# SMART TRAVEL APPLICATION

By

Teoh Le Teng

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)

FEB 2026

# ACKNOWLEDGEMENT

Firstly, I would like to express my sincere appreciation to Ms. Leong Ying Mei, my project supervisor for her invaluable guidance, constructive feedback, and continuous encouragement throughout the development of this project. Her expertise and support have been essential in determining the direction and development of my Final Year Project.

I would also like to thank the Faculty of Information and Communication Technology (FICT), Universiti Tunku Abdul Rahman (UTAR) for providing me with the opportunity, facilities, and resources to undertake this project. Special thanks to my instructors and classmates for their insightful recommendations and moral support over the course of this project.

Finally, my sincere appreciation also goes to my family and friends for their constant encouragement, patience, and understanding during the challenging moments of my journey. Without their encouragement, this project could not have been completed.

# COPYRIGHT STATEMENT



This Final Year Project proposal is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal 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.

# ABSTRACT

This final year project proposes the development of an AI-driven Smart Travel Application that is designed to provide an intelligent and centralised application for planning and managing travel experiences. Existing travel applications frequently struggle from fragmented services, limited personalisation, lack of real-time travel updates, and insufficient offline features support. This project aims to resolve these limitations by integrating advanced technologies into a single user-friendly interface mobile application that developed with Flutter. The outcome of the application provides AI-generated travel plans based on destinations and durations, real-time chatbot assistance, offline accessibility for travel plans, and camera-based landmarks and buildings recognition. This application also provide automatically adjusts travel plans in response to flight delays or weather changes. The Smart Travel Application enhance travel confidence, safety and convenience for international travellers by resolving the weaknesses and limitations of existing applications. This final year project proves the standard for future intelligent travel assistance with comprehensive integration and personalisation.

**Area of Study:** Mobile Application Development, Artificial Intelligence in Travel Technology

**Keywords:** Smart Travel Application, AI-powered Application, Landmarks and Buildings Recognition, Offline Features, Automation Real-Time Update, Real-Time Weather, Map, Real-Time Currency, Real-Time Notification, Firebase

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| *AI* | Artificial Intelligence |
| *API* | Application Programming Interface |
| *ERD* | Entity Relationship Diagram |
| *LLM* | Large Language Model |
| *NLP* | Natural Language Processing |
| *OTA* | Online Travel Agency |
| *RTM* | Requirement Traceability Matrix |
| *UAT* | User Acceptance Testing |
| *UI* | User Interface |

# CHAPTER 1
# INTRODUCTION

In recent years, traveling has become an essential part of modern lifestyle [1]. People are traveling more frequently than ever before for leisure, business trip, education or cultural exploration [2]. Then, it has evolved from a luxurious lifestyle into a daily activity with people spending more of their free time to travel domestically or travel internationally [2]. The convenience of access to transportation, worldwide connectivity, and rising interest in lifestyle experiences are all contributing to the growth of travel sector. As the travel sector has become more common, so does the demand for tools and service that make the travel experience efficient and personalised [3].

The tourism industry has gone through an important digital transformation from traditional methods to utilise technology for all aspects of travel such as planning, booking and experiencing travel [4]. Traditional travel planning which was previously relied on traditional travel agencies to print brochures, guidebooks, personalised travel planning, booking services and travel advice through face-to-face interactions has evolved to online platforms and mobile applications [5]. Meanwhile, online travel agencies (OTAs) and booking platforms such as Agoda, Booking.com, and Traveloka are currently allowing users to search and book the flights, hotels and certain local experiences directly from their smartphones or computers [6]. These applications frequently provide basic travel services such as hotel reservations, flight bookings, and package discounts to make them useful for certain aspects of the trip [7], [8], [9]. However, these applications still lack complete functionality such as travel plan recommendations, real-time travel plan adjustments, or automated responses to flight delays and weather disruptions [7], [8], [9]. Moreover, existing applications frequently do not extend further basic booking and travel plan features [7], [8], [9]. They fail to evaluate the traveller's preferences or provide personalised recommendations based on travel objectives, traveller's duration and departure, and personal budgets [10]. Therefore, users typically switch between multiple applications just to manage a single travel which will cause an inefficient and annoying user experience [11]. For example, travellers might utilise Agoda to book for accommodation, Traveloka to book the flight ticket, Google Maps for the directions, and third-party applications for translation and plan for travel budget [7], [9], [12]. This fragmentation may contribute to delays, duplication

of data entry, and cluttered planning.

To solve these challenges, a new technologies called Artificial Intelligence (AI) have begun to improve how travel is planned and managed in the travel sector [13]. AI involves an array of intelligent systems such as machine learning, natural language processing (NLP), computer vision, and predictive analytics [14]. These kinds of technologies provide more advanced personalisation, contextual recommendations, and intelligent automation. For example, machine learning may evaluate behaviour of users, interests, and travel history to propose budget-friendly destinations and activities. NLP provide interactive AI chatbots that can respond to user questions in real time across multiple languages. Predictive analytics may predict disruptions such as flight delays or poor weather and adjust travel planning accordingly. Lastly, computer vision provides support camera- based translation and landmark recognition.

In conclusion, this final year project proposes the development of a comprehensive AI-driven Smart Travel Application built with Flutter by using Dart language. This application will function as an intelligent travel assistant that combine all the necessary travel functions into a single cross platform mobile application. The core features of this application include AI-based travel planning that suitable for user destinations and durations, real-time flight and weather updates, and integrated search and booking for flight, accommodations, and transportation. One of the core innovations is an AI driven travel plan generator that generates complete location-based travel schedules. For example, if the user plans 3 days 2-night trip in Kuala Lumpur, the system will automatically propose a detailed plan for Day 1, Day 2, and Day 3. Besides, additional features include offline access to travel plans, building and landmark recognition using the device base camera, and travel plans based on the real-time updates such as delays or cancellations or weather disruption. It aims to set a new benchmark for mobile travel applications by combining intelligent automation and real-time adaptation. Therefore, it not only addresses traditional concern such as fragmented services, reactive interfaces, and poor offline support. But also introduces a higher level of personalisation and predictive assistance that existing applications frequently need. In conclusion, this AI driven Smart Travel Application is designed to enhance traveller confidence and security, improve decision-making, and deliver a smooth and wonderful experience around the world anytime and anywhere.

## Key terms definitions:

**[i] Artificial Intelligence (AI)** refers to computer systems to perform task that typically require human intelligence.

**[ii] Natural Language Processing (NLP)** is a kind of AI that assists computers to understand and interact with human language.

**[iii] Predictive analytics** utilizes for data, statistical model, and machine learning to estimate future events or trends.

**[iv] Computer vision** is an area of AI that allows machines to analyse and process visual information from images or videos.

## 1.1 Problem Statement and Motivation

Although the tourism sector has experienced an important digital transformation in recent years, travellers still encounter considerable difficulties when planning and managing their trips.

**Problem Statement 1: Fragmented Services**

The most common concern is the service fragmentation which require users to utilise multiple applications to book flight, reserve accommodations, mange budgets, finding local attractions, translate languages and access travel information [6], [7], [8], [9]. For example, user might use Agoda for hotel reservations, Traveloka for flight booking, Google Maps for directions and more. This fragmented technique frequently leads to inefficiencies, delays and an insufficient of user experience when dealing with emergency adjustments, unfamiliar destinations and poor network connectivity.

**Problem Statement 2: Lack of Real-time Adaptation and Intelligent Automation**

Most of the existing travel applications only provide basic booking features and without real-time adaptation or intelligent automation [7], [8], [9]. They do not automatically respond to flight delays, cancellations or weather interruptions and causing users to manually reschedule their entire itinerary.

**Problem Statement 3: Lack of Personalised Planning and User Assistance**

Existing application do not offer personalised daily travel plans based on travel objectives, destination, duration [7], [8], [9]. The lack of contextual intelligence causes travellers to spend more time to research and reorganise their itineraries and reducing overall satisfaction. Also, the lack of interactive assistance tools reduces user satisfaction and travel experience.

**Problem Statement 4: Poor Offline Support**

Another significant limitation is that the high dependency on continuous internet access. Most of the existing applications lose functionality in low or no connectivity areas which limiting users from accessing essential services. Therefore, this can lead the foreign travellers to communication breakdown, safety concern in unfamiliar places with language and cultural differences.

### 1.1.1 Motivation

These challenges highlighted the critical need for an intelligent, integrated solution that can assist travellers throughout their entire travel. Therefore, this final year project is motivated by the opportunity to develop a comprehensive AI driven Smart Travel Application that combines multiple travel services into a single and user-friendly platform. The application will utilise advance technology to provide dynamic travel planning, real-time updates, and offline accessibility. In addition, the features will assist users travel smarter and greater confidence. It aims to create a smooth, proactive, and personalised travel assistant that sets a new benchmark in mobile travel assistance. Also, it aims to enhance travel decision-making and deliver a smooth and intelligent experience for modern travellers in anytime and anywhere.

## 1.2 Project Objectives

This final year project aims to design and develop an integrated AI-driven Smart Travel Application with utilising Flutter. It enhances the entire travel experience by combining intelligent features into a single platform. Also, it focuses on solving the common travel challenges such as fragmented services, lack of real-time updates, poor offline support, and travel planning personalisation. Therefore, this AI-driven Smart Travel Application aim to provide a smarter, smoother, and user centralised travel experience. The key objectives of this final year project are: -

**i. To develop an all-in-one AI-driven Smart Travel Application**

This objective focuses on developing a centralised mobile application that integrates the primary fragmented travel related services into a single mobile application to enhance user experience.

**ii. To implement real-time intelligent and predictive automation updates**

This objective focuses on monitoring disruptions and predicting problems to improve responsiveness, and adjust travel plans to ensure minimal inconvenience to the user in the case of flight delays or poor weather by providing notifications.

**iii. To enhance personalisation and user interaction through AI technologies**

This objective focuses on utilising advance technologies to provide intelligent automation personalised travel experiences. Besides, the system will generate personalised travel itineraries depending on travel destinations and durations. It will also include real-time chatbot assistance to respond to user inquiries and provide the solution of the inquiries.

**iv. To provide offline support for essential features**

This objective focuses on enabling offline access for essential features such as travel itineraries. This is to ensure that these features are workable wherever travellers are, regardless of poor or no internet connectivity.

## 1.3 Project Scope and Direction

The scope of Smart Travel Application involves designing, developing and implementing a completely integrated AI-driven mobile application utilising Flutter focused to enhance the travel experience for worldwide travellers. The application integrates core travel features into a single platform while managing important travel concerns such as fragmented services, lack of real-time updates, poor offline support and lack of personalisation travel planning.

**Core Features of the Smart Travel Application: -**

### 1.3.1 AI-driven personalised travel planning recommendations

The application will provide AI-driven personalised travel planning recommendations features that allow users to create, edit and save the personalised travel itineraries based on travel destination, and travel duration. The system can develop a suitable travel plan that matched to each user's specific needs, and improving the entire travel planning process and enhance the user satisfaction with it.

### 1.3.2 Camera-based landmark and building recognition

The application can recognise popular landmarks and provide users with relevant information for the device camera-based landmark and building recognition. For example, it can provide historical facts, opening hours, visitor tips and more. This enhances the user exploration and allows them to engage more deeper with their travel destinations.

### 1.3.3 AI chatbot assistance

The application will contain an AI chatbot to assist user in real time throughout their travel. The chatbot will be able to answer travel related questions and provide solutions to common problem through the system. This conversation support system will enhance the user experience by providing immediate feedback and interactive assistance when needed.

### 1.3.4 Integration of multiple service booking

The application will combine several booking services into a single platform for allowing the user to search, compare and book the services such as flight, hotel, transportation through external Application Programming Interfaces (API). This integration minimises the need for switching between different application to make the booking process easier, efficient and user friendly.

### 1.3.5 Real-time travel updates

The application will provide a real-time notification about travel events such as flight status updates, weather conditions, and emergency alerts. These live updates guarantee that user is aware of any changes that may affect their travel plans. The system will automatically adjust the travel plan of the user in response such as delays for reducing inconvenience and allow for more satisfaction travel experience.

### 1.3.6 Offline accessibility

The application will provide offline access to important resources to fit users travelling in the place with limited or no internet access. Users will be able to access features without having an internet connection to ensure ongoing support throughout the travel such as travel plan and flight ticket.

### 1.3.7 Real-time weather prediction

The application will include a location-based real-time weather function to keep user informed about current and upcoming weather conditions. It provides a detailed 24-hour hourly prediction as well as a 7-days weather prediction based on the user's exact location. This feature provides travellers plan their daily activities better and prepare for unforeseen weather changes. It makes informed decisions on the move to minimise unexpected travel disruptions.

### 1.3.8 Integrated interactive map

The application will feature an integrated interactive map to assist users in navigating unfamiliar environments. It allows users to track their current location, search for specific locations, and instantly discover nearby locations of interest such as attractions, restaurants, and accommodations. A built-in map keeps users within a single platform. It makes the exploration and navigation process more straightforward and convenient for the traveller.

### 1.3.9 Real-time currency conversion

The application will provide a real-time currency conversion feature to make it easier for users to manage their finances while travelling abroad. The system provides users to accurately convert between their home currency and the local currency of their destination by retrieving live exchange rates. This feature is particularly useful for assisting travellers make quick purchasing decisions, compare local prices, and maintain their travel budget without needing to open a separate financial application.

## Project Direction

The Smart Travel Application is developed to function as a centralised and AI-driven travel assistant that combine important features into a single mobile application. The project aims to solve common travel challenges by integrating intelligent automation using advance technology. For example, which it allows for interactive travel plan development, AI personalised travel planning, flight tickets to response real-time interruptions such as flight delays or weather changes. The application will enhance the user experience by providing offline access to travel plans and flight tickets. Therefore, the project aims to enhance traveller confidence, safety and convenience by focussing on developing an intelligent, smart, and user centred travel solution. It creates a new standard for mobile travel applications in the present digital world.

## 1.4 Project Impact, Significance and Contributions

The development of the Smart Travel Application is expected to bring an important impact on how the traveller plan, manage and experience travel in today's digital world. This project will address the significant limitations in existing travel applications such as fragmented services, lack of personalised travel planning, dependency on internet connectivity, lack of real-time response. This application contributes to enhance convenience, safety and satisfaction for both leisure and business traveller in travel sector. It centralises important services into a mobile application by delivering a complete AI-driven travel.

One of the most significant components of the project is the implementation of advance technologies to provide intelligent automation, real-time personalisation, and smart travel planning. The ability of the application demonstrates an evolution from static planning to smart and adaptable travel assistance which will generate personalised travel plan, recommend local attractions and restaurants, recognise landmarks and buildings by using device camera, and adjust travel plan in response to real-time updates such as flight delay or weather changes.

The project contributes inclusivity and accessibility features in the travel sector such as offline functionality. These features are beneficial for international travellers especially who encounter language barriers, and low or no connectivity areas.

In conclusion, this project contributes to researchers and the travel technology sector by providing a model solution that combines usability, intelligence, and adaptability. It has the potential to serve as new standard for future mobile travel applications to enhance user confidence and inspire further research and development in intelligent travel assistants. It represents the growing demand for intelligent, smooth and personalised digital experiences in the post-digital transformation era.

## 1.5 Report Organisation

This report is organised into four chapters. Each of the chapter will address a specific component of development of the AI-driven Smart Travel Application.

**Chapter 1: Introduction –** Provides an overview of the project that include the problem statement, objectives, project scope, and significance of the proposed solution.

**Chapter 2: Literature Review –** Conducts a comparative analysis of existing travel applications and discusses the strengths and weaknesses of current technologies. And justifying the need for an AI-driven and all-in-one solution.

**Chapter 3: System Analysis and Methodology –** Outlines the Agile development methodology adopted for this project. It details the user requirements, system specifications, and the functional and non-functional requirements.

**Chapter 4: System Design –** Illustrates the system architecture, database schema, and user interface designs. It focuses on the structural blueprint of the application before the implementation phase.

**Chapter 5: System Implementation –** Describes the actual development process, focusing on the integration of the Flutter Framework, Firebase backend services, and the Google Gemini API for AI-powered features.

**Chapter 6: System Evaluation and Discussion –** Details the complex testing procedures, including functional and offline resilience testing. It also evaluates the final system against the initial project objectives.

**Chapter 7: Conclusion and Recommendation –** Summarise the overall achievements of the project, acknowledges current system limitations, and proposes future enhancements for further development.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Previous Related Works on Travel Applications

In recent years, multiple type of travel application has been appeared to support the growing demand for convenient trip planning, booking and navigation. While many of these applications focus on specific functions such as flight and hotel reservations, itinerary management or destination reviews. However, it often lacks integration, personalisation, and intelligent automation. Therefore, this chapter provides an overview of popular travel applications that attempt to simplify travel such as Agoda, Booking.com, Traveloka, Trip.com, Tripadvisor, CheckMyTrip, TripIt: Travel Planner, Wanderlog, and Kayak.

### 2.1.1 Agoda, Booking.com, Traveloka and Trip.com

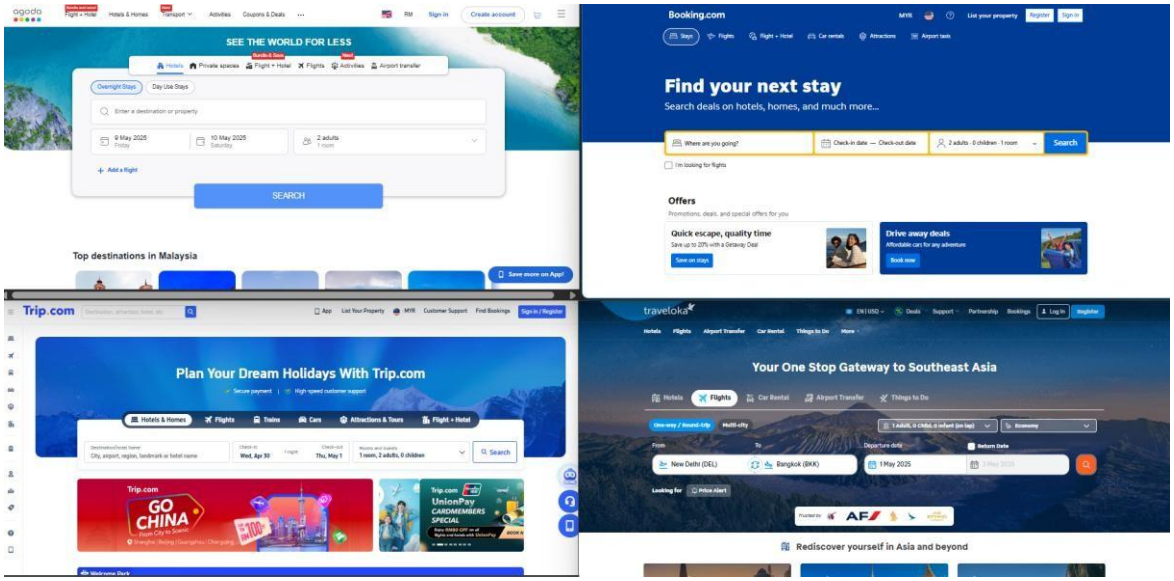



Figure 2.1 Agoda, Booking.com, Traveloka, Trip.com

Agoda, Booking.com, Traveloka and Trip.com are different brand that belong to the most popular worldwide travel applications that provide the booking services for flight, hotels, transportation as well as local experiences. These applications

provide user friendly interfaces, multi-language assistance, currency converters, and promotional discounts. At the same time, Traveloka and Trip.com also provide specific services such as insurance, car rentals, and local event reservations. Although these applications are effective in integrating booking services for flights, hotels and accommodations, but they fail to provide the important services in travel sector to assist user such as smart travel planning, AI based personalisation, real-time travel disruption management, offline access to essential information, and camera-based landmarks and buildings recognition. In conclusion, these applications typically serve the same core functions of assisting travel bookings however lack a completely intelligent and integrated travel management system [7], [8], [9], [15].

