

**CLEADR: AI-ENHANCED AR NAVIGATION APP
FOR SEAMLESS DRIVING**
BY
BRANDON TING EN JUNN

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

© 2025 Brandon Ting En Junn. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my supervisor, Ts Dr Saw Seow Hui and my moderator, Mr Luke Lee Chee Chien who gave me an opportunity to involve in the Augmented Reality and Artificial Intelligence fields of study. They enlightened me with their experienced knowledge and wisdom in terms of guidance and constructive criticism. I was able to overcome many of the challenges faced in this project, all thanks to their valuable advice. A million thanks to my supervisor and moderator.

Other than that, I would like to extend my heartfelt thanks to my academic advisor, Ts Dr Lim Seng Poh who encouraged and supported me on this challenging journey. He always offered the valuable academic advise that I could receive. I could not have completed this project up to its current quality without his thoughts and views on the project. My academic advisor was not only the teacher I aspired to be, but also the first one I could sincerely call a friend.

Additionally, I am deeply grateful to Ms Tay Yan Bin for her unrequited love and support throughout this project. She would always share her thoughts and opinions on the project, but never had she failed to support me, regardless of my decisions. I sincerely thank her for always standing beside me during tough times.

Lastly, I would like to extend my deepest appreciation to my family and friends for their support and companionship. They provided the greatest motivation for me to complete this project with the required perseverance and confidence. Without them, I am certain that I would not be able to persevere through the challenges faced in this project.

ABSTRACT

Navigation systems have become an essential tool for drivers to navigate through journeys with ease and confidence. However, navigation systems are far from perfect as they still pose some limitations such as ambiguous directions, insufficient real-time assistance, and poor User Experience (UX). These limitations of navigation systems often lead to confusion and uncertainty for the driver. Hence, this project proposed a solution by integrating Augmented Reality (AR) and Artificial Intelligence (AI) into a navigation system. Specifically, a mobile AR navigation application integrated with AI assistance was proposed. The development of the system adopted the agile Extreme Programming (XP) methodology that allowed for quick and iterative development, as well as ample flexibility in responding to changing requirements. Core technologies involved were Flutter, Unity, PyTorch, and TensorFlow. The development of the system was structured into 3 core modules: the Maps Module, Navigation Module, and Intelligence Module. Core features of the system included the AR navigation and lane identification. AR navigation provided clearer directions by projecting them onto the real-world environment, while lane identification provided context-awareness for effective lane change instructions. A system performance evaluation was conducted with performance metrics such as response time and accuracy. The system was responsive with an overall response time of 1.80 seconds. Additionally, a lane identification model was trained from a custom Malaysian highway roads dataset that consisted of a total 22,806 images. The model managed to obtain an accuracy of 99.21%. In summary, this project successfully developed a mobile AR navigation application integrated with AI assistance to achieve the outlined project objectives. Moreover, the project contributed a lane identification model accustomed for Malaysian highway roads with reliable accuracy.

Area of Study: Augmented Reality, Artificial Intelligence

Keywords: AR Navigation, Lane Identification, Navigation System, Mobile Application, Deep Learning, Computer Vision, Flutter, Unity, PyTorch, TensorFlow

TABLE OF CONTENTS

TITLE PAGE	i
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	ix
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1: INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement and Motivation	2
1.3 Objectives	4
1.4 Project Scope	4
1.5 Contributions	5
1.6 Report Organisation	6
CHAPTER 2: LITERATURE REVIEW	7
2.1 Review of Technologies	7
2.1.1 Mobile Operating Systems	7
2.1.2 Mobile Application Development Frameworks	9
2.1.3 AR Development Frameworks	12
2.1.4 AI Development Frameworks	14
2.1.5 Summary of Technologies Review	16
2.2 Review of Existing Systems	18
2.2.1 Google Maps	19
2.2.2 Waze Navigation & Live Traffic	20
2.2.3 Sygic GPS Navigation & Maps	21
2.2.4 MBUX Navigation System	23
2.2.5 Intelligent Collision Avoidance and Manoeuvring System	24

2.2.6	Summary of Existing Systems Review	25
CHAPTER 3:	SYSTEM METHODOLOGY	26
3.1	Methodology	26
3.2	System Requirements	28
3.2.1	Hardware	28
3.2.2	Software	29
3.3	Functional Requirements	30
3.4	Non-Functional Requirements	31
3.5	Project Milestones	32
3.5.1	Project I	32
3.5.2	Project II	33
CHAPTER 4:	SYSTEM DESIGN	34
4.1	System Architecture	34
4.2	System Flowcharts	37
4.2.1	System Flowchart	37
4.2.2	Maps Module Flowchart	38
4.2.3	Navigation Module Flowchart	39
4.2.4	Intelligence Module Flowchart	40
4.2.5	Lane Identification Model Flowchart	41
4.2.6	Services Flowchart	42
4.3	Storyboard	43
4.4	Use Case Diagram	44
4.5	Activity Diagram	45
4.6	Wireframe Design	46
4.7	Low-Fidelity Prototype	47
4.8	High-Fidelity Prototype	48
CHAPTER 5:	SYSTEM IMPLEMENTATION	49
5.1	Software Installation and Setup	49
5.1.1	Android Studio	49
5.1.2	Visual Studio Code	50

5.1.3	Flutter	50
5.1.4	Unity	52
5.1.5	CUDA	53
5.1.6	cuDNN	54
5.1.7	Python	54
5.1.8	Jupyter Notebook	55
5.1.9	PyTorch	56
5.1.10	WSL	56
5.1.11	TensorFlow	57
5.2	Settings and Configurations	57
5.2.1	Dependencies	57
5.2.2	Google Maps API	59
5.2.3	Flutter Unity Widget	61
5.2.4	AR + GPS Location Asset	67
5.2.5	TFLite	67
5.3	System Operation	69
5.4	Implementation Issues and Challenges	85
5.5	Concluding Remarks	85
CHAPTER 6:	SYSTEM EVALUATION AND DISCUSSION	86
6.1	System Performance Evaluation	86
6.1.1	Response Time	86
6.1.2	Accuracy	87
6.2	System Testing Setup and Results	88
6.3	Project Challenges	92
6.4	Objectives Evaluation	93
6.5	Concluding Remarks	94
CHAPTER 7:	CONCLUSION AND RECOMMENDATIONS	95
7.1	Conclusion	95
7.2	Recommendations	97
REFERENCES		98

APPENDICES

	A-1
Appendix A: Wireframe Design	A-1
Appendix B: Low-Fidelity Design	B-1
Appendix C: High-Fidelity Design	C-1
Appendix D: GitHub Repository	D-1
Appendix E: Poster	E-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 1.1	Samsung In-Vehicle AR Experience Concept	2
Figure 2.1	Mobile Operating System Market Share Worldwide in 2024	7
Figure 2.2	Android Logo	8
Figure 2.3	iOS Logo	8
Figure 2.4	Android Studio Logo	9
Figure 2.5	Java Logo	9
Figure 2.6	Kotlin Logo	9
Figure 2.7	Gradle Logo	9
Figure 2.8	Xcode Logo	10
Figure 2.9	Swift Logo	10
Figure 2.10	Flutter Logo	10
Figure 2.11	Dart Logo	11
Figure 2.12	Visual Studio Code Logo	11
Figure 2.13	React Native Logo	11
Figure 2.14	Unity Logo	12
Figure 2.15	C# Logo	12
Figure 2.16	ARCore Logo	12
Figure 2.17	ARKit Logo	13
Figure 2.18	Unreal Engine Logo	13
Figure 2.19	C++ Logo	13
Figure 2.20	PyTorch Logo	14
Figure 2.21	Python Logo	14
Figure 2.22	Jupyter Notebook Logo	14
Figure 2.23	EfficientNet Architecture	15
Figure 2.24	TensorFlow Logo	15
Figure 2.25	Keras Logo	16
Figure 2.26	Google Maps Logo	19
Figure 2.27	Live View	19

Figure 2.28	Waze Navigation & Live Traffic Logo	20
Figure 2.29	Sygic GPS Navigation & Maps Logo	21
Figure 2.30	AR SmartCam	22
Figure 2.31	Road Sign Detection	22
Figure 2.32	MBUX AR Navigation	23
Figure 2.33	Intelligent Collision Avoidance and Manoeuvring System	24
Figure 3.1	Agile XP Process Flow Diagram	27
Figure 3.2	Agile XP Release Cycle Process	27
Figure 3.3	Project I Gantt Chart	32
Figure 3.4	Project II Gantt Chart	33
Figure 4.1	System Architecture	34
Figure 4.2	System Flowchart	37
Figure 4.3	Maps Module Flowchart	38
Figure 4.4	Navigation Module Flowchart	39
Figure 4.5	Intelligence Module Flowchart	40
Figure 4.6	Lane Identification Model Flowchart	41
Figure 4.7	Services Flowchart	42
Figure 4.8	A Trip to Kuala Lumpur	43
Figure 4.9	Use Case Diagram	44
Figure 4.10	Activity Diagram	45
Figure 4.11	Wireframe Design	46
Figure 4.12	Low-Fidelity Prototype	47
Figure 4.13	High-Fidelity Prototype	48
Figure 5.1	NDK 27.0.12.077973	49
Figure 5.2	Flutter Extension	50
Figure 5.3	Flutter	51
Figure 5.4	Flutter Project	51
Figure 5.5	Java and Gradle Conflict Warning Message	52
Figure 5.6	Unity 2022.3.55f1	52
Figure 5.7	Unity Project	53
Figure 5.8	CUDA	53
Figure 5.9	CUDA Directory	54

Figure 5.10	Python	54
Figure 5.11	Jupyter Notebook	55
Figure 5.12	“current_lane” Directory	55
Figure 5.13	WSL	56
Figure 5.14	Dart Packages	58
Figure 5.15	Google Maps API Key Settings	59
Figure 5.16	AndroidManifest.xml	60
Figure 5.17	Player Settings	61
Figure 5.18	Build.cs Configuration 1	62
Figure 5.19	Build.cs Configuration 2	62
Figure 5.20	Exported Unity Project 1	63
Figure 5.21	Exported Unity Project 2	63
Figure 5.22	settings.gradle	64
Figure 5.23	build.gradle (Project)	65
Figure 5.24	build.gradle (Application) Configuration 1	65
Figure 5.25	build.gradle (Application) Configuration 2	66
Figure 5.26	MainActivity.kt	66
Figure 5.27	AR + GPS Location Asset	67
Figure 5.28	export.sh	68
Figure 5.29	manifest.json	68
Figure 5.30	Loading Screen	69
Figure 5.31	Loading – Failed Screen	70
Figure 5.32	Loading – Failed – Application Settings Screen	71
Figure 5.33	Maps Screen	72
Figure 5.34	Maps – Selected Screen	73
Figure 5.35	Maps – Search Screen	74
Figure 5.36	Maps – Search – Selected Screen	75
Figure 5.37	Maps – Directions Screen	76
Figure 5.38	Maps – Navigation Screen	77
Figure 5.39	Maps Navigation – Horizontal Screen	78
Figure 5.40	AR Navigation – Blue Arrow Screen	79
Figure 5.41	AR Navigation – Yellow Arrow Screen	80
Figure 5.42	AR Navigation – Red Arrow Screen	81

Figure 5.43	AR Navigation – Horizontal – Blue Arrow Screen	82
Figure 5.44	AR Navigation – Horizontal – Yellow Arrow Screen	82
Figure 5.45	AR Navigation – Horizontal – Red Arrow Screen	83
Figure 5.46	AR Navigation – Lane Change Screen	84

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Summary of Mobile Operating Systems Review	16
Table 2.2	Summary of Mobile Application Development Frameworks Review	17
Table 2.3	Summary of AR Development Frameworks Review	17
Table 2.4	Summary of AI Development Frameworks Review	18
Table 2.5	Summary of Existing Systems Review	25
Table 3.1	Laptop Specifications	28
Table 3.2	Smartphone Specifications	28
Table 3.3	Software Specifications	29
Table 6.1	Average Response Time	87
Table 6.2	Lane Identification Model Accuracy	88
Table 6.3	Search Place Test Case	89
Table 6.4	Select Destination Test Case	90
Table 6.5	Preview Route Test Case	90
Table 6.6	Start Maps Navigation Test Case	91
Table 6.7	Start AR Navigation Test Case	91

LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>API</i>	Application Programming Interface
<i>APK</i>	Android Package Kit
<i>AR</i>	Augmented Reality
<i>ETA</i>	Estimated Time of Arrival
<i>GPS</i>	Global Positioning System
<i>GPU</i>	Graphics Processing Unit
<i>IDE</i>	Integrated Development Environment
<i>MBUX</i>	Mercedes-Benz User Experience
<i>ONNX</i>	Open Neural Network Exchange
<i>SDK</i>	Software Development Kit
<i>TFLite</i>	TensorFlow Lite
<i>UI</i>	User Interface
<i>UX</i>	User Experience
<i>WSL</i>	Windows Subsystem for Linux
<i>XP</i>	Extreme Programming

Chapter 1

Introduction

This chapter includes the background, problem statement and motivation, objectives, project scope, contributions, and report organisation.

1.1 Background

Navigation is the field of study that determines the position of an entity and guiding it towards a specific destination through a sequence of well-defined steps, instructions, or directions [1]. In the past, ancient navigation used celestial navigation that involved the observation of celestial bodies such as stars to determine positions and directions. Technological advancements have made it possible for navigation to be more accurate, reliable, and accessible through navigation systems. Navigation systems such as mobile navigation applications are widely used to navigate journeys with ease. Service-oriented companies such as Grab and foodpanda rely heavily on mobile navigation applications to sustain their daily food delivery operations.

Navigation systems are supported by the Global Positioning System (GPS) that consists of over 30 satellites orbiting Earth [2]. GPS provides real-time location information with accuracy to support the navigation functionality. In 1990, Mazda's Eunos Cosmo was the 1st production car that utilised a GPS-based navigation system to provide the navigation functionality for drivers [3]. It pioneered the revolution of automotive navigation in the industry. Subsequently, the evolution of navigation systems has led to the emergence of lightweight and convenient mobile navigation applications. Moreover, the dynamic rerouting feature in mobile navigation applications are able to suggest alternative routes when unexpected events happen to minimise the Estimated Time of Arrival (ETA) delays.

Augmented Reality (AR) is a technology that provides an interactive and immersive way of conveying information. It serves as a bridge between the virtual and real world through overlaying digital objects onto the physical environment. For example, Figure 1.1 shows a concept by Samsung on the future of in-vehicle experience with AR. The

concept shows the AR navigation feature that projects the directions directly on the road from the driver's point of view.

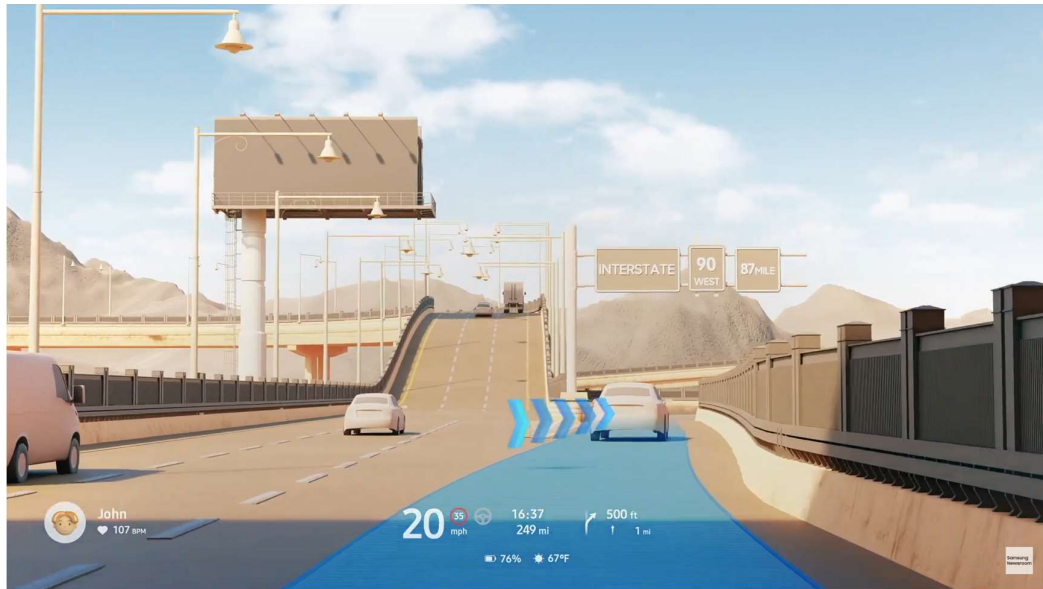


Figure 1.1: Samsung In-Vehicle AR Experience Concept [4]

Artificial Intelligence (AI) is a technology that enables machines to simulate human intelligence. Intelligent machines are able to perform tasks such as classification, regression, clustering, association, and dimensionality reduction [5]. AI improves the efficiency and effectiveness of performing daily tasks. Some practical applications of AI are chatbot, autonomous driving, recommendation systems, and language translation. For example, Tesla's Autopilot feature allows for autonomous driving through computer vision to assist with steering, acceleration, and braking of the car.

1.2 Problem Statement and Motivation

The term "Death by GPS" was discussed by Milner [6] that referred to individuals who become lost or face danger due to the over-reliance on navigation systems. According to Lin et al. [7], they found that navigation systems have the inability to navigate through complex geographics contexts. Complex geographic contexts refer to intricate environments that can easily cause confusion and uncertainty. Due to the over-reliance on navigation systems, problems such as ambiguous directions, insufficient real-time assistance, and poor User Experience (UX).

Ambiguous Directions

Navigation systems instruct the driver through 2 primary channels: visual displays and audio cues. Visual displays show the map and routes, while audio cues provide voice instruction for upcoming turns. For example, consider a scenario where the navigation system instructs the driver to take a right turn in 200 metres. As the driver approaches, it turns out there are two closely spaced right turns. This scenario causes confusion and uncertainty due to the ambiguous directions given by the navigation system. Though a simple example, real-life scenarios often involve complex geographic contexts, such as intricate intersections, multiple exits, and lane changes.

Insufficient Real-Time Assistance

Navigation systems provide relevant real-time information such as weather updates, speed limits, and traffic congestion reports to inform the driver about the changing road conditions. This assists the driver prepare for precautionary actions such as slowing down in bad weather, adjusting to varying speed limits, and anticipating traffic congestion. However, navigation systems lack in providing real-time assistance in the form of monitoring and detection of road conditions. For example, navigation systems are unable to reassure whether the driver is on the correct lane to exit on the highway. The lack of reassurance creates uncertainty, potentially leading to confusion.

Poor UX

UX refers to the holistic relationship between the product, service, or company and the person, typically a user that includes their perceptions, emotions, and interactions to accomplish a certain goal [8]. The quality attributes of UX that were mentioned in [8] include accessibility, usability, credibility, satisfaction, and usefulness. Problems of navigation systems that cause confusion and uncertainty can have a negative impact on the UX for the driver.

1.3 Objectives

- **To Develop a Mobile AR Navigation Application.**

The primary objective of this project is to develop a mobile navigation application that integrates AR navigation features to provide improved clarity in conveying directions. Ultimately, it focuses on reducing confusion and uncertainty of navigation systems. In addition to that, a mobile application allows for wide accessibility without the need for specialised hardware such as embedded systems, integrated cameras, ultrasonic sensors, and radars.

- **To Increase the Effectiveness of AR Navigation through AI Integration.**

AI was integrated into this project to support the AR navigation feature. This project utilises AI to simulate context-aware AR directions that is able monitor and detect road conditions, then reassures the driver was on the correct lane.

- **To Improve the UX.**

This project revolves around improving the clarity of directions in navigations systems for a better UX. By reducing the likelihood of making a wrong turn, an improved UX can be achieved for navigation systems.

1.4 Project Scope

This project delivered a mobile AR navigation application integrated with AI. Several assumptions and requirements were established to define the feasibility and boundaries of the project.

First, the target user of this project was the driver. Hence, the primary transportation mode focused was driving. Other transportation modes such as walking, biking, and public transit were not considered in this project. This project relied heavily on the Google Maps Application Programming Interface (API) to provide maps, places, routes, and navigations on a global scale. Additionally, the AR navigation feature was set to distinguish roads that were minimum 50 metres and maximum 500 metres apart from each other. On the other hand, the AI was trained on a custom Malaysian highway roads dataset with a maximum of 5 lanes.

Second, the mobile device should support both AR and GPS. Moreover, the rear camera of the mobile device should be functional to capture the front view of the road. It was important for the mobile device to always be mounted on the dashboard so that the front view of the road could always be captured.

Third, the mobile device should be Android, specifically Android versions 14.0 and above. This project complied with the target API level requirements of Google Play at the time of development. The mobile device should have Internet access, location and camera permissions enabled.

1.5 Contributions

First, this project proposed a mobile AR navigation application integrated with AI to address the problems of navigation systems. It reduced the confusion and uncertainty of navigation systems through providing improved clarity in conveying directions. This provided an alternative solution for drivers. In addition to that, this project demonstrated the feasibility of integrating AR and AI into navigation systems.

Second, this project explored an alternative implementation approach to AR navigation. Typically, AR navigation is implemented indoors, constrained to predefined and specific areas that require the design of digital environment from the physical environment. Therefore, this project overcame the limitation of area-specific restrictions, enabling AR navigation on a global scale.

Third, this project was open-sourced on GitHub. It served as a steppingstone for AR and AI-based mobile navigation applications. It could also be customised to cater for specific use cases, allowing for code reuse and flexibility. Moreover, this project could be used as a tool for daily navigation needs.

Fourth, this project developed a lane identification model that was trained on a custom Malaysian roads dataset up to a maximum of five lanes. It was able to classify the current lane based on the front view of the road.

1.6 Report Organisation

This report is organised into 7 chapters:

- Chapter 1: Introduction
- Chapter 2: Literature Review
- Chapter 3: System Methodology
- Chapter 4: System Design
- Chapter 5: System Implementation
- Chapter 6: System Evaluation and Discussion
- Chapter 7: Conclusion and Recommendations

The first chapter includes the background, problem statement and motivation, objectives, contributions, project scope, and report organisation. The second chapter reviews the technologies and existing systems. The third chapter outlines the methodology, system requirements, functional requirements, non-functional requirements, and project milestones. The fourth chapter illustrates the system architecture, system flowcharts, storyboard, use case diagram, activity diagram, wireframe design, low-fidelity prototype, and high-fidelity prototype. The fifth chapter demonstrates the software installation and setup, settings and configurations, system operation, implementation issues and challenges, and concluding remarks. The sixth chapter discusses system performance evaluation, system testing setup and results, project challenges, objectives evaluation, and concluding remarks. The seventh chapter summarises the conclusion and recommendations.

Chapter 2

Literature Review

This chapter reviews the technologies and existing systems.

2.1 Review of Technologies

Technologies such as mobile operating systems, mobile application development frameworks, AR development frameworks, and AI development frameworks were reviewed to be selected for the development of the system.

2.1.1 Mobile Operating Systems

Android and iOS are the most widely used operating systems for mobile devices. Figure 2.1 shows the bar chart for the mobile operating system market share worldwide in 2024. According to Figure 2.1, Android accounted for 71.56%, while iOS held for 27.81% of the market share worldwide. International Data Corporation [10] stated that the market share worldwide for Android was projected to grow by 2.3% in 2025, totaling up to 1.26 billion devices. Based on the statistics, Android is the dominant operating system for mobile devices.

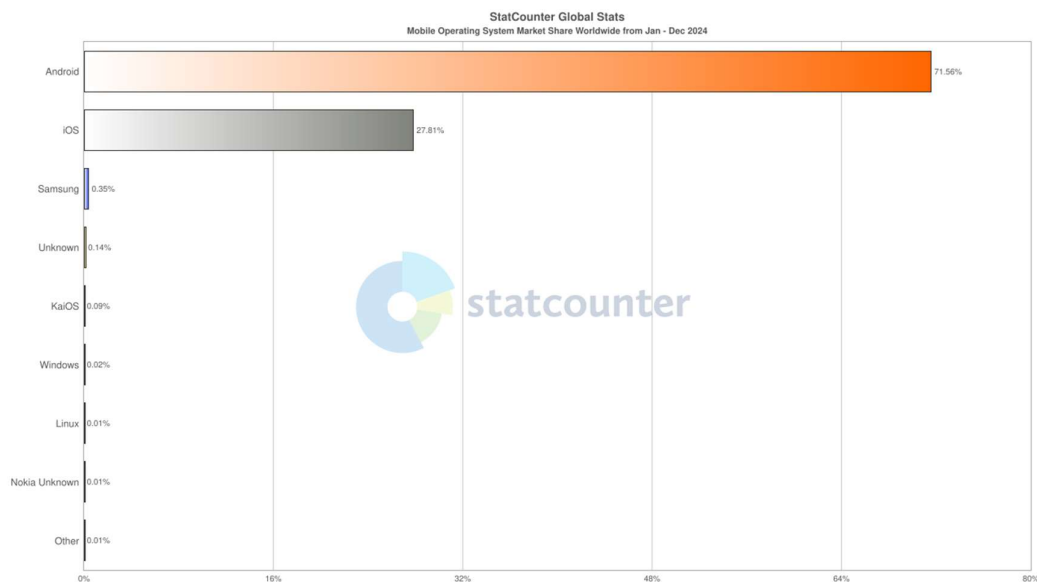


Figure 2.1: Mobile Operating System Market Share Worldwide in 2024 [9]

Android

Android was developed in 2003 by Android Inc., a startup company in Palo Alto, California [11]. However, due to the insufficient fundings, Google acquired Android for 50 million in United States dollars in 2005 [11]. Smartphone manufacturers such as Samsung, Google, OnePlus, Xiaomi, Oppo, and Vivo use Android as their mobile operating system. Figure 2.2 shows the logo of Android.



Figure 2.2: Android Logo [12]

Android advocates for an open system that allows flexible customisation and modification to the operating system. For example, Android mobile devices can install mobile applications through the Android Package Kit (APK) files. However, it also makes Android devices vulnerable to security threats. According to Lee [13], there were 128 scamming cases targeted at Android users, a totaling loss of 2.4 million in Singapore dollars which was equivalent to 1.8 million in United States dollars, due to the installation of malicious APK files advertised from Facebook or TikTok scammers.

iOS

iOS is the mobile operating system developed by Apple that was originally named “iPhone OS” for powering the 1st-generation iPhone in June 2007 [14]. iOS is the state-of-the-art mobile operating system that pioneered the era of smartphones. Figure 2.3 shows the logo of iOS.



Figure 2.3: iOS Logo [15]

Conversely, its key characteristics is the closed system that ensures tight security. For example, mobile applications can only be installed on iOS devices through the official App Store by Apple. No other third-party platform can provide mobile

application installations on iOS devices. Therefore, it is able to achieve high security but comes with the cost of limited customisation.

2.1.2 Mobile Application Development Frameworks

Android Studio

Android Studio is the official Integrated Development Environment (IDE) developed by Google and JetBrains for Android application development [16]. Figure 2.4 shows the logo of Android Studio.



Figure 2.4: Android Studio Logo [17]

It supports platforms such as Windows, macOS, and Linux. The primary programming languages used are Java and Kotlin. The IDE is equipped with a range of tools for Android development such as a code editor, debugger, Android emulator, Android Software Development Kit (SDK), and Gradle. Gradle is an automated build tool that manages tasks such as compilation, packaging, testing, deployment, and publishing of Android applications [18]. Figure 2.5, Figure 2.6, and Figure 2.7 show the logos of Java, Kotlin, and Gradle respectively.



Figure 2.5: Java Logo [19]



Figure 2.6: Kotlin Logo [20]



Figure 2.7: Gradle Logo [21]

Xcode

Xcode is the official IDE developed by Apple for iOS application development. It can also develop other Apple platform applications such as iPadOS, macOS, tvOS, visionOS, and watchOS. Figure 2.8 shows the logo of Xcode.



Figure 2.8: Xcode Logo [22]

The primary programming languages used are Objective-C and Swift. Xcode has the tools of a code editor, debugger, simulator, Apple SDK, compiler, and linker [22]. However, it is only available on the Mac App Store, limiting the platform to macOS [22]. Xcode is renowned for its ability to build projects faster than Android Studio projects due to its optimised compiler and linker on the multicore architecture of Apple silicon hardware [22]. Figure 2.9 shows the logo of Swift.



Figure 2.9: Swift Logo [23].

Flutter

Flutter is an open-source SDK developed by Google that was released in December 2018 [24]. Figure 2.10 shows the logo of Flutter.



Figure 2.10: Flutter Logo [25]

Flutter advocates for cross-platform development from a single codebase for multiple platforms such as Android, iOS, web, Windows, macOS, Linux, and embedded systems [26]. The primary programming language used is Dart. Flutter can be seamlessly integrated through plug-ins with code editors or IDEs such as Visual Studio Code or Android Studio. Flutter can achieve high performance due to its

framework architecture of compiling to native code with its rendering engine, Skia to bypass the platform User Interface (UI) layers [27]. This allows for smooth, consistent 60 frames per second experiences across multiple platforms [27]. However, its performance is not always up to par with fully native development environments such as Android Studio or Xcode [28]. Figure 2.11 and Figure 2.12 show the logos of Dart, Visual Studio Code, and Skia respectively.



Figure 2.11: Dart Logo [29]



Figure 2.12: Visual Studio Code Logo [30]

React Native

React Native is an open-source framework released in March 2015 developed by Meta Platforms, which was formerly known as Facebook [31]. Figure 2.13 shows the logo of React Native.

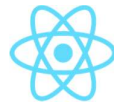


Figure 2.13: React Native Logo [32]

Similar to Flutter, it supports cross-platform development for Android, iOS, web, Windows, and macOS [32]. It is based on the JavaScript library, React, that builds the UI by updating only parts of the screen that change. The primary programming language is JavaScript. React Native has moderate performance on its own, but lower performance when compared to Flutter due to its reliance on the communication bridge between JavaScript and native modules that introduces latency and inconsistency [33].

2.1.3 AR Development Frameworks

Unity

Unity is a cross-platform game engine developed by Unity Technologies in June 2005, originally for the Mac OS X and iPhone [34]. Figure 2.14 shows the logo of Unity.



Figure 2.14: Unity Logo [35]

Unity has extended its support to other platforms such as mobile, desktop, consoles, web, and embedded systems [36]. Unity is renowned for mobile and indie game development due to its user-friendly interface [37]. Unity can be installed on platforms such as Windows, macOS, and Linux. The programming language used is C#. Unity Scripts are written in C# that can be used to manipulate the behaviour of components known as GameObjects. Popular games such as Pokémon Go, Among Us, Genshin Impact, Heartstone, Fall Guys: Ultimate Knockout, and Cuphead were developed with Unity. Figure 2.15 shows the logo of C#.



Figure 2.15: C# Logo [38]

Unity can be used to develop simulations and extended reality applications. For example, Unity provides the necessary tools such as AR foundation to develop mobile AR applications through utilising SDKs such as ARCore and ARKit [39]. Figure 2.16 and Figure 2.17 show the logos of ARCore and ARKit respectively.



Figure 2.16: ARCore Logo [40]



Figure 2.17: ARKit Logo [41]

Unreal Engine

Unreal Engine is a cross-platform game engine developed by Epic Games in May 1998 [42]. Figure 2.18 shows the logo of Unreal Engine.



Figure 2.18: Unreal Engine Logo [43]

Unreal Engine supports platforms such as Android, iOS, Windows, macOS, Linux, PlayStation, Xbox, AR, and virtual reality [44]. Unreal Engine is renowned for being the preferred choice of making high quality games, commonly known as triple-A games. Unreal Engine can be installed on platforms such as Windows, macOS, and Linux. The programming language used is C++. It is used to write game logic and behaviour of Unreal Engine applications. Figure 2.19 shows the logo of C++.



Figure 2.19: C++ Logo [45]

Popular games such as Black Myth: Wukong, PUBG: Battlegrounds, Fortnite, Final Fantasy VII, Rocket League, and Borderlands were developed with Unreal Engine. It has photorealistic rendering capabilities to create the stunning, high-quality visuals with advanced lighting, textures, and effects of a triple-A game [46]. However, Unreal Engine comes with a steep learning curve for beginners compared to Unity [47].

2.1.4 AI Development Frameworks

PyTorch

PyTorch is an open-source machine learning framework developed by Meta AI in 2017 that was based on the Torch library [48]. It is popularly used to build and train deep learning models such as neural networks. Figure 2.20 shows the logo of PyTorch.



Figure 2.20: PyTorch Logo [49]

The primary programming language is Python. It can also be used through Jupyter Notebook. PyTorch also allows for more efficient computations through the C++ API [50]. Intelligent applications such as the Tesla's Autopilot were developed using PyTorch [51]. PyTorch offers two primary features which are tensor computational graphs and automatic differentiation to develop state-of-the-art deep learning models [50]. Additionally, it has Graphics Processing Unit (GPU) support for platforms such as Windows, macOS, and Linux [52]. Figure 2.21 and Figure 2.22 show the logos of Python and Jupyter Notebook respectively.



Figure 2.21: Python Logo [53]



Figure 2.22: Jupyter Notebook Logo [54]

PyTorch has integrated popular deep learning models that are used for different tasks such as classification, semantic segmentation, instance segmentation, keypoint detection, and video classification [55]. Hyperparameter settings can be set to use specific pre-trained weights of each model for faster training time and improved performance. Classification tasks include models such as EfficientNet, MobileNet,

ResNet, and VGG. EfficientNet is known for its strong accuracy while maintaining computational efficiency [56]. Figure 2.23 shows the architecture of EfficientNet. This makes EfficientNet a suitable candidate model for mobile devices that have limited computing resources.

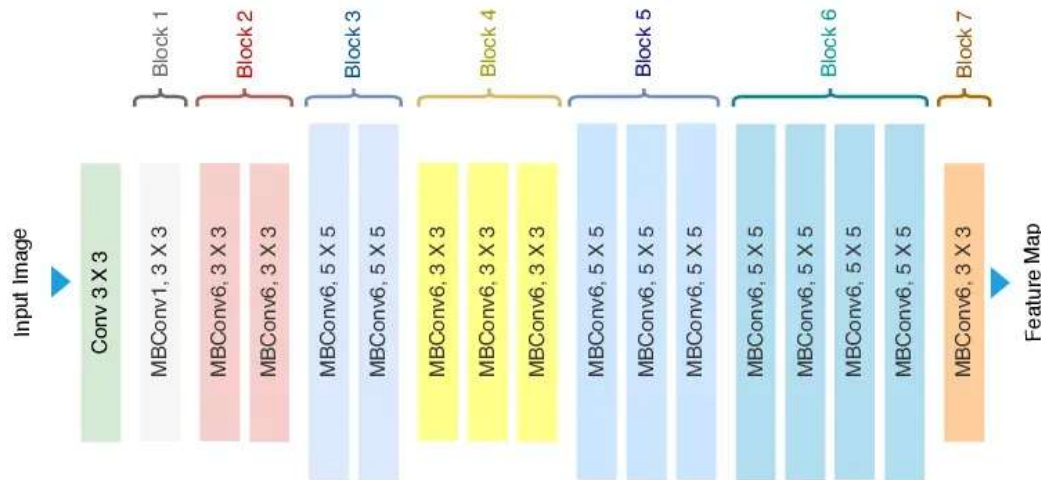


Figure 2.23: EfficientNet Architecture [57]

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google in 2015 [58]. It can be used to train, perform inference, and deploy deep learning models. Figure 2.24 shows the logo of TensorFlow.



Figure 2.24: TensorFlow Logo [59]

The primary programming language used by TensorFlow is Python, JavaScript, C++, and Java [60]. Other than that, Jupyter Notebook can also be used to run TensorFlow. In 2019, Google announced TensorFlow 2.0 that introduced significant improvements with power tool integrations and a strong emphasis on ease of use [61]. For example, TensorFlow 2.0 has tight integrations with Keras, a high-level deep learning API that makes it easier for machine learning development [61]. Additionally,

TensorFlow provides high performance runtime for on-device machine learning known as TensorFlow Lite (TFLite), now known as LiteRT [62]. However, GPU support for TensorFlow is only available on Linux platforms since TensorFlow version 2.11 [63]. Figure 2.25 shows the logo of Keras.





Figure 2.25: Keras Logo [64]

2.1.5 Summary of Technologies Review

Mobile Operating Systems

Table 2.1 shows the summary of the review on mobile operating systems between Android and iOS. It compares and contrasts the characteristics of the developer, release year, market share worldwide, system architecture, customisation, performance, and security.

Table 2.1: Summary of Mobile Operating Systems Review

Operating System	Android	iOS
Characteristic		
Developer	Google	Apple
Release Year	2003	2007
Market Share Worldwide (2024)	71.56%	27.81%
System Architecture	Open system	Closed system
Customisation	High	Limited
Performance	Varies by device	Consistent, optimised
Security	Vulnerable	Tight, secure

Android was selected for this project due to it being the dominant mobile operating system. In addition to that, the open system of Android allows for wide customisation.

Mobile Application Development Frameworks

Table 2.2 shows the summary of the review on mobile application development frameworks between Android Studio, Xcode, Flutter, and React Native. It shows the comparisons on the aspects of cross-platform, supported platform, and performance.

Table 2.2: Summary of Mobile Application Development Frameworks Review



Framework	Android Studio	Xcode	Flutter	React Native
Aspect				
Cross-Platform	✗	Apple platforms only	✓	✓
Supported Platforms	Windows, macOS, Linux	macOS only	Windows, macOS, Linux	Windows, macOS, Linux
Performance	Native for Android	Native for iOS	Good	Moderate

Flutter was selected for this project due to its support for cross-platform development and good performance. However, Android Studio was also required due to Flutter being dependent on the Android SDK for Android application development. Cross-platform development allows for seamless expansion to other platforms such as iOS.

AR Development Frameworks

Table 2.3 shows the summary of the review on AR development frameworks between Unity and Unreal Engine. It shows the comparisons on the aspects of cross-platform, supported platform, and ease of use.

Table 2.3: Summary of AR Development Frameworks Review

Framework	Unity	Unreal Engine
Aspect		
Cross-Platform	✓	✓
Supported Platforms	Windows, macOS, Linux	Windows, macOS, Linux



Ease of Use	Beginner-friendly	Steep learning curve
--------------------	-------------------	----------------------

Unity was selected for this project as its beginner-friendly interface allows for easy implementation of AR. Moreover, it supports cross-platform development.

AI Development Frameworks

Table 2.4 shows the summary of the review on AI development frameworks between PyTorch and TensorFlow. It shows the comparisons on the aspects of development tools, specialty, and GPU support.

Table 2.4: Summary of AI Development Frameworks Review

Framework	PyTorch	TensorFlow
Aspect		
Development Tools	✓	✓
Specialty	Computational graphs, automatic differentiation	High-level API (Keras), High-performance deployment (TFLite)
GPU Support	Windows, macOS, Linux	Linux

PyTorch and TensorFlow were included in this project to utilise each of their specialty features such as computational graphs, automatic differentiation, and high-performance deployment. PyTorch was responsible for the development and training of the AI model while TensorFlow was responsible for the deployment of the AI model on mobile devices. Moreover, PyTorch has GPU support for various platforms to reduce training time. EfficientNet-B0 was selected for this project as the classification model due to its efficient computation and lightweight architecture.

2.2 Review of Existing Systems

Existing systems such as Google Maps, Waze Navigation & Live Traffic, Sygic GPS Navigation & Maps, Mercedes-Benz User Experience (MBUX) Navigation System, together with the Intelligent Collision Avoidance and Manoeuvring System were reviewed for the proposed system.

2.2.1 Google Maps

Google Maps was released in April 2006 as a mobile navigation application [65]. As of August 2024, Google Maps had over 1 billion monthly active users worldwide, with an approximately 67% of mobile device users prefer Google Maps as their primary navigation tool [66]. Google Maps has emerged to become one of the most popular mobile navigation applications on the market. Figure 2.26 shows the logo of Google Maps.



Figure 2.26: Google Maps Logo [67]

Google Maps provides AR navigation for walking through the Live View feature. Figure 2.27 shows the demonstration of the feature. It projects the directions directly on the road from the pedestrian's point of view. This provides clear and descriptive directions to the user, making it easy to follow. However, this feature is only available for walking. Moreover, the feature is restricted to only areas that support the Google Street View.



Figure 2.27: Live View [68]

Google Maps provides real-time information such as indications of accidents and road traffic reports with accuracy and reliability. Multiple data sources such as satellite data, road sensors, and user-reported incidents are gathered by Google Maps to provide the information [69]. However, it does not provide monitoring and detection of road conditions.

On the other hand, Google Maps is supported globally as it is available in over 220 countries and territories that span more than 30 million miles (48.3 million kilometres) of roads [68]. Additionally, Google Maps performs monthly updates on the maps for over 200 countries [68]. These extensive efforts allow Google Maps to obtain complete, accurate, and reliable information on geographical roads and places on Earth for its navigation functionality. Despite these efforts, Google Maps is still susceptible to causing confusion and uncertainty during navigation. For example, its lane assist feature indicates the which lane to take but fails to reassure the driver whether the vehicle is in the correct lane, therefore the likelihood of a wrong turn always exists. Moreover, Google Maps is known for its UI designs to be cluttered with information, causing confusion and uncertainty that contributes to poor UX.

