System Requirements Specifications (SRS)

for

Wanderers

Hazim Khoiruddin
Isaac Chun Jun Heng
J'sen Ong Jia Xuan
Kim Seo Jin
Leong Mininn Miko
Raghav Rajendran Nair
Yu Zi Hao Albert

Team Wanderers
College of Computing and Data Science, Nanyang Technological University

Submitted to—
Goh Tong Hai
Li Shenggui
College of Computing and Data Science, Nanyang Technological University

1 Table of Contents

2 Problem Statement

Planning group trips can often be a stressful and inefficient process. Travelers, especially when in a group, rely on multiple fragmented tools such as messaging apps, spreadsheets, and third-party booking sites to handle various aspects of the trip, such as accommodation, finances, schedule planning and communication, to name a few. Using multiple fragmented tools often leads to miscommunication, budgeting issues, and logistical confusion. Furthermore, managing shared expenses among group members is often cumbersome, requiring manual tracking and calculations that can result in disputes and misunderstandings.

Without a unified system for travel management, these fragments operate in solos, leading to limited interoperability. As a result, users spend a significant amount of time switching between these apps, waiting for feedback from others on a potential item, finding places to go, the list goes on. This causes the planning time to sometimes extend beyond the intended departure date or take up a considerable amount of time.

Additionally, there has been a significant shift in social trends with the rise of free and easy travel. It is no longer uncommon to go on a trip with a bunch of strangers met during school semester exchange or in the middle of a solo trip. The birth and success of local companies such as “Sotravel” highlights the strong appetite for social adventures as a community in Singapore. When travelling with new people, the planning process becomes even more daunting.

These social shifts and the general rise of travelling highlight the clear need for a comprehensive, collaborative platform. Wanderers addresses the problem of constructing an all-in-one solution that simplifies itinerary coordination, centralizes group discussions, and automates cost-sharing, ultimately enhancing travel experiences, especially for groups.

3 Overview

3.1 Background

Travel planning is an essential yet complex aspect of group trips, requiring seamless coordination across multiple individuals. While various tools such as messaging apps, spreadsheets, and expense trackers exist, they operate in silos, leading to inefficiencies in managing itineraries, costs, and communication.

A significant challenge travellers face is the lack of a centralized platform that integrates itinerary management, communication, and expense tracking. Many travellers rely on WhatsApp for discussions, Excel for planning, and Splitwise for expense sharing, resulting in fragmented trip organization.

The demand for collaborative travel solutions has grown with the increasing digitalization of travel planning. However, existing applications primarily focus on either social networking (e.g., TripAdvisor, Instagram) or individual itinerary creation (e.g., Google Trips) rather than providing a comprehensive, real-time collaborative experience.

To address these gaps, Wanderers is designed as a one-stop collaborative travel planning platform where travellers can collaboratively build itineraries, communicate within the app, and manage expenses in real-time. By consolidating these functionalities, Wanderers streamlines the planning process, reducing friction in group coordination and ensuring a smoother travel experience.

3.2 Overall Description

Wanderers is a user-friendly collaborative travel planning platform to streamline itinerary planning, group collaboration and budgeting processes. The software items included in this plan encompass all components of the application. The frontend development is created with Next.js for responsive and interactive user experience. The backend development is powered by Express.js, which ensures efficient API handling and seamless data management. Supabase is used as the database for synchronization and secured data storage. Lastly, the application integrates external APIs such as Google Maps/Places API for location services and travel recommendations, which are the features of the application. The intended use of this application is to offer users a convenient, simplified travel planning platform with features such as collaborative editing and transparent budget splitting.

4 Investigation & Analysis Methodology

4.1 System Investigation

Upon evaluation, there are significant usability and functionality gaps in the current group travel planning environment. Existing solutions often rely on disjointed tools such as spreadsheets, messaging apps, and third-party expense trackers, leading to fragmented communication and inefficient coordination. Travelers frequently struggle with synchronizing itinerary updates, tracking shared expenses accurately, and maintaining a centralized discussion space for their trip.

Additionally, the lack of a dedicated collaborative platform for trip planning results in miscommunication, duplicate planning efforts, and confusion over cost contributions. Many available solutions do not offer real-time itinerary updates, forcing users to manually adjust plans across multiple platforms. These inefficiencies highlight the need for Wanderers, which aims to provide a unified, interactive, and user-friendly solution for seamless trip coordination, expense management, and group communication.

4.2 Analysis Methodology

4.2.1 Feasibility study and requirements elicitation

The successful development of the Wanderers application requires the formation of a project team with expertise in software development, UI/UX design, and those with a vested interest in travelling and planning group trips. Throughout the project, this team will facilitate continuous collaboration, iterative feedback, and refinement of features.

A critical step in the analysis phase involves interviewing key stakeholders, including frequent travellers, travel planners, and finance-conscious users. These interviews will provide insights into existing challenges in group travel planning, such as coordination difficulties, expense tracking inefficiencies, and fragmented communication tools. Discussions with UX designers and software engineers experienced in developing interactive and collaborative platforms will help assess the practicality, scalability, and usability of the proposed features.

A Feasibility and Risk Assessment study will be conducted to evaluate the viability of Wanderers, focusing on technical feasibility, user experience optimization, financial considerations, data security compliance, and potential implementation risks. This study will ensure that Wanderers is developed as a robust, intuitive, and effective solution for modern travel planning needs.

4.2.2 System analysis and requirements specification

4.2.2.1 Perform an analysis of the problem using object-oriented techniques

An external view of the Wanderers system will be developed using Unified Modelling Language (UML) to represent the core components and their interactions. This will include user roles, itinerary management, expense tracking, and real-time collaboration functionalities. This System Requirements Specification (SRS) document will form part of the documentation for the project.

Some desired features of the **Wanderers** platform include:

- **Create, view, and edit itineraries** collaboratively
- **Adding/removing** itineraries
- **Manage shared expenses** with automatic cost-splitting
- **Real-time synchronization** of itinerary updates across group members
- **Chat function** for in-app communication
- **Accessibility** across desktop and mobile devices with internet access

4.2.2.2 Scope and Limitations

4.2.2.2.1 Business Analysis

- **Business Rules**

  Users must create an account to access collaborative features. The group members should have equal access to shared trip plans unless restricted by the trip owner. There should be an automated expense calculations to ensure fairness and transparency.

- **Business System Interfaces**

  Wanderers is a collaborative travel planning platform, an integration with third-party APIs such as Google Maps for locations services is needed. JSON Web Token (JWT) is used to ensure secure authentication and authorization on our platform.

- **Business Function**

  We aim to provide a centralized platform for trip planning, expense sharing, and itinerary management. This helps to facilitate real-time collaboration and communication between group members. It can also automate financial reconciliation to avoid disputes or conflicts between users of our application and reduce their effort.

- **Business Ownership and Sponsorship**

  The project is owned and maintained by the Wanderers development team. Funding sources may include internal financing, grants, or partnerships with travel agencies.

- **Budget Requirements**

  We require the infrastructure of hosting, database, and security services. Tools, and third-party API integrations for the development of the web application. Maintenance & support for future web application update is required.

4.2.2.2.2 Requirement Analysis

- **System I/O Description**

  The system takes user inputs such trip details, expenses and chat messages while generating outputs like itineraries, expense breakdown and chat conversation.

