**CAB203 GroupUp! Design Document and Software Requirement Specification Sheet**

**Log for modifications to Design Document**

| **Name** | **Modifications (+/-)** | **Date Changed** |
| --- | --- | --- |
| **Jinyoung Choi** | * **Initial ideation of the project** * **Project description** * **User stories** * **Project functionality** * **Frontend planning** * **Pages, modal, input** * **Validation checks** * **Timeline and milestone** * **User stories update** | **12th of March**  **14th of March** |
| **Kevin Nguyen** | * **Backend data ideation** * **Project Functionality** | **14th March** |
| **Daniel Paza** | **+ explanation of SDLC**  **+ agile approach methodology description**  **+ simplified project requirements**  **+ explanation of projects theme for “Digital Wellbeing”**  **+ relevance of group up to digital wellbeing**  **+ User story format, syntax and example** | **14Th of March** |
| **Daniel Paza** | * Fixed some user stories * Added acceptance criteria with functional and nonfunctional requirements to user stories | **15th of March** |
| **Daniel Paza** | * **Added a user story and acceptance criteria** * **Added dot points for UI considerations from lecture content** | **16th Of March** |
| **Daniel Paza** | * **Re defined the project purpose** * **Added another user story along with acceptance criteria** * **Defined the project scope including objective, inclusions, exclusions, limitations and constraints** * **Fixed professionalism of specification document formatting** * **Added section explaining functional and nonfunctional requirements** | **17th of March** |
|  |  |  |

**Table of Contents**

[Description of Project 4](#_2u9x1os3fj98)

[Project Purpose: 4](#_7aul4hwczfek)

[Project Scope: 5](#_wovcgtqimp6s)

[1. Objective: 5](#_te2q05scnvz1)

[2. Inclusions: 5](#_nmfv1rig9jdl)

[Project Considerations (From Lecture) 5](#_zakfwd9wluwm)

[3. Exclusions: 6](#_h4d536fyqt6h)

[4. Limitations/Constraints: 6](#_jhoj4n20rhqs)

[Software Development Life Cycle (SDLC) 6](#_wa4oed8lrlkc)

[Projects Agile approach for the SDLC 6](#_up4yio55wa3t)

[Naming convention for GIT branches 7](#_77opldl5vu4m)

[Features and Requirements 8](#_czd3dipizfie)

[User story Structure: 8](#_ye1ajzrmz7eq)

[Example of a well formed user story 8](#_eqhkmfcly2k6)

[What are functional requirements? 9](#_w6mjhxqajqf9)

[What are non functional Requirements? 9](#_ge5yvafqtm0p)

[Example of extracting functional and non functional requirements: 9](#_tuqtdfje05dq)

[User stories and Acceptance Criteria: 10](#_mkg3pgbloea6)

# **Description of Project**

GroupUp! is an innovative social networking and event organising app designed to connect individuals with shared interests, hobbies, or activities. Unlike traditional event-hosting platforms that primarily focus on connecting friends or established groups, GroupUp! introduces a unique twist by facilitating events with random participants who share a common interest.

**Understanding the context of digital wellbeing:**

Digital well being refers to the impact of digital technologies on what it means to live a life that is good for a human being. It encompasses the effects of technology on individuals' health, comfort, and overall well-being. Digital wellbeing involves managing and balancing the use of technology in daily life to prevent negative impacts such as digital fatigue, eye strain, and mental health issues. It emphasises making conscious choices about digital consumption and using technology in a way that enhances physical and mental health. Our assignment emphasises the importance of designing software with digital wellbeing in mind. It suggests that we create applications that support the physical and mental health of our stakeholders by promoting mindful usage and or enhancing the quality of users' lives i.e facilitating wellbeing, preventing isolation and or reducing anxiety etc.

