VIHARA - TOUR PLANNER

Software Requirement Specification

Submitted By:

Aditya Shriwal 22BCE2132
Archit Yadav 22BCE2146
Aman Chauhan 22BCE0476

Table of Contents

1 INTRODUCTION

VIHARA is a tool for planning your holidays and tour through online by the Customer. It provides the proper management tools and easy access to the customer information.

1.1 Purpose

The purpose of "Vihara" is to provide an all-in-one web-based solution for travellers to plan their trips effortlessly. It will include tourist destination information, optimized travel routes, budget-friendly hotel recommendations, booking options, and personalized itineraries.

This SRS for Tour Planner can also be used for future as basis for detailed understanding on how project was started. It provides a blueprint to upcoming new developers and maintenance teams to assist in maintaining and modifying this project as per required changeability.

1.2 Scope of the Project

"Vihara" focuses on providing:

1. Information about popular and offbeat tourist places.
2. Route optimization based on distance, time, and cost.
3. Hotel and accommodation recommendations with booking options.
4. Integration of travel modes (flights, trains, cabs).
5. Cost estimation for the entire trip.
6. Weather based plan change

7. Suggestions for nearby restaurants and shopping areas.
8. Personalized itineraries based on user preferences.
9. Emergency SOS help provinces.

1.3 Definitions, Acronyms and abbreviations

SRS	Software Requirement Specifications
TP	Tour Planner
DBMS	Database Management System
Blueprint	A design technical plan
AI	Artificial Intelligence
HTTP/HTTPS	Hyper Text Transfer Protocol/Secure
API	Application Programming Interface
UI/UX	User Interface/ User Experience
FR	Functional Requirement
NFR	Non-Functional Requirement

1.4 References

[1] Travel APIs Documentation (e.g., Skyscanner API, Booking.com API).

[2] Google Reviews.

[3] ER Diagram Tutorial: https://www.tutorialspoint.com/dbms/er_diagram_representation.htm

[4] Requirement Engineering:http://morse.inf.unideb.hu/valseg/gybitt/07/ch02.html

[5] Data Flow Diagram: http://myyee.tripod.com/cs457/dfd.htm

[6] Requirement Engineering: https://en.wikipedia.org/wiki/Requirements_engineering

1.5 Overview

The remaining sections of this documentations describes the overall descriptions which includes product perspective and functions, characteristics of users. It also consists of Assumptions, and Constraints. Overall description is listed in section 2. Section 3 includes Specific Requirements which consists of Functional and Non-functional requirements, External Interface Requirements, Software System Attributes, Performance Requirements, Capacity Requirements, Availability Requirements, Safety Requirements and Requirement Traceability Matrix.

2 OVERALL DESCRIPTION

2.1 Product Perspective

Vihara is an innovative travel planning solution, leveraging AI/ML algorithms for optimized routes and cost-efficient planning. The product integrates multiple APIs for real-time data retrieval, ensuring seamless user experience.

2.2 Product Functions

Our Product General functions are:

1. **Tourist Place Discovery**: Displays a curated list of destinations for selected cities or regions.
2. **Route Optimization**: Finds the most time and cost-efficient travel route.
3. **Hotel Booking**: Recommends accommodations based on user preferences (budget, ratings, location).
4. **Itinerary Planner**: Generates personalized travel plans.
5. **Cost Estimation**: Provides an estimated budget for the trip.

2.3 User Characteristics

1. Travelers of all ages and interests.
2. Friendly with Solo, Family, Friend.
3. Individuals seeking budget-friendly and customized travel experiences.

2.4 Constraints

1. Internet connectivity is required for full functionality.
2. Integration with external APIs (e.g., travel and hotel booking services).
3. Budget limitations for API usage and hosting services.

2.5 Assumption and Dependencies

1. Users provide accurate input regarding preferences.
2. APIs for hotel and travel services are reliable and provide updated data.
3. The application is used on modern browsers with JavaScript enabled.

If incase of any difficulties, SRS should be flexible enough to change accordingly.

3 Specific Requirement

3.1. External Interface Requirements

These define how the system will interact with external components such as APIs, users, and third-party services.

