Analysis Report

1. Introduction

This report presents a detailed analysis of the travel planning (Roadrunner) project. It formalizes the system specifications derived from the initial requirements elicitation and serves as a comprehensive agreement between the development team and the stakeholders.

The purpose of this report is to produce a model of the system that is correct, complete, consistent, and verifiable. It addresses all relevant issues related to the project, including a thorough examination of boundary conditions, exceptional cases, and specific functional needs. By clarifying the system specifications and resolving any ambiguities found during the analysis phase, this report establishes the foundation for the subsequent design and implementation phases.

2. Current system

Currently, as far as our research goes, no single unified system exists for the proposed solution. The current system refers to the manual, fragmented process of modern travel planning. Users are forced to navigate a distributed ecosystem of unconnected websites and applications to research destinations, compare transport and accommodation options, and organize daily activities.

This traditional approach has several flaws, listed below:

- Users sift through massive amounts of irrelevant information.
- Data is distributed across blogs, booking sites, and map applications.
- Most web content is filled with ads and sponsored content that are not useful to the users.
- Existing tools offer generic filters (price, rating) but fail to understand nuanced preferences like "quiet cafes" or "historical battlefields."

3. Proposed system

The proposed system is an AI-powered travel assistant that centralizes and personalizes the entire planning lifecycle. By leveraging Large Language Models (LLMs) and Data Mining, the system allows users to input a natural language "User Story" (i.e.., "I want a quiet 3-day trip to Rome focused on art and food").

The system generates optimized, end-to-end itineraries including daily schedules. It integrates third-party APIs like Wikipedia, Google Places (New), OpenStreetMap to provide data and acts as a smart intermediary, transforming travel planning from a stressful chore into an effortless, engaging experience.

3.1 Overview

The software functions as a personalization layer and aggregator sitting on top of the existing travel data ecosystem.

- The product acts as a central hub, connecting users to data from various APIs (e.g. Wikipedia, Wikivoyage and Google Places).
- It converts unstructured user inputs into structured travel plans. It features a dynamic user profiling system that learns from past interactions to refine future recommendations.
- The interaction model is hybrid, offering both a conversational interface (Chatbot) for intent gathering and a graphical interface (Interactive Map) for route visualization and modification.

3.2 Functional Requirements

- FR-001: The system shall accept and parse a user's travel preferences, constraints, and interests provided in natural language text ("User Story").
- FR-002: The system shall automatically generate a complete travel plan, including an optimized sequence of POIs and a daily schedule.
- FR-003: The system shall create and maintain a persistent user profile, storing explicit preferences and implicit data (past trips, feedback).
- FR-004: The system shall display the generated plan on an interactive map, clearly marking routes and allowing POI interaction.
- FR-005: The system shall provide a drag-and-drop interface for users to manually edit their plan (add/remove/reorder POIs).
- FR-006: The system shall retrieve and display real-time POI information (opening hours, ratings, descriptions) upon request.
- FR-007: The system shall allow users to provide explicit feedback (ratings/likes) on plans to update their profile weights.
- FR-008: The system shall provide "Explainable AI" justifications for recommendations (e.g., "Recommended because you like Architecture").
- FR-009: The system shall provide an interface for users to request complete data deletion (GDPR compliance).
- FR-010: The system shall identify and label activities with age ranges for family-friendly filtering.
- FR-011: The system shall provide a quick total price guess for the suggested trip.

3.3 Nonfunctional Requirements

- Initial plan generation must complete within 30 seconds.
- UI interactions (drag-and-drop, clicks) must provide visual feedback within 500ms at most.
- The system must support at least 100 concurrent users without performance degradation.
- External API calls must timeout after 8 seconds to prevent system hangs.
- Target uptime of 99.5%.
- The system must degrade gracefully (e.g., show cached data) if third-party APIs fail.

- Mean Time To Repair (MTTR) should be less than 4 hours, supported by containerized deployment and backups.
- Data must be encrypted in transit (HTTPS/TLS) and at rest (Database encryption).
- User passwords must be hashed using Argon2 or scrypt with salting.
- API keys must be managed via a secret management system, not hardcoded.