# **Project Purpose:**

GroupUp! aims to promote digital wellbeing by facilitating meaningful connections among individuals with shared interests, hobbies, or activities. By connecting users based on common interests, the app encourages social interactions that can contribute to users' overall well-being. Based on secondary research, studies have shown that meaningful social connections can have positive effects on mental health and emotional well-being, reducing feelings of loneliness and isolation. Therefore, the **purpose of this project must be to enhance real life interactions amongst software users to reduce social isolation**. Alongside this purpose, groupUp **must also foster interactions between a diverse range of individuals so that the app promotes social inclusion and diversity**. Of which will all contribute to a greater level of social well being for the users involved. As GroupUp! encourages users to engage in offline activities and social events a **further purpose of this project is to promote a balanced approach to technology use for all users**. By facilitating real-world social interactions and offline activities, GroupUp! helps users maintain a greater level of wellbeing by indirectly guiding a healthy balance between online and offline experiences. Finally by leveraging the social networking and event organising features of GroupUp!, the **project is expected to provide users with opportunities to build social support networks and connections**. Studies show that social support plays a crucial role in promoting resilience and coping with stress, therefore contributing to overall well-being.

# Project Scope:

## 1. Objective:

* Defined in project purpose (above)

## 2. Inclusions:

* GroupUp! will include features such as event creation and discovery
* The platform will only be developed for PC
* The project scope must/extend the requirements and project considerations below using JAVA/JAVAFX:
  + A visual interface using JavaFX, providing windows for key functions.
  + An authentication system for user sign-up and sign-in, integrated with both the graphical interface and data models.
  + A system for storing, retrieving, and updating user data, integrated with both the graphical interface and data models.
  + One or more application windows where users can perform the main tasks of the application, also integrated with the graphical interface and data models.

### Project Considerations (From Lecture)

* Choose a legible font: Sans-serif fonts preferred (e.g. Arial, Helvetica, Vedana)
* Use a reasonable font size: Ensure it is affected by system scale. Fonts appear smaller on high-DPI devices
* One-handed Operation: Some users won’t be able to use two hands. Your application
* should function with only a keyboard.
* Keyboard Shortcuts: Assign keyboard shortcuts to common actions.
* Clear Error Messages: When there is invalid user input, clearly highlight where it is and what is wrong.
* Avoid Error Messages: Where possible, prevent errors from occurring or suggest solutions.
* Constrain Input: Only allow input of the correct type to be entered – e.g. no text in a phone number.
* Highlight Errors: If there is an error in the user input, highlight exactly where it is.
* Clearly Label Required Fields

## 3. Exclusions:

* GroupUp! does not extend to providing professional mental health services or counselling within the app.
* The project will not focus on advanced AI algorithms for personalised recommendations beyond basic interest matching.

## 4. Limitations/Constraints:

- Development will be constrained by available resources, including time, and technical expertise.

- The app's effectiveness in promoting digital wellbeing may be limited by external factors such as user behaviour, preferences and the actual availability of users to genuinely appear at in person/registered events.

- GroupUp! Is expected to face challenges in ensuring both digital and physical user safety and privacy, as a result robust security measures must be considered for user data and compliance with relevant regulations.

# **Software Development Life Cycle (SDLC)**

Systematic (development of software will be structured and organised)) process to develop high quality software (software that meets stakeholder requirements) that involves multiple phases (7 agile phases) with distinct processes, methods and deliverables (each phase is independent of one another).

# **Projects Agile approach for the SDLC**

Throughout this assignment an agile methodology will be used to facilitate the SDLC of our group's assignment. Agile is an iterative approach (meaning one feature at a time) to software development that breaks the developmental process into increments called sprints. Sprints in our case will be held as physical meetings after CAB302’s tutorial everyday Monday. Sprints will generally consist of identifying the functionalities/s of our program which we want to implement, how previously delegated tasks are progressing and whether more time is needed for specific tasks etc.

The Agile software development life cycle will follow the phases below:

1. Initiation: The assignment vision and objectives are established, and initial requirements from canvas are gathered (within this document). The team forms and begins planning for the first sprint.

2. Sprint Planning: The team during the sprint collaboratively selects and prioritizes user stories or features to be developed during the sprint. Tasks are defined, and the sprint goal is set. Its expected that during our first sprint, the documentation and specification sheet (this doc) will likely be worked on/expanded to gain a deeper understanding of functionalities and requirements.

3. Development: Developers work on implementing the selected features assigned to them, often in pairs or small groups. During this phase group members should appropriately create and work on GIT branches which are named relevant to their user stories. Best practices are to create a branch in the naming convention of:

### **Naming convention for GIT branches**

*“WIP-(simplified explanation of user story functionality)”*

I.e *WIP-sign-up-page-front-end*

4. Testing: Testers verify the functionality of the developed features, ensuring that they meet any defined acceptance criteria and quality standards (requirements for specific minimum features will be defined below).

