Final Year Project Report

A report submitted in the partial fulfillment of degree of BSE

Trip Nexus

Bahria University, Islamabad
May 2026

Supervisor
Engr. Subas Bilal

Group Members

Muhammad Hafeezullah Iqbal	**01-131222-032**
Talha Bin Tahir	**01-131222-047**

Software Engineering Department

TRIP NEXUS

Group Members

Muhammad Hafeezullah Iqbal (01-131222-032)

Talha Bin Tahir (01-131222-047)

Supervisor: Engr. Subas Bilal

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: Talha Bin Tahir Enrolment No: 01-131222-047

Student Name: Muhammad Hafeezullah Iqbal Enrolment No: 01-131222-032

Programme of Study: Bachelor of Software Engineering

Project Title: Trip Nexus

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 _____ 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: _____ Name: Engr. Subas Bilal_____

CERTIFICATE OF ORIGINALITY

This is to certify that the intellectual contents of the project titled TripNexus – AI-Enabled Smart Tourism Platform with Blockchain-Verified Reviews are entirely the product of our own work, except where material has been properly cited and acknowledged in the references section. Any material drawn from external sources, including research journals, published books, technical documentation, and online resources, has been used solely to support, elaborate upon, compare, and extend our own work.

Furthermore, this project has not been submitted in whole or in part for any other academic degree or qualification at this University or any other institution. Should this declaration be found to be false at any stage, the University reserves the right to cancel the degree in accordance with its applicable rules and regulations.

Name of the Student: **Talha Bin Tahir**

Signature: _____ Date: _____

Name of the Student: **Muhammad Hafeezullah Iqbal**

Signature: _____ Date: _____

TripNexus

Sustainable Development Goals

The SDGs listed below have been identified as directly relevant to TripNexus based on its implemented features, design goals, and measurable impact pathways. Ticked entries (✓) indicate primary alignment.

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 & Complex Activities

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

The apps for tourism currently available on the market tend to address only one component of the journey effectively, while other components become quite fragmented. If a tourist would like to plan his tour, hire a capable guide, book a package and trust the ratings, he needs to use different apps, which have certain disadvantages. However, the worst issue is related to the reliability of the ratings because they are located in a single place and may be altered or forged without any traces for the tourist to trace.

The Final Year Project offers TripNexus, which is an artificial intelligence-enhanced role-based mobile tourism application built using the Flutter programming environment with MongoDB Atlas as the persistence database technology. It has an integration of the Mistral large language model API to provide AI-based itinerary suggestions and landmark recognition features, the Open-Meteo service for acquiring weather information of multiple cities, the Google News RSS feed parser to extract news and event information in each area, and the development of a unique Solidity smart contract in the Polygon Amoy testnet to ensure immutable blockchain-based review storage.

The application covers the full cycle of travel services including pre-trip planning, automatic generation of travel itineraries, recognition of landmarks, selection of packages, booking, advisor chat communication, and lastly blockchain submission of travel review after completion of the travel experience. Travel agencies and individual travel advisors control the cataloging of packages, booking processes involving the states of pending, approved, and completed, as well as analyzing reviews of the particular travel packages on the blockchain. This application uses a trust model that writes only completed booking reviews onto the blockchain and keeps all other booking data off the chain in MongoDB.

A total of 22 functional requirements and 8 non-functional requirements have been mapped with executed test cases. Tests specifically focused on edge cases have been performed for the blockchain review process flow with respect to the following scenarios: no gas transaction, incorrect contract address, and duplicate submission. Acceptance criteria have been achieved. Hence, using the developed prototype, it has been demonstrated that it is possible to develop a mobile application that includes AI and marketplace functionalities as well as blockchain-based reviews, all accomplished by a small student team.

Keywords: Smart Tourism, Flutter, AI Itinerary Planning, Landmark Recognition, MongoDB, Mistral AI, Blockchain Reviews, Polygon Amoy, Web3Dart, Software Engineering, Hybrid Trust Architecture.

Dedication

In the name of Allah, the Most Gracious, the Most Merciful.

This work is wholeheartedly dedicated to Almighty Allah, whose limitless mercy, wisdom, and blessings have made all the efforts fruitful. Glory be to him.

To our beloved parents, who through their unselfish love, silent supplications, and sacrifices have created the base upon which everything is erected. Every hardworking day and sleepless night has been fueled by the affection of your care and the faith that you have always placed in us.

To our teachers and mentors, who shaped not only our technical understanding but also our character as engineers and as people. Your guidance continues to influence how we think, question, and build.

Acknowledgments

Our thanks and gratitude are for the Almighty God who bestowed upon us the wisdom, perseverance, and patience to be able to complete our project successfully from scratch until its completion.

Heartfelt gratitude and appreciation is extended to our supervisor, Ms. Engr. Subas Bilal, who has made us face many challenges that made us work harder and better to give an effective result at the end. She made her valuable comments, helped us to get out of confusion, and gave her valuable time not only in understanding every aspect of the project but also made us understand how to convert our broad idea into a presentable product.

A mention must be made to express thanks to the Department of Software Engineering at Bahria University, Islamabad, for providing a supportive learning environment that helps one to develop a thinking ability and rewards for putting in extra efforts. The theoretical knowledge acquired during the studies on requirement analysis, system design, database management, software testing, and project management became very useful in dealing with challenges during the development process of TripNexus.

It is important for us to extend our sincere thanks to our parents who continued praying for us throughout the most difficult period of designing and debugging the TripNexus application and kept motivating us despite the fact that we showed little success.

Finally, it is important to express our heartfelt thanks to our friends who offered us constructive criticism, participated in our user tests, and provided us with practical cases.

Table of Contents

FYP COMPLETION CERTIFICATE ..... **i**

TripNexus ..... **iii**

Sustainable Development Goals ..... **iii**

Range of Complex Problem Solving & Complex Activities ..... **iv**

Abstract ..... **v**

Dedication ..... **vi**

Acknowledgments ..... **vii**

List of Abbreviations and Acronyms ..... **xiii**

Introduction **1**

1.1. Background and Motivation 1

1.2. Problem Statement 2

1.3. Objectives 2

1.4. Main Contributions 3

1.6. Methodology Overview 3

1.7. Success Criteria 4

1.8. Report Organisation 4

Background Study / Literature Review **5**

2.1. Smart Tourism Platforms 5

2.2. AI-Assisted Travel Planning 5

2.3. Blockchain for Review Integrity 6

2.4. Flutter in Mobile Development 7

2.5. Comparative Analysis of Existing Platforms 7

2.6. Research Gap 7

2.7. Chapter Summary 8

System Requirements **9**

3.1. Use Case Model 9

3.2. Functional Requirements 19

3.3. Non-Functional Requirements 20

3.4. Interface Requirements 21

3.4.1. User Interface Requirements 21

3.4.2. Software and Component Interface Requirements 21
3.4.3. Hardware and Device Interface Requirements 22
3.5. Database Requirements 22
3.6. Project Feasibility 23
3.6.1. Technical Feasibility 23
3.6.2. Operational Feasibility 23
3.6.3. Legal and Ethical Feasibility 23
3.7. Analysis Models 23
3.7.1. Activity Diagrams 23
3.7.2. Sequence Diagrams 26
3.8. Risk Analysis 27
3.9. Conclusion 27
System Design 28
4.1. Design Approach 28
4.2. Design Constraints 28
4.3. System Architecture 29
4.4. Logical Design and Class Model 30
4.5. Dynamic View – Sequence Diagrams 31
4.6. Component Design 35
4.7. Data Models 36
4.8. User Interface Design 36
4.8.1. Design Principles 36
4.8.2. System Prototype 36
4.9. Security by Design 45
4.10. Conclusion 45
System Implementation 46
5.1. Development Stack and Environment 46
5.2. Authentication and Session Management 47
5.3. Tourist Module Implementation 47
5.3.1. Trip Management 47
5.3.2. AI Itinerary Planning 47
5.3.3. Landmark Scanner 47
5.3.4. Weather and News Modules 48

5.4. Advisor Module Implementation 48
5.5. Blockchain Review Implementation 48
5.6. Implementation Challenges and Resolutions 50
5.7. Code Organisation 50
5.8. Conclusion 50
System Testing & Evaluation 51
6.1. Test Strategy 51
6.2. Component Testing 51
6.3. Unit Testing 51
6.4. Integrated Testing 51
6.5. System Testing 52
6.6. Test Cases 52
6.6.1. Functional Test Cases 52
6.6.2. Blockchain Edge-Case Tests 54
6.6.3. Non-Functional Test Cases 54
6.7. Results & Evaluation 55
6.8. Threats to Validity 56
6.9. Conclusion 56
Conclusion 57
7.1. Contributions 57
7.2. Reflections 57
7.3. Future Work 58
7.4. Final Remarks 58
References 59
Appendices 60
Appendix A: Requirement Traceability Matrix 60
Appendix B: Deployment and Demo Checklist 61

List of Figures

Figure 3.1: Use Case Diagram (TripNexus)
Figure 3.2: Activity Diagram — Tourist Booking Workflow
Figure 3.3: Activity Diagram — Advisor Package Lifecycle Management
Figure 3.4: Sequence Diagram — Edit and Save Trip by Traveler
Figure 4.1: High-Level System Architecture — Four-Tier View
Figure 4.2: Class Diagram (Domain Model)
Figure 4.3: Sequence Diagram — Tourist Registration
Figure 4.4: Sequence Diagram — Tourist Login
Figure 4.5: Sequence Diagram — AI Trip Planning Flow
Figure 4.6: Sequence Diagram — Local Advisor Create Package
Figure 4.7 - Sign Up Page
Figure 4.8 - Login Page
Figure 4.9 - Tourist Dashboard
Figure 4.10 - User Profile Page
Figure 4.11 - Trip Planner Page
Figure 4.12 - Generated Plan UI
Figure 4.13 - Landmark Scanner
Figure 4.14 - Weather Page
Figure 4.15 - News & Event Page
Figure 4.16 - Advisor Dashboard
Figure 4.17 - Advisor Package Creating Page
Figure 4.18 - Traveler Review Submission Page
Figure 4.19 - Traveler Review Confirmation Page
Figure 5.1: Blockchain Review Submission Flow — from Dialog to PolygonScan Verification

List of Tables

Table 2.1: Comparative Analysis of Existing Tourism Platforms
Table 3.1: Use Case Description – User Registration
Table 3.2: Use Case Description – User Login and Session Restoration
Table 3.3: Use Case Description – Create Personalised Trip
Table 3.4: Use Case Description – View and Manage Saved Trips
Table 3.5: Use Case Description – Landmark Recognition via Image
Table 3.6: Use Case Description – Browse and Book Advisor Package
Table 3.7: Use Case Description – Advisor Package Lifecycle Management
Table 3.8: Use Case Description – Submit Blockchain Review
Table 3.9: Use Case Description – View Weather Forecasts and Local News
Table 3.10: Functional Requirements Catalog
Table 3.11: Non-Functional Requirements Catalog
Table 3.12: Core Database Collections and Responsibilities
Table 3.13: Risk Analysis and Mitigation Strategies
Table 5.1: Development Technology Stack
Table 6.1: Functional Test Suite Summary
Table 6.2: Blockchain Edge-Case Test Results
Table 6.3: Non-Functional and Reliability Test Cases
Table 6.4: Acceptance Test Checklist
Table A.1: Requirement Traceability Matrix

List of Abbreviations and Acronyms

Acronym	Full Form
AI	Artificial Intelligence
API	Application Programming Interface
BSE	Bachelor of Software Engineering
DB	Database
DID	Decentralised Identifier
FR	Functional Requirement
FYP	Final Year Project
HEC	Higher Education Commission (Pakistan)
LLM	Large Language Model
MATIC	Native token of Polygon Blockchain
MVC	Model-View-Controller
NFR	Non-Functional Requirement
NoSQL	Non-Relational Database
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SHA	Secure Hash Algorithm
SSI	Self-Sovereign Identity
UI	User Interface
UX	User Experience
WBS	Work Breakdown Structure
XML	Extensible Markup Language
2FA	Two-Factor Authentication

Chapter 1
Introduction

Over the last ten years, the tourism sector worldwide has seen tremendous digital disruption. Today's tourists require apps to help them navigate their trips and make informed decisions on destinations as well as whom they should interact with. Unfortunately, despite all the changes, there still exists a significant disparity between what people need when travelling and what they have today in terms of apps. Existing tools tend to excel in only one domain - finding a place to stay, suggesting itineraries, providing information on local events and activities - but cannot cover the whole experience.

Another equally worrying factor is the issue of the rating system's validity. Ratings play an important role in determining destinations one should visit, but having a centralized rating system does not offer any natural protection from manipulation. Manipulation can easily be done by the owner of the business or even the website owner himself through editing, deleting, or even creating new reviews without a single trace left behind.

These are the twin challenges that have been directly addressed through the development of TripNexus. The platform represents a role-based model that connects tourists with travel consultants, all within one seamless process, and a blockchain-enabled review system that solves the trust challenge.

1.1. Background and Motivation

The motivation for this project emerged from four recurring deficiencies observed across commonly available travel applications:

1. Fragmented planning experience: Travellers routinely switch between separate applications for itinerary generation, weather checking, local event discovery, and service booking. This context-switching degrades decision quality and increases the cognitive cost of travel planning.
2. Low review authenticity: Centralised ratings platforms provide no mechanism for users to verify that a review has not been altered since submission. The structural inability to detect manipulation reduces the credibility of the entire review ecosystem.
3. Weak role integration: Tourist and advisor journeys are rarely connected within a single operational workflow. Most platforms serve only one side, leaving advisors dependent on external channels to manage bookings and communicate with travellers.
4. Insufficient travel context: Many applications present static destination information without real-time inputs such as current weather risk, active local

events, or safety-relevant news updates that would inform smarter itinerary decisions.

TripNexus addresses all four deficiencies through a unified mobile architecture that consolidates AI planning intelligence, advisor operations, and verifiable review trust into one product.

1.2. Problem Statement

The existing tourism applications have merely tackled half of the problem in addressing the multifaceted nature of the issue. When considering tourists, there are fragmented services when it comes to itinerary planning, reservation management, and review systems that lack any means of validating the veracity of the inputs entered. The challenge that our group aims to address is the following:

"How can a single mobile platform deliver AI-assisted travel planning, a structured advisor-driven package marketplace, and cryptographically verifiable review trust without sacrificing usability or implementation feasibility for a real-world deployment path?"

1.3. Objectives

The following objectives were defined to address the problem statement systematically:

1. Develop a role-based mobile application supporting tourists, independent advisors, and travel agencies within a single unified platform.
2. Implement secure user authentication with SHA-256 hashed password storage and persistent session management that survives application restarts.
3. Enable tourists to create and manage trip plans, generate AI-powered itineraries using Mistral API, and save plans for later retrieval.
4. Integrate AI-powered landmark recognition so that tourists can identify destinations and cultural sites from captured or selected images.
5. Build a package marketplace where tourists can discover, filter, and book advisor-listed travel packages with integrated per-package chat support.
6. Provide advisors with complete package lifecycle tools covering creation, editing, status management, booking decisions, and traveller communication.
7. Implement a blockchain review subsystem that writes completed booking reviews immutably to a Solidity smart contract on Polygon Amoy testnet.
8. Include real-time weather forecasting and local event or news feeds with user-configurable local notification support.
9. Evaluate the system through requirement-mapped functional, integration, blockchain edge-case, and acceptance test execution.

1.4. Main Contributions

- **Complete dual-role tourism workflow:** Implementation of the entire dual-role tourism workflow in Flutter, where tourist and advisor operations are performed using a consistent source code base.
- **Hybrid blockchain trust model:** An efficient design implementing a blockchain-based review using a Solidity smart contract alongside booking status tracking on an off-chain MongoDB database.
- **Integrated AI intelligence:** Usage of the Mistral AI model in generating natural language itineraries and landmark detection based on images taken by the tourist in their daily activity.
- **Requirement-driven development:** Traceable development process from requirements gathering to executing test cases, allowing for efficient evaluation.
- **Reference prototype and roadmap:** A prototype that can be deployed in a fully academically sound fashion, including a roadmap for production-ready usage.

1.5. Scope

- Tourist management using role-based systems of tourists, personal advisors, and advising organization.
- Tour creation, AI itinerary planner, scanning landmarks, weather alerts, and news alerts.
- Package listing, tour booking process, advising-tourist interaction, and tour booking status process.
- Blockchain-based review submissions and retrievals via smart contract on Polygon Amoy testnet.
- Local weather and news alerts
- Test execution throughout system and results documentation.

1.6. Methodology Overview

The development process adopted an iterative methodology consisting of five phases:

10. Requirement analysis through actors' identification and use-case development.
11. Designing the system architecture and domain entities, integrating all other external systems.
12. The development phase was conducted using four vertical slices, providing deliverables in each iteration.
13. Validation was carried out through various test levels including functional, integration, and blockchain-specific testing, which corresponded to requirement specifications.

14. Impact analysis consisted of defect identification and edge case failures during demonstration trials.

1.7. Success Criteria

All of the points below had to be verified for successful completion of the project:

- The tourists got the chance to undergo the entire process from planning till booking without facing any conflict of roles.
- The advisers could provide packages, manage bookings, and get on-chain review statistics.
- The bookings done successfully could choose to give their review on the blockchain using a transaction hash.
- The second attempt to submit the review could not be allowed by the system at the user interface level.
- Core functions were covered with tests.

