Database System Final Project

Instructor: Ms. Zubaira Naz

Batch CS23 and AI23

Database Project Description

Here's the updated document incorporating all the changes for your **Travel Guide** project:

Travel Guide Project Document

1. **Project Title**: Travel Guide
2. **Project Description**: The Travel Guide project aims to create a platform where users can discover, share, and document their travel experiences. The primary objective is to provide users with a comprehensive database of destinations and attractions while allowing them to contribute their own recommendations. Key features include user-added locations, reviews for both locations and hotels, personalized bucket lists, and journals for documenting travel experiences.
3. **Scope**: The Travel Guide project will support the following key features:
    - User registration and authentication.
    - A database of destinations and attractions with detailed information.
    - Users can add new locations to the database for others to see.
    - Users can rate and review locations and hotels.
    - Users can create multiple bucket lists to plan future travels, with privacy settings (public/private).
    - Users can maintain journals to document their travel experiences, including memories and photos.
    - Integration with external APIs for real-time data on locations and hotels.
4. **Stakeholders**:
    - **Primary Users**: Travelers who want to explore new destinations, share experiences, and plan future trips.
    - **Secondary Users**: Friends or family of primary users who may view their bucket lists or journals.
    - **System Administrators**: Individuals responsible for maintaining the database, approving user-added locations, and ensuring data quality.
5. **Entity-Relationship Diagram (ERD)**:
    - **Entities**:
        - User: Stores user information.
        - Location: Stores information about destinations and attractions.
        - Hotel: Stores information about hotels.
        - BucketList: Stores user-created bucket lists.
        - BucketListLocations: Links bucket lists to specific locations.
        - Journal: Stores user-created journals.

        - JournalEntries: Links journals to specific locations and contains user experiences.
        - LocationReviews: Stores user reviews for locations.
        - HotelReviews: Stores user reviews for hotels.
    - **Relationships**:
        - One-to-many between User and Location, BucketList, Journal, LocationReviews, HotelReviews.
        - Many-to-many between BucketList and Location, and one-to-many between Journal and JournalEntries.
6. **Schema Design**:
    - **Tables**:
        - **User**: `UserID (PK), Username, Password, Email, RegistrationDate`
        - **City :** cityid(fk),country,cityname
        - **Location**: `LocationID (PK), LocationName, Cityid(fk), Description`
        - **Hotel**: `HotelID (PK), HotelName,cityid (FK), Price, Rating`
        - **BucketList**: `ListID (PK), UserID (FK), ListName, Privacy (Public/Private), CreationDate`
        - **BucketListLocations**: `BucketListID (FK), LocationID (FK)`
        - **Journal**: `JournalID (PK), UserID (FK), JournalName, Privacy (Public/Private), CreationDate`
        - **JournalEntries**: `EntryID (PK), JournalID (FK), LocationID (FK), EntryText, Photos, DateVisited, CreatedAt`
        - **LocationReviews**: `ReviewID (PK), LocationID (FK), UserID (FK), Rating, Comment, SubmissionDate`
        - **HotelReviews**: `ReviewID (PK), HotelID (FK), UserID (FK), Rating, Comment, SubmissionDate`
    - **Primary Keys**: Each table will have its respective primary key.
    - **Foreign Keys**: Relationships will be established using foreign keys to maintain data integrity.

Extra notes:

Here are some APIs you might consider integrating into your **Travel Guide** project to enhance its functionality:

1. Google Places API

- **Purpose**: To retrieve information about points of interest, including attractions, restaurants, hotels, and other locations.
- **Use Cases**:
    - Fetch detailed information about tourist attractions and locations.
    - Retrieve user reviews and ratings for various places.
    - Get images of locations.

2. OpenWeatherMap API

- **Purpose**: To provide weather data for various locations.
- **Use Cases**:
    - Display current weather conditions and forecasts for destinations on the platform.

    - Show weather forecasts for users' planned trips or bucket list locations.

3. Booking.com API (or other hotel APIs)

- **Purpose**: To provide information about hotel availability, pricing, and amenities.
- **Use Cases**:
    - Allow users to search for hotels near a specific location.
    - Display hotel ratings, reviews, and booking options directly on your platform.

4. Mapbox or Google Maps API

- **Purpose**: To create interactive maps that show locations and routes.
- **Use Cases**:
    - Display user-added locations and their coordinates on a map.
    - Allow users to see routes between different locations on their bucket lists or journals.

5. Flickr API (or Unsplash API)

- **Purpose**: To fetch photos related to locations.
- **Use Cases**:
    - Display user-uploaded photos alongside location information.
    - Automatically pull in beautiful images of destinations for users to browse.

6. TripAdvisor API (if available)

- **Purpose**: To provide reviews and recommendations for restaurants, attractions, and hotels.
- **Use Cases**:
    - Enhance user reviews with TripAdvisor ratings and reviews for locations.

7. Yelp API (if available)

- **Purpose**: To access business information and user reviews.
- **Use Cases**:
    - Provide restaurant reviews and ratings, enhancing the dining aspect of travel guides.

Considerations:

- **API Key Management**: Most APIs require you to sign up and obtain an API key for access. Make sure to keep your keys secure.
- **Rate Limits**: Be aware of any rate limits set by the API providers to avoid exceeding usage caps.
- **Documentation**: Refer to the API documentation for detailed instructions on how to make requests and handle responses.

Project requirements details:

You should keep these things in mind while deciding your project

Note: You should choose a project that has unique value and make sure to highlight that idea. Avoid selecting projects that are easily available on GitHub or have tutorials on YouTube, such as an e-commerce store or a Twitter clone.

Group Size: Maximum 3 students

Allowed Tech Stack :

- A web development framework like React, Next, Node, Express, Django, Flask, etc.
- For app development, you have to use Android Studio, React Native, Kotlin, swift, etc.
- For a database, you have to use MySQL or MongoDB.

Normalization: The database should be normalized and all the details of this process have to be documented at the end of the project.

Front-End Requirements:

User Interface (UI) Design

- **Responsive Design**: The website should fully support different screen sizes (desktop, tablet, mobile).
- **Navigation**: Implement an intuitive navigation structure (e.g., side menu or navigation bar) to allow users to switch between different features easily.
- **Form Inputs**: Provide forms for data input, validation, and submission, such as creating, updating, and deleting records.
- **Feedback & Alerts**: Include visual feedback (e.g., success/failure messages) for user actions such as submitting a form, invalid inputs, or system errors.
- **Data Display**: Design data display screens using widgets such as tables, lists, and cards to show information retrieved from the database in an organized way.

User Roles and Access

- **Role-Based Access Control**: Define different user roles (e.g., Admin, Editor, Viewer) to limit access to specific functionalities. The UI should change dynamically based on the logged-in user's permissions.
- **Admin Panel**: Provide an interface for system administrators to manage users, settings, and the database structure (if applicable).

Documentation:

- **Data Dictionary:** Provide a comprehensive description of each table, column, and relationship within the database.
- **ERD Explanation:** Write an explanation of the ERD, including the rationale behind design choices.
- **User Manual:** Outline how users will interact with the database system, including instructions for performing key tasks.

- **Queries:** Use SQL or MongoDB to create databases and to write queries.

- **Front-End Documentation**: Include details on how the Flutter website is structured, the key components, and how API interactions occur.

General Instructions

- Make ERD on draw.io
- Save your ERD in pdf or as an image.
- Make all database tables in MySQL workbench using queries.
- Must bring your laptops at the time of evaluation and viva. You may ask to write any query or any manipulation with your database according to your ERD.
- In case of any problem while working on a project, please ask TAs or consult your Instructor.

GOOD LUCK!