- **User Requirements**

  Users should have the ability to create, edit, and share trip itineraries collaboratively, track expenses and communicate with other collaborators in real time.

- **Functional Requirements**

  At a high level, there should be user authentication and role-based access control for read/writing privileges. Users should also be able to handle trip creation and management with convenience. Additionally, there should be an integrated expense tracking and settlement in the application.

- **Security Requirements**

  This project requires a secure storage and encryption of sensitive user data, secure API communication with external services. There should also be

role-based permissions for all users for security and fine-grained control of actions users can take.

4.2.2.2.3 Data Analysis

- **Data Collection Process and Validation**

  The data collection process generally involves getting user input via forms and uploaded files with API integration for real-time travel data. Form validation is also required to ensure data integrity as well as verification of transaction records for expenses.

  User-generated data is stored in a PostgreSQL database hosted on Supabase. Form validation is also required for such data to ensure data integrity as well as verification of transaction records for expenses.

4.2.2.2.8 Process Analysis

- **Data/Process Flow Analysis**

  The user request-response cycle involves frontend interactions (Next.js), backend processing (Express.js) and data retrieval from Supabase.

- **Process Decomposition**

  Breaking down features into modular services, including trip planning, expense tracking, and messaging.

- **System Interfaces**

  The system connects with Google Maps API and internal APIs for user authentication and itinerary updates.

4.2.2.2.9 Application Architecture

- **Information Structure**

  Data is structured into users, itineraries, activities and APIs to ensure efficient retrieval and updates.

- **Usability**

  The system follows responsive design principles for accessibility across desktop and mobile devices.

- **UI Design**

Our user interface is developed with Next.js built on the React framework and styled using Tailwind CSS, while integrating third-party component libraries such as Shadcn/ui to create a seamless, user-friendly interface for trip planning and expense sharing.

- **Implementation**

  The backend, built with Express.js, connects to a PostgreSQL database via Supabase and communicates with the Next.js frontend using the Fetch API, ensuring seamless real-time synchronization between the various systems.

4.2.3 Object-oriented design using UML

A comprehensive object-oriented design for the Wanderers application will be developed using UML (Unified Modelling Language) for both graphical representation and documentation. This design will focus on the core functionalities of collaborative trip planning and expense management, capturing the interactions between users and the system. At its core, users will interact with a web-based interface to create, modify, and manage travel itineraries, facilitated by real-time updates and synchronization with the back-end system.

Additional features will include collaborative editing of trip plans, built-in discussion rooms for group communication, and an integrated cost-splitting system to track and divide shared expenses. Security will be ensured through user authentication mechanisms, including account-based access and role-based permissions for trip organizers and participants.

4.2.3.1 Use Case Diagram

The below Use Case diagram for Wanderers application illustrates how users and the system's functionalities interact within the system.

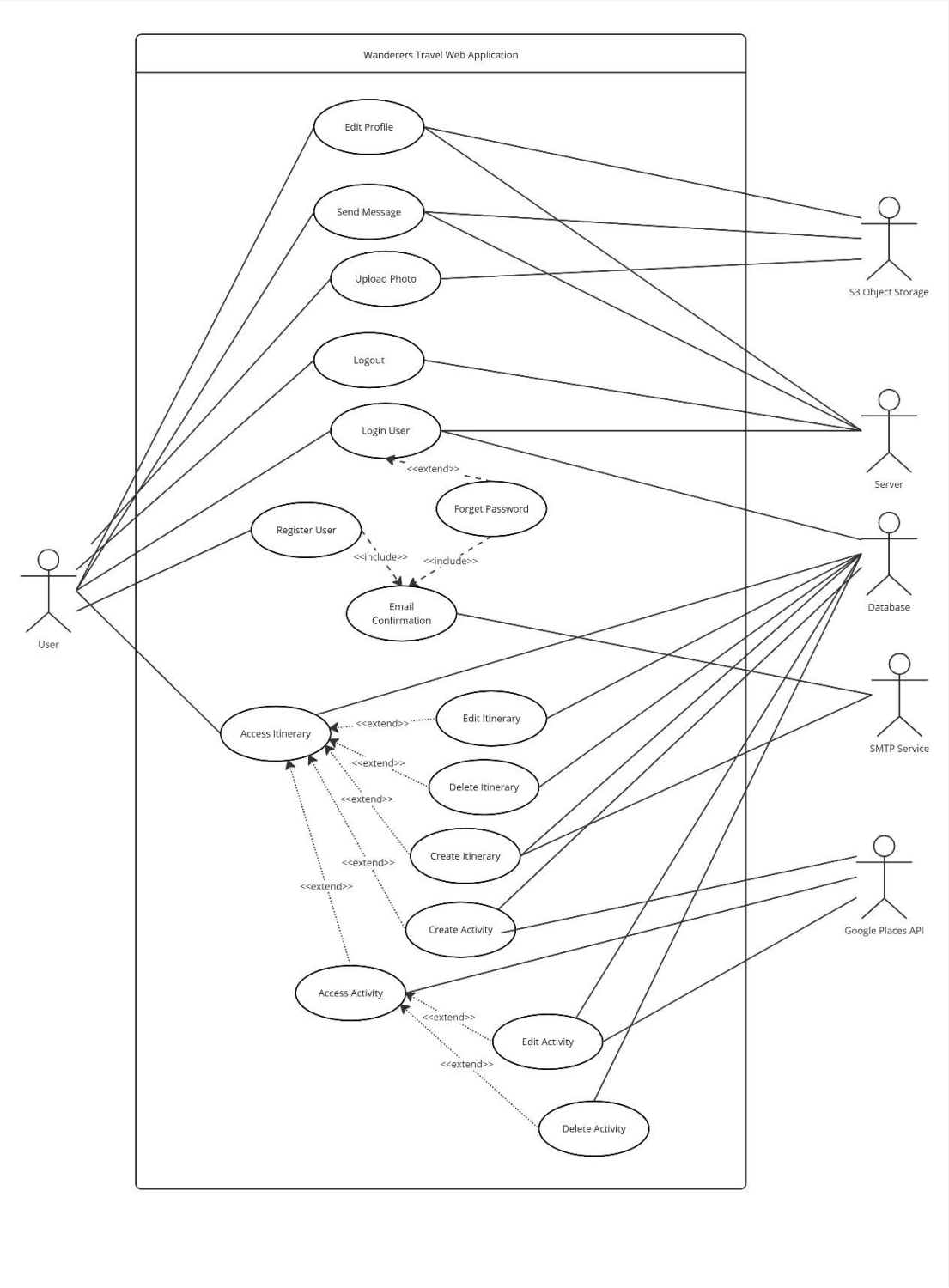

Figure 1: UML Diagram for Wanderers

4.2.4 Prototyping

The Object-Oriented Rapid Prototyping (OORP) method will be used to develop a limited yet functional prototype of the Wanderers application. This prototype will serve as a proof of concept, allowing stakeholders to evaluate key functionalities before full-scale development.

The prototype will focus on demonstrating:

- **Trip Creation and Itinerary Management** – Users will be able to create, edit, and share itineraries.
- **Collaborative Editing and Cost Splitting** – A simplified version of expense tracking and shared contributions.
- **Chat function** – Basic chat functionality within a travel group.