3.1.1. User Interface (UI) Requirements:

- The website must be intuitive and user-friendly, allowing users to perform key functions such as searching, booking, and payments easily.
- **Responsive Design**: The UI must adapt seamlessly to various screen sizes (mobile, tablet, desktop).
- **Accessibility**: The website must comply with WCAG 2.0 standards, ensuring that users with disabilities can access the website. This includes screen reader support, high-contrast modes, and keyboard navigation.
- **Login/Register Pages**: Secure login and registration forms with clear input validation for email, password strength, etc.

3.1.2. Third-Party API Integration Requirements:

- **Weather API (e.g., Gemini API)**: The system must fetch real-time weather data for specific destinations and modify travel plans accordingly.
- **Transport Services API**: Integration with flight, train, or car rental services for real-time transport information.
- **Map Integration (Google Maps API)**: To show the user their current location, hotel locations, nearby emergency services, and other points of interest.

3.2. User Requirements

These specify how different types of users will interact with the system, what roles they will have, and what functionalities are required.

3.2.1. User Roles and Access:

- **Customer (Family, Solo, Friend, Couple):**
    - Can search for destinations and hotels based on preferences.
    - Can book hotels and transport options.
    - Receives weather-based travel recommendations.

- Can use emergency SOS feature for assistance.

3.3. Software Requirements

This outlines the software technologies and services necessary to build and run the travel tour website.

3.3.1. Frontend Software Requirements:

- **Framework**: Angular (for UI/UX, responsive components).
- **HTML/CSS/JavaScript**: For creating responsive and interactive web pages.
- **FontAwesome**: For adding icons and user-friendly design elements.
- **API Integration Libraries**: Axios or Fetch API for interacting with third-party APIs like weather, payment, and transport.

3.3.2. Backend Software Requirements:

- **API Development**: Go (for API services to handle user data, bookings, and communication with third-party APIs).
- **Database**: MongoDB or PostgreSQL (to store user information, bookings, hotels, destinations, etc.).
- **Authentication & Authorization**: JWT for secure user authentication and role-based access control.
- **Weather-Based Travel Plan Logic**: Python/TensorFlow (optional) for recommendation systems based on weather, budget, and user preferences.

3.4. Communication Requirements

This section outlines how data is transmitted and secured between components and external entities.

3.4.1. Client-Server Communication:

- **Protocol**: HTTPS (Secure HTTP) for all communications to ensure encryption of user data, especially during login, booking, and payment.
- **Real-Time Data Transfer**: Use of WebSockets or RESTful APIs to fetch real-time weather updates and transport information.

3.4.2. Third-Party API Communication:

- **Weather API**: Secure API key-based authentication for accessing weather data (e.g., using the Gemini API).

3.4.3 Internal System Communication:

- **Backend to Database**: Secure connection between backend (Go services) and the database (MongoDB/PostgreSQL) using SSL/TLS to prevent data breaches.
- **Frontend to Backend**: All communications between the frontend (Angular) and backend (APIs) should be over secure channels using HTTPS.

4. Functional Requirements

These define the primary operations and processes that your system will perform. Each functional requirement corresponds to user tasks, features, and system behaviors.

4.1. User Registration and Login:

- **FR1.1**: The system must allow users to register by providing their email, password, name, and other optional details (e.g., phone number).
- **FR1.2**: The system must send a confirmation email for user account verification.
- **FR1.3**: Users must be able to log in using their registered email and password.
- **FR1.4**: The system must provide password recovery functionality for users who forget their passwords.
- **FR1.5**: User authentication must be secure using JWT tokens to allow authorized access.

4.2. Search for Destinations and Hotels:

- **FR2.1**: Users must be able to search for destinations by providing input such as location, travel dates, and preferences.
- **FR2.2**: The system should display available hotels and tour packages based on user preferences and real-time data.
- **FR2.3**: The system should allow filtering based on price range, type of accommodation, and amenities.
- **FR2.4**: Weather data (integrated through an API) should be factored into search results, recommending alternate travel plans if needed due to adverse weather conditions.

4.3. Weather-Based Travel Recommendations:

- **FR3.1**: The system should fetch real-time weather data from the weather API (e.g., Gemini API) based on the destination.

- **FR3.2**: If severe weather conditions are detected, the system should recommend alternative travel dates or nearby destinations.
- **FR3.3**: Users must be able to access detailed weather forecasts as part of their travel plan.

