Horizon – A Personalized Flight Journal

Group Members

Ahmed Tasadduq (01-131222-008)

Zainab Asif (01-131222-050)

Supervisor: Engr. Aamir Sohail

A Final Year Project submitted to the Department of Software Engineering, Faculty of Engineering Sciences, Bahria University, Islamabad in the partial fulfillment for the award of degree in Bachelor of Software Engineering

May 2026

FYP Completion Certificate

Student Name: Ahmed Tasadduq Enrollment No: 01-131222-008

Student Name: Zainab Asif Enrollment No: 01-131222-050

Programme of Study: Bachelor of Software Engineering

Project Title: **Horizon – A Personalized Flight Journal**

It is to certify that the above students' project has been completed to my satisfaction and to my belief, its standard is appropriate for submission for evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at ***13%*** that is within the permissible limit set by the HEC. I have also found the thesis in a format recognized by the department.

Supervisor's Signature: _____

Date: 16th April 2026 Name: Engr. Aamir Sohail

Certificate of Originality

This is certify that the intellectual contents of the project

Horizon – A Personalised Flight Journal

are the product of my/our own work except, as cited properly and accurately in the acknowledgements and references, the material taken from such sources as research journals, books, internet, etc. solely to support, elaborate, compare, extend and/or implement the earlier work. Further, this work has not been submitted by me/us previously for any degree, nor it shall be submitted by me/us in the future for obtaining any degree from this University, or any other university or institution. The incorrectness of this information, if proved at any stage, shall authorities the University to cancel my/our degree.

Name of the Student: Ahmed Tasadduq

Signature: _____ Date: 16th April 2026

Name of the Student: Zainab Asif

Signature: _____ Date: 16th April 2026

Project Title: Horizon – A Personalized Flight Journal

Sustainable Development Goals

(Please tick the relevant SDG(s) linked with FYDP)

SDG No	Description of SDG	SDG No	Description of SDG
SDG 1	No Poverty	SDG 9 ✓	Industry, Innovation, and Infrastructure
SDG 2	Zero Hunger	SDG 10	Reduced Inequalities
SDG 3	Good Health and Well Being	SDG 11 ✓	Sustainable Cities and Communities
SDG 4	Quality Education	SDG 12	Responsible Consumption and Production
SDG 5	Gender Equality	SDG 13	Climate Change
SDG 6	Clean Water and Sanitation	SDG 14	Life Below Water
SDG 7	Affordable and Clean Energy	SDG 15	Life on Land
SDG 8	Decent Work and Economic Growth	SDG 16	Peace, Justice and Strong Institutions
		SDG 17	Partnerships for the Goals

Range of Complex Problem Solving			
	Attribute	**Complex Problem**	
1	Range of conflicting requirements	Involve wide-ranging or conflicting technical, engineering and other issues.	✓
2	Depth of analysis required	Have no obvious solution and require abstract thinking, originality in analysis to formulate suitable models.	✓
3	Depth of knowledge required	Requires research-based knowledge much of which is at, or informed by, the forefront of the professional discipline and which allows a fundamentals-based, first principles analytical approach.	✓
4	Familiarity of issues	Involve infrequently encountered issues	✓
5	Extent of applicable codes	Are outside problems encompassed by standards and codes of practice for professional engineering.	
6	Extent of stakeholder involvement and level of conflicting requirements	Involve diverse groups of stakeholders with widely varying needs.	
7	Consequences	Have significant consequences in a range of contexts.	✓
8	Interdependence	Are high level problems including many component parts or sub-problems	
Range of Complex Problem Activities			
	Attribute	**Complex Activities**	
1	Range of resources	Involve the use of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies).	✓
2	Level of interaction	Require resolution of significant problems arising from interactions between wide ranging and conflicting technical, engineering or other issues.	✓
3	Innovation	Involve creative use of engineering principles and research-based knowledge in novel ways.	✓
4	Consequences to society and the environment	Have significant consequences in a range of contexts, characterized by difficulty of prediction and mitigation.	
5	Familiarity	Can extend beyond previous experiences by applying principles-based approaches.	✓

Abstract

People fly a lot these days, but it's not always easy for them to get flight information, travel history, and journey documentation from different sources. There are already a lot of flight tracking apps that give you real-time data, but they usually don't let you customize your experience with features like journaling, travel stats, and journey management. Horizon — Flight Tracker & Journal is an app for mobile devices that combines flight tracking, travel logging, and personalized insights all in one place.

People can look up flights by flight number or route, get real-time information about flights and the weather from outside aviation and weather APIs, and keep a full record of their travels.

One of the main features includes the ability to add a trip travel log, where users can describe their journeys through a statistics dashboard displaying various attributes, including but not limited to the number of flights made, distance traveled, and visited airports, and modules for learning more information about the chosen destinations and available facilities within the airport This application was developed using cross-platform development, and Thus involves using Flutter for multiple platforms (iOS and Android). The Backend component of this application is hosted on Azure Cloud infrastructure, comprising a Backend component developed using Node.js and Azure SQL Database.

This report presents the overall development process of the Horizon app, including the requirement analysis phase, architectural and interface design, implementation, testing, and deployment processes. The resulting system presents a functional prototype lying between traditional flight trackers and travel platforms and providing all the necessary resources for efficient travelling for the modern user.

Keywords: Flight tracking, mobile application, travel journal, real-time data integration, Flutter, cross-platform development, cloud architecture, aviation data API

Dedication

To Ahmed, whose love for aviation gave this idea its wings.

To Sir Aamir, for guiding us along the way.

And to our parents, for their constant support, no matter the distance.

Acknowledgements

We wish to express our sincere gratitude to our esteemed supervisor, Sir Aamir, for his invaluable guidance and mentorship throughout every stage of this project. His expertise and unwavering support were instrumental in the successful completion of this endeavor.

Furthermore, we extend our appreciation to the Bahria School of Engineering & Applied Sciences, Department of Software Engineering, for providing the comprehensive academic foundation essential for the realization of this project.

The completion of this work would not have been possible without the steadfast support of our families. We are deeply indebted to our parents for their enduring trust, patience, and encouragement, particularly when we were miles away.

Individual recognition is also due to several others for their personal support. Ahmed extends his heartfelt thanks to his grandfather for his continued support and faith in his abilities. Zainab is profoundly grateful to her siblings for providing essential emotional strength. We also wish to acknowledge Shahum for his consistent availability and contributions to our extensive project discussions.

On a personal note, Zainab extends her deepest appreciation to Ahmed for his relentless dedication, companionship, and brilliant technical mind that solved our toughest hurdles. Finally, we recognize the collective synergy and diligent effort invested in bringing this project to fruition.

We feel highly blessed by the Almighty and pray that He keeps showering His bounties upon us.

List of Figures

List of Tables

Table of Contents

1. Introduction

1.1 Motivation

Air travel is an important aspect of contemporary society, but gathering reliable real-time data about the status of their flights and other pertinent information is not always easy. For instance, the need to navigate through various apps in order to check flight status, weather conditions, and previous travel activity presents an additional hurdle for users.

Such limitations pose some challenges when trying to keep track of your own flights and organize and log your journey experiences. Besides, most of these platforms do not offer a personalized approach since they lack functionality related to journals, statistics, and other important aspects.

Horizon – Flight Tracker & Journal app would help users to solve these problems by offering a solution that unites all features in one application.

1.2 Objectives

The primary goal of this project is to create a flight tracking application on mobile devices where users can get all the required information.

The secondary goals are as follows:

1.2.1 **System Integration & Real-Time Data Processing:** To integrate robust third-party aviation and meteorological APIs, enabling the reliable retrieval, parsing, and display of live flight statuses, routing data, and localized weather conditions.

1.2.2 **Frontend Architecture & UI Optimization:** To engineer a highly responsive, cross-platform user interface utilizing the Flutter framework, employing advanced state management techniques to ensure seamless data synchronization and rendering. Monitoring weather conditions at both origin and destination airports.

1.2.3 **Algorithmic Data Aggregation:** To implement an analytics module that computes and visualizes user-centric travel metrics, including cumulative flight distances, frequency of travel, and unique airport interactions.

1.3 Main Contributions

Key achievements of this project are as follows:

- Development of a cohesive application for flight tracking, including search, tracking, and logging.
- Use of real-time flight and weather data with the help of third-party APIs.
- Development of personalized travel logs using user data.
- Designing a modern interface with the help of FlutterFlow.
- Development of the backend with the help of Azure services and Firebase authentication.
- Developing an integration solution for utility-based flight tracking applications and personalized travel logs.

1.4 Report Organization

The report comprises seven chapters.

Chapter 1 provides information regarding the introduction, motivations, objectives, and contributions of the project.

Chapter 2 highlights the background study and analysis of current flight tracking systems.

Chapter 3 elaborates on the system requirements such as functional and non-functional requirements.

Chapter 4 talks about the design, architecture, and interface of the system.

Chapter 5 includes the details regarding implementation.

Chapter 6 covers the testing method and results.

Chapter 7 gives a conclusion to the report.

2. Background Study/Literature Review

The rapid growth of the aviation industry has led to an increased demand for efficient and accessible flight tracking solutions. With the rise of mobile technology, several applications and platforms have been developed to provide users with real-time flight information. However, these systems vary significantly in terms of functionality, usability, and personalization.

This chapter presents an overview of existing flight tracking systems, key concepts related to the domain, and a critical analysis of current solutions to highlight the need for the proposed system

2.1 Key Concepts

2.1.1 Flight Tracking Systems

Flight tracking applications are software that provides real-time information about flights including the take-off and landing time, delay time, route of the flight, aircraft information and much more. Flight tracking software depends heavily on an external API for the data required.

A modern flight tracking application strives to improve the user experience through added features such as notification, map, and history.

2.1.2 Real-Time Data Integration

The Term real-time data integration is used to describe the procedure involved in collecting and displaying live data from other resources. The sources include flights schedules, data from an API, and weather feeds among others. It is one of the most important features in flight tracking software *[1].*

2.1.3 User Experience Based on Mobile Device

The widespread use of smart phones by many users has made mobile applications one of the most common ways of gathering travel activity-related information. The most important factor for creating an application that offers ease of use is its usability and responsiveness.

2.1.4 Travel Logging and Personalization

The travel logging option allows the user to log their own travel activities, be it the past or future trips. By using personalization techniques, it is possible to enhance user engagement with the help of customized insights.

2.2 Existing Systems

There are some existing flight tracking systems, which offer diverse functions.

2.2.1 FlightRadar24

FlightRadar24 is the flight tracking system software. It can help to track flights and their status by using world map view.

Advantages:

- Real-time tracking information
- Convenient world map-based user interface
- High number of flights covered

Disadvantages:

- Complex user interface for novices
- Lack of personalization settings
- Focused mostly on flights' fans

2.2.2 FlightAware

FlightAware is the flight tracking software, offering real-time flight information, airport information, and analytics capabilities.

Advantages:

- High accuracy of updating flights' statuses
- High-quality back-end with accurate data management
- Good analytics capabilities

Disadvantages:

- Complicated mobile app interface
- Lack of customer-oriented elements
- Insignificant flight journals

2.2.3 Application from Airlines

There are some applications made exclusively for airlines which assist in monitoring flights, bookings, and all kinds of alerts regarding travel for their customers.

Advantages:

- Access to accurate information about certain airlines
- It consists of a built-in booking system

Disadvantages:

- One airline only
- No capability to monitor different airlines
- Does not have history

2.3 Comparative Analysis

While the existing systems have been highly developed in terms of flight tracking, they fail to provide a common framework for integrating all these aspects into one.

Table 2.3.1: Comparative Analysis of Existing

Feature	FlightRadar24	FlightAware	Airline Apps	Proposed System
Live Flight Tracking	✓	✓	✓	✓
Support for Multiple Airlines	✓	✓	✕	✓
User-Friendly Interface	✕	✕	✓	✓
Flight History	✕	✕	✕	✓
Customization	✕	✕	✕	✓
Stats Dashboard	✕	✓	✕	✓

2.4 Identified Gap

After reviewing the existing systems, the following shortcomings have been identified:

- No complete system that incorporates tracking, journaling, and analytical functions
- Insufficient personalization options for users
- No integration of travel history and statistics
- Too complicated user interfaces for professional systems
- Limited functionality for airline-related apps

2.5 Proposed Solution

In order to overcome those limitations, it is proposed that the following be implemented through a new system called Horizon – Flight Tracker & Journal:

- Flight tracker
- Search function for airports and airlines
- Travel history and journaling
- Statistics and Dashboards
- Modern, minimalist, and easy to use interface

This ensures that users can organize their entire travel experience within one application.

2.6 Conclusion

In this chapter, we have discussed the current available methods to track flights and analyzed their advantages and disadvantages. Although current methods offer sufficient information about flights, they fail to address customization, integration, and usability.

In contrast, the new proposed method combines all three aspects and provides an innovative solution for today’s travelers.

3. System Requirements

Functional and non-functional requirements of the system are discussed in this chapter, as well as system interaction and interface issues. The above requirements are based on the user requirements and goals identified in the previous chapters.

3.1 Use Case Diagram

Description for Use-Case Diagram (Figure 3.1.1)

The Use-Case diagram illustrates the interactions between the primary actor, the User, and the Horizon system.

It outlines the core functionalities available to the user, which include searching for flights using a flight code or route, viewing comprehensive flight details, and manually adding flights to their account.

Furthermore, the diagram captures the user's ability to access airport and airline information, manage personal travel logs and notes, and review their travel statistics and analytics.