Primary elements to be included in the prototype:

- **Web-based Forms**: These will act as the primary method of input for users, allowing them to create trips, modify itineraries, and split expenses.
- **Database and API Interaction**: The forms will interact in real-time with a back-end database, showcasing the application's ability to store, retrieve, and update travel plans and expense data.
- **User Interface Mock-ups**: These will illustrate the proposed design and layout of the Wanderers interface, providing stakeholders with a visual understanding of the user experience.

The prototype will be developed with a focus on flexibility to accommodate feedback and iterative improvements.

Key objectives during this phase:

- Validate the application's technical feasibility and functional alignment with user needs.
- Gather early user feedback to inform subsequent design and development phases.
- Provide hands-on experience for stakeholders to interact with the concept, ensuring alignment with business objectives and user expectations.

The prototype will be implemented as a web-based interface connected to a lightweight database to simulate data persistence and user interactions. It will be presented to the

development and implementation team for feedback, refinement, and validation of the core system requirements.

5 Constraints

5.1 Scalability

Scalability is a critical factor in the development of the Wanderers application, as it needs to accommodate an increasing number of users, trip data, and real-time collaboration requests. The application must be designed to handle multiple concurrent users modifying itineraries and managing expenses simultaneously.

To ensure efficient data retrieval and synchronization, the system architecture should support real-time updates, seamless integration with cloud-based databases, and optimized network performance. Additionally, asynchronous processing techniques will be explored to minimize lag in collaborative features such as itinerary editing and expense tracking.

5.2 Project Schedule

The project's planning, development, implementation, and quality assurance will take place over three months from January 2025 to March 2025.

5.3 API Limitations

Our Google Places API is protected with a rate limitation library to prevent users from repeatedly making calls to it. Users are given 20 requests per 10 minutes before they are rate limited. When rate limited, affected users will be unavailable to create any Activities within an Itinerary until the rate limit request window is reset.

Our webapp also requires users to be authenticated before they can fully use the webapp's features as a valid JWT token is needed to access the protected API. Meanwhile, unauthenticated users will only be able to access the public API routes, limiting the webapp's functionalities.

5.4 Data Protection

In alignment with prevailing data protection regulations, including but not limited to the Personal Data Protection Act (PDPA) of Singapore. Compliance with these regulations is imperative, necessitating explicit user consent for data collection, transparent communication about the purpose of data processing, and the implementation of robust security measures to protect personal information. The application must align with the respective data protection laws in all jurisdictions where it operates, ensuring a privacy-centric approach and enabling users to exercise control over their data as mandated by the applicable regulatory frameworks.

6 Operational Requirements

6.1 Help Desk Support

Users will have access to customer support for any technical issues related to system performance, account access, or feature-related inquiries. Support channels will include email assistance and a knowledge base page with frequently asked questions. Future enhancements may include AI-powered chatbots for instant troubleshooting.

6.2 Application Services and Technical support

The development team will have continuous access to source code for bug fixes and system improvements. Dedicated network administrators and database engineers will ensure system stability, data integrity, and real-time synchronization between users. Scheduled maintenance windows will be implemented to ensure minimal downtime.

6.3 Administration Features

System access and security levels will be managed through role-based authentication. Users will have permissions based on their roles (e.g., trip organizer vs. participant), ensuring secure access to trip data and expense records. Only authorized administrators will have full access to system logs and user management settings.

6.4 System Interface independence

Wanderers will function independently of any external travel platforms while allowing integration with third-party APIs (e.g., Google Maps for location-based itinerary features and financial tools for expense management). The platform will remain operational even if external integrations are temporarily unavailable.

6.5 System hardware fail over and routine back up

Computer operations centre will handle system hardware tasks such as data tape back-up, hardware maintenance, fail over, scheduled system patches and maintenance.

Regular data backups will be performed to prevent loss of trip plans and expense records. The system will employ cloud-based storage solutions to ensure high availability, disaster recovery, and failover mechanisms in case of unexpected outages. A reliable server will be utilised to ensure minimal effort is required for servicing.

6.6 Audit Trail

All critical user actions (such as itinerary modifications, expense updates, and user role changes) will be logged with timestamps and user identifiers. This ensures transparency and accountability, allowing users to track changes made to trip details and expenses over time.

7 Functional Requirements

The Wanderers application is a comprehensive system designed to streamline the process of planning, managing, and coordinating group travel. It offers a range of self-service features that allow users to collaboratively organize trips, track expenses, and manage itineraries efficiently. All system interfaces adhere to ISO-accepted industry standards for the World Wide Web, ensuring accessibility, security, and seamless user experience across various devices.

7.1 Functionalities

The app includes the following functionalities:

7.1.1 Register

7.1.1.1 The system must allow users to register using an **email** and **password** authentication method.
7.1.1.2 The system must validate that the **email** provided by the user must be unique and valid.
7.1.1.2.1 Must contain a '@' and '.com' in the email.
7.1.1.3 The system must validate that the **password** provided by the user must meet the following conditions:
7.1.1.3.1 Must contain at least one uppercase letter.
7.1.1.3.2 Must contain at least one lowercase letter.
7.1.1.3.3 Must contain at least one digit.
7.1.1.3.4 Must contain at least one special character.
7.1.1.3.5 Must be at least 8 characters long and not exceed 20 characters.
7.1.1.4 The system must generate a unique verification token and send a verification link to email provided via SMTP after register.
7.1.1.5 The system must expire a verification token if the user has not confirmed their registration within 2 days.
7.1.1.6 The system must redirect the user to the authenticated homepage for login after user has clicked on the account confirmation link.

7.1.2 Login

7.1.2.1 The system must allow users to log into their account by entering their email and password.

7.1.2.2 The system must authenticate the **username** and **password** entered by the user against the credentials stored in the database.

7.1.2.6 The system must grant access to the user and redirect them to the homepage if their credentials are validated.

7.1.2.7 The system must prevent a user from logging in when incorrect credentials are supplied.

7.1.2.8 The system must hash the user's password securely during authentication.

7.1.2.9 The system must allow the user to reset their password if the user has forgotten their password.

7.1.2.10 The system shall authenticate the user within **5 seconds** after the login attempt is made.

7.1.3 Logout

7.1.3.1 The system must destroy the user's active session and clear any session data, including authentication tokens.

7.1.3.2 The system must redirect the user to the login page upon successful logout.

7.1.3.3 The system must not allow users to access any protected pages until they log in to the system again.

7.1.3.4 The system shall handle logout requests within **10 seconds**.

7.1.4 Reset / Change Password

7.1.4.1 The system must allow users to reset their password using a secure email-based authentication method.

7.1.4.2 The system must validate the email address entered:
- 7.1.4.2.1 It follows the correct email format.
- 7.1.4.2.2 It belongs to a registered user in the system.

7.1.4.3 The system must generate a unique password reset token and store it securely in the database upon successful validation

7.1.4.4 The system must send an email to the user with a password reset link containing the unique token via SMTP when it is requested.

7.1.4.5 The user must be allowed enter a new password when resetting their password provided it hits the following criteria:

- 7.1.4.5.1 The password entered is not the same as the last three used password.
- 7.1.4.5.1 Must contain at least one uppercase letter.
- 7.1.4.5.2 Must contain at least one lowercase letter.
- 7.1.4.5.3 Must contain at least one digit.
- 7.1.4.5.4 Must contain at least one special character.
- 7.1.4.5.5 Must be at least 8 characters long and not exceed 20 characters.

7.1.4.6 The system must update their credentials and invalidate the reset token upon successful reset by user.

7.1.5 Profile Management

7.1.5.1 The user must be able to view and edit their profile details, including name, and profile picture.
7.1.5.2 When user inputs their username and the cursor exit the input box, the following must happen:

- 7.1.5.2.1 The system must validate if the username is already taken.
- 7.1.5.2.2 The system must not allow saving of changes if the username is not valid.

7.1.5.3 The user must be able to change their profile picture.
7.1.5.4 The system must allow image file types of only .png and .jpg
7.1.5.5 The system must validate that the image file size is below 10 MB.
7.1.5.6 The system must allow users to delete their profile picture.

7.1.6 Trip Planning

7.1.6.1 The user must be able to create a new trip by specifying trip details such as destination, duration, and group members.
7.1.6.2 The system must validate the following before a trip/itinerary can be created

- 7.1.6.2.1 The destination is a valid destination in the world.
- 7.1.6.2.2 The duration has a valid time range and is at least 1 minute.

7.1.6.3 The user must be able to add, edit, and delete activities in the itinerary
7.1.6.4 The user must be able to attach images or links to each activity.
7.1.6.5 The system must automatically save changes in the itinerary when changes are made by an authorised user.
7.1.6.6 The user must be able to invite other members to collaborate on an itinerary
7.1.6.7 The system shall display real-time updates when an itinerary is modified within **10 seconds.**

7.1.7 Cost Splitting

7.1.7.1 The user must be able to input trip-related expenses with details such as amount, category, and payer.
7.1.7.2 The system must provide an option for equal cost splitting or do not split.
7.1.7.3 The users must be able to adjust individual contributions for each expense.
7.1.7.4 The system must generate a summary of each member's owed amount.
7.1.7.5 The user must be able to mark expenses as paid manually.
7.1.7.6 The system must provide a notification when an expensive is marked as settled.

7.1.8 Discussion & Communication

7.1.8.1 The system must allow users to send messages to each other in a chatroom
7.1.8.2 The user must be able to send text messages in any language in the chatroom.
7.1.8.3 The system shall convey messages from client to client within **1 second.**

8 Input Requirements

8.1 User identifier key and user access

Each user is assigned a globally unique identifier upon account creation in the Wanderers application. Users must verify their account through email confirmation. A randomly generated token will be sent to the user's email address. Only when the user verifies that the email is valid and belongs to the user, will the account be activated. This identifier is also used to securely map their trip itineraries, expenses, preferences, and chat history within the system. User accounts may be deactivated upon request or due to prolonged inactivity.

8.2 Expense Logging

User will enter trip expenses, specifying details such as currency, amount, payer, payee and category. The system will validate the amounts are numerical and in correct format. Expense entries will be stored in the database and be synchronised for all members automatically.

8.3 Action Codes

All user actions, such as creating a trip, modifying an itinerary or adding expenses, will be logged with unique transaction codes. These codes serve as a reference for tracking user activity, debugging issues, and ensuring system accountability. Transaction logs will be stored securely in the backend database and can be used for generating usage reports and audits when necessary.

9 Process Requirements

The following are among the inherent requirements that Wanderers system must be able to handle.

9.1 Database transaction

The system must be able to send, receive, and trigger transactions to the Supabase for user authentication, trip management, expense tracking, and itinerary updates. All database operations will follow ACID (Atomicity, Consistency, Isolation, Durability) principles.

9.2 Data integrity

The application will commit fully completed transactions and rollback any unfinished or time out transactions to maintain data integrity.

9.3 Data validation

Data errors from the user's end and from the back-end database-processing end must be gracefully handled. There will be data validation and error-handling routines as part of the Wanderers System.

9.4 Performance

The system will be optimized to handle high concurrency and provide 24/7 availability to resolving locking issues. It must send, receive and display user messages to assist the over-all user experience.

9.5 Data repository

The Wanderers application will store all user, itinerary, and financial data in a Supabase-hosted PostgreSQL database. All uploaded files and images will be stored in Supabase Storage for efficient retrieval and backup.

10 Output Requirements

10.1 Transaction summary and confirmation

Users will receive a summary and confirmation after performing key actions, such as creating or updating an itinerary, logging expenses, or modifying trip details.

10.2 Exception reports

The system will generate exception reports to document anomalies or unusual activities such as multiple failed login attempts, API request failures and data fetching errors. These reports will help in troubleshooting and security monitoring.

10.3 Registration Reports and summaries

Administrators will have access to real-time and historical reports summarizing user activity, trip planning trends, and system performance metrics. These reports will assist in analyzing platform engagement, identifying popular features, and optimizing the user experience.

11 Hardware Requirements

11.1 Network

Stable internet connection is required to access and operate Wanderers.

11.2 Client Computers

The Wanderers application will be accessible on a variety of devices, including smartphones, tablets, laptops, and desktop computers.

The platform is accessible on Mac, Windows, and Unix-based client computers.

The platform will support the latest versions of popular internet browsers (e.g., Chrome, Safari, Firefox, Edge) across both Android and iOS mobile devices. The application will be optimized for different screen sizes and resolutions to ensure a responsive and user-friendly interface for all device types.

11.3 Server

Robust server infrastructure to host the application, database, and related services.

11.4 Production support systems

Wanderers utilises Vercel deployments for production and staging environment previews, allowing developers to preview their progress on a per commit/pull-request basis. Other than that, we intend to host our server on an Azure Linux machine that can handle requests and act as a central system, fully utilising Microsoft Azure's consistent uptime. This infrastructure includes automated backups and helps keep Wanderers running 99.9% of the time.

12 Software Requirements

12.1 Client Operating Systems

- UNIX (any flavour)
- MacOS
- Windows

12.2 Client Application

Wanderers is compatible with all major browsers:

- Google Chrome
- Microsoft Edge
- Safari
- Opera
- Mozilla Firefox

12.3 Network system

Network software and protocols for systems to communicate:

- TCP/IP
- HTTP
- HTTPS
- FTP

12.4 Backend system

The application backend is built with Express.js, a lightweight Node.js framework, to handle API requests and manage business logic. Supabase, a PostgreSQL-based database, is used to store data as it provides real-time synchronization and efficient querying capabilities. Supabase Storage serves as the file and media storage for users to upload and retrieve documents.

12.5 Licenses

All third-party software used in the development, testing, and production of the application will be properly licensed, ensuring compliance with legal and industry standards.

13 Deployment Requirements

The deployment of the Wanderers application is a crucial phase that ensures the platform remains accessible with minimal downtime and high availability. Below are the key deployment requirements that outline the necessary steps for a smooth deployment:

13.1 Environment Set-up

Production Server Specifications: The production server should run a stable Linux distribution (e.g., Ubuntu, CentOS), with at least 16 GB of RAM, a quad-core processor, and SSD storage of at least 256 GB for smooth performance, particularly for managing user data and real-time collaboration.

Staging Environment: A staging environment will be set up to mirror the production environment. This will allow pre-deployment testing and validation of new features, ensuring that the application runs reliably.

13.2 Deployment Process

Automation: By utilizing GitHub Actions & workflows as part of our Continuous Integration and Continuous Delivery pipeline with multi environment setup, we can quickly test our features in the staging environment before releasing the feature to the production environment.

Version Control: All code will be managed in a Git repository, with a clear branching model and

release management strategy. Rulesets are created to manage allowed actions by developers (e.g. master branch is locked, only Squash & Rebase is allowed for Pull Requests).

13.3 Data Migration & Backup

Data migration strategy: A detailed migration plan will be created to ensure a smooth transition of any existing data to the new system, with minimal disruption to users. This will include the necessary steps for data validation and transfer. For every release, the migration history will be squashed and baselined to ensure the migrations folder do not become too large.

Backup procedures: Automated daily backups will be scheduled to back up critical data and configuration files using Supabase's data backup services. "pg_dump" scripts will be created as part of our CI/CD pipeline ensuring that the latest data is saved before new releases are rolled out.

13.4 Dependency Management

Software Dependencies: All software dependencies will be clearly documented and locked in the respective package managers (e.g., npm for JavaScript, pip for Python). This ensures that builds are consistent across environments.

Service Dependencies: Configuration for any third-party services and integrations (external APIs) will be managed securely via environment variables and secret management systems.

13.5 Security Measures

Encryption: TLS encryption will be enforced for all communication between clients and servers. Sensitive data stored in databases will be protected with at-rest encryption.

Access Controls: Access to the production environment will be strictly controlled using SSH keys and VPNs. Only authorized personnel will have access to the system's sensitive areas.

Security Testing: Regular security audits and penetration testing will be carried out to identify vulnerabilities and secure the platform before deployment.

13.6 Compliance and Regulations

Legal Compliance: Wanderers will comply with GDPR (General Data Protection Regulation) to ensure proper handling of user data, and any relevant local laws related to digital platforms and travel data.

Data Protection: Policies for user data protection will be implemented, including user consent management, data anonymization, and adherence to the right to erasure.

13.7 Rollback Strategy

Contingency Planning: In case of a failed deployment, a rollback mechanism will be in place, allowing the team to revert to the last stable version of the application to minimize downtime and service interruptions.

13.8 Documentation and Training

Deployment Documentation: Comprehensive documentation will be created for the deployment process, including step-by-step guides, troubleshooting procedures, and emergency protocols. This documentation will be maintained and regularly updated.

Staff Training: Technical teams will receive training on the deployment process, tools, and contingency procedures to ensure smooth deployments and quick resolution of any issues that arise.

13.9 Continuous Deployment

CI/CD Pipeline: A robust CI/CD pipeline will be implemented to enable frequent, safe deployments of
the application as new updates and features are ready.

Appendix A – Use Case Descriptions

Register User

Use Case ID:	1		
Use Case Name:	Register User		
Created By:	Kim Seojin	Last Updated By:	Kim Seojin
Date Created:	01/02/2025	Date Last Updated:	01/02/2025

Actors:	User, Database, Server, SMTP Service
Description:	*As a User, I want to register an account and verify my email to access the system.* This use case enables user to register on the platform by providing a unique username and password, ensuring secure access to the system's features and services
Trigger:	The user clicks on the Register button to create an account.
Preconditions:	1. User has launched the website. 2. User does not have an account registered and verified with the system.
Postconditions:	1. User account 2. User will receive a confirmation email. 3. User must verify via email within 2 days of receiving activation link before account becomes active.
Normal Flow:	1.1: User navigates to the account registration page. 1.2: On the registration page, the user enters username, email and password to be registered. 1.3: User clicks submit. 1.4: Server validates the user input. 1.5: Server registers new account but set it as unverified. 1.6: Server generates unique verification token and sends the verification link to user email via SMTP Service. 1.7: User clicks on the verification link. 1.8: Server sets the user account as verified. 1.9: User is redirected to the login page.
Alternative Flows:	1.7 User does not click on the verification link within 2 days 1. System deletes the account.
Exceptions:	1.0.E.2 Invalid email format. 1. System prompts error message "Please enter a valid email" 1.0.E.2 Invalid password format. 1. 2. System prompts error message "Password must be at least 8 characters and have at least one non letter or digit character."
Includes:	-

Priority:	High
Frequency of Use:	High
Business Rules:	-
Special Requirements:	SR – 1: The system should be able to verify if the username is taken within 5 seconds.
Assumptions:	-
Notes and Issues:	-

Login User

Use Case ID:	2		
Use Case Name:	Login User		
Created By:	Kim Seojin	Last Updated By:	Kim Seojin
Date Created:	01/02/2025	Date Last Updated:	01/02/2025

Actors:	User, Database, Server
Description:	*As a User, I want to be able log in to the system so that I can use Wanderer's services.* This use case allows users to login to the platform with their registered userID and password.
Trigger:	1. The user wants to access their account and navigate to the login page. 2. The user clicks the "Login" button on the website.
Preconditions:	1. User has launched the website. 2. User must have an account registered and verified with the system.
Postconditions:	1. User has successfully logged in to the website. 2. System redirects the user to the homepage.
Normal Flow:	2.1: The user clicks on the Login button on the landing page. 2.2: System displays the login page. 2.3: User enters their username in the 'username' field 2.4: User enters their password in the 'password' field 2.5: User clicks on the 'Login' button 2.6: Server authenticates the username and password against the database 2.7: Upon successful authentication, user gain access to the application 2.8: System redirects user to homepage
Alternative Flows:	2.2: User click on "Forget Password" (Use Case 3) 1. System redirect user to password recovery page 2.5: User enters incorrect username or password 1. Displays error message "Invalid ID or Password. Please try again"
Exceptions:	-
Includes:	-
Priority:	High

Frequency of Use:	High
Business Rules:	1. Users must have verified accounts before logging in. 2. Passwords must be hashed and securely stored.
Special Requirements:	SR – 1: The system should be able to verify the username and password within 5 seconds.
Assumptions:	-
Notes and Issues:	-

Forget Password

Use Case ID:	3		
Use Case Name:	Forget Password		
Created By:	Raghav	Last Updated By:	Raghav
Date Created:	03/02/2025	Date Last Updated:	04/02/2025

Actors:	User, Server, SMTP Service
Description:	*As a User, I want to reset my password with the system if I forgot my existing password* This use case allows a user to reset their password with the system.
Trigger:	1. The user clicks on the "Forgot Password" link on the login page
Preconditions:	1. The User must have an existing account in the system (Use Case 1) 2. The user must have access to the email associated with the account
Postconditions:	1. “Password Reset Successful” message will be displayed 2. User’s password is updated, and they can log in using new credentials
Normal Flow:	3.1: User clicks on "Forgot Password" 3.2: “Forgot Password” page is displayed 3.3: User enters their registered email address 3.4: User clicks on “Get verification code” button. 3.5: Server creates a token and saves it in the database. 3.6: SMTP service sends an email together with a link with the token attached to the registered email address 3.7: User clicks password reset link. 3.8: User is redirected to Password Reset Form 3.9: User enters new password 3.10: User clicks on the “Reset Password” button 3.11: If password successfully reset, User will be redirected to the login page
Alternative Flows:	3.2: User clicks on “Cancel” button 1. User is redirected to the home page.
Exceptions:	3.0.E.4

	1. If the user enters an email that is not registered, the system displays an error message: "No account found with this email address." 3.0.E.9 1. The new password does not meet criteria. System displays message “Your password should at least contain one upper case letter, one lower case letter, one digit, and one special character”
Includes:	-
Priority:	Low
Frequency of Use:	Low
Business Rules:	-
Special Requirements:	Email containing password resetting link must be sent within 10 seconds
Assumptions:	-
Notes and Issues:	-

Email Confirmation

Use Case ID:	4		
Use Case Name:	Email Confirmation		
Created By:	Raghav	Last Updated By:	Raghav
Date Created:	03/02/2025	Date Last Updated:	04/02/2025

Actors:	User, Server, Database, SMTP Service
Description:	*As a User, I want to verify my email address with the system after I have registered an account* This use case allows a user to verify their registered email address to ensure their account can be used in Wanderers.
Trigger:	1. When user registers account 2. User tries to login, but the registered email is unverified.
Preconditions:	1. User must have successfully registered an account with a valid email address
Postconditions:	1. If successful, the user’s email is confirmed, and they can log in. Token will then be deleted from database. 2. If unsuccessful, the user must request a new verification email
Normal Flow:	4.1: User clicks “Verify email address” button 4.2: Server generates a unique email verification token 4.3: SMTP service sends email together with a link with the token attached to the registered email address 4.4: A message: "A confirmation email has been sent to your email address. Please check your inbox." will be displayed. 4.5: User clicks on email verification link

	4.6: Server verifies the token and redirects the user to the login page with a success message.
Alternative Flows:	-
Exceptions:	4.0.E.3 1. If the SMTP server fails to send the email, the server logs the error and retries. If retries fail, the system notifies the user. 4.0.E.6 1. If the reset link is expired (=> 2 days), System notifies the user and prompts them to repeat email verification process
Includes:	-
Priority:	Low
Frequency of Use:	Once
Business Rules:	-
Special Requirements:	Email containing verification link must be sent within 10 seconds
Assumptions:	-
Notes and Issues:	-

Edit Profile

Use Case ID:	5		
Use Case Name:	Edit Profile		
Created By:	Hazim	Last Updated By:	Hazim
Date Created:	01/02/2025	Date Last Updated:	04/02/2025

Actors:	User, Server, Database, S3 Object Storage
Description:	*As a User, I want to be able to edit or customize my profile that can be seen by others* This use case allows the user to edit their account information, which includes their profile picture and profile description.
Trigger:	The user clicks on the “Edit Profile” button
Preconditions:	1. User is logged in (Use Case 2)
Postconditions:	1. Account information is updated
Normal Flow:	5.1: User clicks ‘Edit Profile’ button 5.2: “Edit Profile’ page is loaded and showed to User 5.3: User clicks *Change Profile Picture* 5.4: User selects an image file 5.5: Server validates the file type and size and displays a preview of the new picture 5.6: The user edits their profile description in the provided text area. 5.7: User submits the form 5.8: Server validates all inputs 5.9: The server uploads the new profile picture to the S3 Object

	Storage 5.10: Server saves the updated description and image URL into the database 5.11: System confirms the successful update
Alternative Flows:	•
Exceptions:	5.0.E.4: User selects a non-image file 1. An error message “Only image files (.png, .jpg, .jpeg) are accepted.” will be displayed 2. User is prompted to input another file. 5.0.E.4: Uploaded image file is more than 5MB 1. The system will display an error message “Files cannot exceed more than 5MB.” 2. User is prompted to input another file.
Includes:	-
Priority:	Medium
Frequency of Use:	Low
Business Rules:	-
Special Requirements:	-
Assumptions:	This use case assumes that the user is updating their profile information and changed one of the fields
Notes and Issues:	-

Create Itinerary

Use Case ID:	6		
Use Case Name:	Create Itinerary		
Created By:	Hazim	Last Updated By:	Hazim
Date Created:	01/02/2025	Date Last Updated:	04/02/2025

Actors:	User, Server, Database, SMTP Service
Description:	*As a User, I want to be able to create an itinerary to start my planning of my trip* This use case allows the user to create a new itinerary on the platform to start planning their trips.
Trigger:	The user clicks on the “Create Itinerary” button
Preconditions:	User is logged in (Use Case 2)
Postconditions:	A new itinerary is created
Normal Flow:	6.1: ‘Create Itinerary’ page is loaded and launched. 6.2: The user interface displays a form with required input fields (Trip Name, Trip Location, Trip Start Date, Trip End Date and Itinerary Visibility) and an optional input field (Invite tripmates) 6.3: User inputs the required fields 6.4: User enters email addresses of friends to send invites 6.5: User submits the form 6.6: Server validates all required fields

	6.7: Server creates an itinerary in the database 6.8: The system displays a successful confirmation message 6.9: The SMTP Service sends emails to the inputted emails with a link to view the itinerary page 6.10: The system redirects the user to the itinerary details page
Alternative Flows:	6.4: The user did not invite any tripmates 1. When the user submits the form, the database creates the itinerary and the SMTP Service sends no emails
Exceptions:	6.0.E.5: Field Validation Error 1. If the user submits the form with invalid or missing data 2. System displays error message “Please fill in required fields” 3. Return to step 6.2
Includes:	-
Priority:	High
Frequency of Use:	Low (once per trip)
Business Rules:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Access Itinerary

Use Case ID:	7		
Use Case Name:	Access Itinerary		
Created By:	J’sen	Last Updated By:	J’sen
Date Created:	3/2/2025	Date Last Updated:	3/2/2025

Actors:	User, Database
Description:	*As a User, I want to access the itinerary so that I can get detailed trip information. This allows me to stay informed, manage my travel plans efficiently and collaborate with other trip participants.* A user can access an existing itinerary to view details, schedules and shared plans.
Trigger:	The user selects an itinerary from their list of trips they have created or through the explore page.
Preconditions:	1. The user must be logged in. 2. If the user is accessing their own itinerary, they must have at least one itinerary created. 3. If the user is accessing another person’s itinerary from the explore page: a. The itinerary must be public or shared with the user.

Postconditions:	1. The user successfully views the itinerary details, including trips dates, locations, activities, shared notes and the total trip cost.
Normal Flow:	7.1: User clicks on “View Itinerary” button 7.2: The system retrieves the itinerary details from the database. 7.3: The ‘View Itinerary’ page is loaded and shown to User 7.4: Details such as schedules, locations, activities, shared notes and the total trip cost are shown to user.
Alternative Flows:	7.1: The user accesses the itinerary from a shared link 1. The user clicks on a shared itinerary link 2. If logged in, they are redirected to the itinerary view. 3. If not logged in, they are prompted to log in before viewing the itinerary (if required).
Exceptions:	7.0.E.1: The itinerary does not exist or has been deleted 1. The system displays an error message indicating the itinerary is unavailable.
Includes:	-
Priority:	High
Frequency of Use:	Frequent. Users will typically access their itinerary multiple times before and during their trip.
Business Rules:	-
Special Requirements:	-
Assumptions:	Users have at least one itinerary saved in their account or can access itineraries via the Explore page.
Notes and Issues:	-

Edit Itinerary

Use Case ID:	8		
Use Case Name:	Edit Itinerary		
Created By:	J’sen	Last Updated By:	J’sen
Date Created:	3/2/2025	Date Last Updated:	3/2/2025

Actors:	User, Database
Description:	*As a User, I want to access and edit my itinerary so that i can update trips details, modify schedules and ensure all travel information is accurate and up to date.* A user can edit an existing itinerary including update trip details, add or remove activities, adjust schedules and manage trip expenses to calculate total trip cost.
Trigger:	The user selects an itinerary from “My Trips” page.
Preconditions:	1. The user must be the owner of the itinerary. 2. User has been granted editing permissions by the owner. 3. The itinerary must exist and be accessible.

Postconditions:	1. The system updates the itinerary with the user's changes. 2. The updated itinerary is available for viewing by the owner and other collaborators.
Normal Flow:	8.1: User navigates to "My Trip" page 8.2: User selects an itinerary 8.3: The system retrieves the itinerary details from database 8.4: The user makes changes, such as: • Modify trip details (title, description, dates). • Adding or removing activities. • Adjusting the trip schedules. • Updating shared notes. • Adding, modifying or removing expenses. 8.5: If the user modifies expenses, the system automatically recalculates the total trip cost. 8.6: The system saves the changes and updates the itinerary.
Alternative Flows:	-
Exceptions:	8.0.E.1: The itinerary has been deleted 1. The system informs the user that the itinerary no longer exists.
Includes:	-
Priority:	High
Frequency of Use:	Frequent. Users may update itineraries and expenses multiple times during trip planning.
Business Rules:	-
Special Requirements:	-
Assumptions:	Users have at least one editable itinerary in their account. Users make changes to the itinerary.
Notes and Issues:	-

Delete Itinerary

Use Case ID:	9		
Use Case Name:	Delete Itinerary		
Created By:	Albert	Last Updated By:	Albert
Date Created:	2/2/2025	Date Last Updated:	2/2/2025

Actors:	User, Database
Description:	*As a User, I want to delete an itinerary so that the System will not show any outdated or irrelevant itinerary.* A feature to remove the itinerary details and its contents including activities associated with the itinerary itself. The itinerary is archived instead of permanently deleted, allowing the user to restore it if needed.
Trigger:	The user clicks on the Delete button on his itinerary page.

Preconditions:	1. The user is already logged in. (Use Case 2) 2. The user must own the itinerary.
Postconditions:	1. The itinerary and its children are marked as archived. 2. The user will be redirected to the home page. 3. The user is shown a success notification.
Normal Flow:	9.1: User identifies his own Itinerary to be deleted and clicks the delete button. 9.2: A dialog popup appears to ask User to confirm his action. 9.3: User confirms to delete. 9.4: System marks the itinerary as archived. 9.5: UI is updated with a message notifying the user about the successful deletion. 9.6: System displays successful message “Itinerary deleted. You can still undo changes by clicking here” 9.7: System redirects User to home page.
Alternative Flows:	9.2.1: User cancels delete action 1. Closes dialog popup and returns to the itinerary page
Exceptions:	9.0.E.3: User does not own the itinerary 1. System displays an error message “You do not own the itinerary” 2. Itinerary is not deleted 9.0.E.3: System generic error 1. System displays an error message “An error has occurred, please try again later” 2. Error is logged 3. Itinerary is not deleted
Includes:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	1. Itinerary is archive instead of deleted with the isDeleted flag so the user can restore any accidental deletions. 2. Only the owner can delete (archive) their itinerary.
Special Requirements:	1. System must show the successful deletion message with the undo action and display it for at least 5 seconds to allow the user to restore the itinerary.
Assumptions:	-
Notes and Issues:	1. Determine data retention policy (how long to store the archive itinerary before permanently deleting it)

Create Activity in an Itinerary

Use Case ID:	10
Use Case Name:	Create Activity in an Itinerary

Created By:	Albert	Last Updated By:	Albert
Date Created:	3/2/2025	Date Last Updated:	3/2/2025

Actors:	User, Database, PlacesAPI
Description:	*As a User, I want to create activity entry(s) so that I can share more details such as the location and cost that is associated to the itinerary.* The user can create multiple activities within an itinerary. Each activity includes details such as title, description, cost, and location.
Trigger:	User clicks on the Add Activity button.
Preconditions:	1. The user must be logged in 2. The user must own the itinerary 3. The itinerary exists and is not archived
Postconditions:	1. New activity entry(s) will be created under the itinerary. 2. The activity is displayed in the itinerary's activity list without refreshing the page 3. The user can continue adding more activities or close the form.
Normal Flow:	10.1: User identifies his own Itinerary and clicks the add activity button. 10.2: A modal form popup appears for user input 10.3: User inputs the activity title, description, cost. 10.4: User search for location using PlacesAPI 10.5: System returns a list of autocompleted results for user to select. 10.6: User clicks on submit button. 1. System validates the input fields. 2. System creates an activity in database and the UI is updated with a new activity entry.
Alternative Flows:	10.6: User has more activities to add: 1. User clicks on Add More activities button 2. System displays an additional form to ask User to input the activity title, description, cost and location.
Exceptions:	10.0.E.6: Field Validation Error 1. If the user submits the form with invalid or missing data 2. System displays error message "Please fill in required fields"
Includes:	-
Priority:	Medium
Frequency of Use:	High
Business Rules:	1. Activity title, description, cost and location are required. 2. Only members with permissions can add activities
Special Requirements:	1. After the activity is created, the activity should be shown on the itinerary without refreshing the page. 2. Form inputs must be validated before submission and validated again before server processes the data.
Assumptions:	1. User has an activity to add to the itinerary.
Notes and Issues:	

Access Activity in an Itinerary

Use Case ID:	11		
Use Case Name:	Access Activity in an Itinerary		
Created By:	Isaac	Last Updated By:	Isaac
Date Created:	3/2/2025	Date Last Updated:	3/2/2025

Actors:	User, Database
Description:	*As a User, I want to be able to access a particular activity in the itinerary* The user can access a particular event planned in an itinerary to view more details about that activity such as name, location. This use case is a child use case of **Edit Itinerary.**
Trigger:	1. The user clicks on the activity button/bounding box
Preconditions:	1. The user is logged in. 2. The user has access to the itinerary. 3. The itinerary must have at least one activity to view.
Postconditions:	1. The user interface shows details about the activity.
Normal Flow:	11.1: User clicks on the activity button/bounding box 11.2: The system fetches latest activity information from database 11.3: The user interface shows information about the activity. 11.4: User clicks on ‘Close’ button 11.5: User is redirected back to the itinerary details page.
Alternative Flows:	11.4: The user decides to edit part of an activity 1. The user carries on the flow from Use Case 12 (Edit Activity in an Itinerary)
Exceptions:	11.0.E.2: The system is unable to fetch results from the database 1. A dialog showing “Unable to fetch activity details, please try again later” is shown.
Includes:	-
Priority:	High
Frequency of Use:	High
Business Rules:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Edit Activity in an itinerary

Use Case ID:	12		
Use Case Name:	Edit Activity in an Itinerary		
Created By:	Isaac	Last Updated By:	Isaac
Date Created:	3/2/2025	Date Last Updated:	3/2/2025