2.2.2 Waze Navigation & Live Traffic

Waze Navigation & Live Traffic, widely known as Waze, is a mobile navigation application that was acquired by Google in June 2013 [70]. According to [71], Waze had an estimated 140 million monthly users as of April 2024. Waze is widely known for its relevant real-time information and minimalistic UI. Figure 2.28 shows the logo of Waze.



Figure 2.28: Waze Navigation & Live Traffic Logo [72]

Waze does not provide any AR navigation feature. However, its strength is the ability to provide relevant real-time information such as police roadblock, speed cameras, traffic and accident reports, and weather updates that are customisable to the user's preference. Despite that, it still has the limitation of insufficient real-time

assistance on monitoring and detection of road conditions. To maintain its source of relevant real-time information, Waze implemented a community-driven approach [73]. For example, Waze has a feature that allows its users to update and report incidents to publicly warn other users. Therefore, it depends on the number of Waze users in the given area to provide the relevant real-time information. If there are insufficient users in the area, Waze performs poorly in terms of its real-time information.

2.2.3 Sygic GPS Navigation & Maps

Sygic GPS Navigation & Maps, also known as Sygic, is a mobile navigation application that utilises an offline map for its navigation. The offline map allows for the navigation functionality to perform without the need for Internet connection, unlike Google Maps and Waze. However, multiple offline maps are required to be downloaded locally on the mobile device, occupying the storage space and frequent manual updates to the offline maps. According to Sygic [74], it has 2.5 million monthly active users. However, Sygic is a paid subscription application, making its features widely inaccessible compared to free applications such as Google Maps and Waze. Figure 2.29 shows the logo of Sygic.

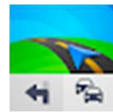


Figure 2.29: Sygic GPS Navigation & Maps Logo [75]

Sygic has an AR SmartCam feature that provides AR navigation. Figure 2.30 shows a demonstration of the feature. However, the directions shown are not inaccurate and cluttered, causing confusion and uncertainty during navigation. Therefore, the feature does not provide clear directions from the AR SmartCam feature. Additionally, [76] stated that its map navigation provided misleading directions and information on multiple occasions, indicating the poor reliability of Sygic's navigation functionality.



Figure 2.30: AR SmartCam [77]

Sygyic provides real-time assistance of monitoring and detection through its road sign detection feature. For example, when a speed limit road sign is passed, it is detected and highlighted in the navigation UI as a reminder for the user. Figure 2.31 shows a demonstration of the road sign detection feature.

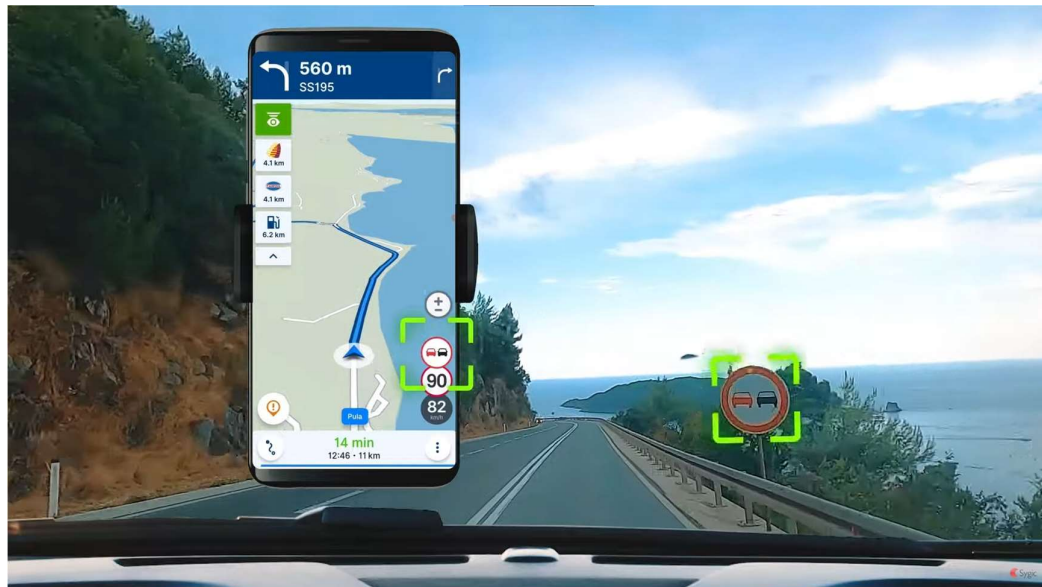


Figure 2.31: Road Sign Detection [78]

However, its real-time assistance is only limited to detecting road signs. Additionally, the reliability of the feature depends on external factors such as placement, light disparity, weather conditions, and cleanliness of the windshield [79].

2.2.4 MBUX Navigation System

The MBUX Navigation System is exclusive to the Mercedes-Benz series that has a built-in navigation system. It features AR navigation through utilising built-in sensors and cameras of the car. Figure 2.32 shows the demonstration of the MBUX AR navigation feature.

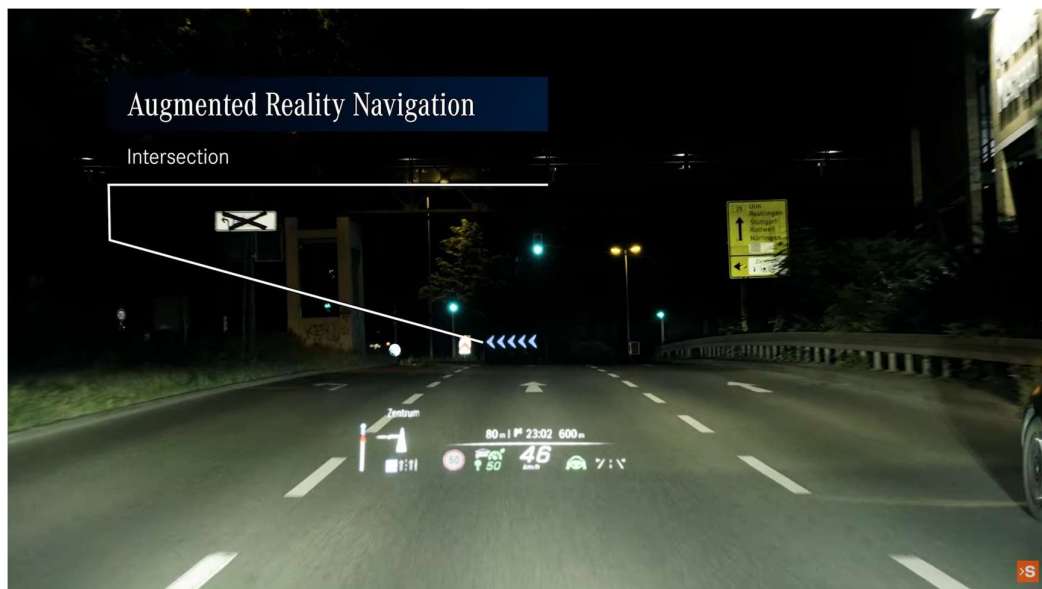


Figure 2.32: MBUX AR Navigation [80]

The AR directions are projected on the road from the driver's point of view, providing clear and certain navigation. Ease of navigation can be achieved even in complex intersections or unfamiliar streets. The MBUX Navigation System enhances the UX by ensuring a confident driving experience for the driver [81].

The MBUX Navigation System also provides real-time information such as indications of accidents, traffic congestion, and road closures. Dynamic route adjustments known as dynamic rerouting are able to reduce the impact of unexpected events that normally delay the ETA [81]. However, the MBUX Navigation System shows insufficient real-time assistance in terms of monitoring and detection.

2.2.5 Intelligent Collision Avoidance and Manoeuvring System

Bram-Larbi et al. [82] proposed a prototype system that was able to perform intelligent collision avoidance and manoeuvring assistance to the driver. It had integrated AR and AI into a heads-up display to inform the driver of potential collisions and the recommended actions. Figure 2.33 shows a demonstration of their prototype system in a simulation.






Figure 2.33: Intelligent Collision Avoidance and Manoeuvring System [82]

The changing colours of the recommended actions indicate the increasing risk levels, specifically from blue, green, amber, to red. For example, a manoeuvring option with the colour red is riskier than the manoeuvring option with the colour blue which is less urgent and easier to perform. Additionally, their proposed AI Co-Driver acted as an information acquirer from the vehicle's sensors to support their prototype system. It provides real-time assistance in the forms of monitoring and detection that aided the driver in the decision-making process during imminent collision scenarios.

2.2.6 Summary of Existing Systems Review

This project proposed a system after the reviewing the strengths and weaknesses of existing systems. The proposed system considered features such as AR navigation, real-time assistance in the form of monitoring and detection, UX, map coverage, and real-time information. Table 2.5 shows the summary of the review on existing systems.

Table 2.5 Summary of Existing Systems Review

System	Google Maps	Waze	Sygie	MBUX Navigation System	Intelligent Collision Avoidance and Manoeuvring System	Proposed System
Feature						
AR Navigation	Walking only	✗	Poor	✓	✓	✓
Real-Time Assistance	✗	✗	Road sign	✗	Collision and manoeuvre	Lane change
UX	✗	✓	✗	✓	N/A	✓
Map Coverage	✓	✓	Poor, online maps	✓	N/A	✓
Real-Time Information	✓	Community-driven	✓	✓	N/A	✓

Chapter 3

System Methodology

This chapter outlines the methodology, system requirements, functional requirements, non-functional requirements, and project milestones.

3.1 Methodology

This project adopted the Extreme Programming (XP) agile software development methodology. The methodology focuses on quick and iterative development that has incremental changes for each system release rather than making big changes all at once. Its flexible nature allows for quick response to changing user requirements. Efficient coding practices and extensive testing were advocated without the need for proper documentation in this project.

Agile has 5 key principles which are customer involvement, incremental delivery, people not process, embrace change, and maintain simplicity. XP has 10 key practices which are incremental planning, small releases, simple design, test-first development, refactoring, pair programming, collective ownership, continuous integration, sustainable pace, and on-site customer. There were some limitations on following these principles and practices, but they served as the foundational pillars during the development of this project.

Figure 3.1 illustrates the process flow diagram of the agile XP methodology for every system release. It consists of 4 key phases which are planning, analysis, design, and implementation. The process repeats iteratively for the incremental changes of each system release.

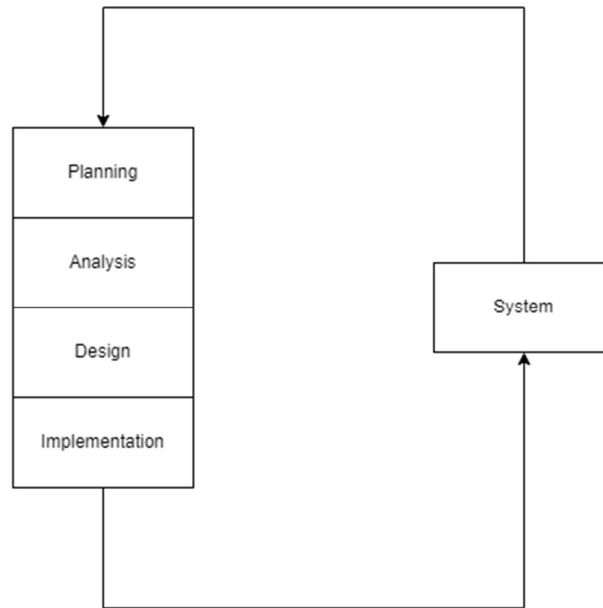


Figure 3.1: Agile XP Process Flow Diagram

Figure 3.2 illustrates the release cycle process of the agile XP methodology. User requirements are gathered in the form of use stories, then broken down into several tasks. Subsequently, the tasks are developed, integrated, and tested for the system release. At the end of each cycle, a system evaluation is performed to determine the user stories of the next release cycle.

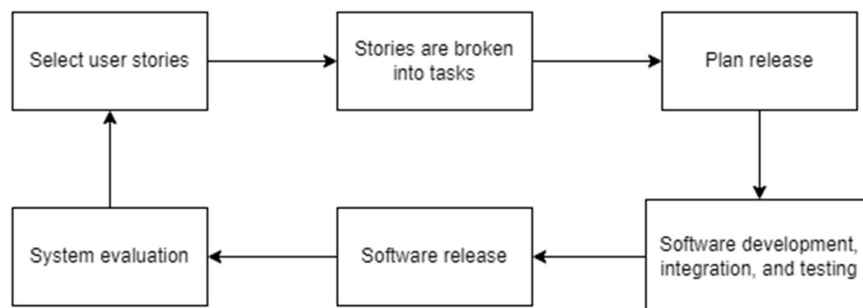


Figure 3.2: Agile XP Release Cycle Process

3.2 System Requirements

3.2.1 Hardware

This project required 2 hardware devices, namely a computer and a mobile device. The computer was used for the development of the system, while the mobile device was used for the testing of the system. Specifically, a Lenovo laptop and a Redmi smartphone were used for the development of this project. Table 3.1 and Table 3.2 show the hardware specifications of the laptop and smartphone respectively.

Table 3.1: Laptop Specifications

Specification	Description
Model	Lenovo IdeaPad Gaming 3 15ARH05 (82EY)
Processor	AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
Operating System	Windows 10 Home Single Language 64-bit
Graphics	NVIDIA GeForce GTX 1650 4GB GDDR6
Memory	16GB DDR4 RAM
Storage	512GB SATA SSD





Table 3.2: Smartphone Specifications





Specification	Description
Model	Xiaomi Redmi Note 12 Pro+
Processor	Octa-core (2x2.6 GHz Cortex-A78 & 6x2.0 GHz Cortex-A55)
Operating System	Android 12, upgradable to Android 14, HyperOS
Graphics	Mali-G68 MC4
Memory	12GB RAM
Storage	256GB
Rear Camera	Triple: 200 MP, f/1.7, 24mm (wide), 1/1.4", 0.56µm, PDAF, OIS 8 MP, f/2.2, 120° (ultrawide), 1/4.0", 1.12µm 2 MP, f/2.4, (macro) Features: Dual-LED dual-tone flash, HDR, panorama Video: 4K@30fps, 1080@30/60/120fps, 720p@960fps

3.2.2 Software

This project required the installation of several software tools on the laptop for the development of the proposed system. Table 3.3 shows a list of software tools, and their versions used in this project.

Table 3.3 Software Specifications

Software	Version	Description
Mobile Development		
Android Studio 	2024.2.2	<ul style="list-style-type: none"> Official IDE for Android development. Tools such as Android SDK, Android emulator, and Gradle.
Visual Studio Code 	1.98.2	<ul style="list-style-type: none"> Open-source text editor developed by Microsoft. Lightweight and fast. Extensive plug-in support from Extension Marketplace.
Flutter 	3.27.2	<ul style="list-style-type: none"> Open-source SDK developed by Google. Cross-platform development. Good performance.
AR Development		
Unity 	2022.3.55f1	<ul style="list-style-type: none"> Cross-platform development. Beginner-friendly.
AI Development		
CUDA	12.4	<ul style="list-style-type: none"> CUDA stands for Compute Unified Device Architecture. Parallel computing platform developed by NVIDIA. Required for utilising the NVIDIA GPU.
cuDNN	8.9.7	<ul style="list-style-type: none"> cuDNN stands for CUDA Deep Neural Network. GPU-accelerated library for deep neural networks. Developed by NVIDIA for running CUDA-enabled GPUs.

Python 	3.11.9	<ul style="list-style-type: none"> • High-level programming language. • High compatibility with most machine learning frameworks.
PyTorch 	2.6.0+cu124	<ul style="list-style-type: none"> • Open-source machine learning framework developed by Meta AI. • Tensor computation graphs. • Automatic differentiation.
Jupyter Notebook 	7.4.0	<ul style="list-style-type: none"> • Web-based interactive computing environment. • Widely used in data science and machine learning.
Windows Subsystem for Linux (WSL)	2.4.12.0	<ul style="list-style-type: none"> • WSL stands for Windows Subsystem for Linux. • Provides the compatibility layer for running Linux distributions on Windows without virtual machines.
TensorFlow 	2.19.0	<ul style="list-style-type: none"> • Open-source machine learning framework developed by Google. • High performance deployment such as TFLite.

3.3 Functional Requirements

This project listed 5 functional requirements to be implemented in the proposed system. They focus on defining the features of the system to achieve the objectives of this project. The functional requirements are as follows:

1. The user can tap on the map to select a destination.
2. The user can search for places to select a destination.
3. The user can preview the route to the selected destination.
4. The user can start a navigation session to the selected destination.
5. The user can switch between maps navigation mode and AR navigation mode.

First, the destination selection feature allows the user to select a destination by tapping on the map. When a destination is selected, information such as place name and coordinates are shown. The map shows the proximity area based on the current location of the user.

Second, the place search feature allows the user to type and search for specific places. It returns search results based on the query from the user. The user can choose to select the destination based on the search results.

Third, the route preview feature shows the routes to the selected destination. It shows information such as duration of travel, distance from destination, and ETA.

Fourth, a navigation session can be started by the user to receive directions from the system.

Fifth, the toggle switch is provided to allow the user to switch maps navigation mode or AR navigation mode.

3.4 Non-Functional Requirements

This project listed 4 non-functional requirements to ensure the quality of the proposed system. They focus on aspects such as fault tolerance, performance, and flexibility of the proposed system. The non-functional are as follows:

1. The system should check and inform any required service errors to the user.
2. The system should show places search results within five seconds.
3. The system should provide alternative routes to the selected destination.
4. The system should dynamically reroute when a wrong turn is taken.

First, the system checks and validates whether the required services such as Internet connectivity, terms and conditions, Google Maps API, location permission, and camera permission are available. If any of these services are not available, it informs the user of the failed services and prompts the user to try again. This achieves fault tolerance to avoid unexpected errors later on.

Second, the search results are returned and shown to the user in 5 seconds. This achieves performance as the user does not need to wait a long time for search results.

Third, the route preview shows and suggests multiple routes to the destination. By default, the best route is selected. However, it gives the user the option to take

alternative routes. This achieves flexibility for the user to pick and choose the other route if needed.

Fourth, dynamic rerouting is performed when a wrong turn is taken. This achieves fault tolerance as the user is able to receive new directions when a wrong turn is taken.

3.5 Project Milestones

3.5.1 Project I

Project I delivered a prototype system to demonstrate the feasibility of the proposed features. Figure 3.3 shows the Gantt chart of Project I.



Figure 3.3: Project I Gantt Chart

Project I spanned across a total of 6 weeks, specifically from October 2024 to December 2024. It implemented the features of destination selection, route preview, and navigation session. Additionally, it explored experimental real-time assistance of a pothole detection model. The timeline of Project I is as follows:

- Week 1 – Plan and sketch the prototype system.
- Week 2 – Develop the maps and navigation features.
- Week 3 – Prepare pothole dataset.
- Week 4 – Train and fine-tune pothole detection model.
- Week 5 – Test and export pothole detection model.
- Week 6 – Integrate all prototype features into the prototype system.

3.5.2 Project II

Project II delivered the proposed system that achieved the objectives of this project. It was developed and refined from the prototype system in Project I. A few changes were made to improve the proposed system. Figure 3.4 shows the Gantt chart of Project II.

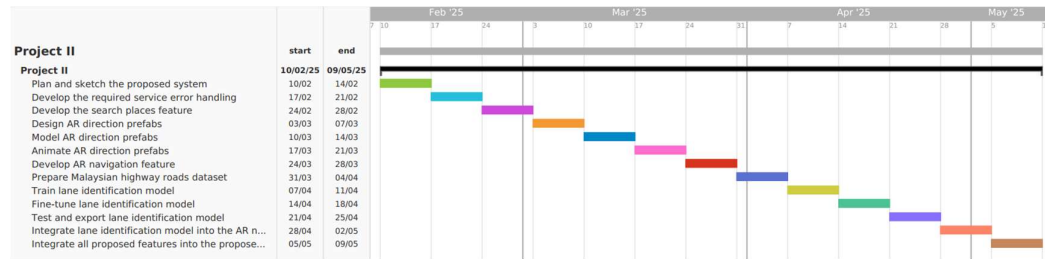


Figure 3.4: Project II Gantt Chart

Project II spanned across a total of 13 weeks, specifically from February 2025 to May 2025. It finished the remaining features such as place search and AR navigation. Additionally, a lane identification model was developed as real-time assistance to support the AR navigation more effectively and efficiently. The lane identification model was inspired and derived from the pothole detection model in Project I. The timeline of Project II is as follows:

- Week 1 – Plan and sketch the proposed system.
- Week 2 – Develop the required service error handling.
- Week 3 – Develop the search places feature.
- Week 4 – Design AR direction prefabs.
- Week 5 – Model AR direction prefabs.
- Week 6 – Animate AR direction prefabs.
- Week 7 – Develop AR navigation feature.
- Week 8 – Prepare Malaysian highway roads dataset.
- Week 9 – Train lane identification model.
- Week 10 – Fine-tune lane identification model.
- Week 11 – Test and export lane identification model.
- Week 12 – Integrate lane identification model into the AR navigation feature.
- Week 13 – Integrate all proposed features into the proposed system.

Chapter 4

System Design

This chapter illustrates the system architecture, system flowcharts, storyboard, use case diagram, activity diagram, low-fidelity prototype, and high-fidelity prototype.

4.1 System Architecture

Figure 4.1 illustrates the system architecture of this project. It has five main components which utilises technologies such as Flutter, Google Maps, Unity, PyTorch, and TensorFlow.

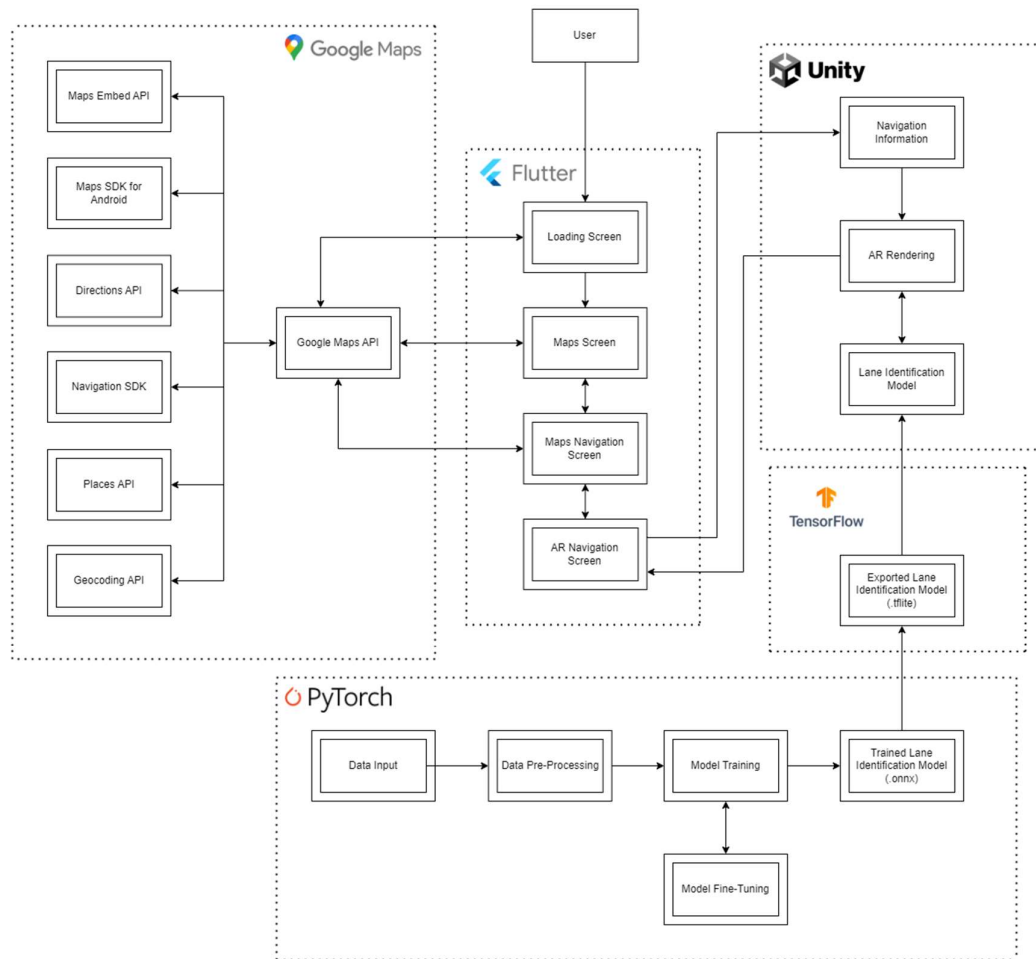


Figure 4.1: System Architecture

Flutter manages the mobile application UI, while Google Maps provides the information on maps, places, routes, and navigation. Unity handles rendering AR components such as AR directions. PyTorch develops the lane identification model, while TensorFlow exports the model to the TFLite format for mobile device runtime.

First, Flutter manages the UI of the mobile application that serves as the interface bridge between the user and the system. Other than that, it also serves as the main communication bridge between the components. It has the subcomponents of “Loading Screen”, “Maps Screen”, “Maps Navigation Screen”, and “AR Navigation Screen”. “Loading Screen” checks the availability of the required services which are Internet connectivity, terms and conditions, Google Maps API, location permission, and camera permission. If any of these services fail, it redirects to the failed loading screen that informs about the failed services and prompts to retry. “Maps Screen” loads a map from the Maps SDK for Android. It also handles place search, destination selection, route preview, and navigation sessions. “Maps Navigation Screen” shows directions and guidance on the map while “AR Navigation Screen” shows directions and guidance in the form of AR directions.

Second, Google Maps provides information on maps, places, routes, and navigation through the Google Maps API. It communicates with services such as Maps Embed API, Maps SDK for Android, Directions API, Navigation SDK, Places API, and Geocoding API. Maps Embed API is used by the system to check the availability of the Google Maps API. Maps SDK for Android provides maps that can be loaded on Android devices. Directions API retrieves routes to the selected destination. Navigation SDK enables the starting of a navigation session on the map. Places API and Geocoding API are used to search for places and translate place names or addresses to latitude and longitude respectively.

Third, Unity handles rendering of AR components. It has the subcomponents of “Navigation Information”, “AR Rendering”, and “Lane Identification Model”. “Navigation Information” receives navigation information about directions and guidance in terms of maneuver, distance, and target lane. “AR Rendering” renders the AR direction prefabs for the “AR Navigation Screen” as the corresponding AR directions and guidance. “Lane Identification Model” runs the inference in real-time

through the AR capture camera as monitoring and detection whether to render AR direction prefabs that instructs the user to change towards the left or right lane.

Fourth, PyTorch develops the lane identification model from a custom Malaysian highway road dataset annotated from [83]. It has the subcomponents of “Data Input”, “Data Pre-Processing”, “Model Training”, “Model Fine-Tuning”, “Trained Lane Identification Model (.onnx)”. “Data Input” imports and “Data Pre-Processing” pre-processes the dataset for “Model Training” to train the lane identification model. “Model Fine-Tuning” tests and fine-tunes the model hyperparameters, then “Trained Lane Identification Model (.onnx)” exports the model to the Open Neural Network Exchange (ONNX) file format.

Lastly, TensorFlow exports the ONNX model to the TFLite model. This prepares a lane identification model in the TFLite file format to be embedded in Unity for mobile device inferences.

4.2 System Flowcharts

4.2.1 System Flowchart

Figure 4.2 illustrates the overall flowchart of the system. The flow begins with the Maps Module that handles the map, place search, destination selection, and navigation session initiation. If navigation session is not started, it will return to the Maps Module. If navigation session is started, it will go to the Navigation Module that handles the maps navigation and AR navigation. If the navigation session is cancelled, it will return to the Maps Module. If the navigation session is not cancelled, it will check if the destination is reached. If the destination is not reached, it returns to the Navigation Module. If the destination is reached, the navigation session is completed and terminates.

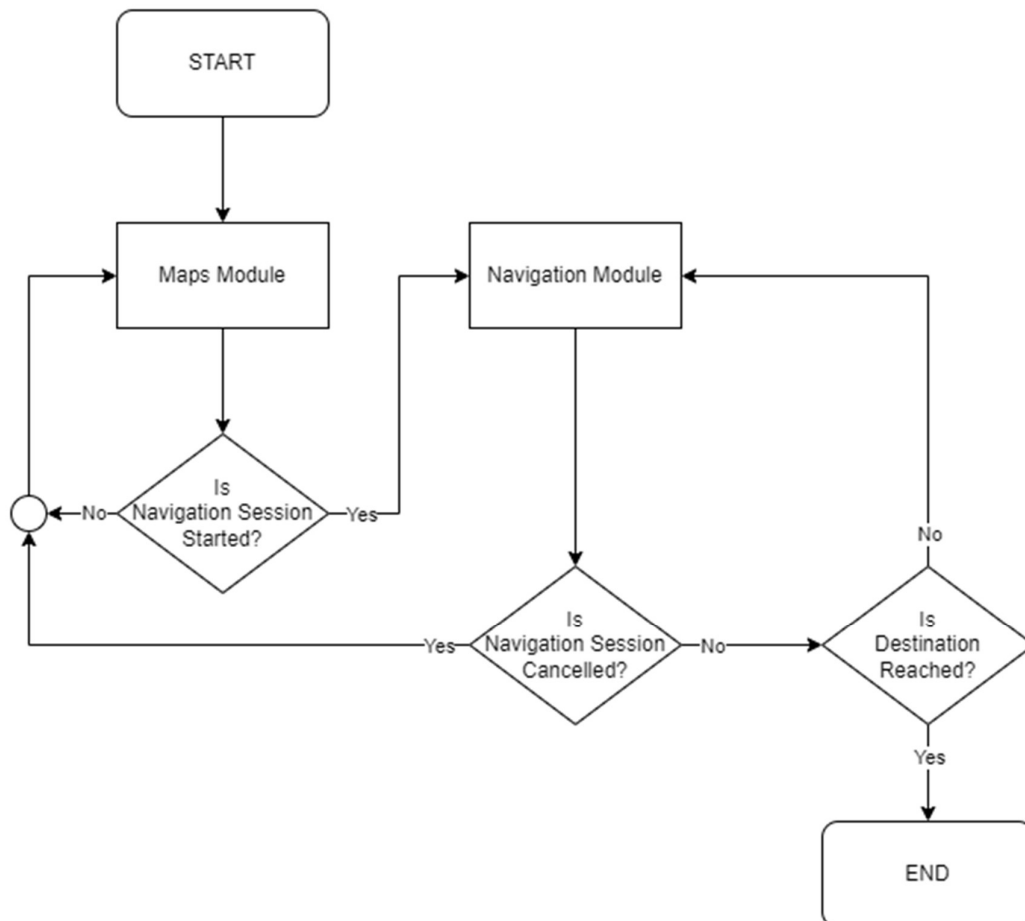


Figure 4.2: System Flowchart

4.2.2 Maps Module Flowchart

Figure 4.3 illustrates the detailed flowchart of the Maps Module. The Maps Module handles the required services error checking, place search, destination selection, route preview, and navigation session initiation.

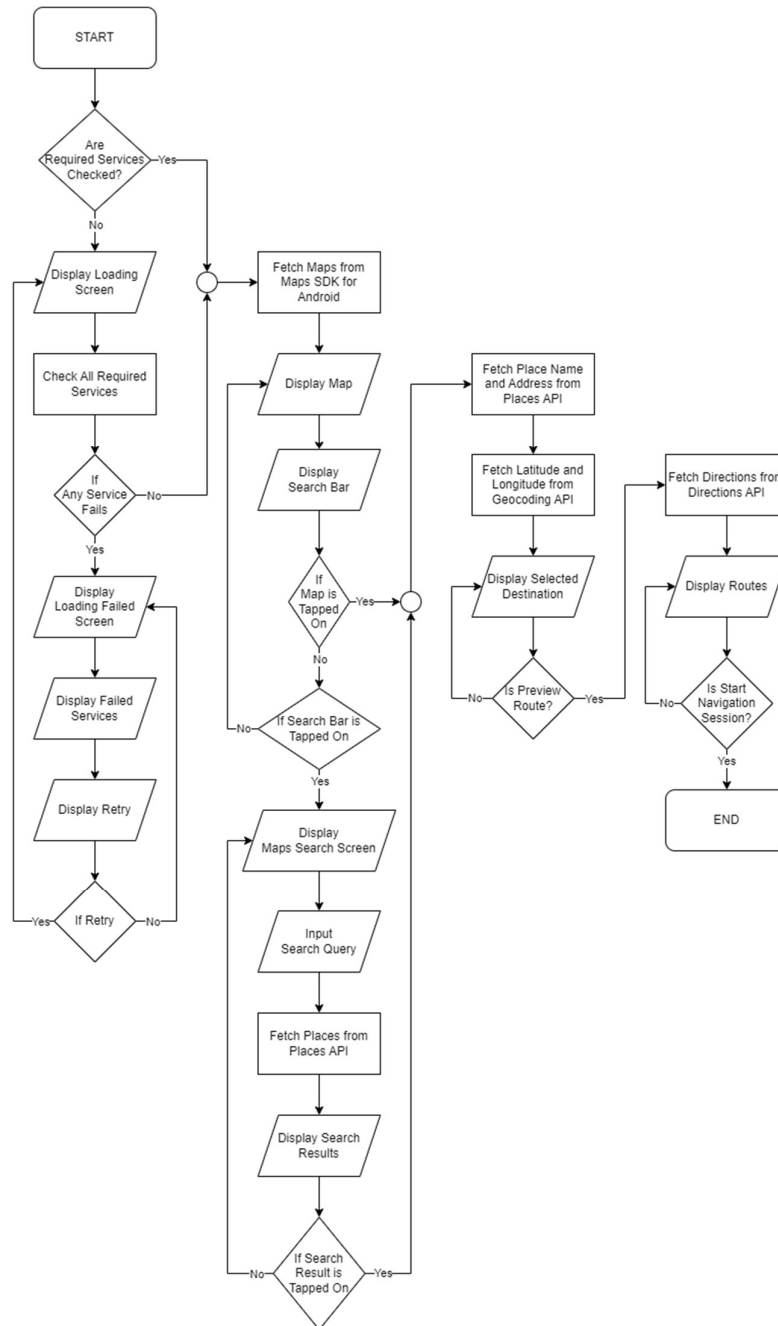


Figure 4.3: Maps Module Flowchart

4.2.3 Navigation Module Flowchart

Figure 4.4 illustrates the detailed flowchart of the Navigation Module. The Navigation Module handles the navigation session initiated from the Maps Module. It includes the maps navigation UI and AR navigation UI. Additionally, it utilises the real-time assistance of the lane identification model from the Intelligence Module for context-aware and intelligent AR rendering.

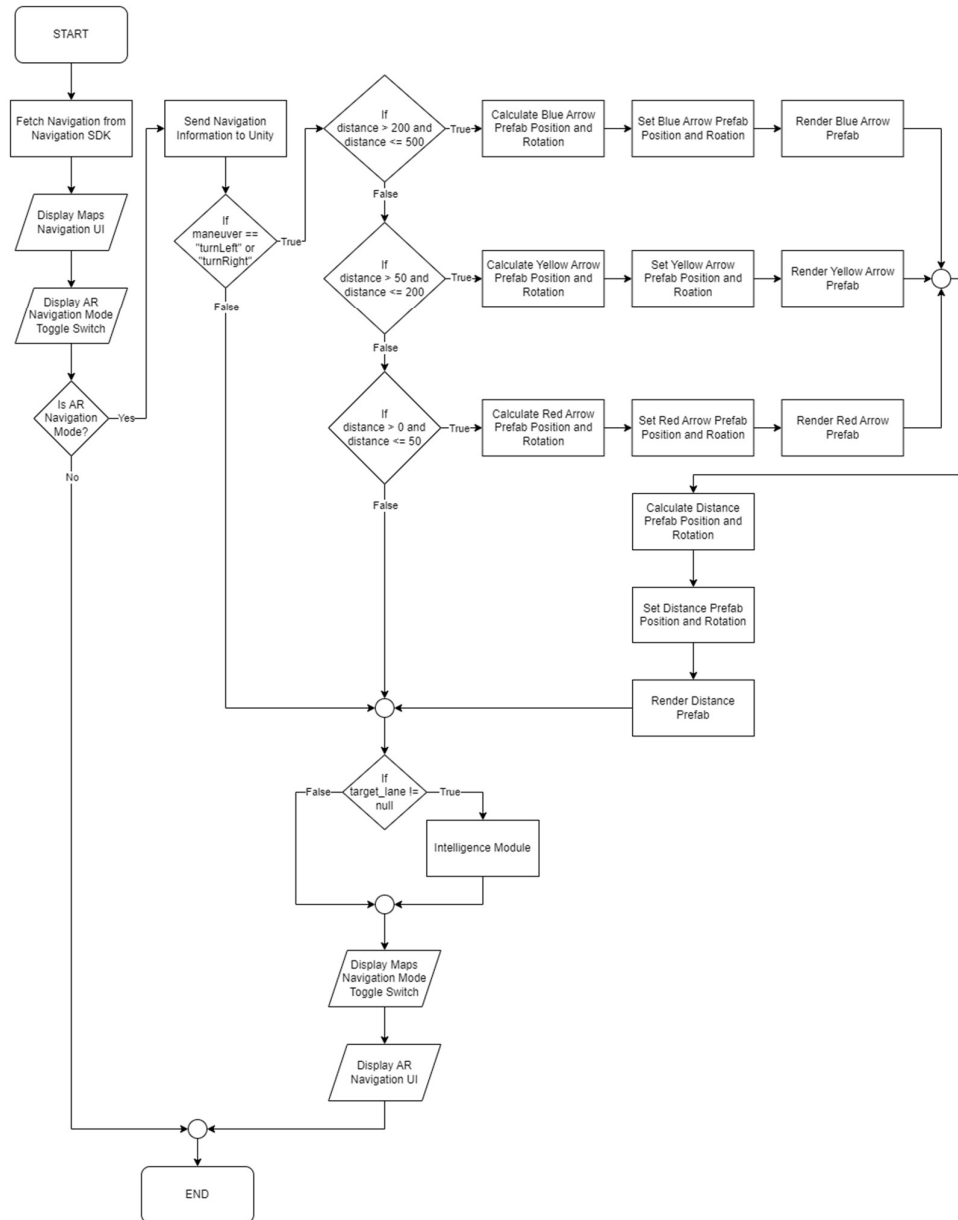


Figure 4.4: Navigation Module Flowchart

4.2.4 Intelligence Module Flowchart

Figure 4.5 illustrates the detailed flowchart of the Intelligence Module. The Intelligence Module loads and initialises the lane identification model, then performs inference on the model with the pre-processed captured AR camera frame to obtain the current lane number and confidence level from the prediction. Then, if the confidence level is equal to or greater than 50% and the current lane number is not equal to the target lane number, it renders the prefab as AR directions to indicate changing to the left or right lanes after performing position and rotation calculations. If there is not enough confidence level or the current lane number is already equivalent to the target lane number, it skips rendering the lane change prefab.

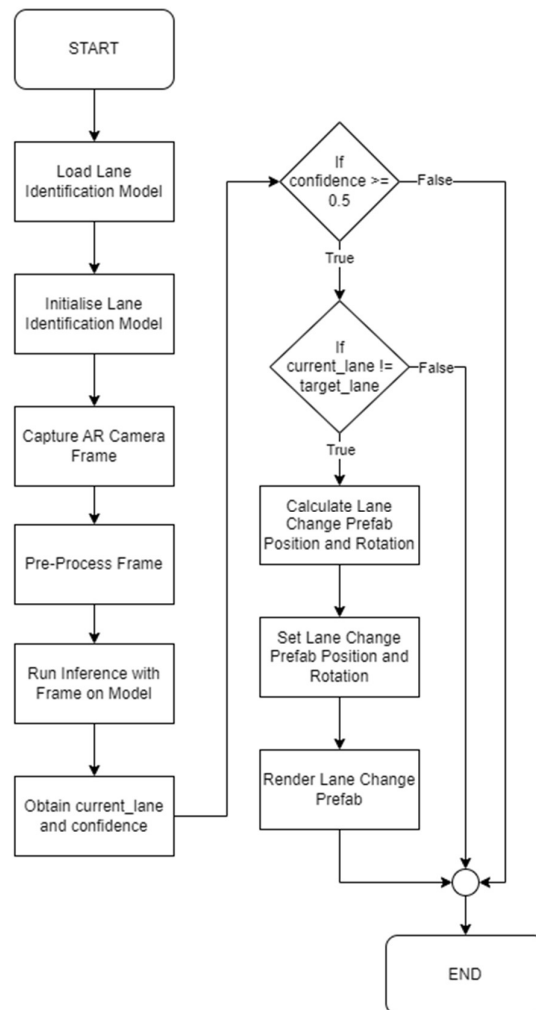


Figure 4.5: Intelligence Module Flowchart

4.2.5 Lane Identification Model Flowchart

Figure 4.6 illustrates the detailed flowchart of developing the lane identification model for the Navigation Module. A video is obtained to be cleaned and processed as training, evaluation, and test sets. Then, data augmentation and transformation are performed to be loaded for model training. An EfficientNet-B0 architecture is used with its pre-trained weights. Training hyperparameters are set to train, evaluate, and test the model. After each training session, the model is saved. After a few hyper-tunings, the model is exported to the ONNX format, then to the TFLite format for the lane identification model to run inferences on mobile devices.

