Software Requirements Specification

for

Farrin – Travel Companion Application

Version 1.0

Prepared by

GROUP: <undefined>

Michael Bryan	**620151954**	**michael.bryan02@mymona.uwi.edu**
Kaela Calvert	**620154861**	**kaela.calvert@mymona.uwi.edu**
Joel Plummer	**620146814**	**joel.plummer@mymona.uwi.edu**
Jahnea Francis	**620155185**	**jahnea.francis@mymona.uwi.edu**

Course Instructor: **Dr. Carl Beckford**

Course: **SWEN3920 – Group Project**

Supervisor: ***Dr. Phillipa Bennet***

Date: **tbd**

TABLE OF CONTENTS

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Michael Bryan Kaela Calvert Joel Plummer Jahnea Francis	Initial Specification	02/03/2025

1 Overall Description

1.1 Product Context and Need

The Farrin App was developed in response to the complexities and challenges travellers worldwide face when planning international trips. As globalisation continues facilitating travel across borders, the demand for efficient and comprehensive trip-planning tools has grown. Traditionally, travelers have had to navigate multiple websites and resources to gather essential information such as flight options, accommodation availability, visa requirements, and travel advisories. This fragmented approach can be overwhelming and time-consuming, often deterring individuals from pursuing their travel plans.

The Farrin App is created by a team committed to revolutionising the travel planning experience for individuals globally. The organisation's mission is to provide a centralised, user-friendly platform that consolidates all necessary travel information and resources into one convenient location. By catering to travelers from any country of origin, the app aims to simplify planning international trips, regardless of the traveler's starting point.

1.2 Product Functionality

The Farrin App is designed to offer a comprehensive suite of functionalities that streamline the international trip planning process for travelers worldwide. At a high level, the major functions of the system include:

- **User Management:** Facilitates user registration, login, and profile management, allowing travelers to maintain personal information and travel preferences.
- **Trip Planning:** This tool enables users to select destinations, plan solo or group trips, and create detailed itineraries that include activities, accommodations, and transportation.
- **Destination Information:** Provides comprehensive data on selected destinations, including visa requirements, travel advisories, and climate information.
- **Transportation and Accommodation Booking:** Aggregates and displays real-time options for flights, cruises, and accommodations, allowing users to filter based on their preferences and directing them to websites where they can make the actual bookings.
- **Personalised Recommendations:** A machine learning model that utilises user data and preferences to generate tailored destination and activity recommendations and budget estimates.
- **Travel Goals and Sharing:** This feature allows users to set and track travel goals, save locations with planned timelines, and share itineraries with friends or family members.
- **Integration with External Systems:** Connects with various external APIs to fetch real-time data on flights, accommodations, weather, and currency exchange rates, ensuring up-to-date information for users.

These high-level functions are designed to provide a centralised and user-friendly platform that simplifies all aspects of international trip planning, empowering travelers to plan their journeys efficiently and confidently.

1.3 Stakeholders and Users Characteristics

Current Key Stakeholders:

- **Travelers:** The primary users of The Farrin App, who will utilise the system to plan and manage their international trips.
- **App Developers:** The team responsible for creating, maintaining, and updating The Farrin App.
- **API Providers:** External systems that supply real-time data to The Farrin App, such as flight booking systems, hotel booking systems, and weather data providers.

Possible Future Stakeholders, we want to leave room for:

- **Travel Agencies and Partners:** Organisations that may integrate with The Farrin App to provide services such as flight and accommodation bookings.
- **Investors and Sponsors:** Individuals or organisations that fund The Farrin App's further development and operation.

We have ideated five (5) categories users of The Farrin App could fall in, each with distinct behaviours/characteristics. See below:

1. **Casual Travelers**
    - **Frequency of Use:** Occasional
    - **Subset of Product Functions Used:** Primarily use the trip planning, destination information, and transportation/accommodation booking features.
    - **Technical Expertise:** Basic to moderate; comfortable with using mobile apps.
    - **Experience:** May have limited experience with international travel planning.
    - **Importance:** High; these users form the app's core user base and are essential for its success.
2. **Frequent Travelers**
    - **Frequency of Use:** Regular
    - **Subset of Product Functions Used:** Utilise all features, including personalised recommendations, travel goals, and sharing functionalities.
    - **Technical Expertise:** Moderate to advanced; adept at using technology for travel planning.
    - **Experience:** Experienced in international travel and planning.

- **Importance:** High; these users provide valuable feedback and help refine the app's features.

3. **Group Travel Organisers**

    - **Frequency of Use:** Varies, often project-based or corporate-based.
    - **Subset of Product Functions Used:** Focus on group trip planning, itinerary sharing, and coordination features.
    - **Technical Expertise:** Moderate; comfortable with managing group activities through apps.
    - **Experience:** May have experience in organising group trips.
    - **Importance:** Medium; these users help expand the app's utility for group travel.

4. **Travel Enthusiasts and Bloggers**

    - **Frequency of Use:** Regular to frequent
    - **Subset of Product Functions Used:** Utilise all features, focusing on destination information, recommendations, and sharing capabilities.
    - **Technical Expertise:** Advanced; skilled in using technology and sharing content online.
    - **Experience:** Highly experienced in travel and content creation.
    - **Importance:** Medium; these users can help promote the app through their networks and content.

5. **Business Travelers**
    - **Frequency of Use:** Regular
    - **Subset of Product Functions Used:** Focus on efficient trip planning, transportation, and accommodation booking.
    - **Technical Expertise:** Moderate to advanced; accustomed to using technology for business purposes.
    - **Experience:** Experienced in business travel and planning.
    - **Importance:** Medium; these users represent a significant travel market segment

.

The most important users for The Farrin App app are Casual and Frequent Travelers, as they form the core user base essential for the app's success. Casual Travelers represent the largest potential market, and Frequent Travelers provide valuable feedback to refine the app's features, such as our recommendation model. While Group Travel Organisers, Travel Enthusiasts and Bloggers, and Business Travelers enhance the app's utility and reach, they are less critical than the core user groups focused on simplifying international travel planning for the general traveler.

1.4 Operating Environment

The Farrin App web application is designed to operate on various modern web browsers, including, but not limited to, Google Chrome, Mosilla Firefox, Microsoft Edge, and Safari, ensuring broad user accessibility. The website will be optimised for desktop and laptop computers and mobile devices to provide a consistent user experience across different screen sises. Users will need a stable internet connection through Wi-Fi or mobile data to access the website and utilise its features, which include real-time data retrieval from external APIs such as flight and accommodation booking systems, weather data providers, and currency exchange rate services. The Farrin App website will be hosted on cloud infrastructure to ensure scalability, reliability, and global accessibility, allowing users to plan their trips efficiently from any location with internet access.

1.5 Design and Implementation Constraints

- **Real-Time Data Integration:** The website's functionality relies heavily on real-time data from external APIs, such as flight and accommodation booking systems, weather data providers, and currency exchange rate services. This constraint necessitates using specific communication protocols and APIs and considering the API's rate-limiting and relevance, i.e., how up to date its data is, which may limit the developers' flexibility in choosing alternative data sources or integration methods. Additionally, it requires robust error handling and fallback mechanisms to ensure the website remains functional even when external services are unavailable.

- **Security and Privacy Requirements:** Given the sensitive nature of user data, including personal information and travel preferences, The Farrin App website must adhere to strict security and privacy standards. Additionally, to improve the recommendation model, the system will require users to allow the recording of their interactions with the website. Users can choose which parameters can be used in their recommendations, ensuring transparency and control over their data. This constraint limits the developers' options regarding data storage and transmission methods, requiring the implementation of encryption, secure communication protocols (such as HTTPS), and compliance with data protection regulations like GDPR. These security measures may also impact the website's performance and development timeline, as additional time and resources will be needed to ensure compliance and protect user data.

1.6 Assumptions and Dependencies

Assumptions

1. **User Engagement:** It is assumed that users will actively engage with The Farrin App website, providing feedback and allowing the recording of their interactions to improve the recommendation model. If user engagement is lower than expected, the effectiveness of the recommendation system may be compromised.

2. **API Reliability:** The Farrin App website assumes that the external APIs for flight and accommodation bookings, weather data, and currency exchange rates will be reliable and

maximally available. Significant downtime or unreliability of these APIs could affect the website's functionality and user experience.

3. **Internet Connectivity:** The Farrin App website is assumed to be accessible via stable internet connectivity. If users frequently experience connectivity issues, this could impact the website's usability and effectiveness.

4. **Browser Support:** The website assumes that most users will use supported browsers (Google Chrome, Mosilla Firefox, Microsoft Edge, and Safari) and keep them updated to the latest versions. If a significant number of users use unsupported or outdated browsers, it may limit the website's functionality and user experience.

Dependencies

1. **Third-Party APIs:** The Farrin App website uses third-party APIs for real-time flight data, accommodations, weather, and currency exchange rates. Any changes in these APIs, such as updates, discontinuation, or changes in terms of service, could require modifications to the website's integration and functionality.

2. **Cloud Infrastructure:** The website's performance and scalability depend on the reliability and performance of the cloud infrastructure provider. Any issues with the cloud service could impact the website's availability and performance.

3. **Security and Compliance Tools:** Implementing security measures and compliance with data protection regulations depends on the availability and effectiveness of security tools and services. Any limitations or changes in these tools could affect the website's ability to protect user data and comply with legal requirements.

2 Specific Requirements

2.1 External Interface Requirements

2.1.1 Hardware Interfaces

The Farrin App website does not have direct hardware interfaces as it is a web-based application. However, it is designed to be accessible on various devices, including desktop computers, laptops, smartphones, and tablets. The website will interact with these devices through standard web browsers, ensuring compatibility with common input devices such as keyboards, mice, touchscreens, and trackpads. No special libraries are required for hardware communication, as the website will rely on the browser's capabilities to handle user input and display output.

2.1.2 Software Interfaces

The Farrin App website will interface with the operating systems of the devices on which it is accessed through web browsers. The supported operating systems include Windows (version 10 and above), macOS (version 10.14 and above), iOS (version 12 and above), and Android (version 8.0 and above). The website will use standard web technologies to interact with these operating systems via the browser. No specific software libraries or tools are required for this interface, as the website will leverage the browser's built-in capabilities to render pages and handle user interactions.

2.1.3 Communications Interfaces

The Farrin App website will utilise several communication standards to ensure seamless data exchange and user interaction. The primary communication protocol will be HTTP/HTTPS for web browsing, ensuring secure data transmission between the user's browser and the website's server. The website will use RESTful APIs over HTTPS to fetch real-time data for integration with external APIs, such as those for flight and accommodation bookings, weather data, and currency exchange rates. Email communication will be facilitated through SMTP to send users confirmation links and notifications. Data will be encrypted using industry-standard encryption protocols like TLS to protect user information during transmission. The website will also support WebSocket for real-time updates, ensuring users receive timely information on availability and pricing changes.

2.2 Functional Requirements

Requirement ID: FR-1.1.1 - User Registration
Use Case: UC-1 Register
Rationale: Users must be able to create an account to access system features.
User Requirement: The system must allow new users to register using email and password.
System Requirements:

1. The system must provide a registration form for users to enter their email and password.
2. The system must enforce password security standards.
3. The system must store user credentials securely in the database, i.e. hashing.

Acceptance Criteria:

4. Users can successfully register with an authenticated email and password.
5. Invalid emails or weak passwords are rejected. Password criteria: mixture of upper- and lower-case digits, at least one digit and at least one special symbol (!@#$%^&*).

Relates to/Dependencies: UC-1: Register
Priority: High
Team Owner: Kaela Calvert

Requirement ID: FR-1.1.2 - Email Verification
Use Case: UC-1 Register
Rationale: Email verification ensures that the registered user is the owner (who has access) to the email provided while registering.
User Requirement: The system must verify user email addresses through a confirmation link sent to the email.
System Requirements:

1. The system must generate and send a unique URL to the provided email.
2. The system must send the confirmation link via the provided email.

Acceptance Criteria:

3. Users receive a verification email within 5 minutes of registration.
4. A user successfully verifies email access or ownership by entering the verification code.
5. A travel profile is generated and activated, and the user is redirected to the login page.

Relates to/Dependencies: UC-1: Register
Priority: High
Team Owner: Michael Bryan

Requirement ID: FR-1.1.3 - User Login
Use Case: UC-2 Login
Rationale: A registered user wants to access their account.

User Requirement: The system must allow registered users to log in using email and password.
System Requirements:

1. The system must provide a login form for users to enter their email and password.
2. The system must authenticate users by checking credentials against stored data.
3. The system must restrict access for users with invalid credentials.

Acceptance Criteria:

4. Users with valid credentials can log in successfully.
5. Users with invalid credentials receive an error message.

Relates to/Dependencies: UC-2: Login
Priority: High
Team Owner: Joel Plummer

Requirement ID: FR-1.1.4 - Password Reset
Use Case: UC-3: Reset Password
Rationale: Users should be able to regain access to their accounts if they forget their passwords.
User Requirement: The system must allow users to reset their passwords through email verification.
System Requirements:

1. The system must provide a "Forgot Password" option on the login page.
2. The system must generate and send an email with a password reset code.
3. The system displays a message to direct the user to check their email for a reset code (4-6 characters).
4. The system must allow users to set a new password after verification.

Acceptance Criteria:

5. Users receive a password reset email within 5 minutes of the request.
6. Users can successfully update their passwords and log in.

Relates to/Dependencies: UC-3: Reset Password
Priority: High
Team Owner: Jahnea Francis

Requirement ID: FR-1.2.1 - Travel Profile Management
Use Case: UC-4: Edit Travel Profile
Rationale: Users need a way to store and manage their travel-related data efficiently.
User Requirement: The system must create a travel profile and allow users to maintain personal information such as name, email, date of birth, and interests.
System Requirements:

1. The system must provide a form to input and update user profile details.

2. The system must validate the entered data before saving.
3. The system must store user profile data securely in the database.

Acceptance Criteria:

4. The user can create a profile and update details successfully.
5. Updated data is stored correctly in the system 100% of the time.

Relates to/Dependencies: UC-4: Edit Travel Profile
Priority: High
Team Owner: Joel Plummer

Requirement ID: FR-1.2.2 - User Preference Questionnaire
Use Case: UC-1 Register
Rationale: A detailed questionnaire helps tailor travel recommendations.
User Requirement: Upon registration and the first-ever login, the system must administer a detailed questionnaire to gain insights into user preferences, demographics, past travel destinations, and possible destinations of interest.
System Requirements:

1. The system must display a multi-step questionnaire upon first user login.
2. The system must validate and store the responses in the user's profile.
3. The system must use the responses to enhance personalised recommendations.

Acceptance Criteria:

4. Users complete the questionnaire without skipping the required fields.
5. Stored responses influence recommendation accuracy.

Relates to/Dependencies: UC-1: Register
Priority: High
Team Owner: Kaela Calvert

Requirement ID: FR-1.2.3 - Travel Preferences Management
Use Case: UC-5: Manage Travel Preferences
Rationale: Users should be able to customise their preferred destinations and budget.
User Requirement: The system must store user travel preferences, including a list of preferred destinations, budget ranges, and travel styles.
System Requirements:

1. The system must provide options to input and modify travel preferences.
2. The system must allow users to select from predefined budget ranges.
3. The system must store and retrieve preferences for future recommendations.

Acceptance Criteria:

4. Users successfully save and update their preferences.
5. Preferences are reflected in personalised recommendations.

Relates to/Dependencies: UC-6: Manage Travel Preferences
Priority: High
Team Owner: Michael Bryan

Requirement ID: FR-1.2.4 - Past Trips Management
Use Case: UC-6: Manage Past Trips List
Rationale: Users should be able to maintain a record of their travel history.
User Requirement: The system must allow users to add, edit, and view past travel experiences.
System Requirements:

1. The system must provide an interface for recording past trip details.
2. The system must store past trips with dates, destinations, and activities.
3. The system must use past trip data to enhance future recommendations.

Acceptance Criteria:

4. Users can add and edit past trip information successfully.
5. Past trip information contributes to recommendation accuracy.

Relates to/Dependencies: UC-7: Manage Past Trips List
Priority: Medium
Team Owner: Jahnea Francis

Requirement ID: FR-1.2.5 - Travel Bucket List and Goals
Use Case: UC-8: Manage Travel Bucket List and Travel Goals
Rationale: Users need to track future travel aspirations.
User Requirement: The system must allow users to create and manage travel goals and bucket list destinations.
System Requirements:

1. The system must provide an interface for managing travel goals.
2. The system must allow priorities and timelines for bucket list destinations.
3. The system must track progress toward travel goals.

Acceptance Criteria:

4. Users can add, edit, and prioritise travel goals.
5. The system displays appropriate progress indicators for goals.

Relates to/Dependencies: UC-8: Manage Travel Bucket List and Travel Goals
Priority: Medium
Team Owner: Joel Plummer

Requirement ID: FR-1.2.6 - Data Access Control (DAC) Mechanism
Use Case: UC-9: Discretion Data Sharing Preference for Model
Rationale: Users should have control over how their data is used for recommendations.
User Requirement: The system must implement a DAC mechanism that allows users to control whether the Farrin model can use their interaction patterns and preferences to improve the recommendation model.
System Requirements:

1. The system must provide an interface where users can opt in or opt out of data collection.
2. The system must store user consent settings securely and reliably.
3. The system must adjust recommendation algorithms based on user consent settings.

Acceptance Criteria:

4. Users can successfully modify data-sharing preferences.
5. The system respects user consent choices 100% of the time.

Relates to/Dependencies: UC-9: Discretion Data Sharing Preference for Model
Priority: High
Team Owner: Michael Bryan

Requirement ID: FR-2.1.1 - Trip Creation
Use Case: UC-10: Create Trip
Rationale: Users need a starting point for planning their travels.
User Requirement: The system must allow users to create new trip plans with basic details.
System Requirements:

1. The system must provide an interface for initiating new trips.
2. The system must allow users to input basic trip parameters (destination, dates, trip type).
3. The system must save trip drafts for further planning.