3.4 Pseudo requirements

- **Implementation Language:** Python or Spring Boot (Backend), React/JavaScript (Frontend).
- **Platform:** Online-only web application (no offline mode) with mobile layout support (responsive UI)..
- **Compliance:** Must adhere to GDPR and KVKK regulations regarding data privacy and the Right to be Forgotten.
- **Data Dependencies:** Functionality is strictly bound by the limits of the free/tier-based APIs used (Google Places, OSM, Wikivoyage) and publicly available data.
- **Browser Support:** Must function on Chromium-based browsers, Firefox, and Safari.

3.5 System models

3.5.1 Scenarios

- Example Scenario 1: User enters "I have a 2-day business trip to London. I have 4 hours free on Saturday afternoon and love modern art." The system parses "London," "2-day" (context), "Saturday afternoon" (time constraint), and "Modern Art" (interest). It returns a route visiting the Tate Modern and a nearby highly-rated coffee shop.
- Example Scenario 2: User enters "Planning a weekend in Ankara with two kids, need parks and kid-friendly museums." The system filters for POIs tagged with appropriate age ranges and low physical difficulty, avoiding crowded nightlife spots.

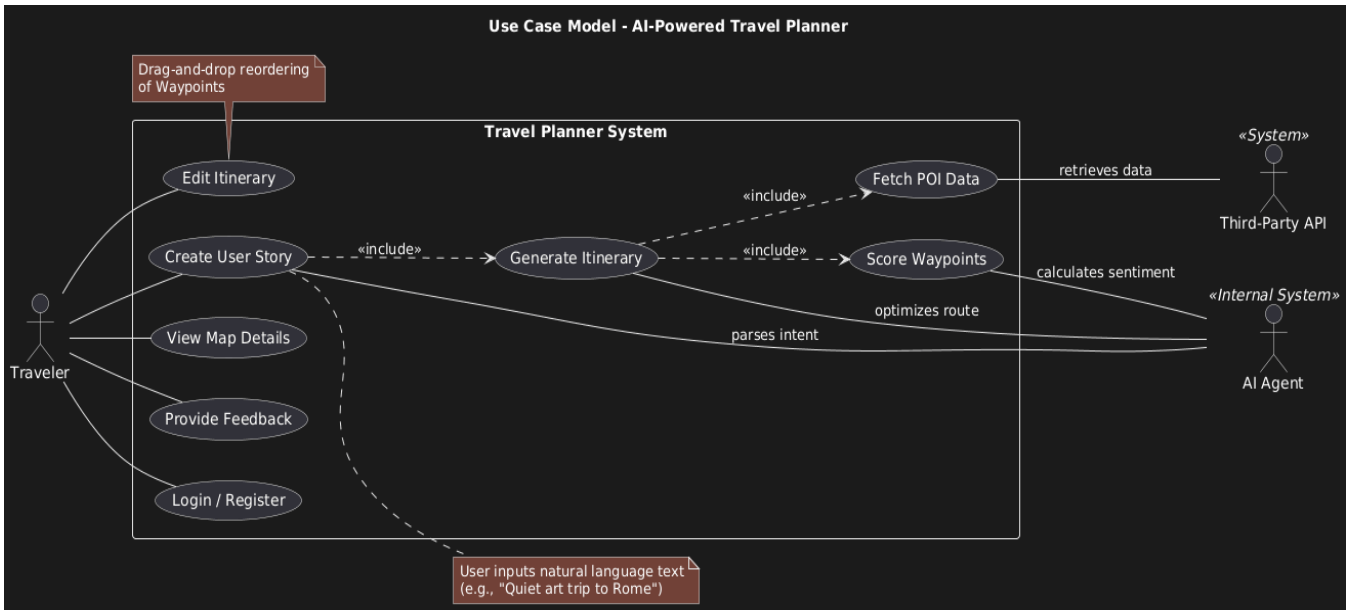
3.5.2 Use case model

Actors:

- Traveler: The primary user requesting itineraries.
- System (AI Agent): The internal actor processing logic.
- Third-Party API: External actor providing data.

