Software Requirements Specification

for

2W2Meet

Version 1.0 approved

Prepared by Song Yick Qi

Nanyang Technological University

5 April 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Song Yick Qi	5/4/2025	Compiled documentation from 2W2Meet lifecycle. Credit to the respective authors.	1.0

1. Introduction

Meeting planning can be frustrating and time-consuming, especially when coordinating schedules, locations, and real-world factors like weather. Our product, 2W2Meet, aims to simplify this process by creating a user-friendly scheduling tool that integrates public APIs to provide weather forecasts and location-based insights. Designed for students, professionals, and social groups, this tool will make planning meetings more efficient.

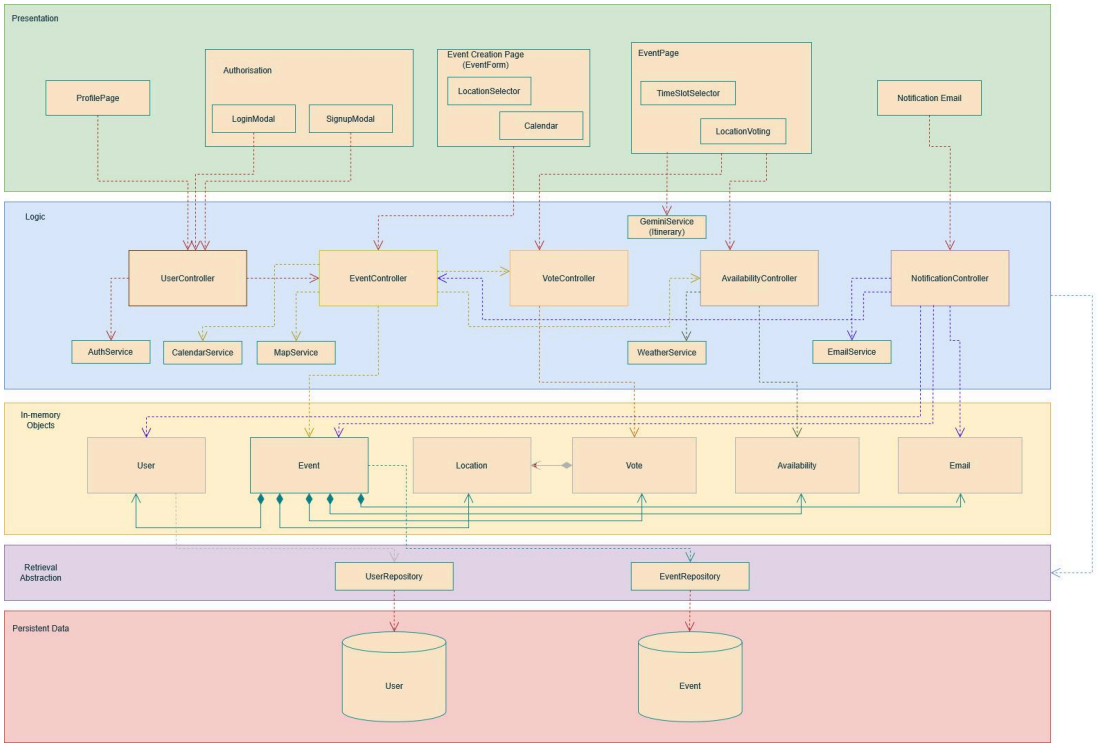
1.1 Document Conventions

Where appropriate, the terms defined in the Data Dictionary (see Appendix A) are capitalised. Boldface terms or phrases as will be seen in the Functional Requirements (see Section 4) are the keyphrases (or titles) for each of the major subsections.

2. Overall Description

2.1 Product Perspective

A high-level overview of the system architecture is provided below:-



Presentation

This layer is primarily responsible for the user interface and interaction components of the application, which users will interact with directly.

- **LandingPage**: The initial view where users can learn about the application and navigate to other functions.
- **EventCreationPage**: A dedicated interface for users to create new events, specifying details like time, date, and venue.
- **EventPage**: Displays detailed information for a specific event, including options to manage or modify event details.
- **ProfilePage**: A user interface that shows the users all the events they have interacted with, and provides convenient navigation to those events.
- **Notification Email**: Dispatched by the application backend, it informs the user about the details of finalised events. In a stricter sense, it is not an application component, but it provides some form of user interaction, e.g., google map link of the meetup location.

Logic

This layer contains all the controller classes that mediate between the presentation layer and the object layer, processing user input and executing application logic.

- **UserController**: Manages user login, registration, and session-persistent credentials.
- **VoteController**: Handles voting-related actions, like submitting votes for preferred event times or dates.
- **EventController**: Manages event-related functions, such as creating, modifying, and deleting event data.
- **AvailabilityController**: Checks and manages the availability of participants or resources for events.
- **NotificationController**: Manages sending notifications to users about event updates, reminders, and changes.

In-Memory Objects

This layer consists of the entities used by the app logic to implement its functionality. These entities are retrieved from a database.

- **User**: Represents a user with properties such as name, email, and preferences.

- **Event**: Represents an event with properties like date, time, and venue.
- **Location**: Manage geographical details and locations associated with events.
- **Vote**: Handle voting for event times and manage specific time slots for events.
- **Availability**: Manages available times for events.
- **Email**: Stores email address for event notification purposes.

Retrieval Abstraction

Wraps provider-dependent calls and procedures related to database operations into provider-independent API to encourage flexibility and declarativity in controllers. The API is responsible for the retrieval and update of persistent data whenever necessary in the logic layer.

Persistent Data

Stores all persistent data including user profiles, event details, locations, and other necessary data to ensure data consistency and durability.

2.2 Product Functions

The list of major product functions is provided below:-

UC1 - User management
- UC1.1 - Sign up
- UC1.2 - Login
- UC1.3 - View Profile - moderate

UC2 - Event Creation Page
- UC2.1 - Submit Event Creation Form - high

UC3 - Unique Event Page
- UC3.1 - View Unique Event Page - high
    - UC3.1.1 - Retrieve Event Information - high
    - UC3.1.2 - Request Venue Data - moderate
    - UC3.1.3 - Request Meetup Suggestions - moderate

UC4 - Respond to event
- UC4.1 - Login - high
- UC4.2 - Respond to Event - high
    - UC4.2.1 - Indicate availability - high
    - UC4.2.2 - Vote for Meetup Location - high

    - UC4.2.3 - Copy and Share Unique URL - low
    - UC4.2.4 - Opt in for notification - low

UC5 - Access Landing Page

UC6 - Finalise event

- UC6.1 - Collate Event Details - high
- UC6.2 - Dispatch Event Details - moderate

2.3 Operating Environment

2W2Meet has been locally tested and verified on both Windows 11 and MacOS systems. This web application only requires a modern browser to run; the inherent backward compatibility and interoperability of complex browser systems affords 2W2Meet with excellent portability and independence from the underlying Operating Systems.

2.4 Design and Implementation Constraints

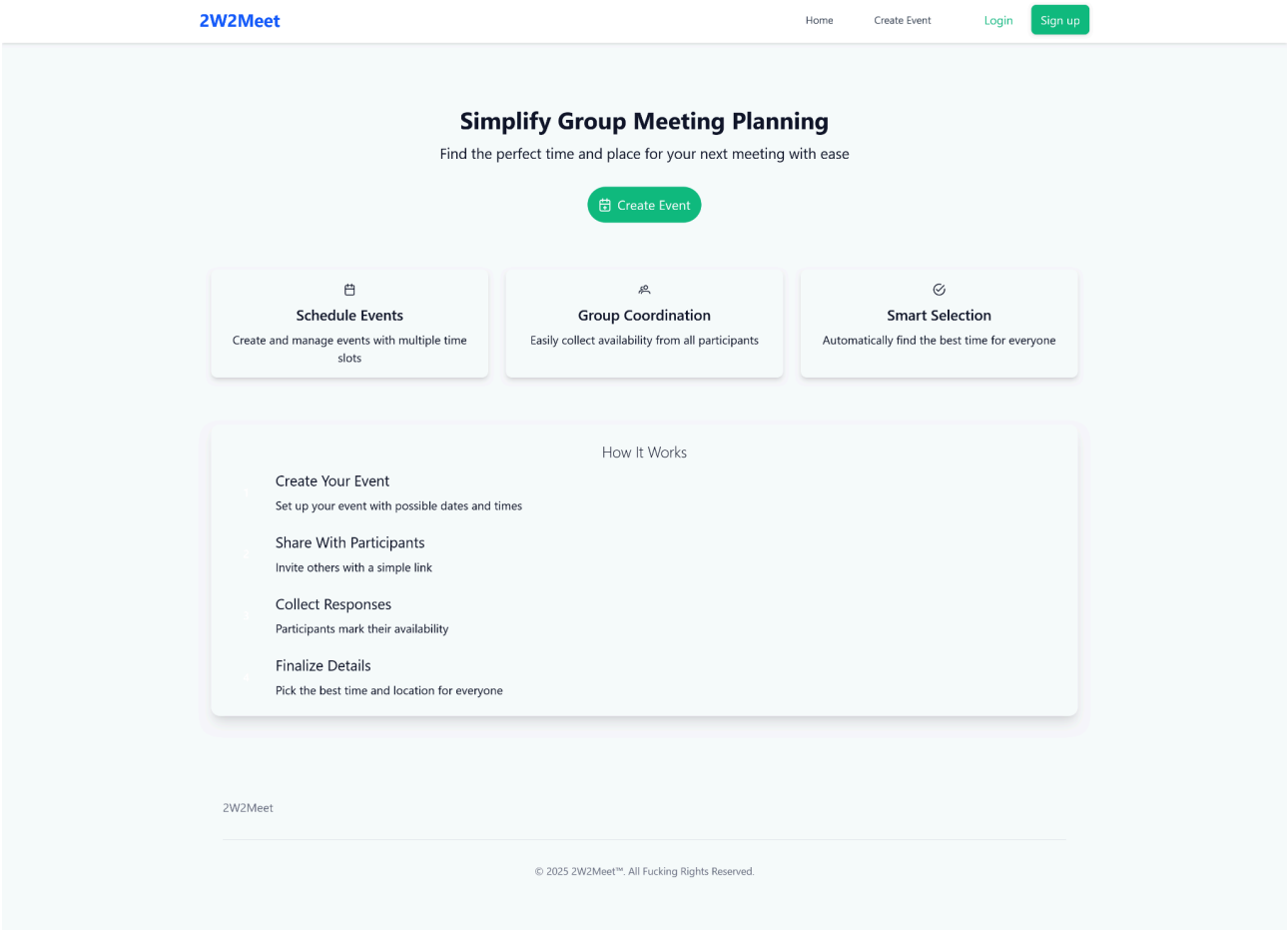
API Usage Limitations

The institution policy dictates that 2W2Meet to rely on external APIs for major functionalities. While all of the APIs used provide a generous extent of free services, 2W2Meet lacks the budget and therefore scalability to stand in-line with competitors with corporate-scale resources and standards.