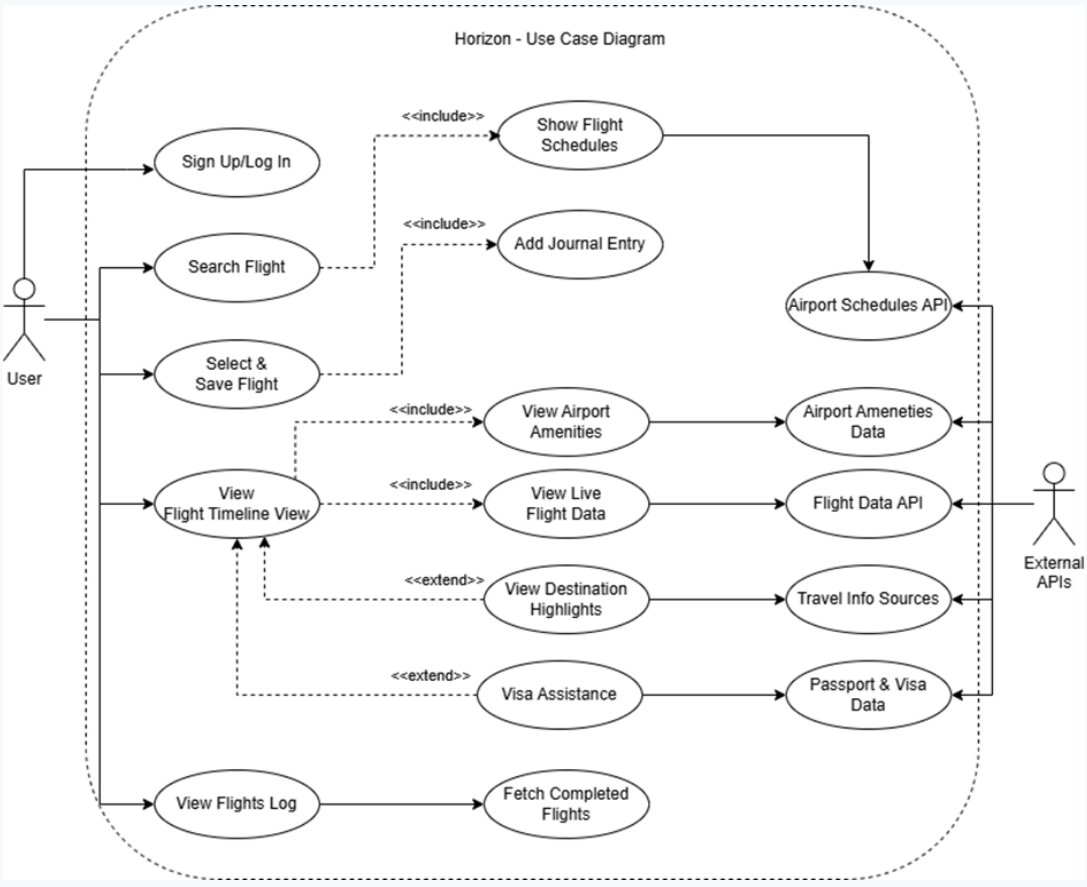


Figure 3.1.1: Use-Case Diagram

3.2 User Stories

3.2.1 Register User Account:

Use Case ID:	UC-01
Use Case Name:	Register Account for Application access
Actor(s):	User
Pre-Conditions:	The user has not registered in the system, and the system is connected to the Internet.
Priority:	High
Primary Scenario:	The user submits the registration information, and the system processes this data and successfully registers the new user account.

User Steps		**System Behavior**
1	User selects the “register” choice in the interface of the system	**2** The system presents the registration form that needs specific information.
3	The user fills in all details and submits them	**4** The system verifies the data input.
		5 If all actions are done correctly, the system registers a new user account
Exception Flows		
Actor Action		**System Response**
3a. The user makes a mistake while filling in some details.		**4a.** The system informs about errors and asks the user to give the right details.
3b. User provides proper data but there is no connection in the system at the moment.		**4b.** The system notifies the user that something goes wrong and advises to try again later.

3.2.2 Search Flight (by Flight Number)

Use Case ID:	UC-02
Use Case Name:	Search Flight (by Flight Number)
Actor(s):	User
Pre-Conditions:	User must have logged into the system and the system must be active along with being connected to external flight data API.
Priority:	High
Primary Scenario:	Flight no and Departure Date is searched for by the system and corresponding real-time flight details are displayed.
User Steps	**System Response**
1. The user selects “Search Flight” feature from the drop-down menu.	2. System asks the user to input flight number and date
3. User enters flight number and date and hits submit button.	4. System validates input fields and invokes real-time flight data through flight API.
	5. System retrieves real-time flight details such as departure time, arrival time, gate number, etc.
Exception Flows	
Actor Action	**System Response**
3a. User enters an invalid flight number or date.	**4a.** System displays error message and asks user to re-input details.
4b. API could not be invoked due to lack of connectivity or any other issue.	**5b**. System displays temporary out of service message and asks the user to visit again later.

3.2.3 Search Flight (by Route)

Use Case ID:	**UC-03**
Use Case Name:	Search Flight (by Route)
Actor(s):	User
Pre-Conditions:	User is logged into the system, system is online, and external flight data API is online.
Priority:	High
Primary Scenario:	User provides information about departure and arrival airports and travel date. All flights, which match the requested route, are shown with up-to-date information on the status
User Steps	**System Response**
1. User clicks on “Search Flight (by Route)” from the menu.	2. System provides user with input fields for Departure Airport, Arrival Airport, and Date.
3. User fills out provided fields and submits the request	4. System validates the input data and makes a request to external API in order to retrieve information about matching flights.
	5. System processes received data, filtering only those flights which match the requested route, and provides corresponding information to user.
Exception Flows	
Actor Action	**System Response**
3a. User enters incorrect data (invalid airport or date)	**4a**. System warns user of the mistake and asks for correction of provided information.
4b. External API does not respond or provides no matches.	**5b**. System informs user that there are no matching flights.

3.2.4 Add Flight to Account

Use Case ID:	**UC-04**
Use Case Name:	Add Flight to Account
Actor(s):	User
Pre-Conditions:	The user is logged into the system and has already searched for flights using valid criteria. The system and external flight API are accessible.
Priority:	High
Primary Scenario:	The user chooses a flight from the list of options, and it is saved in the account of the user. After this, information related to the flight and the time left for take-off is shown.
User Steps	**System Response**
1. The user is presented with the list of flights available after searching for flights.	2. The system provides flight details (number, route, timing, status).
3. The user chooses the flight they wish to add to their account.	4. The system validates the existence of the chosen flight within the returned search results.
	5. The system saves the flight information in the user's account and presents flight status and countdown to departure.
Exception Flows	
Actor Action	**System Response**
3a. The user picks an invalid or outdated flight.	**4a**. The system alerts the user that the flight cannot be saved to their account.
4b. An error occurs while saving the flight due to network or API failure.	**5b**. The system informs the user that there is an issue saving the flight and suggests retrying at a later time.

3.2.5 View Upcoming Flights

Use Case ID:	**UC-05**
Use Case Name:	View Details of Upcoming Flights
Actor(s):	User
Pre-Conditions:	The user has logged into his account and there are at least one flight registered in his account. The application and flight API services are working fine.
Priority:	High
Primary Scenario:	The user selects a specific flight from his list of future flights. The application fetches the details of the chosen flight including all information on airline, aircraft, departure/arrival airport, and flight history.
User Steps	**System Response**
1. User goes to "My Flights" tab	2. System displays all future flights associated with this user.
3. User selects a specific flight to be viewed	4. Application retrieves the details of the selected flight including airline, model of the aircraft used, departure/arrival airports, and the flight schedule
	5. Also, the system will display additional airport information including weather forecast and time zone information along with route history.
Exception Flows	
Actor Action	**System Response**
1a. There are no flights associated with the user account	**2a**. System notifies that there are no future flights
4b. External retrieval of flight details fail	**5b**. System notifies of temporary unavailability of flight details.

3.2.6 Remove Flight From Account

Use Case ID:	**UC-06**
Use Case Name:	Remove Flight from Account
Actor(s):	User
Pre-Conditions:	The user must be logged into his/her account and have at least one flight saved in their account.
Priority:	Medium
Primary Scenario:	The user decides to delete the selected flight from their list of saved flights.
User Steps	**System Response**
1. The user picks up one flight from their "Upcoming Flights" list.	2. The system displays information regarding the selected flight along with the "Remove" option.
3. The user hits the "Remove" button.	4. The system asks for confirmation from the user to remove the selected flight.
5. The user agrees with the process.	6. The system deletes the selected flight from the user's account.
Exception Flows	
Actor Action	**System Response**
3a. The user cancels the removal process when the confirmation dialog pops up.	**4a**. The system cancels the process and retains the selected flight in the user's list.
5b. The system encounters technical difficulties when removing the flight.	**6b**. The system displays an error message to the user.

3.2.7 Archive Completed Flights

Use Case ID:	**UC-07**
Use Case Name:	Archive Completed Flights
Actor(s):	System (Primary), User (Secondary)
Pre-Conditions:	The user has successfully logged into the system and has at least one complete flight according to the departure and arrival times..
Priority:	Medium
Primary Scenario:	The system automatically archives completed flights, that is, the arrival time of the flight has expired, in “Flight Log” for the user. The user can access the details of the archived flight later.
User Steps	**System Response**
1. The user arrives at their destination	2. The system automatically detects that the arrival time of the flight has expired.
	3. The system archives the flight in “Flight Log,” meaning that it moves from the list of “Upcoming Flights.”
4. The user accesses “Flight Log”	5. The system shows a list of archived flights.
Exception Flows	
Actor Action	**System Response**
2a. The system does not detect the completion of the flight.	**3a**. The flight remains in the list of “Upcoming Flights.”
4b. The user manually archives the flight.	**5b**. The system updates the database and archives the flight.

3.2.8 View Travel Log

Use Case ID:	**UC-08**
Use Case Name:	View Travel Log
Actor(s):	User
Pre-Conditions:	The user is logged in and has one archived flight record.
Priority:	Medium
Primary Scenario:	The user navigates to the Travel Log to view the list of completed flights. The system presents a history of the flights and enables filtering by airline, airport, or country.
User Steps	**System Response**
1. The user navigates to the “Travel Log” section.	2. The system displays all archived flight records.
3. The user applies filters for airline, airport, or country.	4. The system displays the filtered results.
5. The user selects one flight record to get detailed information about it.	6. The system displays the selected flight information and its journal entry.
Exception Flows	
Actor Action	**System Response**
1a. The user has no archived flights.	**2a**. The system displays a message stating “No past flights available.”
4b. Filters return no matching results.	**5b**. The system displays a message indicating “No results found.”

3.2.9 Edit Journal Entry (Notes)

Use Case ID:	**UC-09**
Use Case Name:	Edit / Delete Journal Entry
Actor(s):	User
Pre-Conditions:	The user must be logged into the website and has a completed journal entry for a particular flight.
Priority:	Medium
Primary Scenario:	Passenger selects the available flights from the Travel Log, sees the journal entry, and edits/deletes it if needed.
User Steps	**System Behavior**
1. User chooses the "Travel Log" tab and selects the flight for which he/she has to update the journal entry.	2. Displays the journal entry related to the chosen flight with 'Edit' and 'Delete' links.
3. User wishes to edit the entry and makes necessary changes to the entry	4. Validates and executes the entry as per the user's request.
5. The user desires to delete the journal entry	6. Requests the user to confirm his decision.
7. User confirms his decision	8. Deletes the journal entry and shows the confirmation message.
Exception Flows	
Actor Action	**System Behavior**
3a. User fails to edit the journal entry and proceeds further.	4a. Keeps the last recorded entry intact.
7b. User refuses to confirm his decision	8b. Aborts the process.
4c. Failure happens while updating/deleting the journal entry	5c. Tells the user of the error and advises him to try again.

3.2.10 View Airport Amenities

Use Case ID:	**UC-10**
Use Case Name:	View Airport Amenities
Actor(s):	User
Pre-Conditions:	The user is logged into the system and has chosen one of the departure/arrival airports associated with a current or future flight. The application and airport API have been made available.
Priority:	Medium
Primary Scenario:	The user chooses an airport in order to check the amenities available there. The system retrieves information from the various sources and presents the amenities list categorized accordingly.
User Steps	**System Behavior**
1. The user selects an airport from their flight details or search results.	2. The system requests available amenities for the chosen airport via external APIs.
3. The user views the amenities list.	4. The system displays categorized results such as lounges, Wi-Fi access, restaurants, cafés, and retail shops.
	5. The system may also display ratings, maps, or location details for each facility.
Exception Flows	
Actor Action	**System Behavior**
1a. The user selects an airport with incomplete or unavailable amenity data.	2a. The system displays a message: “Amenities data unavailable for this airport.”
2b. API call fails or returns an error.	3b. The system shows an error message and suggests retrying later.

3.2.11 Get Insights about Destination

Use Case ID:	**UC-11**
Use Case Name:	Get Destination Insights
Actor(s):	User
Pre-Conditions:	The user has an upcoming or selected destination city. The system has access to external APIs for tourism or location data.
Priority:	Medium
Primary Scenario:	User requires information regarding their destination city. The system provides information related to their destination city such as places of attraction, cultural aspects, and fun activities
User Steps	**System Behavior**
1. The user chooses the “Destination Insights” feature on the basis of their booked flights or destination information.	2. The system accesses destination information through travel and tourism APIs.
3. The user reviews the information provided.	4. The system provides information on top attractions, culture points, events, and sports and recreation options.
	5. The system may provide mapping or booking information.
Exception Flows	
Actor Action	**System Behavior**
2a. The selected destination has limited or outdated data.	3a. The system displays a message stating “Limited destination data available.”
2b. API call fails or times out.	3b. The system displays an error message prompting the user to retry later.