Primary Use Cases:

1. Create User Story: User chats with LLM to define trip parameters.
2. Generate Itinerary: System processes story and produces a route.
3. Edit Itinerary: User modifies the generated route via GUI.
4. View Map Details: User clicks POIs for metadata.
5. Provide Feedback: User rates the trip to refine the profile.



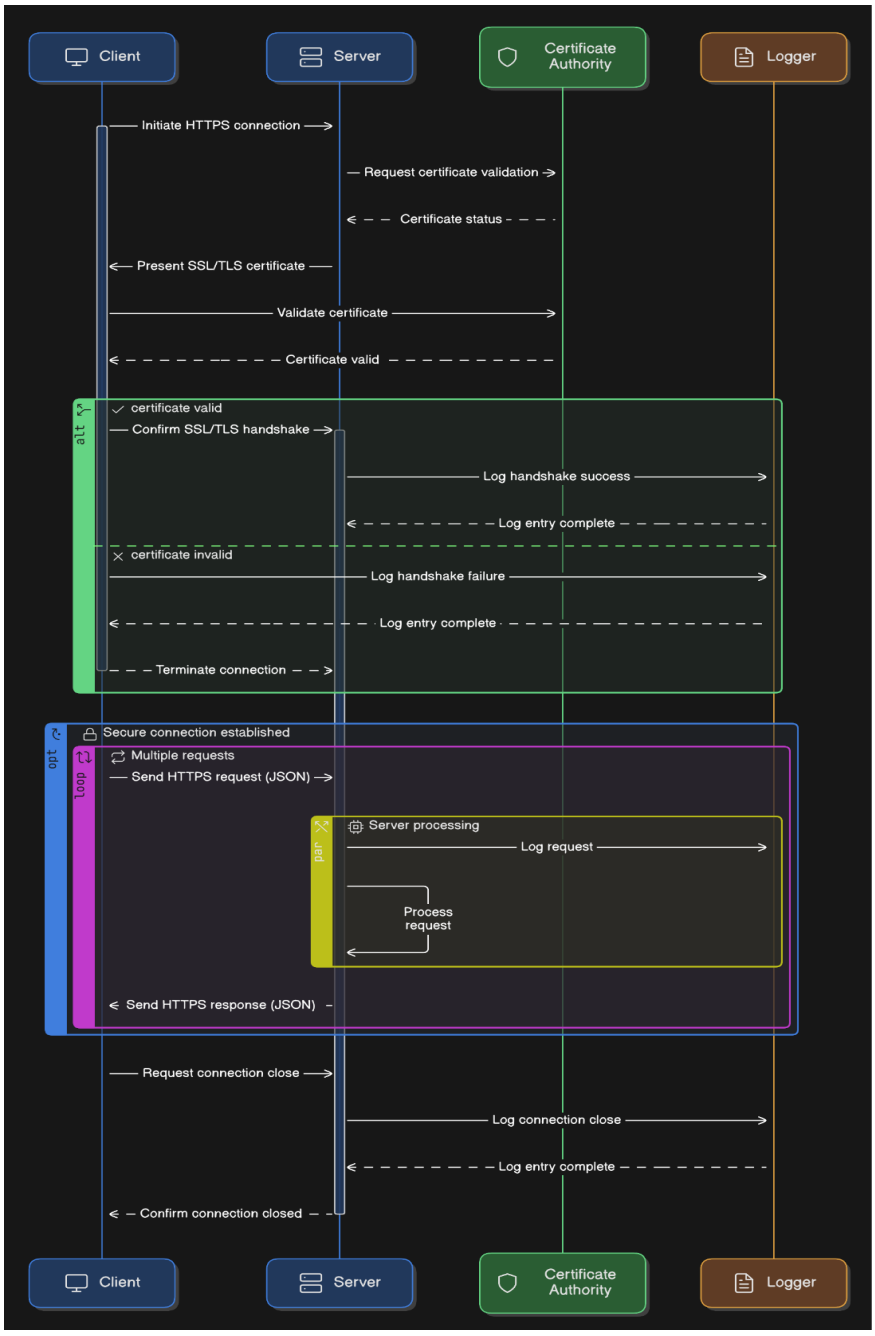
3.5.3 Object and class model

- User: Contains Profile, Credentials, History.