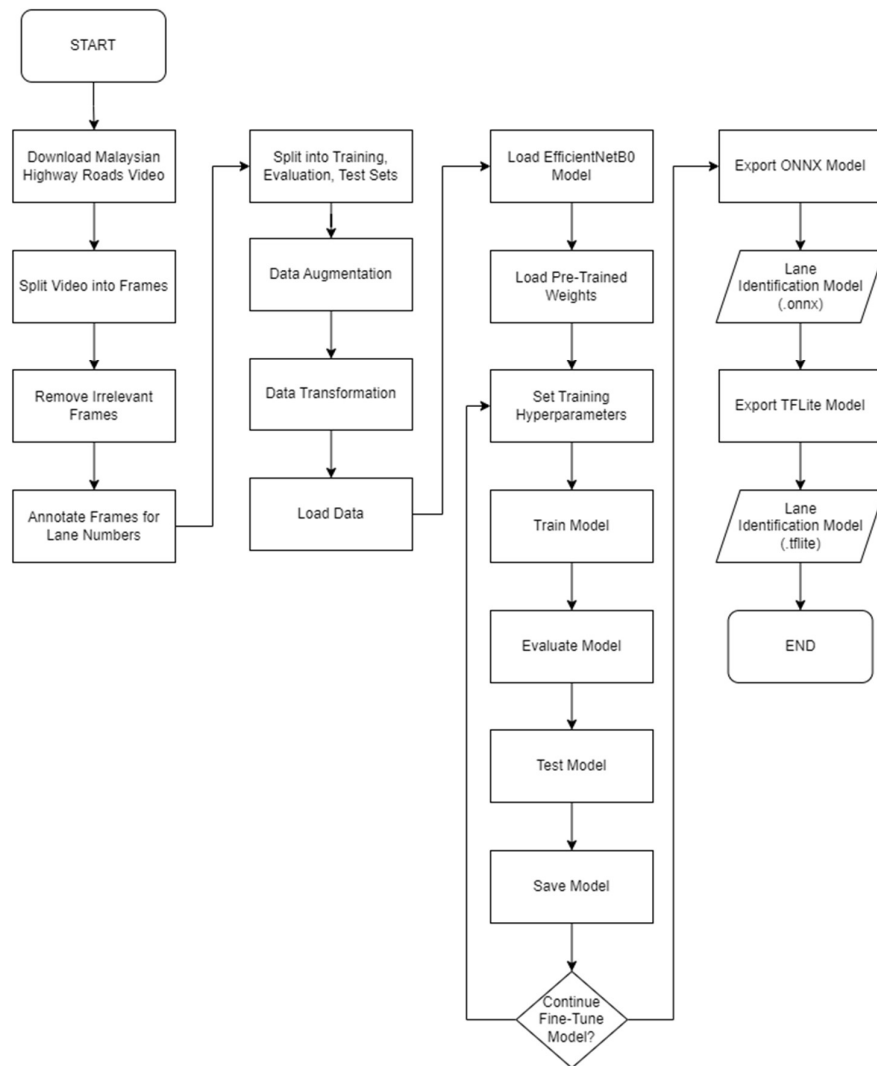


Figure 4.6: Lane Identification Model Flowchart

4.2.6 Services Flowchart

Figure 4.7 illustrates the detailed flowchart of the “Check All Required Services” from Figure 4.3. It checks the availability of services such Internet connectivity, terms and conditions, Google Maps API, location access, and camera access.

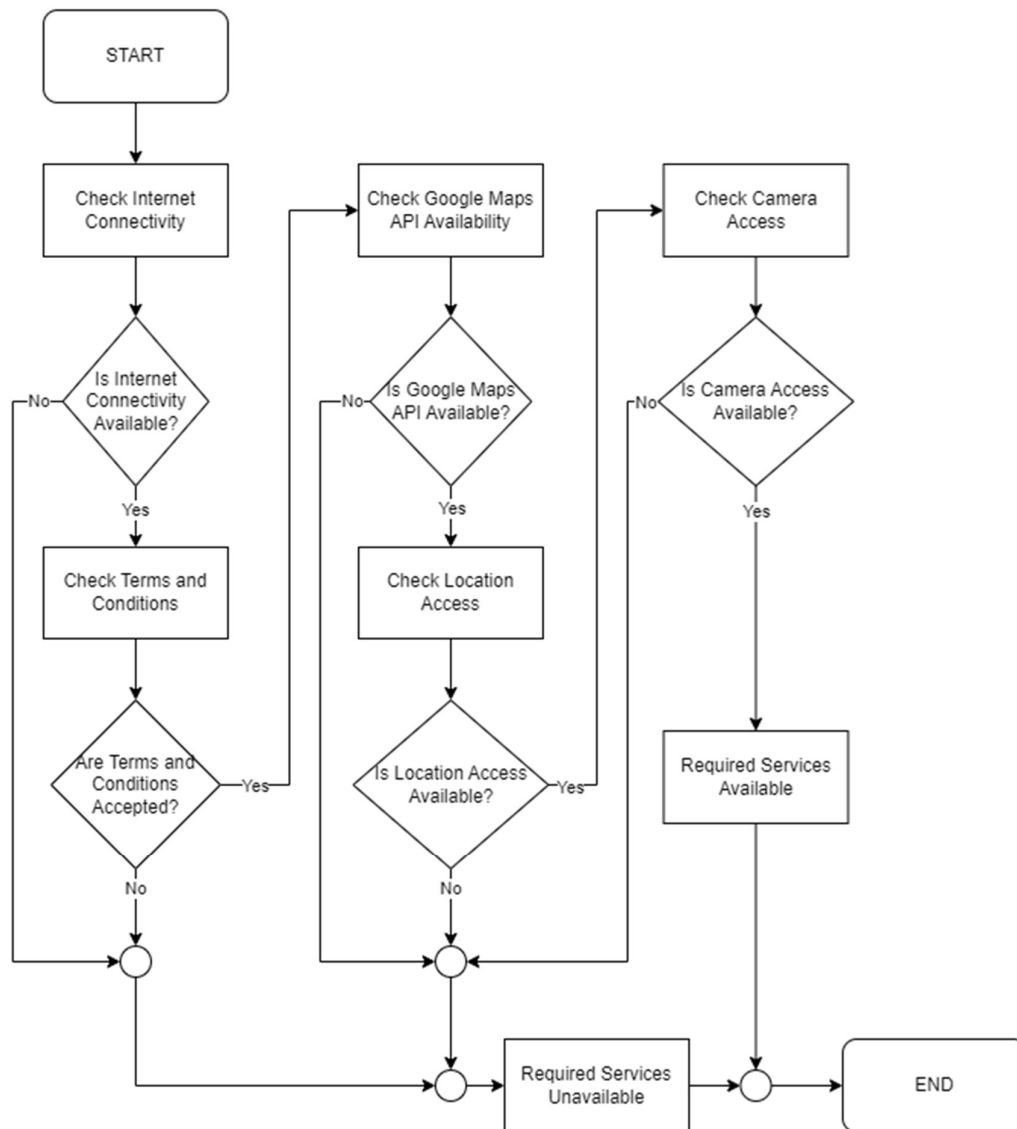


Figure 4.7: Services Flowchart

4.3 Storyboard

Figure 4.8 illustrates a storyboard that describes a scenario for the proposed system. Adam drives to Kuala Lumpur for a holiday trip. He uses a mobile navigation application to receive directions and guidance to the destination. On an upcoming junction, it instructs Adam to make a right turn in 200 metres. However, there are two closely spaced right turns ahead. This causes ambiguous directions given by the application, leading to confusion and uncertainty for Adam. He takes a bold guess, and this ends up prolonging his journey due to taking the wrong turn. This proposed system focuses on this problem to provide clearer directions, reducing confusion and uncertainty during navigation sessions.



Figure 4.8: A Trip to Kuala Lumpur

4.4 Use Case Diagram

Figure 4.9 illustrates the use case diagram that includes actors and use cases, along with their associations in the system. The actors are “Driver” and “Google Maps API”. The use cases are search place, select destination, preview route, start maps navigation, and start AR navigation. Each of the use cases only has the “extend” relationship associating with other use cases as shown in Figure 4.9.

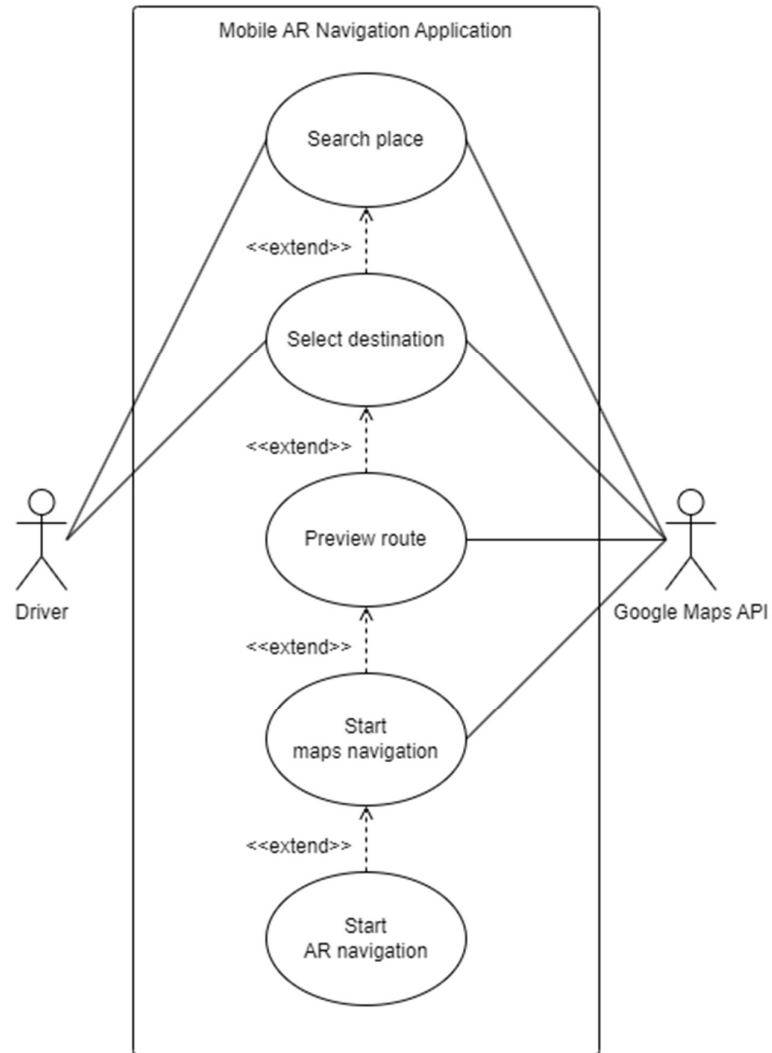


Figure 4.9: Use Case Diagram

4.5 Activity Diagram

Figure 4.10 illustrates the activity diagram of the system. It includes the workflow of the corresponding activities between the “Driver”, “Mobile AR Navigation Application”, and Google Maps API.

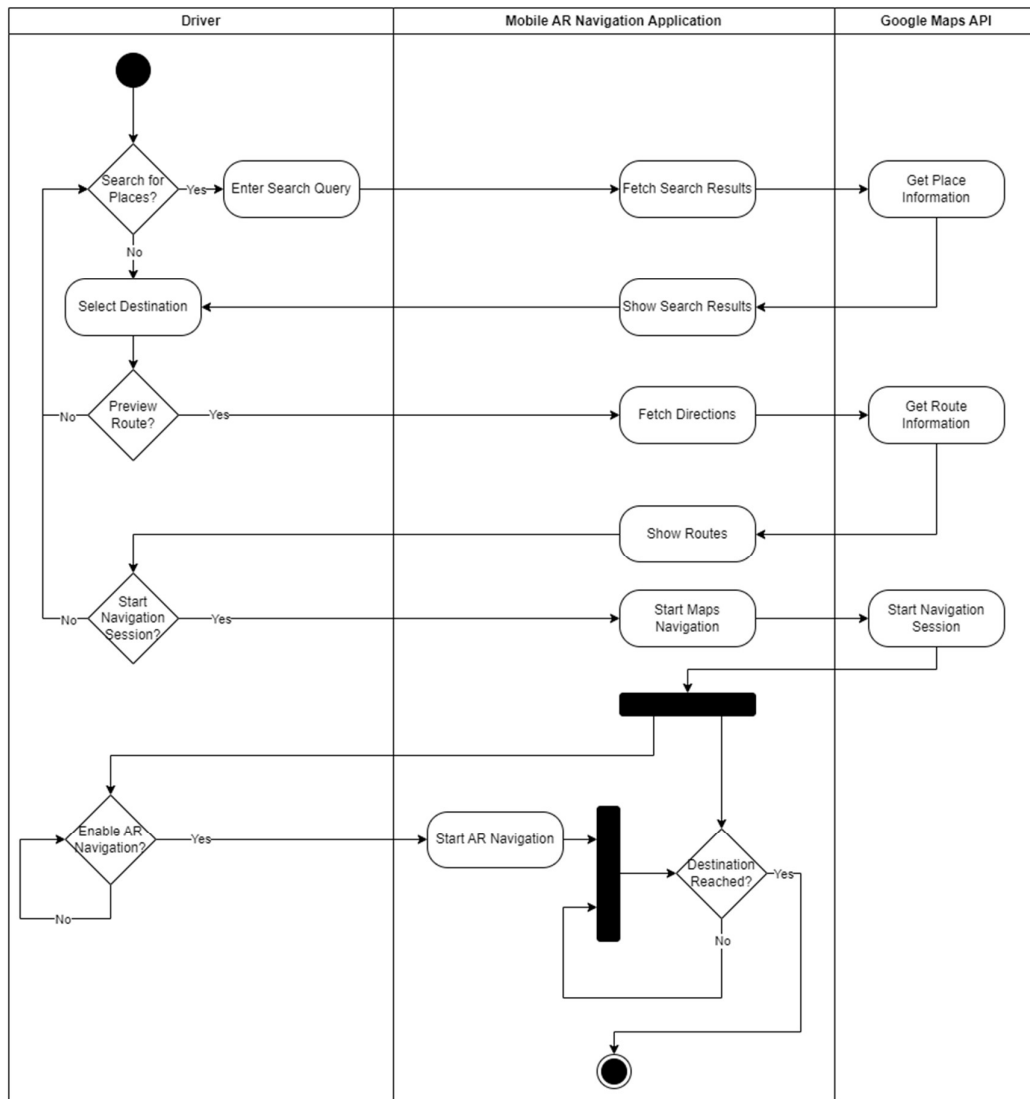


Figure 4.10: Activity Diagram

4.6 Wireframe Design

Figure 4.11 illustrates the wireframe design of the mobile application UI for the system. It highlights the arrangement of content on a screen layout. A detailed wireframe design of each screen can be found in Appendix A.



Figure 4.11: Wireframe Design

4.7 Low-Fidelity Prototype

Figure 4.12 illustrates the low-fidelity prototype of the mobile application UI for the system. It highlights the sketch or mock-up to represent the screen layout. A detailed low-fidelity prototype of each screen can be found in Appendix B.

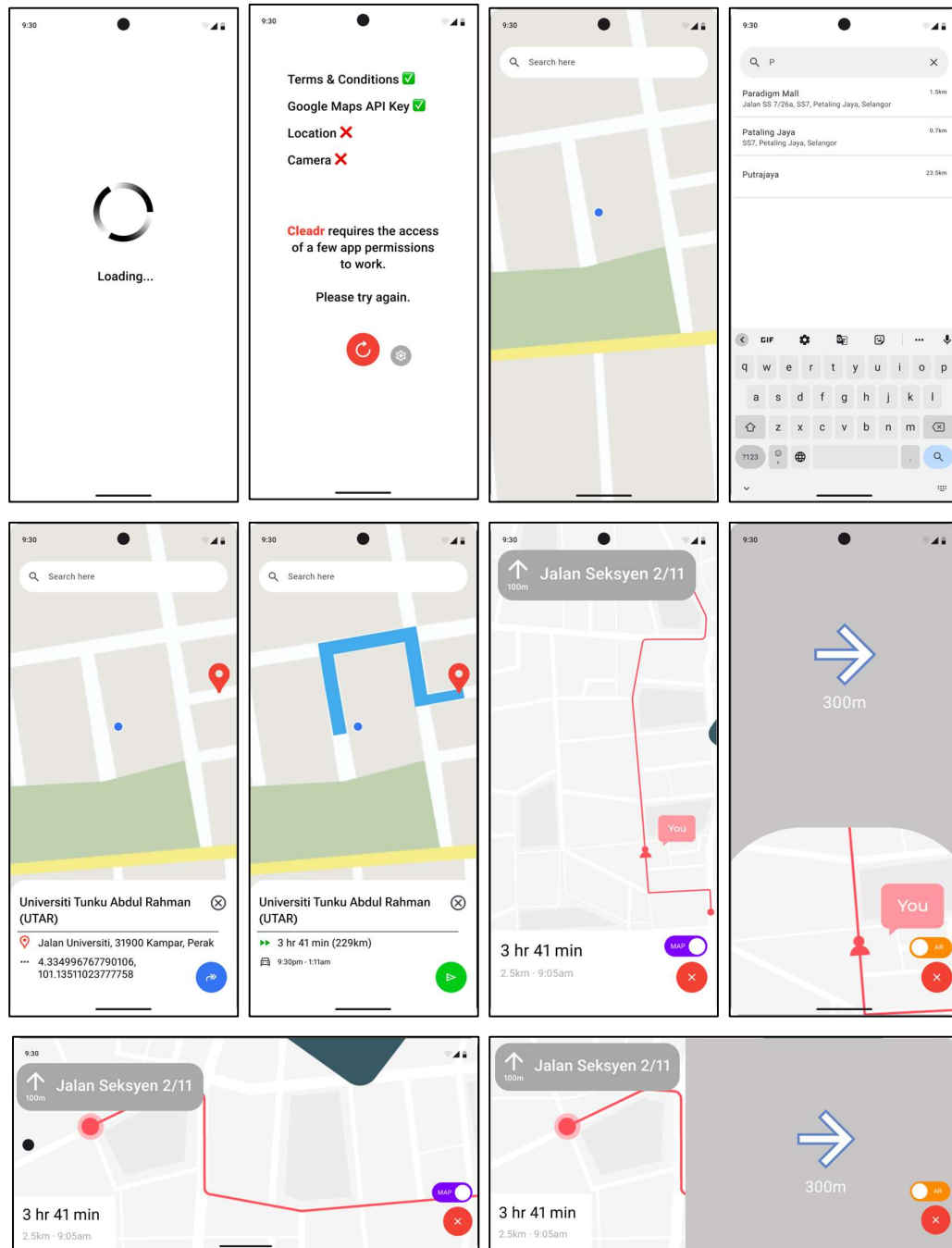


Figure 4.12: Low-Fidelity Prototype

4.8 High-Fidelity Prototype

Figure 4.13 illustrates the high-fidelity prototype of the mobile application UI for the system. It resembles the final product of the system for this project. A detailed high-fidelity prototype of each screen can be found in Appendix C.

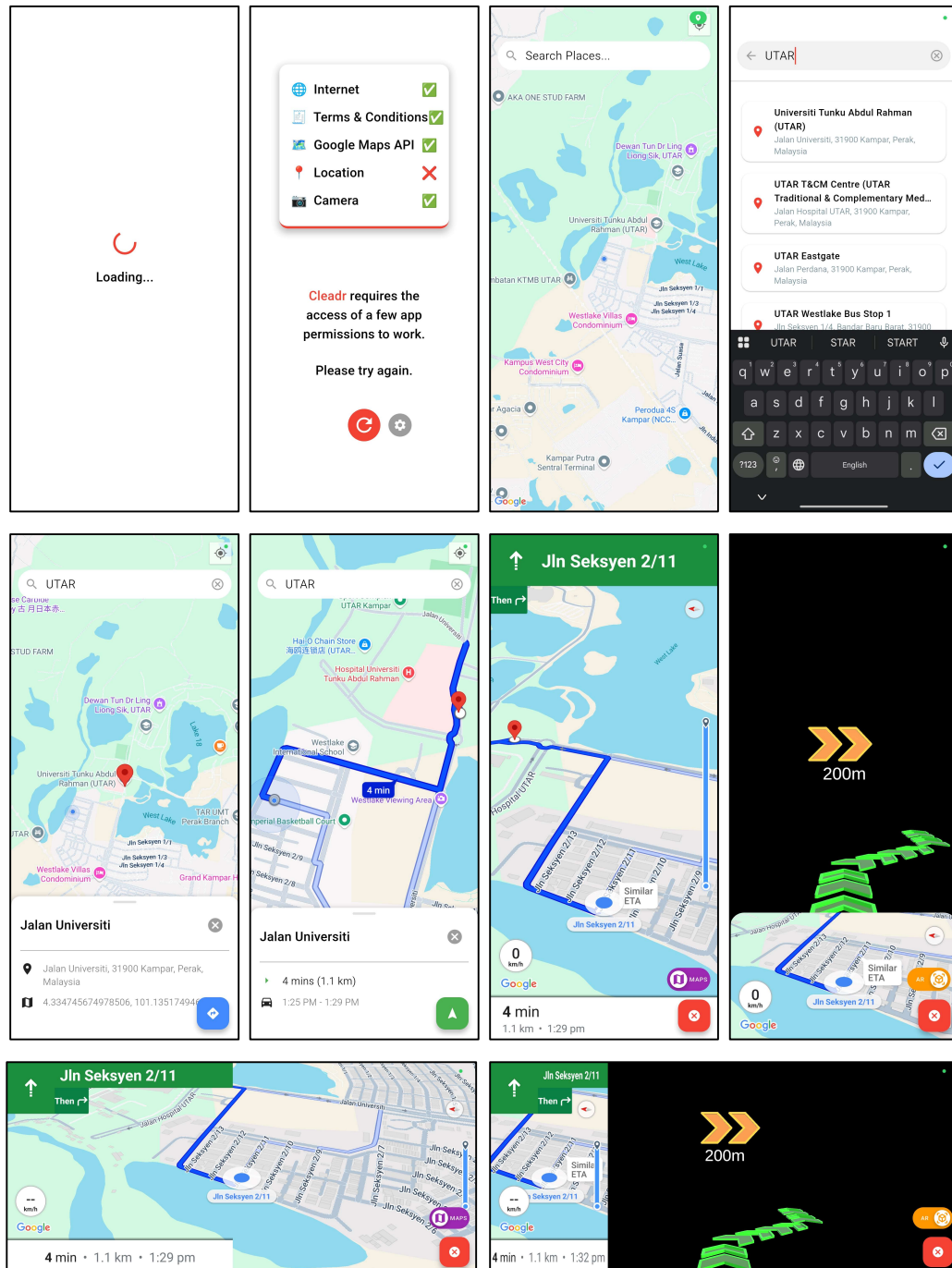


Figure 4.13: High-Fidelity Prototype

Chapter 5

System Implementation

This chapter demonstrates the software installation and setup, settings and configurations, system operation, implementation issues and challenges, and concluding remarks. Appendix D shows the GitHub repository for the system implementation.

5.1 Software Installation and Setup

This project installed and set up several software tools such as Android Studio, Visual Studio Code, Flutter, Unity, CUDA, cuDNN, Python, Jupyter Notebook, PyTorch, and WSL to develop the proposed system.

5.1.1 Android Studio

Android Studio was installed with the default settings through the official installer from [84]. NDK version 27.0.12.077973 was installed through the SDK Manager by Android Studio. Figure 5.1 shows the installed NDK version.

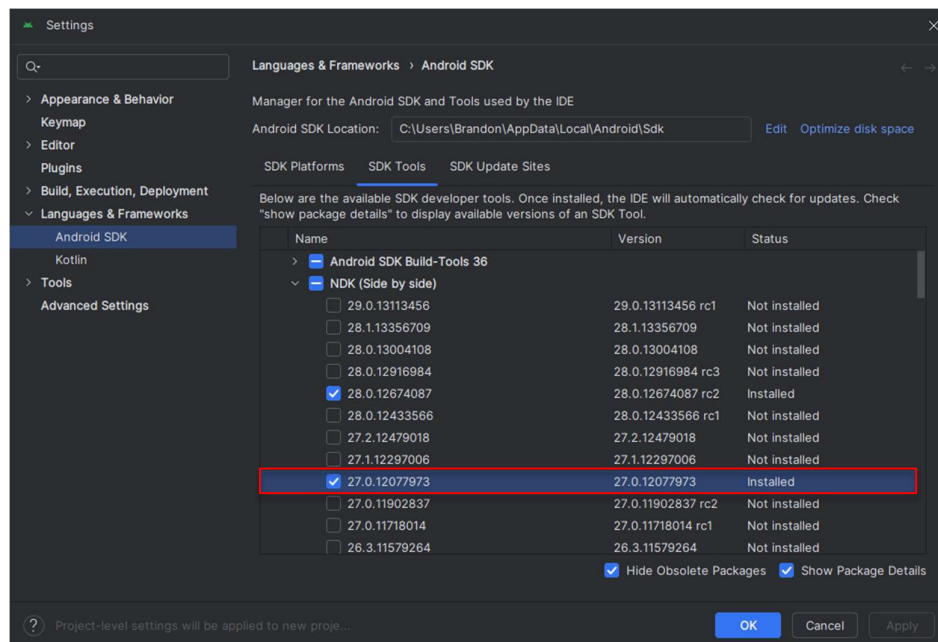


Figure 5.1: NDK 27.0.12.077973

5.1.2 Visual Studio Code

Visual Studio Code was installed with the default settings through the official installer from [85]. The Flutter extension was installed through the Extension Marketplace of Visual Studio Code. Figure 5.2 shows the installed Flutter extension.

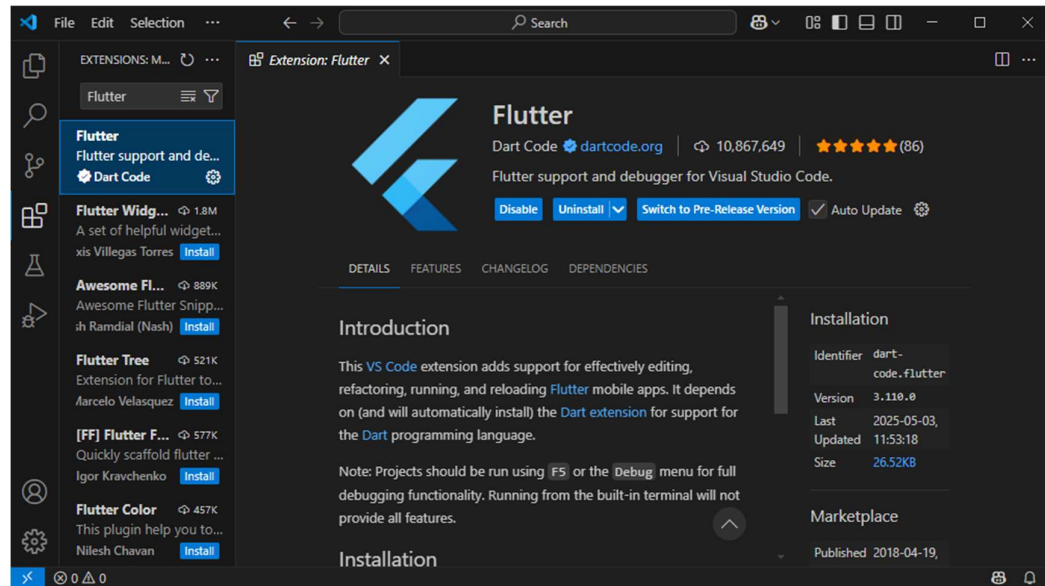


Figure 5.2: Flutter Extension

5.1.3 Flutter

Flutter was installed through the Flutter extension of Visual Studio Code. The installation steps were referred to [86]. The installation steps are as follows:

1. Open Command Palette in Visual Studio Code.
2. Type and select “Flutter: New Project”.
3. Download SDK and Clone Flutter.
4. Add SDK to PATH.
5. Open a new terminal and run “doctor --android-licenses”.
6. Accept the licenses.
7. Run “flutter doctor” on the terminal to check whether Flutter is successfully installed.

Figure 5.3 shows the installed Flutter. Note that the Visual Studio status may be allowed to fail as it is only needed for developing Windows applications.



```

C:\Users\Brandon>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.2, on Microsoft Windows [Version 10.0.19045.5737], locale en-MY)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.12.3)
[✓] Android Studio (version 2024.2)
[✓] VS Code (version 1.99.3)
[✓] Connected device (3 available)
[✓] Network resources

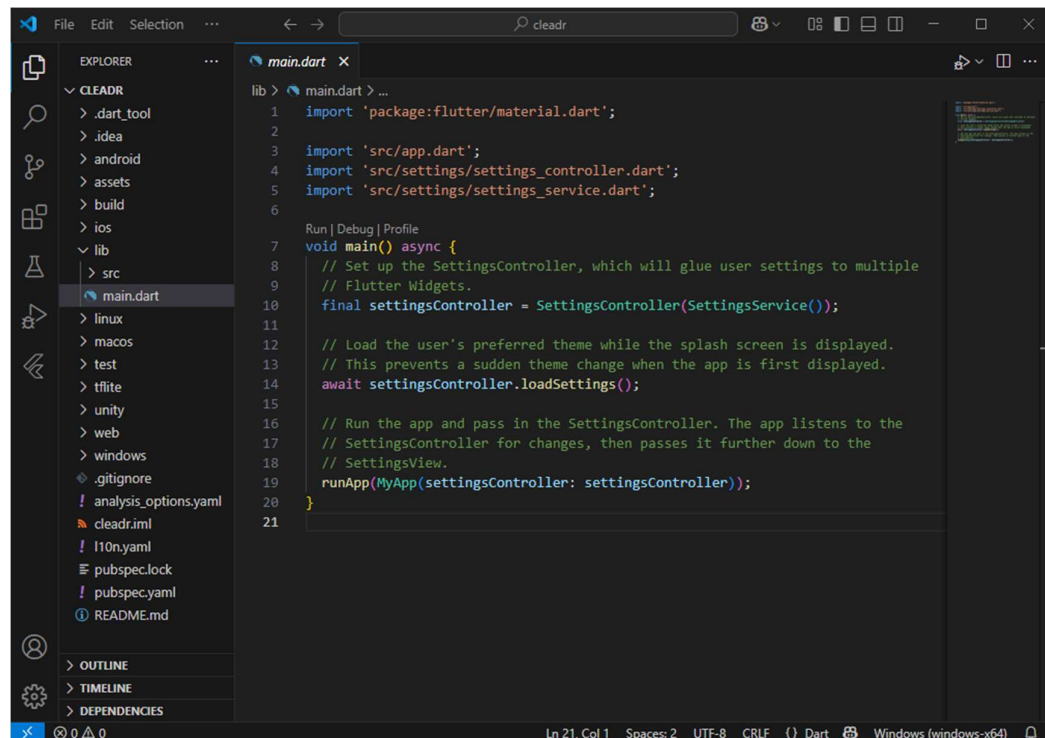
• No issues found!

C:\Users\Brandon>

```

Figure 5.3: Flutter

A new Flutter project was created. The folders named “unity” and “tflite” were manually created for future use. Figure 5.4 shows the directory and overview of the Flutter project.



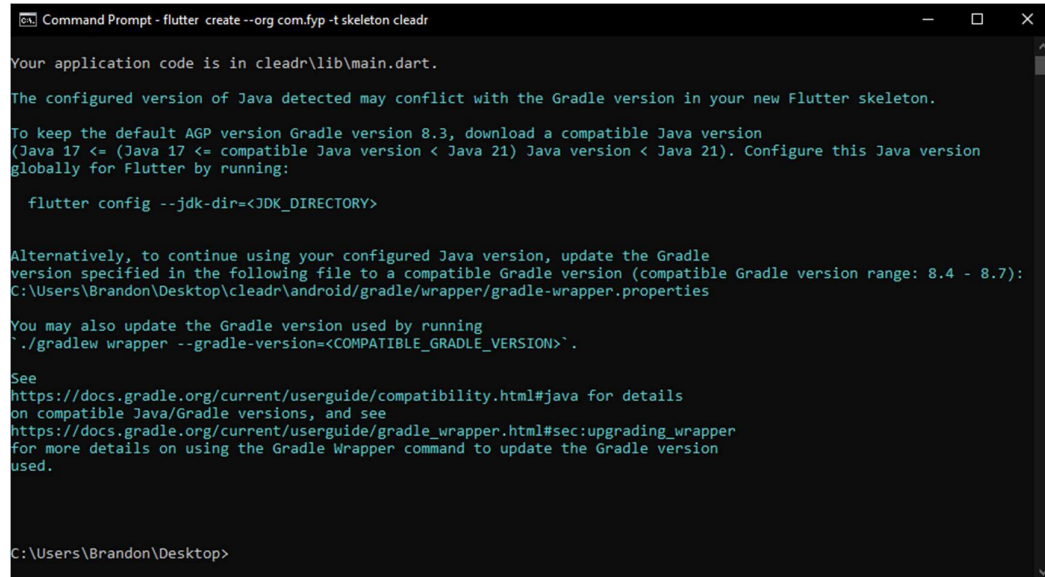
```

lib > main.dart > ...
1  import 'package:flutter/material.dart';
2
3  import 'src/app.dart';
4  import 'src/settings/settings_controller.dart';
5  import 'src/settings/settings_service.dart';
6
7  Run | Debug | Profile
8  void main() async {
9    // Set up the SettingsController, which will glue user settings to multiple
10   // Flutter Widgets.
11   final settingsController = SettingsController(SettingsService());
12
13   // Load the user's preferred theme while the splash screen is displayed.
14   // This prevents a sudden theme change when the app is first displayed.
15   await settingsController.loadSettings();
16
17   // Run the app and pass in the SettingsController. The app listens to the
18   // SettingsController for changes, then passes it further down to the
19   // SettingsView.
20   runApp(MyApp(settingsController: settingsController));
21 }

```

Figure 5.4: Flutter Project

The versions of Java and Gradle in this project were made sure to be compatible with each other. Figure 5.5 shows the warning message regarding the version conflict between Java and Gradle. This conflict can be solved by either updating Java or Gradle.



```

Command Prompt - flutter create --org com.fyp -t skeleton cleadr

Your application code is in cleadr\lib\main.dart.

The configured version of Java detected may conflict with the Gradle version in your new Flutter skeleton.

To keep the default AGP version Gradle version 8.3, download a compatible Java version
(Java 17 <= (Java 17 <= compatible Java version < Java 21) Java version < Java 21). Configure this Java version
globally for Flutter by running:

  flutter config --jdk-dir=<JDK_DIRECTORY>

Alternatively, to continue using your configured Java version, update the Gradle
version specified in the following file to a compatible Gradle version (compatible Gradle version range: 8.4 - 8.7):
C:\Users\Brandon\Desktop\cleadr\android\gradle\wrapper\gradle-wrapper.properties

You may also update the Gradle version used by running
'./gradlew wrapper --gradle-version=<COMPATIBLE_GRADLE_VERSION>'.

See
https://docs.gradle.org/current/userguide/compatibility.html#java for details
on compatible Java/Gradle versions, and see
https://docs.gradle.org/current/userguide/gradle_wrapper.html#sec:upgrading_wrapper
for more details on using the Gradle Wrapper command to update the Gradle version
used.

C:\Users\Brandon\Desktop>

```

Figure 5.5: Java and Gradle Conflict Warning Message

5.1.4 Unity

Unity was installed with the default settings through the official installer from [87]. Unity version 2022.3.55f1 was installed through Unity Hub. Figure 5.6 shows the installed Unity version.

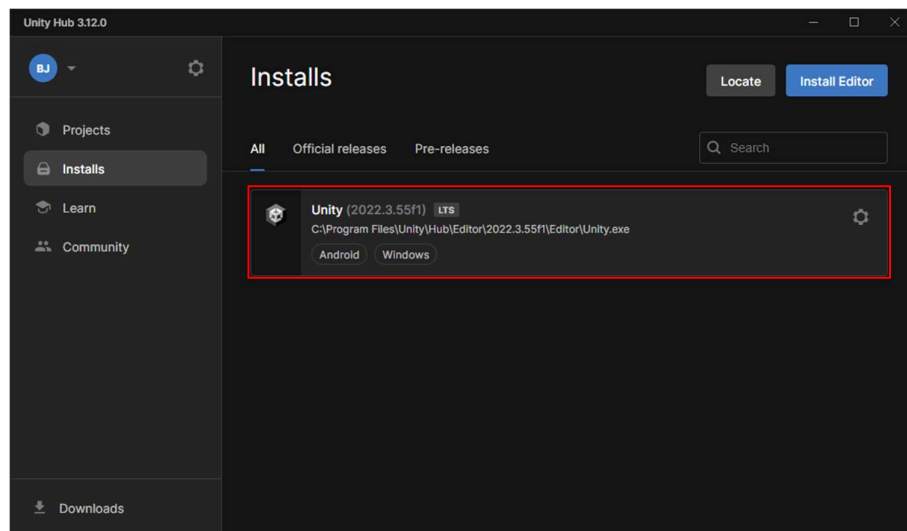


Figure 5.6: Unity 2022.3.55f1

A new Unity project was created in the “unity” folder of the Flutter project. The AR Mobile template module was attached to the Unity project. Figure 5.7 shows the overview of the Unity project.

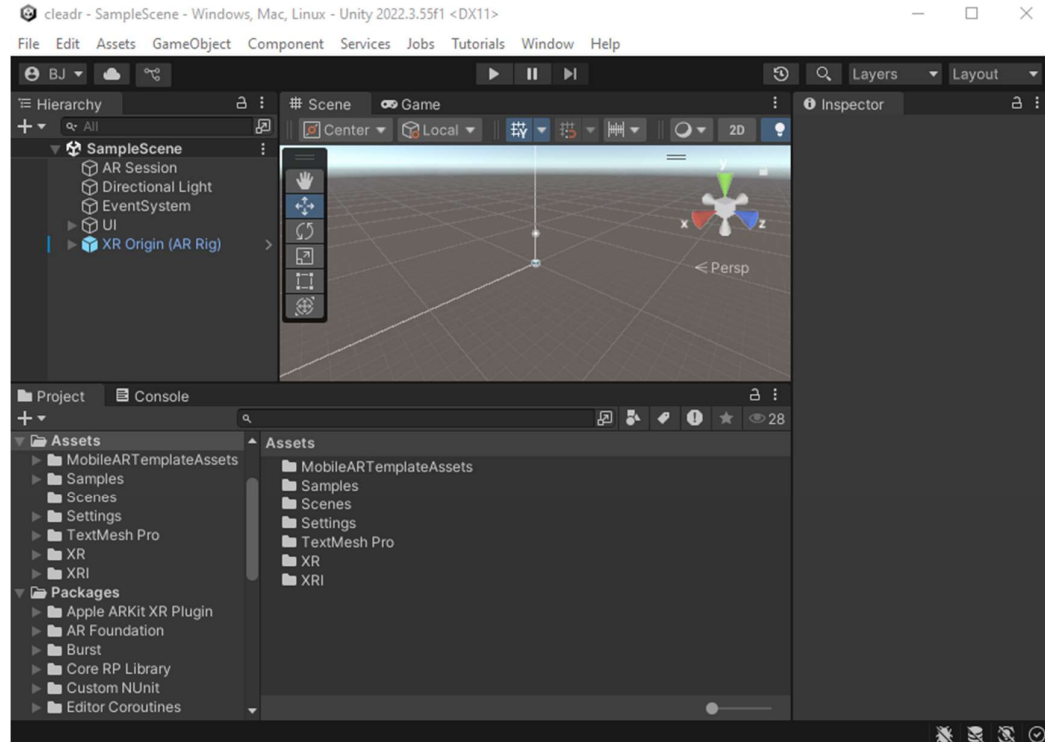


Figure 5.7: Unity Project

5.1.5 CUDA

CUDA was installed with the default settings through the official installer from [88]. Figure 5.8 shows the installed CUDA.

```

C:\Users\Brandon>nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Tue_Feb_27_16:28:36_Pacific_Standard_Time_2024
Cuda compilation tools, release 12.4, V12.4.99
Build cuda_12.4.r12.4/compiler.33961263_0

C:\Users\Brandon>

```

Figure 5.8: CUDA

5.1.6 cuDNN

cuDNN was installed on CUDA with the official library from [89]. The library files were included in the “bin”, “include”, and “lib” folders of the CUDA directory respectively. Figure 5.9 shows the directory of CUDA.