Acceptance Criteria:

4. Users can successfully create a new trip.
5. Created trips are available for detailed planning.

Relates to/Dependencies: UC-10: Create Trip
Priority: High
Team Owner: Kaela Calvert

Requirement ID: FR-2.1.2 - Travel Documentation Verification
Use Case: UC-11: Verify Travel Requirements
Rationale: Users need to know travel documentation requirements before planning a trip.
User Requirement: The system must verify and display the travel documentation (visas and/or passports) requirements for the user's specified country of citizenship to the chosen destination.
System Requirements:

1. The system must allow a user to choose which of their home countries they want to be the origin or starting point of a trip, i.e., countries to which they have citizenship.
2. The system must retrieve travel documentation requirements from an external API.
3. The system must display the requirements from the origin country to the country of destination in an easy-to-read format.

Acceptance Criteria:

4. Users receive accurate documentation requirements based on their nationality.
5. Users can confirm they meet the requirements before proceeding.

Relates to/Dependencies: UC-11: Verify Travel Requirements
Priority: High
Team Owner: Jahnea Francis

Requirement ID: FR-2.1.3 - Travel Advisories
Use Case: UC-11: Verify Travel Requirements
Rationale: Users must know their destination's safety and health advisories and requisite travel documentation.
User Requirement: The system must provide travel advisories for the selected destination, including health and required vaccination information.
System Requirements:

1. The system must retrieve travel advisories from a reliable source.
2. The system must display advisory information to users.
3. The system must allow users to confirm they meet the requirements.

Acceptance Criteria:

4. Users can view travel advisories before proceeding.
5. The advisory information is up-to-date and accurate.

Relates to/Dependencies: UC-11: Verify Travel Requirements
Priority: High
Team Owner: Michael Bryan

Requirement ID: FR-2.1.4 - View Created Trips
Use Case: UC-12: View Created Trips
Rationale: Users need to access and manage their existing trips.
User Requirement: The system must allow users to view, filter, and access all trips they have created.
System Requirements:

1. The system must retrieve and display all user trips.
2. The system must organise trips by status (upcoming, in-progress, past).
3. The system must allow users to select trips for viewing or editing.

Acceptance Criteria:

4. Users can view a complete list of their trips.
5. Users can access detailed trip information from the list.

Relates to/Dependencies: UC-12: View Created Trips
Priority: High
Team Owner: Joel Plummer

Requirement ID: FR-2.1.5 - Detailed Trip Planning
Use Case: UC-13: Plan a Trip
Rationale: Users need to develop comprehensive travel plans.
User Requirement: The system must allow users to create detailed itineraries for their trips.
System Requirements:

1. The system must provide interfaces for adding all trip components.
2. The system must support solo and group trip planning.
3. The system must save all trip details securely.

Acceptance Criteria:

4. Users can successfully create comprehensive trip plans.
5. Trip plans include all required travel elements.

Relates to/Dependencies: UC-13: Plan a Trip
Priority: High
Team Owner: Kaela Calvert

Requirement ID: FR-2.1.6 - Solo and Group Trip Planning
Use Case: UC-13: Plan a Trip
Rationale: Users should be able to plan solo and group trips.
User Requirement: The system must allow users to plan solo and group trips and provide options to invite other registered users to coordinate travel plans.
System Requirements:

1. The system must provide an option between solo and group trip planning.
2. The system must allow users to invite registered members to a group trip.
3. The system must allow invited users to accept or decline the invitation.

Acceptance Criteria:

4. Users can create a solo or group trip successfully.
5. Group members receive invitations and can join the trip.

Relates to/Dependencies: UC-13: Plan a Trip
Priority: High
Team Owner: Jahnea Francis

Requirement ID: FR-2.1.7 - Transportation Booking Integration
Use Case: UC-14: Add Transport to Trip Itinerary
Rationale: Users need to find and add transportation to their itineraries.
User Requirement: The system must allow users to search, compare, and add transportation options to their trip plans.
System Requirements:

1. The system must integrate with transportation APIs.
2. The system must provide search and filter functionality.
3. The system must allow users to add booked transportation to their itineraries.

Acceptance Criteria:

4. Users can search for transportation options successfully.
5. Transportation details are accurately added to itineraries.

Relates to/Dependencies: UC-14: **Manage Transportation in Trip Itinerary**

Priority: High
Team Owner: Michael Bryan

Requirement ID: FR-2.1.8 - Accommodation Booking Integration
Use Case: UC-15: **Manage Accommodation in Trip Itinerary**

Rationale: Users need to find and add accommodations to their itineraries.
User Requirement: The system must allow users to search, compare, and add accommodation options to their trip plans.
System Requirements:

1. The system must integrate with accommodation APIs.
2. The system must provide search and filter functionality.
3. The system must allow users to add booked accommodations to their itineraries.

Acceptance Criteria:

4. Users can search for accommodation options successfully.
5. Accommodation details are accurately added to itineraries.

Relates to/Dependencies: UC-15: **Manage Accommodation in Trip Itinerary**

Priority: High
Team Owner: Kaela Calvert

Requirement ID: FR-2.1.9 – Event Management
Use Case: UC-16: **Manage Events in Trip Itinerary**
Rationale: Users should be able to organise activities within their solo or group travel itineraries.
User Requirement: The system must allow users to add events (activities, reservations, etc.) to their trip itinerary and coordinate them with invited group members when applicable.
System Requirements:

1. The system must allow users to add events to a trip itinerary.
2. The system must associate events with a specific trip (solo or group).
3. If the trip is a group trip, all invited and accepted users must see the shared event updates.
4. The system must allow users with edit permissions to modify or remove events from the itinerary.

Acceptance Criteria:

5. Users can successfully add an event to a solo or group trip itinerary.
6. Events are correctly displayed on the shared trip view(itinerary) for all group participants.
7. Group members receive notifications when a new event is added.

Relates to/Dependencies: UC-16: **Manage Events in Trip Itinerary**
Priority: High
Team Owner: Joel Plummer

Requirement ID: FR-2.1.10 - Weather Information Integration
Use Case: UC-16: Add Weather Info to Trip Itinerary
Rationale: Users need weather information to plan appropriately for their trip.
User Requirement: The system must provide climate and weather information for the selected destination.
System Requirements:

1. The system must retrieve weather data from an external provider.
2. The system must display historical climate data and current weather updates.
3. The system must allow users to add weather info to their itinerary.

Acceptance Criteria:

4. Users can view accurate climate and weather data.
5. Weather details can be added to the itinerary successfully.

Relates to/Dependencies: UC-17: **Manage Weather Info in Trip Itinerary**

Priority: Medium
Team Owner: Jahnea Francis

Requirement ID: FR-3.1.1 - Personalised Recommendations
Use Case: UC-18: View Feed
Rationale: Personalised recommendations help users discover destinations that align with their preferences.
User Requirement: The system must generate personalised destination recommendations based on the following criteria:

- User preferences, Travel history, and Past user interactions (e.g., locations viewed or liked) - 40%
- Demographic Popularity - 20%
- Budget alignment - 15%
- Climate preferences - 15%
- Convenience (including flight availability and ease of public and rideshare commute at the destination) - 10%

System Requirements:

1. The system will analyse user preferences, travel history, and interactions to generate recommendations.
2. The system must weigh each factor according to the specified distribution.
3. The system must display recommended destinations in an easily accessible format.

Acceptance Criteria:

4. Users receive tailored recommendations based on stored travel preferences.
5. The recommendation algorithm updates dynamically with new user interactions.

Relates to/Dependencies: UC-18: View Feed
Priority: High
Team Owner: Michael Bryan

Requirement ID: FR-3.1.2 - Activity Recommendations
Use Case: UC-18: View Feed
Rationale: Users should receive suggestions for activities at selected destinations.
User Requirement: The system must recommend activities based on user preferences and destination data.
System Requirements:

1. The system must analyse user preferences and match them with activity options.
2. The system must retrieve relevant activity data from an external provider.
3. The system must display suggested activities in a user-friendly format.

Acceptance Criteria:

4. Users receive activity suggestions that align with their preferences.
5. Recommendations update dynamically based on user interactions.

Relates to/Dependencies: UC-18: View Feed
Priority: Medium
Team Owner: Kaela Calvert

Requirement ID: FR-3.1.3 - Budget Estimation
Use Case: UC-18: View Feed
Rationale: Users need estimated travel costs to make informed decisions.
User Requirement: The system must provide budget estimates based on selected destinations and travel styles.
System Requirements:

1. The system must calculate estimated travel costs using historical and real-time data.
2. The system must allow users to modify budget parameters for more accurate estimates.
3. The system must display budget recommendations in an organised format.

Acceptance Criteria:

4. Users can view estimated budgets based on selected travel styles.
5. Estimates adjust dynamically when modifying parameters.

Relates to/Dependencies: UC-18: View Feed
Priority: Medium
Team Owner: Joel Plummer

Requirement ID: FR-3.1.4 - Sentiment Analysis of Destinations
Use Case: UC-18: View Feed
Rationale: Users should have an overview of traveller opinions about destinations.
User Requirement: The system must analyse user reviews to generate a sentiment summary for destinations.
System Requirements:

1. The system must collect and analyse user reviews from various sources.
2. The system must generate sentiment scores and summaries for each destination.
3. The system must display sentiment results in an easy-to-understand format.

Acceptance Criteria:

4. Users can view aggregated sentiment summaries for destinations.
5. Sentiment scores accurately reflect overall user reviews.

Relates to/Dependencies: UC-18: View Feed
Priority: Medium
Team Owner: Jahnea Francis

Requirement ID: FR-3.2.1 - Interest-Based Destination Browsing
Use Case: UC-19: Filter Feed by Interest

Rationale: Users should be able to find destinations based on specific interests.
User Requirement: The system must allow users to browse destinations based on the following specific interests:

- Adventure: Activities such as hiking, kayaking, and zip-lining.
- Relaxation: Spa resorts, beachfront hotels, and scenic retreats.
- Cultural experiences: Historical landmarks, museums, and local festivals.
- Nature: National parks, wildlife reserves, and scenic landscapes.

System Requirements:

1. The system must categorise destinations into predefined interest groups.
2. The system must allow users to browse and filter destinations by interest.
3. The system must provide relevant activity recommendations within each category.

Acceptance Criteria:

4. Users can browse destinations based on their selected interests.
5. Interest-based categories display accurate destination options.

Relates to/Dependencies: UC-19: Filter Feed by Interest
Priority: High
Team Owner: Michael Bryan

Requirement ID: FR-3.3.1 - View Travel Bucket List
Use Case: UC-8: **Manage Travel Bucket List and Travel Goals**
Rationale: Users need to view and manage their saved destinations.
User Requirement: The system must provide a dedicated view for bucket list destinations.
System Requirements:

1. The system must retrieve and display all bucket list destinations.
2. The system must provide filtering and sorting options.
3. The system must allow users to initiate planning from the bucket list.

Acceptance Criteria:

4. Users can view their complete bucket list.
5. Users can successfully interact with bucket list items.

Relates to/Dependencies: UC-8: **Manage Travel Bucket List and Travel Goals**
Priority: Medium
Team Owner: Kaela Calvert

Requirement ID: FR-3.3.2 - View Travel Goals
Use Case: UC-8: **Manage Travel Bucket List and Travel Goals**
Rationale: Users need to track progress toward their travel goals.

User Requirement: The system must provide a dedicated view for travel goals.
System Requirements:

1. The system must retrieve and display all travel goals.
2. The system must show progress indicators for goals.
3. The system must allow users to update goal status.

Acceptance Criteria:

4. Users can view their complete list of travel goals.
5. Goal progress is accurately displayed.

Relates to/Dependencies: UC-8: **Manage Travel Bucket List and Travel Goals**
Priority: Medium
Team Owner: Joel Plummer

Requirement ID: FR-5.1.1 - External System Integration
Use Case: Multiple Use Cases
Rationale: The system requires external data for flights, accommodations, and other travel services.
User Requirement: The system must integrate with multiple external providers for travel services.
System Requirements:

1. The system must connect with flight booking systems via APIs.
2. The system must retrieve hotel availability from integrated hotel booking systems.
3. The system must fetch weather and currency exchange data from external sources.

Acceptance Criteria:

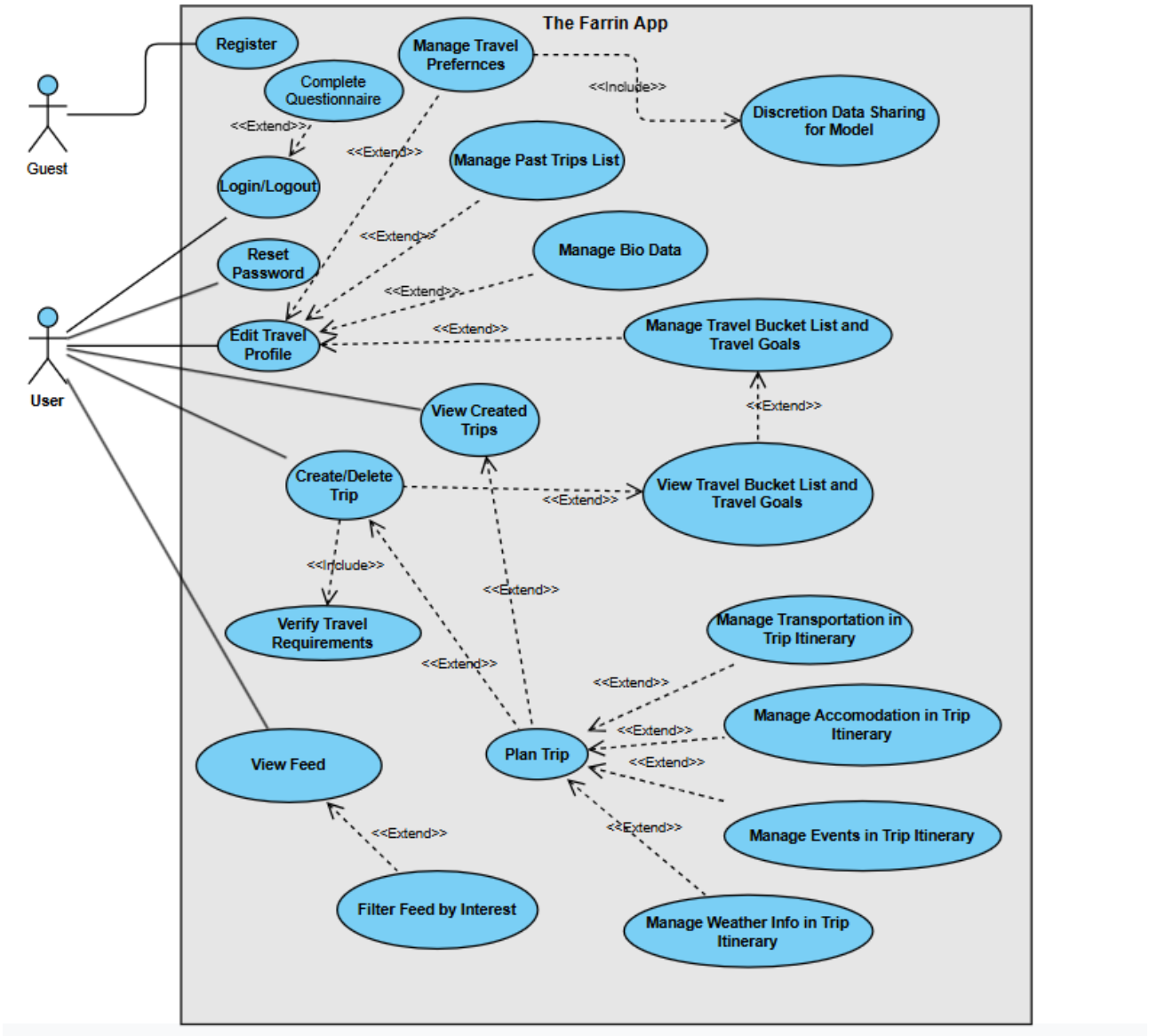
4. Users receive accurate, real-time data from external providers.
5. API integrations operate with minimal downtime.

Relates to/Dependencies: UC-11, UC-14, UC-15, UC-16, UC-17
Priority: High
Team Owner: Jahnea Francis

2.3 Behaviour Requirements

2.3.1 Use Case View

Use Case Diagram:



Use Case Narratives:

Authentication and Registration

Fully Dressed Use Case: Register

Use Case: UC-1 Register

Primary Actor: Guest

Stakeholders and Interests:

- Guest: Wants to create an account to access travel planning features with minimal effort and security concerns.
- App Developers: Want to collect accurate user information while ensuring data security and usability.
- System Administrators: Want to maintain data integrity and security compliance.

Preconditions: The guest has internet access and access to the Farrin App.

Success Guarantee (Postconditions):

- The account is created and activated
- The user profile is established with initial preferences
- The user's email is verified
- The user can log into the system

Main Success Scenario (for Basic Flow):

1. The guest selects the "Register" option on the app's welcome screen.
2. The system presents a registration form requesting an email and password.
3. The guest enters an email address and creates a password according to the security criteria (mixture of upper and lower-case letters, at least one digit and at least one special symbol).
4. The system validates the email format and password strength.
5. The system generates and sends a verification link to the provided email.
6. The guest receives the email and clicks the verification link.
7. The system verifies the email and activates the account.
8. “Guest” becomes a “User”.
9. The system creates a new user travel profile and stores the user in the user repository.
10. The system redirects the user to the login page.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the last stable state.
3. User continues from where they left off.

*a2. The system cannot recover state:

1. The system signals an error to the User.
2. The user must restart the registration process.

3a. The user attempts to register with an email that is already in the system:

1. The system notifies the user that the email is already registered.
2. The system offers options to recover the password or use a different email.

3a. User enters a password that doesn't meet security requirements:

1. The system displays specific security criteria that aren't met.
2. The user corrects the password and continues.

4a. User submits form with invalid data:

1. The system highlights the invalid fields.
2. The user corrects the highlighted fields and resubmits.

5a. The system cannot send a verification email:

1. The system notifies the user of the issue.
2. The system offers an alternative verification method or to try again later.

