Project Specification: Group Travel Planner

Course: CS-UY 4513 - Software Engineering

Professor: Dr. DePasquale

Project Title: Group Travel Planner

Date: 09/26/2025

1.0 Project Overview

The goal of this project is to design and develop a multi-module Software-as-a-Service (SaaS) system using Ruby on Rails. The system will allow for organization and listing of travel plans, with the ability to choose destinations, flights, hotels, and the ability to list them for other users to join. This primarily acts as a social platform for users to plan recreational trips together publicly or privately.

2.0 Core Requirements

The project must adhere to the mandatory characteristics outlined in the project specification template document.

2.1 User-Based System

The application must be a user-based system, requiring a secure user authentication and authorization framework. User accounts will be managed via standard email/password registration and login

2.2 User Roles

The system will support three distinct user roles, each with specific permissions:

- **Users:** Joins group to attend trip, cannot change itinerary
- **Organizer:** Chooses trip details and can manage users within a trip
- **Admin:** Full control and management of events made by organizers, ability to ban users.

2.3 Persistent Storage

The application must use a relational database to persist all data.

Proposed Database Schema :

- Users(**UserID**, FirstName, LastName, Username (unique), Password, Age, Gender)
- Roles(**RoleID**, RoleName, RolePermissions)
- UserRoles(**UserRoleID**, UserID, RoleID)
- ItineraryGroup(**ItineraryGroupID**, GroupName, Date, Location, IsPrivate (boolean), OrganizerID)

- Hotels(**HotelID**, Name, Location, Rating, Cost, ArrivalTime, DepartureTime)
- Flights(**FlightID**, FlightNumber, DepartureLocation, ArrivalLocation, DepartureTime, ArrivalTime, Cost)
- Messages(**MessageID**, UserID, ItineraryGroupID, Text, Time, IsRead(boolean))
- Itinerary_Attendees(UserID, ItineraryGroupID)
- Itinerary_Hotels(HotelID, ItineraryGroupID)
- Itinerary_Flights(FlightID, ItineraryGroupID)

2.4 Modular Architecture

The system must be logically divided into a minimum of 3 and a maximum of 5 modules. These modules must demonstrate clear dependencies and rely on each other for full functionality.

Proposed Modules:

1. **User Management and Authentication:**
    - TManages user registration, login, and profile information
    - Defines and assigns user roles (Users, Organizers, Admin)
    - **Dependency**: Core dependency for all other modules
2. **Itinerary Management:**
    - Allows organizers to create and edit itineraries
    - Allows users to join itineraries
    - **Dependency:** User management and Authentication
3. **Search / Filtering Management**
    - Allows users to search for itineraries using keywords & other filters (date, party size, public or private(password required))
    - **Dependency:** Itinerary Management
4. **Communication / Notification Management:**
    - Logs messages between members of itineraries
    - Tracks messages between sender and receivers
    - Stores messages for notifying members of itineraries
    - **Dependency:** Itinerary Management

2.5 API Interfaces

Each module must expose a RESTful API interface. This is crucial for enabling the modules to communicate with each other and for potential future integrations. The API should follow standard REST conventions.

Example Endpoints:

- **User & Identity Management API:**
    - POST /api/user/register
    - POST /api/user/login
    - GET /api/user/<id>
    - GET /api/user/<id>/roles

    - POST /api/user/<id>/roles/<role_id>
- **Itinerary Management API:**
    - GET /api/itineraries/
    - GET /api/itineraries/create
    - POST /api/itineraries/create
    - POST /api/itineraries/update/<id>&<user_id>
    - POST /api/itineraries/update/<id>&<user_id>
- **Search / Filtering API:**
    - GET /api/itineraries/search
    - POST /api/itineraries/search?<keyword_args>
- **Communication/Notification Management API**
    - GET /api/messages/<user_id>&<group_id>
    - POST /api/messages/<user_id>&<group_id>
    - POST /api/messages/<user_id>&<group_id>
    - GET api/notifications/<user_id>&<new/all>

3.0 Technical Stack

- **Language:** Ruby
- **Framework:** Ruby on Rails
- **Database:** SQLite
- **Testing:** RSpec

4.0 Deliverables

The final submission must include:

1. **Complete Source Code:** A well-structured and commented codebase, including all necessary configuration files.
2. **Project Documentation:** A `README.md` file that explains the system's architecture, setup instructions, and how to run the application.
3. **API Documentation:** A separate document or file detailing all API endpoints, their expected parameters, and response formats.
4. **Presentation:** A brief presentation and demonstration of the application's core features.