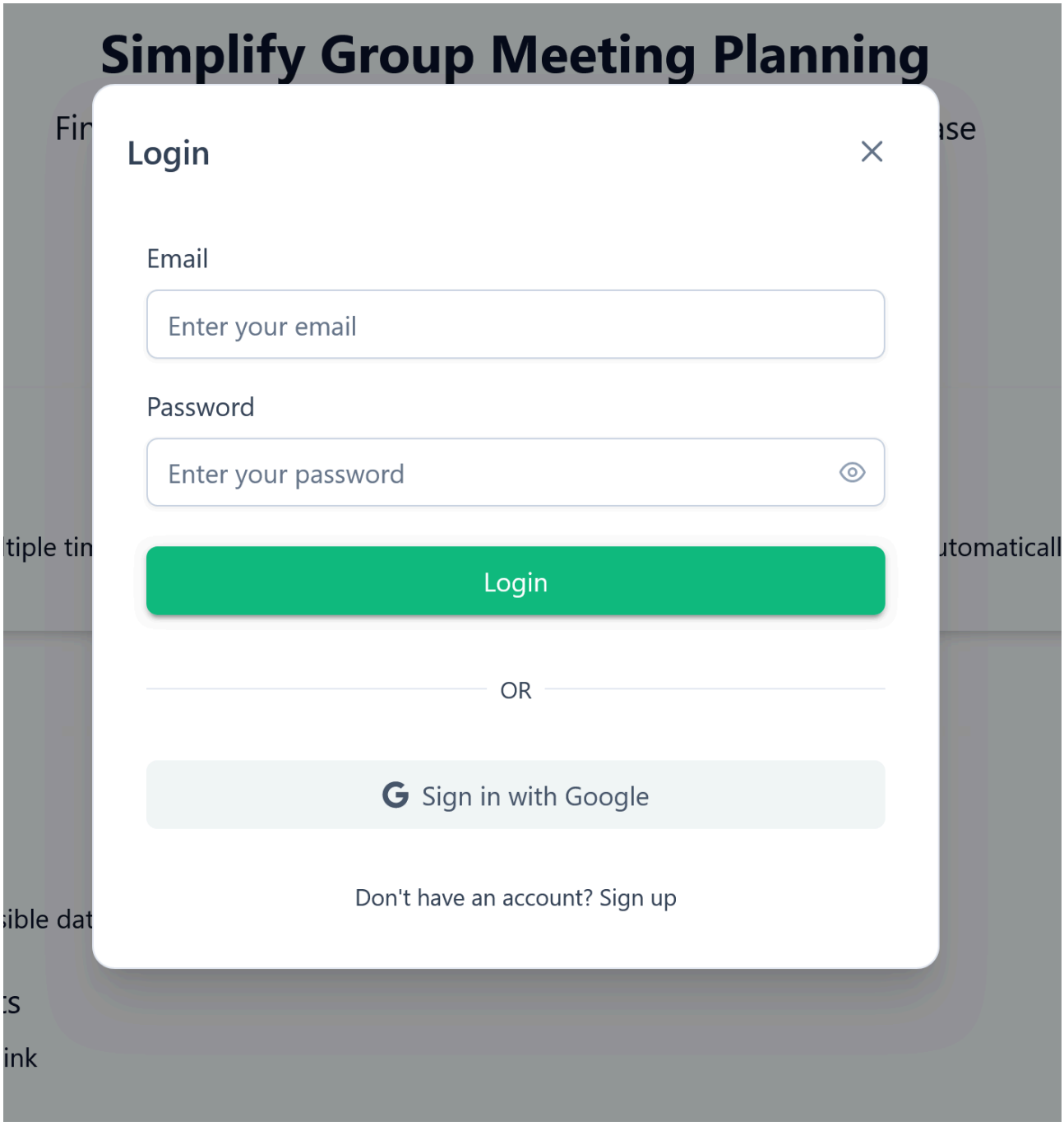
2.5 User Documentation

Please refer to the video link attached below for a comprehensive walkthrough of 2W2Meet.
<*video-demo*>

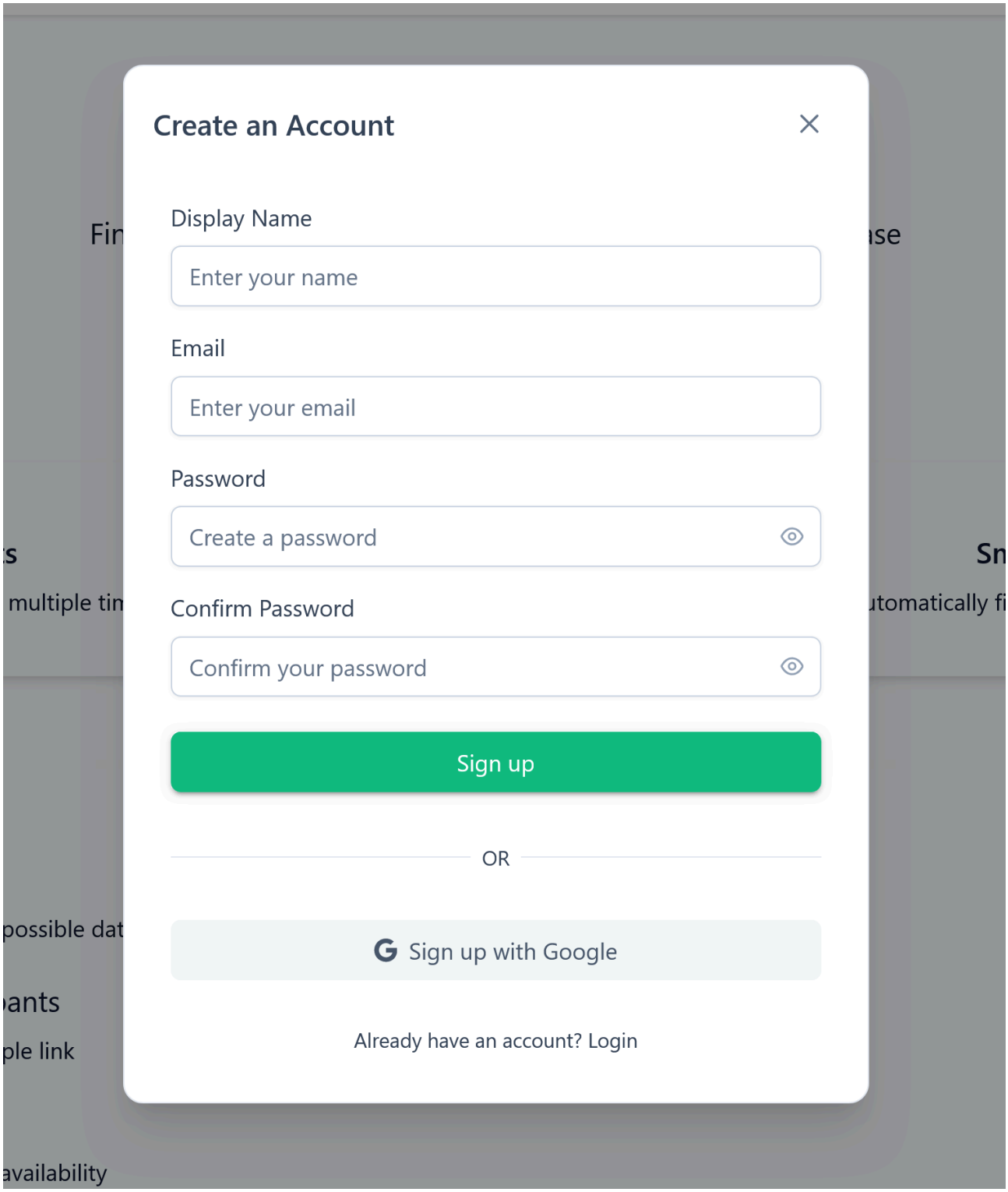
3. User Interface



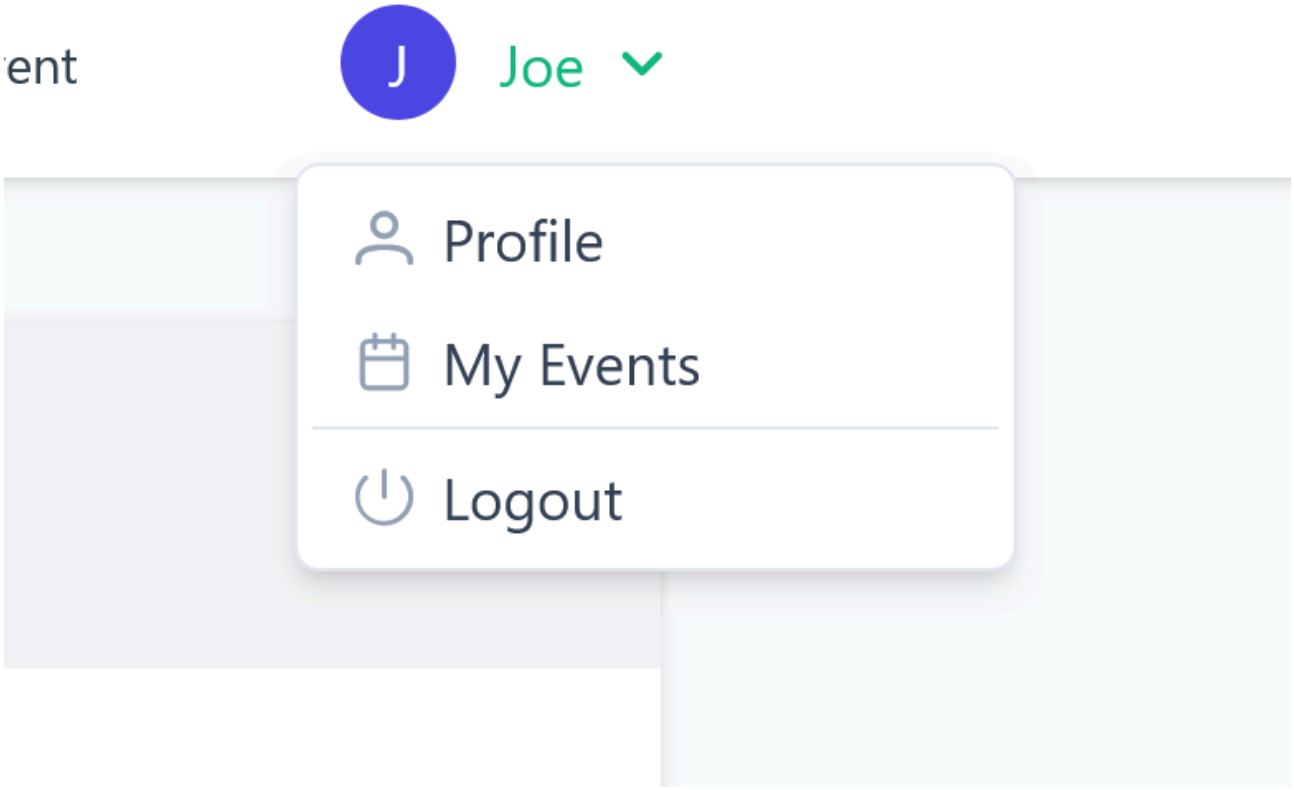
Landing Page. Provides navigation to the essential features of 2W2Meet, such as event creation, login and sign up.