6a. User doesn't receive verification email within 5 minutes:

1. User requests email resend.
2. The system regenerates the verification link and resends the email.

6b. Verification link expires before the user clicks it:

1. The system displays an expired link message.
2. The system offers to send a new verification link.
3. User requests a new link and completes verification.

Special Requirements:

- Registration process must be completed within 5 seconds (not including email verification).
- Interface must be responsive across devices (desktop, tablet, mobile).
- Password encryption must use industry-standard hashing algorithms.
- Email verification must support international email providers.

Technology and Data Variations List:

- Email verification can be done via link or code entry.
- Registration can be completed via social media accounts (future implementation).

Frequency of Occurrence: High during app launch periods; moderate ongoing as new users discover the platform.

Open Issues:

- How much data control should users have from the beginning?

- What is the retention policy for user registration data?
- Should we implement two-factor authentication for registration?

The Two-Column Variation

Use Case UC1: Register

Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user selects the "Register" option on the app's welcome screen.	
	2. The system presents a registration form requesting email and password.
3. The user enters an email address and creates a password according to the security criteria.	
	4. The system validates the email format and password strength.
	5. The system generates and sends a verification link to the provided email.
6. The user receives the email and clicks the verification link.	
	7. The system verifies the email and activates the account.

Fully Dressed Use Case: Login/Logout

Use Case: UC-2 Login/Logout

Primary Actor: User (Traveller)

Stakeholders and Interests:

• **User:** Wants quick, secure access to their account and travel information, and the ability to safely log out

• **App Developers:** Want to ensure secure authentication/session management while maintaining a smooth user experience.

• **System Administrators:** Want to prevent unauthorised access while providing legitimate users with reliable access and proper session cleanup.

• **Security Team:** Want to protect user accounts from unauthorised access, ensure proper session termination, and manage data collection consent.

• **AI Development Team:** Want user consent for data collection to improve recommendation algorithms.

Preconditions:

- The user has already registered and has a verified account in the system.

Success Guarantee (Postconditions):

Login:

- The user is successfully authenticated
- The user gains access to their personalised travel profile
- The system logs the login activity
- The user's session is established with appropriate permissions.
- First-time users complete or skip the preference questionnaire with clear data sharing implications

Logout:

- The user's session is safely terminated
- All authentication tokens are invalidated
- The user is redirected to a public page
- Logout activity is logged for security

Main Success Scenario:

Login Flow:

1. The user navigates to the login screen.
2. The system presents the login form with fields for email and password.
3. The user enters their email address and password.
4. The system validates the credentials against stored data.
5. The system authenticates the user and establishes a session.
6. **[First-time login check]** The system checks if this is the user's first login.
7. The system redirects the user to their personalised dashboard.

Logout Flow:

8. The user selects the logout option from the navigation menu.
9. The system invalidates the user's session and authentication tokens.
10. The system logs the logout activity.
11. The system redirects the user to the login page with a confirmation message.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the login page.
3. The user attempts to log in again.

3a. The user is not registered:

1. The system offers a link to the registration page.
2. The user navigates to registration.

4a. The user enters incorrect credentials:

1. The system displays an error message.
2. The system allows the user to retry or reset their password.
3. The user corrects the credentials or chooses to reset the password.

4b. The user has forgotten their password:

1. The user selects the "Forgot Password" option.
2. The system redirects to the password reset flow (UC-3).

5a. The user's account is locked due to multiple failed attempts:

1. The system displays a message about the account being temporarily locked.
2. The system provides options for account recovery.
3. The user follows the account recovery process.

6a. NEW First-time login detected:

1. The system displays a welcome message and explains the travel preference questionnaire.
2. The system presents the questionnaire with a clear explanation of data usage.
3. The system explains that questionnaire completion enables personalised recommendations.
4. The system provides options: "Complete Questionnaire" or "Skip for Now."

6a1. The user chooses to complete the questionnaire:

1. The system presents the multi-step questionnaire (travel style, interests, budget, climate preferences).
2. The user completes the questionnaire.
3. The system saves responses and sets the data sharing preference to "true."
4. The system creates personalised preference profile.
5. The system proceeds to feed with personalised recommendations enabled.

6a2. The user chooses to skip the questionnaire:

1. The system displays a warning: "Skipping will result in generic, non-personalised recommendations."
2. The system asks for explicit confirmation: "Are you sure you want to skip? You can complete this later in Settings." 3a. User confirms skip:
    - The system sets the data sharing preference to "false"
    - The system proceeds to feed the view with generic recommendations
    - The system displays a notice about enabling personalisation in Settings 3b. User cancels skip:
    - Return to step 6a1

6a3. User partially completes questionnaire:

1. The system saves partial responses.
2. The system asks if the user wants to complete later or finish now.
3. If later: Proceed as in 6a2 (skip scenario)
4. If finished: Continue questionnaire from where left off.

7a. The system detects a login from an unrecognised device:

1. The system requests additional verification.
2. The user completes the additional verification steps.
3. The system proceeds with authentication.

9a. Logout fails due to system error:

1. The system displays an error message, but still invalidates the local session.
2. The system redirects to the login page with a warning about potential active sessions.
3. The user is advised to change the password if concerned about security.

Special Requirements:

- The login process must complete within 5 seconds
- The logout process must complete within 3 seconds
- The system must support secure password storage using industry-standard hashing algorithms
- The login interface must be responsive across different devices
- The system must implement rate limiting to prevent brute force attacks

- The system must maintain a login/logout activity log for security purposes
- Questionnaire must be dismissible, but with clear consequences explained
- Data sharing preferences must be explicit and easily changeable
- Generic recommendations must still be functional and useful

Technology and Data Variations List:

• Authentication can be performed via email/password or via social media integration (future) • The system supports various password complexity requirements • Session management can use cookies or tokens depending on the platform • **Questionnaire responses stored separately from core user data • Recommendation engine switches between personalised and generic modes based on data sharing preference**

Frequency of Occurrence:

Very high - login is one of the most frequently used functions. First-time login questionnaire occurs once per user.

Open Issues:

- Should we implement multi-factor authentication?
- What is the timeout period for inactive sessions?
- How many failed login attempts should trigger account lockout?
- Should we implement "Remember Me" functionality?
- What type of additional verification should be used for logins from new devices?
- Should users be able to skip the questionnaire indefinitely, or should we periodically prompt them?
- How often should we remind users about enabling personalisation if they opted out?
- Should partial questionnaire completion still enable some level of personalisation?
- What constitutes "generic" vs "personalised" recommendations? Should generic still use location/popular destinations?
- Should we allow users to complete the questionnaire in multiple sessions?
- How do we handle users who want to change their data sharing preference later?

Key Design Decisions:

1. **Questionnaire is optional** - Users can skip to maintain account accessibility
2. **Clear consequence communication** - Users understand what they're giving up by skipping
3. **Data sharing is explicit** - No hidden data collection
4. **Generic recommendations still functional** - App remains useful without personalisation
5. **Settings accessibility** - Users can always change their mind later
6. **Partial completion supported** - Reduces abandonment, allows gradual engagement

This approach balances user autonomy with business needs for personalisation data, while maintaining transparency about data usage.

The Two-Column Variation

Use Case UC2: Login

Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the login screen.	
	2. The system presents the login form with fields for email and password.
3. The user enters their email address and password.	
	4. The system validates the credentials against stored data.
	5. The system authenticates the user and establishes a session.
	6. The system redirects the user to their personalised dashboard.

Fully Dressed Use Case: Reset Password

Use Case: UC-3 Reset Password

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants to regain access to their account quickly and securely when they've forgotten their password
- App Developers: Want to provide a secure password recovery mechanism that maintains account integrity
- System Administrators: Want to ensure the password reset process doesn't introduce security vulnerabilities
- Customer Support: Wants to minimise account access issues that require manual intervention

Preconditions: The user has a registered account in the system.

Success Guarantee (Postconditions):

- The user's password is successfully reset
- The user can log in with the new password
- The old password no longer works
- The system logs the password change activity
- The user receives notification of the password change

Main Success Scenario:

1. The user selects the "Forgot Password" option on the login screen.
2. The system presents a form requesting the user's email address.
3. The user enters their registered email address.
4. The system validates that the email exists in the database.
5. The system generates a password reset code and sends it to the user's email.
6. The user receives the email and enters the reset code in the provided form.
7. The system verifies the reset code.
8. The system presents a form for the user to create a new password.
9. The user enters and confirms a new password.
10. The system validates that the new password meets security requirements.
11. The system updates the user's credentials with the new password.
12. The system notifies the user that the password has been successfully reset.
13. The system redirects the user to the login page.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to a stable state in the password reset flow.
3. The user continues from where they left off.

3a. The user enters an email that is not registered:

1. The system displays a generic message stating that if the email exists, a reset link will be sent.
2. The system does not send any email.
3. The user can try another email or navigate back to registration.

4a. The system cannot send the reset email:

1. The system notifies the user of a technical issue.
2. The system offers alternative recovery methods or suggests trying again later.

5a. The user doesn't receive the reset email within 5 minutes:

1. The user requests the code to be resent.
2. The system regenerates a new reset code and sends another email.

5b. The reset code expires before the user enters it:

1. The system informs the user that the code has expired.
2. The system offers to send a new reset code.
3. The user requests a new code and continues the process.

6a. The user enters a password that doesn't meet security requirements:

1. The system displays the specific security criteria that aren't met.
2. The user corrects the password and continues.

6b. The user enters a new password that matches their old password:

1. The system informs the user that the new password must be different from the previous one.
2. The user enters a different password.

Special Requirements:

- The password reset process must complete within 5 minutes (excluding user response time).
- Reset codes must expire after 30 minutes for security purposes.
- The system must not reveal whether an email exists in the database to prevent enumeration attacks.
- Password reset emails must be sent within 1 minute of the request.
- The system must follow secure password handling practices.

Technology and Data Variations List:

- Password reset can be done via email code or a secure link.
- Reset codes can be numeric or alphanumeric.
- The system supports various password complexity requirements.

Frequency of Occurrence: Moderate - common but less frequent than login.

Open Issues:

- Should we implement alternative verification methods (e.g., security questions)?
- What is the optimal expiry time for reset codes?
- Should we notify users of unsuccessful reset attempts?
- How many reset attempts should be allowed within a specific timeframe?
- Should we force a password change on the next login after a reset?

The Two-Column Variation

Use Case UC-3: Reset Password

Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	**System Responsibility**
1. The user selects the "Forgot Password" option on the login screen.	
	2. The system presents a form requesting the user's email address.
3. The user enters their registered email address.	
	4. The system validates that the email exists in the database.
	5. The system generates a password reset code and sends it to the user's email.
6. The user receives the email and enters the reset code in the provided form.	

Actor Action (or Intention)	**System Responsibility**
	7. The system verifies the reset code.
	8. The system presents a form for the user to create a new password.
9. The user enters and confirms a new password.	
	10. The system validates that the new password meets security requirements.
	11. The system updates the user's credentials with the new password.
	12. The system notifies the user that the password has been successfully reset.
	13. The system redirects the user to the login page.

Profile Management

Fully Dressed Use Case: Edit Travel Profile

Use Case: UC-4 Edit Travel Profile

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants to maintain up-to-date personal information for better service and recommendations
- App Developers: Want to collect accurate user data to improve personalisation and system functionality

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated

Success Guarantee (Postconditions):

- The user's profile information is updated in the system
- All changes are saved to the database
- The updated information is reflected in all relevant parts of the application
- The recommendation system incorporates the updated preferences

Main Success Scenario:

1. The user navigates to the profile management section of the application.
2. The system displays the current profile information.
3. The user can manage their complete travel profile by:

    a) Updating biographic data (name, email, date of birth)
    b) Configuring travel preferences (interests, climate preferences, budget constraints, travel styles, and data sharing settings)
    c) Maintaining their travel history (past destinations visited)
    d) Planning future travel (bucket list destinations and travel goals)

4. The user submits the changes.
5. The system validates the modified data.
6. The system saves the updated profile information to the database.
7. The system confirms a successful update to the user.

Extensions (or Alternative Flows): *a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the profile management page.
3. The user may need to re-enter unsaved changes.

3a. Some profile information cannot be retrieved:

1. The system displays the available information and indicates missing data.
2. The system allows the user to proceed with updating available fields.

4a. The user attempts to update with invalid data:

1. The system highlights the invalid fields.
2. The user corrects the highlighted fields and resubmits.

5a. The system detects validation errors:

1. The system highlights the fields with errors.
2. The system provides specific error messages.
3. The user corrects the errors and resubmits.

Special Requirements:

- Profile updates must be processed within 3 seconds.
- The interface must be responsive across devices (desktop, tablet, mobile).
- The system must maintain a history of profile changes for audit purposes.
- The system must comply with relevant data protection regulations.

Technology and Data Variations List:

- Profile images can be uploaded in JPEG, PNG, or GIF formats with sise limits.
- Date formats can be adjusted based on user location preferences.
- The system supports various language options for the interface.

Frequency of Occurrence: Moderate - users typically update their profiles occasionally.

Open Issues:

- What profile fields should be mandatory vs. optional?
- Should users receive a notification when their profile is updated?
- How should profile data be handled if a user requests account deletion?
- Should certain profile changes (e.g., email) require re-verification?

The Two-Column Variation

Use Case UC4: Edit Travel Profile
Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the profile management section of the application.	
	2. The system displays the current profile information.
3. The user modifies personal information such as name, email, date of birth, and interests.	
4. The user submits the changes.	
	5. The system validates the modified data.
	6. The system saves the updated profile information to the database.
	7. The system confirms a successful update to the user.

Fully Dressed Use Case: Manage Bio Data

Use Case: UC-5 Manage Bio Data

Primary Actor: User (Traveller)

Stakeholders and Interests:

- **User**: Wants to maintain accurate personal information for travel planning and account management
- **App Developers**: Want to ensure user data integrity and compliance with data protection regulations
- **Travel Partners (Prospective)**: Need accurate user information for booking validations and communications
- **Legal/Compliance Team**: Require proper data management practices and user consent for data changes

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated
- The user has appropriate permissions to modify their own bio data

Success Guarantee (Postconditions):

- The user's biographical information is updated in the system
- All changes are validated and saved to the database
- The updated information is reflected across all user interfaces
- Age is automatically recalculated based on the new date of birth (if modified)
- Audit log records the bio data changes for security purposes

Main Success Scenario:

Read/View Bio Data:

1. The user navigates to their profile's "Personal Information" or "Bio Data" section.
2. The system displays current bio data (first name, last name, email, date of birth, and calculated age).

Update Bio Data:

3. The user selects the "Edit Personal Information" option.
4. The system displays an editable form with the current bio data populated.
5. The user modifies any combination of:
    - **First Name** (text field)
    - **Last Name** (text field)
    - **Email Address** (email field with validation)
    - **Date of Birth** (date picker/calendar interface)
6. The user submits the changes.
7. The system validates the modified bio data.
8. The system saves the updated information to the database.
9. The system automatically recalculates and updates the derived age based on the new date of birth.
10. The system displays confirmation of a successful update.

11. The system logs the bio data modification for audit purposes.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the bio data management page.
3. Any unsaved changes are lost, and the user must re-enter modifications.

5a. The user attempts to enter an invalid first/last name:

1. The system validates that names contain only alphabetic characters and approved punctuation.
2. The system highlights invalid name fields with error messages.
3. The user corrects the names and continues.

5b. The user attempts to enter an invalid email address:

1. The system validates email format and checks for proper structure.
2. The system highlights the email field with the format error message.
3. The user corrects the email format and continues.

5c. The user attempts to enter an invalid date of birth:

1. The system validates that the date is not in the future and the user is at least 13 years old.
2. The system highlights the date field with the appropriate error message.
3. The user corrects the date and continues.

7a. Email address already exists in the system:

1. The system detects that the email is registered to another account.
2. The system displays the error message: "This email is already associated with another account."
3. The user must enter a different email address or contact support.

7b. The user enters a date of birth that makes them under 13:

1. The system displays an age restriction message.
2. The system prevents saving and requests a valid date of birth.
3. The user must enter a date that meets minimum age requirements.

8a. Database save operation fails:

1. The system displays error message about save failure.
2. The system retains the user's entered data in the form.

3. The user can retry saving or contact support.

9a. Age calculation results in significant change (e.g., 25 to 65):

1. The system displays an additional confirmation dialogue highlighting the age change.
2. The system asks the user to confirm the date of birth correctness.
3. User confirms or corrects the date of birth.

Special Requirements:

- First and last names must contain only alphabetic characters, spaces, hyphens, and apostrophes
- Email address must be validated for proper format and uniqueness
- Date of birth validation must ensure user meets minimum age requirements (13+ years)
- Age must be automatically recalculated and updated when date of birth changes
- All biodata changes must be logged for audit and security purposes
- Email changes may require additional verification (consider security implications)
- The interface must clearly show which fields are required vs. optional
- Changes must be processed within 3 seconds under normal system load

Technology and Data Variations List:

- Name fields can use text inputs with character validation
- Email field can include real-time format validation and domain verification
- Date of birth can use calendar widgets, dropdown menus, or date pickers
- Age display can be updated dynamically as the date of birth is modified
- Form validation can be client-side, server-side, or both for optimal user experience

Frequency of Occurrence:

Low to Moderate - users occasionally update personal information due to life changes, corrections, or account maintenance.

Open Issues:

- Should email address changes require additional verification (e.g., confirmation email to both old and new addresses)?
- How should the system handle name changes that might indicate identity verification needs?
- Should there be limits on how frequently bio data can be modified?
- Should the system maintain a history of previous bio data for audit purposes?
- How should the system handle special characters in names for international users?
- Should the date of birth change trigger additional identity verification for security?

