System Analysis & Design – Project: Waygo

1. Introduction

Waygo is a smart travel planning application designed to help users organize their trips efficiently and intuitively. It centralizes all travel-related aspects such as itinerary planning, budgeting, destination discovery, and activity suggestions. All these features into one user-friendly platform.

2. Objectives

- Simplify trip organization by gathering key features in one app.

- Provide personalized recommendations for activities and destinations.

- Offer a visual and interactive experience through maps and itineraries.

- Allow budget tracking and travel sharing among users.

- Practice software development management and teamwork.

3. System Analysis

3.1. Actors

Actor	Description
User	Main actor who uses the system to plan and manage trips.
External API	Provides data for maps.

3.2. Use Cases

Main Use Cases for the User:
1) Authenticate (Register / Login)
2) Manage Profile
3) Create Trip
4) Search Destination
5) Plan Itinerary
6) Manage Budget
7) Add to Favorites
8) Get Activity Suggestions
9) View Map
10) Receive Notifications
11) Share Trip

12) View Travel History
13) Give Feedback

4. System Design

4.1. Class Diagram Overview

Class	Key Attributes	Key Methods	Relations
User	userID, name, email, password, preferences	login(), register(), updateProfile()	→ Trip
Trip	tripID, title, startDate, endDate, destination, budget	createTrip(), editTrip(), deleteTrip()	→ Destination → Activity → Budget
Destination	destinationID, name, country, description, coordinates	searchDestination(), viewMap()	→ Trip

Activity	activityID, name, type, cost, rating	addActivity(), removeActivity()	→ Trip
Budget	total, expensesList	addExpense(), getRemainingBudget()	→ Trip
Notification	notifID, message, date, status	sendNotification(), markAsRead()	→ User
APIService	apiKey, endpoint	getMapData(), getWeather(), getSuggestions()	Used by Trip : Destination, Activity

4.2. Relationships

- Association: between User and Trip (1 user → several trips)
- Composition: Trip owns its Budget and Activities (deleted together)
- Dependency: APIService provides external data to multiple classes
- Aggregation: User aggregates Notifications independently of Trips

4.3. Design Summary

The design follows a modular and object-oriented approach, making the system:
- Scalable (easy to add features like transport or accommodation modules)
- Maintainable (clear separation of responsibilities)
- Collaborative (each feature can be handled by a different teammate on GitHub)

5. Future Improvements

- Integration of AI for smart trip recommendations.

- Synchronization with external booking systems (flights, hotels).

- Multi-language and offline support.

- Addition of a social sharing feed (community of travelers).

6. GitHub & Project Management

Each team member :
- Manage a feature branch for their module (e.g., trip-module, budget-module)
- Contribute progressively with regular commits
- Use GitHub Issues for task tracking
- Provide a README.md with setup instructions and team roles