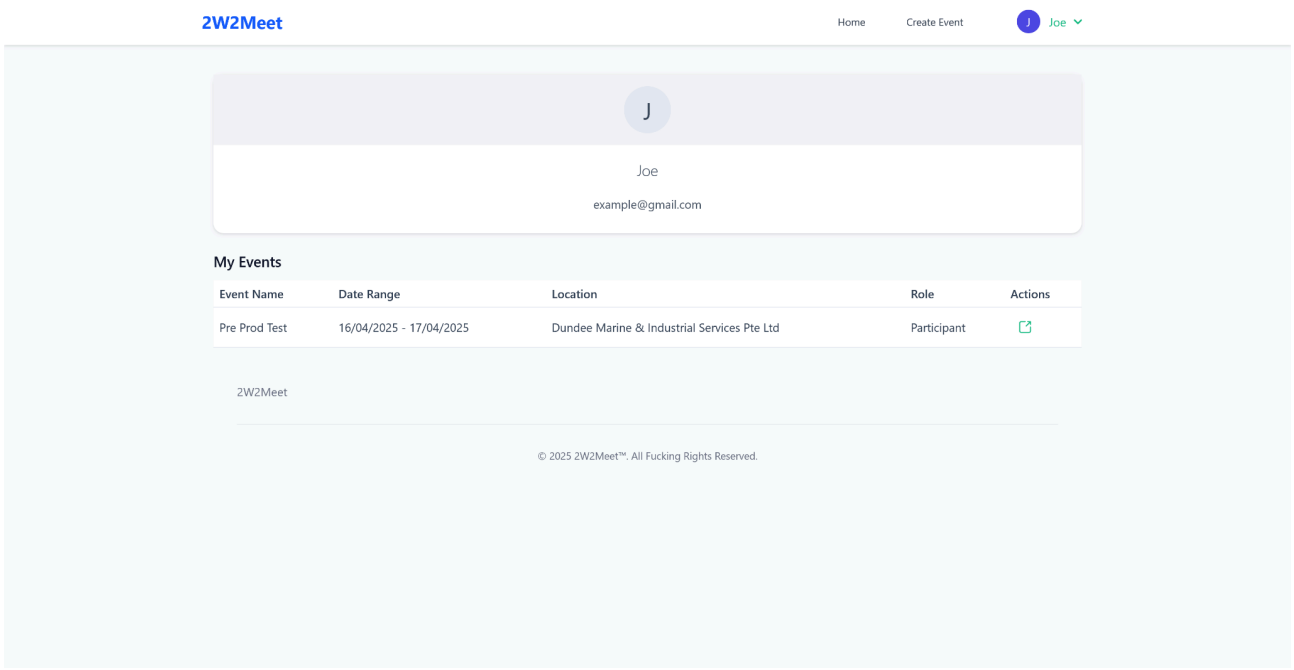
Login popup. For users that already have an account with 2W2Meet.



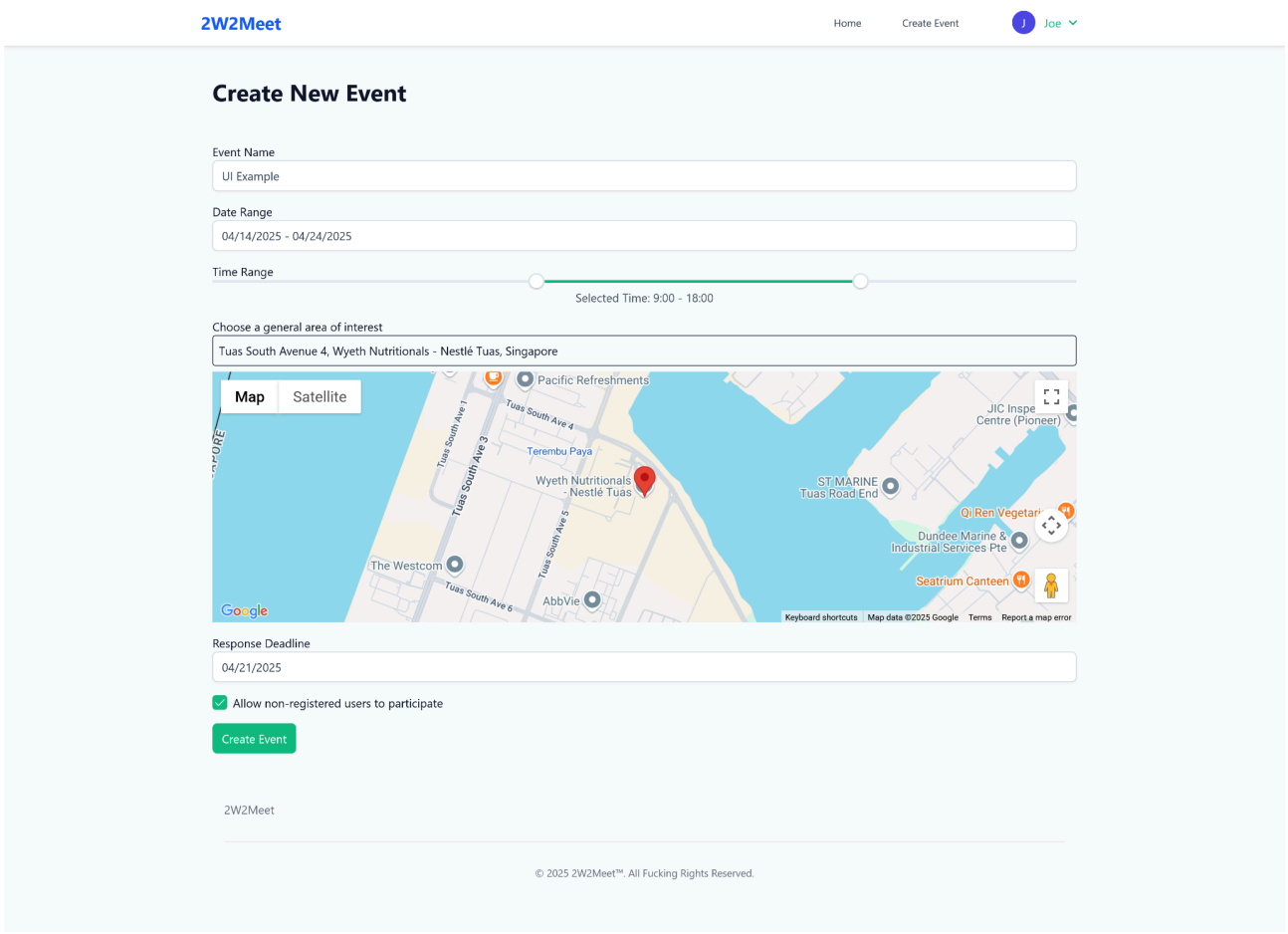
Signup popup. For users that are about to join 2W2Meet.



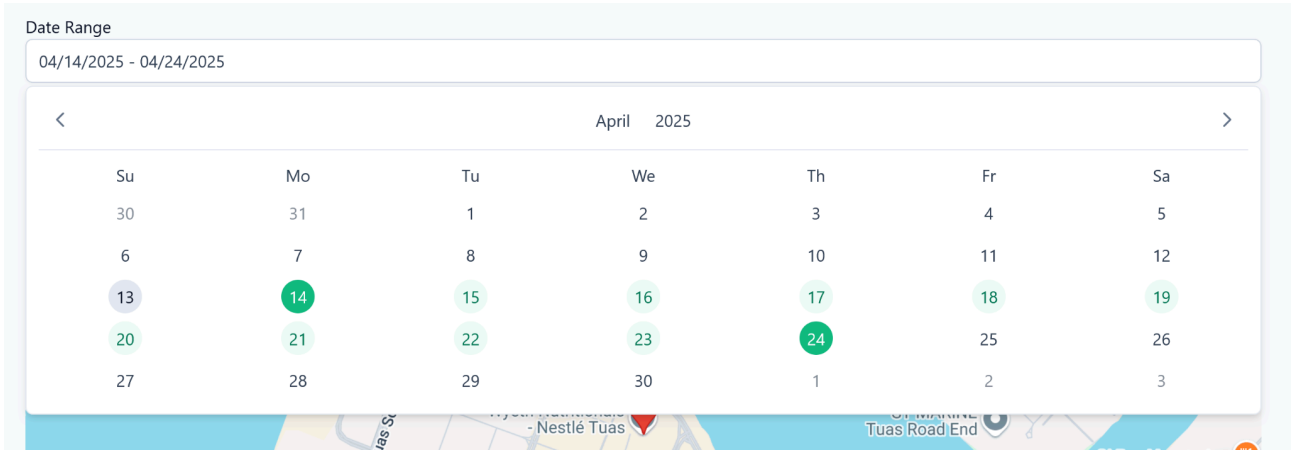
User menu. After logging in, the top bar displays the user's profile and actions.



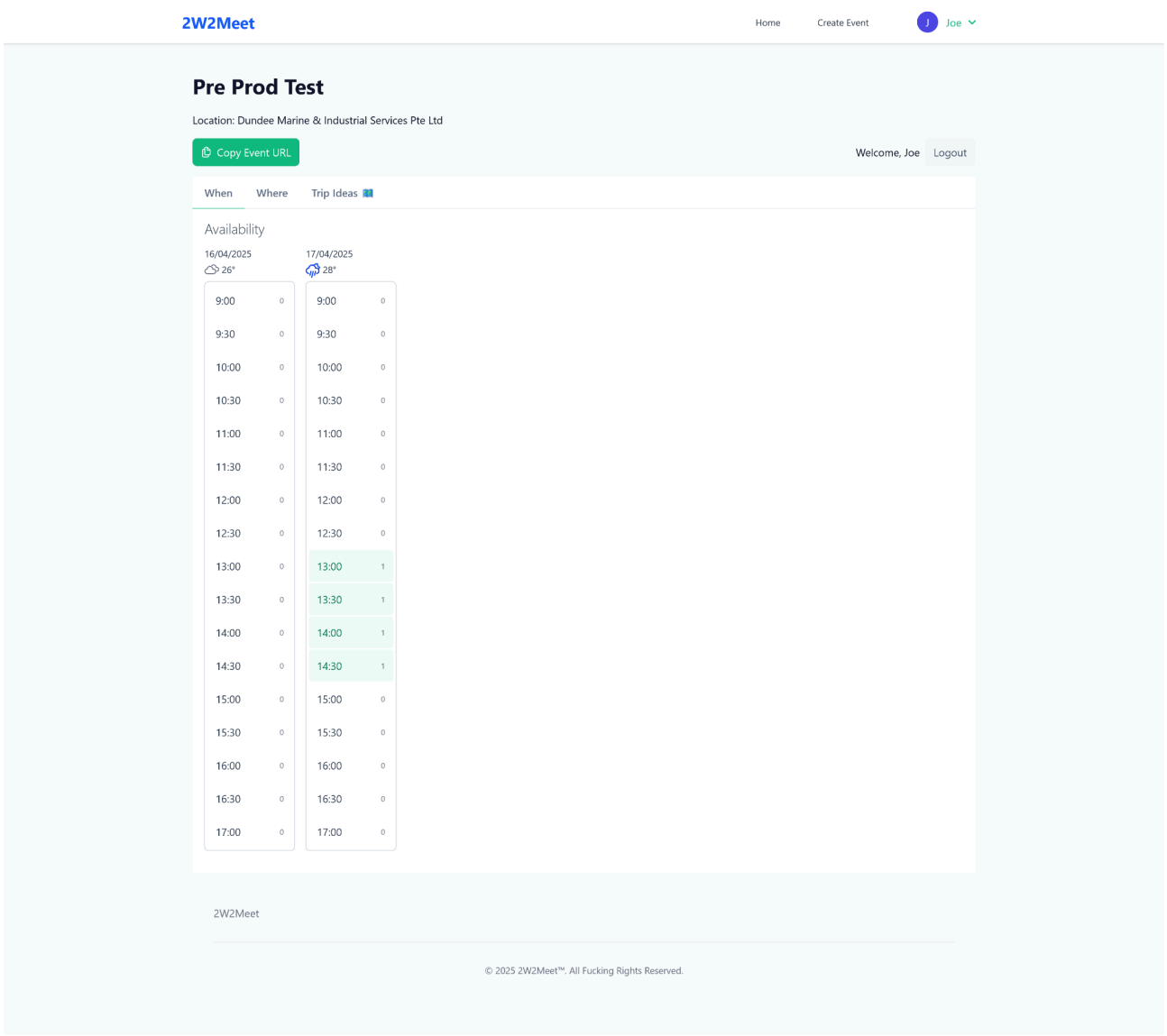
Profile page. It contains a list of the user's events. Each entry contains a link to the corresponding event.



