will process and evaluate the case.

ChainGuard

▲ Pending Violation

🕒 Violation History

🔍 **Appeals**

👤 Profile

🚪 Sign Out

Submit Appeal

Violation ID

Violation Date
mm/dd/yyyy

Reason for Appeal
Incorrect Information

Detailed Explanation

Supporting Evidence

Upload a file or drag and drop
PNG, JPG, PDF up to 10MB

Cancel Submit Appeal

Figure 3.29. Appeal Submission Page

Figure 3.30 showcases the edit user profile dashboard, where drivers can maintain their personal information. The interface provides fields for updating essential details including name, email address, contact number, residential address, and vehicle license plate information. Users can easily modify any outdated information and save their updates with a single click using the "Save Changes" button, ensuring their profile remains current.

The screenshot displays the 'Edit Profile' page within the ChainGuard application. On the left, a sidebar menu includes options for 'Pending Violation', 'Violation History', 'Appeals', 'Profile' (which is currently selected and highlighted in blue), and 'Sign Out'. The main area of the page is a light gray box titled 'Edit Profile'. Inside this box, there is a circular profile picture placeholder with a camera icon. Below the picture, there are six text input fields labeled 'First Name', 'Last Name', 'Email', 'Phone Number', 'Address', and 'License Plate'. A blue 'Save Changes' button is positioned at the bottom right of the form area.

Figure 3.30. Edit User Profile Page

3.4.8.2. Web Application Wireframe for Traffic Authorities

This sub-section focuses on the web application wireframes for traffic authorities (administrator). It consists of 5 wireframes which are Admin Dashboard, Manual Violation Logging, Violation Record, Appeal Management and User Account Management.

Figure 3.31 shows the administrator dashboard, which serves as a comprehensive analytics interface for violation management. The interface presents critical metrics including total violations recorded, revenue from collected fines, number of pending appeals, and active user count. Two visual representations enhance data interpretation: a breakdown of violations by category and a graph showing monthly violation trends. For reporting purposes, administrators can export these analytical insights, which facilitates documentation and further analysis.

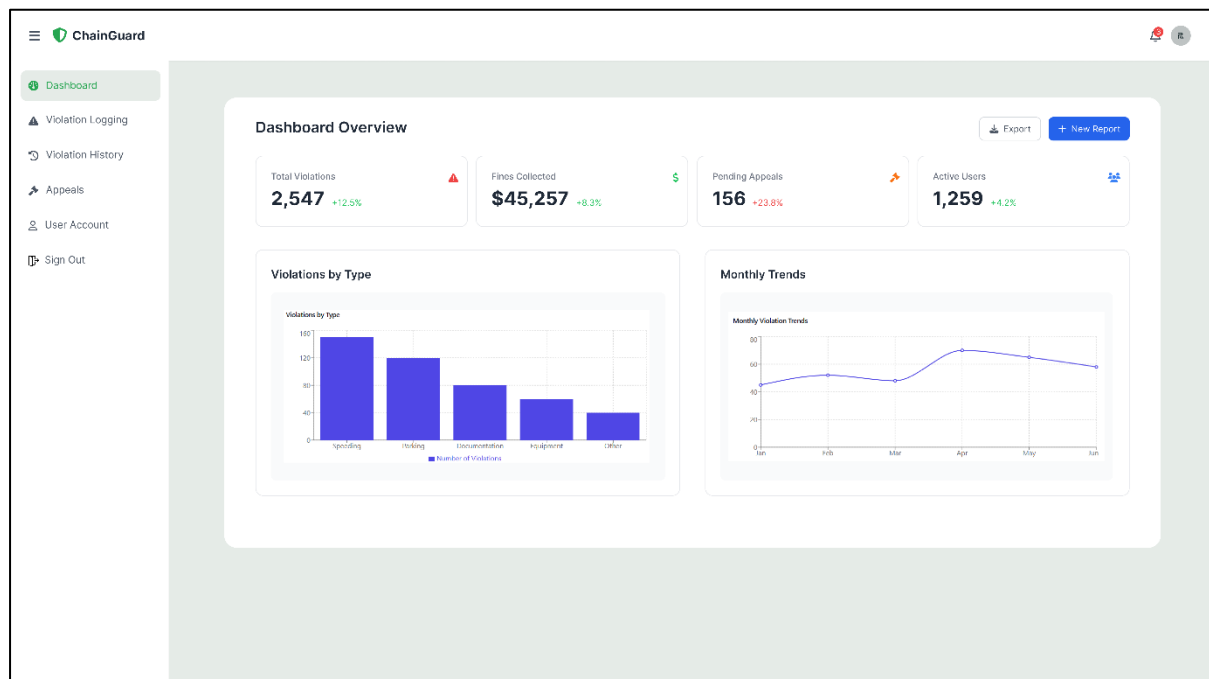
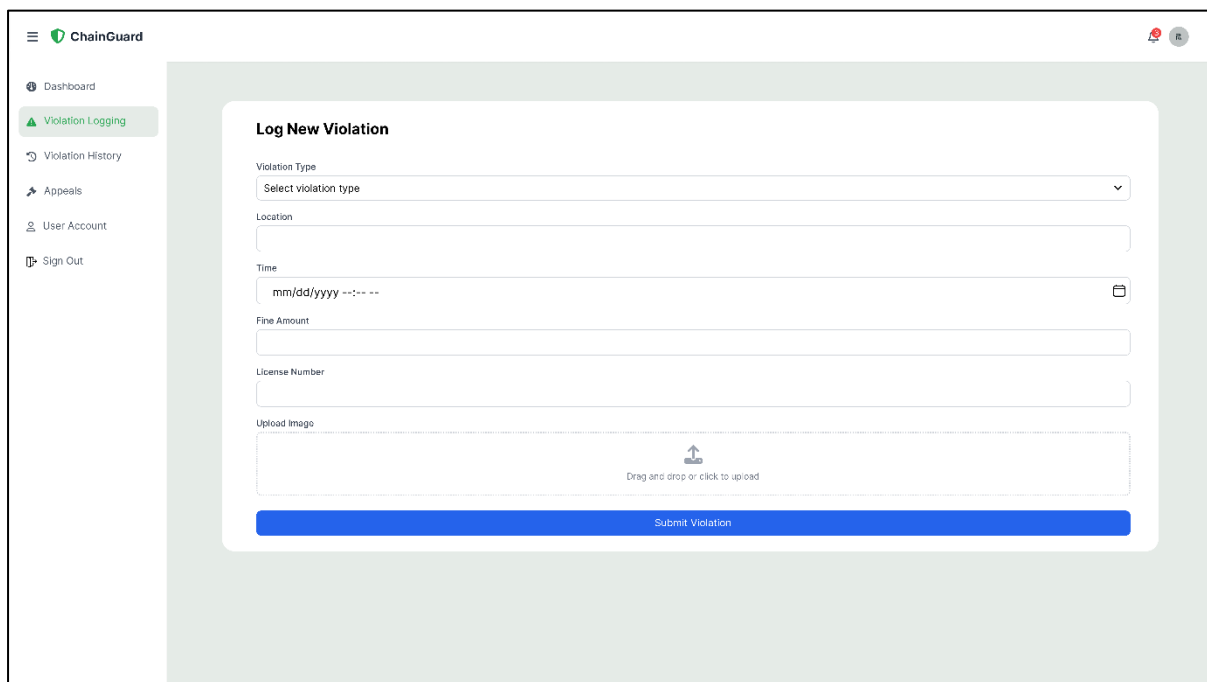


Figure 3.31. Admin Dashboard Page

Figure 3.32 illustrates the manual violation logging interface, which enables administrators to log violations directly into the system. The interface allows administrators to input key details such as the violation type, location, date and time, fine amount, license plate number, and image as proof. Upon completion, the violation record can be uploaded to the system and securely stored in the blockchain ledger, ensuring the data remains immutable and tamper-proof.



The screenshot displays the ChainGuard web application interface. On the left is a sidebar menu with the following items: 'Dashboard', 'Violation Logging' (highlighted with a green background), 'Violation History', 'Appeals', 'User Account', and 'Sign Out'. The main content area is titled 'Log New Violation' and contains a form with the following fields: 'Violation Type' (a dropdown menu with 'Select violation type' as the placeholder), 'Location' (a text input field), 'Time' (a date and time picker showing 'mm/dd/yyyy --:-- --'), 'Fine Amount' (a text input field), 'License Number' (a text input field), and 'Upload Image' (a large dashed border area with a central upload icon and the text 'Drag and drop or click to upload'). At the bottom of the form is a prominent blue button labeled 'Submit Violation'.

Figure 3.32. Manual Violation Logging Page

Figure 3.33 shows the violation record page, which displays all logged violations. Each record includes key details such as the date, violation type, location, fine amount, and current status. This page enables administrators to view and manage violation records within the system, supporting traffic enforcement activities with streamlined access to critical data.

Violation Records					View All
Date	Violation	Location	Fine	Status	
Jan 15, 2025	Speeding	CTFI, UNIMAS	RM 150	Pending	
Jan 10, 2025	Red Light	FCSIT, UNIMAS	RM 150	Paid	
Jan 10, 2025	Red Light	FCSIT, UNIMAS	RM 150	Paid	
Jan 10, 2025	No seat belt	FCSIT, UNIMAS	RM 300	Paid	
Showing 1 to 2 of 10 results					Previous 1 2 Next

Figure 3.33. Violation Record Page

Figure 3.34 demonstrates the Appeals Management page, where the administrator can review and process appeals submitted by drivers. The interface displays a table containing key details, such as the driver's name, email, license number, and associated violation ID. Administrators are provided with clear options to approve or reject each appeal. The page includes a search function for filtering appeals and pagination for easy browsing of multiple records.

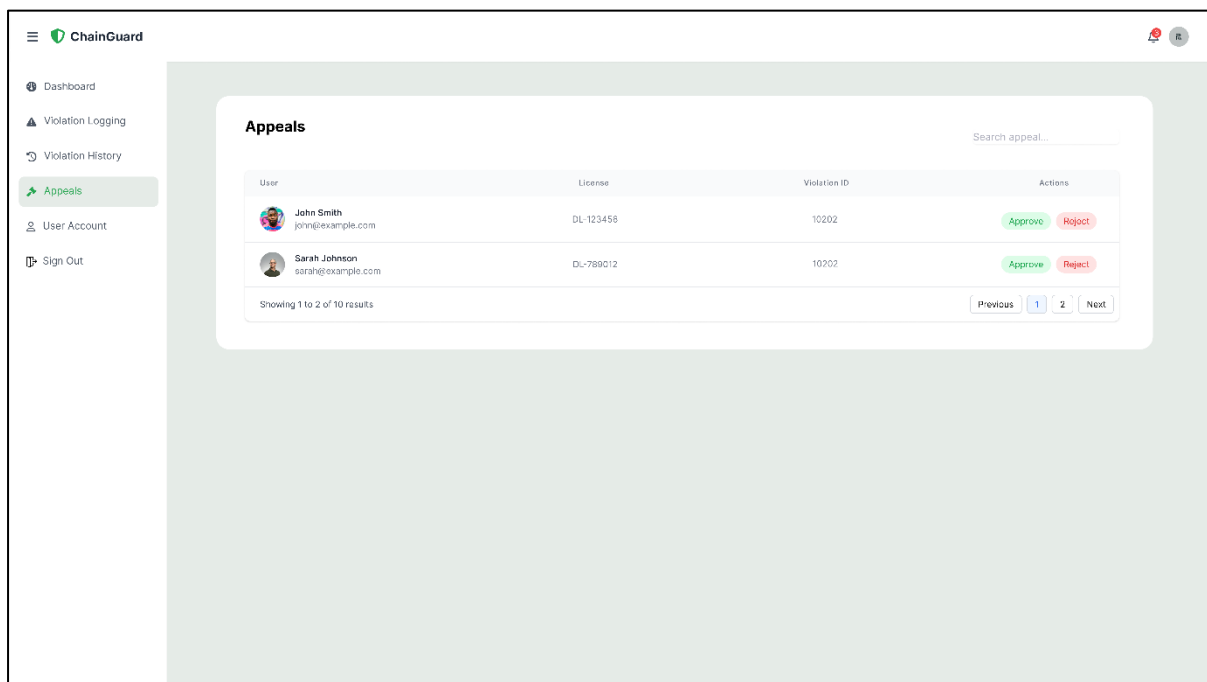


Figure 3.34. Appeal Management Page

Figure 3.35 illustrates the User Accounts Management page within the administrative dashboard. This interface provides the administrator with the ability to view and manage all registered driver accounts. Each user is listed along with their name, email address, license number, and account status (e.g., Active or Inactive). The table also includes action buttons for editing, viewing details, and deleting accounts. A search bar at the top enables efficient filtering of user accounts, while pagination at the bottom facilitates navigation through a large user database.

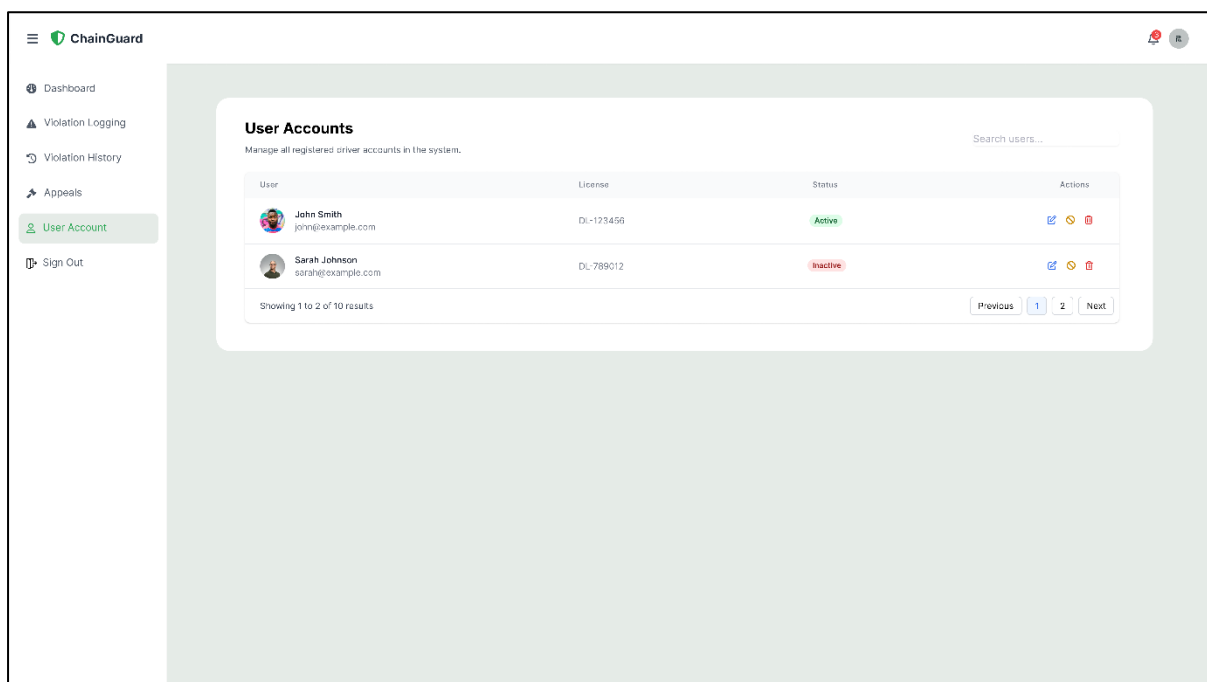


Figure 3.35. User Account Management Page

3.5. Development

After the design phase, all the requirements are identified through an in-depth analysis of the proposed system's architecture, data flows, relationships, and prototyping. The next critical phase is developing the proposed system, which uses software and hardware identified in the design phase. This phase is divided into two parts, where the first part is the frontend design based on the wireframe created. The second part, backend integration, where all the crucial parts such as the web server, databases, and blockchain network will be integrated with the front end. At the end of this phase, the codebase will be sanitized and cleaned up to ensure maintainability and readability.

3.6. Testing

Following the principles of Scrum methodology, the testing phase was conducted iteratively with testing performed at the end of each sprint, allowing early identification and resolution of issues. It ensures the system's functionality and reliability throughout the development process. Various types of testing were performed during this phase. Unit testing was conducted to validate the functionality of individual modules, while integration testing ensured smooth interaction between the frontend, backend, and blockchain components. Additionally, user acceptance testing involved end-users, including administrators and drivers, to validate the system's usability and performance in real-world scenarios. The iterative nature of testing ensured that each feature met the specified requirements and operated seamlessly within the system.

3.7. Deployment

The deployment phase involved releasing the completed system into a live environment for real-world usage. This process began with setting up the production infrastructure, hosting the web application on a secure server, and configuring the blockchain network for immutable data storage. A structured deployment plan was followed, which included migrating data from the development environment to production and performing final checks to ensure all components were fully functional and secure. Post-deployment monitoring was carried out to identify and address any issues to ensure the stability and performance of the system. This phase also will focus on the delivery of essential technical documentation of the application.

3.8. Review

The review phase marked the final step in the project lifecycle and served as a retrospective in line with Scrum practices. Sprint reviews were conducted at the end of each sprint to evaluate deliverables, gather feedback, and identify areas for improvement. A final system review was carried out to assess the overall solution against the initial objectives and requirements which ensure the system's completeness and quality. This phase also highlighted the system's strengths, identified areas for future enhancements, and documented lessons learned. The insights gained during the review process provide a strong foundation for the system's scalability and sustainability which ensures its relevance and effectiveness in addressing traffic enforcement challenges.

3.9. Summary

In conclusion, Chapter 3 focuses on the chosen methodology, Scrum, in previous chapter and use it as the foundation for the proposed system, Blockchain-Based Traffic Violation Logging and Penalty Management System. The chapter began with the requirements phase, where a survey was conducted to gather feedback on the proposed system from potential users. This phase also included a detailed analysis of existing systems and the identification of features to be incorporated into the proposed system. Next, the design phase laid the foundation for the system's structure and functionality, which involves creating scenario illustrations, designing the web application and software architecture, and developing key visual and structural components, including flowcharts, context diagrams, data flow diagrams (DFDs), entity-relationship diagrams (ERDs), data dictionaries, and wireframes. Following the design phase, the development phase brought the system to life through the implementation of both frontend and backend components. These components were iteratively tested during the testing phase to ensure reliability, usability, and seamless integration. Finally, the review phase reflected on the overall development process, evaluating the system's performance against the initial objectives and identifying areas for future improvement.

CHAPTER 4: IMPLEMENTATION

4.1. Introduction

This chapter focuses on the development and implementation of the proposed system based on the requirement analysis and design in previous chapter, Chapter 3. With the utilization of the proposed hardware and software mentioned in Chapter 3, the development follows strongly to the Scrum methodology, which split the development and implementation phases into sprints. This iterative approach allows the system to be built and tuned to fulfil the requirements of the proposal, and also ensure the development is completed in timely manner. In addition, this chapter also underline the front-end and back-end development processes

4.2. Installation and Configuration of ChainGuard

Setting up the ChainGuard system, a Blockchain-Based Traffic Violation Logging and Penalty Management System, necessitates the installation and configuration of several specific software packages, libraries, and dependencies. These serve as prerequisites for the development and deployment stages, which will be elaborated upon in subsequent sections. The system was developed and tested on Debian-based Linux distributions, namely Ubuntu 24.04.2 LTS and Raspberry Pi OS (Bookworm).

ChainGuard's architecture consists of distinct layers: the front-end web application is built using standard web technologies including Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS), with server-side logic and web functionalities handled by Hypertext Preprocessor (PHP) running on an Apache Web Server. For efficient off-chain data storage, MongoDB is utilized as the database. An essential middleware component is the API layer, implemented as a Node.js Express server, which leverages the Hyperledger Fabric Software Development Kit (SDK) to securely interact with

the blockchain network. The core of the system resides in the Hyperledger Fabric blockchain network, configured for testing as a two-organization network with a single peer per organization to simulate diverse user interactions securely and efficiently. Custom chaincode (smart contracts), developed in Go, is deployed onto this network to automate the logic for creating and managing traffic violation records on the immutable ledger, with CouchDB serving as the state database for each peer to store and query chaincode data effectively. Finally, Docker is employed to containerize the backend components, specifically the API layer and the Hyperledger Fabric network elements, ensuring consistency and simplifying deployment across different environments. The following sub-chapters will provide detailed guidance on the installation and setup procedures for each of these components.

4.2.1. Hyperledger Fabric Network

The foundation of the ChainGuard system lies in the Hyperledger Fabric blockchain network. Setting up this network involves several key steps, beginning with the installation of essential prerequisite software on the chosen Linux environment (Ubuntu 24.04.2 LTS or Raspberry Pi OS Bookworm).

4.2.1.1. Prerequisites Installation

The implementation of the Hyperledger Fabric blockchain network begins with establishing a robust development environment, which required the installation of several essential tools and dependencies that would support the blockchain infrastructure on the chosen Debian-based Linux environment. Firstly, Python and Pip must be installed to provide scripting capabilities and dependency management for various components of the system, such as Docker and Docker Compose. The following command shows the installation of Pip, with Python pre-installed on the operating system.

```
sudo apt install python3-pip
```

Next, Git is installed for version control, curl for data transfer operations, and Docker along with Docker Compose for containerization various components of the Hyperledger Fabric blockchain network. Docker plays a crucial role in Hyperledger Fabric deployment as it allows for consistent runtime environments across different systems, ensuring that the blockchain network operates identically regardless of the underlying infrastructure. The attached command shows the installation of Git, curl, Docker, and Docker Compose.

```
sudo apt-get install git curl docker-compose -y
```

After the Docker and Docker Compose installation, it is essential to ensure that they are enable and running in the background. It is also recommended to run Docker and Docker Compose with administrator privilege, which can ease the overall container management process. Once the administrator privilege is set, ensure that the system is rebooted to take effect of the changes.

```
sudo systemctl start docker  
sudo systemctl enable docker  
sudo systemctl status docker  
sudo usermod -a -G docker rj  
sudo reboot  
docker ps
```

The Go Programming language must be installed, as it serves as the foundation for the Hyperledger Fabric chaincode development. The chaincodes, also known as smart contracts in other blockchain framework, are primarily written in Go language, which makes it an essential component of the development stack. To install the Go language, it is critical to ensure that the correct version of the Go language is installed. Start by navigating to the official website of Go language download page (<https://go.dev/dl/>) and locate the stable version with the correct

system architecture, and operating system. For this development, the archive ARM64 version is chosen as for the installation, as shown in Figure 4.1. The link is copied by right clicking the package text, then navigate to the Downloads directory and use Wget to download the package to the system. Once the download is completed, the package is extracted and the Go binary directory must be added to the system's PATH to ensure that Go commands could be executed from any location in the file system, as shown in the commands below.

```
# Install Go Language

cd Downloads/

wget https://go.dev/dl/go1.24.2.linux-arm64.tar.gz

sudo tar -C /usr/local/ -xzf go1.24.2.linux-arm64.tar.gz

nano $HOME/.profile

export PATH=$PATH:/usr/local/go/bin

source $HOME/.profile

echo $PATH
```

go1.24.2						
File name	Kind	OS	Arch	Size	SHA256 Checksum	
go1.24.2.src.tar.gz	Source			29MB	9dc77ffadc16d837a1bf32d99c624cb4df0647cee7b119ed9e7b1bcc85f2e00	
go1.24.2.darwin-amd64.tar.gz	Archive	macOS	x86-64	76MB	238d9c865d89ff6af229d2e3b8b5e85e688318d69f4086fb85a96e41c216ea83	
go1.24.2.darwin-amd64.pkg	Installer	macOS	x86-64	77MB	535ed9ff283fee39575a7fb9b6d8b1901b6dc648d06dc71fd7d3faeefda78030	
go1.24.2.darwin-arm64.tar.gz	Archive	macOS	ARM64	73MB	b70f8b3c5b4ccb0ad4ffa5ee91cd38075df20fdb953a1daedd47f50fbcff47a	
go1.24.2.darwin-arm64.pkg	Installer	macOS	ARM64	73MB	4732f607a47ce4d898c0af01ff68f07e0820a6b50603aef5d5c777d1102505e2	
go1.24.2.linux-386.tar.gz	Archive	Linux	x86	73MB	4c382776d52313266f3026236297a224a6688751256a2dff3af524d8d6f6c0ba	
go1.24.2.linux-amd64.tar.gz	Archive	Linux	x86-64	75MB	68097bd680839cb9d464a0edce4f7c333975e27a90246890e9f1078c7e702ad	
go1.24.2.linux-arm64.tar.gz	Archive	Linux	ARM64	71MB	756274ea4b68fa5535eb9fe2559889287d725a8da63c6aee4d5f23778c229f4b	
go1.24.2.linux-armv6l.tar.gz	Archive	Linux	ARMv6	73MB	438d5d3d7dc239b58d893a715672eab670b9730b1fd1c8fc858a46722a598a	
go1.24.2.windows-386.zip	Archive	Windows	x86	82MB	13d86cb818bba331da75fcd18246ab31a1067b44fb4a243b6df93087eda7f37	
go1.24.2.windows-386.msi	Installer	Windows	x86	67MB	8a702d9f7104a15bd935f4191c58c24c0b6389e060b9d5661b93915114a2bef0	
go1.24.2.windows-amd64.zip	Archive	Windows	x86-64	83MB	29c553aabee0743e2ffa3e9fa0cda00ef3b3cc4ff0bc92007f31f80fd69892e1	
go1.24.2.windows-amd64.msi	Installer	Windows	x86-64	69MB	acefb191e72fea0bd0b1a3f5f8f6f5ab18b42b3bcbec0c7183f189f25953aff275	

Figure 4.1. Go Language Website

go1.24.2.linux-amd64.tar.gz	Archive	Linux	x86-64	75MB	68097bd688839cbc9d464a0edce4f7c333975e27a90246890e9f1078c7e702ad
go1.24.2.linux-arm64.tar.gz	Open link in new tab		ARM64	71MB	756274ea4b68fa5535eb9fe2559889287d725a8da63c6aee4d5f23778c229f4b
go1.24.2.linux-armv6.tar.gz	Open link in new window		ARMv6	73MB	438d5d3d7dcb239b58d893a715672eabe670b9730b1fd1c8fc858a46722a598a
go1.24.2.linux-armv7.tar.gz	Open link in incognito window				
go1.24.2.windows-amd64.zip	Open link as	>	s x86	82MB	13d86cb818bba331da75fcd18246ab31a1067b44fb4a243b6df93097eda7f37
go1.24.2.windows-arm64.zip	Save link as...		s x86	67MB	8a702d9f7104a15bd935f4191c58c24c8b6389e066b9d5661b93915114a2bef0
go1.24.2.windows-armv6.zip	Copy link address		s x86-64	83MB	29c553aabee0743e2ffa3e9fa8cda00ef3b3cc4ff0bc92007f31f80fd69892e1
go1.24.2.windows-armv7.zip	AdGuard AdBlocker	>	is x86-64	69MB	acefb191e72fea0bdb1a3f5f8f6f5ab18b42b3bbce0c7183f189f25953aff275
Other Ports ^	Inspect				

Figure 4.2. Copy link address

Additionally, JQ, a lightweight command-line JSON processor, is installed to facilitate parsing and manipulation of JSON data, which is common in blockchain configurations and transactions.

```
# Install JQ
sudo apt-get install jq
```

To ensure all required components were correctly installed, a validation step was performed by checking the version of each tool. This verification process confirmed that the development environment was properly configured and ready for the Hyperledger Fabric network implementation. The expected output is shown in Figure 4.3.

```
# Validation
python --version
pip --version
docker --version
docker-compose --version
go version
curl --version
git --version
jq --version
```

```

rj@rpi:~$ python --version
Python 3.11.2
rj@rpi:~$ pip --version
pip 23.0.1 from /usr/lib/python3/dist-packages/pip (python 3.11)
rj@rpi:~$ docker --version
Docker version 20.10.24+dfsg1, build 297e128
rj@rpi:~$ docker-compose --version
docker-compose version 1.29.2, build unknown
rj@rpi:~$ go version
go version go1.24.2 linux/arm64
rj@rpi:~$ curl --version
curl 7.88.1 (aarch64-unknown-linux-gnu) libcurl/7.88.1 OpenSSL/3.0.15 zlib/1.2.13 brotli/1.0.9 zstd/1.5.4 libidn2/2.3.3 l
ibpsl/0.21.2 (+libidn2/2.3.3) libssh2/1.10.0 nghttp2/1.52.0 librtmp/2.3 OpenLDAP/2.5.13
Release-Date: 2023-02-20, security patched: 7.88.1-10+deb12u12
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp smb smbs
smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNEG
O SSL threadsafe TLS-SRP UnixSockets zstd
rj@rpi:~$ git --version
git version 2.39.5
rj@rpi:~$ jq --version
jq-1.6

```

Figure 4.3. Prerequisites Installation Validation Output.

4.2.1.2. Hyperledger Fabric Blockchain Network Installation

With the development environment in place, the next phase involved installing the Hyperledger Fabric framework. A dedicated directory structure is created to organize the fabric-related files and ensure a clean workspace for the upcoming development. The Hyperledger Fabric installation script is downloaded from the official repository and made executable through command line interface. This script automates the process of downloading the framework's binaries, Docker images, and sample code that will serve as the foundation for the proposed system's blockchain network.

```

# Create dedicated directory for the network

mkdir -p $HOME/go/src/github/<github_name>

cd $HOME/go/src/github/<github_name>

# Pull the install-fabric.sh script

curl -sSLO

https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-
fabric.sh && chmod +x install-fabric.sh

# Start the installation

./install-fabric.sh

```

To simplify Docker container management throughout the project lifecycle, Portainer is installed. Portainer provides a user-friendly web interface for monitoring and managing Docker containers, which is particularly useful for overseeing the multiple containers that make up the proposed systems. Before the deployment, ensure that Portainer's external port mapping is changed to port 9000 due to Port 9443 is reserved for the blockchain network. Netstat command can be used to check the occupied ports in the system. After Portainer is deployed, it can be access through a web browser at https://<device_IP>:9000, where admin credentials are established for secure access to the Docker management interface as shown in Figure 4.4 and 4.5.

```
# Install Portainer for easy docker management

docker volume create portainer_data

netstat -tulnp

docker run -d -p 8000:8000 -p 9000:9443 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data
portainer/portainer-ce:lts
```

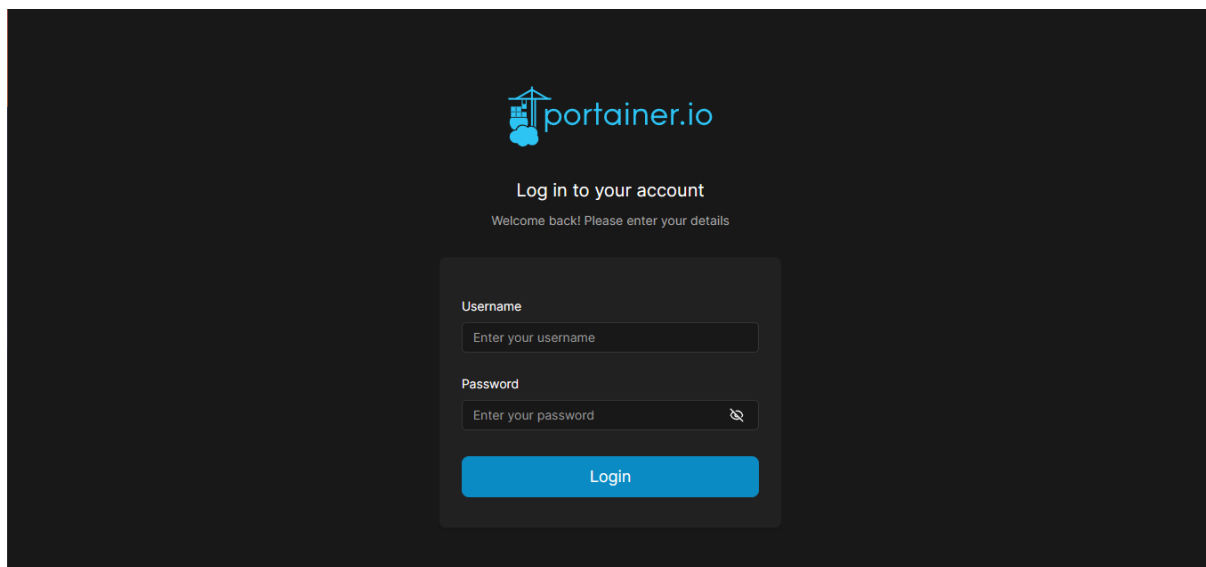


Figure 4.4. Login Interface of Portainer

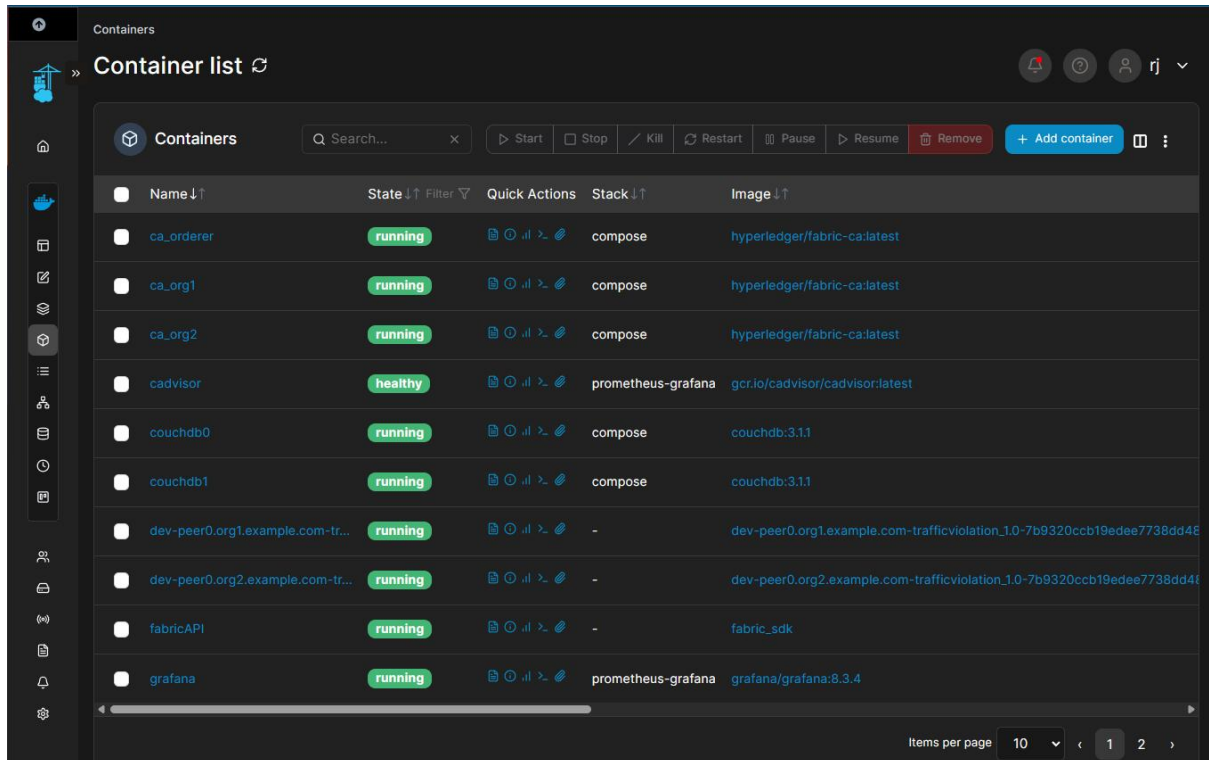


Figure 4.5. List of Containers Running.

Furthermore, the proposed system repository which contains all the web application, API layer, and custom configurations for the blockchain network is cloned from GitHub. The cloning process involves copying the repository link from GitHub as shown in Figure 4.6, and navigate to the home directory for cloning the repository using git clone.

```
# Clone the project repository
cd
git clone https://github.com/Kae-RZ/FYP_81049_ChainGuard.git
```

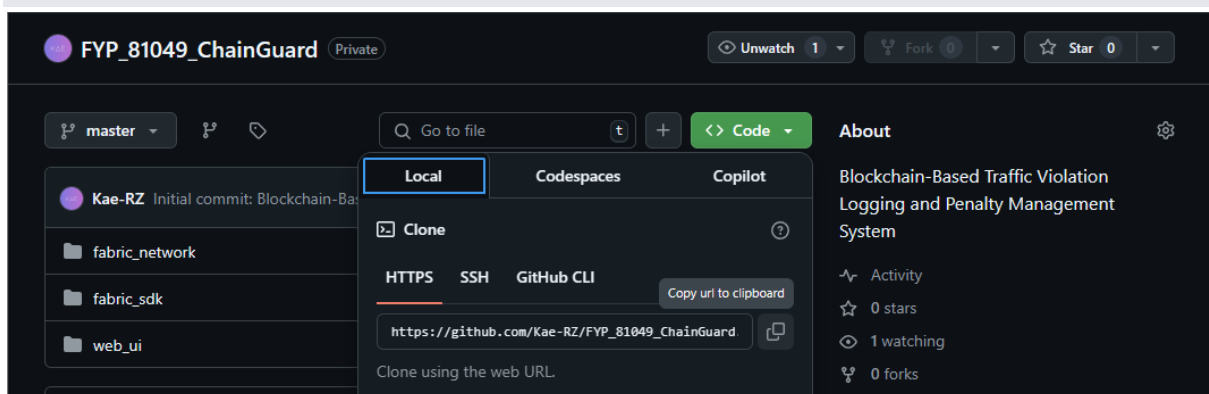


Figure 4.6. GitHub Repository Page.

The standard Hyperledger Fabric test network configuration need to be configured to meet the specific requirements of the project. This process involved modifying the Docker Compose file that defines the network's container services, peer nodes, and ordering services. The changes are made on the inclusion of CouchDB as state database due to the default LevelDB does not support rich queries on state data. To ease the changes on configuration file, the updated configuration file is attached in the cloned repository under the fabric_network directory. Hence, before update the file, a backup of the original configuration file was created before replacing it with the custom version from the project repository. This ensured that the changes could be reverted if necessary while preserving the integrity of the original Fabric samples.

```
# Change the docker compose file for the network  
cd $HOME/go/src/github/Kae/fabric-samples/test-network/compose  
  
# Make a backup  
cp compose-test-net.yaml compose-test-net-bk.yaml  
  
# Copy the compose-test-net.yaml from the cloned repository  
cp $HOME/FYP_81049_ChainGuard/fabric_network/compose-test-net.yaml  
$HOME/go/src/github/Kae/fabric-samples/test-network/compose
```

The core business logic of the blockchain web application is implemented as a smart contract, known as chaincode in Hyperledger Fabric. A dedicated directory structure was created for the traffic violation chaincode, and the implementation file was copied from the cloned project repository. Go modules are then initialized for the chaincode to manage dependencies properly, which ensure that the smart contract could be compiled and deployed successfully.

```
# Chaincode (Smart Contract) Configuration  
cd $HOME/go/src/github/Kae/fabric-samples/
```

```

mkdir chaincode && cd chaincode

mkdir trafficviolation && cd trafficviolation

cp $HOME/FYP_81049_ChainGuard/fabric_network/trafficviolation.go
$HOME/go/src/github/Kae/fabric-samples/chaincode/trafficviolation/

# Initialize the Go modules

go mod init trafficviolation

go mod tidy

```

In addition, to optimize query performance in the CouchDB state database, a custom index was created for the driverID field. CouchDB indexes significantly improve the performance of queries that filter or sort data based on specific attributes, which is crucial for applications requiring efficient data retrieval.

```

# Create the META-INF directory for CouchDB indexes

mkdir -p $HOME/go/src/github/Kae/fabric-
samples/chaincode/trafficviolation/META-INF/statedb/couchdb/indexes

cd /META-INF/statedb/couchdb/indexes

nano indexDriverID.json

# Index Definition

{
  "index": {
    "fields": ["driverID"]
  },
  "name": "driverID-index",
  "type": "json"
}

```

With all the required components, the Hyperledger Fabric network can be initialized. The network script was executed with parameters to bring up the containers, create a channel for transactions, and enable the Certificate Authority (CA) for identity management through

Membership Service Provider (MSP). The command starts the Docker containers for the peer nodes, ordering services, and certificate authorities, established the necessary network connections, and created a channel for secure communication between the network participants. If there is a need to down or reset the network, the network down command can be used to stop and remove all associated containers.

```
# Start the network

cd $HOME/go/src/github/Kae/fabric-samples/test-network/

./network.sh up createChannel -ca

# If want bring down the network

cd $HOME/go/src/github/Kae/fabric-samples/test-network/

./network.sh down
```

4.2.1.3. Network Monitoring – Prometheus and Grafana

To ensure operational visibility and performance tracking of the blockchain network, Prometheus, an open-source monitoring system designed for time-series data collection and alerting, and Grafana, an open-source platform widely used for visualizing metrics and creating interactive analysis dashboards, monitoring tools are deployed. The Docker Compose configuration for these tools is modified to accommodate project-specific requirements, particularly to avoid port conflicts with other components.

```
# Start Prometheus and Grafana

cd $HOME/go/src/github/Kae/fabric-samples/test-network/prometheus-grafana

nano docker-compose.yaml
```

Based on the docker-compose.yaml, there are two main changes: cAdvisor's volume binding and Grafana port binding. Container Advisor (cAdvisor) is an open-source tool developed by

Google for discovering, collecting, processing and exporting real-time information about the running containers on a host machine. The change on volume binding for cAdvisor is to ensure proper system monitoring capabilities, while Grafana's port binding is to avoid conflicts with the Node.js API that would be used to interact with the Hyperledger Fabric SDK. The changes are listed in below. After saving the configuration changes, the monitoring stack was launched through command line interface. Grafana can be accessed through web browser at http://<device_IP>:3002 and Prometheus can be accessed at http://<device_IP>:9090 as shown in Figure 4.7 and Figure 4.8.

```
# Search for and edit the following (Configuration for cAdvisor - System Monitoring):

cadvisor:

  image: gcr.io/cadvisor/cadvisor:latest

  privileged: true

  container_name: cadvisor

  volumes:

    - /:/rootfs:ro

    - /var/run:/var/run:ro

    - /sys:/sys:ro

    - /var/lib/docker:/var/lib/docker:ro

    - /sys/fs/cgroup:/sys/fs/cgroup:ro

  ports:

    - 8080:8080

  restart: always

# Search for and edit the following (Grafana port need to change as port 3000 is reserved for Node JS API [fabric SDK])

grafana:
```

```
image: grafana/grafana:8.3.4

container_name: grafana

user: "104"

depends_on:

  - prometheus

ports:

  - 3002:3000

volumes:

  - grafana_storage:/var/lib/grafana
  - ./grafana/provisioning/:/etc/grafana/provisioning/

env_file:

  - ./grafana/config.monitoring

restart: always

# Start the Prometheus and Grafana containers

docker-compose up -d
```

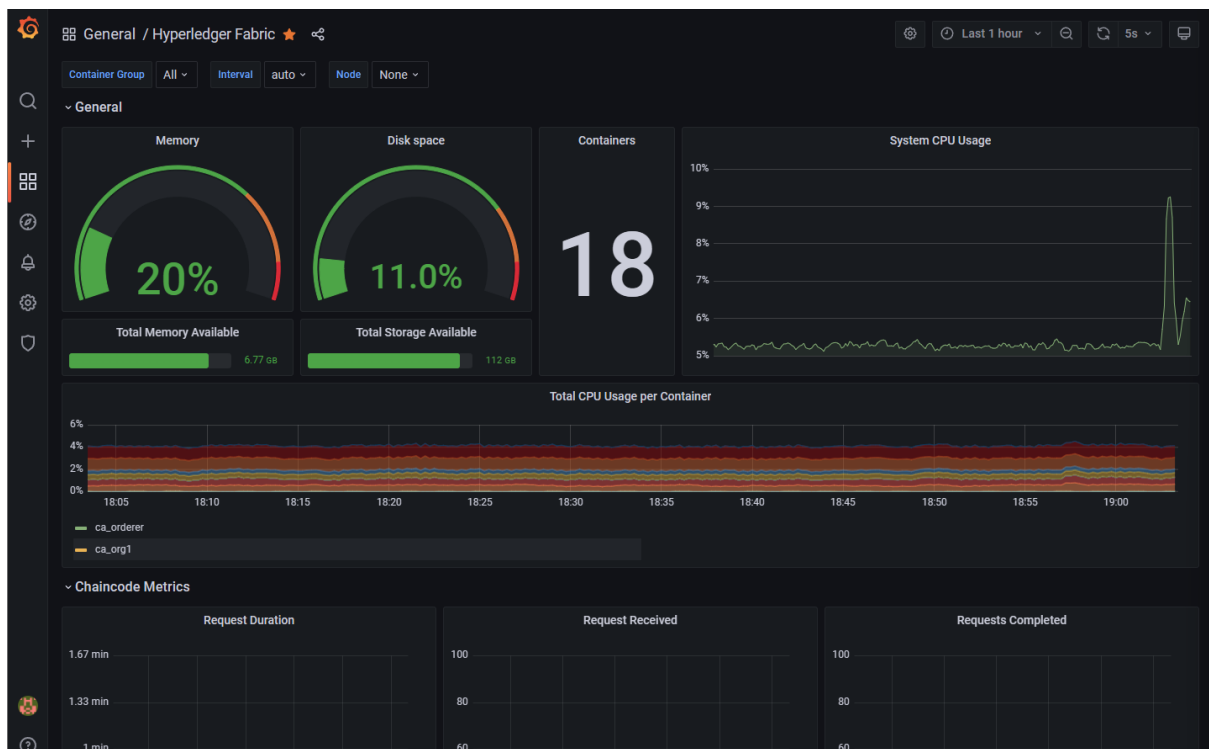


Figure 4.7. Grafana Dashboard.



Figure 4.8. Prometheus Dashboard.

4.2.1.4. Chaincode Deployment

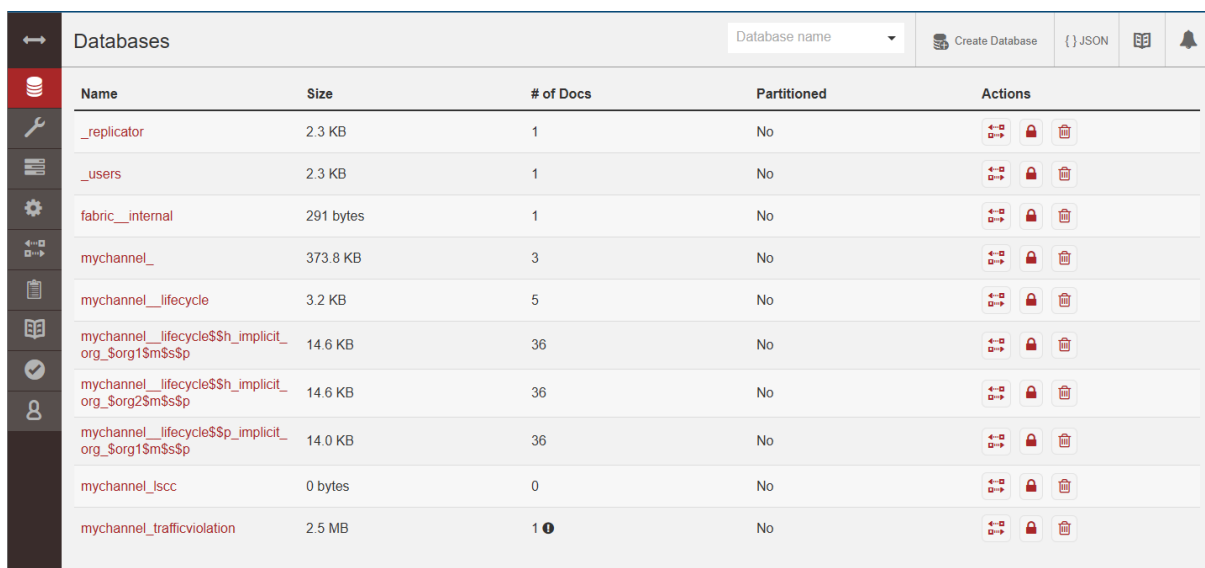
The final step in the network setup involved deploying the traffic violation chaincode onto the application channel. This is again accomplished using the `network.sh` script with the `deployCC` command, specifying the chaincode name (`-ccn trafficviolation`), the source code path (`-ccp ../chaincode/trafficviolation/`), and the language (`-ccl go`). This command packages, installs, approves, and commits the chaincode definition on the channel, making it ready to process transactions. Upon successful deployment, the CouchDB state database can be access via a web browser at `http://<device_IP>:5984/_utils/#`, where the default credentials can be obtained in the `compose-test-net.yaml` file. This interface allows for direct observation of the state database contents, providing valuable insights during development and deployment as shown in Figure 4.8.

```
# Deploy the chaincode

cd $HOME/go/src/github/Kae/fabric-samples/test-network/

./network.sh deployCC -ccn trafficviolation -

ccp ../chaincode/trafficviolation/ -ccl go
```



Databases					Database name	Create Database	{ } JSON		
	Name	Size	# of Docs	Partitioned	Actions				
	_replicator	2.3 KB	1	No					
	_users	2.3 KB	1	No					
	fabric__internal	291 bytes	1	No					
	mychannel_	373.8 KB	3	No					
	mychannel__lifecycle	3.2 KB	5	No					
	mychannel__lifecycle\$\$h_implicit_ org_\$org1\$m\$\$p	14.6 KB	36	No					
	mychannel__lifecycle\$\$h_implicit_ org_\$org2\$m\$\$p	14.6 KB	36	No					
	mychannel__lifecycle\$\$p_implicit_ org_\$org1\$m\$\$p	14.0 KB	36	No					
	mychannel_iscv	0 bytes	0	No					
	mychannel_trafficviolation	2.5 MB	1	No					

Figure 4.9. CouchDB Interface.

4.2.1.4. Hyperledger Explorer

After successfully implementing the Hyperledger Fabric network with its core components and smart contract, the next phase involved deploying Hyperledger Explorer. Hyperledger Explorer is a vital tool that provides a web-based interface for visualizing and monitoring the blockchain network, offering intuitive insights into blocks, transactions, and network participants. This visualization layer enhances the project's usability and provides stakeholders with a comprehensive view of the blockchain's operation.

The implementation begins by creating a dedicated directory for the Hyperledger Explorer components within the project structure. Several configuration files are downloaded from the official Hyperledger Explorer repository to establish the foundation for the Explorer setup. These files define how Explorer connects to and interacts with the existing Fabric network. The `config.json` file specifies the network profiles that Explorer will connect to, while the `test-network.json` connection profile contains specific details about the blockchain network's endpoints, organizations, and channel configurations. The `docker-compose.yaml` file defines the container services required for running Explorer, including the Explorer web application and its PostgreSQL database.

```
# Deploy Hyperledger Explorer

cd $HOME/go/src/github/Kae
mkdir explorer && cd explorer

# Download the required files

wget https://raw.githubusercontent.com/hyperledger/blockchain-
explorer/main/examples/net1/config.json
```

```
wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/examples/net1/connection-profile/test-network.json -P connection-profile
wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/docker-compose.yaml
```

Furthermore, to enable Explorer to authenticate with the Fabric network, the cryptographic materials, such as certificates and keys, from the test network are copied to the Explorer directory. The ownership of these files is adjusted to ensure that the current user has appropriate permissions to access and use these cryptographic materials, which is essential for secure interaction with the blockchain network.

```
# Set permissions for the copied files
sudo cp -r ../fabric-samples/test-network/organizations .
sudo chown -R $(id -u):$(id -g) organizations
sudo chown -R $(id -u):$(id -g) connection-profile/
```

An environment configuration file is created to specify the operating parameters for Explorer, including the port number and the paths to various configuration files. This is to avoid modifying the downloaded files, and possible creating unwanted errors during deployment.

```
# Create an environment configuration file
nano .env
```

The following environment variables are defined to customize the Explorer deployment. These settings ensure that the Explorer runs on port 8081, uses the specified configuration file, and can access the necessary cryptographic materials and connection profiles.

```
PORT=8081
EXPLORER_CONFIG_FILE_PATH=./config.json
EXPLORER_PROFILE_DIR_PATH=./connection-profilego
FABRIC_CRYPTO_PATH=./organizations
```

To automate the setup process, a custom script is copied from the cloned project repository and made executable. This script automates several configuration tasks, including cryptographic material management, private key handling, configuration updates, Docker operations and error handling.

```
# Copy the script from the cloned repository and place in the same folder for  
the setting up  
cp $HOME/FYP_81049_ChainGuard/setup_explorer.sh  
$HOME/go/src/github/Kae/explorer  
chmod +x setup_explorer.sh  
./setup_explorer.sh
```

Upon successful execution of the setup script, Hyperledger Explorer is fully deployed and accessible through a web browser at <http://localhost:8081>. The interface provides a comprehensive dashboard displaying key metrics such as block height, transaction count, and network status, along with detailed views of individual blocks and transactions.

4.2.2. Node.js Hyperledger Fabric Software Development Kit (SDK)

After setting up the Hyperledger Fabric blockchain network, the next critical component involves the creation of an application programming interface (API) that will interact with the blockchain through the Hyperledger Fabric SDK. The cruciality of this interface is to serve as the communication layer between client application, which is the traffic violation web application and the Hyperledger Fabric blockchain network, enabling operations such as querying the ledger and submitting transactions.

4.2.2.1. Setting Up Node.js Express Server (API)

The installation begins with the installation of Node.js and NPM (Node Package Manager), which form the foundation of the API server. The API server used Express.js, a

Node.js web application framework due to the simplicity of the server-side application development and deployment. The following commands are used to install Node.js and NPM, and then verify their installation by checking their version. Node.js is chosen for this implementation because of it offers a lightweight and efficient runtime environment, and Hyperledger Fabric SDK for Node.js offers comprehensive blockchain interaction capabilities.

```
# Set up the Node.js Express Server (API)
```

```
sudo apt install nodejs npm -y
```

```
node -v
```

```
npm -v
```

In addition, a dedicated directory is created for the Fabric SDK implementation, and all necessary files are copied from the cloned project repository.

```
cd $HOME
```

```
mkdir fabric_sdk && cd fabric_sdk
```

```
cp $HOME/FYP_81049_ChainGuard/fabric_sdk/* $HOME/fabric_sdk
```

Withing the same directory, two subdirectories are created to store network connection information and digital identities, as known as the wallet identities, which is needed for the blockchain interaction. Once the directories are created, the connection profiles for both organizations in the blockchain network are then copied from the test network configuration directory to the newly created network directory. This step must be executed with the Hyperledger Fabric blockchain is running. These connection profiles contain essential information such as endpoint URLs, TLS certificates, and organization details required for the SDK to establish secure connections with the blockchain network.

```
# Create the wallet and network directories
```

```
mkdir network/ wallet/
```

```
# Copy the connection profiles from the test-network to the network directory
cp $HOME/go/src/github/Kae/fabric-samples/test-
network/organizations/peerOrganizations/org1.example.com/connection-org1.json
network/connection-org1.json
cp $HOME/go/src/github/Kae/fabric-samples/test-
network/organizations/peerOrganizations/org2.example.com/connection-org2.json
network/connection-org2.json
```

A new Node.js project is initialized, and the necessary dependencies are installed through NPM. These commands create a package.json file with default values and installs Express server, a minimal web framework for creating the API endpoints, fabric-network, the core Hyperledger Fabric SDK for connecting to the blockchain, fabric-ca-client, a controller for interaction with the Certificate Authority (CA) for Fabric blockchain network, and uuid, an unique identifier generator for transactions. A connection test is performed to verify that the SDK can successfully communicate with the blockchain network. The expected output is shown Figure 4.10.

```
npm init -y
npm install express fabric-network fabric-ca-client uuid
# Test the API connection
node testConnection.js
```

```
rj@rpi:~/fabric_sdk $ node testConnection.js
Evaluating GetClientID transaction...
GetClientID result: eDUwOT06Q049YWRtaW4sT1U9Y2xpZW50OjpDTj1jYS5vcmcxLmV4YW1w
bGUuY29tLE89b3JnMS5leGFtcGxlLmNvbSxMPUR1cmhhbSxTVD10b3J0aCBDYXJvbGluYSxDPVVT
rj@rpi:~/fabric_sdk $
```

Figure 4.10. Output of API Connection Testing.

Once the connection is tested and passed successfully, the API server can be started by the following command. The Express server launches and starts listening for incoming HTTP requests, ready to process operations against the blockchain network.

```
// Start the API Server
node app.js
```

4.2.2.2. Dockerizing the API for Deployment

To enhance portability and simplify deployment, the Express API is containerized using Docker. This approach ensures consistent operation across different environments and simplifies scaling and management. The implementation begins with creating a `.dockerignore` file to specify which files and directories should be excluded from the Docker build context and prevents unnecessary files from being included in the build.

```
cd $HOME/fabric_sdk
nano .dockerignore

# Paste this inside .dockerignore
node_modules
npm-debug.log
.git
.gitignore
```

Next, a Dockerfile is created to define the container image specifications. The Dockerfile contains the following instructions: uses Node.js 18 as the base image, sets up the working directory in the container, copies `package.json` and `package-lock.json` from the current directory to install dependencies, installs all required NPM packages, copies the application

code into the container, exposes port 3000 for accessing the API, and defines the command to start the application.

```
# Create a Dockerfile for the API
```

```
nano Dockerfile
```

```
# Paste this inside Dockerfile
```

```
FROM node:18
```

```
# Create app directory
```

```
WORKDIR /usr/src/app
```

```
# Install app dependencies
```

```
COPY package*.json ./
```

```
RUN npm install
```

```
# Bundle app source
```

```
COPY . .
```

```
# Expose port the app runs on
```

```
EXPOSE 3000
```

```
CMD [ "node", "app.js" ]
```

The docker image is then built and the following command creates a Docker image named “fabric_sdk” based on the instructions in the Dockerfile.

```
# Build the Docker image
```

```
docker build -t fabric_sdk .
```

To run the container, the following command is executed. This command uses the host network to simplify communication with the Hyperledger Fabric network, mounts the local

wallet and network directories as volumes in the container, ensuring persistence of credentials and connection information, names the container “fabricAPI” for easier reference, and uses the previously built fabric_sdk image.

```
# Start the API  
docker run --network="host" -v $(pwd)/wallet:/usr/src/app/wallet -v  
$(pwd)/network:/usr/src/app/network --name fabricAPI fabric_sdk
```

To verify the container is running correctly, Docker’s process status and logs are checked. These commands should show the fabricAPI container in the running state and reporting logs on the API server is listening on the port 3000, indicating successful deployment. For stopping and removing the fabricAPI container, the commands are also attached the following to facilitate container lifecycle management, allowing for clean shutdown and removal when updating the application or performing maintenance.

```
# Check docker process  
docker ps  
  
# Check docker logs for fabricAPI  
docker logs fabricAPI  
  
# Stop the API  
docker stop fabricAPI  
  
# Remove the API  
docker rm fabricAPI
```

This containerized approach significantly enhances the deployment flexibility of the API, making it easier to integrate with containerized environments, orchestration systems like Kubernetes, and continuous integration/continuous deployment (CI/CD) pipelines. It also

ensures that the API operates in a consistent environment regardless of the underlying host system, reducing potential compatibility issues.

4.2.3. Web Application

4.2.3.1. Setting Up the Web Application Environment

After implementing Hyperledger Fabric network and the Hyperledger Fabric SDK API layer, the final component of the system involves the development of a web-based user interface that will allow users to interact with the traffic violation logging system. This web application provides a user-friendly interface for viewing, managing, and interacting with the blockchain-stored records for both traffic authorities and drivers.

The implementation starts with installing the necessary web server components. These packages provide Apache2 (a robust and widely-used web server), PHP (server-side scripting language), libapache2-mod-php (Apache module for processing PHP files), php-mongodb (PHP extension for MongoDB connectivity), php-curl (PHP extension for making HTTP requests to the Fabric SDK API), and php-mbstring (PHP extension for handling multibyte string operations).

```
# Install Apache2, PHP, and MongoDB  
sudo apt update && sudo apt install apache2 php libapache2-mod-php php-mongodb  
php-curl php-mbstring -y
```

Next, a dedicated directory for the web application is created within Apache's default web root. The following commands create the application directory, set appropriate execution permissions, and assign ownership to the current user for ease of development. The web application files are then copied from the cloned project repository to the newly created directory.

```
# Navigate to the web server root directory
```

```
cd /var/www/html

# Create a directory for the web application
sudo mkdir chainguard

# Change ownership of the directory to the current user
sudo chmod +x chainguard && cd chainguard
sudo chown rj:rj /var/www/html/chainguard

# Copy the web application files from the cloned repository to the web server
root directory
cp $HOME/FYP_81049_ChainGuard/web_ui/* /var/www/html/chainguard
```

4.2.3.2. Setting Up the MongoDB Database

While the blockchain stores the core violation records on the ledger, a MongoDB database is deployed to handle off-chain data such as user account, violation IDs, and appeal IDs. MongoDB is chosen for its flexibility, performance, and easy deployment within Docker container. Before setting up MongoDB database, a directory is created for data persistency, and the MongoDB container is launched. The following commands creates a named container 'mongodb', runs it in detached mode, maps the container's MongoDB port to the host, configures automatic restart on system boot, mounts a local directory for persistent data storage, and uses the latest official MongoDB image.

```
# Install MongoDB Docker Container

mkdir mongodb_data

docker run --name mongodb -d -p 27017:27017 --restart always -v
/var/www/html/chainguard/mongodb_data:/data/db mongo:latest
```

For database management and visualization, MongoDB Compass is installed from the official website (<https://www.mongodb.com/try/download/compass>) as shown in Figure 4.11. For connecting to the database, the connection string `mongodb://localhost:27017` or the server's IP address can be mapped in MongoDB Compass for access. Figure 4.12. shows the mapping of the MongoDB in MongoDB Compass. In addition, if any issues occur with the MongoDB container, diagnostic commands are available for troubleshooting.

```
# Any error regarding MongoDB
```

```
docker container logs mongodb
```

```
docker rm mongodb
```

```
docker stop mongodb
```

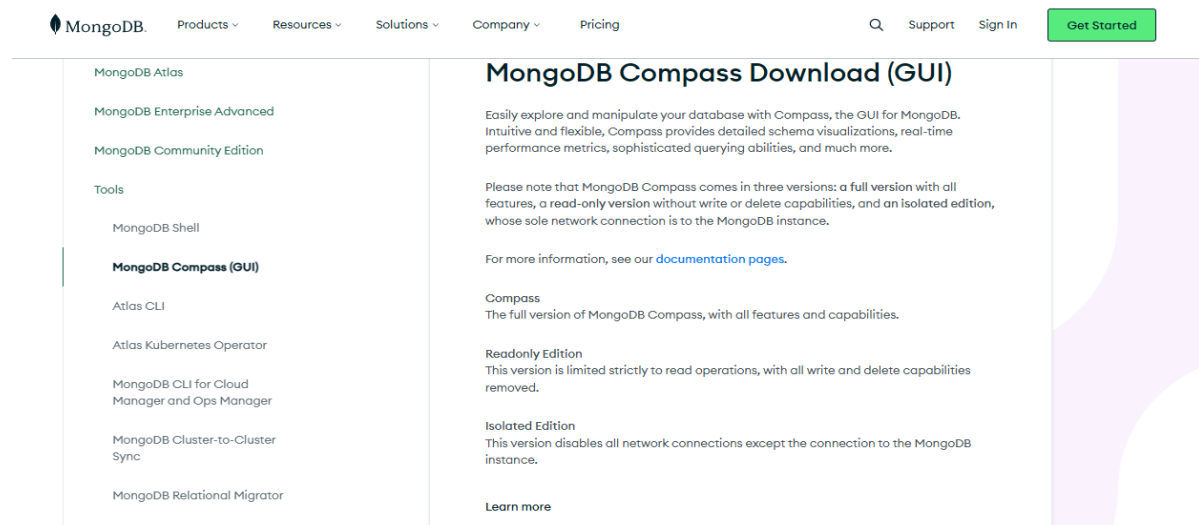


Figure 4.11. MongoDB Compass Download Page.

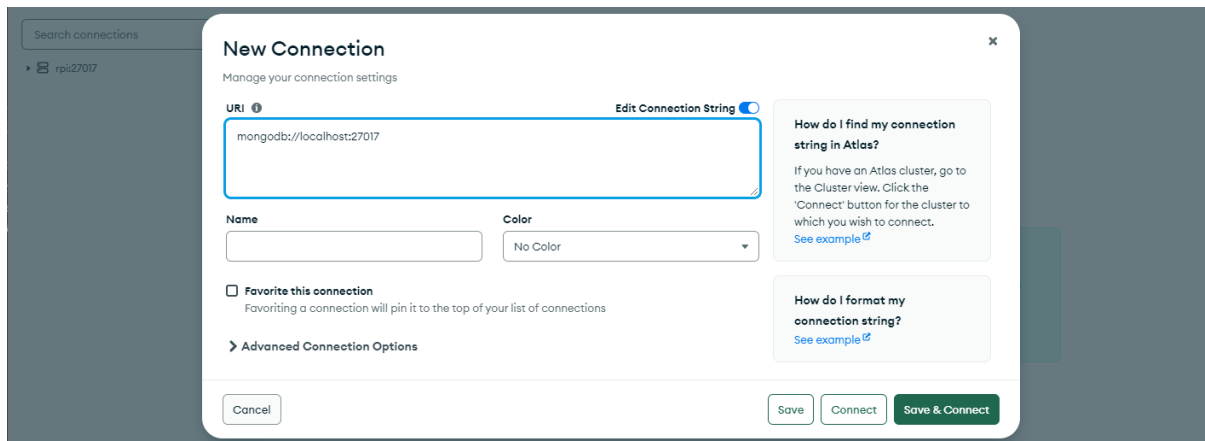


Figure 4.12. MongoDB Connection Setup.

4.2.3.3. PHP Dependencies with Composer

To manage PHP dependencies efficiently, Composer is installed through a series of commands that download the Composer installer, verify the installer's integrity against the official hash, and install Composer globally on the system. With Composer installed, the required PHP packages are added to the project. These commands install mongodb/mongodb (Official PHP MongoDB driver), stripe/stripe-php (for payment processing integration), and phpmailer/phpmailer (for sending email notifications). Apache is restarted to ensure all PHP extensions are properly loaded, and the final command displays a list of all installed packages.

```
# Composer Installation

cd ~

curl -sS https://getcomposer.org/installer -o /tmp/composer-setup.php

HASH=`curl -sS https://composer.github.io/installer.sig`

echo $HASH

php -r "if (hash_file('SHA384', '/tmp/composer-setup.php') === '$HASH') { echo
'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-
setup.php'); } echo PHP_EOL;"

sudo php /tmp/composer-setup.php --install-dir=/usr/local/bin --
filename=composer
```

```
# Composer Package Dependencies Installation
```

```
cd /var/www/html/chainguard
```

```
composer require mongodb/mongodb
```

```
sudo systemctl restart apache2
```

```
composer require stripe/stripe-php
```

```
composer require phpmailer/phpmailer
```

```
composer show
```

4.2.3.4. Apache Web Server Configuration

To properly serve the web application, Apache's default site configuration is modified. The configuration is updated to set the document root to the application directory, disable directory listing for security, enable .htaccess file usage with AllowOverride All, grant access permissions, and configure logging for troubleshooting. Additionally, an .htaccess file is created for URL rewriting and default page specification, which disables directory browsing and redirects the root to the login page.

```
# Apache Web Server Configuration
```

```
# Set apache2 root directory
```

```
cd /var/www/html/chainguard
```

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

```
# Paste this inside the file
```

```
<VirtualHost *:80>
```

```
    ServerAdmin webmaster@localhost
```

```
    DocumentRoot /var/www/html/chainguard
```

```
    <Directory /var/www/html/chainguard>
```

```
        Options -Indexes
```

```

        AllowOverride All

        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

# Enable the site configuration
sudo nano /var/www/html/chainguard/.htaccess

# Paste this inside the file
# Disable directory browsing
Options -Indexes

# Redirect root to the login page
DirectoryIndex /php/login.php

```

Moreover, the Apache rewrite module is enabled to support URL rewriting, and appropriate permissions are set on the application directory. These steps enable the mod_rewrite module for URL manipulation, set the web server user (www-data) as the owner of application files, apply appropriate read and execute permissions, and restart Apache to apply all changes.

```

# Enable URL rewriting

sudo a2enmod rewrite

sudo chown -R www-data:www-data /var/www/html/chainguard

sudo chmod -R 755 /var/www/html/chainguard

sudo systemctl restart apache2

```

With all the components set, the web application can be accessible through the web browser by navigating to `http://<ip_address>/`, where the `<ip_address>` is the server's IP address. This implementation creates a full-stack solution that combines a blockchain backend for immutable record storage, a Node.js API for blockchain interaction, a web-based user interface for system access, a MongoDB database for user management and application data, and security features including authentication, authorization, and secure communication. This result in a system that provides a robust solution for managing traffic violation records with the enhanced security, transparency, and immutability afforded by blockchain technology, while maintaining user-friendly access through the web interface.

4.2.4. Automation For System Deployment

After all the required components are installed, the system can be started or shut down with scripts. The implementation of this project uses an automation framework that significantly simplifies the deployment, management, and termination processes of the Hyperledger Fabric network and its associated components. This framework consists of custom shell script that automate the complex setup and shutdown sequences, ensuring consistency and reliability across different environments. It consists of three shell scripts, which are “setup_fabric.sh” for complete deployment of the Hyperledger Fabric network and all supporting services, “setup_explorer.sh” for configuring and deploying the Hyperledger Explorer, and “shutdown_fabric.sh” for performing a clean shutdown of all the system components.

First of all, the “setup_fabric.sh” begins with prerequisite checks for required software and directories. It then proceeds to configure and initialize the Hyperledger Fabric network with appropriate channels deploy monitoring tools like Prometheus and Grafana for system health tracking, and install and deploy the traffic violation chaincode. In addition, it also sets up MongoDB to store off-chain application data, configures the Fabric SDK with connection details and certificates, and enrolls admin users for both organizations. Finally, it launches the Fabric API service in a Docker container and triggers the Explorer setup process. Throughout this process, the script uses detailed error handling and logs each step with a color-coded and timestamped messages to help with tracking and logging.

For the configuration of Hyperledger Explorer, the “setup_explorer.sh” script starts by handling the management of cryptographic material transfer from the network to the Explorer directory, allowing it to access and monitor all the transactions within the blockchain network. It automatically identifies and configures private key paths, updates connection profiles with

correct certificate paths, and launches the Explorer Docker container with proper configuration. One of the key sections of this script is that it automatically detects and sets up cryptographic materials using tools like jq to handle JSON files, eliminating the need for manual intervention in configuring the Explorer, which is traditionally a complex and error-prone process.

Furthermore, the “shutdown_fabric.sh” script ensures clean termination of all components by shutting down the Hyperledger Fabric network with proper cleanup, terminating monitoring services such as Prometheus, Grafana and Explorer, and removing API container. All operations include verification steps to confirm successful completion, checking the state of each component before attempting shutdown operations. This approach prevents errors and ensures a clean system state after termination, making subsequent deployments more reliable. Moreover, all the automation scripts include detailed logging functionality, which include timestamped entries for all operations, color-coded console output for immediate status recognition, detailed command execution logs, and persistent log files stored in the “~/fabric_logs/” directory. In terms of error handling, the scripts also log error capture with debugging information, facilitating troubleshooting and providing an audit trail for deployment activities.

The standard deployment workflow begins with copying the automation scripts to appropriate directories. The setup script is then executed to deploy the entire system, after which the system can be accessed through the provided endpoints. When necessary, the system can be terminated using the shutdown script. The complete process can be summarized in a few commands:

```
# Copy the automation scripts from cloned GitHub repository  
cp $HOME/FYP_81049_ChainGuard/setup_fabric.sh $HOME  
cp $HOME/FYP_81049_ChainGuard/shutdown_fabric.sh $HOME
```

```
cp $HOME/FYP_81049_ChainGuard/setup_explorer.sh
$HOME/go/src/github/Kae/explorer

# Start the system

./setup_fabric.sh

# Shutdown the system

./shutdown_fabric.sh
```

Upon successful deployment, the system provides access to various services through standardized endpoints. The main ChainGuard user interface is accessible via the web application at <http://<ip>/>, while container management is available through Portainer at <https://<ip>:9000/>. System metrics can be visualized using Grafana at <http://<ip>:3002/>, and the state database interface is provided by CouchDB at http://<ip>:5984/_utils/. The blockchain network can also be monitored using Hyperledger Explorer at <http://<ip>:8081>. These endpoints provide a complete set of tools for managing and monitoring the blockchain network and its components, allowing a better system management and maintenance.

4.3. System Architecture and Network Design

The traffic violation logging and penalty management system employs a multi-tier architecture that leverages blockchain technology to provide immutable, transparent, and secure record-keeping. Before diving into the functionalities of the system, it is essential to understand the underlying system architecture and network design that forms the foundation of the application. This section will focus on the Hyperledger Fabric blockchain network configuration and the overall system flow, providing a detailed overview of how different components used in this system interact to create a cohesive solution.

4.3.1. Hyperledger Fabric Blockchain Network Configuration

The system is built on a Hyperledger Fabric blockchain network with a customized configuration designed to meet the requirements of a traffic violation management system. The network consists of two organizations representing different stakeholders in the traffic management ecosystem: Organization 1 representing Traffic Authorities and Organization 2 representing Drivers. Each organization operates one peer node within the network, which maintains a copy of the distributed ledger and executes the smart contracts that enforce the business logic across the network, as shown in Figure 4.13.

Following the permissioned blockchain architecture of Hyperledger Fabric, each organization has a designated admin responsible for enrolling and registering users. This ensures that only authorized entities can participate in the network, maintaining security and compliance with regulatory requirements. The business logic governing traffic violation records is implemented through chaincode also known as smart contract for other blockchain framework, written in Go language. This chaincode defines the rules for creating, updating, and querying violation records, ensuring consistency and integrity across the network.

Both organizations utilize CouchDB as the state database rather than the default LevelDB. CouchDB supports rich queries against the blockchain data, allowing for more complex data retrieval patterns essential for a traffic violation management system. This configuration required specific modifications to the docker-compose.yaml file to properly configure and initialize the CouchDB instances for both organizations.

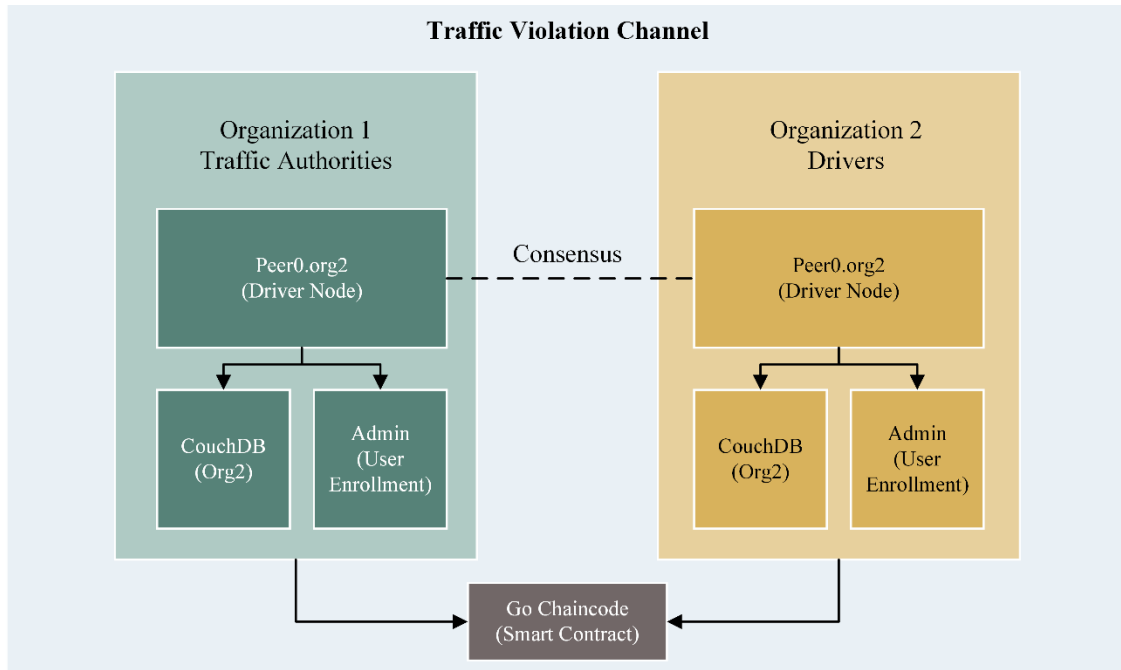


Figure 4.13. Hyperledger Fabric Network Configuration

4.3.2. System Flow and Component Integration

The system follows a multi-tier architecture with several components interacting in sequence to provide a complete solution for managing traffic violations. The flow follows a linear path: Web Application ↔ MongoDB (Database) ↔ Express.js (API) ↔ Fabric Blockchain Network. This architecture can be visualized through the component diagram shown in Figure 4.14.

The flow of information through the system occurs in a systematic manner. Users interact with the web application through their browsers, providing a familiar and accessible interface for all stakeholders. Off-chain data such as user information is handled by MongoDB,

which serves as the application database separate from the blockchain. The Express.js API layer functions as a bridge, receiving requests from the web application and translating them into blockchain transactions using the Fabric SDK. Finally, the Fabric blockchain network processes these transactions and maintains the immutable record of traffic violations.

This separation of concerns ensures that each component can be developed, tested, and maintained independently while working together to provide a cohesive system. The modular design also allows for future extensions or replacements of individual components without disrupting the entire system, providing flexibility and scalability as requirements evolve.

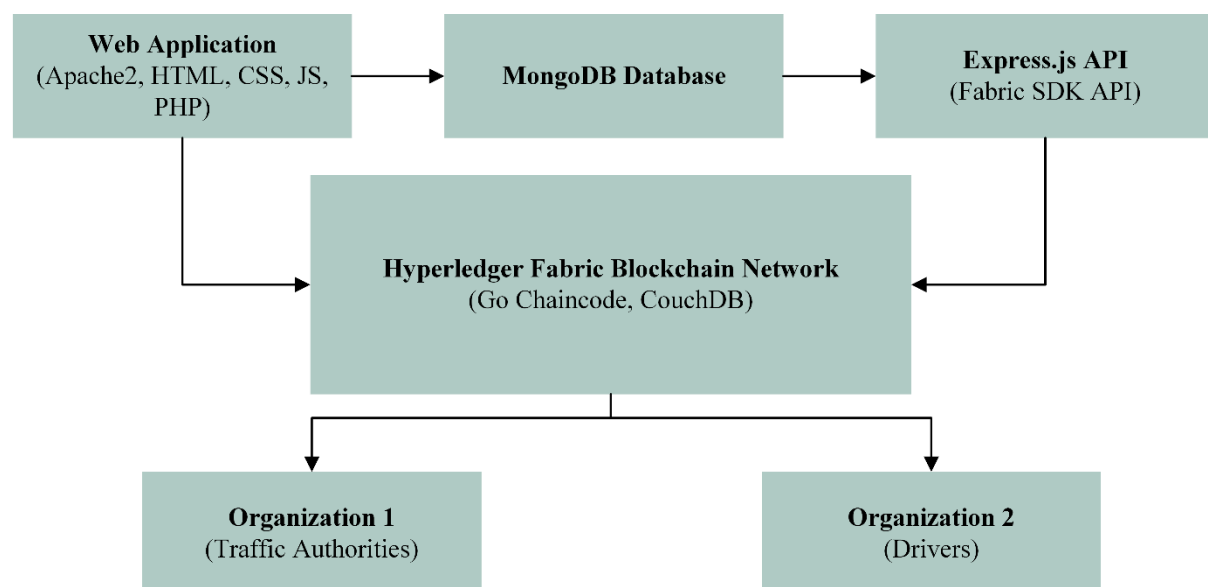


Figure 4.14. ChainGuard System Architecture.

4.4. Proposed System Implementation

This section details the implementation of ChainGuard, the blockchain-based traffic violation logging and penalty management system's user interfaces and functionalities. Each component of the application is showed with detailed descriptions of its purpose, technical implementation, user interactions, and integration with the blockchain network.

4.4.1. Login

After setting up the requirements, the system can be accessed at <http://localhost>. This will lead the user to a secure login page as shown in Figure 4.15, that serves as the gateway to the ChainGuard platform. This interface requires all users to authenticate with their credentials such as email and password before accessing the system features, which ensures that only authorized individuals can interact with the blockchain-based violation records.

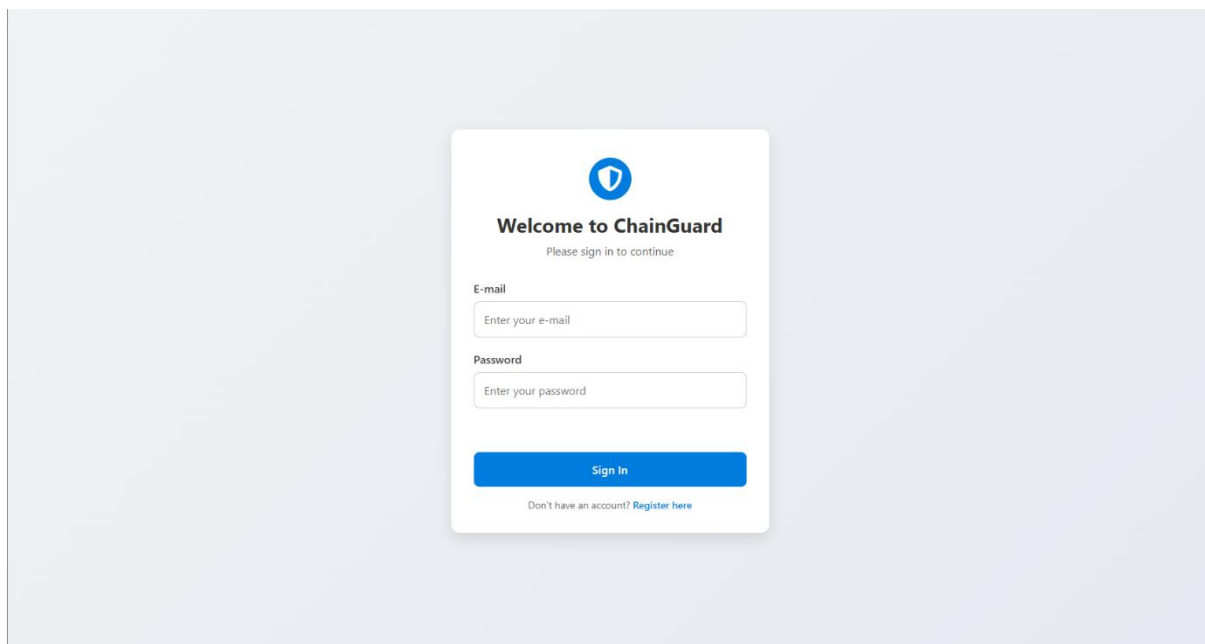


Figure 4.15. ChainGuard Login Page.

4.4.2. Driver Registration

For new users, the driver registration page as shown in Figure 4.16. enables account creation by collecting necessary personal and vehicle information. This interface guides the users through a structured registration process, requiring essential details such as name, contact information, license details, and vehicle registration data. If there is any missing information, the system will prompt the user to complete all the required field. The driver registration interface also checks for existing user by checking the email and contact number, blocking multiple account creation with same credentials, and preventing possible spamming and bot account creation. With the registration complete, it establishes the driver's identity within the blockchain network for future violation tracking and management, creating a unique digital identity that will be associated with all subsequent transactions and records within the system.

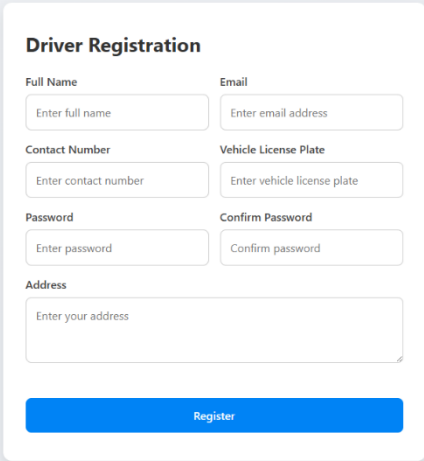
The image shows a 'Driver Registration' form centered on a light blue background. The form is a white card with a title 'Driver Registration' at the top. It contains several input fields: 'Full Name' (with placeholder 'Enter full name'), 'Email' (with placeholder 'Enter email address'), 'Contact Number' (with placeholder 'Enter contact number'), 'Vehicle License Plate' (with placeholder 'Enter vehicle license plate'), 'Password' (with placeholder 'Enter password'), 'Confirm Password' (with placeholder 'Confirm password'), and 'Address' (with placeholder 'Enter your address'). A blue 'Register' button is located at the bottom of the form.

Figure 4.16. Driver Registration Page.

4.4.3. Pending Violation

Upon successful authentication, drivers are directed to their primary dashboard. Figure 4.17 shows the pending violations which requires the driver immediate attention. This interface presents critical information including violation details, fine amounts, and payment deadlines, helping drivers prioritize their responses. The pending violation dashboard uses visual indicators to highlight urgent items. The dashboard also offers straightforward links to the payment processing module, allowing drivers to settle fines without delay. By integrating essential violation information and simplifying the path to resolution, this interface functions as the principal tool for managing active violations and maintaining adherence to traffic regulations.

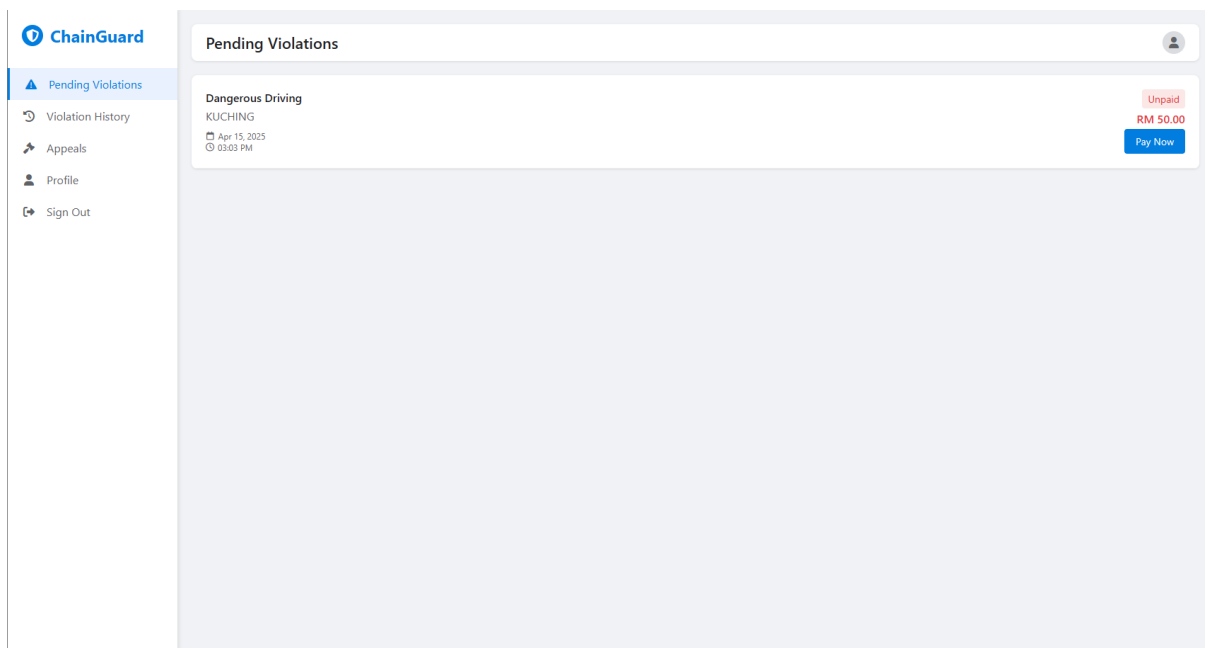


Figure 4.17. Pending Violation Page.

4.4.4. Driver Payment Process

When a driver chooses to settle a fine, the process begins with a loading screen shown in Figure 4.18 and then transitions seamlessly to the secure Stripe payment interface shown in Figure 4.19. This project employs a test payment method because integrating a live gateway would incur per-transaction fees or require an annual subscription. Throughout the workflow, transactions are processed safely and efficiently, and each payment status is recorded immutably on the blockchain. Once the payment is confirmed, the system updates the violation status automatically and returns the driver to the pending violations dashboard.

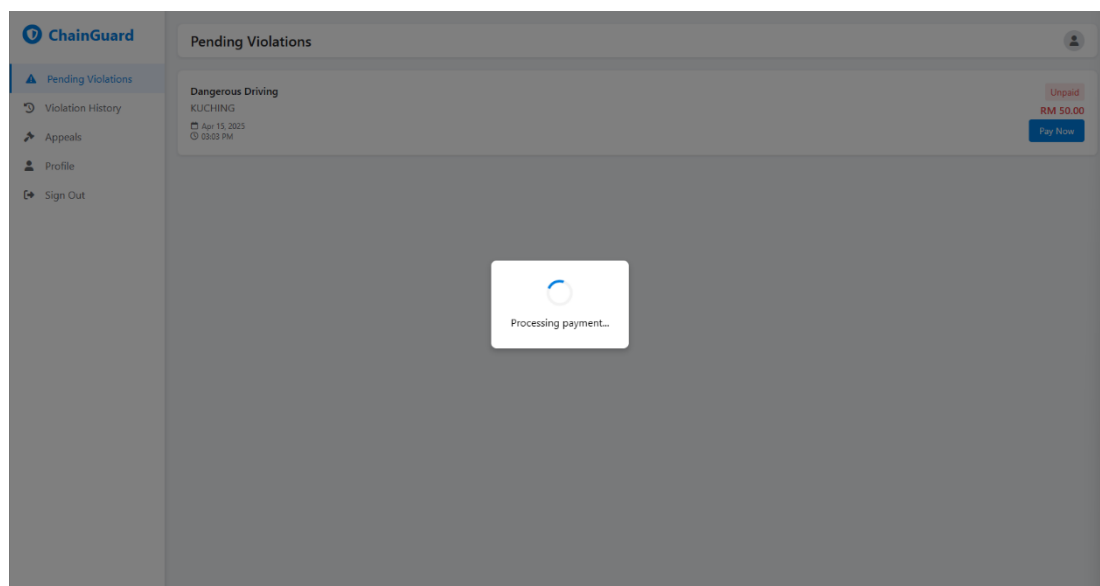


Figure 4.18. Payment Loading Screen.

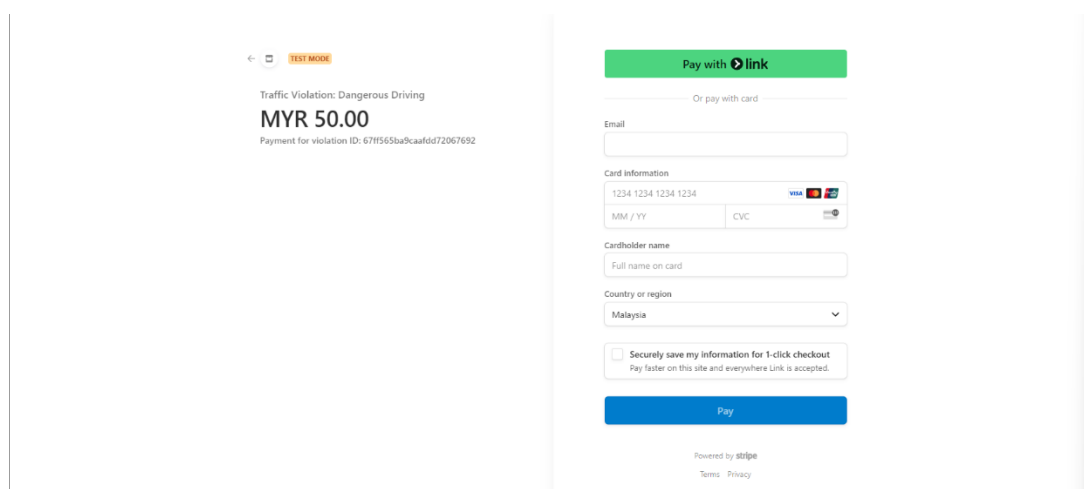
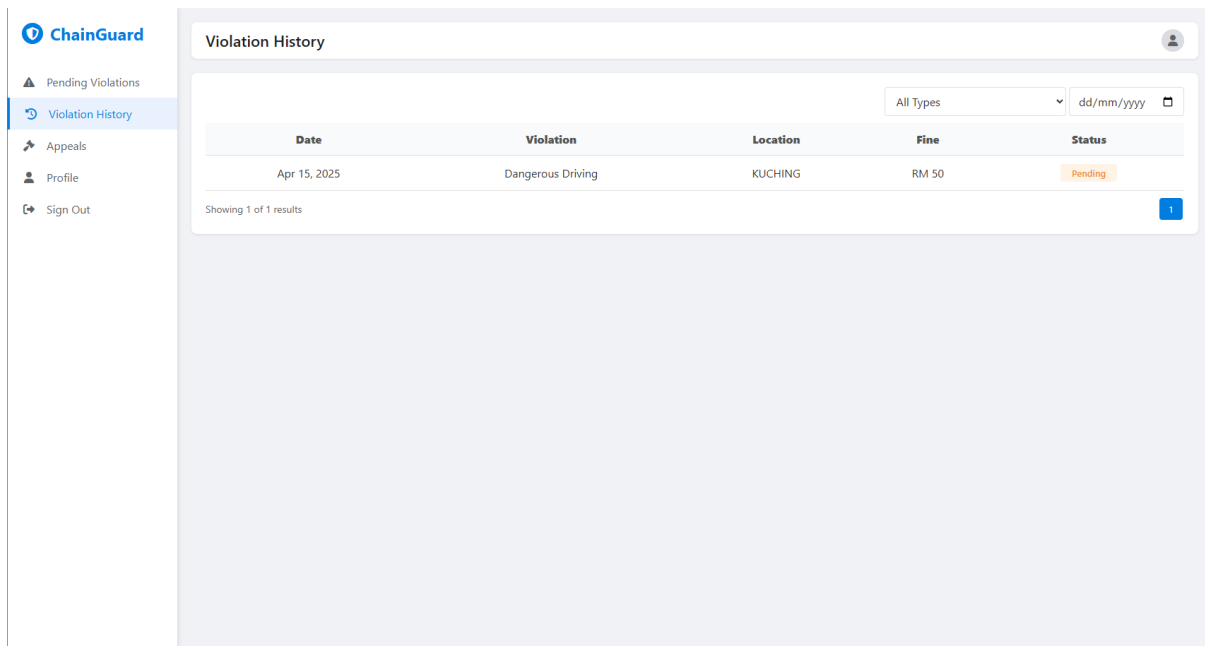


Figure 4.19. Stripe Payment Gateway Page

4.4.5. Violation History

Additionally, the driver's violation history page shown in Figure 4.20 serves as a complete record list, allowing drivers to track all current and past records with ease. This interface also implements filtering and search capabilities to help drivers locate specific violations quickly and efficiently. Moreover, each violation record can be expanded by clicking on the record itself as shown in Figure 4.21 to reveal specific details including time, location, classification, evidence and processing status. The violation ID also can be copied through the pop-up violation details, allowing drivers to use the ID for appeal if they wish to make an appeal. The immutable nature of blockchain ensures that this history remains accurate and tamper-proof, providing drivers with reliable access to their complete violation records.



Date	Violation	Location	Fine	Status
Apr 15, 2025	Dangerous Driving	KUCHING	RM 50	Pending

Showing 1 of 1 results

Figure 4.20. Driver's Violation Record Page.

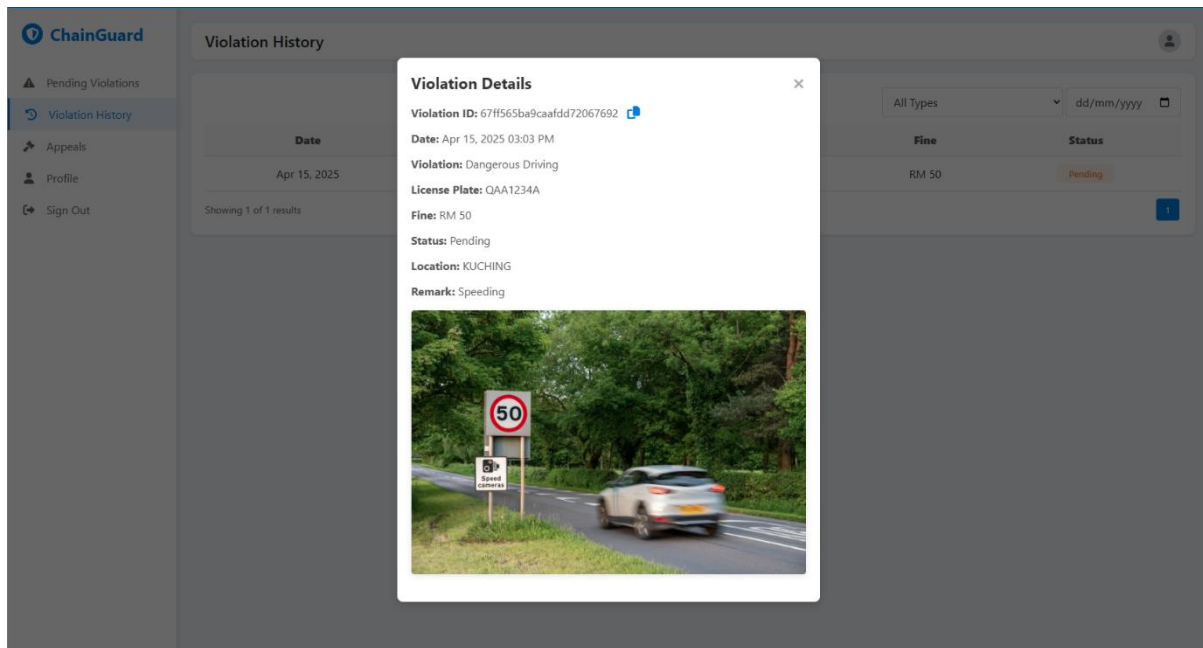


Figure 4.21. Driver's Violation Record Details

4.4.6. Appeal Submission

Drivers who wish to contest a violation may do so through the appeal submission page shown in Figure 4.22, where they are required to provide supporting evidence and a clear justification. All fields must be completed to ensure a comprehensive basis for the appeal. Upon successful submission, the system displays a confirmation notification, as illustrated in Figure 4.23, offering clear feedback and enhancing user confidence. The appeal history page in Figure 4.24 displays all submitted appeals along with their current statuses, allowing drivers to monitor the progress of each case in real time. For more detailed information, the appeal details page in Figure 4.25 presents specific data including the violation ID, driver's name, reason for appeal, current status, and date of submission. This integrated appeal management system promotes transparency, accountability, and user engagement throughout the dispute resolution process.

ChainGuard

- Pending Violations
- Violation History
- Appeals**
- Profile
- Sign Out

Appeals

Violation ID:
67ff565ba9caafdd72067692

Appeal Reason:
Wrong violation

Supporting Evidence:
Support image



Figure 4.22. Driver's Appeal Submission Page.

ChainGuard

- Pending Violations
- Violation History
- Appeals**
- Profile
- Sign Out

Appeals

Appeal submitted successfully.

Violation ID:

Appeal Reason:

Supporting Evidence:
Support image

Cancel Submit Appeal

Figure 4.23. Driver's Appeal Submission Successful Notification.

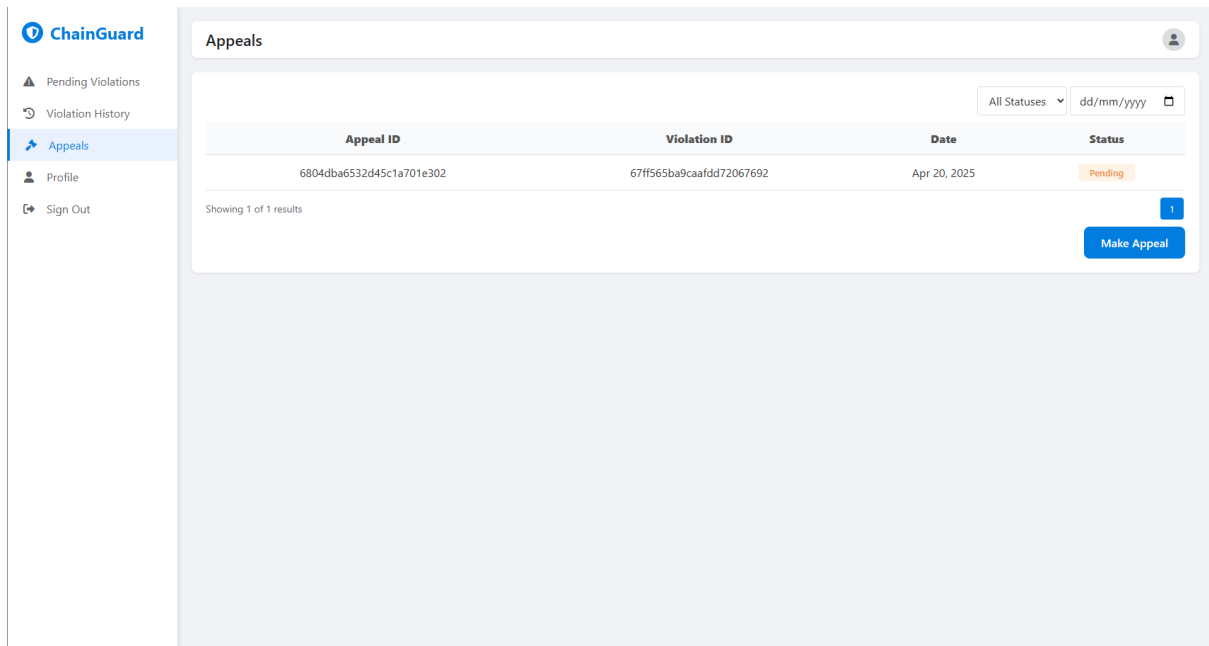


Figure 4.24. Driver's Appeal History Page.

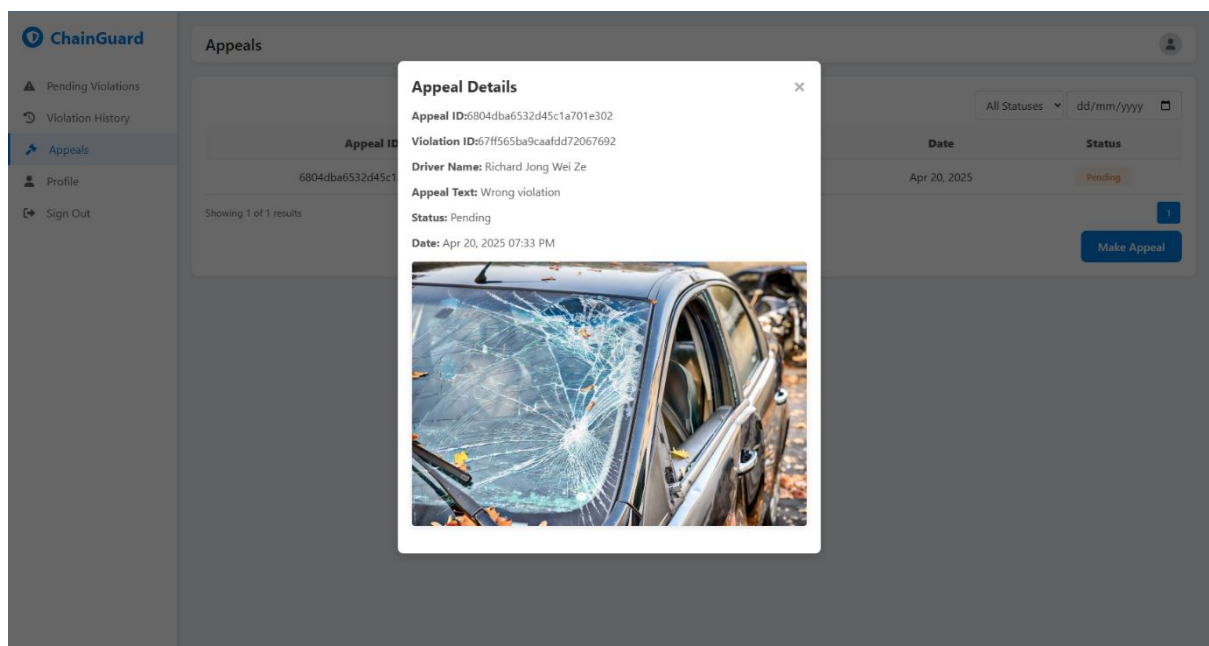


Figure 4.25. Driver's Appeal Details.

4.4.7. Driver Profile

The driver's profile page, as shown in Figure 4.26, serves as a centralized interface for viewing and managing personal information within the ChainGuard system. It displays key user details such as the driver's full name, email address, contact number, license plate number, address, and a unique wallet ID. This layout offers a clear and organized overview of essential data, allowing users to verify their information at a glance. An "Edit Profile" button is conveniently provided, granting easy access to make updates when necessary. By consolidating critical user information into a single interface, this page enhances navigation efficiency and supports seamless interaction with other system features.

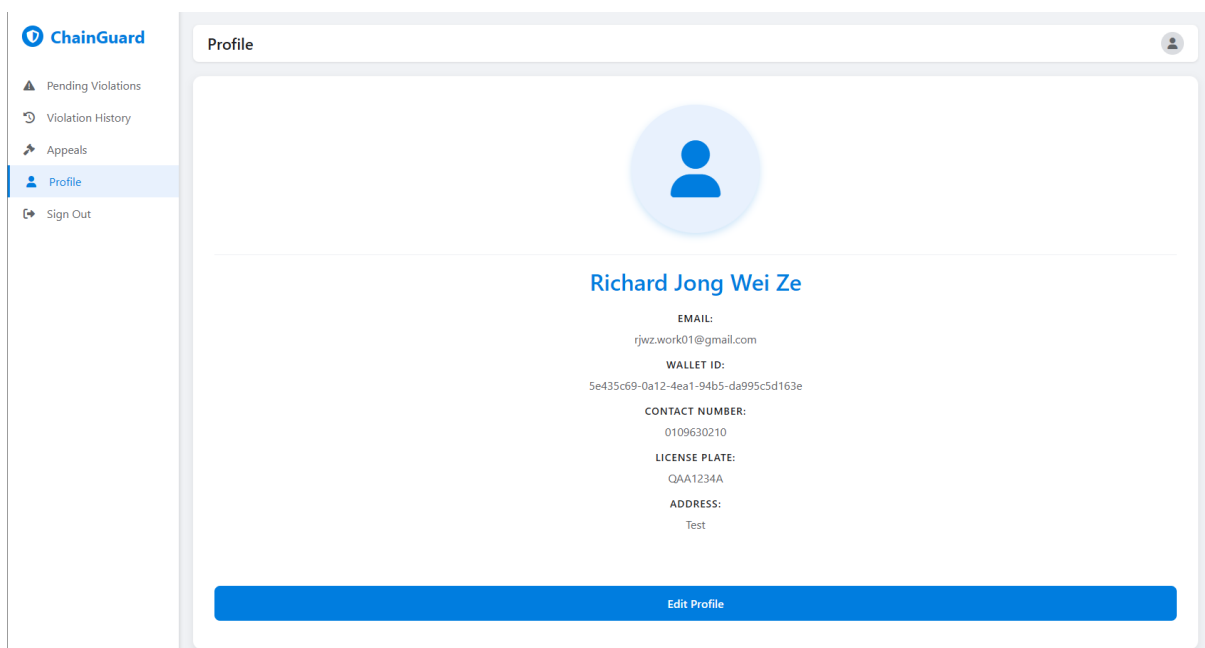


Figure 4.26. Driver's Profile Page.

4.4.8. Edit Driver Profile

The edit profile interface, illustrated in Figures 4.27, enables drivers to manage and maintain accurate personal records within the ChainGuard system. This interface provides editable fields for key user details such as full name, email address, contact number, license

plate, and address. Drivers can update this information as needed to ensure their profiles remain current and reflective of any changes. Additionally, the system displays a unique wallet ID, which is fixed and cannot be modified by the user. Validation checks are implemented to maintain data accuracy, and clear feedback is provided upon successful updates, ensuring that all changes are securely saved to the blockchain. This functionality helps maintain reliable communication channels and ensures that system notifications, including violation alerts and appeal responses, are delivered to the correct recipients.

ChainGuard

- ▲ Pending Violations
- 🕒 Violation History
- 🔍 Appeals
- 👤 Profile**
- 🚪 Sign Out

Edit Profile


[Change Photo](#)

Personal Information

Full Name
Richard Jong Wei Ze

Email Address
rjwz.work01@gmail.com

Contact Number
0109630210

License Plate
QAA1234A

Address
Test

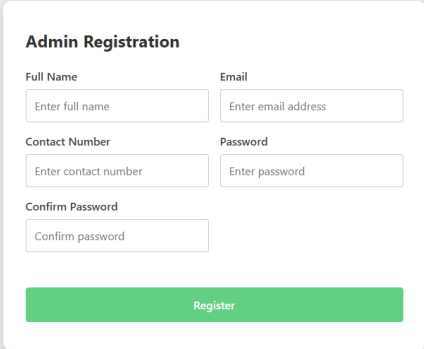
Wallet ID
5e435c69-0a12-4ea1-94b5-da995c5d163e
Cannot be changed

[Cancel](#) [Save Changes](#)

Figure 4.27. Edit Driver's Profile Page.

4.4.9. Admin Registration

The admin registration interface, shown in Figure 4.28, provides a streamlined onboarding process for system administrators within the ChainGuard platform. This carefully designed form captures essential administrative credentials through five key fields: Full Name, Email, Contact Number, Password, and Confirm Password. Each field features clear labelling and appropriate input validation to ensure data integrity. The registration process is secured through password confirmation requirements, helping prevent entry errors during account creation. Upon completing the form, administrators can finalize their registration by clicking the green "Register" button, which submits their credentials for verification and account activation. This interface serves as the gateway for authorized personnel to access the administrative dashboard, where they can manage driver records, review violation reports, process appeals, and perform other system oversight functions critical to maintaining the system operational integrity.

The image shows a web form titled "Admin Registration" centered on a light gray background. The form is a white card with rounded corners. It contains five input fields arranged in three rows. The first row has "Full Name" and "Email". The second row has "Contact Number" and "Password". The third row has "Confirm Password". Each field has a placeholder text: "Enter full name", "Enter email address", "Enter contact number", "Enter password", and "Confirm password". At the bottom of the form is a green button with the text "Register".

Admin Registration	
Full Name	Email
Enter full name	Enter email address
Contact Number	Password
Enter contact number	Enter password
Confirm Password	
Confirm password	
Register	

Figure 4.28. Admin Registration Page.

4.4.10. Admin Dashboard

Figure 4.29 shows the admin dashboard. It provides real-time analysis and key performance indicators for traffic authorities in managing the violation records. At the top, there are four statistics cards display critical metrics which are Total Violations, Fines Collected, Pending Appeals, and Users. In addition, the dashboard features three graphical data visualizations: a bar chart showing Violations by, a line graph illustrating Monthly Trends for selected year, and a Daily Activity chart displaying granular violation data by day. This interface also allows the traffic authorities to export the statistical reports for offline analysis or reporting purposes.

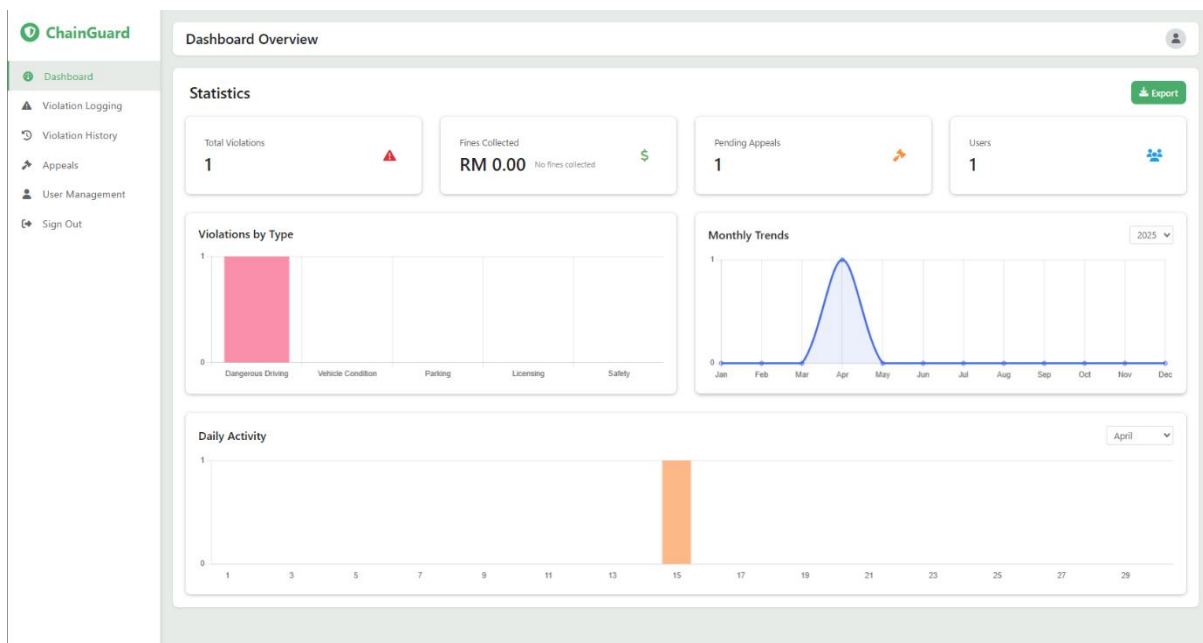


Figure 4.29. Admin Dashboard.

4.4.11. Violation Logging

As shown in Figure 4.30, the violation logging page provides a structured interface for recording new violation into the blockchain network, which ensures immutable documentation of each log. This interface guides the traffic authorities through the process of entering violation details, uploading evidence, classifying the violation, and calculating appropriate fines. In

addition, the integration with Hyperledger Explorer in Figure 4.31 offers traffic authorities immediate transparency into blockchain activities, allowing them to observe block creation in real-time as new violations are recorded. This visual confirmation of blockchain transactions ensures traffic authorities can verify that each violation has been properly committed to the distributed ledger. When a violation is logged, the system automatically dispatches email notifications to affected drivers as shown in Figure 4.32, ensuring immediate awareness of new violations. This interface implements validation checks to maintain data quality and consistency across all violation records.

The screenshot displays the 'ChainGuard' application interface. On the left, a sidebar menu includes 'Dashboard', 'Violation Logging' (highlighted with a green triangle), 'Violation History', 'Appeals', 'User Management', and 'Sign Out'. The main panel is titled 'Log New Violation' and contains a form for recording violations. The form fields are: 'Violation Type' (a dropdown menu currently set to 'Dangerous Driving'), 'Location' (a text input field), 'Time' (a date and time picker showing 'dd/mm/yyyy --:-- --'), 'License Plate Number' (a text input field), 'Penalty Amount' (a text input field), 'Remark' (a text area), and 'Violation Image' (an image upload area with the text 'Upload image'). At the bottom right of the form, there are three buttons: 'Cancel', 'Refresh Violations', and 'Save Changes'.

Figure 4.30. Violation Logging Page

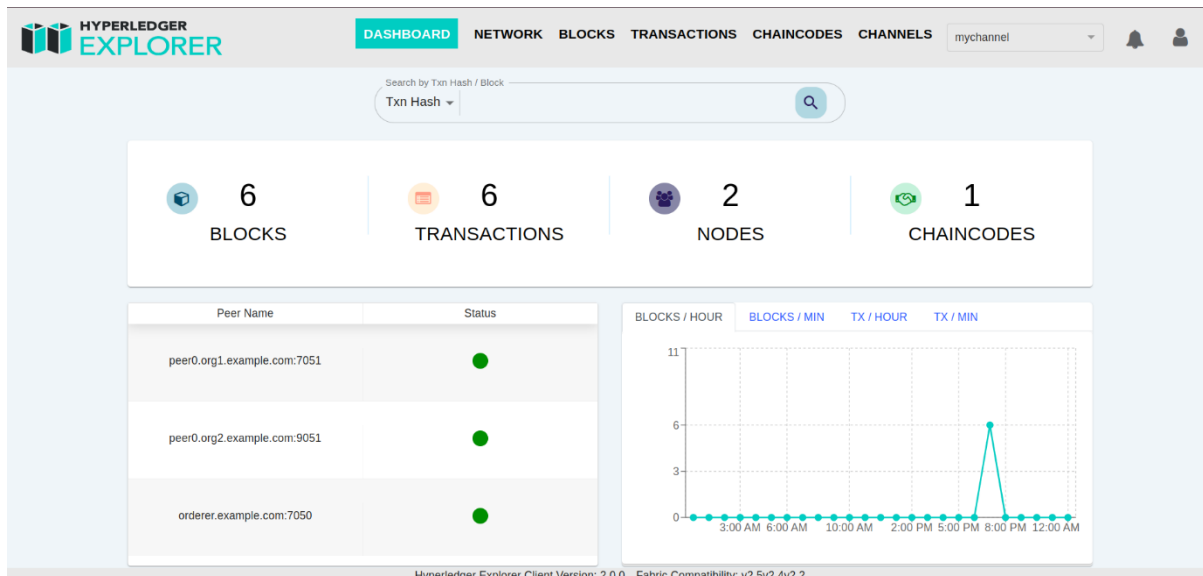


Figure 4.31. Hyperledger Explorer Dashboard

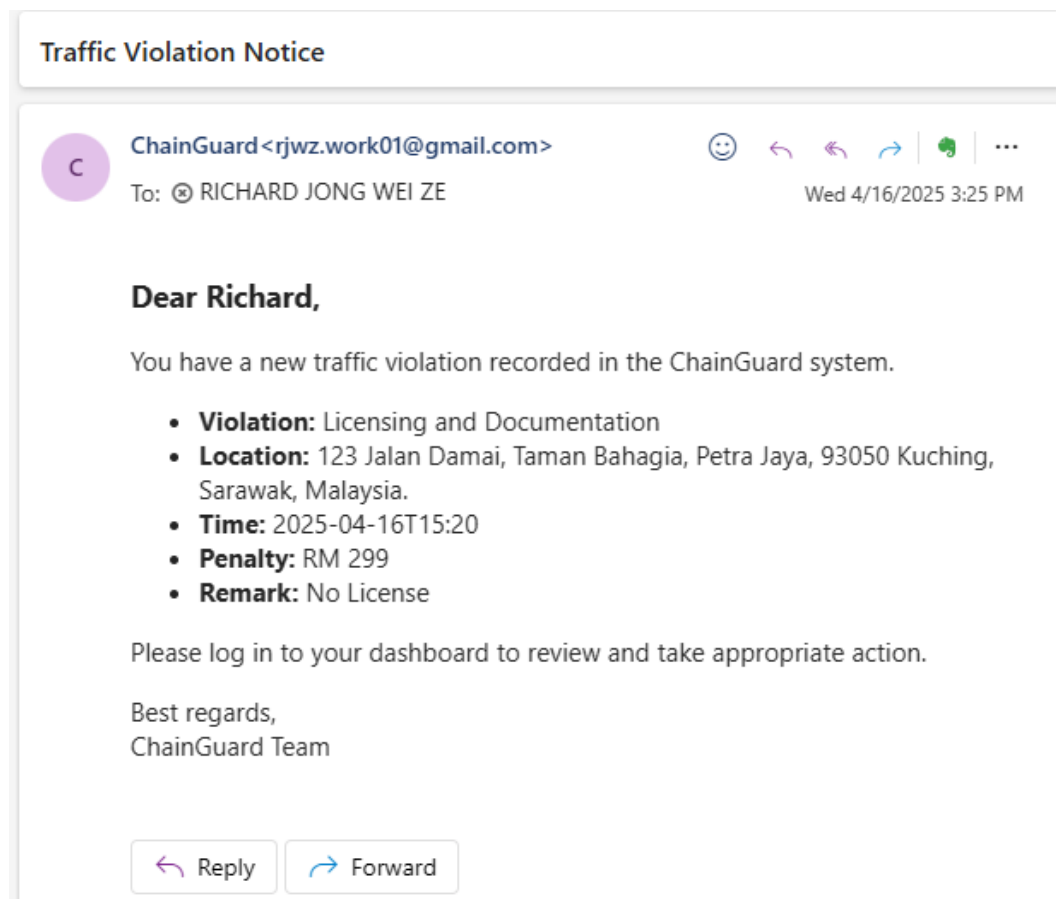


Figure 4.32. Driver's Email Notification

4.4.12. Violation Record Management

Traffic authorities can access detailed records of all violations in the system, as shown in Figure 4.33, and can view specific cases more closely when needed, as seen in Figure 4.34. This interface includes useful tools like search, filters, and sorting options to help quickly find particular violations. These features improve efficiency and ensure that no case is overlooked during the enforcement or appeal process.

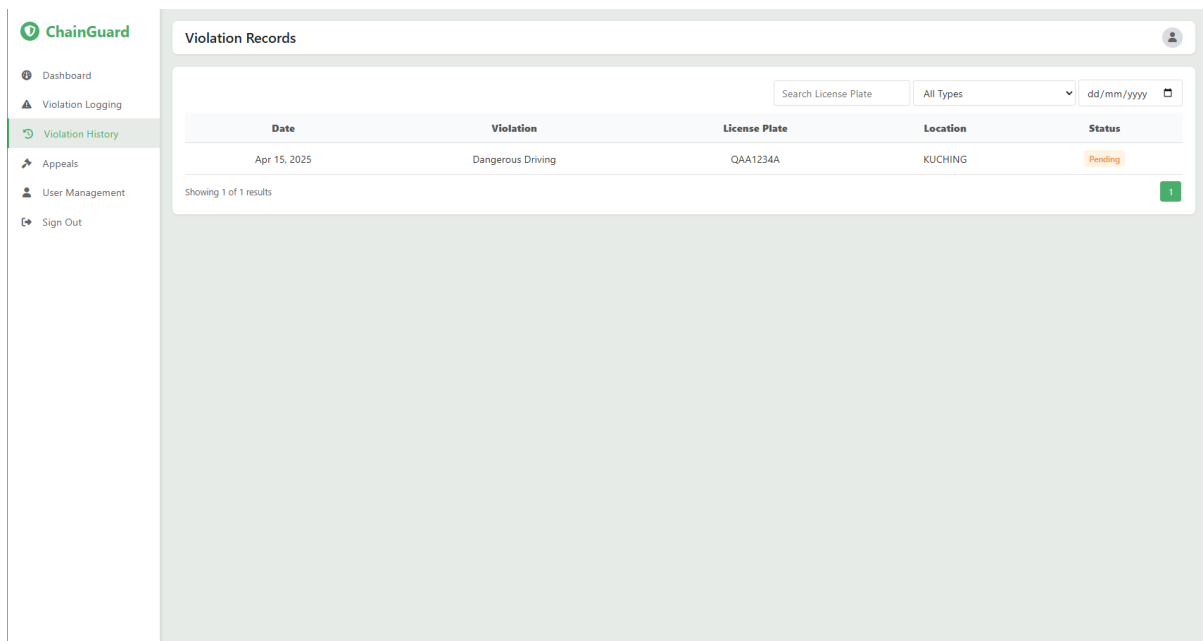


Figure 4.33. Violation Record Page

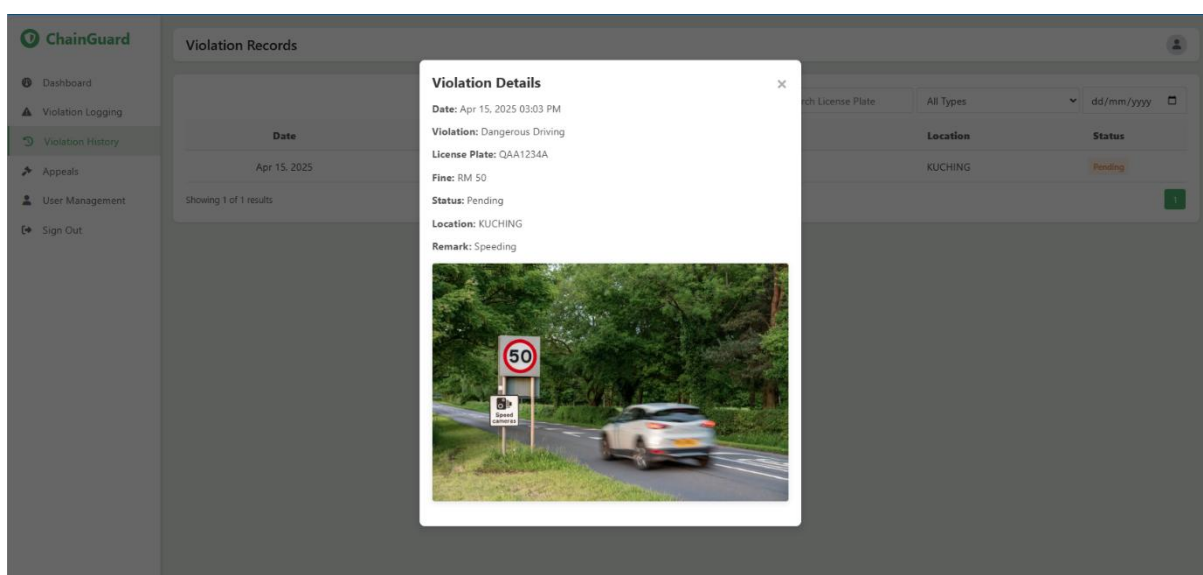


Figure 4.34. Details of Selected Violation

4.4.13. Appeal Management

The appeal management interface allows traffic authorities to view and handle all submitted appeals effectively. As shown in Figure 4.35, the appeals list displays all contested violations, while Figure 4.36 shows the full details of each appeal, including the driver's explanation and any supporting documents. Authorities can review the information provided, record their decisions, and update the appeal status directly within the system. All communication and decisions are automatically recorded to ensure transparency and accountability throughout the process. Additionally, authorities can sort and prioritize appeals based on factors such as the date of submission or the seriousness of the violation, which allow easier and more efficient way to manage the cases.

User	License	Violation ID	Date	Actions
Richard Jong Wei Ze rjwz.work01@gmail.com	QAA1234A	67ff565ba9caafdd72067692	Apr 20, 2025	

Showing 1 of 1 results

Figure 4.35. Appeal List Page

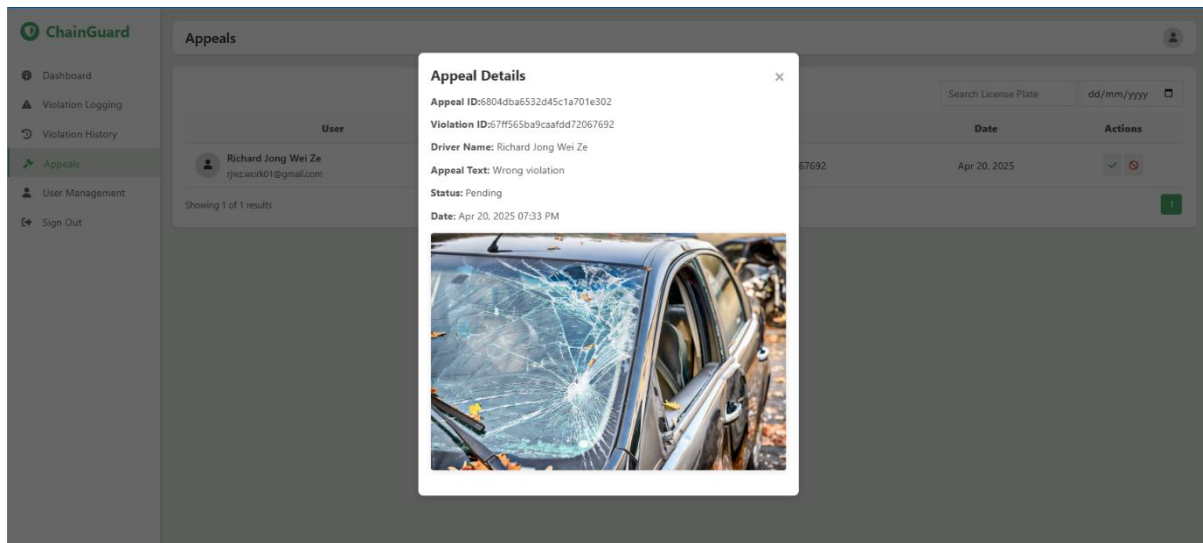


Figure 4.36. Details of Selected Appeal

4.4.14. User Account Management

Furthermore, the user account management page as shown in Figure 4.37 enables the traffic authorities to oversee all the registered accounts within the ChainGuard system. Each individual user profile can be examined by pressing the profile as displayed in Figure 4.38, which allows the traffic authorities to verify the identity information. When needed, they can update user information to correct any errors, as shown in Figure 4.39, ensuring that user data remains accurate and up to date.

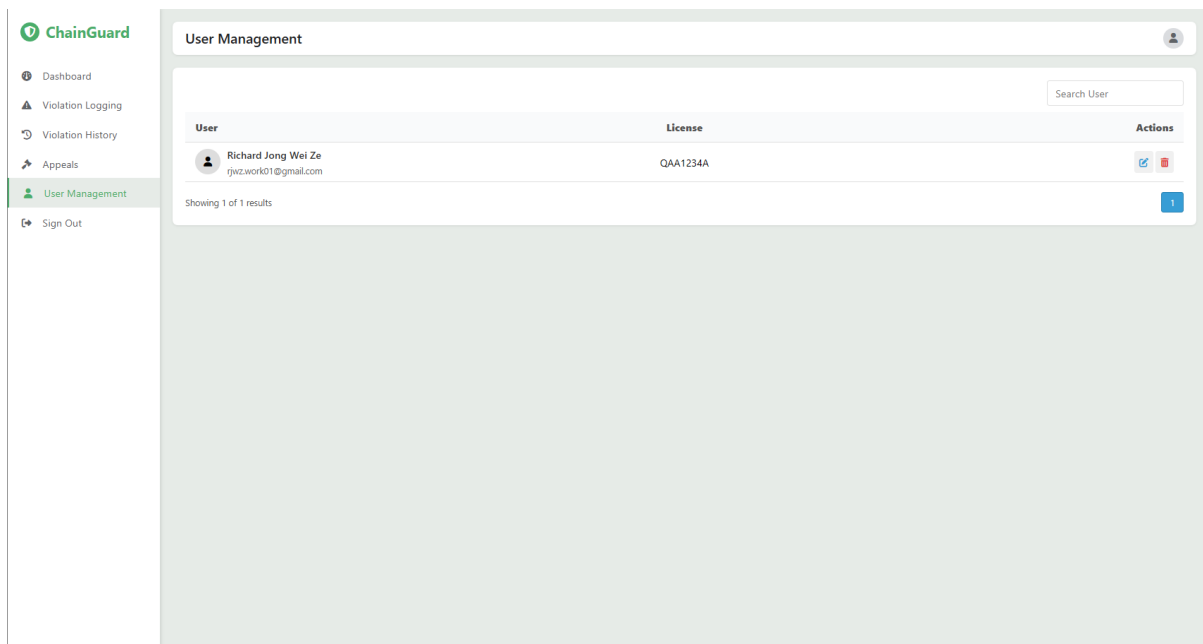


Figure 4.37. User Account Management Page.

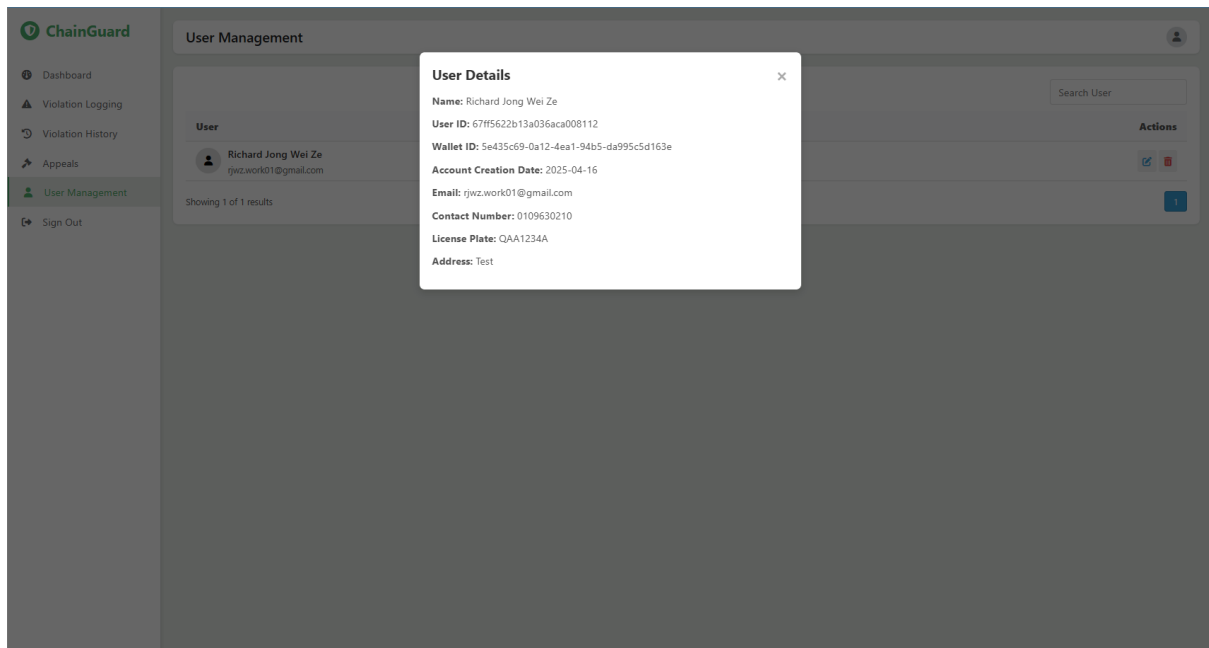


Figure 4.38. User Account Details.

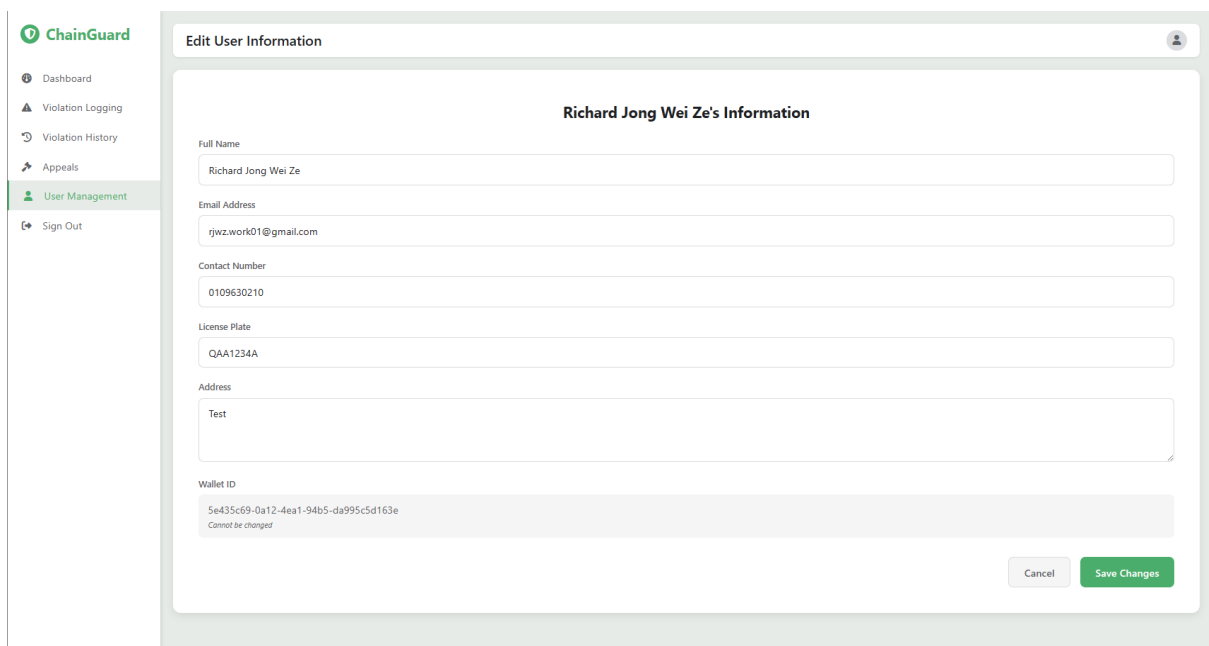


Figure 4.39. Edit User Account Page.

4.5. Summary

As a conclusion, this chapter provides a complete overview of the development and implementation of the proposed blockchain-based traffic violation logging and penalty management system, ChainGuard. It begins by introducing the system's core objectives and principles, then outlines the installation and configuration steps required to deploy the ChainGuard infrastructure. The chapter proceeds to describe the system architecture, detailing how Hyperledger Fabric components interact to ensure secure and transparent record-keeping. In the implementation section, fourteen key interfaces are presented which from user authentication and violation logging to payment processing and appeal management, while the integration with Hyperledger Explorer is emphasized for real-time transaction verification. Together, these sections demonstrate ChainGuard's effectiveness as a secure, efficient platform that benefits both traffic authorities and drivers.

CHAPTER 5: TESTING

5.1. Introduction

Testing is one of the critical phases in the development lifecycle of any system, especially in a blockchain-based system, where it deals with sensitive data such as traffic violation records. The testing strategy implemented for this Hyperledger Fabric-based traffic violation management system, ChainGuard ensures the reliability, security, and performance of all the system components across multiple layers of the architecture, which encompasses the Hyperledger Fabric blockchain network, Node.js Fabric SDK integration, PHP backend services with JavaScript frontend components, and MongoDB database operations. Given the distributed nature of permissioned blockchain systems and the complexity of multi-stakeholder interactions between drivers and traffic authorities, detailed testing protocols are established to validate individual component functionality and end-to-end system workflows.

This chapter consists of two main sections, which are functional testing, where all the functionalities of the system are tested to ensure the system operates effectively and efficiently, and non-functional testing, which focuses on evaluating the system's performance characteristics, reliability, usability, and overall quality attributes that determine how well the system performs under various conditions. The functional testing section encompasses unit testing to validate individual system components, ensuring that each function performs correctly in isolation before integration with other modules. The non-functional testing section includes reliability testing to assess system stability and fault tolerance, compatibility testing to ensure the system is widely compatible, as well as user acceptance testing to validate that the system meets end-user expectations and business requirements.

5.2. Functional Testing

Functional testing forms the foundation of the quality assurance process for the blockchain-based traffic violation logging and penalty management system, focusing on verifying that each system component performs its intended functions according to the specified requirements. This testing phase validates the business logic implementation systematically, blockchain transaction processing, and data consistency across the distributed ledger network. This functional testing approach covers both positive and negative test scenarios to ensure robust error handling and system resilience. Each functional module undergoes thorough testing to verify correct behaviour under normal operating conditions while also validating appropriate error responses when invalid inputs or exceptional conditions are encountered.

5.2.1. Unit Testing

Unit testing focuses on validating individual system functions in isolation, ensuring that each components performs correctly before integration with other system modules. The unit testing covers all the user-facing functionalities across both driver and traffic authority interfaces, as well as shared system functions that server multiple user roles.

5.2.1.1. Test Cases for Shared Function

For shared function testing, this section will focus on the sign in function for both driver and traffic authority. Table 5.1 highlights the test case for unit testing of Log In in “ChainGuard”.

Table 5.1. Unit Test Case for Log In.

ID	1	Module Name	Log In			
Objective	To test the log in functionality					
<i>Name</i>	<i>Richard Jong Wei Ze</i>		<i>Date</i>	<i>24th May 2025</i>	<i>Status</i>	<i>Pass</i>
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					

3	User must have a valid user account registered in the system.				
Test Scenario		User wants to sign in to an existing account.			
ID	Test Case	Input Data	Expected Result	Actual Result	Status
1	Navigate to the system through web browser.	Enter to user IP address in web browser.	The log in page is displayed	As Expected	Pass
2	Enter a valid traffic authority's email, and password.	- Email: admin@admin.com - Password: Admin_pwd	User is navigated to admin dashboard.	As Expected	Pass
3	Enter a valid driver's email, and password.	- Email: 81049@siswa.unimas.my - Password: QAA1234A_	User is navigated to driver's pending violation dashboard.	As Expected	Pass
4	Enter an invalid email with correct format	- Email: user@gmail.com - Password: QAA1234A_	Error message "Invalid email or password. Please try again" is showed.	As Expected	Pass
5	Enter an invalid email with incorrect format	- Email: user - Password: QAA1234A_	Error message "Invalid email or password. Please try again" is showed.	As Expected	Pass
6	Enter an invalid password with correct email	- Driver's Email: 81049@siswa.unimas.my - Driver's Password: ABCD1234 - Admin's Email: admin@admin.com - Admin's Password: Admin_pwd	Error message "Invalid email or password. Please try again" is showed.	As Expected	Pass

5.2.1.2. Test Cases for Drivers' Application

This section focuses on the driver side functionalities, including registration, pending violation viewing, payment processing, violation history viewing, appeal history viewing, appeal submission, driver profile viewing, and driver profile management. The testing of driver

functionalities ensures that all user interactions with the ChainGuard system are properly validated, secure, and provide accurate results through the blockchain network.

Table 5.2 presents the unit test cases for driver registration functionality, validating the account creation process and ensuring proper integration with the Hyperledger Fabric ledger. Table 5.3 demonstrates the testing results for pending violation viewing and payment processing, which are critical functions that allow drivers to access their outstanding violations and complete secure payment transactions. Table 5.4 details the unit test cases for viewing violation history, ensuring that drivers can accurately retrieve their complete violation records from the blockchain. Table 5.5 shows the testing of appeal viewing functionality, validating that drivers can properly access the status and details of their submitted appeals. Table 5.6 presents the unit test cases for appeal submission processes, confirming that drivers can successfully submit appeals with proper blockchain recording. Table 5.7 covers the testing of driver profile viewing and profile information update functionalities, ensuring secure and accurate personal data management. Finally, Table 5.8 provides the unit test cases for the sign-out functionality, validating proper session termination and security protocols.

Table 5.2. Unit Test Case for Driver Registration.

Table 3.2: Unit Test Case for Driver Registration.						
ID	2	Module Name	Log In			
Objective	To test the driver registration functionality					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
Test Scenario		User wants to register a driver account				
ID	Test Case	Input Data		Expected Result	Actual Result	Status
1	Navigate to the system through web browser.	Enter to user IP address in web browser and press “Register here”.		The driver registration page is displayed	As Expected	Pass
2	Enter valid information	• Full Name: Richard Jong Wei Ze		User is navigated to log in page and received an email	As Expected	Pass

	for the registration	- Email: 81049@siswa.unimas.my - Contact Number: 0109630210 - Vehicle License Plate: QAA1234A - Password: User1234@ - Address: Kuching, Sarawak.	notification of the success of account registration.		
3	Enter an invalid name with number	Full Name: Richard123	Error message "Please enter a valid name" is showed.	As Expected	Pass
4	Enter an invalid email	Email: 81049.siswa.unimas.my	Error message indicating the format of the email is wrong.	As Expected	Pass
5	Enter an invalid contact number more than 11 numbers	Contact Number: 010963021011	Error message "Please enter a valid Malaysian contact number" is showed.	As Expected	Pass
6	Enter an invalid contact number less than 9 numbers	Contact Number: 01096302	Error message "Please enter a valid Malaysian contact number" is showed.	As Expected	Pass
7	Enter an invalid contact number that is not start with 0	Contact Number: 1109630210	Error message "Please enter a valid Malaysian contact number" is showed.	As Expected	Pass
7	Enter an invalid contact number that is mix with letter	Contact Number: 010963d210	Error message "Please enter a valid Malaysian contact number" is showed.	As Expected	Pass
9	Enter an invalid license plate with less than 3 letters	Vehicle License Plate: QA1234A	Error message "Please enter a valid license plate" is showed.	As Expected	Pass
10	Enter an invalid license plate with disallowed letters (I and O)	Vehicle License Plate: QAI1234A	Error message "Please enter a valid license plate" is showed.	As Expected	Pass

11	Enter an invalid license plate with incorrect format	Vehicle License Plate: 1234QAA	Error message "Please enter a valid license plate" is showed.	As Expected	Pass
12	Enter a password with no special character.	Password: User1234	Error message "Password must be at least 8 characters long, contain at least 1 letter, and at least 1 special character." is showed.	As Expected	Pass
13	Enter a password with no letter	Password: 12345678@	Error message "Password must be at least 8 characters long, contain at least 1 letter, and at least 1 special character." is showed.	As Expected	Pass
14	Enter a password with less than 8 characters	Password: 1234	Error message "Password must be at least 8 characters long, contain at least 1 letter, and at least 1 special character." is showed.	As Expected	Pass
15	Enter the matching password	- Password: User1234@ - Confirm Password: User1234@	User is navigated to log in page and received an email notification of the success of account registration.	As Expected	Pass
16	Enter a not matching password	- Password: User1234@ - Confirm Password: User1234	Error message "Passwords do not match!" is showed.	As Expected	Pass

Table 5.3. Unit Test Case for Pending Violation Viewing and Payment Processing.

ID	3	Module Name	Pending Violation Viewing and Payment Processing			
Objective	To test the pending violation viewing and payment processing functionalities					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an account.					
4	User must have at least 1 pending violation recorded in the system.					
Test Scenario		User wants to view the pending violation and proceed to payment				
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	View the pending violation once login to the system	Log in to the system with valid credentials	The pending violation(s) is showed.	As Expected	Pass	
2	Payment Processing	Press the “Pay Now”	The payment dashboard is showed.	As Expected	Pass	
3	Enter all the required information with Visa card number	- Email: 81049@siswa.unimas.my - Card Information:4242 4242 4242 4242 - Date: 5/55 - CVC: 555 - Cardholder Name: Richard	The payment is processed and navigate the user back to the pending violation dashboard.	As Expected	Pass	
4	Enter all the required information with declined card number	- Email: 81049@siswa.unimas.my - Card Information: 4000 0000 0000 0002 - Date: 5/55 - CVC: 555 - Cardholder Name: Richard	The payment failed to process and notified the user the payment has been declined.	As Expected	Pass	
5	Leave any/all the credential empty	Leave any/all the credential empty	Error message is showed indicating the missing field(s) is required.	As Expected	Pass	

Table 5.4. Unit Test Case for Viewing Violation History.

ID	4	Module Name	Violation History Viewing			
Objective	To test the viewing violation history functionality					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an account.					
4	User must have at least 1 violation recorded in the system.					
Test Scenario		User wants to view the violation history.				
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	View the violation history	Press the “Violation History” at the side navigation bar.	The violation history is showed	As Expected	Pass	
2	View the details of selected violation	Select a violation record and press it.	The violation details pop-up is showed.	As Expected	Pass	
3	Copy the violation ID	Press the copy icon besides the violation ID.	A message “Violation ID copied to clipboard” is showed	As Expected	Pass	
4	Search violation records with date filter	Select a date from the date picker.	The violation record(s) within the selected date is showed.	As Expected	Pass	
5	Choose a date with no violations	Select a date for which no violation records exist.	No violation record will be shown.	As Expected	Pass	
6	Filter violation records by type	Query violation records based on the type of violation	The violation record(s) with the selected type is showed.	As Expected	Pass	
7	Filter violation records to show that no violations of the selected type exist.	Select a violation type with no record.	No violation record will be shown.	As Expected	Pass	

Table 5.5. Unit Test Case for Viewing Appeal.

ID	5	Module Name	Appeal Viewing			
Objective	To test the viewing appeal functionality					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an account.					
4	User must have at least 1 appeal recorded in the system.					
Test Scenario		User wants to view the appeal.				
ID	Test Case		Input Data	Expected Result	Actual Result	Status
1	View the appeal list		Press the “Appeals” at the side navigation bar.	A list of appeals is showed	As Expected	Pass
2	View the details of selected appeal		Select an appeal and press it.	An appeal details pop-up is showed.	As Expected	Pass
3	Search appeal with date filter		Select a date from the date picker.	The appeal within the selected date is showed.	As Expected	Pass
4	Choose a date with no appeal		Select a date for which no appeal exists.	No appeal will be shown.	As Expected	Pass
5	Filter appeals by status		Query appeals based on the status of appeal (Pending, Approved, Rejected)	The appeal(s) with the selected status is showed.	As Expected	Pass
6	Filter appeals to show that no violations of the selected status.		Select the appeal status with no record.	No appeal will be shown.	As Expected	Pass

Table 5.6. Unit Test Case for Appeal Submission.

ID	6	Module Name	Appeal Submission				
Objective		To test the submission of appeal functionality					
Name		Richard Jong Wei Ze	Date		24 th May 2025	Status	Pass
No	Prerequisites						
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.						
2	User must have a stable internet connection.						
3	User must have an account.						
4	User must have at least 1 violation recorded in the system.						
Test Scenario		User wants to submit an appeal.					
ID	Test Case		Input Data		Expected Result	Actual Result	Status

1	View the appeal submission page	Press the “Make Appeal” button at the appeal page	An appeal application page is showed.	As Expected	Pass
2	Enter all the required fields	- Violation ID: 6833c8db16e9472b8d0fcce2 - Appeal Reason: Testing1 - Supporting Evidence: image.jpg	An appeal is created successfully with a green notification message and a new block is created on the blockchain ledger.	As Expected	Pass
3	Leave any/all the field(s) empty	Leave empty for any/all the required field(s)	Error message is showed, notifying the user the information is missing.	As Expected	Pass
4	Enter a violation ID with appeal created before and is in the state of pending.	- Violation ID: 6833c8db16e9472b8d0fcce2 - Appeal Reason: Testing2 - Supporting Evidence: image.jpg	Error message is showed, notifying the user the appeal for the selected violation has been created before and waiting for action.	As Expected	Pass
5	Enter a violation ID with appeal created and approved before.	- Violation ID: 6833c8db16e9472b8d0fcce2 - Appeal Reason: Testing3 - Supporting Evidence: image.jpg	Error message is showed, notifying the user the appeal for the selected violation has been approved.	As Expected	Pass
6	Enter a violation ID with appeal created and rejected before.	- Violation ID: 6833c8db16e9472b8d0fcce2 - Appeal Reason: Testing4 - Supporting Evidence: image.jpg	An appeal is created successfully with a green notification message.	As Expected	Pass

Table 5.7. Unit Test Case for Viewing Driver Profile and Update Profile Information.

ID	7	Module Name	Profile Viewing and Management			
Objective		To test the profile viewing and editing functionalities				
Name		Richard Jong Wei Ze	Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an account.					
Test Scenario		User wants to view and edit profile.				
ID	Test Case	Input Data		Expected Result	Actual Result	Status
1	View the profile page	Press the “Profile” button at the side navigation bar.		A user profile page is showed.	As Expected	Pass
2	View Edit Profile Page	Press the “Edit Profile” button at the bottom of the page.		An edit profile page is showed.	As Expected	Pass
3	Update any personal information	Full Name: Richard Jong		A green notification bar with a message “Profile updated successfully!” is showed.	As Expected	Pass
4	No changes to any personal information with “Save Changes” button press	No changes to any field and pressing the “Save Changes” button.		A warning message “No changes were made to your profile.” is showed.	As Expected	Pass
5	Cancel the profile editing	Press the “Cancel” button at the bottom of the page.		User is navigated to the “Profile” page.	As Expected	Pass

Table 5.8. Unit Test Case for Driver Sign Out.

ID	8	Module Name	Driver Sign Out			
Objective	To test the sign out functionality on driver side					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					

3	User must have an account.				
Test Scenario		User wants to sign out of the system.			
ID	Test Case	Input Data	Expected Result	Actual Result	Status
1	Sign out of the system	Press the “Sign Out” button at the side navigation bar.	User is navigated to the login page.	As Expected	Pass
2	Navigate back to the system after sign out	Press the back icon in the browser to navigate back to the system	User is navigated to the login page.	As Expected	Pass

5.2.1.3. Test Cases for Traffic Authorities’ Application

This section focuses on the testing of traffic authority functionalities, including admin registration, statistics and reporting through the main dashboard, violation logging, violation record management, appeal management, and user account management. The traffic authority application serves as the administrative backbone of the ChainGuard system, providing authorized personnel with the tools necessary to efficiently manage traffic violations, process appeals, and maintain system oversight.

Table 5.9 presents the unit test cases for admin registration functionality, ensuring that traffic authority personnel can securely create administrative accounts with appropriate role-based permissions. Table 5.10 demonstrates the testing results for additional admin registration scenarios, validating comprehensive coverage of administrative account creation processes. Table 5.11 details the unit test cases for logging violation records, confirming that traffic officers can accurately record new violations with proper blockchain integration and data validation. Table 5.12 shows the testing of admin violation record management functionality, ensuring that authorized personnel can efficiently update, modify, and manage existing violation records while maintaining audit trails. Table 5.13 presents the unit test cases for appeal management processes, validating that traffic authorities can properly review, process, and respond to driver appeals with appropriate status updates. Table 5.14 covers the testing of

user management functionalities, ensuring that administrators can effectively manage driver accounts, including account oversight and administrative capabilities while adhering to data protection requirements. Finally, Table 5.15 provides the unit test cases for admin sign-out functionality, validating proper session termination and security protocols for administrative users.

Table 5.9. Unit Test Case for Admin Registration.

ID	9	Module Name	Admin Registration					
Objective		To test the admin registration functionality						
Name		Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass	
No	Prerequisites							
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.							
2	User must have a stable internet connection.							
Test Scenario		User wants to register as an admin of the system.						
ID	Test Case		Input Data		Expected Result		Actual Result	Status
1	Navigate to the registration page through web browser.		Enter IP/php/admin_registration.php in web browser		The admin registration page is displayed		As Expected	Pass
2	Enter valid information for the registration		- Full Name: admin - Email: admin@admin.com - Contact Number: 0109630210 - Password: Admin_pwd		User is navigated to log in page		As Expected	Pass
3	Enter an invalid name with number		Full Name: admin123		Error message “Please enter a valid name” is showed.		As Expected	Pass
4	Enter an invalid email		Email: admin.admin.com		Error message indicating the format of the email is wrong.		As Expected	Pass
5	Enter an invalid contact number more than 11 numbers		Contact Number: 010963021011		Error message “Please enter a valid Malaysian contact		As Expected	Pass

			number” is showed.		
6	Enter an invalid contact number less than 9 numbers	Contact Number: 01096302	Error message “Please enter a valid Malaysian contact number” is showed.	As Expected	Pass
7	Enter an invalid contact number that is not start with 0	Contact Number: 1109630210	Error message “Please enter a valid Malaysian contact number” is showed.	As Expected	Pass
7	Enter an invalid contact number that is mix with letter	Contact Number: 010963d210	Error message “Please enter a valid Malaysian contact number” is showed.	As Expected	Pass
8	Enter a password with no special character.	Password: admin1234	Error message “Password must be at least 8 characters long, contain at least 1 letter, and at least 1 special character.” is showed.	As Expected	Pass
9	Enter a password with no letter	Password: 12345678@	Error message “Password must be at least 8 characters long, contain at least 1 letter, and at least 1 special character.” is showed.	As Expected	Pass
10	Enter a password with less	Password: 1234	Error message “Password must be at	As Expected	Pass

	than 8 characters		least 8 characters long, contain at least 1 letter, and at least 1 special character.” is showed.		
11	Enter the matching password	- Password: Admin_pwd - Confirm Password: Admin_pwd	User is navigated to log in page and received an email notification of the success of account registration.	As Expected	Pass
12	Enter a not matching password	- Password: Admin_pwd - Confirm Password: admin_Pwd	Error message “Passwords do not match!” is showed.	As Expected	Pass

Table 5.10. Unit Test Case for Admin Registration.

ID	10	Module Name	Statistics and Reporting			
Objective		To test the statistics and reporting functionalities				
Name		Richard Jong Wei Ze	Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an admin account.					
Test Scenario		User wants to view the real-time statistics and reporting.				
ID	Test Case	Input Data		Expected Result	Actual Result	Status
1	Navigate to the main dashboard.	Login with admin’s credentials		The statistics dashboard is displayed	As Expected	Pass
2	Enter the year for monthly trend graph.	Select year from the year picker		The monthly trend graph will change based on the year selected.	As Expected	Pass
3	Enter the month for daily trend graph.	Select month from the month picker		The daily trend graph will change based on the	As Expected	Pass

			month selected.		
4	Print the statistics data for reporting.	Press the “Export” button at top right of the page.	A pop out is showed with the statistics arranged neatly and asking for printing or cancellation.	As Expected	Pass

Table 5.11. Unit Test Case for Logging Violation Record.

ID	11	Module Name	Violation Logging					
Objective		To test the violation logging functionality						
Name		Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass	
No	Prerequisites							
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.							
2	User must have a stable internet connection.							
3	User must have an admin account.							
Test Scenario		User wants to log new violation record.						
ID	Test Case		Input Data		Expected Result		Actual Result	Status
1	Navigate to the violation logging page.		Press the “Violation Logging” button at the side navigation bar.		The violation logging page is displayed		As Expected	Pass
2	Enter all the valid information for logging.		- Violation Type: Poor Vehicle Condition - Location: FCSIT, UNIMAS - Time: 24/05/2025 07:30PM - License Plate Number: QAA1234A - Penalty Amount: 300 - Remark: Testing - Violation Image: testing.jpg		A green notification bar is showed with “The violation is logged successfully.” message, and a block is created on the ledger.		As Expected	Pass
3	Leave any/all the field empty.		Leave any/all field empty.		Error message “Location is required. Time is required. License plate number is required. Penalty		As Expected	Pass

			amount is required. Violation image is required.” is showed.		
4	Logging for driver that has not register to the system.	- Violation Type: Poor Vehicle Condition - Location: FCSIT, UNIMAS - Time: 24/05/2025 07:30PM - License Plate Number: QAA4567B - Penalty Amount: 300 - Remark: Testing2 - Violation Image: testing2.jpg	A notification message is showed indicating the driver has not register to the system and stored the logging as off-chain data on MongoDB	As Expected	Pass
5	Refresh the violation logging function.	Press the “Refresh Violations” button in the bottom right of the page.	A message “Violation refreshed successfully.” is showed, indicating all the data is pushed to the ledger, and new block(s) is created based on the number of data.	As Expected	Pass
5	Cancel the violation logging.	Press the “Cancel” button in the bottom right of the page.	All the filled data is cleared and refreshed.	As Expected	Pass

Table 5.12. Unit Test Case for Admin Violation Record Management.

Table 5.12: Unit Test Case for Admin Violation Record Management.						
ID	12	Module Name	Violation Management			
Objective	To test the violation management functionality					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an admin account.					
4	There must has at least 1 record of violation record logged in the system.					

Test Scenario		User wants to manage and search for violation records.			
ID	Test Case	Input Data	Expected Result	Actual Result	Status
1	View the violation list	Press the “Violation History” at the side navigation bar.	A list of violation record is showed	As Expected	Pass
2	View the details of selected violation record	Select a violation record and press it.	A violation details pop-up is showed.	As Expected	Pass
3	Search violation with date filter	Select a date from the date picker.	The violation record(s) within the selected date is showed.	As Expected	Pass
4	Choose a date with no violation record	Select a date for which no violation record exists.	No violation record will be shown.	As Expected	Pass
5	Filter violation record by types of violation	Query violation records based on the types of violation (All Types, Dangerous Driving, Poor Vehicle Condition, Parking Violations, Licensing and Documentation, Safety Regulations)	The violation record(s) with the selected type is showed.	As Expected	Pass
6	Filter violation record by license plate	Query violation records with license plate number	The violation record(s) with the corresponding license plate number is showed.	As Expected	Pass
7	Filter violation record by types of violation (No Data for Selected Type)	Query violation records based on the types of violation (All Types, Dangerous Driving, Poor Vehicle Condition, Parking Violations, Licensing and Documentation, Safety Regulations)	No violation record will be shown.	As Expected	Pass
8	Filter violation record by license plate (No Record Found for License Plate)	Query violation records with license plate number	No violation record will be shown.	As Expected	Pass

Table 5.13. Unit Test Case for Appeal Management.

ID	13	Module Name	Appeal Management			
Objective	To test the appeal management functionality					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an admin account.					
4	There must has at least 1 record of violation record and appeal logged in the system.					
Test Scenario		User wants to manage appeal.				
ID	Test Case	Input Data	Expected Result	Actual Result	Status	
1	View the appeal list	Press the “Appeal” at the side navigation bar.	A list of violation record is showed	As Expected	Pass	
2	View the details of selected appeal	Select an appeal and press it.	An appeal details pop-up is showed.	As Expected	Pass	
3	Search appeal with date filter	Select a date from the date picker.	The appeal(s) within the selected date is showed.	As Expected	Pass	
4	Choose a date with no appeal	Select a date for which no appeal exists.	No appeal will be shown.	As Expected	Pass	
5	Filter appeals by license plate	Query appeals with license plate number	The appeal(s) with the corresponding license plate number is showed.	As Expected	Pass	
6	Filter appeals by license plate (No Record Found for License Plate)	Query appeals with license plate number	No appeal will be shown.	As Expected	Pass	
7	Approve appeal	Press the green tick icon to approve the appeal	Appeal’s action is changed to “Approved”, and a new block is created in the ledger.	As Expected	Pass	
8	Reject appeal	Press the red cross icon to reject the appeal	Appeal’s action is changed to “Rejected”, and a new block is	As Expected	Pass	

			created in the ledger.		
--	--	--	------------------------	--	--

Table 5.14. Unit Test Case for User Management.

ID	14	Module Name	User Management			
Objective		To test the user management functionality				
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an admin account.					
4	There must has at least 1 driver account in the system.					
Test Scenario		User wants to manage user account.				
ID	Test Case	Input Data		Expected Result	Actual Result	Status
1	View the user list	Press the “User Management” at the side navigation bar.		A list of user is showed	As Expected	Pass
2	View the details of selected user	Select a user and press it.		A user details pop-up is showed.	As Expected	Pass
3	Search user with name, email and license plate number	Enter name, email or license plate number in the search bar.		The related user is showed with the filter.	As Expected	Pass
4	Delete user account	Press the red bin icon under the actions section		Everything related to the user is deleted, and new block is created on the ledger.	As Expected	Pass
5	Edit user profile	Press the blue edit icon under the actions section		Edit user profile page is showed.	As Expected	Pass
6	Update any personal information	Full Name: Richard Jong Wei Ze		A green notification bar with a message “Profile updated successfully!” is showed.	As Expected	Pass
7	No changes to any personal information with “Save	No changes to any field and pressing the “Save Changes” button.		A warning message “No changes were made to the	As Expected	Pass

	Changes” button press		selected profile.” is showed.		
8	Cancel the profile editing	Press the “Cancel” button at the bottom of the page.	User is navigated to the “User Management” page.	As Expected	Pass

Table 5.15. Unit Test Case for Admin Sign Out.

ID	15	Module Name	Admin Sign Out			
Objective		To test the sign out functionality on admin side				
Name		Richard Jong Wei Ze	Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	User must have a stable internet connection.					
3	User must have an admin account.					
Test Scenario		User wants to sign out of the system.				
ID	Test Case	Input Data		Expected Result	Actual Result	Status
1	Sign out of the system	Press the “Sign Out” button at the side navigation bar.		User is navigated to the login page.	As Expected	Pass
2	Navigate back to the system after sign out	Press the back icon in the browser to navigate back to the system		User is navigated to the login page.	As Expected	Pass

5.3. Non-Functional Testing

Non-functional testing plays an essential role in evaluating the ChainGuard system’s performance characteristics, reliability, and overall quality attributes beyond basic functionality. While functional testing validates what the system does, non-functional testing focuses on how well the system performs under various conditions, which ensure that the blockchain-based traffic violation logging and penalty management system can operate effectively in real-world deployment scenarios. In addition, the non-functional testing approach covers multiple critical aspects including reliability and availability testing to access the system stability under continuous operation and failure conditions, compatibility testing to ensure

cross-platform functionality across different system architectures and operating environments, and usability testing to validate user experience and interface effectiveness.

5.3.1. Reliability and Availability Testing

Reliability and availability testing focuses on evaluating the ChainGuard system's ability to maintain consistent performance and remain operational under various stress conditions and failure scenarios. This testing category is particularly critical for blockchain-based systems where network failures, resource constraints, and connectivity issues can significantly impact system functionality and data integrity. Table 5.16 presents the test cases for reliability and availability testing.

Table 5.16. Test Case for Reliability and Availability Testing.

ID	16	Application Name	ChainGuard			
Objective	To test the reliability and availability of the system					
Name	Richard Jong Wei Ze		Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	The device is running on Debian-based Linux operating system.					
3	The device is connected to stable internet connection, preferably ethernet connection.					
Test Scenario		Verify the reliability and availability of the system under various conditions.				
ID	Test Case	Input Data		Expected Result	Actual Result	Status
1	Continuous Testing	Continuous usage of the system with periodic API interactions with monitoring via Grafana.		Consistent uptime and runs smoothly without any issue.	As Expected	Pass
2	API Failure Handling	Simulate interrupt upload violation records to ledger		The interrupted data is temporarily stored as off-chain data in MongoDB. Admin can refresh the violation records to reupload the interrupted data to the ledger.	As Expected	Pass
3	Network Connectivity Loss	Disconnect the system from the internet connection		The system auto-recovered on reconnect once the connection is re-established	As Expected	Pass

4	Resource Utilization	Track the CPU and memory usage with Prometheus and Grafana. - Average CPU usage: 5 to 6% - Average Memory Usage: 20% out of 8GB memory	Low CPU and memory usage for the whole system.	As Expected	Pass
5	Temperature Tracking	Continuous monitoring of the temperature with command script Average temperature: $56^{\circ}\text{C} \pm 2^{\circ}\text{C}$	The average temperature throughout continuous operation, with no thermal throttling occurring	As Expected	Pass
6	Error Handling	Simulate with malformed data and unexpected input.	Error message is showed, blocking the action from further operation	As Expected	Pass
7	Database Connectivity	Temporarily block MongoDB service	Error message is showed, indicating the database is not available, and blocking user's action.	As Expected	Pass
8	User Input Stress Test	Simulate excessive submissions.	The system is able to handle large data submission without crashing.	As Expected	Pass
9	Blockchain Ledger Storage	Upload data to the system and verify the transaction logs via Hyperledger Explorer/Grafana.	The statistics in Hyperledger Explorer and Grafana is updated if there is new transaction occurs.	As Expected	Pass
9	Multiple Instance in Single Machine	Multiple browser tab open with the system and attempt to sign in to each opened system	User is logged out when new sign in is detected in the same machine.	As Expected	Pass

5.3.2. Compatibility Testing

Compatibility testing ensures that the ChainGuard system operates consistently across different hardware architectures, operating systems, and web browsers, providing users with a reliable experience regardless of their technical environment. This testing approach is essential

for ensuring broad accessibility and deployment flexibility in diverse traffic authority infrastructures. Table 5.17 tabulates the test cases for compatibility testing.

Table 5.17. Test Case for Compatibility Testing.

ID	17	Application Name	ChainGuard			
Objective		To test the compatibility of the system				
Name		Richard Jong Wei Ze	Date	24 th May 2025	Status	Pass
No	Prerequisites					
1	The requirements of the blockchain network, API layer, and UI are installed and configured correctly.					
2	The device is connected to stable internet connection, preferably ethernet connection.					
Test Scenario		Verify the compatibility of the system under various scenario.				
ID	Test Case	Input Data	Expected Result		Actual Result	Status
1	System Architecture	Run system on ARM64 (Raspberry Pi 5) and x86_64 (Laptop)	All services work as expected via Docker, beside Hyperledger Explorer on Raspberry Pi due to depreciated package.		As Expected	Pass
2	Operating System	Deploy system on Ubuntu 22.04 LTS and Raspberry Pi OS Bookworm	No compatibility issues; system runs on both OS types		As Expected	Pass
3	Web Browser Compatibility	Access system using Chrome, Microsoft Edge, and Brave browsers	UI renders correctly and functions as expected on all browsers		As Expected	Pass

5.3.3. User Acceptance Testing

Usability testing evaluates the user experience quality and interface effectiveness of the ChainGuard system from both driver and traffic authority perspectives. This testing approach focuses on assessing the intuitiveness of system interfaces, the efficiency of user workflows, and the overall satisfaction of end users interacting with the blockchain-based traffic violation management system.

The usability testing methodology involves systematic evaluation through structured user feedback collection with Google Form, which showed in Appendix C: Sample Feedback

Form, focusing on task completion ease, system navigation efficiency, and overall user satisfaction metrics. The testing covers both role-specific task evaluations and general system usability assessments to ensure comprehensive coverage of user experience factors. The survey form consists of three sections, which are Section 1: User Demographics, Section 2: Admin/User Task Completion Feedback, Section 3: Overall System Usability. Based on Figure 5.1, a total of 10 respondents participated in the usability testing, comprising 3 administrative users (traffic authorities) and 7 regular users (drivers), providing a balanced perspective on system usability across different user roles.

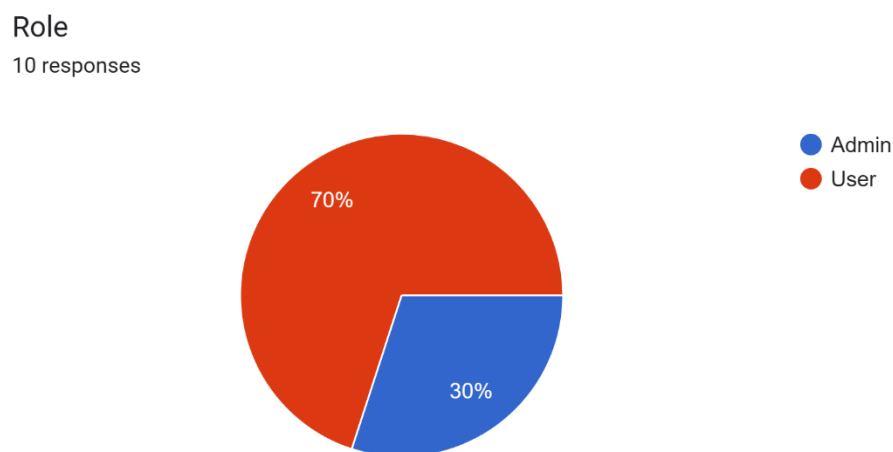


Figure 5.1. Number of Participants and Role Assigned.

In addition, Figure 5.2 and Figure 5.3 presents the usability testing result for the ChainGuard blockchain-based traffic violation logging and penalty management system, covering feedback from administrative users and regular drivers across task-specific and overall system evaluations. The admin task completion feedback shows high user satisfaction, with all core administrative functions receiving mostly "Strongly Agree" responses. Report generation, appeal revelation, and traffic violation creation tasks received positive feedback from all 3 administrative respondents, showing good task completion ease and workflow efficiency. The account registration and dashboard login functions showed mixed responses,

with some "Agree" ratings indicating that minor improvements could improve the initial user experience for administrators.

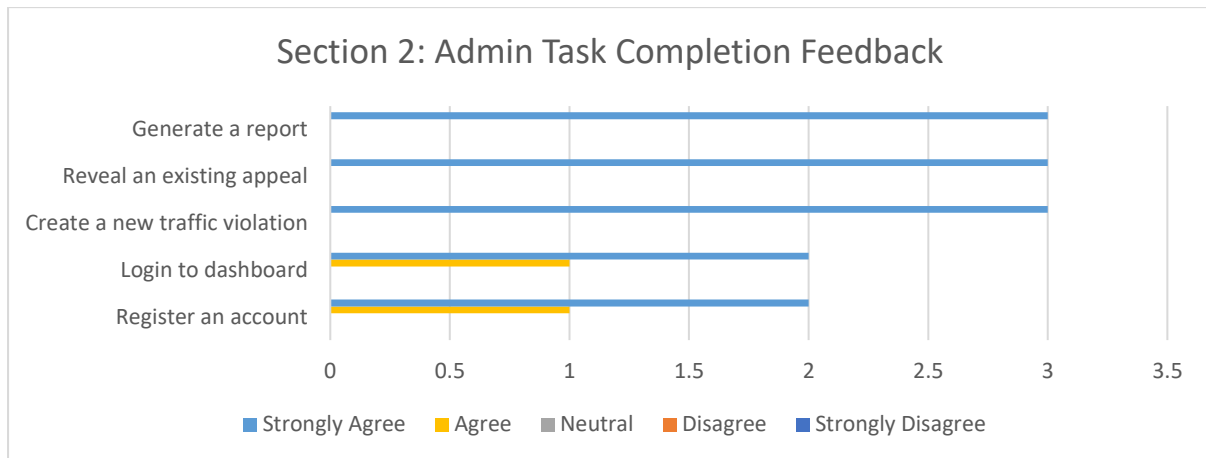


Figure 5.2. Admin Task Completion Feedback Analysis.

The user task completion feedback shows positive results from the 7 driver participants, with all primary user functions receiving strong approval ratings. Appeal submission, violation viewing, portal login, and account registration tasks received consistent "Strongly Agree" responses from all participants, indicating clear interface design and efficient task completion workflows for end users. The penalty payment function received both "Strongly Agree" and "Agree" responses, suggesting that while the functionality works well, there may be opportunities to improve the payment process.

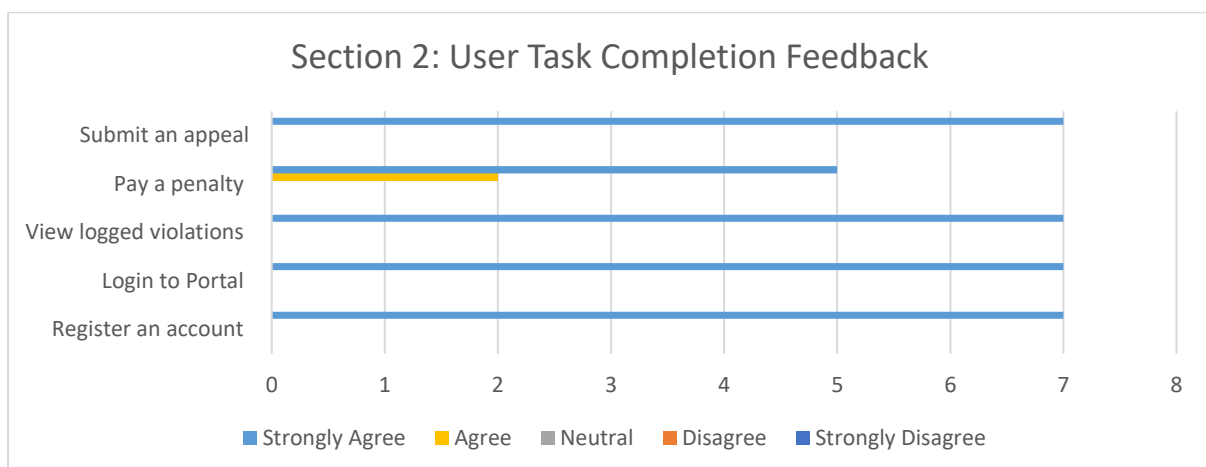


Figure 5.3. User Task Completion Feedback Analysis.

The overall system usability assessment as shown in Figure 5.4 supports the positive task-specific feedback, with all 10 respondents showing high satisfaction levels across key usability measures. System satisfaction, learnability, functional integration, ease of use, and frequency of intended use all received mostly "Strongly Agree" responses, with minimal "Agree" ratings observed only in the learning curve and usage frequency categories. These results confirm the effectiveness of the ChainGuard system's user-centric design approach and demonstrate successful achievement of the usability testing objectives. Both administrative and regular users can efficiently navigate and use the blockchain-based traffic violation management system with high levels of satisfaction and task completion success.

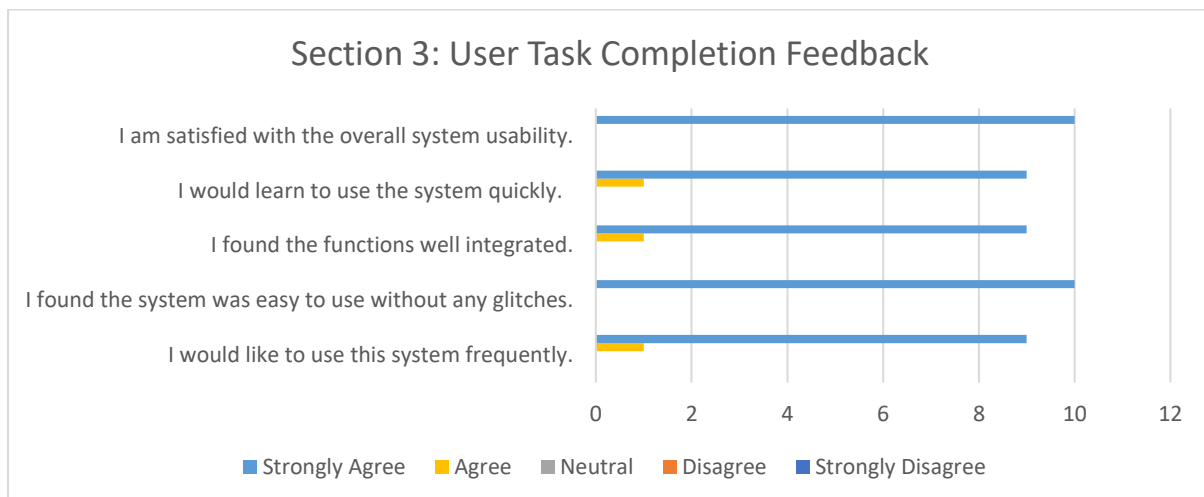


Figure 5.4. Overall System Usability Analysis.

5.4. Discussion on Results

The testing results presented in this chapter demonstrate the ChainGuard system successfully meets its design objectives and functional requirements across all testing categories. The functional testing results indicate robust implementation of core system functionalities, with all unit test cases achieving successful completion rates. This demonstrates the reliability of both driver-facing and traffic authority applications.

The unit testing results covers in-depth validation of driver functionalities and administrative operations. Tables 5.1 through 5.8 demonstrate successful implementation of all

driver-related features, including login authentication, registration processes, violation management, payment processing, appeal submission and viewing, profile management, and secure sign-out procedures. The testing confirms proper integration with the Hyperledger Fabric blockchain ledger and smart contracts across all driver interactions. Tables 5.9 through 5.15 validate the administrative functionalities, showing effective implementation of admin registration, violation record logging and management, appeal processing, user management capabilities, and administrative session handling. All administrative operations demonstrate proper role-based access controls through smart contracts and maintain appropriate audit trails.

The non-functional testing results provide valuable insights into system performance characteristics under various operational conditions. Table 5.16 presents the reliability and availability testing outcomes, demonstrating the system's resilience to network disruptions, API failures, and extended operational periods. The testing confirms consistent performance metrics and validates the system's ability to maintain operational stability under stress conditions. Table 5.17 demonstrates compatibility testing results, showing the system's versatility across different deployment environments with successful operation validated on both ARM64 and x86 architectures. Cross-platform compatibility across Ubuntu and Raspberry Pi OS environments ensures broad deployment applicability, while browser compatibility testing confirms consistent user experience across Chrome, Microsoft Edge, and Brave browsers.

The usability testing feedback from 10 respondents provides positive results regarding user experience quality. Administrative users expressed high satisfaction with core traffic authority functions, particularly in violation logging and report generation capabilities. Driver users demonstrated positive feedback regarding system navigation and task completion ease, with payment processing and appeal submission receiving favourable ratings. The overall

system usability scores indicate strong user acceptance potential, with respondents expressing willingness to use the system frequently and satisfaction with the integrated functionality approach.

Some areas for improvement emerged from the testing process. For instances, network connectivity loss scenarios revealed opportunities for enhanced offline capability implementation, while stress testing identified potential optimization areas for concurrent user handling. The usability feedback suggested minor interface refinements that could further improve user experience, particularly in the user payment process and administrative dashboard navigation flow.

5.5. Summary

This chapter focuses on the detailed testing methodology and results analysis for the developed system, ChainGuard, a blockchain-based traffic violation logging and penalty management system. The testing approach consists of two sections, which are functional and non-functional testing, ensuring thorough validation of the system capabilities across multiple operational scenarios. The functional testing verified all core features for both driver and authority roles, with all test cases passing successfully. Non-functional testing assessed performance, compatibility across different platforms, reliability, availability, and usability through user feedback. Overall, the results confirm that ChainGuard meets its intended functional requirements and demonstrates strong performance, security, and user experience, indicating its potential for further development and future deployment.

CHAPTER 6: CONCLUSION AND FUTURE WORKS

6.1. Introduction

Chapter 6 is the final chapter which presents the conclusion and evaluation of the developed decentralized traffic violation logging system, analysing the achievements against the initially proposed objectives and identifying areas for future improvement. The discussion covers the successful implementation of blockchain-based system which utilized smart contract for automated penalty issuance and the development of a transparent platform accessible to all stakeholders including traffic authorities and drivers. Additionally, this chapter addresses the limitations and constraints encountered during the development process, providing insights into the challenges faced when working with Hyperledger Fabric technology and the complexities of establishing a permissioned blockchain network for traffic management applications. The final section of this chapter outlines potential future work, highlighting opportunities to enhance the system's scalability, usability, and real-world applicability based on the limitations identified.

6.2. Objectives and Achievements

As outlined in Chapter 1, this project aimed to develop a comprehensive blockchain-based traffic violation logging and penalty management system. This section evaluates the achievement of the stated objectives and discusses the successful implementation of key system components. The primary objective of this project was to develop a decentralized system for securely logging and storing traffic violation data utilizing blockchain technology. The specific objectives established for this project were:

1. To implement smart contracts for automated and consistent penalty issuance -

Creating an autonomous system where, once a violation is logged by traffic authorities, penalties are issued automatically through predefined smart contract logic, ensuring uniform enforcement and minimizing the risk of human error.

2. To develop a transparent and accessible platform for all stakeholders - Building a

comprehensive system that provides traffic authorities and drivers with transparent access to violation records, penalty information, and system operations.

Both objectives have been successfully achieved through comprehensive system development and implementation. The following Table 6.1 demonstrates how each objective was accomplished through specific technical implementations and system components.

Table 6.1. Analysis of Achievements.

Objective	Achievement Status	Implementation Details
Smart Contract for Automated Penalty Issuance	Successfully Achieved	• Hyperledger Fabric blockchain network established successfully • Custom chaincode (smart contracts) developed and deployed • Automated violation logging and implemented with pre-defined chaincode logic • Consistent penalty enforcement mechanisms integrated • Smart contract functions tested for reliability and accuracy
Transparent and Accessible Platform Development	Successfully Achieved	• Consistent penalty enforcement mechanisms integrated • Hyperledger Fabric SDK implemented as API layer • Seamless integration between web application and blockchain network • User-friendly interfaces developed for both authorities and drivers • Role-based access control implemented for different stakeholder types • Real-time data access and query capabilities established

Discussion on Achievements

The successful implementation of both objectives demonstrates the practical viability of blockchain technology in traffic management systems. The first objective, implementing smart contracts for automated penalty issuance, was achieved through the successful deployment of Hyperledger Fabric blockchain network and custom chaincode development. Unlike public blockchains such as Ethereum with established development frameworks like Hardhat, Hyperledger Fabric's permissioned nature presented unique challenges that were successfully overcome. The chaincode functions as intended smart contracts, automating the violation logging process and ensuring consistent penalty enforcement across all network transactions.

The second objective, developing a transparent and accessible platform, was accomplished through comprehensive system integration. The Hyperledger Fabric SDK serves as the critical API layer, establishing seamless communication between the web application frontend and the blockchain network backend. This integration enables stakeholders to access violation records and system information transparently. Additionally, the implementation of monitoring tools including Hyperledger Explorer, Prometheus, and Grafana provides real-time visibility into block creation and network operations, ensuring administrative oversight and driver access to relevant information.

6.3. Project Limitations and Constraints

Despite the successful achievement of the project goals, several limitations and constraints emerged during the development and implementation phases that impact the system's deployment readiness and operational effectiveness. These constraints primarily root

from the test environment configuration, infrastructure limitations, and the inherent characteristics of blockchain technology that affect real-world applicability.

1. Lack of Persistent Wallet and Identity Storage

The primary limitation involves persistent storage challenges for wallet identities and blockchain credentials. The test network environment utilizes testing certificates that reset automatically upon network shutdown, creating continuity issues for user authentication and system access. This ephemeral nature of the test environment prevents cryptographic materials and ledger data from being persistently stored by default, requiring user identities to undergo re-registration or re-enrollment processes after each network restart.

2. Limited Scalability of the Blockchain Network

Scalability constraints present another limitation affecting the system's real-world applicability. The test network configuration used throughout this project restricts the number of participating nodes and overall transaction throughput capacity. Real-world deployment scenarios involving larger user populations and higher transaction volumes would necessitate more robust and scalable blockchain infrastructure implementations, which were not explored within the current project parameters.

3. Complexity in Blockchain Network Management

The blockchain network management complexity requires substantial technical expertise encompassing scripting, programming, and containerization technologies such as Docker. The current system configuration assumes that administrative personnel possess advanced technical knowledge, while limited online documentation and resources compound these complexity challenges.

4. Use of Deprecated Libraries and Tools

Technical infrastructure limitations further constrain system functionality and maintenance capabilities. Several critical components rely on deprecated libraries, particularly Hyperledger Explorer, which no longer receives active maintenance support. This deprecation results in compatibility issues, such as connectivity problems between newer Raspberry Pi 5 systems with older PostgreSQL databases when interfacing with the explorer tool. The open-source nature of these libraries depends on active community maintenance, and the absence of dedicated maintainers leads to gradual system depreciation and reduced functionality over time.

5. Latency in Transaction Processing and Confirmation

Performance limitations affect the system's responsiveness in time-critical scenarios. Blockchain transactions experience inherent latency due to block creation processes and consensus mechanisms required for transaction validation. In real-time traffic violation scenarios, these delays could significantly impact the timely recording and processing of violations and penalties, potentially affecting enforcement effectiveness and user experience quality.

6.4. Future Works

To further enhance the capabilities, reliability, and real-world applicability of the developed decentralized blockchain traffic violation logging system, ChainGuard, several areas have been identified for future improvement. These suggestions aim to address the current limitations, expand the system functionalities, and support broader deployment and adoption.

1. Implementation of Persistent Wallet and Certificate Storage

One of the key areas for future enhancement is the integration of persistent storage mechanisms for wallet identities and cryptographic certificates. The current test network resets identities upon each restart due to its use of temporary development credentials, which affects long-term usability and continuity. In a production-ready environment, persistent identity storage is essential to ensure user records remain valid across sessions. Future work may explore incorporating secure storage solutions along with infrastructure enhancements such as DNS-based domain naming, SSL/TLS encryption for secure communication, and proper certificate authority (CA) management to support identity consistency and system security.

2. Enhancement for the Blockchain Network Scalability

The current network operates with a minimal node configuration, which is 2 organizations and each organization has one peer, suitable for development and testing. However, scaling the network to support a real-world traffic management system would require improvements in node distribution, consensus mechanisms, and infrastructure resilience. Future work can focus on deploying a multi-peer network setup that supports greater transaction throughput, fault tolerance, and cross-organizational communication. Additional research can also be directed toward tuning the ordering service and exploring consensus algorithms better suited for high-frequency traffic data.

3. Transaction Processing Time Optimization

Another potential improvement lies in reducing the time it takes for transactions to be processed and committed to the blockchain ledger. In real-time traffic scenarios, such as automated enforcement systems, delays in transaction confirmation can impact the system's effectiveness. Future development could investigate the use of Layer-2 solutions or protocol-

level optimizations within Hyperledger Fabric to achieve lower latency. Techniques such as batch processing, improved endorsement policies, or more efficient block generation parameters may also be considered to improve performance.

4. System Integration with External Agencies

To enhance the system's practical applicability, future work could focus on integrating the blockchain platform with external entities such as law enforcement databases, municipal court systems, and digital payment gateways. Such integration would allow end-to-end automation—from violation detection to penalty payment, while maintaining transparency and data integrity. Additionally, combining the system with Automated Enforcement Systems (AES), such as traffic light or speed camera monitoring, could enable automated violation logging directly to the blockchain, further reducing human input and increasing efficiency.

5. Simplification of Blockchain Management and User Interaction

Currently, the deployment and management of the network require a moderate level of technical knowledge involving Docker, CLI tools, and smart contract management. Future work can aim to simplify these processes by introducing graphical user interfaces (GUIs) for network administration, automated deployment scripts, and more intuitive user workflows. This would make the system more accessible to traffic authorities and IT administrators without deep blockchain expertise, ultimately promoting wider adoption and ease of use.

6.5. Conclusion

The development of the blockchain-based traffic violation logging and penalty management system has shown that Hyperledger Fabric can be effectively used for government-related applications. The project successfully met its main goals, including using

smart contracts to automate penalty issuance and building a transparent system that can be accessed by all stakeholders. Although the system faced some limitations, such as lack of persistent storage, limited scalability, and technical setup complexity. These challenges point to clear areas for future improvement. Overall, the project demonstrates how blockchain can solve common problems in traffic violation management and provides a strong starting point for modernizing similar systems in the public sector.

REFERENCES

- About the Ubuntu project.* (n.d.). Ubuntu. <https://ubuntu.com/about>
- AES - JPJ Portal.* (n.d.). Jabatan Pengangkutan Jalan. <https://www.jpj.gov.my/en/web/main-site/aes>
- Basic installation.* (n.d.). Ubuntu. <https://ubuntu.com/server/docs/basic-installation>
- Benjamin, L., Rahim, R., Gimino, G., & Vethasalam, R. (2024). *Road accidents cost Malaysia RM25bil in economic value in 2023, says Transport Minister.* The Star: <https://www.thestar.com.my/news/nation/2024/07/11/road-accidents-cost-malaysia-rm25bil-in-economic-value-in-2023-says-transport-minister>
- Ben Salem, A. (2023). *Scrum: Mastering the 3 pillars, 5 values and 7 key principles of agile project management.* Scrum-Master.Org. <https://scrum-master.org/en/scrum-mastering-the-3-pillars-5-values-and-7-key-principles-of-agile-project-management/>
- Bultmann, P. (2024). *The history of Scrum.* Agile42. <https://www.agile42.com/en/blog/scrum-history>
- Chan, M. (2024). *Road accidents in Malaysia in 2023 – 600,000 cases recorded; 6,443 deaths, 65% involved motorcyclists.* paultan.org: <https://paultan.org/2024/05/17/road-accidents-in-malaysia-in-2023-600000-cases-recorded-6443-deaths-65-involved-motorcyclists/>
- Chen, R. (n.d.). *What is Hyperledger Fabric?* <https://www.kaleido.io/blockchain-blog/what-is-hyperledger-fabric>
- Chintanonweb. (2024). *Node.js exposed: Unraveling the V8 engine's role.* DEV Community. <https://dev.to/chintanonweb/nodejs-exposed-unraveling-the-v8-engines-role-4eh9>
- CSS history and versions.* (2024). GeeksforGeeks. <https://www.geeksforgeeks.org/css-history-versions/>
- Devi, V. (2024). *One death every two hours due to motorbike accidents in Malaysia, says JPJ.* <https://www.thestar.com.my/news/nation/2024/02/29/one-death-every-two-hours-due-to-motorbike-accidents-in-malaysia-says-jpj>
- Diew, J. (2019). *Review - ASUS TUF Gaming FX505 (i7-8750H, GTX 1060, 8GB, 128GB SSD + 1TB HDD).* Tech Critter. <https://www.tech-critter.com/review-asus-tuf-gaming-fx505/>
- Express.js Tutorial.* (2025). GeeksforGeeks. <https://www.geeksforgeeks.org/express-js/>

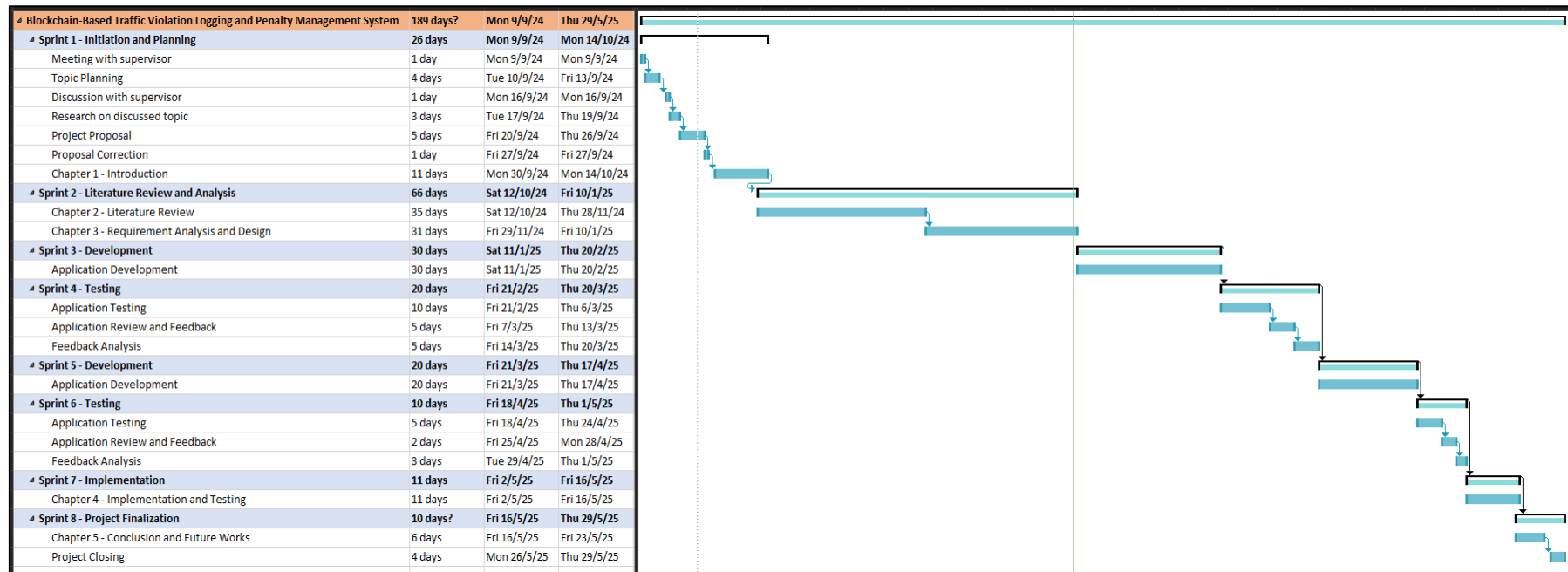
- Genting accident: Tour bus driver has no licence, 27 traffic violations, say cops.* (2024).
<https://www.thestar.com.my/news/nation/2024/06/29/genting-accident-tour-bus-driver-has-no-licence-27-traffic-violations-say-cops>
- Girbea, A. (2018). *Asus TUF Gaming FX505 review (FX505GE - i7, GTX 1050 Ti)*.
 Ultrabookreview.com. <https://www.ultrabookreview.com/21758-asus-tuf-fx505-review/>
- Gurnov, A. (2024). *What is a Project Schedule in Project Management?* wrike:
<https://www.wrike.com/project-management-guide/faq/what-is-a-project-schedule-in-project-management/>
- Happy, M. P. T. R. (2018). *MongoDB: a next gen. technology (Geek to geek)*.
<https://www.linkedin.com/pulse/mongodb-next-gen-technology-geek-maulik-pandya/>
- Hariz Ikhwan (2021). *Traffic Offences in Malaysia: How to Check and Settle Your Traffic 'Samans' Online*. <https://www.motorist.my/article/1160/traffic-offences-in-malaysia-how-to-check-and-settle-your-traffic-samans-online>
- Hassan, W. M. H. a. W. (2023). *Rethinking Traffic Enforcement in Malaysia: Adopting Global Best Practices for Safer Roads*. LinkedIn: <https://www.linkedin.com/pulse/rethinking-traffic-enforcement-malaysia-adopting-best-wan-hassan/>
- Heller, M. (2022, July 8). *What is Visual Studio Code? Microsoft's extensible code editor*. InfoWorld. <https://www.infoworld.com/article/2335960/what-is-visual-studio-code-microsofts-extensible-code-editor.html>
- History of JavaScript*. (2024). GeeksforGeeks. <https://www.geeksforgeeks.org/history-of-javascript/>
- Hoffmann, J. (n.d.). *The three Pillars of scrum*.
<https://resources.scrumalliance.org/Article/three-pillars-of-scrum>
- Hyperledger Fabric in Blockchain*. (2023). GeeksforGeeks.
<https://www.geeksforgeeks.org/hyperledger-fabric-in-blockchain/>
- Hyperledger vs Ethereum in Blockchain*. (2024). GeeksforGeeks.
<https://www.geeksforgeeks.org/blockchain-hyperledger-vs-ethereum/>
- Introduction — Hyperledger Fabric Docs main documentation*. (n.d.). <https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html>
- Iqbal, F. (2024). *What makes VSCode a popular IDE - Zee Palm, LLC - medium*. Medium.
<https://medium.com/zee-palm/what-makes-vscode-a-popular-ide-2f0109452a64>

- Johnson, C. (2022). *Java Basics: What is Visual Studio Code?* JRebel by Perforce.
<https://www.jrebel.com/blog/what-is-visual-studio-code>
- Johnston, S. (2024). 11 Years of Docker: Shaping the next decade of development | Docker.
Docker. <https://www.docker.com/blog/docker-11-year-anniversary/>
- Juillet, R. (2022). *Scrum Method: definition and benefits*. Bocasay.
<https://www.bocasay.com/scrum-method-benefits-web-development/>
- JPJ MySIKAP. (n.d.). JPJ.MY. https://www.jpj.my/misc/jpj_mysikap.htm
- Kaur, M. (2024). *Three traffic cops nabbed for allegedly asking RM100 bribe from British couple*. The Star. <https://www.thestar.com.my/news/nation/2024/01/30/three-traffic-cops-nabbed-over-alleged-bribing-involving-british-couple>
- Lau, A. (2023). *The paradox of Malaysian traffic summonses and discounts*.
<https://www.malaymail.com/news/opinion/2023/12/11/the-paradox-of-malaysian-traffic-summonses-and-discounts/106916>
- Marr, B. (2021). *Blockchain: A very short history of Ethereum everyone should read*. Bernard Marr. <https://bernardmarr.com/blockchain-a-very-short-history-of-ethereum-everyone-should-read/>
- Marchewka, J. T. (2016). *Information Technology Project Management: Providing Measurable Organizational Value* (5th ed.). Wiley Publishing.
- Moe, N. B., & Dingsøyr, T. (2008). Scrum and team effectiveness: theory and practice. In *Lecture notes in business information processing* (pp. 11–20).
https://doi.org/10.1007/978-3-540-68255-4_2
- MongoDB. (n.d.). *About us - Our story | MonGoDB*. <https://www.mongodb.com/en-us/company>
- Morhan, S. (2024). *2.4 million drivers potential road bullies: Survey*.
<https://thesun.my/local-news/24-million-drivers-potential-road-bullies-survey-GC12760314>
- MyBayar PDRM (n.d.). Royal Malaysia Police (RMP).
<https://play.google.com/store/apps/details?id=com.pdrm.mybayarsamanpdrm>
- MyBayar PDRM (n.d.). MyBayar PDRM. <https://mybayar.rmp.gov.my/en>
- Noviantika, G. (2024, July 2). *What is Ubuntu? A quick beginner's guide*. Hostinger Tutorials. https://www.hostinger.com/tutorials/what-is-ubuntu#What_Is_Ubuntu

- Online Vehicle Registration* (n.d.). MyGOV - the government of Malaysia's official portal.
<https://www.malaysia.gov.my/portal/content/30685>
- PHP Introduction*. (2024). GeeksforGeeks. <https://www.geeksforgeeks.org/php-introduction/>
- Road Accidents and Fatalities In Malaysia*. (n.d.). Ministry of Transport Malaysia:
<https://www.mot.gov.my/en/land/safety/road-accident-and-facilities>
- Road Transport Department. (n.d.). *Act 333*. JPJ Portal Jabatan Pengangkutan Jalan Malaysia:
<https://www.jpj.gov.my/en/web/main-site/akta-333>
- Reza, S. (2019). *GitHub - sakibreza/Traffic-Rules-Violation-Detection-System: An Image Processing and Computer Vision based traffic rules violation detection system from CCTV camera footage (including GUI App)*. GitHub.
<https://github.com/sakibreza/Traffic-Rules-Violation-Detection-System>
- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*.
<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100>
- Sergeenkov, A. (2021). *Who founded Ethereum?* CoinMarketCap Academy.
<https://coinmarketcap.com/academy/article/who-founded-ethereum>
- Sherman, R. (2015). Chapter 18 - Project Management. In R. Sherman, *Business Intelligence Guidebook* (pp. 449-492). Boston: Morgan Kaufmann.
doi:<https://doi.org/10.1016/B978-0-12-411461-6.00018-6>
- Smart contracts on blockchain*. (2024). IBM. <https://www.ibm.com/topics/smart-contracts>
- Spydra. (2024). Hyperledger Fabric vs Ethereum: A Comprehensive comparison. *Medium*.
<https://medium.com/coinmonks/hyperledger-fabric-vs-ethereum-a-comprehensive-comparison-d557d2f86231>
- Susnjara, S., & Smalley, I. (2024). Docker. *What is Docker?*
<https://www.ibm.com/topics/docker>
- T, G., & Villa, M. (2022). *The basics of Package.json*. The NodeSource Blog - Node.js Tutorials, Guides, and Updates. <https://nodesource.com/blog/the-basics-of-package-json>
- Tan, D. (2022). *PDRM introduces MyBayar Saman app and online portal, offers 50% discount as introductory offer*. Paul Tan's Automotive News.
<https://paultan.org/2021/03/25/pdrm-introduces-mybayar-saman-app-and-online-portal-offers-50-discount-as-introductory-offer/>

- Team, I. (2024). *What does Proof-of-Stake (POS) mean in crypto?* Investopedia.
<https://www.investopedia.com/terms/p/proof-stake-pos.asp>
- Unpaid traffic fines reach RM4 billion, Singaporeans top offender list.* (2024).
<https://thesun.my/local-news/unpaid-traffic-fines-reach-rm4-billion-singaporeans-top-offender-list-BO12852773>
- Visual Studio Code User Interface.* (2021). Visual Studio Code.
<https://code.visualstudio.com/docs/getstarted/userinterface>
- What is Docker?* (2024). Docker Documentation. <https://docs.docker.com/get-started/docker-overview/>
- What is Ethereum?* (n.d.). Coinbase. <https://www.coinbase.com/learn/crypto-basics/what-is-ethereum>
- What is Scrum?* (n.d.). scrum.org: <https://www.scrum.org/resources/what-scrum-module>
- What is Scrum?* (n.d.). AWS: <https://aws.amazon.com/what-is/scrum/>
- Who developed HTML?* (2024). GeeksforGeeks. <https://www.geeksforgeeks.org/who-developed-html/>
- Why Visual Studio code?* (2021). <https://code.visualstudio.com/docs/editor/whyvscode>
- Yin, A. C. P. (2021, August 9). *PDRM launches MyBayar Saman Portal and App, offers 50% introductory discount.* RinggitPlus. <https://ringgitplus.com/en/blog/personal-finance-news/pdrm-launches-mybayar-saman-portal-and-app-offers-50-introductory-discount.html>
- Zheng, Z., Xie, S., Dai, H., Chen, H., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *2017 IEEE International Congress on Big Data (BigData Congress)*, (pp. 557-564). Honolulu.
doi:10.1109/BigDataCongress.2017.85

Appendix A: Gantt Chart



Appendix B: Sample Questionnaire



Final Year Project: Survey on Digital Traffic Summons Management System using Blockchain Technology

Dear respected respondents,

My name is Richard Jong Wei Ze, a final year student from Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS). This survey is conducted for my final year project titled "Blockchain-based Traffic Violation Logging and Penalty Management System". The purpose of this project is to developed a decentralized system for securely logging and storing traffic violation data by utilizing blockchain technology.

Other objectives include:

1. To implement smart contracts for automated and consistent penalty issuance.
2. To develop a transparent and accessible platform for all the stakeholders, including traffic authorities, and drivers.

The survey takes approximately 10 to 15 minutes to complete. All information provided will be kept confidential, and will only be used for academic purposes. If you have any inquiries regarding the survey, feel free to contact me at 010-9630210 or through email at 81049@siswa.unimas.my.

Your participation in this survey is highly appreciated and will greatly contribute to the success of this project.

Warm regards,
Richard Jong Wei Ze.

rjwz.work01@gmail.com [Switch account](#)

 Not shared





Final Year Project: Survey on Digital Traffic Summons Management System using Blockchain Technology

rjwz.work01@gmail.com [Switch account](#)



Not shared

* Indicates required question

Section A: Demographic Information

Age Group *

- ☐ 16 - 25
- ☐ 26 - 35
- ☐ 36 - 45
- ☐ 46 - 55
- ☐ 56 and above

State of Residence *

Choose

Driving Experience (Years) *

- ☐ Less than 1
- ☐ 1 - 5
- ☐ 6 - 10
- ☐ More than 10

Vehicle Type: *

- ☐ Car
- ☐ Motorcycle
- ☐ Commercial vehicle
- ☐ Other: _____

[Back](#)

[Next](#)

Page 2 of 5

[Clear form](#)



Final Year Project: Survey on Digital Traffic Summons Management System using Blockchain Technology

rjwz.work01@gmail.com [Switch account](#)



Not shared

* Indicates required question

Section B: Current System Assessment

Please use the scale below to rate the questions in this section:

1 = *Not at all*

2 = *Slightly*

3 = *Moderately*

4 = *Very*

5 = *Extremely*

How easy do you find checking your traffic summons? *

	1	2	3	4	5	
Not at all	☐	☐	☐	☐	☐	Extremely

Do you find the current summons payment process convenient? *

	1	2	3	4	5	
Not at all	☐	☐	☐	☐	☐	Extremely

How confident are you in the accuracy of your traffic violation records? *

	1	2	3	4	5	
Not at all	☐	☐	☐	☐	☐	Extremely

How easily can you access your complete traffic violation history? *

	1	2	3	4	5	
Not at all	☐	☐	☐	☐	☐	Extremely

[Back](#)

[Next](#)

Page 3 of 5

[Clear form](#)



Final Year Project: Survey on Digital Traffic Summons Management System using Blockchain Technology

rjwz.work01@gmail.com [Switch account](#)



Not shared

* Indicates required question

Section C: Digital System Features and Improvements

Please use the scale below to rate the questions in this section:

1 = **Strongly Disagree**

2 = **Disagree**

3 = **Neutral**

4 = **Agree**

5 = **Strongly Agree**

Would you prefer to receive instant notifications about your traffic summons? *

1 2 3 4 5
Strongly Disagree ☐ ☐ ☐ ☐ ☐ Strongly Agree

Would a tamper-proof system increase your trust in traffic enforcement? *

1 2 3 4 5
Strongly Disagree ☐ ☐ ☐ ☐ ☐ Strongly Agree

Do you think automated summons issuance would be more efficient than manual processing? *

1 2 3 4 5
Strongly Disagree ☐ ☐ ☐ ☐ ☐ Strongly Agree

Would unchangeable proof help reduce your concerns about summons disputes? *

1 2 3 4 5
Strongly Disagree ☐ ☐ ☐ ☐ ☐ Strongly Agree

Would you find it more convenient to pay summons through a web application? *

1 2 3 4 5
Strongly Disagree ☐ ☐ ☐ ☐ ☐ Strongly Agree

Would you find it helpful to have access to your complete traffic violation history? *

1 2 3 4 5
Strongly Disagree ☐ ☐ ☐ ☐ ☐ Strongly Agree

Would you find it valuable to have the ability to track your summons payment status? *

1 2 3 4 5
Strongly Disagree ☐ ☐ ☐ ☐ ☐ Strongly Agree

[Back](#)

[Next](#)

Page 4 of 5

[Clear form](#)



Final Year Project: Survey on Digital Traffic Summons Management System using Blockchain Technology

rjwz.work01@gmail.com [Switch account](#)



Not shared

* Indicates required question

Section D: Privacy and Public Acceptance

Please use the scale below to rate the questions in this section:

- 1 = *Not at all*
- 2 = *Slightly*
- 3 = *Moderately*
- 4 = *Very*
- 5 = *Extremely*

How concerned are you about your personal information security in a new traffic system? *

1 2 3 4 5
Not at all ☐ ☐ ☐ ☐ ☐ Extremely

Would a digital system make traffic enforcement more trustworthy for you? *

1 2 3 4 5
Not at all ☐ ☐ ☐ ☐ ☐ Extremely

How comfortable would you be with automatic summons issuance through traffic cameras? *

1 2 3 4 5
Not at all ☐ ☐ ☐ ☐ ☐ Extremely

Would you be willing to adopt a new digital traffic summons system? *

1 2 3 4 5
Not at all ☐ ☐ ☐ ☐ ☐ Extremely

Do you think a new system would better motivate people to follow traffic rules? *

1 2 3 4 5
Not at all ☐ ☐ ☐ ☐ ☐ Extremely

THANK YOU

[Back](#)

[Submit](#)

Page 5 of 5

[Clear form](#)

Appendix C: Sample Feedback Form

Final Year Project: ChainGuard User Satisfaction Survey

Dear respected respondents,

My name is Richard Jong Wei Ze, a final year student at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS). This feedback survey is part of my final year project titled "Blockchain-Based Traffic Violation Logging and Penalty Management System (ChainGuard)." The goal of this survey is to gather your feedback on the usability and functionality of the system after testing it.

The survey will take approximately 5 minutes to complete. All responses are anonymous and confidential and will be used solely for academic purposes to improve the quality and effectiveness of the system.

If you have any questions, feel free to contact me at 010-9630210 or via email at 81049@siswa.unimas.my.
Thank you for your valuable time and contribution!

Warm regards,
Richard Jong Wei Ze

[Next](#)[Clear form](#)

This content is neither created nor endorsed by Google. - [Terms of Service](#) - [Privacy Policy](#)

Does this form look suspicious? [Report](#)

Google Forms

Final Year Project: ChainGuard User Satisfaction Survey

* Indicates required question

Section 1: User Demographics

Role *

- ☐ Admin
- ☐ User

[Back](#)

[Next](#)

[Clear form](#)

This content is neither created nor endorsed by Google. - [Terms of Service](#) - [Privacy Policy](#)

Does this form look suspicious? [Report](#)

Google Forms

Final Year Project: ChainGuard User Satisfaction Survey

* Indicates required question

Section 2: Admin Task Completion Feedback

I found the task "Register an account" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task "Login to dashboard" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task "Create a new traffic violation" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task "Reveal an existing appeal" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task "Generate a report" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

[Back](#)

[Next](#)

[Clear form](#)

Final Year Project: ChainGuard User Satisfaction Survey

* Indicates required question

Section 2: User Task Completion Feedback

I found the task "Register an account" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task "Login to Portal" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task " View logged violations" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task " Pay a penalty" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the task "Submit an appeal" easy to complete. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

[Back](#)

[Next](#)

[Clear form](#)

Final Year Project: ChainGuard User Satisfaction Survey

* Indicates required question

Section 3: Overall System Usability

I would like to use this system frequently. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the system was easy to use without any glitches. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I found the functions well integrated. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I would learn to use the system quickly. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

I am satisfied with the overall system usability. *

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

[Back](#)

[Submit](#)

[Clear form](#)