**Data Management Plan for Flight Buddy Wiki URL:**<https://github.com/srikarmiriyala/GDPProject-02/wiki/Data-Management-Plan-(Iteration-2)>

**Data Management Plan for Flight Buddy:**

**Summary of Data to Store:**

**Entities and Their Attributes:**

1. **User**
   * **UserID** (PK): Primary key, unique identifier for each user.
   * **Name, Email, Password, Phone**: Basic user information for account management and contact.
2. **Destination**
   * **DestinationID** (PK): Primary key, unique identifier for each destination.
   * **Name, Description, Country**: Information about each travel destination.
3. **Trip**
   * **TripID** (PK): Primary key, unique identifier for each trip.
   * **DestinationID** (FK): Foreign key linking to the destination.
   * **StartDate, EndDate**: The duration of the trip.
   * **Field**: Additional details or specific field related to the trip (purpose might need clarification).
4. **Bookings**
   * **BookingID** (PK): Primary key, unique identifier for each booking.
   * **UserID, TripID, FlightID** (FKs): Foreign keys linking to the user, trip, and flight respectively.
   * **BookingDate, Status**: Date of booking and current status (e.g., confirmed, canceled).
5. **Flight**
   * **FlightID** (PK): Primary key, unique identifier for each flight.
   * **Airline, DepartureTime, ArrivalTime, DepartureAirport, ArrivalAirport, Price**: Details about the flight schedule and pricing.
6. **Activity**
   * **ActivityID** (PK): Primary key, unique identifier for each activity.
   * **DestinationID** (FK): Foreign key linking to the destination.
   * **Name, Description, Type**: Details about the activity available at a destination.
7. **Review**
   * **ReviewID** (PK): Primary key, unique identifier for each review.
   * **UserID, TripID** (FKs): Foreign keys linking to the user and trip.
   * **Rating, Comments, Date**: User feedback about a trip.
8. **Payment**
   * **PaymentID** (PK): Primary key, unique identifier for each payment.
   * **BookingID** (FK): Foreign key linking to the booking.
   * **FlightID** (FK): Foreign key linking to the flights.
   * **Amount, PaymentDate, PaymentMethod**: Details about the transaction.
9. **Packing Checklist**
   * **ChecklistID** (PK): A unique identifier for each checklist.
   * **TripID** (FK): Foreign key linking to the Trip entity to associate the checklist with a specific trip.
   * **UserID** (FK): Foreign key linking to the User entity, indicating which user the checklist belongs to.
   * **ItemName**: Name of the item to be packed (e.g., passport, clothes).
   * **Status**: Indicates whether the item has been packed ('packed') or not ('unpacked').

**Relationships between Tables:**

* **User to Bookings**: One-to-Many (A user can have multiple bookings).
* **Trip to Bookings**: One-to-Many (A trip can have multiple bookings associated with it, possibly representing different users or different aspects like flights and accommodations).
* **Flight to Bookings**: One-to-Many (A flight can be part of multiple bookings).
* **Destination to Trip**: One-to-Many (A destination can host multiple trips).
* **Destination to Activity**: One-to-Many (A destination can offer multiple activities).
* **User to Review**: One-to-Many (A user can write multiple reviews on different trips).
* **Trip to Review**: One-to-Many (A trip can receive multiple reviews from different users).
* **Booking to Payment**: One-to-One (Typically, each booking results in a single payment transaction, although this could vary based on business rules).
* **Packing Checklist to User**: One-to-Many (Each user can have multiple packing checklists, one for each trip).
* **Packing Checklist to Trip**: One-to-Many (Each trip can have its own packing checklist).

**Initial Security Measures:**

**Access Control:**

* Deploy role-based access control (RBAC) to limit access to sensitive information according to user roles.
* Enforce robust authentication and authorization protocols to validate users and manage their access privileges.

**Data Encryption:**

* Utilize advanced encryption algorithms to encrypt sensitive data, including passwords and payment details.
* Employ SSL/TLS encryption for secure transmission of data over networks, thwarting potential eavesdropping attempts.
* Implement encryption technologies provided by the database management system to secure data at rest.

**Mapping of Functional Requirements to Data Storage:**

1. **User Registration and Profiles:**
   * Store user profile data including username, email, password (encrypted), profile picture, and preferences.
2. **Destination Planning and Selection:**
   * Store destination data including destination name, description, images, attractions, and weather information.
3. **Flight Planning and Selection:**
   * Store flight data including departure/destination airports, dates/times, airlines, flight numbers, seat availability, and prices.
4. **Interactive Maps and Itinerary Creation:**
   * Use destination data for interactive maps and itinerary creation features.
5. **Detailed Flight Guides:**
   * Utilize flight data to provide detailed flight guides for exploring destinations and airlines.
6. **Real-time Booking and Payment Confirmation:**
   * Store user flight reservations and payment information (encrypted) securely.
7. **Calendar View for Flight Details:**
   * Utilize stored flight data to display flight details and schedules in a calendar view for planning.
8. **Search and Filter Functionality:**
   * Utilize stored destination and flight data for search and filter functionality for flight options.
9. **Packing Checklist:** -Store user-specific packing checklists, integrate with trip profiles, and enable user interaction for adding, editing, and syncing essential travel items and documents.