4.4. Booking:

- **FR4.1**: Users must be able to book hotels, transport, and tour packages through the website.

4.5. User Profile Management:

- **FR5.1**: Users should be able to update their personal information, including name, email, phone number, and password.
- **FR5.2**: Users must have access to their booking history and the ability to cancel or modify future bookings.

5. Non-Functional Requirements

These specify the quality attributes of the system and how the functional requirements will be achieved in terms of **safety**, **security**, **performance**, **capacity**, **availability**, and more.

5.1. Safety Requirements:

- **NFR1.1**: The system must ensure that user data and transactions are handled securely, protecting sensitive information (e.g., payment details) from leaks or breaches.
- **NFR1.2**: The SOS feature must function reliably in emergency situations, ensuring that user location and distress signals are transmitted accurately. We are giving only recommendation.

5.2. Security Requirements:

- **NFR2.1**: All user data must be encrypted during transmission using HTTPS/SSL, especially during login, booking, and payment processes.
- **NFR2.2**: Sensitive user data (e.g., passwords, payment details) must be securely stored using encryption algorithms (e.g., AES-256).
- **NFR2.3**: The system must prevent unauthorized access through the use of secure authentication methods, such as JWT-based tokens.

5.3. Performance Requirements:

- **NFR3.1**: The website should load within **3 seconds** for users on a standard broadband connection.
- **NFR3.2**: The backend API should respond to user requests (e.g., searching for hotels or processing payments) within **2 seconds** on average.
- **NFR3.3**: The system must support **real-time weather updates** and fetch weather data in less than **1 second** for dynamic travel recommendations.

5.4. Capacity Requirements:

- **NFR4.1**: The database must support up to **1 thousand user profiles** and their associated booking history.
- **NFR4.2**: The system should handle up to **1,000 hotel and destination listings** across various regions.

5.5. Software Attributes Requirements:

- **NFR5.1**: The system must be **scalable**, allowing it to expand to accommodate more users, destinations, or features as the business grows.
- **NFR5.2**: The system should be built on a **modular architecture**, ensuring that new features (e.g., additional APIs or services) can be added without major disruptions.
- **NFR5.3**: The platform should be **compatible across browsers** (e.g., Chrome, Firefox, Safari) and devices (e.g., desktops, tablets, smartphones).
- **NFR5.4**: The codebase must be **maintainable**, following best practices for clean, documented code, to allow future developers to easily update or troubleshoot the system.

5.6. Availability Requirements:

- **NFR6.1**: The system must be **available 24/7** to accommodate users across different time zones.