1.8. Report Organisation

Structure of this report is described here. The purpose of Chapter 2 is providing background information on the subject matter, doing literature research for smart tourism, AI-enabled trip planning and blockchain reviews and analyzing existing solution to identify gaps. Chapter 3 will provide full-fledged requirements specification covering functional and non-functional requirements, interface specification, database structure, feasibility analysis, and risk management. Chapter 4 is devoted to discussion about design of the system including system architecture, components design, logical and dynamic views of the system, database modeling and design of the user interface. Implementation chapter (Chapter 5) will describe the implementation process and discuss important decisions taken during implementation phase as well as issues identified during the development process. Testing of the system will be covered in Chapter 6.

Chapter 2
Background Study / Literature Review

In this chapter, we place our research TripNexus into context by reviewing what is currently known about smart tourism, AI-based travel apps, blockchain technology used in providing online credibility, and mobile app development. We highlight the strengths and weaknesses of current knowledge, while considering how the design considerations of our project address them. This insight was necessary because our goal was to develop an innovative solution, not merely a duplicate of existing technology.

2.1. Smart Tourism Platforms

The concept of smart tourism entails the use of technological advancements that improve the travel experience, aid destination management, and encourage responsible tourist conduct. In other words, based on Buhalis & Amaranggana [1], a smart tourism destination is considered a system that combines the ecosystem of tourism stakeholders with smart city facilities to enable the delivery of valuable data to tourists rather than providing raw information. The features associated with a successful tourism platform include personalization, responsiveness, interoperability, and crowdsourcing.

According to UN World Tourism Organization, digital preparedness is critical in determining competitiveness in the market [2]. Huge number of travelers today rely heavily on the use of mobile-first applications, and they find it difficult to switch among different apps during a single travel period. The above consideration was the main reason why TripNexus incorporated all the functionality related to itinerary management, browsing packages, communicating with an adviser, and submitting reviews into a single application.

2.2. AI-Assisted Travel Planning

There has been rapid adoption of the application of large language models in the travel industry since the year 2022. There have been studies done on the subject that have shown that merely incorporating an AI system in the app is not sufficient; the success of such an app relies heavily on the degree of contextual information that the system has been provided about the person it is talking to [3]. Studies analyzing different types of AI systems for travel assistants have shown that providing detailed information about the destination, length of stay, preferences, and budget of the traveler has produced outputs which are far more satisfying than generic inputs.

From various studies conducted regarding travel recommendation systems based on artificial intelligence, it is evident that there are mainly three methods employed in developing such technologies, which include collaborative filtering where recommendations rely on other users' decisions, content-based filtering where recommendations are aligned with some characteristics of the suggested products, and generative prompting where the machine learning model recommends the options [4].

Among the three methods mentioned above, the latter deserves more attention as it has the potential to provide several days of recommendations and ensure that suggestions flow seamlessly between each day, making them much easier to read than listing recommendations.

Interactive aspect of the process of discovery which allows users to make and share pictures rather than perform search queries through the usage of keywords has been proven by multiple user experience studies as a significant milestone toward making the process of discovering information about visited landmarks much easier and quicker [5]. In case travelers could just make pictures from their smartphones and at the same time learn everything about history and culture of particular landmarks, it would mean that the process of discovering information became not only intuitive but very entertaining as well. This is the main reason why we decided to integrate landmark scanner into our mobile application.

2.3. Blockchain for Review Integrity

The problem of manipulations by way of review of centralized platforms has been thoroughly investigated by researchers. The findings about online reviews indicate that the managers of these online platforms, who act as central authorities, regulate everything about the reviews, whether it involves showing some reviews or deleting others depending on their preferences [6]. The very worst case possible is that the manager can delete any negative reviews, alter existing reviews, and even accept fake reviews, which might favor certain sellers. But if blockchain technology is adopted in order to store reviews, the central authority loses its power, as all the reviews will be stored forever and will not be editable or deletable anymore.

The first two challenges that may occur when working with a blockchain network that complies with Ethereum rules involve properly managing the private keys required for signatures and having sufficient money in the account for covering the costs associated with gas usage prior to making an attempt at sending a transaction [7]. Both cases have been observed to generate deceptive errors during the testing of prototypes. As for TripNexus, checking the wallet balance before sending a review is done in order to prevent the occurrence of an error. If that happens, users are given instructions on how to obtain the needed funds from a faucet.

Further theories justifying the implementation of the architecture for a hybrid trust framework by TripNexus come from research on the trade-off present in the characteristics of distributed ledger technology [8]. Research has shown that there is no such thing as an optimal design for a blockchain for any use case. Where the importance of trust and efficiency are equal, as in the case of booking sites and review sites, it becomes vital to partition immutable data and efficient data.

2.4. Flutter in Mobile Development

The choice of using Flutter is justified by the high level of popularity of this cross-platform application framework for mobile phones. This is because it allows for building complex UIs using the same code. Based on the information provided in the official documentation [9], Flutter offers a variety of widgets and expressiveness in its UI, as well as quick development thanks to a hot reload function which was important while developing by a team of two people within the academic semester limitations. In addition, the need for mobile application development is justified as in developing countries mobile phone access to digital tourism applications prevails [10].

2.5. Comparative Analysis of Existing Platforms

Table 2.1 presents a comparative analysis of representative tourism platform categories, evaluating each against the specific requirements that motivated TripNexus.

Table 2.1: Comparative Analysis of Existing Tourism Platforms

Platform Category	Key Strengths	Limitation Relevant to This Project
Booking-focused apps (e.g., Booking.com, Expedia)	Strong global reservation ecosystem with extensive inventory	Limited AI trip generation; no blockchain review trust; advisor interaction not supported
Map and navigation apps (e.g., Google Maps)	Excellent location intelligence and proximity search	No package booking lifecycle; no advisor-traveller communication channel
Itinerary planning tools (e.g., TripIt, Wanderlog)	Structured planning boards and trip organisation features	Weak trust layer; no immutable review model; no advisor marketplace
Review aggregation platforms (e.g., TripAdvisor)	Broad community-sourced content at scale	Review authenticity cannot be independently verified by end users
Travel social platforms (e.g., Polarsteps)	High community engagement and content discovery	Lacks structured advisor operations, booking states, and review integrity mechanisms

2.6. Research Gap

After analyzing the results, it is evident that there exists an enduring need within the realm of integration, and this need cannot be met by any technology platforms that exist today. All current types of technological solutions concentrate on optimizing one aspect

alone, but the others remain underutilized or underserved. TripNexus was designed to address this need.

2.7. Chapter Summary

In this chapter, the theoretical and empirical underpinnings for TripNexus were developed through an overview of principles in the smart tourism design field, studies on AI-enabled trip planning, blockchain-based trust frameworks, and the competitive analysis of existing products. These insights indicate that the problem addressed by TripNexus is real and not yet solved. The proposed combination of on-chain and off-chain technologies is based on well-established research results from the distributed systems literature. At the same time, the approach to integrating AI technology with the application aligns with the state of the art in LLM travel applications. The following chapter translates these findings into a complete requirements specification.

Chapter 3
System Requirements

The following section presents the entire set of functional and non-functional requirements, interface requirements, database requirements, feasibility studies, analysis models and risks related to TripNexus. These documents collectively form the testable specification which forms the basis of the design and implementation of the system in chapters 4 & 5.

3.1. Use Case Model

In the use case model for the system, there are two main actors – Tourist and Local Guide. Tourist actor plays the role of a consumer while Advisor plays the role of a producer of packages/booking process. The secondary actors in the system supply intelligence, environment details and persistent storage respectively.

Some of the major use cases in the system include registration and logging into the system, AI generated itinerary, browsing and booking packages, managing packages of the advisor, advisor's booking decision, acquiring weather & news feed, sending notifications and management of privacy settings.

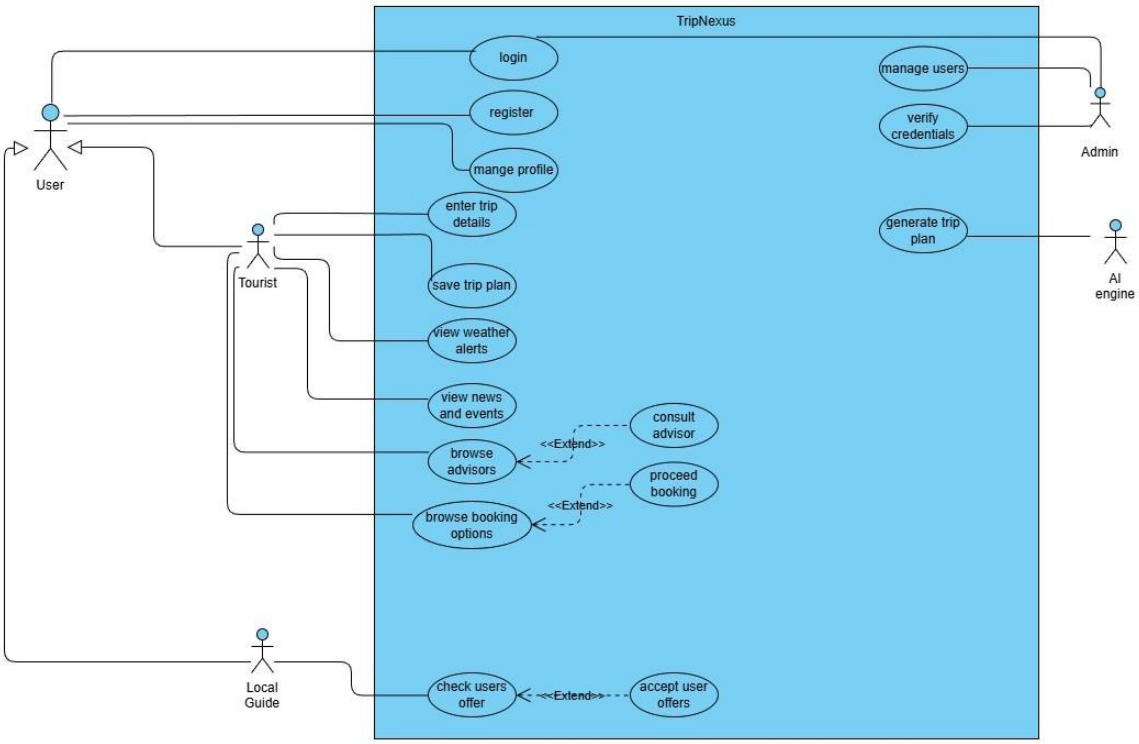

Figure 3.1: Use Case Diagram (System Level - TripNexus)

The following sections describe each use case in detail through the interaction between the major actors in the system and the software product called TripNexus. Each use case includes use case id, actors involved, pre-conditions, priority and flows..

Table 3.1 outlines the steps involved in the registration process when a new Tourist or Advisor creates an account on TripNexus.

Table 3.1: Use Case Description – User Registration

Use Case ID	**UC-001**
Use Case Name	User Registration
Actor(s)	Tourist, Local Advisor
Pre-Conditions	User must have a valid email address.
Priority	High
Basic Flow	A new user registers by providing their details, selecting a role, and activating their TripNexus account.
Actor Actions	**System Response**
1. User selects "Sign Up" on the mobile app.	2. System displays the registration form.
3. User enters name, email, password, and selects role (Tourist / Advisor).	4. System validates email format, password strength, and checks email uniqueness.
5. User submits the registration form.	6. System creates an account, hashes the password using SHA-256, and confirms successful registration.
7. User is redirected to their role-specific dashboard.	8. System initialises the session and stores the user record.
Alternative Course of Action	
Actor Action	**System Response**
4.a User enters an invalid email format.	4.a System displays: "Please enter a valid email address."
4.b User enters a weak password.	4.b System displays password guidelines and prompts for a stronger password.

Use Case ID	UC-001
4.c User registers with an already existing email.	4.c System displays: "An account with this email already exists. Please log in."

Table 3.2 outlines the login process and session recovery, which enables users to skip authentication for future app usage.

Table 3.2: Use Case Description - User Login and Session Restoration

Use Case ID	UC-002
Use Case Name	User Login and Session Restoration
Actor(s)	Tourist, Local Advisor
Pre-Conditions	User must be registered.
Priority	High
Basic Flow	A registered user logs in with valid credentials and is routed to their role-specific dashboard. Session is restored automatically on subsequent app launches.
Actor Actions	**System Response**
1. User enters email and password and taps "Login."	2. System hashes the entered password and compares it with the stored hash.
3. Credentials match; user is authenticated.	4. System stores the session in SharedPreferences and navigates to the role-specific dashboard.
5. User closes and re-opens the app.	6. System reads the stored session and restores the dashboard without requiring re-login.
Alternative Course of Action	
Actor Action	**System Response**
2.a User enters incorrect email or password.	2.a System displays: "Invalid email or password. Please try again."
2.b Session data is missing or expired.	2.b System redirects the user to the login screen.

Table 3.3 shows how a Tourist generates a custom itinerary for their multi-day trip via the AI feature with Mistral API integration.

Table 3.3: Use Case Description – Create Personalised Trip

Use Case ID	**UC-003**
Use Case Name	Create Personalised Trip
Actor(s)	Tourist, Mistral AI Service
Pre-Conditions	User must be logged in as a Tourist.
Priority	High
Basic Flow	Tourist creates a personalised multi-day itinerary using a trip creation form and AI-generated suggestions from the Mistral API.
Actor Actions	**System Response**
1. Tourist taps "Plan a Trip" on the dashboard.	2. System displays the trip creation form (destination, dates, budget, interests, eco-friendly flag).
3. Tourist fills in the trip details and submits the form.	4. System validates inputs and submits a structured prompt to the Mistral API.
5. Tourist reviews the AI-generated multi-day itinerary.	6. Mistral API returns a day-by-day itinerary; system displays it in a readable format.
7. Tourist saves the itinerary.	8. System stores the plan in the saved_trip_plans collection and confirms success.
Alternative Course of Action	
Actor Action	**System Response**
3.a Tourist leaves required fields empty.	3.a System highlights missing fields and prompts the tourist to complete them.
4.a Mistral API is unavailable or returns an error.	4.a System displays a friendly error message and allows the tourist to retry.

Table 3.4 deals with the management of previously saved AI itineraries for viewing and deletion purposes.

Table 3.4: Use Case Description – View and Manage Saved Trips

Use Case ID	UC-004
Use Case Name	View and Manage Saved Trips
Actor(s)	Tourist
Pre-Conditions	Tourist must have at least one saved trip or itinerary.
Priority	Medium
Basic Flow	Tourist browses previously saved trip plans, views details, and deletes records they no longer need.
Actor Actions	**System Response**
1. Tourist navigates to the "Saved Plans" screen.	2. System retrieves and displays all saved itineraries for the tourist from the saved_trip_plans collection.
3. Tourist taps a saved plan to view its full content.	4. System displays the complete day-by-day itinerary for the selected plan.
5. Tourist selects a plan and taps "Delete."	6. System removes the record from the database and refreshes the saved plans list.
Alternative Course of Action	
Actor Action	**System Response**
2.a No saved plans exist.	2.a System displays: "You have no saved plans yet. Start by planning a new trip."

Table 3.5 describes how the AI-powered landmark scanner identifies cultural and historical sites from a captured or selected image.

Table 3.5: Use Case Description – Landmark Recognition via Image

Use Case ID	UC-005
Use Case Name	Landmark Recognition via Image

Use Case ID	UC-005
Actor(s)	Tourist, Mistral AI Service
Pre-Conditions	Camera permission must be enabled or an image must be available in the gallery.
Priority	Medium
Basic Flow	Tourist captures or uploads a photo of a landmark and receives AI-generated identification and cultural information about it.
Actor Actions	**System Response**
1. Tourist opens the Landmark Scanner and selects camera or gallery.	2. System opens the camera or gallery picker.
3. Tourist captures or selects an image.	4. System encodes the image to base64 and submits it to the Mistral vision endpoint with an identification prompt.
5. Tourist views the landmark name, historical background, and travel tips.	6. Mistral API returns the analysis; system displays it on screen.
7. Tourist taps "Save Landmark."	8. System stores the image path and analysis text in the saved_landmarks collection.
Alternative Course of Action	
Actor Action	**System Response**
4.a Mistral API cannot identify the landmark with sufficient confidence.	4.a System displays a partial result with a note that the identification may be uncertain.
4.b API call fails due to network error.	4.b System displays an error message and offers a retry option.

Table 3.6 discusses the entire package discovery and booking flow, involving browsing of the marketplace and booking requests submission to the advisor.

Table 3.6: Use Case Description – Browse and Book Advisor Package

Use Case ID	UC-006
Use Case Name	Browse and Book Advisor Package
Actor(s)	Tourist, Local Advisor
Pre-Conditions	Tourist must be logged in. At least one active package must exist in the system.
Priority	High
Basic Flow	Tourist browses available advisor-published travel packages, views details and on-chain reviews, and creates a booking request.
Actor Actions	**System Response**
1. Tourist navigates to the Marketplace screen.	2. System retrieves and displays all active packages with search, filter, and sort options.
3. Tourist applies filters and taps on a package to view its full details.	4. System displays package itinerary, inclusions, price, and associated blockchain reviews.
5. Tourist taps "Book This Package."	6. System creates a booking record in the package_bookings collection with status set to Pending.
7. Tourist receives a confirmation that the booking request has been submitted.	8. System notifies the advisor of the new booking request.
Alternative Course of Action	
Actor Action	**System Response**
3.a No packages match the applied filters.	3.a System displays: "No packages found. Try adjusting your filters."
6.a Tourist has already booked the same package.	6.a System prevents duplicate booking and informs the tourist.

Table 3.7 discusses the lifecycle of the package from the side of the advisor. It includes package creation, publishing, booking, and booking management.

