Personalized Travel Recommendation based on user preferences

Software Requirements Specification

INT 219
FRONT END WEB DEVELOPER

Anshika Srivastava

Student Registration Numbers
12305830

Prepared for
Continuous Assessment 3
Spring 2025

Table of Contents

1. Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document. (Note: the following subsection annotates are largely taken from the IEEE Guide to SRS).

1.1 Purpose

This SRS defines the requirements for TravelMate, a web-based travel recommendation system. It is intended for faculty evaluation, project developers, and students. It outlines functional and non-functional specifications.

1.2 Scope

TravelMate provides personalized travel destination recommendations based on user preferences: Activity, Budget, Location, Trip Duration, Climate, and Visa-Free options. It offers:

- User authentication (signup/login)
- Preference collection and profile update
- Intelligent destination filtering
- Destination presentation with images and Wikipedia links

It does not include real-time booking or payment systems.

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- UI: User Interface
- DB: Database
- TravelMate: Name of the system

1.4 References

(1) IEEE Guide to SRS

(2) PHP and MySQL Documentation

(3) Wikipedia (for destination data)

1.5 Overview

This SRS is structured to define general product context, user expectations, interfaces, functionality, and limitations.

2. General Description

This section provides an overview of the TravelMate system, its environment, constraints, user expectations, and other relevant background information. It helps the reader understand the context in which the system operates and prepares for the specific requirements defined later.

2.1 Product Perspective

TravelMate is a **standalone web-based recommendation system** that enables users to receive personalized travel suggestions based on a variety of preferences. It is an original development project and does not depend on any existing systems or third-party APIs.

The system has a **modular architecture** consisting of:

- A **frontend** developed using HTML, CSS, and JavaScript for user interaction.
- A **backend** written in PHP that handles data processing and logic.
- A **MySQL database** that stores user data and preferences.

All destination data (e.g., images and Wikipedia links) are **locally mapped or statically configured**, making the application independent of external APIs.

2.2 Product Functions

The major functions of TravelMate include:

- **User Authentication:** Users can create an account or log in to access personalized features.
- **Preference Management:** After logging in, users can choose preferences such as activity, budget, location, trip duration, climate, and visa-free status.
- **Recommendation Generation:** Based on selected preferences, the backend filters through a curated list of destinations and presents suitable options.
- **Destination Display:** Recommendations are shown with images and clickable links to corresponding Wikipedia pages for more information.

2.3 User Characteristics

The intended users of TravelMate are:

- **General public** looking for vacation or travel ideas.
- **Students or young professionals** with varying levels of travel experience.
- Users are assumed to have basic familiarity with:
  - Web browsing
  - Filling out online forms
  - Understanding destination categories like “budget” or “visa-free”

No advanced technical knowledge is expected.

2.4 General Constraints

- Internet connection required
- Browser compatibility assumed
- Data based on pre-curated image and wiki mappings

2.5 Assumptions and Dependencies

- Users will provide truthful and valid input during registration and preference selection.
- Static destination data (image filenames and links) will be **manually maintained** by developers.
- Sessions are used to maintain login state; **cookies must be enabled** in the browser.
- Deployment platforms like **InfinityFree** or **XAMPP** are used during development and testing.
- The recommendation logic uses **if-else filtering**, not machine learning or real-time geolocation.

3. Specific Requirements

This section defines the **detailed requirements (D-requirements)** of the TravelMate system. These requirements guide the **software design, implementation, and testing**. Each requirement is uniquely identified, testable, complete, and consistent. The structure ensures traceability and maintainability throughout the development lifecycle.

3.1 External Interface Requirements

3.1.1 User Interfaces

• Login Page (`login.html`)
Provides input fields for email and password. Validates user credentials and starts a session.

• Signup Page (`signup.php`)
Allows new users to register by providing name, email, password, and initial preferences.

• Profile Page (`profile.php`)
Displays user details and preference form. Shows recommendations based on saved preferences.

• Contact Page (`contact.html` & `contact.php`)
Contains a form to send queries to the website admin using email.

• About Page (`about.html`)
Describes the purpose and mission of TravelMate.

3.1.2 Hardware Interfaces

No special hardware interfaces are required. The application is browser-based and works across standard desktop or mobile devices.

3.1.3 Software Interfaces

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** PHP 8+
- **Database:** MySQL (user, preference storage)
- **Server:** Apache or any PHP-supported web server

3.1.4 Communications Interfaces

- **HTTP/HTTPS protocol** is used for all browser-server communication.
- **Form POST** method is used for login, signup, and saving preferences.

3.2 Functional Requirements

3.2.1 User Authentication

3.2.1.1 Introduction

Allows users to securely log in or sign up to access personalized features.

3.2.1.2 Inputs

- Email and password (Login)
- Name, email, password, preferences (Signup)

3.2.1.3 Processing

- PHP scripts validate and hash passwords.
- Checks credentials against the database.
- Creates a session on success.

3.2.1.4 Outputs

- Redirects to profile.php if successful.
- Displays error message if invalid credentials.

3.2.1.5 Error Handling

- Invalid email format
- Password mismatch
- User already exists or incorrect login

3.2.2 Preference Submission and Update

3.2.2.1 Introduction

Enables users to save or update their preferences related to travel.

3.2.2.2 Inputs

- Activity, Budget, Location, Duration, Climate, Visa-Free status (via form)

3.2.2.3 Processing

- save_preferences.php updates the users table using UPDATE SQL query.

3.2.2.4 Outputs

- Redirects back to profile.php
- Shows updated preferences

3.2.2.5 Error Handling

- Empty or missing fields
- SQL/database update failure

3.2.3 Destination Recommendation

3.2.3.1 Introduction

Generates travel suggestions based on stored preferences using predefined logic.

3.2.3.2 Inputs

- User preferences fetched from database

3.2.3.3 Processing

- PHP if-else statements check conditions
- Matches preferences to relevant destination list
- Maps image and Wikipedia link using static arrays

3.2.3.4 Outputs

- Destination cards with:
  - Image (from /images/)
  - Destination name
  - Wikipedia link

3.2.3.5 Error Handling

- If no match found, fallback destinations are shown

- Image missing: default image shown
- Wikipedia link missing: fallback to basic str_replace logic

---

3.2.4 Contact Form Handling

3.2.4.1 Introduction

Allows users to send inquiries or feedback.

3.2.4.2 Inputs

- Name, email, message (via form)

3.2.4.3 Processing

- PHP processes the form
- Sends email using mail() or logs query in backend

3.2.4.4 Outputs

- Show success or error message

3.2.4.5 Error Handling

- Empty fields
- Invalid email
- Mail server failure

3.5 Non-Functional Requirements

3.5.1 Performance

The system shall ensure that all main pages (login, profile, recommendations) load within **3 seconds** under standard network conditions.

3.5.2 Reliability

The system must reliably maintain **user sessions** throughout their interaction. In case of errors (e.g., unauthorized access), users must be **redirected to appropriate pages** such as the login screen.

3.5.3 Availability

The TravelMate web application must be available **24/7** assuming the hosting server and database are operational.

3.5.4 Security

The system must implement **session-based login** for all authenticated routes. User **passwords must be securely stored** in the database (hashed) and never visible in plaintext.

3.5.5 Maintainability

The codebase should follow a **modular structure**:

- Separate files for backend logic (`.php`)
- Frontend styling (`.css`)
- Page markup (`.html`)

Commenting and naming conventions should enhance readability and future updates.

3.5.6 Portability

- The application must work smoothly on all modern browsers (Chrome, Firefox, Edge, Safari).
- It should be accessible across desktop, tablet, and mobile devices with responsive design.