6.Requirement Traceability Matrix

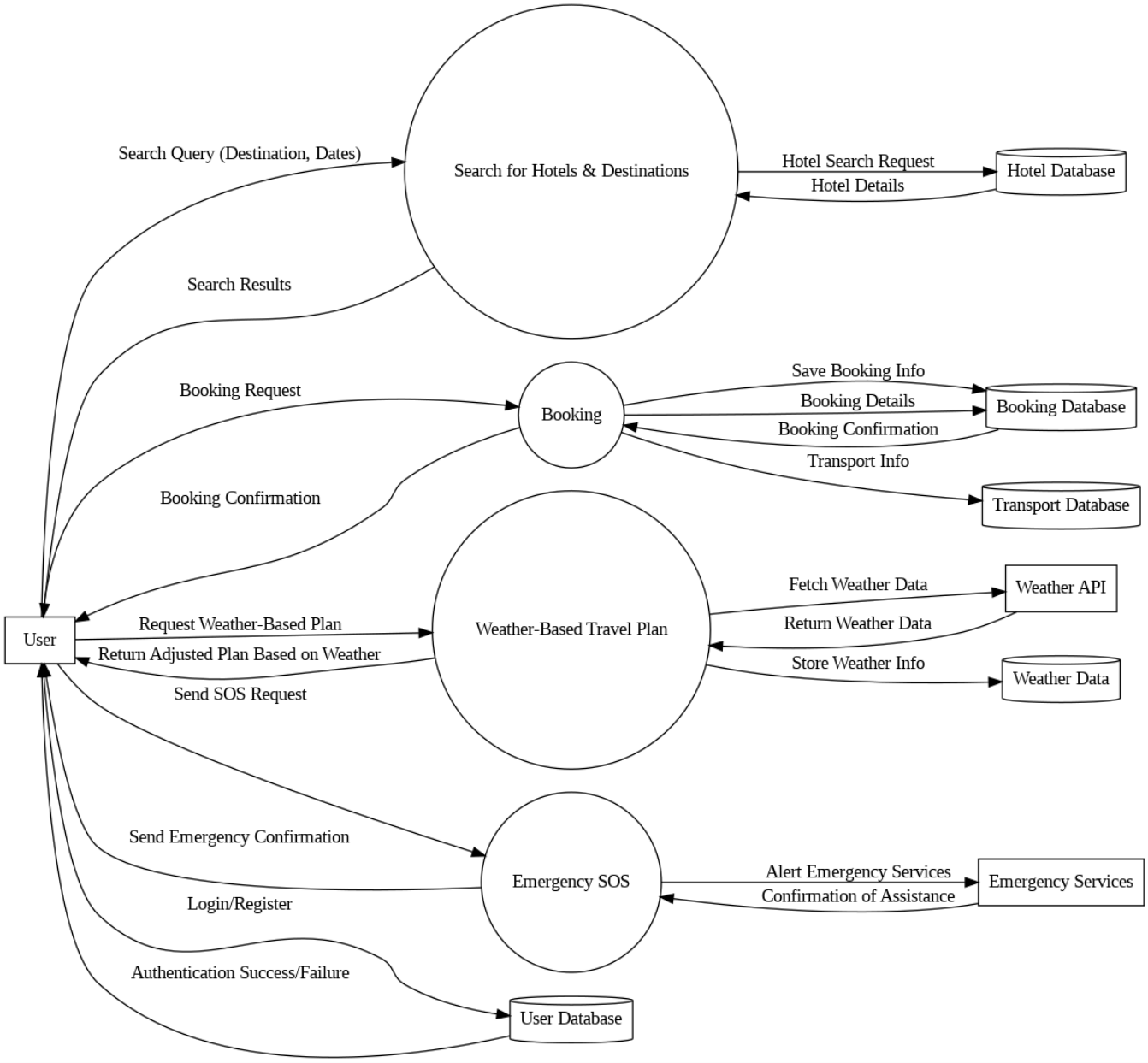
The Requirement Traceability Matrix (RTM) reflects the correlation between Non-Functional Requirements (NFR) and Functional Requirements (FR). The RTM is a documentation that associates the requirements entirely throughout the validation process. Traceability is regarded to be one of the most important considerations for tracing the requirements.

In the table below we will be tracing the relation between Functional Requirements and Non-Functional Requirements.