3.2.12 View Visa Requirements

Use Case ID:	**UC-12**
Use Case Name:	Visa Assistance
Actor(s):	User
Pre-Conditions:	The user is signed in and has specified their nationality information while registering. The system is active and connected to the database that contains the foreign visa requirements.
Priority:	High
Primary Scenario:	The user chooses his/her flight or destination city. The computer verifies the nationalities of the passenger and the destination nation, fetches information on visas and travel requirements, and presents the findings to the user.
User Steps	**System Behavior**
1. The passengers can open up the "Visa Assistance" option or even view it automatically by selecting the flight.	2. The system reads the stored nationality and destination data.
3. The user reviews displayed visa details.	4. The system fetches visa-requirement data from external APIs (e.g, Passport Index) and displays whether a visa is required, visa-on-arrival eligibility, or e-visa options.
	5. The system provides helpful links or references to embassy or e-visa websites.
Exception Flows	
Actor Action	**System Behavior**
2a. User has not provided nationality.	3a. The system prompts the user to update their profile with nationality information.
4b. API service fails or returns incomplete data.	5b. The system displays an error message and advises retry later.

3.2.13 View Weather & Time zone difference

Use Case ID:	**UC-13**
Use Case Name:	View Weather & Time-Zone Differences
Actor(s):	User
Pre-Conditions:	At least one flight or airport is chosen by the user. The app has internet access.
Priority:	Medium
Primary Scenario:	User Selects a flight/airport in order to see their current conditions. The system uses external APIs and fetches information on the current weather, temperature, and time zones at the airports of departure and arrival.
User Steps	**System Behavior**
1. The user opens the screen with a selected flight or airport information	2. The system invokes external APIs for obtaining weather forecasts and time-zones
3.The user sees the retrieved information	4. The system displays the results containing weather condition, temperature, local time, time difference between airports.
	5. The system refreshes weather info every once in a while.
Exception Flows	
Actor Action	**System Behavior**
1a. User opens the feature offline.	2a. The system presents stored data, or informs that updates require an internet connection.
2b. An error occurs at an API call	3b. The system displays an error.

3.2.14 Receive Alerts about flights

Use Case ID:	**UC-14**
Use Case Name:	Receive Flight Alerts
Actor(s):	User
Pre-Conditions:	The user needs to be signed in to the application, having some upcoming flights on his account, and notifications should be enabled. The system needs to be connected to real-time flight data API.
Priority:	High
Primary Scenario:	The system will check flight details periodically for any delay, gate changes, or cancellation of the flights and let the user know. This can happen via the application itself or push or mail.
User Steps	**System Behavior**
1. There are one or more saved flights in his profile.	2. Periodically system checks flight data via the API call.
3. If there is any modification in the flight status.	4. Notification is made to the user if any changes happened.
5. The user views the alert.	6. System will show updated data along with highlighting the flight which got changed.
Exception Flows	
Actor Action	**System Behavior**
1a. If the notifications are disabled.	2a. The system logs the event but does not send an alert.
4b. If the API did not provide any updates on flight details.	5b. Retry will be done periodically.

3.2.15 Manage Notification Preferences

Use Case ID:	**UC-15**
Use Case Name:	Manage Notification Settings
Actor(s):	User
Pre-Conditions:	The user is authenticated and has configured one or more notifications on their profile.
Priority:	Medium
Primary Scenario:	The user accesses their profile settings to enable, disable, or modify notification preferences (push, email, or in-app). The system updates and stores these preferences.
User Steps	**System Behavior**
1. The user navigates to the “Notification Preferences” page in the application	2. The system displays the list of notification options available for configuration.
3. The user toggles notification preferences.	4. The system configures and saves their preferences.
5. The user exits the “Notification Preferences” page.	6. The system configures their preferences going forward..
Exception Flows	
Actor Action	**System Behavior**
3a. The user switches off their notifications.	4a. The system prompts a warning message before saving their preferences.
4b. An issue with the network connection or backend when configuring the preferences.	5b. The system notifies the user about the error and retains the previously saved preferences.

3.2.16 Update Profile

Use Case ID:	**UC-16**
Use Case Name:	Update Profile
Actor(s):	User
Pre-Conditions:	The user should be logged into his/her account having an active profile within the system.
Priority:	Medium
Primary Scenario:	The user edits profile settings by changing personal information such as name, country or authentication preferences. The system updates profile information upon validation.
User Steps	**System Behavior**
1. User visits the “Profile Settings”	2. System pulls out existing profile information.
3. User edits personal information (name, country, authentication method).	4. System validates information provided.
5. User saves updated information.	6. System updates profile successfully.
Exception Flows	
Actor Action	**System Behavior**
3a. User inputs incomplete or incorrect information.	4a. System marks erroneous fields with warnings.
4b. Technical problem occurs when saving changes.	5b. System generates a warning message and rolls back the transaction.

3.2.17 Delete Account

Use Case ID:	**UC-19**
Use Case Name:	Delete Account
Actor(s):	User
Pre-Conditions:	User must be logged in and have the system accessible.
Priority:	High
Primary Scenario:	The user makes a decision to permanently delete his/her account. The system warns and deletes data from the user's account.
User Steps	**System Behavior**
1. The user clicks on "Account Settings" and selects "Delete Account”	2. The system provides a warning that it will lose all your data.
3. The user accepts the warning	4. The system deletes all accounts' information (Profile Information, Flights, Logs, Journals).
	5. The system confirms and navigates the user to the homepage or log-in page.
Exception Flows	
Actor Action	**System Behavior**
3a. The user clicks "Cancel" on deleting his/her account.	4a. The system does not delete the account.
4b. The system cannot delete the account because of some back-end errors.	5b. The system gives a warning that the deletion has failed and it retains the account data until the deletion process is successful.

3.3 Functional Requirements

The system provides the following functional capabilities:

3.3.1 User Authentication

- Users shall be able to register and log in securely.
- The system shall authenticate users using Firebase Authentication.
- Users shall remain logged in across sessions unless they log out OR Delete Account.

3.3.2 Searching Flights

- It shall be possible to search flights using:
    - Flight Number
    - Departure and Arrival Airports
- The software shall show search result for flights.
- The software shall allow searching flights for a duration of 10-12 hours in future from the present time.

3.3.3 Details of the Flights

- The system must provide flight information as follows:
    - Timings (departure and arrival times)
    - Flight duration
    - Airline information (flight details)
    - Details about flight status
- The system must also provide weather information at departure and arrival locations.

3.3.4 Manual Flight Entry

- Users shall be able to manually add flight details.
- Users shall input:
    - Flight number
    - Date
    - Departure and arrival airports
- The system shall store manually entered flights in the user profile.

3.3.5 Airport Module

- Users shall be able to search for airports.
- The system shall display airport details including:
    - Location
    - Weather
    - Flight board (departures/arrivals)
- The system shall provide access to airport amenities (e.g., restaurants, lounges, services).

3.3.6 Airline Module

- Users will be allowed to see information about airlines.
- The system will show information about airlines like their names and codes.

3.3.7 Travel Log / Notes

- Users shall be able to add notes related to flights.
- Users shall be able to store and edit personal travel logs.

3.3.8 Statistics Dashboard

The system will incorporate travel statistics in the following manner:

- Total number of flights
- Distance traveled
- Total Duration of Flight taken
- Total number of airports Visited
- Total Number of airlines Flown

3.3.9 Flight Management

- The user will be able to:
    - See saved flights
    - Check upcoming flights
    - See past flights

3.4 Interface Requirements

3.4.1 User Interface

- The system shall provide a **mobile-based interface** built using Flutter.
- The UI shall follow a **minimal, modern design**.
- The interface shall support:
    - Theme for dark mode
    - Easy navigation across screens
    - Scrollable layouts and responsive design

3.4.2 Software Interface

- The system shall integrate with:
    - Flight data APIs (for schedules and tracking)
    - Weather APIs (for airport weather data)
- Firebase Infrastructure and Services shall be used for:
    - Authentication
    - Database storage

3.4.3 Hardware Interface

- The application should work on Android and iOS operating systems.
- No specialized hardware is required

3.5 Database Requirements

The system shall maintain a database to store user-related and flight-related data.

Stored Data Includes:

- User profiles
- Saved flights

Database Technology:

- Azure SQL (SQL Server)

Thus ensuring:

- Quick access to data
- Secure user-specific data storage
- Scalability for future expansion

3.6 Non-Functional Requirements

3.6.1 Performance

- System will react to user inputs in timely manner.
- Search results shall be displayed quickly.

3.6.2 Usability

- System will have user-friendly GUI.
- System navigation will be simple and consistent.

3.6.3 Reliability

- The system shall operate without crashes or failures.
- System data will be reliably stored and accessed.

3.6.4 Security

- System will ensure safe authentication of users.
- System user data will be kept secure and accessible only for Authorized users.

3.6.5 Scalability

- The system shall support increasing users and data.
- Backend services shall be scalable using cloud infrastructure.

3.6.6 Availability

- System shall be available at all times except during maintenance.

3.7 Project Feasibility

3.7.1 Technical Feasibility

The project is technically feasible due to the availability of:

- Flutter for cross-platform development [3]
- Firebase for authentication and database
- Azure for backend hosting [4]
- External APIs for flight and weather data

3.7.2 Operational Feasibility

The system is user-friendly and requires minimal training. Users can easily interact with the application through a mobile interface.

3.7.3 Legal & Ethical Feasibility

- The system uses publicly available APIs.
- User data is handled securely and ethically.
- No copyright or licensing violations are involved.

3.8 Conclusion

This chapter outlined the functional and non-functional requirements of the system along with interface and database specifications. These requirements form the foundation for the system design and implementation discussed in the next chapter.

4. System Design

This chapter outlines the design of the Horizon – Flight Tracker & Journal software application. This includes an outline of the system architecture, components, data storage and retrieval techniques, and UI design.

4.1 Design Approach

The system adheres to a design that is modular and service-oriented.

The frontend is implemented using Flutter, with an emphasis on component reuse and responsiveness. The backend design is based on REST architecture with the use of Node.js and Azure. *[3] [4]*

Some of the key design considerations are:

- Separation of concerns (UI, Logic, and Data)
- Component reuse
- Scalability using the cloud
- A user-friendly user interface design

4.2 Design Constraints

- Flutter development should be utilized for cross-platform support
- Using external APIs for real-time updates is necessary
- The app relies on internet connectivity for most of its functions
- The UI should continue to be responsive in different device screen sizes

4.3 System Architecture

The system follows a **three-tier architecture**:

1. Presentation Layer (Frontend)

- Built using Flutter
- Handles user interaction and UI rendering

2. Application Layer (Backend)

- Node.js backend hosted on Azure
- Handles business logic and API integration

3. Data Layer (Database)

- Firebase / Azure SQL
- Stores user data, flights, and logs

External Services

- Flight APIs (for real-time flight data) [1]
- Weather APIs (for airport conditions) [5]

Description for High Level Architecture Diagram (Figure 4.3.1)

This diagram depicts the three-tier architecture of the system. The architecture is divided into a **Presentation Layer** representing the frontend built with *Flutter*, an **Application Layer** detailing the *Node.js* backend hosted on *Azure App Service*, and a **Data Layer** that utilizes *Firebase* and *Azure SQL* for data storage. Additionally, the diagram highlights the system's reliance on external services, specifically integrating flight and weather APIs for real-time data retrieval.

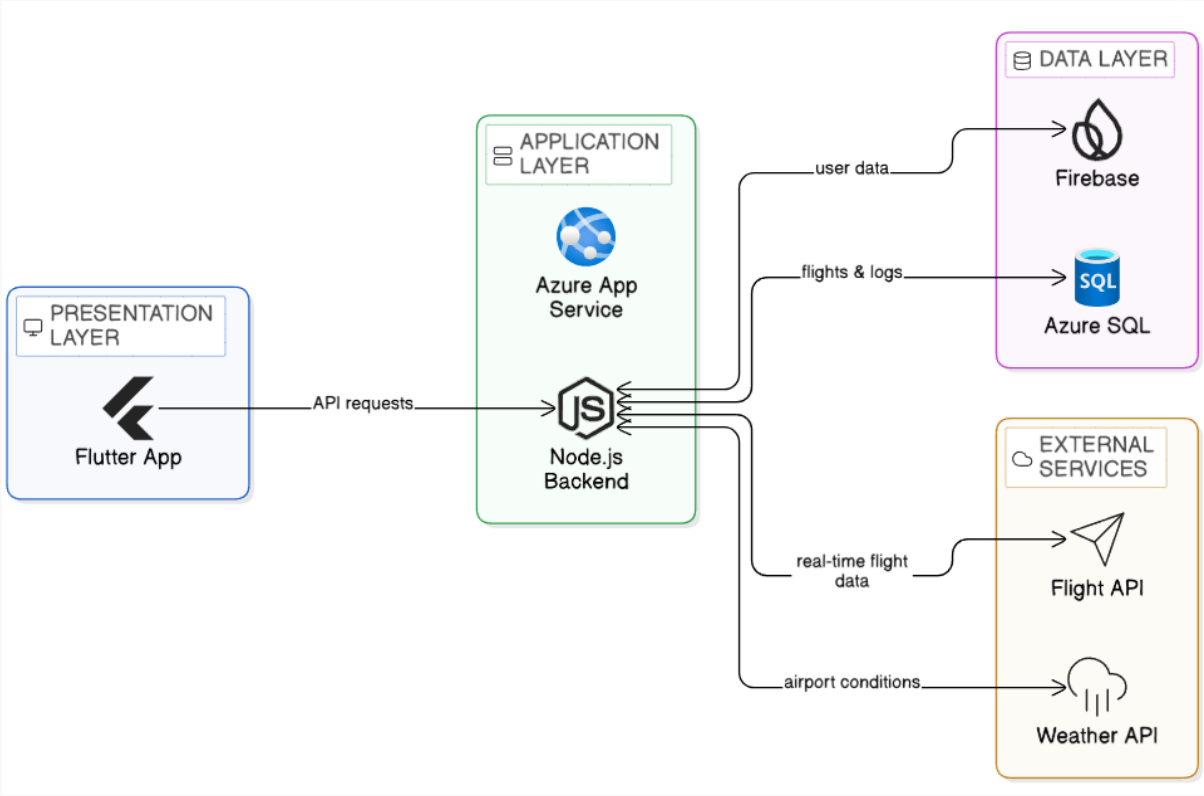


Figure 4.3.1: High Level Architecture Diagram

4.4 Logical Design

The logical design defines the structure of system components and their relationships.

Description for Logical System Design - Class Diagram (Figure 4.4.1)

The Class Diagram defines the static structure of the system's components and their interrelationships

It maps out the main entities comprising the system, primarily the **User, Flight, Airport, and Airline.**

The diagram also illustrates how these entities interact, demonstrating that a single user can be linked to multiple flights, each flight is connected to specific airports and airlines, and completed flights can contain corresponding journal entries.

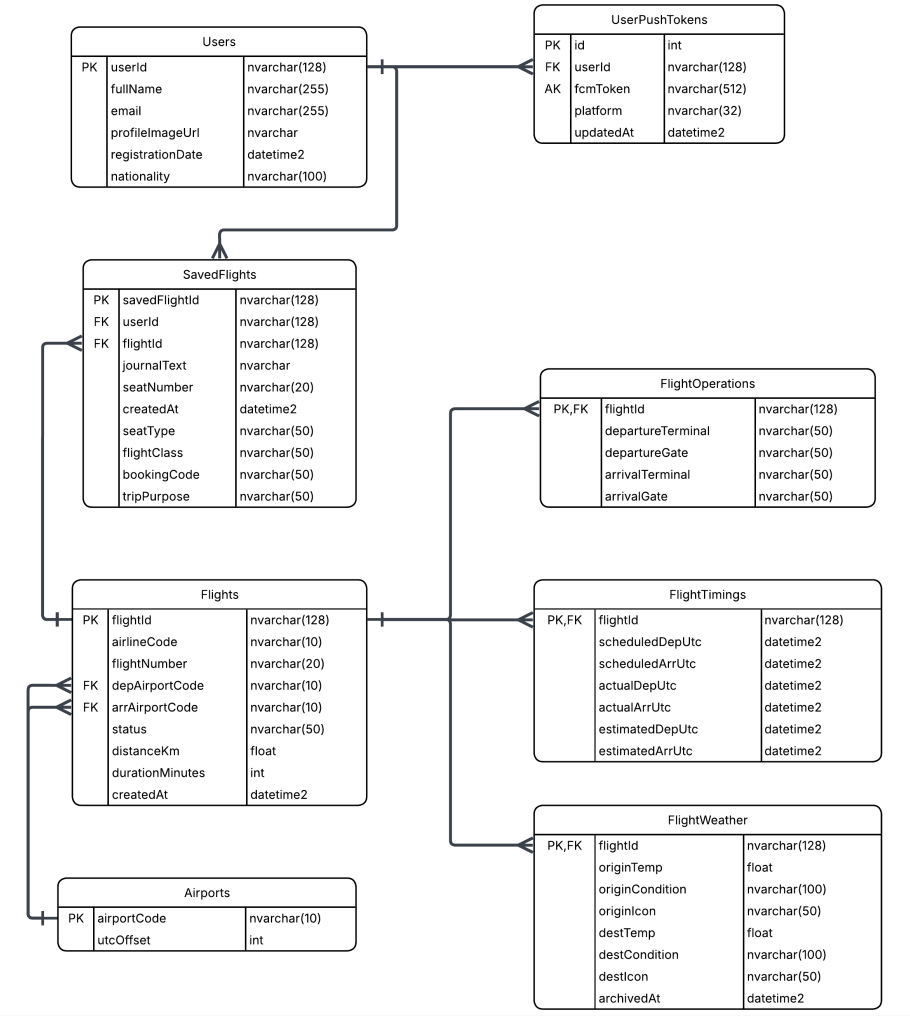


Figure 4.4.1: Logical System Design - Class Diagram

4.5 Dynamic View

The dynamics of the system have been modeled using sequence and activity diagrams.

Key Processes:

- Flow for Searching Flights
- Adding a Flight
- Details of Flights
- Travel Logs Management

These diagrams show interactions among various components at runtime.

Description for Sequence Diagrams (Figures 4.5.1 - 4.5.5)

The following Sequence Diagrams model the dynamic behavior of the system by illustrating the step-by-step interactions between various components at runtime.

These diagrams detail the chronological flow of messages for key system processes, including searching for and adding a flight, viewing specific flight details, removing a flight, accessing travel logs and analytics, and manually editing flight data.

Diagram 1: Add a Flight to User Profile (Search → Select → Save)

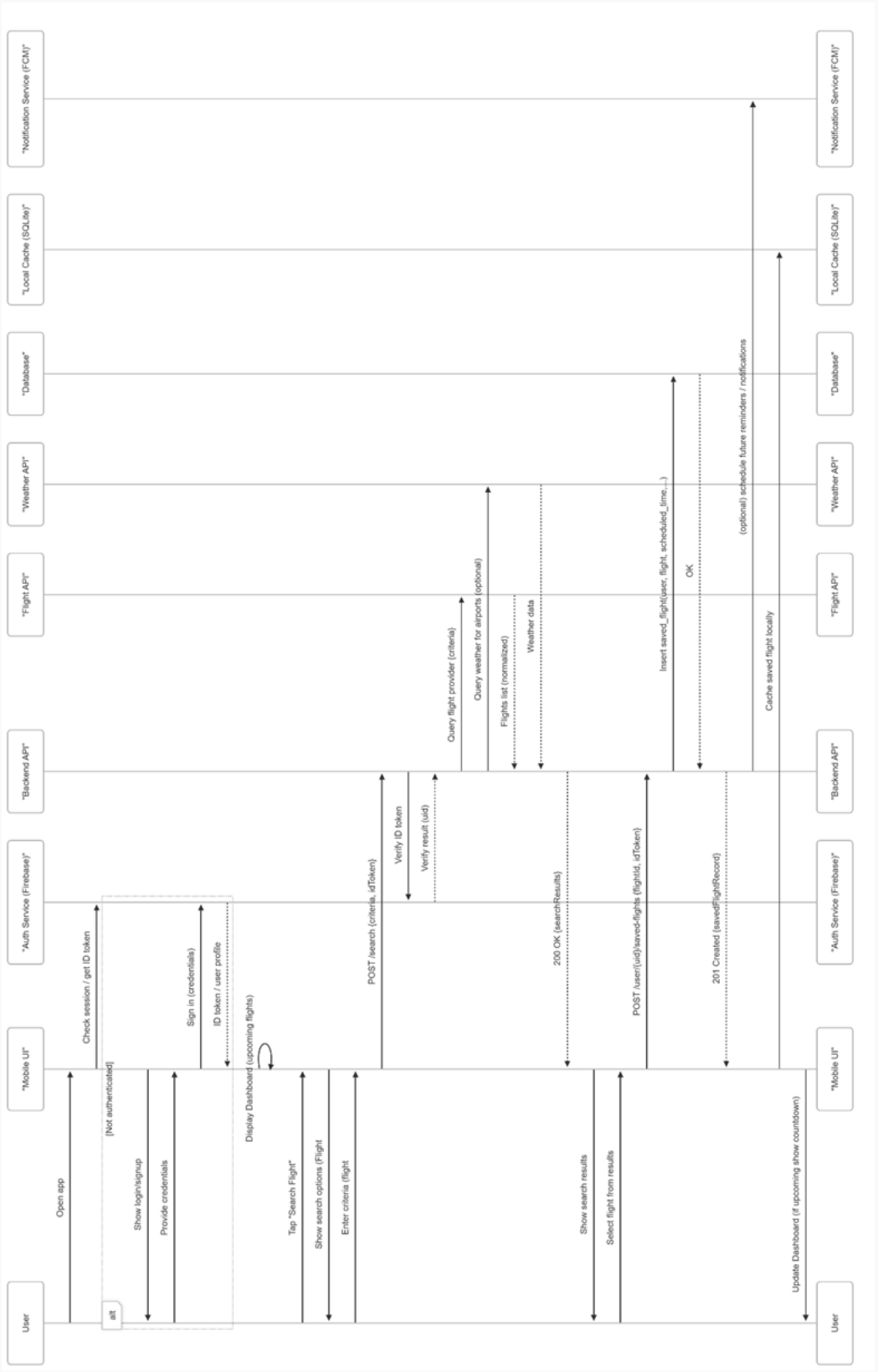


Figure 4.5.1: Add a Flight to User Profile – Sequence Diagram

Diagram 2: View Details of an Upcoming Flight from Dashboard

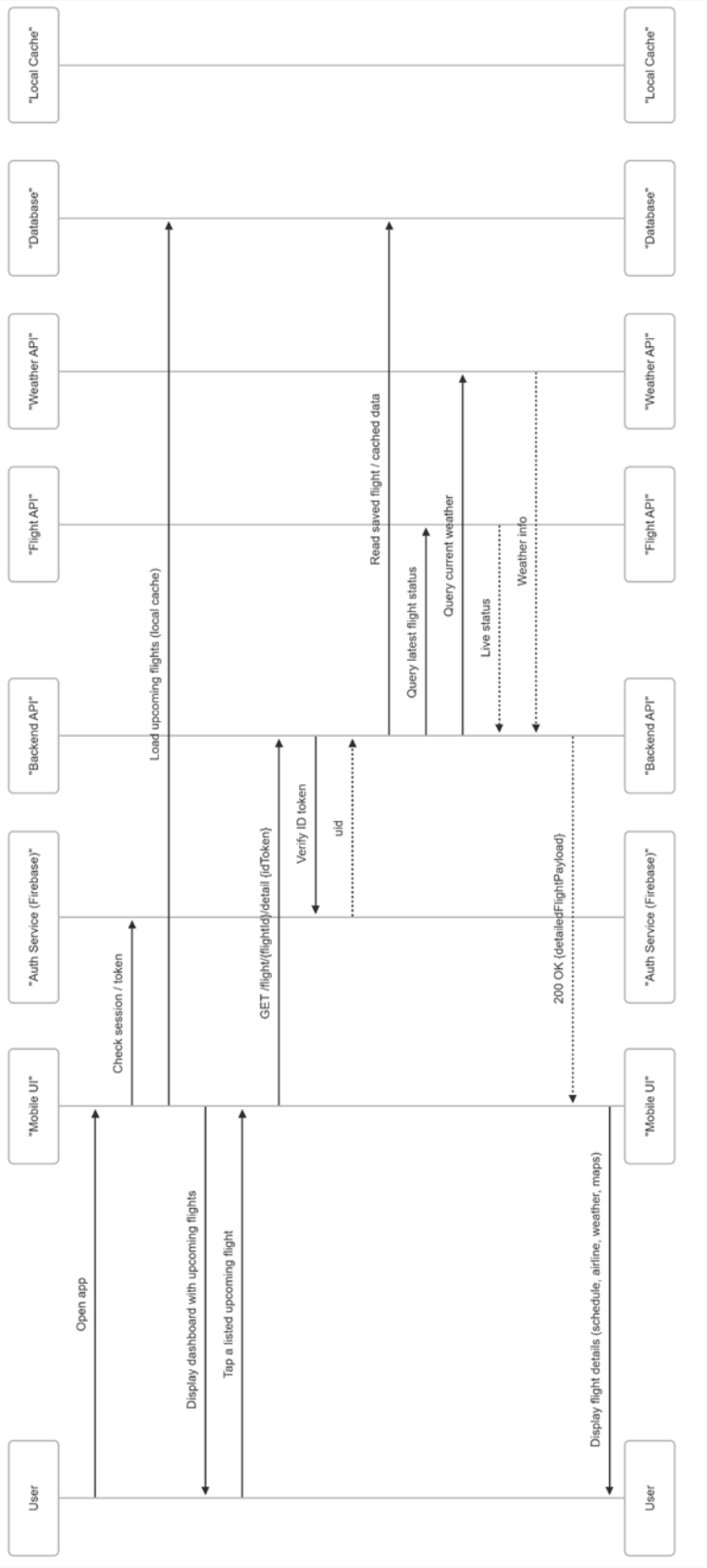


Figure 4.5.2: View Details of an Upcoming Flight from Dashboard – Sequence Diagram

Diagram 3: Delete a Flight (from Dashboard or Flight Log)

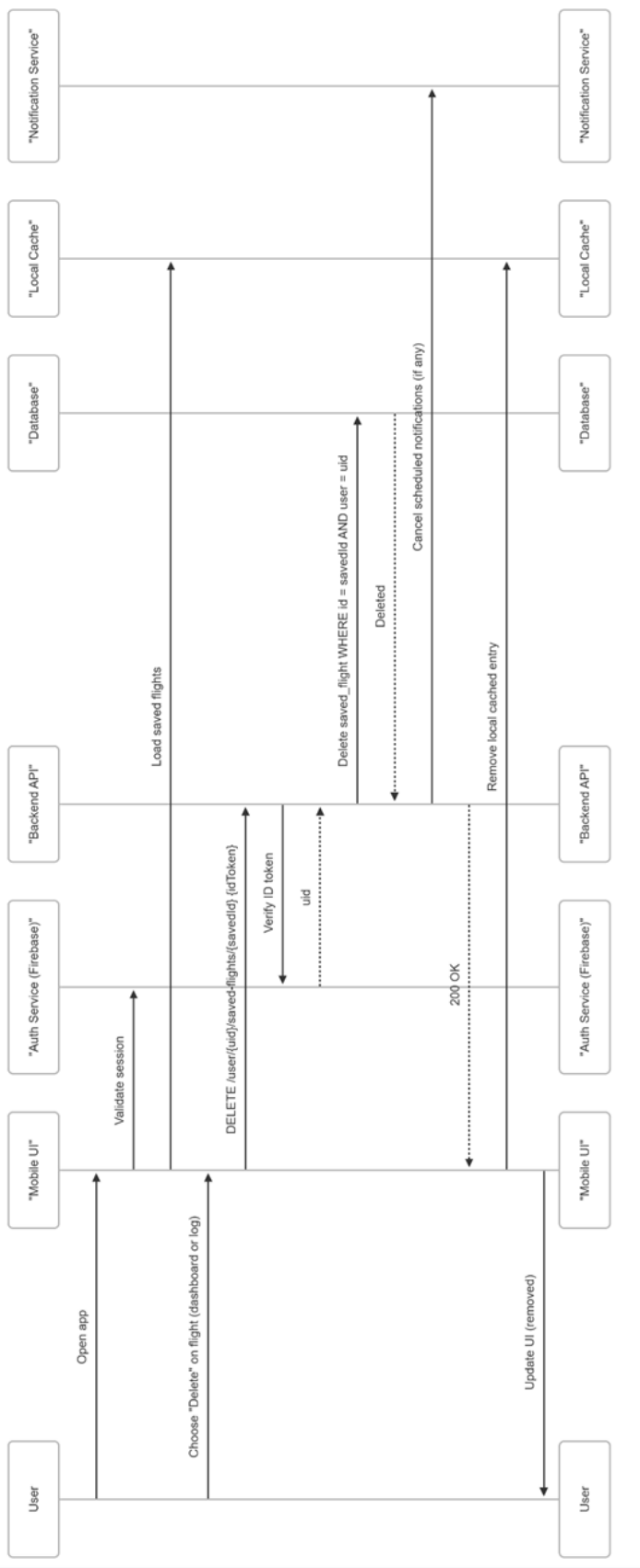


Figure 4.5.3: Delete a Flight (from Dashboard or Flight Log) – Sequence Diagram

Diagram 4: View Flight Log/Analytics

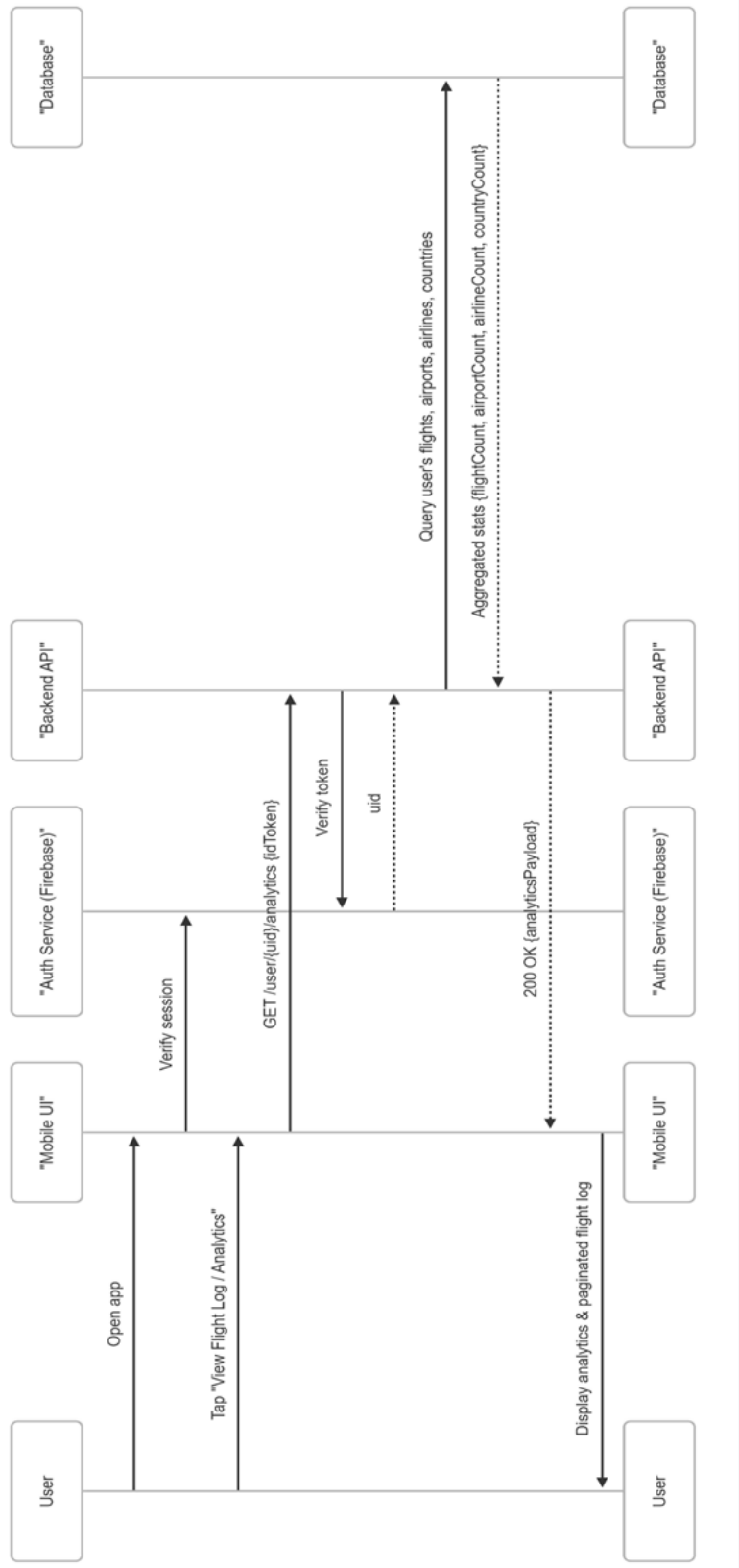


Figure 4.5.4: View Flight Log/Analytics Profile – Sequence Diagram

Diagram 5: Manually Edit Flight Data

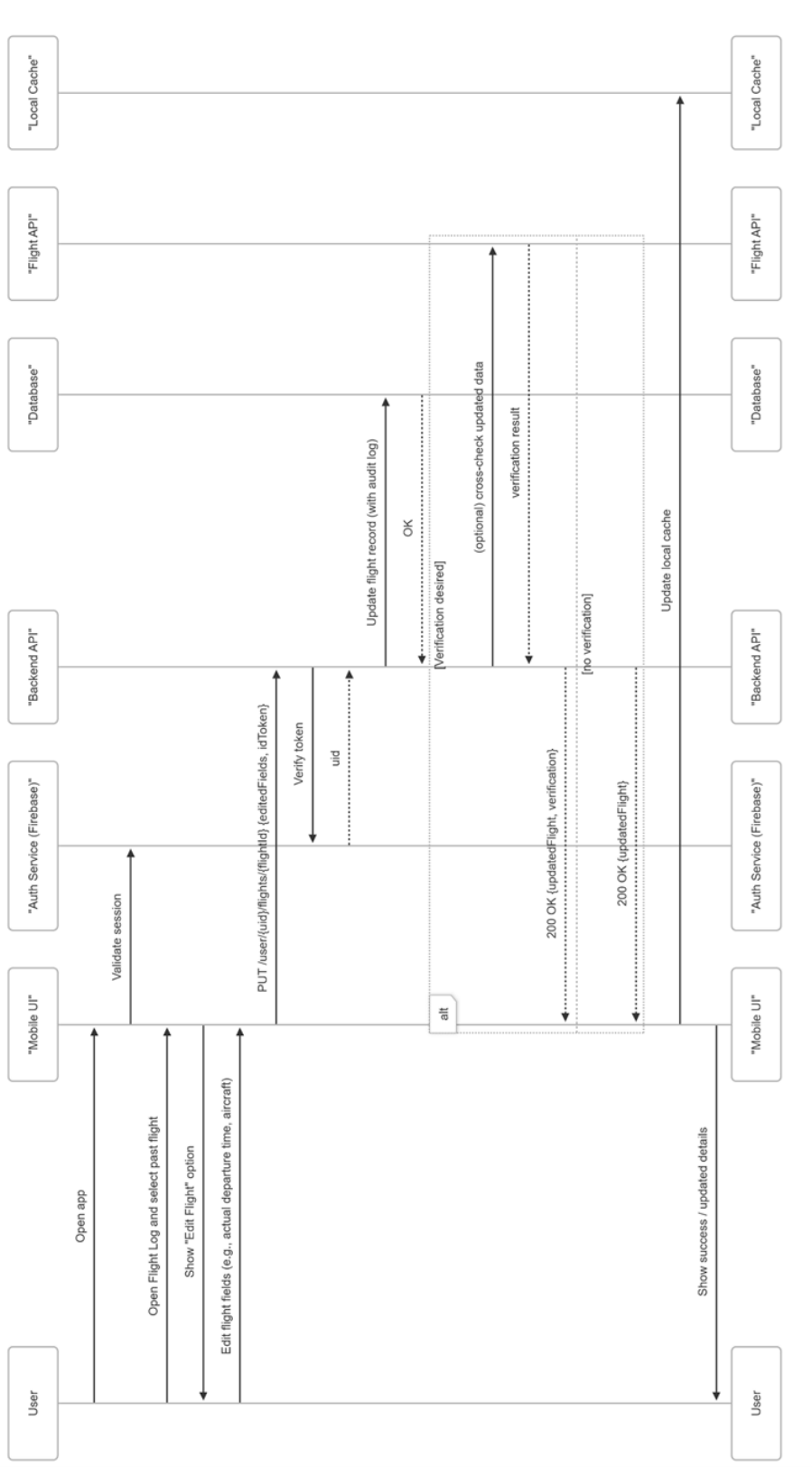


Figure 4.5.5: Manually Edit Flight Data – Sequence Diagram

4.6 Component Design

The system is divided into the following modules:

4.6.1 User Management Module

- Handles authentication and profile management
- Uses Firebase Authentication

4.6.2 Flight Search Module

- Handles search by flight number and route
- Communicates with external APIs

4.6.3 Flight Management Module

- Saves and tracks flights
- Displays upcoming and completed flights

4.6.4 Travel Log Module

- Stores notes and travel history
- Allows editing and viewing logs

4.6.5 Airport & Airline Module

- Displays airport and airline details
- Includes amenities and additional info

4.6.6 Statistics Module

- Calculates user travel insights
- Displays analytics such as distance and flights

Package Diagram:

Description for Package Diagram (Figure 4.6.1)

The Package Diagram illustrates the modular division of the system into distinct functional components. It visually organizes the application into core modules:

- User Management Module for authentication
- Flight Search Module for API-driven queries, and
- Flight Management Module for tracking saved flights.

It also outlines the Travel Log Module for journaling, the Airport & Airline Module for facility details, and the Statistics Module responsible for calculating user travel insights.

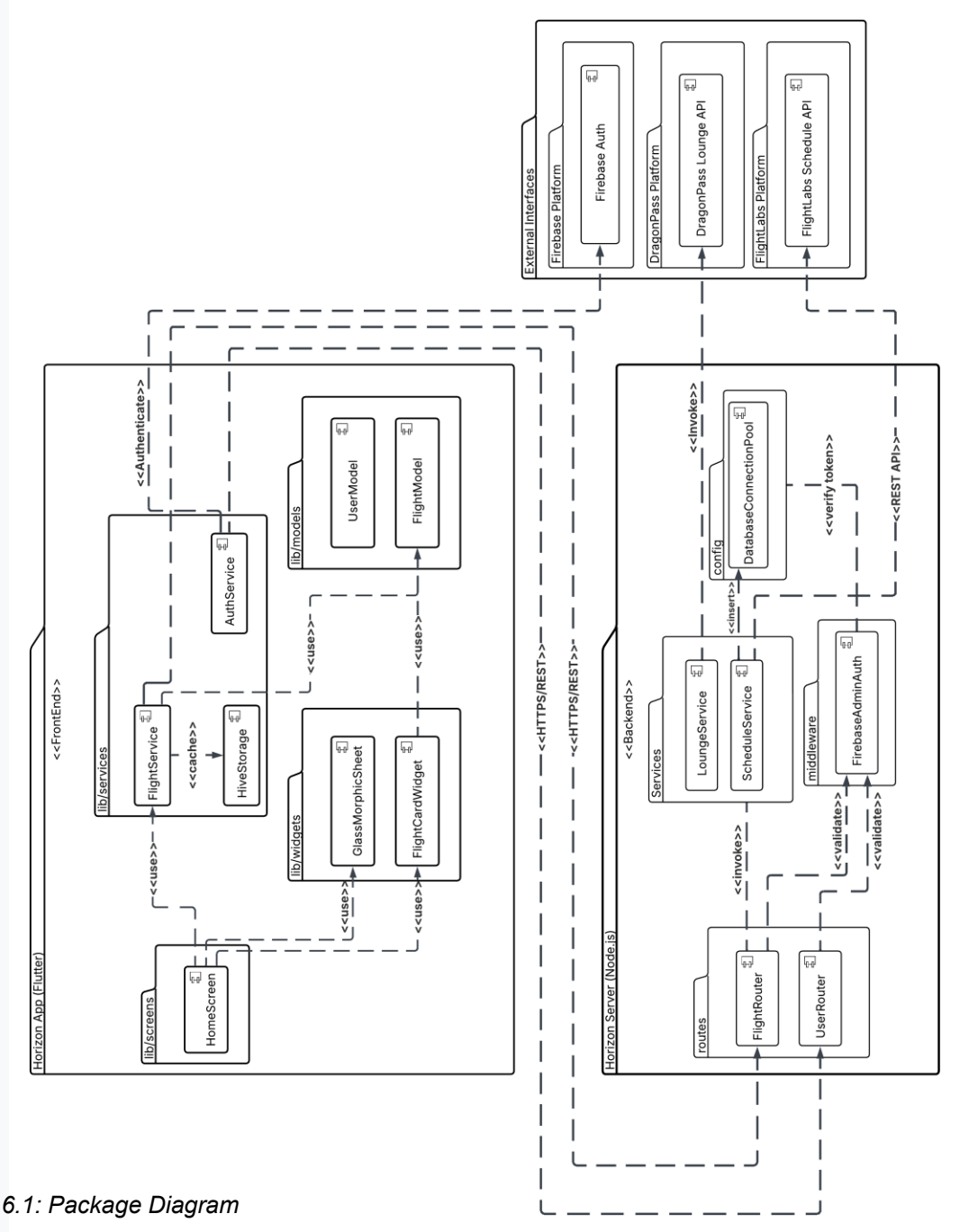


Figure 4.6.1: Package Diagram

Deployment Diagram:

Description for Deployment & ER Diagrams (Figures 4.6.2)

The Deployment Diagram provides a high-level view of how the software artifacts are mapped to the physical or cloud infrastructure, detailing the distribution of the application's components.

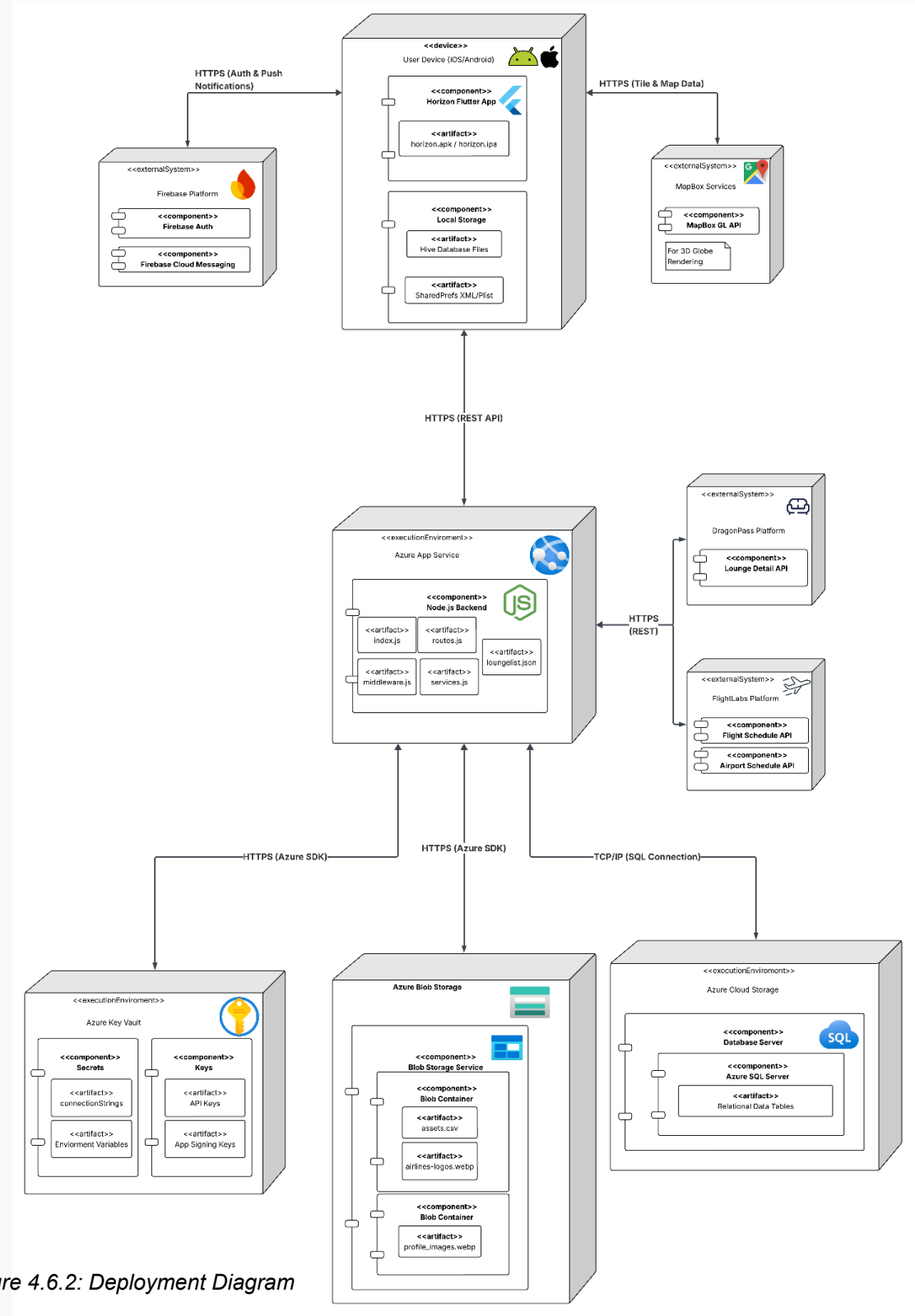


Figure 4.6.2: Deployment Diagram

4.7 Data Models

Description for Deployment & ER Diagrams (Figures 4.6.2 & 4.7.1)

The Entity-Relationship (ER) Diagram represents the structured data models utilized by the system to effectively manage and store information across the database.

ER Diagram:

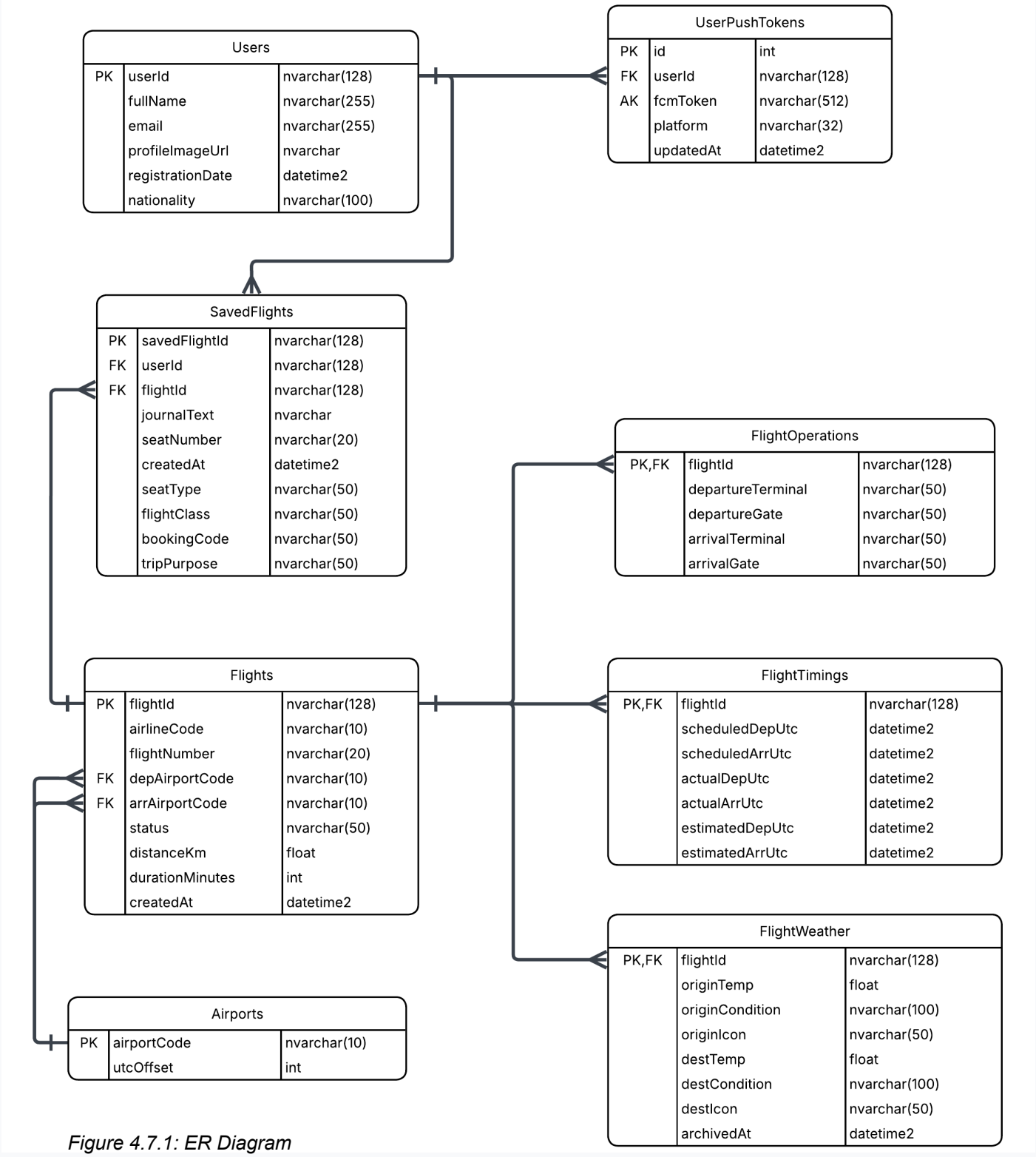


Figure 4.7.1: ER Diagram

4.8 User Interface Design

Modern aesthetics and usability was the designated focus for designing this UI.

4.8.1 Design Characteristics:

- Dark theme interface
- Minimal and clean layout
- Smooth navigation between screens
- Consistent typography and spacing

4.8.2 Key Screens Designed:

4.8.2.1 Search Flight Screen

- Search bar for flight number or airport
- Suggestions for airlines and airports
- Dynamic search results

4.8.2.2 Flight Code Entry Screen

- Input field for flight number
- Date selection option

4.8.2.3 Airport Selection Screen

- Searchable airport list
- Selection for departure and arrival

4.8.2.4 Flight Results Screen

- Displays list of available flights
- Cards showing flight details

4.8.2.5 Flight Details Screen

- Detailed flight information
- Weather and airport data
- Options for adding notes

4.8.2.6 Travel Log Screen

- Displays past flights
- Allows journaling

4.8.2.7 Statistics Dashboard

- Shows user insights such as:
    - Total flights
    - Airlines Used
    - Airports Visited

Description for User Interface Design Screens (Figures 4.8.1 - 4.8.21)

The following figures showcase the modern aesthetics and usability of the application's user interface. The design strictly adheres to a minimal and clean layout featuring a dark theme, consistent typography, and responsive scrollable screens.

The visual prototypes map out core user journeys, including dynamic flight search results, comprehensive flight details with weather data, a travel log for journaling, and a robust statistics dashboard displaying metrics like total flights, top airports, and routes.

Additionally, specialized modules for layover details, visa requirements, and airport lounge access are visually depicted to highlight the app's holistic travel management capabilities.

Figure 4.8.1: Login Screen Interface

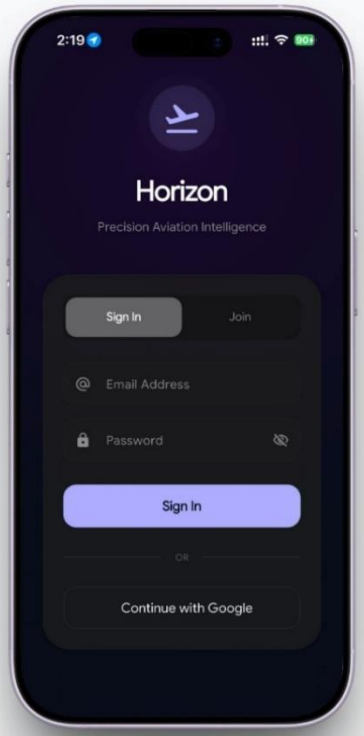


Figure 4.8.2: Flights Dashboard with Map View

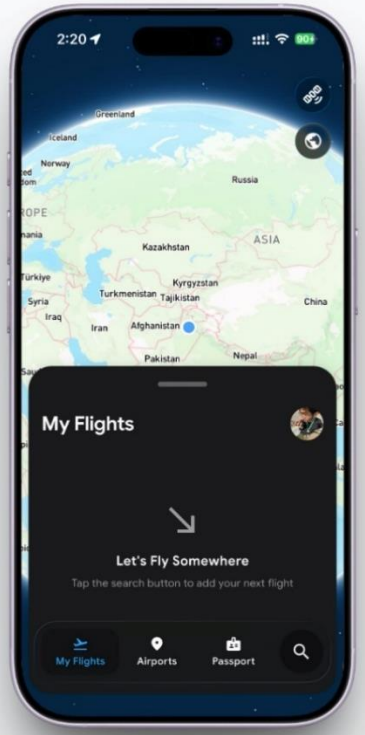
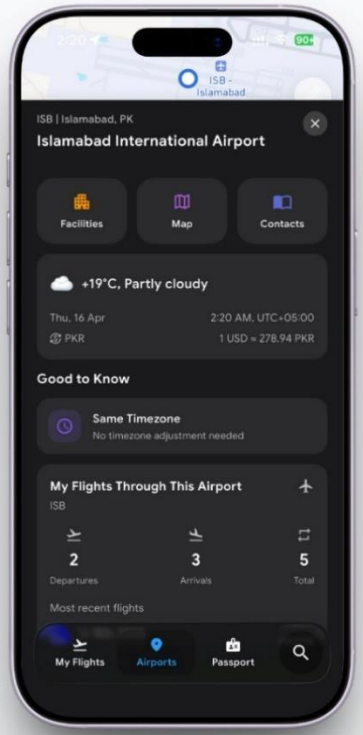


Figure 4.8.3 Airport Information & Insights

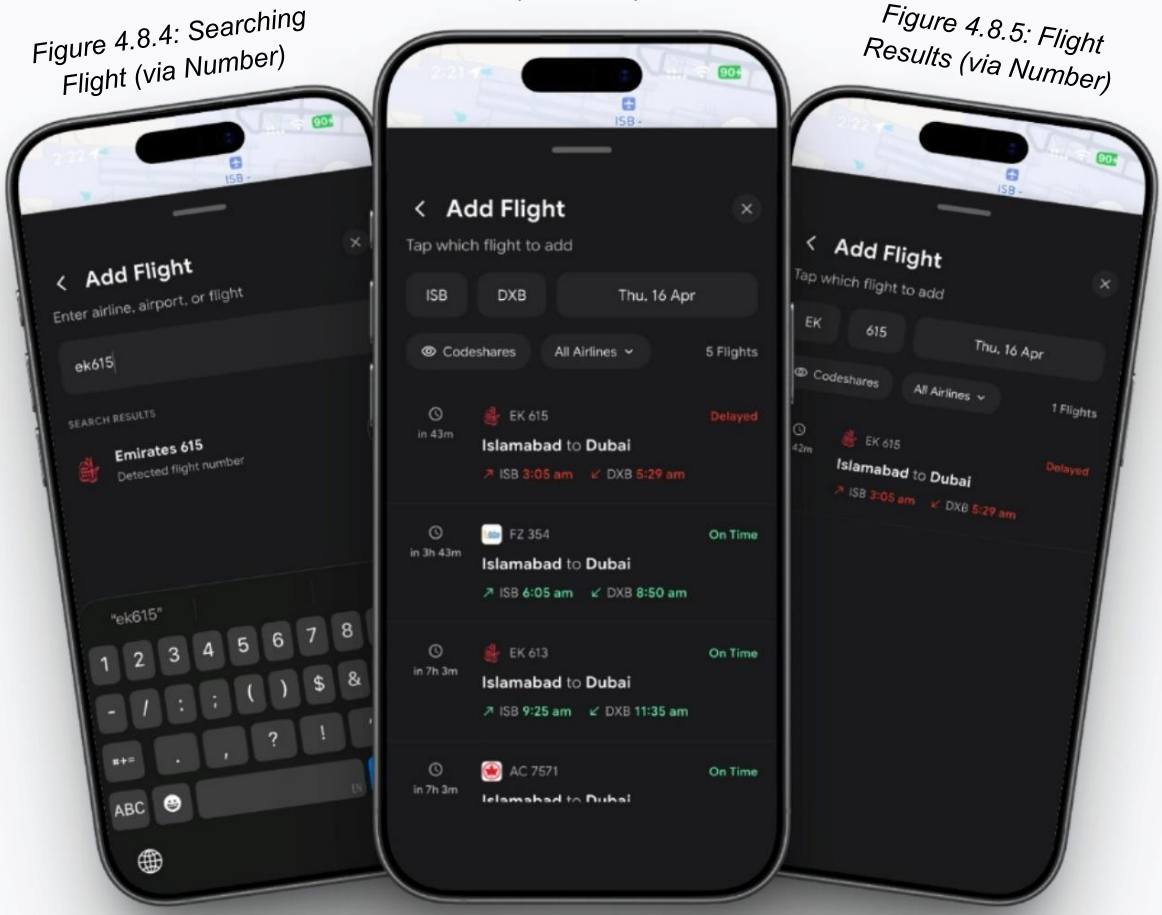


These figures showcase selected screens and their key functions like login, flight information, and airport details that reflect the modern nature of the system design.

Figure 4.8.6: Flight Results (via Route)

Figure 4.8.4: Searching Flight (via Number)

Figure 4.8.5: Flight Results (via Number)



The figures demonstrate several steps in the flight search process, such as querying and the dynamic representation of flight results.

Figure 4.8.7: Flight Overview

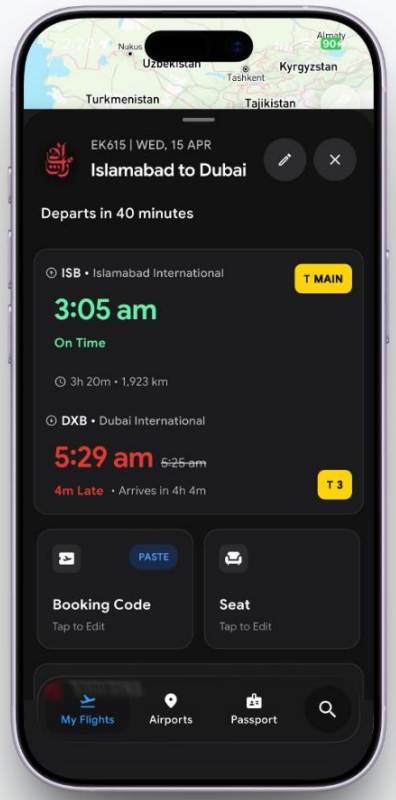


Figure 4.8.8: Airline Details

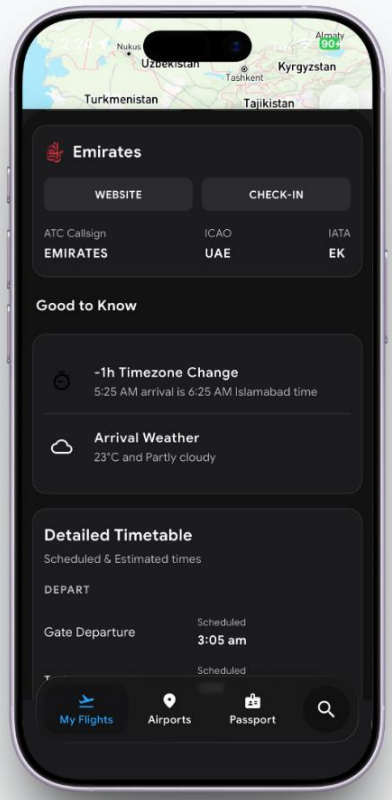
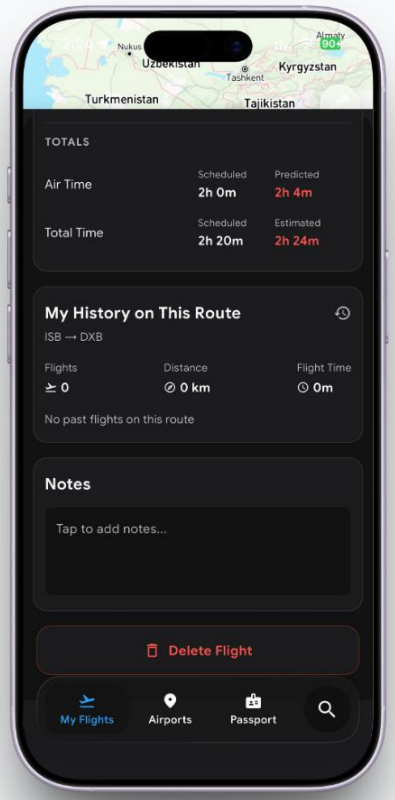
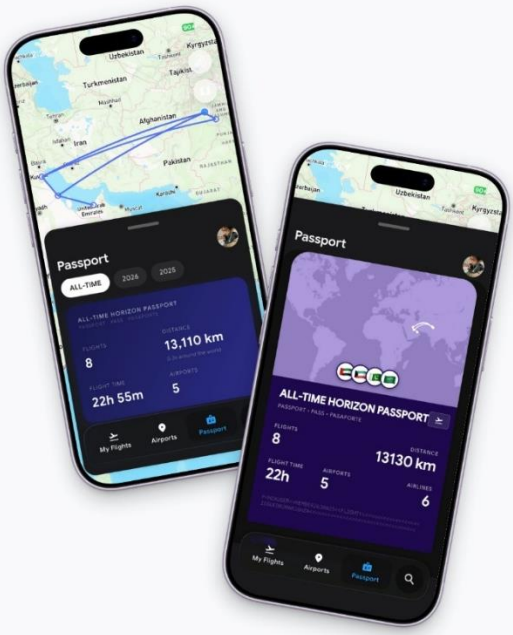


Figure 4.8.9: Flight Stats and Notes



The following figures depict screens on the flight details interface containing timing, airline information, and personal details like history and notes of the user.



The figures illustrate the passport module, highlighting the visual travel map and a summarized dashboard of user flight statistics and travel history.

Figure 4.8.10: Travel Map Overview

Figure 4.8.11: Flight Passport Dashboard

Figure 4.8.12: Top Airports

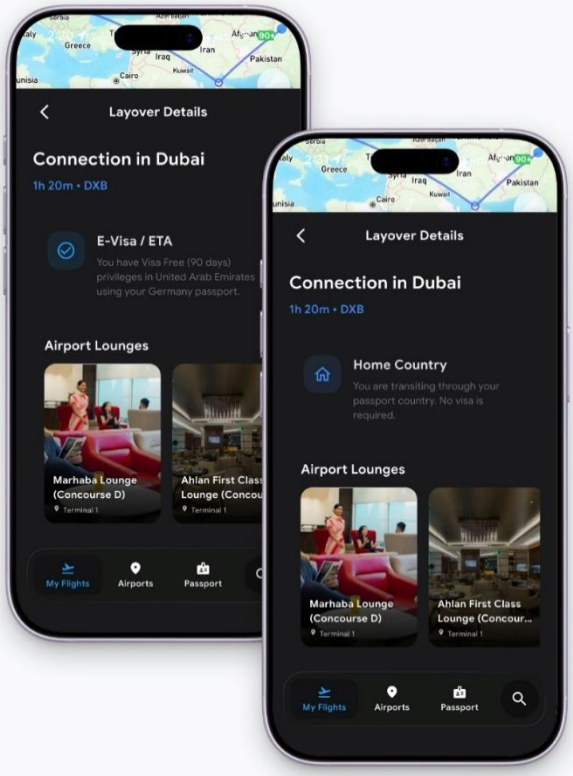
Figure 4.8.13: Top Airlines Overview

Figure 4.8.14: Top Routes Overview

Figure 4.8.15: Countries and Regions Stats

Figure 4.8.16: Past Flights History

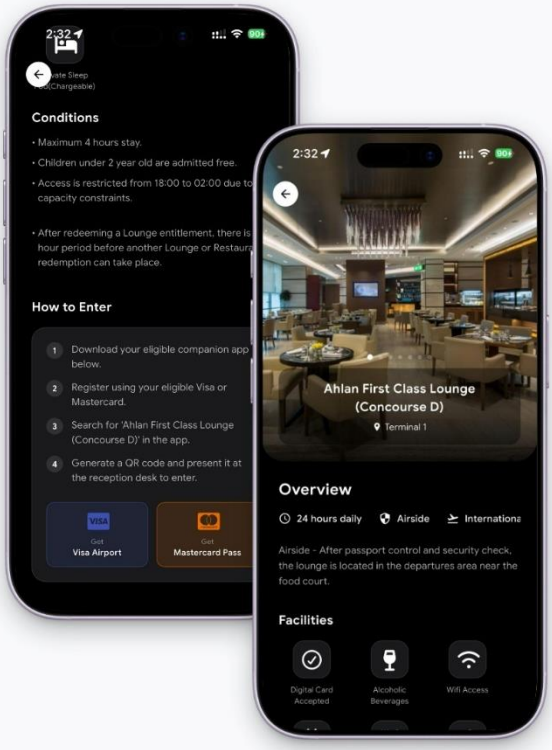
The figures above illustrate the statistics module, which shows user travel statistics such as frequently visited airports, most common airlines, routes, regional distribution, and previous flight information.



The figures present the layover details module where there are various transit and visa situations for layovers and connections.

Figure 4.8.17:
Visa-Free Transit Details

Figure 4.8.18:
Home-Country Transit Details



The figures showcase the layover module, providing the user with transit related information such as visa requirements and airport lounge facilities, with their detailed information.

