



Faculty of Computer Science and Information Technology

AUGMENTED REALITY-ASSISTED EVACUATION FOR FLOOD VICTIMS

TAN MIN HONG

**Bachelor of Computer Science with Honours
(Multimedia Computing)**

2025

**AUGMENTED REALITY-ASSISTED EVACUATION
FOR FLOOD VICTIMS**

TAN MIN HONG

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

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

iii

DECLARATION

I hereby declare that this project together with all the content is my original work where I have not copied and plagiarism from other's work except due to the research-based information and related materials that were adjusted and take out from other resources, which have clearly been cited or stated respectively.

I confirm that the work presented in this thesis is original and has been prepared by me. Any assistance or guidance received in the preparation of this thesis has been properly acknowledged.

Signed,



.....

Tan Min Hong (81349)

Bachelor of Computer Science with Honours (Multimedia Computing)

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak (UNIMAS)

23 June 20

ACKNOWLEDGMENT

The accomplishment of this project is highly dependent on my supervisor's motivation and guidance, for which I would like to express my gratitude and appreciation to Faculty of Computer Science and Information Technology, FYP Lab for support and assistance in this project. I would also like to express my gratitude and appreciation to my supervisor Ts. Dr. Chiu Po Chan. Besides, I would also like to thank my examiner, Dr. Irwandi Hipni Mohamad Hipiny and the Final Year Project coordinator, Prof. Wang Yin Chai who provide valuable information and tips during lectures. I also want to thank my family members for their encouragement in getting me through this important journey. Their never-ending support has successfully boosted my motivation to complete this project. Finally, I would like to thank all my friends for their invaluable assistance. I am grateful for their assistance, guidance, time, and effort.

Abstract

Floods pose significant threats to human life and infrastructure, particularly in urbanized regions like Kota Samarahan, Sarawak. Traditional flood management systems often lack real-time, localized guidance, leaving communities vulnerable during emergencies. This project proposes and implements FloodGuard, a mobile application that utilizes Augmented Reality (AR) to guide flood victims along safe evacuation routes. Developed using Unity and AR Foundation, with integration of Cesium 3D Tiles and GPS-based mapping, the system overlays critical evacuation information in real-world environments to improve situational awareness and decision-making. A Rapid Application Development (RAD) methodology was employed to create a user-centered design informed by feedback from UNIMAS students. Key features include hazard detection, AR-guided navigation, and an emergency preparedness checklist. Usability testing using the System Usability Scale (SUS) resulted in an average score of 92.75, indicating excellent usability and a positive user experience. Limitations in geographic scalability and AR detection under poor conditions were noted. Future enhancements aim to expand area coverage, integrate real-time flood data, and improve accessibility features. FloodGuard demonstrates the potential of AR technology to transform disaster response, offering a practical, immersive tool for safer and more efficient flood evacuations.

Abstrak

Banjir merupakan ancaman serius terhadap keselamatan nyawa dan infrastruktur, terutamanya di kawasan bandar seperti Kota Samarahan, Sarawak. Sistem pengurusan banjir tradisional sering kali tidak menyediakan panduan masa nyata yang bersifat setempat, menyebabkan komuniti terdedah semasa kecemasan. Projek ini mencadangkan dan melaksanakan FloodGuard, sebuah aplikasi mudah alih yang menggunakan teknologi Realiti Terimbuh (AR) untuk membimbing mangsa banjir ke laluan pemindahan yang selamat. Sistem ini dibangunkan menggunakan Unity dan AR Foundation, dengan integrasi Cesium 3D Tiles dan pemetaan berasaskan GPS bagi memaparkan maklumat penting pemindahan dalam persekitaran sebenar untuk meningkatkan kesedaran situasi dan membuat keputusan. Kaedah Pembangunan Aplikasi Pantas (RAD) digunakan untuk menghasilkan reka bentuk berpusatkan pengguna berdasarkan maklum balas pelajar UNIMAS. Ciri utama termasuk pengesanan bahaya, navigasi berpanduan AR, dan senarai semak persediaan kecemasan. Ujian kebolegunaan menggunakan Skala Kebolegunaan Sistem (SUS) menunjukkan purata skor sebanyak 92.75, yang menandakan tahap kebolegunaan yang sangat baik dan pengalaman pengguna yang positif. Walau bagaimanapun, terdapat beberapa kekangan dari segi skop geografi dan keberkesanan pengesanan AR dalam keadaan pencahayaan yang rendah. Penambahbaikan masa depan merangkumi peluasan kawasan liputan, integrasi data banjir masa nyata, dan peningkatan ciri kebolehcapaian. FloodGuard membuktikan potensi teknologi AR dalam merevolusikan tindak balas bencana, menawarkan alat yang praktikal dan imersif untuk pemindahan banjir yang lebih selamat dan cekap.

Table of Contents

Abstract.....	vi
Abstrak	vii
Table of Contents	viii
Table of Figure	xi
INTRODUCTION	1
1.1 Introduction.....	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Scope	3
1.5 Brief Methodology	3
1.6 Significance of Project.....	3
1.7 Project Schedule	4
1.8 Project Outcome	6
1.9 Conclusion.....	6
Chapter 2	7
Literature Review	7
2.1 Augmented Reality.....	7
2.2 Analysis of Existing Application/Framework.....	7
2.2.1 AR Way Finder	8
2.2.2 AR Flood	9
2.2.3 FloodVue Framework.....	10
2.2.4 Existing Augmentation Reality Application Comparison.....	11
2.3 Map technologies	12
2.4 AR Development Frameworks	17
2.5 Conclusion.....	21
Chapter 3	22
Methodology	22
3.1 Introduction.....	22
3.2 Rapid Application Development (RAD).....	22
3.3 Development	30

3.4	Prototype	31
3.3	Implementation.....	38
3.4	Software limitations.....	39
3.5	Hardware limitations	40
3.6	Testing	41
3.7	Conclusion.....	41
Chapter 4		42
	Implementation	42
4.1	Introduction.....	42
4.2	System Implementation.....	42
4.2.1	Unity.....	42
4.2.2	Cesium	46
4.3	Key Features	50
4.3.1	Coordination (Latitude, Longitude).....	50
4.3.2	Calculation.....	50
4.3.3	Navigation Line	55
4.4	System Architecture.....	60
4.5	User Interfaces (UI).....	62
4.6	Conclusion.....	66
Chapter 5		68
	Testing and Evaluation	68
5.1	Introduction.....	68
5.2	System Usability Scale.....	68
5.3	System Usability Scale Analysis	68
5.4	SUS Two Factor Analysis	71
5.5	Results and Discussion.....	72
5.6	Scenario based testing overview	74
5.7	Conclusion.....	75
Chapter 6		77
	Conclusion and Future Work	77

6.1	Introduction.....	77
6.2	Objective Achieved	77
6.3	System Limitation	77
6.4	Future enhancement.....	78
6.5	Final thoughts.....	79
	Reference	80
	Appendix.....	83

Table of Figure

Figure 1.1: Reality-virtuality spectrum retrieved from Ruggiero (2018).	1
Figure 1.2: Project Schedule Diagram 1	4
Figure 1.3: Project Schedule Diagram 2	5
Figure 2.1: Screenshot of application AR Way Finder	8
Figure 2.2: Screenshot of application AR Flood	9
Figure 2.3: Schematic of framework for global flood risk assessment.....	10
Figure 2.4: Google map view of Uni Garden area.....	13
Figure 2.5: Here map view of Uni Garden area.....	14
Figure 2.6: Topology map view of Uni Garden area	15
Figure 3.1: Rapid application development model.....	23
Figure 3.2: Chart of respondents who have experienced floods and its impact	24
Figure 3.3: Chart of most desired features in app.....	25
Figure 3.4: Chart of importance of user-friendly and simplicity in an emergency app ...	25
Figure 3.5: Use Case Diagram of Floodguard	26
Figure 3.6: Sequence Diagram of Floodguard.....	27
Figure 3.7: Activity Diagram of Floodguard	28
Figure 3.8: System Flow Diagram of Floodguard	29
Figure 3.9: Home Page of Floodguard Application	31
Figure 3.10: Evacuation Path Page of Floodguard Application.....	32
Figure 3.11: Map View of Floodguard Application	33
Figure 3.12: Detected Hazard	34
Figure 3.13: Reached destination.....	35
Figure 3.14: Destination selection to safety point page of Floodguard Application	36
Figure 3.15: Loading View Page of Floodguard Application	37
Figure 3.16: Emergency kits checklist page of Floodguard Application	38
Figure 4.1: Unity - terrain system.....	45
Figure 4.2: Accessing Cesium 3D Token.....	46
Figure 4.3: Cesium 3D Tiles Limitation	47
Figure 4.4: Hybrid Solution for Unity and Cesium	48
Figure 4.5: Sub Scene Feature in Cesium.....	49
Figure 4.6: Origin position in CesiumGeoreference.....	50

Figure 4.7: Code snippet for calculation using cesium for unity	51
Figure 4.8: Code snippet for obtaining user location.....	53
Figure 4.9: Position on Lat (1.4603032039173787), Lon (110.40976117233218)	55
Figure 4.10: Position on Lat (1. 4615193598316287), Lon (110. 41387269287807).....	55
Figure 4.11: Pseudocode on how system update the user and camera positions	57
Figure 4.12: Categorizing walkable surfaces with obstacles	58
Figure 4.13: Nav Mesh Agent.....	59
Figure 4.14: Overview of the System Architecture	60
Figure 4.15: Home Page of Floodguard Application	62
Figure 4.16: Loading View of Floodguard Application.....	62
Figure 4.17: Checklist page of Floodguard Application.....	63
Figure 4.18: Notify Window page of Floodguard Application	65
Figure 4.19: Mapping page of Floodguard Application.....	65
Figure 5.1: Average score for the SUS survey	69

Table of Table

Table 2.1: Comparison between AR Way Finder,AR Flood,FloodVue Framework.....	12
Table 2.2: Comparison between Google Maps, Here Maps and Topography Maps.....	16
Table 2.3: Comparison between Framework Development.....	20
Table 3.1: Software Limitation for the system.....	39
Table 3.2: Hardware Limitation for the system	40
Table 4.1: Comparison of Unity's Render Pipelines.....	44
Table 5.1: Recorded result from Google Form	69
Table 5.2: Calculation based on recorded result	70
Table 5.3: SUS Score Interpretation Table.....	71

CHAPTER 1

INTRODUCTION

1.1 Introduction

Augmented reality (AR) technology provides immersive and interactive experiences that enable individuals to visualize flood evacuation routes in real time. The use of augmented reality (AR) in flood evacuation planning can improve real-time decision-making during emergencies, particularly in regions like Kota Samarahan, where flooding hazards are increasing due to urbanization and unpredictable rainfall patterns. AR enables the overlay of vital info like flood levels and safe evacuation routes, providing users with interactive, location-specific guidance. AR experience can help user navigate flood conditions more effectively, ensuring safer evacuations. This technology has proven valuable in disaster management, including earthquake and wildfire scenarios, where AR simulations enhance situational awareness and improve evacuation strategies (Khanal et

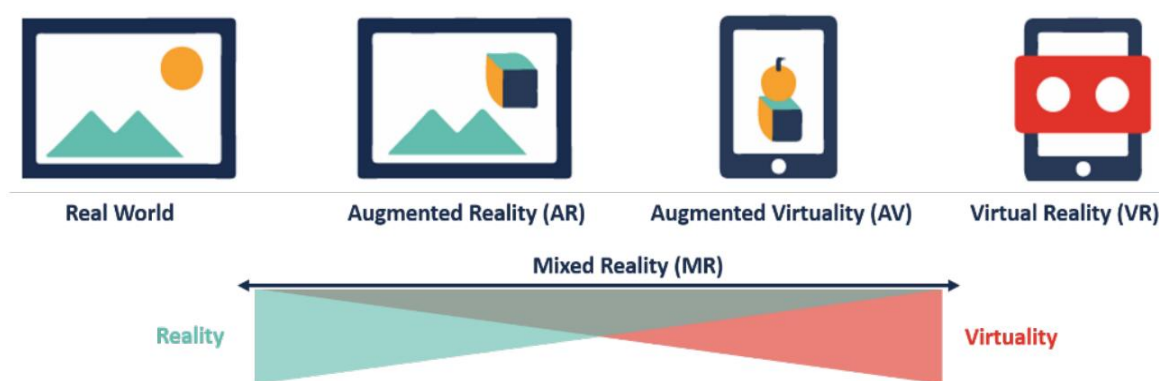


Figure 1 – Reality-virtuality spectrum

Figure 1.1 Reality- virtuality spectrum (Ruggiero, 2018)

al., 2022).

Research published by the International Organization for Migration (2024) emphasizes how AR and virtual reality (VR) tools can play a significant role in engaging communities for disaster risk reduction. These technologies not only educate individuals, but also simulate real-world crisis scenarios, offering them hands-on experience on how to respond in an emergency. For flood-prone locations like Kota Samarahan, where flood hazards can alter rapidly due to unpredictable rainfall, AR technology could offer real-time, location-specific instructions, potentially save lives and reduce panic.

The use of AR technology could transform disaster management. Studies by (Khanal et al., 2022) show that AR-based simulations can significantly improve preparedness by enabling individuals to practice evacuation procedures in a safe, controlled environment before a real emergency occurs.

1.2 Problem Statement

Flooding is one of the most urgent environmental issues in Malaysia, particularly in urbanized areas like Kota Samarahan, Sarawak. The region has witnessed increasingly frequent and severe flood events due to rapid urbanization, and inconsistent rainfall patterns. The flash flood in UniGarden, January 2015, serves as a sharp reminder of the urban residents' vulnerability to extreme weather events (Er, Lim, & Bong, 2017). Factors such as blocked drainage systems, and extensive vegetation development resulting in disruptions to daily life, damage to infrastructure, and significant threats to public safety.

Current flood management systems in Kota Samarahan primarily focus on physical infrastructure improvements, such as flood barriers and drainage systems. While these efforts address long-term mitigation needs, they fail to provide immediate support for residents during flood emergencies. The lack of localized evacuation assistance leaves communities unprepared to respond. Residents often face confusion in identifying safe evacuation routes, increasing the risk to life and property.

Incorporating advanced technologies into disaster management systems offers a promising solution to these challenges. As highlighted by Zhu and Li (2021), Virtual Reality (VR) and Augmented Reality (AR) technologies are increasingly being adopted for emergency management to enhance pre-disaster preparedness, improve real-time responses during emergencies, and support post-disaster recovery efforts by providing immersive simulations and real-time guidance. Therefore, there is a need to propose an augmented reality-assisted evacuation system to guide flood victims along safe paths using interactive visualizations.

1.3 Objectives

The aim of this project is to propose an augmented reality-assisted flood evacuation system. The specific objectives include:

1. To develop an augmented reality-assisted flood evacuation
2. To evaluate the usability of the proposed augmented reality-assisted evacuation

1.4 Scope

This project primarily focuses on the residents of Unimas and the surrounding areas of Kota Samarahan, Sarawak, who are particularly vulnerable to flooding. It aims to develop an augmented reality (AR) system to assist in evacuation during flood emergencies. The project will involve testing the AR system under simulated flood conditions to evaluate its effectiveness in guiding residents through safe evacuation routes, improving situational awareness, and enhancing emergency response efforts.

1.5 Brief Methodology

This study adopts a developmental and evaluative approach to design and assess a mobile-based Augmented Reality (AR) application aimed at guiding flood victims to safe evacuation routes. The development process involves 4 phases, which are gathering requirements from UNIMAS students, analyzing existing AR navigation systems, and designing the application architecture with Google Maps or HERE Maps APIs for route mapping and AR frameworks for overlaying evacuation cues and conducting evaluation testing upon the completion of the system. The system includes an AR module for visualizing safe routes, a backend database using MySQL for storing emergency data, and PHP for API integration. User testing with UNIMAS students in simulated flood scenarios evaluates the app's usability, navigation clarity, and AR accuracy, with feedback used to refine the application.

1.6 Significance of Project

This project addresses critical needs in disaster response and safety for flood-affected areas by increasing safety for flood victims by reducing confusion and helping users avoid hazardous areas. The AR application can aid emergency teams by directing evacuees more effectively, allowing users to distribute their resources better and potentially saving more lives. Through the integration of AR technology, the proposed system not only addresses the immediate needs of flood evacuation but also aligns with broader disaster risk reduction objectives. By enhancing preparedness and response capabilities, the system has the potential to reduce casualties, minimize property damage, and strengthen community resilience in flood-prone areas. This placed AR at the cutting edge of improving flood management strategies and protecting urban populations that are at risk.

1.7 Project Schedule

				OCT				NOV				DEC				JAN			
FYP 1	Duration	START	END	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Requirement Analysis	65days	10/10/2024	12/13/2024																
Discuss FYP Title with Supervisor	8days	10/10/2024	10/17/2024																
Bried Proposal Preparation	3days	10/18/2024	10/20/2024																
Milestone : Brief Proposal Complete	0days	10/20/2024	10/20/2024																
Feedback and Revision	8days	10/21/2024	10/28/2024																
Full Proposal Development	17days	10/29/2024	11/14/2024																
Milestone: Full Proposal Complete	0days	11/14/2024	11/14/2024																
Chapter 1 Introduction	7days	11/15/2024	11/21/2024																
Milestone: Chapter 1 Complete	0days	11/21/2024	11/21/2024																
Chapter 2 Literature Review	22days	11/22/2024	12/13/2024																
Milestone: Chapter 2 Complete	0days	12/13/2024	12/13/2024																
Design	23days	12/14/2024	1/5/2025																
Construct Data Flow Diagram	4days	12/14/2024	12/17/2024																
Identify Features and Function	4days	12/18/2024	12/21/2024																
Prototype Design	15days	12/22/2024	1/5/2025																
Milestone: Chapter 3 Complete	0days	1/5/2025	1/5/2025																
Report	15days	1/6/2025	1/20/2025																
Final Report & Paper Preparation	7days	1/6/2025	1/12/2025																
Milestone: Report Submission	0days	1/12/2025	1/12/2025																
Presentation	8days	1/13/2025	1/20/2025																

Figure 1.2: Project Schedule Diagram 1

				MAR				APR				MAY				JUN				JUL			
FYP 2	Duration	START	END	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Implementation	97days	3/10/2025	6/14/2025																				
AR Feature Development	44days	3/10/2025	4/22/2025																				
Navigation and Pathfinding System	8days	4/23/2025	4/30/2025																				
UI/UX Development	15days	5/1/2025	5/15/2025																				
Milestone : Chapter 4 Complete	0days	5/15/2025	5/15/2025																				
Database Integration	8days	5/16/2025	5/23/2025																				
API Development	8days	5/24/2025	5/31/2025																				
Milestone : Chapter 5 Complete	0days	5/31/2025	5/31/2025																				
Testing	15days	6/1/2025	6/15/2025																				
System testing	15days	6/1/2025	6/15/2025																				
Milestone : Chapter 6 Complete	0days	6/15/2025	6/15/2025																				
Final report & Presentation	15days	6/16/2025	6/23/2025																				
FYP Symposium	8days	6/24/2025	7/1/2025																				

Figure 1.3: Project Schedule Diagram 2

1.8 Project Outcome

Augmented Reality-Assisted Flood Evacuation can significantly affect disaster management and community safety. The application offers a range of AR-based tools, including safe route guidance, and situational alerts, to help flood victims make informed, life-saving decisions. Users will receive help from increased situational awareness, improved evacuation efficiency, and strengthened resilience against flooding events.

1.9 Conclusion

This chapter introduces the use of augmented reality (AR) technology for flood evacuation, focusing on its ability to provide real-time, interactive guidance. The project addresses the increasing flood risks in Kota Samarahan, exacerbated by urbanization and unpredictable rainfall. Current flood management systems lack immediate, localized evacuation support, leading to confusion and safety risks during emergencies.

The proposed solution is an AR-assisted evacuation system designed to enhance situational awareness and guide residents through safe routes. The system will be developed using AR frameworks. This project aims to improve disaster response, increase safety, and strengthen community resilience in flood-prone areas.