Table 3.7: Use Case Description – Advisor Package Lifecycle Management

Use Case ID	UC-007
Use Case Name	Advisor Package Lifecycle Management
Actor(s)	Local Advisor
Pre-Conditions	Advisor must be logged in.
Priority	High
Basic Flow	Advisor creates, edits, publishes, and manages travel packages through a full lifecycle from creation to completion.
Actor Actions	**System Response**
1. Advisor opens the dashboard and taps "Create Package."	2. System displays the package creation form (title, destination, duration, price, inclusions, itinerary).
3. Advisor fills in all required fields and submits the form.	4. System validates the input and saves the package with a Draft status.
5. Advisor publishes the package.	6. System updates the status to Live and makes the package visible to tourists.
7. Advisor views incoming bookings and updates each booking status (Accepted, Rejected, or Completed).	8. System updates the booking record and notifies the relevant tourist of the status change.
Alternative Course of Action	
Actor Action	**System Response**
3.a Advisor leaves required fields empty.	3.a System highlights missing fields and prevents submission.
4.a Network error occurs during form submission.	4.a System displays an error message and retains the entered data for resubmission.

Table 3.8 provides information about the blockchain-based reviewing process where a tourist leaves an immutable and tamper-proof review on the Polygon Amoy smart contract after making a booking.

Table 3.8: Use Case Description – Submit Blockchain Review

Use Case ID	**UC-008**
Use Case Name	Submit Blockchain Review
Actor(s)	Tourist, Polygon Blockchain Network
Pre-Conditions	Tourist must have a booking with Completed status and a configured wallet address. The booking must not already have a review submitted.
Priority	High
Basic Flow	Tourist submits a post-trip review for a completed booking, which is written permanently to the Solidity smart contract on Polygon Amoy testnet.
Actor Actions	**System Response**
1. Tourist navigates to bookings and locates a Completed booking.	2. System displays the "Submit Review" button only on bookings with Completed status.
3. Tourist taps "Submit Review," enters a rating and comment, and confirms.	4. System performs a wallet balance pre-check to ensure sufficient test MATIC is available.
5. Tourist confirms the submission.	6. System calls the addReview function on the smart contract, signs the transaction, and broadcasts it to the Polygon Amoy network.
7. Tourist views the transaction hash and PolygonScan explorer link.	8. System updates the reviewSubmitted flag to true in MongoDB, preventing duplicate submissions.
Alternative Course of Action	
Actor Action	**System Response**
4.a Wallet balance is insufficient to cover the gas fee.	4.a System displays: "Insufficient MATIC balance. Please top up your wallet using the Polygon Amoy faucet."

Use Case ID	UC-008
6.a Blockchain transaction fails or times out.	6.a System displays an error message. The reviewSubmitted flag remains false, allowing a retry.
3.a Tourist attempts to submit a review on an already-reviewed booking.	3.a System shows a "Review Already Submitted" indicator and disables the review dialog.

Table 3.9 involve getting weather forecasts for saved cities in real time, browsing local news and configuring notifications.

Table 3.9: Use Case Description – View Weather Forecasts and Local News

Use Case ID	UC-009
Use Case Name	View Weather Forecasts and Local News
Actor(s)	Tourist
Pre-Conditions	Tourist must be logged in. At least one city must be saved in the weather module.
Priority	Medium
Basic Flow	Tourist views real-time weather forecasts for saved cities and browses locally relevant news and event articles.
Actor Actions	**System Response**
1. Tourist navigates to the Weather & News section.	2. System fetches current weather data from the Open-Meteo API for all saved cities.
3. Tourist switches between city tabs to view forecasts.	4. System displays a seven-day weather forecast with temperature, conditions, and relevant alerts.
5. Tourist navigates to the News tab.	6. System fetches and parses the Google News RSS feed, classifies articles by type, and displays them in a list.
7. Tourist toggles notification preferences for weather and news alerts.	8. System saves the notification settings persistently and schedules local notifications accordingly.
Alternative Course of Action	

Use Case ID	UC-009
Actor Action	**System Response**
2.a Open-Meteo API is unavailable.	2.a System displays the most recently cached weather data with a note indicating it may not be current.
6.a RSS feed returns no results for the selected location.	6.a System displays: "No news available for this location at the moment."

3.2. Functional Requirements

Table 3.10 contains the full list of functional requirements. Every requirement has a unique identifier FR and its implementation priority.

Table 3.10: Functional Requirements Catalog

ID	Requirement Statement	Priority
FR-01	The system shall support role-based registration for Tourist, Advisor Individual, and Advisor Agency roles.	High
FR-02	The system shall hash all user passwords using SHA-256 before database storage.	High
FR-03	The system shall support user login and persistent session restoration across application restarts.	High
FR-04	The system shall route authenticated users to their role-specific dashboard upon login or session restore.	High
FR-05	A tourist shall be able to create, view, and delete personal trip records with destination and date fields.	High
FR-06	A tourist shall be able to define an optional budget and eco-friendly travel flag per trip.	Medium
FR-07	A tourist shall be able to generate AI-powered multi-day itinerary plans using Mistral API.	High
FR-08	A tourist shall be able to save AI-generated itinerary plans and retrieve them in a saved plans view.	High
FR-09	A tourist shall be able to scan landmark images from camera or gallery and receive AI analysis.	Medium

ID	Requirement Statement	Priority
FR-10	A tourist shall be able to save and delete recognised landmark records.	Medium
FR-11	A tourist shall be able to browse live advisor packages with search, filter, and sort capabilities.	High
FR-12	A tourist shall be able to view full package details including itinerary, inclusions, and blockchain reviews.	High
FR-13	A tourist shall be able to create a booking for a selected advisor package.	High
FR-14	Tourist and advisor shall exchange messages within a package-specific chat channel.	Medium
FR-15	An advisor shall be able to create, edit, publish, and delete travel packages.	High
FR-16	An advisor shall view and update booking status through Pending, Accepted, Rejected, and Completed states.	High
FR-17	Blockchain review submission shall only be available on bookings with Completed status.	High
FR-18	The system shall record the reviewSubmitted flag per booking and prevent duplicate review submissions.	High
FR-19	An advisor shall view blockchain review analytics including average rating and review count per package.	Medium
FR-20	The system shall provide weather forecasts for saved cities and a news/events feed from RSS sources.	Medium
FR-21	The system shall issue local notifications for weather and news updates based on user-controlled toggles.	Medium
FR-22	The system shall provide privacy and security settings screens with persistent user-controlled toggles.	Medium

3.3. Non-Functional Requirements

Table 3.11 contains non-functional requirements related to quality attributes of the system. The mentioned requirements limit the options of implementing certain system aspects in further chapters.

Table 3.11: Non-Functional Requirements Catalog

ID	Category	Requirement Statement
NFR-01	Security	Passwords must be hashed using SHA-256 before storage; plaintext passwords must never be persisted or transmitted.
NFR-02	Availability	The application must handle startup errors and API failures gracefully without crashing or presenting an unrecoverable state.
NFR-03	Performance	Application startup must initialise key services in parallel to minimise user-perceived launch delay.
NFR-04	Usability	The UI must provide role clarity through separate navigation structures and support low-effort navigation between core features.
NFR-05	Maintainability	The codebase must follow a modular layered structure separating screens, services, models, and widget components.
NFR-06	Data Integrity	Booking status transitions and the reviewSubmitted flag must remain internally consistent throughout all state changes.
NFR-07	Auditability	Blockchain transaction hashes must be visible in the UI and linkable to a public block explorer for independent verification.
NFR-08	Portability	The Flutter architecture must support deployment across Android and iOS targets without platform-specific code duplication.

3.4. Interface Requirements

3.4.1. User Interface Requirements

- Special role navigation and dashboard designs that conceal any actions made across roles.
- Mobile interface design that remains consistent with typography, spacing, and colors used across the app.
- A clear indication of changes to booking status and submission status of reviews on the blockchain.
- Convenient feedback for all time-taking operations such as AI content generation, blockchain interactions, and data fetching using the internet.

3.4.2. Software and Component Interface Requirements

- The Mistral chat completion API endpoint for itinerary generation using AI and Mistral vision API endpoint for landmark analysis.
- The Open-Meteo API endpoint for geocoding and weather forecasting.
- The Google News RSS feed API endpoint with XML parsing capability for news discovery.

- The MongoDB Atlas connection via the mongo_dart package for all database operations.
- The Polygon Amoy RPC endpoint via Infura and web3dart for interaction with smart contracts.

3.4.3. Hardware and Device Interface Requirements

- The permission to access the device camera and gallery for landmark scanning.
- Connection to the internet for performing operations that rely on the use of APIs and blockchain.
- The permission to send out notifications on the mobile app regarding weather and news.

3.5. Database Requirements

The MongoDB Atlas database serves as the database of choice for all database operations within TripNexus. Collections within the database have been designed to ensure a clear boundary for efficient queries without inter-collection dependencies. This is detailed in Table 3.12.

Table 3.12: Core Database Collections and Responsibilities

Collection	Purpose	Key Fields
users	User identity, role, and authentication data	email, passwordHash, role, fullName, walletAddress
trips	Tourist-created trip records	userId, destination, startDate, endDate, budget, activities, ecoFriendly
saved_trip_plans	AI-generated itinerary persistence	userId, destination, duration, interests, planText, createdAt
saved_landmarks	Landmark scan and analysis records	userId, imagePath, analysisText, savedAt
packages	Advisor package catalogue	advisorId, title, description, price, duration, status, itinerary
package_bookings	Booking workflow state records	packageId, travelerId, advisorId, status, reviewSubmitted, bookedAt
package_messages	Per-package chat messages	packageId, senderId, receiverId, messageText, isRead, sentAt

3.6. Project Feasibility

3.6.1. Technical Feasibility

The selected tech stack, including Flutter, MongoDB Atlas, Mistral API, Open-Meteo, web3dart, and Polygon Amoy, meets all the requirements functionally. All integration points were tested based on documentation and availability of APIs during the initial development stages. Technical feasibility has been established.

3.6.2. Operational Feasibility

Creating a trip, booking a package, and submitting a blockchain review can be executed using user-friendly and guided interfaces for their roles, without any need for special technical skills. The same applies to the advisor's flow of activities. The metaphors used relate to the task at hand and the package management system. Operational feasibility has been established.

3.6.3. Legal and Ethical Feasibility

The current academic prototype addresses some minimalistic security measures through password hashing and role-based access control. In a production environment, it would be necessary to formulate a comprehensive privacy policy, ensure compliance with data protection regulations, provide a backend implementation for the identity verification pages, and manage secrets for encryption purposes. Blockchain reviews will contribute to ethical data governance. Conditional feasibility has been established pending pre-production requirements.

3.7. Analysis Models

The following analysis models were designed to model the internal structure and behavior of the TripNexus application. These models have been utilized heavily in developing the application architecture and components discussed in Chapter 4.

3.7.1. Activity Diagrams

Activity diagrams for Trip Creation with AI Planning and Advisor Booking Lifecycle Management have been drawn. The two diagrams below illustrate how the Traveler creates the trip and the Advisor creates the packages.

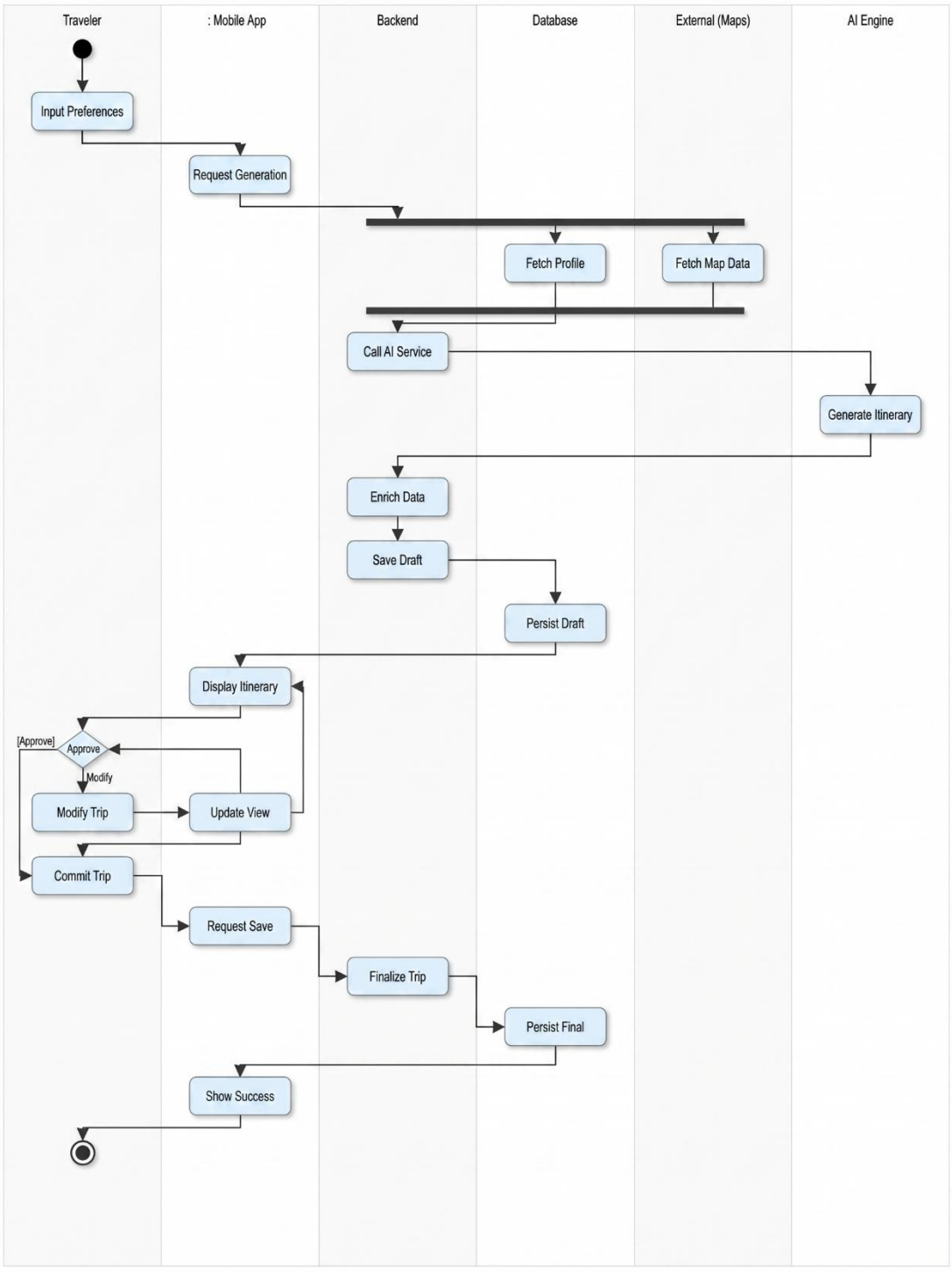

***Figure 3.2**: Activity Diagram — Tourist Booking Workflow*

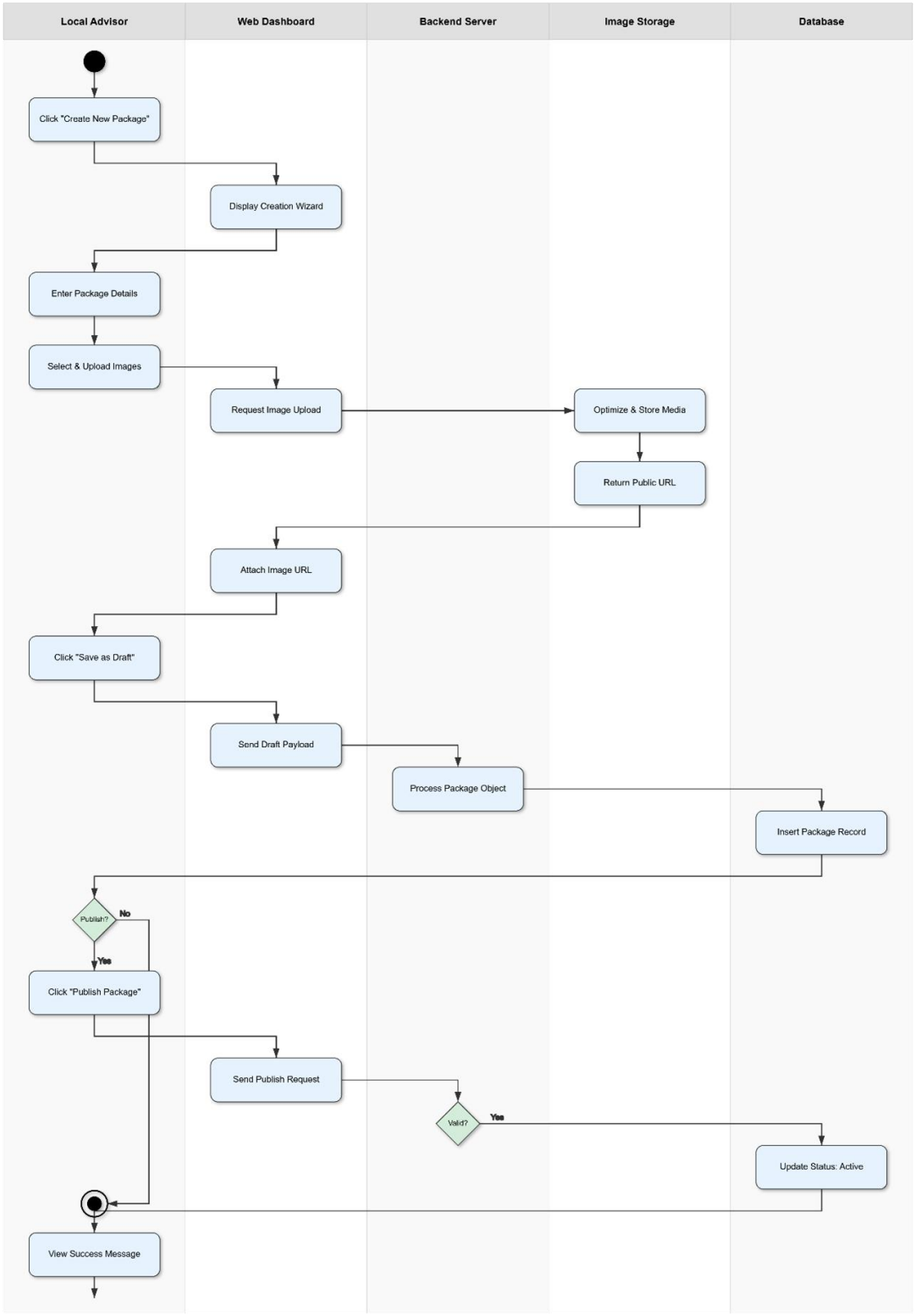

Figure 3.3: *Activity Diagram — Advisor Package Lifecycle Management*

3.7.2. Sequence Diagrams

Sequence diagrams are created for the edit and save the AI Trip itinerary for Tourist. This diagram document the temporal ordering of service interactions and the message-passing contracts between components.

