**Software requirements specification (SRS) wiki URL:** <https://github.com/srikarmiriyala/GDPProject-02/wiki/Software-requirements-specification-(SRS)>

**Software requirements specification (SRS): Team 02**

**Project Information:**

**Project Charter:**  
**Summary of the Problem:** The problem involves the need for a comprehensive Travel Planning and Itinerary Management Application, known as TravelBuddy, to address the inefficiencies and challenges faced by travelers in planning and organizing trips. Currently, the process of trip planning is disjointed and inefficient, requiring users to gather information from multiple sources and tools. There's a lack of a centralized platform, leading to manual organization of trip details and cumbersome user experience.

**Motivation for Solving the Problem:** Implementing TravelBuddy will revolutionize the way travelers plan and manage their trips, significantly enhancing the efficiency and convenience of the travel planning process. By providing a centralized platform for trip planning, booking accommodations, and discovering attractions, TravelBuddy aims to streamline the entire travel experience. This system will address current operational challenges, improve user satisfaction, and elevate the overall quality of travel planning.

**Required Functionality:** TravelBuddy should include the following features:

**User Registration and Profile Management:** Allow users to create accounts, manage their profiles, and customize preferences.

**Travel Organization & Planning:** Easy-to-use tools for choosing locations, times, and activities. For effective travel planning, interactive maps and tools for creating itineraries are available.

**Explore Your Destination:** Detailed travel guides that provide details on nearby landmarks, attractions, and services.

**Booking Interface:** The app's booking interface enables smooth real-time booking for trips through integration with travel APIs.

**Functionality on the backend:** Account management and user authentication for safe access. User preferences, reservations, and itinerary data are stored and retrieved. Real-time booking and data retrieval through integration with external travel APIs.

**Database:** Making use of a MongoDB database to store and retrieve data efficiently. By putting these features into place, TravelBuddy hopes to solve the problems and inefficiencies that exist in the trip planning process and give users a smooth and pleasurable travel planning experience.

Through the integration of these features, TravelBuddy seeks to deliver users a smooth and pleasurable travel planning journey, effectively tackling existing obstacles and inefficiencies within the trip planning process.

**Developer About Pages:**

* **Sai Srikar:** Expertise in Java, Spring, Spring Boot, and JDBC. Aiming to transition into a full-stack Cloud Java engineer role.
* **Sowmya:** Independent learner with interests in Data Science and DevOps tools.
* **Pravallika:** Strong problem-solving skills with a solid grasp of complex mathematical concepts.
* **Nishchala:** Strong analytical and problem-solving abilities with experience in cybersecurity and data analysis.
* **Yeswanth:** Skilled multitasker with interests in Python, Java, JavaScript, and exploring new technologies like IoT and AI.
* **Uday kiran:** Expertise in complex algorithms and data structures with a problem-solving mindset.

**Problem Statement:**

The current process of planning trips is disjointed and inefficient, as travelers have to gather information from multiple sources and tools. There's no centralized platform, so users must manually gather and organize trip details from various sources, making the experience cumbersome and time-consuming.

**1. Challenges and Inefficiencies:** The current trip planning process is fraught with numerous challenges and inefficiencies. These encompass difficulties in selecting destinations, dates, and activities, as well as obstacles in obtaining comprehensive destination information, securing immediate bookings for accommodations and transportation, and handling trips in offline or low connectivity scenarios.

**2. Lack of Centralized Platform:** A significant inefficiency in the current system is the lack of a unified platform for trip planning. Without a centralized interface to gather all trip information, travelers must navigate various sources separately, leading to confusion and inefficiency.

**3. Inadequate Destination Guides:** The absence of thorough destination guides adds complexity to the trip planning experience for travelers. Without detailed insights into local attractions, facilities, and noteworthy places, individuals may find it challenging to make well-informed decisions about their travel itineraries, risking the possibility of overlooking significant experiences.

**4. Absence of Real-time Booking Integration:** The current method lacks integration with travel APIs for instant booking of flights, accommodations, and transportation. Consequently, travelers are compelled to toggle between various tabs and applications to finalize bookings, resulting in a fragmented and inconvenient booking journey.

**5. Issues with Offline Functionality:** Existing trip planning tools suffer from insufficient offline features, creating obstacles for travelers facing restricted internet connectivity during their trips. Without offline functionality, users cannot access crucial trip details or make reservations without an internet connection, adding further complexity to the planning process.

**6. Impact on Traveler Experience:** Together, these obstacles and shortcomings have a detrimental effect on the overall experience of travelers, resulting in annoyance and discontentment. It is essential to tackle these issues to improve the trip planning process and guarantee a smooth and pleasurable experience for travelers.

