Database Design

Overview

The WanderLogixs web application is designed to efficiently manage and organize structured, interconnected data related to users' travel experiences, including trip details, expenses, itineraries, media, and memories. For this purpose, we have chosen PostgreSQL as the most appropriate database technology.

PostgreSQL is an ideal choice due to its robust support for relational databases, which aligns seamlessly with the structured nature of the data. It excels in managing complex relationships, ensuring that data integrity is maintained. This is crucial for associating user profiles with their respective trips, expenses, itineraries, media, and memories. PostgreSQL's proven ability to handle one-to-one, one-to-many, and many-to-many relationships provides the necessary flexibility for our application.

Furthermore, PostgreSQL's data consistency and security features are paramount for storing sensitive user information, such as authentication credentials, while offering a mature ecosystem for efficient data retrieval and storage operations. Its support for structured data types allows us to store various data formats, including dates, decimals, and text, precisely matching the diverse information associated with travel experiences.

PostgreSQL stands out as the optimal choice for the WanderLogixs web application, offering the structured foundation required to create a user-friendly, secure, and efficient platform for managing and reliving travel experiences.

Database Specifications

To design a relational database for the WanderLogixs web application using PostgreSQL, we need to create an Entity-Relationship diagram (ERD) that outlines the tables, their columns, data types, and relationships.

Users

The Users collection stores user information for authentication and personalization. It enables users to have personalized dashboards and secure access to their travel data.

Relationships:
User (UserID) - User (One-to-One): Each user has a unique ID.
User (UserID) - Trip (One-to-Many): Each user can have multiple trips.
User (UserID) - Memory (One-to-Many): Each user can have multiple memories.

Document Structure:

```
{
    “UserID”: “string (auto-generated id)”,
    “Username”: string”,
    “Email”: “string”,
    “Password”: “string”
}
```

Trips

The Trips collection is central to the application, recording trip details, and forming the core structure of a user's travel history. It associates trips with users through foreign keys.

Relationships:
Trip (TripID) - User (One-to-One): Each trip is associated with one user.
Trip (TripID) - Expense (One-to-Many): Each trip can have multiple expenses.
Trip (TripID) - Itinerary (One-to-Many): Each trip can have multiple itinerary entries.
Trip (TripID) - Media (One-to-Many): Each trip can have multiple media uploads.

Document Structure:

```
{
    "TripID": "string (auto-generated id)",
    "UserID": "string (foreign key: User.UserID)",
    "Destination": "string",
    "StartDate": "date",
    "EndDate": "date",
    "Purpose": "string"
}
```

Expense

The Expense collection is designed to track expenses related to specific trips, categorized for cost management. It's associated with trips through foreign keys.

Relationships:
Expense (ExpenseID) - Trip (One-to-One): Each expense is associated with one trip.

Document Structure:

```
{
    "ExpenseID": "string (auto-generated id)",
     "TripID": "string (foreign key: Trip.TripID)",
    "Date": "date",
    "Category": "string",
    “Amount": "decimal(10, 2)",
    "Description": "text"
}
```

Itinerary

The Itinerary collection is used to create and manage day-by-day schedules for trips, helping users plan their activities. It's linked to trips through foreign keys.

Relationships:
Itinerary (ItineraryID) - Trip (One-to-One): Each itinerary entry is associated with one trip.

Document Structure:

```
{
    "ItineraryID": "string (auto-generated id)",
    "TripID": "string (foreign key: Trip.TripID)",
    "Date": "date",
    "Activity": "string",
    "Notes": "text",
 }
```

Media

The Media collection allows users to upload and associate media content with trips, enriching the travel experience. It's connected to trips through foreign keys.

Relationships:
Media (MediaID) - Trip (One-to-One): Each media entry is associated with one trip.

Document Structure:

```
{
    "MediaID": "string (auto-generated id)",
    "TripID": "string (foreign key: Trip.TripID)",
    "Date": "date",
    "MediaType": "string",
    "FileURL": "string",
    "Description": "text"
 }
```

Share/Memory

The Share/Memory collection is used for users to share and save travel memories, such as stories and notes. It's associated with both users and trips through foreign keys.

Relationships:
Memory (MemoryID) - User (One-to-One): Each memory is associated with one user.
Memory (MemoryID) - Trip (Many-to-Many): Memories can be associated with multiple trips, and trips can have multiple memories.

Document Structure:

```
{
    "MemoryID": "string (auto-generated id)",
    "UserID": "string (foreign key: User.UserID)",
    "Date": "date",
    "Title": "string",
    "Text": "text"
}
```

Purpose, Implementation and Interactions

Users

Purpose:
The Users collection stores user information for authentication and personalization. It's central to the application's user management and security.

Implementation:
This collection is implemented using a relational database structure where each user is assigned a unique autogenerated ID. User information such as username, email, and password (hashed and salted) is stored securely.

Interactions:
Users interact with this collection when registering, logging in, and updating their profiles. The collection is used for authentication and to personalize the user experience.

Trips

Purpose:
The Trips collection is central to the application, recording trip details, and forming the core structure of a user's travel history. It associates trips with users through foreign keys.

Implementation:
This collection is implemented as part of a relational database. Each trip has a unique autogenerated ID, and it is associated with a user (foreign key) who created the trip. The trip details, including the destination, start and end dates, and purpose, are stored for each trip.

Interactions:
Users interact with this collection to create, update, and view their trip details. It's a fundamental component for organizing and managing travel experiences.

Expense

Purpose:
The Expense collection is designed to track expenses related to specific trips, categorized for cost management. It's associated with trips through foreign keys.

Implementation:
Expenses are stored in a relational database with a unique ID for each expense. Each expense is linked to a specific trip (foreign key). The collection records the date, category, amount, and description of expenses.

Interactions:
Users interact with this collection to record and categorize their expenses during trips. It helps in cost tracking and reporting.

Itinerary

Purpose:
The Itinerary collection is used to create and manage daybyday schedules for trips, helping users plan their activities. It's linked to trips through foreign keys.

Implementation:
This collection is part of the relational database structure. It stores unique IDs for each itinerary entry and associates them with specific trips. Details such as the date, planned activities, and notes are recorded.

Interactions:
Users interact with this collection to plan their daily activities during trips, keeping their itineraries organized and accessible.

Media

Purpose:
The Media collection allows users to upload and associate media content with trips, enriching the travel experience. It's connected to trips through foreign keys.

Implementation:
Media files and their details are stored in the relational database. Each media item has a unique ID and is linked to a specific trip or a day within a trip. The collection records the date, media type, file URL, and descriptions.

Interactions:
Users interact with this collection to upload photos, videos, and notes related to their trips, enhancing their travel memories.

Share/Memory

Purpose:
The Share/Memory collection is used for users to share and save travel memories, such as stories and notes. It's associated with both users and trips through foreign keys.

Implementation:
Memories and shared content are stored in the relational database. Each memory has a unique ID and is associated with a user and, optionally, with specific trips. The collection records the date, title, and content of the memories.

Interactions:
Users interact with this collection to create and share travel memories, including stories and notes. Memories can be linked to specific trips, allowing users to relive their experiences.