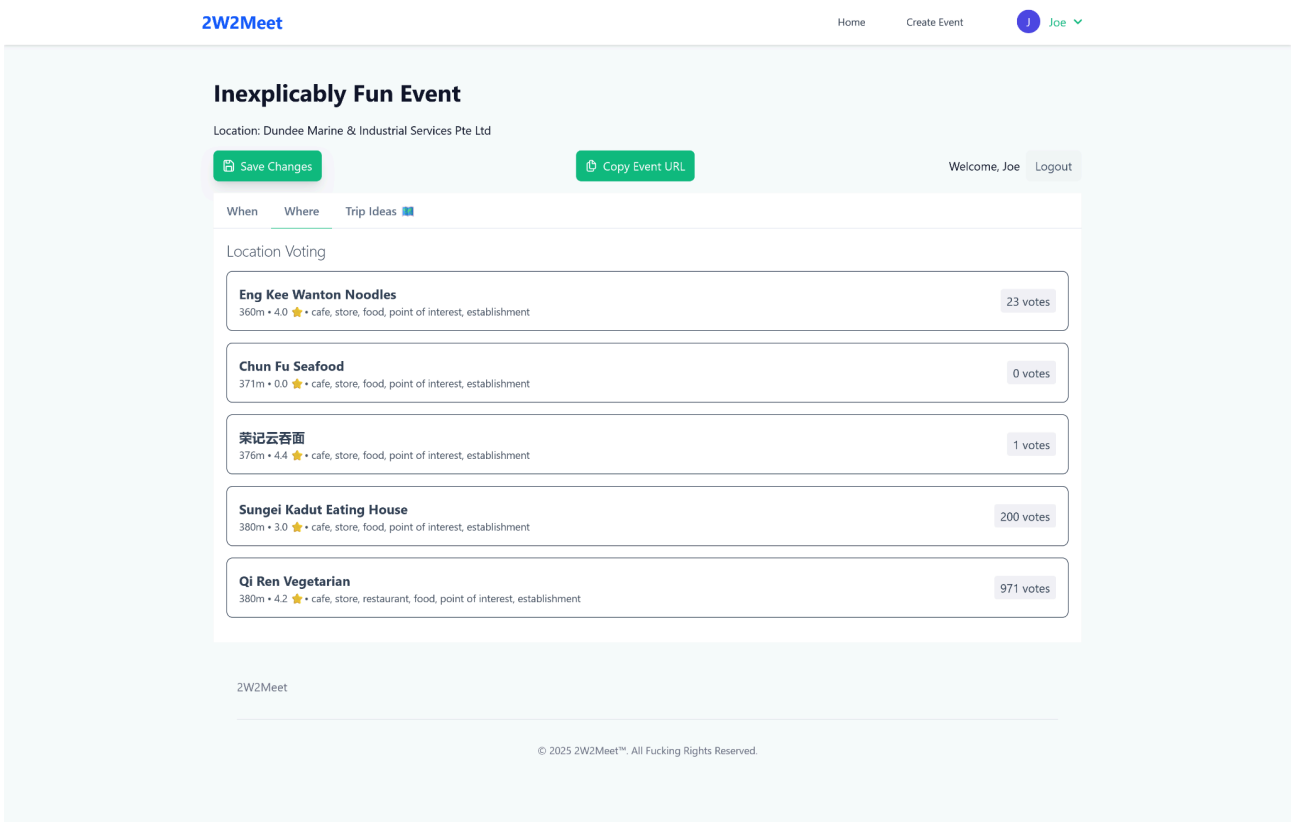
Event form. Event organisers fill these up to collect participants' decisions.



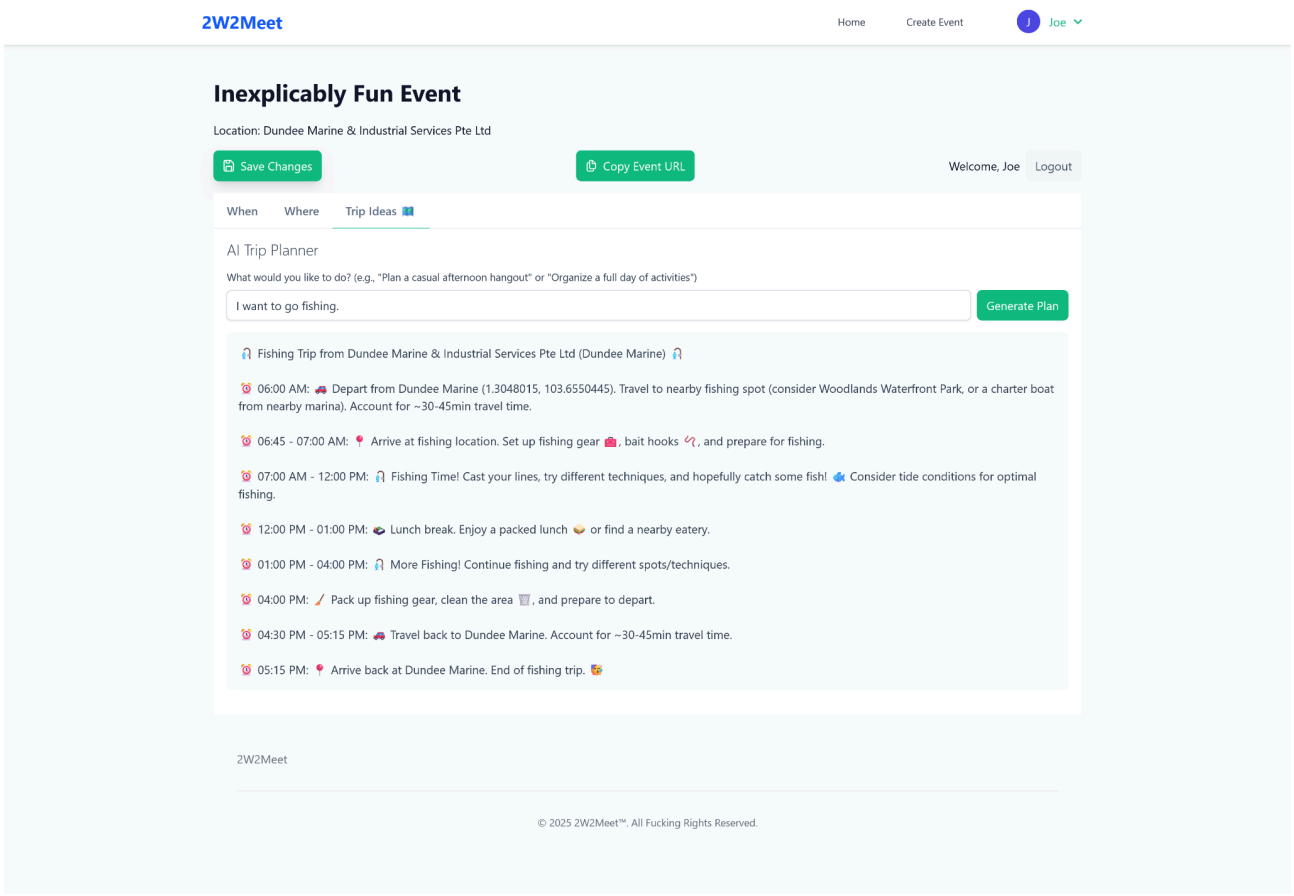
Calendar interface. Not shown in the last capture, this interface allows organisers to conveniently specify a date range.



Availability indicator. Participants mark themselves available (highlighted in green) by clicking on a time slot. The number of available participants (shown to the right of each time slot) are updated in real time.



Meetup location voting. Participants can vote for where to meet prior to the event.



AI-powered trip planner. 2W2Meet leverages a powerful generative AI model to suggest itinerary based on event context and user's need.

4. System Features

4.1 Functional Requirements

1. User Management

1.1. The system must allow Users to **sign up** for a user account.

1.1.1. The system must generate and/or retain any and all necessary information for identifying the User across different sessions.

1.1.2. The system must maintain the set of all Events with which the User has interacted. In conjunction with 1.1.2, the system must also retain the Role of the User with respect to that Event.

1.2. The system must allow Users to attempt to **log in** with their user account.

1.2.1. After logging in, upon the User's request, the system must retrieve and present all Events with which the User has interacted.

1.2.2. Unless otherwise specified by the User, the system shall use the User's associated email address to deliver all event changes details.

1.2.3. Unless otherwise configured by the User, the system must remember the User's credentials in order to automatically log him in on the next session.

1.3. When a User is not logged in, the system must allow guest mode interactions with Events.

1.3.1. An Organiser must be allowed to create an Event without an account.

1.3.1.1. The Organiser information of such Events shall be undefined.

1.3.2. Unless otherwise specified by the Organiser upon Event creation, a Participant must be allowed to join the event.

1.3.2.1. The guest mode Participant shall be given the same access rights to the Event as any other Participants.

1.3.2.2. The guest mode Participant shall be assigned a temporary, Event-specific credential bound to the nickname he specified.

2. Event Management

2.1. The system must retain the **Event Name** for each of the recorded Events.

2.1.1. The Event Name must be a non-empty string.

2.1.2. The Event Name must not be longer than 50 characters.

2.1.3. If the User inputs a name longer than 50 characters, the system must truncate it to 50 characters.

2.1.3.1. The system must display a warning message: “Event name truncated to 50 characters.”

2.1.4. If the User did not specify a name, the default Event Name shall be “Untitled Event”.

2.1.5. If the input of Event Name does not satisfy all of the criteria stated above, the system must reject the input and no write to the database shall be made.

2.2. The system must retain the **Allowable Date Range** for each of the recorded Events.

2.2.1. The two (2) dates designating the Allowable Date Range must be specified by the User.

2.2.1.1. The Allowable Date Range shall be empty and invalid otherwise.

2.2.1.2. If the Allowable Date Range is not specified by the User, the system must present the error message: “Please select a range of dates on which the event may be held”.

2.2.2. Neither of the two dates of the Allowable Date Range shall be earlier than the date on which the Event is created.

2.2.3. If the input of Allowable Date Range does not satisfy all of the criteria stated above, the system must reject the input and no write to the database shall be made.

2.3. The system must retain the **Allowable Time Range** for each of the recorded events.

2.3.1. If not specified by the user, the system shall apply the default Allowable Time Range of 9:00 AM - 5:00 PM.

2.4. The system must retain the **Venue** for each of the recorded events.

2.4.1. The Venue must be specified by the User.

2.4.1.1. The Venue shall be empty and invalid otherwise.

2.4.1.2. If the Venue is not specified by the User, the system must present the error message: “Please select a venue”.

2.4.2. If the input of Venue does not satisfy all of the criteria stated above, the system must reject the input and no write to the database shall be made.

2.5. The system must retain the **Response Deadline** for each of the recorded events.

2.5.1. The Response Deadline must be specified by the User.

2.5.1.1. The Response Deadline shall be empty and invalid otherwise.

2.5.1.2. If the Response Deadline is not specified by the User, the system must present the error message: “Please select a deadline”.

2.5.2. The Response Deadline must not be earlier than or equal to the date on which the Event is created.

2.5.3. The Response deadline must not be later than any and all of the dates designated by the Allowable Date Range.

2.5.4. If the input of Response Deadline does not satisfy all of the criteria stated above, the system must reject the input and no write to the database shall be made.

3. Landing Page

3.1. The Landing Page (hereinafter “the page” within section (1)) must include a message on the purpose of the website.

3.1.1. The message: “Create events, share them with others, and find the best time and place to meet.”

3.2. The page must include a button with the inner text “Create Event”.

3.2.1. The button must redirect an **Organiser** to the **Event Creation Page**.

3.3. The page must include a “How It Works” section.

3.3.1. The section must include a brief explanation of how to use the website (e.g., “1. Create an event. 2. Share the link. 3. Choose the best time and place.”).