Actors:	User, Database
Description:	*As a User, I want to be able to make changes to an activity that I have planned in an itinerary.* The user can access and edit the activity and change details about the name, timing, location, pricing and other relevant details in the activity.
Trigger:	1. The user clicks on the 'Edit Activity' button from the 'View Activity view'
Preconditions:	1. The user is logged in. 2. The user has access to the itinerary. 3. The itinerary must have at least one activity to view. 4. User is currently viewing the activity.
Postconditions:	1. The user interface shows the updated activity 2. The information about the updated activity is updated in the database.
Normal Flow:	12.1: User clicks on 'Edit Activity' button 12.2: The user interface shows all editable fields of the activity 12.3: User makes changes to activity details such as: • Name • Location • Timing/Date • Description/Details 12.4: User clicks on 'Save Activity' button 12.5: System validates the user input 12.6: System saves the updated activity in the database. 12.7: The user interface shows a "The activity has been saved!" message. 12.8: User is redirected back to "View Itinerary" page
Alternative Flows:	
Exceptions:	12.0.E.5: Field Validation Error 1. If the user submits the form with invalid or missing data 2. System displays error message "Please fill in required fields" 12.0.E.6: Unable to access Database

	1. System displays error message “Unable to save this activity. Please try again later.
Includes:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Delete Activity in an Itinerary

Use Case ID:	13		
Use Case Name:	Delete Activity in an Itinerary		
Created By:	Miko	Last Updated By:	Miko
Date Created:	28/1/2025	Date Last Updated:	4/2/2025

Actors:	User, Database
Description:	*As a User, I want to ensure a lean itinerary to organize only relevant activities, removing those irrelevant ones.* A feature that deletes an activity/event within an itinerary
Trigger:	1. User must be viewing an activity. 2. User clicks the ‘Delete Activity’ button.
Preconditions:	1. There should be an itinerary that existed. (Use Case 7) 2. There should be at least 1 activity/event in the itinerary for the user to delete if not, deletion is not possible (Use Case 11) 3. A warning message that tells user that the deletion of an activity/event is permanent and cannot be recovered
Postconditions:	1. The deleted entry will disappear from the database 2. The deleted entry also disappears from the UI 3. A message will be prompted to the user about successful deletion
Normal Flow:	13.1: User clicks the ‘Delete Activity’ button 13.2: The user interface shows a window alert with message “Are you sure you want to delete this activity?” 13.3: User clicks on ‘Confirm’ button. 13.4: The system deletes the activity entry in the database. 13.5: System displays "Your activity has been deleted! Message. 13.6: User is redirected to the itinerary view page.
Alternative Flows:	13.2: User cancels the delete option 1. System closes the alert 2. User is redirected to the view activity page
Exceptions:	13.0.E.4: Database unable to delete the activity

	1. System displays the unsuccessful deletion message to the User.
Includes:	-
Priority:	Medium
Frequency of Use:	High during heavy edits of the itinerary Low when the itinerary is completed/fixed
Business Rules:	-
Special Requirements:	A prompt must be created to remind users of the consequences of deletion. • If there is a single delete button that controls all the deletion function, the button would need to be disabled and prompt an error message as there is no entry to be deleted • If there is a delete button created along with the activity, there will be no such special feature to be added as that delete button is part of the entry that disappears from the UI
Assumptions:	• There should be at least 1 existing activity/event to fulfil the deletion
Notes and Issues:	-

Logout

Use Case ID:	14		
Use Case Name:	Logout		
Created By:	Miko	Last Updated By:	Miko
Date Created:	28/2/2025	Date Last Updated:	4/2/2025

Actors:	User, Server
Description:	*As a User, I want to logout so that my session will not persist.* A feature that helps User to logout of the website so that they can end their current session.
Trigger:	The User clicks on the 'Logout' button on the home page
Preconditions:	1. User has an existing account (Use Case 1) 2. User has logged into the website (Use Case 2) 3. User is on the home page.
Postconditions:	1. User will be prompted to login page with a message informing the user their successful logout
Normal Flow:	14.1: User clicks on the 'Logout' button from home page. 14.2: The user interface displays a "Are you sure you want to log out?" message 14.3: User clicks on the "Log Out" button. 14.4: Server deactivates the user's session. 14.5: User is redirected to the login page.
Alternative Flows:	-
Exceptions:	14.0.E.4: System cannot log User out

	1. System displays the unsuccessful logout message to the User.
Includes:	-
Priority:	High
Frequency of Use:	Medium
Business Rules:	-
Special Requirements:	-
Assumptions:	• The User should know that clicking on the logout button will log them out without a prompt to warn them.
Notes and Issues:	-

Send Messages

Use Case ID:	15		
Use Case Name:	Send Messages		
Created By:	Miko	Last Updated By:	Miko
Date Created:	28/1/2025	Date Last Updated:	4/2/2025

Actors:	User, S3 Object Storage
Description:	*As a User, I want to be able to communicate tips and information with other users by sending them messages* A feature that allows a User to send messages to another in an itinerary, allowing for collaboration among users
Trigger:	User types a message in the typing bar and hits enter.
Preconditions:	1. User has an existing account (Use Case 1) 2. User has logged into the website (Use Case 2) 3. User is on the view itinerary page.
Postconditions:	1. The message is saved in the S3 Object Storage. 2. The message will display in the chat.
Normal Flow:	15.1: User clicks on the typing bar 15.2: User types a message and then hits send. 15.3: Message is sent and stored in S3 Object Storage 15.4: Message is displayed in the chat box.
Alternative Flows:	-
Exceptions:	15.0.E.2: User cannot send a message 1. The User will be notified that the message cannot be sent. 15.0.E.3: Unable to store message in S3 Object Storage 2. The User will be notified that the message cannot be sent.
Includes:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	-

Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Register User

Use Case ID:	16		
Use Case Name:	Upload Photo		
Created By:	Albert	Last Updated By:	Albert
Date Created:	3/2/2025	Date Last Updated:	3/2/2025

Actors:	User, Database, S3 Object Storage
Description:	*As a User, I want to upload photos to the system so that users will be engaged visually.* A feature to allow User to upload photos to the System to add interactivity with an itinerary or chat.
Trigger:	User clicks on Upload Photo or the round avatar image.
Preconditions:	1. The User has an existing account (Use Case 1) 2. The User has logged into the website (Use Case 2)
Postconditions:	1. The photo URL is saved in the Database 2. The photo is saved in the S3 Object storage bucket.
Normal Flow:	16.1: User clicks on Upload Photo button 16.2: The system displays a file explorer 16.3: User selects an image file 16.4: System validates the file type and size 16.5: The photo is saved in the S3 Object storage bucket. 16.6: The photo URL is saved in the database. 16.7: The user interface shows a “Your picture was uploaded successfully!” message.
Alternative Flows:	-
Exceptions:	16.0.E.4: User selects a non-image file 3. An error message “Only image files (.png, .jpg, .jpeg) are accepted.” will be displayed 4. User is prompted to input another file. 16.0.E.4: Uploaded image file is more than 5MB 3. The system will display an error message “Files cannot exceed more than 5MB.” 4. User is prompted to input another file.
Includes:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	1. File upload should only accept image/jpeg, image/png mime types. 2. File size should not exceed 5MB.
Special Requirements:	

Assumptions:	1. User has a photo to upload
Notes and Issues:	-