Chapter 2

Literature Review

2.1 Augmented Reality

Augmented Reality (AR) technologies work by combining or enhancing digital content in the real world. It builds a connection between the physical and virtual worlds. The technology bridges the gap between these two realms by augmenting the physical environment with virtual layers. AR technology works in real time and uses interactive displays to show augmented 3D graphics in the real world. AR does not create a virtual world nor replace the physical world, instead it enriches and complements the real world (Kiryakova, Angelova, Yordanova, 2018). AR includes three components such as tracking, rendering process, and interaction (Grubert, Langlots, Zollmann, & Regenbrecht, 2017). These features are vital in AR applications. The tracking components are utilized to locate the device's position when rendering digital components in physical environments. Multimedia information can be rendered by tracking data. The interactivity component enables the user to modify and interact with both digital and physical information.

2.2 Analysis of Existing Application/Framework

This section reviews existing systems related to Augmented Reality applications and framework. These applications highlight various technological approaches to enhancing donation processes and provide insights into their strengths, limitations, and areas for improvement.

2.2.1 AR Way Finder

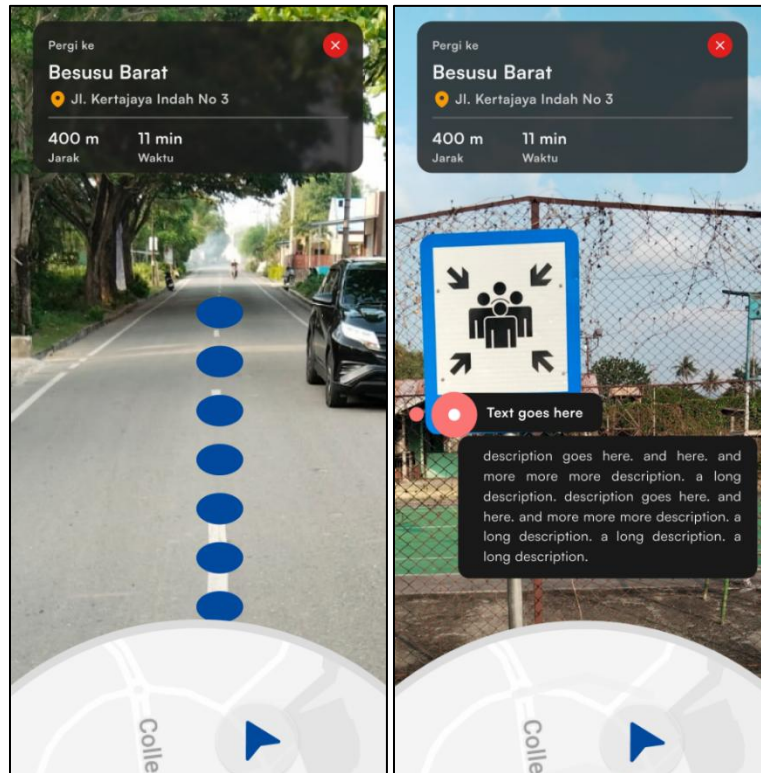


Figure 2.1: Screenshot for application AR Way Finder

AR Way Finder is an augmented reality application designed to assist users in identifying the safest and quickest evacuation routes during disasters. By utilizing a smartphone's camera, the app overlays digital directions—such as arrows and signs—onto the real-world view, guiding users effectively to safety. This intuitive interface ensures that individuals can navigate complex environments during emergencies with ease. This product includes training and can be socialized throughout Indonesia.

In addition, AR Way Finder technology has been successfully adapted for other use cases, such as library navigation systems. According to R et al. (2023), AR-based wayfinding systems revolutionize user experiences by overlaying directional information onto real environments via smartphone cameras, simplifying navigation and improving accessibility. These systems employ advanced features like GPS, real-time path suggestions, and visual overlays to enhance situational awareness, making them highly effective in complex environments. Such functionalities can be extended to disaster evacuation scenarios to further improve safety and efficiency.

2.2.2 AR Flood



Figure 2.2: Screenshot for application AR Flood

AR Flood is an application that enables users to visualize flood data through 3D renderings of flood-prone areas, enhancing awareness and preparedness. Users can select specific points of interest, and the app dynamically updates maps with 3D visualizations and flood data for the chosen location. This feature aids in understanding potential flood impacts on various landmarks, facilitating better preparedness measures.

2.2.3 FloodVue Framework

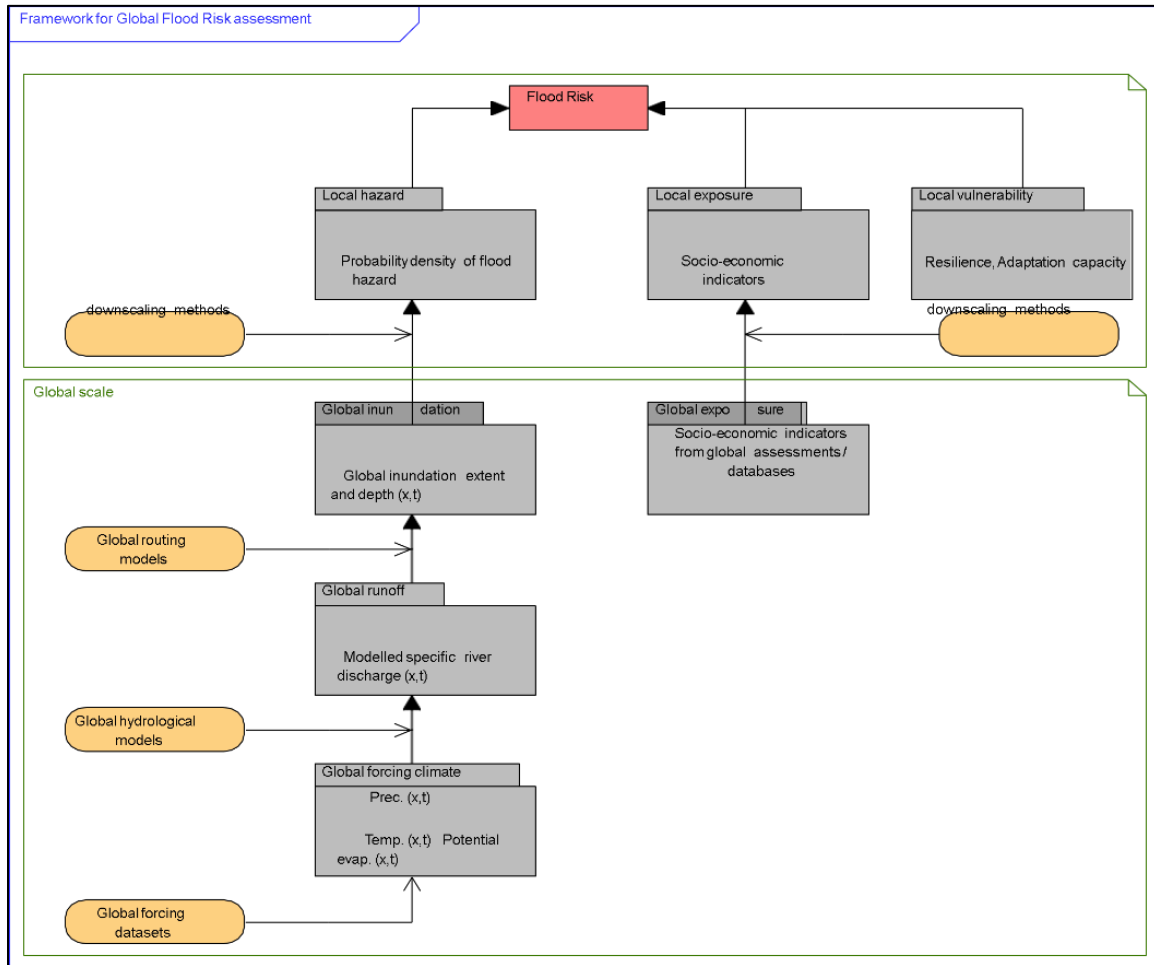


Figure 2.3: Schematic of framework for global flood risk assessment (Winsemius, 2013).

The FloodVue application utilizes the FloodVue Framework (FVF) to explain interactive visualization of flood data, primarily aimed at aiding infrastructure planning and enhancing public awareness. This framework integrates AR technology to simulate flood scenarios, presenting users with immersive and real-time information about flood extents, depths, and impacts. It leverages data from global hydrological and flood routing models, augmented with localized downscaling methods to enhance the resolution of flood predictions, enabling precise and actionable insights for urban and regional planning.

The FloodVue Framework (FVF) aligns with the global flood risk assessment framework proposed by Winsemius et al. (2013), which integrates hazard, exposure, and vulnerability factors to produce high-resolution flood risk estimates. FVF uses similar methodologies, including

hydrological modeling, inundation downscaling, and socioeconomic data incorporation, to assess flood risks at both macro and micro scales.

2.2.4 Existing Augmentation Reality Application Comparison

The table below compares three existing applications which are AR Way Finder, AR Flood, and FloodVue Framework that leverage augmented reality (AR) and related technologies for disaster management, flood awareness, and urban planning. Each application serves distinct purposes, ranging from real-time evacuation guidance to flood visualization and risk assessment. The comparison highlights their primary functions, technology used, key features, geographic focus, target users, technological capabilities, offering insights into their potential use cases and contributions to disaster preparedness and mitigation.

Feature/Aspect	AR Way Finder	AR Flood	FloodVue
Primary Function	Evacuation guidance through augmented reality.	Visualization of flood data with 3D renderings.	Interactive visualization of flood data for infrastructure planning and public awareness.
Technology Used	Augmented Reality (AR) with smartphone camera and GPS.	3D rendering technology integrated with AR.	Augmented Reality combined with hydrological and flood routing models.
Key Features	Overlays digital directions (e.g., arrows, signs) onto real-world views for evacuation guidance.	Allow users to select points of interest and displays flood-prone areas with 3D maps.	Simulates flood scenarios with detailed information on extents, depths, and impacts.
Data Integration	Basic GPS and AR mapping for real-time route guidance.	Dynamic maps that update with selected locations' flood data.	Uses global hydrological data, flood-routing models, and localized downscaling methods.

Geographic Focus	Indonesia.	General/global, adaptable to various locations.	Global applications with a focus on localized data for urban and regional planning.
Target Users	Individuals and communities needing evacuation assistance during disasters, particularly in Indonesia.	General public, policymakers, and planners for flood awareness and preparedness.	Urban and regional planners, policymakers, and the public for strategic flood management.
Available On	Google Play (Android).	Apple App Store (iOS).	Specialized tool for professionals; not available for the public to download.

Table 2.1: Comparison between AR Way Finder, AR Flood, FloodVue Framework

2.3 Map technologies

Mapping technologies play a vital role in emergency response by providing accurate and timely spatial data. Google Maps, HERE Maps, and topography maps are three key platforms that offer diverse capabilities relevant to flood evacuation systems.

Google Maps: Known for its extensive road network data and real-time traffic updates, Google Maps is widely used for navigation. Its AR overlay feature enhances pedestrian and vehicle navigation, making it a valuable tool for real-time evacuation guidance.

HERE Maps: Recognized for its offline functionality and customizable APIs, HERE Maps is highly effective in areas with limited connectivity. It supports high-definition mapping and dynamic hazard visualization through AR integration, making it suitable for rapidly changing environments.

Topography Maps: These maps provide critical data on elevation, slopes, and terrain, which are essential for identifying flood-prone areas and safe high-ground locations. Their integration with AR enables dynamic visualization of risk zones and escape routes.

2.3.1 Google Map

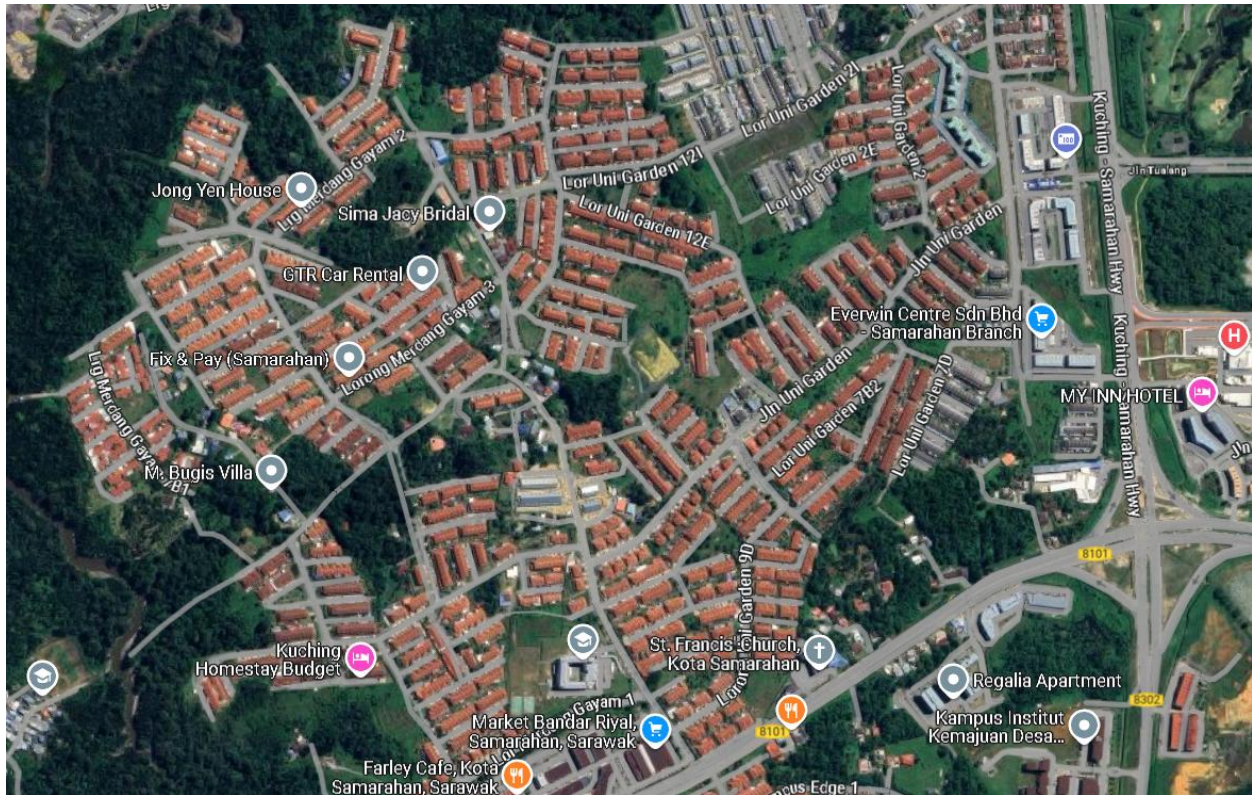


Figure 2.4: Google map view of Uni Garden area

Google Maps is one of the most widely used navigation platforms due to its robust and reliable mapping capabilities. It offers features such as real-time traffic updates, satellite imagery, and geolocation services, which are crucial for effective route planning. In disaster scenarios, Google Maps can provide detailed road network data and identify the fastest and safest evacuation routes. Its integration with augmented reality (AR) has been explored in applications like pedestrian navigation, where users receive real-time AR overlays on their screens for guidance. This functionality can be adapted to flood evacuation systems, allowing victims to see visualized escape routes layered onto their immediate surroundings, thereby enhancing situational awareness.

Existing studies have demonstrated the potential of Google Maps in disaster preparedness and evacuation. Sonwane (2014) developed a GPS-integrated disaster management system using Google Maps to track a user's location and provide shortest-path guidance to shelters during emergencies. Their system emphasized real-time functionality and was successfully tested in simulated flood scenarios. Similarly, Soni et al. (2014) proposed an early disaster warning and evacuation system using Google Street Map, leveraging cloud messaging to deliver alerts and

evacuation routes to users. Their use of ray-casting algorithms to identify disaster-prone areas further validated the system's accuracy and reliability.

These implementations highlight how geolocation technologies like Google Maps can be enhanced with AR for improved disaster response. For instance, Sonwane's GPS-guided evacuation system can serve as a foundation for developing AR visualizations that overlay escape routes and safe zones directly onto a user's environment. Soni et al.'s integration of cloud-based notifications and real-time tracking can complement AR systems by ensuring victims receive timely updates about danger zones and evacuation points.

By incorporating these proven methodologies, AR-assisted flood evacuation systems can combine the mapping precision of Google Maps with immersive AR overlays to create a more intuitive and effective disaster management tool. This would not only enhance situational awareness but also streamline evacuation processes, ultimately saving lives during emergencies.

2.3.2 Here Map

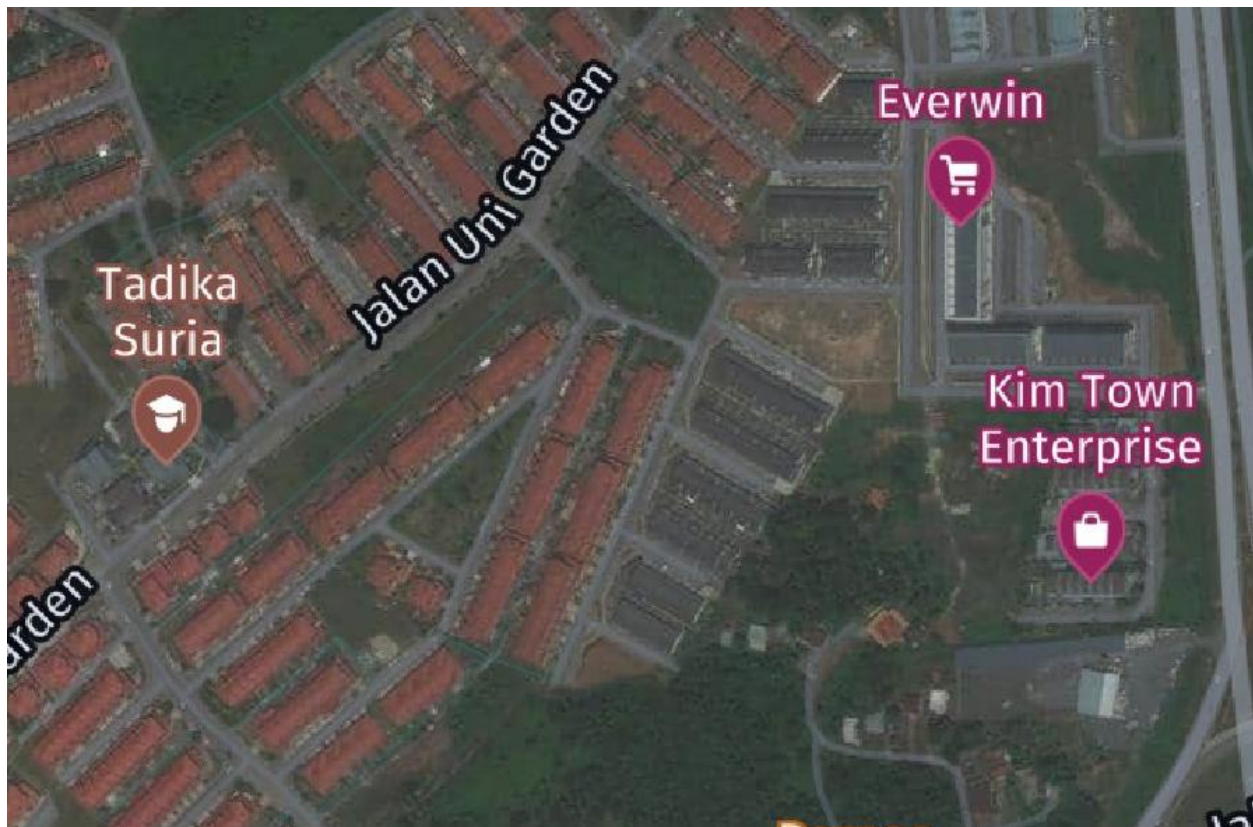


Figure 2.5: Here map view of Uni Garden area

HERE Maps, known for its precision and scalability, is another significant mapping platform that supports emergency management applications. It excels in offline functionality, an essential feature during disasters when internet access might be disrupted. HERE Maps provide high-definition mapping, customizable APIs, and advanced route planning, which can be tailored to include flood-prone areas and evacuation paths. The platform's integration with AR technology enables overlaying real-time hazard data onto maps, offering an intuitive interface for users. Additionally, its ability to process geospatial data in real time makes it suitable for dynamic environments like flood zones, where conditions can change rapidly.

2.3.3 Topography Map



Figure 2.6: Topology map view for Uni Garden area

Topography maps provide a detailed representation of land features, including elevation, terrain, and slopes, which are critical in understanding flood dynamics. During heavy rainfall or flooding, areas with low elevation or poor drainage are most vulnerable. By incorporating topographic data into an AR-assisted evacuation system, users can visualize potential water flow patterns, areas prone to inundation, and safe high-ground locations. These maps enhance route planning by accounting for elevation changes, ensuring that evacuation paths lead to safer, elevated

areas. When combined with AR technology, topography maps can dynamically highlight risk zones and safe zones in real time, enabling more informed decision-making during emergencies.