5. Review and Feedback: At the end of the sprint, the team demonstrates the completed work to each other, gathering group feedback for further refinement.

6. Sprint Retrospective: The team reflects on their performance during the sprint, identifying areas for improvement and discussing strategies to enhance efficiency and collaboration.

7. Repeat: The cycle repeats with the start of a new sprint.

# **Features and Requirements**

## User story Structure:

User stories MUST (to be discussed) follow the syntax below:



· Each user story should have ONE MAIN FUNCTIONALITY and be short and concise.

· The “because” of a user story should be written if possible, otherwise it can be omitted.

· Acceptance criteria are specific conditions or requirements that must be met for a user story to be considered completed and accepted. These should be written as data points after the user story is finalised.

## Example of a well formed user story

*“As a user I want to view all upcoming open music events based on a specific music genre because I want to choose events which interest me.”*

Acceptance Criteria:

1. When creating a music event logged in users must be able to select at least one type of music genre for the event by using the entry field or drop-down menu to choose from the list of predetermined music genres.

2. All users should be able to use the drop-down and entry field menu to select one or more music genres. The selection should then prompt the system to automatically filter through all events and get/display the relevant music events with the specified genre on the main event screen.

3. For all users the genre of a music event must be clearly displayed when that event is shown on the main event page.

4. Implement a basic reset button or icon on the drop down and search menu that when clicked will automatically reset the event page and display all music events of any genre.

5. Implement a feature on the dropdown/search menu that notifies the user of the number of events that are available for a specific music genre or if there is none available.

For the acceptance criteria above there were other requirements externally defined. As a result some acceptance criteria cannot be extracted from the user story alone. However, as a group we will get to define our own external acceptance criteria.

The acceptance criteria whilst generally okay are not feasible for the extent of this project because they do not explicitly define the functional and non functional requirements. Both non functional and functional requirements are imperative to the software development life cycle as they make analysing and extracting the core functionalities simple and straightforward.

# What are functional requirements?

From the user story, these specify the functionality that the software must provide to fulfil user needs. Functional requirements describe what the system should do, such as actions users can perform, data manipulation, and system responses to inputs. For example, a functional requirement for a social media platform could be "users can post messages to their timeline."

# What are non functional Requirements?

From the user stories, these define the quality attributes or constraints that the software must satisfy. Non-functional requirements address aspects such as performance, reliability, security, usability, and scalability. An example of a non-functional requirement is "the system must be able to handle 1000 concurrent users without significant performance degradation."

# Example of extracting functional and non functional requirements:

User Story: As a registered user, I want to be able to log in to my account so that I can access my personalised content.

Acceptance Criteria:

Functional Requirements:

1. The login page should have input fields for entering the username/email and password.
2. Upon successful login, the user should be redirected to their dashboard page.

Non-functional Requirements:

1. The login process should have a maximum response time of 3 seconds.
2. The system should support a minimum of 1000 concurrent login requests without crashing.

# User stories and Acceptance Criteria:

1. *“As a logged in user, I want to create diverse events such as study groups, sports activities, parties, and dating gatherings because I want to provide a wide range of options to cater to different interests and preferences.”*

*Acceptance Criteria:*

Functional Requirements:

* Many different event genres should be available/selectable when an event is being created
* The genre/type of event must be visible when an event is selected from a list

Non-Functional Requirements:

* Backend systems should support a minimum of 100+ concurrently stored events without affecting the performance, reliability and functions of the app.
* System filtering should never exceed 1 minute when filtering for events based on a specific type

1. *“As a logged in user, I want to ensure that when I create an event it includes a comprehensive range of fields and details because I want potential guests to have a clear understanding of what to expect of the event.”*

*Acceptance Criteria:*

Functional Requirements:

* During event creation, the hoster should be able to specify the title, date, time, location, description, maximum number of guests, event type/genre, supporting images and cost (if applicable).
* Allow the host to add custom fields to the event registration form, capturing additional information from attendees if needed (e.g., dietary restrictions, emergency contact information).
* The host should have the option to provide contact information (e.g., email, phone number) for attendees to reach out with questions or inquiries about the event

Non-Functional Requirements:

* Event description should be limited to 1000 characters
* Backend system must support multiple image uploads for a singular event and of specific types i.e .JPG, .PNG etc

1. *“As a user, I want to browse through event listings on the app's event page so that I can discover events that align with my interests and availability.”*

*Acceptance Criteria:*

Functional Requirements:

* Provide a central window/section in the software designed to allow users to browse through all listed events
* Allow all users to click on any visible event, when done it should navigate them to a page which contains a more detailed and comprehensive view of the event.
* All users must be able to access a type of filtering system to search for events based on title, event genre, location etc.