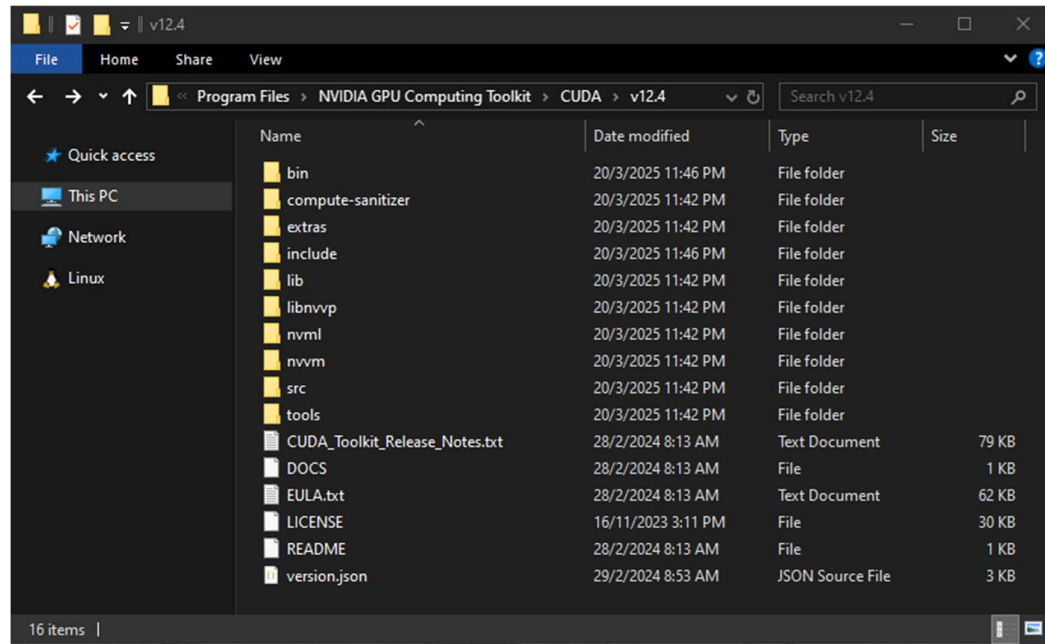


Figure 5.9: CUDA Directory

5.1.7 Python

Python was installed with the default settings through the official installer from [90]. Figure 5.10 shows the installed Python.

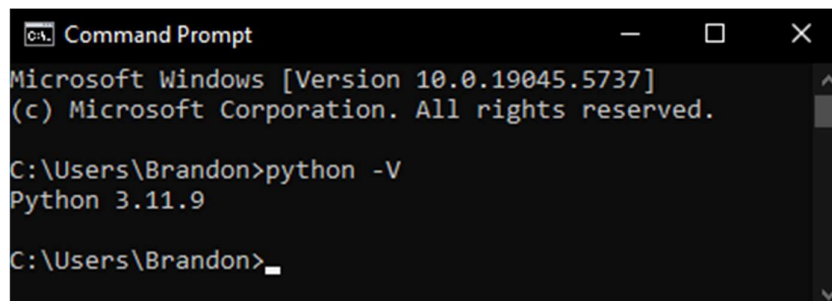
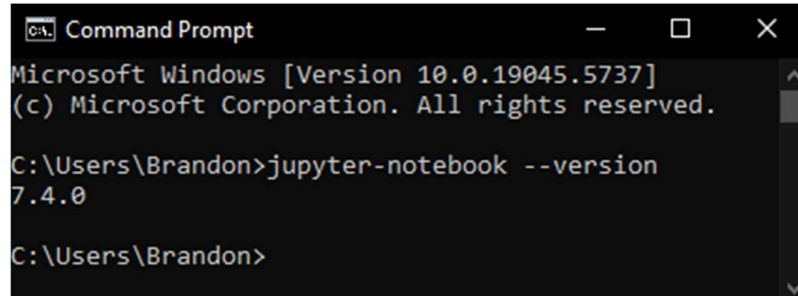


Figure 5.10: Python

5.1.8 Jupyter Notebook

Jupyter Notebook was installed through the pip package manager of Python. Figure 5.11 shows the installed Jupyter Notebook.



```
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Brandon>jupyter-notebook --version
7.4.0

C:\Users\Brandon>
```

Figure 5.11: Jupyter Notebook

A folder named “current_lane” was created in the “tflite” folder of the Flutter project. Then, a new Jupyter Notebook file named “model.ipynb” was created in the “current_lane” folder. Figure 5.12 shows the directory of the “current_lane” folder.

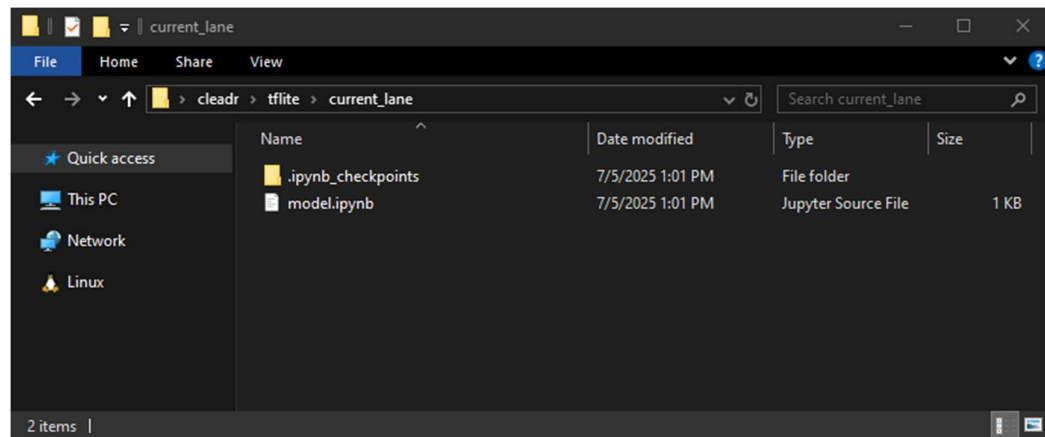


Figure 5.12: “current_lane” Directory

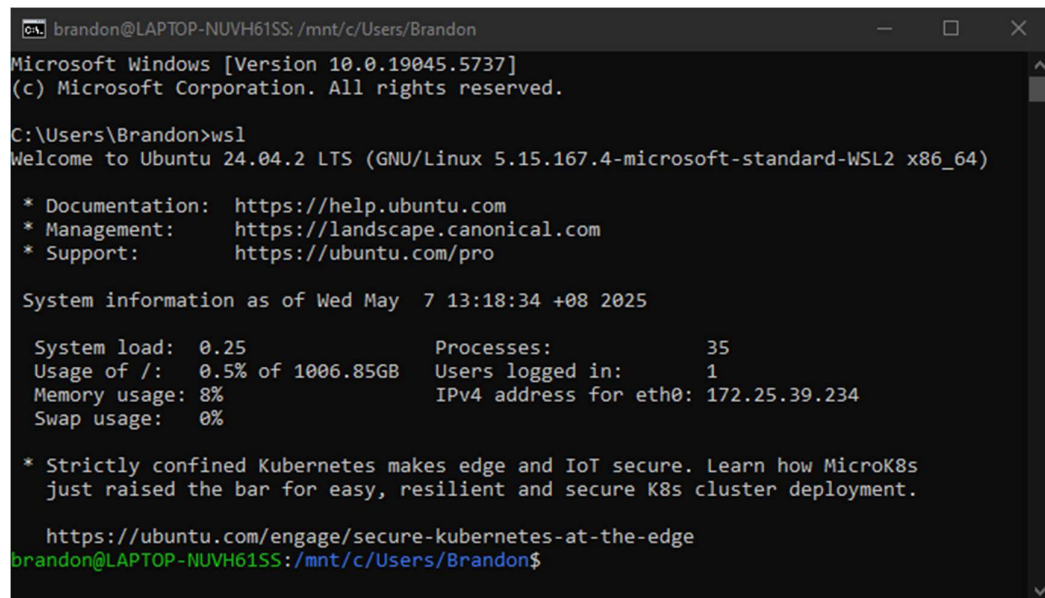
5.1.9 PyTorch

PyTorch was installed through the pip package manager of Python. The dependencies required by PyTorch were also installed. The dependencies are as follows:

1. torch 2.6.0+cu124
2. scikit-learn 1.6.1
3. tqdm 4.67.1
4. onnx 1.17.0
5. opencv-python 4.11.0.86
6. mss 10.0.0

5.1.10 WSL

WSL was installed through the terminal. Figure 5.13 shows the installed WSL.



```
brandon@LAPTOP-NUVH61SS: /mnt/c/Users/Brandon
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Brandon>wsl
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed May  7 13:18:34 +08 2025

System load:  0.25           Processes:            35
Usage of /:   0.5% of 1006.85GB Users logged in:          1
Memory usage: 8%           IPv4 address for eth0: 172.25.39.234
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge
brandon@LAPTOP-NUVH61SS:/mnt/c/Users/Brandon$
```

Figure 5.13: WSL

5.1.11 TensorFlow

TensorFlow was installed through the pip package manager of Python on WSL. The dependencies required by TensorFlow were also installed. The dependencies are as follows:

1. tensorflow 2.19.0
2. tf-keras 2.19.0
3. onnx 1.17.0
4. onnxruntime 1.18.1
5. onnx-simplifier 0.4.33
6. onnx_graphsurgeon 0.5.56
7. simple_onnx_processing_tools 1.1.32
8. onnx2tf 1.27.0
9. ml_dtypes 0.5.1
10. flatbuffers 24.3.25
11. psutil 5.9.5
12. ai-edge-litert 1.2.0

5.2 Settings and Configurations

This project involved some settings and configurations for the development of the system. Settings and configurations of software include dependencies, Google Maps API, Flutter Unity Widget, AR + GPS Location Asset, and TFLite.

5.2.1 Dependencies

6 dependencies from the official Dart package repository were included in the Flutter project. The dependencies are as follows:

- google_navigation_flutter 0.4.0
- flutter_unity_widget 2022.2.1 (local)
- http 1.3.0
- geolocator 13.0.2
- location 7.0.1
- permission_handler 11.3.1

The “flutter_unity_widget” package was installed locally from the GitHub repository [91]. The repository was cloned and included in the Flutter project. Figure 5.14 shows the included Dart packages.

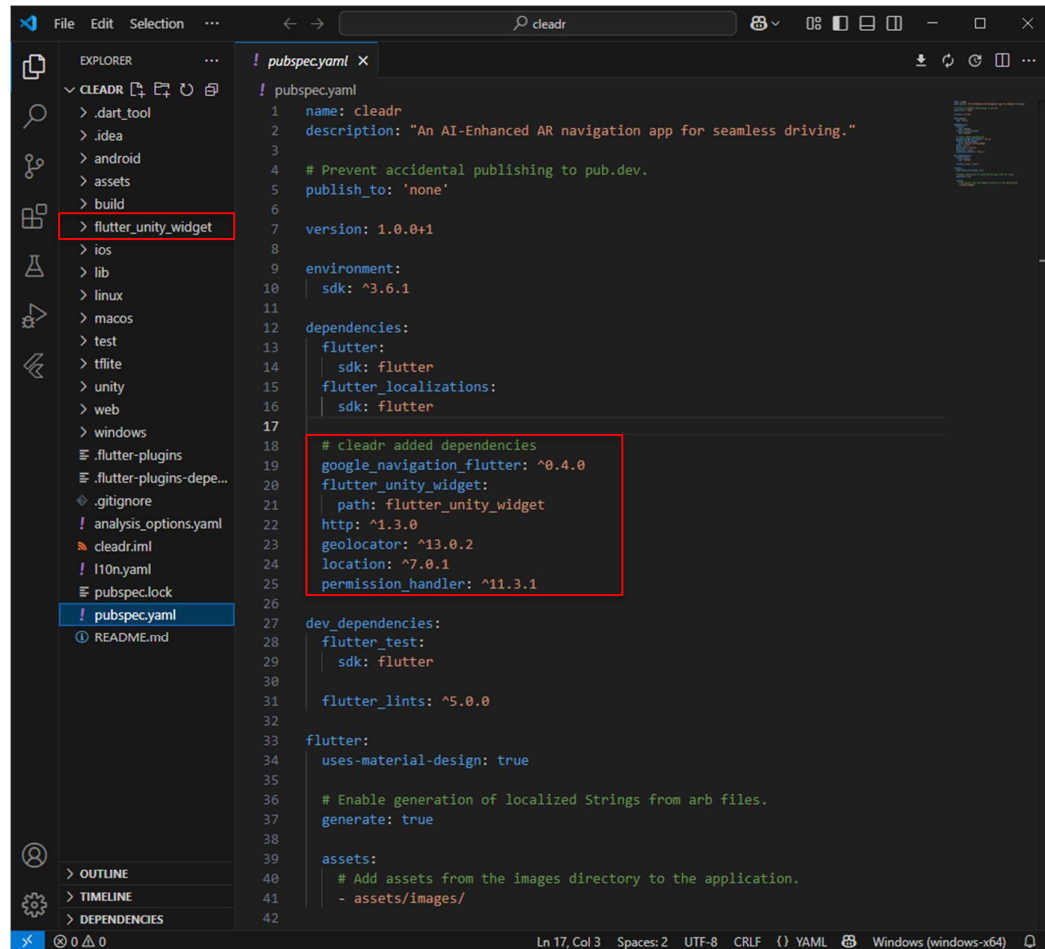


Figure 5.14: Dart Packages

5.2.2 Google Maps API

A Google Maps API key was obtained from the Google Cloud Console [92] for the Flutter project. Services such as Maps Embed API, Maps SDK for Android, Directions API, Navigation SDK, Places API, and Geocoding API were enabled for this project. Figure 5.15 shows the Google Maps API key settings.

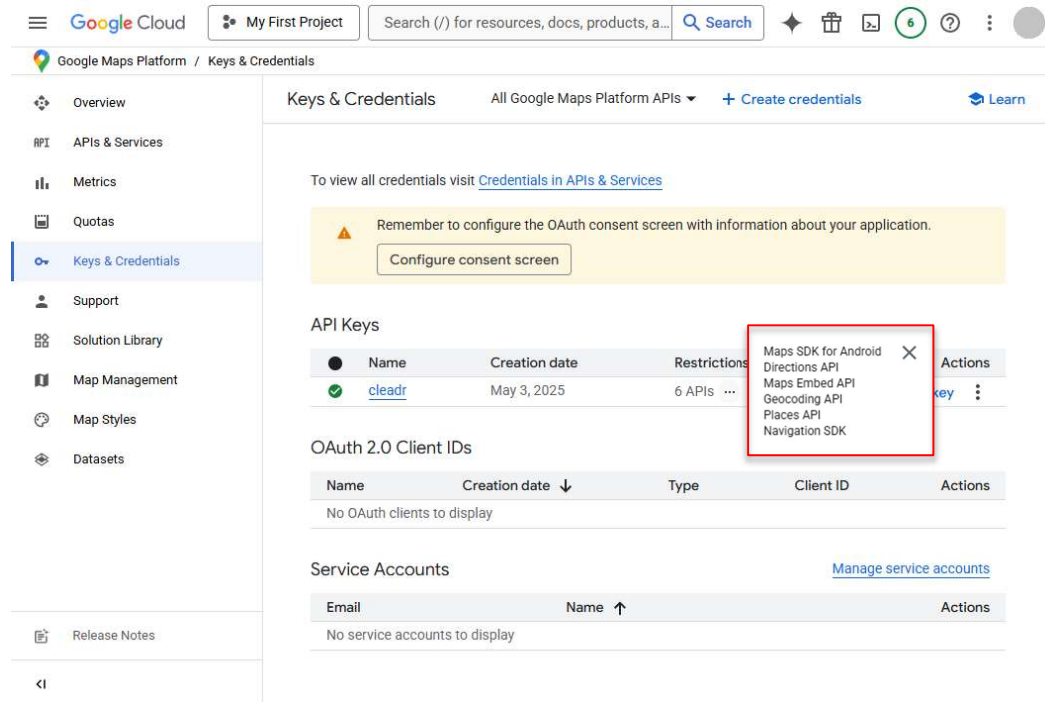


Figure 5.15: Google Maps API Key Settings

The Google Maps API key was included in the Flutter project, specifically the “android/app/src/main/AndroidManifest.xml” file. Figure 5.16 shows the code snippet of the AndroidManifest.xml file.

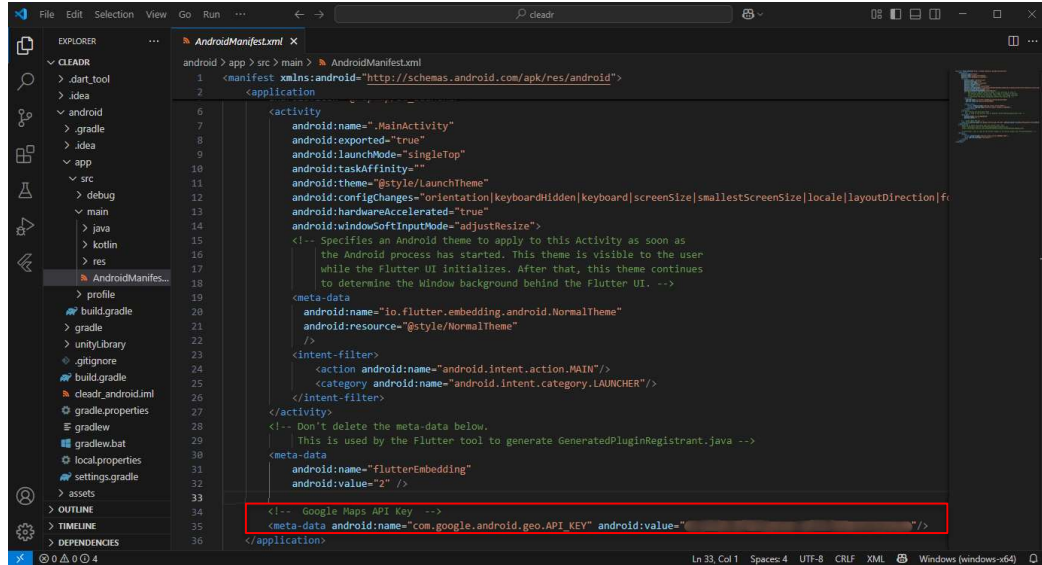


Figure 5.16: AndroidManifest.xml

5.2.3 Flutter Unity Widget

In the Unity Project, a package from the “unitypackages” was installed from [91]. Upon installation, the “Player” settings were configured. Figure 5.17 shows the configurations.

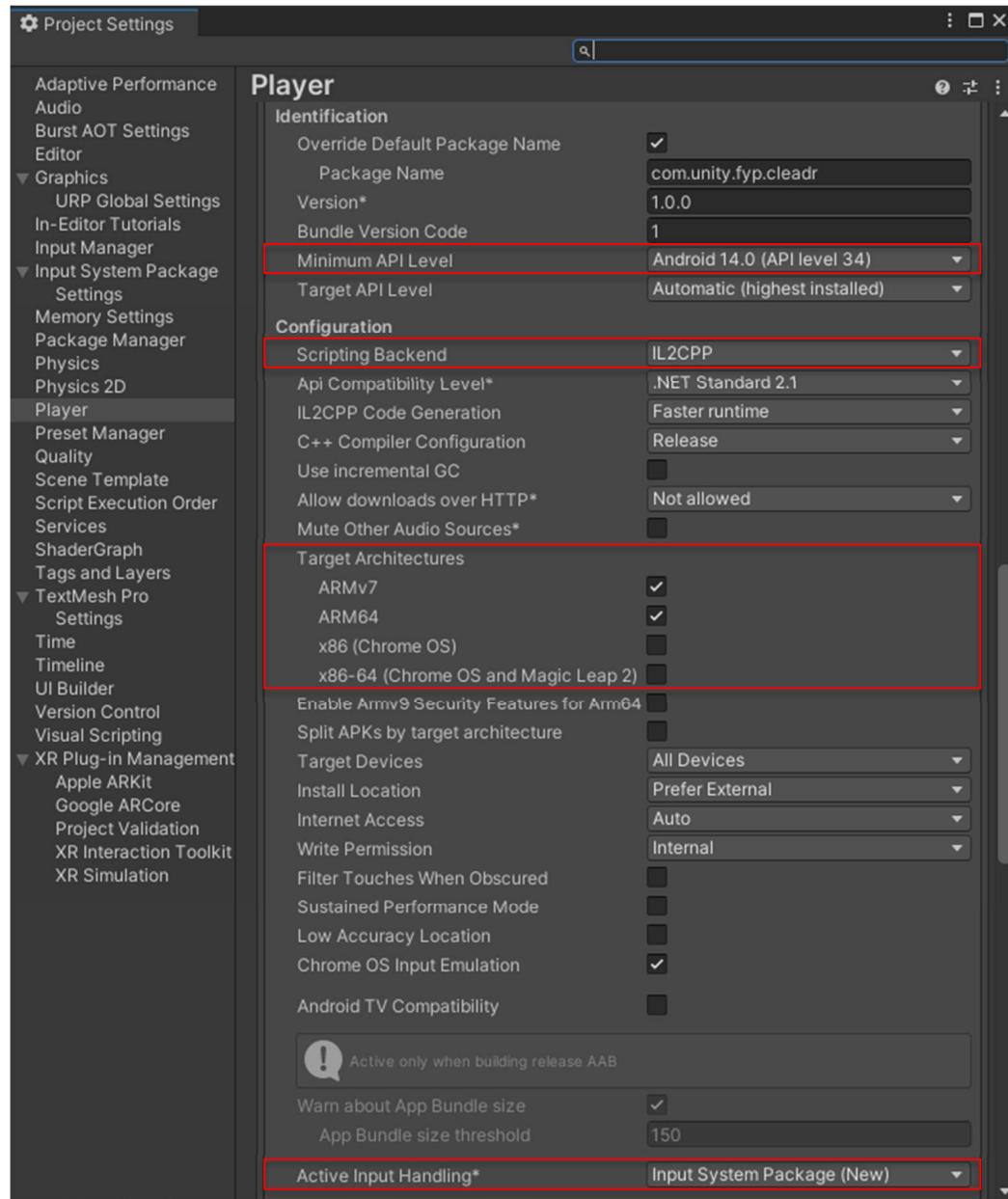


Figure 5.17: Player Settings

In the “Project” tab, configurations were made to the “Assets/FlutterUnityIntegration/Editor/Build.cs” file. Figure 5.18 and Figure 5.19 show the code snippets of Build.cs.

```

15 namespace FlutterUnityIntegration.Editor
16 {
17     // Unity Script reference
18     public class Build : EditorWindow
19     {
20         private static readonly string ProjectPath = Path.GetFullPath(Path.Combine(Application.dataPath, "."));
21         private static readonly string APKPath = Path.Combine(ProjectPath, "Builds/" + Application.productName + ".apk");
22         private static readonly string AndroidExportPath = Path.GetFullPath(Path.Combine(ProjectPath, "../../android/unitylibrary"));
23         private static readonly string WindowsExportPath = Path.GetFullPath(Path.Combine(ProjectPath, "../../windows/unitylibrary/data"));
24         private static readonly string IOSExportPath = Path.GetFullPath(Path.Combine(ProjectPath, "../../ios/unitylibrary"));
25         private static readonly string WebExportPath = Path.GetFullPath(Path.Combine(ProjectPath, "../../web/unitylibrary"));
26         private static readonly string IOSExportPluginPath = Path.GetFullPath(Path.Combine(ProjectPath, "../../ios_xcode/unitylibrary"));
27
28         private bool _pluginMode = false;
29         private static string _persistentKey = "Flutter-unity-widget-pluginMode";
30
31         // Sets the NDK version
32         // Note: This uses Android Studio's NDK path (Android Studio -> Tools -> SDK Manager -> SDK Tools -> NDK)
33         private static string ndkVersion = "27.0.12077773";
34         private static string ndkPath = $"{Environment.GetFolderPath(Environment.SpecialFolder.UserProfile).Replace("\\", "/")}/AppData/Local/Android/Sdk/ndk/{ndkVersion}";
35     }

```

Figure 5.18: Build.cs Configuration 1

```

332 private static void ModifyAndroidGradle(bool isPlugin)
333 {
334     // Modify build.gradle
335     var buildFile = Path.Combine(AndroidExportPath, "build.gradle");
336     var buildText = File.ReadAllText(buildFile);
337     buildText = buildText.Replace("com.android.application", "com.android.library");
338     buildText = buildText.Replace("bundle {", "splits {");
339     buildText = buildText.Replace("enableSplit = false", "enable false");
340     buildText = buildText.Replace("enableSplit = true", "enable true");
341     buildText = buildText.Replace("implementation fileTree(dir: 'libs', include: ['*.jar'])", "implementation(name: 'unity-classes', ext: 'jar')");
342     buildText = buildText.Replace("unityStreamingAssets.tokenize(", ")", "");
343     // Disable the Unity ndk path as it will conflict with Flutter
344     // buildText = buildText.Replace("ndkPath \"", "// ndkPath \"");
345     buildText = buildText.Replace("\"C:/Program Files/Unity/Hub/Editor/2022.3.55f1/Editor/Data/PlaybackEngines/AndroidPlayer/NDK\"", $"\"{ndkPath}\"");
346
347     // check for namespace definition (Android gradle plugin 8+), add a backwards compatible version if it is missing.
348     if (!buildText.Contains("namespace"))
349     {
350         buildText = buildText.Replace("compileOptions {",
351             "if (project.android.hasProperty(\"namespace\")) {\n        namespace 'com.unity3d.player'\n    }\n\n    compileOptions {");
352     }
353 }

```

Figure 5.19: Build.cs Configuration 2

The Unity project was built and exported for the Flutter project named as “unityLibrary” in the “android” folder. “Export Android (Debug)” and “Export Android (Release)” were used for the debug and release build respectively. Figure 5.20 and Figure 5.21 show the exported build of the Unity project.

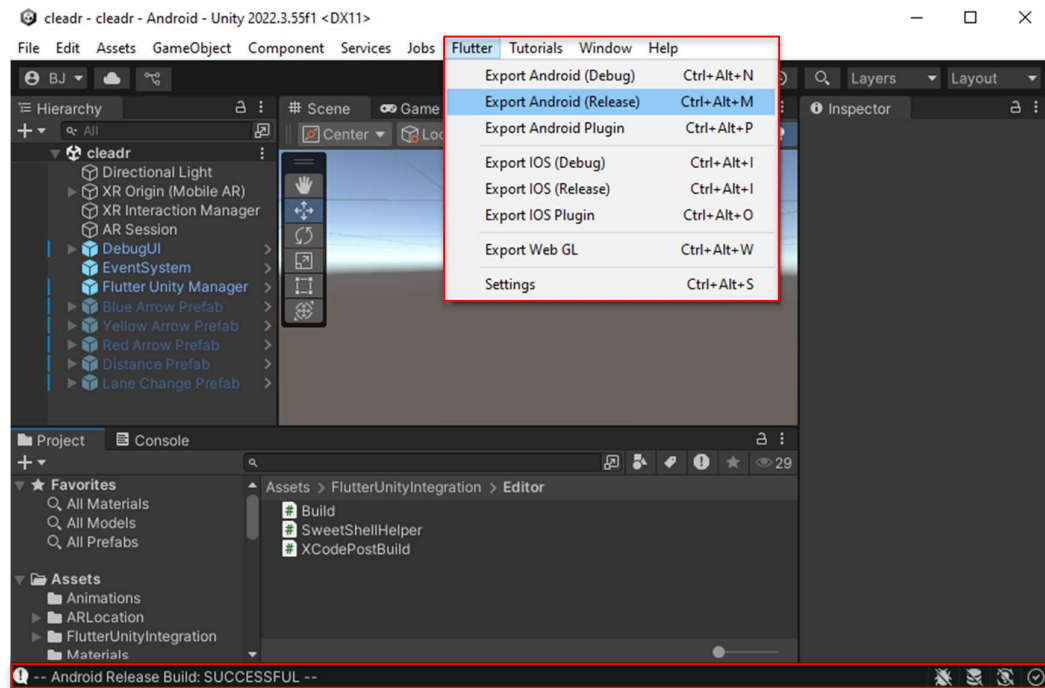


Figure 5.20: Exported Unity Project 1

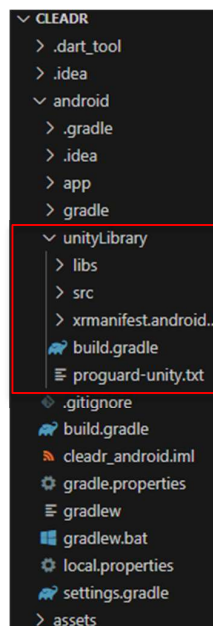


Figure 5.21: Exported Unity Project 2

Configurations were being made to the Flutter project. Figure 5.22 to Figure 5.26 show the code snippet of the configurations.

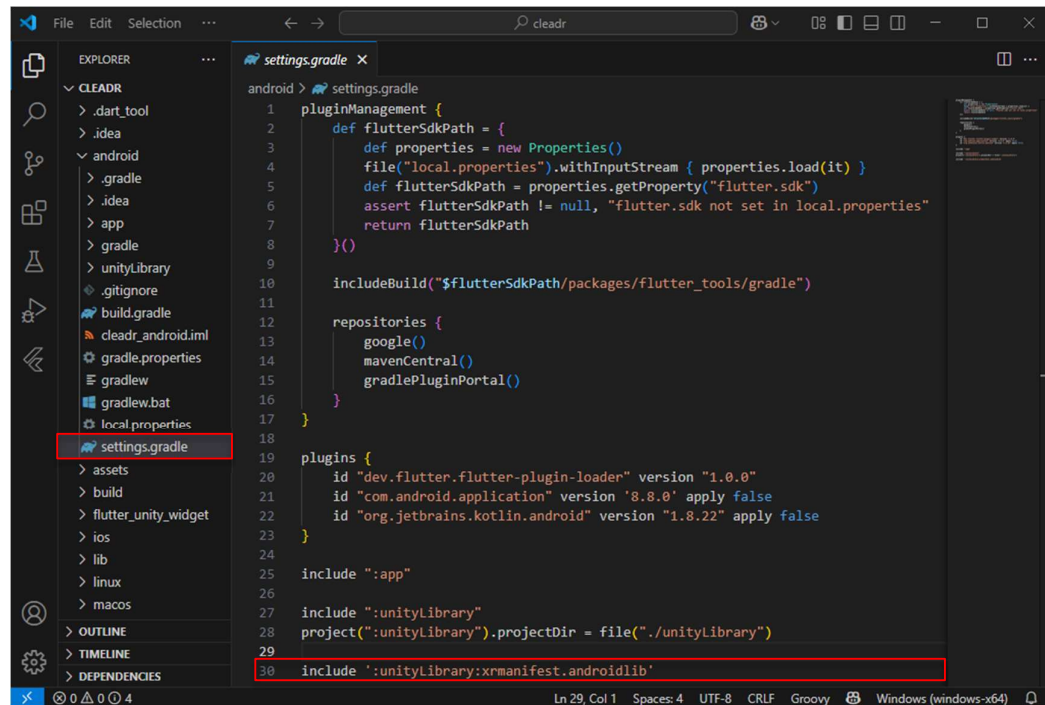


Figure 5.22: settings.gradle

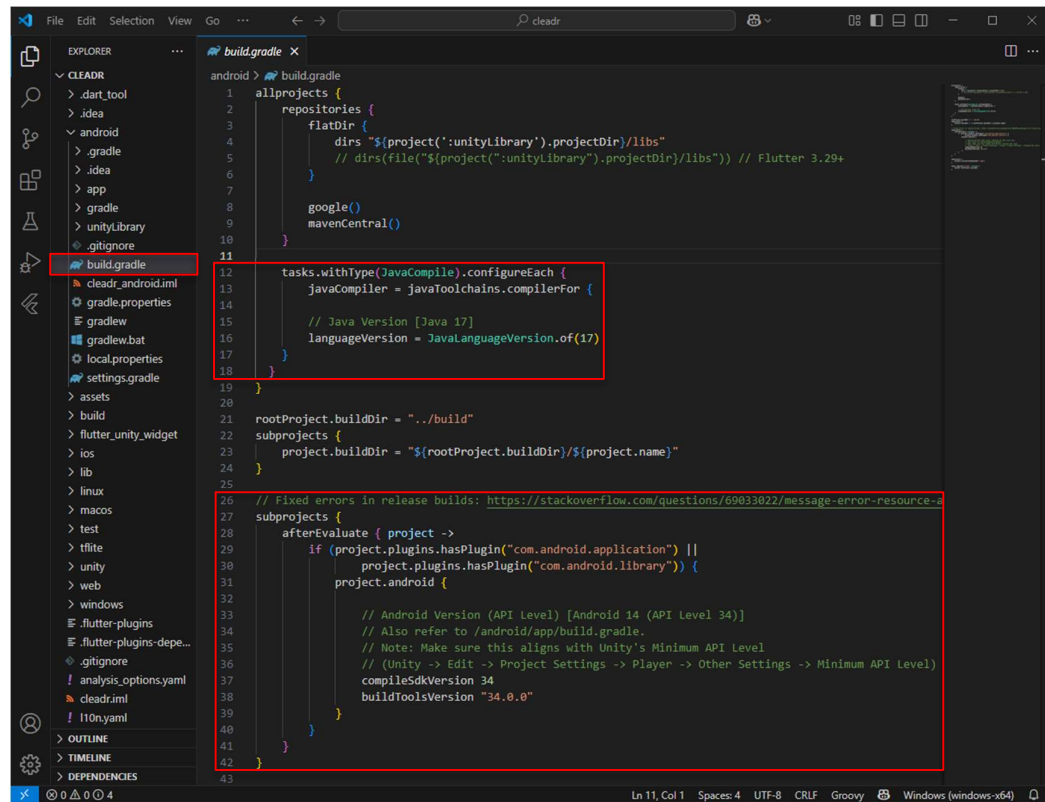


Figure 5.23: build.gradle (Project)

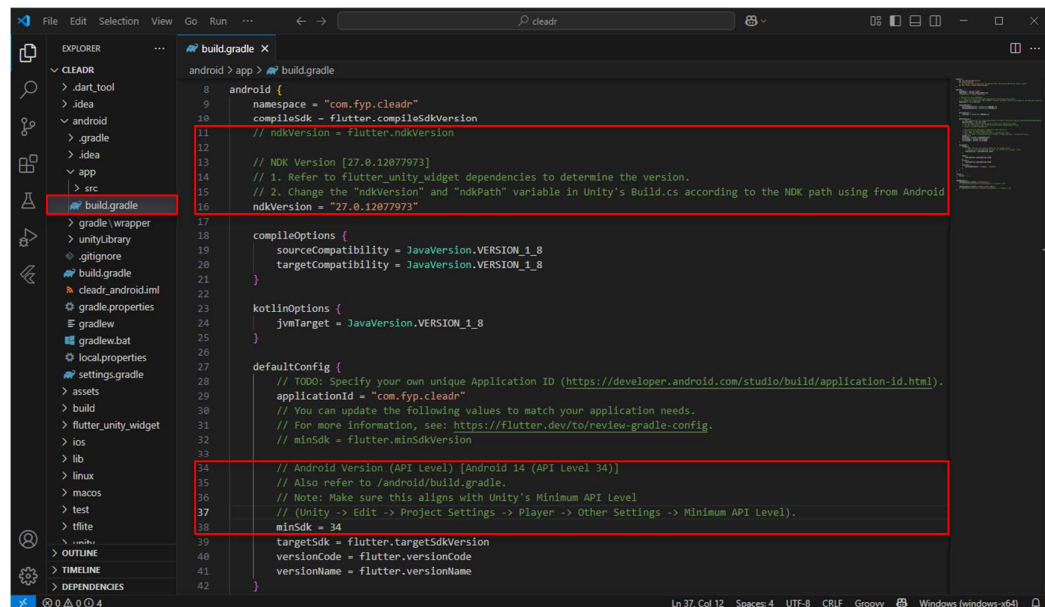


Figure 5.24: build.gradle (Application) Configuration 1

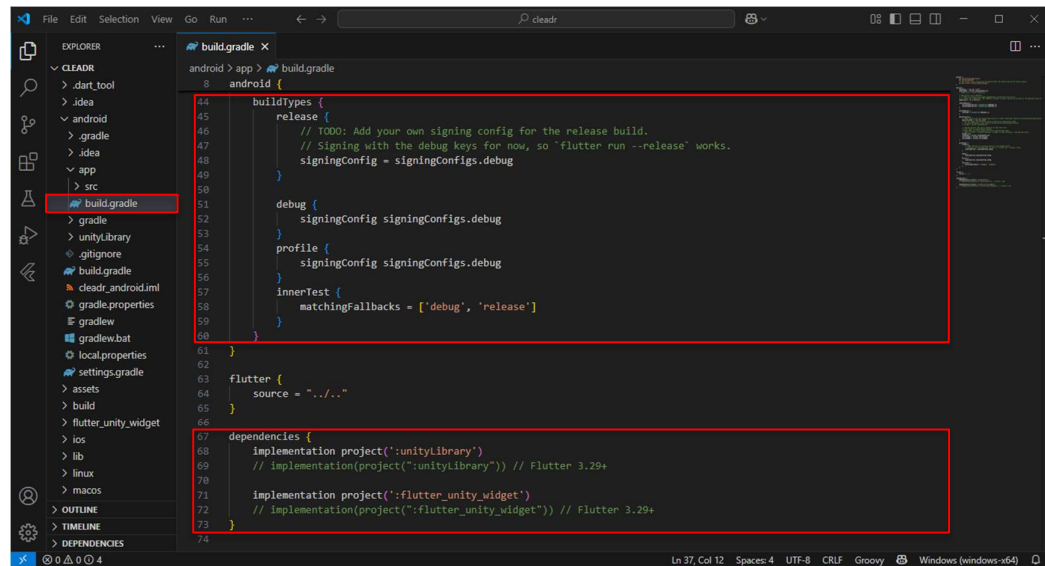


Figure 5.25: build.gradle (Application) Configuration 2

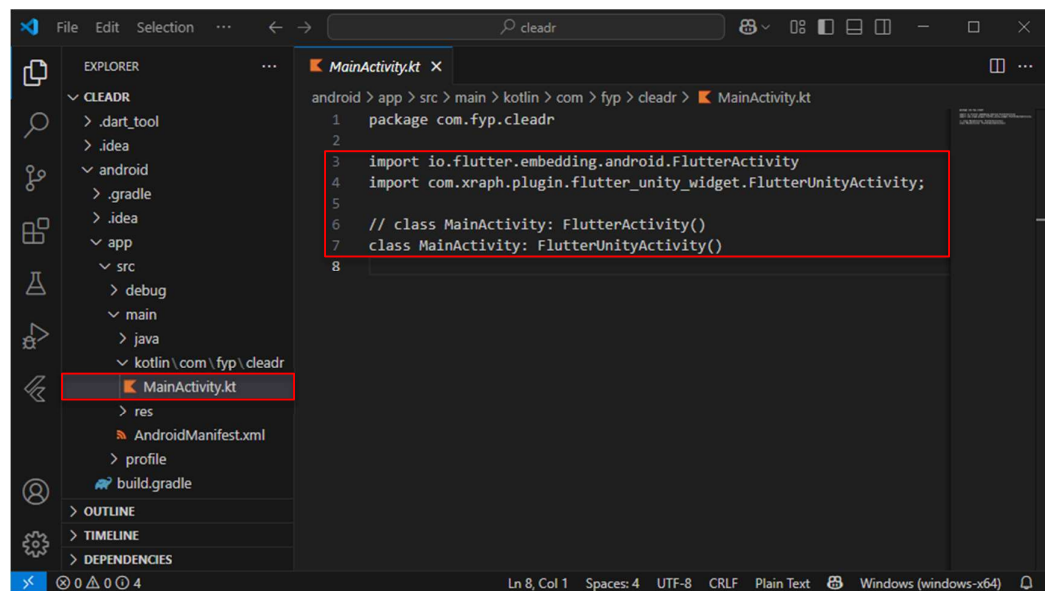


Figure 5.26: MainActivity.kt

5.2.4 AR + GPS Location Asset

AR + GPS Location Asset was installed from the Unity Asset Store [93]. Figure 5.27 shows the installed Unity asset.