2.3.4 Mapping Comparison

The comparison highlights the unique strengths of Google Maps, HERE Maps, and Topography Maps in supporting an AR-assisted flood evacuation system. Google Maps excels in real-time traffic updates and urban navigation, relying on internet connectivity for optimal use. HERE Maps provides offline capabilities, customizable APIs, and real-time hazard data visualization, making it ideal for emergencies in low-connectivity areas. Topography Maps, focused on elevation and terrain details, are essential for identifying flood-prone and safe high-ground areas. Integrating these technologies with AR enhances their usability, providing adaptive and informed evacuation solutions during floods

Feature	Google Maps	HERE Maps	Topography Maps
Strengths	Real-time traffic updates, satellite imagery, and extensive road network data.	Offline functionality, high-definition mapping, and customizable APIs.	Detailed representation of terrain, elevation, and slopes.
AR Integration	AR overlays for real-time navigation and route guidance.	AR overlays with real-time hazard data visualization.	AR-enhanced visualization of flood risk zones and safe areas.
Internet Dependency	Requires a stable internet connection for optimal performance.	Operates efficiently in offline mode.	Typically offline but requires integration with other systems for AR visualization.
Relevance in Floods	Identifies real-time evacuation routes based on road network and traffic conditions.	Provides adaptive guidance in areas with poor connectivity.	Highlights flood-prone areas and safe high-ground zones based on terrain.
Customization	Limited by predefined features; customization possible via APIs.	Highly customizable via APIs for tailored solutions.	Requires integration with other systems for enhanced usability.

Table 2.2: Comparison between Google Maps, Here Maps and Topography Maps

2.4 AR Development Frameworks

2.4.1 ARCore (Android) / ARKit (iOS)

ARCore and ARKit rely on advanced capabilities to create seamless augmented reality experiences by leveraging the hardware and software features of mobile devices. Both frameworks use motion tracking, which combines data from the device's inertial measurement units (IMU), such as accelerometers and gyroscopes, with visual input from the camera. This integration, known as visual-inertial odometry (VIO), allows the device to accurately estimate its position and orientation in the environment, even without GPS or external markers. Additionally, these frameworks excel in environmental understanding, enabling devices to detect and map real-world features like horizontal and vertical surfaces. This functionality supports the precise placement of virtual objects in a way that aligns with the physical environment. Light estimation further enhances the realism of augmented reality by analyzing ambient lighting conditions and adjusting the appearance of virtual elements to match the real world. As Cervenak et al. (2019) emphasize, ARKit, for instance, can function as an indoor positioning system, showcasing its potential for precise spatial awareness. Through these combined technologies, ARCore and ARKit provide the tools necessary for developers to create applications that integrate virtual content seamlessly into physical spaces, opening up possibilities for a wide range of practical uses, including navigation, education, and emergency response systems.

2.4.2 Unity with Vuforia

Unity, a cross-platform development tool, and Vuforia, together provide powerful solutions for developing immersive augmented reality applications. Unity serves as the foundational platform where developers create, manage, and deploy AR experiences across various devices and operating systems. It provides a comprehensive set of tools, including a robust editor, asset management, and scripting capabilities, enabling developers to design highly interactive and visually appealing AR environments. Vuforia enhances this process by introducing specialized AR functionalities, such as real-time object tracking, target detection, and the ability to recognize and interact with predefined markers or 3D objects. According to Centria (2018), the integration of Vuforia with Unity streamlines the creation of interactive applications, supporting features such as real-time object tracking, target detection, and cross-platform deployment. This integration enables seamless workflow management, allowing developers to incorporate virtual elements that interact dynamically with the real world. Together, Unity and Vuforia empower the development of AR-

assisted systems for flood evacuation by enabling the visualization of critical data, such as water levels and evacuation routes, in real-time. These capabilities enhance situational awareness, facilitate better decision-making, and ensure that users receive accurate and actionable information during emergencies.

2.4.3 Unreal Engine with AR

Unreal Engine offers another robust platform for developing AR applications, renowned for its exceptional graphics capabilities and flexibility in creating high-quality immersive experiences. This powerful engine supports AR development for both Android and iOS through seamless integration with ARCore and ARKit, enabling developers to harness the full potential of these frameworks. Unreal Engine's Blueprint visual scripting system simplifies the development process, allowing creators to design complex interactions and AR behaviors without extensive coding knowledge. Its advanced rendering engine is capable of producing photorealistic 3D visualizations, bringing virtual environments to life with highly detailed textures, dynamic lighting, and realistic animations. As McIntosh (2023) highlights, Unreal's advanced rendering engine can create realistic 3D visualizations of flood-affected areas, including detailed evacuation routes, flood zones, and emergency shelters. By leveraging Unreal's robust capabilities, developers can craft immersive AR experiences that provide users with an intuitive and engaging way to interpret critical information. These visualizations are essential for enhancing situational awareness, supporting effective evacuation planning, and ensuring that vital data is communicated clearly and interactively during emergencies.

2.4.4 AR.js

AR.js is a lightweight, open-source library that enables augmented reality (AR) experiences directly in web browsers, eliminating the need for additional software or applications. Built on top of Three.js, a widely used JavaScript library for 3D rendering, AR.js supports both marker-based and markerless AR functionalities, making it a versatile solution for developers. According to Nitika et al. (2021), WebAR frameworks like AR.js leverage technologies such as WebRTC and WebGL to enhance interactivity and rendering performance, ensuring smooth and responsive AR experiences in browsers. These technologies enable efficient resource utilization, even on devices with limited computational power, thus broadening the accessibility of AR applications. AR.js integrates seamlessly with standard web technologies like HTML, CSS, and

JavaScript, allowing developers to create interactive AR applications with ease. It also supports A-Frame, simplifying the development of immersive AR scenes through an intuitive HTML-like syntax. Additionally, AR.js harnesses the WebXR API to ensure compatibility with modern browsers, enabling users to access AR content directly by visiting a web page. This combination of lightweight performance, cross-platform accessibility, and the ability to run on low-end devices makes AR.js particularly suitable for applications focused on delivering inclusive, real-time AR experiences. The work by Nitika et al. (2021) underscores the growing potential of WebAR as a powerful tool for making AR technology accessible to wider audiences, further enhancing its utility for diverse use cases.

2.4.5 Framework Development Comparison

Feature	ARCore (Android) / ARKit (iOS)	Unity with Vuforia	Unreal Engine with AR	AR.js
Platform	Android (ARCore), iOS (ARKit)	Cross-platform (Android, iOS, Windows)	Cross-platform (Android, iOS, Windows)	Web-based (Cross-platform)
Key Features	Motion tracking, environmental understanding, light estimation, real-time data overlays	3D model integration, object recognition, image tracking, advanced rendering	High-definition graphics, ARKit and ARCore support, real-time rendering, advanced physics and lighting	Marker-based AR, JavaScript library for browser-based AR experiences, lightweight
Best Use Case for Flood Assistance	Mapping flood zones, detecting safe evacuation routes, real-time flood data overlays	Creating interactive AR maps, visualizing evacuation routes, flood zones, and emergency shelters	High-quality visualizations of flood-affected areas, evacuation routes, and flood warnings with realistic 3D renderings	Lightweight AR for web-based access to flood assistance information such as evacuation routes, shelter locations, and alerts

Table 2.3: Comparison between Framework Development

2.5 Conclusion

To summarize, the integration of augmented reality (AR) into flood evacuation systems can significantly enhance disaster response by improving situational awareness, real-time navigation, and decision-making capabilities. However, without a customized approach tailored to the unique challenges of flood-prone areas, existing technologies may fall short in addressing the critical needs of affected communities. From the analysis, many existing AR applications and frameworks have demonstrated innovative features that enhance usability and engagement. The next chapter will focus on the implementation of the proposed AR-assisted evacuation solution.

Chapter 3

Methodology

3.1 Introduction

This chapter describes the methodology adopted for the development of the Augmented Reality-Assisted Evacuation System for Flood Victims. The chosen approach is Rapid Application Development (RAD) which emphasizes iterative development and user feedback to ensure the system is both efficient and user-friendly.

The chapter begins by detailing the principles of RAD and its suitability for this project. Subsequently, it elaborates on the research design, outlining the structured methods used to design, and developing the system. The development and implementation sections highlight the tools, technologies, and processes employed to create the AR-assisted application.

Moreover, the testing phase ensures the system meets its objectives by evaluating usability, accuracy, and effectiveness through user feedback and simulated scenarios. Lastly, the system flow diagram provides a clear representation of how data flows through the application, ensuring transparency and traceability in the methodology.

This structured approach guarantees the development of a robust, practical, and scalable solution to address the critical issue of flood evacuation in Kota Samarahan, Sarawak.

3.2 Rapid Application Development (RAD)

Rapid Application Development (RAD) is a dynamic software development methodology that emphasizes quick prototyping, iterative design, and active user participation. Its structured approach is divided into four distinct phases: Requirements Analysis, User Design, Rapid Construction, and Cutover.

The Requirements Planning phase serves as the foundation of the project. In this stage, developers and the system end-users collaborate to define the system's needs. Unlike traditional methodologies that may spend extensive time on planning, RAD condenses this phase to expedite development.

Following by user design phase that involves the creation of prototypes and models that represent the desired system functionalities. During this interactive process, users engage directly

with developers to provide feedback on design elements, ensuring the evolving system aligns with their needs and preferences. This continuous user involvement facilitates the development of a user-centric system that is both intuitive and practical.

The Rapid Construction phase is where the actual development takes place. Building upon the prototypes and user feedback from the previous phase, developers construct the system's components using efficient coding practices and development tools. This phase is characterized by iterative cycles of development, testing, and refinement, allowing for the swift incorporation of changes and enhancements.

Finally, the Cutover phase involves the transition from development to full-scale deployment. This includes final testing, user training, data conversion, and the implementation of the system into the live environment. The goal is to ensure a seamless transition with minimal disruption, allowing users to quickly adapt to the new system and begin leveraging its functionalities effectively.

By progressing through these phases, RAD facilitates the development of robust systems that are responsive to user needs and adaptable to changing requirements, making it particularly suitable for projects requiring rapid delivery and flexibility.

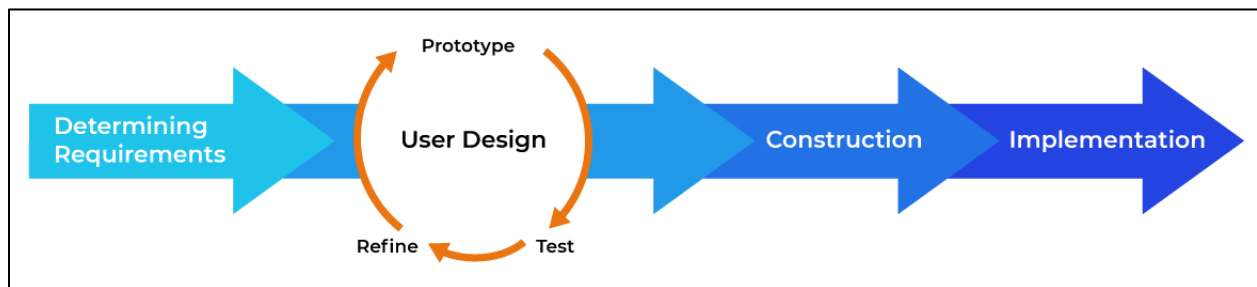


Figure 3.1: Rapid application development model

3.2.1 Determining Requirements

The requirement for this project focuses on developing and evaluating an Augmented Reality (AR) application to assist flood victims in real-time evacuation. The design adopts a developmental and evaluative approach, emphasizing user-centered functionality and iterative refinement to ensure the system is both effective and practical.

To gather User Requirements, a survey was conducted targeting students from UNIMAS and residents of Uni Garden. A total of 30 participants were involved, representing a balanced demographic. The objectives of the survey were twofold: to determine the prevalence and impact of flooding on individuals and to collect valuable insights for the development of the AR-assisted evacuation system. This survey findings play a crucial role in guiding the design and functionality of the application, ensuring it effectively addresses the specific challenges and needs of users. By incorporating these insights, the system aims to be a practical, reliable, and user-focused tool for managing flood emergencies.

3.2.2 User Requirement Analysis

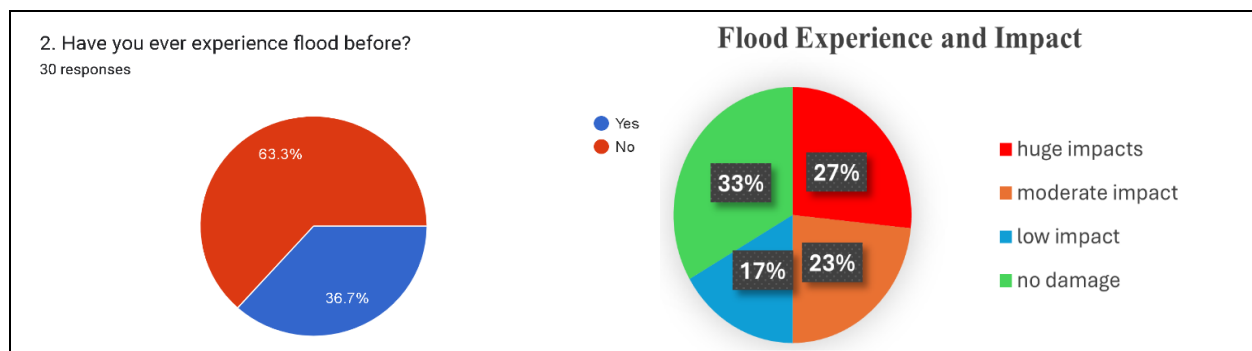


Figure 3.2: Chart of respondents Who Have Experienced Floods and its impact

The chart shows the proportion of respondents who have experienced floods. Out of 30 respondents, 20 individuals (66.67%) reported experiencing floods, while 10 respondents (33.33%) stated they had not. Among the 20 respondents who experienced floods, 8 respondents (40%) reported severe impacts, such as significant property damage or displacement, while 7 respondents (35%) stated the flood had a moderate impact, including minor property damage or temporary disruption. Additionally, 5 respondents (25%) indicated a low impact, experiencing minimal disruption or inconvenience.

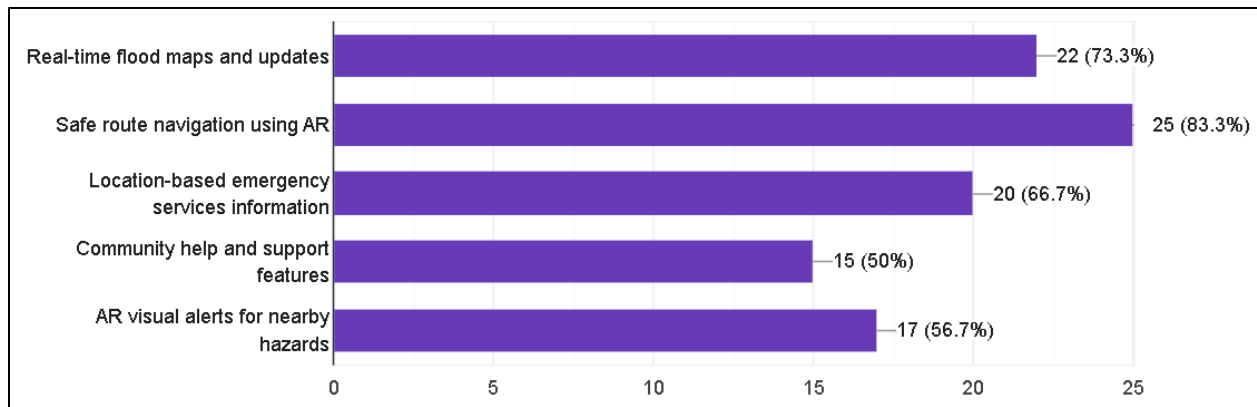


Figure 3.3: Chart of most desired features in app for user

This chart shows the most wanted features by respondents in an AR-assisted flood app. Real-time flood maps and safe route navigation were the most selected features, chosen by 22 respondents (73.33%) and 25 respondents (83.3%) which shows the importance of immediate assistance for avoiding danger and finding safe evacuation routes during floods. Location-based emergency services information was selected by 20 respondents (66.7%) while community help, and support features were the least chosen with only 15 respondents (50%).

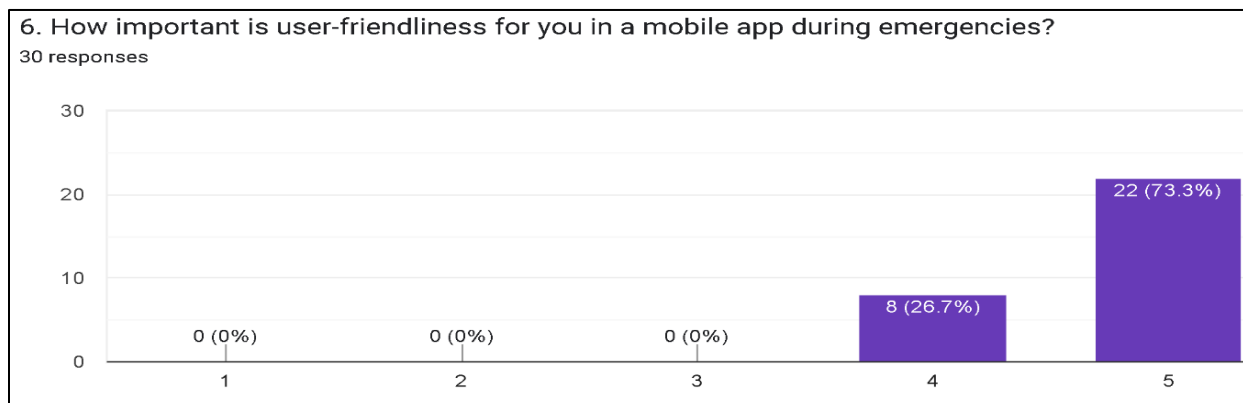


Figure 3.4: Chart of the importance of user-friendliness and simplicity in an emergency app

Respondents emphasized the need for a user-friendly and simple app during emergencies. Key preferences include features that allow users to share their flood-handling experiences, easy navigation with a clean user interface, and straightforward access to essential information, such as tips on flood preparedness. These insights highlight the importance of designing an app that is intuitive and accessible, even under stressful conditions.

3.2.3 Use Case Diagram

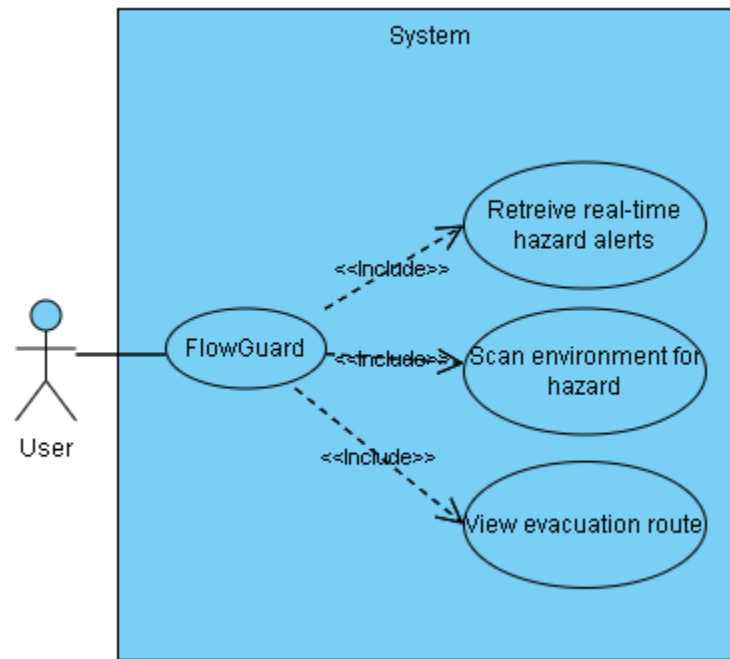


Figure 3.5: Use case diagram of Floodguard

The use case diagram represents the interaction between the Floodguard application, and its primary actor is app User. The User utilizes the Floodguard application to access various safety-related functionalities, including viewing evacuation routes, receiving real-time hazard alerts, and scanning the surroundings for potential hazards. The <<include>> relationships indicate that these functionalities are integral components of the Floodguard system, ensuring that users can access all necessary safety features seamlessly.

3.2.4 Sequence Diagram

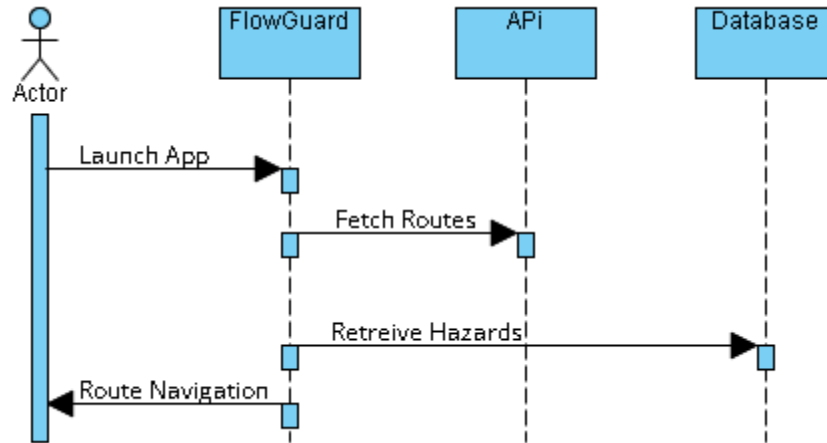


Figure 3.6: Sequence diagram of Floodguard

The sequence diagram illustrates the step-by-step interaction between the User, the Floodguard application, external APIs, and the database during route navigation. It begins with the User launching the application, prompting the Floodguard application to communicate with external APIs to fetch the safest route. The system then retrieves real-time hazard data from the database and overlays this information on the fetched route. Finally, the system displays the AR navigation overlay to the User, allowing them to follow the suggested route while being aware of potential hazards. This sequence ensures efficient coordination between the system components to provide users with a reliable and safe navigation experience.

3.2.5 Activity diagram

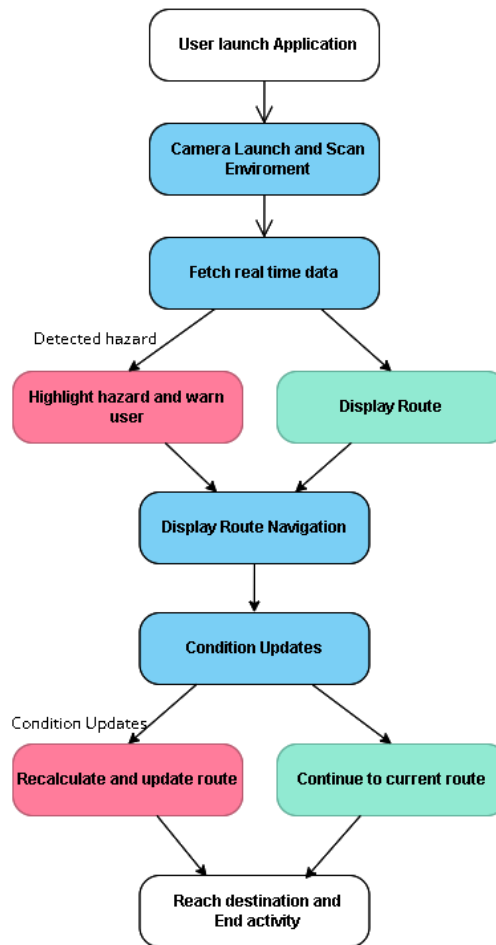


Figure 3.7: Floodguard Activity Diagram

The activity begins when the user launches the application, which activates the camera to scan the environment for hazards. Simultaneously, the system fetches real-time data to identify potential risks and generate safe evacuation routes.

Once a hazard is detected, the system either highlights the danger and warns the user or directly displays a safe route to follow. The route is visualized through augmented reality overlays, guiding the user to an emergency center. During navigation, the application continuously monitors environmental conditions in real time. If changes are detected that interrupt the current route, the system recalculates and provides an updated path to ensure the user's safety. In the absence of any significant changes, the user continues the current route.

The process concludes successfully when the user reaches the designated safe destination, marking the end of the evacuation activity. This dynamic and interactive system ensures that users receive up-to-date and accurate guidance throughout the evacuation process.

3.2.6 System Flow Diagram

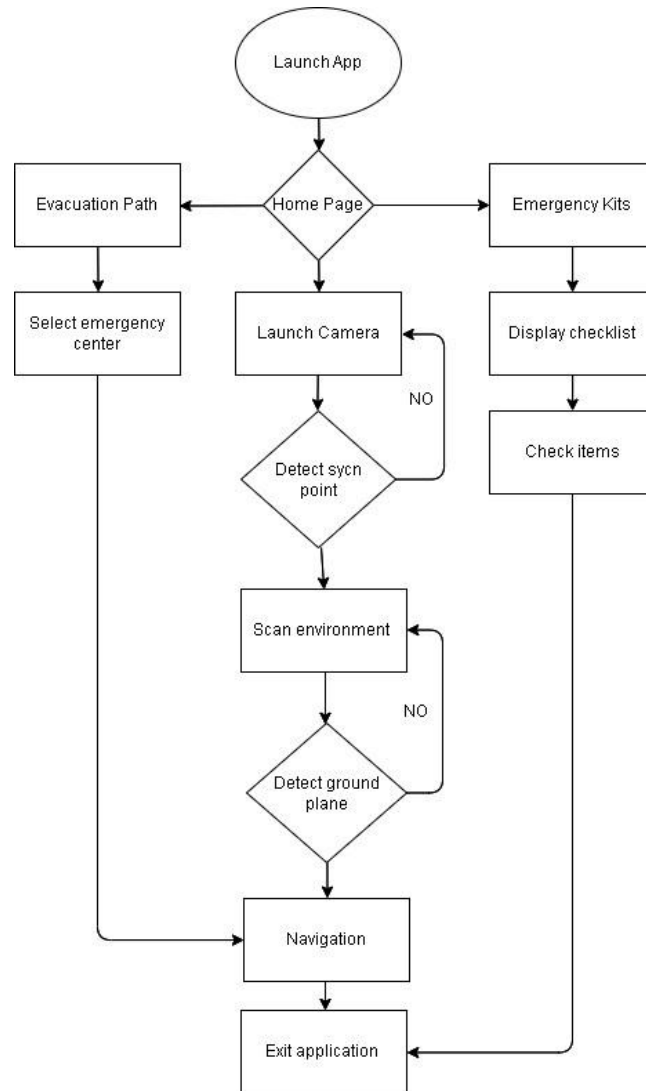


Figure 3.8: Floodguard System Flow Diagram

The system begins with the user launching the application, which takes them to the home page. From the home page, the user is presented with three options: Evacuation Path, Emergency Kits, and Launch Camera. If the user selects the Evacuation Path option, they are guided to choose an emergency center, after which the system navigates them to the selected location. Alternatively,

if the user selects the Emergency Kits option, the system displays a checklist of essential items for emergencies, allowing the user to review and check off items from the list.

If the user opts for the Launch Camera feature, the system attempts to detect a synchronization point in the environment. If the sync point is not detected, the system loops until successful detection. Once the sync point is identified, the system scans the environment and proceeds to detect the ground plane. If the ground plane is not detected, the system loops back to scanning the environment until successful detection. After the ground plane is detected, the system transitions to the Navigation feature, providing guidance and directions for the user. Finally, the user can exit the application upon completing their tasks.

3.3 Development

The development phase focuses on building the core components of the Augmented Reality-Assisted Evacuation System based on the requirements and design established earlier. This involves creating the AR modules, integrating mapping APIs, and developing the backend infrastructure.

3.4 Prototype

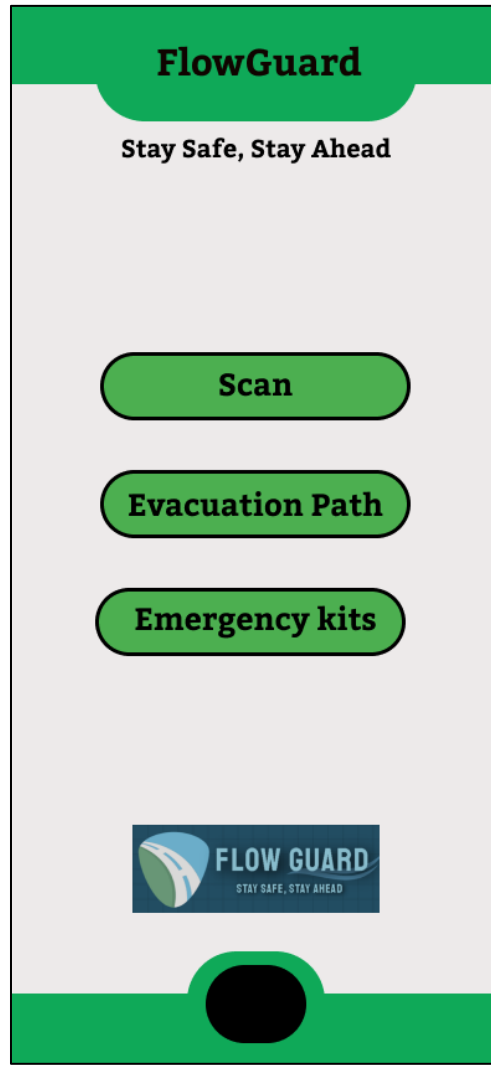


Figure 3.9: Home page for Floodguard Application

The Home Page serves as the main interface of the application, where users can access its core functionalities. It features three main options: "Scan," which allows users to scan surroundings to obtain evacuation-related information; "Evacuation Path," which guides users to the nearest evacuation center using both augmented reality (AR) and map navigation; and "Emergency Kits," which provides a checklist of essential items to ensure users are prepared for emergencies. This page acts as the starting point for users to explore and utilize the application effectively. The design applied multimedia coherence principle which avoids unnecessary elements, focusing on essential features such as "Scan," "Evacuation Path," and "Emergency Kits," ensuring users are not overwhelmed with irrelevant information during emergencies.

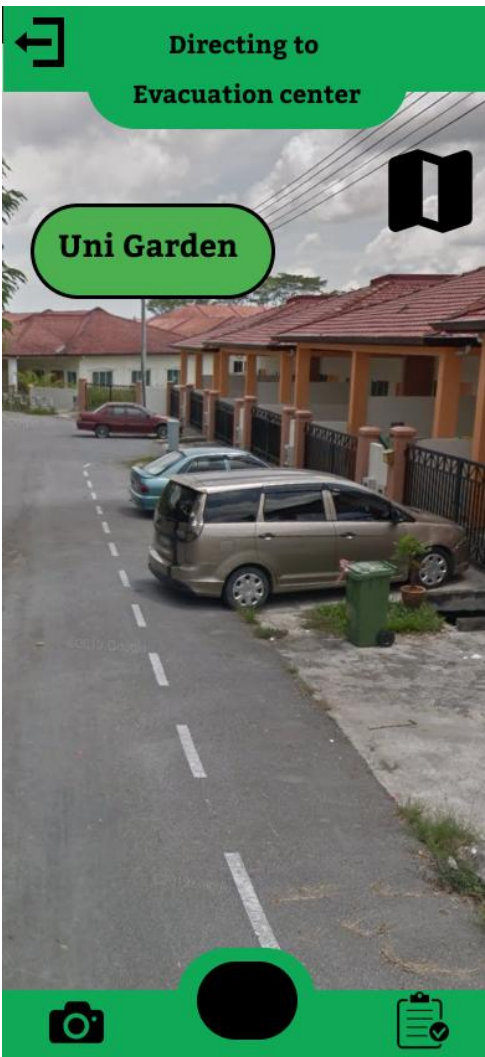


Figure 3.10: Evacuation Path for Floodguard Application

The Evacuation Path page utilizes an AR interface to guide users to the evacuation center. It displays the user's current location overlaid with directional arrows on a real-world view, ensuring intuitive navigation. The page shows the remaining distance, helping users track their progress. This page applies visual elements like AR-based directions (e.g., "20m" label) enhance user understanding by combining graphics with minimal text. This offers users clear, and real-time directions without overloading text.

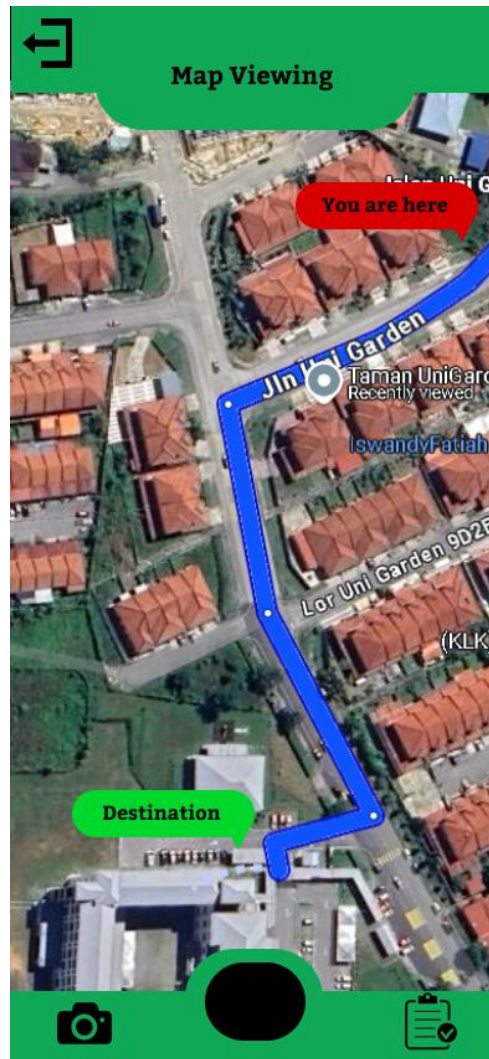


Figure 3.11: Map View for Floodguard Application

The Map View Page offers users an alternative bird's-eye view of the route to the selected evacuation center. It displays a detailed map with the user's current location and destination clearly marked. A highlighted route guides the user step-by-step to their evacuation center, while a "You are here" marker ensures easy orientation. This page complements the AR navigation by providing a more traditional navigation option, catering to diverse user preferences.

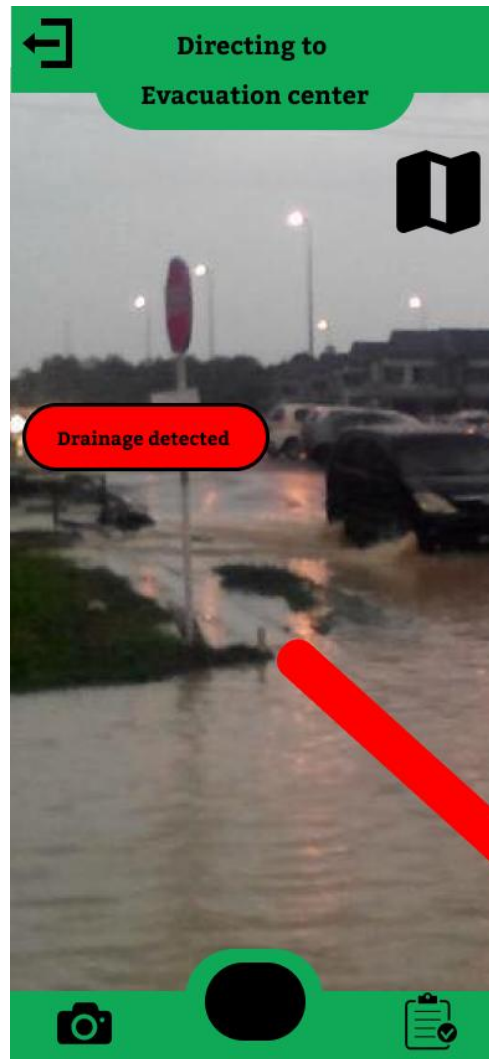


Figure 3.12: Detected Hazards

When the system scans the environment and detects a potential hazard, such as an overflowing or blocked drainage, it highlights the area with a red box. The use of the red box is deliberate, as red is universally recognized as a color that signifies danger or caution, ensuring it immediately captures the user's attention. This visual alert informs the user to avoid the marked hazard, minimizing the risk of injury or unsafe navigation. By clearly indicating the presence of danger, the system proactively prevents users from entering hazardous zones.



Figure 3.13: Reach Destination

Once the user successfully reaches the designated safe location, the system provides a confirmation message displayed prominently on the screen, such as “You have reached destination.” This feature ensures users are aware of their successful arrival at the evacuation center or safe zone, completing the journey with a sense of clarity and assurance.

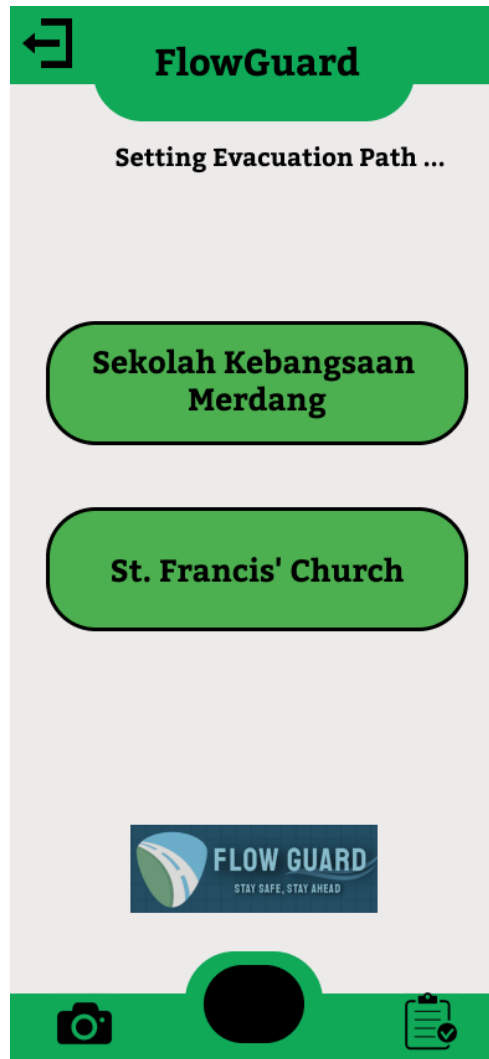


Figure 3.14: Destination Selection to safety point

The Selection Page allows users to choose their preferred evacuation center from a list of available options. It displays the names of evacuation centers, such as local schools or churches, and enables users to select a destination by tapping on the respective button. Once a center is selected, the application sets up navigation to that location. This page simplifies decision-making by presenting clear options and initiating the evacuation process smoothly.



Figure 3.15: Loading view page

The Loading Page is displayed after the user selects an evacuation center and serves to initialize the evacuation path. It features a loading message, such as "Setting Evacuation Path..." accompanied by the name of the selected center. During this time, the application processes navigation details and prepares both the AR and map navigation systems. This page reassures users that the application is actively working to provide the best route to safety.



Figure 3.16: Emergency Kits checklist page

People learn better when words are combined with relevant images, rather than using text alone. In this checklist page, it ensures users are fully prepared for emergencies by displaying a comprehensive list of essential items, serves as a practical reminder for users to verify their emergency kit, helping them stay equipped during critical situations.

3.3 Implementation

All the prototypes are implemented into a complete system in mobile application named Floodguard.

3.4 Software limitations

Software Component	Requirement	Limitation
Internet Connectivity	Stable internet connection for real-time data fetching and updates.	Dependence on stable internet for fetching real-time mapping and hazard data.
AR Framework Compatibility	ARKit (iOS 11+), ARCore (Android 7.0+), Vuforia, Unreal Engine for AR.	Compatibility issues with older devices or unsupported AR frameworks.
Data Processing Efficiency	Devices with high computational power (modern processors and GPUs).	Slower performance on devices with limited computational power when handling large datasets.

Table 3.1: Software Limitation for the system

The Floodguard system relies on various software components, including mapping APIs (e.g., Google Maps, HERE Maps), augmented reality frameworks (ARCore or ARKit), and backend services for data processing. A key limitation lies in the dependency on reliable internet connectivity to fetch real-time mapping and hazard data. In flood-prone areas, network disruptions are common, which may hinder the application's ability to provide timely updates. Furthermore, integrating AR technology with mapping APIs can lead to compatibility issues, especially when managing large datasets or handling complex overlays. Limited computational efficiency in older devices can also slow down AR rendering, reducing the application's responsiveness and accuracy. These factors may impact on the system's real-time performance and user experience in critical situations.

3.5 Hardware limitations

Hardware Component	Requirement	Limitation
Processor (CPU)	Intel Core i5 and above	Older processors may struggle with real-time AR rendering and dynamic data updates.
Memory (RAM)	8GB RAM and above	Devices with less than 8GB RAM may experience delays in data handling and application responsiveness.
Graphics (GPU)	4GB memory and above	Devices without sufficient GPU power may fail to display AR visualizations smoothly.
Battery Life	Long-lasting battery	High battery consumption due to GPS, AR, and camera usage may limit the system's usability over time.

Table 3.2: Hardware Limitation for the system

The Floodguard system requires modern hardware capabilities, including high-performance cameras, GPS modules, and processors, to support AR features and real-time data processing. A primary limitation is the dependence on devices with AR-enabled hardware, such as specific Android or iOS models that support ARCore or ARKit. In areas where users may rely on older smartphones, limited AR compatibility can restrict access to the application. Additionally, high battery consumption caused by continuous GPS tracking, camera usage, and data processing can pose challenges during prolonged evacuation scenarios. Environmental factors, such as poor lighting or water damage to the hardware during floods, may further degrade the system's functionality, reducing its reliability in emergency conditions.

3.6 Testing

Technical evaluation and refine will be conducted until the application reached the acceptable performance level by user. 10 users who are new, unfamiliar to the UniGarden environment users will be selected randomly to try the Floodguard. After that, the user will answer the Brooke (1986) System Usability Scale (SUS) questionnaire. It consists of 10 Likert scale questions scaling from strongly disagree to strongly agree (see Appendix A). It is selected as a testing tool due to its robustness and rapidness in providing a reliable result on the usability of the mobile app.

3.7 Conclusion

This chapter outlines the development of the AR-assisted flood evacuation system, utilizing the Rapid Application Development (RAD) approach. The methodology combines iterative prototyping, user feedback, and structured design to ensure a user-centric and practical solution. By leveraging tools such as use case diagrams, sequence diagrams, and prototypes, the system effectively addresses the need for real-time navigation and situational awareness during flood emergencies. A mobile application prototype, Floodguard, was developed to demonstrate the proposed system's capabilities. The next chapter will focus on the technical implementation of the key components discussed in this chapter, including AR integration and real-time data processing.

Chapter 4

Implementation

4.1 Introduction

This chapter gives more information about how the AR navigation system was developed and implemented. This section starts with a description of the infrastructure of the system, which is referred to as the backend, frontend, and support technologies that are used. Some of the technologies used throughout the chapter are Unity and Cesium. After describing the infrastructure, it goes on to describe the process of developing the system. This is done through a series of key steps that include integration of maps, implementation of navigation line, and the 3D environment all built with basic blocks. In addition, this section describes some of the challenges faced when developing the system. These challenges include how to integrate into Unity and how they were solved to move forward.

4.2 System Implementation

4.2.1 Unity

Unity was chosen as the primary development platform due to its flexibility in building AR applications and its wide support for mobile deployment. Unity's AR Foundation framework was used to manage cross-platform AR features such as plane detection, tracking, and camera input. The project was developed using Unity version 6000.0.43f1, which provided compatibility with the latest AR and GPS packages.

In the AR navigation system, Unity served as the core engine for integrating real-world elements with digital overlays. Key features like real-time user location tracking, AR camera setup, prefab instantiation, and LineRenderer for navigation paths were all implemented within Unity. The use of XR Origin, AR Session, and AR Camera components made it possible to synchronize the physical environment with the digital content.

Unity's built-in NavMesh system was actively used in conjunction with a custom GPS-to-Unity world space mapping system to enable real-world navigation. The NavMesh system handled the pathfinding logic within the virtual environment, allowing agents to navigate around obstacles

and terrain. At the same time, GPS coordinates were continuously converted into Unity world positions to dynamically track the user's real location and render paths accordingly. This hybrid approach enabled both realistic in-app movement and real-world spatial alignment.

Universal Render Pipeline (URP)

The AR navigation application developed in this project utilizes Unity's Universal Render Pipeline (URP) as its rendering solution. The decision to adopt URP was based on its balance between graphical quality, performance efficiency, and compatibility across a range of mobile platforms. As the app operates on mobile devices and relies heavily on GPS and AR capabilities, performance and responsiveness are critical.

URP is designed to be highly optimized for mobile and low- to mid-range hardware, making it an ideal choice for an AR navigation context where seamless rendering and low battery consumption are required. Its streamlined architecture and forward renderer reduce GPU load, resulting in smoother frame rates and lower thermal output. This is especially advantageous for AR applications, where camera feed, GPS, and real-time navigation run concurrently and must be handled within constrained device resources.

In comparison, Unity's Built-in Render Pipeline provides a general-purpose solution that, while highly compatible, lacks the optimization and scalability benefits offered by SRP-based pipelines like URP. Meanwhile, the High Definition Render Pipeline (HDRP) delivers state-of-the-art rendering quality ideal for high-end desktop and console platforms but is too resource-intensive for mobile devices, making it unsuitable for mobile AR navigation.

Comparison of Unity's Render Pipelines:

Feature	Built-in Render Pipeline	Universal Render Pipeline (URP)	High-Definition Render Pipeline (HDRP)
Platform Target	General-purpose (PC, console, mobile)	Mobile-first and low to mid-range devices	High-end PC and console
Rendering Quality	Moderate	Good quality with optimization for mobile	State-of-the-art rendering and visual fidelity
Performance	Adequate for basic needs	Highly optimized for mobile and AR applications	Very demanding and resource intensive
Flexibility	Fixed and less customizable	Highly customizable via SRP, shader graphs, and post effects	Highly customizable for advanced effects
Best Use Case	Simple 3D projects	Mobile and AR applications where performance is critical	High-end PC or console games and simulations

Table 4.1: Comparison of Unity's Render Pipelines

By choosing URP, the AR navigation app achieves an ideal balance between graphical fidelity and performance efficiency, ensuring that it operates smoothly across mobile platforms. This allows for seamless GPS-based navigation, precise rendering of terrain and navigation lines, and an overall responsive and visually coherent experience for end-users.

Terrain



Figure 4.1: Unity - terrain system

In this project, Unity's built-in Terrain system was used to create a realistic virtual environment representing the Uni Garden area. The Terrain system allows developers to design outdoor landscapes by adjusting the ground's shape, applying textures, and enabling physical interaction through colliders. It provides an efficient way to simulate real-world terrain for applications such as AR navigation.

For this project, a custom terrain was created with dimensions and resolution tailored to match the scale of the targeted flood-prone area. Tree and grass details were disabled to optimize performance, especially for mobile use. The terrain acts as the foundation for visual rendering and pathfinding, providing a realistic ground layer that aligns with the user’s GPS-based position.

By using the Terrain component, the system is able to render an immersive scene that helps users better understand their surrounding environment during a simulated flood evacuation.

4.2.2 Cesium

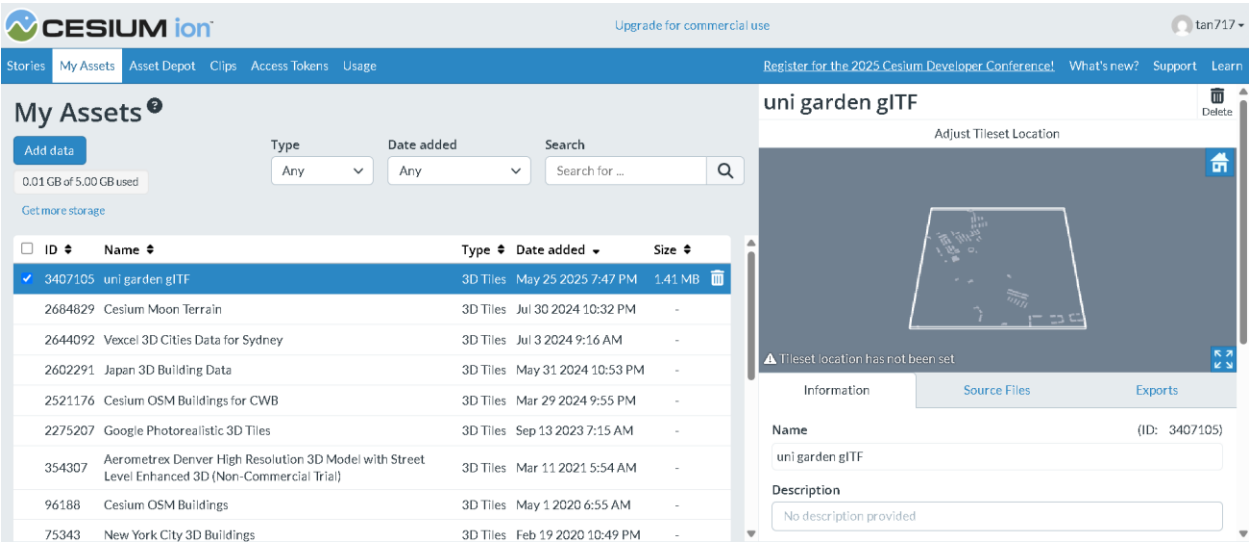


Figure 4.2: Accessing Cesium 3D Token

Cesium for Unity was used in this project to provide a geospatially accurate base map for the AR navigation system. By integrating Cesium 3D Tiles, the application could stream high-resolution terrain, satellite imagery, and building models directly into the Unity scene. This allowed for realistic spatial awareness, where virtual elements aligned meaningfully with real-world geography, enhancing the immersion and accuracy of the AR experience.

To enable this functionality, the project was configured with a Cesium Ion access token, which authenticated the Unity application and granted access to Cesium’s cloud-based asset streaming services. This token allowed the application to retrieve global 3D Tilesets, including terrain elevation, photogrammetry, and open geospatial datasets, directly from Cesium Ion. By leveraging Cesium’s real-world georeferencing and data streaming capabilities, the AR

navigation system achieved seamless integration between GPS-based positioning and photorealistic virtual environments.

Limitations Cesium 3D Tiles



Figure 4.3: Cesium 3D Tiles Limitation

The Cesium plugin was installed into the Unity environment and configured with a Cesium Ion access token to stream real-world 3D Tiles. A CesiumGeoreference component was also added to align Unity's world space with real-world coordinates (latitude, longitude, and altitude), enabling accurate placement of virtual objects in geographic space.

However, during the development phase, Cesium's 3D Tiles data proved inconsistent for navigation purposes. In particular, the geometry coverage was uneven as some regions contained detailed photogrammetric blocks, while others were incomplete or entirely missing. As illustrated in the accompanying image, the left side of the map lacks 3D structures, showing flat or empty tiles, whereas the right side contains full 3D building geometry. This discrepancy is often due to the resolution of available data or the dynamic level-of-detail (LOD) loading that prioritizes urban centers over smaller, less-mapped areas.

Such inconsistencies made it difficult to rely solely on Cesium tiles for precise AR navigation. Missing or misaligned blocks interfered with the placement of path lines, destination markers, and interactive objects, leading to potential confusion in user guidance. As a result, alternative visual aids and fallback strategies were considered to ensure a reliable user experience in low-data areas.

Solution for incomplete Cesium 3D tiles



Figure 4.4: Hybrid Solution for Unity and Cesium

To address the limitations of Cesium’s 3D Tiles, particularly in areas with missing or incomplete geometry, manually placed block models were introduced into the Unity scene. These models, built using scaled Unity primitives, were strategically placed to represent key locations such as important houses, and navigation targets. Placement was not global; rather, blocks were specifically added to Uni Garden area, where accurate spatial representation was essential for testing and demonstration. As shown in the map, only the Uni Garden region is populated with manual block models, while the surrounding areas remain uncovered.

These manually positioned blocks became critical references for the AR navigation system. They were used to define anchor points for route lines, guide agent movement, and support pathfinding logic in places where Cesium's data was unreliable or absent. This hybrid approach leveraging Cesium for global terrain and georeferencing, and Unity blocks for structural fidelity and logic offered both flexibility and control. It improved the system's overall stability and ensured that navigation paths, destination markers, and environment-aware features functioned reliably within the critical Uni Garden area.

Sub-Scene

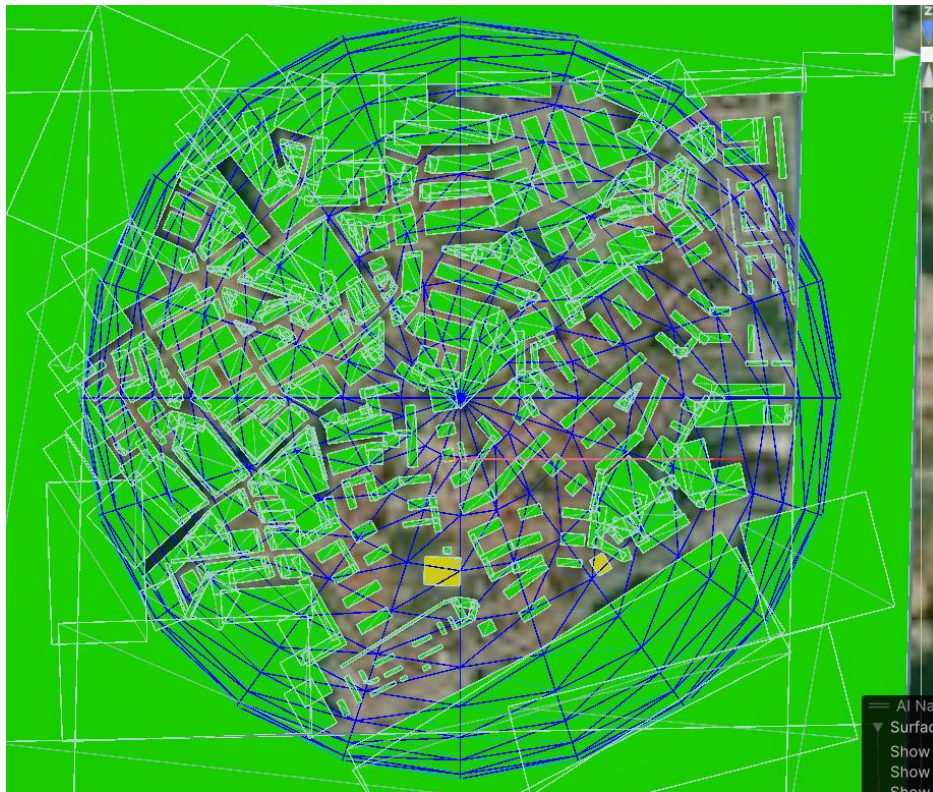


Figure 4.5: Sub Scene Feature in Cesium

Sub Scenes were utilized to manage and optimize the scene structure within Unity. A Sub Scene is a separate scene file embedded within a main scene, allowing parts of the environment or content to be organized modularly and loaded or unloaded at runtime. This approach is especially valuable when working with large-scale geospatial data, as provided by Cesium for Unity. By placing Cesium-related components such as Cesium3DTilesets, terrain, or georeferenced objects inside a Sub Scene, I was able to streamline the editor workflow, reduce main scene complexity, and support dynamic loading. Additionally, Sub Scenes improve performance in the Unity Editor by

enabling selective editing and better memory management, which is crucial when dealing with real-world, high-resolution spatial data.

4.3 Key Features

4.3.1 Coordination (Latitude, Longitude)

GPS-based navigation system was integrated into Unity to enable both real-time and manual location-based interactions. The system captures live GPS data (latitude and longitude) from the device to determine the user's current real-world position. Alternatively, editor can also input coordinates manually, allowing for flexible testing and simulation.

These geographic coordinates are then converted into Unity world space, enabling precise placement and movement of key components such as the AR Camera and the player avatar. As a result, the player or camera can be automatically positioned at any walkable real-world location inside the virtual 3D scene. This functionality allows the application to simulate or reflect real-world navigation accurately, making it particularly suitable for location-based AR experiences. The system also supports dynamic updates, ensuring the agent and camera smoothly transitions to new locations when the GPS position changes in real time.

4.3.2 Calculation

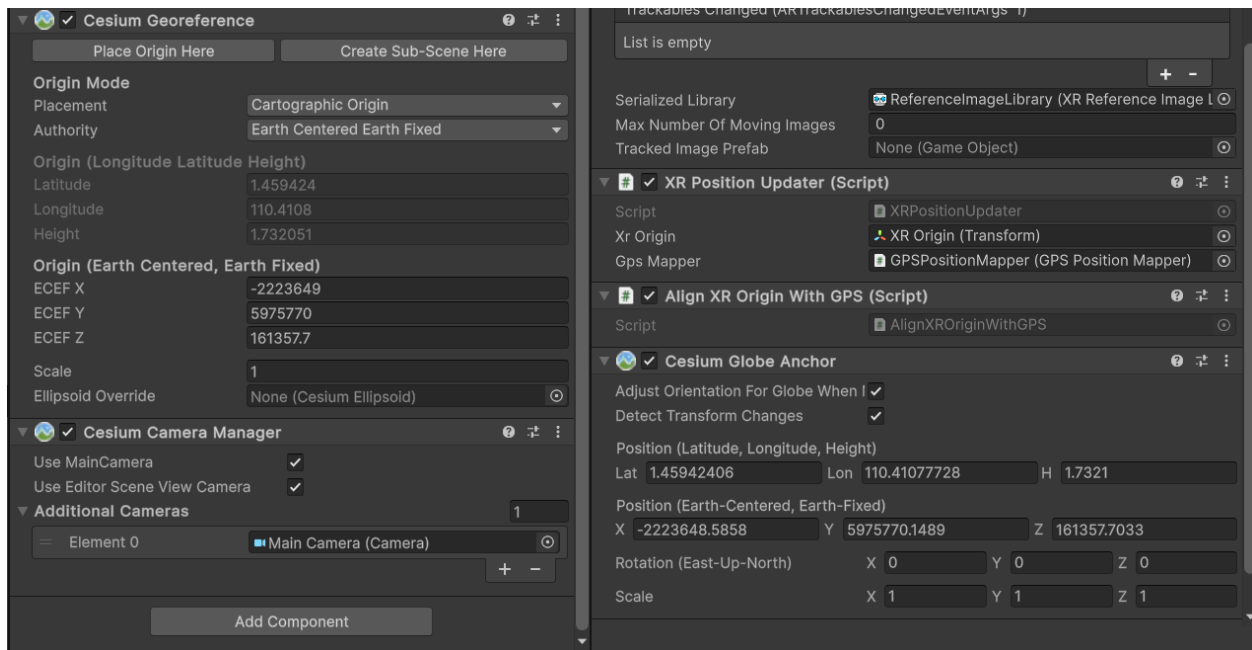


Figure 4.6: Origin position in CesiumGeoreference

In the GPS-to-Unity conversion process, the flow begins with acquiring real-world geographic data specifically latitude, longitude, and height either through real-time GPS access or manual input. Once obtained, these coordinates are passed to the CesiumGlobeAnchor component, which acts as the bridge between geographic and 3D space. The CesiumGlobeAnchor then converts the latitude, longitude, and height values into ECEF (Earth-Centered, Earth-Fixed) coordinates using the WGS84 ellipsoid model. This ECEF position represents a point in 3D space relative to the center of the Earth. Finally, the converted position is aligned with the CesiumGeoreference origin, which anchors the Unity scene to a specified real-world location, allowing the object such as a player or AR camera to be correctly placed and visualized within the Unity environment.

```

1  using UnityEngine;
2  using CesiumForUnity;
3  using Unity.Mathematics;
4
5  public class GPSPositionMapper : MonoBehaviour
6  {
7      public static GPSPositionMapper Instance;
8
9      public CesiumGeoreference cesiumGeoreference;
10
11     public bool IsReady =>
12         GPSLocationManager.Instance != null &&
13         GPSLocationManager.Instance.IsLocationReady &&
14         cesiumGeoreference != null;
15
16     private GameObject globeAnchorHelper;
17     private CesiumGlobeAnchor globeAnchorComponent;
18
19     private void Awake()
20     {
21         if (Instance == null)
22             Instance = this;
23         else
24             Destroy(gameObject);
25     }
26
27     /// Converts GPS to Unity world position using Cesium's georeference.
28     public Vector3 MapToWorld(double lat, double lon, double height = 0)
29     {
30         if (cesiumGeoreference == null)
31         {
32             Debug.LogError("CesiumGeoreference not assigned.");
33             return Vector3.zero;
34         }
35
36         GameObject temp = new GameObject("TempAnchor");
37         temp.hideFlags = HideFlags.HideAndDontSave;
38         temp.transform.SetParent(cesiumGeoreference.transform, false); // Make it a child of CesiumGeoreference
39
40         var anchor = temp.AddComponent<CesiumGlobeAnchor>();
41         anchor.longitudeLatitudeHeight = new double3(lon, lat, height);
42
43         Vector3 unityPosition = anchor.transform.position;
44
45         Destroy(temp);
46         return unityPosition;
47     }
48 }
49
50
51

```

Figure 4.7: Code snippet for calculation using cesium for unity

The `GPSPositionMapper` script serves as a core utility within the FloodGuard AR navigation system, acting as the critical link between the real world and the Unity-based augmented reality environment. Its primary role is to transform GPS-derived latitude, longitude, and altitude data into Unity's 3D coordinate space, ensuring that virtual elements such as user positions, navigation targets, and path indicators align accurately with their real-world counterparts. This allows FloodGuard to overlay navigation cues precisely on the terrain, regardless of the user's physical location.

The script is implemented as a singleton, making it globally accessible throughout the application. This design choice guarantees that any component — be it the pathfinding system, the line rendering script, or other navigation-related elements — can reliably access the conversion methods and obtain an accurate world position.

To perform its conversion, the script depends on two key components. The first is the `GPSLocationManager`, which provides real-time latitude, longitude, and altitude data from the device's GPS sensor. The second is the `CesiumGeoreference`, a component that facilitates seamless translation between geographic coordinates and Unity's Cartesian coordinate system. Together, these elements enable the script to map GPS data into Unity with precision.

To ensure stability and reliability, the script incorporates an `IsReady` boolean flag that confirms both the `GPSLocationManager` and the `CesiumGeoreference` have been fully initialized before any calculations occur. This approach prevents invalid position requests and guarantees that position conversion only takes place when the necessary services are available, making the `GPSPositionMapper` a fundamental and dependable link in FloodGuard's AR navigation workflow.

```

public LocationInfo CurrentLocation => Input.location.lastData;

private void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
        return;
    }
}

private void Start()
{
    #if UNITY_EDITOR
        latitude = manualLatitude;
        longitude = manualLongitude;
        locationReady = true;
        Debug.Log($"[GPSLocationManager] Using manual coordinates in Editor: {latitude}, {longitude}");
    #else
        StartCoroutine(StartLocationService());
    #endif
}

IEnumerator StartLocationService()
{
    if (!Input.location.isEnabledByUser)
    {
        Debug.LogWarning("[GPSLocationManager] Location services not enabled.");
    }
    #if UNITY_ANDROID
        using (AndroidJavaClass unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer"))
        {
            AndroidJavaObject currentActivity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");
            AndroidJavaObject intent = new AndroidJavaObject("android.content.Intent", "android.settings.LOCATION_SOURCE_SETTINGS");
            currentActivity.Call("startActivity", intent);
        }
    #endif
    yield break;

    Input.location.Start(1f, 1f);

    int maxWait = 20;
    while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)
    {
        yield return new WaitForSeconds(1);
        maxWait--;
    }

    if (maxWait <= 0 || Input.location.status == LocationServiceStatus.Failed)
    {
        Debug.LogWarning("[GPSLocationManager] Location service failed or timed out.");
        yield break;
    }

    locationReady = true;
    UpdateLocation();
    InvokeRepeating(nameof(UpdateLocation), 0f, 2f); // update every 2 seconds
}

private void UpdateLocation()
{
    if (Input.location.status == LocationServiceStatus.Running)
    {
        latitude = Input.location.lastData.latitude;
        longitude = Input.location.lastData.longitude;
        Debug.Log($"[GPSLocationManager] Updated Location: {latitude}, {longitude}");
    }
    else
    {
        Debug.LogWarning("[GPSLocationManager] Location service stopped unexpectedly.");
        CancelInvoke(nameof(UpdateLocation));
        locationReady = false;
    }
}
}

```

Figure 4.8: Code snippet for obtaining user location

The `GPSLocationManager` script is responsible for acquiring and managing real-time geographic location data from the user's device, specifically the latitude and longitude coordinates required for location-based augmented reality (AR) interactions in the FloodGuard application. This script ensures that the system continuously receives updated GPS data, which is essential for rendering user positions, destinations, and navigation paths in the correct spatial context within the Unity environment.

Implemented as a singleton, the script ensures only one instance of the location manager exists during runtime, accessible globally via `GPSLocationManager.Instance`. This centralized design allows other scripts, such as `GPSPositionMapper`, to retrieve the current location data easily and consistently.

The core functionality begins in the `Start()` method, which differentiates between two execution environments: the Unity Editor and a physical mobile device. When running in the Unity Editor, the script uses predefined manual Latitude and manual Longitude values, allowing developers to simulate location data without GPS hardware. This feature is particularly useful for debugging or UI testing in desktop development environments.

On a real device, the script starts a coroutine named `StartLocationService()` to initialize the device's GPS module. Before beginning, it checks whether the user has enabled location services. If not, and if running on Android, the script automatically opens the device's system settings to prompt the user to enable GPS. Once permission is granted, the script starts the Unity location service using `Input.location.Start()`, configured to update every one meter or one second of movement. The service is given up to 20 seconds to initialize; if it fails or takes too long, the script logs a warning and exits the process gracefully.

Upon successful initialization, the script sets a `locationReady` flag to true and begins updating the latitude and longitude values every two seconds via the `UpdateLocation()` method. This method accesses the latest available GPS data from `Input.location.lastData`, extracting and storing the latitude and longitude into the class's internal variables. These values are exposed through `CurrentLatitude` and `CurrentLongitude` properties, which intelligently return either manual or live data based on the platform context. Additionally, if the GPS service unexpectedly stops or becomes unavailable, the update cycle is cancelled, and the system is marked as not ready, avoiding stale or inaccurate position data.

4.3.3 Navigation Line

Navigation System Using NavMesh

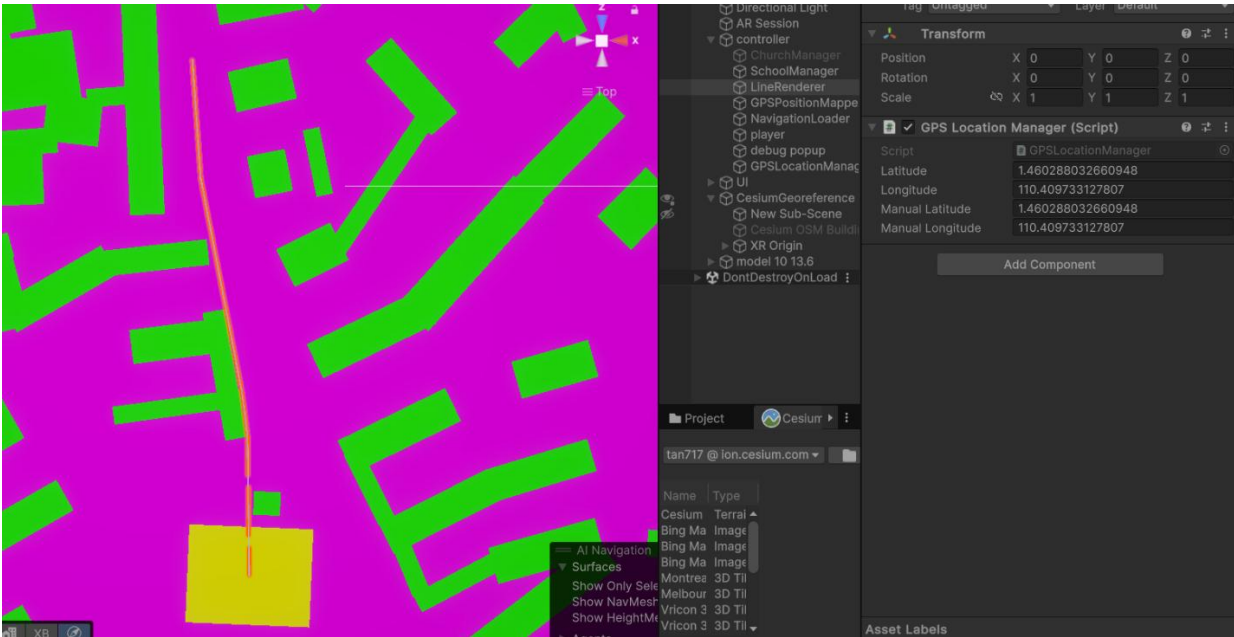


Figure 4.9: Position on Lat (1.4603032039173787), Lon (110.40976117233218)

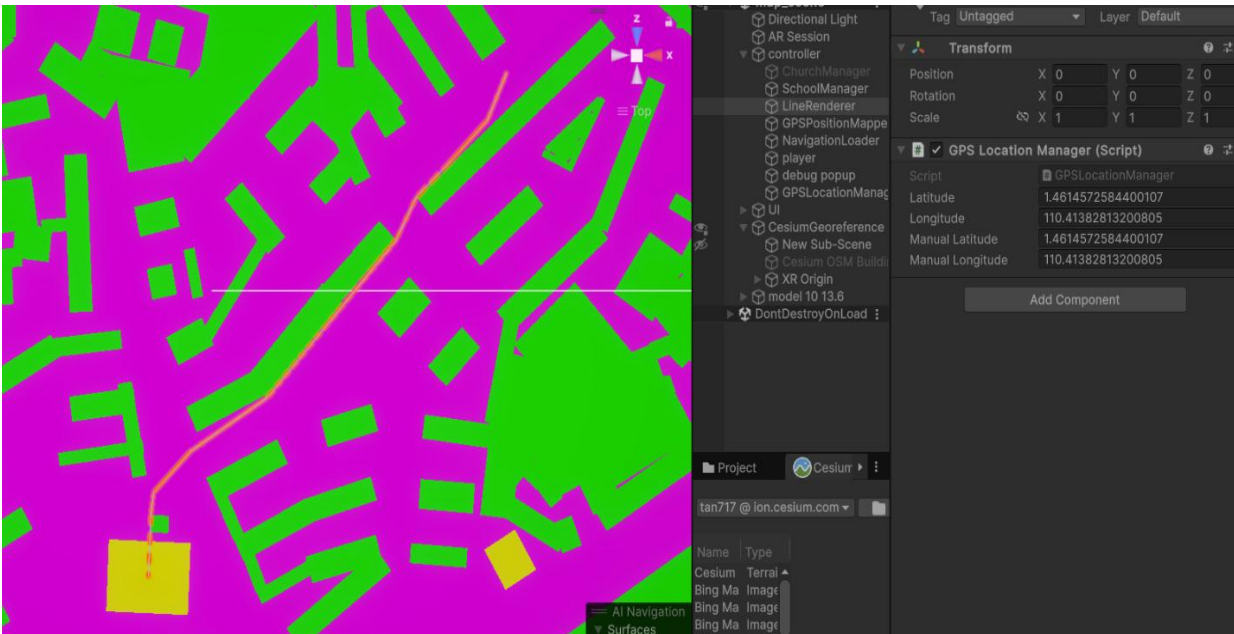


Figure 4.10: Position on Lat (1.4615193598316287), Lon (110.41387269287807)

Unity's built-in Navigation Mesh (NavMesh) system was leveraged to enable robust pathfinding and AI navigation within the FloodGuard AR application. The NavMesh provides a pre-calculated layout of walkable surfaces across the virtual environment, making it ideal for simulating navigation between points while avoiding terrain and obstacles. In its original form, the NavMesh allows for seamless pathfinding and movement within Unity's 3D space.

To adapt this system for augmented reality and real-world navigation, an overlay technique was implemented using a line-based navigation system. This approach combines the benefits of NavMesh pathfinding with GPS-derived positioning, allowing navigation paths to be visualized in AR. The line renderer draws a path that connects GPS-defined coordinates, converting latitude and longitude values into Unity world positions through the GPSPositionMapper. This method allows the route line to be updated both in real time and in manual testing scenarios, making it highly adaptable for a range of use cases.

Through this dual approach employing NavMesh for internal path calculations and GPS-based line rendering for external positioning — the FloodGuard app achieves a seamless blend of virtual path planning and live AR navigation. The effectiveness of this approach is illustrated in Figures 4.9 and 4.10, which show the path line generated from two different sets of GPS coordinates.

These examples highlight the system's ability to adapt to varying starting positions, ensuring accurate and reliable navigation across different real-world locations. Together, the NavMesh and GPS-based line rendering form the foundation for FloodGuard's AR navigation, making it both spatially accurate and highly interactive for end-users.

```

FUNCTION Update():
    // Get current GPS coordinates
    latitude = current GPS latitude
    longitude = current GPS longitude
    // If player not yet spawned
    IF player is not spawned:
        IF GPS and Mapper are both ready:
            // Convert GPS to Unity world position
            worldPosition = convert (latitude, longitude) to Unity coordinates
            // Move camera to this position
            set camera position to worldPosition
            // Spawn the player at the same position (with synced height)
            syncY = 0
            set player position = worldPosition (with Y = syncY)
            set player rotation = camera rotation
            LOG: player GPS location
            LOG: player spawned
            set playersSpawned = true
        ELSE:
            // Player already spawned, update player location continuously
            IF GPS and Mapper are both ready:
                worldPosition = convert (latitude, longitude) to Unity coordinates
                // Move camera to current location
                set camera position = worldPosition
                // Sync player position and rotation
                syncY = 0
                set player position = worldPosition (with Y = syncY)
                set player rotation = camera rotation
            // --- Target Position Initialization and Stabilization Logic ---
            IF player AND target AND line exist:
                IF ground not snapped:
                    IF target exists AND not yet logged:
                        LOG: Initial target position
                        mark target position logged = true
                        // Snap the target to ground
                        snap target to ground
                        mark ground snapped = true
                    IF target exists AND not yet logged for stabilization:
                        LOG: Final target position
                        mark target position logged = true
                IF Vuforia not initialized:
                    mark Vuforia initialized
                    start stabilization timer
                    LOG: Starting stabilization...
                IF Vuforia initialized AND path not yet calculated:
                    update stabilization timer
                    LOG: Stabilizing...
                    IF timer exceeds threshold:
                        LOG: Timer complete, calculating path
                        calculate path
                        mark path calculated = true
                        store last known player position
            // If player moved significantly since last path update
            IF path already calculated AND distance from last position is large:
                LOG: Player moved, recalculate path
                calculate path again
                update last player position

```

Figure 4.11: screenshot illustrating how system update the user and camera positions

Figure 4.11 shows the main Update() method of the navigation script. This method is responsible for synchronizing the camera and player positions based on the device's GPS data and triggering the navigation pathfinding. Initially, it uses the GPSLocationManager and GPSPositionMapper to locate the user's current position, updating the camera and aligning the player's position and rotation accordingly. Once this initialization is complete, the script snaps both the player and target to the ground, ensuring accurate placement on the terrain. After a brief stabilization period, it calculates the navigation path using Unity's NavMesh. As the user moves, the script monitors the position and dynamically recalculates the path when necessary, updating the navigation line in real-time.

NavMesh Surface

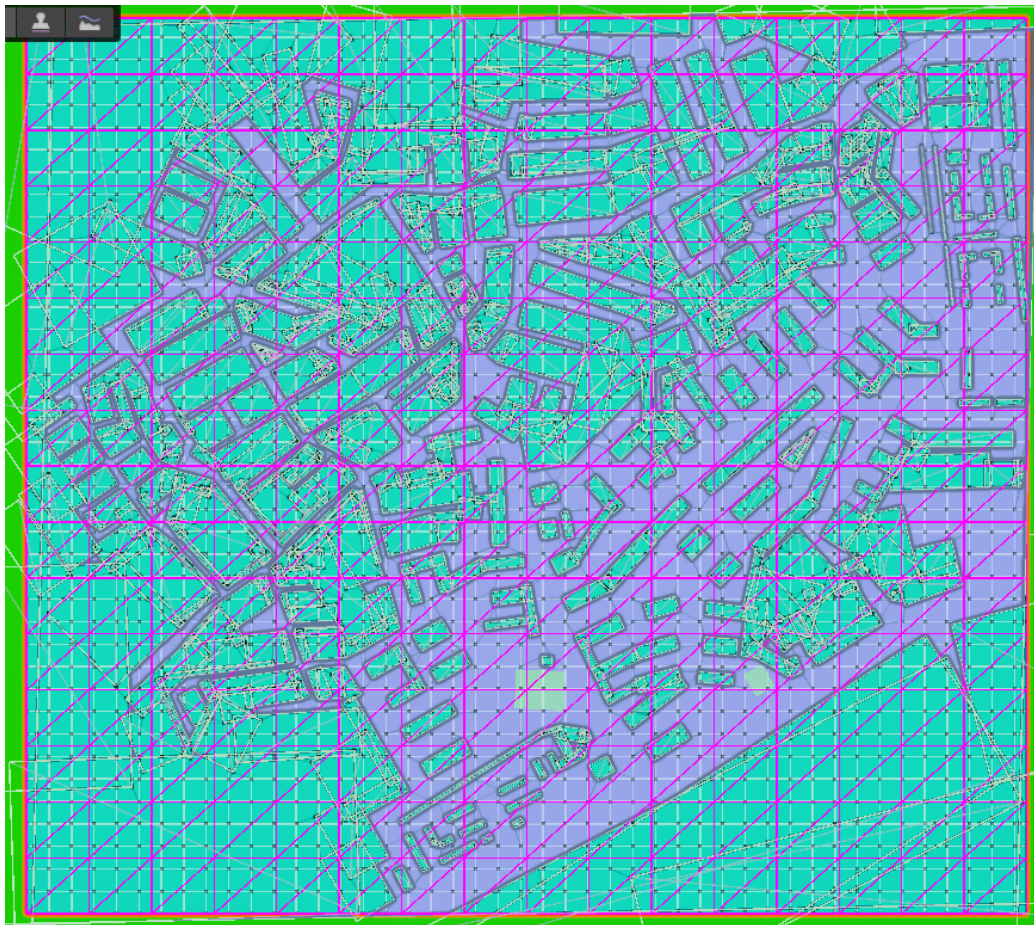


Figure 4.12: Categorizing walkable surfaces with obstacles

A NavMesh Surface is a core component in Unity's navigation framework, designed to define and manage walkable areas within a 3D scene. It works by analyzing the geometry of the

environment such as terrain, road meshes, and other surfaces — and generating a Navigation Mesh (NavMesh), which is a data structure that describes where agents can move and where they must avoid obstacles. By applying a NavMesh Surface, Unity can accurately “bake” this information into the scene, creating a map of walkable areas that its pathfinding system can use. In this project, NavMesh Surfaces were added to terrain and road meshes to mark them as navigable spaces, allowing Unity’s navigation system to compute realistic and efficient paths for the AR user. This process ensures that movement within the scene respects the contours of the terrain, avoids obstacles, and allows for seamless, context-aware navigation between starting points and destinations. In this way, the NavMesh Surface plays a pivotal role in achieving accurate and reliable pathfinding in the FloodGuard AR navigation application.

NavMesh Agent

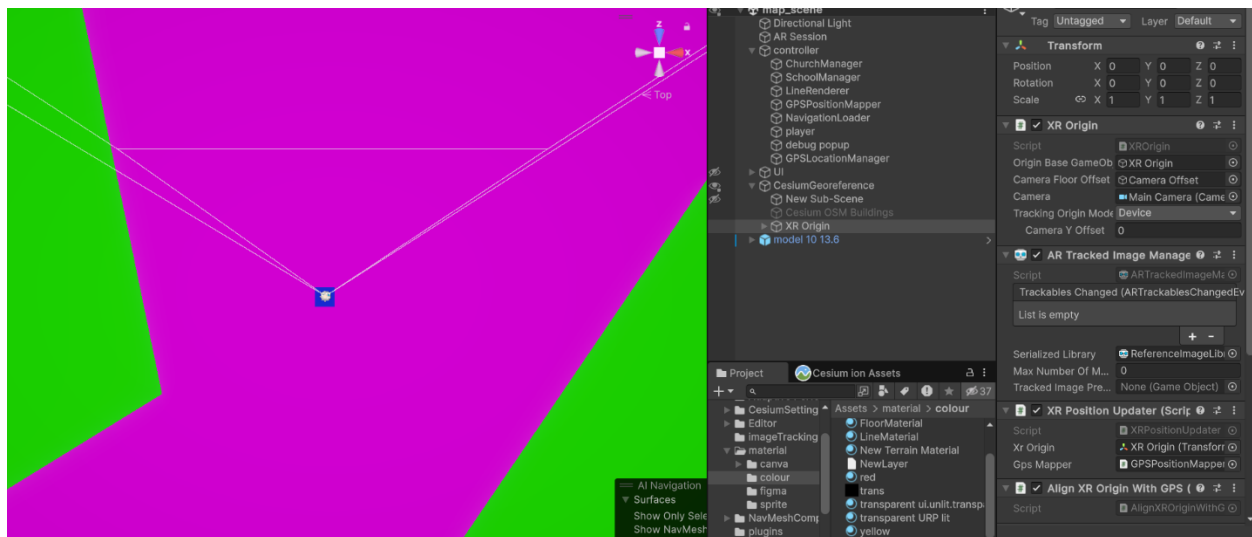


Figure 4.13: Nav Mesh Agent

The NavMesh Agent component in Unity was used to enable intelligent and realistic movement within the scene. By attaching the NavMesh Agent to the player object, the system gained access to Unity’s built-in pathfinding capabilities, allowing it to move along a pre-baked NavMesh surface. During the development phase, a placeholder agent was used to simulate user movement within the virtual environment, navigating between a defined starting point and an end point. The agent reliably calculated the shortest path across the available walkable surfaces, automatically avoiding obstacles and providing smooth cornering behavior. This approach created a robust foundation for integrating navigation into the AR experience.

To extend this concept beyond the virtual space, the system was enhanced to support coordinate-based positioning. By retrieving GPS data — either in real-time or through manual input the player’s position could be accurately aligned with a specific geographical location within the Unity scene. In this setup, the player (and by extension, the NavMesh Agent) became the starting point for AR navigation, allowing the calculated path to dynamically adjust based on actual GPS coordinates. The navigation line would then connect the player’s current position to the target destination, making it easy for the user to follow the route in the real world.

Figure 4.13 illustrates this concept by showing the player object, represented as the NavMesh Agent, serving as the starting point for pathfinding within the scene. This seamless integration of Unity’s NavMesh Agent and GPS coordinate mapping forms the core of the FloodGuard AR navigation experience, combining intelligent pathfinding with precise, context-aware geolocation to guide users effectively in both virtual and physical environments.

4.4 System Architecture

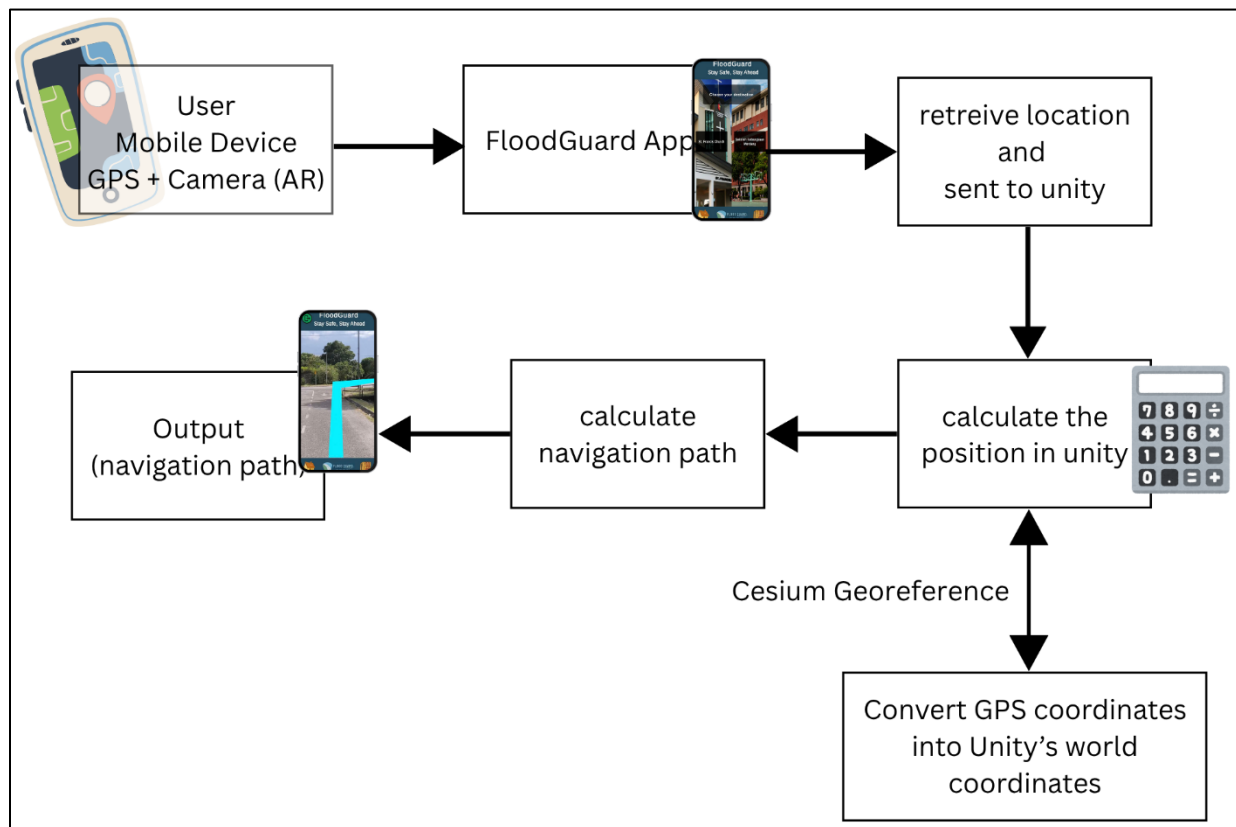


Figure 4.14: Overview of the System Architecture

FloodGuard is built upon a tightly integrated architecture that combines Unity, Cesium, and GPS services to deliver a precise and context-aware augmented reality (AR) navigation experience. At its core, the system captures the user's GPS coordinates and relays this information to Unity, where it is transformed into 3D world positions through the Cesium for Unity plugin. Cesium acts as a global georeferencing engine, translating latitude, longitude, and altitude data into a highly accurate 3D coordinate space within Unity. This allows the FloodGuard application to position virtual navigation elements precisely over the user's actual location, regardless of where the app is launched.

Unity serves as the primary engine for rendering and interaction, managing the scene's terrain, user avatar, and navigation elements. It uses the NavMesh system to define walkable areas within the environment, while its line rendering tools draw dynamic navigation paths that connect the user's position to a target destination. Meanwhile, AR Foundation provides access to camera data and device motion tracking, allowing the app to overlay these navigation guides and other relevant information directly onto the user's camera feed.

Through this seamless interaction between GPS, Cesium, Unity, and AR Foundation, FloodGuard delivers an AR navigation experience that is both highly accurate and responsive, allowing users to move through the real world while following contextually relevant visual cues. This architecture ensures that navigation information is always aligned with the user's actual surroundings, making the app suitable for flood evacuation or any other location-based guidance scenario.

4.5 User Interfaces (UI)

Home Page

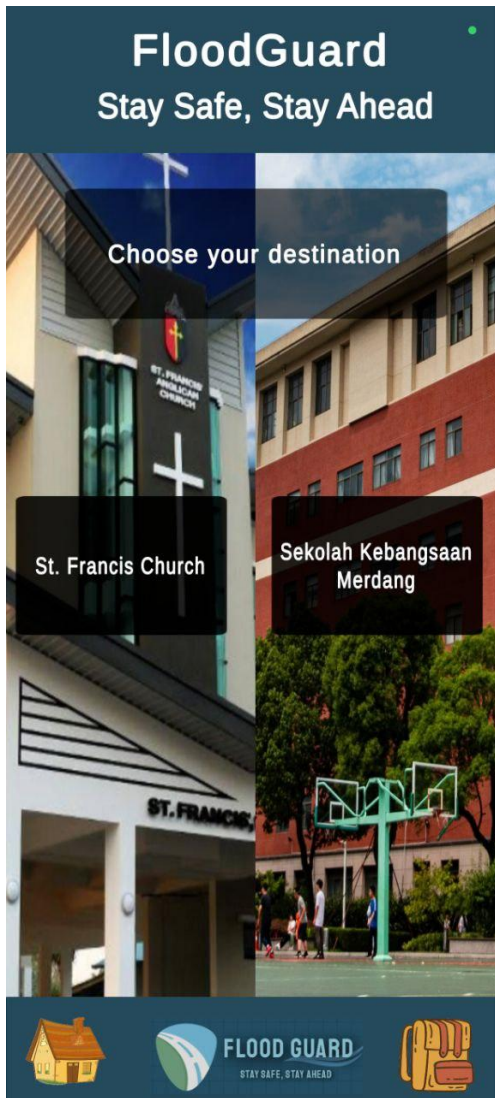


Figure 4.15: Home Page



Figure 4.16: Loading View Page

These images represent the homepage and loading view page of the FloodGuard mobile application, designed to help users stay safe during flood emergencies. At the top, the app name “FloodGuard” is prominently displayed, accompanied by the slogan “Stay Safe, Stay Ahead,” establishing its purpose and building trust. The central interface features two key images: St. Francis Church and Sekolah Kebangsaan Merdang, each labeled clearly and functioning as interactive buttons that allow users to select their evacuation destination. A semi-transparent box in the middle provides contextual guidance with the message, “Choose your Destination” and

“Setting Evacuation Path to Sekolah Kebangsaan Merdang,” enhancing user orientation. At the bottom of the screen, a home icon and a checklist icon serve as navigation tools, returning the user to the homepage or accessing emergency preparations. The FloodGuard logo is also placed at the bottom to reinforce branding. This homepage applies several multimedia principles, which are observed through the clean, distraction-free layout that focuses attention on essential functions; the signaling principle highlights interactive areas using text boxes; and the spatial contiguity principle ensures labels are placed close to their respective images for intuitive understanding. Overall, the design promotes clarity, consistency, and ease of navigation, supporting user decision-making during a crisis.

Checklist Page



Figure 4.17: Checklist View Page

This checklist page is designed as a reminder to help users prepare essential items for flood emergencies. On the top-left corner, a green exit button allows users to return to the previous screen, promoting easy and intuitive navigation. The main section of the page lists five critical emergency items which are Food and Beverage, Flashlight, Phone & Power bank, First Aid Kit, and Whistle, each accompanied by a clear black-and-white icon and corresponding text label. This layout applies key multimedia learning principles: the coherence principle is reflected in the minimal and focused design that eliminates unnecessary distractions; the signaling principle is demonstrated through the visual emphasis on each item with icons and aligned text that guide user attention; and the spatial contiguity principle is used effectively by placing each icon next to its label, helping users easily understand and retain the information. Additionally, the consistency principle is maintained through uniform icon style and layout structure. At the bottom of the page, three icons represent navigation options: home, the FloodGuard logo, and a highlighted checklist icon, helping users seamlessly access other app sections. This checklist page is a practical and user-friendly guide that supports flood preparedness through clear visuals and structured content.

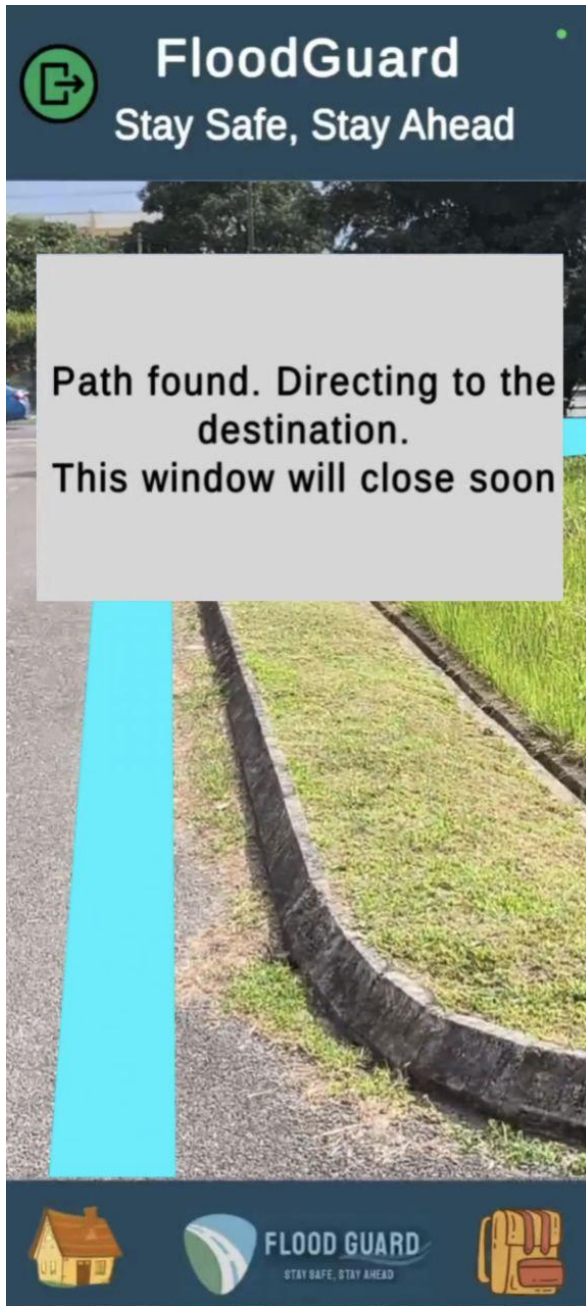


Figure 4.18: Notify window



Figure 4.19: mapping page

In the FloodGuard AR navigation system, visual feedback plays a key role in guiding users effectively. When a navigation route is successfully calculated based on the user's current GPS coordinates and the chosen destination, the system triggers a notification window displaying the message: "Path found. Directing to the destination. This window will close soon." This message

appears as immediate confirmation that the system has computed a valid route and that navigation is about to begin.

Before this message is shown, the system performs several internal checks to ensure the route is valid. Specifically, it verifies whether the user (player) is positioned on a walkable area within the environment. If not, the system halts the pathfinding process and displays the message: “Player is not on a walkable area. Please move.” A similar check is conducted for the destination point; if it is not located on a valid navigable surface, the system informs the user with: “Target location is unreachable. Check placement.”

Only when both the player and target are confirmed to be on walkable surfaces does the system proceed with calculating the path. If the path is found to be incomplete—such as when there is not enough navigable space or obstacles prevent a clear route—the system displays: “Path is incomplete. Try moving to a wider area.” In cases where the calculation fails entirely, a fallback message appears: “Failed to calculate path.”

Once a complete path is successfully generated, a blue navigation line is rendered on the ground to visually direct the user. This line is dynamically calculated using real-time GPS data that is mapped into Unity’s world space via georeferencing. However, due to the asynchronous nature of GPS updates, there may be a brief delay or visual offset between the user’s actual position and the line’s location. As fresh coordinates are received, the system adjusts the navigation line accordingly, gradually improving alignment.

As shown in Figure 4.18, after the initial message disappears, the navigation interface transitions into a cleaner state which only the blue path remains active. This supports a more immersive and uncluttered AR experience while continuing to guide the user visually to the destination. The consistent integration of dynamic path rendering and conditional feedback ensures both usability and reliability in real-world navigation scenarios.

4.6 Conclusion

This chapter outlines the technical implementation of FloodGuard, an augmented reality (AR)-based mobile application designed to assist flood evacuation in real time. The implementation focused on integrating AR features, GPS-based tracking, and navigational guidance within Unity using AR Foundation and Cesium for Unity.

Unity was selected as the primary development platform due to its robust support for AR features, while Cesium provided geospatially accurate 3D terrain and satellite data. Key components included the Unity Terrain System, Cesium 3D Tiles integration, and NavMesh for pathfinding. Limitations in Cesium's data completeness were addressed by manually adding Unity primitives to critical areas.

The system's core features—real-time coordinate mapping, GPS-based location tracking, and navigation line rendering were developed using C# scripts (e.g., `GPSLocationManager`, `GPSPositionMapper`). These scripts enabled real-world positioning of the user and accurate path visualization. Navigation lines were implemented using Unity's `LineRenderer` and `NavMesh Agent` components, enabling intelligent movement across walkable surfaces.

Additionally, the app's architecture followed a modular design to improve performance and manage geospatial components through Unity's Sub Scenes. User interface elements, including the home page, evacuation selector, and checklist page, were designed with multimedia learning principles to support ease of use during emergencies.

Overall, this chapter illustrates how FloodGuard was built as a hybrid system that merges real-world GPS data with a virtual navigation interface to provide immersive and practical flood evacuation guidance.

Chapter 5

Testing and Evaluation

5.1 Introduction

This chapter describes the results that were obtained during testing and evaluation of the systems. The methodology used when testing is System Usability Scale (SUS) along with user testing and performance testing. After that, the accuracy of navigation is described followed by feedback on SUS surveys such as effectiveness and responsiveness. Finally, the system limitations will also be discussed in this chapter.

5.2 System Usability Scale

The System Usability Scale (SUS) is a reliable, industry-standard tool used to assess the usability of a certain system. It provides a quick and effective way to evaluate whether users find a system easy or difficult to use. SUS consists of a ten-item questionnaire with five response options for respondents; from Strongly Disagree (1) to Strongly Agree (5).

SUS chooses because it is simple to administer, interprets well even with a small sample size, and generates a single score that reflects the overall usability of the system. It is especially suitable for systems like Floodguard, which are designed to be used in emergency or high-stress environments, where usability and clarity are critical.

5.3 System Usability Scale Analysis

The prototype testing has been conducted at Uni Garden area with a total of 10 respondents involved in the testing. The average SUS score was calculated, and analysis was conducted. Most users agreed that the tool was easy to use and would recommend it to others.

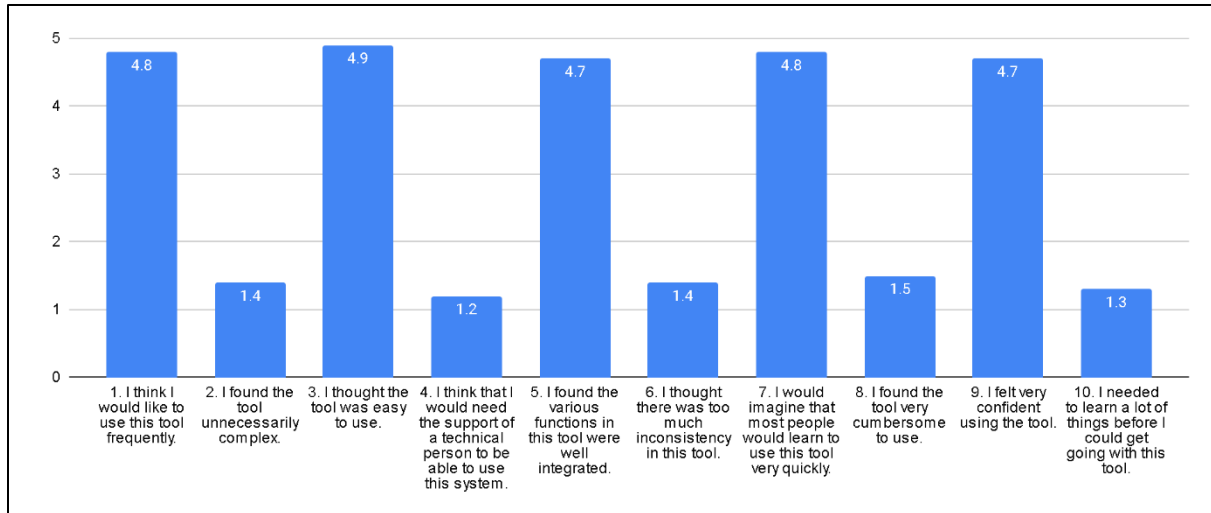


Figure 5.1: Average Score for SUS Survey

Result Calculated from Google Form										
Respondent	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	4	1	5	1	4	1	5	1	5	1
2	5	2	4	1	5	2	4	1	3	1
3	5	1	5	1	5	1	5	1	5	1
4	5	2	5	1	4	1	4	2	5	1
5	5	1	5	1	5	1	5	2	5	2
6	5	2	5	2	5	2	5	2	5	2
7	4	1	5	1	4	1	5	1	4	1
8	5	1	5	2	5	2	5	1	5	1
9	5	1	5	1	5	2	5	2	5	2
10	5	2	5	1	5	1	5	2	5	1

Table 5.1: Recorded result from Google Form

Questions	Average Score For each question	Scoring Formula	Result
1	4.8	4.8-1	3.8
2	1.4	5-1.4	3.6
3	4.9	4.9-1	3.9
4	1.2	5-1.2	3.8
5	4.7	4.7-1	3.7
6	1.4	5-1.4	3.6
7	4.8	4.8-1	3.8
8	1.5	5-1.5	3.5
9	4.7	4.7-1	3.7
10	1.3	5-1.3	3.7
Total			37.1
Total * 2.5			92.75

Table 5.2: Calculation based on recorded result

The System Usability Scale (SUS) was used to evaluate the usability of the FloodGuard application based on responses from 10 participants. The survey was conducted using Google Forms, where each participant completed a standardized 10-item questionnaire rated on a 5-point Likert scale, with 1 indicating "Strongly Disagree" and 5 indicating "Strongly Agree."

Each question corresponds to a key usability principle such as ease of use, complexity, confidence, integration, and learnability. The SUS scoring process involves adjusting responses: for odd-numbered (positive) questions, 1 is subtracted from the score, while for even-numbered (negative) questions, the score is subtracted from 5. The adjusted scores are then summed and multiplied by 2.5 to obtain a final SUS score on a scale of 0–100.

In this study, the total adjusted score was 37.1, which resulted in a final SUS score of 92.75. Since SUS scores above 68 are considered above average, and scores above 90 represent excellent usability, this result reflects that FloodGuard is a highly usable, efficient, and user-friendly system. Further breakdown of average responses per question supports this interpretation. Most positive statements received average scores of 4.8 or higher, such as “I think I would like to use this tool frequently” (4.8) and “I found the tool easy to use” (4.9). In contrast, negative statements like “I found the tool unnecessarily complex” (1.4) and “I found the tool very cumbersome to use” (1.5) received low scores, indicating strong agreement on the system’s simplicity and clarity.

These results suggest that users were able to interact with the system confidently and intuitively, without needing technical assistance (1.2 average on needing support). The system was also found to be consistent and easy to learn, as reflected by high scores on quick learnability (4.8) and confidence while using the tool (4.7). Therefore, both the raw SUS calculations and the average response analysis validate that FloodGuard delivers a high level of usability, making it suitable for emergency scenarios where ease of use and speed are critical.

SUS Score Range	Grade	Adjective Rating	Usability Description
90 – 100	A+	Excellent	Exceptional usability; users are delighted.
80 – 89	A	Good–Excellent	High usability; users are very satisfied.
70 – 79	B	Good	Above average usability; still room for improvement.
60 – 69	C	OK/Fair	Average usability; acceptable but not impressive.
50 – 59	D	Poor	Below average usability; needs improvement.
< 50	F	Awful	Very poor usability; users are frustrated.

Table 5.3: SUS Score Interpretation Table

The table presents the interpretation of SUS (System Usability Scale) scores based on established usability benchmarks. SUS scores range from 0 to 100 and are commonly categorized into descriptive bands such as "Poor," "Good," or "Excellent" to reflect the perceived usability of a system. These categories are often aligned with academic-style grades (A–F) and adjective ratings to help non-technical stakeholders understand the results more easily. According to the table, a SUS score of 90–100 is rated as “Excellent” usability, receiving a grade of A+, which indicates that the system is exceptionally user-friendly and well-designed. In this evaluation, the system received a SUS score of 92.75, placing it firmly in the “Excellent” category. This result suggests that users found the system highly usable, intuitive, and satisfying to use, with minimal difficulties in operation or learning.

5.4 SUS Two Factor Analysis

The System Usability Scale (SUS) is a widely used tool for evaluating the perceived usability of a system. Traditionally, SUS is interpreted as a single-dimensional scale that generates a score out of 100, reflecting overall user satisfaction. However, Lewis and Sauro (2009) proposed a more detailed approach by dividing SUS into two distinct subscales: Usability and Learnability. This two-factor structure provides deeper insight into how users interact with a system:

- Usability refers to how effectively users can operate the system during regular use.
- Learnability focuses on how easily new users can understand and begin using the system, particularly on their first attempt.

The 10 SUS items are categorized as follows:

- Usability-related items: Q1, Q2, Q3, Q5, Q6, Q7, Q8, Q9
- Learnability-related items: Q4 and Q10

By calculating separate scores for these subscales, developers and researchers can determine not only how usable a system is, but also how quickly it can be learned with a distinction that is especially valuable in interactive systems such as mobile or AR applications, where both usability and fast onboarding are critical.

Using the adjusted responses already computed in the overall SUS analysis, the subscale scores were derived as follows:

$$\begin{aligned}
 \text{Usability Score} &= \left(\frac{\text{Sum of Adjusted Usability Items}}{8} \right) \times 2.5 \\
 \text{Usability Score} &= \left(\frac{3.8 + 3.6 + 3.9 + 3.7 + 3.6 + 3.8 + 3.5 + 3.7}{8} \right) \times 2.5 \\
 \text{Learnability Score} &= \left(\frac{\text{Sum of Adjusted Learnability Items}}{2} \right) \times 2.5 \\
 \text{Learnability Score} &= \left(\frac{3.8 + 3.7}{2} \right) \times 2.5
 \end{aligned}$$

The usability score was 92.5, and the learnability score was 93.75. These high scores suggest that the system is not only highly usable during regular operation, but also very easy to learn, even for first-time users. This insight is valuable because it confirms that the app offers a low learning curve and supports quick onboarding — crucial in emergency-focused applications where users may interact with the system for the first time during high-stress situations.

By adopting this two-dimensional perspective, the evaluation provides deeper understanding of user experience and can guide targeted improvements — for instance, refining user onboarding tutorials if learnability scores are significantly lower than usability.

5.5 Results and Discussion

This section presents an analysis of the System Usability Scale (SUS) responses collected from 10 participants who interacted with the FloodGuard application. The SUS is a standard questionnaire comprising ten statements, alternating between positive and negative framing which

are used to evaluate the overall usability of a system. Each item was rated on a 5-point Likert scale, where 1 represents “Strongly Disagree” and 5 represents “Strongly Agree” as shown previously in Figure 5.1

The objective of this survey was to evaluate the ease of use, integration, and user confidence while interacting with FloodGuard. The average scores for each question were computed to assess the system’s performance across usability dimensions.

The results from the SUS survey indicate overwhelmingly positive feedback regarding the usability of the FloodGuard system. Participants expressed high levels of satisfaction with the application’s ease of use and learnability. For example, the statement “I thought the tool was easy to use” received an average score of 4.9, while “I would imagine most people would learn to use this tool very quickly” averaged 4.8, and “I felt very confident using the tool” scored 4.7. These findings suggest that users found the system intuitive, user-friendly, and easy to navigate without requiring prior technical expertise.

In terms of complexity and support requirements, the low average scores for negatively worded statements such as “I found the tool unnecessarily complex” (1.4), “I think I would need the support of a technical person to be able to use this system” (1.2), and “I found the tool very cumbersome to use” (1.5) further support the notion that the system posed minimal usability barriers. These scores indicate that users did not perceive the system as overly complicated, nor did they feel the need for external assistance while operating it.

Furthermore, the responses also highlight the effectiveness of the system’s design in terms of feature integration and consistency. The statement “The various functions in this tool were well integrated” received an average score of 4.7, while the negatively framed statement “There was too much inconsistency in this tool” averaged only 1.4. These results show that users acknowledged a high level of consistency throughout the system and appreciated how its features worked together seamlessly. Collectively, these findings affirm the FloodGuard system’s strong usability and reinforce its potential for practical application in real-world emergency scenarios.

The strong positive trend, with average scores above 4.7 on positive statements and below 1.5 on negative ones, reflects high user satisfaction. These results suggest that FloodGuard is effective in delivering a user-friendly interface with well-integrated functionalities. Users were

able to confidently navigate the application without external support, reinforcing the design's strength in promoting self-reliance during critical emergency scenarios.

The consistency of the responses also highlights the robustness of the application across different users and contexts. These findings validate the design choices made during development, particularly in areas of AR navigation, UI clarity, and emergency preparedness support.

5.6 Scenario based testing overview

To assess the real-world performance of the FloodGuard application, scenario-based testing was conducted with multiple users across different environments. These scenarios were designed to simulate actual flood evacuation conditions and observe user interaction with the app's core navigation features.

These varied user scenarios provided valuable insights into the robustness of FloodGuard under different environmental conditions. Testing in optimized locations like Uni Garden demonstrated the system's full capabilities, while tests in suboptimal areas highlighted limitations related to GPS signal strength and Cesium map coverage. These findings reinforce the importance of field testing and support the strong usability scores reported earlier.

To supplement this section, a video playlist has been prepared showcasing selected user test sessions including navigation in the Uni Garden area, destination selection, and AR guidance in action. These recordings provide visual confirmation of the system's performance and serve as supporting evidence for the evaluation presented. <https://youtube.com/playlist>

Recorded Scenario in Effective Environment (Uni Garden)

One of the test sessions was recorded in Uni Garden, a flood-prone area targeted by the system's spatial configuration. In this scenario, the user launched the FloodGuard app from the sidewalk at Uni Garden and selected Sekolah Kebangsaan Merdang as the evacuation destination. The app initialized the AR-based navigation line, which was visually overlaid onto the physical environment through the user's mobile camera. As the user walked toward the evacuation point, the path was continuously updated using GPS data, with AR cues helping to guide the user in real time. The session successfully concluded at the destination, validating both the AR guidance and system responsiveness in an optimized environment.

Unrecorded Scenario in Effective Environment

Several users also tested the application in Uni Garden without recording their sessions. Despite the absence of video documentation, user feedback and observation confirmed similar outcomes accurate route rendering, stable AR overlays, and positive navigation flow. These users also reported high confidence in following the path, aligning with SUS results that suggested ease of use and low complexity.

Scenario in Non-Optimized (Ineffective) Environments

Other participants tested FloodGuard in environments outside the predefined area, such as residential zones or locations with weak GPS signal and limited Cesium map data. In these conditions, the system occasionally struggled to render accurate AR overlays or maintain continuous path alignment. Users experienced some delay in route rendering and inconsistent camera tracking due to lack of terrain mapping or reduced geospatial fidelity. Despite this, most were still able to understand the interface and navigate short distances, albeit with reduced accuracy.

5.7 Conclusion

The testing and evaluation phase of the Floodguard application demonstrated promising outcomes in terms of usability, functionality, and user satisfaction. Using the System Usability Scale (SUS), 10 users evaluated the prototype, with the results indicating a strong overall positive user experience. The average SUS score showed high marks in areas such as ease of use, confidence in usage, and willingness to reuse, while concerns like system complexity and need for technical support scored notably low, both favorable signs of a well-designed system.

Users particularly appreciated the intuitive interface, responsive performance, and the helpfulness of AR navigation in simulated flood conditions. However, minor limitations were reported, particularly involving AR marker detection accuracy in poor lighting conditions. Despite these limitations, users confirmed the system met its intended goals of improving emergency navigation and preparedness through AR visualization and real-time pathfinding.

The outcome of the evaluation confirms that the objectives of the project were achieved. The AR-based system successfully assists users in navigating flood scenarios and enhances their

situational awareness during emergencies. These results not only validate the technical feasibility but also the real-world applicability of AR in disaster response systems.

Chapter 6

Conclusion and Future Work

6.1 Introduction

This is the last chapter that summarizes the research, implementation, and main findings of the project. This chapter addresses the major contributions of the system, i.e., the success in navigation through AR technology. In addition, this chapter discusses potential future improvements and extensions of the system. These include increased accuracy in navigation, 3D modelling for more complex environments, better representations, and a broader set of functions for other applications. Finally, the thesis summarizes the important things learned from the built system and explains its strengths, weaknesses, and added value. Additionally, it highlights some areas for future work and the possibility of enhancement in both AR-based and human artifact navigation systems and their real-world applicability and wider range of effects.

6.2 Objective Achieved

Objectives	Achievements
To develop an augmented reality-assisted flood evacuation	The mobile application Floodguard was successfully developed using Unity, AR Foundation, and Cesium for Unity. It integrates real-time GPS data, and AR-based navigation. The system overlays evacuation paths in the user's real-world environment, effectively assisting users during flood emergencies.
To evaluate the usability of the proposed augmented reality-assisted evacuation	A usability study was conducted using the System Usability Scale (SUS). The results showed a high level of user satisfaction and system effectiveness.

6.3 System Limitation

While the developed AR-assisted evacuation system, Floodguard, demonstrates promising functionality and usability, it is important to acknowledge its limitations. These limitations help define the scope of the system and identify areas for future improvement.

The most significant limitation of the system lies in its geographical restriction. Currently, Floodguard is designed and implemented exclusively for use within the Uni Garden area in Kota

Samarahan, Sarawak. All AR markers, routes, evacuation points, and navigation functionalities are tailored specifically to the layout and emergency response needs of this locality.

As a result:

- The system cannot be used effectively outside the Uni Garden area without modification.
- Navigation and evacuation paths are static and limited to the mapped infrastructure of Uni Garden.
- The AR content and wayfinding markers are not scalable or adaptable to other geographical regions unless custom data and mapping are developed.

This limitation was necessary due to time constraints, resource availability, and the need for a focused proof-of-concept. However, it also highlights a key opportunity for expansion in future versions of the application.

6.4 Future enhancement

To improve and expand the system in the future, several enhancements are proposed:

- **Expansion to Larger Areas**

One of the main goals for future development is to expand the system's coverage beyond Uni Garden. This includes mapping additional neighborhoods, towns, or entire flood-prone regions, allowing more users to benefit from AR-assisted evacuation guidance.

- **Dynamic Area Mapping**

Future versions could include dynamic data loading based on user location, allowing the app to adapt to different areas using GPS and cloud-based mapping services.

- **Integration with Real-Time Flood Alerts**

The system can be integrated with government or weather agencies' flood alert APIs to automatically update evacuation paths based on live flood conditions.

- **Multilingual and Accessibility Features**

To improve usability for a wider audience, the system could support multiple languages and include features for users with disabilities (e.g., voice navigation, text-to-speech).

- **Cross-Platform Deployment**

Future work may involve developing versions for both Android and iOS to maximize reach and usability across different devices.

By addressing these enhancements, Floodguard can evolve into a scalable and reliable tool for disaster preparedness and emergency response in flood-prone regions across Malaysia and beyond.

6.5 Final thoughts

The development of Floodguard, a mobile application leveraging augmented reality to assist in flood evacuation, has been a meaningful and impactful journey. This project successfully demonstrated how immersive technologies such as AR can be applied to critical real-world problems like natural disasters, offering solutions that are not only innovative but lifesaving.

Throughout this project, I have grown significantly in both technical and practical aspects. From learning how to integrate GPS and AR components in Unity to understanding user-centered design through survey-driven feedback, this project has been both a technological and personal milestone. It reinforced the importance of usability, accessibility, and user experience, especially in high-stress emergency scenarios.

Floodguard may only serve a localized region for now, but its concept and framework are scalable. The challenges encountered such as limitations in hardware compatibility, internet dependence, and area limitation on Uni Garden highlighting the real-world hurdles of deploying such systems in unpredictable environments. However, each challenge also presented an opportunity to learn and iterate toward a better solution.

This project reflects not just an academic requirement fulfilled, but a prototype with the potential to evolve into a fully operational emergency aid tool. It has sparked a broader vision for how technology can contribute meaningfully to community resilience and disaster preparedness. I hope to continue refining and expanding this work in the future, perhaps collaborating with local disaster response agencies or integrating with real-time flood data systems.

Reference

- Centria-Ammattikorkeakoulu. (2018a). Application development with vuforia and unity3d. Theseus. <http://www.theseus.fi/handle/10024/145435>
- Cervenak, R., Masek, P., & Department of Telecommunications, Brno University of Technology, Brno, Czech Republic. (2019). ARKit as indoor positioning system. In 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). <https://doi.org/10.1109/ICUMT48472.2019.8970761>
- Er, H., Lim, L. L. P., & Bong, C. H. J. (2017). A hydrology and hydraulic case study on January 2015 Flash flood in UniGarden, Kota Samarahan, Sarawak. In University Malaysia Sarawak, Proceedings of the 37th IAHR World Congress (p. 2092). IAHR. <https://www.researchgate.net/publication/319186594>
- Grubert, J., Langlotz, T., Zollmann, S., & Regenbrecht, H. (2016). Towards Pervasive Augmented Reality: Context-Awareness in Augmented Reality. In University of Applied Sciences Coburg, University of Otago, University of Otago, & University of Otago, IEEE Transactions on Visualization and Computer Graphics [Journal-article]. <https://doi.org/10.1109/TVCG.2016.2543720>
- How Virtual and Augmented Reality can Inspire Youth to Take | Environmental Migration Portal. (2024, June 5). <https://environmentalmigration.iom.int/blogs/how-virtual-and-augmented-reality-can-inspire-youth-take-action-climate-change-focus-disaster-risk-reduction-and-evacuation>
- Khanal, S., Medasetti, U. S., Mashal, M., Savage, B., & Khadka, R. (2022). Virtual and Augmented Reality in the Disaster Management Technology: A Literature Review of the Past 11 years. Frontiers in Virtual Reality, 3. <https://doi.org/10.3389/frvir.2022.843195>
- Kiryakova, G., Angelova, N., & Yordanova, L. (2018). The Potential of Augmented Reality to Transform Education into Smart Education. TEM Journal, 556–565. <https://doi.org/10.18421/tem73-11>
- Lovreglio, R., Lovreglio, R., & School of Engineering and Advanced Technology, Massey University. (2018). A Review of Augmented Reality Applications for Building Evacuation. In 17th International Conference on Computing in Civil and Building

Engineering, ICCCB E 2018, 17th International Conference on Computing in Civil and Building Engineering, ICCCB E 2018 [Conference-proceeding].

- McIntosh, G. (2023). AR Sandbox Built Within Unreal Engine 5. Wellington Faculty of Engineering Symposium. <https://ojs.victoria.ac.nz/wfes/article/view/8413>
- N. Nitika, T. K. Sharma, S. Rajvanshi and K. Kishore, "A study of Augmented Reality performance in web browsers (WebAR)," 2021 2nd International Conference on Computational Methods in Science & Technology (ICCMST), Mohali, India, 2021, pp. 281-286, doi: 10.1109/ICCMST54943.2021.00065. keywords: {Uniform resource locators;Productivity;Solid modeling;Three-dimensional displays;Publishing;Libraries;Browsers;Augmented Reality;Web-based augmented reality;ar.js;three.js;WebRTC;WebGL},
- R, A., K, S. V., K, A. J., Vp, S., & Menon, H. P. (2023). Augmented Reality based WayFinder System in Libraries. Augmented Reality Based WayFinder System in Libraries, 1740–1744. <https://doi.org/10.1109/icacrs58579.2023.10405191>
- Soni, A., Jr., Sharma, A., Jr., Kumar, P., Jr., Verma, V., Jr., & Prof. Shiv Sutar. (2014). Early Disaster warning & Evacuation system on mobile phones using Google Street Map. In International Journal of Engineering and Technical Research (IJETR): Vol. Volume-2 (Issue Issue-4, pp. 9–10) [Journal-article]. <https://www.erppublication.org>
- Sonwane, V. S. & CSE Department, PES College of Engineering, AURANGABAD. (2014). Disaster management system on mobile phones using Google Map. In IJCSIT International Journal of Computer Science and Information Technologies (Vols. 5–5, pp. 6760–6763). <https://www.ijcsit.com6760>
- Winsemius, H. C., Van Beek, L. P. H., Jongman, B., Ward, P. J., & Bouwman, A. (2013). A framework for global river flood risk assessments. Hydrology and Earth System Sciences. <https://doi.org/10.5194/hess-17-1871-2013>
- Z. Oufqir, A. El Abderrahmani and K. Satori, "ARKit and ARCore in serve to augmented reality," 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 2020, pp. 1-7, doi: 10.1109/ISCV49265.2020.9204243.

- Zhu, Y., & Li, N. (2020). Virtual and augmented reality technologies for emergency management in the built environments: A state-of-the-art review. *Journal of Safety Science and Resilience*, 2(1), 1–10. <https://doi.org/10.1016/j.jnlssr.2020.11.004>
- Lewis, J. R., & Sauro, J. (2009). The factor structure of the System Usability Scale. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 53(1), 1257–1261. <https://doi.org/10.1177/154193120905301208>

Appendix

<https://docs.google.com/forms>

System Usability Scale (SUS)

Please enter your participant number: _____

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the website today.

Explanation	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
Likert Scale	1	2	3	4	5

1. I think I would like to use this tool frequently.					
2. I found the tool unnecessarily complex.					
3. I thought the tool was easy to use.					
4. I think that I would need the support of a technical person to be able to use this system.					
5. I found the various functions in this tool were well integrated.					
6. I thought there was too much inconsistency in this tool.					
7. I would imagine that most people would learn to use this tool very quickly.					
8. I found the tool very cumbersome to use.					
9. I felt very confident using the tool.					
10. I needed to learn a lot of things before I could get going with this tool.					