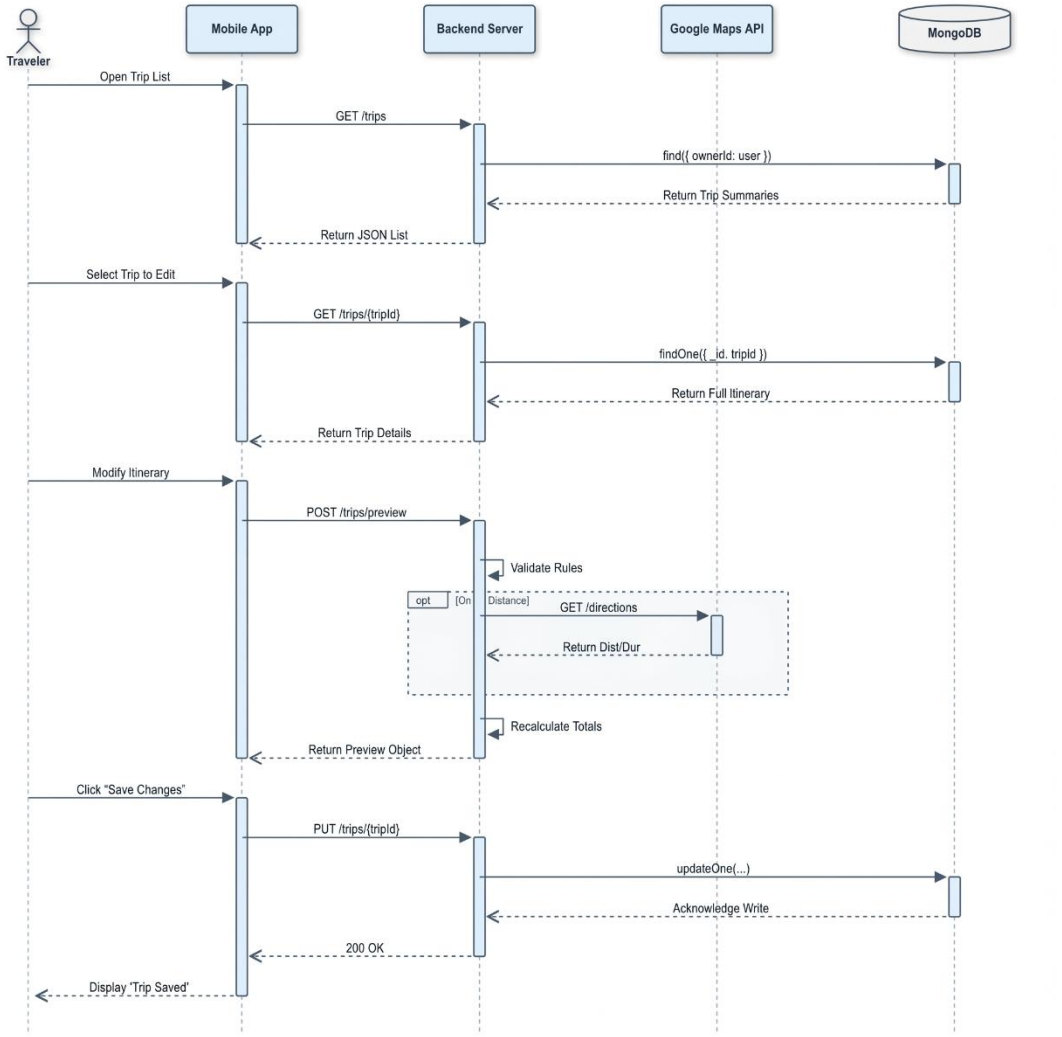

Figure 3.4: *Sequence Diagram — Edit and Save Trip by Traveler*

3.8. Risk Analysis

Table 3.13 below provides a summary of the risk assessment carried out during the planning phase of the project.

Table 3.13: Risk Analysis and Mitigation Strategies

ID	Risk Description	Impact	Mitigation Strategy
R-01	External API outage or rate limit exceeded	High	Graceful error states, retry logic, and cached fallback content where technically feasible
R-02	Blockchain transaction failure due to insufficient test MATIC	High	Balance pre-check before submission; user guidance message with faucet instructions on failure
R-03	Hardcoded sensitive configuration exposed in prototype build	High	Documented limitation with a clear migration path to a secure key-vault configuration system
R-04	MongoDB schema mismatch causing data parsing errors	Medium	Defensive model parsing with null-safe defaults and migration-safe field handling throughout services
R-05	Role authorisation gap allowing cross-role screen access	High	Role checks enforced at navigation entry points and at service method level for all critical operations
R-06	User notification fatigue reducing engagement	Medium	Independent user-controlled toggle switches per notification category in the settings screen

3.9. Conclusion

This chapter established a complete, testable, and traceable requirements specification for TripNexus. Twenty-two functional requirements and eight non-functional requirements define the full intended behaviour of the system, supported by interface specifications, database design, feasibility assessments, structured analysis models, and a risk register. This specification provides the authoritative baseline against which the system design, implementation, and testing results are evaluated in subsequent chapters.

Chapter 4
System Design

In this chapter, we will discuss the design aspects, the architectural model, components involved, logical and dynamic design, data design, and the design of user interfaces in the TripNexus solution. Iterative design and implementation processes were employed alongside the design process of features of this solution based on requirements analysis and testing feedback.

4.1. Design Approach

For designing the TripNexus application, the vertical slices were chosen as an approach to develop fully functional prototypes. In contrast to the horizontal slices, the delivery of the prototype involves adhering to five major principles:

1. Clear roles in every transaction, so that none of the tourists' actions could be performed from the context of an advisor and vice versa;
2. Services boundaries which are modular and guarantee that no changes to the UI will be needed to switch an external provider;
3. Minimum chain of data transfer to ensure good operational performance, preserve user privacy and provide immutable reviews;
4. Safeguards against any change to the data schemas in the data parse during reading from any service;
5. The state management for the whole booking life cycle provides the consistency between all view states..

4.2. Design Constraints

- Limited academic project timeframe made it impossible to fully harden and implement additional features concerning security and scalability as mentioned in further work.
- A backend server with API is not implemented in the architecture to make the architecture less complicated and allow communicating directly with cloud-based services; however, there are plans to introduce centralized gateway in the future.
- Varied availability and costs associated with the testing environment due to variability of the third-party APIs and testnet blockchain introduced environment-specific error conditions and made it difficult to repeat tests.
- Blockchain transactions can only be done in the Polygon Amoy testnet environment to avoid costs and potential risks related to main net deployment.

4.3. System Architecture

The architecture of the TripNexus can be described using the following four tiers. The Presentation Layer is comprised of the Flutter UI and its components, navigation and reusable widgets. The Service Layer contains classes responsible for various types of service: authentication, packages, weather and news, AI services, notifications and interactions with different blockchain. Each class in this layer has a distinct interface contract. The Data Layer comprises three kinds of data storage: MongoDB Atlas collection, SharedPreferences and notifications storage. The External Layer consists of services that are used in the Services Layer to perform their tasks such as Mistral, Open-Meteo, Google News RSS and Polygon Amoy RPC.

The boundaries of the services were established via interface contracts for the model objects allowing switching concrete implementations of the service without changing the presentation layer. This directly addresses NFR-05 (Maintainability).

Figure 4.1: High-Level System Architecture — Four-Tier View

4.4. Logical Design and Class Model

The Domain Model revolves around role-aware entities linked together through Booking and Messaging. The main entities and associations between them are the following:

- **User** (abstract) carries identity, email, role, and status fields and is the root entity for all role-specific flows (Traveler, LocalAdvisor, Admin).
- **Traveler** inherits from User, maintains an array of saved trips, and is responsible for creating trips and requesting advisor help.
- **LocalAdvisor** inherits from User, carries business name and expertise fields, and handles package creation and request responses.
- **Trip** is created by either a Traveler or an AIEngine and contains a title, date boundaries, status, and manages the generation of itineraries.
- **DayPlan** is strictly contained within a Trip and organizes a specific day number, date, and its nested activities.
- **Activity** is strictly contained within a DayPlan and carries identifying details, name, and type.
- **Request** acts as a communication bridge, linking a sending Traveler to a receiving LocalAdvisor, and tracks the lifecycle status of the inquiry.
- **Admin** inherits from User, carries permission arrays, and monitors system actions via audit logs.
- **WeatherIntelligence**, **Event**, and **Landmark** associate with a Trip to provide contextual details, geographical points of interest, and safety summaries.

Key domain relationship rules: User entity is an abstract class extended by Traveler, LocalAdvisor, and Admin classes; One Traveler generates zero or many trips; AI Engine generates zero or many trips; One Trip is consisted of one or many DayPlans (composition); One DayPlan is consisted of zero or many Activities (composition); One Trip is consisted of Events and Landmarks besides Day Plans and WeatherIntelligence (0..*); One Traveler sends zero or many Requests handled by exactly one Local Advisor; An Admin monitors composition of zero or many AuditLogs.

.

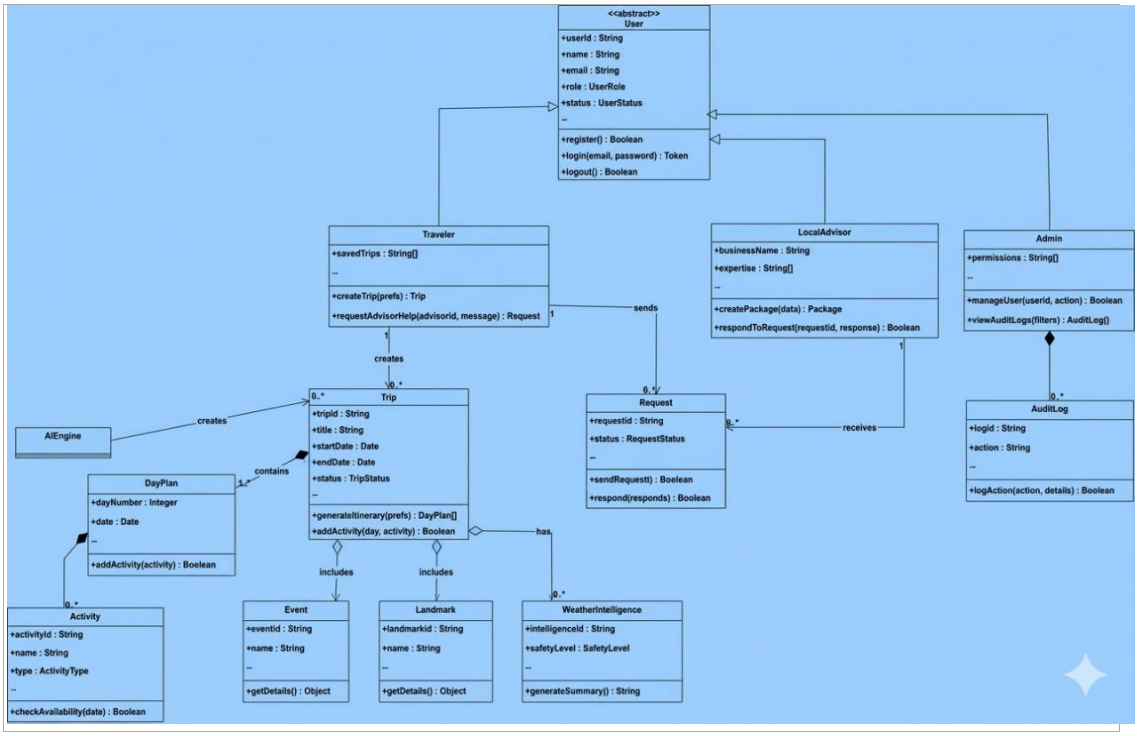

Figure 4.2: Class Diagram (Domain Model)

4.5. Dynamic View – Sequence Diagrams

The sequence diagram illustrates the chronological order of communication between components and service contracts that facilitate the passing of messages. The major sequence diagrams illustrated include: Tourist registration and login sequence with session management, AI trip planning request-reply sequence, and package booking and advisor decision-making sequence.