- UserStory: Contains the raw text input and extracted Tags (Constraints, Interests).
- Route: A collection of Waypoints ordered by time/distance. Includes KPIs (Total Distance, Est. Cost).
- Waypoint (POI): Represents a physical location. Example attributes are: Coordinates, Category, Rating, OpeningHours.

3.5.4 Dynamic models

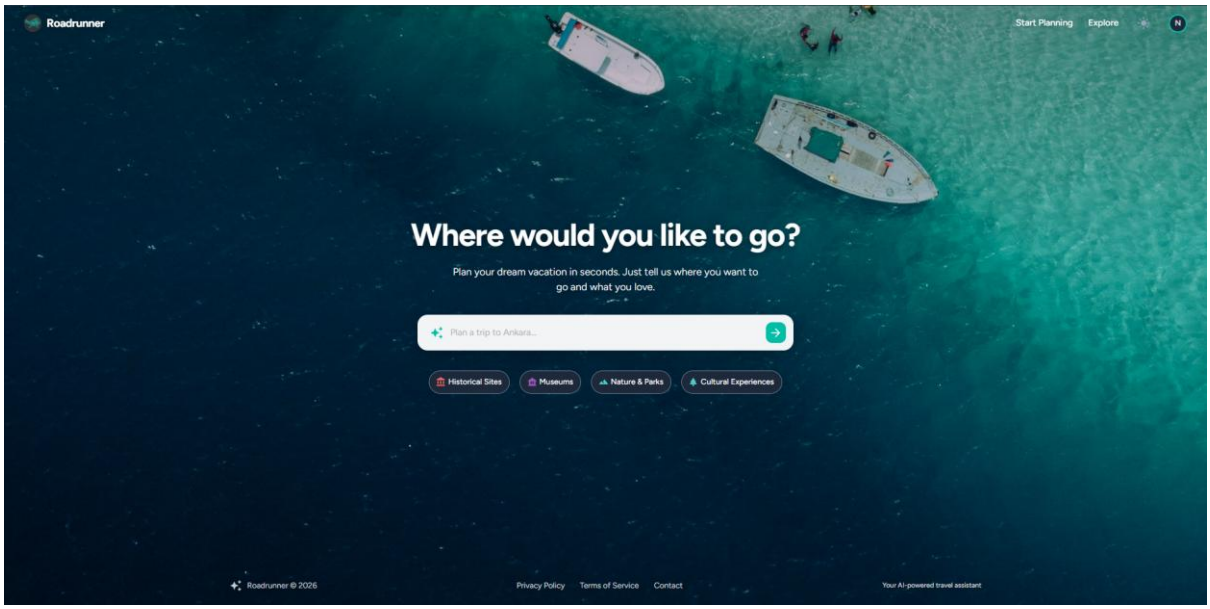
The dynamic behavior follows a pipeline approach:

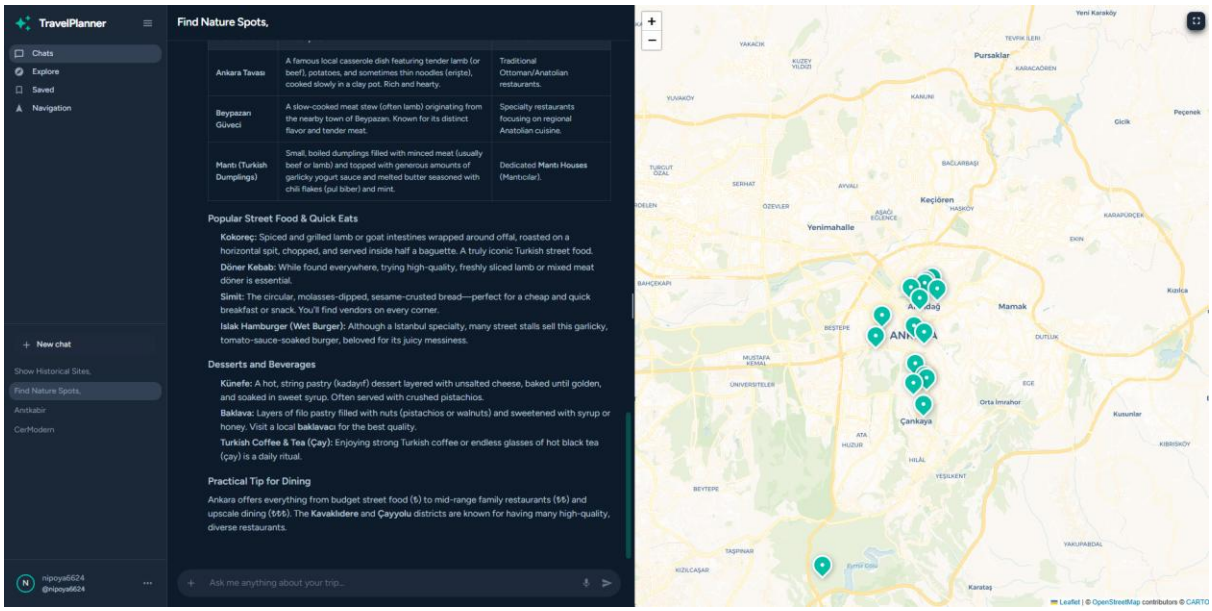
1. User submits text.
2. System extracts entities (Location, Date) and Intent Labels (Budget, Vibe).
3. System queries APIs for POIs within the target Vertex (radius).
4. Waypoints are scored based on User Profile + Blogger Data sentiment.
5. JSON data is sent to the frontend for rendering.



3.5.5 User interface - navigational paths and screen mock-ups

- Landing Page -> Login/Register -> Dashboard (Chat/Map View).
- Dashboard -> Generate New Trip -> Modify Trip (if requested) -> View Route/Navigation -> Save/Share.





4. Glossary

- Waypoint: A specific Point of Interest (POI) such as a museum, restaurant, or park.
- Vertex: A defined geographical area (circle) within which the travel is planned.
- User Story: The natural language input explaining the user's ideal travel preferences and constraints.

- Blogger Data: Data scraped from travel blogs used to generate "vibe" labels and sentiment scores for locations.
- XAI (Explainable AI): A feature that presents the "why" behind specific AI recommendations.
- RAG (Retrieval-Augmented Generation): A technique enhancing LLM responses with external proprietary data.
- GDPR: General Data Protection Regulation (EU privacy law).
- KVKK: Kişisel Verilerin Korunması Kanunu (Turkish Data Protection Law).

5. References

- Google AI for Developers, 2025. https://ai.google.dev/gemini-api/docs/
- Google Places API (New), 2025. https://developers.google.com/maps/documentation/places/web-service/op-overview
- S. Heyer, “RAG API,” Medium, Sep. 13, 2024.
- “Wikivoyage,” Wikivoyage.org, 2025.
- “Map Features - OpenStreetMap Wiki,” wiki.openstreetmap.org.
- İbrahim Sarıçiçek, “Foursquare Verileri ile Ankara Restoranları Mekansal Analizleri,” Medium, Nov. 29, 2021.
- C.-Y. Tsai and J.-H. Wang, “A Personalized Itinerary Recommender System: Considering Sequential Pattern Mining,” Electronics, 2025.
- A. Chen et al., “TravelAgent: An AI Assistant for Personalized Travel Planning,” arXiv preprint, 2024.