**Design:**

**Use Cases:  
  
Use Cases for Travel Buddy System:**

**Use Case 1: User Registration and Profile Creation**

* **Functionality:**
  + Collect registration details: email, password, personal info.
  + Grant access to personalized dashboards.
* **Input:** User details: email, password, name, DOB.
* **Execution:** Validate and securely store data.
* **Output:** Confirmation of successful registration, access to dashboard.

**Use Case 2: Travel Preferences Customization**

* **Functionality:**
  + Access profile settings to customize preferences.
  + Securely save and update preferences.
* **Input:** User preferences for seating.
* **Execution:** Save preferences securely.
* **Output:** Confirmation of successful customization.

**Use Case 3: Trip Planning and Destination Selection**

* **Functionality:**
  + Accept user input for destination, dates, budget.
  + Present destination, flight options.
* **Input:** User preferences: destination, dates, budget.
* **Execution:** Retrieve and present relevant options.
* **Output:** Present various options, confirm selected itinerary.

**Use Case 4: Interactive Map Exploration**

* **Functionality:**
  + Grant access to interactive maps.
  + Offer real-time info and recommendations.
* **Input:** User inputs: destinations, preferences.
* **Execution:** Retrieve data, present interactive map.
* **Output:** Display interactive map with recommendations.

**Use Case 5: Flight Booking and Payment**

* **Functionality:**
  + Allow flight search, selection, booking, payment.
* **Input:** Destination, dates, preferences, payment info.
* **Execution:** Retrieve flights, process booking, payment.
* **Output:** Confirmation of successful booking, receipt.

**Use Case 6: Calendar View and Itinerary Management**

* **Functionality:**
  + Grant access to calendar view of itineraries.
  + Synchronize data, provide notifications.
* **Input:** Itinerary details.
* **Execution:** Populate calendar, issue notifications.
* **Output:** Display itinerary with notifications.

**Use Case 7: Secure Data Storage and Retrieval**

* **Functionality:**
  + Securely store, retrieve user data.
* **Input:** User data: profile, preferences, bookings.
* **Execution:** Implement secure storage, retrieval.
* **Output:** Confirmation of secure storage, retrieval.

**Use Case 8: Intuitive User Interface**

* **Functionality:**
  + Design user-friendly interface.
  + Ensure compatibility, responsiveness.
* **Input:** User interactions.
* **Execution:** Design based on UX principles.
* **Output:** Present user-friendly interface.

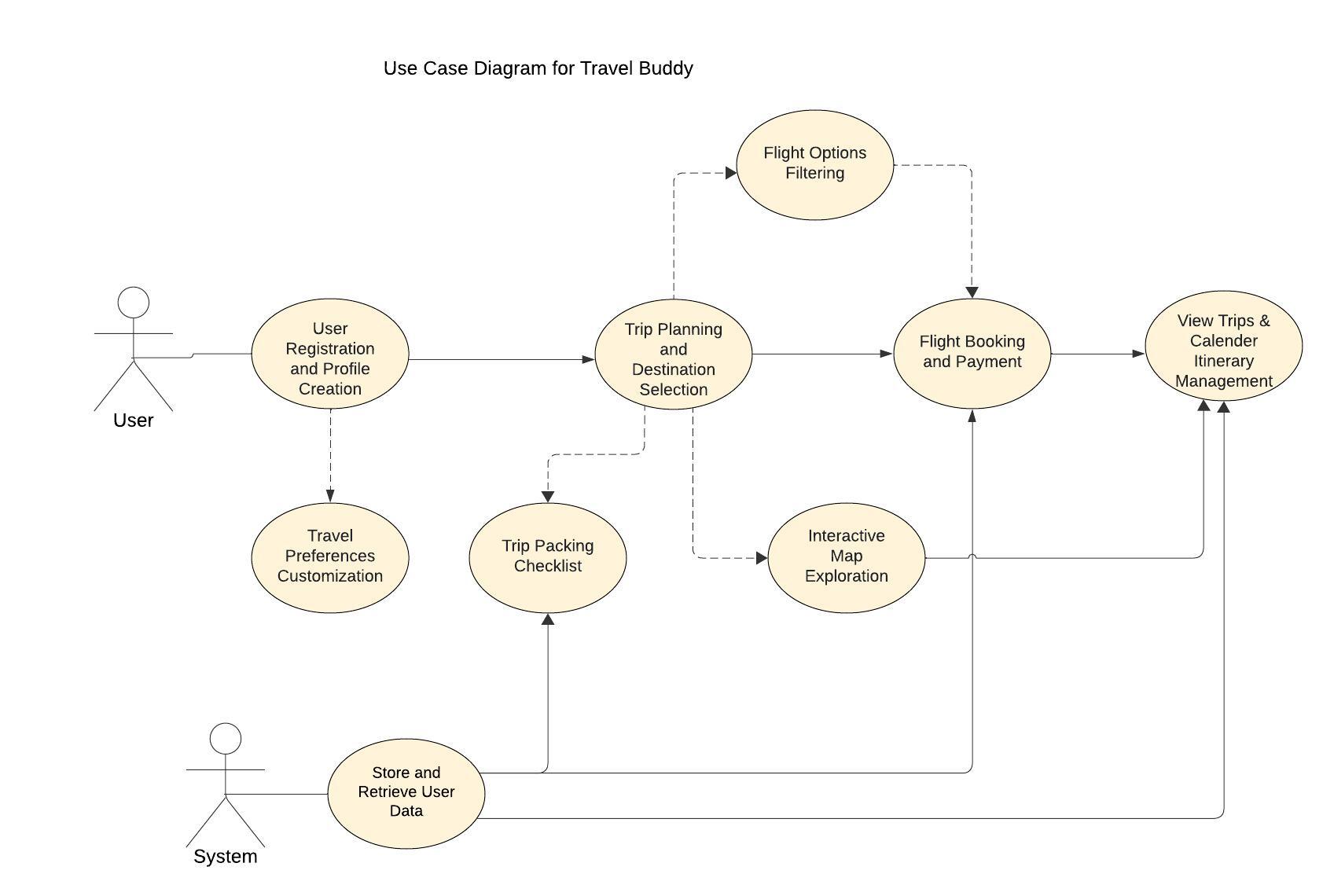
**Use Case 9: Flight Options Filtering**

* **Functionality:**
  + Allow search, filter flights by criteria.
* **Input:** Search criteria: airline, time, price.
* **Execution:** Retrieve, filter options.
* **Output:** Display filtered options for selection.

**Use Case 10: Trip Packing Checklist**

* **Functionality:**
  + Provide users with a built-in checklist feature to assist in packing and preparing for trips, ensuring they include essential items and documents.
* **Input:**
  + User creates and customizes a packing checklist with necessary items and documents (e.g., passport, tickets, clothing, toiletries, chargers).
* **Execution:**
  + The system securely saves the user's checklist. Users can access, update, and check off items as they pack, ensuring they do not forget anything.
* **Output:**
  + Confirmation of the successfully saved checklist, along with notifications for each item checked off, indicating it has been packed.

These use cases cover various functionalities of the TravelBuddy platform, from user registration to trip planning, booking, payment, and data management.



**Functional Requirements:**   
  
**Functional Requirements for Travel Buddy (Iteration 2):**

**The system SHALL:**

* Allow users to register and create profiles.
* Provide tools for planning and selecting destinations, dates, and flight options.
* Offer interactive maps and flight itinerary creation features.
* Present detailed flight guides for exploring destinations and airlines.
* Facilitate real-time booking for flights through integration with airline APIs and confirm payment.
* Display flight details and schedules in a calendar view to aid in planning.
* Store and retrieve user data, including preferences, flight reservations, and itineraries, securely.
* Ensure a clean and intuitive user interface for seamless navigation and flight management.
* Offer search and filter functionality for flight options.
* Offer a built-in checklist feature to help users pack and prepare for their trips, including essential items and documents.

**The system SHOULD:**

* Synchronize user data across platforms in real-time for consistency.
* Track the progress of flight booking tasks throughout their lifecycle.

**The system MAY:**

* Enable collaboration among users by allowing sharing updates and comments on flight plans.
* Set deadlines for flight booking tasks based on user input.
* Send reminders to users about upcoming flight deadlines.
* Integrate weather forecast data for selected destinations to help users plan their trips accordingly.
* Provide tools for users to set and track their travel budgets, including expenses for flights, accommodations, and activities.

**Non-functional Requirements:**

* Ensure robust performance for actions like searches and bookings, completing them in seconds.
* Design an intuitive and user-friendly interface to enable efficient navigation without extensive training.
* Make the application compatible with various devices and operating systems, ensuring consistent performance across different browsers.
* Develop the system using modular and well-documented code for easy maintenance and updates.
* Maintain code quality and stability over time by implementing automated testing and continuous integration.
* Secure sensitive user data, such as personal information and payment details, through encryption to prevent unauthorized access.
* Optimize device performance by implementing battery-efficient algorithms and minimizing resource usage, ensuring smooth navigation and responsiveness.