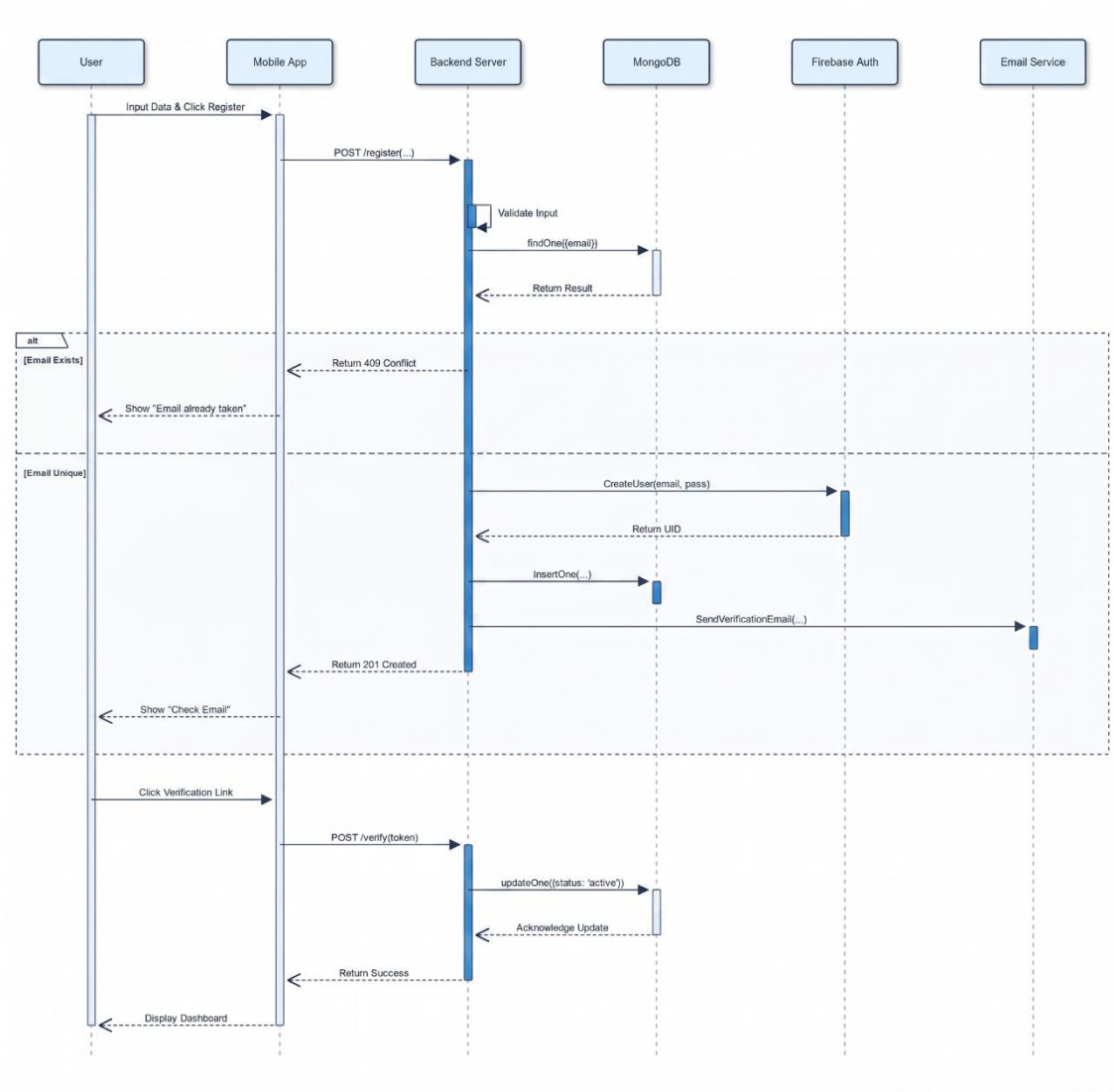

Figure 4.3: Sequence Diagram — Tourist Registration

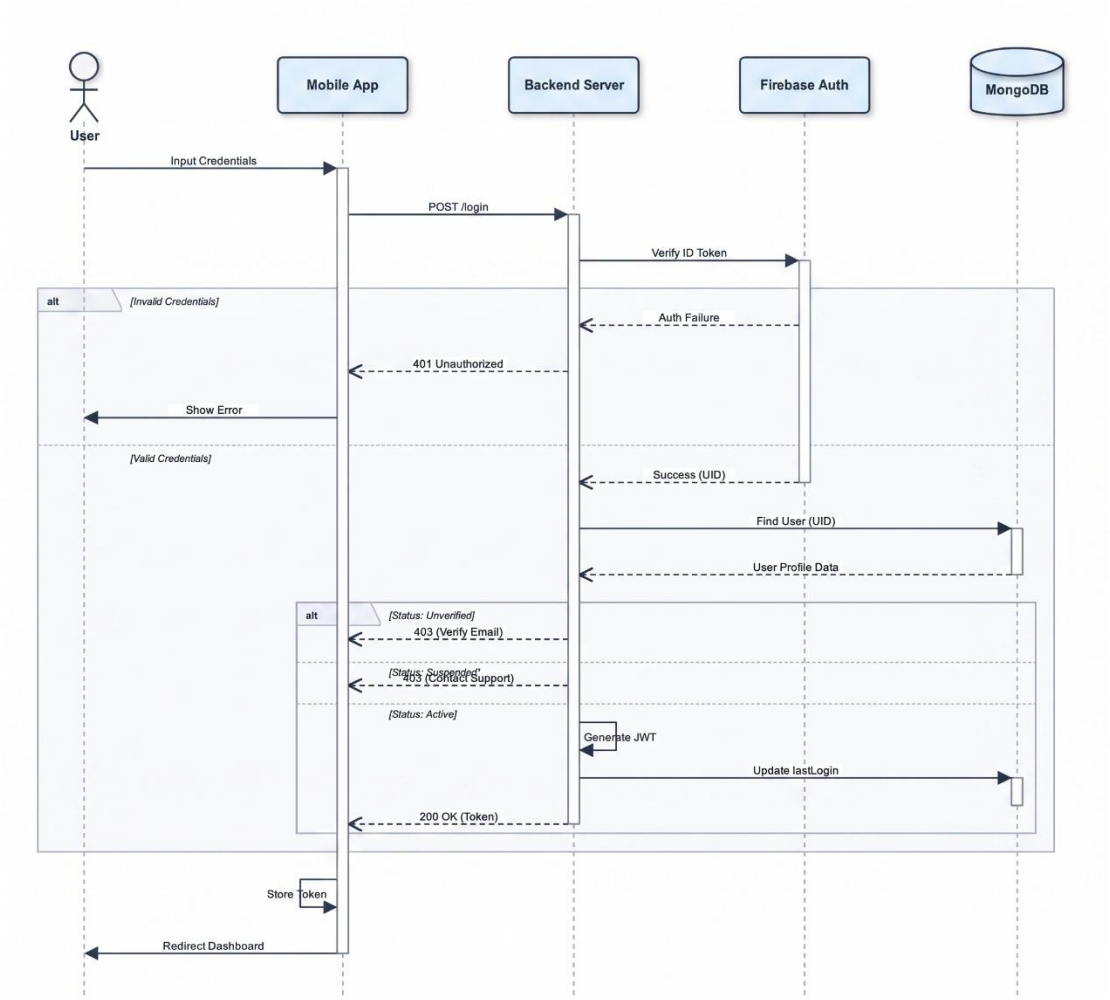

Figure 4.4: Sequence Diagram — Tourist Login

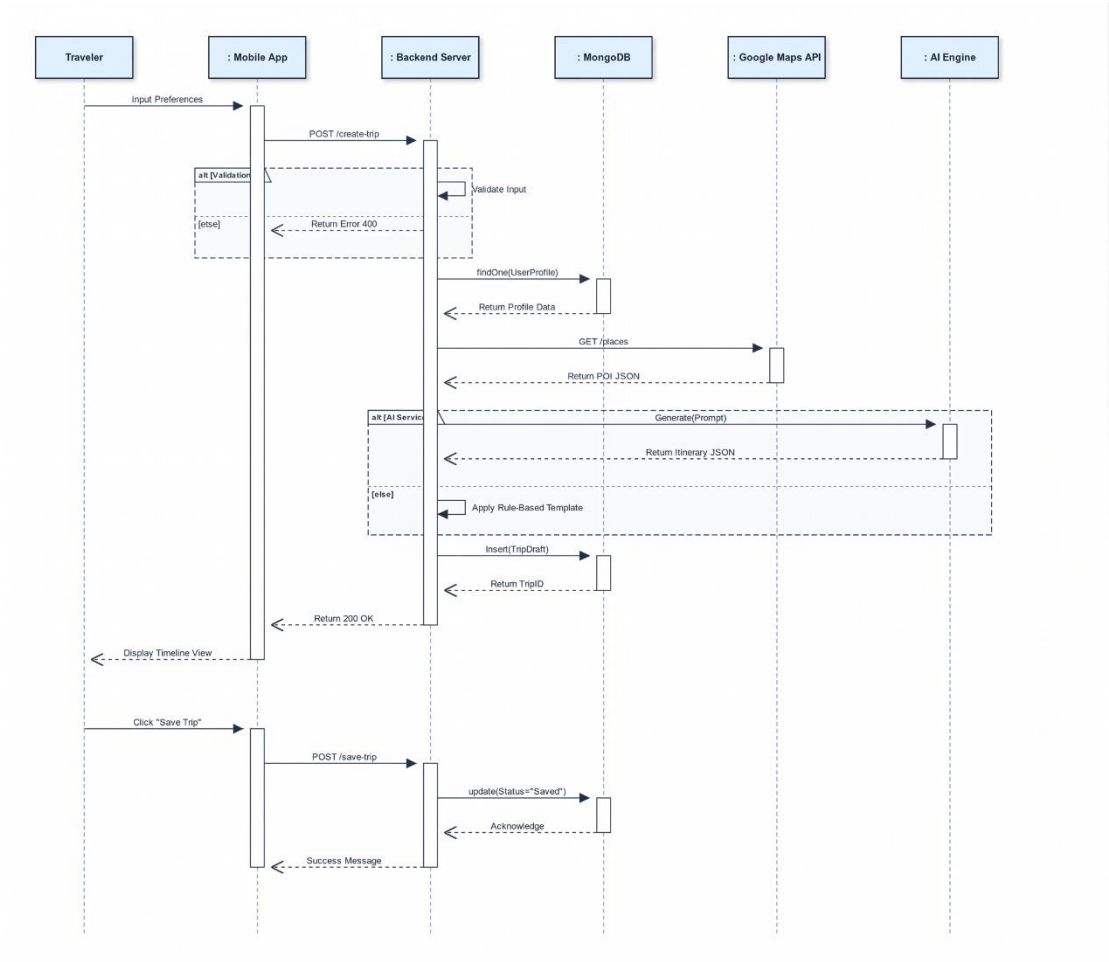

Figure 4.5: Sequence Diagram — AI Trip Generation Flow

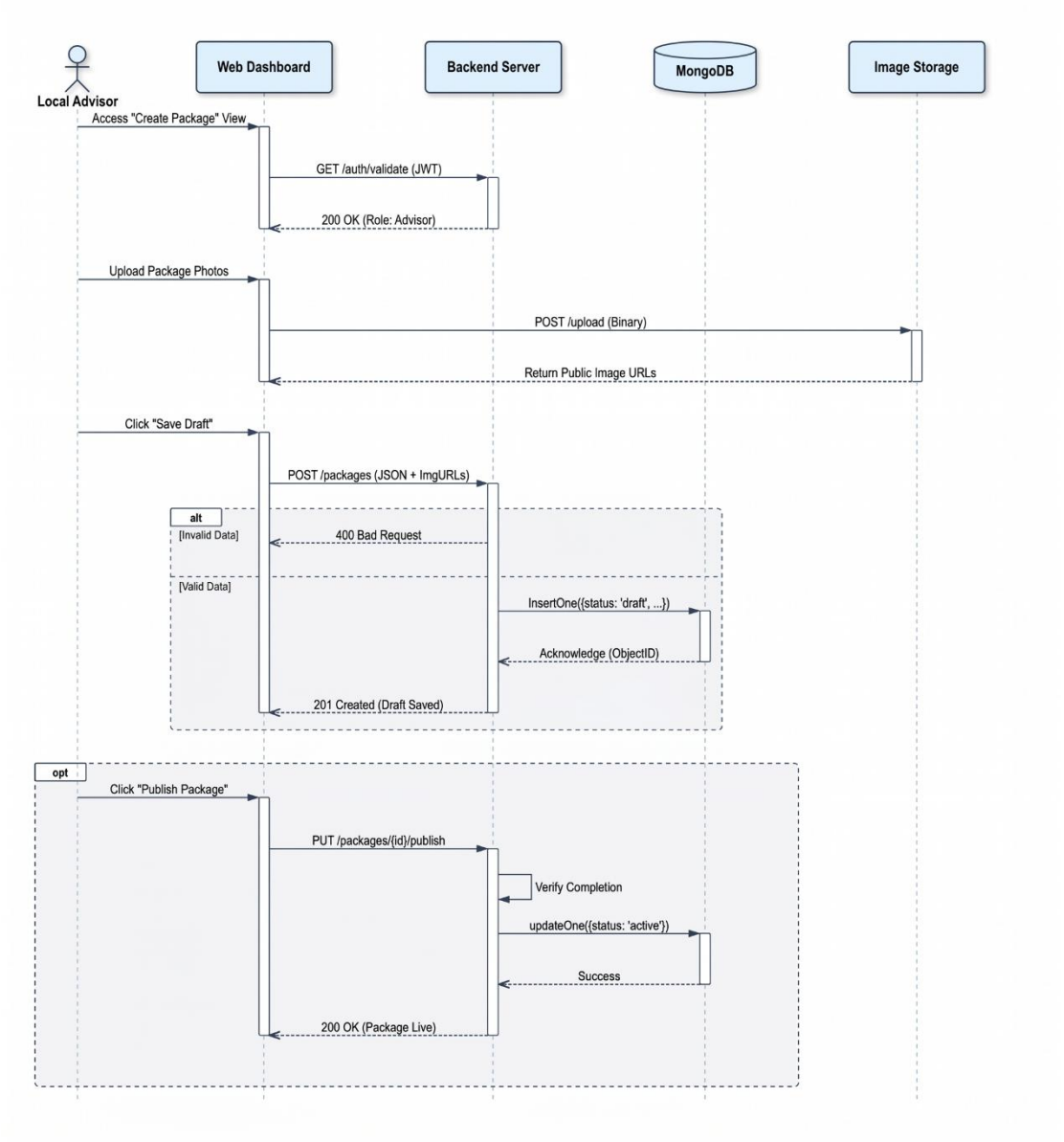

Figure 4.6: Sequence Diagram — Local Advisor Create Package

4.6. Component Design

The architecture of the TripNexus software consists of six primary services with well-defined service boundaries. The first of these is the Auth Service, which handles authentication, password hashing, session management, and route determination depending on the role. The second is the Tourist Planning Service, which handles trip management, AI planning, and landmark scanning operations. The third service is the Marketplace Service, which handles package searching, booking generation, and chatting messages. The fourth service is the Advisor Operations Service, which provides CRUD operations of packages, booking operations, and analysis operations. The fifth service is the Blockchain Service, which manages review contract initialization, transactions, and reviews search.

Coupling is made intentionally loose through service contracts and models rather than tight coupling of the services themselves. Thus, tests can be done independently and individual components substituted without disrupting the whole system.

4.7. Data Models

Package bookings is the main database where all the information about the tourist, advisor and package booking process are stored in one document. Package's state transitions and review statuses will be maintained in that document. Field 'reviewSubmitted' boolean is the integrity constraint that links off chain booking system to on chain review entry. This is shown in Figure 4.5 below.

Booking state transition diagram demonstrates booking as a finite state machine containing four states; pending state (the state at creation of a booking record), accepted state (advisor accepts), rejected state (advisor rejects) and completed state (advisor is satisfied with the work). Only one state gives access to blockchain review system – the completed state.

4.8. User Interface Design

4.8.1. Design Principles

- Role clarity: navigation paths for tourist and advisor are entirely separated, guaranteeing that no action outside the role is presented within the interface.
- Information hierarchy: booking status indicators, package price, and review state are visually prioritised above secondary information.
- Feedback design: prolonged operations, such as AI creation and blockchain transaction, have loading indicators and snackbar feedback messages.
- Visual consistency: common theme configurations are used to ensure consistent color schemes, typeface scales, grid spacing, and user interfaces.

4.8.2. System Prototype

The UI prototype design went through four major rounds. The first prototype, named Foundation, consists of login flow and role-based dashboard screens. The second prototype, named Planning, covers trip management, AI planning, and landmark scanning screens. The third prototype, named Marketplace, includes package browsing and booking screens, chat feature, and advisor management screen. The last prototype, named Trust, involves blockchain review dialog, transaction result screen, and advisor analysis screen.

Registration & Login Page

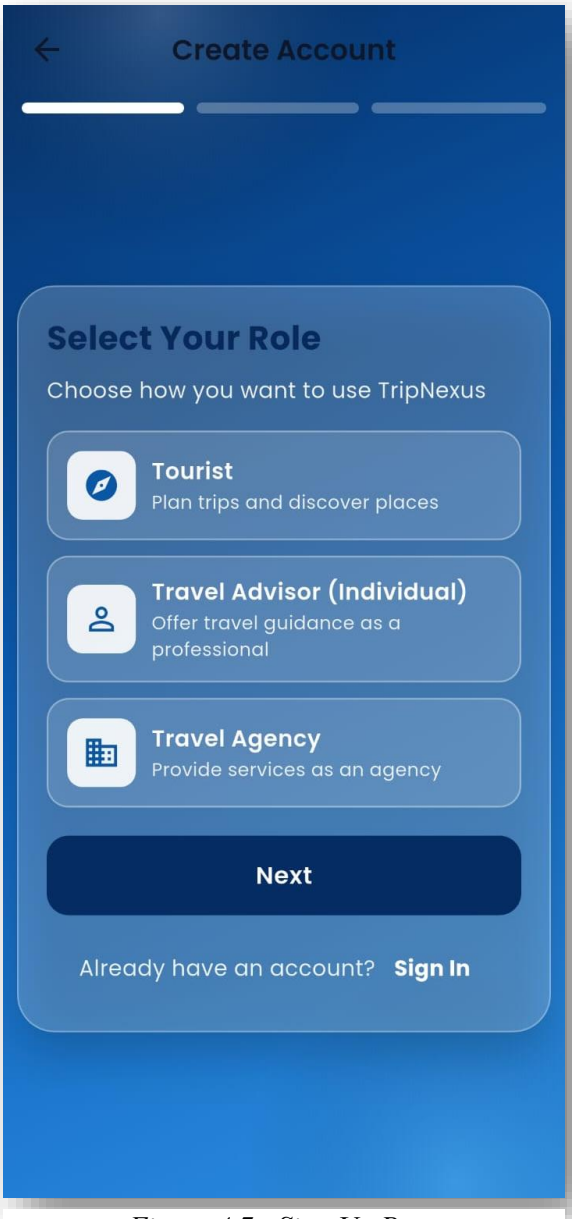

Figure 4.7 - Sign Up Page

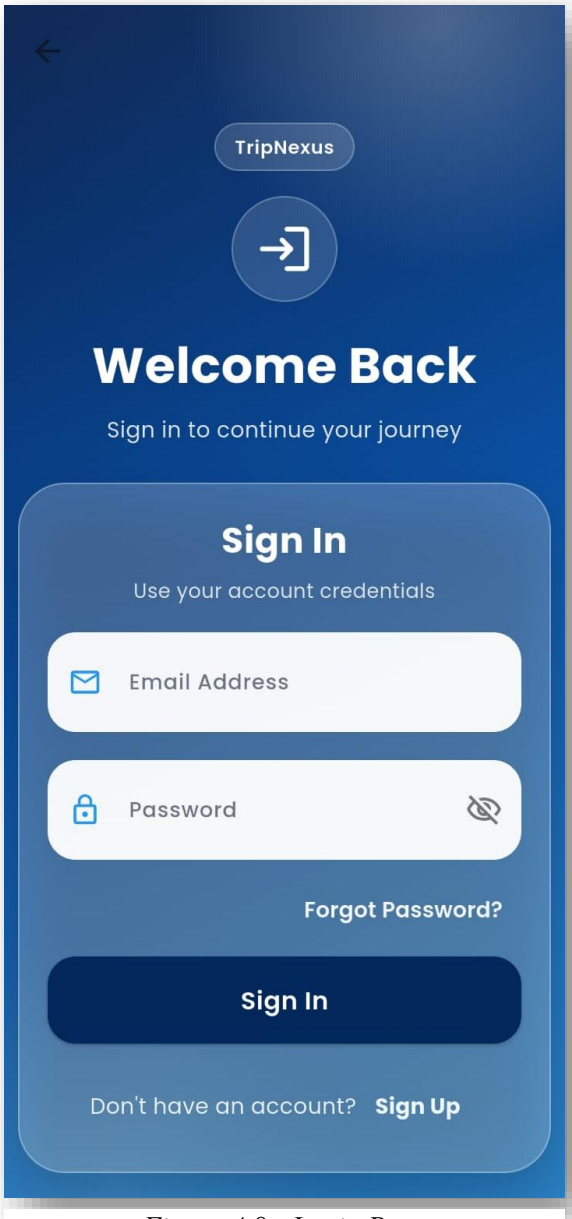

Figure 4.8 - Login Page

Traveler Dashboard & Profile Page

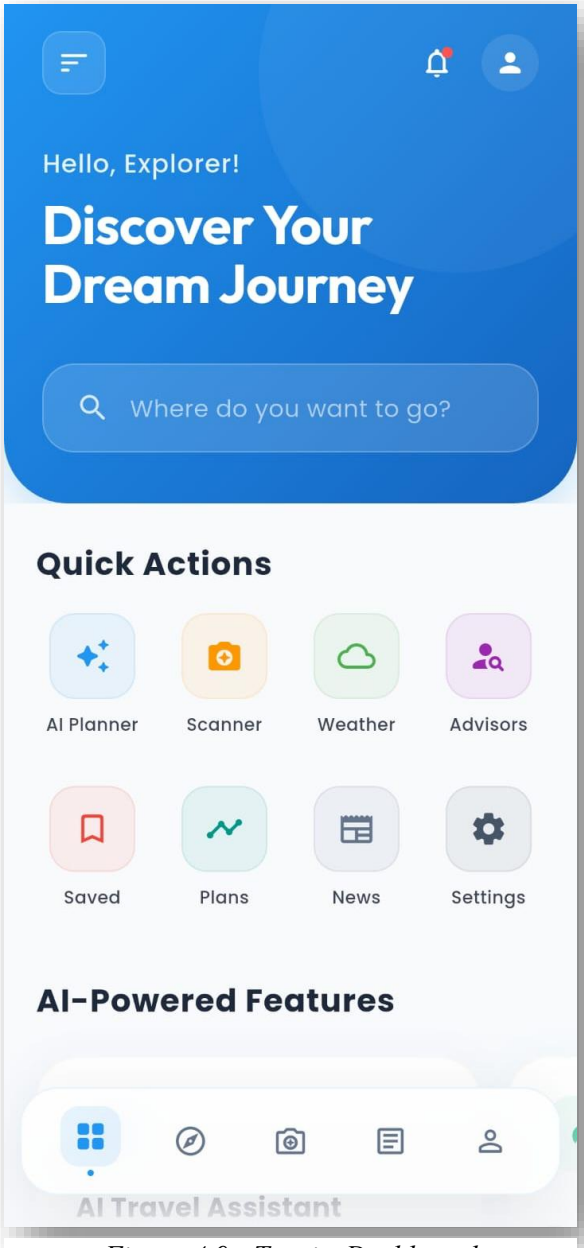

Figure 4.9 - Tourist Dashboard

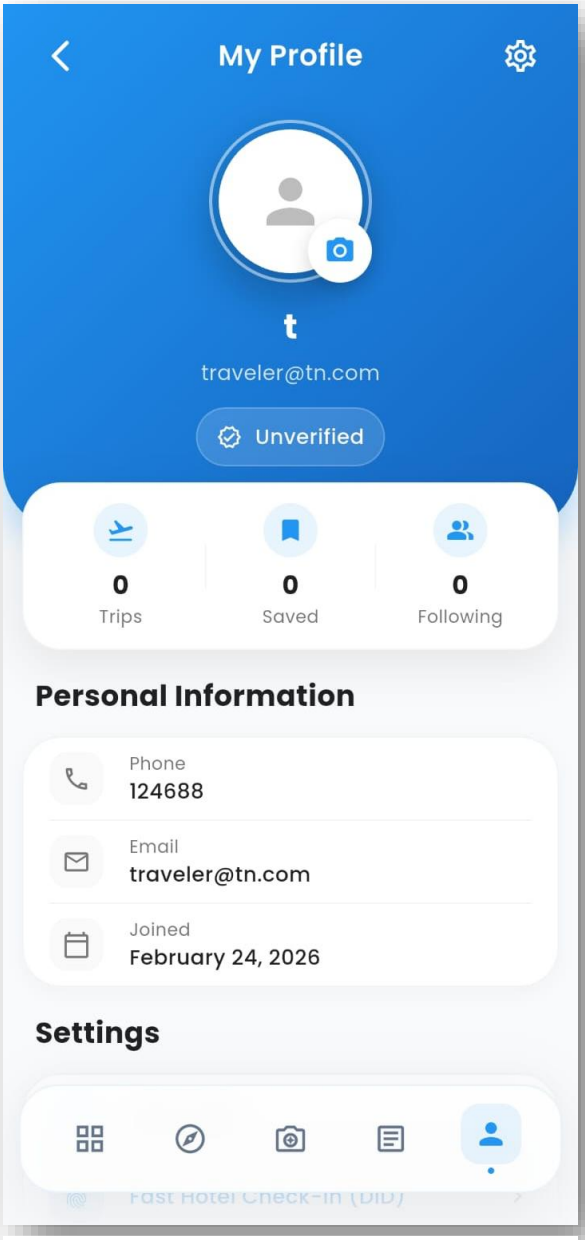

Figure 4.10 - Profile Page

AI Trip Planner Page & Generated Itinerary

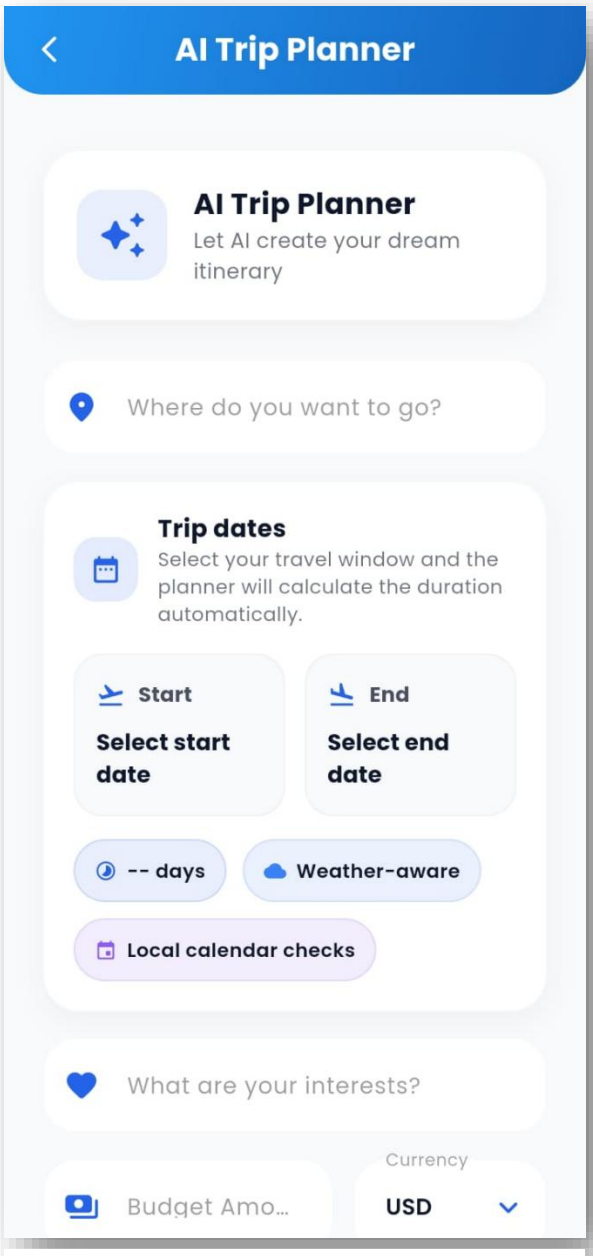

Figure 4.11 - Trip Planner Page

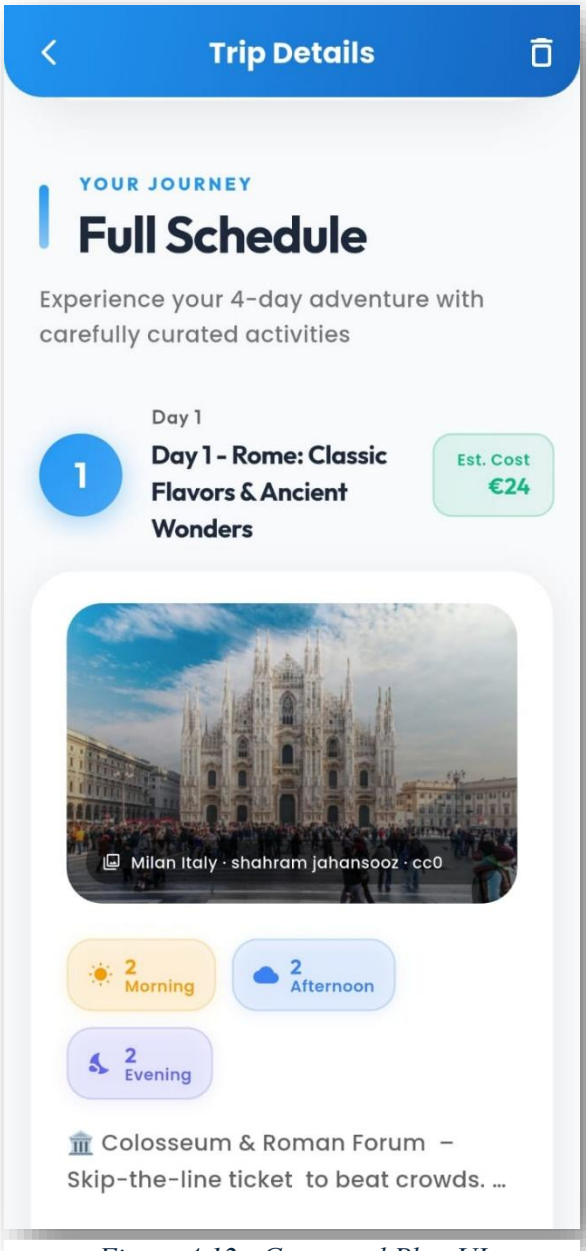

Figure 4.12 - Generated Plan UI

Landmark Scan Page

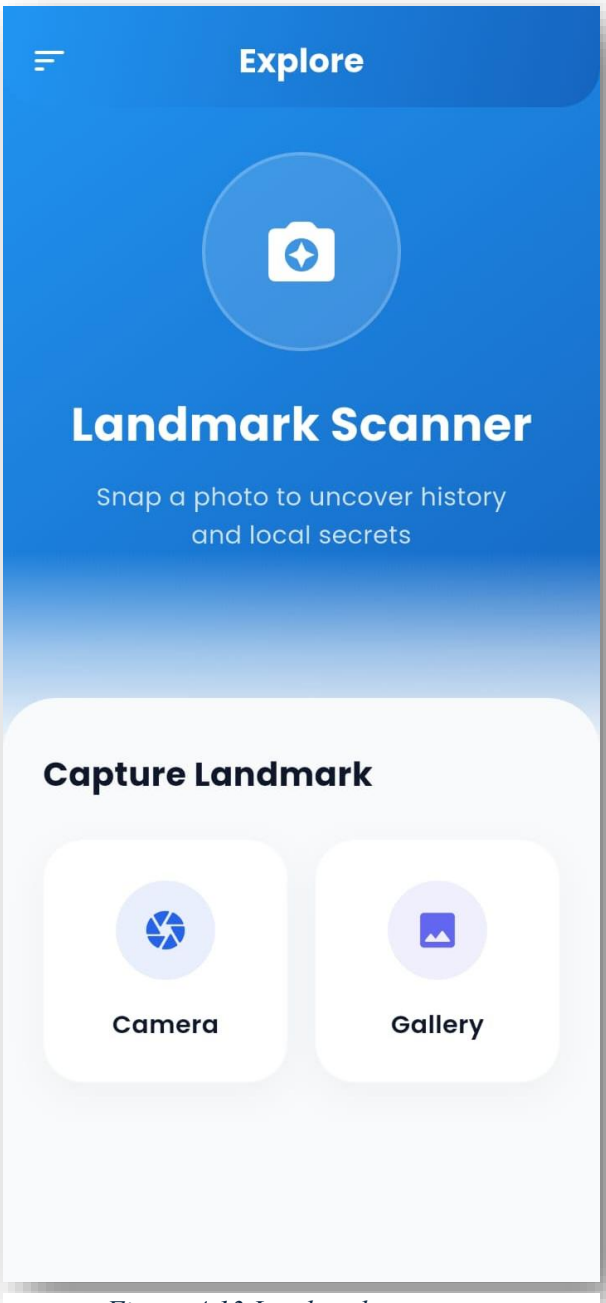

Figure 4.13 Landmark scan page

Weather & News detail Page

Figure 4.14 - Weather Page

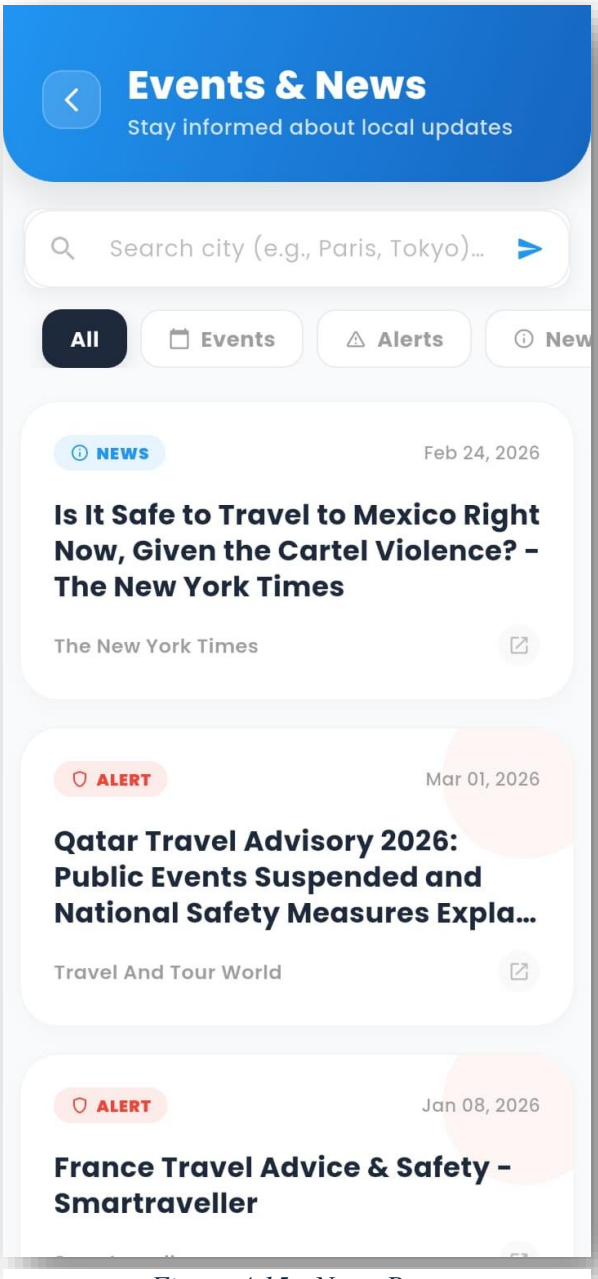

Figure 4.15 - News Page

Advisor Dashboard Page

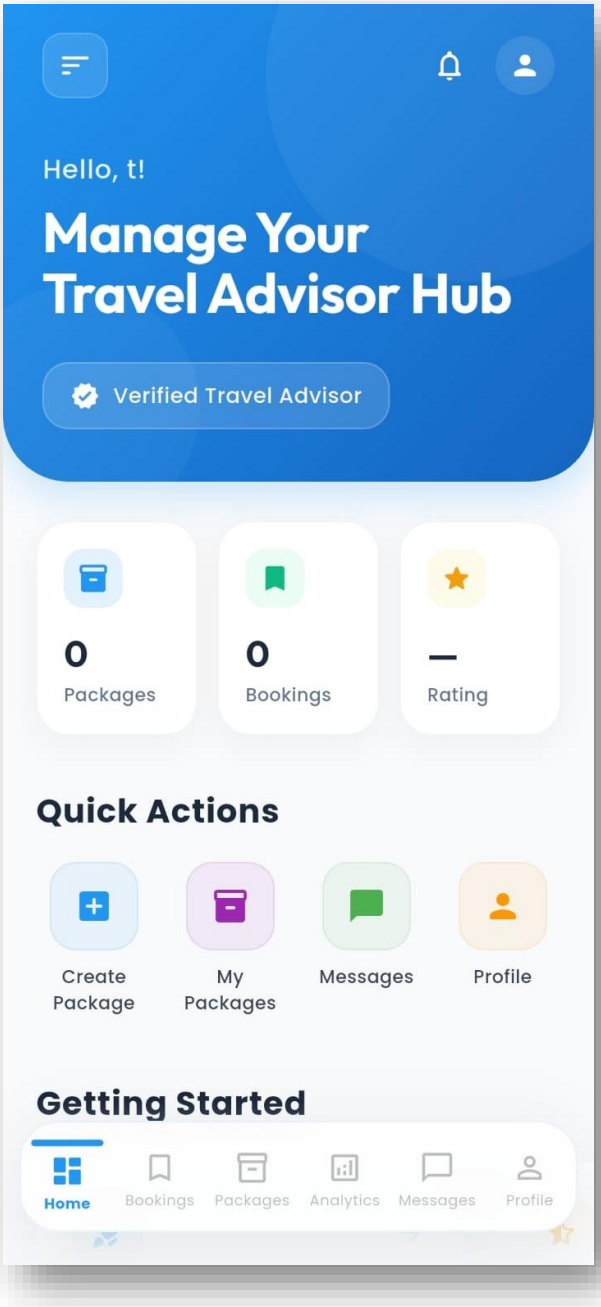

Figure 4.16 - Advisor Dashboard

Advisor Package Creating Page

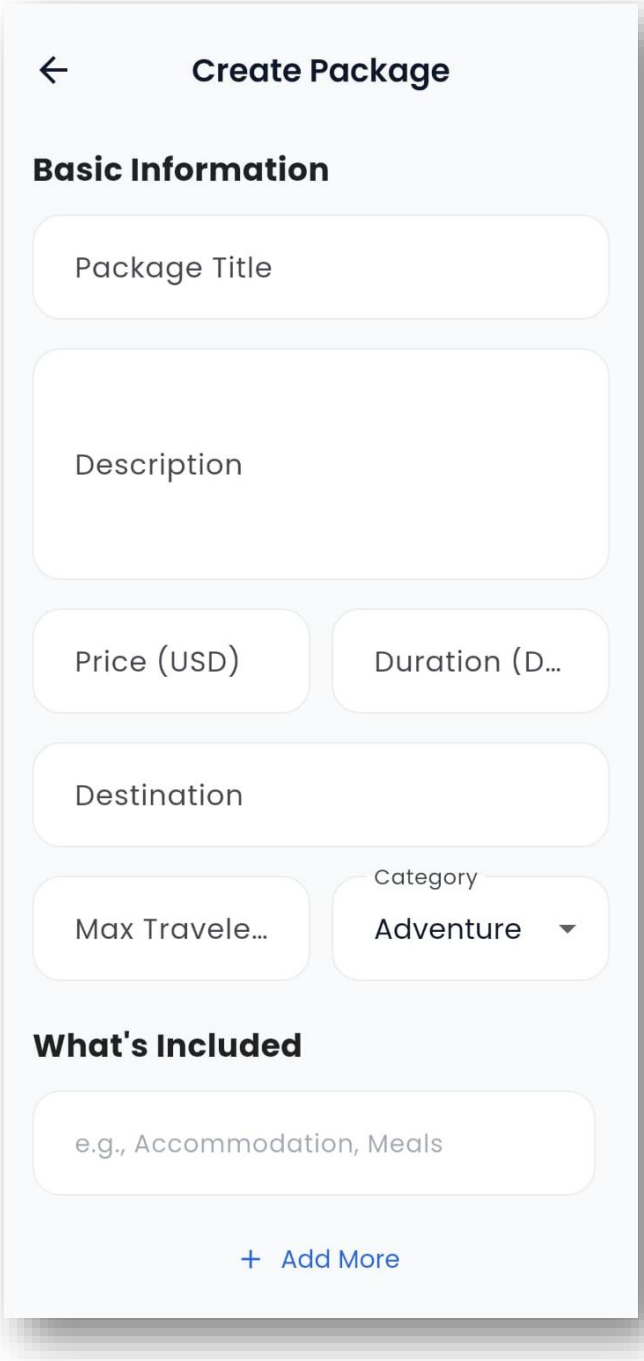

Figure 4.17 - Advisor Package Creating Page

Traveler Review Submission Page & Confirmation

(Gives a hash id for submitted review to be view on polygonScan officical site)

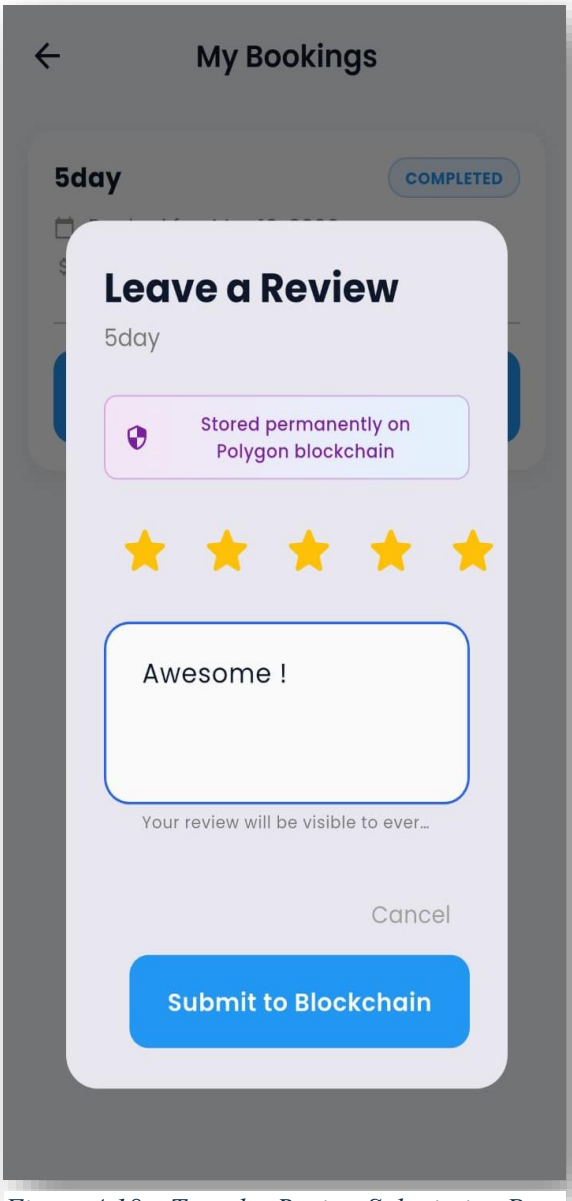

Figure 4.18 - Traveler Review Submission Page

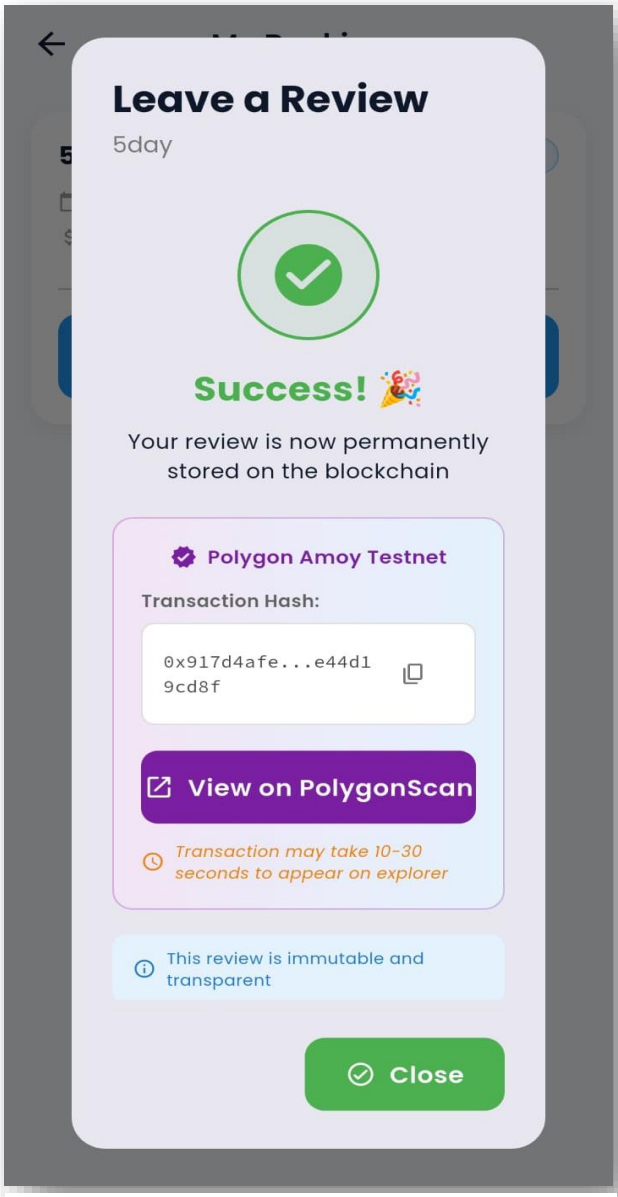

Figure 4.19 - Traveler Review Confirmation Page

4.9. Security by Design

The process of SHA-256 password hashing is done not only at registration before storing in a database but also during logging-in prior to comparing credentials. In no way can a clear password be transmitted over any connection and saved anywhere in software. The routing is aware of the role of the user both when checking permissions via navigation guard and using service itself. Only wallet address, package ID, numeric rating, comments and timestamp are included in the list of blockchain review payload fields, thus making the leakage of sensitive information restricted to an immutable ledger. The migration strategies foreseen for secret management, identity confirmation, and API mediation are among primary security concerns of the future application.

4.10. Conclusion

This design approach produced a holistic design that implements all of the necessary features in testable architecture components. The core design decision here was the use of the hybrid trust model approach which guarantees not only operability of off-chain booking but also ensures the immutability of reviews within the on-chain part of the system. One has to point out explicitly that such an approach solves the problem outlined in Chapter 1.

Chapter 5
System Implementation

The chapter describes the implementation of TripNexus system within all major components along with the descriptions of technologies used, feature implementations, solutions found to encountered issues in development, etc. This chapter is based on architecture described in Chapter 4 and functional requirements discussed in Chapter 3.

5.1. Development Stack and Environment

Table 5.1 gives the complete list of technologies utilized in the TripNexus implementation process. All the technologies selected are appropriate to perform the desired tasks and can be easily implemented within the student project framework.

Table 5.1: Development Technology Stack

Component	Technology	Purpose
Frontend Framework	Flutter (Dart ^3.10.0)	Cross-platform mobile UI and all application logic
Database	MongoDB Atlas via mongo_dart	Persistent application data storage for all business entities
AI Integration	Mistral API (chat + vision)	Itinerary text generation and landmark image analysis
Weather Data	Open-Meteo API	Real-time multi-city weather intelligence and forecasts
News Feed	Google News RSS + XML parsing	Local events and news article discovery and categorisation
Blockchain Platform	Polygon Amoy Testnet	Immutable review storage through smart contract deployment
Smart Contract	Solidity + web3dart client	ReviewContract deployment, transaction signing, and retrieval
Local Storage	SharedPreferences	Session persistence, settings, and notification history
Notifications	flutter_local_notifications + timezone	Scheduled weather and news alert delivery

5.2. Authentication and Session Management

In the registration module, there are three types of registration that depend on the chosen role: Tourist, Advisor Individual, and Advisor Agency. The registration form changes according to the selected role; also, for security reasons, passwords are encrypted via the SHA-256 hashing algorithm before being saved into MongoDB because the purpose is to ensure that passwords won't be transferred from the client to the server and saved in plaintext. In order to log into the application, the login procedure verifies the correctness of the entered credentials by computing the SHA-256 hash of the entered password.

The management of the user session includes the storage of the information about the user – their ID, display name, and role – in SharedPreferences in the form of a JSON object snapshot. At the very beginning of loading the application, the splash screen checks whether the data is already present and then performs further actions accordingly. Consequently, after launching the application, the logged-in user gets automatically sent to their dashboard.

5.3. Tourist Module Implementation

5.3.1. Trip Management

Trip management is accomplished through storing the destination point, start and end dates, list of activities inputted by the user, a budget field (optional), and eco-travel status. All trips are stored into the trips MongoDB collection, allowing users to see the list, which offers an opportunity to delete individual records from it. The eco-travel status field was added to support SDG 12 and provide an option for users to plan their trip using this criterion.

5.3.2. AI Itinerary Planning

The planning interface asks users to enter the destination name, travel duration (number of days), travel preferences, and maximum budget (optional). All fields constitute a prompt template, which is passed to the Mistral chat completion API endpoint [12]. Consequently, a user gets a travel plan with daily suggestions on what and where they should visit during their travel. Plans may be saved into the saved_trip_plans database collection for further usage. There is a loading animation on the interface until the API returns data, and an error message pops up in case it fails to receive any because of network problems.

5.3.3. Landmark Scanner

The landmark scanner module receives the images as inputs from the device’s camera or gallery through the image_picker library. The selected image is then encoded in base64 and transmitted alongside the prompt that asks about the landmark and its importance in history and culture, as well as tourist recommendations for visiting the landmark on the Mistral vision server endpoint. The resulting text output is displayed

on the application interface and can be optionally saved to the saved_landmarks table with the file path and date.

5.3.4. Weather and News Modules

The weather module takes advantage of the geocoding function provided by the Open-Meteo API. By converting the entered city name into geographical coordinates, it obtains the weather forecast data for the next seven days for the user's saved city. The weather module allows the user to save his/her list of cities and navigate between tabs with weather forecasts for those cities. In the news module, the RSS feeds from Google News can be utilized. By parsing XML data with the help of xml package, one is able to classify news articles as local, safety, and general.

5.4. Advisor Module Implementation

There are four main functions of the advisor module. Package management encompasses all functions involving CRUD activities on packages, which have the title, description, destination, duration, price, offered services, and travel itinerary. Advisors can choose to enable or disable packages that should be published. Booking management involves compiling all bookings associated with the advisor's packages and automatically filling the package title and traveler name for each booking record. The booking management function follows the state machine model, with the Pending status being followed by either Accepting or Rejecting, and then accepting leads to completing the status. Completing the status alone provides access to the blockchain review submission form for tourists. The chat function allows for communication between a chosen traveler and the advisor through a particular package.

5.5. Blockchain Review Implementation

In the solidity contract, there is a declaration of ReviewContract, where the declaration of the Review struct is performed and includes the traveller wallet address, package identifier string, rating number, comment string, and UNIX timestamp. There are three declared functions within the contract, such as addReview, getReviews, and getReviewCount, which add the review into the blockchain through the creation of the review object and sending the ReviewAdded event, returning all review objects for certain package identifier and returning the number of reviews for specified package, respectively.

As for the Flutter part, the instance of BlockchainService is created only once at the start-up time of an application using the Polygon Amoy RPC node from Infura, credentials that are generated based on the private key defined in the configuration file, and ABIs that are loaded from application configuration. After submitting the user's review via a review dialog window, BlockchainService generates a transaction which calls the addReview method with specified data and sends it using credentials.

After completing the submission procedure on-chain, the booking record within the MongoDB database is updated by changing the field "reviewSubmitted" to the value

"true". This value will be checked while loading the list of bookings to prevent the reviewer from re-clicking on the review button for the same booking. On the analytics page, the on-chain reviews for the advisor's packages are fetched using the method "getReviews".

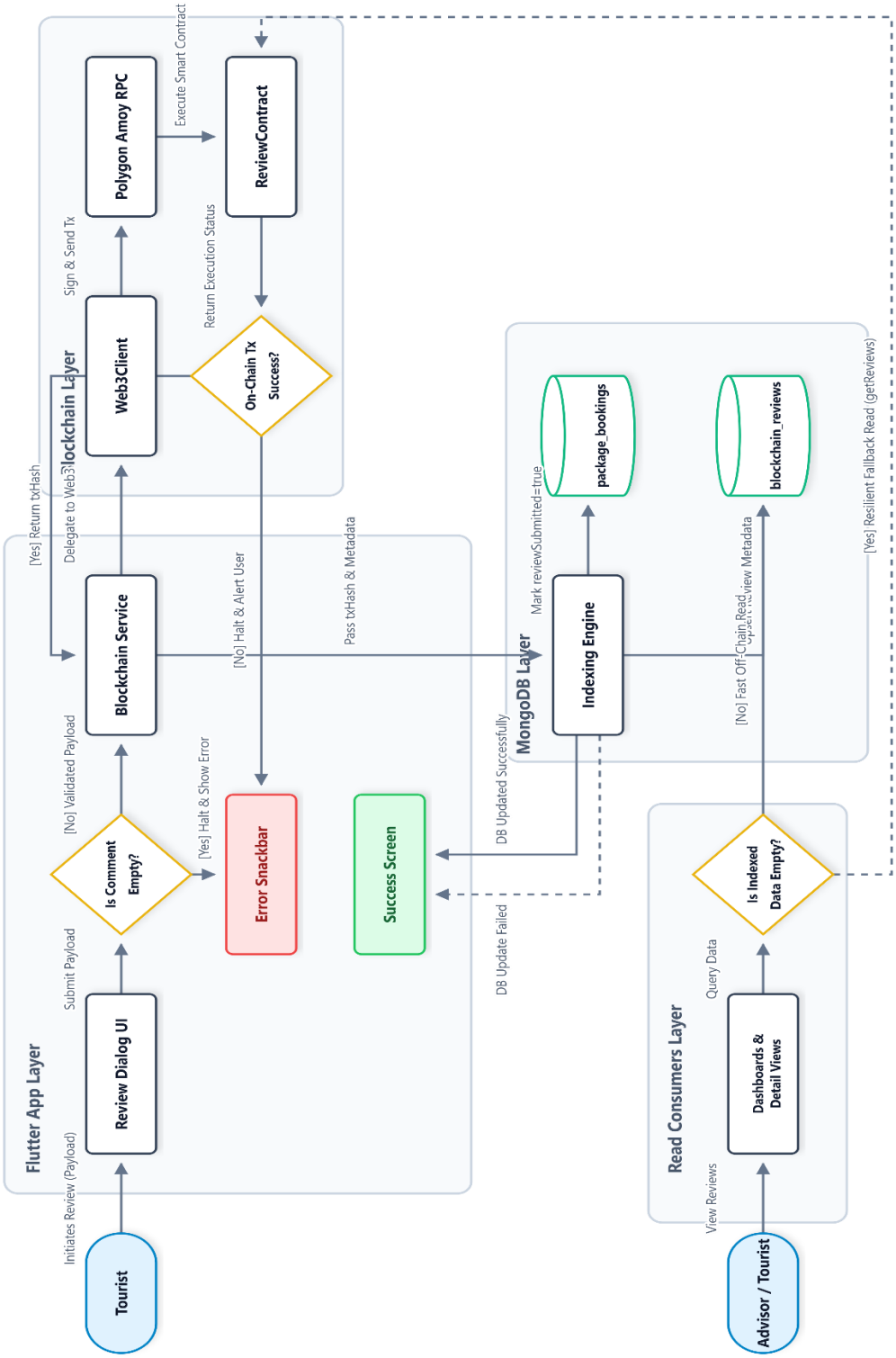

Figure 5.1: Blockchain Review Submission Flow — from Dialog to PolygonScan Verification

5.6. Implementation Challenges and Resolutions

- **Normalisation of ObjectId:** Some MongoDB document ids were shown as nested BSON objects rather than strings at the moment of calling the services. Fixed using a centralized normalisation process over all ids before executing each and every call.
- **Initialisation Race Condition for Services:** Problems occurred with accessing the service states during initialisation, while it was still ongoing. Fixed with an appropriate control flow around the singleton instance initialisation that locks its state and denies access for the services until completion.
- **Transaction Reliability Issues:** The first runs produced generic errors with a message indicating problems with wallet containing no test MATIC set up for the service. Fixed with a condition checking and an error message with instructions to acquire more tokens from the faucet website.
- **UX Problem Caused by Latency:** The introduction of a latency problem caused by both AI models calls and transaction submission led to an inconsistent UI state. Solved with loading animation indication and buttons disabling in case of pending request or transaction.

5.7. Code Organisation

Project architecture follows the design of four logical layers explained in Chapter 4. The lib/screens layer includes all the UI screens for particular roles. The lib/services layer contains all the integration classes and services used by the application. The lib/models layer stores all domain objects and implements parsing and serialization methods. Finally, the lib/widgets layer includes reusable UI components that are used by many screens.

5.8. Conclusion

The architecture of the project adheres to the structure of four logical layers described in Chapter 4. Specifically, the lib/screens layer comprises all UI screens for the role specified. All integration classes and services are present in the lib/services layer. The domain objects along with their parsers and serializers are stored in the lib/models layer. Lastly, lib/widgets layer consists of reusable UI components.

Chapter 6
System Testing & Evaluation

The testing strategy, specification of the test cases, execution of test runs and the evaluation results of TripNexus are provided in this chapter. The tests were performed according to the requirements established in Chapter 3.

6.1. Test Strategy

In designing the testing strategy for TripNexus special emphasis was put on the application characteristics. The software has a complex integration architecture involving use of various APIs, services, databases, and networks such as: language model API, weather data provider, a cloud database, and a blockchain network. Not only the correct execution of the TripNexus functionality in normal operating environment was verified but also its correct performance in the presence of faults. Five test categories were used for verification of TripNexus functionality: component validation, unit testing, integration testing, end-to-end testing, and specific edge-case testing related to the interactions with blockchain. The most important test type is the manual scenario execution due to integration nature of TripNexus.

6.2. Component Testing

Every single major system component including AuthService, PackageService, MistralIntegrationService, WeatherService, NewsService, BlockchainService and NotificationService was validated at the component level. Every component was checked for proper startup and primary functioning with a valid input. Additionally, all failure cases and invalid inputs were tested in the context of the component. All major component modules were tested successfully before moving to the integration phase.

6.3. Unit Testing

Unit level testing is not applicable for TripNexus due to its architectural design. Nevertheless, some of the most important logic and map functions were tested separately, namely: parsing string roles to Role enum, mapping booking statuses to UI states, normalising ObjectId values for two input variants (strings and objects) and incrementing/decrementing counter for notifications.

6.4. Integrated Testing

In this phase, the behaviors of interaction chains among services regarding the most significant workflows were tested. These include testing the interaction chain between the authentication service and route restoration service, which should ensure that the persisted session state influences the choice of dashboard displayed on the application startup screen. Another interaction chain was tested to ensure it runs successfully from end to end: starting from when tourists book packages until when they get accepted and completed by advisors. The chain for submission of the blockchain review together

with the MongoDB booking status update was tested in order to verify if the changes happen in the proper order.

6.5. System Testing

System testing was performed as scenario flows beginning with the login of users into the application for the first time as either tourists or advisors. This process involved testing whether the user can successfully log in, navigate through all modules of the application, and persists its state in case of moving screens, as well as handling failures occurring in the external services successfully. Flow success and graceful failure handling of the external services were verified during the process.

6.6. Test Cases

6.6.1. Functional Test Cases

Table 6.1 contains all functional test cases for all twenty functional requirements. All tests were executed and passed successfully.

Table 6.1: Functional Test Suite Summary

TC ID	Scenario	Mapped FR	Expected Outcome	Result
TC-FR-01	Tourist registration with valid data	FR-01, FR-02	Account created; SHA-256 hash stored in DB	Pass
TC-FR-02	Advisor Individual registration flow	FR-01	Advisor role set; advisor dashboard route confirmed	Pass
TC-FR-03	Duplicate email registration attempt	FR-01	Registration rejected with descriptive error message	Pass
TC-FR-04	Login with valid credentials	FR-03, FR-04	Role-correct dashboard opened immediately	Pass
TC-FR-05	Session restore after application force-close	FR-03	App restores from SharedPreferences without re-login	Pass
TC-FR-06	Create trip with destination and date range	FR-05, FR-06	Trip inserted in DB; appears in trip list view	Pass
TC-FR-07	Generate and save AI trip plan	FR-07, FR-08	Multi-day plan created and persisted in saved_trip_plans	Pass

TC ID	Scenario	Mapped FR	Expected Outcome	Result
TC-FR-08	Landmark scan from gallery image with save	FR-09, FR-10	AI analysis stored with image path in saved_landmarks	Pass
TC-FR-09	Package browse with category filter applied	FR-11, FR-12	Filtered packages displayed; package details accessible	Pass
TC-FR-10	Tourist creates booking for selected package	FR-13	Pending booking record created in package_bookings	Pass
TC-FR-11	Tourist and advisor exchange chat messages	FR-14	Message visible to advisor; unread count indicator updates	Pass
TC-FR-12	Advisor creates and publishes travel package	FR-15	Package saved with Live status and visible to tourists	Pass
TC-FR-13	Advisor booking lifecycle state transitions	FR-16	Status: Pending → Accepted → Completed confirmed in DB	Pass
TC-FR-14	Review button visibility rule on bookings list	FR-17	Review action visible only on Completed-status bookings	Pass
TC-FR-15	Blockchain review success path execution	FR-17, FR-18	Tx hash displayed; reviewSubmitted set to true in MongoDB	Pass
TC-FR-16	Duplicate review UI prevention check	FR-18	Button replaced by Submitted indicator; dialog not opened	Pass
TC-FR-17	Advisor blockchain analytics screen load	FR-19	Review count and average rating correctly aggregated	Pass
TC-FR-18	Weather and news screens load live data	FR-20	Both modules return and display current data successfully	Pass
TC-FR-19	Local notification delivery confirmation	FR-21	Notification appears in device tray and app history	Pass
TC-FR-20	Privacy settings toggle persistence check	FR-22	Toggle states saved across application restarts	Pass

6.6.2. Blockchain Edge-Case Tests

Table 6.2 documents targeted edge-case tests for the blockchain review subsystem, covering failure conditions specific to the on-chain interaction model.

Table 6.2: Blockchain Edge-Case Test Results

TC ID	Condition Tested	Expected Behaviour	Result
TC-BC-01	Zero test MATIC in configured wallet	Transaction fails; user shown insufficient-funds message with faucet link	Pass
TC-BC-02	Contract address changed to invalid value	Review fetch fails gracefully; empty state displayed without crash	Pass
TC-BC-03	Second review submission attempt on reviewed booking	UI blocks attempt; reviewSubmitted flag prevents dialog from opening	Pass
TC-BC-04	Analytics load after review submission	On-chain review reflected in advisor analytics with correct rating and timestamp	Pass

6.6.3. Non-Functional Test Cases

Table 6.3 summarizes the results of non-functional and reliability testing according to the criteria established in Chapter 3.

Table 6.3: Non-Functional and Reliability Test Cases

TC ID	Category	Test Action	Evaluation Result
TC-NFR-01	Performance	Cold start on mid-range Android device	Acceptable launch time for demo-scale usage; services initialise in parallel
TC-NFR-02	Availability	Disable internet connection before app launch	Graceful error path displayed; application does not crash
TC-NFR-03	Data Integrity	Create booking then update status from advisor dashboard	All status transitions recorded correctly in MongoDB
TC-NFR-04	Auditability	Submit review and copy transaction hash from UI	Hash visible; PolygonScan explorer link opens correct transaction

TC ID	Category	Test Action	Evaluation Result
TC-NFR-05	Usability	Navigate between tourist module tabs and return	Consistent navigation state; no broken layouts observed
TC-NFR-06	Notifications	Toggle notification off; trigger update cycle	No notification delivered when toggle is disabled
TC-NFR-07	Security	Inspect users collection after registration	SHA-256 hash stored; no plaintext password present
TC-NFR-08	Known Limitations	Open DID and 2FA screens	UI simulation confirmed; no real wallet backend is called

6.7. Results & Evaluation

All of the twenty functional test cases passed their first execution without errors. All four blockchain edge cases returned the required failure handling behavior. Non-functional testing validated acceptable start-up speed, offline operation, data protection, and auditing features in the prototype.

The technical highlights were those concerning the use of the blockchain technology. The proper association of the booking state 'Completed' with the right to submit a review, proper updating of the MongoDB 'flag', and reading of on-chain reviews by the advisor into their analytics panel confirm that the proposed hybrid trust architecture is implemented. The acceptance testing checklist is completely fulfilled, as seen from Table 6.4.

Table 6.4: Acceptance Test Checklist

Acceptance Criterion	Verification Method	Status
Multi-role registration and login works for all three roles	Scenario walkthrough and MongoDB data inspection	Met
Tourist can generate and save an AI trip plan	Functional test TC-FR-07	Met
Advisor can complete full package lifecycle from creation to completion	Functional tests TC-FR-12, TC-FR-13	Met
Completed booking enables blockchain review with visible transaction hash	Functional test TC-FR-15 with PolygonScan verification	Met
Duplicate review attempts are structurally prevented at UI level	Functional test TC-FR-16	Met

Acceptance Criterion	Verification Method	Status
Advisor dashboard reflects blockchain review analytics from on-chain data	Functional test TC-FR-17	Met
Weather, news, and notification features function correctly	Functional tests TC-FR-18, TC-FR-19	Met

6.8. Threats to Validity

- Dependency on third-party services may influence the replicability of the test results because of factors related to connectivity, the validity of API keys, and the current status of the blockchain in the testnet environment.
- The behavior of Polygon Amoy testnet under production-level load may differ from actual costs and transaction processing time, as well as actual network traffic.
- Small-scale nature of an academic prototype does not account for real-life traffic; performance figures need to be tested under production-level traffic.
- The DID and 2FA authentication features are prototypes and do not confirm actual production-level security implementation..

6.9. Conclusion

Tests have shown that the prototype TripNexus application is fully working. All the functional requirements have been proven through the usage of test cases presented in Chapter 6. The issues that were not considered include the technical challenges such as managing secrets and other components of the production system such as identity providers and API gateways.

Chapter 7
Conclusion

This chapter offers a summary of the key contributions of TripNexus in terms of implementation of a solution not only highlighting technical feasibility but also demonstrating usability of an AI-driven mobile application for tourist trip planning which includes adviser-led commerce of services and reviewing mechanism based on cryptographic proofs of trust. Thus, the end product delivered by TripNexus is a fully-functioning, testable and evaluable Flutter app.

7.1. Contributions

The primary technical and academic contributions of TripNexus are as follows:

6. The development of a dual-role tourism workflow implemented in Flutter, including both the tourist journey and adviser journey in the same codebase and in the context of a single platform.
7. The design and implementation of a practical blockchain-based trust architecture involving recording of booking completion and reviews as immutable blockchain data in Polygon Amoy testnet and keeping booking information stored in MongoDB Atlas as off-chain data.
8. Efficient usage of Mistral AI's language and visual models for analysis of multi-day tourist routes and landmark images in order to improve the user experience.
9. A requirement-oriented software engineering process where all twenty-two functional requirements can be traced from the passing test cases.
10. The implementation of a deployable academic prototype along with a roadmap of necessary actions for reaching production grade status of the app regarding secrets management, authentic identities, backend API Gateway architecture and automated testing.

7.2. Reflections

- The implementation of dual-role workflow allows for full coverage of all processes in terms of in-app task execution without using third-party apps.
- The use of blockchain-based trust proves to be non-trivial enough to contribute significantly to the academic body and innovation in the area of students projects. In other words, this design demonstrates the knowledge of blockchain practices rather than mere theory.
- A modular service architecture helped with isolation of problematic areas and introducing changes in just one module. Hence, this design choice proved to be correct.
- The feature set of this app makes six Sustainable Development Goals achievable.

7.3. Future Work

The following extensions are suggested as the major lines of work towards production-grade TripNexus:

- **Payment Gateway:** Introduce a PCI compliant payment service provider to replace the cash-based booking flow with digital transactions.
- **Multilingual Support:** Localise the app into Urdu and any other suitable regional language to cater to the intended user audience better.

7.4. Final Remarks

It is demonstrated by TripNexus that a two-person student project working within a typical university timeline is capable of designing, implementing, testing, and validating a fairly complex mobile application, including AI services integration and blockchain infrastructure. In particular, the blockchain-based review subsystem may be considered as an example for other researchers and students in terms of hybrid trust models in mobile applications.

The project presented as an already developed, fully-functional, and ready-to-evaluate prototype was designed and developed with all possible engineering rigour and feasibility of practical deployment in mind.

References

[1] D. Buhalis and A. Amaranggana, "Smart Tourism Destinations Enhancing Tourism Experience Through Personalisation of Services," in Information and Communication Technologies in Tourism 2015, I. Tussyadiah and A. Inversini, Eds. Cham: Springer, 2015, pp. 377–389.

[2] UN World Tourism Organization (UNWTO), "Strategy on Innovation, Investment and Digital Transformation," UNWTO, Madrid, Spain, 2021. [Online]. Available: Digital Transformation | UN Tourism

[3] A. Chen, X. Ge, Z. Fu, Y. Xiao, and J. Chen, "TravelAgent: An AI Assistant for Personalized Travel Planning," *arXiv preprint arXiv:2409.08069*, 2024. [Online]. Available: https://arxiv.org/abs/2409.08069

[4] E. Arıbaş, "Transforming Personalized Travel Recommendations: Integrating Generative AI with Personality Models," *Electronics*, vol. 13, no. 23, p. 4751, 2024. [Online]. Available: https://doi.org/10.3390/electronics13234751

[5] Y.-C. Chen, K.-M. Yu, T.-H. Kao, and H.-L. Hsieh, "Deep learning based real-time tourist spots detection and recognition mechanism," *Science Progress*, vol. 104, no. 3, Sep. 2021. [Online]. Available: https://www.google.com/search?q=https://doi.org/10.1177/00368504211044228

[6] D. Martens and W. Maalej, "ReviewChain: Untampered Product Reviews on the Blockchain," in *Proc. 1st Int. Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, Gothenburg, Sweden, 2018, pp. 22–29. [Online]. Available: https://dl.acm.org/doi/10.1145/3194113.3194120

[7] Q. Wang and S. Chen, "Account Abstraction, Analysed," in *Proc. 2023 IEEE Int. Conf. on Blockchain (Blockchain)*, Hainan, China, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10411457/

[8] N. Kannengiesser, S. Lins, T. Dehling, and A. Sunyaev, "What Does Not Fit Can Be Made to Fit! Trade-Offs in Distributed Ledger Technology Designs," in Proc. 52nd Hawaii Int. Conf. on System Sciences (HICSS), Maui, HI, USA, 2019, pp. 1–10.

[9] Flutter Team, "Flutter: Build apps for any screen," Flutter Documentation, 2025. [Online]. Available: https://docs.flutter.dev/

[10] A. K. Sharma and R. Sharma, "Smart tourism in the digital age: overcoming barriers and unlocking new possibilities," *Revista de Gestão*, vol. 32, no. 3, pp. 224–237, 2025. [Online]. Available: https://doi.org/10.1108/REGE-02-2025-0030

[11] MongoDB Inc., "MongoDB Atlas: Fully Managed Cloud Database," 2024. [Online]. Available: https://www.mongodb.com/docs/atlas/

[12] Mistral AI, "Mistral AI API Reference and Documentation," 2024. [Online]. Available: https://docs.mistral.ai/

Appendices

Appendix A: Requirement Traceability Matrix

Table A.1 maps each functional and non-functional requirement to its corresponding executed test cases, providing complete requirement coverage evidence for evaluation purposes.

Table A.1: Requirement Traceability Matrix

Requirement ID	Mapped Test Cases
FR-01 to FR-04	TC-FR-01, TC-FR-02, TC-FR-03, TC-FR-04, TC-FR-05
FR-05 to FR-06	TC-FR-06
FR-07 to FR-08	TC-FR-07
FR-09 to FR-10	TC-FR-08
FR-11 to FR-12	TC-FR-09
FR-13	TC-FR-10
FR-14	TC-FR-11
FR-15	TC-FR-12
FR-16	TC-FR-13
FR-17	TC-FR-14, TC-FR-15
FR-18	TC-FR-16, TC-BC-03
FR-19	TC-FR-17, TC-BC-04
FR-20	TC-FR-18
FR-21	TC-FR-19
FR-22	TC-FR-20
NFR-01 to NFR-08	TC-NFR-01 through TC-NFR-08

Appendix B: Deployment and Demo Checklist

The following checklist should be completed before any live demonstration or formal evaluation session:

11. Verify MongoDB Atlas connection string is correctly configured and that network access is permitted from the deployment environment.
12. Confirm Mistral API key is active and has sufficient request quota for the demonstration session.
13. Confirm Open-Meteo network reachability for the weather module from the target device.
14. Grant notification permissions on the target device for the weather and news alert features.
15. Verify the Polygon Amoy wallet has sufficient test MATIC balance for at least one review submission. Obtain test MATIC from the Polygon Amoy faucet if the balance is insufficient.
16. Execute the minimum acceptance test set: register and login both roles; create one trip; generate and save one AI plan; scan and save one landmark; create one live package; complete one full booking lifecycle through Pending to Completed; submit one blockchain review and record the transaction hash; confirm advisor analytics reflect the submitted review.

TripNexus_FYP finalfile.docx

Document

turnitin

Document Details

Submission ID
trn:oid:::1:1528927896

Submission Date
Apr 20, 2026, 9:50 AM

Download Date
Apr 20, 2026, 9:52 AM

File Name
TripNexus_FYP finalfile.docx

File Size
9.91 MB

55 Pages

13,737 Words

89,291 Characters

turnitin Page 2 of 87 · Integrity Overview Submission ID trn:oid:::1:1528927896

4% Overall Similarity

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

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

- **18** Not Cited or Quoted **3%**
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations **0%**
  Matches that are still very similar to source material
- **0** 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

11% Internet sources
3% Publications

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

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.

turnitin Page 2 of 86 - AI Writing Overview Submission ID 1:9649226641

*% 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.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

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.