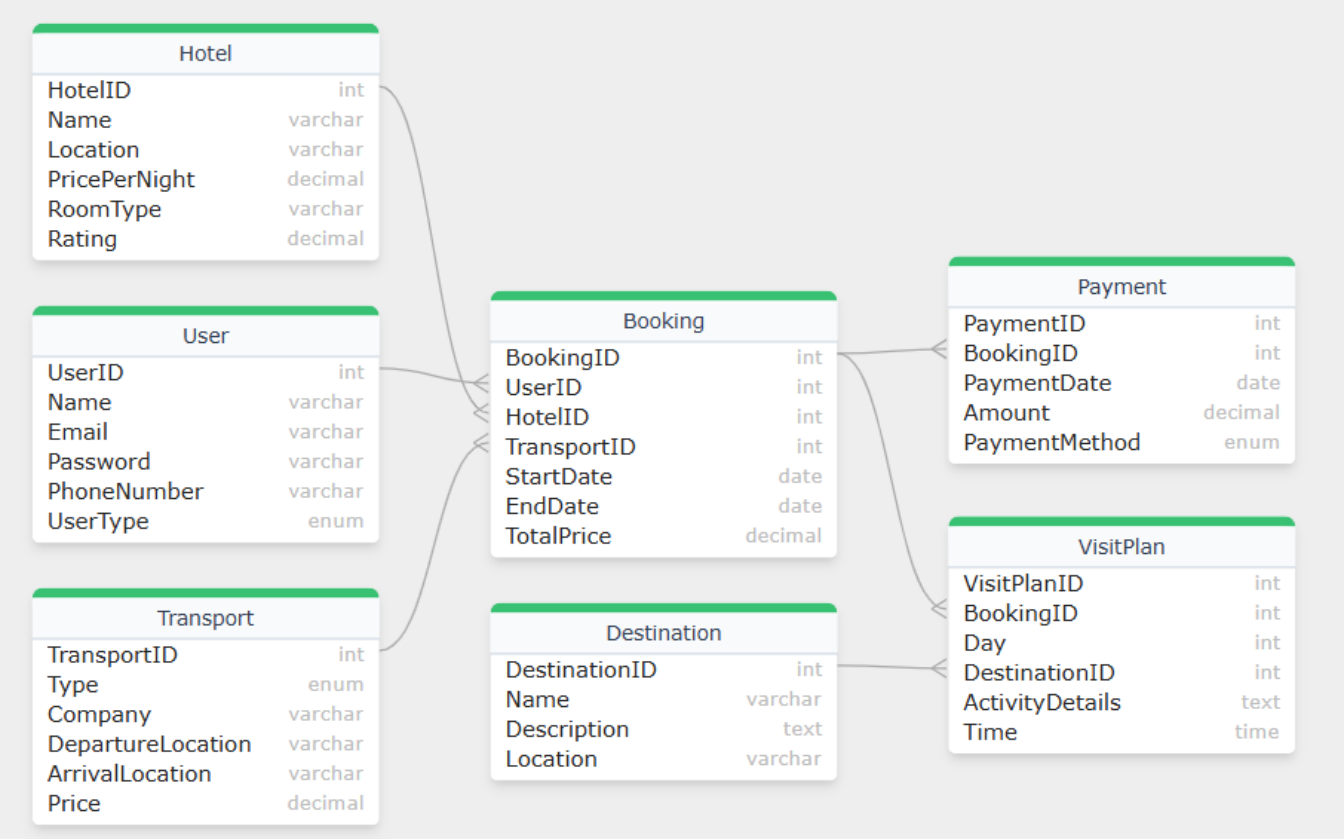
	Security	Performance	Availability	Capacity
Allow users to register by providing email, password, and name.	User data must be encrypted.	User registration within 1 second.	Registration always available.	Support up to 1M users for registrati
Allow users to log in using their registered email and password.	Passwords must be encrypted.	Login response time within 1 second.	Login service must be always available.	Login service should support 1M us
Use JWT tokens for secure user authentication.	JWT tokens securely stored.	Token creation and storage under 1 second.	JWT token service must be reliable.	Token storage scalable for many use
Allow users to search for destinations.	Search data must be encrypted.	Search results within 2 seconds.	Search function must be available 24/7.	Search can support 10,000 queries/s
Display available hotels.	Hotel data encrypted during transfer.	Hotel info fetched quickly.	Hotel data access must have high uptime.	Hotel data scalable to large dataset
Allow users to filter search results.	Search filters should be secure.	Filter results within 1 second.	Search filters always available.	Filters handle large user base.
Recommend alternate plans based on weather conditions.	Weather API access must be secure.	Real-time weather updates.	Weather data available in real-time.	Weather data scaled to many users
Fetch real-time weather data by API.	Weather data encrypted.	API response within 1 second.	Weather API must be accessible at all times.	API handles high query volume.
Recommend alternative dates or destinations based on weather forecasts.	Weather recommendations must be secure.	Recommendations updated immediately.	Recommendations available 24/7.	Recommendation system handles large ata.
Allow users to book hotels, transport, and tour packages.	Booking details encrypted.	Booking process for up to 10k users.	Booking service must be always available.	Booking system handles many user
Allow users to access booking history.	Booking history must be secure.	History loads in under 3 seconds.	Booking history accessible 24/7.	Booking history scalable.
Allow admin to manage user accounts, add/update/delete users.	Admin actions must be auditable.	Admin actions reflect immediately.	Admin actions available at all times.	Admin actions handle large data se
Allow admin to view and manage all bookings.	Admin bookings management must be secure.	Bookings load in 2 seconds.	Booking data available always.	Bookings system scales to many use

APPENDIXES

DATA FLOW DIAGRAM



ENTITY RELATIONSHIP DIAGRAM



WORK BREAKDOWN STRUCTURE: