**1. Introduction**

**1.1 Project Overview**

The **Web Application for Recommending Tourist Locations in a City** aims to help users discover, plan, and organize their travel activities. By inputting details such as city, time period, budget, and personal preferences, users receive a curated list of tourist attractions and activities, that are easily arranged in a schedule structure.

**2. Functional Requirements**

**2.1 User Authentication**

* Users can log in using email/password.
* Users can log in using Google authentication.
* Site functionalities differ based on login status.
* Users receive a confirmation email upon sign-up.

**2.2 Site Navigation & Structure**

* **Navbar:** Contains buttons for Create Schedule, View Schedules, View Cities & Activities, Log In/Out, and Contact.
* **Main Page:** Displays upcoming schedules (if any) or a “Start Schedule” button.
* **Create Schedule Page:** Allows users to select the time period, city, and activities.
* **View Created Schedules Page:** Lists saved schedules.
* **View Schedule Details Page:** Displays schedule breakdown, weather, and map routes.
* **Modify Schedule Page:** Allows users to adjust schedules.
* **City Browser Page:** Provides details on a selected city and its attractions.
* **Profile Settings Page:** Allows users to manage account settings.
* **Contact Page:** Provides support or feedback options.

**2.3 Core Functionalities**

**2.3.1 Schedule Creation Process**

* Users select a **city** and view top attractions.
* Users select a **time period** via a calendar.
* Users receive weather forecasts for selected dates.
* Users input or select a **hotel location**.
* Users browse and filter **activities** by category, budget, and location.
* Users can **drag and drop** activities into a daily schedule.
* Activity recommendations are optimized based on:
  + Default time durations (e.g., restaurants 2.5 hours, parks 3 hours)
  + Travel time between locations
  + Weather and estimated crowdedness
* The app generates a **daily route map** linking activities, with a direct link to Google Maps.
* Users can **rename** their schedule (default: “City Name - Period”).

**2.3.2 Viewing and Modifying Schedules**

* Users can view all created schedules in chronological order.
* Each schedule displays:
  + Title, city, period, hotel, and activities.
  + A daily itinerary with weather information.
  + A map with recommended travel routes.
  + A button to modify the schedule.
* The modify schedule feature allows users to:
  + Change selected activities.
  + Adjust time slots and durations.
  + Update hotel locations.
  + Regenerate optimal travel routes.

**2.3.3 City Browser**

* Users can search for cities.
* The system displays:
  + General city information (source TBD).
  + Top attractions and activities with filtering options.
  + An option to start a schedule for that city.

**3. Non-Functional Requirements**

**3.1 Performance**

* The application should respond within **3 seconds** for major actions.
* Search operations (cities, activities) should return results in **under 2 seconds**.

**3.2 Security**

* Passwords must be stored securely using encryption.
* Authentication should use OAuth for Google sign-in.
* User data should be protected with role-based access controls.

**3.3 Usability**

* The UI should be **responsive** for mobile and desktop.
* Drag-and-drop scheduling should be intuitive.
* Map and location services should provide **real-time** updates.

**3.4 Scalability**

* The application should support **thousands of users** without performance degradation.
* API requests should be optimized to reduce load.

**3.5 Reliability & Maintainability**

* The system should have **99.9% uptime**.
* Logs should track errors and user actions for debugging.

**4. Constraints & Dependencies**

**4.1 Technical Constraints**

* **Google Places API** and **Google Distance Matrix API** are required for location-based services.
* **SQL Server** is used for data storage.
* **ASP.NET** and **C#** are the primary backend technologies.
* **Bootstrap & JavaScript** are used for frontend development.

**4.2 Legal & Compliance Constraints**

* The application must comply with **GDPR** and **data privacy laws**.
* Users must be informed about **data collection** and given consent options.

**4.3 Project Dependencies**

* Internet connectivity is required for real-time API interactions.
* External data sources (e.g., weather, city info) may affect availability.

**5. Future Considerations**

* Multi-language support.
* AI-based personalized recommendations.
* Social sharing for travel plans.