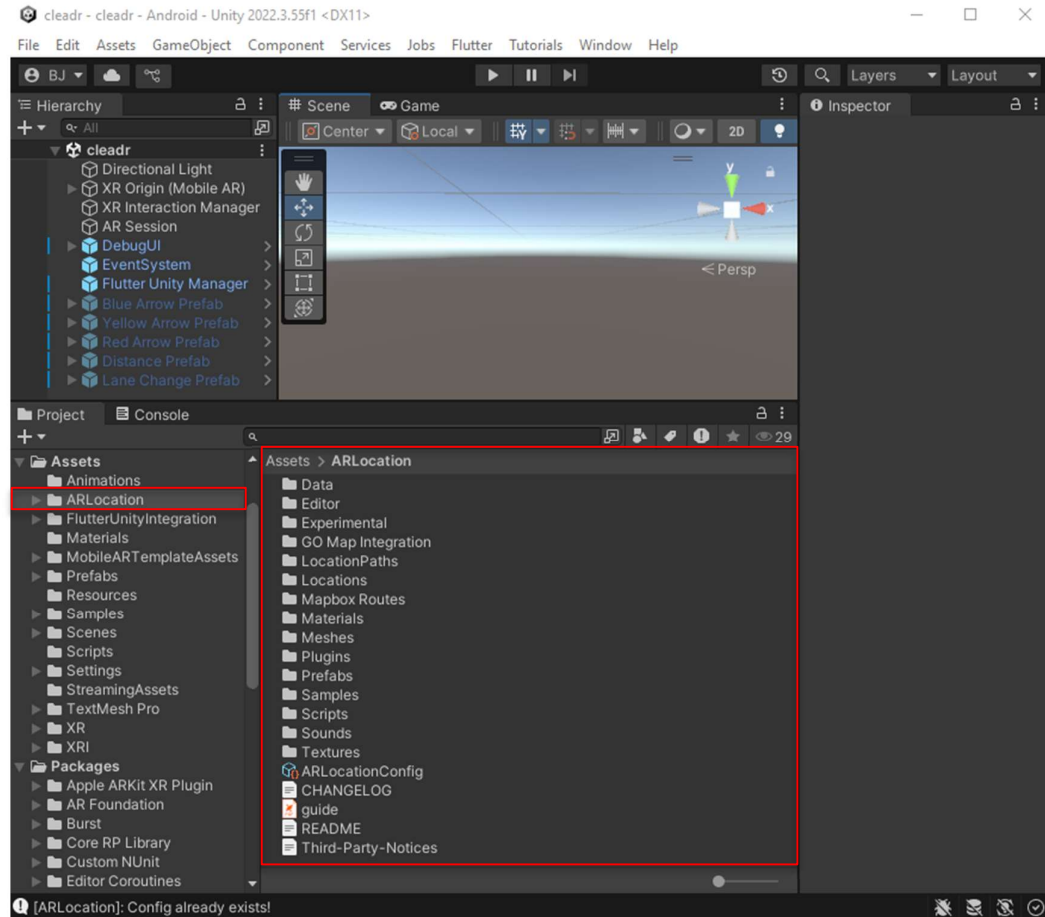


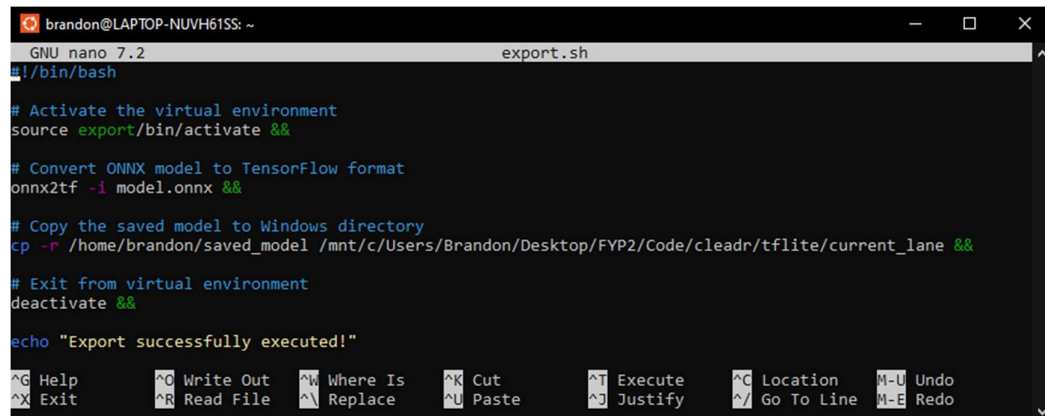
Figure 5.27: AR + GPS Location Asset

5.2.5 TFLite

In WSL, it was used to export the model of ONNX file format to TFLite file format. The conversion steps are as follows:

1. Open WSL.
2. Create a Python virtual environment.
3. Move the ONNX file (model.onnx) to WSL.
4. Activate the virtual environment.
5. Run “onnx2tf -i model.onnx”.
6. TFLite file is created.

A bash script named “export.sh” was created in WSL to automate the conversion steps. Figure 5.28 shows the code snippet of “export.sh”.



```

brandon@LAPTOP-NUVH61SS: ~
GNU nano 7.2 export.sh
#!/bin/bash

# Activate the virtual environment
source export/bin/activate &&

# Convert ONNX model to TensorFlow format
onnx2tf -i model.onnx &&

# Copy the saved model to Windows directory
cp -r /home/brandon/saved_model /mnt/c/Users/Brandon/Desktop/FYP2/Code/cleadr/tflite/current_lane &&

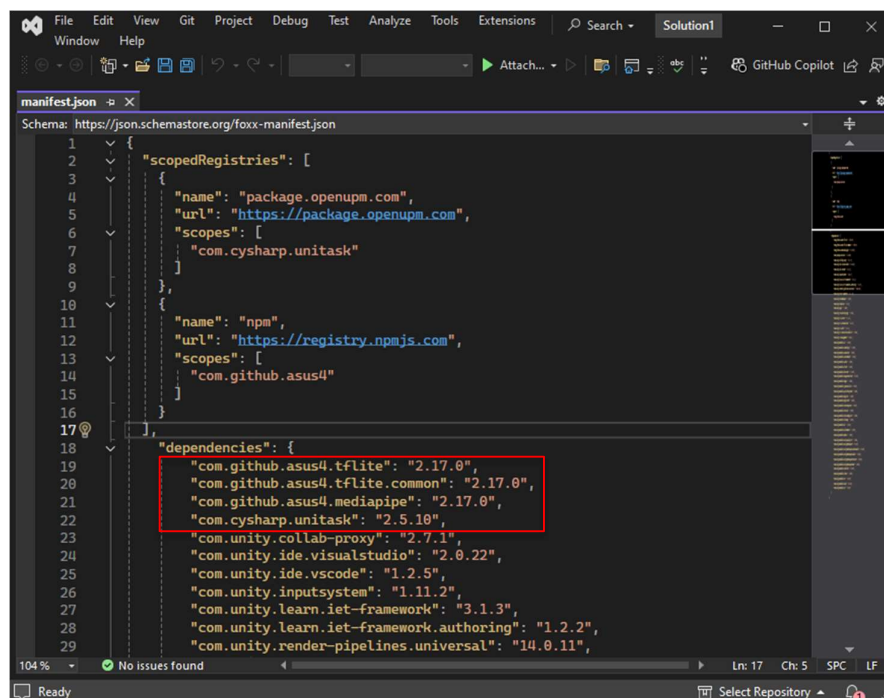
# Exit from virtual environment
deactivate &&

echo "Export successfully executed!"

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  M-U Undo
^X Exit      ^R Read File ^M Replace   ^J Paste     ^_ Justify  ^_/ Go To Line M-E Redo
  
```

Figure 5.28: export.sh

In the Unity project, the TFLite package was installed from [94]. Configurations were made to the “Packages/manifest.json” file to install the package through the Unified Package Manager for Unity. Figure 5.29 shows the code snippet of “manifest.json”.



```

File Edit View Git Project Debug Test Analyze Tools Extensions Search Solution
Window Help
manifest.json Schema: https://json.schemastore.org/foxx-manifest.json
1  {
2    "scopedRegistries": [
3      {
4        "name": "package.openupm.com",
5        "url": "https://package.openupm.com",
6        "scopes": [
7          "com.cysharp.unitask"
8        ]
9      },
10     {
11       "name": "npm",
12       "url": "https://registry.npmjs.com",
13       "scopes": [
14         "com.github.asus4"
15       ]
16     }
17   ],
18   "dependencies": {
19     "com.github.asus4.tflite": "2.17.0",
20     "com.github.asus4.tflite.common": "2.17.0",
21     "com.github.asus4.mediapipe": "2.17.0",
22     "com.cysharp.unitask": "2.5.10",
23     "com.unity.collab-proxy": "2.7.1",
24     "com.unity.ide.visualstudio": "2.0.22",
25     "com.unity.ide.vscode": "1.2.5",
26     "com.unity.inputsystem": "1.11.2",
27     "com.unity.learn.iet-framework": "3.1.3",
28     "com.unity.learn.iet-framework.authoring": "1.2.2",
29     "com.unity.render-pipelines.universal": "14.0.11",
  
```

Figure 5.29: manifest.json

5.3 System Operation

Figure 5.30 to Figure 5.46 show the demonstrations and descriptions of the system.

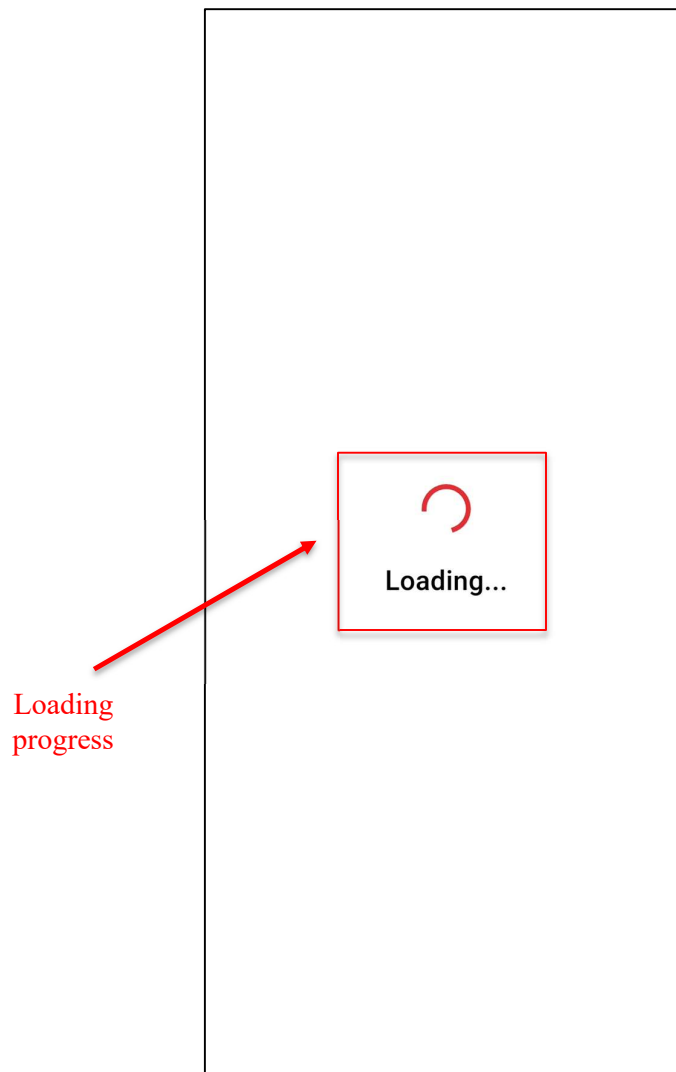


Figure 5.30: Loading Screen

Figure 5.30 shows the Loading Screen that indicates the user to wait for the application to be launched. The application checks for the availability of the required services in the background such as Internet connection, acceptance of terms and conditions, Google Maps API, location permission, and camera permission before launching the application.

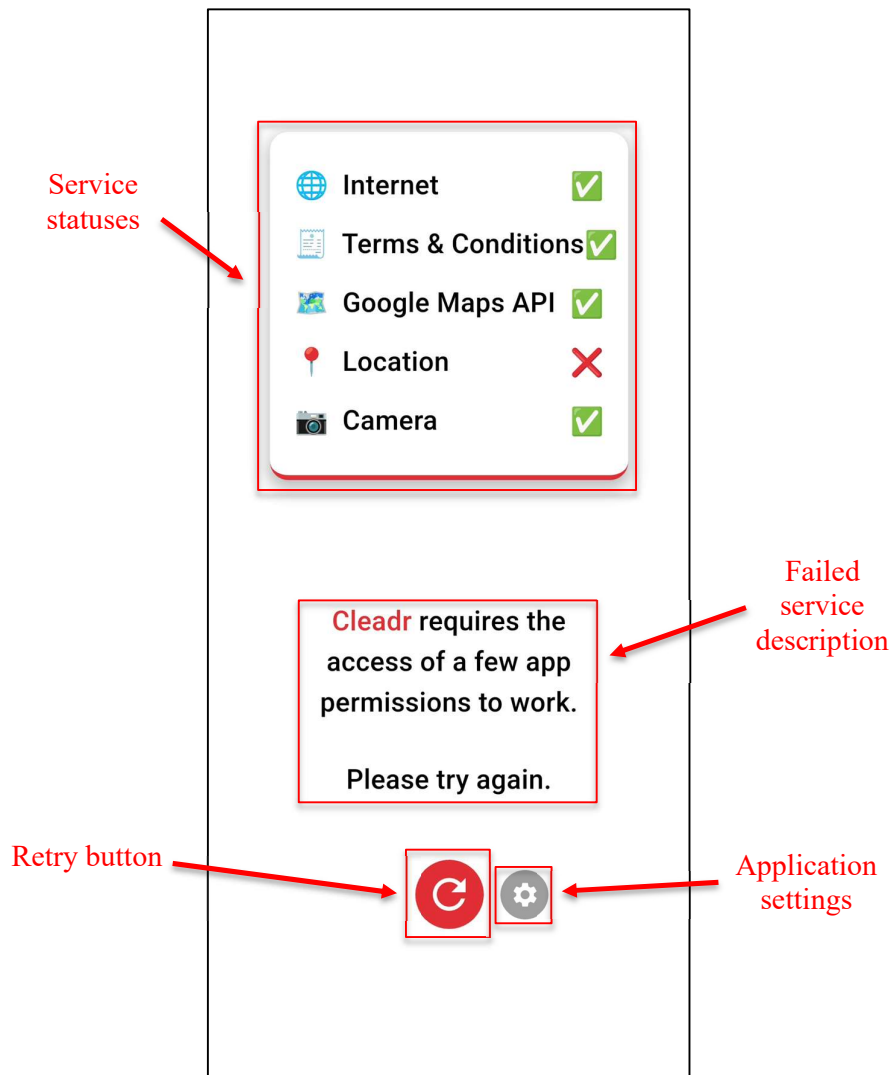


Figure 5.31: Loading – Failed Screen

Figure 5.31 shows the Loading – Failed Screen when the application has failed to be launched due to the failed services. As shown in Figure 5.31, the failed service is the location permission. Therefore, the user is given a retry button once the location permission has been given to the application. Additionally, an application settings button redirects the user to the application settings page for quick access to the permission settings.

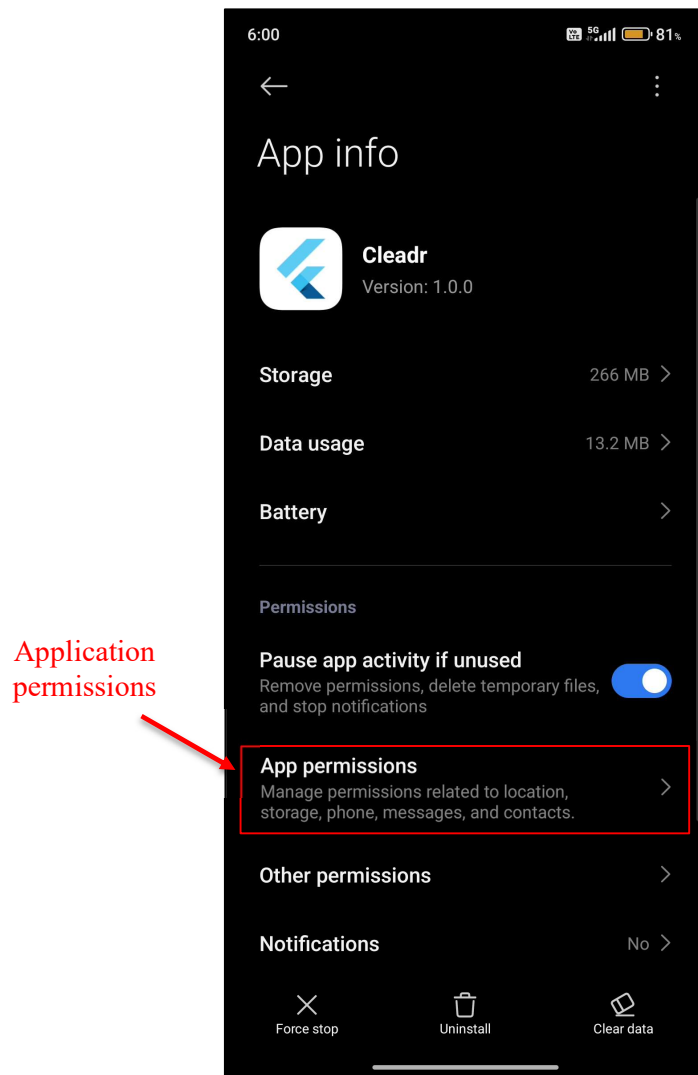


Figure 5.32: Loading – Failed – Application Settings Screen

Figure 5.32 shows the Loading – Failed – Application Settings Screen that was redirected from the application settings button of the Loading – Failed Screen. The user can allow the required application permissions through the application settings.

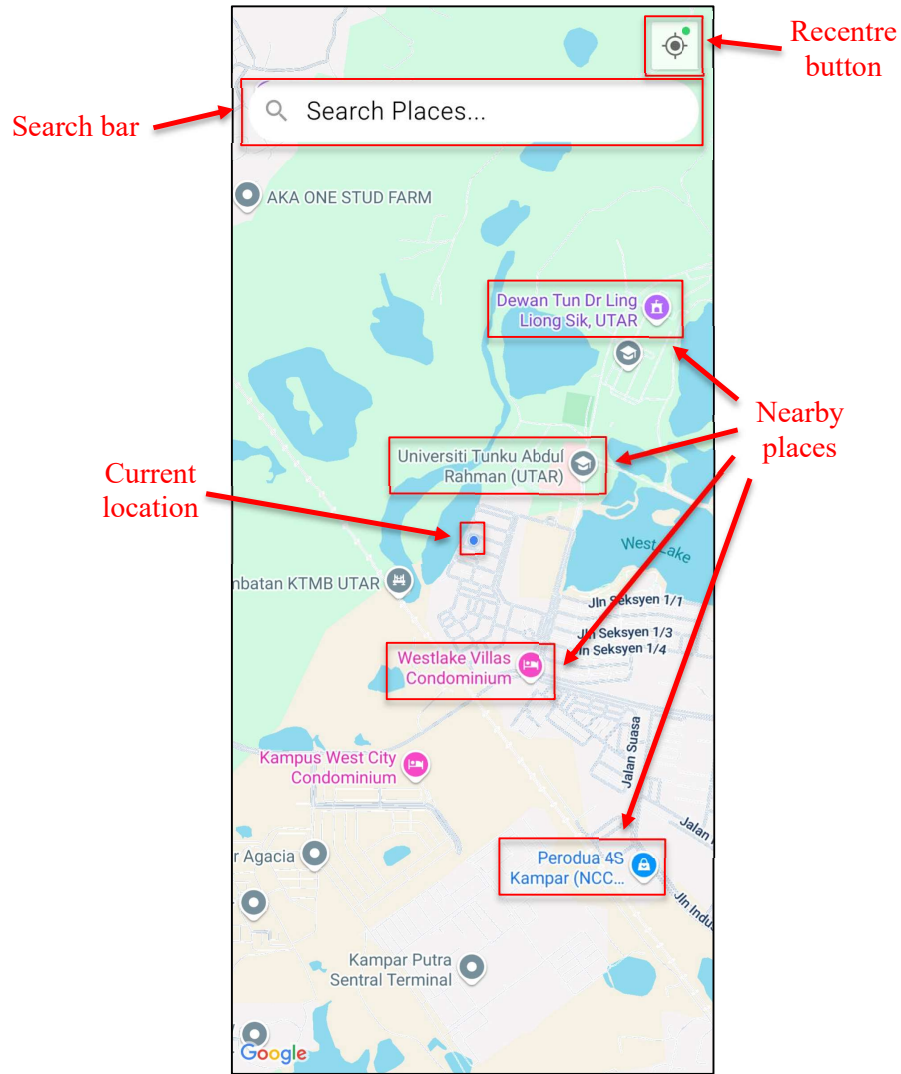


Figure 5.33: Maps Screen

Figure 5.33 shows the Maps Screen that loads the map based on the current location of the user. It has the recentre button, search bar, current location, and nearby place pins.

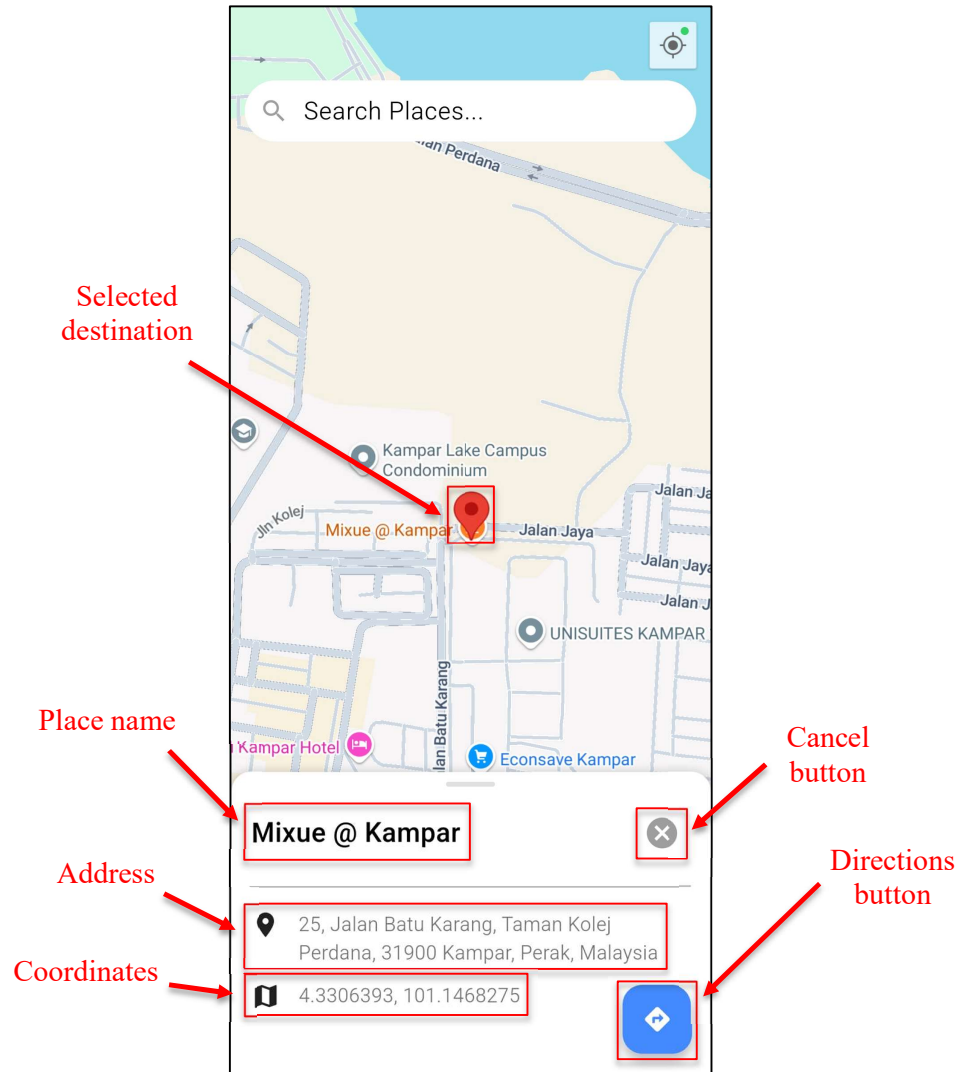


Figure 5.34: Maps – Selected Screen

Figure 5.34 shows the Maps – Selected Screen that indicates the selected destination by tapping on the map. It shows the place name, address, and coordinates of the selected destination. Note that the place name may not be available for every location, instead the address will be shown as the place name. The user can also cancel the selected destination to choose another destination. Once the user has confirmed, the directions button will preview the route from the current location to the destination.

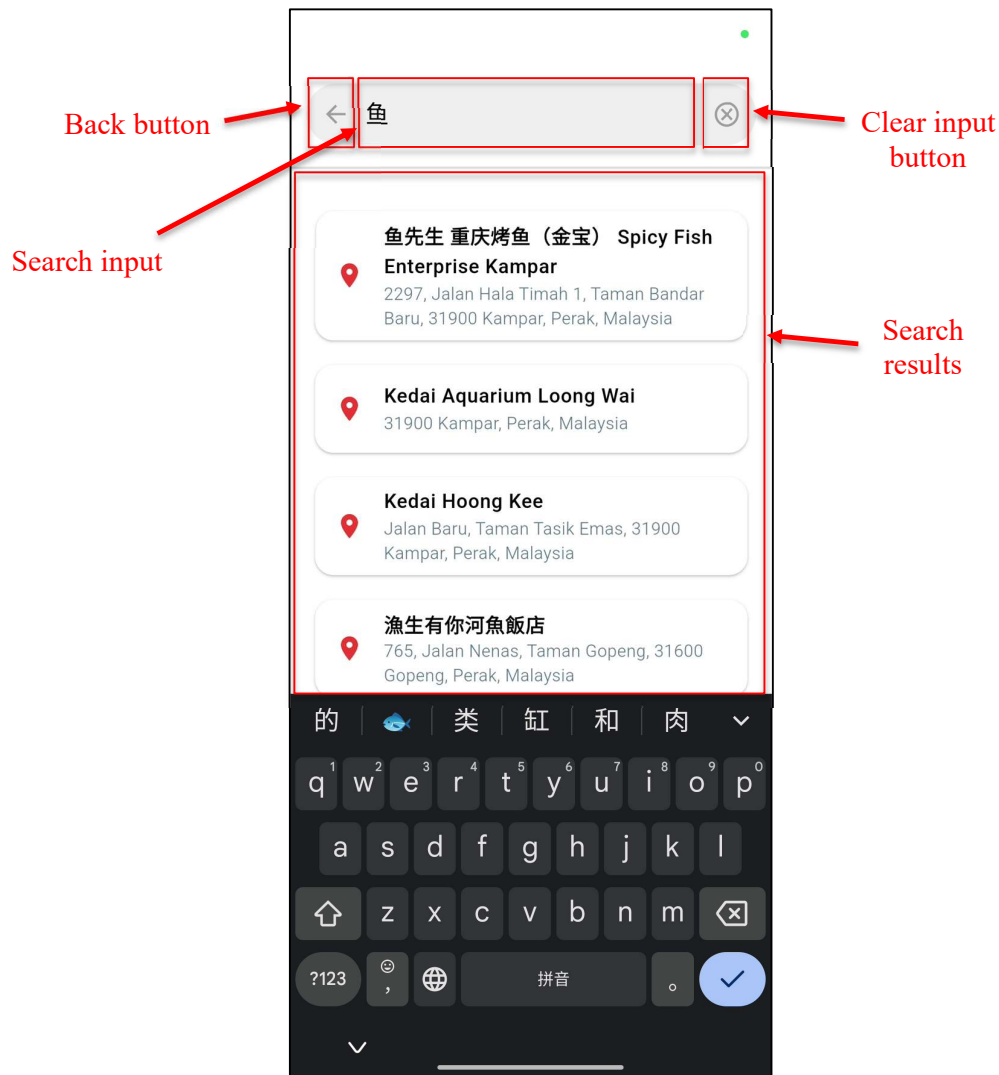


Figure 5.35: Maps – Search Screen

Figure 5.35 shows the Maps – Search Screen that allows the user to type and input queries to search for places. It autocompletes the query and returns the most relevant search results. Additionally, the user can navigate back or clear the input. Once a desired search result is found, the user can tap on the search result to select it as the destination.

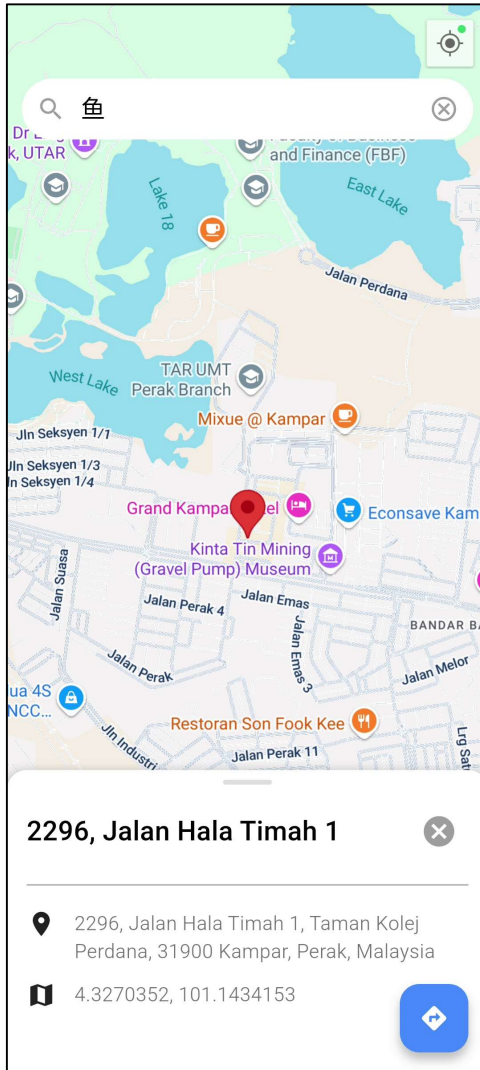


Figure 5.36: Maps – Search – Selected Screen

Figure 5.36 shows the Maps – Search – Selected Screen that the user selected from the search results. Note that the search result selected does not contain a place name. Therefore, the address name is used as the place name.

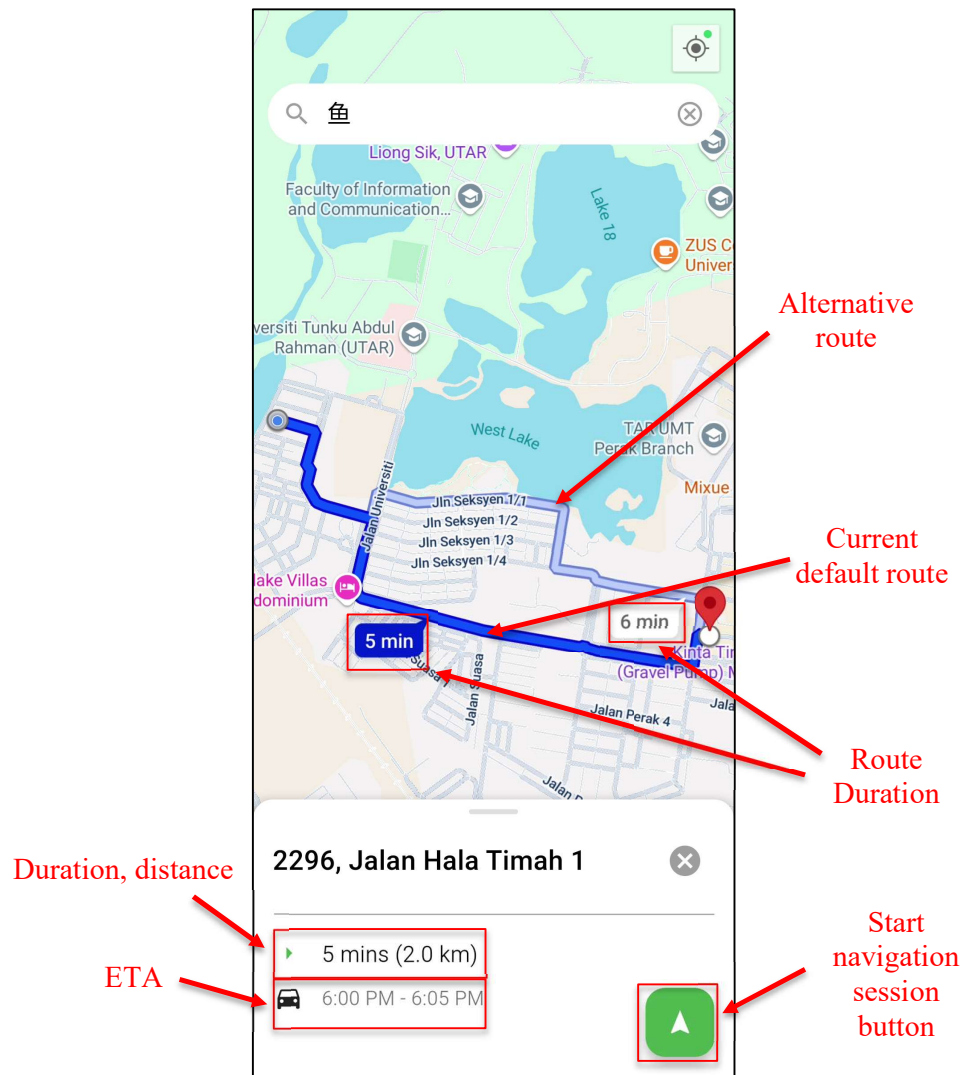


Figure 5.37: Maps – Directions Screen

Figure 5.37 shows the Maps – Directions Screen that previews the route from the current location to the destination. It shows the current default route and the alternative route. The user can switch to the alternative route if wanted by tapping on it. Additionally, the duration required of each route is shown on the map. The duration, distance, and ETA are also shown as navigation information about the route. Once the user has confirmed the destination route, the journey can be started through the start navigation session button.

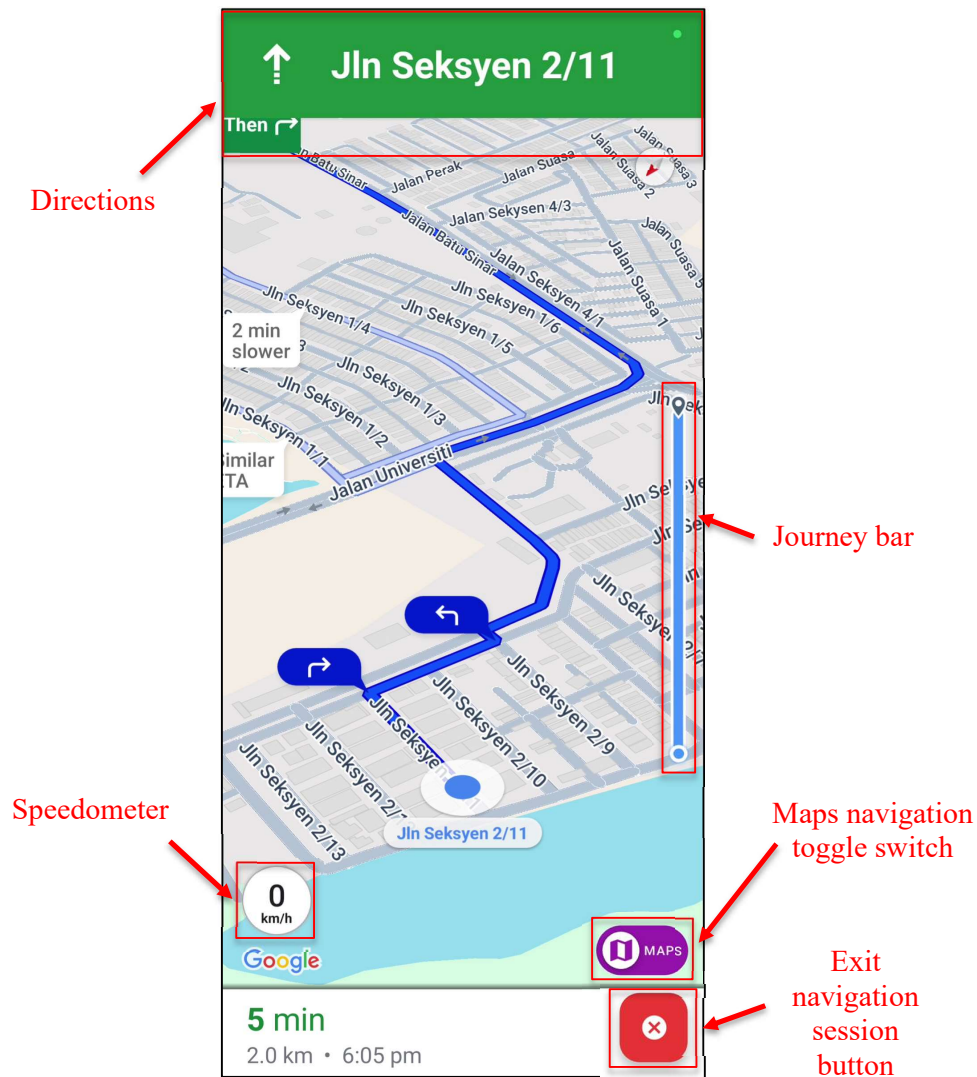


Figure 5.38: Maps Navigation Screen

Figure 5.38 shows the Maps Navigation Screen after the starting of a navigation session. It shows the directions on the top, the journey bar on the right, and the speedometer on the bottom left. The user can exit the navigation session at any time through the exit navigation session button. Additionally, the user can tap on the maps navigation toggle switch to change the navigation mode to AR navigation.

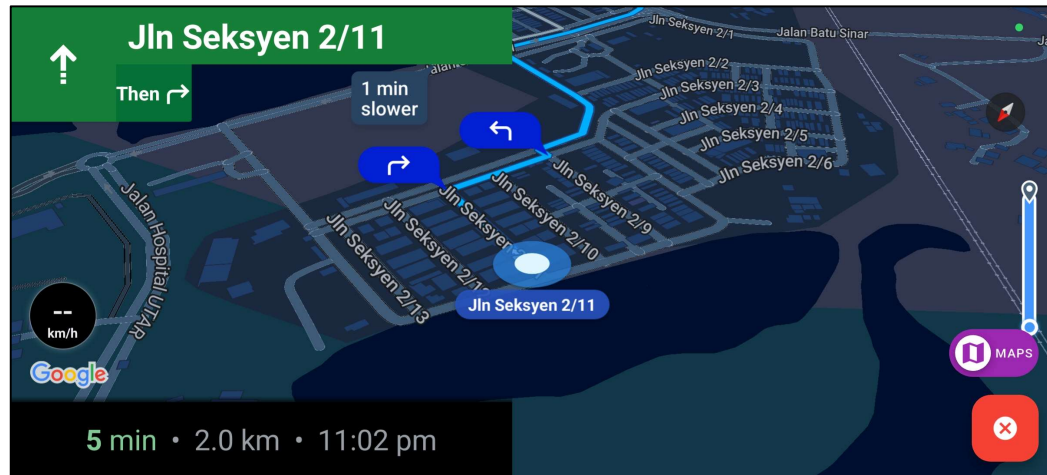


Figure 5.39: Maps Navigation – Horizontal Screen

Figure 5.39 Maps Navigation – Horizontal Screen shows the horizontal view of the maps navigation. Additionally, it shows a dark themed UI during the night.

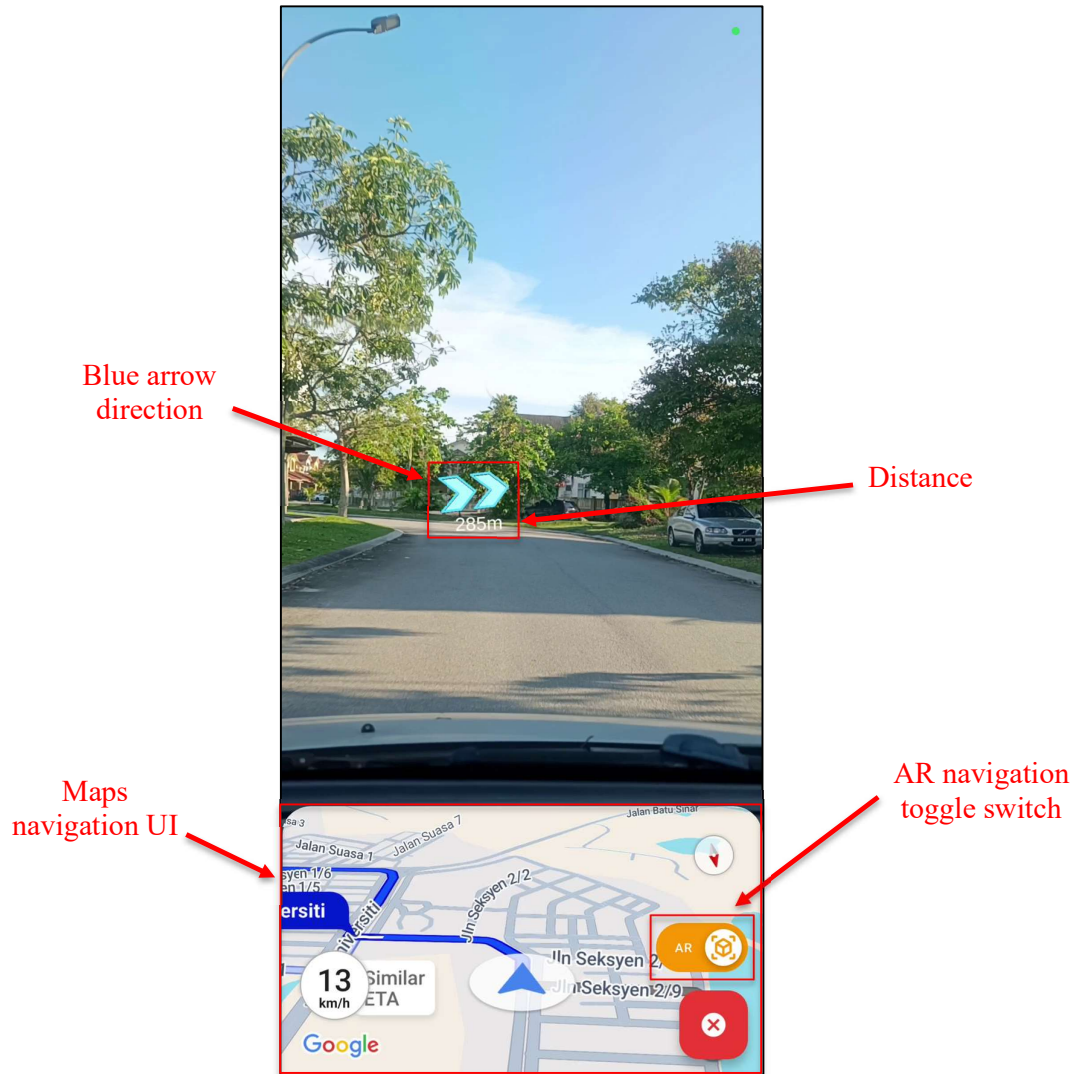


Figure 5.40: AR Navigation – Blue Arrow Screen

Figure 5.40 shows the AR Navigation – Blue Arrow Screen when switched to AR navigation. It opens the camera to render AR directions for the AR navigation. The blue arrow direction shows to turn right with its distance below. The blue colour represents that the turn is still far but to alert the user of the upcoming turn. The blue arrow direction is shown between distances of 200 metres to 500 metres. Additionally, a maps navigation UI is attached at the bottom for better navigational reference. The user can switch back to the maps navigation mode through the AR navigation toggle switch.