The Two-Column Variation

Use Case UC-5: Manage Bio Data

Primary Actor: User (Traveller)

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the "Personal Information" section of their profile.	
	2. The system displays current bio data (first name, last name, email, date of birth, calculated age).
3. The user selects "Edit Personal Information."	
	4. The system displays an editable form with current bio data populated.
5. The user modifies first name, last name, email, and/or date of birth.	
6. The user submits the changes.	
	7. The system validates the modified bio data (format, uniqueness, age requirements).
	8. The system saves the updated information to the database.
	9. The system automatically recalculates and updates the derived age.
	10. The system displays confirmation of a successful update.
	11. The system logs the bio data modification for audit purposes.

Note: This use case specifically manages only biographical data (first name, last name, email, date of birth) and does not include travel preferences, which are handled in a separate "Manage Travel Preferences" use case (UC-6).

Fully Dressed Use Case: Manage Travel Preferences

Use Case: UC-6 Manage Travel Preferences

Primary Actor: User (Traveller)

Stakeholders and Interests:

- **User**: Wants to specify and update travel preferences for better recommendations and planning
- **App Developers**: Want to collect preference data to enhance personalisation algorithms
- **AI Model Development Team**: Need accurate preference data to improve the recommendation model accuracy

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated

Success Guarantee (Postconditions):

- The user's travel preferences are updated in the system with a new timestamp
- The recommendation system incorporates the updated preferences
- All preference changes are saved to the database
- The updated preferences are reflected in future trip recommendations

Main Success Scenario:

1. The user navigates to the travel preferences section of their profile.
2. The system displays current travel preferences (budget range, primary interest, travel style, climate preference, and data sharing setting).
3. The user modifies their preferences:
    - Updates **budget range** (lower and upper limits in integers)
    - Selects **primary interest** (Adventure, Relaxation, Cultural Experience, or Nature)
    - Chooses **primary travel style** (Casual, Frequent, Business, Enthusiast, or Organiser)
    - Sets **preferred climate** (Tropical, Dry, Continental, Polar, Mediterranean, Arid, Semi-Arid, Monsoon, or Tundra)
4. **Include: Discretion Data Sharing for model (UC-9)**
5. The user submits the changes.
6. The system validates the preference data (budget limits, enum values).
7. The system saves the updated preferences to the database with the current timestamp.
8. The system confirms a successful update to the user.
9. The system updates the recommendation parameters based on new preferences.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the preferences management page.
3. The user may need to re-enter unsaved changes.

3a. No previous preferences exist:

1. The system displays the default preference form with empty/default values.
2. The user sets initial preferences for all required fields.

3b. User sets invalid budget range:

1. The system detects that the lower limit exceeds the upper limit.
2. The system highlights the budget fields with an error.
3. The user corrects the budget range and continues.

5a. The system detects validation errors:

1. The system highlights the fields with errors (invalid budget range, missing selections).
2. The system provides specific error messages for each field.
3. The user corrects the errors and resubmits.

6a. Database save fails:

1. The system displays error message about save failure.
2. The system retains user's entered data in the form.
3. The user can retry saving or navigate away.

8a. User disables data sharing:

1. The system warns that recommendations will become generic.
2. The system asks for confirmation of data sharing disable.
3. If confirmed, system updates preference and switches to generic recommendation mode.

Special Requirements:

- Preference updates must be processed within 3 seconds
- Budget limits must be positive integers with lower limit ≤ upper limit
- The interface must display all enum options for interests, styles, and climates
- Changes to data sharing preference should immediately affect recommendation personalisation
- The system must store the timestamp of each preference update

Technology and Data Variations List:

- Budget input can use sliders, number inputs, or currency selectors
- Interest and style selection can use radio buttons, dropdown menus, or visual cards
- Climate preferences can be displayed with weather icons or descriptive text
- The data sharing toggle can include explanatory tooltips about personalisation impact

Frequency of Occurrence:

Moderate - users typically update preferences periodically or when their travel interests change.

Open Issues:

- Should the system suggest preference adjustments based on user booking behaviour?
- How should conflicting climate preferences be handled in recommendations?
- Should budget preferences be currency-specific or universal?
- Should there be validation on realistic budget ranges?
- How often should the system prompt users to review/update their preferences?

The Two-Column Variation

Use Case UC-6: Manage Travel Preferences

Primary Actor: User (Traveller)

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the travel preferences section of their profile.	
	2. The system displays current travel preferences (budget range, primary interest, travel style, climate preference, data sharing setting).
3. The user modifies budget limits, selects primary interest, chooses travel style, sets preferred climate, and toggles data sharing preference.	
4. The user submits the changes.	
	5. The system validates the preference data (budget range validity, enum selections).
	6. The system saves the updated preferences to the database with the current timestamp.
	7. The system confirms a successful update to the user.
	8. The system updates the recommendation parameters based on new preferences.

Fully Dressed Use Case: Manage Past Trips List

Use Case: UC-7 Manage Past Trips List

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants to maintain a record of past travels and use it for future planning
- App Developers: Want to collect travel history data to improve recommendation algorithms
- Analytics Team: Wants to analyse travel patterns for system enhancements
- Marketing Team: Wants to understand user travel history for targeted offers

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated

Success Guarantee (Postconditions):

- The user's past trip information is maintained accurately
- The recommendation system incorporates the travel history
- All changes are saved to the database
- The updated history is reflected in the user's travel profile

Main Success Scenario:

1. The user navigates to the past trips section of their profile.
2. The system displays a list of recorded past trips with details including destinations, travel dates, accommodations, and activities.
3. The user reviews the list and may select a trip to view or edit.
4. The user updates information for selected trips or adds new past trips or removes an old one.
5. The user submits any changes.
6. The system validates the modified trip data.
7. The system saves the updated travel history to the database.
8. The system confirms a successful update to the user.
9. The system updates the recommendation parameters based on the revised travel history.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the past trips management page.
3. The user may need to re-enter unsaved changes.

3a. No past trips exist:

1. The system displays an empty trips list with options to add trip history.
2. The user can choose to add past trips or return to the profile.

4a. The user requests to delete a past trip:

1. The system asks for confirmation.
2. The user confirms deletion.
3. The system removes the trip from the user's history.
4. The system updates recommendations accordingly.

5a. The user attempts to add a trip with invalid dates:

1. The system highlights the invalid date fields.
2. The user corrects the dates and continues.

6a. The system detects validation errors:

1. The system highlights the fields with errors.
2. The system provides specific error messages.
3. The user corrects the errors and resubmits.

Special Requirements:

- Past trip updates must be processed within 3 seconds.
- The interface must allow for easy browsing of multiple trips.
- The system should provide a visual timeline of past travels.
- Trip data should be exportable in common formats (PDF, CSV).

Technology and Data Variations List:

- Trip entry can be manual or imported from other travel platforms.
- Date selection can use various calendar interfaces.
- Photo attachments can be supported for trip memories.

Frequency of Occurrence: Low to moderate - users typically update past trips after completing travel.

Open Issues:

- How far back should the system allow past trip recording?
- Should the system offer integration with other travel logging platforms?
- How should incomplete past trip information be handled in recommendations?
- Should there be limits on the number of past trips that can be recorded?

The Two-Column Variation

Use Case UC-7: Manage Past Trips List
Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the past trips section of their profile.	
	2. The system displays a list of recorded past trips with details including destinations, travel dates, accommodations, and activities.
3. The user reviews the list and may select a trip to view or edit.	
4. The user updates information for selected trips or adds new past trips.	
5. The user submits any changes.	
	6. The system validates the modified trip data.
	7. The system saves the updated travel history to the database.
	8. The system confirms a successful update to the user.
	9. The system updates the recommendation parameters based on the revised travel history.

Fully Dressed Use Case: Manage Travel Bucket List and Travel Goals

Use Case: UC-8 Manage Travel Bucket List and Travel Goals

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants to track future travel aspirations and set achievable travel goals
- App Developers: Want to understand future travel intentions for system enhancements
- Marketing Team: Wants insight into desired destinations for targeted offerings

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated

Success Guarantee (Postconditions):

- The user's travel goals and bucket list are updated in the system
- All changes are saved to the database
- The updated goals are reflected in the user's profile
- The recommendation system incorporates the user's travel aspirations

Main Success Scenario:

1. The user navigates to the travel goals section of their profile.
2. The system displays current travel goals and bucket list destinations.
3. The user modifies their travel goals or adds/removes destinations to their bucket list.
4. The user may assign priority levels or tentative timelines to bucket list destinations.
5. The user submits the changes.
6. The system validates the modified data.
7. The system saves the updated travel goals and bucket list to the database.
8. The system confirms a successful update to the user.
9. The system adjusts recommendations to include bucket list destinations.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the travel goals management page.
3. The user may need to re-enter unsaved changes.

3a. No travel goals or bucket list destinations exist:

1. The system displays empty lists with options to add goals and destinations.
2. The user can choose to add entries or return to the profile.

4a. The user sets a goal that conflicts with travel preferences:

1. The system identifies the conflict.
2. The system suggests resolution options.
3. The user resolves the conflict or confirms the exception.

5a. The user assigns unrealistic timelines:

1. The system guides feasibility.
2. The user adjusts or confirms the timeline.

7a. The system detects validation errors:

1. The system highlights the fields with errors.
2. The system provides specific error messages.
3. The user corrects the errors and resubmits.

Special Requirements:

- Goal updates must be processed within 3 seconds.
- The interface must allow for prioritisation and categorisation of goals.

- The system should provide visual progress tracking toward goals.
- Destinations should be linkable to specific travel goals.

Technology and Data Variations List:

- Goal setting can use various formats (checklists, timelines, maps).
- Destination selection can be made via a map interface or text search.
- Goals can be categorised by type (adventure, cultural, relaxation, etc.).

Frequency of Occurrence: Moderate - users typically update travel goals periodically.

Open Issues:

- How should progress toward travel goals be tracked and visualised?
- Should the system suggest bucket list destinations based on user preferences?
- How should the system handle expired goals or timelines?
- Should users be able to share bucket lists with other users?

The Two-Column Variation

Use Case UC-8: Manage Travel Bucket List and Travel Goals
Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the travel goals section of their profile.	
	2. The system displays current travel goals and bucket list destinations.
3. The user modifies their travel goals or adds/removes destinations to their bucket list.	
4. The user may assign priority levels or tentative timelines to bucket list destinations.	
5. The user submits the changes.	
	6. The system validates the modified data.
	7. The system saves the updated travel goals and bucket list to the database.
	8. The system confirms a successful update to the user.
	9. The system adjusts recommendations to include bucket list destinations.

Fully Dressed Use Case: Discretion Data Sharing Preference for Model

Use Case: UC-9 Discretion Data Sharing Preference for Model

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants control over personal data usage while maintaining personalised services
- App Developers: Want access to user data to improve recommendation algorithms
- AI/ML Engineers:
    - Want to train models with comprehensive data while respecting privacy.
    - Wants to minimise liability related to data usage (Legal concerns)
    - Want to ensure compliance with data protection regulations (Data privacy)

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated

Success Guarantee (Postconditions):

- The user's data sharing preference (opt-in or opt-out) is updated in the system
- The change is saved to the database
- The system respects the user's data usage choice
- The recommendation system operates within the user's consent parameters

Main Success Scenario:

1. The user navigates to the privacy settings section of their profile.
2. The system displays the current data sharing preference for the recommendation model (either opted in or opted out).
3. The system provides an explanation of how data is used to improve recommendations.
4. The user reviews the information about data usage.
5. The user toggles their preference to either opt in or opt out of data collection for the recommendation model.
6. The user submits the change.
7. The system validates the selection.
8. The system saves the updated data sharing preference to the database.
9. The system confirms a successful update to the user.
10. The system adjusts data collection and recommendation parameters based on the new preference.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the privacy settings page.
3. The user may need to reset their preference if it wasn't saved.

3a. No previous data sharing preference exists:

1. The system displays the default privacy setting (opted out by default).
2. The user sets their initial preference.

4a. The user requests more information about data usage:

1. The system displays detailed explanations of what data is collected and how it's used.
2. The user reviews the information and continues setting their preference.

5a. The user opts out of data collection:

1. The system displays a message about the impact on recommendation quality.
2. The user confirms the decision or changes their selection.

Special Requirements:

- Privacy preference updates must be processed within 3 seconds.
- The interface must clearly explain data usage in non-technical language.
- The system must implement the change immediately upon saving.
- The system must maintain an audit trail of consent changes.
- The system must comply with relevant data protection regulations.

Technology and Data Variations List:

- Consent can be managed via a single toggle switch (on/off).
- Privacy explanations can include visual aids and examples.
- Consent records must be timestamped and versioned.

Frequency of Occurrence: Low to moderate - users typically set privacy preferences once and update occasionally.

Open Issues:

- Should previously collected data be deleted if a user opts out?
- Should the system periodically remind users to review their privacy settings?
- How should the impact of privacy choice on recommendation quality be communicated?
- Is a notification needed when data has been used to improve the model?

The Two-Column Variation

Use Case UC-9: Discretion Data Sharing Preference for Model
Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the privacy settings section of their profile.	
	2. The system displays the current data sharing preference for the recommendation model (either opted in or opted out).
	3. The system explains how data is used to improve recommendations.
4. The user reviews the information about data usage.	
5. The user toggles their preference to either opt in or opt out of data collection for the recommendation model.	
6. The user submits the change.	
	7. The system validates the selection.
	8. The system saves the updated data sharing preference to the database.
	9. The system confirms a successful update to the user.
	10. The system adjusts data collection and recommendation parameters based on the new preference.

Trip Planning Module Requirements

Fully Dressed Use Case: Create/Delete Trip

Use Case: UC-10 Create/Delete Trip

Primary Actor: User (Traveller)

Stakeholders and Interests:

• **User**: Wants to initiate new trip plans for destinations and remove unwanted or outdated trips • **App Developers**: Want to provide streamlined trip creation/deletion processes that encourage user engagement while preventing accidental data loss • **Travel Partners**: Want their services integrated into the trip planning process • **Marketing Team**: Wants to collect data on user trip creation/deletion patterns for targeted offerings • **Data Analytics Team**: Needs trip lifecycle data to understand user behaviour patterns

Preconditions:

• The user has a registered account and is logged in • The user has successfully authenticated • **For deletion**: The user has at least one existing trip in their account

Success Guarantee (Postconditions):

Create Trip: • A new trip plan is created with a selected destination • The trip plan has basic trip parameters established (destination, dates, trip type) • The trip is saved to the user's account • The user can proceed to detailed trip planning

Delete Trip: • The selected trip is permanently removed from the user's account • All associated trip data (itinerary, bookings, events) are deleted • The deletion is logged for audit purposes • The user's trip list is updated to reflect the removal

Main Success Scenario:

Create Trip Flow:

1. The user selects the "Create Trip" option from the main navigation.
2. The system presents a destination selection interface with filters for continents, countries, and cities.
3. The user selects a destination using the filtering system.
4. **Include: Verify Travel Requirements (UC-10)**
    - The system automatically initiates the travel requirements verification process.
    - The user completes the verification process.
5. The system creates a new trip planning workspace.
6. The user enters basic trip details, including tentative dates and trip type (solo or group).
7. The system saves the trip to the user's account.
8. The system offers options to:
    - Continue with detailed planning (Plan Trip)
    - Save for later and return to the dashboard

Delete Trip Flow:

9. The user navigates to their trips list or trip details page.
10. The user selects the "Delete Trip" option for a specific trip.

11. The system displays a confirmation dialogue with trip details and deletion consequences.
12. The user confirms the deletion by typing the trip name or clicking confirmation.
13. The system permanently removes the trip and all associated data.
14. The system displays a success message and updates the trips list.
15. The system logs the deletion for audit purposes.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the appropriate process with the entered data preserved.
3. The user continues the process.

3a. The user cannot find the desired destination:

1. The system offers a search function to find specific destinations.
2. The user searches for and selects the destination.
3. The process continues with the selected destination.

4a. The destination verification indicates the user does not meet travel requirements:

1. The system displays warning notifications about unmet requirements.
2. The system offers options to either continue with trip creation (acknowledging the requirements) or select a different destination.
3. The user decides how to proceed.

6a. The user attempts to enter invalid dates:

1. The system highlights the invalid date fields.
2. The system provides guidance on valid date selection.
3. The user corrects the dates and continues.

8a. The user chooses to continue with detailed planning:

1. The system transitions to the Plan Trip use case (UC-12).
2. The Plan Trip process continues with the basic trip details already established.

8b. The user wants to save the trip with minimal details:

1. The user selects the "Save and Complete Later" option.
2. The system saves the trip with available details.
3. The system adds the trip to the user's "In Progress" trips list.

11a. The user attempts to delete a trip with active bookings:

1. The system warns about active bookings that will be lost.
2. The system lists all bookings and their cancellation policies.
3. The user acknowledges the consequences or cancels the deletion.

12a. The user cancels the deletion:

1. The system closes the confirmation dialog.
2. The system returns to the trip list/details page.
3. No changes are made to the trip.

13a. Deletion fails due to system error:

1. The system displays an error message.
2. The system suggests trying again or contacting support.
3. The trip remains in the user's account unchanged.

14a. User immediately regrets deletion:

1. The system offers a brief "Undo" option (30 seconds).
2. If selected, the system restores the trip from temporary storage.
3. If not selected within the time limit, deletion becomes permanent.

Special Requirements:

- Destination selection interface must provide visualisation options (map, list, etc.)
- The system must seamlessly integrate the travel requirements verification
- **Deletion must require explicit confirmation to prevent accidental loss**
- **Active bookings must be highlighted before deletion**
- The trip creation process should complete within 3 minutes for common destinations
- **The deletion process must complete within 10 seconds**
- **The system must maintain an audit log of all trip deletions**

Technology and Data Variations List:

- Destination selection can use map interfaces, search functionality, or category browsing
- Date selection can use various calendar interfaces with season highlighting
- Trip type selection can use visual indicators for solo vs. group planning
- **Deletion confirmation can use various methods: typing the trip name, multiple clicks, or security questions**
- **Undo functionality can use temporary storage or soft-delete mechanisms**

Frequency of Occurrence:

- **Create**: High - primary entry point for creating new trips.
- **Delete**: Low to Moderate - users occasionally remove outdated or unwanted trips

Open Issues:

- Should trips be automatically categorised based on selected destination characteristics?
- How should the system handle destinations with severe travel restrictions?
- Should the system suggest optimal travel dates based on destination climate and events?
- How much information should be required vs. optional during initial trip creation?
- **Should deleted trips be recoverable for a certain period?**
- **What data should be preserved when deleting trips with shared group members?**
- **How should the system handle the deletion of trips with confirmed bookings?**
- **Should there be different deletion permissions for group trip organisers vs. members?**

The Two-Column Variation

Use Case UC-10: Create/Delete Trip

Primary Actor: User (Traveller)

Create Trip Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user selects the "Create Trip" option from the main navigation.	
	2. The system presents a destination selection interface with filters for continents, countries, and cities.
3. The user selects a destination using the filtering system.	
	4. The system automatically initiates the travel requirements verification process (UC-10).
5. The user completes the verification process.	
	6. The system creates a new trip planning workspace.
7. The user enters basic trip details, including tentative dates and trip type (solo or group).	
	8. The system saves the trip to the user's account.
	9. The system offers options to continue with detailed planning or save for later.

Delete Trip Main Success Scenario:

Actor Action (or Intention)	System Responsibility
10. The user navigates to their trips list and selects "Delete Trip" for a specific trip.	
	11. The system displays a confirmation dialogue with trip details and deletion consequences.
12. The user confirms the deletion.	
	13. The system permanently removes the trip and all associated data.
	14. The system displays a success message and updates the trips list.
	15. The system logs the deletion for audit purposes.

Fully Dressed Use Case: Verify Travel Requirements

Use Case: UC-11 Verify Travel Requirements

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants accurate information about travel requirements to plan trips confidently
- App Developers: Want to provide reliable travel advisory information to enhance user experience
- Government Agencies: Want to ensure travellers are aware of entry requirements
- Travel Insurance Partners: Want users to be informed about travel risks and requirements

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated
- The user has selected a destination for verification (either directly or as part of Create Trip)

Success Guarantee (Postconditions):

- The user has viewed and understood travel requirements for the selected destination
- The user has confirmed whether they meet the requirements
- The destination can be proceeded with in the trip creation process if the requirements are met
- The system has recorded the verification process

Main Success Scenario:

1. The user selects a destination for potential travel.

2. The system retrieves and displays travel advisories and requirements from the user's home country to the selected destination.
3. The system presents detailed information about visa requirements, passport validity, vaccination needs, and travel restrictions.
4. The user reviews the requirements and confirms they meet them.
5. The system acknowledges the user's confirmation of meeting requirements.
6. The system enables continuation of the trip creation process.
7. If invoked independently (not as part of Create Trip), the system offers the option to add the destination to the user's trip bucket list.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the requirements verification page.
3. The user continues the verification process.

2a. The system cannot retrieve travel advisory information:

1. The system displays an error message.
2. The system offers alternative sources for travel requirement information.
3. The user can try again later or proceed with limited information.

3a. The system detects that the selected destination has severe travel restrictions:

1. The system prominently displays warning notifications.
2. The system provides detailed information about the restrictions.
3. The user decides whether to proceed with consideration of the destination.

4a. The user determines they do not meet the requirements:

1. The system offers information about how to meet the requirements.
2. The user can still proceed with trip creation or bucket list addition with a warning flag.
3. The system flags the destination as requiring additional preparation.

6a. The user is in the Create Trip flow and wants to continue despite not meeting all requirements:

1. The system displays a confirmation dialogue with warnings.
2. The user confirms their understanding of the risks.
3. The system allows continuation with appropriate flags.

7a. The user is in standalone verification mode and chooses not to add the destination to their bucket list:

1. The system acknowledges the decision.

2. The system offers alternative destinations with similar characteristics but fewer requirements.

Special Requirements:

- Travel requirement information must be updated at least daily.
- The system must clearly distinguish between mandatory requirements and recommendations.
- Warning notifications for high-risk destinations must be prominently displayed.
- The interface must present complex requirement information in an easy-to-understand format.

Technology and Data Variations List:

- Travel advisories can be sourced from multiple government agencies and aggregated.
- Requirement display can include visual indicators for severity levels.
- The language of advisories can be translated based on user preferences.

Frequency of Occurrence: High - users will check requirements for every new destination considered, both directly and as part of the trip creation process.

Open Issues:

- How frequently should travel requirement data be updated?
- Should the system store historical requirement data for comparison?
- How should conflicting advisory information from different sources be reconciled?
- Should users be notified if requirements change for destinations in their bucket list?
- How should the verification process differ when accessed directly versus through Create Trip?

The Two-Column Variation

Use Case UC-11: Verify Travel Requirements
Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user selects a destination for potential travel.	
	2. The system retrieves and displays travel advisories and requirements from the user's home country to the selected destination.
	3. The system presents detailed information about visa

Actor Action (or Intention)	System Responsibility
	requirements, passport validity, vaccination needs, and travel restrictions.
4. The user reviews the requirements and confirms they meet them.	
	5. The system acknowledges the user's confirmation of meeting requirements.
	6. The system enables continuation of the trip creation process.
	7. If invoked independently, the system offers the option to add the destination to the user's trip bucket list.

Fully Dressed Use Case: View Created Trips

Use Case: UC-12 View Created Trips

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants to access and manage all trips they have created
- App Developers: Want to provide an organised interface for trip management
- System Administrators: Want to ensure efficient data retrieval and display
- Analytics Team: Wants to track user engagement with created trips

Preconditions:

- The user has a registered account and is logged in
- The user has successfully authenticated

Success Guarantee (Postconditions):

- The user can view a list of all their created trips
- Trips are organised by relevant categories (e.g., upcoming, past, in-progress)
- The user can select a trip to view or edit its details
- Trip status information is accurately displayed

Main Success Scenario:

1. The user selects the "View Created Trips" option from their profile or dashboard.
2. The system retrieves all trips associated with the user's account.
3. The system categorises the trips by status (upcoming, in-progress, past) and sorts them by date.

4. The system displays the organised trip list with key information for each trip (destination, dates, trip type).
5. The user browses through their trips.
6. The user selects a specific trip from the list.
7. The system retrieves detailed information for the selected trip.
8. The system displays the trip details, including itinerary, bookings, and group members (if applicable).
9. The user can choose to continue planning the selected trip, which extends to Plan Trip (UC-13).

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the trips list page.
3. The user continues browsing trips.

2a. The user has not created trips:

1. The system displays a message indicating no trips have been created.
2. The system offers the option to create a new trip via Create Trip (UC-12).
3. The user can proceed to trip creation.

5a. The user wants to filter or search for specific trips:

1. The system provides filtering options (by destination, date, status).
2. The user applies filters to narrow down the trip list.
3. The system updates the displayed trips based on selected filters.

6a. The user wants to perform actions on a trip without viewing details:

1. The system provides quick action options for each trip (edit, delete, share).
2. The user selects a quick action.
3. The system performs the requested action.

9a. The user chooses to continue planning the selected trip:

1. The system transitions to the Plan Trip use case (UC-13).
2. The user continues detailed trip planning.

9b. The user chooses to edit basic trip information:

1. The system provides an option to modify basic trip details.
2. The user updates the information.
3. The system saves the changes.

9c. The user chooses to delete a trip:

1. The system requests confirmation of the deletion.
2. The user confirms deletion.
3. The system removes the trip from the user's account.

Special Requirements:

- Trip list must load within 3 seconds, even for users with many trips.
- The interface must provide clear visual indicators of trip status.
- The system should allow offline viewing of basic trip information.
- Trip organisation should allow for customisation based on user preferences.

Technology and Data Variations List:

- Trip list can be displayed in various formats (cards, list, calendar view).
- Trip filtering can employ text search or category-based filtering.
- Trip details can be summarised or expanded based on user preference.

Frequency of Occurrence: High - users frequently access their created trips.

Open Issues:

- How should the system handle trips that were shared with the user versus created by them?
- Should there be a limit on the number of active trips a user can maintain?
- What archiving policies should be implemented for past trips?
- Should the system suggest trip modifications based on changed circumstances (e.g., travel advisories)?

The Two-Column Variation

Use Case UC-12: View Created Trips
Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user selects the "View Created Trips" option from their profile or dashboard.	
	2. The system retrieves all trips associated with the user's account.
	3. The system categorises the trips by status (upcoming, in-progress, past) and sorts them by date.
	4. The system displays the organised trip list with

Actor Action (or Intention)	System Responsibility
	key information for each trip (destination, dates, trip type).
5. The user browses through their trips.	
6. The user selects a specific trip from the list.	
	7. The system retrieves detailed information for the selected trip.
	8. The system displays the trip details, including itinerary, bookings, and group members (if applicable).
9. The user can choose to continue planning the selected trip, which extends to Plan Trip (UC-12).	

Fully Dressed Use Case: Plan a Trip

Use Case: UC-13 Plan a Trip

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants to create a structured trip plan that organises travel elements efficiently
- Travel Companions: Want to participate in group planning with shared access to the itinerary
- App Developers: Want to provide comprehensive planning tools that keep users engaged
- Travel Partners: Want their services to be integrated into the trip planning process

Preconditions:

- The user has a registered account and is logged in
- The user has created a basic trip (through Create Trip UC-10)
- Travel requirements for the destination have been verified

Success Guarantee (Postconditions):

- A complete trip plan is created with a detailed itinerary
- The trip plan includes dates, activities, accommodations, and transportation
- Group members are invited and can access the shared plan (if applicable)
- The trip plan is saved and can be edited later
- The trip is accessible through View Created Trips (UC-12)

Main Success Scenario:

1. The user continues from Create Trip or accesses a previously created trip from View Created Trips.
2. The system displays the basic trip details and planning workspace.
3. If not already specified in Create Trip, the user chooses between solo or group trip planning.
4. If choosing a group trip, the user invites other registered users by generating and sharing invitation links.
5. Invited users receive notifications and can accept or decline the invitation.
6. The system creates or enhances the itinerary structure with placeholders for trip dates, activities, accommodations, and transportation.
7. The user fills in or refines trip details and itinerary elements (transportation, events, accommodation, and weather).
8. The system saves the trip plan.
9. The system presents options for further planning steps (add transportation, accommodations, events, weather information).
10. The user and group members (if applicable) can view and edit the trip plan.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the trip planning workspace with saved progress.
3. The user continues planning.

3a. The user starts with solo planning but later wants to convert to group planning:

1. The user accesses group planning options from the trip plan.
2. The system converts the trip to a group plan.
3. The user continues by inviting other users.

4a. The user invites non-registered users:

1. The system generates invitation links that include registration prompts.
2. The system sends invitations to the provided contact information.
3. Non-registered users must register before accessing the group plan.

5a. Invited users decline the invitation:

1. The system notifies the trip creator of declined invitations.
2. The user can invite alternative travel companions.

7a-8a. The user needs to save progress before completing the planning:

1. The user selects the save option at any point.
2. The system saves the current state of the trip plan.
3. The user can return later to continue planning via View Created Trips.

9a. The user chooses to add transportation to the itinerary:

1. The system initiates the Add Transport to Trip Itinerary use case (UC-14).

9b. The user chooses to add accommodation to the itinerary:

1. The system initiates the Add Accommodation to Trip Itinerary use case (UC-15).

9c. The user chooses to add events to the itinerary:

1. The system initiates the Add Event to Trip Itinerary use case (UC-16).

9d. The user chooses to add weather information to the itinerary:

1. The system initiates the Add Weather Info to Trip Itinerary use case (UC-17).

Special Requirements:

- Trip planning interface must be collaborative with real-time updates for group planning.
- The system must support offline editing with synchronisation when connectivity is restored.
- The invitation system must be secure to prevent unauthorised access to trip plans.
- The planning interface must be intuitive with drag-and-drop functionality for organising elements.

Technology and Data Variations List:

- Trip planning can be viewed in calendar, list, or map visualisation formats.
- Collaboration can be supported through various notification methods.
- Invitations can be sent via email, SMS, or in-app messaging.

Frequency of Occurrence: High - central to the application's purpose.

Open Issues:

- What permissions should different group members have?
- How should conflicts in group planning be resolved?
- Should there be a limit on the number of members in a group plan?
- How should the system handle abandoned or inactive trip plans?

The Two-Column Variation

Use Case UC-13: Plan a Trip
Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user continues from Create Trip or accesses a previously created trip from View Created Trips.	
	2. The system displays the basic trip details and planning workspace.
3. If not already specified, the user chooses between solo or group trip planning.	
4. If choosing a group trip, the user invites other registered users by generating and sharing invitation links.	
	5. Invited users receive notifications and can accept or decline the invitation.
	6. The system creates or enhances the itinerary structure with placeholders for trip dates, activities, accommodations, and transportation.
7. The user fills in or refines trip details and itinerary elements.	
	8. The system saves the trip plan.
	9. The system presents options for further planning steps (add transportation, accommodations, events, weather information).
10. The user and group members (if applicable) can view and edit the trip plan.	

Fully Dressed Use Case: Manage Transportation in Trip Itinerary

Use Case: UC-14 Manage Transportation in Trip Itinerary

Primary Actor: User (Traveller)

Stakeholders and Interests:

- **User**: Wants to find, book, modify, and remove transportation options for their trip with full control over booking details
- **Travel Companions**: Want to coordinate transportation plans and stay informed of changes if it's a group trip
- **Transportation Providers**: Want their services to be accurately represented and booking changes to be properly handled

- **App Developers**: Want to provide comprehensive transportation management with real-time pricing and seamless CRUD operations
- **Customer Support**: Needs a clear audit trail of transportation changes for user assistance

Preconditions:

- The user has a registered account and is logged in
- The user has created a trip plan with an itinerary
- The system has access to transportation API services
- **For Update/Delete operations**: The user has existing transportation entries in their itinerary

Success Guarantee (Postconditions):

- Transportation details are properly managed (created, read, updated, or deleted) in the trip itinerary
- Booking information is accurately recorded and maintained in the system
- The updated itinerary reflects current transportation timing and status
- Other trip elements (accommodations, activities) remain properly aligned with transportation changes
- **All transportation modifications are logged for audit purposes**
- **Group members are notified of relevant transportation changes**

Main Success Scenario:

Create Transportation:

1. The user accesses the itinerary page for their trip plan.
2. The user selects the "Add Transportation" option.
3. The system displays a search interface for transportation options with filters.
4. The system retrieves and displays available transportation options from multiple providers, including flights and cruises.
5. The user applies filters based on preferences such as transportation type, price, duration, and provider.
6. The system updates the results based on selected filters.
7. The user selects a preferred transportation option.
8. The system redirects the user to the official booking source website.
9. The user completes the booking on the external website.
10. The user returns to the app and selects "Add Booked Transportation."
11. The user enters the booking details, including confirmation numbers, departure/arrival times, and provider information.
12. The system validates the transportation details.
13. The system saves the transportation details to the trip itinerary.

14. The system updates the trip timeline based on the transportation schedule.

Read/View Transportation:

15. The user selects "View Transportation" from the itinerary menu.
16. The system displays all existing transportation entries with complete details.
17. The user can view individual transportation details, booking status, and related information.

Update Transportation:

18. The user selects "Edit" on an existing transportation entry.
19. The system displays the transportation modification form with current details populated.
20. The user modifies transportation details (times, confirmation numbers, provider info, personal notes, etc.).
21. The system validates the updated information for conflicts and accuracy.
22. The user submits the changes.
23. The system saves the updated transportation information.
24. The system updates the trip timeline based on the modified schedule.
25. The system notifies group members of transportation changes (if applicable).

Delete Transportation:

26. The user selects "Delete" on an existing transportation entry.
27. The system displays a detailed confirmation dialogue with cancellation policy warnings and potential impacts.
28. The user reviews the deletion consequences and confirms the removal.
29. The system permanently removes the transportation from the itinerary.
30. The system adjusts the trip timeline and checks for conflicts with other itinerary items.
31. The system notifies group members of the transportation removal (if applicable).
32. The system logs the deletion for audit purposes.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the appropriate transportation management page with data preserved where possible.
3. The user continues the process from the last successful step.

3a. The system cannot retrieve transportation options:

1. The system displays an error message explaining the connectivity issue.
2. The system offers the option to try again or to manually enter transportation details.
3. The user chooses how to proceed.

5a-6a. No transportation options match the user's filters:

1. The system suggests broadening the search criteria or adjusting filters.
2. The system displays alternative suggestions based on relaxed criteria.
3. The user adjusts filters and continues or chooses manual entry.

7a. The user wants to compare multiple options before deciding:

1. The system allows saving potential options to a comparison list.
2. The user can review and compare saved options side-by-side.
3. The user selects the preferred option from the comparison.

8a-9a. The booking fails on the external website:

1. The user returns to the app without booking confirmation.
2. The system offers to search for alternative options or try the same booking later.
3. The user can choose to continue with different transportation or save the search for later.

11a. The user books transportation outside the app:

1. The user selects the "Add External Transportation" option.
2. The user manually enters all transportation details, including provider, times, and confirmation information.
3. The system validates the manually entered details and saves them to the itinerary.

20a. The user attempts to update with conflicting times:

1. The system detects conflicts with existing accommodations, events, or other transportation.
2. The system displays detailed conflict information and suggests resolutions.
3. The user resolves conflicts by adjusting times or confirms the overlapping schedule.

21a. The system detects validation errors in updates:

1. The system highlights fields with errors and provides specific error messages.
2. The system suggests corrections based on common patterns.
3. The user corrects the errors and resubmits the changes.

27a. The user attempts to delete a transportation with a non-refundable booking:

1. The system displays detailed cancellation policy information and potential financial losses.
2. The system calculates and shows estimated cancellation fees.
3. The user acknowledges potential losses and confirms deletion or cancels the operation.

29a. Deletion affects other itinerary items:

1. The system identifies and displays dependent items (connecting flights, airport transfers, etc.) that may be affected.
2. The system suggests alternative arrangements or warns about potential issues.
3. The user acknowledges the impacts and confirms deletion or modifies the deletion scope.

30a. Transportation deletion creates timeline gaps:

1. The system identifies gaps in the travel timeline that may need addressing.
2. The system suggests transportation alternatives to fill the gaps.
3. The user can immediately search for replacement transportation or address the gaps later.

Special Requirements:

- Transportation search results must update in real-time with accurate pricing and availability
- The system must handle currency conversions for international bookings with current exchange rates
- The interface must clearly distinguish between different transportation types, classes, and booking statuses
- Booking redirections must pass relevant search parameters to external sites for a seamless user experience
- For group trips, the system must indicate which members have booked transportation and coordination status
- All transportation modifications must be logged with timestamps and user identification for audit trails
- Cancellation policies must be clearly displayed before any deletion operations
- The system must handle time sone conversions for international transportation accurately
- Modification notifications to group members must be configurable and respect user preferences

Technology and Data Variations List:

- Transportation data can be sourced from multiple API providers with fallback options for service interruptions
- Booking information can be entered manually, imported from confirmation emails, or integrated via booking APIs
- Transportation visualisation can include map routes, timeline views, or detailed itinerary formats
- Update operations can support bulk modifications for efficiency
- Deletion operations can support soft delete with recovery options
- Version control can track all changes to transportation entries

Frequency of Occurrence:

- **Create**: High - essential component of trip planning

- **Read**: Very High - users frequently review transportation details
- **Update**: Moderate - users occasionally modify bookings due to changes
- **Delete**: Low to Moderate - users sometimes cancel or remove transportation

Open Issues:

- How should the system handle transportation cancellations that incur fees?
- Should the system attempt to automatically synchronise transportation bookings across group members?
- How should multi-leg journeys be represented and managed in the itinerary?
- Should the system offer price tracking features for transportation options?
- What level of modification history should be maintained for each transportation entry?
- How should the system handle conflicting updates from multiple group members?
- Should deleted transportation entries be recoverable for a certain period?
- How should the system prioritise transportation options when multiple APIs provide different results?

The Two-Column Variation

Use Case UC-14: Manage Transportation in Trip Itinerary

Primary Actor: User (Traveller)

Create Transportation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user accesses the itinerary page for their trip plan.	
2. The user selects the "Add Transportation" option.	
	3. The system displays a search interface for transportation options with filters.
	4. The system retrieves and displays available transportation options from multiple providers, including flights and cruises.
5. The user applies filters based on preferences such as transportation type, price, duration, and provider.	
	6. The system updates the results based on selected filters.
7. The user selects a preferred transportation option.	
	8. The system redirects the user to the official

Actor Action (or Intention)	System Responsibility
	booking source website.
9. The user completes the booking on the external website.	
10. The user returns to the app and selects "Add Booked Transportation."	
11. The user enters the booking details, including confirmation numbers, departure/arrival times, and provider information.	
	12. The system validates the transportation details.
	13. The system saves the transportation details to the trip itinerary.
	14. The system updates the trip timeline based on the transportation schedule.

Read Transportation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
15. The user selects "View Transportation" from the itinerary menu.	
	16. The system displays all existing transportation entries with complete details.
17. The user can view individual transportation details, booking status, and related information.	

Update Transportation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
18. The user selects "Edit" on an existing transportation entry.	
	19. The system displays the transportation modification form with current details populated.
20. The user modifies transportation details (times, confirmation numbers, provider info, personal notes, etc.).	
	21. The system validates the updated information for conflicts and accuracy.
22. The user submits the changes.	
	23. The system saves the updated

Actor Action (or Intention)	System Responsibility
	transportation information.
	24. The system updates the trip timeline based on the modified schedule.
	25. The system notifies group members of transportation changes (if applicable).

Delete Transportation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
26. The user selects "Delete" on an existing transportation entry.	
	27. The system displays a detailed confirmation dialogue with cancellation policy warnings and potential impacts.
28. The user reviews the deletion consequences and confirms the removal.	
	29. The system permanently removes the transportation from the itinerary.
	30. The system adjusts the trip timeline and checks for conflicts with other itinerary items.
	31. The system notifies group members of the transportation removal (if applicable).
	32. The system logs the deletion for audit purposes.

Fully Dressed Use Case: Manage Accommodation in Trip Itinerary

Use Case: UC-15 Manage Accommodation in Trip Itinerary

Primary Actor: User (Traveller)

Stakeholders and Interests:

- **User**: Wants to find, book, modify, and remove accommodations for their trip with complete booking management
- **Accommodation Providers**: Want their properties to be accurately represented and booking modifications properly handled
- **App Developers**: Want to provide comprehensive accommodation management with real-time availability and seamless CRUD operations
- **Booking Services**: Need an accurate representation of their accommodation inventory and booking modifications

- **Customer Support**: Requires a clear accommodation change history for user assistance

Preconditions:

- The user has a registered account and is logged in
- The user has created a trip plan with an itinerary
- The system has access to the accommodation API services
- For Update/Delete operations: The user has existing accommodation entries in their itinerary

Success Guarantee (Postconditions):

- Accommodation details are properly managed (created, read, updated, or deleted) in the trip itinerary
- Booking information is accurately recorded and maintained in the system
- The updated itinerary reflects current accommodation timing and availability status
- Accommodation is properly aligned with transportation and activity timing
- All accommodation modifications are logged with detailed audit information
- Group members are appropriately notified of accommodation changes that affect them
- Booking confirmations and modifications are properly tracked

Main Success Scenario:

Create Accommodation:

1. The user accesses the itinerary page for their trip plan.
2. The user selects the "Add Accommodation" option.
3. The system displays a search interface for accommodation options with comprehensive filters.
4. The system retrieves and displays available accommodations from multiple providers with real-time availability.
5. The user applies filters based on preferences such as price, location, amenities, rating, and accommodation type.
6. The system updates the results based on selected filters and displays detailed property information.
7. The user selects a preferred accommodation option.
8. The system redirects the user to the official booking source website with pre-filled search parameters.
9. The user completes the booking on the external website.
10. The user returns to the app and selects "Add Booked Accommodation."
11. The user enters the booking details, including confirmation numbers, check-in/out times, property information, and room details.
12. The system validates the accommodation details for consistency and conflicts.
13. The system saves the accommodation details to the trip itinerary.

14. The system updates the trip timeline based on the accommodation schedule.

Read/View Accommodation:

15. The user selects "View Accommodations" from the itinerary menu.
16. The system displays all existing accommodation entries with complete booking details, photos, and amenities.
17. The user can view individual accommodation details, including booking status, cancellation policies, and contact information.

Update Accommodation:

18. The user selects "Edit" on an existing accommodation entry.
19. The system displays the accommodation modification form with current booking details populated.
20. The user modifies accommodation details (dates, room preferences, guest count, special requests, contact info, etc.).
21. The system validates the updated information against availability and booking policies.
22. The user submits the accommodation changes.
23. The system saves the updated accommodation information and checks for availability conflicts.
24. The system updates the trip timeline based on the modified accommodation schedule.
25. The system notifies relevant group members of accommodation changes that affect shared bookings.

Delete Accommodation:

26. The user selects "Delete" on an existing accommodation entry.
27. The system displays a comprehensive confirmation dialogue with cancellation policy details, potential fees, and booking impacts.
28. The user reviews the deletion consequences, including financial implications and confirms the removal.
29. The system permanently removes the accommodation from the itinerary.
30. The system adjusts the trip timeline and identifies any gaps in accommodation coverage.
31. The system notifies group members of the accommodation removal and suggests alternative arrangements if needed.
32. The system logs the deletion with full details for audit and potential recovery purposes.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the appropriate accommodation management page with booking data preserved.
3. The user continues the process from the last confirmed step.

3a. The system cannot retrieve accommodation options:

1. The system displays a detailed error message explaining the connectivity or service issue.
2. The system offers options to try again with different providers or to manually enter accommodation details.
3. The user chooses to retry, switch providers, or proceed with manual entry.

5a-6a. No accommodations match the user's filter criteria:

1. The system suggests broadening the search criteria by relaxing specific filters.
2. The system displays alternative accommodation types or nearby locations that might meet the user's needs.
3. The user adjusts filters, considers alternatives, or chooses manual entry.

7a. The user wants to compare multiple accommodation options:

1. The system allows saving potential accommodations to a detailed comparison list with amenities and pricing side-by-side.
2. The user can add multiple properties to compare features, locations, and prices.
3. The user selects the preferred accommodation from the comprehensive comparison.

8a-9a. The booking fails on the external accommodation website:

1. The user returns to the app without successful booking confirmation.
2. The system offers to search for alternative accommodations or attempt the same booking later.
3. The user can choose to try different accommodations, retry later, or manually enter external booking details.

11a. The user books accommodation outside the app through different channels:

1. The user selects the "Add External Accommodation" option.
2. The user manually enters comprehensive accommodation details including property info, booking confirmation, and contact details.
3. The system validates the manually entered accommodation information and integrates it with the itinerary.

20a. The user attempts to modify dates outside available periods:

1. The system checks real-time availability for the requested new dates across multiple providers.
2. The system displays alternative available dates or suggests similar properties with availability.
3. The user chooses from available alternatives or modifies their date requirements.

21a. The system detects conflicts with accommodation modifications:

1. The system identifies conflicts with transportation schedules, other accommodations, or group member bookings.
2. The system provides detailed conflict information and suggests specific resolutions.
3. The user resolves conflicts by adjusting dates, choosing alternatives, or confirming overlapping arrangements.

27a. The user attempts to delete accommodation within the cancellation penalty period:

1. The system displays detailed cancellation fees, policies, and exact financial implications.
2. The system calculates total costs, including penalties, and shows refund amounts.
3. The user acknowledges potential charges and confirms deletion or cancels the operation to avoid fees.

29a. Accommodation deletion creates coverage gaps:

1. The system identifies dates without accommodation and calculates the gap duration.
2. The system suggests immediate alternative accommodations for the uncovered dates.
3. The user can immediately search for replacement accommodations or address gaps in trip planning later.

30a. Group accommodation deletion affects multiple members:

1. The system identifies all group members affected by the accommodation removal.
2. The system displays impact analysis and suggests coordination steps for group accommodation.
3. The user confirms understanding of group impacts and proceeds or modifies the deletion scope.

Special Requirements:

- Accommodation search results must display real-time availability and accurate pricing with currency conversion
- The system must handle different accommodation types (hotels, vacation rentals, hostels) with appropriate booking flows
- Property photos, amenities, and guest reviews must be displayed during selection
- Booking redirections must preserve search parameters and user preferences for seamless external booking
- For group trips, the system must support multiple room bookings and shared accommodation coordination
- All accommodation modifications must maintain detailed change logs with timestamps and modification reasons
- Cancellation policies and fees must be prominently displayed before any deletion operations • The system must handle different check-in/check-out policies and time zones accurately
- Group notification preferences must be respected for accommodation change communications

Technology and Data Variations List:

- Accommodation data can be aggregated from multiple booking platforms with real-time synchronisation
- Property information can include virtual tours, 360-degree photos, or detailed floor plans
- Booking management can integrate with property management systems for direct communication • Modification tracking can include before/after comparisons and change justifications
- Availability checking can use predictive algorithms for better date suggestions
- Price monitoring can alert users to better deals for their selected accommodations

Frequency of Occurrence:

- **Create**: High - essential component of trip planning for overnight stays
- **Read**: Very High - users frequently review accommodation details and amenities
- **Update**: Moderate - users modify bookings due to plan changes or better options
- **Delete**: Low to Moderate - users cancel accommodations due to itinerary changes

Open Issues:

- How should the system handle accommodation modifications that require direct contact with properties?
- Should the system offer automatic rebooking when accommodations become unavailable?
- How should multi-property stays (different hotels each night) be managed efficiently?
- Should the system provide accommodation recommendations based on transportation and activity locations?
- What level of integration should be maintained with property management systems for real-time updates?
- How should the system handle group accommodation when members have different modification permissions?
- Should accommodation history include price change tracking for user cost optimisation?
- How should the system balance user privacy with group coordination for shared accommodations?

The Two-Column Variation

Use Case UC-15: Manage Accommodation in Trip Itinerary

Primary Actor: User (Traveller)

Create Accommodation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user accesses the itinerary page for their trip plan.	
2. The user selects the "Add Accommodation" option.	
	3. The system displays a search interface for accommodation options with comprehensive filters.
	4. The system retrieves and displays available accommodations from multiple providers with real-time availability.
5. The user applies filters based on preferences such as price, location, amenities, rating, and accommodation type.	
	6. The system updates the results based on selected filters and displays detailed property information.
7. The user selects a preferred accommodation option.	
	8. The system redirects the user to the official booking source website with pre-filled search parameters.
9. The user completes the booking on the external website.	
10. The user returns to the app and selects "Add Booked Accommodation."	
11. The user enters the booking details, including confirmation numbers, check-in/out times, property information, and room details.	
	12. The system validates the accommodation details for consistency and conflicts.
	13. The system saves the accommodation details to the trip itinerary.
	14. The system updates the trip timeline based on the accommodation schedule.

Read Accommodation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
15. The user selects "View Accommodations" from the itinerary menu.	

Actor Action (or Intention)	System Responsibility
	16. The system displays all existing accommodation entries with complete booking details, photos, and amenities.
17. The user can view individual accommodation details, including booking status, cancellation policies, and contact information.	

Update Accommodation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
18. The user selects "Edit" on an existing accommodation entry.	
	19. The system displays the accommodation modification form with current booking details populated.
20. The user modifies accommodation details (dates, room preferences, guest count, special requests, contact info, etc.).	
	21. The system validates the updated information against availability and booking policies.
22. The user submits the accommodation changes.	
	23. The system saves the updated accommodation information and checks for availability conflicts.
	24. The system updates the trip timeline based on the modified accommodation schedule.
	25. The system notifies relevant group members of accommodation changes that affect shared bookings.

Delete Accommodation Main Success Scenario:

Actor Action (or Intention)	System Responsibility
26. The user selects "Delete" on an existing accommodation entry.	
	27. The system displays a comprehensive confirmation dialogue with cancellation policy details, potential fees, and booking impacts.
28. The user reviews the deletion	

Actor Action (or Intention)	System Responsibility
consequences, including financial implications and confirms the removal.	
	29. The system permanently removes the accommodation from the itinerary.
	30. The system adjusts the trip timeline and identifies any gaps in accommodation coverage.
	31. The system notifies group members of the accommodation removal and suggests alternative arrangements if needed.
	32. The system logs the deletion with full details for audit and potential recovery purposes.

Fully Dressed Use Case: Manage Events in Trip Itinerary

Use Case: UC-16 Manage Events in Trip Itinerary

Primary Actor: User (Traveller)

Stakeholders and Interests:

- **User**: Wants to plan, organise, modify, and remove activities and events during their trip with complete control
- **App Developers**: Want to provide comprehensive event planning features with seamless CRUD operations and group coordination
- **Group Trip Organisers**: Need tools to manage event participation and coordinate group activities effectively
- **Local Tourism**: Benefits from accurate event representation and booking facilitation

Preconditions:

• The user has a registered account and is logged in • The user has created a trip plan with an itinerary • The user has appropriate permissions to manage events (owner or editor) • **For Update/Delete operations**: The user has existing events in their itinerary

Success Guarantee (Postconditions):

- Event details are properly managed (created, read, updated, or deleted) in the trip itinerary
- Group members are appropriately notified of event changes and can respond accordingly
- The updated itinerary reflects accurate event timing and participation status

- Events are properly aligned with other trip elements (transportation, accommodation, other activities)
- All event modifications are logged with detailed change history
- Participant responses and attendance tracking are maintained accurately
- Event conflicts and scheduling issues are identified and resolved

Main Success Scenario:

Create Event:

1. The user accesses the itinerary page for their trip plan.
2. The user selects the "Add Event" option.
3. The system displays a comprehensive event creation form with activity suggestions for the destination.
4. The user enters detailed event information, including name, date and time, location, description, and duration.
5. The user specifies event requirements: whether the event is mandatory or optional and selects specific group members or all members.
6. The user sets event parameters such as cost estimates, booking requirements, and preparation needs.
7. The user submits the complete event details.
8. The system validates the event information for conflicts with transportation, accommodation, and other events.
9. The system saves the event to the trip itinerary with all specified parameters.
10. The system sends detailed notifications to relevant group members about the new event with RSVP options.
11. The system updates the trip timeline to include the new event and checks for optimal scheduling.
12. Group members receive event details and can indicate their participation status and any special requirements.

Read/View Events:

13. The user selects "View Events" from the itinerary menu.
14. The system displays all existing events with comprehensive details, including participation status, timing, and logistics.
15. The user can view individual event details, participant responses, location information, and related bookings.

Update Event:

16. The user selects "Edit" on an existing event entry.
17. The system displays the event modification form with all current details populated and edit permissions verified.
18. The user modifies event details such as timing, location, description, participant requirements, or event type.

19. The system validates the updated information for schedule conflicts and participant availability.
20. The user submits the event changes with modification reasons if significant changes are made.
21. The system saves the updated event information and identifies all affected participants.
22. The system updates the trip timeline based on the modified event schedule and checks for new conflicts.
23. The system sends change notifications to affected group members with updated event details and new RSVP requests if needed.
24. Group members can review changes and update their participation status accordingly.

Delete Event:

25. The user selects "Delete" on an existing event entry.
26. The system displays a detailed confirmation dialogue showing event details, participant responses, and potential impacts on the itinerary.
27. The user reviews the deletion consequences, including effects on group members and any associated bookings.
28. The user confirms the event deletion and optionally provides a reason for removal.
29. The system permanently removes the event from the itinerary and all related participant data.
30. The system updates the trip timeline and identifies any scheduling gaps or opportunities for optimisation.
31. The system notifies all previously invited group members of the event cancellation with explanation if provided.
32. The system logs the deletion with comprehensive details for audit purposes and potential group coordination needs.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the appropriate event management page with form data preserved where possible.
3. The user continues the process from the last completed step.

3a. The user wants to add a suggested activity from destination recommendations:

1. The user selects "Add Suggested Activity" instead of creating a completely new event.
2. The system presents a curated list of popular and recommended activities for the specific destination and travel dates.
3. The user selects a suggested activity, and the system pre-populates the event form with activity details.
4. The user customises the pre-populated details and continues with the standard event creation process.

4a. The user attempts to schedule an event that conflicts with existing transportation or accommodation:

1. The system detects scheduling conflicts with confirmed transportation times or accommodation check-in/check-out schedules.
2. The system displays detailed conflict information with specific timing issues and suggests alternative scheduling options.
3. The user can adjust event timing, confirm the overlapping schedule with the understanding of logistics, or choose different activities.

6a. The user wants to make event attendance mandatory for all group members:

1. The user selects the "Required Attendance" option and specifies the attendance policy.
2. The system marks the event as mandatory and prepares appropriate notifications.
3. Group members receive notifications clearly indicating that the event requires attendance and any consequences for non-participation.

8a. The system detects validation errors or incomplete event information:

1. The system highlights specific fields with errors and provides detailed error messages and suggestions.
2. The system offers to save the event as a draft for completion later if substantial information is provided.
3. The user corrects the identified information and resubmits or saves as a draft for later completion.

10a. Some group members cannot receive event notifications:

1. The system logs failed notification attempts with specific delivery failure reasons.
2. The system attempts to deliver notifications through alternative communication channels (email, in-app alerts, etc.).
3. The event creator is informed of notification delivery status and can manually contact members who didn't receive notifications.

18a. The user changes event from optional to mandatory or vice versa:

1. The system detects the significant change in event requirements and highlights the implications.
2. The system sends special notifications to all group members explaining the requirement change and any new expectations.
3. Group members receive updated participation requirements and can respond to the changed event status.

20a. Event modifications create conflicts with other itinerary items:

1. The system identifies specific conflicts with transportation, accommodation, or other events and displays detailed conflict analysis.
2. The system suggests specific resolutions including alternative timing, modified logistics, or event restructuring.
3. The user resolves conflicts through suggested solutions or confirms the overlapping schedule with full understanding of implications.

26a. The user attempts to delete an event with confirmed external bookings or tickets:

1. The system warns about ticketed events, restaurant reservations, or other confirmed bookings that may be lost.
2. The system displays specific booking details and cancellation policies for each associated reservation.
3. The user acknowledges potential financial losses and booking complications, or cancels the deletion to handle bookings separately first.

29a. Event deletion affects group coordination or other dependent events:

1. The system identifies other events or activities that depend on the deleted event for logistics or coordination.
2. The system suggests modifications to dependent events or warns about potential coordination issues.
3. The user addresses dependent events before confirming deletion or acknowledges the coordination impacts.

Special Requirements:

- Event creation must integrate with destination activity databases for accurate suggestions and information
- The system must detect and provide clear warnings about scheduling conflicts with all other itinerary elements
- For group trips, the system must provide comprehensive RSVP functionality with attendance tracking and reminder capabilities
- The interface must support recurring events (e.g., daily breakfast, regular group meetings) with bulk management options
- **Event modification notifications must be configurable based on user preferences and change significance**
- **The system must handle different time sones accurately for international travel events**
- **Participation tracking must support partial attendance and alternative arrangements for group members**
- **Event logistics must integrate with transportation and accommodation for seamless coordination**

Technology and Data Variations List:

- Event locations can be selected from integrated maps, entered as text, or chosen from destination activity databases
- Event notifications can be delivered via multiple channels: email, in-app alerts, calendar integration, or SMS
- Events can be automatically categorised (dining, sightseeing, transportation, entertainment, etc.) with appropriate icons and organisation
- Participation tracking can integrate with external calendar systems for comprehensive schedule management
- Event modification history can provide detailed change logs for group coordination and dispute resolution
- Location integration can provide real-time information about venues, weather, and accessibility

Frequency of Occurrence:

- **Create**: High - essential component of detailed trip planning
- **Read**: Very High - users frequently check event schedules and details
- **Update**: Moderate to High - users often modify event details as plans evolve
- **Delete**: Moderate - users remove events due to changes in preferences or logistics

Open Issues:

- How should the system handle events that require booking or have limited availability?
- Should the system offer automated event reminders at configurable intervals before event start times?
- How should conflicting events be visually represented in the itinerary to help users identify scheduling issues?
- Should events have privacy settings for selective visibility within group trips (e.g., surprise events)?
- What level of integration should be maintained with external booking systems for event tickets and reservations?
- How should the system handle event modifications when some group members have already made related bookings?
- Should the system provide alternative event suggestions when users delete popular activities?
- How should event costs be tracked and split among group members for financial transparency?

The Two-Column Variation

Use Case UC-16: Manage Events in Trip Itinerary

Primary Actor: User (Traveller)

Create Event Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user accesses the itinerary page for their trip plan.	
2. The user selects the "Add Event" option.	
	3. The system displays a comprehensive event creation form with activity suggestions for the destination.
4. The user enters detailed event information, including name, date and time, location, description, and duration.	
5. The user specifies event requirements: whether the event is mandatory or optional and selects specific group members or all members.	
6. The user sets event parameters such as cost estimates, booking requirements, and preparation needs.	
7. The user submits the complete event details.	
	8. The system validates the event information for conflicts with transportation, accommodation, and other events.
	9. The system saves the event to the trip itinerary with all specified parameters.
	10. The system sends detailed notifications to relevant group members about the new event with RSVP options.
	11. The system updates the trip timeline to include the new event and checks for optimal scheduling.
	12. Group members receive event details and can indicate their participation status and any special requirements.

Read Event Main Success Scenario:

Actor Action (or Intention)	System Responsibility
13. The user selects "View Events" from the itinerary menu.	
	14. The system displays all existing events with comprehensive details including participation status, timing, and logistics.
15. The user can view individual event details,	

Actor Action (or Intention)	System Responsibility
participant responses, location information, and related bookings.	

1. Update Event Main Success Scenario:

Actor Action (or Intention)	System Responsibility
16. The user selects "Edit" on an existing event entry.	
	17. The system displays the event modification form with all current details populated and edit permissions verified.
18. The user modifies event details such as timing, location, description, participant requirements, or event type.	
	19. The system validates the updated information for schedule conflicts and participant availability.
20. The user submits the event changes with modification reasons if significant changes are made.	
	21. The system saves the updated event information and identifies all affected participants.
	22. The system updates the trip timeline based on the modified event schedule and checks for new conflicts.
	23. The system sends change notifications to affected group members with updated event details and new RSVP requests if needed.
	24. Group members can review changes and update their participation status accordingly.

Delete Event Main Success Scenario:

Actor Action (or Intention)	System Responsibility
25. The user selects "Delete" on an existing event entry.	
	26. The system displays a detailed confirmation dialogue showing event details, participant responses, and potential impacts on the itinerary.
27. The user reviews the deletion consequences, including effects on group	

Actor Action (or Intention)	System Responsibility
members and any associated bookings.	
28. The user confirms the event deletion and optionally provides a reason for removal.	
	29. The system permanently removes the event from the itinerary and all related participant data.
	30. The system updates the trip timeline and identifies any scheduling gaps or opportunities for optimisation.
	31. The system notifies all previously invited group members of the event cancellation with an explanation if provided.
	32. The system logs the deletion with comprehensive details for audit purposes and potential group coordination needs.

Fully Dressed Use Case: Manage Weather Information in Trip Itinerary

Use Case: UC-17 Manage Weather Information in Trip Itinerary

Primary Actor: User (Traveller)

Stakeholders and Interests:

- **User**: Wants accurate and up-to-date weather information to plan appropriate activities, packing, and schedule adjustments
- **App Developers**: Want to provide reliable weather data integration that enhances trip planning decision-making
- **Weather Data Providers**: Want their meteorological information to be accurately represented and properly attributed

Preconditions:

- The user has a registered account and is logged in
- The user has created a trip plan with an itinerary including specific dates and destinations
- The system has access to reliable weather data API services with current and historical information
- **For Update/Delete operations**: The user has existing weather information entries in their itinerary

Success Guarantee (Postconditions):

- Weather information is properly managed (created, read, updated, or deleted) in the trip itinerary
- Weather data is accurately associated with specific dates, locations, and activities in the itinerary
- Users can access comprehensive weather forecasts and historical climate data during trip planning
- Weather information is used to provide intelligent activity suggestions and scheduling recommendations
- All weather data modifications are logged with update timestamps for accuracy tracking
- Weather-based recommendations are updated when weather information changes
- Group members have access to shared weather information for coordinated planning

Main Success Scenario:

Create/Add Weather Information:

2. The user accesses the itinerary page for their trip plan.
3. The user requests comprehensive weather information for the selected destination and travel dates.
4. The system retrieves detailed weather data, including historical climate patterns, current forecasts, and extended predictions for the destination.
5. The system displays comprehensive weather information, including temperature ranges, precipitation likelihood, humidity levels, wind conditions, and seasonal climate patterns.
6. The user reviews the weather information and identifies relevant data for trip planning.
7. The user selects "Add Weather Info to Itinerary" with options to include all dates or specific date ranges.
8. The system integrates selected weather data with the corresponding trip dates in the itinerary.
9. The system analyses weather conditions and offers intelligent activity suggestions based on expected weather patterns.
10. The user can customise weather display preferences and set up weather-based alerts for significant changes.
11. The system saves weather information preferences and creates weather-informed activity recommendations.

Read/View Weather Information:

11. The user selects "View Weather Information" from the itinerary menu or views weather data integrated with daily schedules.
12. The system displays comprehensive weather information organised by date with detailed forecasts and climate context.
13. The user can view weather information in multiple formats: daily summaries, hourly forecasts, or extended climate patterns.

14. The system provides weather context for planned activities and suggests optimal timing based on conditions.

Update Weather Information:

15. The user selects "Refresh Weather Data" or "Update Weather Information" from the weather management options.
16. The system retrieves the latest weather forecasts and climate data from multiple reliable sources.
17. The system compares new weather data with existing information and identifies significant changes.
18. The system updates weather information in the itinerary and highlights any notable forecast changes.
19. The system analyses updated weather conditions and revises activity recommendations if necessary.
20. The user reviews updated weather information and can adjust activity plans based on new forecasts.
21. The system notifies group members of significant weather changes that may affect shared activities.

Delete Weather Information:

22. The user selects "Remove Weather Info" for specific dates or the entire trip.
23. The system displays confirmation options for partial or complete weather data removal.
24. The user confirms weather information removal for selected dates or the entire itinerary.
25. The system removes specified weather data from the selected dates while preserving weather-independent activity information.
26. The system adjusts activity recommendations to remove weather-specific suggestions while maintaining general recommendations.
27. The system updates the itinerary display to reflect weather-independent planning mode.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the weather information management page with user preferences preserved.
3. The user continues the weather management process from the last successful operation.

3a. The system cannot retrieve current weather data:

1. The system displays a detailed error message explaining the data connectivity issue and provides alternative data sources.
2. The system offers options to use cached weather data, try alternative weather services, or access historical climate averages.

3. The user chooses to use available data, retry with different sources, or proceed without current weather information.

4a. The weather forecast indicates severe weather conditions during planned travel dates:

1. The system displays prominent weather warnings with specific risk information and official advisories.
2. The system suggests alternative travel dates, modified activities, or contingency planning options.
3. The user can acknowledge the weather risks and proceed with planning or adjust trip dates and activities accordingly.

5a. The user wants to add weather information for only specific dates or activities:

1. The user selects specific dates from the itinerary calendar or chooses activities requiring weather information.
2. The system adds detailed weather information only for the selected dates or activity periods.
3. The system maintains weather-independent planning for unselected dates while providing weather context where requested.

8a. The user wants more detailed weather information, including specialised data:

1. The user selects "Detailed Weather Analysis" options for comprehensive meteorological data.
2. The system displays specialised weather information, including UV index, air quality, seasonal patterns, and historical weather trends.
3. The user can customise detailed weather displays and set specific weather parameter alerts for optimal trip planning.

16a. Weather data sources provide conflicting forecasts:

1. The system identifies discrepancies between different weather services and displays confidence levels for predictions.
2. The system presents multiple forecast scenarios with probability ranges and recommends the most reliable predictions.
3. The user can choose preferred weather sources or review multiple forecasts for comprehensive planning.

18a. Significant weather changes affect planned activities:

1. The system identifies activities that may be significantly impacted by weather forecast changes.
2. The system suggests specific activity modifications, alternative indoor options, or schedule adjustments.
3. The user can accept suggested changes, modify activities independently, or acknowledge weather risks and proceed with original plans.

24a. Weather information removal affects weather-dependent activity recommendations:

1. The system identifies activities and recommendations that were based on weather data and will be affected by removal.
2. The system warns about activity suggestions that may no longer be relevant without weather context.
3. The user confirms understanding of recommendation changes or chooses to maintain weather data for activity planning.

Special Requirements:

- Weather data must be updated regularly with the latest forecasts and should indicate data freshness and reliability.
- The system must display both short-term forecasts and long-term climate patterns with clear differentiation between prediction types.
- Weather information must be presented with intuitive visual indicators, including icons, colour coding, and graphical representations.
- The system should provide weather-based activity alternatives and suggest optimal timing for outdoor activities.
- Weather alerts and significant forecast changes must be delivered through configurable notification channels.
- International weather data must handle different measurement units and time sones accurately
- Weather information must integrate seamlessly with activity planning and scheduling features.
- Historical weather data should be available for reference and trip planning optimisation.

Technology and Data Variations List:

- Weather data can be sourced from multiple meteorological services, with fallback options for service reliability
- Climate visualisation can include detailed charts, interactive maps, colour-coded calendars, or graphical forecast displays
- Weather alerts can be integrated with push notification systems, email alerts, or in-app notification centres
- Weather data accuracy can be enhanced through machine learning algorithms that analyse multiple forecast sources
- Integration with satellite weather imagery can provide real-time weather visualisation
- Weather-based activity recommendations can use AI to optimise activity scheduling based on conditions

Frequency of Occurrence:

- **Create**: High - essential for comprehensive trip planning and activity scheduling.

- **Read**: Very High - users frequently check weather information throughout trip planning and during travel.
- **Update**: High - users want current weather information as forecasts change and travel dates approach.
- **Delete**: Low - users occasionally remove weather information to simplify planning or when switching to weather-independent activities.

Open Issues:

- How should long-term weather forecasts (beyond 7-10 days) be presented, given their reduced accuracy?
- Should the system offer real-time weather updates and alerts during the actual trip period?
- How should weather probability and uncertainty (chance of rain, forecast confidence) be communicated effectively to users?
- Should the system allow customisation of weather display units (°C/°F, mm/inches, different wind speed units) based on user preferences?
- What level of weather detail is optimal for trip planning without overwhelming users with meteorological data?
- How should the system handle weather information for multi-location trips with different climate sones?
- Should weather-based activity recommendations be automatically updated, or should users control when recommendations change?
- How should the system balance weather accuracy with usability for users who prefer simplified weather information?

The Two-Column Variation

Use Case UC-17: Manage Weather Information in Trip Itinerary

Primary Actor: User (Traveller)

Add Weather Information Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user accesses the itinerary page for their trip plan.	
2. The user requests comprehensive weather information for the selected destination and travel dates.	
	3. The system retrieves detailed weather data, including historical climate patterns, current forecasts, and extended predictions for the destination.
	4. The system displays comprehensive weather

Actor Action (or Intention)	System Responsibility
	information, including temperature ranges, precipitation likelihood, humidity levels, wind conditions, and seasonal climate patterns.
5. The user reviews the weather information and identifies relevant data for trip planning.	
6. The user selects "Add Weather Info to Itinerary" with options to include all dates or specific date ranges.	
	7. The system integrates selected weather data with the corresponding trip dates in the itinerary.
	8. The system analyses weather conditions and offers intelligent activity suggestions based on expected weather patterns.
9. The user can customise weather display preferences and set up weather-based alerts for significant changes.	
	10. The system saves weather information preferences and creates weather-informed activity recommendations.

Read Weather Information Main Success Scenario:

Actor Action (or Intention)	System Responsibility
11. The user selects "View Weather Information" from the itinerary menu or views weather data integrated with daily schedules.	
	12. The system displays comprehensive weather information organised by date with detailed forecasts and climate context.
13. The user can view weather information in multiple formats: daily summaries, hourly forecasts, or extended climate patterns.	
	14. The system provides weather context for planned activities and suggests optimal timing based on conditions.

Update Weather Information Main Success Scenario:

Actor Action (or Intention)	System Responsibility
15. The user selects "Refresh Weather Data" or	

Actor Action (or Intention)	System Responsibility
"Update Weather Information" from the weather management options.	
	16. The system retrieves the latest weather forecasts and climate data from multiple reliable sources.
	17. The system compares new weather data with existing information and identifies significant changes.
	18. The system updates weather information in the itinerary and highlights any notable forecast changes.
	19. The system analyses updated weather conditions and revises activity recommendations if necessary.
20. The user reviews updated weather information and can adjust activity plans based on new forecasts.	
	21. The system notifies group members of significant weather changes that may affect shared activities.

Delete Weather Information Main Success Scenario:

Actor Action (or Intention)	System Responsibility
22. The user selects "Remove Weather Info" for specific dates or the entire trip.	
	23. The system displays confirmation options for partial or complete weather data removal.
24. The user confirms weather information removal for selected dates or the entire itinerary.	
	25. The system removes specified weather data from the selected dates while preserving weather-independent activity information.
	26. The system adjusts activity recommendations to remove weather-specific suggestions while maintaining general recommendations.
	27. The system updates the itinerary display to reflect weather-independent planning mode.

Fully Dressed Use Case: View Feed

Use Case: UC-18 View Feed

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants relevant travel suggestions tailored to their preferences and history
- App Developers: Want to provide valuable recommendations that increase engagement
- Travel Partners: Want their offerings to be included in recommendations
- Data Scientists: Want to improve recommendation algorithms using user feedback

Preconditions:

- The user has a registered account and is logged in
- The user has completed the basic preference questionnaire
- The recommendation system has sufficient data to generate suggestions

Success Guarantee (Postconditions):

- The user receives personalised destination recommendations
- Recommendations include activity suggestions for each destination
- Budget estimates are provided for recommended destinations
- Sentiment summaries show traveller opinions about destinations
- The user can interact with recommendations for further planning

Main Success Scenario:

1. The user navigates to the recommendations section of the application.
2. The system analyses the user's profile, preferences, and interaction history.
3. The system applies the recommendation algorithm with weighted factors:
    - User preferences, travel history, and interactions (40%)
    - Demographic popularity (20%)
    - Budget alignment (15%)
    - Climate preferences (15%)
    - Convenience factors (10%)
4. The system displays a list of recommended destinations sorted by relevance.
5. The system provides key information for each destination, including activities, budget estimates, and sentiment analysis.
6. The user browses through the recommendations.
7. The system tracks user interactions with recommendations to refine future suggestions.

8. The user can select destinations to begin trip planning or save to their bucket list.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the recommendations page.
3. The user continues browsing recommendations.

3a. The system has insufficient user data for personalised recommendations:

1. The system uses general popularity and seasonal data for initial recommendations.
2. The system prompts the user to complete preference questionnaires for better recommendations.

4a. The recommendation model detects conflicting user preferences:

1. The system prioritises more recent preferences and interactions.
2. The system provides a diverse set of recommendations to address different preference aspects.

5a. The user wants more specific recommendation categories:

1. The user selects filters to focus recommendations (e.g., adventure travel, relaxation, cultural experiences).
2. The system adjusts recommendations based on selected filters.

6a. The user provides explicit feedback on recommendations:

1. The user rates or dismisses specific recommendations.
2. The system immediately incorporates feedback into the recommendation algorithm.
3. The system presents refined recommendations.

7a. The user requests detailed information about a recommended destination:

1. The user selects a destination for more details.
2. The system displays comprehensive destination information, including travel requirements, popular activities, and accommodation options.

Special Requirements:

- Recommendations must update dynamically as user preferences change.
- The recommendation algorithm must balance personalisation with exploration of new options.
- The interface must present recommendations in an engaging, visual format.

- Sentiment analysis must aggregate multiple sources for balanced representation.

Technology and Data Variations List:

- Recommendation display can include image galleries, ratings, and quick-action buttons.
- User feedback can be collected explicitly (ratings) or implicitly (viewing behaviour).
- Sentiment visualisation can use numeric scores, star ratings, or text summaries.

Frequency of Occurrence: High - central feature for user engagement.

Open Issues:

- How should the system balance familiar recommendations with novel suggestions?
- Should recommendations include a confidence score to indicate relevance?
- How frequently should recommendations be refreshed with new options?
- Should users be able to customise the weighting of recommendation factors?

The Two-Column Variation

Use Case UC-18: View Personalised Recommendations

Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the recommendations section of the application.	
	2. The system analyses the user's profile, preferences, and interaction history.
	3. The system applies the recommendation algorithm with weighted factors: user preferences/history/interactions (40%), demographic popularity (20%), budget alignment (15%), climate preferences (15%), convenience factors (10%).
	4. The system displays a list of recommended destinations sorted by relevance.
	5. For each destination, the system provides key information including activities, budget estimates, and sentiment analysis.
6. The user browses through the recommendations.	
	7. The system tracks user interactions with recommendations to refine future suggestions.
8. The user can select destinations to begin trip planning or save to their	

Actor Action (or Intention)	System Responsibility
bucket list.	

Fully Dressed Use Case: Filter Feed by Interest

Use Case: UC-19 Filter Feed by Interest

Primary Actor: User (Traveller)

Stakeholders and Interests:

- User: Wants to discover destinations that align with specific interests
- App Developers: Want to provide intuitive interest-based navigation
- Travel Partners: Want their offerings to be categorised accurately by interest
- Content Curators: Want interest categories to showcase appropriate destinations

Preconditions:

- The user has a registered account and is logged in
- The system has a categorised database of destinations by interest
- Interest categories are regularly updated with relevant destinations

Success Guarantee (Postconditions):

- The user views destinations that match their selected interest category
- Interest-specific activities are highlighted for each destination
- The user can interact with interest-based results for further planning
- User interaction data is collected to improve future recommendations

Main Success Scenario:

1. The user navigates to the destination browsing section of the application.
2. The system displays available interest categories: Adventure, Relaxation, Cultural Experiences, and Nature.
3. The user selects an interest category (e.g., Adventure).
4. The system retrieves destinations tagged with the selected interest.
5. The system displays a list of destinations sorted by relevance to the selected interest.
6. For each destination, the system highlights specific activities and attractions related to the interest.
7. The user browses through the destinations in the chosen interest category.
8. The user can filter results further by additional criteria (region, budget, season).
9. The system tracks user interactions with interest-based results to refine future recommendations.
10. The user can select destinations to begin trip planning or save to their bucket list.

Extensions (or Alternative Flows):

*a. At any time, the system fails:

1. The user refreshes the page or restarts the app.
2. The system recovers to the interest browsing page.
3. The user continues browsing destinations.

3a. The user wants to browse multiple interest categories simultaneously:

1. The user selects multiple interest categories.
2. The system retrieves destinations that match any selected category.
3. The system indicates which interests are matched for each destination.

5a. No destinations match the selected interest and filter combination:

1. The system suggests broadening search criteria.
2. The user adjusts filters and continues.

7a. The user wants more detailed information about activities for a specific interest:

1. The user selects an "Explore Activities" option for a destination.
2. The system displays comprehensive activity listings categorised by interest.
3. The user can view detailed activity information.

8a. The user wants to save the current interest search as a preference:

1. The user selects the "Save This Interest" option.
2. The system adds the interest to the user's preference profile.
3. Future recommendations incorporate the saved interest.

10a. The user creates a new trip plan directly from interest browsing:

1. The user selects "Plan Trip" for a destination.
2. The system initiates the trip planning process (UC-12).
3. The system pre-populates the trip with interest-specific activities.

Special Requirements:

- Interest categories must be clearly defined with visual representations.
- The system must maintain current and accurate interest tagging for all destinations.
- The interface must highlight why each destination matches the selected interest.
- Interest browsing must perform efficiently even with complex filtering.

Technology and Data Variations List:

- Interest browsing can include visual galleries, maps, or list views.
- Interest categories can be expanded with sub-categories for more specificity.
- Interest matching can use machine learning to identify emerging patterns.

Frequency of Occurrence: High - primary discovery mechanism for many users.

Open Issues:

- How should interests be weighted when multiple categories apply to a destination?
- Should users be able to create custom interest categories?
- How frequently should interest categorisations be reviewed and updated?
- Should seasonal variations in interest activities be highlighted?

The Two-Column Variation

Use Case UC-19: Browse Destinations by Interest

Primary Actor: User (Traveller)
...as before...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1. The user navigates to the destination browsing section of the application.	
	2. The system displays available interest categories: Adventure, Relaxation, Cultural Experiences, and Nature.
3. The user selects an interest category (e.g., Adventure).	
	4. The system retrieves destinations tagged with the selected interest.
	5. The system displays a list of destinations sorted by relevance to the selected interest.
	6. For each destination, the system highlights specific activities and attractions related to the interest.
7. The user browses through the destinations in the chosen interest category.	
8. The user can filter results further by additional criteria (region, budget, season).	
	9. The system tracks user interactions with interest-based results to refine future recommendations.
10. The user can select destinations to begin trip planning or save to their bucket list.	

Fully Dressed Use Case: Complete Questionnaire

Use Case: UC-20 Complete Questionnaire
Primary Actor: User (Traveller)

Stakeholders and Interests:

- **User**: Wants to provide initial travel preferences to receive personalised recommendations from the start.
- **App Developers**: Want to collect comprehensive preference data during user onboarding for better personalisation.

- AI Model Development Team: Need initial user preference data to train and improve recommendation algorithms

- **Marketing Team**: Want to understand user travel patterns and preferences for targeted offerings.

- **UX Team**: Want to create a smooth onboarding experience that engages users without overwhelming them.

Preconditions:

- The user has successfully registered and verified their account
- This is the user's **first login** to the system
- The user has been authenticated and logged in
- The system has detected that this is a first-time login session

Success Guarantee (Postconditions):

- The user's initial travel preferences are captured and saved to the system
- A complete Preference record is created with all questionnaire responses
- The user's data sharing preference is set based on questionnaire completion
- The recommendation system is initialised with user preference data
- The user proceeds to their personalised dashboard with tailored recommendations
- The questionnaire completion flag is set to prevent future questionnaire prompts

Main Success Scenario:

1. The user completes the login process on their first login.
2. The system detects that this is the user's first login and displays a welcome message.
3. The system presents the travel preference questionnaire with an explanation of its purpose for personalisation.
4. The system displays the multi-step questionnaire form with the following sections:
    1. **Budget Range**: Lower and upper budget limits for trips
    2. **Primary Travel Interest**: Adventure, Relaxation, Cultural Experience, or Nature
    3. **Travel Style**: Casual, Frequent, Business, Enthusiast, or Organiser
    4. **Preferred Climate**: Tropical, Dry, Continental, Polar, Mediterranean, Arid, Semi-Arid, Monsoon, or Tundra
    5. **Data Sharing Consent**: Agreement to use responses for personalised recommendations
5. The user completes all questionnaire sections.
6. The user reviews their responses and submits the questionnaire.
7. The system validates all questionnaire responses.
8. The system creates a new Preference record with the user's responses.
9. The system sets the user's data sharing preference to "true" (indicating consent for personalisation).
10. The system marks the questionnaire as completed for this user.
11. The system displays a completion confirmation message.
12. The system redirects the user to their personalised dashboard with initial recommendations.

Extensions (or Alternative Flows):

***a. At any time, the system fails:**

1. The user refreshes the page or restarts the app.
2. The system recovers to the questionnaire page with any completed sections preserved.
3. The user continues from where they left off.

3a. The user selects "Skip Questionnaire" or "Complete Later":

1. The system displays a warning message about generic recommendations.
2. The system explains that skipping will result in non-personalised content.

3. The system asks for confirmation: "Are you sure you want to skip? This will affect your experience." 4a. User confirms skip:
    - The system sets data sharing preference to "false"
    - The system marks questionnaire as "skipped"
    - The system redirects to dashboard with generic recommendations
    - The system displays a notice about enabling personalisation in Settings 4b. User cancels skip:
    - Return to step 4 (continue questionnaire)

5a. The user provides incomplete questionnaire responses:

1. The system identifies missing required fields.
2. The system highlights incomplete sections with error indicators.
3. The system displays specific messages for each missing field.
4. The user completes the missing information and continues.

5b. The user sets an invalid budget range (lower > upper):

1. The system validates budget range logic.
2. The system highlights the budget fields with error message.
3. The system suggests correcting the range (e.g., swap values or adjust limits).
4. The user corrects the budget range and continues.

6a. The user wants to change responses before submitting:

1. The system allows navigation between questionnaire sections.
2. The user modifies previous responses.
3. The system updates the review summary with changes.
4. The user submits the updated questionnaire.

7a. System validation detects inconsistent responses:

1. The system identifies potential conflicts (e.g., adventure interest with relaxation style).
2. The system displays clarification questions or suggestions.
3. The user confirms responses or adjusts.
4. The system proceeds with validated responses.

8a. Database save operation fails:

1. The system displays an error message about save failure.
2. The system retains all questionnaire responses in the session.
3. The system offers to retry saving or contact support.
4. The user can retry or complete the questionnaire later through Settings.

Special Requirements:

- The questionnaire must be **only available on the first login** - subsequent logins skip this step
- All questionnaire sections must be completed before submission (no partial completion allowed)
- Budget range validation must ensure lower limit ≤ upper limit and both are positive integers
- The questionnaire must be completable within 5-10 minutes for a good user experience
- Questionnaire responses must immediately enable personalised recommendations
- The system must prevent repeated questionnaire completion attempts
- A clear explanation of data usage must be provided before the questionnaire begins
- The interface must be mobile-responsive for various device types

Technology and Data Variations List:

- Budget inputs can use sliders, number fields, or currency selection interfaces
- Interest and style selections can use visual cards, radio buttons, or drop-down menus

- Climate preferences can display weather icons, descriptive text, or map visualisations
- Progress indicators can show completion status across questionnaire sections
- Data sharing consent can include detailed privacy policy links and explanations

Frequency of Occurrence: Very Low - This occurs exactly **once per user** during their first login experience.

Open Issues:

- Should the questionnaire be mandatory or always skippable?
- How detailed should the budget range options be (specific amounts vs. general ranges)?
- Should there be additional questions about travel frequency, group vs. solo preferences, etc.?
- How should the system handle users who abandon the questionnaire mid-completion?
- Should questionnaire responses be revisitable/editable immediately after completion?
- What happens if a user creates multiple accounts - should the system detect and prevent duplicate questionnaires?

The Two-Column Variation

Use Case UC-19: Complete Questionnaire

Primary Actor: User (Traveller)

Main Success Scenario:

Actor Action (or Intention)	**System Responsibility**
1. The user completes the login process on their first login.	
	2. The system detects the first login and displays a welcome message with a questionnaire explanation.
	3. The system presents the travel preference questionnaire form.
	4. The system displays a multi-step questionnaire (budget, interests, style, climate, data sharing).
5. The user completes all questionnaire sections with their preferences.	
6. The user reviews responses and submits the questionnaire.	
	7. The system validates all questionnaire responses (budget range, required fields).
	8. The system creates a new Preference record with the user's responses.
	9. The system sets the data sharing preference to "true" and marks the questionnaire as completed.
	10. The system displays a completion confirmation message.
	11. The system redirects to personalised dashboard with

Actor Action (or Intention)	System Responsibility
	initial recommendations.

Important Note: This use case is **exclusively for first-time login**. After the initial questionnaire completion, all preference modifications must be done through "UC-5: Manage Travel Preferences" which allows ongoing updates to travel preferences throughout the user's account lifecycle.

3 Non-Functional Requirements

3.1 Performance Requirements

- **NFR-1.1**: The system must handle up to 10,000 concurrent users without performance degradation. This requirement ensures the system can manage high traffic volumes during peak usage times, maintaining responsiveness and user satisfaction.
- **NFR-1.2**: Under normal operating conditions, the system must process user requests and display responses within 3 seconds. This requirement is critical for user experience, as delays beyond this threshold can lead to user frustration and abandonment.
- **NFR-1.3**: The system must ensure that the average page load time does not exceed 2 seconds. Faster page load times contribute to a smoother user experience and better search engine optimisation (SEO) performance.

3.2 Safety and Security Requirements

Safety Requirements:

- **NFR-2.1**: The system must implement a predictive model to continuously monitor the health of system processes and resources, providing early warnings of potential issues to avoid unsafe states. This requirement ensures proactive identification and mitigation of risks leading to system failures or data loss.
- **NFR-2.2**: The system must employ condition monitoring to check the operational status of processes and devices, using assertions and key performance indicators to detect conditions that may lead to hasardous behaviour. This requirement enables the system to identify and address unsafe states before they result in harm or damage.
- **NFR-2.3**: The system must use timestamps to track the sequence of events in distributed environments, detecting incorrect sequences that might indicate a fault or an unsafe state. This requirement helps identify and address issues related to timing and synchronisation, thereby preventing undesired states.
- **NFR-2.4**: The system must implement redundancy through replication and functional redundancy to protect against hardware failures and common-mode failures. This requirement ensures system availability and data integrity in the event of component failures.

- **NFR-2.5**: The system must include a rollback mechanism that allows reverting to a previously saved, known good state upon detecting a failure, combined with checkpointing and transactions. This requirement enables the system to recover from transient faults and maintain system integrity.

Security Requirements:

- **NFR-2.6**: Data Encryption: The system must encrypt all user data at rest and in transit using industry-standard encryption protocols (e.g., AES-256 for data at rest and TLS 1.2+ for data in transit). This requirement ensures the confidentiality and integrity of user data.

- **NFR-2.7**: Compliance with Data Protection Regulations: The system must comply with relevant data protection regulations, including GDPR and CCPA, to ensure the privacy and rights of users are respected. This requirement is essential for legal compliance and building trust with users.

- **NFR-2.8**: User Data Access Control (DAC) for Recommendation Model: The system must provide users with discretionary access control (DAC) over the data they share for use in the recommendation model. Users must have the ability to specify which data elements can be utilised, and they must be able to modify these permissions at any time. This requirement ensures that users maintain control over their data and how it is used within the system, enhancing privacy and trust.

3.3 Other Software Quality Attributes

- **NFR-3.1: Maintainability** - The system must be designed with a modular architecture, allowing for easy updates and maintenance. This will be achieved by separating concerns into distinct modules, using well-documented APIs for inter-module communication, and implementing a version control system to track changes. This approach will facilitate adding new features and resolving issues without affecting the overall system stability.

- **NFR-3.2: Usability** - The system must have an intuitive and user-friendly interface, with a maximum of 3 clicks to reach any primary function. To achieve this, the design will follow established usability principles, such as clear navigation, consistent layout, and user-friendly terminology. User testing and feedback will be incorporated throughout development to ensure the interface meets user expectations and needs.

- **NFR-3.3: Design for Change** - The system design must allow easy adaptation to changing requirements and technologies. This will be accomplished using a microservices architecture, enabling individual components to be updated or replaced without affecting the entire system. Additionally, the system will employ configuration management and feature toggles to allow for seamless deployment of new features and experimentation with different configurations.

- **NFR-3.4: Availability** -

4 Other Requirements

*<This section is **Optional**. Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalisation requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

To do: User Stories and Requirement-Level Models

Appendix

<Please include here a description of what was done to elicit requirements, include dates of interview and who was interviewed, questionnaires, interview logs, observation activities, client responses etc.
Essentially, this section should provide proof of activities that resulted in the compilation of this document>
Failure to include supporting evidence here may be construed as evidence that requirements elicitation and analysis activities were not done!!