3.4. The page must include a footer with additional information.

3.4.1. The footer must include links to “About Us”, “Privacy Policy”, and “Contact Us”.

4. Event Creation Page

4.1. The page must display an **Event Creation Form**.

4.2. The Organiser must be allowed to create an Event by submitting the Event Information entered in the **Event Creation Form**.

4.2.1. A “Submit” button must be displayed at the bottom of the Event Creation Form.

4.2.2. When the Organiser clicks the “Submit” button, the system must validate the event details entered in the Event Creation Form.

4.2.3. If the validation fails, the system must display an error message next to the invalid field: “Unable to create the event. Please try again.”

4.2.4. Upon submission, the system must generate an Event ID for the Event.

4.2.5. Upon successful creation of the Event, the Organiser shall be redirected to the Unique Event Page corresponding to the newly created Event.

4.3. In the Event Creation Form, the Organiser must be allowed to set the **Event Name**.

4.3.1. The setting must be done via a text input field.

4.3.2. Input is validated and corrected in accordance with section 2.1.

4.4. In the Event Creation Form, the Organiser must be allowed to configure the **Allowable Date Range**.

4.4.1. The configuration must be done via a calendar interface that supports the selection of two or more dates simultaneously.

4.4.2. The calendar interface shall not allow the Organiser to select any dates in the past.

4.4.3. Input is validated in accordance with section 2.2.

4.5. In the Event Creation Form, the Organiser must be allowed to configure the **Allowable Time Range**.

4.5.1. The configuration must be done via a slider input with a precision of 30 minutes.

4.5.2. Input is corrected in accordance with section 2.3.

4.6. In the Event Creation Form, the Organiser must select the **Venue**.

4.6.1. The selection must be done via a location selection tool (e.g., Google Maps API).

4.6.2. Input is validated in accordance with section 2.4.

4.7. In the Event Creation Form, the Organiser must set the **Response Deadline**.

4.7.1. The configuration of the **Response Deadline** must be done via a calendar interface.

4.7.2. The calendar interface shall not allow the Organiser to select any dates in the past.

4.7.3. Input is validated in accordance with section 2.5.

5. Unique Event Page

5.1. The page must display the **Event Name** at the top.

5.2. The page must display the **Venue** below the event name.

5.2.1. The location must be displayed in the following format:

5.2.1.1. "Location: [Address or Place Name]" (e.g., "Location: Central Park, New York").

5.3. The page must use a Summary Grid (hereinafter “the grid”) to display the **numbers of available Participants**.

5.3.1. The grid must display Time Slots in 30-minute increments within the configured date and time range.

5.3.2. The grid’s rows must represent the Starting Date.

5.3.2.1. Each row on the grid must correspond to one potential Starting Date.

5.3.2.2. The rows on the grid must be sorted top-to-bottom, in ascending order, by the Starting Dates they correspond to.

5.3.3. The grid’s columns must represent the Starting Time.

5.3.3.1. Each column on the grid must correspond to one potential Starting Time.

5.3.3.2. The columns on the grid must be sorted left to right, in ascending order, by the Starting Times they correspond to.

5.3.4. Time Slots outside the configured range must not be shown.

5.3.5. Each Time Slot must display their corresponding number of available Participants.

5.3.6. Each Time Slot must display their corresponding time of the day.

5.3.7. The horizontal header of the grid must display the corresponding Starting Date on top of each column.

5.3.8. Each Time Slot must be presented as a checkbox, for the Participant to toggle his availability.

5.4. The page must allow Participants to **indicate their availability** by selecting Time Slots on the Summary Grid.

5.4.1. Participants must be able to click on a Time Slot to mark themselves as available.

5.4.2. Selected Time Slots must be highlighted in green.

5.4.3. Participants must be allowed to deselect a Time Slot by clicking it again.

5.4.4. The system must store each Participant's availability in real time.

5.4.5. A Participant must be allowed to change his availability in any of the Time Slots when he revisit the Event using the same user credential.

5.5. The page must display **weather information** for the event location.

5.5.1. The weather information must be displayed as an icon (e.g., "Sunny") for each time slot in the event schedule.

5.6. The page must display a list of recommended **Meetup Location**s nearby.

5.6.1. The list must include the following details for each location:

5.6.1.1. Name (e.g., "Starbucks").

5.6.1.2. Distance from nearest bus stop or MRT station (e.g., "0.5 km").

5.6.1.3. Rating (e.g., "4.5 stars").

5.6.1.4. Category (e.g., "Cafe", "Restaurant").

5.6.1.5. Vote count (e.g. 2).

5.6.2. Each location in the list must include a "Vote" button.

5.6.2.1. Clicking the "Vote" button must increment the vote count for that location.

5.6.2.2. The vote count must be displayed next to the "Vote" button (e.g., "Votes: 5").

5.6.2.3. Each Participant can vote for only one location. If they vote for another location, their previous vote must be removed.

5.6.2.4. A Participant must be allowed to change his vote when he revisit the Event using the same user credentials.

5.6.3. The list must be sorted by distance (closest first).

5.6.4. If no meetup locations are found, the system must display a message: “No nearby meetup locations found.”

5.6.5. Each location in the list must be clickable, opening a new tab with more details (e.g., Google Maps link).

5.7. The page must allow users to share the **Unique URL** with others.

5.7.1. The page must provide a “Copy URL” button that copies the Unique URL to the clipboard.

5.7.2. If the “Copy URL” button fails, display the message: “Failed to copy URL. Please try again.”

5.7.3. After copying, the page must display a confirmation message: “URL copied to clipboard.”

5.8. If the Participant is in guest mode, the page must provide an input field for Participants to **submit their email addresses**.

5.8.1. The page must indicate that the email address submission is optional.

5.8.2. The page must display the purpose of the email address submission as “Be notified when the event arrangement is finalised.”

5.8.3. If an email address is submitted through this input field, the system must maintain the email address with an association to the Event.

5.9. The page must provide an interface to an AI (Artificial Intelligence) trip planning service agent (hereinafter “the agent”).

5.9.1. The system must initially suggest the agent about the event’s Venue and starting Time.

5.9.2. Upon request by the Participant, the agent must provide trip details in natural language.

6. Finalisation and Notifications System

6.1. Upon the expiry of an Event’s **Response Deadline**, the system must notify all the associated email addresses with the finalised details of the event.

6.1.1. The notification must contain the finalised Date and Time of the Event.

6.1.1.1. The finalised Date and Time is that with the highest count of available participants.

6.1.1.1.1. At the time of finalisation, the count of participants may be zero (0).

6.1.1.2. If there is more than one Date and Time with the highest number of available participants, the earliest Time Slot shall be taken as final.

6.1.2. The notification must contain the finalised Meetup Location.

6.1.2.1. The finalised Meetup Location has the highest number of votes.

6.1.2.2. If there is more than one Meetup Location with the highest number of votes, the system will perform a random choice among said locations.

6.1.2.3. If the system was unable to suggest any Meetup Location due to the inapplicability thereof (see statement 5.6.4), the Venue must be used in place of the finalised Meetup Location.

4.2 Use Cases

UC0 - Unique URL

Use Case ID:	UC0.1		
Use Case Name:	Construct Unique URL		
Created By:	Kota Siri	Last Updated By:	Song Yick Qi
Date Created:	3/2/2025	Date Last Updated:	7/2/2025

Actor:	Database, URLGenerator
Description:	EventController generates a unique URL pointing to the Event
Preconditions:	1. EventController supplied URLGenerator with an arbitrary eventId
Postconditions:	1. Valid Unique URL pointing to the Event is generated; **or** 2. EventController receives an error message regarding the failed URL generation.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. URLGenerator retrieves the eventName by eventId from the database. 2. URLGenerator strips all non-alphanumerical characters from eventName. 3. URLGenerator replaces all whitespaces in eventName with hyphen (“-”). 4. URLGenerator creates a UniqueURL with “event/<eventId>/<eventName>” as path. 5. URLGenerator returns UniqueURL to the EventController.
Alternative Flows:	AF0-S1: Nonexistent Event 1. eventId supplied is not found within the Database. 2. Use case exits with precondition 1.
Exceptions:	E1: Database Failure 1. If the Database fails to respond for any reason, URLGenerator throws an exception of unspecified type.
Includes:	N/A
Special Requirements:	N/A
Assumptions:	
Notes and Issues:	[7/2/2025] - changed use case to subordinate of UC0 [8/2/2025] - eventName in path is a slug for human-readability, and is ignored by the System. eventId is the sole identifier and different Events may have the same eventName.

UC1 - User management

UC1.1 - Sign Up

Use Case ID:	UC1.1		
Use Case Name:	Sign up		
Created By:	Kota Siri	Last Updated By:	Kota Siri
Date Created:	13/04/2025	Date Last Updated:	13/04/2025

Actor:	User
Description:	Enables a user to create an account by providing their name, email address, and password. They can log in, create, or join events when the system saves their login information and provides them a userId.
Preconditions:	1. The User has not signed up before. 2. The User is on the Landing Page.
Postconditions:	1. A new User entity is created with a unique userId. 2. The credentials - name, email and password - are stored. 3. User is redirecting to the Landing Page.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. The User clicks on the Sign Up button on the Landing Page. 2. The registration form pop up is then triggered and prompts the user to enter their credentials - name, email, password. 3. The User keys in the details and clicks submit 4. The system then runs a validation check on the credentials entered. 5. A User entity is created with a unique userId. 6. The User is logged in and is taken to the Landing Page.
Alternative Flows:	AF1-S1: This Email is already being used. 1. The system checks that the Email that was keyed in is already being used. 2. The User is prompted to use another email or log in instead.
Exceptions:	E1: Network error 1. The system doesn't respond when the user submits the form. 2. An error is displayed and the submission is cancelled.
Includes:	N/A
Special Requirements:	1. Password must meet the validation requirement. 2. The password should match the confirm password. 3. Unique email must be used.

Assumptions:	The User has a stable internet connection and used a supported browser.
Notes and Issues:	The User can also use the Google Sign Up function but it is totally optional.

UC1.2 - Login

Use Case ID:	UC1.2		
Use Case Name:	Login		
Created By:	Kota Siri	Last Updated By:	Kota Siri
Date Created:	13/04/2025	Date Last Updated:	13/04/2025

Actor:	User
Description:	Allows a User that has already signed up to enter into their account using their unique email and password to create an event.
Preconditions:	1. The User has signed up before. 2. The User is on the Landing Page.
Postconditions:	1. System was able to verify the credentials 2. User is authenticated successfully and is directed to the Landing Page. 3. The system retrieves the Event if an event has been created by this User, and User data.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. The User clicks on the Login button on the Landing Page. 2. A pop up of the Login form is triggered to be displayed. 3. The User is prompted to enter their email and password which the system then validates. 4. Upon successful validation the User is logged in and a session with userId is created. 5. User is taken to the Landing Page.
Alternative Flows:	AF2-S1: Incorrect credentials are entered. 1. System detects a mismatch between the email or password. 2. An error is displayed asking the User to enter the right credentials.

Exceptions:	E1: Network error 1. The system doesn't respond when the user submits the form. 2. An error is displayed and the submission is cancelled. 3. User is prompted to retry.
Includes:	N/A
Special Requirements:	1. Passwords need to be encrypted. 2. Password visibility toggling needs to be allowed for Users. 3. Optional Google Sign-In.
Assumptions:	User remembers their login details.
Notes and Issues:	Considering the possibility of a password reset.

UC1.3 - View Profile

Use Case ID:	UC1.3		
Use Case Name:	View Profile		
Created By:	Song Yick Qi	Last Updated By:	Song Yick Qi
Date Created:	13/04/2025	Date Last Updated:	13/04/2025

Actor:	User, Database
Description:	Allows a User that has already signed up to view their events (created/interacted with).
Preconditions:	1. The User holds a valid credential as the consequence of UC1.1 or UC1.2.
Postconditions:	1. The User is presented with the list of their events.
Priority:	LOW
Frequency of Use:	LOW
Flow of Events:	1. User requests to view profile. 2. UserController uses the User's event list to request each Event from EventController. 3. EventController replies with the list of Events. 4. UserController displays the Event Name, Allowable Date Range, Location, as well as the User's Role and a link to the corresponding Event Page in table form.

Alternative Flows:	AF-S1: User in Guest Mode 1. The User does not hold a valid credential. 2. The system displays an error message “Please log in to view your profile.”
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC2 - Create Event

UC2.1 - Submit Event Creation Form

Use Case ID:	UC2.1		
Use Case Name:	Submit Event Creation Form		
Created By:	Song Yick Qi	Last Updated By:	Song Yick Qi
Date Created:	1/2/2025	Date Last Updated:	1/2/2025

Actor:	Organiser, Map API, Calendar API
Description:	The Organiser fills up and submits the Event Creation Form to create an Event.
Preconditions:	1. The Organiser clicked the “Create Event” button in the Landing Page; **or** 2. The Organizer initiated an HTTP request with “/create/” in the request line.
Postconditions:	1. An Event is created and stored in the system’s database, and the User is redirected to the relevant Unique Event Page; **or** 2. The user receives an explanation of why the Event cannot be created.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. EventController requests the date selection interface from the Calendar API. 2. EventController requests the location selection interface from the Map API. 3. The Organizer fills up the displayed form in the EventCreationPage with the desired event information.

	4. The Organizer clicks the “Submit” button. 5. EventController verifies the Organiser’s input. 6. EventController creates the Event. 7. EventController posts the Event to the Database. 8. With the last-inserted eventId, EventController redirects the Organiser to the Unique Event Page using the included Display Unique Event Page. Organiser becomes Participant.
Alternative Flows:	AF1-S1: Invalid Input 1. The EventController displays an error message and indicates the invalid field(s), accordingly. 2. System returns to step 3 with the field inputs retained. AF1-S2: Truncated Input 1. The EventController displays a warning stating the truncated text and indicates the truncated field. 2. Execution proceeds normally.
Exceptions:	E1: API Failure 1. If any of the APIs fails to respond for any reason, the Event Creation Page will not load. Instead, an error message “API Failure” will be displayed. E2: Database failure 1. If the Database fails to respond for any reason, the Event will not be created. Instead, an error message “Database Failure” will be displayed. 2. System returns to step 3 with the field inputs retained.
Includes:	UC2.1 - Display Unique Event Page
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC3 - Unique Event Page

UC3.1 - View Unique Event Page

Use Case ID:	UC3.1
Use Case Name:	View Unique Event Page

Created By:	Kota Siri	Last Updated By:	Song Yick Qi
Date Created:	3/2/25	Date Last Updated:	10/2/2025

Actor:	Participant
Description:	EventController retrieves and renders the Unique Event Page upon user demand
Preconditions:	1. Participant submits an eventId/Unique URL via the Landing Page; **or** 2. User initiates HTTP request to the EventController with /<eventId>/ in the request line; **or** 3. Event creation succeeded.
Postconditions:	1. Unique Event Page is displayed *correctly*.*
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. EventController enters the included Retrieve Event Information to obtain all data necessary for rendition. 2. EventController creates the Summary Grid and Individual Grid according to the Allowable Time Range and Allowable Date Range of the Event. a. Only Summary Grid is rendered on the page until the Participant logs in. b. EventController traverses the Availability Summary of the Event to color the Summary Grid accordingly. 3. EventController enters the included Request Venue Data to display a graphical indication of the Venue. 4. EventController enters the included Request Meetup Suggestions to construct a poll for Meetup Locations. 5. EventController renders all other input fields, namely a. Participant's name b. Optional password c. Opt-in email address
Alternative Flows:	N/A
Exceptions:	E1: API Failure 1. If any of the APIs fails to respond for any reason, the Unique Event Page will not load. Instead, an error message "API Failure" will be displayed.
Includes:	UC3.1.1 - Retrieve Event Information UC3.1.2 - Request Venue Data UC3.1.3 - Request Meetup Suggestions
Special Requirements:	N/A
Assumptions:	1. There is no issue with the APIs. 2. Event had successfully been created.

Notes and Issues:	Regarding the “correct”-ness of the Unique Event Page, a new data dictionary entry shall be added.

UC3.1.1 - Retrieve Event Information

Use Case ID:	UC3.1.1		
Use Case Name:	Retrieve Event Information		
Created By:	Song Yick Qi	Last Updated By:	Song Yick Qi
Date Created:	7/2/2025	Date Last Updated:	8/2/2025

Actor:	Database
Description:	System serves the desired Unique Event Page by eventId/Unique URL supplied by Participant.
Preconditions:	1. System received an incoming request with a specific eventId; **or** 2. Event creation succeeded.
Postconditions:	1. Requested Event is retrieved from Database; **or** 2. The user receives an error message regarding the failure in serving the Unique Event Page.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. If the request was in plain eventId, EventController obtains the URL using the included Construct Unique URL and proceeds. 2. EventController navigates to the specified path. 3. EventController queries the database via exact match of eventId. 4. Database returns the relevant Event.
Alternative Flows:	AF0-S1: Nonexistent Event 1. Supplied eventId is not found within the Database. 2. If the concerned request was that via URL, EventController redirects the User to a 404 - Not Found page. 3. If the concerned request was that via eventId entered in the Landing Page, EventController displays an error message, exiting this use case, while preserving field input for Participant’s correction.
Exceptions:	E1: Database Failure 1. If the database fails to respond for any reason, the Participant is redirected to a 500 - Internal Server Error page.
Includes:	UC0.1 - Construct Unique URL
Special Requirements:	N/A
Assumptions:	N/A

Notes and Issues:	N/A

UC3.1.2 - Request Venue Data

Use Case ID:	UC3.1.2		
Use Case Name:	Request Venue Data		
Created By:	Kota Siri	Last Updated By:	Kota Siri
Date Created:	3/2/25	Date Last Updated:	16/2/2025

Actor:	Map API
Description:	EventController requests data from Map API to show the Venue
Preconditions:	1. EventController retrieved an Event from Database for rendition of the Unique Event Page.
Postconditions:	1. A map window indicating the Venue is shown on the Unique Event Page.
Priority:	MODERATE
Frequency of Use:	MODERATE
Flow of Events:	1. EventController identifies the Event being managed and first confirms that the event has a valid venue and contains geographical coordinates. 2. EventController then accesses the Event's details and retrieves the geographical coordinates. 3. A request for the location is then sent by the EventController to the Map API. 4. EventController waits for the Map API's response and checks the content type to confirm a successful interaction. 5. The data from the API is then processed and displayed as the component required (eg: image data, map links, etc..).
Alternative Flows:	N/A
Exceptions:	E1: API Failure 1. If any of the APIs fails to respond for any reason, the Unique Event Page will not load. Instead, an error message "API Failure" will be displayed.
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC3.1.3 - Request Meetup Suggestions

Use Case ID:	UC3.1.3		
Use Case Name:	Request Meetup Suggestions		
Created By:	Song Yick Qi	Last Updated By:	Song Yick Qi
Date Created:	3/2/25	Date Last Updated:	10/4/2025

Actor:	Map API
Description:	EventController queries the Map API for a list of suggested meetup locations.
Preconditions:	1. EventController received data from a new Event Creation Form; **and** 2. The Form contains at least the Venue
Postconditions:	2. EventController completes the Event object with a list of suggested meetup locations sorted in ascending order by walking distance from Venue.
Priority:	MODERATE
Frequency of Use:	HIGH
Flow of Events:	1. EventController extracts the geometric parameters (longitude and latitude) from the Venue. 2. EventController sends a request to Map API to get nearby meetup locations. 3. Map API replies with a list of places that fits the criteria. 4. EventController extracts name, location, category, and rating from each of the places. 5. If any of the places has undefined rating, EventController imputes that field with 0.0. 6. EventController fills in the meetupLocations field of the Event object.
Alternative Flows:	N/A
Exceptions:	E1: API Failure 2. If any of the APIs fails to respond for any reason, the Unique Event Page will not load. Instead, an error message “API Failure” will be displayed.
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC4 - Respond to Event

UC4.1 - Join Event

Use Case ID:	UC4.1		
Use Case Name:	Join Event		
Created By:	Song Yick Qi	Last Updated By:	Song Yick Qi
Date Created:	10/2/2025	Date Last Updated:	10/2/2025

Actor:	Participant, Database
Description:	Participant registers or logs in per Event to make responses
Preconditions:	1. Display Unique Event Page exited without errors.
Postconditions:	1. The system verified Participant's credentials and switched to an individual-specific context; **or** 2. The Participant receives an error message regarding the failed login.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. Participant enters a name via the "Your Name" input field. 2. System traverse the Participant List keys to check for matching participant names; if positive, System goes to Step 7. 3. Optionally, Participant may create a password to protect his responses, via the "Password (Optional)" field. 4. Participant clicks the "Login" button. 5. System posts the participant name and password (empty string if not provided) to the database, completing the registration. 6. System goes to Step 8. 7. If a password was set, System compares the input to the "Password (Optional)" field with the Participant List value identified by the participant's name. 8. System sets a cookie that identifies the Participant.
Alternative Flows:	AF3-S1: Incorrect Password 1. The entered name matches one of the Participant List keys, but the entered password does not match the value of that key. 2. System rejects the login, showing an error message "Incorrect Name or Password". 3. System returns to Step 1 with field inputs retained.
Exceptions:	E1: Database Failure 1. If the database fails to respond for any reason, the Participant is redirected to a 500 - Internal Server Error page.
Includes:	N/A
Special Requirements:	N/A

Assumptions:	N/A
Notes and Issues:	N/A

UC4.2 - Respond to Event

Use Case ID:	UC4.2		
Use Case Name:	Respond to Event		
Created By:	Song Yick Qi	Last Updated By:	Kota Siri
Date Created:	10/2/2025	Date Last Updated:	16/2/2025

Actor:	Participant
Description:	Participants interact with the Event to complete its planning.
Preconditions:	1. Login exited without errors; **or** 2. Cookie associated with the Event is found, i.e., the Participant has previously logged in to the Event.
Postconditions:	1. The Availability Summary and Availability are updated and posted to the Database; **or** 2. The Meetup Location Poll is updated and posted to the Database; **or** 3. The Participant leaves the Unique Event Page without any action; **or** 4. The Participant receives an error message regarding any failed update of Event's state.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. From the stored Cookie, the AvailabilityController retrieves the participant name. 2. AvailabilityController retrieve the Participant's Availability from the Participant List values. 3. AvailabilityController colors the Individual Grid according to the Availability. 4. If the Participant chooses to indicate availability, AvailabilityController delegates to the included use case Indicate Availability. 5. If the Participant chooses to vote for Meetup Location, VoteController delegates to the included use case Vote for Meetup Location. 6. If the Participant chooses to share the Event, System delegates to the included use case Copy and Share Unique URL.