Non-Functional Requirements:

* The system should be able to handle a user clicking multiple events but only redirecting to the first one clicked
* Ensure fast loading times and smooth scrolling for the event listings, even when there are a large number of events displayed on the page.

1. *“As a user I want to be able to sign up to GroupUp with an email, name, phone number and password because I want to access the applications features.”*

Functional Requirements:

* Make email address, phone number, username and name mandatory when registering for an account
* Upon successful registration, direct user to the login view

Non-Functional Requirements:

* + Before submitting the registration form for a user, ensure all valid fields are filled
  + Ensure the password is checked against specific criteria such as having at least 1 capital letter,1 number, 1 special character and being at least 6 characters long.
  + Ensure email supplied during registration has not already been used to sign up. If this is the case, notify the user of this conflict and request that they login under this email.

1. *“As a user I want to be able to log into my account so that I can access the website's features and my saved data.”*

Functional Requirements:

* + Ensure that login buttons are always accessible and visible
  + Ensure that any feature when clicked by a non logged in user directs them to the login form
  + After successful login the user should be directed to a home/landing page
  + Upon logging into an account, any data i.e created or booked events associated with the email/account id should be loaded and accessible
  + Implement a system for storing, retrieving, and updating user data

Non-Functional Requirements:

* + Implement a system that notifies the user if their credentials are incorrect and ask for them to be re-entered
  + Ensure that the login process is intuitive and easy to use, with clear instructions provided to the user.

1. *“As a logged in user I want to be able to access the events that I have created in their own section of the software because I want to easily manage any created events.”*

Functional Requirements:

* + Implement a window/section in the software where any events created by a user can be independently viewed and managed
  + Implement another subsection in this main window that filters out created events based on completed, outgoing or failed status

Non-Functional Requirements:

* + Implement a system that only shows the events which the logged in user has created

1. *“As a user I want to know the most up- to-date status of an event because I do not want to register for an event that's already happened or is closed.”*

Functional Requirements:

* + Ensure each event has a status i.e Open for registration, Closed for Registration, Event Ongoing (currently happening)
  + Only events open for registration should be displayed in the event discovery window
  + Only logged in should only be able to register for events that are “Open for registration”
  + Implement refreshing feature that checks for and actively removes events from the discovery window whose event status has just been changed to “Closed for registration”

Non-Functional Requirements:

* + Secure event status data to prevent unauthorised access or manipulation.
  + Implement authentication and authorization mechanisms to ensure that only authorised users can view and modify event statuses.
  + Design the system to accommodate potential increases in user activity and event volume over time.

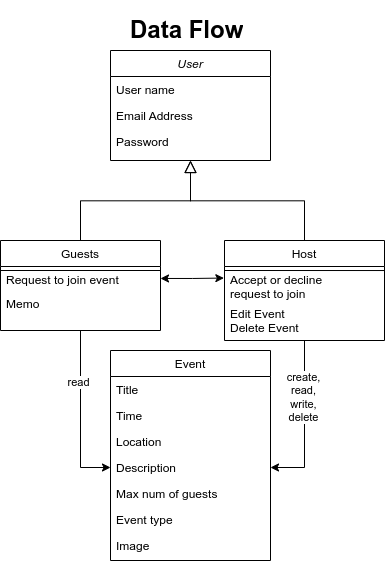
1. As a host, I want to upload an image for my event to add visual appeal and provide further context because I believe it enhances the attractiveness and understanding of the event for potential guests.
2. *Acceptance Criteria:*
3. As a guest, I want to express my interest in attending an event by clicking the "Pending Request" button and optionally write a memo to the host introducing myself or providing additional context about my interests or preferences because I want to indicate my interest and potentially provide helpful information to the host.
4. As a host, I want to receive notifications when guests express interest in attending my event so that I can review their information and assess whether they are a good fit for the event.
5. As a host, I want the authority to accept or decline guests based on my discretion because I want to curate a guest list that aligns with the objectives and atmosphere of the event.
6. As a host, I want to confirm the event once I'm satisfied with the guest list so that all event details, including date, time, location, and participant list, are sent to the guests via email for their reference.
7. As a host, I want confirmed events to be moved to my "my event" page and removed from the event page so that only confirmed participants have access to the event details, promoting a sense of exclusivity and privacy.

**Functionality:**

Host: create, update, delete, confirm events, read guest information, event description, number of guests, confirm guests, decline guests

Guests: read event, join event, add memo for the host

**Backend Data:** Event Title, Time, location, number of guests, Event description, Type of events, Guest memo, Events page, My event page



**Backend plan:**

Display Home Page (prompt login or sign up)

1. open window to login or sign up
   1. Sign up page:
      1. email
      2. Username
      3. Password
   2. Login page:
      1. Email
      2. password

**Frontend plan:**

**Fonts:** Poppins or Roboto will be used consistently throughout the application for a cohesive visual identity and readability.

**Pages:**

* **Login/Sign Up Page:**
  + Users can either log in with existing credentials or sign up for a new account.
  + Includes input fields for username, email and password.
  + Option for account creation.
* **Event Page:**
  + Displays a list of available events.
  + Each event card shows key details like title, date, time, location, and type.
  + Allows users to filter events by type or search for specific events.
  + Option to click on an event card to view more details in a modal.
* **My Events Page:**
  + Shows a personalised list of events the user has created or joined.
  + Includes options to view upcoming events, past events, and event details.
  + Allows users to manage their events, such as editing or cancelling.

**Modals:**

* **Event Detail Modal:**
  + Provides an in-depth view of a specific event when clicked from the event page.
  + Shows all event details including title, date, time, location, description, type, and number of guests.
  + Allows users to join the event, view guest list, and add memo if applicable.
  + Option to close the modal and return to the event page.

**Inputs:**

* **Event Title:**
  + Text input field for users to enter the title of the event.
* **Time:**
  + Dropdowns or date/time picker for selecting event start and end times.
* **Location:**
  + Text input or dropdown for specifying the event location.
* **Number of Guests:**
  + Numeric input field to set the maximum number of guests allowed for the event.
* **Event Description:**
  + Text area for users to provide a detailed description of the event.
* **Type of Events:**
  + Dropdown of checkboxes for users to select the category or type of event (e.g., study group, sports, party, etc.).
* **Images:**
  + Allows users to upload images related to the event, such as flyers or location pictures.

**Validation Checks:**

* **Required Fields:** Ensure all mandatory fields (e.g., title, time, location) are filled out before allowing event creation or submission.
* **Format Validation:** Validate input formats (e.g., email format for sign up, time format for event scheduling).
* **Character Limits:** Enforce character limits for input fields, preventing excessively long entries.
* **Unique Event Titles:** Check for uniqueness of event titles to avoid duplication.
* **Date/Time Conflict:** Warn users if the selected event time overlaps with an existing event they are hosting or attending.

**Timeline and milestones:**

**Version 1 (Week 6-7):**

* **Host Features:**
  + Create Events: Hosts can create new events, specifying details such as title, date, time, location, description, and maximum number of guests.
  + Update Events: Hosts can modify event details as needed, such as changing the date or updating the description.
  + Delete Events: Hosts can cancel events they've created, removing them from the platform.
  + Confirm Events: Hosts can finalise event details and confirm the event, triggering notifications to guests.
* **Guest Features:**
  + Read Events: Guests can browse through the list of events, viewing details such as title, date, time, location, and description.
  + Join Events: Guests can indicate their interest in attending an event by joining it.

**Version 2 (Week 9-10):**

* **Host Features:**
  + Read Guest Information: Hosts gain access to information provided by guests when they express interest in attending an event.
  + Confirm Guests: Hosts have the ability to approve guest requests to join an event, adding them to the guest list.
  + Decline Guests: Hosts can decline guest requests, providing reasons if necessary.
* **Guest Features:**
  + Add Memo for the Host: Guests can include a personal memo when requesting to join an event, introducing themselves or providing additional context.

**Version 3 (Week 12-13):**