### 2.1.2 Tripadvisor

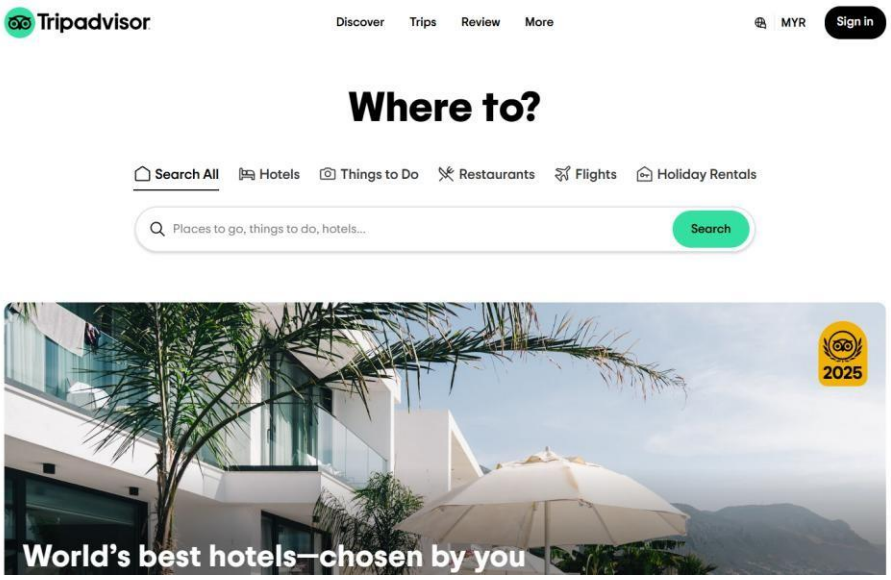



Figure 2.2 Tripadvisor

Tripadvisor is a global application that provides travel related information such as user generate reviews, rankings, and forums. It involves hotels, restaurants, and tourist attractions to provides an experience-based approach to travel planning. This application is known for enhance user exploration of destination through shared the different point of view in the community. However, it does not provide booking service in the system, lack of travel planning features, and does not provide

AI or visual recognition technology to enhance interactivity of the user while travelling [16].

### 2.1.3 CheckMyTrip and TripIt: Travel Planner

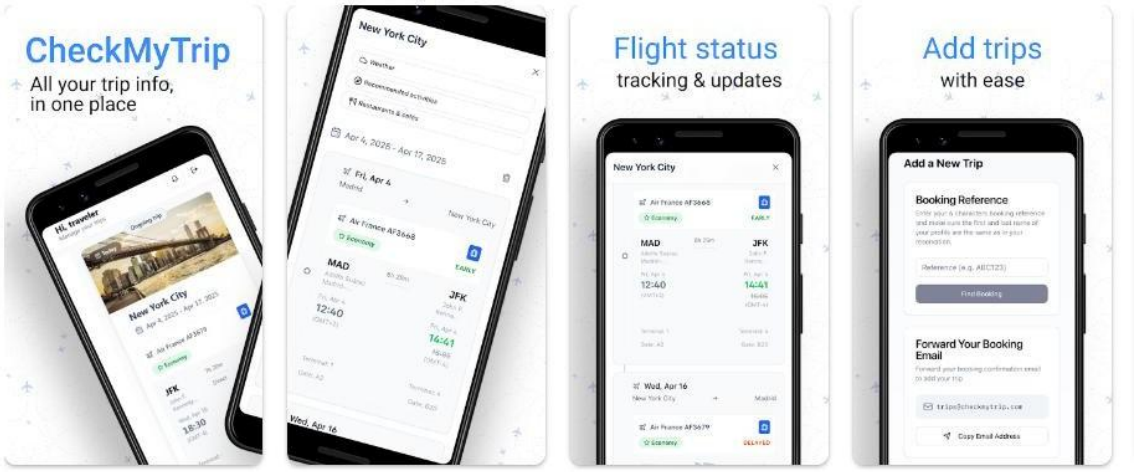



Figure 2.3 CheckMyTrip

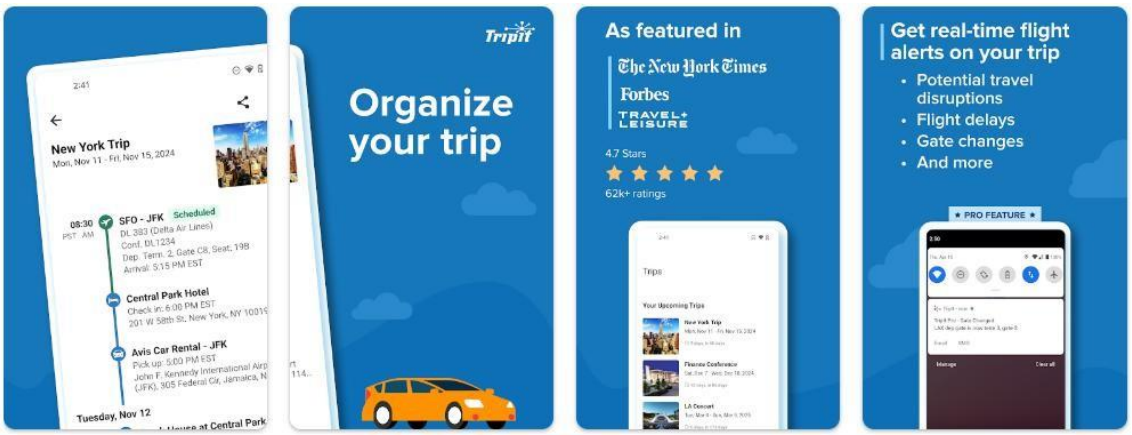



Figure 2.4 TripIt: Travel Planner

CheckMyTrip and TripIt: Travel Planner are the applications that focused on planning and managing travel plan. Both applications integrate booking by synchronising from confirmation emails. They are providing notification, calendar integration, and a centralised itinerary view. These applications are useful especially for business travellers those want to scheduled plans. However, these applications lack integrated booking system, AI-based personalisation, camera-based

landmarks and buildings recognition, real-time disruption management and a fully offline functions to make it more of an interactive application for travel plan tracking rather than a functionality travel assistant [17], [18].

### 2.1.4 Wanderlog

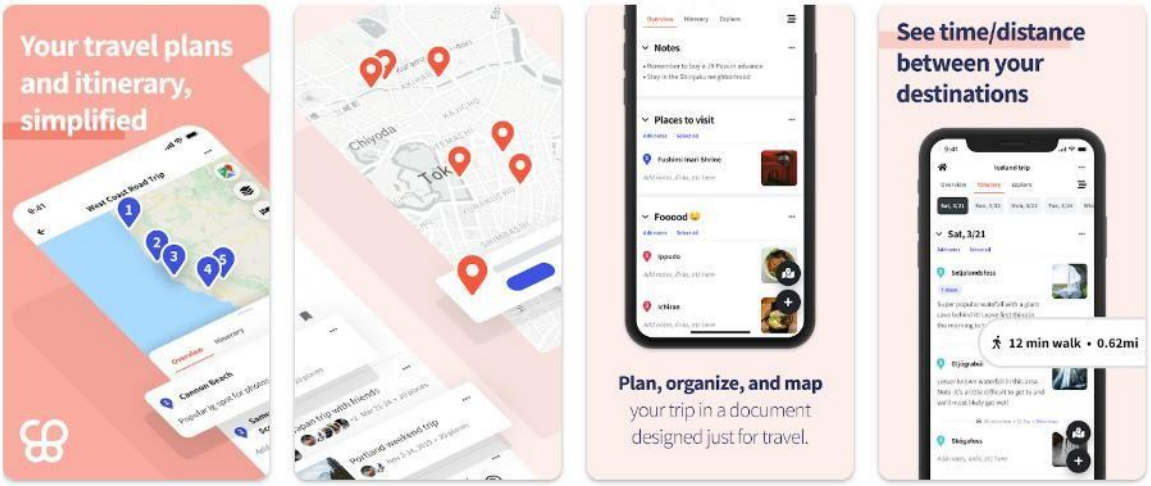



Figure 2.5 Wanderlog

Wanderlog is a collaborative travel planning application that enables users generate and share the itineraries visually to their friends or family through the application. It provides a drag and drops planner, integrated mapping, and offline functionality. In addition, it is especially useful for road trips and group travel. It enhances the efficiency of the travel planning. However, it lacks AI automation, camera recognition, real-time updates, and booking features. Therefore, it is requiring users to manage travel individually [19].

### 2.1.5 Kayak

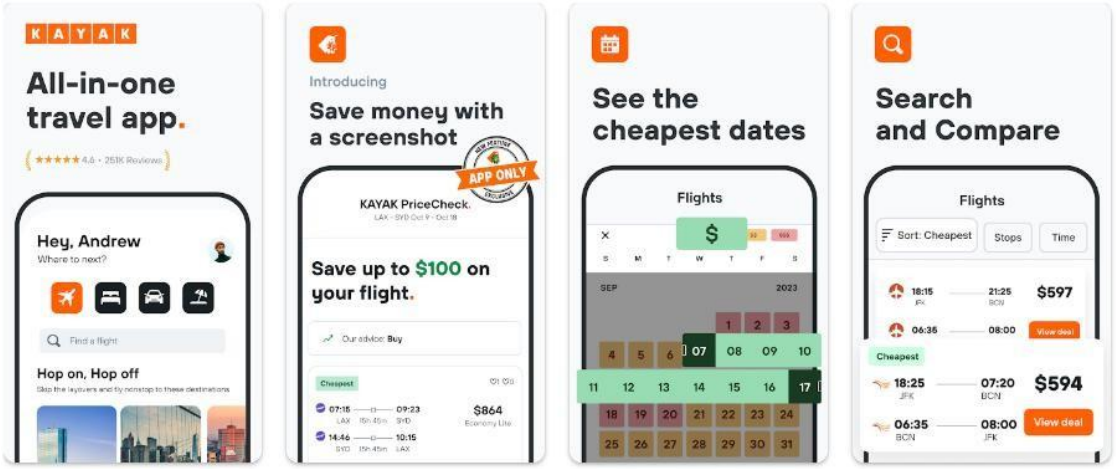



Figure 2.6 Kayak

Kayak is a travel meta-search engine that compare the prices across different booking applications for flights, hotels, and rental cars. It provides features that can assist users to determine the travel options based on their budgets and interests such as price predictions, deal alerts, and an explore map. Therefore, this application benefits in assisting users to determine the cheapest deals. However, it does not provide travel planning, schedule adjustment, offline features, and AI based assistance [20].

## 2.2 Overview of Existing Travel Applications

Existing applications such as Agoda, Booking.com, Traveloka, Trip.com, Tripadvisor, CheckMyTrip, TripIt: Travel Planner, Wanderlog, and Kayak are well-known and common applications that used for travel reservations and itineraries planning. However, each focus of the applications on a different set of functions and lack of integrated, intelligent that can fulfil the expectations of modern travellers.

Table 2.1 Overview of existing travel applications

| **Model / Application** | **Functionalities** | **Features** | **Strengths** | **Weaknesses and Limitations** |
|---|---|---|---|---|
| **Agoda, Booking.com, Traveloka, Trip.com** | -Flight bookings<br>-Hotel bookings<br>-Transportation Bookings<br>-Event or Attraction reservation | -Multi-language support<br>-Currency converters<br>-Discount or Deals<br>-Local payment methods | -Global accessibility<br>-Strong booking integration<br>-User friendly Interface | -No AI travel planning and budget management<br>-No AI chatbot<br>-No real-time travel updates<br>-No camera-based landmarks and buildings recognition<br>-No offline Features |
| **Tripadvisor** | -Reviews and destination discovery<br>-Forums | -User generated information | -Destination research | -No booking system<br>-No AI travel planning and |

| | | | | |
|---|---|---|---|---|
| | -Travel tips | -Photos and rankings<br>-Forums | -Strong community interaction | budget management<br>-No offline features<br>-No camera-based landmarks and buildings recognition<br>-No AI chatbot<br>-No real-time travel updates |
| **CheckMyTrip, TripIt: Travel Planner** | -Travel itinerary tracking<br>-Email booking sync<br>- Notification and reminders | -Calendar sync<br>-Central itinerary view<br>-Notification system | -Organising business travel<br>-Simple travel optimisation | -No booking system<br>-No AI travel planning and budget management<br>-No camera-based landmarks and buildings recognition<br>-No AI chatbot<br>-No real-time travel updates<br>-No offline features |

| | | | | |
|---|---|---|---|---|
| **Wanderlog** | -Collaborative itinerary builder<br>-Visual trip planning | -Drag and drop planner<br>-Offline mapping<br>-Group sharing | -Road trip suitability<br>-Offline usability<br>-Simple User Interface | -No booking system<br>-No AI travel planning and budget management<br>-No camera-based landmarks and buildings recognition<br>-No AI chatbot<br>-No real-time travel updates<br>- No currency conversion support |
| **Kayak** | -Price comparison<br>-Travel deal search<br>-Budget based exploration | -Price prediction<br>-Explore map<br>-Multi-site filtering | -Strong deal search engine<br>-Smart filters | -No AI travel planning and budget management<br>-No camera-based landmarks and buildings recognition<br>-No offline features |

| | | | | |
|---|---|---|---|---|
| | | | | -No AI chatbot<br>-No real-time travel updates |

## 2.3 Comparison between Existing Travel Applications

A detailed features comparison of existing travel applications is shown as below:

Table 2.2 Comparison between existing travel applications

| Application<br><br>Features | Booking Apps (Agoda, Booking.com , Traveloka, Trip.com) | Review Platforms (Tripadvisor) | Planners (CheckMyTrip, TripIt: Travel Planner) | Wanderlog | Kayak |
|---|---|---|---|---|---|
| **Booking system** | ✓ | X | X | X | ✓ |
| **Itinerary planner** | X | X | ✓ | ✓ | X |
| **AI planning and budgeting** | X | X | X | X | X |
| **Real-time travel updates** | X | X | X | X | X |
| **AI chatbot** | X | X | X | X | X |
| **Camera-Based Recognition** | X | X | X | X | X |
| **Offline features** | X | X | X | ✓ | X |
| **Map** | X | X | X | X | X |
| **Currency conversion support** | ✓ | X | X | X | X |

## 2.4 Proposed Solution

The proposed Smart Travel Application aims to combine the strengths of these existing systems while addressing their weaknesses and limitations:

Table 2.3 Proposed solution

| **Existing Application** | **Identified Limitations** | **Proposed Solution** |
|---|---|---|
| **Agoda, Booking.com, Traveloka, Trip.com** | -Highly transactional<br>-Lacks intelligent planning<br>-Real-time guidance<br>- Lack of offline assistance | -Integrates an AI travel planner and budget management tool.<br>-Implements a real-time AI chatbot for instant support.<br>-Features camera-based landmark recognition<br>-Provide offline access for essential travel plan |
| **Tripadvisor** | -Lacks an integrated direct booking system<br>-Intelligent real-time travel features | -Centralises external APIs to provide a direct booking system.<br>- Offer AI driven personalised planning and real-time travel updates.<br>-Features camera-based landmark recognition and offline capabilities |
| **CheckMyTrip, TripIt:Travel Planner, Wanderlog** | - Fragmented experience<br>- Users still need separate apps for bookings, currency, and live guidance | -Incorporates a comprehensive booking system within the app<br>-Enhances user exploration with camera-based landmark recognition<br>-Provides live AI chatbot assistance and real-time currency conversion tools. |
| **Kayak** | -Mainly a search engine<br>-Lacks interactive AI assistance<br>-Offline features | - Employs AI technology for travel plan<br>-Support offline feature accessibility for travellers without internet. |

## 2.5 Summary of Literature Review

First, this chapter reviewed many kinds of existing travel related applications and has highlighted their strengths as well as their weaknesses and limitations. Although, the applications such as Agoda, Booking.com and Trip.com were provided strong booking features but they lack intelligence planning and integration. Tripadvisor develops in community generated information while it lacks booking and automation features. TripIt: Travel planner, CheckMyTrip and Wanderlog are amazing travel planning application whereas they lack real-time travel update and integration services.

In conclusion, the proposed Smart Travel Application fills these gaps by integrating advanced technology, offline features, multi-language support, and intelligent automation. It combines important features such as bookings, travel plan generation, disruption management, and user interaction into a user-friendly application. Therefore, it aims to provide modern travellers with more efficient, safe and satisfying travel experience that personalised to their preferences in the moment.

# CHAPTER 3

# SYSTEM METHODOLOGY

## 3.1 System Methodology


Figure 3.1 Agile Methodology

The Smart Travel Application was developed utilising the Agile Software Development Methodology. It is well-known for its flexibility, iterative workflow as well as adaptability to the project that require continuous feedback and incremental modifications [21]. The Agile method divides the project into multiple sprints, and each of the sprint including requirements analysis, designing solutions, developing modules, testing results, deployment, and review [21]. Therefore, this method ensure that the project progress is frequently evaluated by the supervisor and users. In addition, it also allows for rapid modification to feedback or technical challenges.

In the beginning phases, the priority will be on collecting user requirement and determining the essential features required to solve the typical limitations of the existing travel applications. Once the requirement has been determined, the system design phase will specify the software architecture and workflows using diagrams to visualise the interaction between modules. After that, prototypes will be developed incrementally staring with the travel planning and offline storage modules. Future phases will include more advanced features such as AI chatbot, and camera-based recognition system. As new features are developed, the integration testing will perform to ensure module consistency and system reliability. The iterative enhancement process will continue until the final delivery meets the specified objectives.

## 3.2 Tools to Use

This section describes the core technologies, frameworks, libraries, and services that utilised to develop and support the functionality of the AI-driven Smart Travel Application. The combination of these technologies allows for the implementation of core features such as AI driven itinerary generation, camera-based landmarks and buildings recognition, AI chatbot, integration of multi booking services, and real-time travel updates.

### 3.2.1 Visual Studio Code

Figure 3.2 Visual Studio Code

Visual Studio Code is a lightweight editor for Flutter development and plugin integration [22].

### 3.2.2 Android Studio

Figure 3.3 Android Studio

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development [23]. It provides a unified environment where can build apps for Android

phones, tablets and more [23]. Structured code modules allow to divide project into units of functionality that can independently 2build, test, and debug [23].

### 3.2.3 Flutter


Figure 3.4 Flutter

Flutter is an open-source User Interface (UI) toolkit that developed by Google [24]. It enables developers to develop beautiful and natively for designed mobile applications, web application, and desktop applications from a single codebase [24]. It is used to develop a cross platform mobile application for Android with future potential expansions for iOS or web application in this project [24].

### 3.2.4 Google AI Studio


Figure 3.5 Google AI Studio

Google AI Studio is a web-based development platform to build customise and deploy generative AI models [25]. It provides a user-friendly interface for developing applications that allow powerful natural language processing and content modelling to be integrate into software system based on Large Language models (LLMs) [25]. It serves as the foundation for the AI-driven features in this project specifically for generating travel plan, and powering the chatbot

to deliver dynamic, context-aware replies that improve the traveller's experience [25].

### 3.2.5 Firebase


Figure 3.6 Firebase

Firebase serves as the backend-as-a-service solution for managing user authentication, cloud database, and functions without servers [26]. Firebase Authentication provides secure user login and account management [26]. Firestore is a NoSQL cloud database that stores travel plan, booking records and travel history [26]. Cloud functions are allowing for customised backend logic such as sending itinerary updates or push notification [26].

### 3.2.6 Integration of REST APIs

Multiple REST APIs were integrated to expand the functionality of application and access live data. Flight and hotel bookings APIs was used to search and book the flights and hotel such as Trip.com API, Booking.com API. These APIs enable the application to provide user with real-time travel services and content internationally.

### 3.2.7 Draw.io


Figure 3.7 Draw.io

Draw.io is a free online diagramming software application. It allows developers, network administrators, IT analysts, and designers to build and distribute diagrams using drag-and-drop capabilities [27]. Professionals may use the system to build toggle layers with configurable URLs and align content within the shapes [27].

**Summary of tool to use**

The combination of these technologies enables the AI-Driven Smart Travel Application to provide intelligent automation, multi-language support, offline features, and real-time travel updates. Each of the component was chosen based on the scalability, developer support, and connectivity with mobile application development. In conclusion, it to ensure a reliable and modern travel experience for end users.

## 3.3 Justification of tools to use

Each of the technology was selected based on its functionality, simplicity of integration, and ability to fulfil the primary features of the project. Therefore, the following table are the summarises the justification for each technology.

Table 3.1 Justification of tools to use

| Tools | Purpose | Justification |
|---|---|---|
| **Visual Studio Code** | Integrated Development Environment (IDE) for Flutter | Lightweight, rapid with important extensions for Flutter development |
| **Android Studio** | Android development environment and emulator | Official IDE for Android, provides robust emulation, debugging, and deployment tools for mobile testing |
| **Flutter** | Cross platform application development | Single codebase, fast development, customisable User Interface (UI) |
| **Google AI Studio** | Generative AI model development | Provide advanced natural language understanding and content generation, easy to integrate with Flutter |
| **Firebase** | Backend service for authentication and storage | Simple integration for Flutter, scalable, real-time updates, minimal server setup required |
| **REST APIs** | Real-time booking system, Currency rates, Real-time Weather, Map | Access to the latest global data, speeds up the development, reduces requirement for internally service developing |
| **Draw.io** | System design and diagramming tool | Free web-based tool for creating flowcharts, architecture diagrams, and use case diagrams, supporting clear documentation of system design |

## 3.4 User Requirements

The user requirements for the Smart Travel Application are separated into functional and non-functional requirements. The functional requirements determine the core features and functions that the system must deliver to the users. In the other hands, the non-functional requirements specify the quality features and performance expectations that ensure the system's usability and reliability.

### 3.4.1 Functional Requirements

Table 3.2 Functional requirements

| **Requirement** | **Description** |
|---|---|
| AI travel Assistant | The system shall provide an AI-powered chatbot to offer personalised travel recommendations and answer user queries. |
| Flight Booking and Management | Users shall be able to search for flights, select flight and class, and generate digital boarding pass with real-time barcode and QR code generation. |
| Integrated Map and Route Navigation | The system shall allow users to capture the landmark and provide route visualisation, distance calculation, duration estimates using OpenStreetMap data. |
| Real-time Currency Converter | The system shall provide a tool for users to convert multiple internation currencies based on the latest live exchange rates. |
| Real-time Weather Updates | The system shall fetch and display current weather conditions (temperature, humidity, and prediction) for the user's current location. |
| Booking History and Offline Access | The system shall store booking records locally using shared_preferences to allow users to view their trips even without an internet connection. |
| Smart Notification | The system shall push local notifications to the user regarding flight status and reminders. |
| User Authentication | The system shall support secure login and sign up via email and password or Google Sign-in through Firebase Authentication. |

### 3.4.2 Non-Functional Requirements

Table 3.3 Non-functional requirements

| **Requirement** | **Description** |
|---|---|
| Usability | The system shall feature a simple and user-friendly interface suitable for travellers with different backgrounds. |
| Performance | The system shall ensure that travel plan generation and chatbot responses are delivered within three seconds under normal operating conditions. |
| Reliability | The system shall remain functional even in environments with weak or no internet connectivity, with offline features available at least 90% of the time. |
| Security | The system shall employ secure authentication mechanisms and encrypt sensitive data to protect user privacy. |
| Compatibility | The system shall operate on both Android and iOS platforms ensuring cross-platform accessibility. |
| Scalability | The system shall be designed to handle increasing number of users and data without degradation of performance. |
| Maintainability | The system shall allow updates and improvements to be apply efficiently with minimal disruption to users. |

## 3.5 System Performance Definition

The expected performance of the Smart Travel Application is defined through measurable targets as summarised in Table 3.4 in below section.

Table 3.4 System performance definition

| **System Performance** | **Performance Target Definition** |
|---|---|
| Building and Landmark Recognition Accuracy | At least 80% accuracy with standard lighting and image capture conditions. |
| AI Travel Plan Generation | Generate a comprehensive multiple day personalised travel plan involves comples prompt processing. The system should complete the structured JSON data parsing and UI rendering within 8 to 12 seconds. Displaying a "Thinking…" or loading indicator to manage user expectations gracefully. |
| AI Chatbot | For standard conversational queries, the system must process the user's input and display the AI's response within 2 to 4 seconds under a stable internet connection. |
| Real-Time Updates Delivery | Notifications from flight status and weather APIs provided within 3 seconds of external data retrieval. |
| Offline Access | Retrieving saved booking histories and offline trip plan using shared_preferences must be instantaneous. It ensures that users can access their digital tickets in environments with no network coverage. |
| Map rendering | The flutter_map widget must load map tiles seamlessly without freezing the main UI thread and maintaining a smooth scrolling and zooming experience (close to 60FPS) on standard mobile devices. |

| | |
|---|---|
| Geocoding Speed | Converting user's input into accurate latitude and longitude coordinates must be completed within 2 seconds. |
| Weather Updates | The system must retrieve current weather conditions and temperature based on the user's location within 2 seconds. The UI must dynamically update the weather icons and animations instantly upon receiving the payload. |
| Currency Conversion | Fetching live exchange rates and calculating the converted amount must be executed instantaneously less than 1 second upon user input. It ensures that the numeric keypad interaction feels fully synchronous and lag-free. |
| UI Interactivity | Transitioning between flight selections, date picking, and generating the dynamic barcode/ QR code for the boarding pass must happen with zero noticeable delay. |
| System Reliability | Application uptime maintained at 90% that excluding scheduled maintance. |

## 3.6 System Architecture Diagram

The Smart Travel Application utilises a three-tier system design to enable flexibility, scalability, and consistent user experiences across platforms. The architecture has been divided into three layers which are Presentation, Application, and Data. Each layer is meant to handle certain tasks while working together to produce an integrated travel service platform.

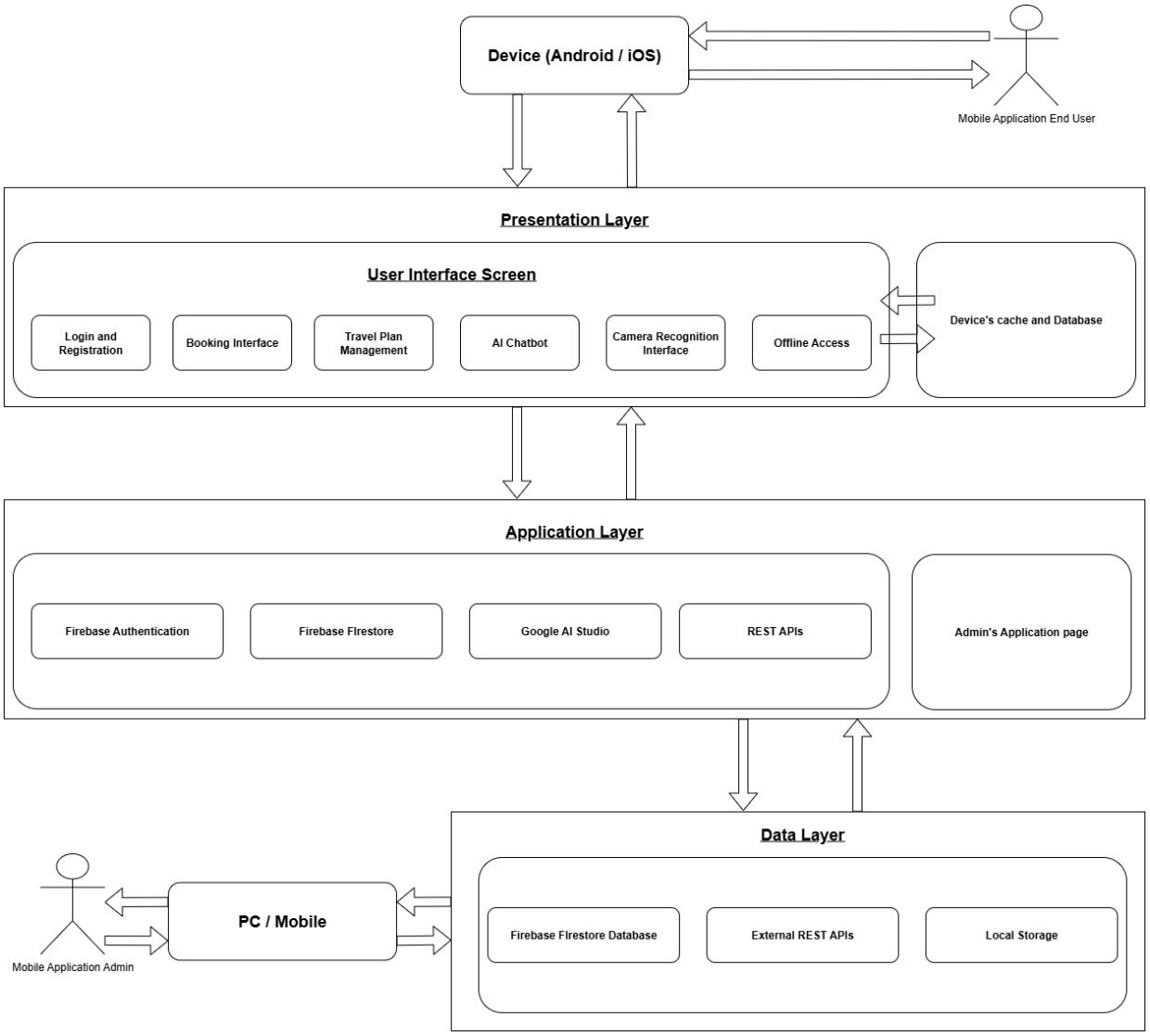



Figure 3.8 System Architecture Diagram

As shown in Figure 3.8, the Presentation Layer represent the user-facing component of the application. It developed with the Flutter framework to support both Android and iOS from a single codebase. Visual Studio Code serves as the basis for development while Android Studio Emulator is utilised to simulate and test mobile environments throughout the process. This layer includes mobile interfaces that allow user to securely log in and register, search and book

flights, hotels, transportation, manage travel plans, communicate with the AI chatbot for travel assistance, use camera-based landmark recognition features, and view saved travel plan and offline features. By combining these functionalities, the Presentation Layer guarantees that users have an accessible and responsive interface that is available even while offline.

The Application Layer serves as the functional core of the system. It is integrating backend services, AI processing, and external service integrations. This layer uses Firebase Authentication to manage secure user logins and registrations while Firebase Firestore saves user profiles, travel plan and booking information in real time. Google AI Studio powers the AI chatbot which is allowing for intelligent conversations with users and supporting multi language communication. At the same time, it provides camera-based detection of landmark and buildings to enhance the travel experience with real time information. External REST APIs are another important component of this layer which is providing up to date travel information such as flights, hotel availability, weather forecasts. To enhance the user experience, the business logic and AI predictive engine evaluate user data dynamically, optimising travel plan, and adjust it in reaction to potential interruptions such as flight delays or weather conditions. This layer ensure that all user requests are effectively handled and that accurate results are sent back to the presentation interface.

The Data Layer serve as the foundation for storage and real-time data management. Firebase Firestore acts as the core database, managing user accounts, booking records, and travel plan while providing secure and scalable access. The system also uses external REST APIs to get real-time information to ensure that users receive reliable data on travel schedules, weather. Additionally, the application integrates local storage to provide offline access to important functions such as saved travel plan. This ensures that service continues even in the lack of an internet connection. It allows travellers to access essential data during their travel.

The architecture is further strengthened by current security protocols such as authentication systems, authorisation controls, and encrypted communication channels. It ensures that user data and system activities are secure. Furthermore, the modular nature of this three-tier architecture emphasises scalability and future expansion. Also, it allows additional features such as advanced AI-powered travel recommendations or improved offline capabilities to be implemented without affecting the system’s basic structure.

## 3.7 Use Case Diagram

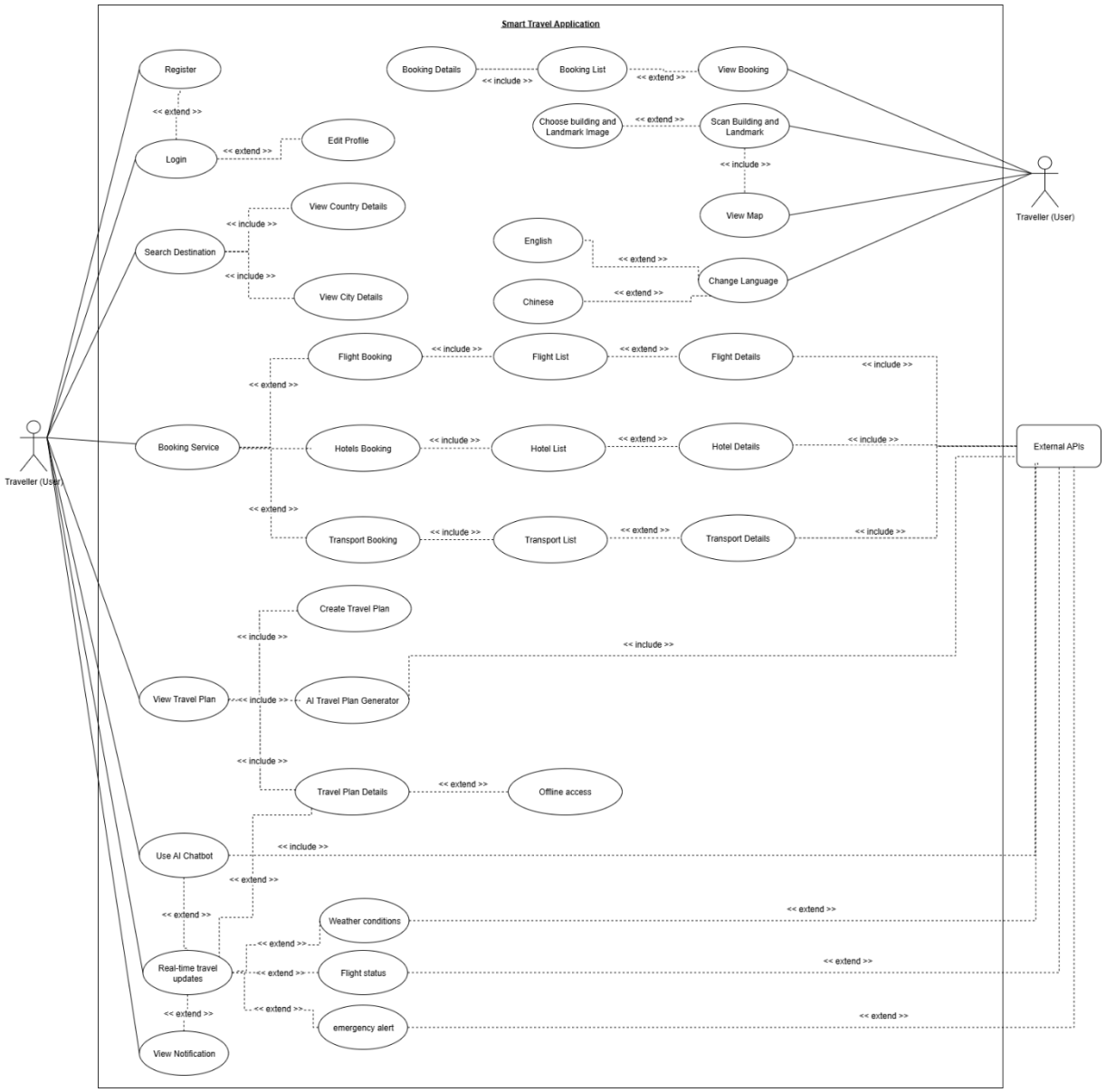



Figure 3.9 Use Case Diagram

### 3.7.1 Use Case Description

#### 3.7.1.1 Login

Table 3.5 Use case description – Login

| **Use Case Description** |
|---|
| **Name:** Login |
| **Brief Description:** The user authenticates themselves to access the system |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller<br>- Include: None<br>- Extends: Register<br>- Generalization: None |
| **Preconditions:**<br>1. User has a registered account in the system. |
| **Basic Flow:**<br>1. User clicks the “Login” option on the home screen.<br>2. System displays the login form.<br>3. User enters email and password.<br>4. User clicks the “Login” button.<br>5. System validates the credentials.<br>6. System grants access and navigates to home page. |
| **Alternate Flow:**<br>1. If user clicks the “Forgot Password” link<br>- System navigates to reset password flow. |
| **Exception Flow:**<br>1. Incorrect input (invalid email or password)<br>- System displays error message. |
| **Postconditions:**<br>1. User session is created.<br>2. User can now access system features. |

### 3.7.1.2 Register

Table 3.6 Use case description – Register

| **Use Case Description** |
|---|
| **Name:** Register |
| **Brief Description:** The user creates a new account to use the system. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller<br>- Include: None<br>- Extends: Login<br>- Generalization: None |
| **Preconditions:**<br>1. User has not registered before. |
| **Basic Flow:**<br>1. User clicks the "Register" option.<br>2. System displays the registration form.<br>3. User enters details (name, email, and password).<br>4. User clicks the "Submit" button.<br>5. System validates the inputs and creates new account.<br>6. System displays success message. |
| **Alternate Flow:** |
| **Exception Flow:**<br>1. Email already exists<br>- System displays error and prompts user to login. |
| **Postconditions:**<br>1. User account is created.<br>2. User can log in to the system. |

### 3.7.1.3 Search and Book Travel Service

Table 3.7 Use case description – Search and book travel service

| **Use Case Description** |
|---|
| **Name:** Search and Book Travel Service |
| **Brief Description:** The user searches for and books flights, hotels, and transportation. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller, Booking APIs (Flights/Hotels/Transport)<br>- Include: None<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User is logged into the system. |
| **Basic Flow:**<br>1. User selects booking type (flight/hotel/transport).<br>2. User enters search details (destination, dates, passengers).<br>3. System retrieves result from external APIs.<br>4. User selects preferred option.<br>5. Users confirm booking and proceed to payment. |
| **Alternate Flow:**<br>1. If no result is found, system suggests alternative dates or locations |
| **Exception Flow:**<br>1. Payment fails<br>-Booking is cancelled, user is prompted to retry. |
| **Postconditions:**<br>1. Booking is confirmed and stored in user's booking page.<br>2. Confirmation details are sent to user. |

### 3.7.1.4 AI Travel Plan Generator

Table 3.8 Use case description – AI travel plan generator

| **Use Case Description** |
|---|
| **Name:** AI Travel Plan Generator |
| **Brief Description:** The system generates a personalised travel plan based on user destination and durations. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller, AI Engine (External APIs)<br>- Include: None<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User has logged in and provided travel details (destination, duration) |
| **Basic Flow:**<br>1. User selects “Generate Travel Plan”<br>2. System analyses user selection<br>3. AI generates daily schedules with attractions, dining, landmark or building.<br>4. System displays travel plan to user. |
| **Alternate Flow:**<br>1. User modifies destination and durations<br>-System regenerates travel plan |
| **Exception Flow:**<br>1. If AI cannot generate travel plan due to no internet connection, system prompts user error message. |
| **Postconditions:**<br>1. Personalised travel plan is created.<br>2. Travel plan can be saved, edited. |

### 3.7.1.5 AI Chatbot Assistance

Table 3.9 Use case description – AI chatbot assistance

| **Use Case Description** |
|---|
| **Name:** AI Chatbot Assistance |
| **Brief Description:** The system provides real-time chatbot support for travel-related queries. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller, AI Engine (Google AI Studio)<br>- Include: External APIs<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User is logged in. |
| **Basic Flow:**<br>1. User clicks the chatbot icon.<br>2. System activates the chatbot window.<br>3. User types a query.<br>4. Chatbot processes the request using AI.<br>5. System displays the response in real time. |
| **Alternate Flow:**<br>1. User asks in another language<br>-System use translation before responding. |
| **Exception Flow:**<br>1. Chatbot cannot understand query<br>-Suggests user to ask again in another way |
| **Postconditions:**<br>1. User receives instant assistance<br>2. Query is logged for improvement of AI responses |

### 3.7.1.6 Offline Access

Table 3.10 Use case description – Offline access

| **Use Case Description** |
|---|
| **Name:** Offline Access |
| **Brief Description:** The system provides offline access to essential features such as saved travel plan. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller<br>- Include: None<br>- Extends: Saved Travel Plan<br>- Generalization: None |
| **Preconditions:**<br>1. User has previously saved travel plans. |
| **Basic Flow:**<br>1. User no internet connection.<br>2. System loads previously saved travel plan.<br>3. User can view and use the travel plan without internet. |
| **Alternate Flow:**<br>1. If network becomes available<br>-System syncs new updates automatically. |
| **Exception Flow:**<br>1. No offline data exists<br>-System prompts user to saved or download before travelling. |
| **Postconditions:**<br>1. User can continue using app without internet.<br>2. Offline changes are synced when internet is restored. |

### 3.7.1.7 Camera-based Landmark and Building Recognition

Table 3.11 Use case description – Camera-based landmark and building recognition

| **Use Case Description** |
|---|
| **Name:** Camera-based Landmark and Building Recognition |
| **Brief Description:** The user can identify landmarks/buildings using the device camera and receive relevant information. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller, AI Engine (External APIs)<br>- Include: None<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User grants camera permission.<br>2. User is logged in. |
| **Basic Flow:**<br>1. User opens camera recognition feature.<br>2. User points device camera at a landmark or building.<br>3. System scans the object using AI.<br>4. System displays name, description, historical facts, and tips. |
| **Alternate Flow:**<br>1. If the landmark and building not recognised<br>-System suggests search online option. |
| **Exception Flow:**<br>1. Camera permission denied<br>-System prompts user to enable it. |
| **Postconditions:**<br>1. User gains useful travel knowledge about the location. |

### 3.7.1.8 Real-Time Travel Updates

Table 3.12 Use case description – Real-time travel updates

| **Use Case Description** |
|---|
| **Name:** Real-Time Travel Updates |
| **Brief Description:** The system provides real-time notifications and adjusts travel plan for disruptions (flight delays, weather) |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Traveller, External APIs<br>- Include: Update Travel Plan<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User is logged in.<br>2. User has active travel plan.<br>3. System is connected to travel APIs |
| **Basic Flow:**<br>1. System checks flight status, weather, and other disruptions.<br>2. System notifies user of update.<br>3. User accepts new adjustment<br>4. System updates travel plan automatically |
| **Alternate Flow:**<br>1. User rejects auto-adjustment<br>-System keeps original travel plan but logs alert. |
| **Exception Flow:**<br>1. API failure<br>-System shows last known data with warning. |
| **Postconditions:**<br>1. User is notified of important travel event.<br>2. Updated travel plan is stored in system. |

### 3.7.1.9 View Map and Search Location

Table 3.13 Use case description – View map and search location

| **Use Case Description** |
|---|
| **Name:** View Map and Search Location |
| **Brief Description:** The system allows users to view an interactive map, search for destinations, and converts text input into geographical coordinates to display a visual marker. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, Nominatim API(OpenStreetMap)<br>- Include: None<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User has launched the application.<br>2. Device is connected to a stable internet network. |
| **Basic Flow:**<br>1. User navigates to the Map module.<br>2. User enters a destination name in the search field.<br>3. System executes a geocoding request to convert text into latitude and longitude coordinates.<br>4. System recenters the map camera to the retrieved coordinates.<br>5. System displays a visual marker at the exact location. |
| **Alternate Flow:**<br>1. Location not found.<br>-System retains the current map view and handles the search silently. |
| **Exception Flow:**<br>1. Network or API failure.<br>-System fails to load map tiles and displays a blank grid or network warning. |
| **Postconditions:**<br>1. Map View is updated successfully with the searched destination. |

### 3.7.1.10 Real-Time Currency

Table 3.14 Use case description – Real-time currency

| **Use Case Description** |
|---|
| **Name:** Real-Time Currency |
| **Brief Description:** The system provides a tool for users to convert between different international currencies in real-time based on local exchange rate definitions. |
| **Actor:** User |
| **Relationship:**<br>- Association: User<br>- Include: None<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User navigates to the Currency Converter page.<br>2. System has successfully loaded the currency exchange rates dictionary. |
| **Basic Flow:**<br>1. Users select the base currency and target currency from dropdown menus.<br>2. Users input a numerical amount into the text field.<br>3. System instantly applies the mathematical conversion formula using backend rates.<br>4. System updates and displays the converted target amount simultaneously. |
| **Alternate Flow:**<br>1. User clears the input field entirely.<br>-System automatically resets the converted amount to “0.00”. |
| **Exception Flow:**<br>1. Invalid user input.<br>-System input formatter automatically blocks non-numeric characters and multiple decimal points to prevent calculation crashes. |
| **Postconditions:**<br>1. The interface displays the accurate and formatted converted currency value. |

### 3.7.1.11 Real-Time Weather

Table 3.15 Use case description – Real-time weather

| **Use Case Description** |
|---|
| **Name:** Real Time Weather |
| **Brief Description:** The system fetches and dynamically displays live weather conditions, including temperature and animated weather icons for current location. |
| **Actor:** User |
| **Relationship:**<br>- Association: User, External Weather API (Open-Meteo)<br>- Include: None<br>- Extends: None<br>- Generalization: None |
| **Preconditions:**<br>1. User has launched the application and navigated it to the weather page.<br>2. Device is connected to a stable internet network. |
| **Basic Flow:**<br>1. Users view the home screen weather banner or detailed weather page.<br>2. System identifies the current location coordinates.<br>3. System sends an asynchronous HTTP request to the external weather API.<br>4. System extracts temperature and weather condition codes from the returned JSON payload.<br>5. System renders the corresponding weather data and triggers dynamic animations on the UI. |
| **Alternate Flow:**<br>1. Location change.<br>-System re-fetches data and updates the UI for the newly assigned location. |
| **Exception Flow:**<br>1. API requests timeout or lack of internet.<br>-System displays ”Updating…” or “Locating…” placeholders to manage user expectations without crashing the app. |
| **Postconditions:**<br>1. Current weather status and related animated icons are successfully displayed to the user. |

## 3.8 Feasibility of the Proposed Method

The preliminary implementation of the Smart Travel Application demonstrates that the proposed method is technically and practically feasible for large-scale development. The practicality is backed by the following factors:

### 3.8.1 Technical Feasibility

The application was successfully constructed using the Flutter and tested on the Android Studio Emulator (Pixel 8a) and a developer workstation without performance issues. The AI integration with Google AI Studio's Gemini API worked properly for both chatbot and travel plan generation as validated through successful test cases. Offline accessibility using local storage (SharedPreferences) functioned as expected. This is ensuring that saved plans remained active even when there was no internet connection. The modular architecture separated important features which reduced complexity and improved maintainability such as User Management, AI travel planning, and Booking Integration.

### 3.8.2 Operational Feasibility

During preliminary testing, the user interface was found to be quick and straightforward with easy navigation between modules (Home, My Trips, Chatbot). The system's error-handling methods ensured smooth user experience even in bad conditions such as offline, API failure fallbacks and empty input validations. Since the application supports both online (AI travel planning, chatbot, booking) and offline (saved travel plans) modes, it is highly suitable for real-world travel scenarios.

### 3.8.3 Economic Feasibility

During testing, open-source frameworks (Flutter, Dart) and free tiers of the Google AI Studio API were employed to reduce the cost. The necessary equipment and software (laptop, emulator, and IDEs) are common and already available. It is reducing the need for further expenses during the first phase. Future expenses will primarily be used for increasing API usage and potential app store deployment and it both manageable within FYP scope.

### 3.8.4 Timeline – Gantt Chart

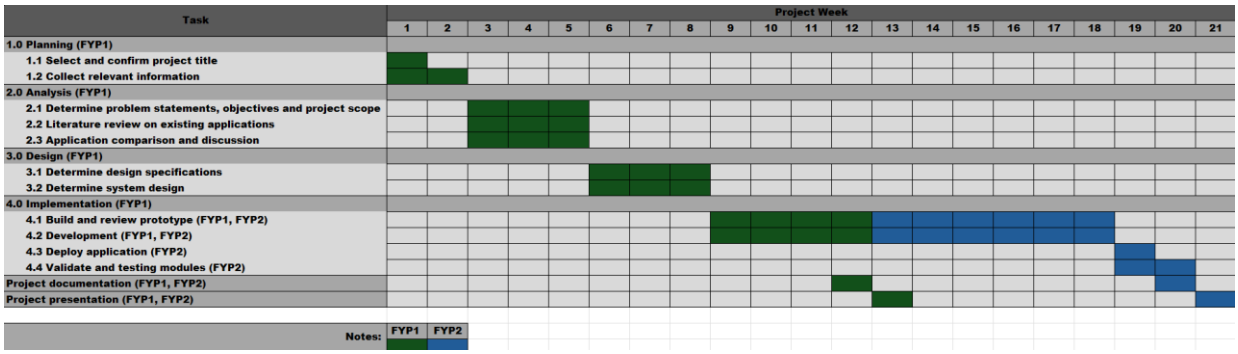



Figure 3.10 Timeline – Gantt Chart

As shown in Figure 3.10, the Gantt chart for the development of the Smart Travel Application is covering both FYP1 and FYP2 phases. During **weeks 1 – week 2 of FYP1**, the project beings with the planning stage which involves selecting a project title and gathering essential information. This is following by the analysis phase in **week 3 – week 5**, which involves defining the problem statement, objectives, and project scope, conducting a literature view on existing applications, and comparing related solutions. The design phase is conducted in **week 6 – week 8,** which focus on determining the design specifications and preparing the system design, including flowcharts, architecture, and requirements. Implementation activities are initiated in **week 9 – week 12** where the prototype was developed and reviewed alongside the initial core features. The documentation is completed in **week 12**, and the FYP1 presentation will take place in **week 13**.

For FYP 2, the project will continue to develop from **week 13 – week 18** to improve and expanding on the prototype. The application will be deployed in **week 19** with features validation and testing taking place in **week 19- week 20** to ensure functionality and reliability. The documentation will be complete in **week 20** and the project concludes with the final presentation in **week 21**. This timeline provides a structured and progressive approach to ensure that the project advances systematically from planning through analysis, design, and implementation. This is allowing sufficient time for refinement, testing, and documentation.

# CHAPTER 4

# SYSTEM DESIGN

## 4.1 Module Segmentation

The Smart Travel Application is divided into several interconnected sections. This modular architecture allows each component to be developed, tested, and maintained separately while also functioning together to provide a smooth travel experience. The main modules are as follows:

Table 4.1 Module segmentation

| Module | Purpose | Core functions | Interaction |
|---|---|---|---|
| User Management | - Handles all user account functions (Registration, login, authentication, profile management) | - Secure login and logout using Firebase Authentication.<br>- Store user travel history.<br>- Provide personalisation by linking user profiles to travel plan and booking data. | - Connects with AI travel planning and booking integration modules to fit results for individual users. |
| AI travel planning | -Provide personalised, AI travel plan fit to user destination and duration. | -Generate travel plan with destination, duration.<br>- Apply predictive analytics for disruption handling<br>- Allow users to edit and save travel plan | - Communicates with Real-Time updates and booking Integration Modules to adjust travel plan dynamically |
| AI Chatbot Assistance | - Provide real-time conversational support and guidance throughout the journey. | - Natural Language Processing (NLP) for multi-language queries.<br>- Answer travel-related questions interactively.<br>- Provide quick access to travel plan, bookings, and updates. | - Serve as a communication that interacts with almost all modules to fetch or update information based on user input. |
| Camera-based landmark and building recognition | Enhance user exploration using computer vision. | Recognise landmarks or buildings through the device camera.<br>Provide contextual details | Connects with Offline Module to Provide cached information if no internet is available. |

| | | | |
|---|---|---|---|
| | | such as history, opening hours, and visitor suggestions. | |
| Booking Integration | Centralises booking services within the application. | Integrate APIs<br>Allow users to search, compare, and book flights, hotels, and transport.<br>Manage booking records within the application. | Supplies travel plan details to AI planning Module and provide booking confirmation to User Management. |
| Real-Time Updates | Keeps users informed about live travel disruptions | Collect data from APIs for flights status, weather, and local alerts.<br>Send push notifications to users regarding travel plan adjustment. | Works closely with AI travel planning to automatically reschedule travel plans. |
| Offline Access | Provides reliability in areas with limited connectivity. | Store travel plan.<br>Allow offline access and synchronise when network become available | Supports AI planning |
| Interactive Map Integration | Visualise travel destinations, routes, and points of interest geographically. | - Render interactive map interfaces using spatial data.<br>- Perform geocoding to search and pinpoint specific landmarks or addresses.<br>- Display custom markers based on user search or saved travel locations. | Supports landmark recognition module |
| Real-Time Weather Updates | Provide current atmospheric conditions and predicts to ensure safe and prepared travel. | - Fetch live meteorological data (temperature, weather conditions) via external endpoints.<br>- Display dynamic and animated weather widgets on the user dashboard. | Complements the AI travel planning module by allowing users to review and adjust their daily itineraries based on incoming weather changes. |

| | | | |
|---|---|---|---|
| Real-Time Currency Conversion | Assist users with financial planning and budget management during international travel. | - Retrieve live foreign exchange rates from external financial APIs.<br>- Perform instant mathematical conversions between base and target currencies.<br>- Support a wide range of global currencies. | Operates within the utility dashboard and can provide financial context for the Flight Booking module to help users estimate costs. |

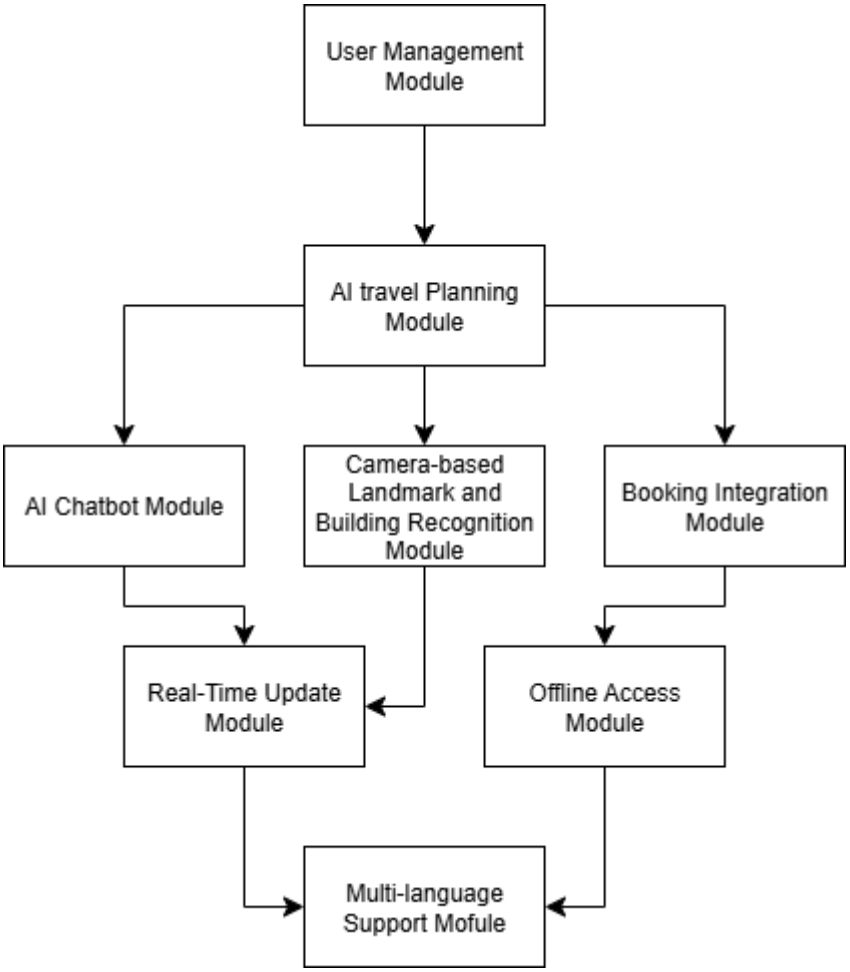



Figure 4.1 Module Segmentation Diagram

As shown in Figure 4.1, it presents the major modules and their interactions. The User

Management Module is at the top and connected to the AI travel planning Module, which is then linked to AI Chatbot Assistance, Camera-based Landmark and Building Recognition, and Booking Integration. The Real-time Updates and Offline Access Modules improve flexibility and reliability while the Multi-Language Support Module enhances the system's accessibility.

### 4.1.1 Summary of module segmentation

The Smart Travel Application is designed using modular interaction to provide flexibility and scalability. The User Management Module enhances the overall experience by linking user data to booking records and travel plan development. The AI travel planning module serves as the system's fundamental engine, combining real-time information, booking services, and personalised features to create dynamic travel plans. The AI Chatbot Module serves as the principal communication gateway that allow users to easily access and engage with system functionalities. To ensure reliability and adaptability, the Offline Access Module and Real-time Updates Module collaborate to deliver continuous services regardless of network availability. Meanwhile, the multi-language Support Module and Camera-based Landmarks and Building Recognition Module promote quality and enhance the user's travel experience by providing multi-language accessibility and contextual destination information. This modular segmentation enhances scalability that makes maintenance easier and ensures clear functional separation which is required for future system enhancement.

## 4.2 System Design Diagram – Flowchart

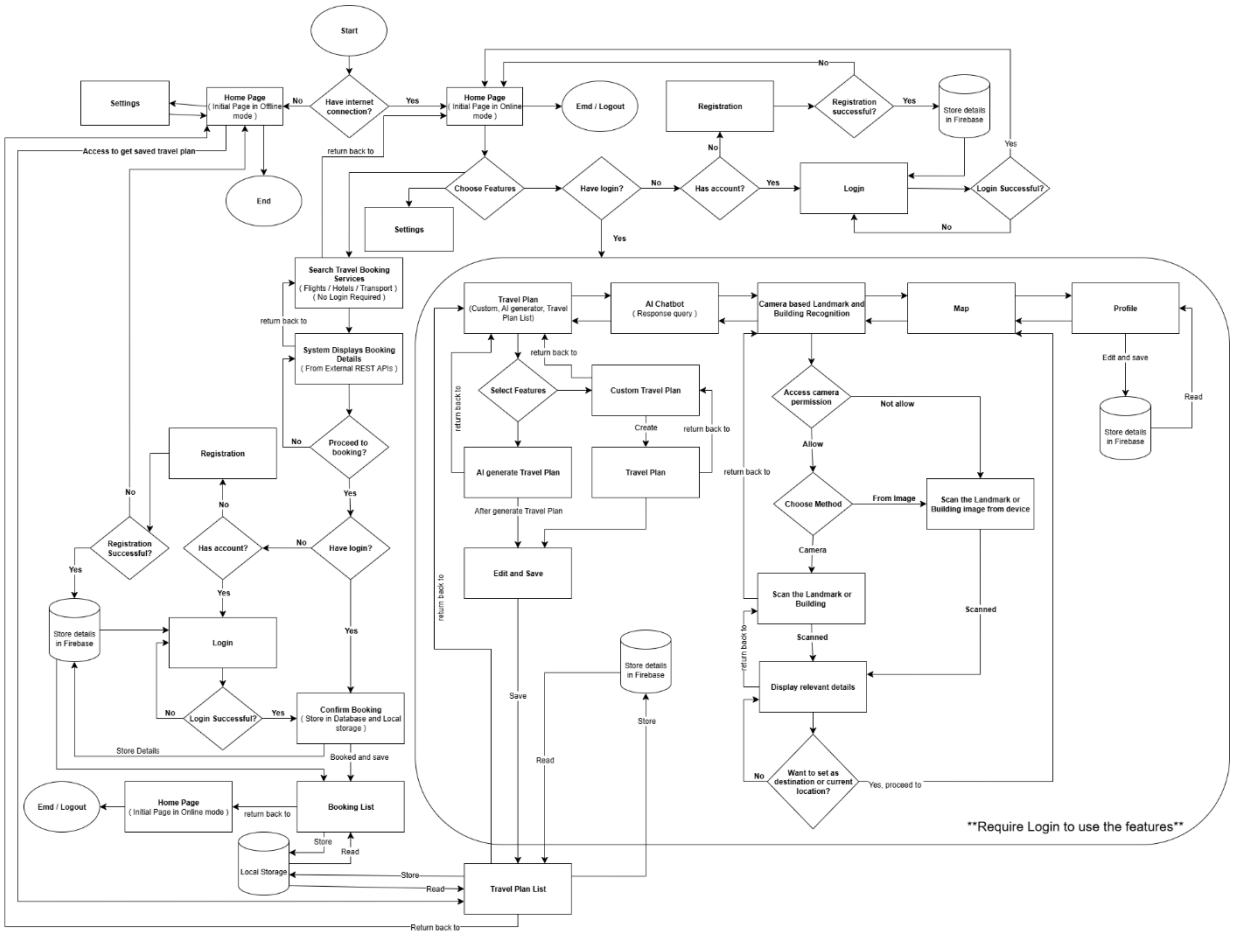



Figure 4.2 System Design Diagram – Flowchart
(clearer refer to appendices)

As shown in Figure 4.2, it demonstrates the system flowchart of the Smart Travel Application which showing the logical workflow from the starting to service execution. The procedure starts with homepage where the system checks for internet connectivity. If an internet connection is available, the application provides complete access to online features such as booking services, AI generated travel plans, AI chatbot, and camera-based landmark and building recognition. In a shortage of connectivity, the system changes to offline mode that allow user to recover and manage previously saved travel plan and essential resources stored locally on the device.

User authentication is handled through registration and login functions with credentials securely checked using Firebase Authentication. Once verified, users can access different features based on their needs. The booking features uses external REST APIs to obtain real-time data on flights, hotels, and transportation. Confirmed reservations are shown to the user and saved in Firebase Cloud Firestore with a mirrored copy kept in local storage for offline access.

The travel plan feature enables users to generate a personalised travel plan or generate an AI based plan using Google AI Studio which considers destination, duration. Furthermore, generated plans are personalised and stored to the travel plan list. It allows user to manage multiple travel plan conveniently.

The AI chatbot feature provides real-time assistance by using Google AI Studio to process and respond to user enquiries about destinations, bookings, or travel adjustment. In addition, the camera-based recognition features let users scan landmarks and building using a live camera feed or by uploading an image. Computer vision models detect landmark and building to show relevant details such as history, business hours, and visitor suggestions. Users can also include these landmarks and buildings as destination on the map. Throughout the process, the application ensures data synchronisation across online and offline modes with Firebase serving as the cloud backend and local storage ensuring availability when offline. The workflow ends when users save their changes, log out or exit the application. This structured design guarantees that all system components work together to deliver a smooth, intelligent, and adaptable travel experience.

## 4.3 Database Design

The database design for the Smart Travel Application was developed to handle essential modules including User Management, Travel Plans, and Booking Records. The design is to ensure that user information, generated travel plans, and booking details are effectively maintained while ensuring data integrity and security.

### 4.3.1 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) demonstrates the logical structure of the database and the relationship between entities.

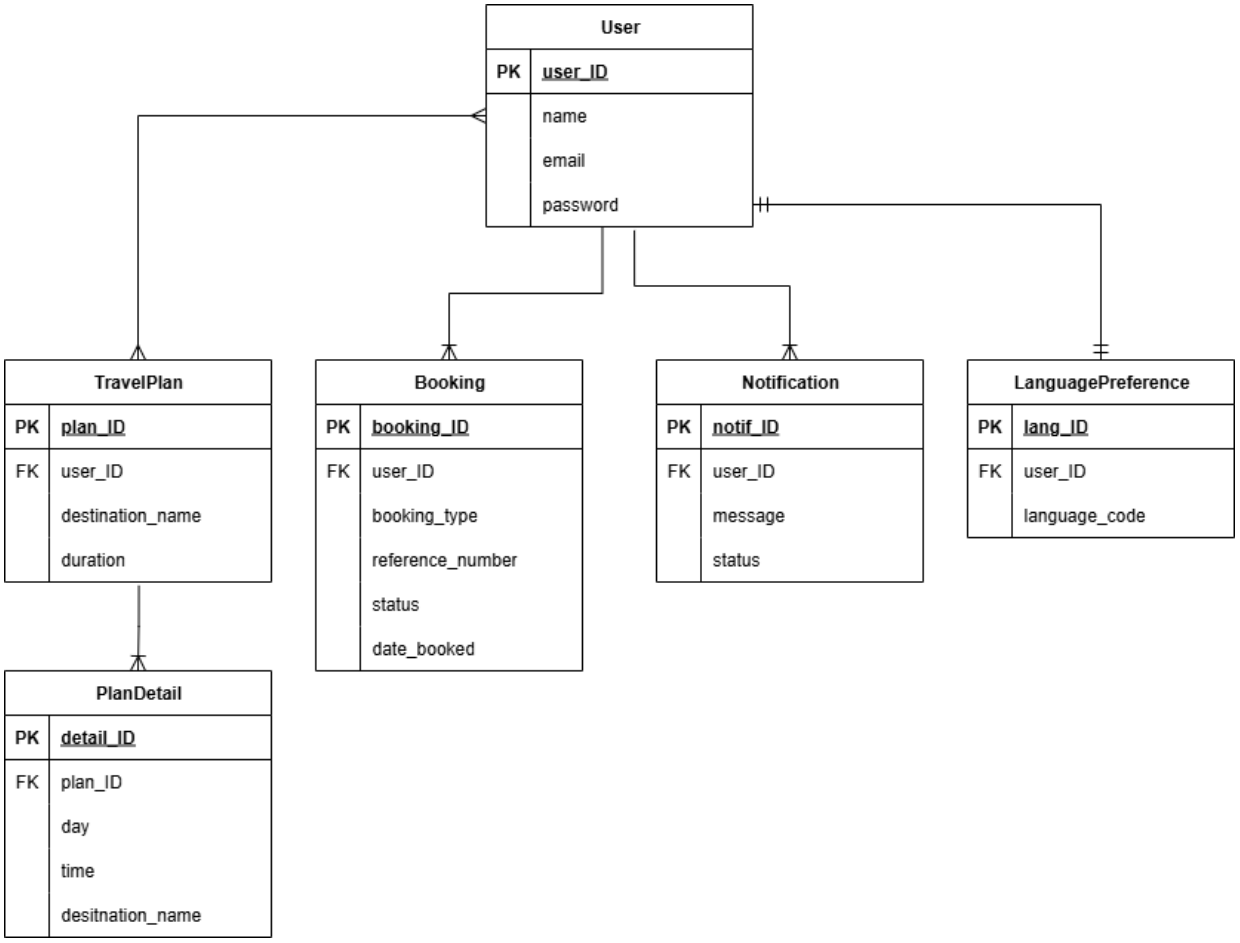



Figure 4.3 Entity Relationship Diagram (ERD)

### 4.3.2 Data Dictionary

**User table**

Table 4.2 Data dictionary – User table

| Attribute Name | Type | NULL | Default | Primary or Foreign Key | Foreign Key Referenced Table |
|---|---|---|---|---|---|
| user_id | INT | NO | NULL | Primary Key | |
| name | VARCHAR (255) | NO | NULL | | |
| email | VARCHAR (255) | NO | NULL | | |
| password | VARCHAR (255) | NO | NULL | | |

**TravelPlan Table**

Table 4.3 Data dictionary – TravelPlan table

| Attribute Name | Type | NULL | Default | Primary or Foreign Key | Foreign Key Referenced Table |
|---|---|---|---|---|---|
| plan_ID | INT | NO | NULL | Primary Key | |
| user_id | INT | NO | NULL | Foreign Key | User (user_ID) |
| destination_name | VARCHAR (255) | NO | NULL | | |
| duration | INT | NO | NULL | | |

**PlanDetail Table**

Table 4.4 Data dictionary – PlanDetail table

| Attribute Name | Type | NULL | Default | Primary or Foreign Key | Foreign Key Referenced Table |
|---|---|---|---|---|---|
| detail_ID | INT | NO | NULL | Primary Key | |
| plan_id | INT | NO | NULL | Foreign Key | TravelPlan(plan_ID) |
| day | INT | NO | NULL | | |
| time | TIME | YES | NULL | | |
| destination_name | VARCHAR (255) | NO | NULL | | |

**Booking Table**

Table 4.5 Data dictionary – Booking table

| Attribute Name | Type | NULL | Default | Primary or Foreign Key | Foreign Key Referenced Table |
|---|---|---|---|---|---|
| booking_ID | INT | NO | NULL | Primary Key | |
| user_id | INT | NO | NULL | Foreign Key | User (user_ID) |
| booking_type | VARCHAR (50) | NO | NULL | | |
| reference_number | VARCHAR (100) | NO | NULL | | |
| status | VARCHAR (50) | NO | ‘pending’ | | |
| date_booked | DATETIME | NO | CURRENT_TIMESTAMP | | |

### Notification Table

Table 4.6 Data dictionary – Notification table

| Attribute Name | Type | NULL | Default | Primary or Foreign Key | Foreign Key Referenced Table |
|---|---|---|---|---|---|
| notif_ID | INT | NO | NULL | Primary Key | |
| user_id | INT | NO | NULL | Foreign Key | User (user_ID) |
| message | TEXT | NO | 'unread' | | |
| status | VARCHAR (50) | NO | 'pending' | | |

### LanguagePreference Table

Table 4.7 Data dictionary – LanguagePreference table

| Attribute Name | Type | NULL | Default | Primary or Foreign Key | Foreign Key Referenced Table |
|---|---|---|---|---|---|
| lang_ID | INT | NO | NULL | Primary Key | |
| user_id | INT | NO | NULL | Foreign Key | User (user_ID) |
| language_code | VARCHAR(10) | NO | 'en' | | |

## 4.4 System Components Specifications

The Smart Travel Application was developed as a purely software-based solution. Therefore, the system components are defined as the core software modules, frameworks, and external services that interact to form the complete ecosystem.

The Frontend Presentation Component (Flutter Framework) serves as an IOS and Android cross-platform client application. It is designed to manage local state, user interactions, and all User-Interface (UI) rendering. Key technical sub-components include flutter_map for geographic visualisation, custom UI clippers for ticket styled layouts, and local caching mechanisms (shared_preferences) to support offline accessibility for saved travel plan and booking history.

The Backend and Authentication Component (Firebase Ecosystem) serve as a cloud-based component to responsible for data persistence, backend logic, and security. It consists of Firebase Authentication which manages secure user sessions. It supports both standard email and password, and Google Sign-In. The Cloud Firebase is a highly scalable NoSQL database specified to manage unstructured user profiles and real-time flight booking transactions securely.

AI Processing Component serve as the intelligent core of the system. The Google Gemini API is specified to process natural language inputs and execute complex logic. It receives dynamically constructed prompts containing specific travel parameters from the frontend, processes the constraints, and returns highly personalised multiple day travel plan formatted as JSON arrays such as destination country, city, and duration.

Real-time data and service component (RESTful APIs) is to ensure the application provides up-to-date travel assistance without increasing the local application size. Several specialised external RESTful API components are integrated such as Nominatim API (OpenStreetMap), Open-Meteo API, ExchangeRate API. Nominatim API was specified for asynchronous geocoding, accurately converting textual destination queries or landmark names into exact latitude and longitude coordinates. Open-Meteo Api was specified to synchronously fetch real time atmospheric data to drive the dynamic weather animations on the UI such as temperature and specific weather condition codes. However, ExchangeRate API was specified to fetch live global financial rates. It enables instantaneous and accurate currency conversion computations.

The System Utility Components represent background modules essential for system integrity and user engagement. It includes the local notification engine (flutter_local_notifications) for triggering offline flight delay alerts, and the Network Time

Protocol (NTP) component. It fetches secure, real-world timestamps to prevent users from manipulating device time during booking interactions.

## 4.5 Database and UI Component Design

This section delineates the architectural logic of the data persistence layer and the visual framework of the application's user interface.

### 4.5.1 Database design (Cloud and Local)

The application utilises a dual-storage method to balance real-time synchronisation and offline reliability. Cloud Firestore (NoSQL architecture) serve as a document-oriented database. Firestore is utilised for its scalability and real-time capabilities. The data is organized into three primary collections which are users' collection, booking collection, and notifications collection. For users' collection, it stores unique user profiles linked by Firebase UID. Fields include display_name, email, member_since, and profile_url (referencing Firebase Storage for images). For bookings collection, it manages transaction records for flights. Each document encapsulates flight parameters such as airline, flight_no, origin_code, destination_code, passenger_count, and class_type. For the Notifications collection, it stores historical flight alerts and system messages. It enables cross-device synchronisation of user alerts.

The Local Persistence (Shared Preferences) is for performance optimisation. The system utilises shared_preferences to cache the offline travel plan and system settings. For offline travel plan, AI generated travel plans are serialised into JSON strings and stored locally. It ensures users can access their schedules without an active internet connection. For system settings, the system stores the user preferences such as notification_enabled and theme configurations.

### 4.5.2 UI Component Design and Design System

To ensure a professional and cohesive user experience, the application follows a strict Design System tailored for the travel industry. A professional high-contrast scheme serves as the UI's basis. Primary Navy stands for the travel services business identity and trust. Secondary Ice Blue used for backgrounds to reduce visual exhaustion and enhance readability.

A specialised ticket-style containers (CustomClipper) class is implemented in boarding pass and booking summary to render a physical ticker aesthetic. It completes with dashed lines and semi-circular cutouts reinforcing the travel theme.

A dynamic weather banner located on the home dashboard. This component utilises TickerProviderStateMixin to provide smooth floating animations for weather icons. It makes visually reflecting real-time atmospheric conditions fetched from the Open-Meteo API.

The AI chat interface was designed with distinct asymmetrical message bubbles (rounded

corners) to differentiate between user input and Gemini AI responses. It ensures a natural conversational flow.

The interactive map layer was integrated using flutter_map. This component features custom marker overlays and polyline rendering to visualise travel routes and landmark locations dynamically.

## 4.6 System Components Interaction Operations

This section describes the operational flow and data exchange between the system's core components during its primary functionalities. The interactions have been carefully designed to utilise asynchronous processing, preventing any user interface freezing during complex cloud communications. It is ensuring a seamless user experience.

### 4.6.1 Interaction Flow: AI-powered Travel Plan Generation

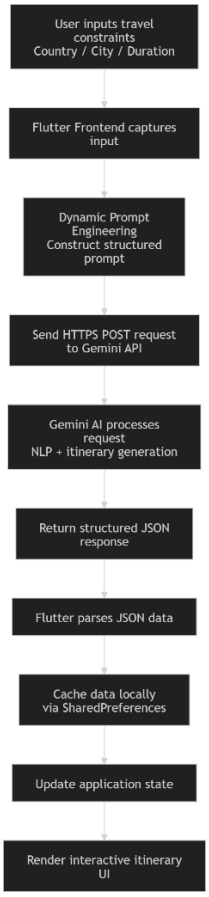



Figure 4.4 AI-powered Travel Plan Generation

The operational sequence for generation intelligent travel plan begins when the Frontend Component captures the user's travel constraints through the Flutter interface such as country, city, and duration. Upon receiving the input, the system performs dynamic prompt engineering and embedding these constrains into a structured text prompt along with strict system-level instruction that demand a JSON output. The application initiates an asynchronous HTTPS POST request to the AI processing Component (Google Gemini API) securely transmitting the

generated prompt along with the required API authorisation keys. The Gemini engine then applies natural language processing to evaluate the constraints, synthesises a logical daily schedule, and formulates a structured JSON response array. Once the API successfully return this payload the Flutter fronted intercepts and serialises the data immediately caching it within the local device storage via the shared_preferences module. This crucial data persistence step ensures that the generated travel plans remain fully accessible even in offline scenarios. Finally, the application updates its internal state and parsing the locally saved JSON to render dynamic and interactable schedule cards on the user interface (UI).

### 4.6.2 Interaction Flow: Real-time Flight Booking and Alert System

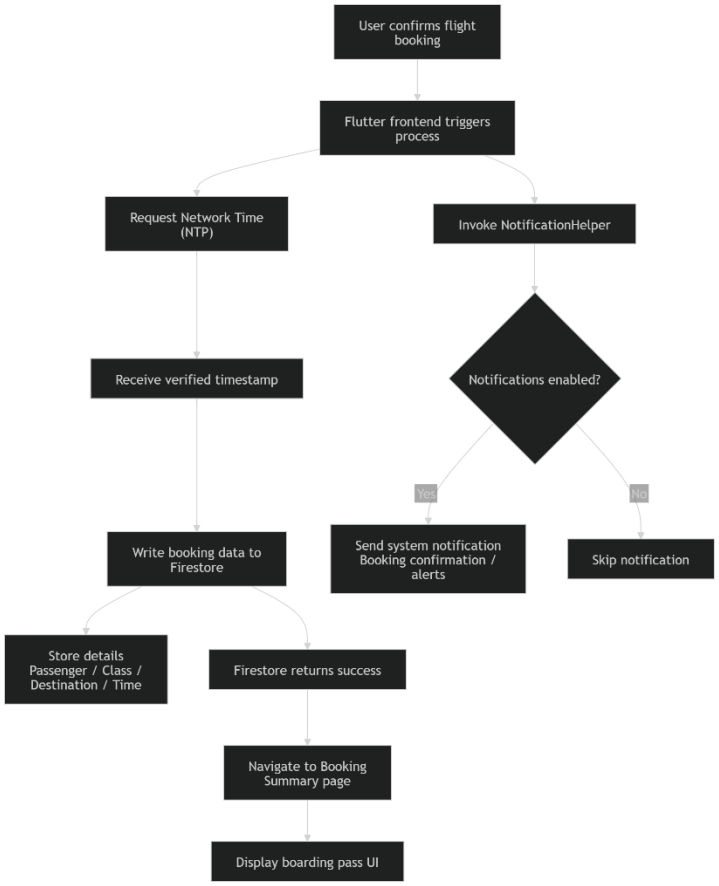



Figure 4.5 Real-time Flight Booking and Alert System

The transactional workflow for the flight booking module involves a highly synchronised interaction between the frontend, backend, and background utility components. The process is initiated when the user reviews their flight selection and triggers the confirmation action within

the Frontend Component. To maintain the absolute integrity of the transaction timestamp and prevent any manipulation of the local device clock, the system immediately communicates with the Network Time Protocol (NTP) Utility Component to fetch a secure, real-world network time. Equipped with this validated timestamp, the application performs a cloud synchronisation handshake with the Backend Component (Cloud Firestore). It actively writes a new transactional document into the bookings collection and encapsulating crucial payload data such as the flight class, passenger count, destination codes, and the NTP timestamp. Concurrently, the application invokes the local NotificationHelper that representing the System Utility Component. This module performs a conditional verification of the user's local preferences. If the notifications_enabled flag is true, it schedules and dispatches an immediate system level banner alert to confirm the booking or simulate a flight delay update. Once the Firestore database returns a success callback for the write operation, the system completes the transaction loop by navigating the user to the Booking Summary page. It renders a graphically rich and custom-clipped boarding pass interface.

# CHAPTER 5

# SYSTEM IMPLEMENTATION

## 5.1 System Requirement

The system requirement phase describes the technical setup and tools that required for the successfully development of the Smart Travel Application. Therefore, this section describes both the hardware and software environment that utilised throughout the project.

### 5.1.1 Hardware Requirements

The following table provides the hardware specifications that used for developing and testing the Smart Travel Application.

Table 5.1 Hardware requirements

| **Hardware** | **Specifications** |
|---|---|
| **Development Laptop** | ASUS TUF Gaming A15 |
| **Processor** | AMD Ryzen 7 |
| **Memory (RAM)** | 16GB DDR4 |
| **Storage** | 512GB SSD |
| **Graphics** | NVIDIA GeFroce GTX 1650Ti 4GB |
| **Operating System** | Windows 11 |

### 5.1.2 Software Requirements

The following table provides the hardware specifications that used for developing and testing the Smart Travel Application.

Table 5.2 Software requirements

| **Software** | **Specifications** |
|---|---|
| **Operating System** | Windows 11 |
| **Development Framework** | Flutter |
| **Programming Language** | Dart Language |
| **Mobile Development Integrated Development Environment (IDE)** | Visual Studio Code |
| **Testing** | Android Emulator, Physical Android Device, Flutter DevTools |

## 5.2 Hardware Setup

The hardware requirements for developing and testing the Smart Travel Application were fundamental with primary goal of providing an effective development environment and testing devices. The following hardware configurations were used during the preliminary work:

### 5.2.1 Development Computer

The application was developed on an ASUS TUF Gaming A15 laptop running Windows 11 with an AMD Ryzen 7 CPU, 16GB DDR4 RAM, 512 GB SSD storage, and an NVIDIA GeForce GTX 1650 Ti (4GB) dedicated graphics card. This arrangement gave more than enough performance for concurrently running the Flutter SDK, Visual Studio Vode, and Android Studio emulator which enabled smooth development, testing, and integration of the Smart Travel Application.

### 5.2.2 Android Emulator

To test the application during the fundamental development phase, an Android Pixel 8a simulator was set up in Android Studio. The emulator provides a controlled environment for evaluating the UI's responsiveness, navigation flows, and AI chatbot integration.

### 5.2.3 Mobile Device (Future Testing)

While initial testing was done on the Pixel 8a emulator, a hardware Androidvor iOS smartphone will be utilised in the future for real-world deployment and validation of offline functionalities. This ensures that the application runs successfully on actual hardware.

### 5.2.4 Internet Connection

A reliable internet connection was necessary for downloading dependencies, integrating the Google AI Studio API, and testing live features such as AI chatbot query and online booking services.

### 5.2.5 USB Cable

To deploy on physical devices, a USB cable will be utilised to connect the Android smartphone to the development PC. It is allowing for direct debugging and application installation.

## 5.3 Software Setup

Before beginning development of the Smart Travel Application, several tools and frameworks were installed and configured to provide an appropriate development environment. The following actions were implemented to prepare for the development process:

### 5.3.1 Flutter SDK Installation

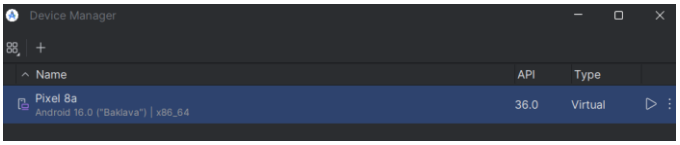
- Go to **Flutter official site** and download **Flutter SDK** for windows
- Then extract to **c:\src\flutter** and add Flutter to **Environment Variables (PATH),** and open **terminal** in VS code and type**:**

```
C:\Users\User\Desktop\smart_travel_app>flutter doctor
```

The Flutter SDK was installed on the development workstation to allow for cross-platform mobile application development. The installation contained Dart language which is require for Flutter development. The Flutter environment variables were configured to allow commands to be executed from the terminal.

### 5.3.2 Android Studio Setup

- **Download and install** Android Studio
- Then **open Android Studio** and go to **Device Manager**.
- Create a **new virtual device** then select **Pixel 8a**
- Choose **API level with Android 16**.

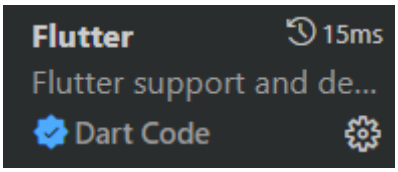



- Launch the emulator

Android Studio was installed to enable emulation, debugging, and Flutter integration. During the preliminary development stage, a Pixel 8a simulator served as the Primary virtual device for testing application functionalities.

### 5.3.3 Visual Studio Code (VS Code)

- **Download and install** VS Code
- Open **VS Code** and go to **Extensions Marketplace**, then install **Flutter** and **Dart extensions**.




Visual Studio Code was chosen as the primary integrated development environment (IDE) for its lightweight performance and strong extension ecosystem. The Flutter and Dart plugins were installed to provide syntax highlighting, debugging and smooth connection with the Flutter SDK.

### 5.3.4 Google AI Studio Configuration

- Go to **Google AI Studio** and **sign in with Google account**.
- Create **a new project** and **generate an API key**, then **copy the API key** into the flutter project environment.

Google AI studio was developed to provide the AI powered features of application especially the AI chatbot and the AI driven travel plan generator. An API key was produced and securely saved. It is allowing interaction with the Gemini Large Language Model (LLM) for natural language processing and personalised travel content development.

## 5.4 Setting and Configuration

The Smart Travel Application relies heavily on device-specific configurations and external API integrations to deliver its full suite of features. This section outlines the environment settings, application permissions, and API configurations required for the system's operation.

### 5.4.1 Application Permissions and Manifest Configuration

To allow the Flutter application to interact with hardware components and external networks, specific permissions were declared in the native configuration files.

Android Configuration (AndroidManifest.xml):

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
```

The following permissions were explicitly added to ensure the application could fetch live data, render maps, and push system notifications. Additionally, the application requires the POST_NOTIFICATIONS permission (introduced in Android 13) to allow the flutter_local_notifications plugin to trigger flight delay alerts locally.

### 5.4.2 External API Integrations and Package Settings

The system's core intelligence and real-time data rely on external endpoints. The configurations for these services were established within the Flutter environment:

#### 5.4.2.1 Google Gemini API Configuration

The google_generative_ai package was integrated to facilitate communication with the Gemini model. A secure API key generated via Google AI Studio was initialized within the application's service layer. The model instance was specifically configured to use gemini-pro, optimized for text-based prompt processing and JSON generation for travel itineraries.

#### 5.4.2.2 Geocoding and Map Settings (OpenStreetMap)

The flutter_map package was configured as the primary rendering engine for the map interface. To comply with OpenStreetMap's usage policies, a custom User-Agent header was implemented during HTTP GET requests to the Nominatim API. It is ensuring reliable and uninterrupted geocoding services (converting landmark text to latitude and longitude coordinates).

#### 5.4.2.3 Weather and Currency Endpoints

The http package was configured to manage asynchronous RESTful requests. Base URLs for the Open-Meteo API (for weather data) and the local exchange rate definitions were defined as global constants. It is allowing the system to dynamically append query parameters (such as target coordinates or currency codes) based on user interactions.

### 5.4.3 Local Storage and State Configuration

To support the application's offline capabilities outlined in the system requirements, the shared_preferences package required specific initialization. Before executing any read or write operations, the asynchronous SharedPreferences.getInstance() method is invoked within the repository layer. This configuration allows the system to establish a secure link to the device's native local storage (XML files on Android and NSUserDefaults on iOS). It is enabling the caching of generated travel plans and booking histories for offline retrieval.

## 5.5 System Operation

### 5.5.1 Home Page

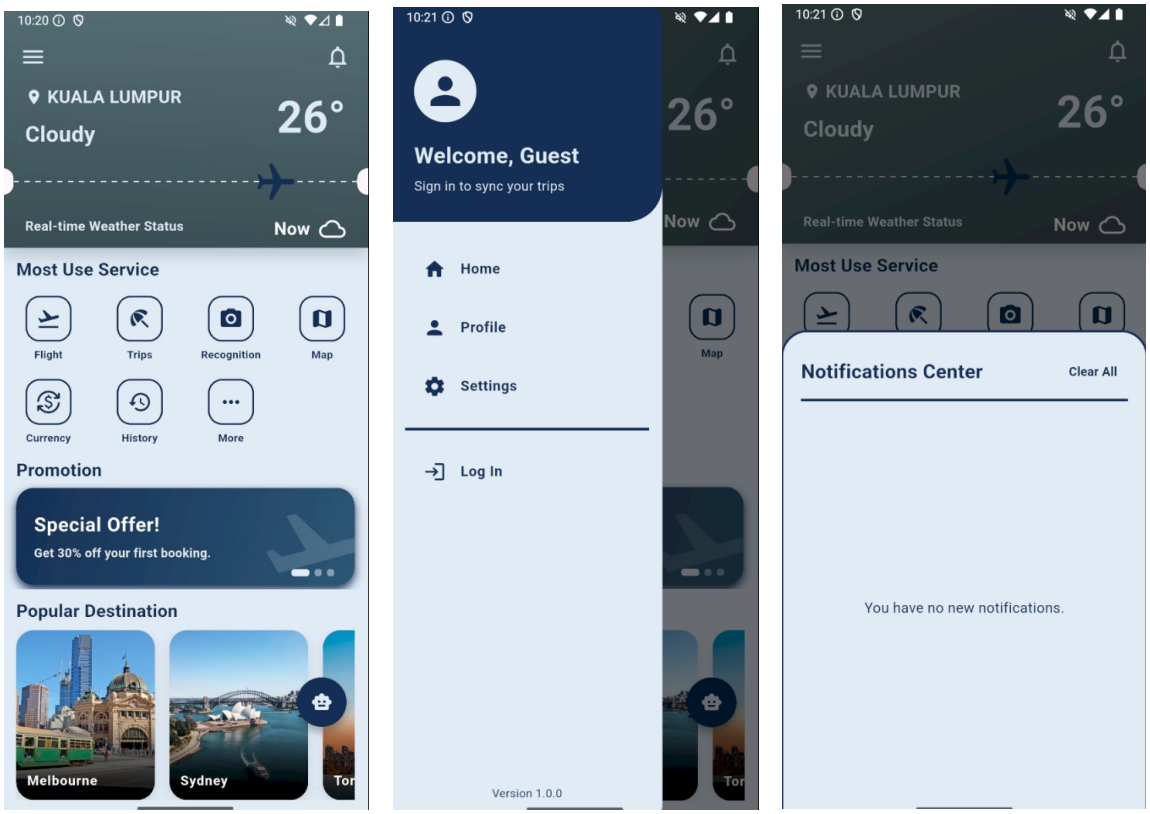



Figure 5.1 Home Page

The Home Page of the Smart Travel Application is designed as the main entry point that enables users to efficiently access all core features in a single interface. At the top, a custom AppBar displays the current location and real-time weather information, allowing users to stay informed about travel conditions. A menu icon is positioned on the top left which opens the navigation drawer, while a notification icon on the top right provides quick access to alerts such as flight delays or weather interruption.

The navigation drawer offers secondary options including Home, Profile, Settings, and Login. This ensures that users can easily manage their accounts and personalize their experience within the application. The drawer layout is simple and intuitive, supporting smooth navigation without interrupting the main workflow.

Below the AppBar, the interface highlights a real-time travel status indicator followed by a “Most Use Service” section. This section contains shortcut icons for frequently used features such as flight booking, trip planning, map navigation, currency conversion, travel history, and additional services. These quick-access buttons are designed to reduce user effort and improve

efficiency when performing common tasks.

In addition, a promotional banner is displayed to showcase ongoing travel deals and discounts, encouraging users to explore cost-saving opportunities. Beneath this, a horizontally scrollable “Popular Destination” section presents recommended travel locations with visual thumbnails which allowing users to browse and select destinations easily.

A floating action button is located at the bottom right corner of the screen, which activates the AI-powered chatbot. This feature provides real-time assistance, helping users with inquiries, travel planning, and decision-making.

### 5.5.2 AI Chatbot

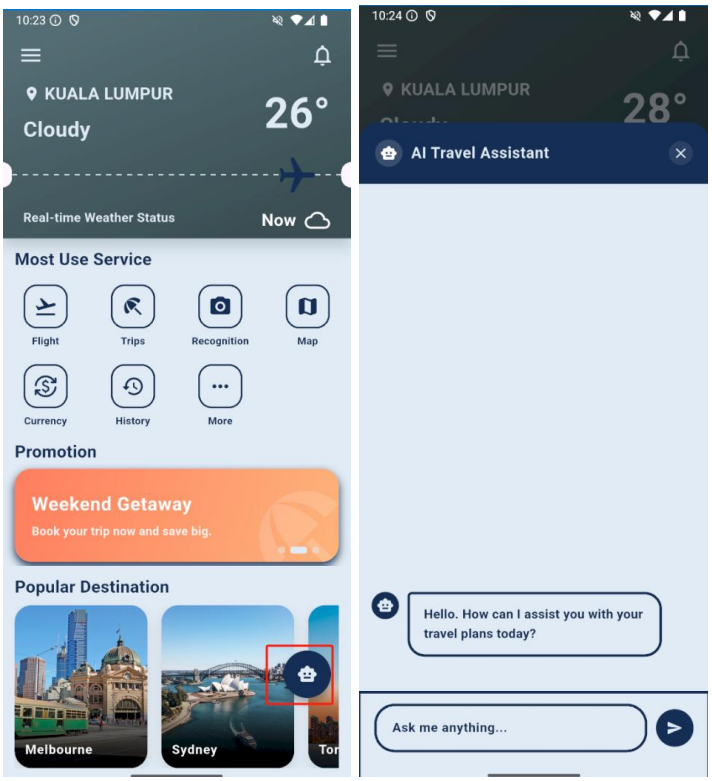



Figure 5.2 AI Chatbot

An AI-powered chatbot was implemented using Google AI Studio's Gemini API. The chatbot responds in real time about travel related queries to user. It provided through a floating action button on the home page and displays as a draggable popup. When the system detects no internet connection, the chatbot button becomes greyed out, suggesting that the functionality is temporarily unavailable.

### 5.5.3 Travel Plan Module

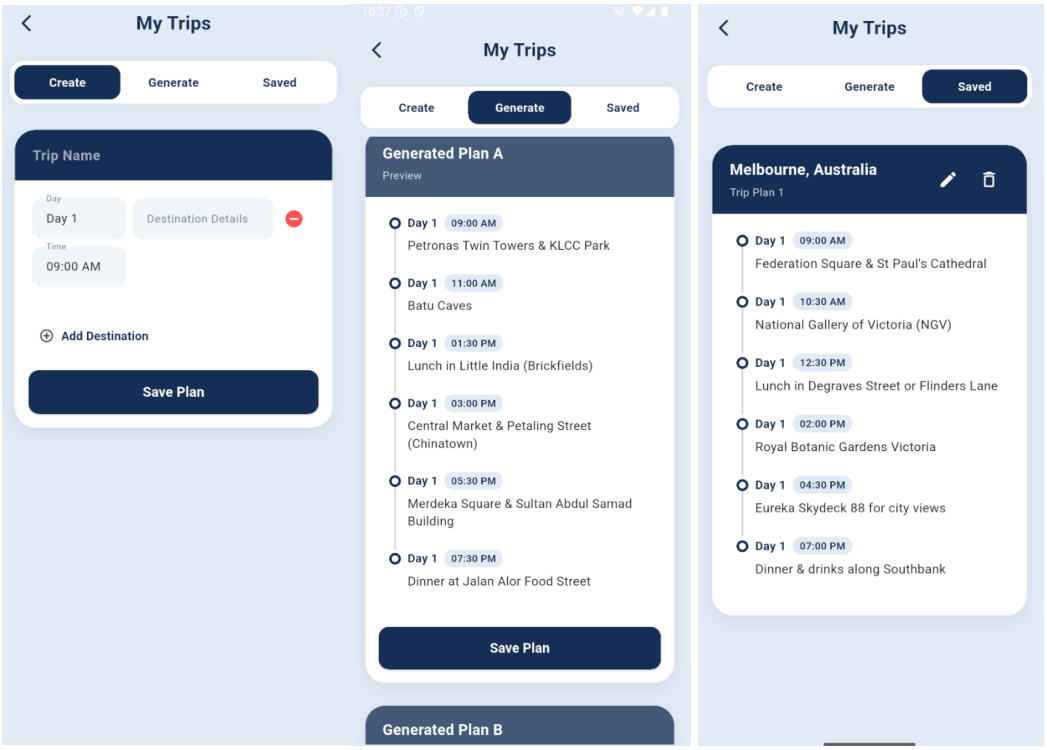



Figure 5.3 Travel Plan

The Travel Plan Module features an interactive and centralised interface divided into three primary segments: “Create”, “Generate”, and “Saved”. In the “Create” segment, users are provided with a dynamic interface to manually construct their travel itinerary. Users can input a custom trip name and define specific structured details for each activity, including the day, time, and destination. The interface allows users to seamlessly add additional rows for new destinations or remove existing ones before finalizing and saving the plan into the application’s local storage.

For automated planning, the “Generate” segment relies on advanced artificial intelligence. Users are required to select a specific country, city, and duration of stay from the provided dropdown menus. After making these selections, tapping the “Generate Plan” button launches the AI-powered travel plan generator utilising the Google Gemini API. To ensure system stability, the application actively monitors the network status. When an internet connection is unavailable, the generation button is immediately deactivated and displays an offline message accompanied by a Wi-Fi off symbol, reminding users to reconnect. Once successfully processed online, the AI-generated travel plans are presented in detailed chronological cards. Users can evaluate these previews and tap the save button to permanently store their preferred itinerary.

All customised and AI-generated itineraries are ultimately transferred to the “Saved” segment.

These chosen plans are displayed with informative headers indicating the destination and the specific plan number. Crucially, these records persist across app sessions as they are stored locally using SharedPreferences, guaranteeing users full offline accessibility to their schedules during their travels. Within this tab, the saved plans come with comprehensive management features. By utilizing the edit and delete icons, users can update the details of each day, modify rows, and save their changes back into local storage. Furthermore, as an intelligent automation feature, long-pressing a saved travel plan triggers a simulated real-time weather alert, pushing a high-importance notification directly to the user's device status bar based on their scheduled destination.

### 5.5.4 Booking Flight Module

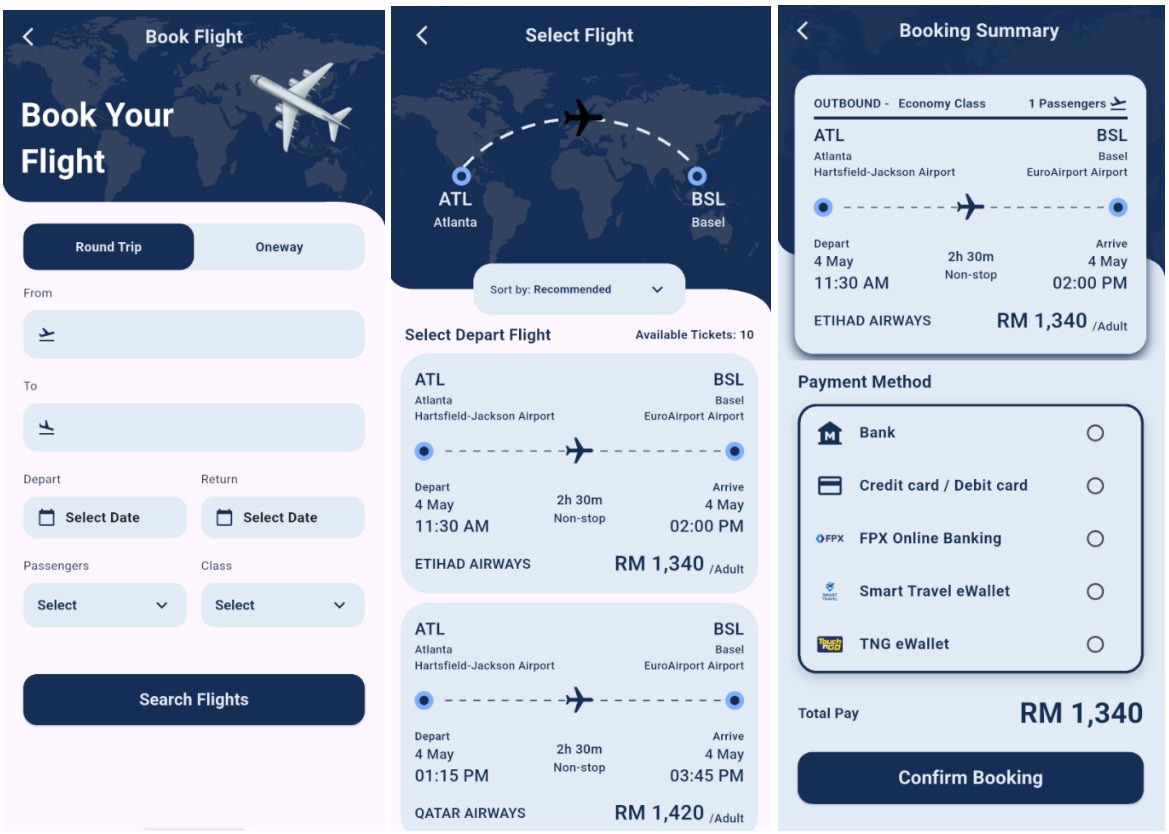



Figure 5.4 Booking Flight Module

The Flight Booking Module provides a streamlined and intuitive workflow for users to search, evaluate, and confirm airline ticket transactions. The operation initiates on the primary flight search interface, where users define their specific travel parameters. They can effortlessly toggle between round-trip and one-way journeys before specifying their departure and arrival destinations, desired travel dates, total passenger count, and preferred cabin class. Once these mandatory fields are populated, activating the “Search Flights” button prompts the system to process the inputted criteria and retrieve a simulated list of available airline routes.

Following the search operation, the system navigates the user to the flight selection interface, which visually enhances the experience with a dynamic route map displaying the planned trajectory between the chosen origin and destination airports. Below this visual aid, the system renders a structured, scrollable list of available flight tickets. Each flight card presents critical information to assist user decision-making, including the operating airline, precise departure and arrival times, total flight duration, layover status, and the calculated fare per adult. Users can also utilize a sorting function to organize the results based on their preferences, such

as recommended flights, before selecting their preferred schedule.

The final phase of the transactional workflow takes place on the booking summary interface. This screen acts as a final verification step, displaying a comprehensive overview of the selected outbound flight details alongside the total calculated payment. To complete the operation, users interact with a simulated payment gateway section, which offers a realistic variety of payment methods including bank transfers, credit or debit cards, online banking, and popular e-wallets. Upon selecting a payment method and tapping the “Confirm Booking” button, the system finalizes the mock transaction, subsequently generating the booking record to be synchronised and stored within the user’s account database.

### 5.5.5 Landmark Recognition

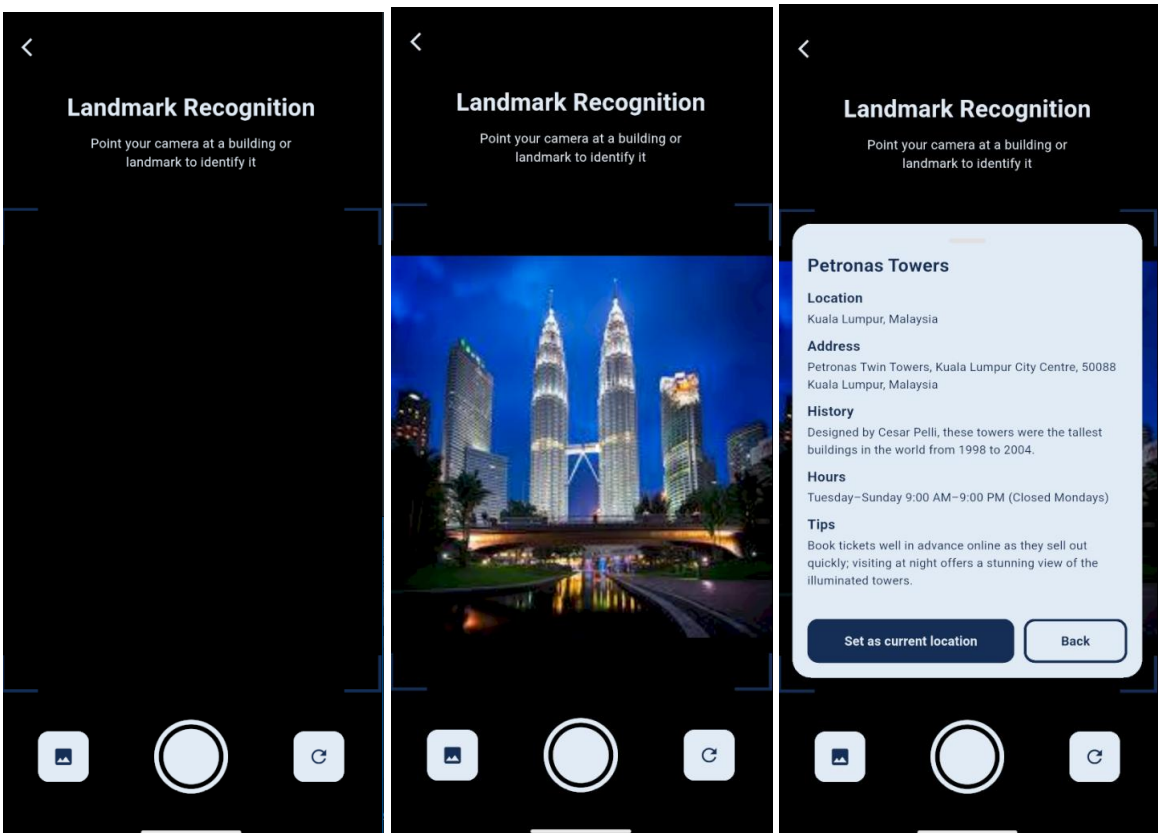



Figure 5.5 Landmark Recognition

The Landmark Recognition Module provides an intelligent, computer-vision-based feature designed to enhance the user's interactive exploration of their physical surroundings. The operation begins on the primary scanner interface, where users are presented with a straightforward layout prompting them to provide visual input. Users can utilise the "Upload Image" functionality to select a photograph of a physical monument, building, or point of interest from their device's local gallery. Once selected, the photograph is dynamically rendered within the designated image preview container on the screen.

After the target image is successfully loaded into the preview container, activating the "Analyse Landmark" button initiates the core AI-driven image processing sequence such as a photograph of the Petronas Twin Towers. The application transmits the visual payload to the integrated AI model for pattern recognition and contextual data extraction. Upon successful processing, the interface automatically updates to present a comprehensive, scrollable informational card directly beneath the analysed image. This generated output provides the user with rich, structured details regarding the identified location, including its historical background, architectural significance, and practical visitor tips, thereby seamlessly bridging the gap between

physical sightseeing and digital intelligence.

### 5.5.6 Map and Navigation

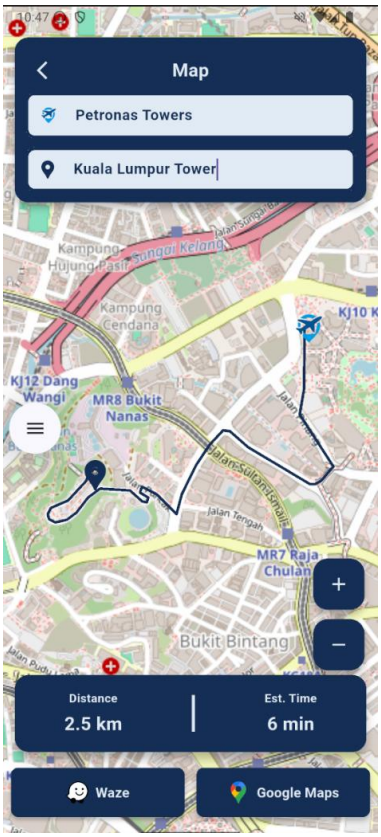



Figure 5.6 Map and Navigation

The Map and Navigation Module provide a vital real-time utility that empowering users to visualize their travel routes and obtain precise spatial context. The operation commences within the primary map interface, where the system utilizes interactive geospatial mapping to render the user's surroundings or predefined journey. Upon establishing an origin and a destination, the system dynamically calculates the optimal route and projects a clear visual trajectory represented by a distinct polyline directly onto a customized map canvas. To assist in trip logistics, the interface simultaneously extracts and displays critical routing metrics within a structured lower information panel, providing users with the exact total distance and the estimated travel duration required for the journey. Once the user reviews these spatial details, they can activate the prominent “Start Navigation” button to initiate the active routing phase, ensuring seamless, informed, and efficient transit between their scheduled locations.

## 5.6 Implementation Issues and Challenges

The implementation of the Smart Travel Application is expected to provide several technological and practical challenges. One of the major challenges is the integration of several REST APIs for flight, hotel, and transportation bookings. Different service providers may impose usage limitations, access limitations, contradictions in their data formats which can cause service disruptions or in complete information retrieval. To prevent this, the application needs to integrate suitable error-handling methods and backup strategies to ensure consistency even when external APIs fail.

Another significant challenge is the development of the AI driven travel plan generator and the camera-based landmark and building recognition feature. The travel plan generator is primarily dependent on the accuracy of the AI model when considering user inputs such as destination and duration. Incorrectly designed algorithms might generate unnecessary or unworkable plans that reduce user confidence. Furthermore, the landmark and building recognition feature may struggle in low light settings, poor camera quality, or partially blocked images. It is causing in misidentification or decreased accuracy. These limitations require continuous model training, dataset refinement, and real-world testing to ensure stability.

Offline accessibility throws additional challenges to ensuring perfect synchronisation between local and loud data. Conflicts may happen if users make changes to travel plan or booking while offline. This might conflict with cloud updates once the device reconnects. To ensure data consistency, a conflict resolution technique must be implemented such as timestamp-based prioritisation or user confirmation.

Lastly, the application must provide consistent performance across devices with different hardware capabilities. The high dependence on AI models and real-time updates may cause a burden on devices with limited processing power, memory, or battery capacity. To ensure smooth execution, optimisation measures will be required such as efficient caching, background task management, and reducing resource intensive operations. Resolving these challenges through iterative development, thorough testing, and continuous user feedback will be essential to developing a reliable and efficient smart travel solution.

## 5.7 Concluding Remark

In conclusion, this chapter has thoroughly documented the transition of the Smart Travel Application from a theoretical architectural design into a fully functional, cross-platform software solution. The frontend interface was successfully actualized utilising the Flutter framework and delivering a highly interactive and visually cohesive user experience. Concurrently, the backend infrastructure was robustly established through Firebase and ensuring secure user authentication and reliable real-time data persistence for flight transactions.

Furthermore, the system's core intelligence and utility features were successfully realised through the seamless integration of external RESTful APIs, the Google Gemini API for dynamic itinerary generation, alongside Open-Meteo and Nominatim for real-time environmental and geographical data. Although several technical challenges were encountered during the implementation phase such as API rate limitations, complex JSON payload parsing, and asynchronous state management. These hurdles were effectively mitigated through robust error handling, local caching strategies, and optimized Dart programming practices.

The application is now prepared for a comprehensive evaluation with all core modules fully implemented, integrated, and stabilised. The subsequent chapter will detail the complex testing methodologies to verify the system reliability, performance and overall alignment with the initial project objectives defined in Chapter 1 such as functional testing and User Acceptance Testing (UAT).

# CHAPTER 6

# SYSTEM EVALUATION AND DICUSSION

## 6.1 Verification Plan

The verification plan describes the testing methodologies that will be used to confirm that the Smart Travel Application fulfils the functionality, performance, and usability requirements. Table 6.1 describes the testing procedures, objectives, and expected outcomes.

Table 6.1 Verification plan

| **Testing Type** | **Objective** | **Example of Inputs or Scenarios** | **Expected Outcome** |
|---|---|---|---|
| Unit Testing | To verify that each individual module functions as schedule | Travel plan generator, AI chatbot queries, Camera-based building and landmark recognition | Each of the module produces correct and consistent results independently |
| Integration Testing | To confirm that module function correctly when combined | Linking travel plan with booking | Smooth data flow and proper interaction between integrated modules |
| Functional Testing | To ensure the system fulfils user requirements | Multiple destination travel plan request, multi-language support, AI chatbot query | System generates valid travel plan and correct response independently |
| Performance Testing | To measure responsiveness and system reliability under different environment | Chatbot query with weak network, real-time API update requests | Chatbot responses less than 3 seconds, real-time updates less than 3 seconds, stable offline feature performance |
| Security Testing | To ensure data protection and secure user authentication | Attempt login with valid or invalid credentials, test data encryption | System prevents unauthorised access and protects sensitive user data |

| | | | |
|---|---|---|---|
| Compatibility Testing | To validate operation across multiple platform and devices | Run application on Android and iOS devices with different hardware | Application performs consistently across supported devices and platforms |
| Regression Testing | To confirm that updates or modifications do not break existing features | Apply a software update and re-test travel plan generation | Existing features continue to function correctly after updates |
| User Acceptance Testing (UAT) | To validate usability and satisfaction from an end-user perspective | Testing by sample users for travel plan and chatbot use | Users report ease of use, accuracy of results, and satisfaction with system flow. |

## 6.2 Requirement Traceability Matrix (RTM)

The Requirement Traceability Matrix (RTM) ensures that all functional and non-functional requirements are covered by appropriate test cases and verification methods. Table 6.2 summarises the mapping between requirements, test cases, and verification method.

Table 6.2 Requirement Traceability Matrix (RTM)

| **Requirement** | **Description** | **Test Case** | **Verification Method** |
|---|---|---|---|
| User Account Management | The system shall allow user to register, login, and manage profiles. | User attempts to register, login, and update profile details | Unit Testing, Functional Testing |
| Personalised Travel Plan Generation | The system shall generate customised travel plan. | User input destination and durations | Functional Testing, UAT |
| AI Travel Chatbot Assistance | The system shall provide an AI-powered chatbot. | User asks chatbot travel-related queries | Unit Testing, Functional Testing |
| Camera-based Building and Landmark Recognition | The system shall recognise building and landmarks using device camera. | User captures images of building and landmark | Unit Testing, UAT |
| Booking Integration | The system shall enable booking API (flights, hotels, and transportation) | User attempts to book a flight or hotel via integrated APIs | Integrating Testing, UAT |
| Real-Time Updates | The system shall deliver real-time notification on weather, flight status, and travel disruptions. | API triggers flight delay, user receives notification | Integrating Testing, Performance Testing |
| Offline Accessibility | The system shall provide offline access to essential resources. | User accesses saved travel plan in offline mode | Functional Testing, UAT |
| Multi-Language Support | The system shall support multiple languages. | User switches system language | Functional Testing, UAT |

| | | | |
|---|---|---|---|
| Usability | The system shall feature a user-friendly interface. | Users test navigation through main menus and functions | UAT |
| Performance | The system shall respond within three seconds (chatbot, travel plan generator). | Measure response time for chatbot and travel plan generator | Performance Testing |
| Reliability | The system shall ensure offline features available at least 90% of the time. | Simulate network loss and attempt to access offline resources | Functional Testing, Performance Testing |
| Security | The system shall employ secure authentication mechanisms and encrypt | Attempt login with correct and incorrect credentials | Security Testing, Unit Testing |
| Compatibility | The system shall operate on both Android and iOS platforms | Deploy and test app on Android and iOS devices | Compatibility Testing |
| Scalability | The system shall be scalable to support increasing numbers of user and data. | Simulate multiple users accessing travel plan generator simultaneously | Load Testing, Performance Testing |
| Maintainability | The system shall allow efficient updates and maintenance. | Deploy a minor system update and test functionality post-update | Regression Testing |

## 6.3 Testing Setup and Result

The following preliminary results were tested on the implemented modules:

### 6.3.1 Splash Screen

Table 6.3 Result – Splash screen

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Splash Screen Display | Launch the app and observe startup sequence | App logo displayed for 3 seconds, then navigates to Home Page. | Expected | Pass |
| Proceed to Home | Wait for splash duration to finish | Home Page loads automatically without user action. | Expected | Pass |

### 6.3.2 Home Page Navigation

Table 6.4 Result – Home page navigation

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Drawer Menu | Tap menu icon on AppBar | Drawer opens with Profile, Settings, and Login/Signup options. | Expected | Pass |
| Notification Button | Tap notification icon | Displays notification notification center with clear all button | Expected | Pass |
| Most Use Services button | Tap all the services button | Navigate to the service that user tapped. | Expected | Pass |

| | | | | |
|---|---|---|---|---|
| Promotions Banner | Swipe promotional banner left/right | Banners rotate with smooth transition. | Expected | Pass |
| Popular Destination | Display all the destination that provide to users. | Navigate and display the details of the destination that user tapped. | Expected | Pass |

### 6.3.3 User Authentication and Profile

Table 6.5 Result – User authentication and profile

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| User Registration | Navigate to Signup, enter valid email and password, tap register | Account is created in Firebase Authentication and Firestore database | Expected | Pass |
| User Login | Enter registered email and password on Login page. | Login successful, user is navigated to Home Page. | Expected | Pass |
| Invalid Credentials | Enter incorrect password or unregistered email. | Feedback to user | Expected | Pass |
| Google Sign-In | Tap Google logo button | Prompts Google Account selection, logs in, and loads user profile. | Expected | Pass |
| Logout | Open drawer menu and tap "Sign out". | Session ends, user is redirected to the Login screen. | Expected | Pass |

### 6.3.4 Flight Booking and Digital Boarding Pass

Table 6.6 Result – Flight booking module

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Flight Selection | Select flight class (Economy/Business) and number of passengers. | Price updates dynamically based on selections. | Expected | Pass |
| Confirm Booking (Cloud Sync) | Tap "Confirm Booking" on the checkout page. | Transaction is securely saved to Cloud Firestore with NTP timestamp. | Expected | Pass |
| Generate Boarding Pass | View the completed booking summary. | Digital boarding pass renders correctly with CustomClipper edges and barcode. | Expected | Pass |
| Booking History Retrieval | Navigate to "History" page. | App fetches previous transactions from Firestore and displays them in a list. | Expected | Pass |

### 6.3.5 Real-Time Travel Utilities

Table 6.7 Result – Real-time travel utilities

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Map Geocoding | Open Map module, search for place. | Nominatim API fetches coordinates, map centers on location, and marker drops. | Expected | Pass |
| Weather Updates | Open Weather widget. | Open-Meteo API fetches live data and displays accurate temperature and animations. | Expected | Pass |
| Currency Conversion | Input a value and select target currency. | System calculates and displays converted amount instantly using live rates. | Expected | Pass |
| Utility Offline Handling | Disable Wi-Fi and attempt to fetch live weather or currency. | UI displays "No Internet Connection" without crashing the app. | Expected | Pass |

### 6.3.6 System Notifications

Table 6.8 Result – Notification system

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Flight Delay Simulation | Trigger the background flight delay mock function. | System pushes a localized delay notification to the device status bar. | Expected | Pass |
| Weather interruption | Trigger the plan background flight delay mock function. | System pushes a localized weather interruption notification to the device status bar. | Expected | Pass |
| Notification Setting | Turn off "Enable Notifications" in settings, then book a flight. | No notification is triggered. | Expected | Pass |

### 6.3.7 Camera-based Landmark Scanner

Table 6.9 Result – Camera-based landmark scanner

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Camera Permissions | Tap the "Scanner" tab for the first time. | Device prompts user to grant camera access permissions. | Expected | Pass |
| Landmark Recognition | Point camera at a clear, well-lit image of a famous landmark | AI processes the image and displays contextual information (history, details). | Expected | Pass |
| Landmark Recognition (upload image) | Upload a clear image | AI processes the image and displays contextual information (history, details). | Expected | Pass |

### 6.3.8 Travel Plan Module

Table 6.10 Result – Travel plan module

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Tab Navigation Switch | Tap the "Create", "Generate", and "Saved" buttons at the top of the screen. | The UI smoothly transitions between the manual entry form, AI form, and saved lists without losing state. | Expected | Pass |

| | | | | |
|---|---|---|---|---|
| Manual Trip Creation | In "Create" tab, enter a "Trip Name", tap "Add Destination" to add rows, fill them out, and tap "Save Plan". | The plan is saved to local storage, input fields clear, and the app automatically switches to the "Saved" tab. | Expected | Pass |
| Empty Input Validation | Attempt to save a manual plan with an empty "Trip Name" or empty destination fields. | System blocks the save action and displays a SnackBar stating "Please fill in all row fields." or "Please enter a Trip Name.". | Expected | Pass |
| AI Plan Generation (Online) | In "Generate" tab, select Country, City, and Days from dropdowns, then tap "Generate Plan". | Button changes to "Crafting Itinerary..." loader, calls Gemini API, and displays parsed travel plan cards. | Expected | Pass |
| AI Generation (Offline Fallback) | Disable internet connection and view the "Generate" tab. | Button becomes disabled, displays an "Offline" icon, and shows a warning text below it. | Expected | Pass |
| Edit Saved Plan Inline | In "Saved" tab, tap the edit icon, modify a destination name, and tap the checkmark icon. | Changes are saved locally and the text fields revert to a static display with updated content. | Expected | Pass |

| | | | | |
|---|---|---|---|---|
| Delete Saved Plan | Tap the trash bin icon on a saved plan card. | An AlertDialog asks "Are you sure you want to delete this trip plan?" The plan is removed upon tapping "Yes". | Expected | Pass |
| Long-Press Weather Alert | Long-press on any saved plan card in the "Saved" tab. | SnackBar shows "Weather alert sent", and a system push notification appears with simulated weather info for the destination. | Expected | Pass |

### 6.3.9 AI Chatbot

Table 6.11 Result – AI chatbot

| Test Case | Test Steps | Expected Result | Actual Result | Pass / Fail |
|---|---|---|---|---|
| Launch Chatbot | Tap floating action button with internet | Chatbot popup appears with initial greeting. | Expected | Pass |
| Send Query | Type in the queries that related to travel | Chatbot responds with travel-related advice or information. | Expected | Pass |
| API Failure Fallback | Simulate API failure by disabling endpoints | Chatbot returns mock fallback responses. | Expected | Pass |
| Offline Mode | Disable internet and attempt to launch chatbot | Chatbot button greyed out and unavailable. | Expected | Pass |

## 6.4 Project Challenges

Throughout the development and testing phases of the Smart Travel Application, several significant technical challenges were encountered and successfully resolved. These challenges primarily revolved around data parsing, asynchronous state management, and native system integrations.

### 6.4.1 Unpredictable AI Response Formatting and Data Parsing

One of the most persistent challenges was ensuring the reliability of the Google Gemini API's output. Although the AI was strictly prompted to return a JSON-formatted itinerary, it occasionally appended conversational text or markdown code blocks. During initial testing, this unpredictability caused the Dart jsonDecode function to throw fatal parsing exceptions and crashing the data flow. To resolve this, a robust sanitization algorithm was implemented in the code to dynamically strip backticks and extract only the valid array using indexOf("[") and string manipulation techniques, ensuring the application successfully parses the data regardless of AI formatting inconsistencies.

### 6.4.2 Complex UI State Synchronization

The "My Trips" module required maintaining highly complex, interconnected states. Because the application allows users to dynamically edit, add, or remove days from their travel plans, the system had to manage dozens of TextEditingController instances alongside the raw JSON data lists (such as _chosenPlans and _generatedPlans). Keeping these UI controllers perfectly synchronized with the underlying local data during Create, Read, Update, and Delete (CRUD) operations posed a significant logical challenge. This was overcome by systematically rebuilding and tracking controller indices during setState rebuilds to prevent memory leaks and out-of-bounds errors.

### 6.4.3 Real-Time Connectivity and Asynchronous Handling

Managing unpredictable network conditions during heavy asynchronous API requests was another major hurdle. If a user lost connection while the AI was generating a 7-day itinerary, the application could hang indefinitely. This was mitigated by integrating the connectivity_plus package to establish a StreamSubscription that continuously monitors the device's network state in the background. Furthermore, a strict 30-second timeout constraint was applied to the HTTP requests. If the network drops, the application gracefully degrades, disables the generation

buttons, and displays clear offline fallback UI elements rather than freezing.

### 6.4.4 Native Hardware and Notification Integration

Implementing the long-press weather alert feature required bridging the Flutter framework with native Android and iOS system-level notifications. The challenge was efficiently fetching user configuration states from SharedPreferences and seamlessly triggering the flutter_local_notifications plugin from a customized GestureDetector. Ensuring the correct high-importance notification channels and permissions were configured without causing background execution conflicts required extensive configuration and debugging on physical testingdevices.

## 6.5 Objectives Evaluation

The primary purpose of the testing and evaluation phase is to verify whether the developed Smart Travel Application successfully aligns with the initial project objectives defined in Chapter 1. Based on the comprehensive functional testing, system integrations, and resolved challenges discussed in this chapter, the system successfully met all its core objectives:

**Objective 1: To develop an all-in-one AI-driven Smart Travel Application**

The objective of centralizing fragmented travel services into a single platform was successfully realized. Developed using the Flutter framework, the application seamlessly integrates core travel modules that include flight booking simulation, interactive mapping, weather updates, and currency conversion into one unified cross-platform mobile interface. The System Verification test cases confirm that users can navigate through all these features smoothly within a single ecosystem.

**Objective 2: To implement real-time intelligent and predictive automation updates**

The system actively monitors external conditions and provides real-time responsiveness. This was achieved through the integration of the Open-Meteo API for live weather tracking and the flutter_local_notifications plugin. As validated in the Notification System testing, the application successfully pushes localized alerts such as simulated flight delays and weather warnings. It is ensuring the user is promptly notified of potential disruptions to minimize travel inconvenience.

**Objective 3: To enhance personalisation and user interaction through AI technologies**

Advanced AI integration was the core highlight of the application with utilising the Google Gemini API. As verified in the “My Trips” testing, the system successfully processes user inputs to generate highly personalized, day-by-day travel itineraries such as destination and duration. Additionally, the AI Chatbot module passed complex testing and demonstrating its ability to process natural language queries and provide real-time, interactive, and personalised travel assistance.

**Objective 4: To provide offline support for essential features**

Ensuring offline reliability was critical for a travel context. By implementing local data persistence utilizing the shared_preferences package and integrating connectivity_plus for network state monitoring. The application successfully provides strongoffline support. As proven

in multiple offline test cases, users can fully access their saved travel plan and booking histories without any internet connectivity, effectively preventing app crashes and data loss.

## 6.6 Concluding Remarks

In conclusion, this chapter has presented a comprehensive testing and evaluation framework for the Smart Travel Application. A complex bottom-up verification strategy was executed that covering 13 different core modules. The evaluation spanned from fundamental user interface interactions and local state management to complex asynchronous operations involving Firebase Cloud Firestore and the Google Gemini API.

The results obtained from the detailed test cases demonstrate that the application is highly strong and performs consistently under various simulated conditions that include strict offline scenarios and unexpected external API constraints. While several technical hurdles were encountered during the testing phase, notably handling unpredictable JSON parsing from the AI model, complex UI state synchronizations, and cross-platform native notification setups—these issues were successfully mitigated through robust error-handling mechanisms and optimized Dart programming practices.

Furthermore, the objective evaluation confirmed that the developed system successfully fulfills all the primary goals defined at the beginning of this project. Also, successfully transitioning from a conceptual design into a fully functional, AI-driven travel companion. The development phase is officially complete with the system fully tested, verified, and stabilised. The subsequent and final chapter will summarize the overall project contributions, discuss the current system limitations, and propose viable recommendations for future enhancements.

# CHAPTER 7

# CONCLUSION AND RECOMMENDATION

## 7.1 Conclusion

The development of the Smart Travel Application has successfully resulted in a strong, comprehensive, and cross-platform mobile solution designed to modernize the travel planning experience. By centralizing fragmented travel services into a single and cohesive ecosystem, the application effectively addresses the common pain points faced by modern travellers such as platform switching, lack of personalization, and vulnerability to network disruptions.

Throughout the project lifecycle from architectural design to rigorous implementation and system testing, all initial objectives outlined in Chapter 1 were successfully achieved. The integration of the Flutter framework along with a Firebase backend ensured a responsive user interface and secure real-time data management. Furthermore, the application successfully harnessed the power of Artificial Intelligence via the Google Gemini API. It is enabling advanced natural language processing for the AI Chatbot and dynamic generation of highly personalised, chronological travel plan.

The system demonstrated high resilience through its offline accessibility features. By utilizing local data persistence, travellers can seamlessly access their saved plans and simulated booking records without relying on continuous internet connectivity. The incorporation of real-time utilities including an interactive map, live weather updates, and currency conversions. It further solidifies the application as a highly practical and all-in-one travel companion. Ultimately, this project proves the technical and operational feasibility of utilising AI-driven automation to enhance travel efficiency and user satisfaction.

## 7.2 Recommendations

While the Smart Travel Application successfully fulfills its current scope and objectives, there are several avenues for future enhancement to transition the system from a highly functional prototype into an enterprise-level commercial product. The following recommendations are proposed for future development:

### 7.2.1 Integration of Live Booking and Payment Gateways

Currently, the flight booking and ticketing module operates as a high-fidelity simulation storing transactions in Firebase. Future iterations should integrate real-world Global Distribution Systems (GDS) APIs such as Amadeus or Skyscanner. Alongside secure payment gateways to facilitate actual financial transactions and live ticket purchasing such as Stripe or PayPal.

### 7.2.2 Augmented Reality (AR) and Advanced Computer Vision

The existing camera-based landmark recognition feature provides valuable contextual information using static image processing. This can be significantly enhanced by incorporating Augmented Reality (AR) frameworks. This would allow users to point their cameras at buildings or streets and see floating digital overlays containing historical facts, ratings, or walking directions in real-time.

### 7.2.3 Social Sharing and Collaborative Travel Planning

Travel is often a group activity. A highly recommended future feature is the introduction of social collaboration. Users should be able to generate a unique shareable link or code for their AI-generated travel plans. It is allowing family and friends to view, comment on, or collaboratively edit the itinerary in real-time within the app.

### 7.2.4 Global Push Notifications via Cloud Messaging

While the current system successfully utilizes local notifications for simulated alerts, integrating Firebase Cloud Messaging (FCM) would elevate the predictive automation feature. FCM would allow the backend server to push live, global updates regarding sudden flight cancellations, gate changes, or severe weather warnings directly to the user's device. It is ensuring maximum safety and convenience.

# REFERENCES

[1] Q. Khan. “Travel and Tourism: An Exploration of the Modern World.” Medium.com. Accessed: Apr. 14, 2025. [Online.] Available: https://medium.com/@qasim06/travel-and-tourism-an-exploration-of-the-modern-world-f2a4ce0ca5ff

[2] C. Tufft, M. Constantin, M. Pacca, and R. Mann. “The way we travel now.” McKinsey.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.mckinsey.com/industries/travel/our-insights/the-way-we-travel-now

[3] J. Sun. “How is AI reshaping the travel experience?” weforum.org. Accessed: Apr. 14, 2025. [Online.] Available: https://www.weforum.org/stories/2023/12/how-is-ai-reshaping-the-travel-tourism/

[4] O. Vlahović, Ž. Rađenović, D. Perović, V. Vujačić, and K. Davidović, “Digital transformation in tourism: The role of E-Booking systems,” *Croatian Reg. Dev. J.*, vol. 5, no. 2, pp. 129–145, Nov. 2024, doi: 10.2478/crdj-2024-0012.

[5] “Traditional Travel Agency Global Market Report 2025,” The Business Research Company, London, UK. Accessed: Apr. 14, 2025. [Online]. Available: https://www.thebusinessresearchcompany.com/report/traditional-travel-agency-global-market-report

[6] C. Vianna. “How do OTAs work.” Xola.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.xola.com/articles/how-do-otas-work/

[7] Agoda. “See The World For Less.” Agoda.com. Accessed Apr. 14, 2025. [Online.] Available: https://www.agoda.com/?cid=1844104&ds=P6TO03BnjdtpVKCI

[8] Booking.com. “Find your next stay.” Booking.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.booking.com/

[9] Traveloka. “Your One Stop Gateway to Southeast Asia.” Traveloka.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.traveloka.com/en-en

[10] I. Gasmi, M. Soui, K. Barhoumi, and M. Abed, "Recommendation rules to personalize itineraries for tourists in an unfamiliar city," *Appl. Soft Comput.*, vol. 150, Jan. 2024, doi: 10.1016/j.asoc.2023.111084.

[11] S. Wankhede. "The Future of Travel Apps: Trends and Services." vocal.media. Accessed: Apr. 14, 2025. [Online.] Available: https://vocal.media/01/the-future-of-travel-apps-trends-and-services

[12] Google. "Find local businesses, view maps and get driving directions in Google Maps." Google.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.google.com.my/maps

[13] M. Ibrahim. "The impact of AI on the tourism industry." ultralytics.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.ultralytics.com/blog/the-impact-of-ai-on-the-tourism-industry

[14] L. Craig. "What is AI? Artificial Intelligence explained." TechTarget.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence

[15] Trip.com. "Plan Your Dream Holidays With Trip.com." my.trip.com. Accessed: Apr. 14, 2025. [Online.] Available: https://my.trip.com/?locale=en-my

[16] Tripadvisor. "Where to?" tripadvisor.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.tripadvisor.com.my/

[17] CheckMyTrip. "All your travel info, in one place." checkmytrip.com. Accessed: Apr. 14, 2025. [Online.] Available: https://checkmytrip.com/

[18] TripIt. "An easier trip, each time." tripit.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.tripit.com/web

[18] TripIt. "An easier trip, each time." tripit.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.tripit.com/web

[19] Wanderlog. "One app for all your travel planning needs." wanderlog.com. Accessed: Apr. 14, 2025. [Online.] Available: https://wanderlog.com/

[20] Kayak. “Compare hotel deals from 100s of sites.” kayak.com. Accessed: Apr. 14, 2025. [Online.] Available: https://www.kayak.com.my/

[21] “What is Agile Methodology?” geeksforgeeks.org. Accessed: Apr. 14, 2025. [Online.] Available: https://www.geeksforgeeks.org/what-is-agile-methodology/

[22] Visual Studio Code. “Your code editor. Redefined with AI.” code.visualstudio.com. Accessed: Apr. 14, 2025. [Online.] Available: https://code.visualstudio.com/

[23] Android Studio. “Android studio.” developer.android.com. Accessed: Sept. 10, 2025. [Online.] Available: https://developer.android.com/studio

[24] Flutter. “Build for any screen.” flutter.dev. Accessed: Apr. 14, 2025. [Online.] Available: https://flutter.dev/

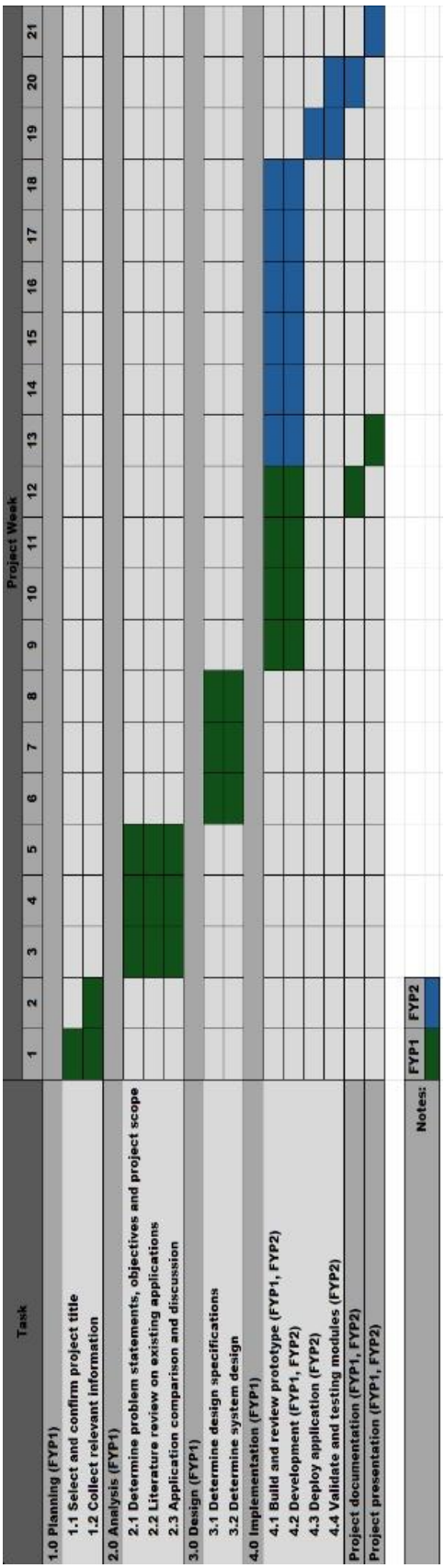
[25] Google AI Studio. “Google AI studio.” aistudio.google.com. Accessed: Sept. 10, 2025. [Online.] Available: https://aistudio.google.com/prompts/new_chat

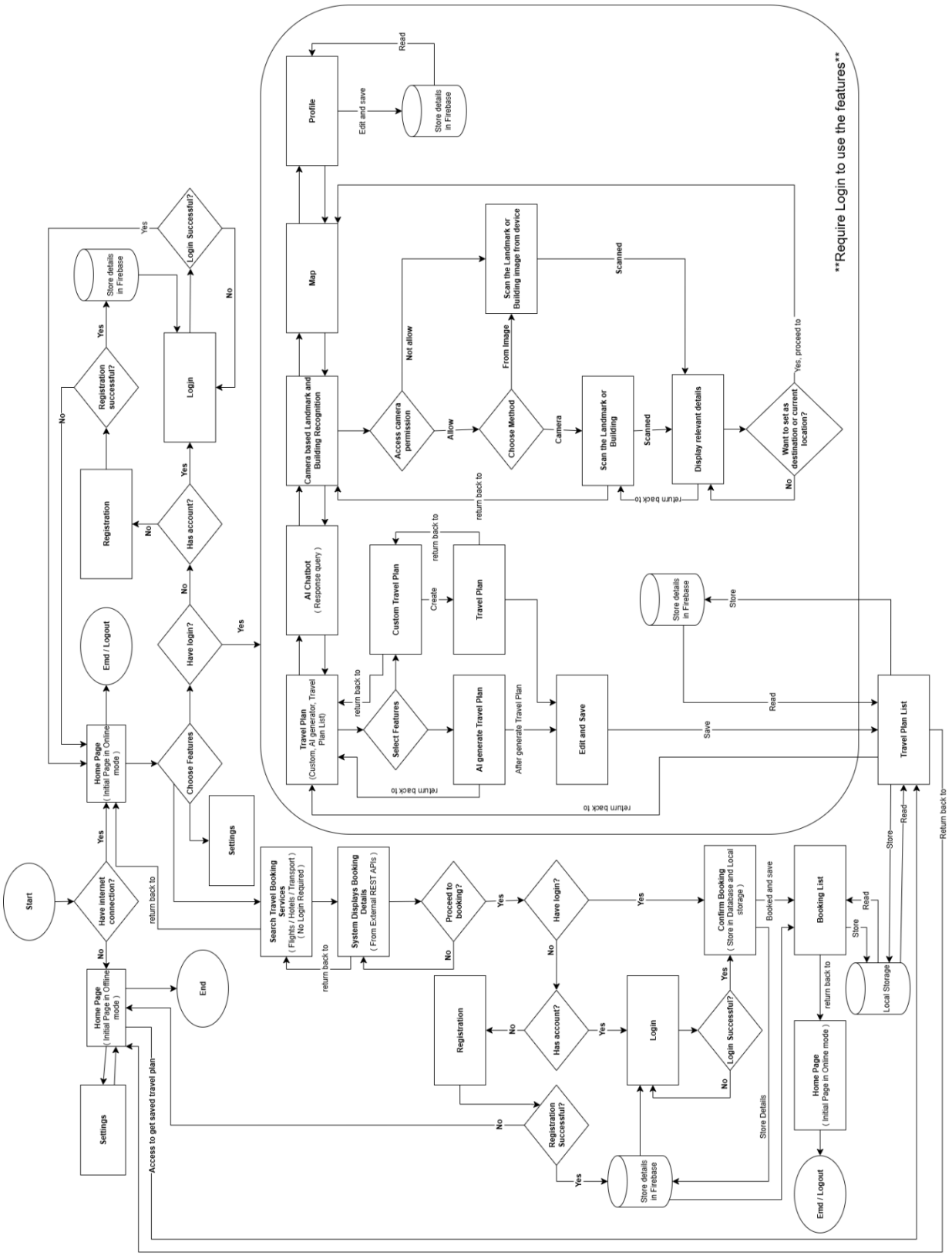
[26] Firebase. “Make your app the best it can be with Firebase and generative AI.” firebase.google.com. Accessed: Apr. 14, 2025. [Online.] Available: https://firebase.google.com/

[27] Draw.io. “Security – first diagramming for teams.” drawio.com. Accessed: Sept. 10, 2025. [Online.] Available: https://www.drawio.com/

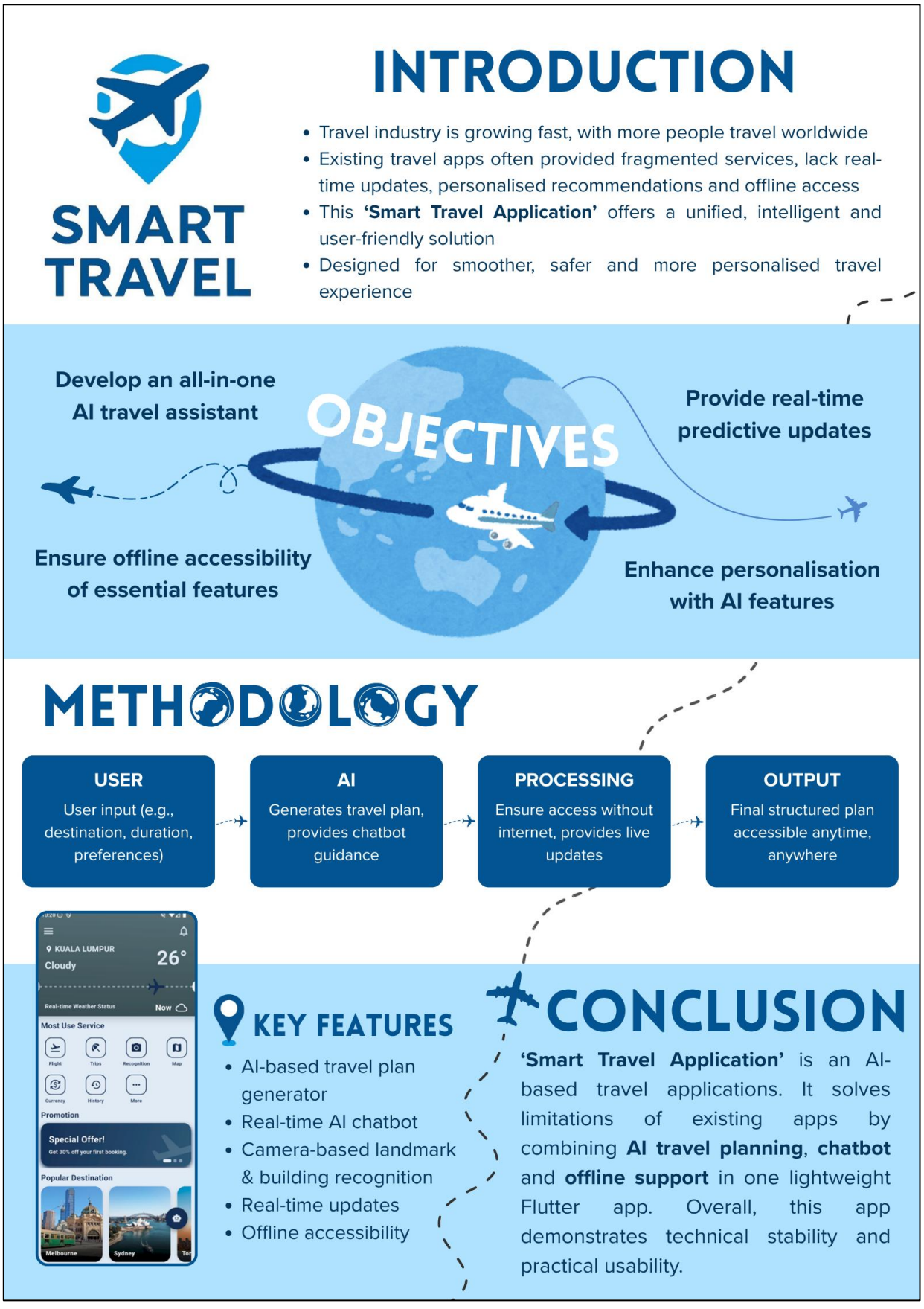
# Appendices

## Appendix A




Start
Have internet connection?
Yes
No
Home Page ( Initial Page in Online mode )
Home Page ( Initial Page in Offline mode )
End
Settings
Access to get saved travel plan
Have login?
Yes
No
Has account?
Yes
No
Registration
Registration successful?
Yes
No
Store details in Firebase
Login
Login Successful?
Yes
No
Store details in Firebase
Emd / Logout
Choose Features
Settings
return back to
Search Travel Booking Services ( Flights / Hotels / Transport ) ( No Login Required )
System Displays Booking Details ( From External REST APIs )
Proceed to booking?
Yes
No
Have login?
Yes
No
Has account?
Yes
No
Registration
Login
Login Successful?
Yes
No
Registration Successful?
No
Yes
Store details in Firebase
Confirm Booking ( Store in Database and Local storage )
Booked and save
Store Details
Booking List
Store
Read
Local Storage
Home Page ( Initial Page in Online mode )
Emd / Logout
Return back to
**Require Login to use the features**
Profile
Edit and save
Read
Store details in Firebase
Map
Camera based Landmark and Building Recognition
Access camera permission
Not allow
Allow
Choose Method
From Image
Camera
Scan the Landmark or Building image from device
Scanned
Scan the Landmark or Building
Scanned
Display relevant details
Want to set as destination or current location?
Yes, proceed to
No
return back to
AI Chatbot ( Response query )
Custom Travel Plan
Create
Travel Plan
return back to
Travel Plan (Custom, AI generator, Travel Plan List)
Select Features
AI generate Travel Plan
After generate Travel Plan
Edit and Save
Store details in Firebase
Store
Read
Save
Travel Plan List
return back to

## Poster


SMART TRAVEL
INTRODUCTION
• Travel industry is growing fast, with more people travel worldwide
• Existing travel apps often provided fragmented services, lack real-time updates, personalised recommendations and offline access
• This 'Smart Travel Application' offers a unified, intelligent and user-friendly solution
• Designed for smoother, safer and more personalised travel experience
Develop an all-in-one AI travel assistant
OBJECTIVES
Provide real-time predictive updates
Ensure offline accessibility of essential features
Enhance personalisation with AI features
METHODOLOGY
USER
User input (e.g., destination, duration, preferences)
AI
Generates travel plan, provides chatbot guidance
PROCESSING
Ensure access without internet, provides live updates
OUTPUT
Final structured plan accessible anytime, anywhere
KUALA LUMPUR
Cloudy
26°
Real-time Weather Status
Now
Most Use Service
Promotion
Special Offer!
Get 30% off your first booking.
Popular Destination
Melbourne
Sydney
KEY FEATURES
• AI-based travel plan generator
• Real-time AI chatbot
• Camera-based landmark & building recognition
• Real-time updates
• Offline accessibility
CONCLUSION
'Smart Travel Application' is an AI-based travel applications. It solves limitations of existing apps by combining AI travel planning, chatbot and offline support in one lightweight Flutter app. Overall, this app demonstrates technical stability and practical usability.