**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:**

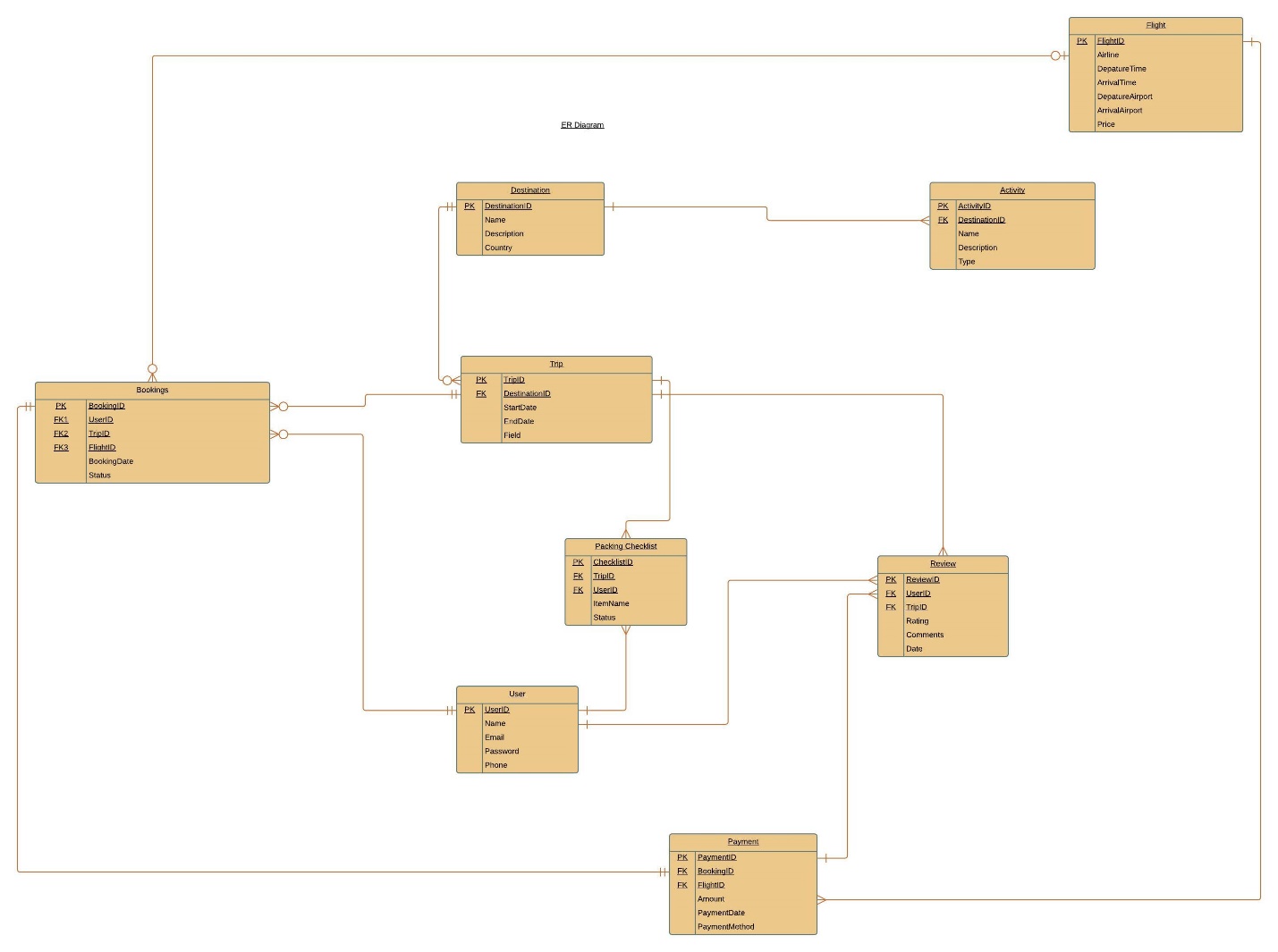
* Deploy role-based access control (RBAC) to limit access to sensitive information according to user roles.
* Enforce robust authentication and authorization protocols to validate users and manage their access privileges.