3.7 Design Constraints

- Tailwind not used (pure CSS)
- Static folder-based structure

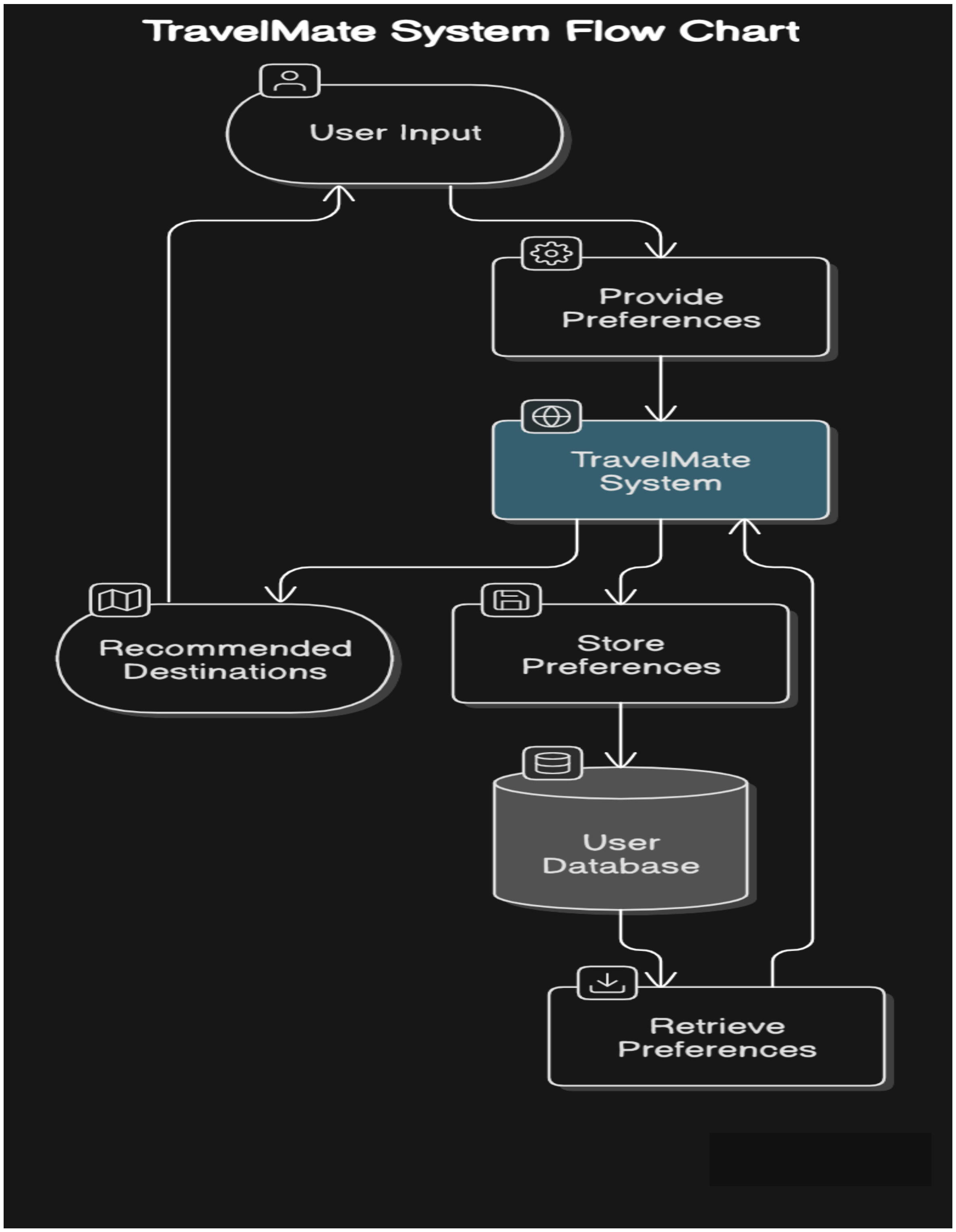
3.9 Other Requirements

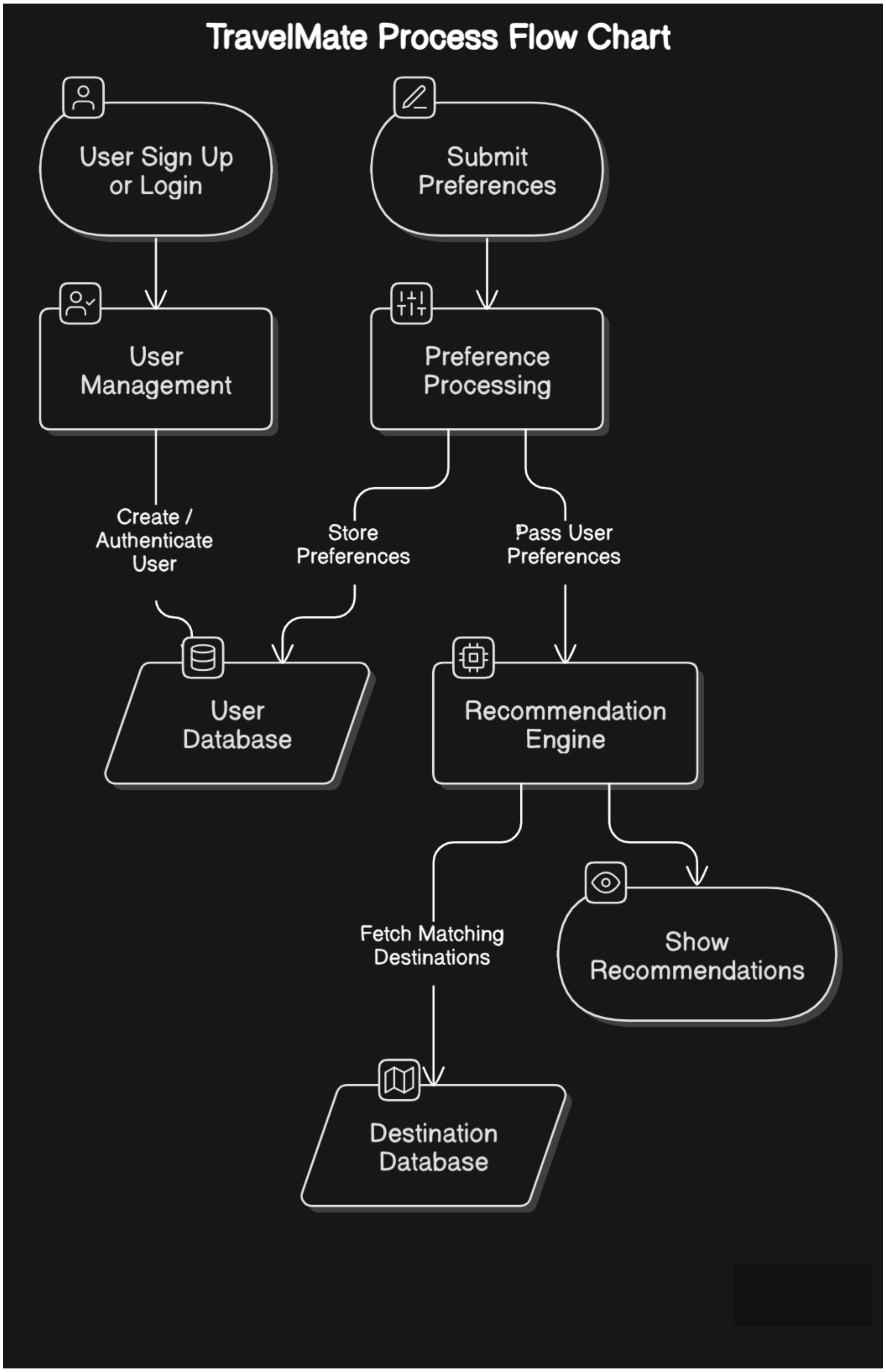
None at this stage

4. Analysis Models

In this section, we describe the various analysis models used to define and refine the functional requirements of TravelMate. Each model helps visualize the flow of data, the interaction of the system with external entities, and the internal structure of the software.

4.1 Data Flow Diagrams (DFD)




TravelMate Process Flow Chart
User Sign Up or Login
Submit Preferences
User Management
Preference Processing
Create / Authenticate User
Store Preferences
Pass User Preferences
User Database
Recommendation Engine
Fetch Matching Destinations
Show Recommendations
Destination Database

GitHub Link: https://github.com/anshikasrivastava3011/Personalized-Travel-Recommendation-based-on-user-preferences

A. APPENDICES

This appendix outlines the two core mapping arrays used in the backend PHP logic of the TravelMate application. The first array is the **destination image mapping**, which connects each travel destination to its corresponding image file or external image URL. This mapping ensures that, based on user preferences, the application can visually represent each recommended place without delay. The second array is the **Wikipedia link mapping**, which accurately links destination names to their appropriate Wikipedia pages. These mappings support a consistent user experience by presenting both visual and informational context for each recommended location. Together, these arrays help the recommendation system operate efficiently using curated data without relying on real-time external APIs.