Figure 5.41: AR Navigation – Yellow Arrow Screen

Figure 5.41 shows the AR Navigation – Yellow Arrow Screen that represents a yellow arrow direction when the upcoming turn is between 50 to 200 metres. Additionally, the yellow arrow is placed closer to the screen that indicates the upcoming turn is getting closer. It indicates that the upcoming turn is incoming, but the turn is not yet reached. Hence, Figure 5.41 shows that the visible turn ahead is not the upcoming turn of the route, distinguishing the turns.

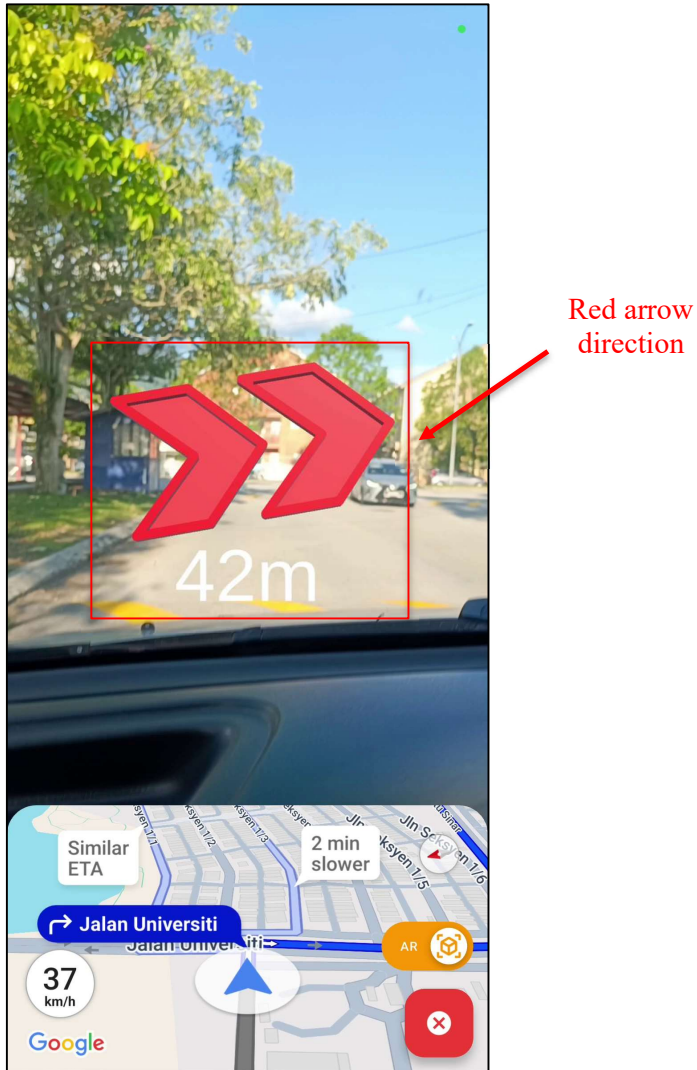


Figure 5.42: AR Navigation – Red Arrow Screen

Figure 5.42 shows the AR Navigation – Red Arrow Screen that represents a red arrow direction when the upcoming turn is between 0 to 50 metres. The red arrow direction places the closest to the screen, indicating to the user to make the next turn. Compared to the yellow arrow direction, the red arrow directions indicates to the user that the closest next turn is the upcoming turn.



Figure 5.43: AR Navigation – Horizontal – Blue Arrow Screen

Figure 5.43 shows the AR Navigation – Horizontal – Blue Arrow Screen that has the landscape view with the AR navigation on the blue arrow direction.



Figure 5.44: AR Navigation – Horizontal – Yellow Arrow Screen

Figure 5.44 shows the AR Navigation – Horizontal – Yellow Arrow Screen that has the landscape view of AR navigation on the yellow arrow direction.



Figure 5.45: AR Navigation – Horizontal – Red Arrow Screen

Figure 5.45 shows the AR Navigation – Horizontal – Red Arrow Screen that has the landscape view with the AR navigation on the red arrow direction.

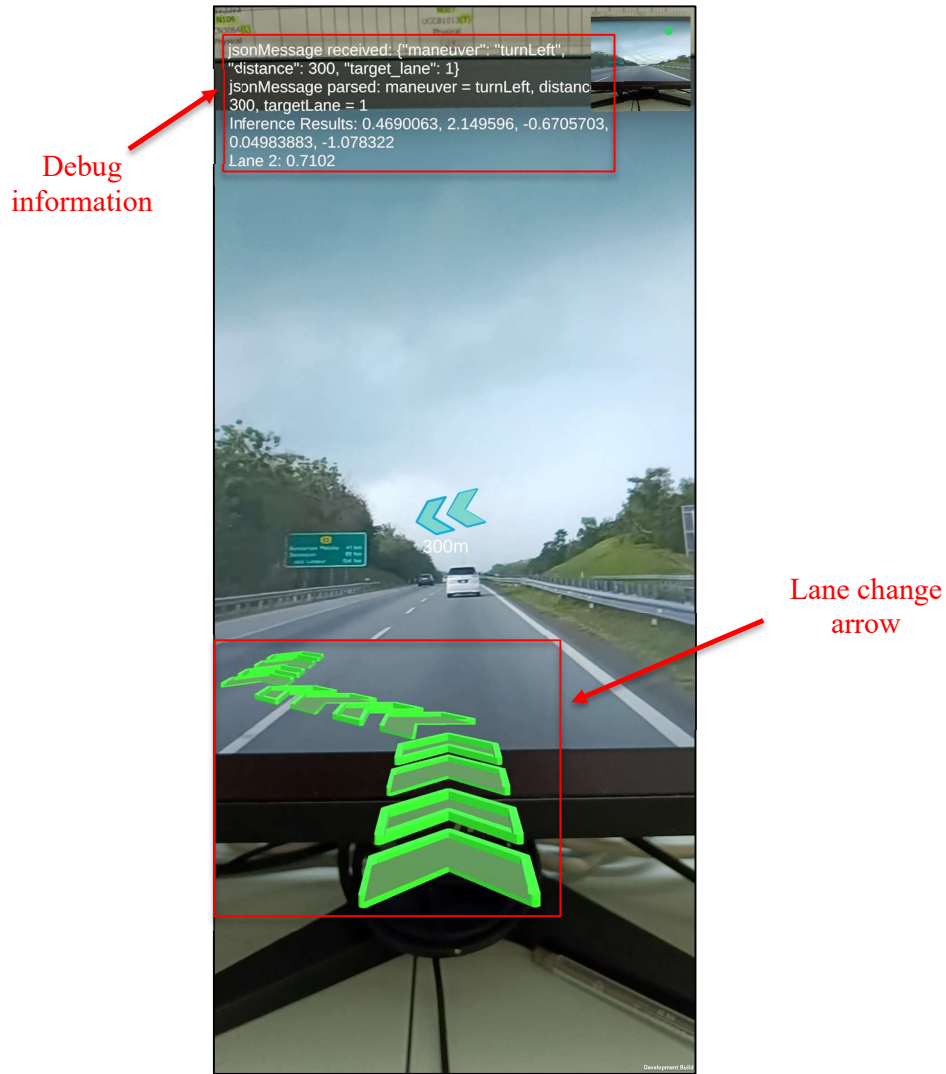


Figure 5.46: AR Navigation – Lane Change Screen

Figure 5.46 shows the AR Navigation – Lane Change Screen for the usage of the lane identification model on controlling the lane change arrow direction. It predicts that the vehicle is currently on lane number 2. The debug information shows that the targeted lane or direction is on the left, lane number 1. Hence, due to the vehicle currently in lane 2, it shows the lane change arrow towards the left. Note that the confidence level of the lane identification model is at least 50% to show the lane change arrow. This shows the feature of context-aware real-time assistance to reassure the user regarding staying on the correct lane.

5.4 Implementation Issues and Challenges

The implementation of this project faced the issue of the support in dependencies. Specifically, there were many conflicts in the compatibility of dependencies and lack of dependencies that support the functionality that the system required. The conflicts between dependencies took a long time to resolve due to extensive research on testing the compatibility between each of the dependencies. Additionally, there were no dependencies found for specific features of this project such as AR navigation and lane identification model. Hence, the features were implemented from scratch in this project.

Throughout the development of this project, another issue was the difficulty in testing the system. This project revolved around the practicality of AR and AI in navigation systems. Hence, it was crucial for the system to be able to achieve expectations to be a reliable system in the real world. This added an extra layer of difficulty to travel around physically for a real-world testing of the system.

5.5 Concluding Remarks

This project successfully implemented the system that used software tools such as Flutter, Unity, PyTorch, and TensorFlow. The system was implemented to integrate the technologies of mobile development, AR, and AI. The system implementation delivered a mobile AR navigation application with integrated AI. Specifically, the features of the system were modularised into 3 main modules which were Maps Module, Navigation Module, and Intelligence Module.

The dependencies for the implementation of the system were also listed to ensure reproducibility. This project documented the overview of the implementation process including the software installation and setup, together with its settings and configurations. The system operation was also shown and labeled to demonstrate the features of this project.

Chapter 6

System Evaluation and Discussion

This chapter discusses the system performance evaluation, system testing setup and results, project challenges, objectives evaluation, and concluding remarks.

6.1 System Performance Evaluation

System performance is an important aspect to ensure that the system achieves the user's expectations. It also indicates how well does a system performs in completing its tasks. System performance can be evaluated in many ways through defining the performance metrics. Hence, this project conducted a system performance evaluation on the proposed system that was based on performance metrics such as response time and accuracy.

6.1.1 Response Time

According to the statistics provided by [95], the average attention span of humans is 8.25 seconds. This indicates that most people can only maintain focus for a short period of time. Therefore, it is crucial for a system to be responsive in order for the users to be engaged and satisfied. If not, the user may become agitated or disinterested in the system. To evaluate the responsiveness of the proposed system, an experiment was conducted to measure the response time of each action. The duration of 8.25 seconds served as the benchmark response time. Table

The experiment was set up to obtain the average response time of each action. Actions such as application launching, destination information loading, search querying and AR navigation update were selected to be tested in the experiment due to its significance in the system. The average response time of each action was obtained out of 5 measurement attempts in seconds. Table 6.1 shows the average response time obtained for each action of the system.

Table 6.1: Average Response Time

Action	Response Time 1	Response Time 2	Response Time 3	Response Time 4	Response Time 5	Average Response Time (s)
Application Launching	4.42	3.72	3.60	4.13	3.15	3.80
Destination Information Loading	1.16	1.22	1.27	2.48	1.18	1.46
Search Querying	1.28	0.97	0.93	0.86	0.87	0.98
AR Navigation Update	1.00	0.77	1.03	1.03	0.96	0.96

Based on the results shown in Table 6.1, the average response time for application launching was 3.80 seconds, destination information loading was 1.46 seconds, search querying was 0.98 seconds, and AR navigation update was 0.96 seconds. The overall response time of the system was 1.80 seconds. Hence, the system was determined as good responsiveness due to its overall response time not exceeding 8.25 seconds.

6.1.2 Accuracy

Accuracy is an important performance metric to determine the correctness and reliability of an AI prediction model. Accuracy is the ratio of correct predictions over total predictions. Hence, the accuracy of the lane identification model was evaluated.

The dataset used for the lane identification model contained a total of 22,806 images that were Malaysian highway road images. There was a total of 5 classes, from Lane 1 to Lane 5. The dataset was split into 3 sets, specifically train (70%), validation (15%), and test sets (15%). After splitting the dataset, the train set consisted of 15,964 images, validation set consisted of 3,421 images, and test set consisted of 3,421 images. The accuracy of the lane identification model was evaluated on the test set.

The performance benchmark was the baseline performance, based on the Zero Rule assumption, assuming that the predictions are always the common class. The majority class of the dataset was Lane 2 with 11,174 images out of a total of 22,806 images. The equation for deriving the baseline performance is shown:

$$\text{Baseline Performance} = \frac{\text{Number of Majority Class}}{\text{Number of All Classes}}$$

The baseline performance for accuracy was 49%. Table 6.2 shows the accuracy comparison of the lane identification model compared to the baseline performance. It shows that the accuracy of the model was 99.21, being 50.21% higher than the baseline performance. This inferred that the model performed exceptionally well on identifying the lane numbers based on the front view of the road.

Table 6.2: Lane Identification Model Accuracy

Model	Baseline Performance	Accuracy
Lane Identification Model	49%	99.21%

6.2 System Testing Setup and Results

The testing of a system is important as it ensures that the system meets the system requirements such as functional and non-functional requirements. Test cases are used to test the system to make sure that each feature of the system is able to perform its intended purpose [96].

Each of the system's modules were evaluated based on several test cases. The test cases of modules such Maps Module, Navigation Module, and Intelligence Module were designed to test their respective features or use cases. Test cases such as search place, select destination, preview route, start maps navigation, and start AR navigation were evaluated for the system.

Each of the test cases contains several elements such as test case ID, test input, expected results, actual results, and status. Test case ID refers to the unique test case identification number while test input is the test data or action to the system, used for executing the test case. Expected results refer to the intended response by the system

while actual results refer to the actual response by the system. Status is indicated as “Fail” or “Pass” that depends on the expected results and actual results of a test case. A “Fail” status is when the actual results are not the same as expected results, while A “Pass” status is when the actual results are the same as expected results.

Each of the test cases contains several elements such as test case ID, test input, expected results, actual results, and status. Test case ID indicates the unique test case identification number while test input is the test data, action, or condition provided to the system. Expected results describe what the system should output or respond to. Actual results are the outputs of the test case in a real scenario. Status indicates either “Fail” or “Pass”, depending on the actual results to the expected results as the evaluation of the system based on the test case. Table 6.3 to Table 6.7 show the test cases of the system.

Table 6.3: Search Place Test Case

Search Place Test Case				
Module Name : Maps Module				
Test Description : To test the search place feature.				
Test Scenario : The user can search for places to select a destination.				
Test Case ID	Test Input	Expected Results	Actual Results	Status
1	Enter a search query.	The system shows the relevant search results with place name and address.	The system shows the relevant search results with place name and address.	Pass
2	Select destination based on search results.	The system redirects to the map and shows the selected search result as the selected destination with place name and address.	The system redirects to the map and shows the selected search result as the selected destination with place name and address.	Pass

Table 6.4: Select Destination Test Case

Select Destination Test Case				
Module Name : Maps Module				
Test Description : To test the select destination feature.				
Test Scenario : The user can tap on the map to select a destination.				
Test Case ID	Test Input	Expected Results	Actual Results	Status
1	Select destination by tapping on the map.	The system shows the selected destination with place name and address.	The system shows the selected destination with place name and address.	Pass
2	Cancel the selected destination.	The system removes the selected destination with its place name and address.	The system removes the selected destination with its place name and address.	Pass

Table 6.5: Preview Route Test Case

Preview Route Test Case				
Module Name : Maps Module				
Test Description : To test the preview route feature.				
Test Scenario : The user can preview the route to the selected destination.				
Test Case ID	Test Input	Expected Results	Actual Results	Status
1	Preview the route from to the destination.	The system shows the route from the current location to the destination with duration, distance, and ETA.	The system shows the route from the current location to the destination with duration, distance, and ETA.	Pass
2	Select the alternative route.	The system switches to the alternative route with the alternative	The system switches to the alternative route with the alternative	Pass

		duration, distance, and ETA.	duration, distance, and ETA.	
3	Cancel the route of the selected destination.	The system removes the route of the selected destination with the duration, distance, and ETA.	The system removes the route of the selected destination with the duration, distance, and ETA.	Pass

Table 6.6: Start Maps Navigation Test Case

Start Maps Navigation Test Case				
Module Name : Maps Module, Navigation Module				
Test Description : To test the start maps navigation feature.				
Test Scenario : The user can start a navigation session to the selected destination.				
Test Case ID	Test Input	Expected Results	Actual Results	Status
1	Start the navigation session.	The system shows the maps navigation UI with its directions, journey, and speed.	The system shows the maps navigation UI with its directions, journey, and speed.	Pass
2	Cancel the navigation session.	The system cancels the navigation session and redirects to the map.	The system cancels the navigation session and redirects to the map.	Pass

Table 6.7: Start AR Navigation Test Case

Start AR Navigation Test Case				
Module Name : Navigation Module, Intelligence Module				
Test Description : To test the start AR navigation feature.				
Test Scenario : The user can switch between maps and AR navigation mode.				
Test Case ID	Test Input	Expected Results	Actual Results	Status
1	Switch to AR navigation mode.	The system switches to the AR navigation UI with its AR	The system switches to the AR navigation UI with its AR	Pass

		directions of blue arrow, yellow arrow, or red arrow depending on the current direction.	directions of blue arrow, yellow arrow, or red arrow depending on the current direction.	
2	Target lane from navigation information.	The system detects whether the user is on the target lane through the lane identification prediction. If no, the lane change AR direction is shown. If yes, the system resumes.	The system detects whether the user is on the target lane through the lane identification prediction. If no, the lane change AR direction is shown. If yes, the system resumes.	Pass
3	Cancel the navigation session.	The system cancels the navigation session and redirects to the map.	The system cancels the navigation session and redirects to the map.	Pass

6.3 Project Challenges

The challenge faced by the project was the hardware limitations of mobile devices. The small and portable size of mobile devices have made it possible for convenient ease of use. However, it also limited the capabilities of mobile devices in terms of processing power and battery life. One of the major challenges faced by the hardware limitations was the battery life. To sustain the AR navigation and lane identification features, the mobile device has to perform heavy computation that contributes to high battery consumption. This makes the mobile device to drain out of battery quickly. Hence, a power supply was always needed to run the system.

Another challenge faced by the project was the budget allocated. This project depended on the Google Maps API service to provide its maps and navigation functionalities. However, Google Maps API only provided free trial periods of up to 3 months. If any services are being used beyond the free trial period, there is the requirement of paying for the services. Therefore, this posed a challenge for the project as it involved the budget throughout the development of the system.

The scale of this project was large as it included technologies such as mobile development, AR, and AI. These are large technologies that require extensive support for them to integrate into 1 system. However, as mentioned during the implementation of this project, there was a lack of dependencies support. Hence, the major features of this project were required to be implemented from scratch, leading to difficulties in managing the system.

The timeline of this project across Project I and Project II spanned across an approximate total of 5 months. As a result, the system implementation of this project was rushed to completion. Despite the lack of time, this project managed to deliver a system that achieved the defined objectives given its scale and complexity.

6.4 Objectives Evaluation

- **To Develop a Mobile AR Navigation Application.**

This project successfully developed a mobile navigation application that had an integrated AR navigation feature. The navigation feature included AR rendered directions such as left turn, right turn and lane change. The AR navigation provided better clarity in providing directions. Additionally, the conventional navigation system features such as place search, destination selection, route preview, and maps navigation were also included. Confusion and uncertainty of navigation sessions were reduced. Therefore, this project achieved the objective of developing a mobile AR navigation application.

- **To Increase the Effectiveness of AR Navigation through AI Integration.**

AI integration into this project led to an increased effectiveness of AR navigation. This provided context-aware AR directions that were able to monitor and detect road conditions through the lane identification model with an accuracy of 99.21%. This reassured the driver of staying in the correct lane according to the navigation information. With the ability to detect lane numbers, it increased the effectiveness of AR navigation by showing relevant AR directions during lane-level guidance, such as exits, merges, and complex intersections.

- **To Improve the UX.**

This project managed to improve the UX through a better design of providing clear directions. The integration of AR navigation enhances clarity that reduced confusion and uncertainty during navigation sessions, ultimately reducing the likelihood of a wrong turn. Additionally, the system provided a minimalistic UI design and good responsiveness to further enhance the UX. The minimalistic UI prevented information overloading on a single screen while maintaining a good overall response time for each action.

6.5 Concluding Remarks

The system was able to provide an overall response time of under 8.25 seconds, which is the attention span of humans, indicating the responsiveness of the system. Additionally, the lane identification model managed to achieve an accuracy of 99.21% compared to the benchmark performance of 49%, outperforming the expected performance.

System testing was conducted through test cases of features. The test cases involved the modules such as Maps Module, Navigation Module, and Intelligence Module. All the test cases were passed, indicating that the system was functional in terms of its features.

Despite the challenges faced such as hardware limitations, budget, project scale, and time, this project managed to fulfill its intended objectives through the proposed system.

Chapter 7

Conclusion and Recommendations

This chapter summarises the conclusion and recommendations.

7.1 Conclusion

This project successfully developed the AR and AI-based mobile navigation application as a proposed solution to the existing problems of navigation systems. It was implemented with features that provided a seamless driving experience for navigation journeys. The system was developed using software tools such as Flutter, Unity, PyTorch, and TensorFlow. The system was developed in modules, specifically Maps Module, Navigation Module, and Intelligent Module. Exclusive features of the system were AR navigation on a global scale and lane identification model with an accuracy of 99.21%. Through the implementation of the proposed system, it had achieved the outlined objectives that were defined in Chapter 1.

With the adoption of the agile XP methodology mentioned in Chapter 3, the system was developed quickly and iteratively through the addition of features in system releases. This methodology enabled the system to be open and flexible to changing requirements. Additionally, Chapter 4 provided the system design diagrams such as system architecture, system flowcharts, storyboard, use case diagram, activity diagram, wireframe design, low-fidelity prototype, and high-fidelity prototype. These diagrams provide the necessary documentation of the system to ensure reproducibility.

Chapter 5 demonstrated the process of developing the system in detail, covering content from software installations to configurations and settings. It also showed and described the demonstrations of the system in operation. Moreover, it also explained the implementation challenges faced by this project such as a lack of supported dependencies, system testing, budget, project scale, and time. However, this project managed to successfully deliver the system that achieved the objectives despite the challenges.

The performance of the system was evaluated with performance metrics such as response time and accuracy to ensure it was met with user expectations. Several test cases were also carefully designed to extensively prepare the system of its intended features. An objective evaluation was conducted to justify that the system achieved the project objectives through the proposed solution to the existing problems in navigation systems.

In conclusion, this final year project managed to provide a comprehensive solution to the confusion and uncertainty problem of navigation systems. The proposed solution achieved the objectives of this project through integrating technologies of mobile development, AR and AI. Additionally, the proposed solution conducted system performance evaluation and system testing to ensure that the system was able to perform well on its intended features. Ultimately, the proposed solution served as an alternative solution for drivers to overcome the issues of confusion and uncertainty of navigation systems to navigate complex geographical contexts with improved clarity for a seamless driving experience.

7.2 Recommendations

Despite that, the system has potential for improvements in various aspects, specifically the AR and AI features. Below are some recommendations for improving the system.

- **Improve AR Directions**

This project had an assumption that the front view of the road was captured at all times for the system. This ensured that the AR directions were correctly rendered to convey the AR directions. However, improvements can be made to the AR navigation feature through various aspects such as the AR placement, AR geolocation, and more guidance. AR placement refers to the AR directions being rendered on the intersection or lane of the road. This further improves the clarity of AR to the driver's point of view. AR geolocation refers to the rendering of AR directions only when the driver is in the visible proximity of the upcoming turn. This provides increased efficiency and relevancy of AR directions. More guidance refers to adding more AR directions for maneuvers such as roundabouts, and U-turns.

- **Improve AI Real-Time Assistance**

The lane identification model of this project achieved a good accuracy, but its feasibility assessment was not sufficient for real life applications. This included external factors such as varying weather, complex traffic scenarios, different road types, and various geographical environments. Therefore, the AI model can be improved through conducting rigorous testing on various real-life scenarios to improve the robustness and adaptability in various situations. Additionally, more types of real-time assistance in the form of monitoring and detection can be introduced and implemented in the system.

REFERENCES

- [1] National Geographic, “navigation | National Geographic Society,” [education.nationalgeographic.org](https://education.nationalgeographic.org/resource/navigation/), Oct. 19, 2023.
- [2] C. Manning, “GPS,” NASA, Sep. 25, 2023. <https://www.nasa.gov/directorates/somd/space-communications-navigation-program/gps/>
- [3] S. Evans, “1993 Eunos/Mazda Cosmo Classic Drive,” MotorTrend, Jan. 31, 2013. <https://www.motortrend.com/reviews/12q2-1993-eunos-mazda-cosmo-drive> (accessed May 05, 2025).
- [4] 삼성전자 뉴스룸 [Samsung Newsroom], “CES 2022: Showcasing the Future of In-Vehicle Experiences,” YouTube, Jan. 05, 2022. <https://www.youtube.com/watch?v=akZ8U6OcksQ> (accessed Aug. 29, 2024).
- [5] Durgesh Kekare, “Supervised Learning vs. Unsupervised Learning: Choosing the Right Approach,” Data Expertise, Dec. 09, 2023. <https://www.dataexpertise.in/5-differences-supervised-unsupervised-learning/> (accessed May 05, 2025).
- [6] G. Milner, Pinpoint: How GPS is Changing Technology, Culture, and Our Minds. W. W. Norton & Company, 2016.
- [7] A. Y. Lin, K. Kuehl, J. Schöning, and B. Hecht, “Understanding ‘Death by GPS,’” Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, May 2017, doi: <https://doi.org/10.1145/3025453.3025737>.
- [8] K. Kaplan, “What Is User Experience (and What Is It Not)?,” Nielsen Norman Group, Nov. 15, 2024. <https://www.nngroup.com/articles/what-is-user-experience/>
- [9] “Mobile Operating System Market Share Worldwide | Statcounter Global Stats,” StatCounter Global Stats, 2024. <https://gs.statcounter.com/os-market-share/mobile/worldwide/2024>
- [10] “Worldwide Smartphone Market Forecast to Grow 2.3% in 2025, Led by Android Growth in China and U.S., Amid 10% China Tariffs, According to IDC,” IDC: The premier global market intelligence company, 2025. <https://my.idc.com/getdoc.jsp?containerId=prAP53217825>

- [11] “History of Android,” GeeksforGeeks, Nov. 07, 2020. <https://www.geeksforgeeks.org/history-of-android/>
- [12] “Brand guidelines | Google Play,” Android Developers. <https://developer.android.com/distribute/marketing-tools/brand-guidelines>
- [13] E. Lee, “Android users lose \$2.4 million to malware scams that use Facebook and TikTok ads as bait,” The Straits Times, Apr. 17, 2025. <https://www.straitstimes.com/singapore/android-users-lose-2-4m-to-malware-scams-that-use-facebook-and-tiktok-ads-as-bait> (accessed May 06, 2025).
- [14] K. Rafalski, “The Complete History of iOS,” www.netguru.com, Apr. 29, 2024. <https://www.netguru.com/blog/ios-history>
- [15] “Build for iOS 18,” Apple Developer. <https://developer.apple.com/ios/>
- [16] “Android Studio: An IDE built for Android,” Android Developers Blog. <https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html>
- [17] X (formerly Twitter), 2025. <https://x.com/AndroidDev/status/1658215682788433931?lang=en> (accessed May 06, 2025).
- [18] “Gradle | Gradle Build Tool Features,” Gradle, Sep. 26, 2023. <https://gradle.org/features/>
- [19] “Java Branding and Licensing Guidelines.” Available: <https://www.oracle.com/a/ocom/docs/java-licensing-logo-guidelines-1908204.pdf>
- [20] “Kotlin brand assets | Kotlin,” Kotlin Help. <https://kotlinlang.org/docs/kotlin-brand-assets.html>
- [21] “Brand,” Develocity, Dec. 18, 2023. <https://gradle.com/brand/> (accessed May 06, 2025).
- [22] Apple, “Xcode - Apple Developer,” Apple.com, 2018. <https://developer.apple.com/xcode/>
- [23] “Swift Resources - Apple Developer,” Apple.com, 2020. <https://developer.apple.com/swift/resources/>
- [24] S. Scrivens, “Google announces Flutter 1.0, the first stable release of its cross-platform mobile development toolkit,” Android Police, Dec. 05, 2018. <https://www.androidpolice.com/2018/12/05/google-announces-flutter-1-0-the->

first-stable-release-of-its-cross-platform-mobile-development-toolkit/
(accessed May 06, 2025).

- [25] “Brand,” flutter.dev. <https://flutter.dev/brand>
- [26] Flutter, “Multi-Platform,” flutter.dev. <https://flutter.dev/multi-platform>
- [27] Flutter, “Flutter architectural overview,” docs.flutter.dev. <https://docs.flutter.dev/resources/architectural-overview>
- [28] GeeksforGeeks, “Flutter vs Native: Which is Best in 2025,” GeeksforGeeks, Dec. 29, 2023. <https://www.geeksforgeeks.org/flutter-vs-native/> (accessed May 06, 2025).
- [29] “Dart brand guidelines,” Dart.dev, 2024. <https://dart.dev/brand>
- [30] “Visual Studio Code and VS Code icons and names usage guidelines,” code.visualstudio.com. <https://code.visualstudio.com/brand>
- [31] facebook, “Release v0.1.0: Initial public release · facebook/react-native,” GitHub, Mar. 27, 2015. <https://github.com/facebook/react-native/releases/tag/v0.1.0> (accessed May 06, 2025).
- [32] React Native, “React Native · A framework for building native apps using React,” reactnative.dev, 2024. <https://reactnative.dev/>
- [33] “Flutter vs React Native: A Comparison,” BrowserStack. <https://www.browserstack.com/guide/flutter-vs-react-native>
- [34] W. of the Abyss, “Unity: Development History and the Influence of This Game Engine on the Game Development...,” Medium, Sep. 28, 2023. https://medium.com/@wota_mmorp/unity-development-history-and-the-influence-of-this-game-engine-on-the-game-development-36dc7a7a3b9d
- [35] “Guidelines for Using Unity Trademarks,” Unity. <https://unity.com/legal/branding-trademarks>
- [36] “What platforms are supported by Unity?,” Unity, Jun. 06, 2024. <https://support.unity.com/hc/en-us/articles/206336795-What-platforms-are-supported-by-Unity>
- [37] M. Dealessandri, “What is the best game engine: is Unity right for you?,” GamesIndustry.biz, Jan. 16, 2020. <https://www.gamesindustry.biz/what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you>
- [38] dotnet, “GitHub - dotnet/vscode-csharp: Official C# support for Visual Studio Code,” GitHub, Apr. 22, 2025. <https://github.com/dotnet/vscode-csharp/tree/main> (accessed May 06, 2025).

- [39] “AR Foundation | AR Foundation | 6.0.0-pre.5,” docs.unity3d.com. <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html>
- [40] AR Core, “ARCore - Google Developers | Google Developers,” Google Developers. <https://developers.google.com/ar>
- [41] Apple, “ARKit 3 - Augmented Reality - Apple Developer,” Apple Developer, 2020. <https://developer.apple.com/augmented-reality/arkit/>
- [42] K. T. Jensen, “25 Years Later: The History of Unreal and an Epic Dynasty,” PCMag, May 22, 2023. <https://www.pcmag.com/news/25-years-later-the-history-of-unreal-and-an-epic-dynasty>
- [43] Epic Games, “Unreal Engine Branding Guidelines and Trademark Usage,” Unrealengine.com, 2019. <https://www.unrealengine.com/en-US/branding>
- [44] ashutosh, “How Unreal Engine Supports Cross-Platform Game Development,” SDLC Corp, Sep. 27, 2024. <https://sdlccorp.com/post/building-vr-experiences-with-unreal-engine-a-developers-guide> (accessed May 06, 2025).
- [45] “Terms of Use : Standard C++,” isocpp.org. <https://isocpp.org/home/terms-of-use>
- [46] M. Klein, “Why Unreal Engine 5 is the Top Choice for Both Indie and AAA Studios,” SDLC Corp, Apr. 16, 2025. <https://sdlccorp.com/post/unreal-engine-5-top-choice-for-indie-and-aaa-studios> (accessed May 06, 2025).
- [47] “Top Mistakes Unreal Engine Beginners Make - Vagon,” Vagon.io, 2024. <https://vagon.io/blog/top-mistakes-unreal-engine-beginners-make> (accessed May 06, 2025).
- [48] IBM, “PyTorch,” Ibm.com, Oct. 04, 2023. <https://www.ibm.com/think/topics/pytorch>
- [49] “Brand Guidelines PyTorch Brand Guidelines.” Accessed: May 06, 2025. [Online]. Available: <https://pytorch.org/assets/brand-guidelines/PyTorch-Brand-Guidelines.pdf>
- [50] “C++ — PyTorch 2.6 documentation,” Pytorch.org, 2023. https://pytorch.org/docs/stable/cpp_index.html
- [51] “PyTorch at Tesla - Andrej Karpathy, Tesla,” www.youtube.com. <https://www.youtube.com/watch?v=oBkltKXtDE> (accessed May 26, 2021).
- [52] “PyTorch,” www.pytorch.org. <https://pytorch.org/get-started/locally/>

- [53] Python Software Foundation, “The Python Logo,” Python.org, 2019. <https://www.python.org/community/logos/>
- [54] jupyter, “GitHub - jupyter/design: Design related materials for Project Jupyter,” GitHub, 2015. <https://github.com/jupyter/design/tree/main> (accessed May 06, 2025).
- [55] “Models and pre-trained weights — Torchvision main documentation,” pytorch.org. <https://pytorch.org/vision/main/models.html>
- [56] N. Klingler, “EfficientNet: Pushing the Boundaries of Deep Learning Efficiency,” viso.ai, Mar. 18, 2024. <https://viso.ai/deep-learning/efficientnet/>
- [57] T. Ahmed and Noor, “Classification and understanding of cloud structures via satellite images with EfficientUNet,” arXiv (Cornell University), Jul. 2021, doi: <https://doi.org/10.1002/essoar.10507423.1>.
- [58] “Towards ML Engineering: A Brief History Of TensorFlow Extended (TFX).” <https://blog.tensorflow.org/2020/09/brief-history-of-tensorflow-extended-tfx.html>
- [59] “Brand Guidelines.” Available: https://www.tensorflow.org/extras/tensorflow_brand_guidelines.pdf
- [60] TensorFlow, “API documentation | tensorflow core v2.4.1,” TensorFlow, Sep. 30, 2024. https://www.tensorflow.org/api_docs
- [61] TensorFlow, “TensorFlow 2.0 is now available! - TensorFlow - Medium,” Medium, Sep. 30, 2019. <https://medium.com/tensorflow/tensorflow-2-0-is-now-available-57d706c2a9ab> (accessed May 06, 2025).
- [62] Google, “LiteRT overview,” Google AI for Developers, 2024. <https://ai.google.dev/edge/litert>
- [63] “Install TensorFlow with pip,” TensorFlow, 2021. <https://www.tensorflow.org/install/pip#macos> (accessed May 06, 2025).
- [64] Keras, “Home - Keras Documentation,” Keras.io, 2019. <https://keras.io>
- [65] “Google Maps,” Archive.org, 2024. <https://web.archive.org/web/20061016100649/http://www.google.com:80/gmm/index.html>
- [66] J. Lindner, “Google Maps Statistics: Latest Data & Summary - Wi-FiTalents,” wifitalents.com, Aug. 06, 2024. <https://wifitalents.com/statistic/google-maps/>
- [67] Maps - Navigate & Explore, “Maps - Navigate & Explore - Apps on Google Play,” Google.com, 2009.

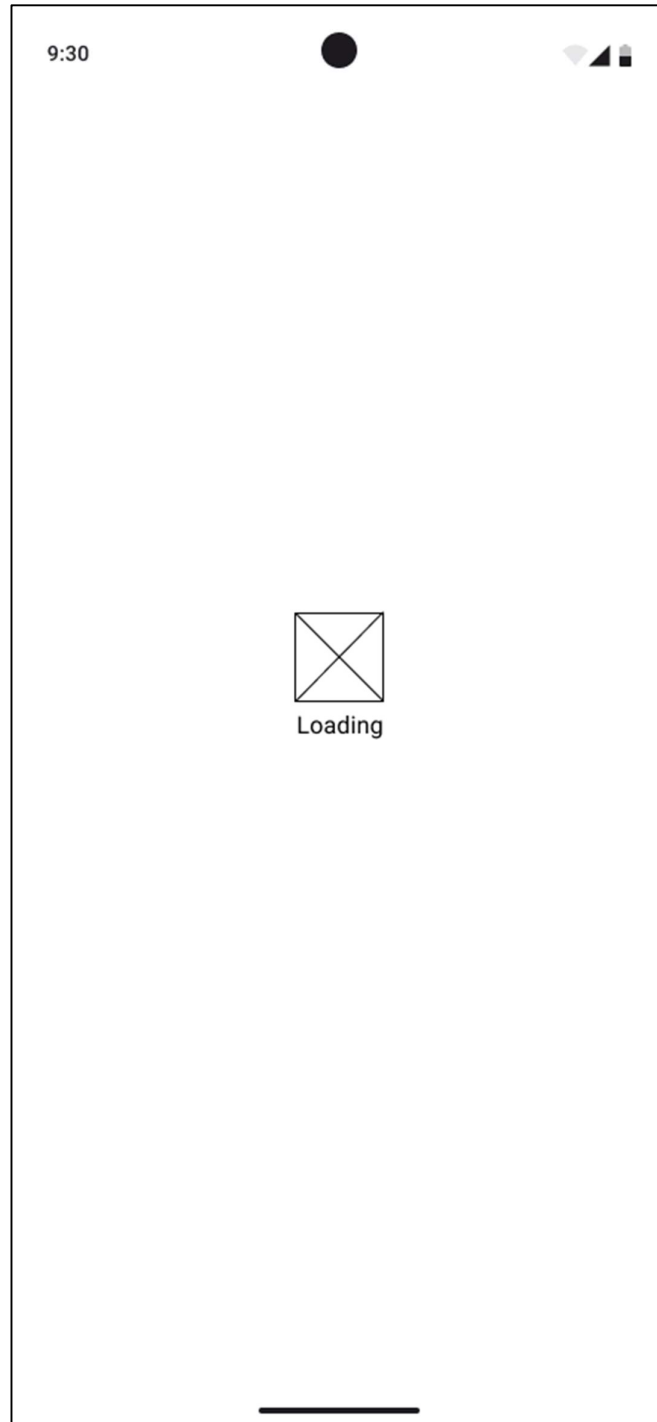
- <https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=en>
- [68] M. Ranieri, “A new sense of direction with Live View,” Google, Oct. 01, 2020. <https://blog.google/products/maps/new-sense-direction-live-view/>
 - [69] “How Google Maps Gets Its Remarkably Accurate Real-Time Traffic Data,” Gadgets 360, Mar. 02, 2017. <https://www.gadgets360.com/apps/features/how-google-maps-gets-its-remarkably-accurate-real-time-traffic-data-1665385>
 - [70] D. Rushe, “Google buys Waze map app for \$1.3bn,” the Guardian, Jun. 11, 2013. <https://www.theguardian.com/technology/2013/jun/11/google-buys-waze-maps-billion>
 - [71] Unlimited Exposure Online, “Do more people use Waze, Apple Maps or Google Maps?,” Medium, Apr. 2024. https://medium.com/@shareknowledge_41127/do-more-people-use-waze-apple-maps-or-google-maps-3239b26a6396
 - [72] “Waze Navigation & Live Traffic - Apps on Google Play,” play.google.com. <https://play.google.com/store/apps/details?id=com.waze&hl=en>
 - [73] V. Page, “Waze: The Pros and Cons,” Investopedia, 2019. <https://www.investopedia.com/articles/investing/060415/pros-cons-waze.asp>
 - [74] “Sygic Sygic GPS Navigation Adds a Cloud-Based Solution for Quickly Detecting and Warning You of Wrong-Way Drivers,” Sygic.com, 2022. <https://www.sygic.com/press/sygic-gps-navigation-adds-a-cloud-based-solution-for-quickly-detecting-and-warning-you-of-wrong-way-drivers> (accessed May 06, 2025).
 - [75] Sygic, “Sygic GPS Navigation & Maps,” Google.com, 2021. <https://play.google.com/store/apps/details?id=com.sygic.aura&hl=en>
 - [76] “Sygic Sucks (and so does Google) – Stuff Yaron Finds Interesting,” Goland.org, Jun. 03, 2014. <https://www.goland.org/sygic/>
 - [77] Sygic a.s., “Real View Navigation from Sygic,” YouTube, Jun. 28, 2017. <https://www.youtube.com/watch?v=yzQCCOAEgqY>
 - [78] Sygic a.s., “Traffic Sign recognition (No overtaking) in Sygic GPS Navigation & Maps,” YouTube, Nov. 21, 2022. <https://www.youtube.com/watch?v=fvrWOhtceyw> (accessed May 06, 2025).
 - [79] “Sygic Traffic Sign recognition,” www.sygic.com. <https://www.sygic.com/what-is/sign-recognition>

- [80] “Mercedes Benz Augmented Reality Navigation,” [www.youtube.com](https://www.youtube.com/watch?v=IU_JLd2C-xM).
https://www.youtube.com/watch?v=IU_JLd2C-xM
- [81] Admin, “Mercedes-Benz of Arrowhead,” Mercedes-Benz of Arrowhead, Apr. 23, 2024. <https://www.arrowheadmb.com/blog/exploring-the-mbux-navigation-system/>
- [82] K. F. Bram-Larbi, Vassilis Charissis, S. Khan, Ramesh Lagoo, D. G. Harrison, and Dimitris Drikakis, “Intelligent Collision Avoidance and Manoeuvring System with the Use of Augmented Reality and Artificial Intelligence,” pp. 457–469, Jan. 2021, doi: https://doi.org/10.1007/978-3-030-73100-7_32.
- [83] Wonderful Road Trip TV, “Driving In Malaysia Highway | Ipoh To Kuala Lumpur MY 🚗 📁 👍,” YouTube, Aug. 17, 2024. <https://www.youtube.com/watch?v=LjzDSIWDXU> (accessed May 06, 2025).
- [84] Android Studio, “Android Studio and SDK tools,” Android Developers, 2019. <https://developer.android.com/studio>
- [85] Microsoft, “Visual Studio Code,” Visualstudio.com, 2024. <https://code.visualstudio.com/>
- [86] “Start building Flutter Android apps on Windows,” docs.flutter.dev. <https://docs.flutter.dev/get-started/install/windows/mobile>
- [87] U. Technologies, “Get Started with Unity - Download the Unity Hub & Install the Editor,” unity.com. <https://unity.com/download>
- [88] “CUDA Toolkit 12.4 Downloads,” NVIDIA Developer, 2025. https://developer.nvidia.com/cuda-12-4-0-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exe_network (accessed May 07, 2025).
- [89] “cuDNN Archive,” NVIDIA Developer, 2023. <https://developer.nvidia.com/rdp/cudnn-archive> (accessed May 07, 2025).
- [90] “Python Release Python 3.11.9,” Python.org. <https://www.python.org/downloads/release/python-3119/>
- [91] juicycleff, “GitHub - juicycleff/flutter-unity-view-widget: Embeddable unity game engine view for Flutter. Advance demo here <https://github.com/juicycleff/flutter-unity-arkit-demo>,” GitHub, 2019.

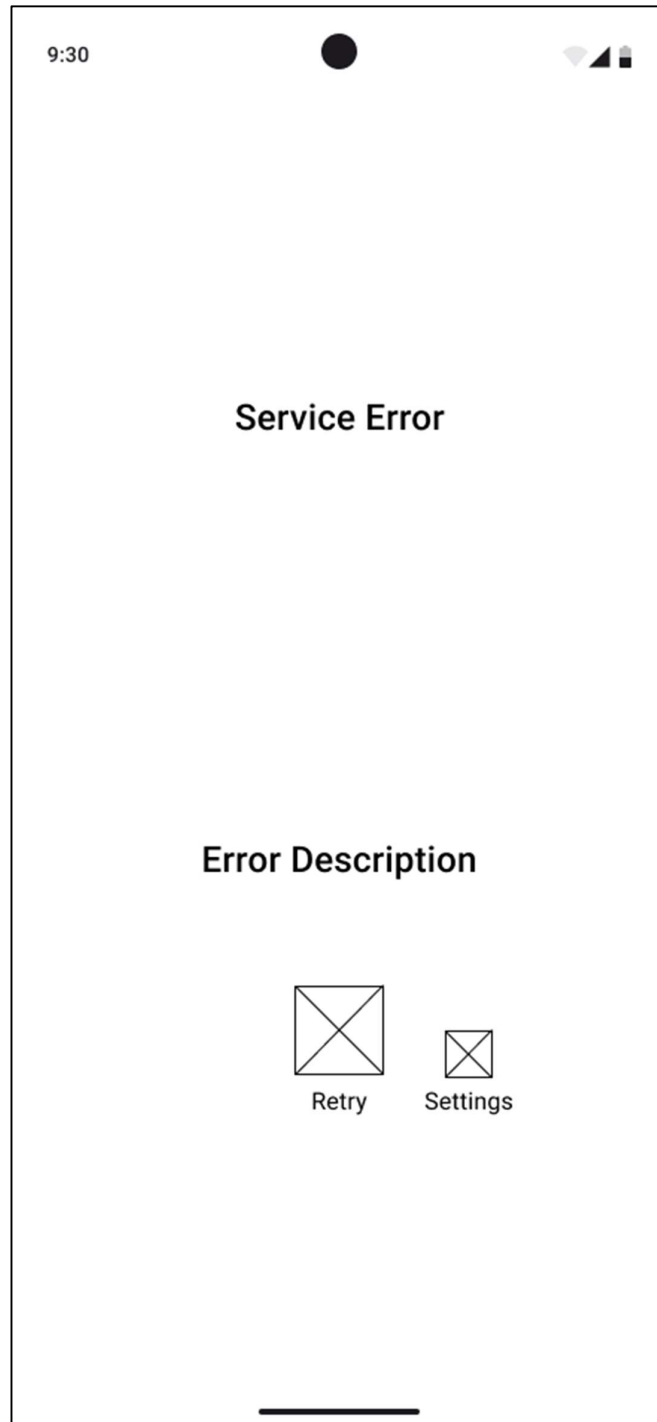
- <https://github.com/juicycleff/flutter-unity-view-widget/tree/master> (accessed May 07, 2025).
- [92] “Google Cloud console - Web UI Admin,” Google Cloud. <https://cloud.google.com/cloud-console>
- [93] “AR + GPS Location | Integration | Unity Asset Store,” @UnityAssetStore, 2021. <https://assetstore.unity.com/packages/tools/integration/ar-gps-location-134882?srsId=AfmBOoqYptyfBCoFuuYTQgXzTRr-> (accessed May 07, 2025).
- [94] asus4, “GitHub - asus4/tf-lite-unity-sample: TensorFlow Lite Samples on Unity,” GitHub, Oct. 17, 2024. <https://github.com/asus4/tf-lite-unity-sample> (accessed May 07, 2025).
- [95] Samba Recovery, “Average Human Attention Span Statistics & Facts [2024] | Samba Recovery,” www.sambarecovery.com, Jun. 25, 2024. <https://www.sambarecovery.com/rehab-blog/average-human-attention-span-statistics>
- [96] GeeksforGeeks, “How to write Test Cases - Software Testing,” GeeksforGeeks, Apr. 25, 2022. <https://www.geeksforgeeks.org/test-case/>

APPENDICES

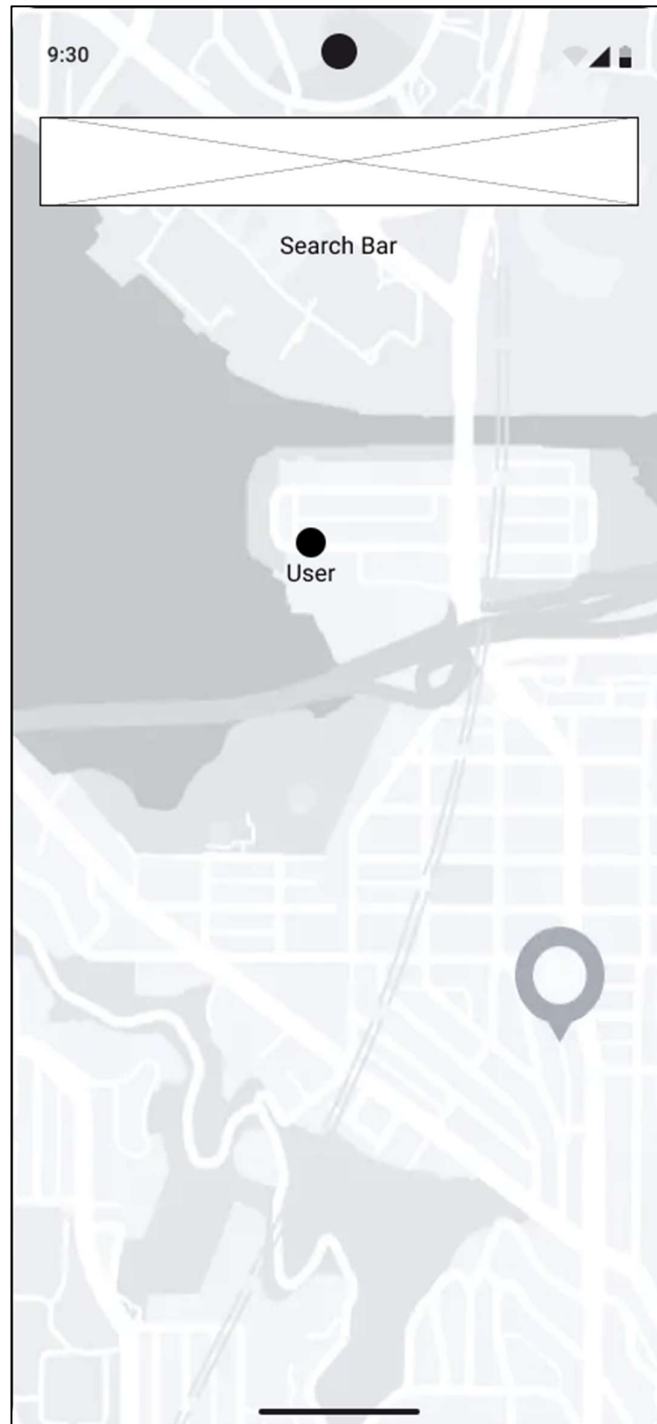
Appendix A: Wireframe Design



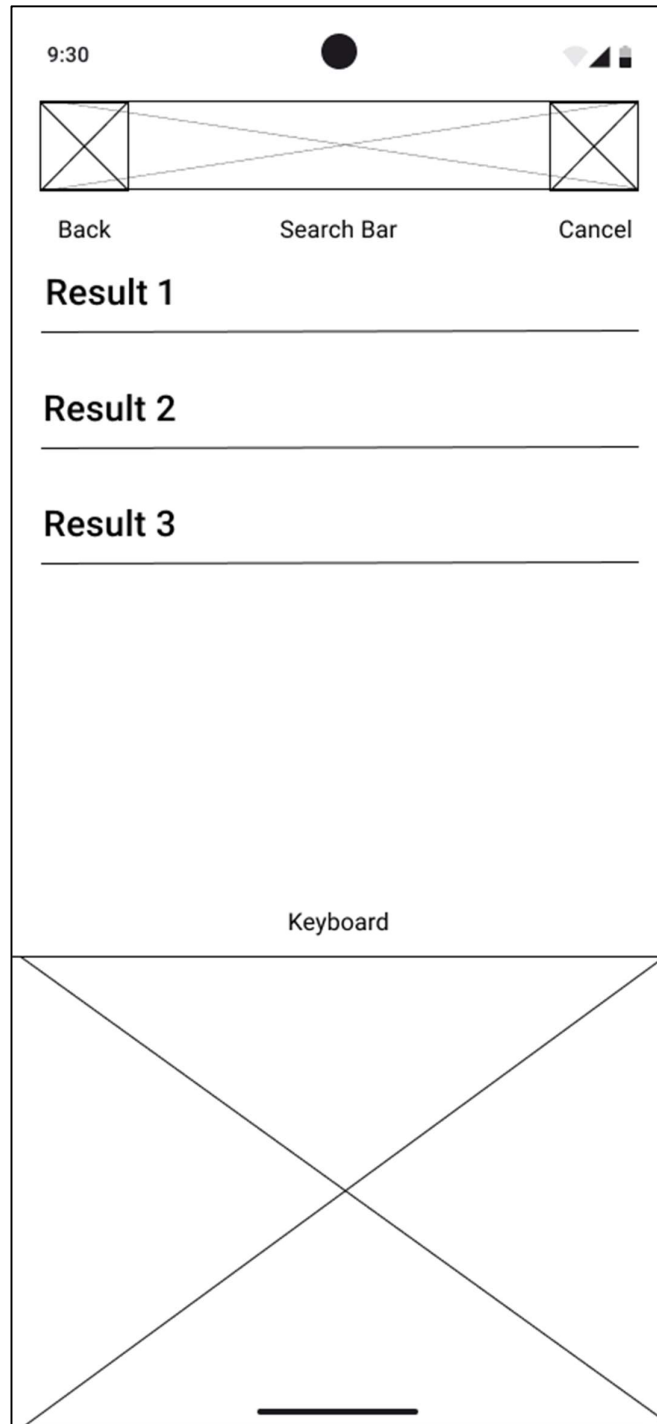
Loading Screen



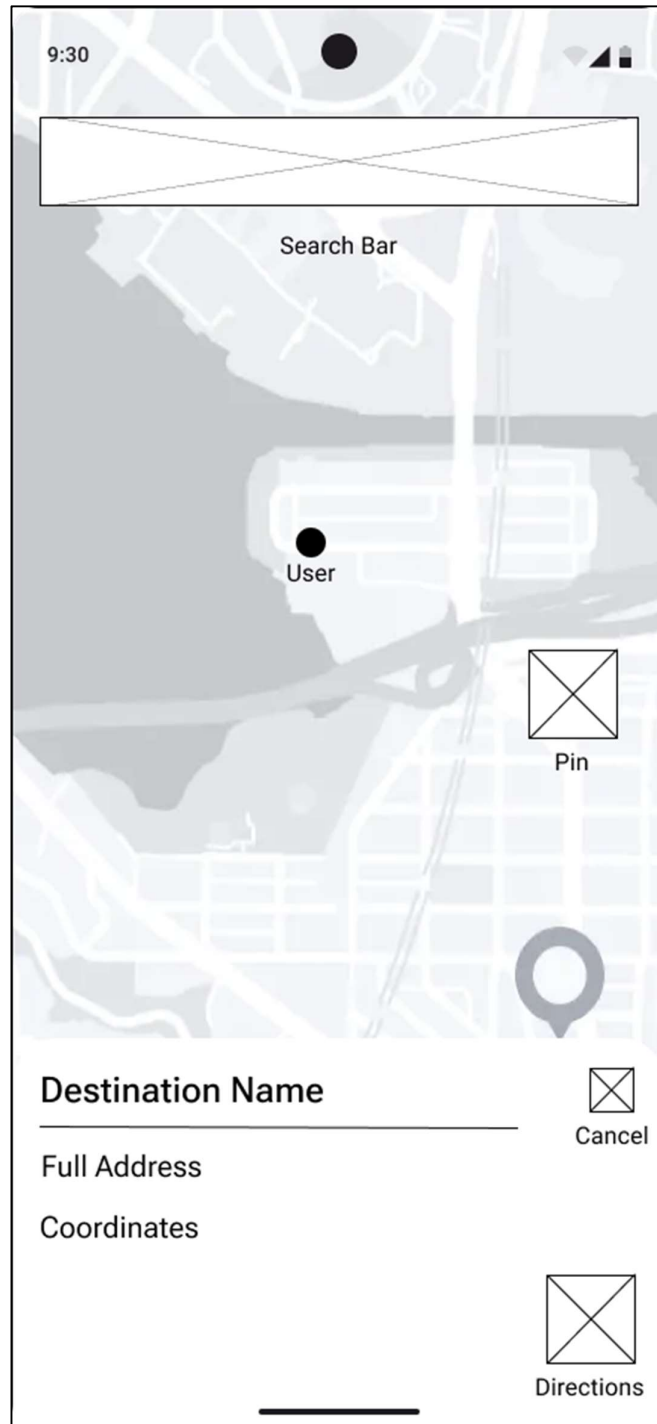
Loading – Failed Screen



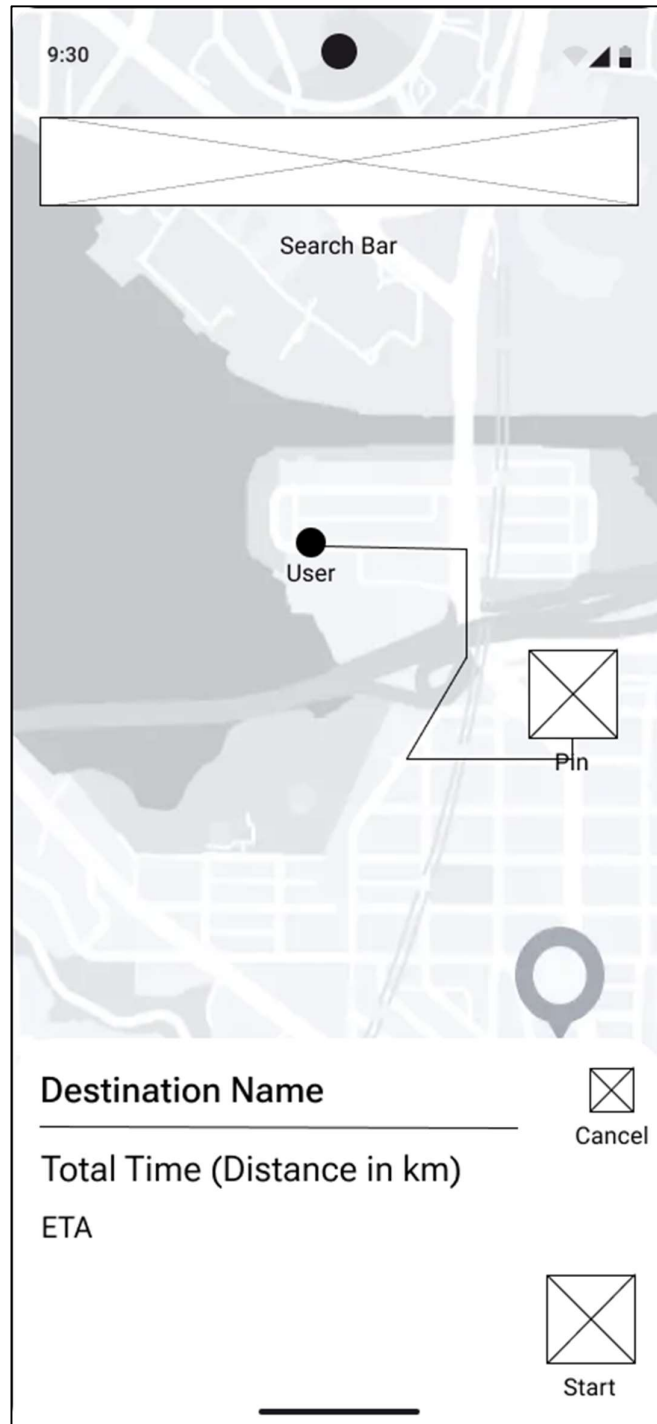
Maps Screen



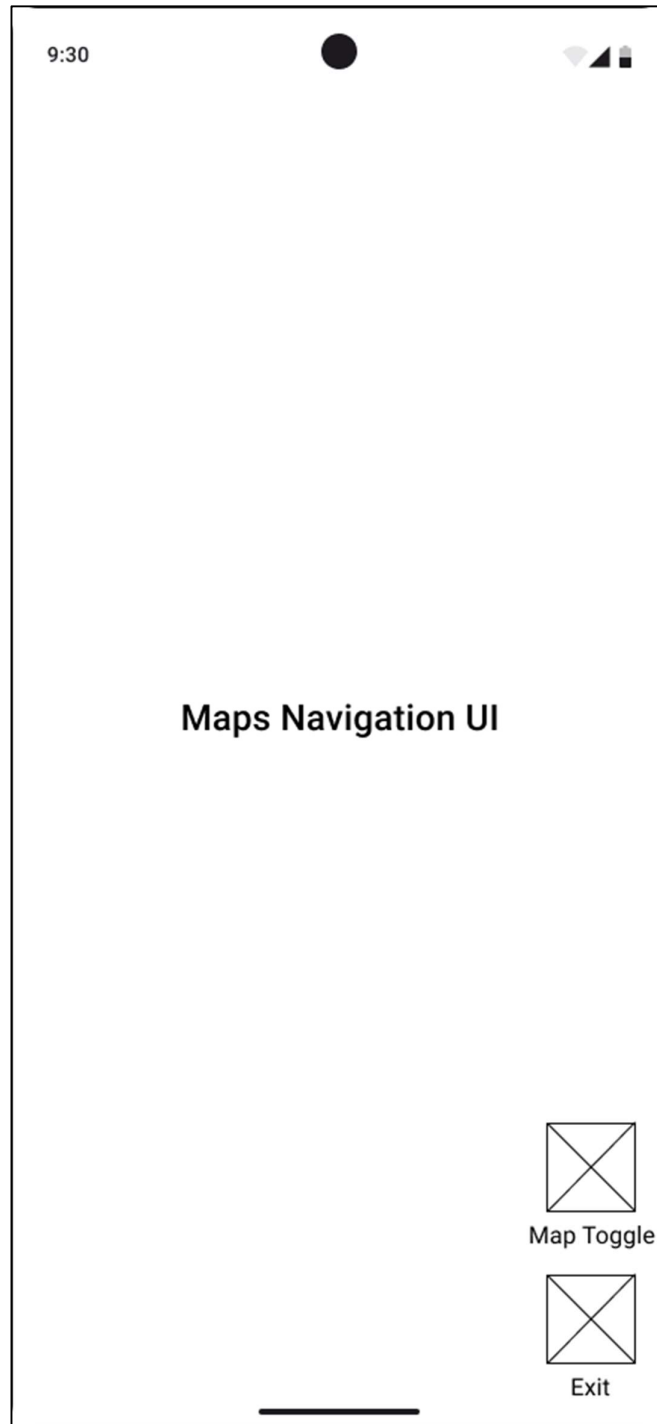
Maps – Search Screen



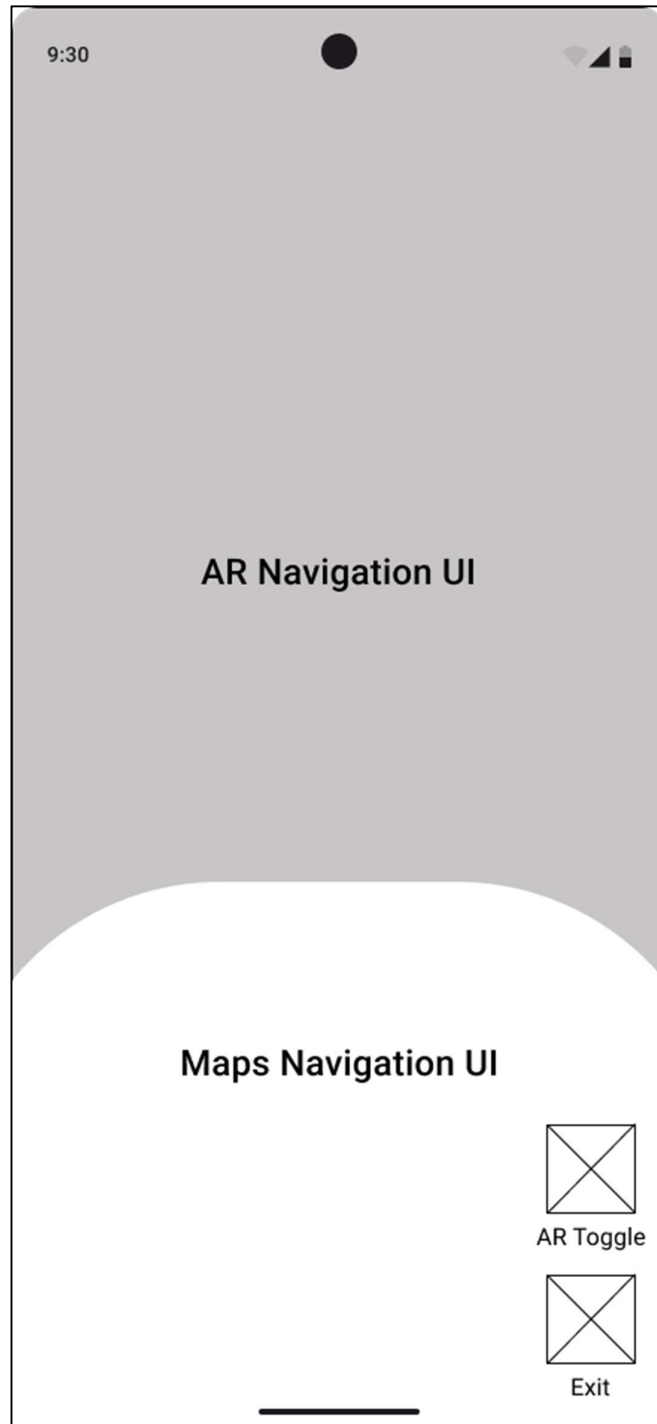
Maps – Selected Screen



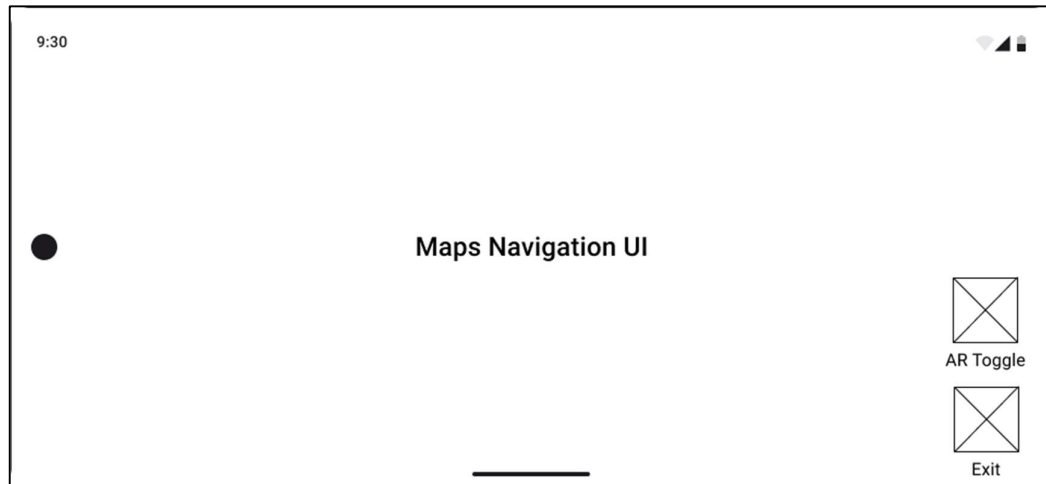
Maps – Directions Screen



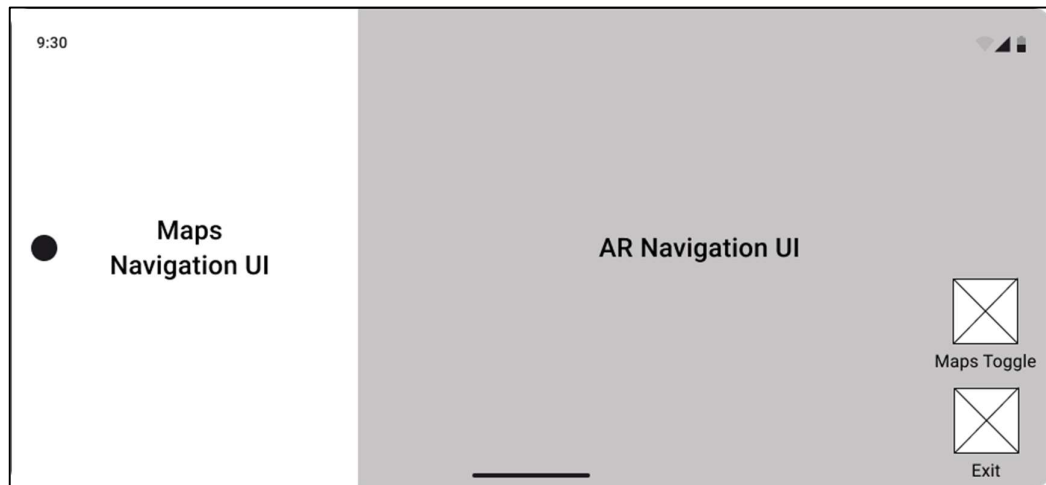
Maps Navigation Screen



AR Navigation Screen

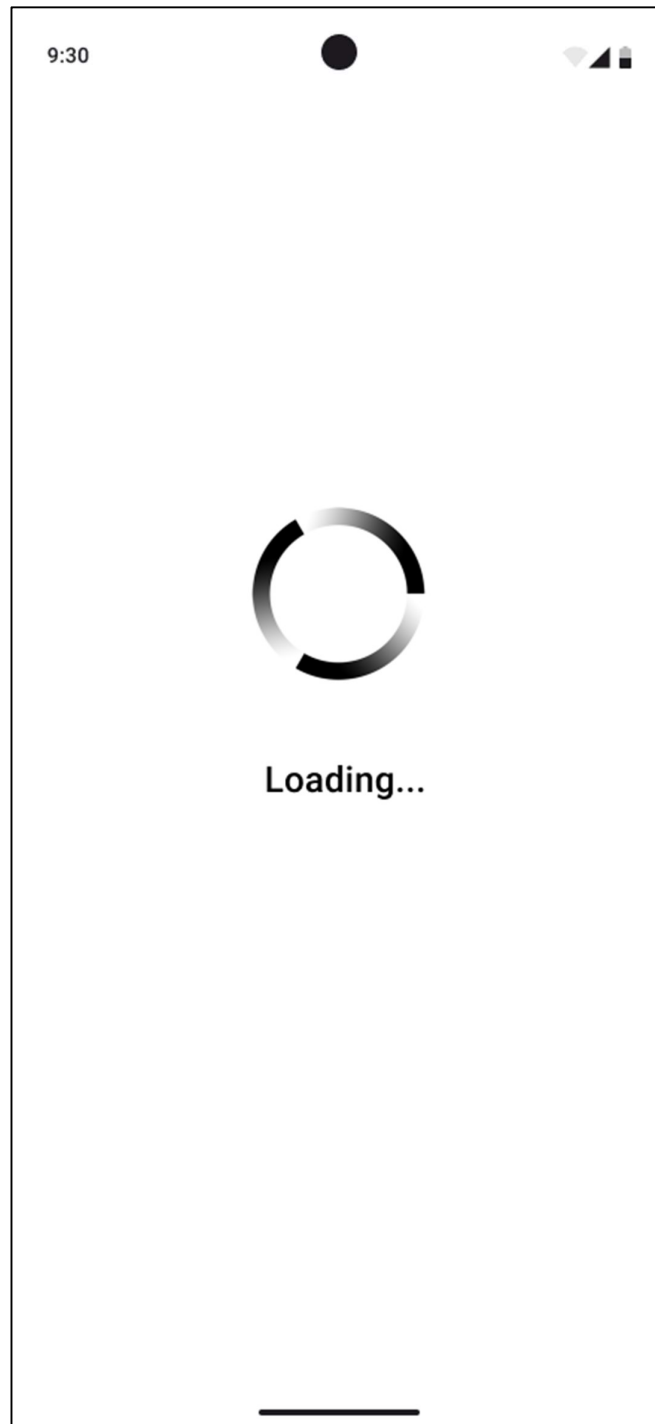


Maps Navigation – Horizontal Screen

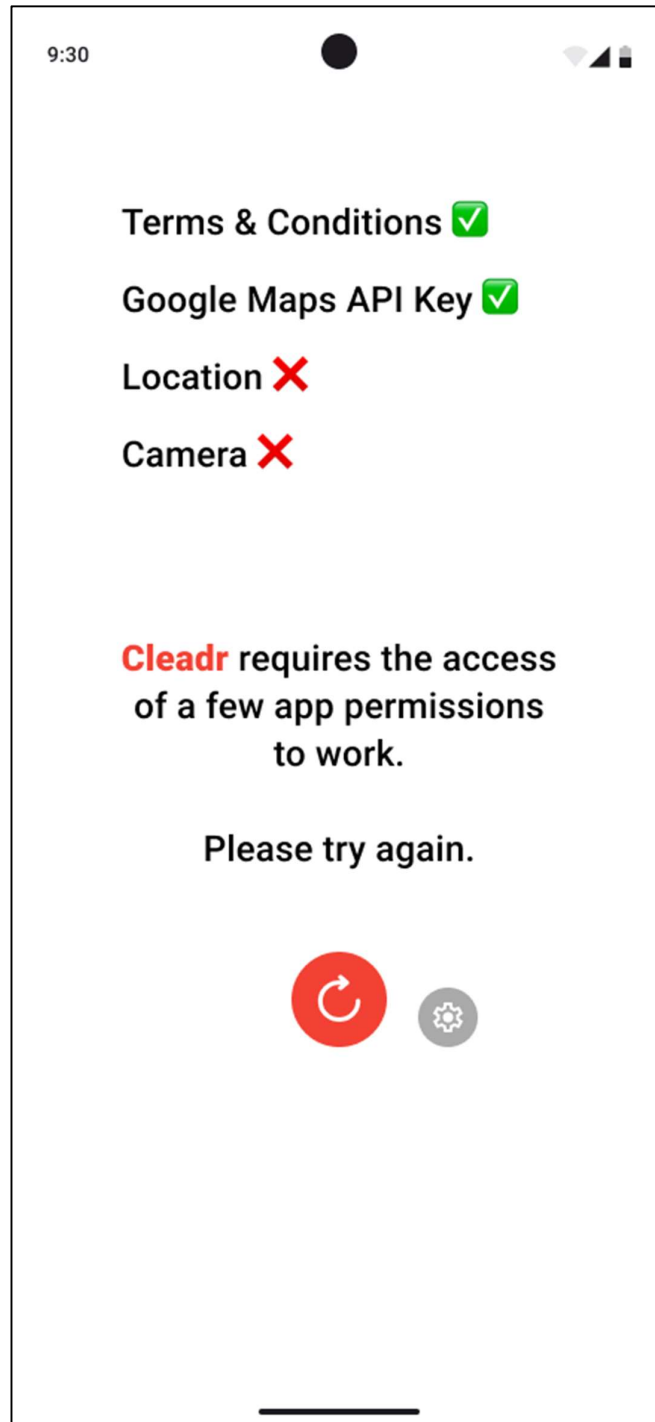


AR Navigation – Horizontal Screen

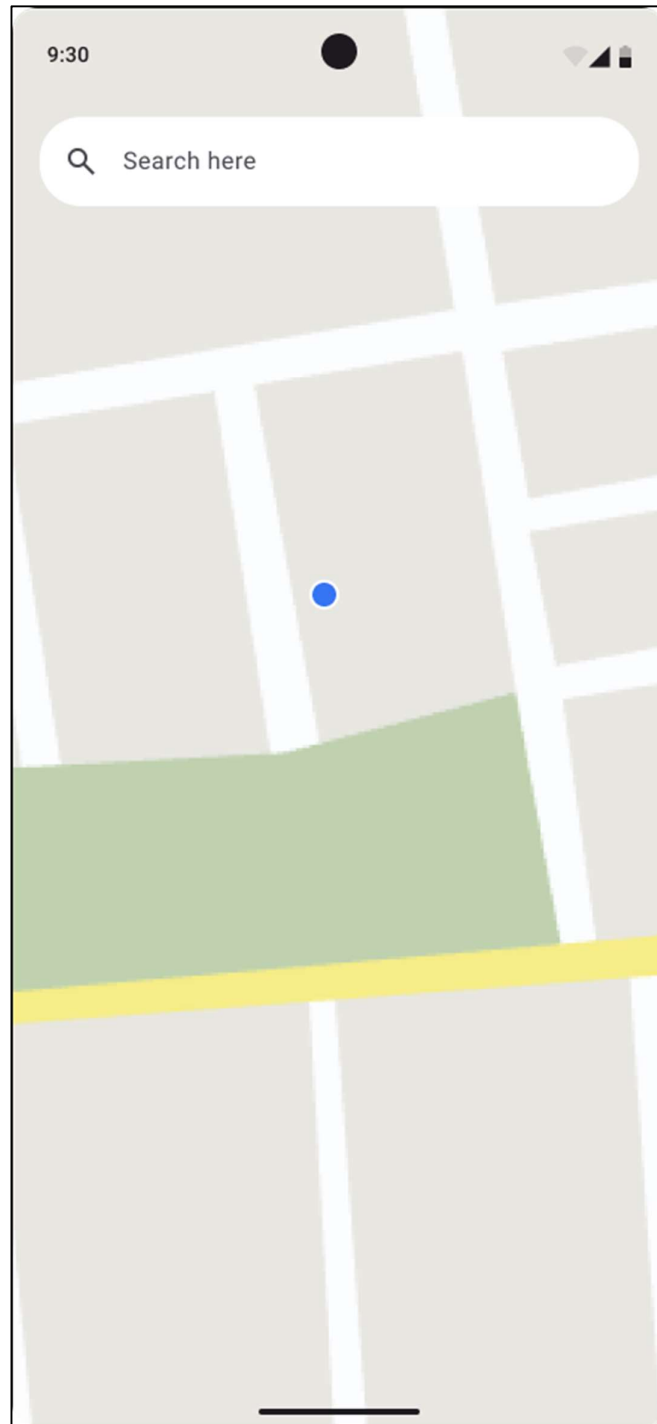
Appendix B: Low-Fidelity Design



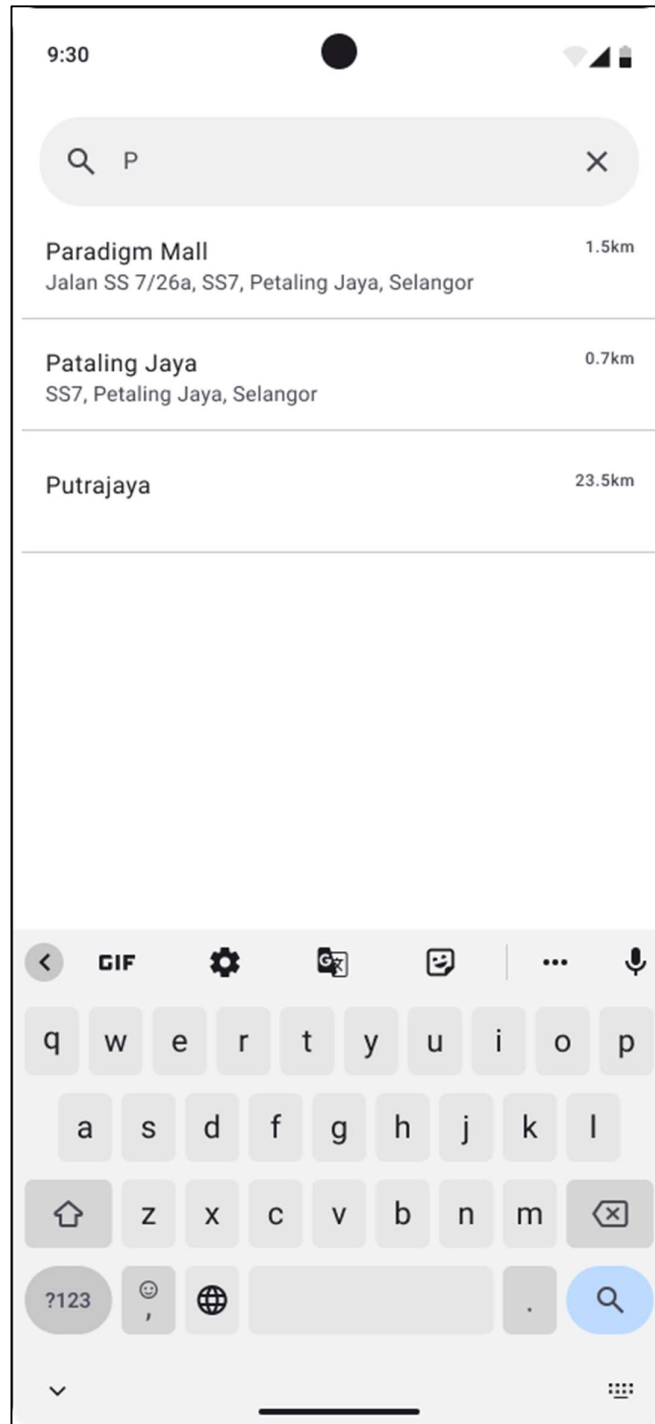
Loading Screen



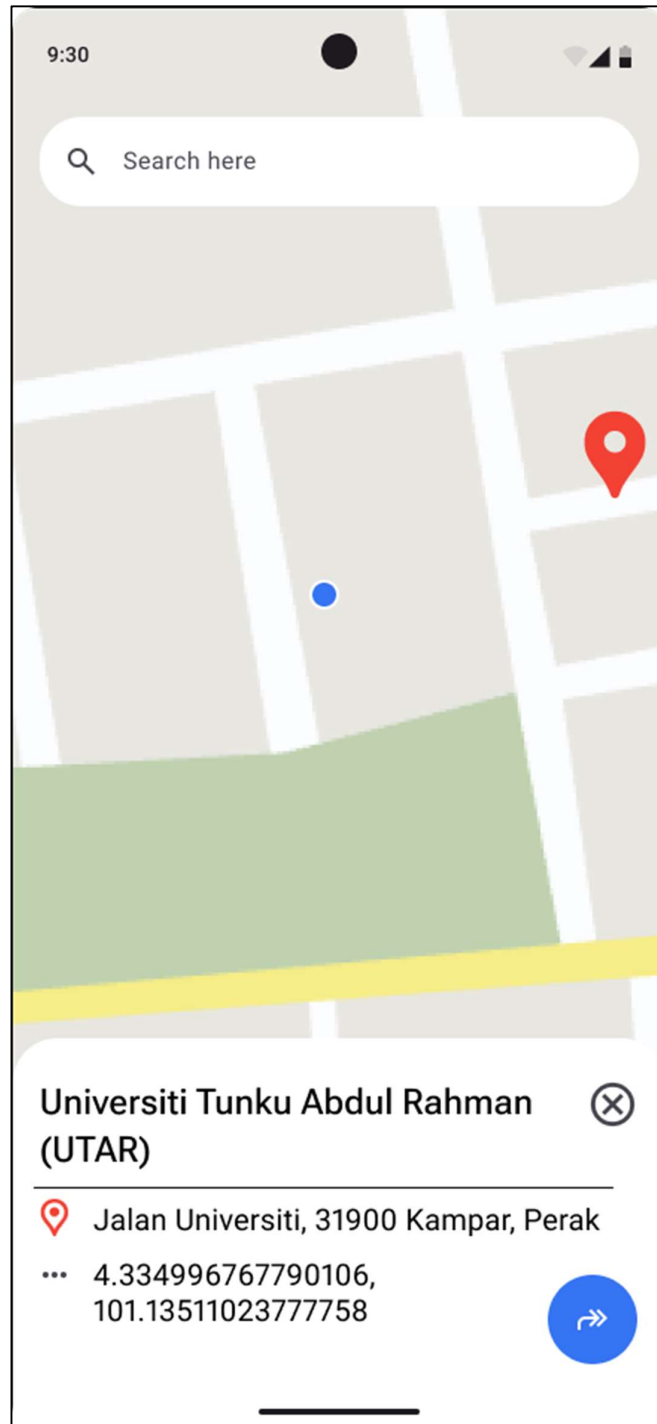
Loading – Failed Screen



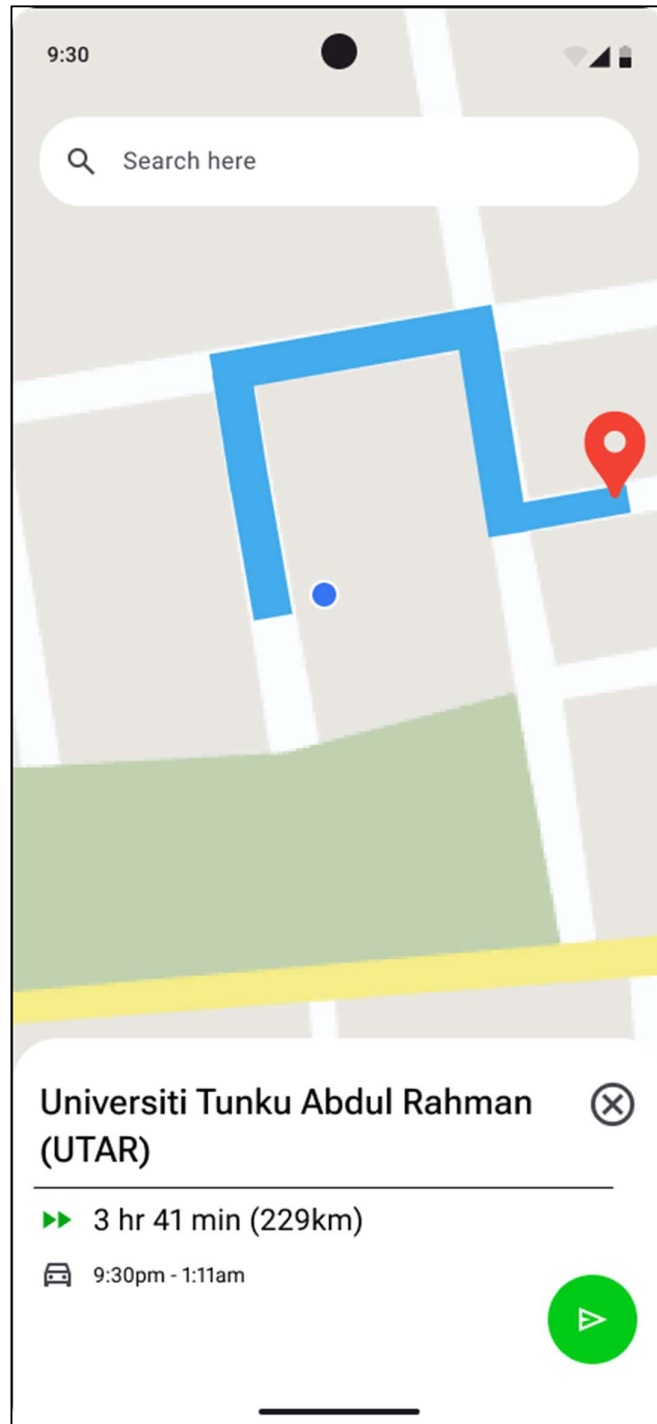
Maps Screen



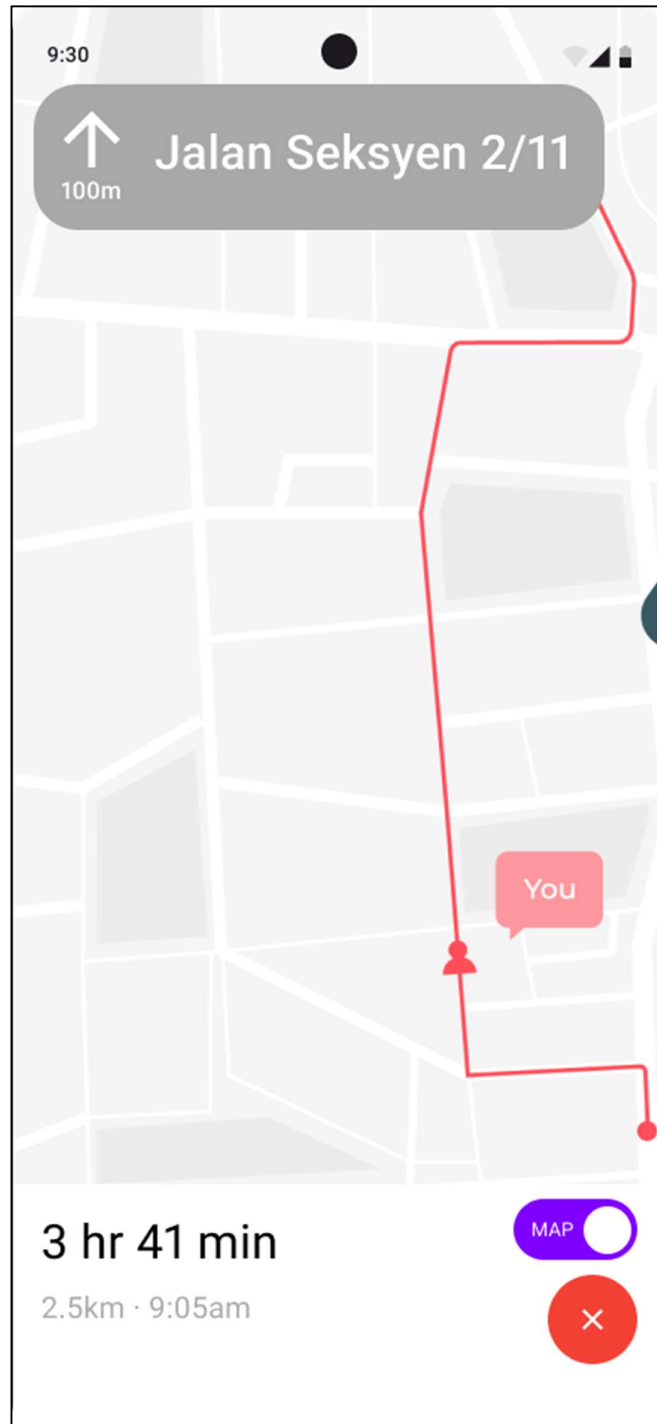
Maps – Search Screen



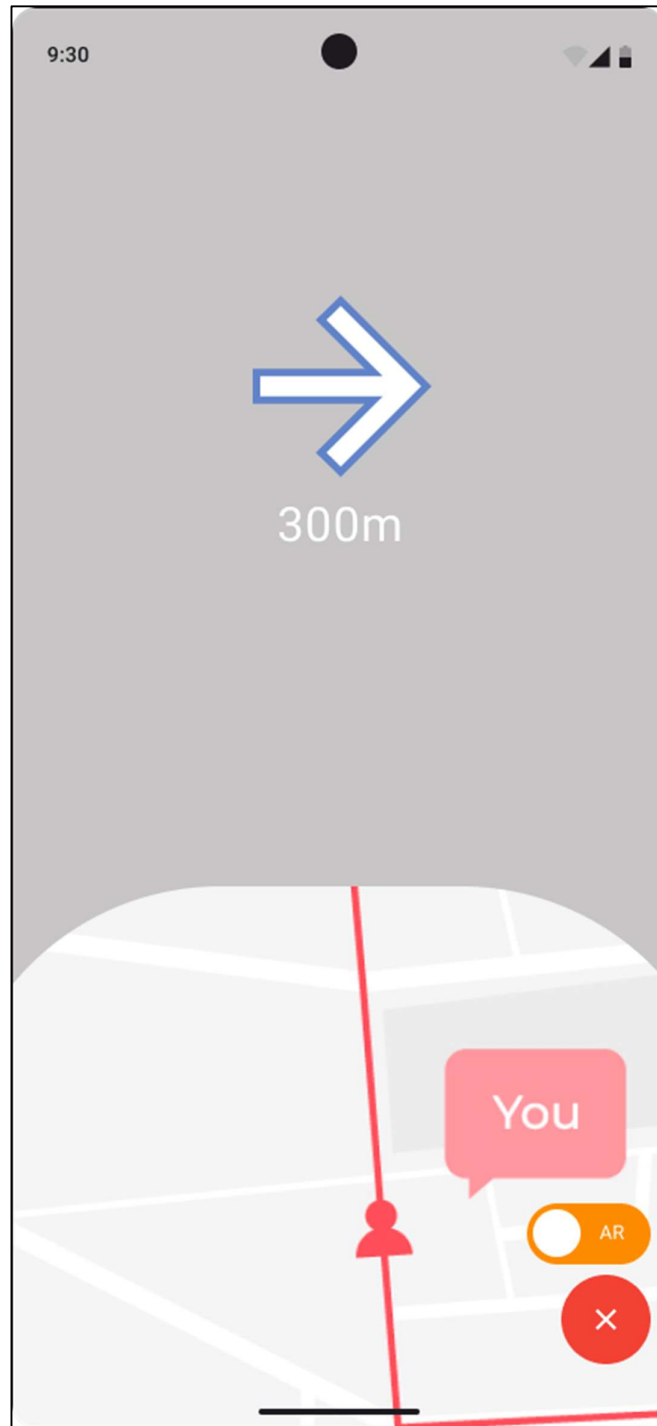
Maps – Selected Screen



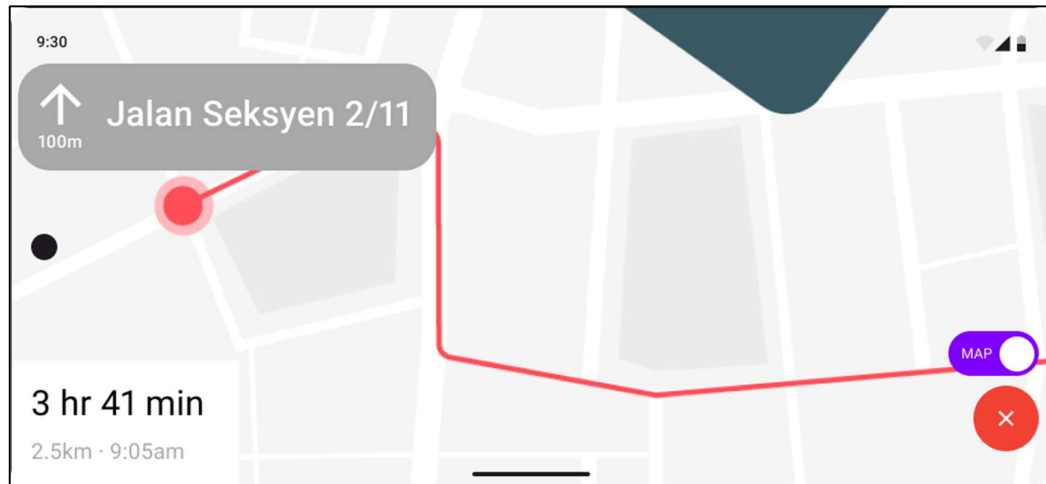
Maps – Directions Screen



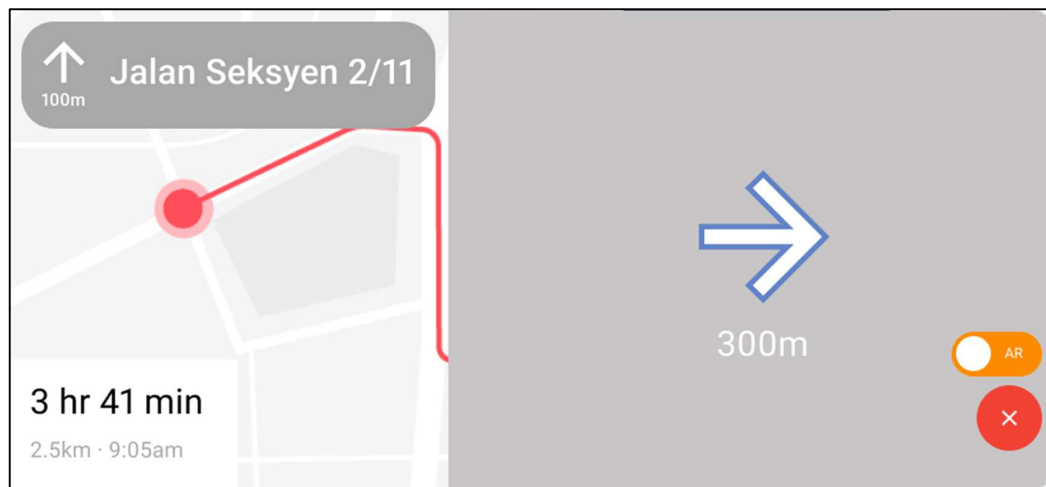
Maps Navigation Screen



AR Navigation Screen

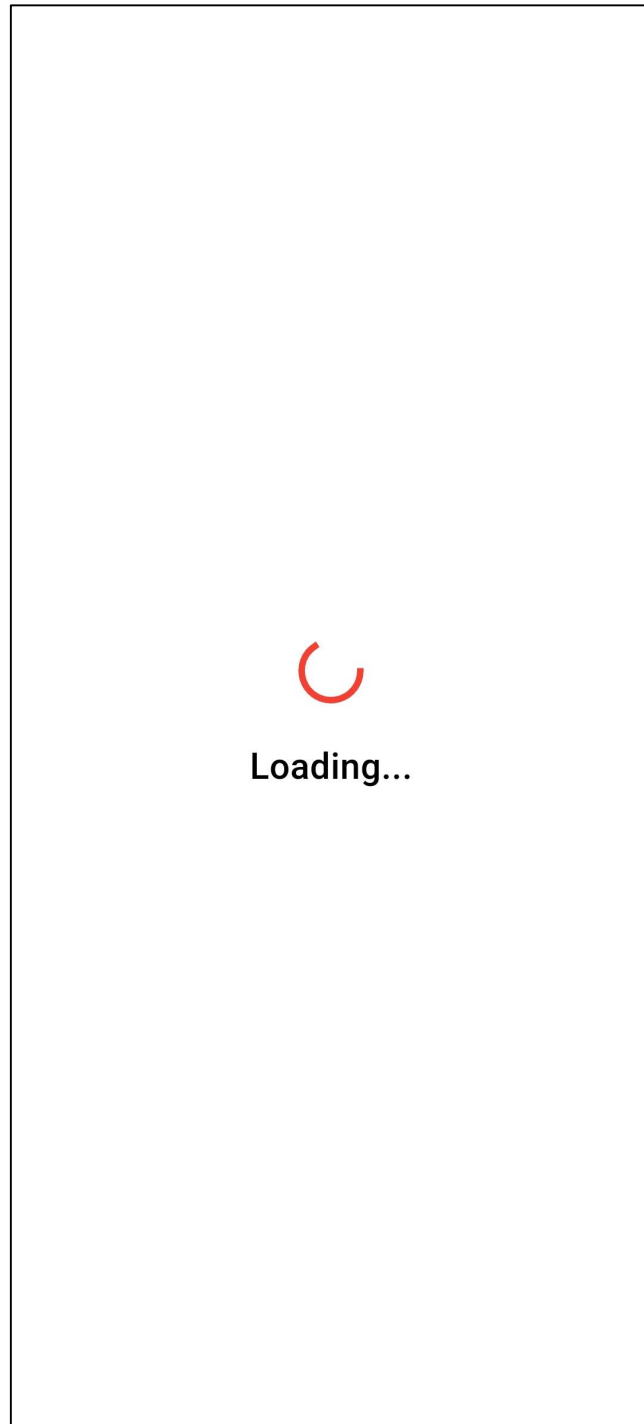


Maps Navigation – Horizontal Screen













AR Navigation – Horizontal Screen

Appendix C: High-Fidelity Design



Loading Screen

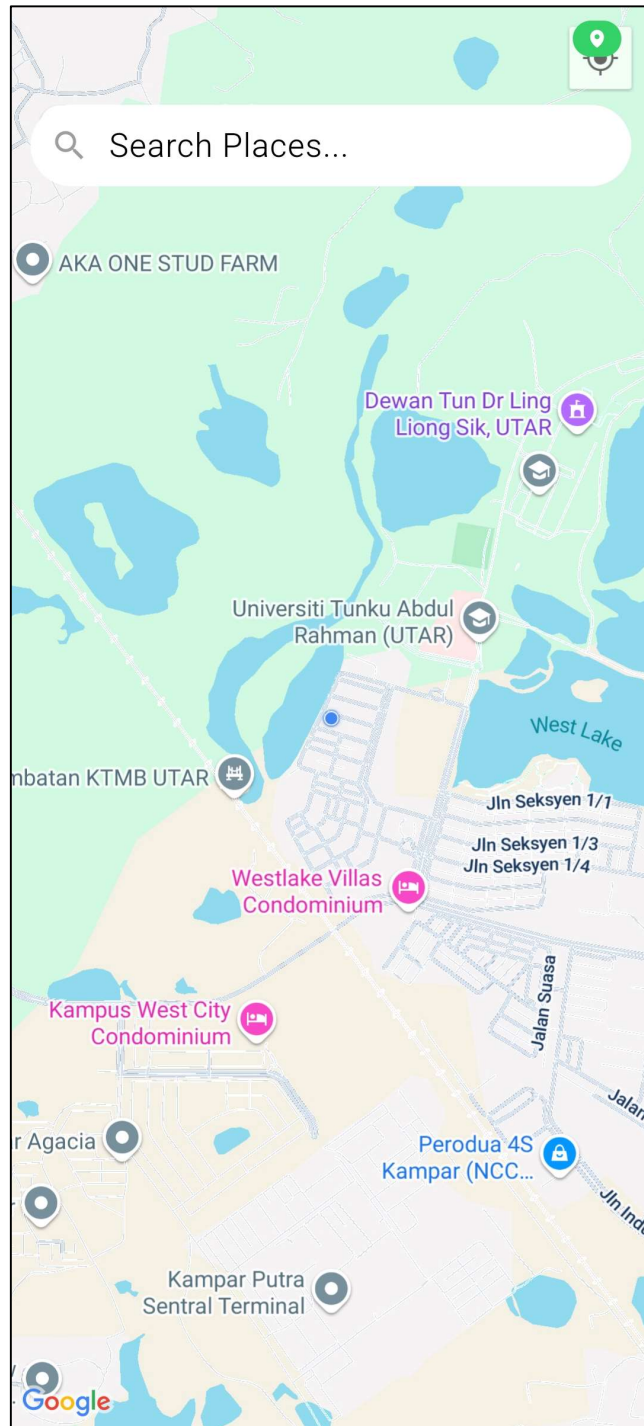
	Internet	
	Terms & Conditions	
	Google Maps API	
	Location	
	Camera	

Cleadr requires the
access of a few app
permissions to work.

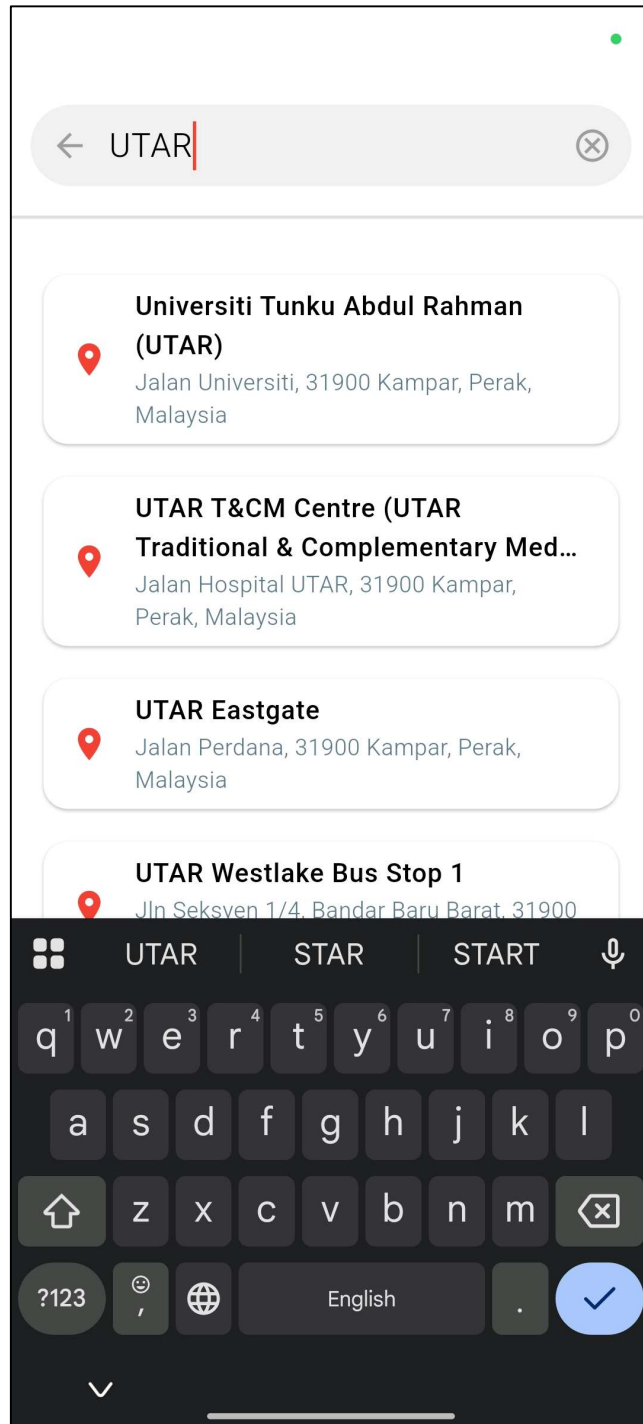
Please try again.



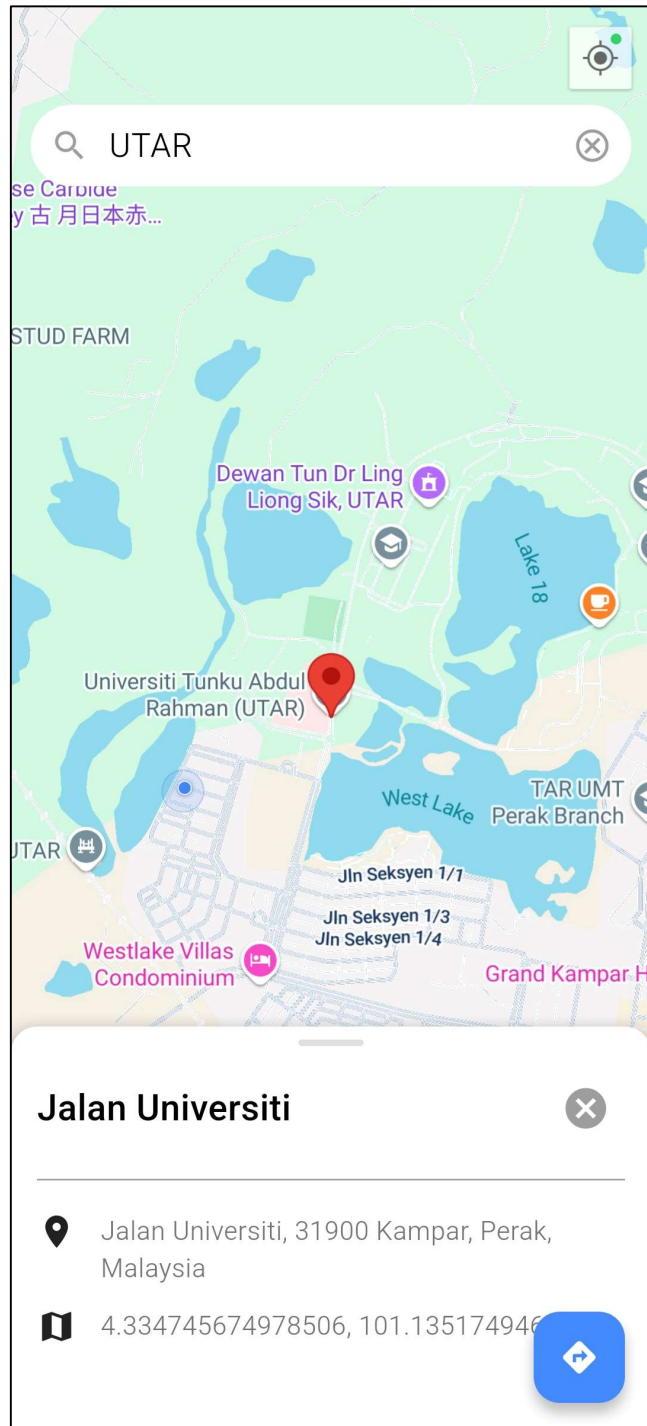
Loading – Failed Screen



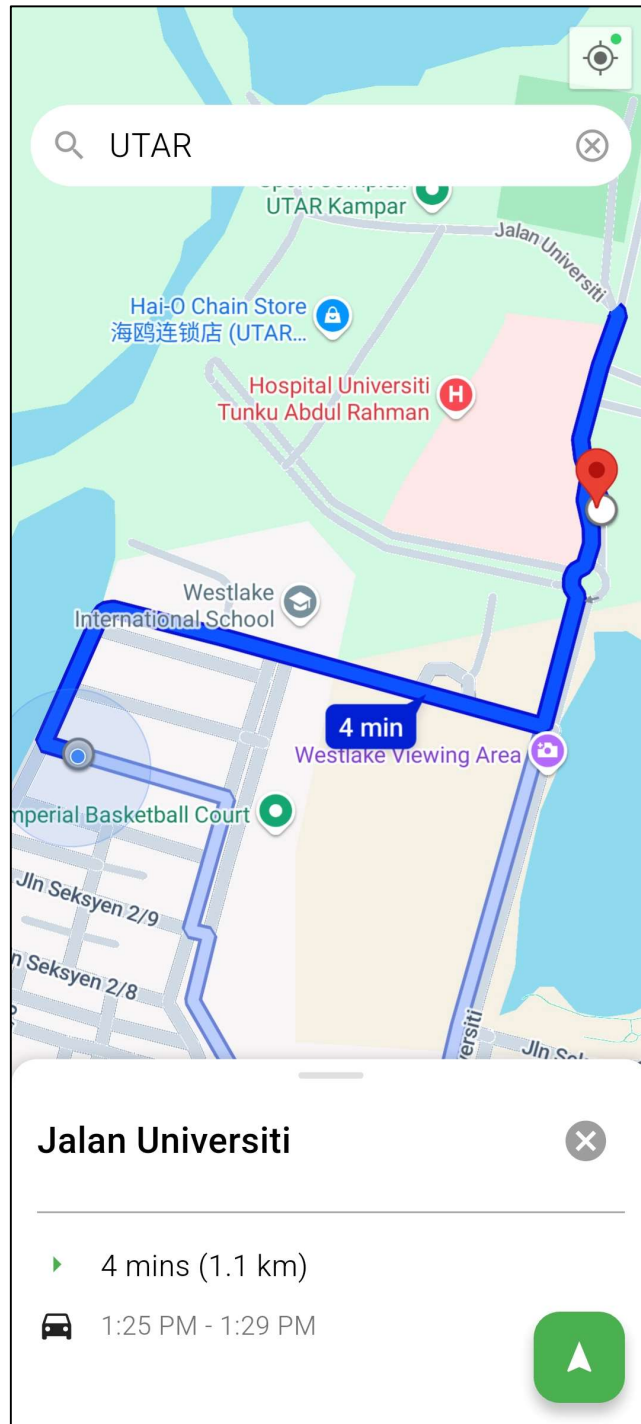
Maps Screen



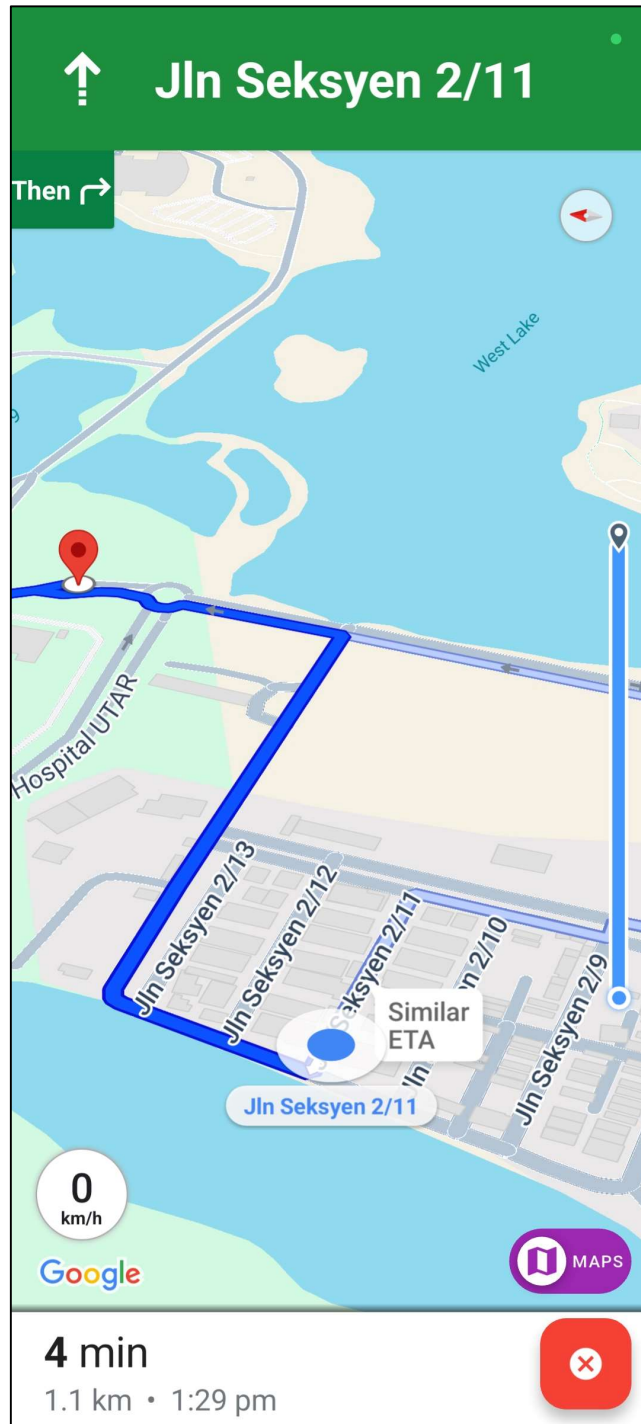
Maps – Search Screen



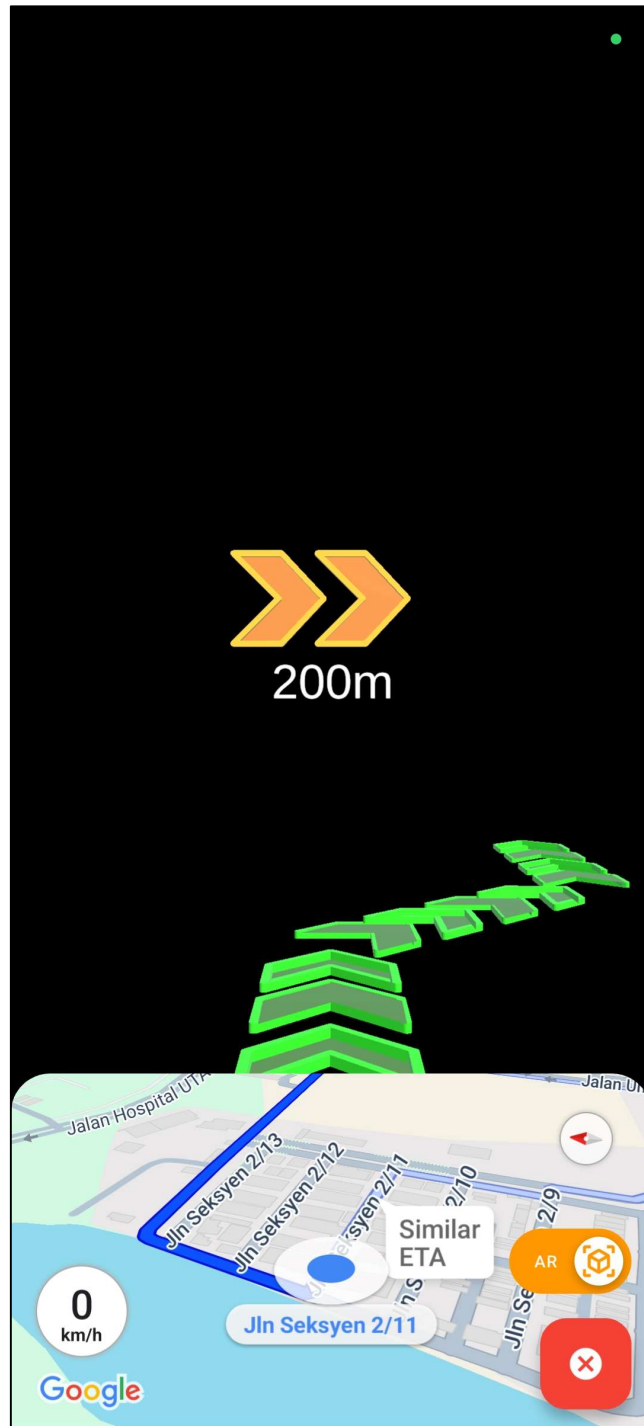
Maps – Selected Screen



Maps – Directions Screen



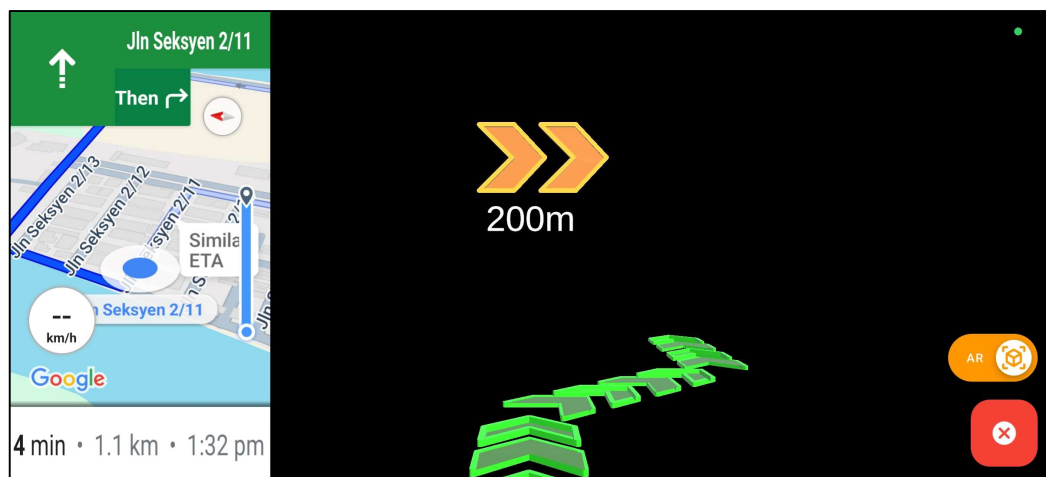
Maps Navigation Screen



AR Navigation Screen

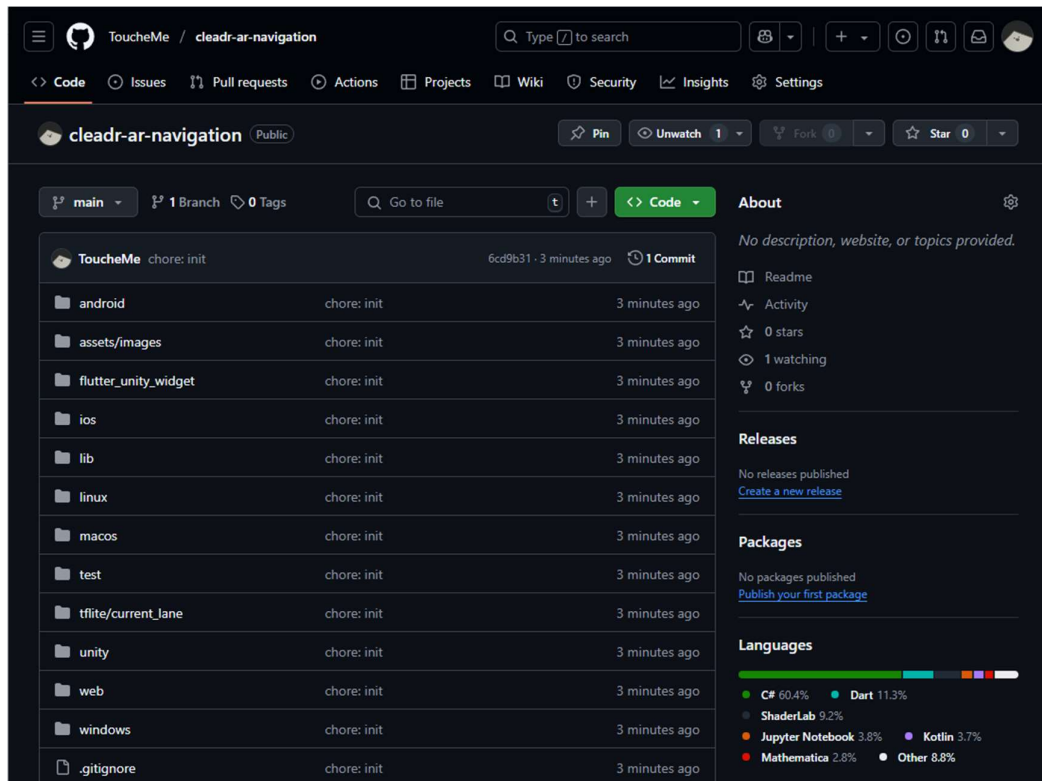


Maps Navigation – Horizontal Screen



AR Navigation – Horizontal Screen

Appendix D: GitHub Repository



<https://github.com/ToucheMe/cleadr-ar-navigation>

Appendix E: Poster

CLEADR :

AI-ENHANCED AR NAVIGATION APP FOR SEAMLESS DRIVING





INTRODUCTION

Navigation systems are important tools in any journey to confidently navigate from point A to point B. However, they are still prone to problems such as ambiguous directions, insufficient real-time assistance, and poor user experience.

FINAL YEAR PROJECT

OBJECTIVES

- To develop a mobile AR navigation application.
- To increase the effectiveness of AR navigation through AI integration.
- To improve the UX.

METHODOLOGY & TECHNOLOGIES

- Agile XP Methodology.
- Flutter.
- Unity.
- PyTorch.
- TensorFlow.



AR NAVIGATION

- Blue Arrow AR Direction
 - Turn Far Ahead . . .
 - 200m to 500m
- Yellow Arrow AR Direction
 - Prepare to Turn . . .
 - 50m to 200m
- Red Arrow AR Direction
 - Turn Now !
 - 0m to 50m



LANE IDENTIFICATION

- Lane number identification from front view of the road from vehicle.
- Exit up ahead, go lane number 1.
- Current lane number detected is lane number 2 → change left lane.
- AR direction of left lane change is displayed to notify the driver.



CONCLUSION

Cleadr is a proposed solution that involved a mobile AR navigation application integrated with AI. It provides enhanced clarity in directions to reduce confusion and uncertainty.



UNIVERSITI TUNKU ABDUL RAHMAN

Bachelor of Computer Science (Honours)
Faculty of Information and Communication Technology

Developer: Brandon Ting En Junn
Supervisor: Ts Dr Saw Seow Hui
Moderator: Mr Luke Lee Chee Chien