	7. If the Participant chooses to opt-in for email notification, NotificationController delegates to the included use case Opt in for notification.
Alternative Flows:	N/A
Exceptions:	E1: Database failure 1. If the system fails to record the participant's selection into the database for any reason, the system will display an error message indicating that the participant's selection was not recorded. 2. If the system fails to load the selection summary for any reason, the system will display an error message. E2: Cookie nonexistent or corrupted 1. The cookie essential for loading Participant-specific content is either incomplete or missing, or the contained participant name is not found within the Participant List. 2. System displays a blocking window with an error message "User credential not found or corrupted". 3. System returns to UC2.1 when the blocking window is dismissed, or when the page is reloaded.
Includes:	UC4.2.1 - Indicate Availability UC4.2.2 - Vote for Meetup Location UC4.2.3 - Copy and Share Unique URL UC4.2.4 - Opt-in for Notification
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC4.2.1 - Indicate availability

Use Case ID:	UC4.2.1		
Use Case Name:	Indicate availability		
Created By:	Megan Chua	Last Updated By:	Megan Chua
Date Created:	1/1/2025	Date Last Updated:	3/2/2025

Actor:	Participant
Description:	Participant selects Time Slot(s) in the Individual Grid to indicate their availability
Preconditions:	3. Display UniqueEventPage exited without errors.
Postconditions:	5. The Availability Summary and Availability are updated and posted to the Database; **or**

	6. The Participant receives an error message regarding the failed update.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. Participants click the desired time slot in the Individual Grid. 2. EventController stores Participants' availability in real-time in the TimeGrid each time a cell in the Individual Grid is selected. 3. EventController posts update of Participants' availability to the database (describes how the system posts the update of the data to the database) system update event data structure 4. 5. AvailabilityController then displays the updated TimeGrid summary on the updated summary matrix by colouring the summary matrix. 6. AvailabilityController also displayed the number of Participants in each TimeSlot selected.
Alternative Flows:	N/A
Exceptions:	E1: Database failure 1. If AvailabilityController fails to record the Participant's selection into the database for any reason, the system will display an error message indicating that the Participant's selection was not recorded. 2. If AvailabilityController fails to load the selection summary for any reason, the system will display an error message.
Includes:	N/A
Special Requirements:	N/A
Assumptions:	Real time updates for Participants' availability are successful.
Notes and Issues:	N/A

UC4.2.2 - Vote for Meetup Location

Use Case ID:	UC4.2.2		
Use Case Name:	Vote for meetup location		
Created By:	Megan Chua	Last Updated By:	Megan Chua
Date Created:	1/1/2025	Date Last Updated:	3/2/2025

Actor:	Participants
Description:	Participants vote for the venue of the meeting
Preconditions:	1. Request Meetup Suggestions exited without errors.
Postconditions:	1. Meetup Location's poll status is updated for the Event and for the acting Participant; **or** 2. VoteController raises an error and displays an error message as to why the poll status cannot be updated.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. The Participant clicks one of the "Vote" buttons from the list of Meetup Locations. 2. VoteController queries the Participant List for the Participant's current vote. 3. If the Participant has previously voted for a Meetup Location: a. If the participant's previous vote is different from what he is voting for this time, VoteController will remove his previous vote, and replace it with the latest value. b. Otherwise, VoteController removes the Participant's vote. 4. Otherwise, VoteController will record the Participant's vote as clicked. 5. VoteController posts the vote back to the database. 6. VoteController will update the displayed vote counts for the modified parts of VoteSummary.
Alternative Flows:	N/A
Exceptions:	E2: Database failure 1. If the database fails for any reason preventing VoteController from updating the Event data, an error message is displayed: "Unable to record vote at the moment. Please try again later.". 2. All other parts of the page continue to function normally.
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC4.2.3 - Copy and Share Unique URL

Use Case ID:	UC4.2.3		
Use Case Name:	Copy and Share Unique URL		
Created By:	Radinka	Last Updated By:	Radinka
Date Created:	1/1/25	Date Last Updated:	3/2/2025

Actor:	Organiser
Description:	System would create a “Copy URL” button to allow the Organiser to share the Event with others.
Preconditions:	1. Display Unique Event Page exited without errors.
Postconditions:	1. Unique URL is successfully copied onto clipboard; **or** 2. The user receives an error message as to why the URL was unable to be copied.
Priority:	LOW
Frequency of Use:	MODERATE
Flow of Events:	1. Participant clicks on the “Copy URL” button. 2. System uses the included Construct Unique URL to obtain the Unique URL. 3. System writes the Unique URL to the Participant’s clipboard. 4. System displays a message indicating the successful copy.
Alternative Flows:	N/A
Exceptions:	URL Construction Failure: 1. Construct Unique URL failed to return a Unique URL. 2. System displays an error message and exits the use case. No Unique URL is copied to the clipboard.
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC4.2.4 - Opt in for notification

Use Case ID:	UC4.2.4		
Use Case Name:	Opt in for notification		
Created By:	Radinka	Last Updated By:	Radinka
Date Created:	1/1/25	Date Last Updated:	3/2/2025

Actor:	Participants

Description:	System prompts participants to optionally submit their email to opt in for notifications.
Preconditions:	2. The User is currently navigated to the Unique Event Page.
Postconditions:	3. The system stores the email address in the database segment associated with the Event.
Priority:	LOW
Frequency of Use:	LOW
Flow of Events:	1. The Participant enters the email address. 2. The Participant attempts to submit the email address. 3. The system validates the email address format. 4. The system stores the email address in the database segment associated with the specific event. 5. The system confirms the email has been recorded by displaying a success message: “Your email has been submitted successfully.”
Alternative Flows:	AF-S1: Invalid Input 1. If the email address entered was invalid, The system rejects the submission with an error message. 2. The system returns to step 1.
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

UC5 - Access Landing page

Use Case ID:	UC5		
Use Case Name:	Access Landing Page		
Created By:	Yuxuan	Last Updated By:	Yuxuan
Date Created:	10/2/2025	Date Last Updated:	10/2/2025

Actor:	Organiser, Participant
Description:	Allows Organisers to access the Event Creation Page via a button and Participants to navigate to unique event pages by entering eventId.
Preconditions:	1. The User has navigated to the landing page URL (path: “/”).
Postconditions:	1. The Organiser is presented with the landing page content, including a "Create Event" button, and is redirected to the Create Event Page after clicking it; **or**

	2. The Participant is redirected to the unique event page corresponding to the entered eventId/unique URL.
Priority:	HIGH
Frequency of Use:	MODERATE
Flow of Events:	Organiser Flow: 1. The Organiser navigates to the landing page URL. 2. The system displays the landing page with a "Create Event" button. 3. The Organiser clicks the "Create Event" button. 4. The system redirects the Organiser to the Create Event Page. Participant Flow: 1. The Participant navigates to the landing page URL. 2. The Participant enters the eventId/URL into the input field. 3. The system verifies the event code and redirects the Participant to the unique event page corresponding to the code.
Alternative Flows:	AF5-S1: Event Page Not Found (Invalid eventId/URL) 1. The system detects that the eventId/URL does not point to an Event in the database. 2. The system displays an error message indicating that the eventId/URL is invalid or not found.
Exceptions:	E1: Landing Page Not Accessible (Server/Network Issue) 1. The User tries to access the landing page URL, but the server fails to load the page (e.g., 500 Internal Server Error). 2. The system displays a server error message. 3. Execution terminates, and the User is unable to proceed further until the issue is resolved.
Includes:	N/A
Special Requirements:	The landing page must be responsive and display correctly on various screen sizes. The landing page should load quickly.
Assumptions:	N/A
Notes and Issues:	

UC6 - Finalise event

UC6.1 - Collate Event Details

Use Case ID:	UC6.1		
Use Case Name:	Collate Event Details		
Created By:	Guan Wen	Last Updated By:	Guan Wen
Date Created:	1/1/2025	Date Last Updated:	3/2/2025

Actor:	System, Database
Description:	The system collates all the event-related data, including participant availability, votes for meetup locations, and weather forecasts, to finalize the event details.
Preconditions:	1. The Organiser has successfully created an event and shared the Unique URL with participants. 2. Participants have submitted their availability and votes before the Response Deadline. 3. Weather and location APIs are accessible for the required data.
Postconditions:	The system finalizes the event's date, time, and meetup location based on participant input and sends notifications to all registered emails with the final event details.
Priority:	HIGH
Frequency of Use:	HIGH
Flow of Events:	1. The system retrieves participant availability data from the database. 2. The system collects votes for meetup locations and retrieves weather data from the API. 3. The system determines the final Time Slot with the highest availability. 4. The system selects the Meetup Location with the highest votes (or defaults to the Venue if no locations are available). 5. The system sends notifications to all participants with final event details.
Alternative Flows:	N/A
Exceptions:	E1: API failure for weather data – display "Weather information unavailable." E2: No participant availability – notify organiser that no participants responded. E3: No meetup locations voted – default to Venue as Meetup Location.
Includes:	N/A

Special Requirements:	The system must handle the collation and notification process within 2 minutes after the Response Deadline.
Assumptions:	It is assumed that participants have provided their availability and votes before the Response Deadline.
Notes and Issues:	Consider edge cases where multiple Time Slots or Meetup Locations have equal responses. Ensure a fair selection mechanism (earliest time slot or random selection for locations).

UC6.2 - Dispatch Event Details

Use Case ID:	UC6.2		
Use Case Name:	Dispatch Event Details		
Created By:	Guan Wen	Last Updated By:	Yuxuan
Date Created:	1/1/2025	Date Last Updated:	3/2/2025

Actor:	System, Email Service API, Database
Description:	After an event is finalized, EventController calls the NotificationController to send an email containing the finalized date, time, and meetup location to all participants who submitted their email addresses. The organizer is notified upon successful dispatch.
Preconditions:	1. The system has successfully collated the event details (date, time, and meetup location). 2. Participants have provided valid email addresses. 3. The Response Deadline has expired.
Postconditions:	All participants who have opted-in receive an email with the final event details.
Priority:	MODERATE
Frequency of Use:	MODERATE
Flow of Events:	1. EventController triggers email dispatch after finalizing the event. 2. NotificationController retrieves participant email addresses from the Database. 3. NotificationController composes an email with the finalized event details. 4. NotificationController sends the email via the Email Service API. 5. Email Service API returns success.
Alternative Flows:	N/A
Exceptions:	E1: Email Service API Failure 1. The NotificationController attempts to send an email through the Email Service API, but the API fails.

	2. The event dispatch process is terminated and the process ends without sending the email. E2: Database Failure 1. The Notification Controller attempts to fetch email of participants from Database, but the Database fails. 2. The event dispatch process is terminated.
Includes:	N/A
Special Requirements:	Emails must be dispatched within 5 minutes of finalizing event details. Email content should be clear, concise, and formatted for readability.
Assumptions:	1. Participants have provided valid email addresses. 2. The Email Service API is operational.
Notes and Issues:	Consider adding a confirmation link in the email for participants to acknowledge receipt. Implement logging for successful and failed email dispatches for troubleshooting.

5. Nonfunctional Requirements

5.1 Performance Requirements

1. The Landing Page should load in under 2 seconds.
2. The Event Creation Page should load in under 2 seconds after clicking "Create Event."
3. The Unique Event Page should load in under 2 seconds after submitting event details.
4. Weather and location data should load in under 2 seconds.

5.2 Usability Requirements

1. Users must be able to create and interact with Events without logging in.
2. The website must be user-friendly. At least 95% of first-time Users must be able to effortlessly navigate around and utilise the system, following the criteria as follows:
   2.1. An Organiser must be able to create an Event in under 3 minutes.
   2.2. A Participant must be able to indicate his availability and vote for the meetup location in under 2 minutes.
3. Buttons and features should be easy to find and understand:
   3.1. The "Create Event" button should stand out on the Landing Page.
   3.2. The address input box should have a map icon next to it.
   3.3. Sorting buttons should have clear icons (e.g., a sorting icon).
   3.4. All the buttons in the website must have an associated tooltip that describes the button’s function.

3.4.1. The tooltip must be displayed when the User hovers and stops over the button.

4. The website's view must be configurable between light and dark mode.
   4.1. The default view configuration must be dark mode.
   4.2. The view configuration must be saved in User's local storage.
5. Error messages must be helpful and tell users what to fix (e.g., "Please select a valid date range").