**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.



**Proposed Prototypes:**   
**List of Proposed Prototypes for Travel Buddy System (Iteration 2):**

1. **User Registration Prototype:**
   * Prototype for user registration and profile creation, including input fields for email, password, name, and date of birth. It should demonstrate the validation of input data and provide feedback upon successful registration.
2. **Travel Preferences Customization Prototype:**
   * Prototype showcasing the customization of travel preferences, particularly focusing on seating preferences. It should allow users to access their profile settings, update seating preferences, and receive confirmation upon successful customization.
3. **Trip Planning and Destination Selection Prototype:**
   * Prototype for trip planning and destination selection, featuring input fields for destination, dates, and budget. It should present various destination and flight options based on user preferences, allowing users to select their itinerary and receive confirmation upon selection.
4. **Interactive Map Exploration Prototype:**
   * Prototype demonstrating interactive map exploration, providing real-time information and recommendations based on user inputs such as destinations and preferences. It should display an interactive map with relevant data and recommendations for user exploration.
5. **Flight Booking and Payment Prototype:**
   * Prototype for flight booking and payment, allowing users to search for flights, select options, and complete the booking process. It should include input fields for destination, dates, preferences, and payment information, providing confirmation upon successful booking and payment.
6. **Calendar View and Itinerary Management Prototype:**
   * Prototype showcasing calendar view and itinerary management, enabling users to view their travel itineraries, synchronize data, and receive notifications. It should populate a calendar with itinerary details and provide notifications for upcoming trips.
7. **Secure Data Storage and Retrieval Prototype:**
   * Prototype demonstrating secure data storage and retrieval, ensuring the confidentiality and integrity of user data. It should securely store and retrieve user profiles, preferences, and bookings, providing confirmation of secure storage and retrieval processes.
8. **Intuitive User Interface Prototype:**
   * Prototype focusing on designing an intuitive user interface based on UX principles. It should feature user-friendly interface elements, compatibility across devices, and responsiveness to user interactions.
9. **Flight Options Filtering Prototype:**
   * Prototype allowing users to search and filter flight options based on criteria such as airline, time, and price. It should retrieve and display filtered options for user selection, providing a seamless filtering experience.
10. **Travel Checklist Generator Prototype:**
    * Prototype generating personalized travel checklists based on the user's destination, travel dates, and preferences. It should provide suggestions for essential items to pack, documents to carry, and tasks to complete before the trip.

**Meeting Minutes:**  
  
**Client Meeting - 1 on May 01 2024**

**1 Who was present?**

- Sai Srikar Miriyala (S567160)

- Sowmya Reddy Allugari (S565488)

- Lakshmi Pravallika Bhupathi (S564196)

- Nishchala Namburi (S564199)

- Yaswanth Kanakala (S567546)

- Udaykiranreddy Devarapally (S567161)

**2 Meeting agenda**

- Start with a team introduction outlining each member's role.

- Let the client representative introduce themselves and their role within the Travel Buddy application.

- Provide a concise overview of the meeting agenda and its objectives.

- Discuss the technology stack for the web application.

- Talk about integrating with a government tool for verification purposes.

- Discuss the project percentage to be completed by the end of this semester.

- Schedule mid and final project presentations.

**3 Specific questions asked (and who asked them)**

a) Sai Srikar Miriyala-

- Which current systems or platforms require integration with the new solution?

- How does the new system simplify user registration and verify eligibility?

- What requirements does this project aim to fulfill?

b) Sowmya Reddy Allugari-

- What specific problems or difficulties do you want the app to solve?

- Do you have any specific technical requirements or preferences for the development of the app?

c) Lakshmi Pravallika Bhupathi -

- What are the primary goals of this project?

- What do you hope to achieve in this meeting?

d) Nishchala Namburi -

- What features are you incorporating?

- What difficulties do existing travel applications encounter?

e) Yaswanth Kanakala -

- Which backend tools can be used?

- What is your timeline for this project?

- How will we plan for effective communication and collaboration during development and implementation?

f) Udaykiranreddy Devarapally -

- What requirements are needed?

- Which needs is this project addressing?

**4 Specific answers given (and who gave them)**

- During the client meeting, Prof. Mark Chai, our project client, addressed numerous queries regarding project implementation.

- He provided responses to the questions posed in the initial meeting.

- What functions are you implementing?

- What is your timeline?

- What are your requirements?

- What challenges do the current travel apps face?

**5 Action items: what have you decided you will show the client in your next meeting**

- During the upcoming client meeting, we'll delve into defining the problem statement in greater detail.

- Develop an initial project plan detailing the project's scope, essential features, schedule, and financial allocation.

- Conduct a user persona and needs analysis to demonstrate our comprehension of various user categories and their distinct requirements.

- User personas and needs analysis: Show how we understand the different users and their specific needs.

**6 When is the next meeting (no more than two weeks away)**

- The upcoming client meeting is set for 05/22/2024.