Figure 4.8.19:
Layover Visa Information

Figure 4.8.20:
Airport Lounge Options

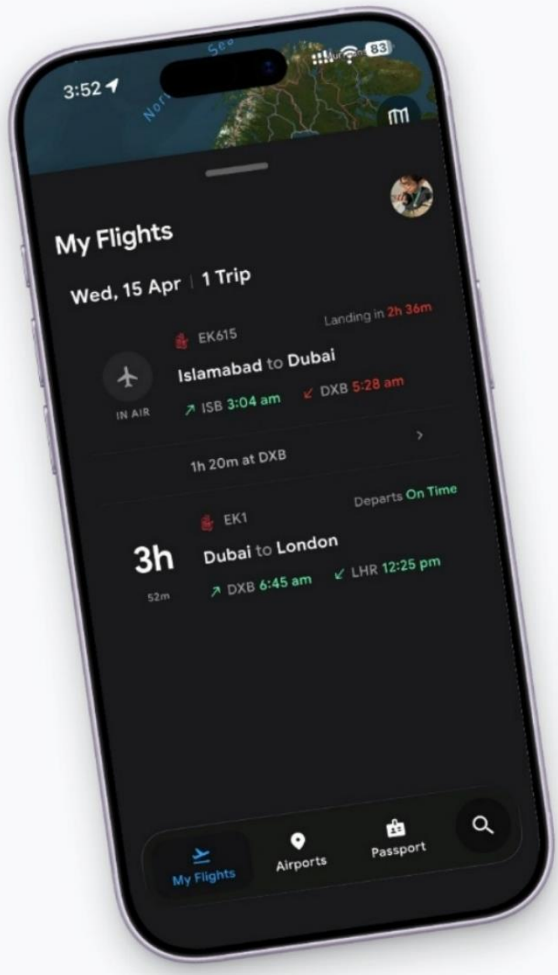


Figure 4.8.21: My Flights Overview Screen

The figure below presents the “My Flights” screen, which provides users with a structured overview of their upcoming journey, including flight segments, departure and arrival times, current status, and layover information. The interface enables users to easily track multi-leg trips in a clear and organized format.

4.9 System Prototype

A functional prototype of the system has been developed using FlutterFlow & Figma.

The prototype demonstrates:

- Navigation between screens
- UI layout and design
- Mock data integration for flight search
- Core user interactions

4.10 Conclusion

This chapter has outlined the overall design of the system, which comprises the architecture, modules, data model, and user interface. The design is scalable and easy to use.

5. System Implementation

In this chapter, we will discuss the implementation process of the Horizon – Flight Tracker & Journal app system. We will talk about the technologies used for developing this app and its key features..

5.1 Tools and Technologies

The system has been developed using modern technologies to ensure scalability, performance, and cross-platform compatibility.

5.1.1 Frontend Development [3]

- **Framework:** Flutter (via FlutterFlow)
- **Language:** Dart
- **Purpose:**
    - Building the mobile user interface
    - Managing navigation and user interaction
    - Designing responsive and visually appealing screens

5.1.2 Backend Development [4]

- **Platform:** Azure App Service
- **Technology:** Node.js (Express)
- **Purpose:**
    - Handling business logic
    - Managing API requests
    - Integrating with external services

5.1.3 Database [6]

- **Technology:** Azure SQL
- **Purpose:**
    - Storing user data
    - Managing flight records
    - Maintaining travel logs and statistics

5.1.4 Authentication

- **Technology:** Firebase Authentication
- **Purpose:**
    - Secure user login and registration
    - Token-based authentication

5.1.5 External APIs

- FlightLabs APIs (for schedules and flight details) [1]
- OpenWeather APIs (for airport weather information) [5]
- DragonPass APIs (for Airport Lounge Details)
- Wikipedia API (for top tourist places for transit layover city)

5.2 Development Approach

An **iterative and incremental approach** was used to develop the system.

Development Phases:

1. UI design using FlutterFlow
2. Implementation of core screens
3. Integration of navigation and user flows
4. Backend setup and API integration
5. Testing and refinement

5.3 Implementation of Key Features

5.3.1 Authentication System

- Users can register and login
- Authentication system handles user sessions (using Firebase Auth/OAuth)
- User data is linked to individual accounts

5.3.2 Flight Search Feature

- Users can search for flights using:
    - Flight number
    - Route (departure and arrival airports)
- Dynamic search suggestions are implemented

5.3.3 Flight Details Module

- Displays detailed flight information including:
    - Timing and duration
    - Airline and aircraft details
    - Status updates
- UI is designed for clarity and readability

5.3.4 Airport and Airline Modules

- Users can view airport and airline information
- Airport screens include:
    - Amenities
    - Weather
    - Flight boards

5.3.5 Manual Flight Entry

- Users can manually input flight details
- Useful for private or unscheduled flights

5.3.6 Travel Log and Notes

- Users can add notes to flights
- Notes are stored and editable
- Provides a personalized journaling experience

5.3.7 Statistics Dashboard

- Displays user travel insights such as:
    - Total flights
    - Distance traveled
    - Airports visited
- Data is calculated dynamically

5.4 UI Implementation

The user interface was implemented with a focus on:

- Clean and minimal design
- Dark theme for modern appearance
- Smooth navigation between screens
- Responsive layouts

UI Features:

- Scrollable screens
- Dynamic lists (flight results, airport lists)
- Bottom navigation bar
- Floating action buttons (e.g., Add Flight)

5.5 Challenges Faced

Challenges encountered during implementation were:

- Managing state and actions in Flutter
- Handling conditional visibility for dynamic UI elements
- Navigation setup across multiple screens
- Limited access to real-time APIs (mock data used)
- Debugging UI spacing and layout issues

These challenges were resolved through iterative testing and refinement.

5.6 Conclusion

In this chapter, the process involved in implementing the system was highlighted using proper tools and technology. The system is now functional due to the implementation of its features.

6. System Testing & Evaluation

The testing methodology followed in the process of evaluating the effectiveness of the system will be discussed in this chapter. The testing phase is crucial since it determines how functional and effective the system will be.

6.1 Test Strategy

Focusing on verifying individual features and overall system behavior, the testing process followed a **manual and iterative testing approach**,.

Testing was performed during development to:

- Identify and fix bugs early
- Validate UI behavior
- Ensure correct data handling

The testing strategy included:

- Functional testing
- UI testing
- Integration testing

6.2 Component Testing

To ensure correct functionality, each module of the system was tested individually:

Components Tested:

- Authentication system
- Flight module for search
- Airport and airline modules
- Travel log and notes
- Statistics dashboard

Each component was tested by simulating user interactions and verifying expected outputs.

6.3 Unit Testing

To verify small parts of the system, basic unit-level testing was performed:

- Input validation (search fields, forms)
- UI components (buttons, text fields)
- Conditional rendering (visibility logic in FlutterFlow)

This ensured that individual elements behaved correctly before integration.

6.4 Integration Testing

To verify that different modules work together correctly, integration testing was conducted.

Examples:

- Search → Results → Flight Details flow
- Airport selection → Date selection → Results
- Adding notes → Viewing in travel log

The system was tested to ensure smooth navigation and proper data flow between screens.

6.5 System Testing

Key Checks:

- Complete user flows
- UI consistency across screens
- Navigation between pages
- Responsiveness of layouts

To ensure stability and usability, we tested the system on different scenarios.

6.6 Test Cases

6.6.1 Test Case #1 — Flight Search

Table 6.6.1: Test Case 1

Field	Description
Test Case ID	TC-01
Feature	Flight Search
Input	Valid flight number (e.g., BA407)
Expected Output	Display flights matching the criteria
Result	Passed

6.6.2 Test Case #2 — Route Search

Table 6.6.2: Test Case 2

Field	Description
Test Case ID	TC-02
Feature	Route Search
Input	Departure + Arrival airport codes
Expected Output	Display Flights matching route criteria
Result	Passed

6.6.3 Test Case #3 — Manual Flight Entry

Table 6.6.3: Test Case 3

Field	Description
Test Case ID	TC-03
Feature	Manual Flight Entry
Input	Flight details entered manually
Expected Output	Flight with manual details saved successfully
Result	Passed

6.6.4 Test Case #4 — Notes Feature

Table 6.6.4: Test Case 4

Field	Description
Test Case ID	TC-04
Feature	Add Notes
Input	User enters note text against a saved flight
Expected Output	Note is saved and displayed when flight is selected from List
Result	Passed

6.6.5 Test Case #5 — User Adding Duplicate Flight

Table 6.6.4: Test Case 5

Field	Description
Test Case ID	TC-05
Feature	Flight Search
Input	Valid flight number (e.g., BA407)
Expected Output	Display of Error explaining Flight is already added
Result	Passed

6.6.6 Test Case #6 — User Searching Non-Existent Flight

Table 6.6.6: Test Case 6

Field	Description
Test Case ID	TC-06
Feature	Flight Search
Input	Random flight number (e.g., XYZ407)
Expected Output	Error Displaying explanation that no flight is available for the specified criteria as per the schedule API
Result	Passed

6.7 Results & Evaluation

The results of the tests show that the system operates according to its basic functions as designed.

Observations:

- All primary features are functioning correctly
- UI interactions are smooth and responsive
- Navigation flows are working properly
- Minor issues observed in UI responsiveness and state updates (FlutterFlow-related)

Limitations:

- Real-time API integration is not fully implemented
- Some features rely on mock data

Even with these limitations, the system is capable of proving the required functionality and design.

6.8 Conclusion

The outcomes show that the software is able to fulfill its intended functions and is highly reliable. Improvements may be implemented following the complete backend and API integration.

7. Conclusion

The goal of this project is to create and build the Horizon - Flight Tracker & Journal, which is an application that aims to make flights more convenient and fun by customizing and designing an interface for the user.

The system was able to integrate several features into one application such as flight tracker, airport information, airline information, flight journal, and statistics. It was able to solve the problems of existing systems that did not provide a holistic approach but only focused on flight tracking.

This existing prototype is built using FlutterFlow, Firebase, and Azure services. Even though there were some limitations in this system such as partially working APIs, this system was still able to show its effectiveness in the workflow and usability.

7.1 Contributions

The key contributions of this project include:

- Development of a **centralized flight tracking platform** that combines multiple features into one application.
- Design and implementation of a modern and intuitive mobile user interface**.**
- Integration of **flight search and airport modules** with structured user flows.
- Implementation of a **travel log and journaling system** for personalized user experience.
- Creation of a **statistics dashboard** to visualize travel data.
- Deployment of a scalable architecture using **Flutter, Firebase, and Azure.**

These contributions demonstrate the ability to design and develop a complete software system from concept to implementation.

7.2 Reflections

It was indeed a challenging project; we were working on something we barely knew about API and a topic which has been overlooked too much.

The major challenges we encountered in our project involved state management integration with FlutterFlow and conditional rendering and navigation. Apart from that another challenge we had in this project was lack of real-time connection with API due to lack of financing for this particular purpose. However, apart from the challenges mentioned above, this project taught us a lot about::

- Mobile application development
- UI/UX design principles
- System architecture and integration
- Problem-solving and debugging

The project also highlighted the importance of the application of agile principles.

7.3 Future Work

The current application has enabled us to understand how our proposed features would operate; but upon reflection on possible features we could add to improve ours further they would include:

- Real-time connection to flight and weather APIs *[1][5]*
- Delay analytics and Aircraft Type visualization in the statistics dashboard
- AI-based delay prediction and travel recommendations

These additions could make the application fully ready for production purposes.

References

[1] FlightLabs, "FlightLabs API Documentation: Flight Schedules & Status" Available: https://www.flightlabs.io/documentation

[2] FlightRadar24, "FlightRadar24: Live Flight Tracker — Real-Time Flight Tracker Map," FlightRadar24 AB]. Available: https://www.flightradar24.com

[3] S. R. Safavi, "Performance Evaluation of Cross-Platform Mobile Frameworks: A Comparative Study of React Native and Flutter," M.S. thesis, Dept. of Information Engineering, University of Padua, Padua, Italy, 2024/2025. Available: https://hdl.handle.net/20.500.12608/87277

[4] Microsoft, "Azure App Service Documentation," Microsoft Learn , Available: https://learn.microsoft.com/en-us/azure/app-service/

[5] OpenWeather, "OpenWeatherMap API Documentation," OpenWeather Ltd. Available: https://openweathermap.org/api

[6] Microsoft, "Azure SQL Database Documentation," Microsoft Learn Available: https://learn.microsoft.com/en-us/azure/azure-sql/

[7] Microsoft, "Azure App Service Documentation," Microsoft Learn, 2024. Available: https://learn.microsoft.com/en-us/azure/app-service/

[8] FlightAware, "FlightAware: Global Flight Tracking & Status," FlightAware LLC, 2024. [Online]. Available: https://www.flightaware.com/aeroapi/portal/documentation

[9] Google, "Flutter Documentation: Build apps for any screen," Google LLC, 2024. Available: https://docs.flutter.dev

[10] FlutterFlow, "FlutterFlow Documentation: Visual Application Builder," FlutterFlow Inc., 2024. Available: https://docs.flutterflow.io

[11] Redwerk, "Why Use Flutter in 2025? Pros and Cons of Flutter App Development," *Redwerk Tech Blog*, Sep. 18, 2025

Available: https://redwerk.com/blog/flutter-app-development-advantages-disadvantages/

Kashif Sultan

fyp-srs._Ahmad Zainab

Student's Task

Document Details

Submission ID
trn:oid:::3618:135822807

Submission Date
Apr 20, 2026, 10:41 AM GMT+5

Download Date
Apr 20, 2026, 10:45 AM GMT+5

File Name
fyp-srs._Ahmad Zainab.pdf

File Size
3.1 MB

65 Pages

7,847 Words

43,001 Characters

13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **96** Not Cited or Quoted 13%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **2** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
2% Publications
13% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.