5.3 Software Quality Attributes

5.3.1 Reliability

1. In the event of failure (e.g., when an API fails), the website should recover promptly and display a helpful message (e.g., "Location services are currently unavailable. Please try again later.").
2. User data (e.g., event history, and site configurations) must not be lost, even if the website crashes.
3. The Unique URL must be valid for at least 1 week after the event date.

5.3.2 Supportability

1. The source code of the system must be well-organized and maintainable.
2. Clear, concise, and complete documentation must be available to the developers to aid them in understanding how the system works.
3. Each data API must be substitutable for any of its competitor APIs.

5.3.3 Scalability

1. The system should be able to record up to 100,000 events without signs of performance degradation.
2. The website must be able to handle up to 500 users concurrently without signs of performance degradation.
3. Each data APIs must be substitutable for its counterpart on a larger scale.

Appendix A: Glossary

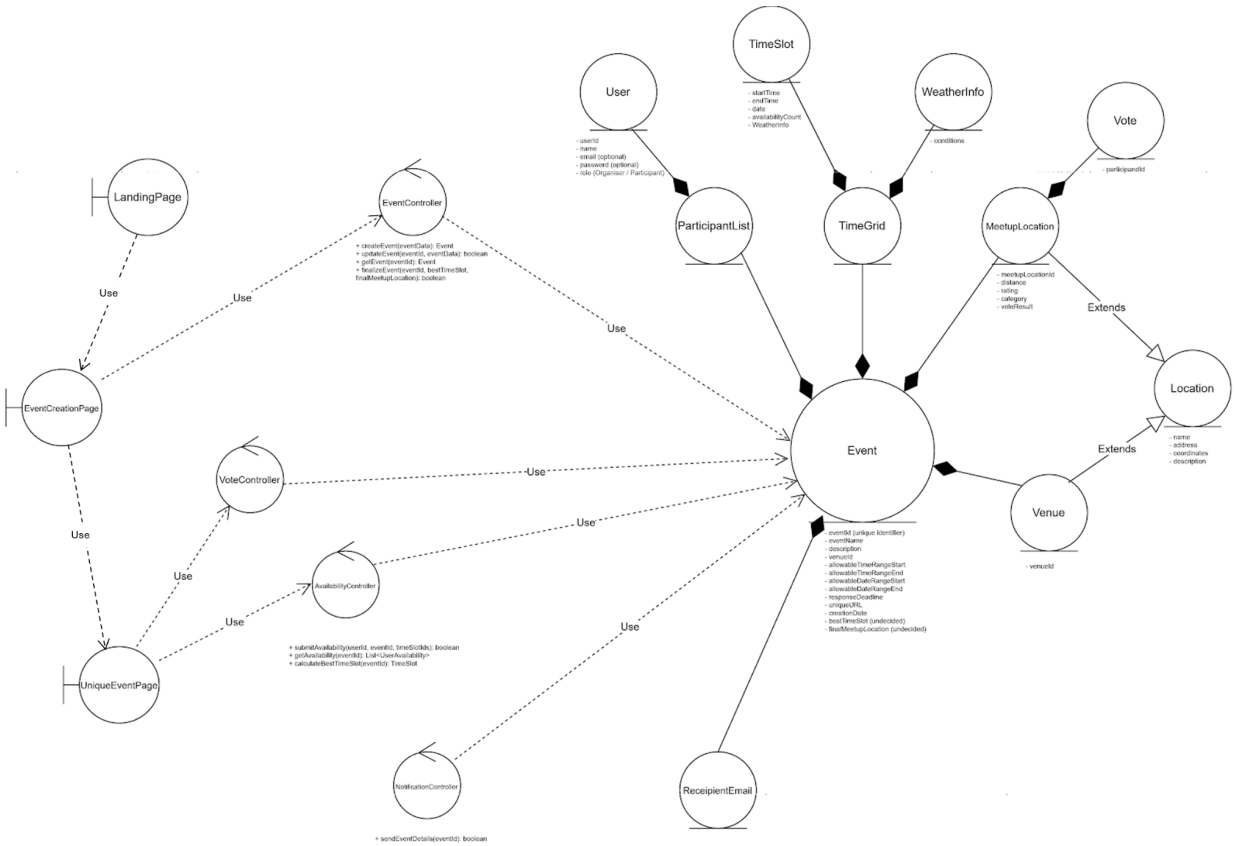
Term	Definition
Event	An object corresponding to a future real-life event being planned by the users. An Event records the possible time, venue, meetup locations, as

	well as the availability and all vote data collected from all its participants.
Event Creation Form	A form that collects the basic information needed for the creation of an Event. The required fields are Allowable Date Range, Venue, and Response Deadline.
Interact	An action, refers to the **User** creating the **Event** and/or voting for his availability and desired meetup location with respect thereto.
Organiser	The creator of an Event, i.e., the User who submitted an Event Creation Form. A User is an Organiser only within the context of the Event he created.
Participant	A User who indicates his availability and/or votes for Meetup Locations for an Event. A User is a Participant only within the context of the Event which he responds to.
User	All the clients that interact with the website. A User may assume the role of either an Organiser or a Participant, depending on the sections of the website he interacts with.
Event Name	A non-unique identifier string for events. It is user-specified upon the creation of an event, and usually describes the nature, type, category or purpose of the event.
Starting Time	A point of time in a day at which an Event may commence, at the convenience of each individual Participant.
Allowable Time Range	A range of time in a day designated by the earliest and latest Event's Starting Time allowed by the Organiser (or the system whenever applicable). For instance, the default range "09:00 AM - 05:00 PM" means that for the associated Event, the Participants may not vote for the start of the Event to be earlier than 09:00 AM or later than 05:00 PM.
Starting Date	The Date on which the Event will commence, as determined by the Organiser.
Allowable Date Range	A contiguous range of dates designated by the earliest and latest Event's Starting Date allowed by the Organiser. For instance, the range "January 1, 1970 - January 5, 1970" means that for the associated Event, the Participants may not vote for the start of the Event to be before January 1, 1970, or after January 5, 1970.
Response Deadline	The date by which a Participant may indicate his availability and/or vote for Meetup Locations for an Event. The related calendar interface only allows for the selection for a specific date, but not time. The time is fixed to 5:00 PM on the selected date.

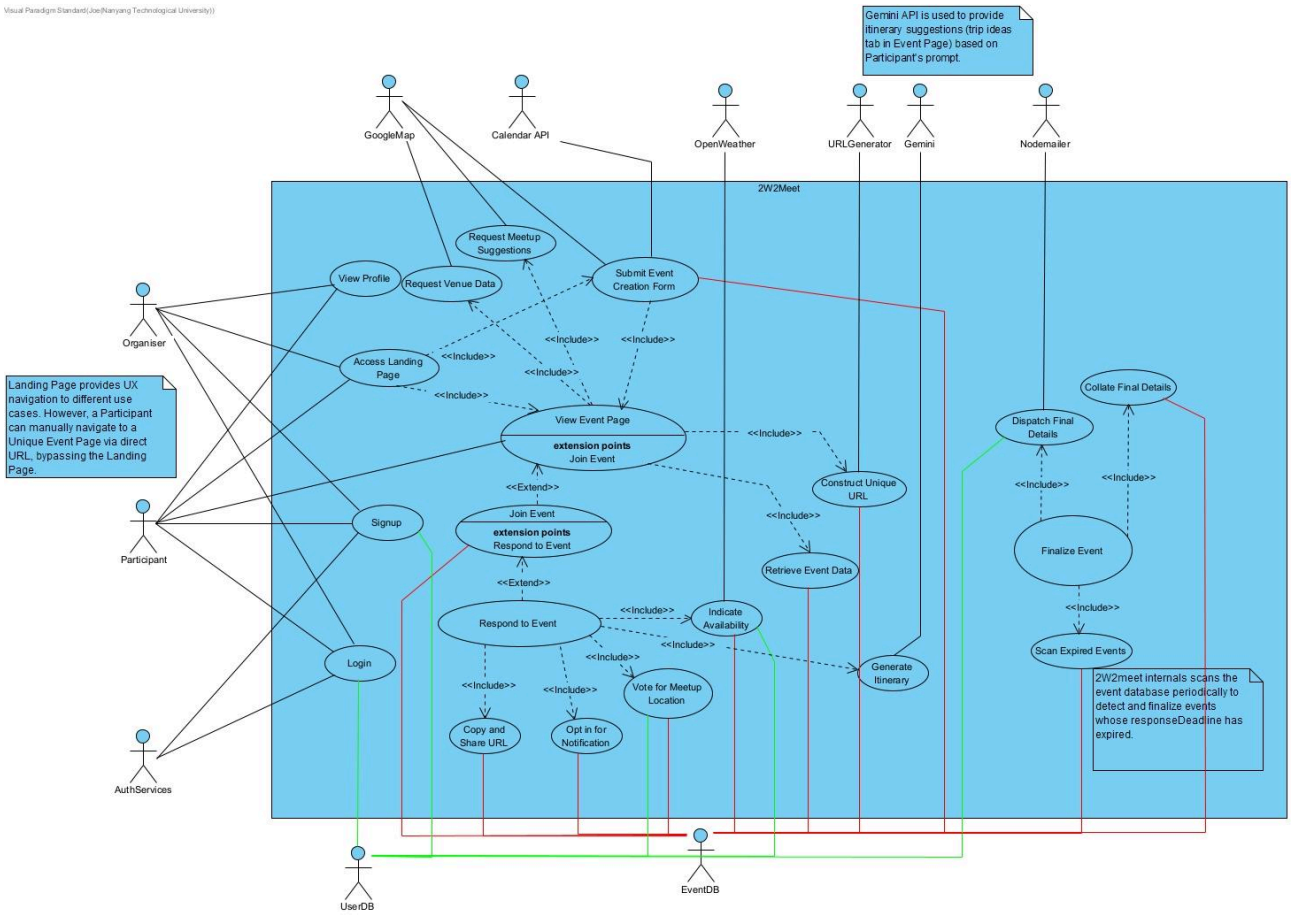
Time Slot	A combination of Starting Date and Starting Time. With both pieces of information, a Time Slot completely determines when an Event will be held
Availability	For any individual Participant and any Time Slot, Availability refers to whether the Participant wishes the event to be held on that Time Slot.
Venue	The location at which the Event is held. The Venue must not be confused with the Meetup Location.
Meetup Location	The location at which all the Participants agree to meet up prior to the start of the Event. The Meetup location may or may not overlap with the Venue.
Unique URL	The URL that uniquely locates a specific Event recorded in the system's database.
Time Grid	Internal representation of availability data for an Event. It is a matrix with as many columns as the number of days in the Allowable Date Range, and as many rows as there are 30-minute intervals in the Allowable Time Range. Each entry corresponds to one Time Slot, and its value represents the number of available Participants.
Individual Grid	Visual component illustrating the Participant's own availability. The grid is composed of Time Slots organised by date as column and time as row. The grid has as many columns as the number of days in the Allowable Date Range, and as many rows as there are 30-minute intervals in the Allowable Time Range. Each Time Slot of the grid will indicate (currently of unspecified form) whether the Participant has set himself as available.
Summary Grid	Visual component illustrating the summary of all Participant's Availability. The grid is composed of Time Slots organised by date as column and time as row. The grid has as many columns as the number of days in the Allowable Date Range, and as many rows as there are 30-minute intervals in the Allowable Time Range. Each Time Slot of the grid will have an indication (currently of unspecified form) of the number of available Participants.
Participant List	Internal representation of participant data. It is a hash map with the Participant's name as key and associated Password, individual Time Grid, and Meetup Location vote as value.

Appendix B: Analysis Models

2W2Meet Conceptual Model



2W2Meet Use Case Diagram



Source: http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc