
SCHOOL OF ENGINEERING AND TECHNOLOGY

COURSEWORK FOR THE BACHELOR OF SOFTWARE ENGINEERING

(HONS); YEAR 3 ACADEMIC SESSION SEPTEMBER 2023; SEMESTER 7,8,9

SWE3024: CODE CAMP

DEADLINE: 16th December 2023 12:00 pm

GROUP: 2

INSTRUCTIONS TO CANDIDATES

- This assignment will contribute 100% to your final grade.
- This is a group assignment. Each group consists of 3-5 members

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

- Coursework submitted after the deadline but within 1 week will be accepted for a maximum mark of 60%.
- Work handed in following the extension of 1 week after the original deadline will be regarded as a non-submission and marked zero.

Students' declaration:

	(Name)	(ID)	(Signature)
We 1)	Cheah Yi Shien	19115195	CYS
2)	Khoo Yukki	21016266.	Yukki
3)	Karen Ng Kai En	19043934	NKE
4)	Daniel Lam Jun Chai	20030250	<i>DL</i>
5)	Nuraisyah Binti Mohammad Iesa	18113910	NMI

received the assignment and read the comments

Academic Honesty Acknowledgement

“We (names stated above) verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties (*refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme*) for any kind of copying or collaboration on any assignment.”

1) Cheah Yi Shien

2) Khoo Yukki

3) Karen Ng Kai En

4) Daniel Lam Jun Chai

5) Nuraisyah Binti Mohammad Iesa

(CYS,DL, yukki, NKE, NMI/

19.12.2023)

Table of Content

1.0 Introduction and overview	4
1.1 Problem Statement	4
1.2 Proposed Solution	5
1.3 System Overview	5
1.4 Roles Assigned.....	6
2.0 System Requirement	7
2.1 Functional Requirement	7
2.2 Quality Attribute	17
3.0 System and Software Design	20
3.1 System Architecture	20
3.1.1 Model-View-Controller.....	20
3.1.2 Client-Server	22
3.1.3 Publish-Subscribe	22
3.2 Use Case Diagram	23
3.3 Sequence Diagram (Main)	24
3.3.1 Sequence Diagram (Sub-Diagram: Create/register/manage event)	25
3.4 Entity Relationship Diagram	26
3.5 User Interface Design.....	27
4.0 Project Planning, Management and Implementation	32
4.1 Project Planning	32
4.2 Project Management	32
4.3 Technology Stack Selection	33
4.4 Implementation.....	34
4.4.1 Navigation Bar	34
4.4.2 Home Page	36
4.4.3 Event & Club	37
4.4.4 Search Function	39
4.4.5 Single Event Page	41
4.4.6 Notification.....	45
4.4.7 Single Club Page	47
4.4.8 Club Subscription.....	50
4.4.9 Profile	51
4.4.10 Register and Login	53
4.4.11 Email Verification	54
4.4.12 Email Notification	55
4.5 Testing and Quality Assurance	57
5.0 Application of Software Engineering Knowledge in Software System Development	65

5.1 Software Architecture.....	65
5.2 Software Testing.....	66
5.3 Requirement Engineering	69
6.0 Conclusion	73

1.0 Introduction and overview

University life is filled with a diverse range of extracurricular activities that allow students to pursue their interests, build connections, and develop valuable skills outside the classroom. However, many students face difficulties in discovering and participating in these enriching opportunities due to fragmented information across various social media platforms. To address this problem, we propose Get2Gether, an event management website tailored to the needs of university students and clubs at Sunway Education Group.

Get2Gether provides a centralized platform for clubs and societies to advertise their events to the wider student population. Students can browse and register for events based on their interests with just a few clicks. Event organizers can easily publicize event details, manage registrations, and collect participant data seamlessly on our user-friendly interface. By consolidating all student events onto a single platform, Get2Gether aims to raise awareness about the vibrant student life at Sunway and motivate greater involvement from both existing and potential club members.

With features catered towards event organizers and participants, Get2Gether seeks to enrich the university experience by fostering stronger communities and connections. Students can look forward to discovering new passions, gaining new skills, and building networks that will benefit them long after graduation. By facilitating participation and engagement across various student-run activities, our platform helps ensure students make the most of their university journey with Sunway.

1.1 Problem Statement

Many university students face challenges in discovering and engaging with various student events and clubs on campus. The current approach of relying on scattered information across social media platforms like Instagram makes it troublesome for students to explore, join and manage their participants in university events. The fragmented system leads to reduced awareness, limited exposure for clubs, and a lower motivation for students to actively participate in campus activities.

Problem 1: Fragmented Information Landscape

The current scattered information across various social media platforms makes it challenging for university students to efficiently discover and engage with student events and clubs, leading to a lack of centralized and easily accessible information.

Problem 2: Limited User Motivation and Engagement

The absence of a streamlined platform for students to search, join, and manage university events results in reduced awareness, exposure, and motivation to actively participate in campus activities, hindering the overall vibrancy of the university community.

1.2 Proposed Solution

In the dynamic world of university life, staying informed about and participating in various events is crucial for student engagement and personal growth. To facilitate this, we present the University Student Event Management Website, a comprehensive platform designed to streamline the process of discovering, joining, and managing university clubs and events. The website will provide a comprehensive and easily accessible database of all student clubs, their events, and relevant information. Users will be able to search for clubs based on interests, join events with a simple click, and manage their involvement through personalized profiles. The University Student Event Management Website is poised to revolutionize the way university students discover, participate in, and organize events. By fostering community engagement, efficiency, and convenience, this platform aims to enrich the university experience for all its users.

1.3 System Overview

Title: Get2Gether By Sunway | **Overview:** An event management website that focuses on advertising upcoming events held by the Club and Societies in Sunway Education Group as well as encouraging freshies and even seniors students to be attending these events and gather together to make their journey unforgettable. This website will have features suited for both the Event Holders and Event Participants. Event Holders will benefit from having a platform to publish their event and gather data of their participants while Event Participants are provided a platform where they can browse upcoming events and register for the ones that they are interested in.

1.4 Roles Assigned

Member	Roles	Job Scope
Nuraisyah Mohammad Iesa	Architect	<ul style="list-style-type: none"> Conducted an analysis of the appropriate requirements for the system. Determined the architectural design of the system by discussing it with the rest of the team. Research and evaluate the appropriate tools and frameworks to be used by the system
Cheah Yi Shien	Tester, User	<p>Tester</p> <ul style="list-style-type: none"> Designed and executed comprehensive test cases across various testing stages Supported development team by providing guidance on code maintainability Evaluated new features to ensure its readiness to release <p>User</p> <ul style="list-style-type: none"> Provided user perspectives during requirements gathering & design phase Conducted frequent usability testing by providing continual feedback Identified user experience defects by contributing to the enhancement of navigation
Daniel Lam Jun Chai	Designer	<ul style="list-style-type: none"> Conduct user research to understand the needs and preferences of both event holders and participants. Design an intuitive and user-friendly interface that enhances the overall user experience Develop a visually appealing and consistent design that aligns with the brand identity of Sunway Education Group. Design a notification system to keep participants informed about upcoming events, changes in event details, and other relevant information.
Karen Ng Kai En	Developer	<ul style="list-style-type: none"> Set up the collaborative environment for development (GitHub) Developed the User Interface of the system using HTML and CSS Developed the underlying logic of the system using PHP

		<ul style="list-style-type: none"> • Ensured the code structure follows the System Architecture • Designed the Database needed for the system • Participated in the code review, testing and debugging • Participated in Documentation
Khoo Yukki	Developer	<ul style="list-style-type: none"> • Set up the collaborative environment for prototype (Figma) • Developed the User Interface of the system using HTML and CSS • Developed the underlying logic of the system using PHP • Ensured the code structure follows the System Architecture • Developed the whole login system included verification and error detection. • Participated in documentation

2.0 System Requirement

The section documents the functional requirement, non-functional requirement and quality attributes for the system.

2.1 Functional Requirement

ID	Name	Description	Roles
FR-01	Create an account	Users register for an account using their university email addresses, either as a participants or organisers.	User
FR-02	Login	Users like participants and organisers are allowed to login into their account.	User
FR-03	Create events	Registered accounts will be able to create events	User
FR-04	View created events	Registered accounts will be able to view created events.	User
FR-05	Manage created events	Registered accounts will be able to manage created events.	User
FR-06	Cancel created events	Registered accounts will be able to cancel their created events.	User

FR-07	Search event availability	Users can browse events by event type, club society and date.	User
FR-08	View available events	Users will be shown the available event details and information	User
FR-09	Register event	User should be able to register an event.	User
FR-10	Manage user profile	Users can manage profiles with personalized details.	User
FR-11	View club details	Users can view club details.	User
FR-12	Subscribe interested club	Users will be able to subscribe to interested club.	User
FR-13	Receive notification	Users will receive notifications for club event updates.	User

Use Case ID: UC-001	Create an account
Brief Description	Users register for an account using their university email addresses, either as a participants or organisers.
Primary Actors	Participants, Organisers
Preconditions: <ol style="list-style-type: none"> 1. Users require an existing Sunway University Email to register. 2. The Email is not registered and does not exist as an account. 3. Users should not be logged in. 	
Actor Action	System Response
1. The user enters their personal details including their Sunway University Email.	2. Verify if the user already has an account.
	3. If an account doesn't exist, make one using the user's information. (An alternate flow is shown in 3(a))
	4. Inform user that account has been successfully created
Alternative Flows: 3(a) In the event that a user currently has an account, then no new accounts can be established.	

Post-Conditions:
1. A new user account is created

Use Case ID: UC-002	Log In
Brief Description	Users like participants and organisers are allowed to login into their account.
Primary Actors	Participants, Organisers
Preconditions: 1. At the moment the user is not logged in. 2. There needs to be a user account.	
Actor Action	System Response
1. User clicks the "Login" button after entering their email address and password.	2. Verify that the user's email address and password correspond to an account in the database.
	3. Once the user's email and password are verified, they are logged in and taken to the system website's home page. (For alternative flow see 3(a))
Alternative Flows: 3(a) If the user's email address and password do not match an account in the database, a new prompt will ask them to do so.	
Post-Conditions: 1. The user can view the account and login.	

Use Case ID: UC-003	Create events
Brief Description	Registered accounts will be able to create events
Primary Actors	Participants, Organizers
Preconditions: <ol style="list-style-type: none"> 1. An active user account is required. 2. User already logged into an account 	
Actor Action	System Response
1. User login to their account	2. Display home page
3. Select create event button	4. Show dropdown box for club selection
5. Select representative club	6. Display create event page
7. Input the event's details including title, date, venue, time, number of participants, image and description.	
8. To successfully alter event details, click the "Confirm" button. (An alternate flow is shown in 8(a))	
Alternative Flows: 8(a) The website will revert back to the system response page if the user selects the "Cancel" button.	
Post-Conditions: Event is created and posted on the event page.	

Use Case ID: UC-004	View created events
Brief Description	Registered accounts will be able to view created events.
Primary Actors	Participants, Organizers
Preconditions: <ol style="list-style-type: none"> 1. An active user account is required. 2. User already logged into an account 3. User is part of the club committee. 	
Actor Action	System Response
1. User login to their account	2. Display home page
3. Select “calendar” button	4. Display event page
5. Select created event	6. Display created events
Alternative Flows:	
Post-Conditions: Event is created and posted on the event page	

Use Case ID: UC-005	Managed created events
Brief Description	Registered accounts can create and manage events.
Primary Actors	Participants, Organizers
Preconditions: <ol style="list-style-type: none"> 1. An active user account is required. 2. User already logged into an account. 3. User is part of the club committee. 	
Actor Action	System Response
1. User login to their account	2. Display home page
3. Select “calendar” button	4. Display event page
5. Select created event	6. Display created events
7. Select “pen” button	8. Display the detailed view of the create events
9. Edit the events and select save button. (An alternate flow is shown in 9(a))	10. The changes are updated and stored in the database.
Alternative Flows: 9(a) The website will revert back to the system response page if the user selects the "Cancel" button .	
Post-Conditions: Event created and posted on the event page	

Use Case ID: UC-006	Cancel created events
Brief Description	Registered accounts will be able to cancel their created events.
Primary Actors	Participants, Organizers
Preconditions: An active user account is required. User already logged into an account	
Actor Action	System Response
1. User login to their account	2. Display home page
3. Select “calendar” button	4. Display event page
5. Select created event	6. Display created events
7. Select “bin” button	8. The changes are updated and stored in the database.
Alternative Flows:	
Post-Conditions: Event created and posted on the event page	

Use Case ID: UC-007	Search event availability
Brief Description	Users can browse events by event type, club society and date.
Primary Actors	Participants, Organizers
Preconditions: 1. An active user account is required. 2. User already logged into an account	
Actor Action	System Response
1. User login to their account	2. Display home page
3. User select the event type or club or date.	4. Display all available events details of the user selected event type or club or date
Alternative Flows:	
Post-Conditions: 1. Searched events show	

Use Case ID: UC-008	View available events
Brief Description	Users will be shown the available event details and information
Primary Actors	Participant and Organizer
Preconditions: <ol style="list-style-type: none"> 1. An active user account is required. 2. User already logged into an account. 3. Available event details and information must be frequently updated 	
Actor Action	System Response
1. User login to their account	2. Display home page
3. The events displayed on the home page will be selected by users.	4. The system will switch to the event page for the user to view events.
5. User can also search the upcoming events available	6. The system will switch to the chosen events and display the selected information (See 6(a) for alternative flow).
Alternative Flows: 6(a) If there's no event available for what the user searched, system will prompt an error and user will need to search for another event.	
Post-Conditions: 1. System will display the available events and information to the participants.	

Use Case ID: UC-009	Register events
Brief Description	User should be able to register an event.
Primary Actors	Participant and Organizer
Preconditions: <ol style="list-style-type: none"> 1. An active user account is required. 2. User already logged into an account. 3. Available event details and information must be frequently updated 	
Actor Action	System Response
1. User login to their account	2. Display home page
3. The events displayed on the home page will be selected by users.	4. The system will switch to the "Event" page for the user to view events.
5. User can select the "register now" button.	6. The system display a form to user.

7. User need to fill up the form and select register.	8. The registration is placed and is stored in the database.
Alternative Flows:	
Post-Conditions: 1. System will display the joined events if user selects “calendar” button in the navigation bar.	

Use Case ID: UC-010	Manage user profile
Brief Description	Users can manage profiles with personalized details.
Primary Actors	Participant and Organizer
Preconditions: 1. An active user account is required 2. User already logged into an account	
Actor Action	System Response
1. User login to their account	2. Display home page
3. The user selects the "Profile" icon on the page.	4. User is moved by the system to the profile webpage.
5. User able to edit their own profile details and select the save button. (See 5(a) for alternative flow).	6. The change is updated and is stored in the database.
Alternative Flows: 5(a) If there's no changes made, the profile details will remain the same.	
Post-Conditions: 1. The user can access all of their user information.	

Use Case ID: UC-011	View club details
Brief Description	Users can view club details.
Primary Actors	Participant and Organizer
Preconditions: 1. An active user account is required 2. User already logged into an account	
Actor Action	System Response
1. User login to their account	2. Display home page

3. The user selects the “load more” button.	4. Display all the clubs that are available.
Alternative Flows:	
Post-Conditions: 1. The user can view all of list of clubs available.	

Use Case ID: UC-012	Subscribe interested club
Brief Description	Users will be able to subscribe to interested club.
Primary Actors	Participant and Organizer
Preconditions: 1. An active user account is required 2. User already logged into an account	
Actor Action	System Response
1. User login to their account	2. Display home page
3. The user selects the club.	4. Display the club profile
5. The user selects the “subscribe” button.	6. The interested club is stored in the user’s database.
Alternative Flows:	
Post-Conditions: 1. The user can view the interested club in the subscribe list and will receive notifications.	

Use Case ID: UC-013	Receive notifications
Brief Description	Users will receive notifications for club event updates.
Primary Actors	Users
Preconditions: 1. An active user account is required 2. User must select their club of interest	
Actor Action	System Response
	1. The system notifies the user with updates about their clubs of interest through email. (See 1(a) for alternative flow).

Alternative Flows:

1(a) If a user does not select their club of interest they will not be notified.

Post-Conditions:

Users are able to receive notifications and reminders about their preferred events.

2.2 Quality Attribute

ID	Name	Description	Roles
01	Performance	Users using the system to register event during normal operation and the system should process the user's registration with a latency average of three seconds.	User
02	Security	An unauthorized user with an invalid email address tries to login the website during operation. The system should not allow user from accessing the website within 2 minutes of detection.	User
03	Usability	A regular user with no technical background visits the event management website during runtime and is able to use each function easily within 2 minutes of usage.	User
04		An organizer would like to make changes to the event details during normal operation. The organizer can make changes to the event within a minute with no issues.	User
05	Modifiability	During the system evolution phase, the developer wishes to add a notification feature to the system. Within a day, the feature is added to the system and tested.	Developer, Architect
06	Testability	The system tester awaits the completion of the error validation code during development and proceeds to perform a test by entering invalid inputs. The results capture those errors with 100% rate of detection within 10 seconds.	Tester

Performance	
Users using the system to register event during normal operation and the system should process the user's registration with a latency average of three seconds.	
Portion of Scenario	Possible Value
Source of Stimulus	Users
Stimulus	Users using the system to register event during normal operation
Artifact	System
Environment	Fully operational
Response	The system should process the user's registration
Response Measure	Average latency of three seconds

Security	
An unauthorized user with an invalid email address tries to login the website during operation. The system should not allow user from accessing the website within 1 minutes of detection.	
Portion of Scenario	Possible Value
Source of Stimulus	An unauthorized user with an invalid email address
Stimulus	Tries to login the website
Artifact	The website
Environment	Fully operational
Response	Should not allow user from accessing the website
Response Measure	Unauthorized user stays logged out within 1 minutes of detection

Usability	
A regular user with no technical background visits the event management website during runtime and is able to use each function easily within 2 minutes of usage.	
Portion of Scenario	Possible Value
Source of Stimulus	Regular user with no technical background

Stimulus	Visits event management website
Artifact	Event management website
Environment	During runtime
Response	All functions on the website are easily used when it is accessed
Response Measure	Within 2 minutes of usage

Usability	
An organizer would like to make changes to the event details during normal operation. The organizer can make changes to the event within a minute with no issues.	
Portion of Scenario	Possible Value
Source of Stimulus	Organizer
Stimulus	Would like to make changes to the event details
Artifact	Event details
Environment	Normal operation
Response	Organizer can make changes to the event with no issue
Response Measure	Within a minute of usage

Modifiability	
During the system evolution phase, the developer wishes to add a notification feature to the system. Within a day, the feature is added to the system and tested.	
Portion of Scenario	Possible Value
Source of Stimulus	Developer
Stimulus	Wishes to add a notification feature to the system
Artifact	Website source code
Environment	System evolution
Response	The notification feature is added into the system and tested
Response Measure	Feature is added within a day

Testability

The system tester awaits the completion of the error validation code during development and proceeds to perform a test by entering invalid inputs. The results capture those errors with 100% rate of detection within 10 seconds.	
Portion of Scenario	Possible Value
Source of Stimulus	System Tester
Stimulus	The completion of error validation code during development
Artifact	Unit of Error Validation Code
Environment	Development time
Response	Perform a test by entering invalid inputs to capture the errors
Response Measure	100% rate of error detection within 10 seconds

3.0 System and Software Design

3.1 System Architecture

Based on the requirements of the system, the most suitable architecture is MVC (Model-View-Controller). The MVC architecture aligns seamlessly with the need for a clear separation of concerns in developing a student event management system. The clear separation of concerns—where the model manages data, the view handles the user interface, and the controller orchestrates user interactions and business logic—enhances maintainability and scalability. This separation allows developers to make changes in one component without affecting the others, promoting code reusability and ease of updates. Additionally, MVC facilitates a more organized development process, making it easier for teams to collaborate efficiently and ensuring a sustainable foundation for future enhancements.

3.1.1 Model-View-Controller

In the context of a website, the MVC architecture translates into a systematic division of tasks. The "Model" corresponds to the data storage layer, managing the storage and retrieval of information such as club details, event schedules, and user profiles. The "View" encapsulates the presentation layer, where the visual elements and user interface are crafted to deliver an engaging and seamless experience. Finally, the "Controller" operates in the application layer, handling user input, implementing business logic, and serving as the bridge between the model and view. This division of responsibilities not only brings clarity to the development process but also ensures that the website is modular, scalable, and adaptable to changing requirements.

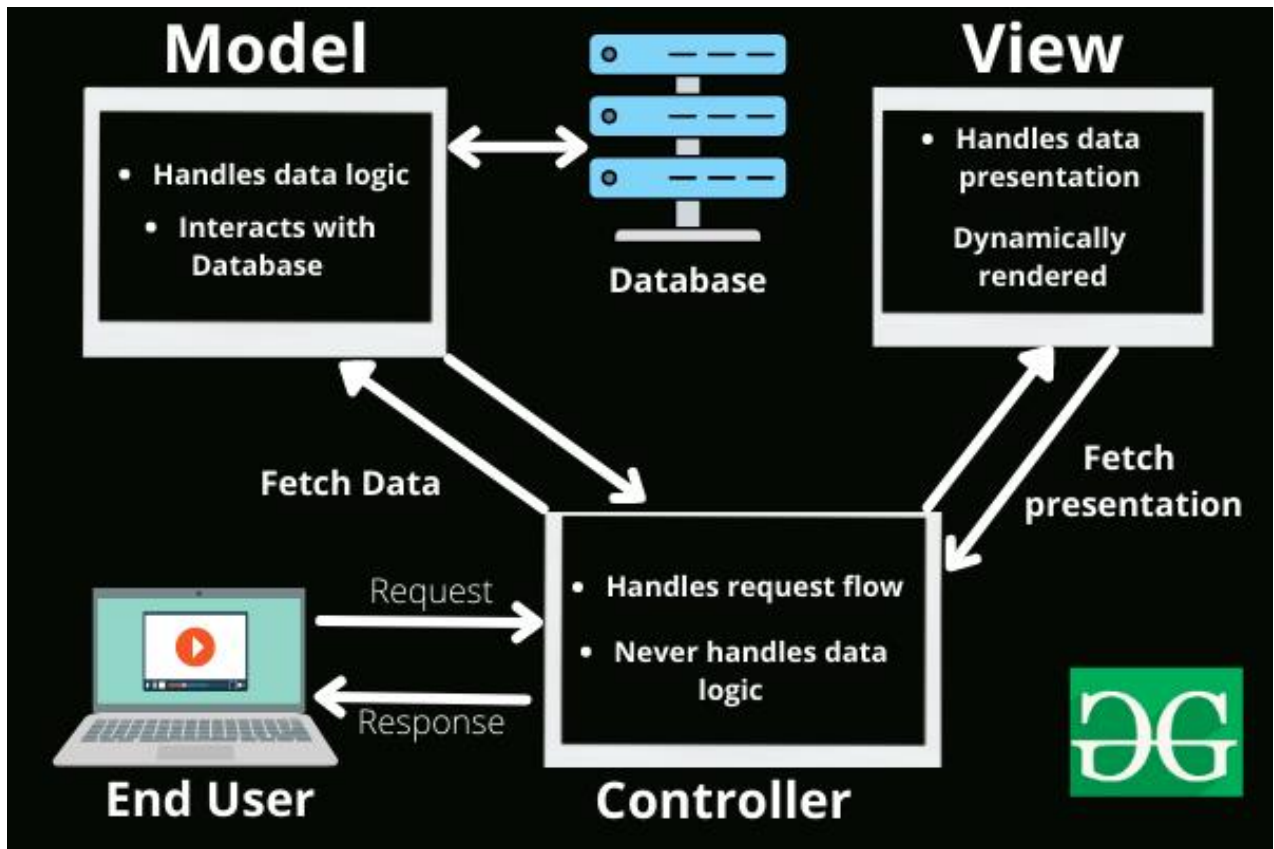


Figure 3.1: Model-View-Controller

Component

Component	Description
Model	Implemented through the data storage layer, efficiently managing the storage and retrieval of data about clubs, events, and user profiles.
View	The presentation layer, where the user interface and experience are crafted to facilitate intuitive interactions and engagement.
Controller	In the application layer, responsible for handling user input, business logic, and orchestrating the flow of data between the model and view.

Rationale

The rationale behind choosing MVC for a student event management system lies in its ability to address the specific needs of the project. This architecture not only supports the modular development and maintainability required for a dynamic web application but also ensures that future modifications or enhancements can be seamlessly integrated into the system. The MVC

architecture, with its clear organization and well-defined roles, provides a solid foundation for building a robust and scalable student event management platform.

3.1.2 Client-Server

Client-server architecture is a widely used computing model that divides a system into two distinct roles: the client and the server. The clients, in this case, are the students and event organizers who access the website. They use their devices, such as laptops or smartphones, as clients to connect to the server hosting the web application, whereas the server acts as the central hub of the event booking system as it hosts the website, databases, and application logic. Event organizers and participants connect to this server to access the services and features provided by the system.

When the event participants visit the website, they act as clients and send requests to the server. These requests may include searching for upcoming events, viewing event details, or registering for events. The server processes these requests and sends back responses with the requested information, event details, or confirmation of registration.

The web application caters to two primary user roles that can be the same account - Event Organizer and Event Participants. Event Organizers use the platform to publish their events and collect data on participants. The server stores this event information and manages the data, making it accessible to both event holders and participants. Event Participants can browse upcoming events and register for the ones they're interested in through the website. With that being said, the two roles are not mutually exclusive, as a user can be both an Event Organizer and Event Participant at the same time. The server would allow for efficient resource sharing to store and manages event data, including event details, participant registrations, and other related information. This resource sharing facilitates collaboration between event organizers and participants.

3.1.3 Publish-Subscribe

A publish-subscribe architecture, where components communicate by publishing events without knowing what subscribers there are, would be significant for an event management website because it allows for loose coupling between the different components. For example, there could be one component focused on providing interfaces for event organizers to publicize

their events, while separate components could subscribe to those events and handle displaying them to end users in different ways - such as in a news feed, personalized recommendations, search results etc.

The publishing component does not need to know how the subscribing components will handle or display the events. As long as they adhere to a standard event format, new subscriber components can be added to create new pages or features. For instance, a new "featured events" component can subscribe to events fitting certain criteria without the publisher being changed. Similarly, new publishing interfaces for partners or mobile apps can be built without affecting existing display and subscription components.

This loose coupling and simplicity of adding new features is crucial for a large event management platform with diverse sources of events and many potential ways to consume them. Publishers of events only care about exposing event data, while subscribers can focus solely on using that data, not coordinating with publishers. Additional cross-cutting concerns like analytics, rate-limiting etc could also be built around the event streams in a non-invasive way. By simplifying dependencies and communication flows, a publish-subscribe architecture enables scalability and velocity of feature development.

3.2 Use Case Diagram

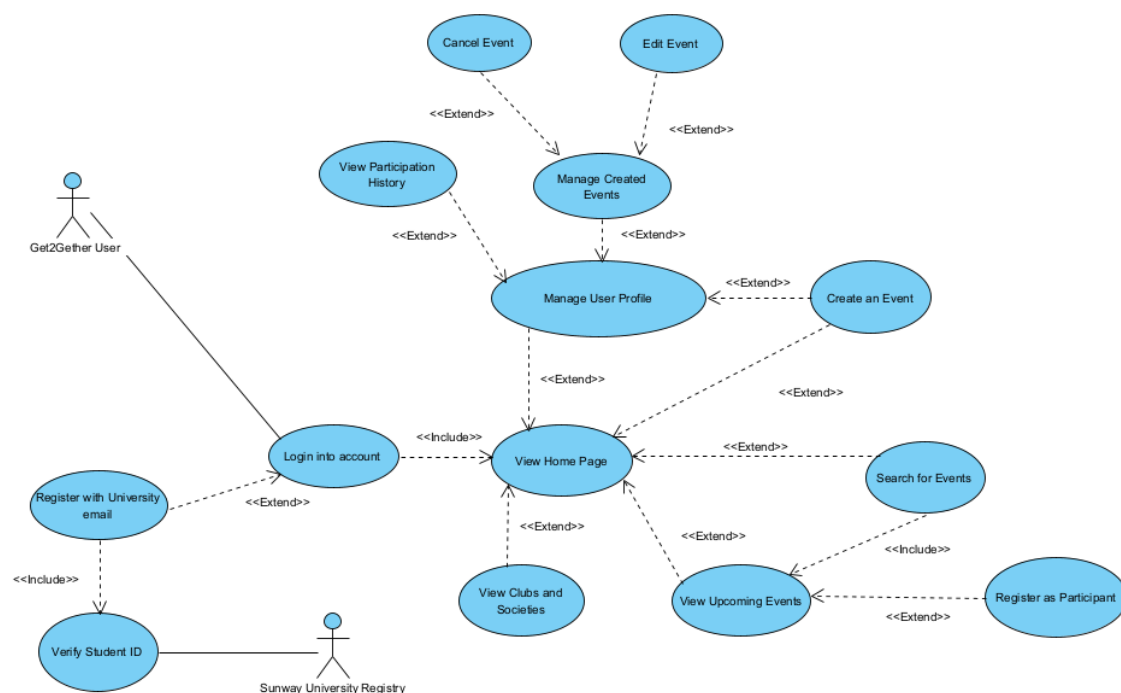


Figure 3.2: Use Case Diagram

3.3 Sequence Diagram (Main)

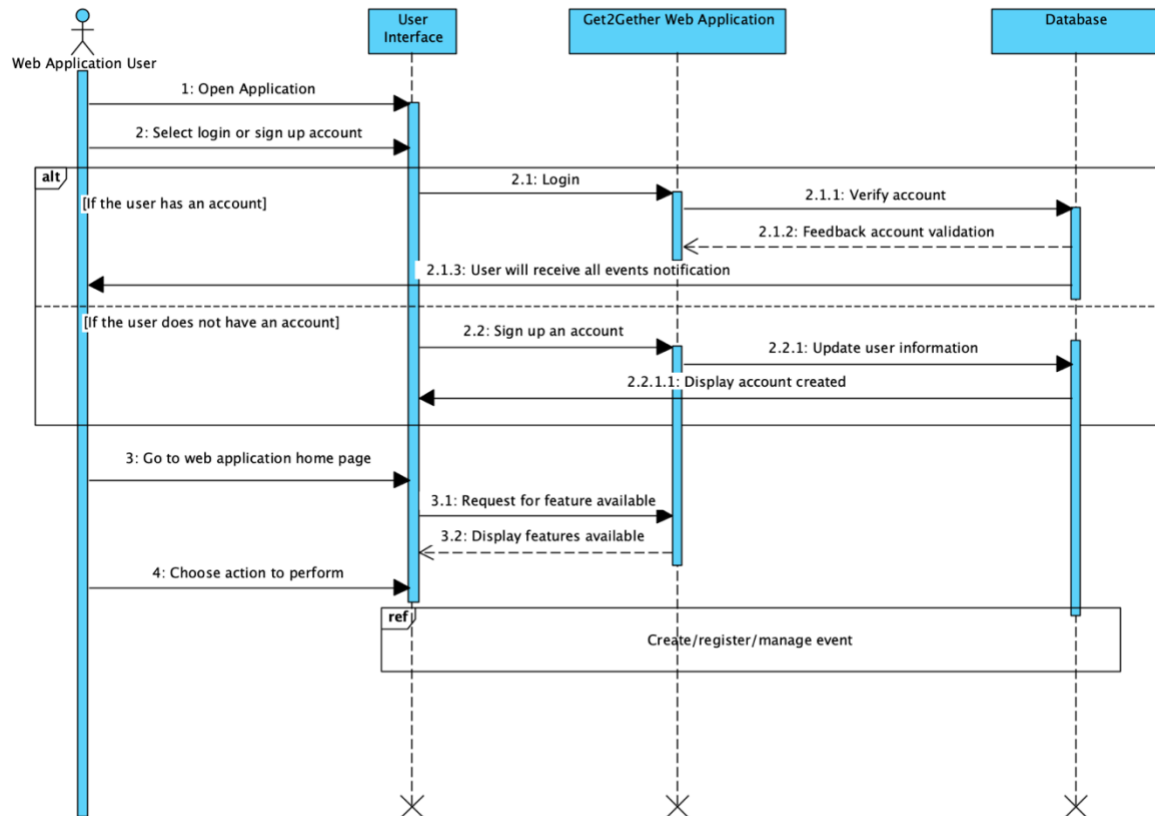


Figure 3.3: Main Sequence Diagram

The sequence diagram above shows an overall view of the web application. As a user of the web application, when accessing the Get2Gether website, the user has the option to either log in or sign up. If the user already has an account, they can log in, and the web application will verify their credentials with the database. Upon successful verification, the user can access the site and receive notifications about available events. If the user doesn't have an account, they can sign up by providing necessary details and inputting the OTP verification code. Once completed, the user gains access to the website. Within the platform, users can create, manage, and register for events.

3.3.1 Sequence Diagram (Sub-Diagram: Create/register/manage event)

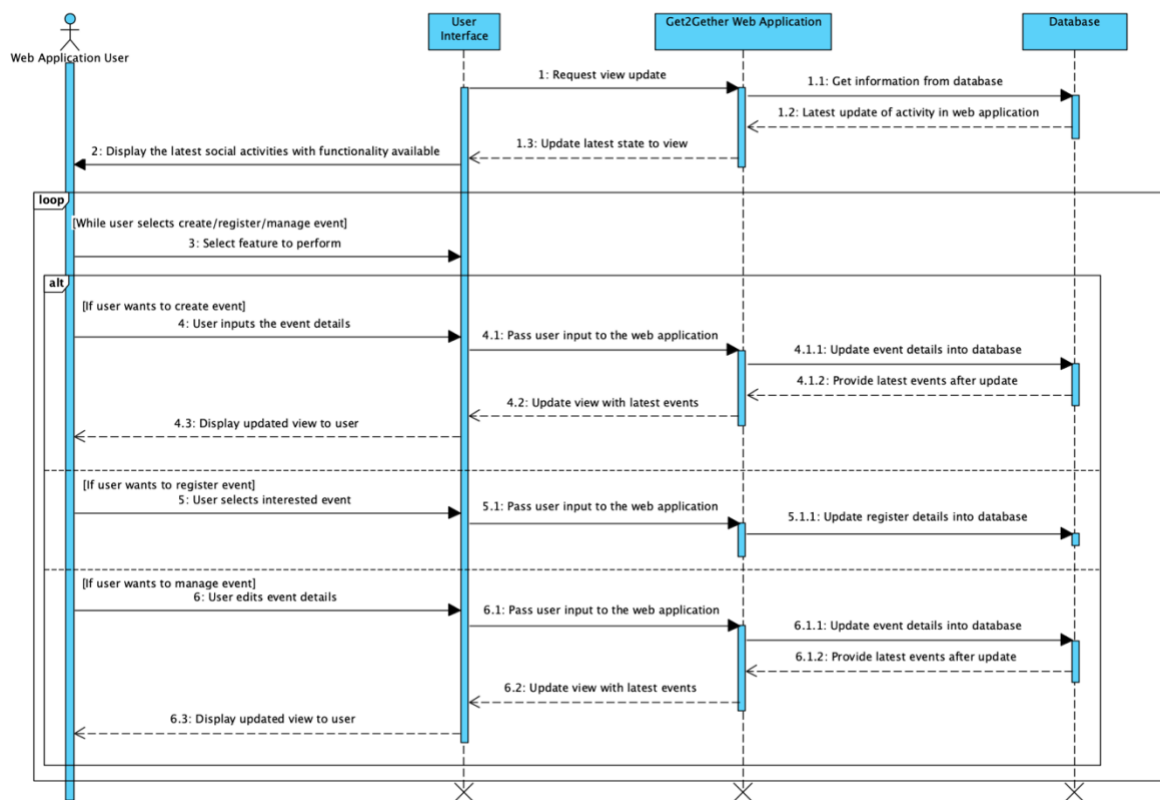


Figure 3.4: Sub Sequence Diagram

The sub-sequence diagram above shows part of the view of the web application. After gaining access to the website, users can create, manage, and register for events. To create an event, users simply need to input the event details, which will be updated in the database. Once a new event is created, it will be displayed to users. For event registration, users only need to select the desired event and complete the registration process, which updates the information in the database. If the user is a committee member or the Point of Contact (PIC), they have the ability to edit event details by updating the information in the database. The updated event details can then be viewed by users.

3.4 Entity Relationship Diagram

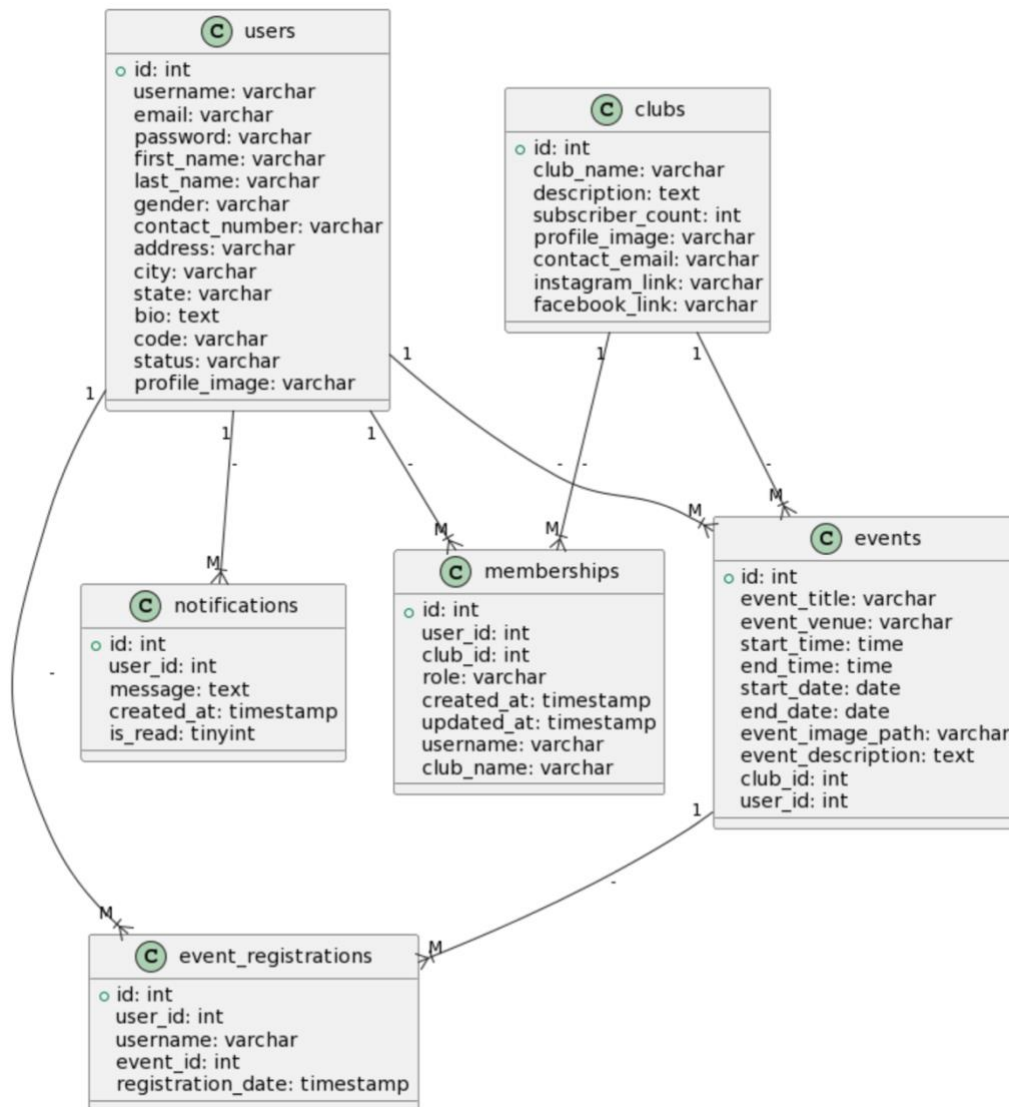


Figure 3.5: ERD

Figure 3.5 shows the Entity Relationship Diagram (ERD) illustrates the relational structure of the "get2gether" database, representing key entities and their associations. The diagram encompasses several tables, including "users," "clubs," "events," "memberships," "event_registrations," and "notifications." Each table is defined with its respective attributes, such as user details, club information, event specifics, membership roles, event registrations, and notifications. Relationships are visually presented through connecting lines, showcasing the associations between entities. The connections include the user's involvement in memberships, participation in events, and receipt of notifications. Clubs are linked to

memberships and events, indicating their organizational roles, while events are tied to both clubs and user participation. The diagram provides a comprehensive overview of the database structure, emphasizing the interconnections that facilitate data retrieval and management within the "get2gether" system.

3.5 User Interface Design

User interface (UI) design is probable the initial element that users encounter when they navigate to a website or utilise an application. The responsibility for the appearance, interactivity, usability, behaviour, and overall feel of a product lies with user interface design. Because UI design can determine whether a user has a positive experience with a product, developers must become well-versed in UI design best practices. Mastering the four c's is a simple method to recall the fundamental principles of user interface design:

1. **Control:** The interface ought to be under the control of the consumers.
2. **Consistency:** Implementing standardised components into your user interface will render it foreseeable and effortless to navigate, even for inexperienced individuals.
3. **Comfortability:** A product should facilitate effortless, comfortable interaction.
4. **Cognitive load:** Caution must be exercised when it comes to inundating users with content. Be as succinct and transparent as feasible.

Prototyping and user interface design are closely connected elements of the product development process. UI design involves refining the aesthetic and visual aspects of a user interface and creating static design assets like wireframes and prototypes. On the other hand, prototyping transforms these design concepts into interactive representations of the interface, facilitating effective communication, usability testing, and iterative design. Prototyping enhances the user experience by enabling designers to test functionality and gather feedback, ensuring the final product aligns with user expectations. In contrast, UI design primarily focuses on aesthetics.

Before starting the prototype development process, it's crucial to establish a clear definition of the project's scope and objectives. Our project aims to create a website with a comprehensive database of student organizations, their activities, and relevant details in a user-friendly manner, targeting Sunway's students as the main audience. The specific functionalities and features we plan to integrate into the prototype were previously outlined in the introductory section of the project documentation. By clearly defining the project's scope, we can identify the essential functionality needed for the prototype. This ensures the development team focuses on fixing the most crucial elements of the design.

The connection between typography and prototype stems from the fact that typography is a fundamental aspect of prototyping in design and user interface development, where the selection of fonts, typefaces, and text layout has a significant impact on the user experience as well as the overall appearance and functionality of the prototype.

Typography is the art and method of arranging and designing text in order to make it visually appealing, readable, and successful in communication. It includes the choosing of typefaces, fonts, spacing, and other design components. Below show the six primary elements or components implemented of typography implemented in Get2Gether system.

1. **Typeface:** Open Sans, Poppins, Roboto
2. **Font:** Open Sans Normal (64px), Open Sans Normal (36px), Open Sans Normal (24px), Open Sans Normal (18px), Open Sans Normal (16px), Open Sans Normal (12px), Poppins Normal (30px), Poppins Normal (17px), Poppins Normal (16px)
3. **Hierarchy:**

	Login, Registration, Email-Verification, Forgot Password Page	Event, Club Profile, User Profile Page	Home Page
Main Heading	Poppins Normal (30px)	Open Sans Normal (64px)	Open Sans Normal (36px)
Subheading	Poppins Normal (17px)	Open Sans Normal (24px)	Open Sans Normal (16px)
Body Text	Poppins Normal (16px)	Open Sans Normal (16px)	Open Sans Normal (12px)

4. **Alignment:** Left Align where all text is aligned to the left margin
5. **Style:** Normal
6. **Colour:** The four main colours utilised mainly are Cloud Burst(#1D294F), Di Serria(#DDAB58), Neptune(#79A6B4), Curious Blue(#1D86C5)

Link to Prototype:

<https://www.figma.com/file/sLY6YM727HRXeTf9rkluo/Prototype?type=design&node-id=29%3A245&mode=design&t=4SUJnneeRUymhBYx-1>

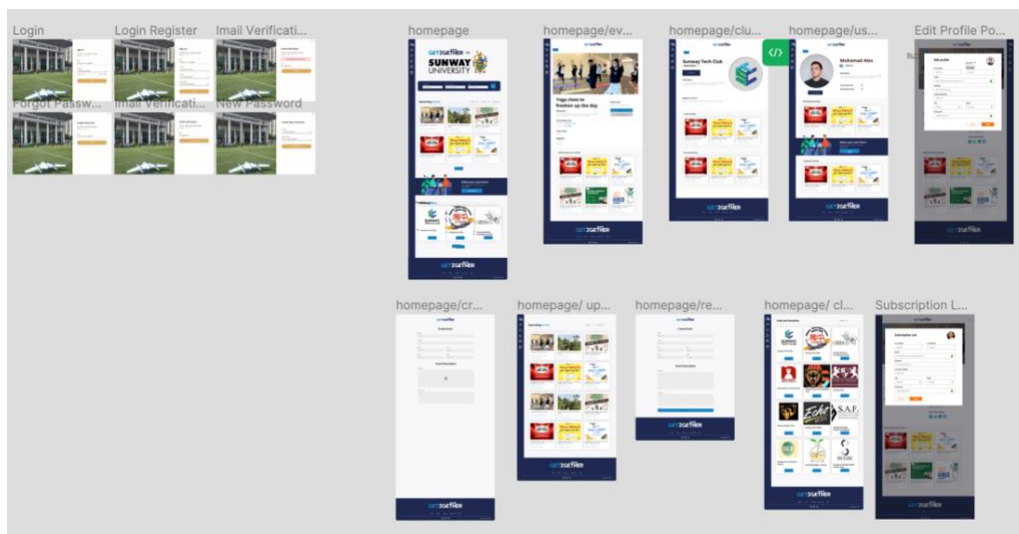


Figure 3.6 Prototype Overview

Additionally, the prototype design follows Shneiderman's Eight Golden Rules, which are also the Eight Golden Rules of Interface Design. To start, consistency is implemented, as shown in Figure 3.5, through the use of familiar colors, icons, menu hierarchy, call-to-actions, and user flows when designing similar scenarios and sequences of actions. Standardizing how information is communicated allows users to apply their knowledge across different clicks, eliminating the need to memorize new representations for the same actions. Subsequently, as depicted in Figure 3.5, the event management website features a navigation bar on the left side of each page after user login. This allows users to quickly and easily navigate to any desired page, satisfying the requirement to support an internal locus of control.

Moreover, the website shown in Figure 3.6 adheres to the rule of offering simple error handling by providing straightforward, intuitive, and step-by-step instructions for resolving issues as swiftly and painlessly as possible in case of an error. For example, if the user enters an incorrect verification code, an error message will appear, informing the user that the entered code is incorrect. Finally, as illustrated in Figure 3.7, a design dialogue to yield closure is identified. Once the event has been successfully created, the user will be notified via email containing pertinent details about the event and a message confirming its creation.



Figure 3.7: Get2Gether Home Page

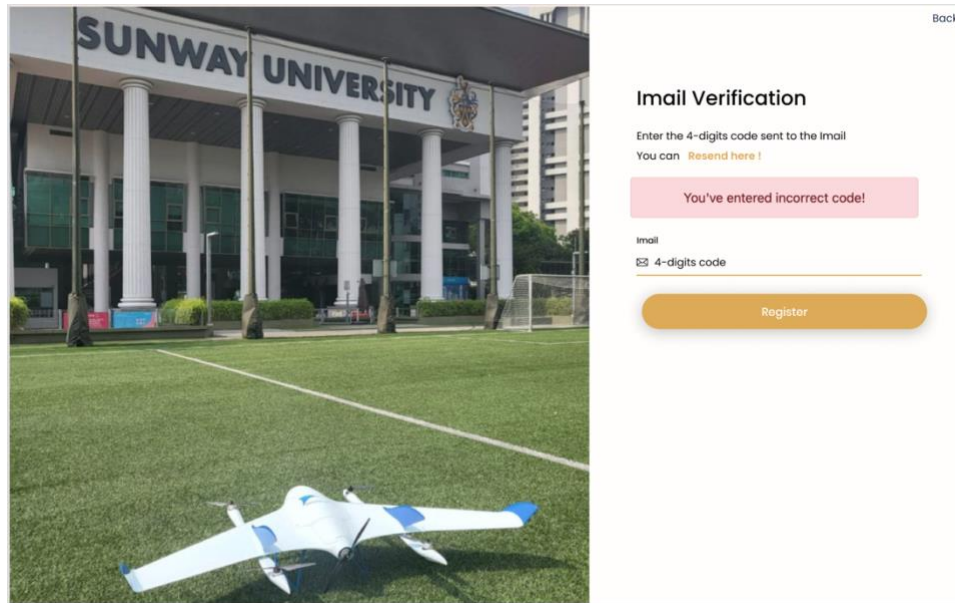


Figure 3.8: Verification error message

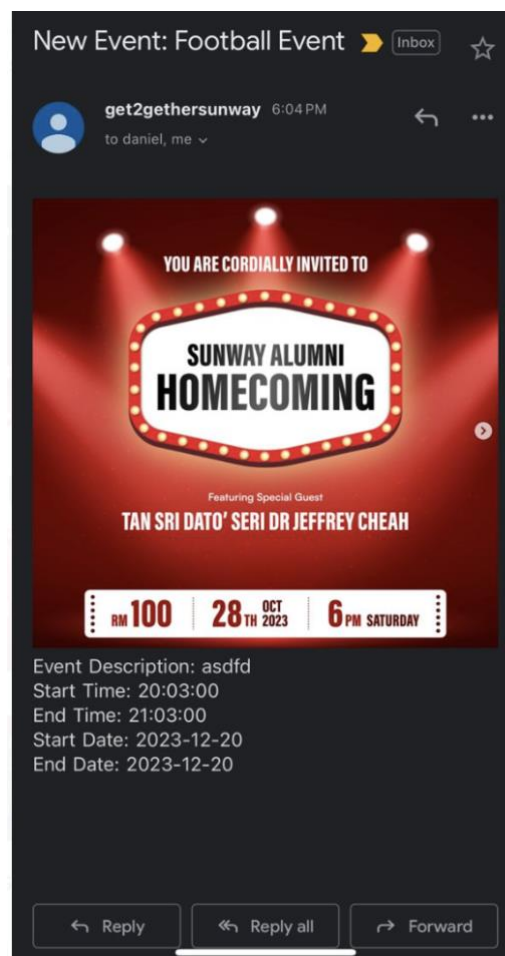


Figure 3.9: Event Notification

4.0 Project Planning, Management and Implementation

The implementation phase of the projects marks a critical stage where the conceptualized ideas and plans are transformed into a functional website. The section provides overall project planning, management, detailed strategies, technologies and methodology employed during the implementation of the Get2Gether's website.

4.1 Project Planning

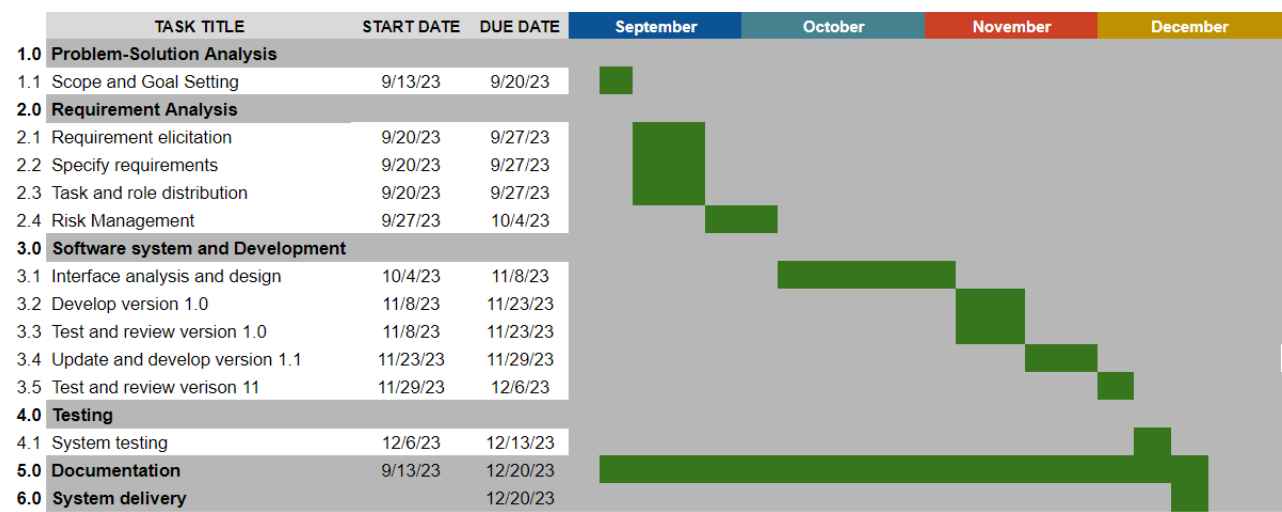


Figure 4.1: Gantt Chart for the entire Get2Gether System

4.2 Project Management

The Agile development methodology was embraced to foster adaptability and collaboration throughout the development lifecycle. The project was divided into sprints, with regular sprint reviews and retrospectives to address evolving requirements and ensure continuous improvement. The initiation phase involved close collaboration with all the group members to outline initial requirements, and a dynamic project scope was established. The use of user stories and a prioritized backlog facilitated a focus on delivering valuable features in each sprint, with time-boxed iterations ranging from 2 to 4 weeks. The designers and developers worked collaboratively during sprint planning sessions, emphasizing continuous attention to technical excellence and adapting to changing circumstances as the project progressed. The team had regular sprint reviews allowing the team to discuss and provide feedback on implemented features, aligning the delivered software with expectations. Moreover, user acceptance testing sessions were conducted regularly to ensure the software met the needs of

both event holders and participants. The release and iteration process followed Agile principles of delivering frequent increments and adjusting the release plan based on user's feedback, while ongoing adaptation allowed the team to respond to the evolving changes. Through the application of Agile methodologies, the project aimed to create a collaborative, adaptive, and customer-focused development process, ultimately resulting in the successful delivery of the event management website

4.3 Technology Stack Selection

In Section 4.3, the technology stack selection is outlined to provide a comprehensive understanding of the tools and languages employed in the development process. Table 4.1 shows breakdown of the selected technologies.

Tools/Language	Description
HTML5 and CSS3	The user interface was crafted using HTML and CSS , to ensure a responsive and visually engaging design. The frontend development process was streamlined with the use of PHP for server-side scripting.
PHP	PHP served as the backbone for implementing the business logic and dynamic functionalities of the website. The Apache web server was employed to host and serve PHP scripts, to ensure efficient communication between the server and the client.
PHPmyAdmin	PHPmyAdmin , in conjunction with SQL , was utilized for the database management system. This combination facilitated the organization and the retrieval of data related to student clubs, events, and user profile.
XAMPPS	XAMPPS , the Apache Web Server was utilized to host and serve PHP Script, to

	ensure efficient communication between the client and the server.
GitHub & Git	GitHub was employed for version control, enabling collaborative development, code tracking, and seamless integration of new features. This choice facilitated team collaboration and version history tracking.
Visual Studio Code (VSCode)	Visual Studio Code (VSCode) was used as the integrated development environment (IDE) of this project.
Figma	Figma played a crucial role in the prototyping phase, allowing for the visualization of the user interface and user experience. Figma's collaborative features facilitated feedback and adjustments during the design process.

Table 4.1: Technology Stacks

4.4 Implementation

GitHub Link: <https://github.com/karen-nke/get2gether-student-event-management>

4.4.1 Navigation Bar

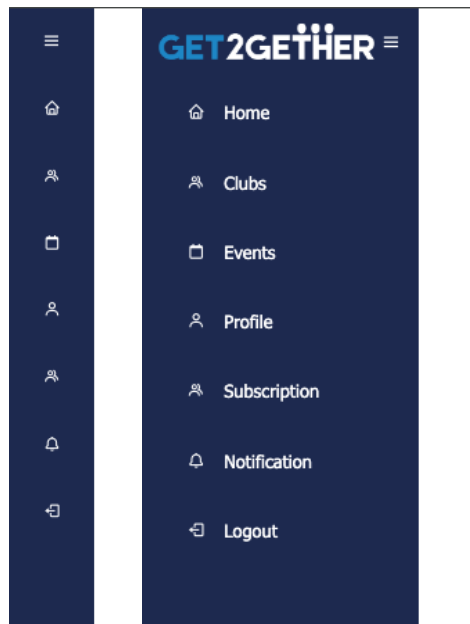


Figure 4.2: Navigation Bar

Figure 4.2 shows the implementation of Navigation Bar before and after extend. After user clicked the burger icon of the menu, the navigation bar will extend and show the Name of each page and the full logo of the system.

```
<?php
if (isset($_SESSION['username'])) {
    // After Logged In
    echo "
    <li>
        <a href='profile.php'>
            <i class='bx bx-user'></i>
            <span class='nav-item'>Profile</span>
        </a>
        <span class='tooltip'>Profile</span>
    </li>

    <li>
        <a href='subscription.php'>
            <i class='bx bx-group'></i>
            <span class='nav-item'>Subscription</span>
        </a>
        <span class='tooltip'>Subscription</span>
    </li>

    <li>
        <a href='notification.php'>
            <i class='bx bx-bell'></i>
            <span class='nav-item'>Notification</span>
        </a>
        <span class='tooltip'>Notification</span>
    </li>
```

Figure 4.3: Code Snippet – Navigation Bar

The Figure 4.3 shows the code snippet to implement the Navigation Bar. Before the user log in into the system, the navigation bar will show only Home, Clubs, Events, Login and Register. After User logged in into the system, the system store the user information in the session. The

Navigation bar will show Home, Clubs, Events, Profile, Subscription, Notification, and Logout.

4.4.2 Home Page

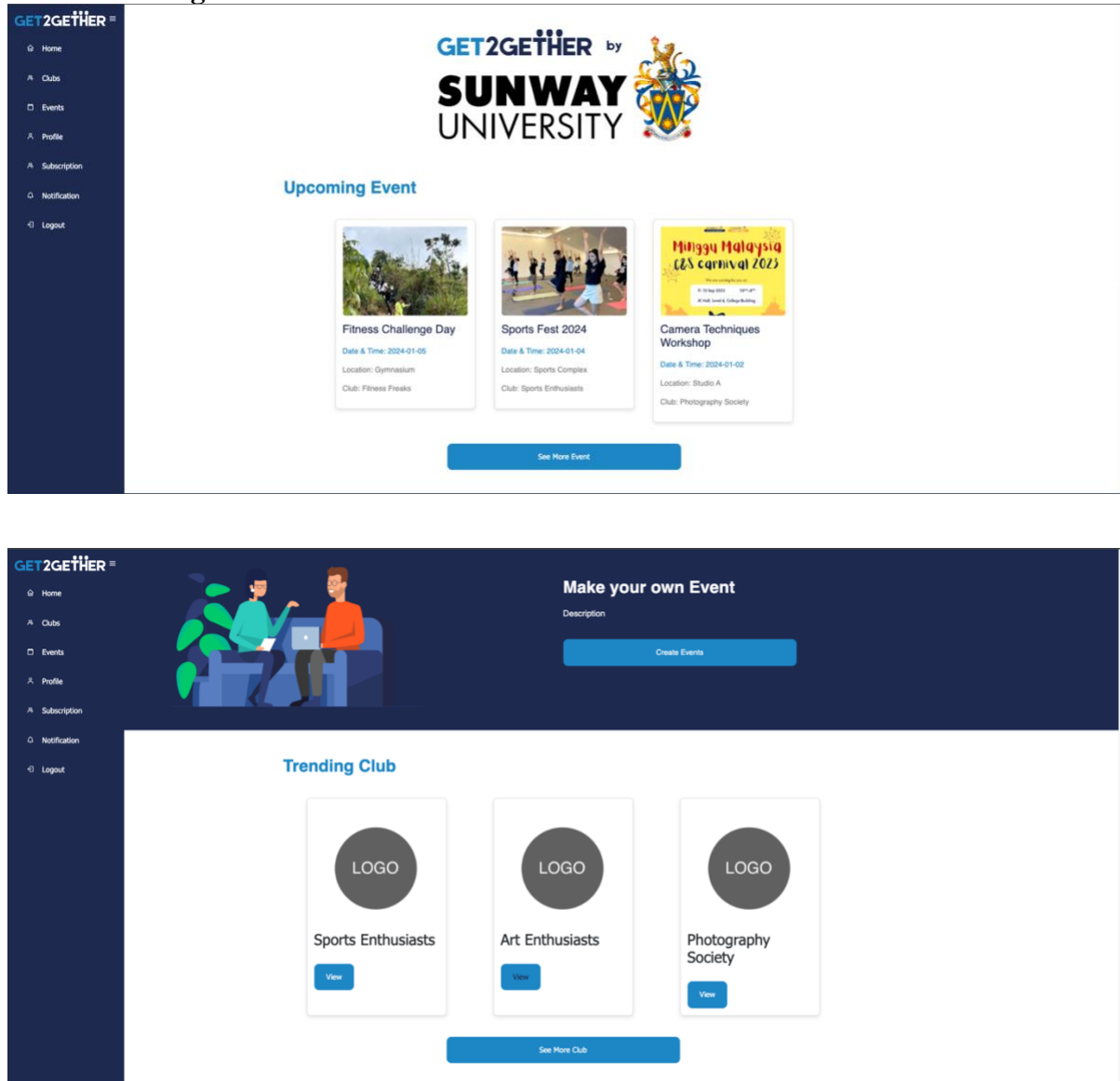


Figure 4.4: Home Page

Figure 4.4 shows the actual implementation of the Home Page. The Homepage contains the Logo of the system, Upcoming Event Section, Trending Clubs Event Section, and a Banner to Create the Event.

```

<div class="page-container">
  <div class="image-container">
    
  </div>

  <?php require_once('Part/event_section.php'); ?>

</div>

<div class="banner-container">
  <div class="image-container">
    
  </div>
  <div class="text-container">
    <p class="title"> Make your own Event</p>
    <p class="desc">Description</p>
    <a href="create_events.php"><button class="btn">Create Events</button></a>
  </div>
</div>

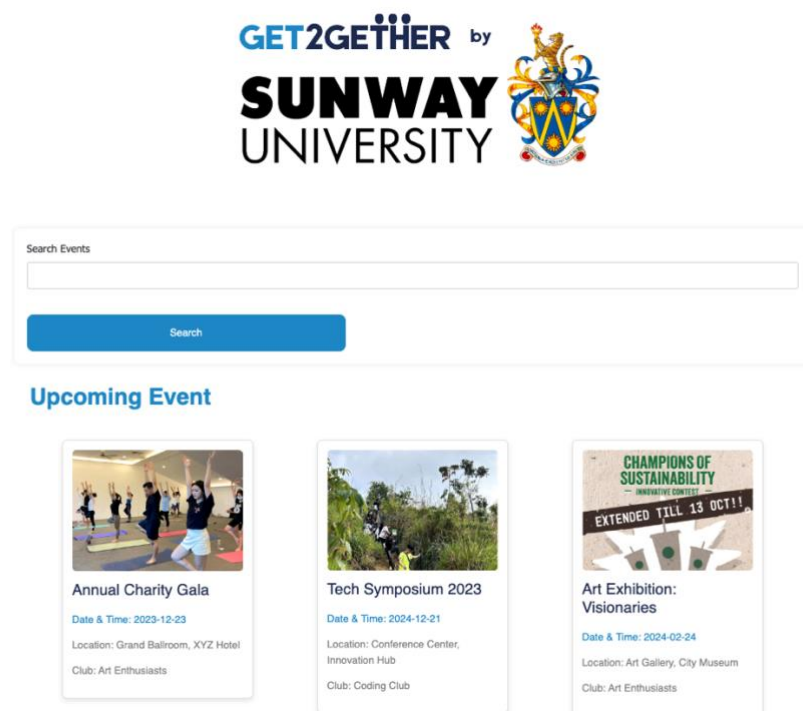
<div class="page-container">
  <?php require_once('Part/club_section.php'); ?>
</div>

```

Figure 4.5: Code Snippet – Reusable Section

Since the Upcoming Event section and Clubs Section will be use across multiple page. To practice reusability, these section were separated into two different file – event_section.php & club_section.php. The file will be call using “require_once(‘file_name.php’) to show the event or the club section which will be show in different other pages.

4.4.3 Event & Club



Past Event



Figure 4.6: Event Pages

Figure 4.6 shows the event page with Search Functionality, all Upcoming Events and all Past Events. The Upcoming event card is clickable while the Past Event card is not clickable. Once user clicked on the event card, user will be lead to the respective event page.

```
<div class="section-container">
  <p class="title">Upcoming Event</p>

  <div class="event-container">
    <div class="event-container">
      <?php foreach ($upcomingEvents as $event): ?>
        <a href="event_single.php?id=<?= $event['id']; ?>">
          <div class="event-card">
            
            <h2 class="title"><?= $event['event_title']; ?></h2>
            <p class="date">Date & Time: <?= $event['start_date']; ?></p>
            <p class="location">Location: <?= $event['event_venue']; ?></p>
            <p class="location">Club: <?= $event['club_name']; ?></p>
          </div>
        </a>
      <?php endforeach; ?>
    </div>
  </div>
</div>
```

Figure 4.7: Code Snippet – Event Card

Figure 4.7 show the code snippet of the Event Card. When user click on the event card, it will lead the user to event_single.php with the details of the selected event by incorporating parameter in the link. The event ID will be parse into the link as parameter, the event_single page will read the ID of the event, and show the correct event details based on the ID.

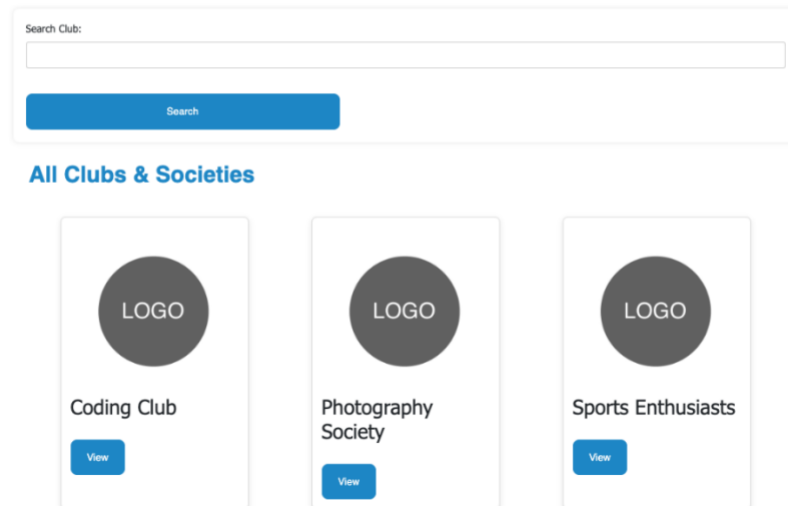


Figure 4.8: Clubs Page

Figure 4.8 shows the Club page of the system where user can view all the available club in the system. The logic implementation of the Club Page is similar to the Event Page.

4.4.4 Search Function

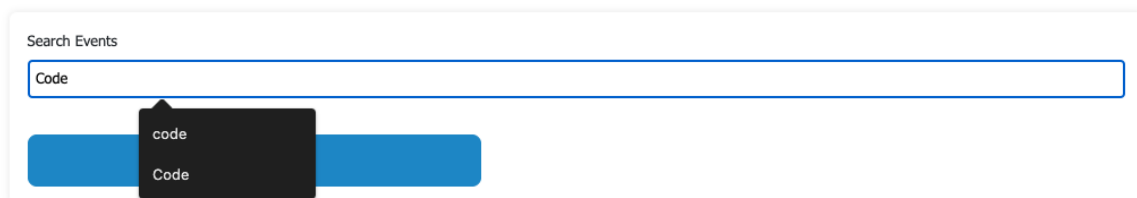


Figure 4.9: Search Function (Event & Club)

Figure 4.9 shows the Search Bar implemented in both Event Page (Figure 4.6) and Club Page (Figure 4.8). The search function allow user to search for the events and clubs based on the keyword.

Search Results



Figure 4.10: Search Result

Figure 4.10 shows the search result using the keyword “Code”. The system will show the event with the searched keyword by the user.

```
// Query the database for clubs or events matching the search term based on the referring page
if (isset($_SERVER['HTTP_REFERER']) && strpos($_SERVER['HTTP_REFERER'], 'clubs.php') !== false) {
    // User came from clubs.php, search for clubs
    $sql = "SELECT * FROM clubs WHERE club_name LIKE '%$searchTerm%' OR description LIKE '%$searchTerm%'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        // Display club search results
        echo '<div class="page-container">';
        echo '<div class="section-container">';
        echo '<div class="image-container">';
        echo '';
        echo '</div>';

        echo '<p class="title">Search Results</p>';
        while ($row = $result->fetch_assoc()) {
            //echo '<div class="event-container">';
            echo '<div class="event-card">';
            echo '';
            echo '<h2>' . $row['club_name'] . '</h2>';
            echo '<p>' . $row['description'] . '</p>';
            echo '<a href="club_single.php?id=' . $row['id'] . '"><button class="btn">View</button></a>';
            //echo '</div>';
            echo '</div>';
        }
        echo '</div>';
        echo '</div>';
    } else {
        echo '<div class="page-container">';
        echo '<h2>No club results found for "' . $searchTerm . '"</h2>';
        echo '</div>';
    }
} elseif (isset($_SERVER['HTTP_REFERER']) && strpos($_SERVER['HTTP_REFERER'], 'events.php') !== false) {
    // User came from events.php, search for events
    $sql = "SELECT * FROM events WHERE event_name LIKE '%$searchTerm%' OR description LIKE '%$searchTerm%'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        // Display event search results
        echo '<div class="page-container">';
        echo '<div class="section-container">';
        echo '<div class="image-container">';
        echo '';
        echo '</div>';

        echo '<p class="title">Search Results</p>';
        while ($row = $result->fetch_assoc()) {
            //echo '<div class="event-container">';
            echo '<div class="event-card">';
            echo '';
            echo '<h2>' . $row['event_name'] . '</h2>';
            echo '<p>' . $row['description'] . '</p>';
            echo '<a href="event_single.php?id=' . $row['id'] . '"><button class="btn">View</button></a>';
            //echo '</div>';
            echo '</div>';
        }
        echo '</div>';
        echo '</div>';
    } else {
        echo '<div class="page-container">';
        echo '<h2>No event results found for "' . $searchTerm . '"</h2>';
        echo '</div>';
    }
}
```

Figure 4.11: Code Snippet – Search Function

Figure 4.11 shows the implementation of the Search Function. The HTTP_REFERER is used to check if the user came from event page or club page, and will show the search result from the event database table or the club database table. %searchTerm% is used to search for the term related to the keyword by the user.

4.4.5 Single Event Page



Figure 4.12: Single Event Page

Figure 4.12 shows the Single Event page to show the event details based on the user selection. The details of the event such as Event Image, Description, Event Date & Time, Event Venue, Organizer will be showed in the page. By clicking on the “View Club Details” button, user will be lead to the organizer club page to view the details of the club. Additionally, user can click on the Register button to register for the event.



Annual Charity Gala

Description

Join us for an elegant evening of fundraising and entertainment.

Event Date & Time

Event Date: 2023-12-23

Event Time: 18:00:00 - 23:00:00

Event Venue

Grand Ballroom, XYZ Hotel

Organizer

Contact Email Address:

Confirm Registration

Figure 4.13: Event Confirmation

Figure 4.13 shows the registration confirmation page. After user clicked on the register button, the user will be prompt to confirm their registration with all the event details. Once user click on the confirm registration, their registration will be recorded in the database.

Organizer

Club Name: Art Enthusiasts

Contact Email Address: art@example.com

View Club Details

Cancel Registration

Figure 4.14: Cancel Registration

Figure 4.14 shows the option for user to Cancel their Registration. Once user registered for the event, the “Cancel Registration” button will be shown instead of “Register” button. User has the option to cancel their registration after they registered for the event.

Organizer

Club Name: Coding Club

Contact Email Address: coding@example.com

View Club Details

Register

Edit Events

Cancel Event

Registered Participants

- DemoAcc2

Boost Event

Remind Participants

Figure 4.15: Event Management (with Permission)

User with the role “PIC” or “Committee” in the club will have the permission to Edit event, Cancel event, View registered participants, Boost Event, Remind Participants. The Edit Event allows the Club to edit and update the information of the event, Cancel Event allow the club to cancel the event and notify the user. User also can Boost the event where a notification will be send to all users in their notification. Ans the Remind participants allow the club to send participation reminder to the registered participants a day before the event.

Edit Event Details

New Event Title

Code Jam Championship

New Event Venue

Coding Arena

New Start Time

03:00 PM

New End Time

05:00 PM

New Start Date

20/12/2023

New End Date

20/12/2023

New Event Description

Compete in a coding challenge and showcase your skills.

Update Details

Back

Figure 4.16: Edit Event

If the user click on the button “Edit Event”, the system will show the form with the current information. User just need to modify the information of the events and click on the button “Update Details”, and the information will be updated in the database.

4.4.6 Notification

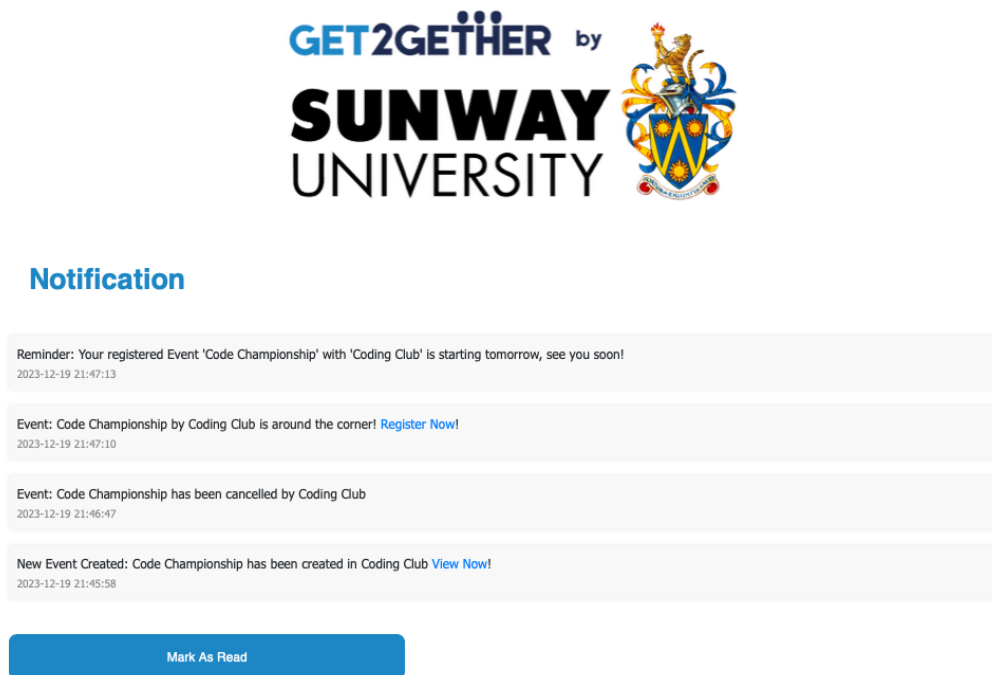


Figure 4:17 – Notification

Figure 4.17 shows the Notification function of the system. User will receive notification when an Event was Created, Event was Cancelled, Club boosted the event, and Reminder as a registered participants. The hyperlink in the notification will lead the user to the respective event page, and only Registered participants of the event will received the reminder notification from the club. When user clicked on “Mark as Read”, the column `is_read` will be updated to 1 in the database, and user the notification will not be shown anymore.

```

// Check if the current date is one day before the event's start date
if (date('Y-m-d', strtotime('tomorrow')) >= date('Y-m-d', $eventStartDate)) {
    $club = fetchClubDetails($clubId, $conn);

    if ($club) {
        $clubName = $club['club_name'];

        // Fetch registered participants
        $participantsSql = "SELECT users.id, users.username FROM event_registrations
        JOIN users ON event_registrations.user_id = users.id
        WHERE event_registrations.event_id = $event_id";
        $participantsResult = $conn->query($participantsSql);

        if ($participantsResult) {
            if ($participantsResult->num_rows > 0) {
                // Send reminder notifications
                while ($participant = $participantsResult->fetch_assoc()) {
                    $userId = $participant['id'];
                    $username = $participant['username'];

                    $reminderMessage = "Reminder: Your registered Event '{$eventTitle}' with '{$clubName}' is starting tomorrow, see you soon!";
                    $escapedReminderMessage = mysqli_real_escape_string($conn, $reminderMessage);

                    $reminderInsertSql = "INSERT INTO notifications (user_id, message, is_read) VALUES ($userId, '$escapedReminderMessage', 0)";

                    if (mysqli_query($conn, $reminderInsertSql)) {
                        // Successful query
                    } else {
                        echo "Error sending reminder for user '$username': " . mysqli_error($conn);
                    }
                }
            }
        }
    }
}

```

Figure 4.18: Code Snippet – Reminder Notification

Figure 4.18 shows the implementation of the notification of the Reminder Notification. Specially for the Reminder Notification, the function y-m-d. strtotime is used to check the current date and the event start date. User can only send the reminded a day before the event start date. Then a message will be recorded into the database table “Notification” that will be show in user notification page (Figure 4.17). Similar logic was implement to all the notification function showed in Figure 4.17.

4.4.7 Single Club Page

Coding Club

Member Count 3

Join Club

Club Description

Explore the world of coding and programming

Contact

coding@example.com

Social Media

Instagram

Facebook

Clubs Events

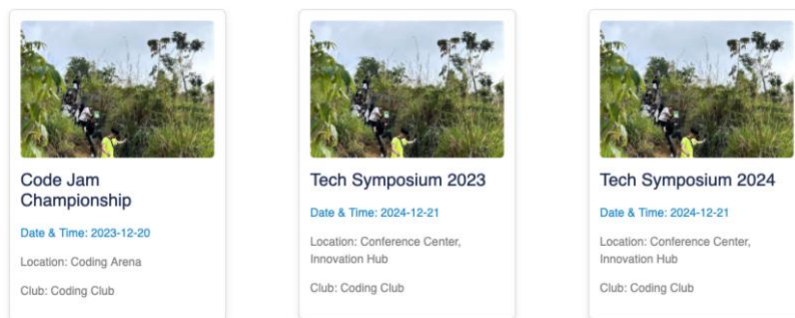


Figure 4.19: Single Club Page

Figure 4.15 shows the Single Club page that shows the Club details based on the club that user selected. The page will shows the Club name, Member Count, Club Description, Contact, Social Media, and the Event hosted by this club. User are able to join the club by pressing the “Join Club” button.

Coding Club

Member Count 3



Figure 4.20: Club Management (with Permission)

Figure 4.20 shows the Club Management function which only available for user with the role “Pic”. The user with the permission are allow the edit the details of the clubs and edit the member of the club.

```
<?php if (isset($_SESSION['username']) && isset($_SESSION['user_id'])) {  
    if ($userRole === 'pic'): ?>  
        <a href="edit_details.php?id=?php echo $club_id; ?>">  
            <button class="btn">Edit Details</button>  
        </a>  
        <a href="edit_members.php?id=?php echo $club_id; ?>">  
            <button class="btn">Edit Members</button>  
        </a>  
    <?php endif;  
} ?>
```

Figure 4.21: Code Snippet – User Role Permission

Figure 4.21 shows the implementation that use to check the user’s role. The button will only be shown is the user role is equal to ‘pic’ in the database.

Edit Club Details

New Description

Explore the world of coding and programming

New Contact Email

coding@example.com

New Instagram Link

https://instagram.com/codingclub

New Facebook Link

https://facebook.com/codingclub

Update Details

Back

Figure 4.22: Edit Club Details

Username	Current Role	Change Role
CodeClub	committee	Member ▾
CodeClub	member	Member ▾

Save Changes

Figure 4.23: Edit Member Roles

Figure 4.22 shows the form use to edit the club details and the Figure 4.23 shows the form use to manage and update the role of the club's member.

4.4.8 Club Subscription

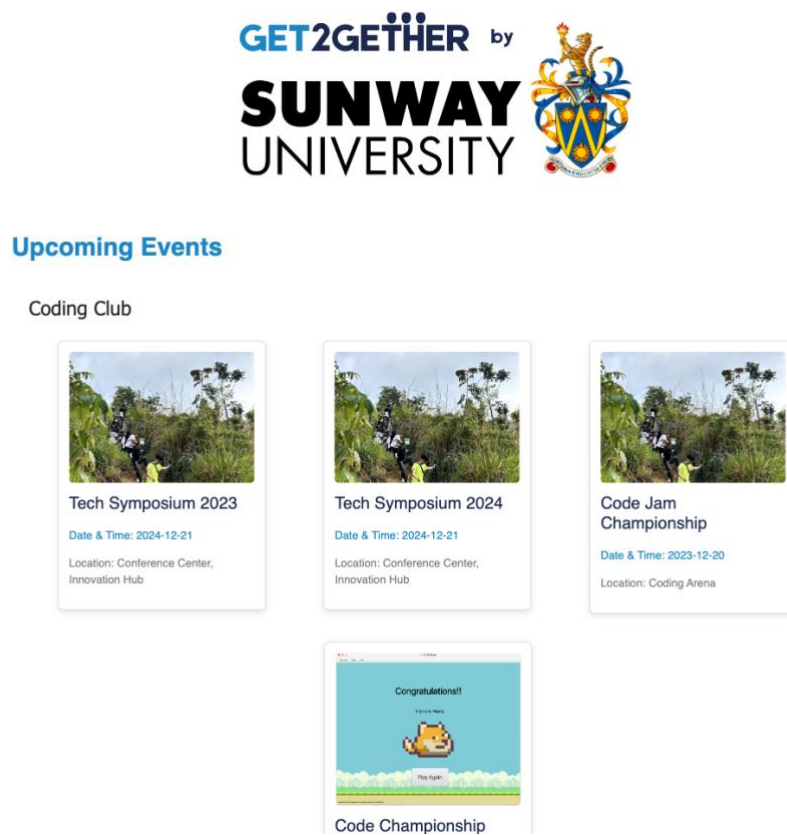


Figure 4.24: Club Subscription

Figure 4.24 shows the Club Subscription page where user can see the upcoming events of the club they joined instead of all of the events.

4.4.9 Profile

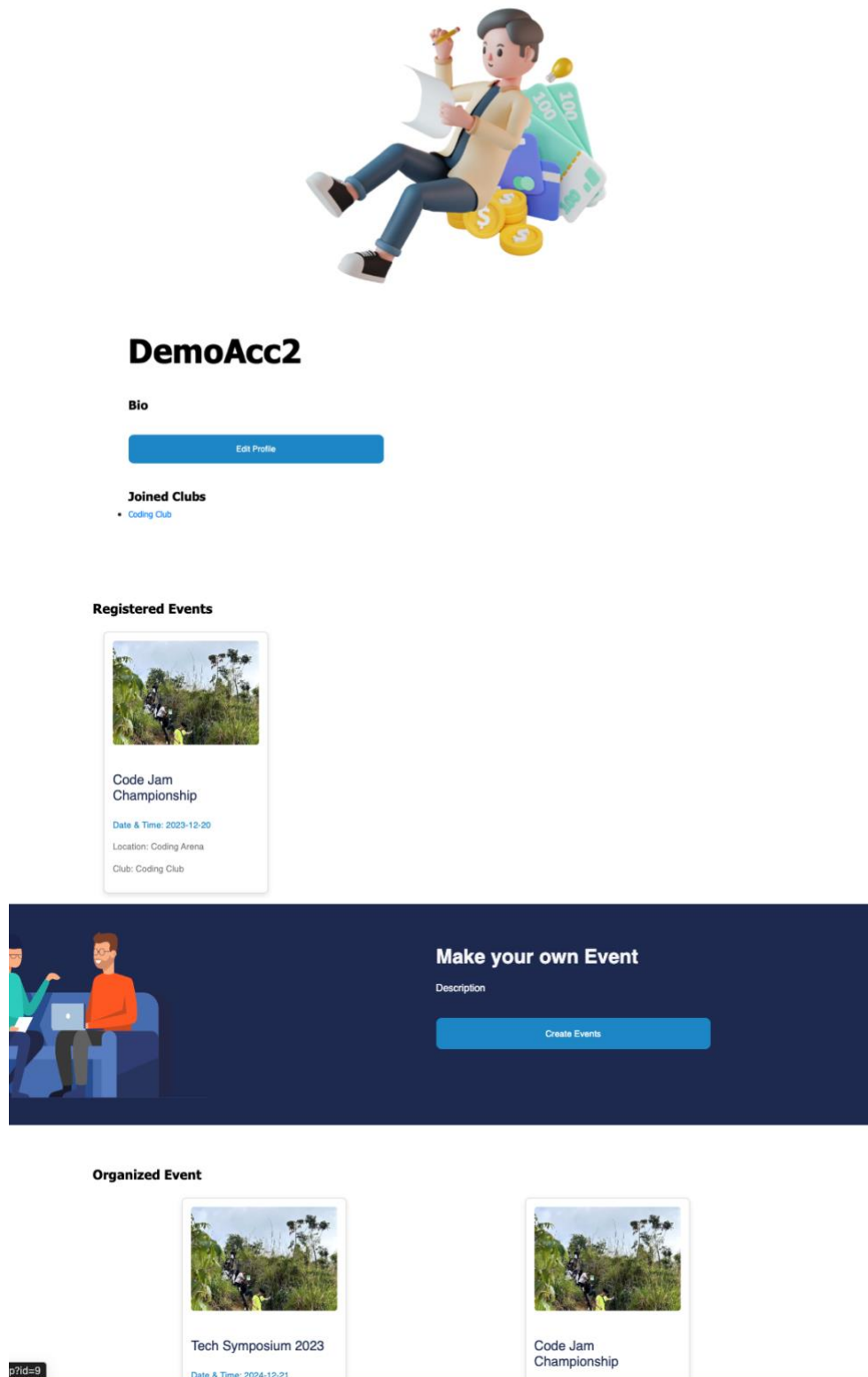
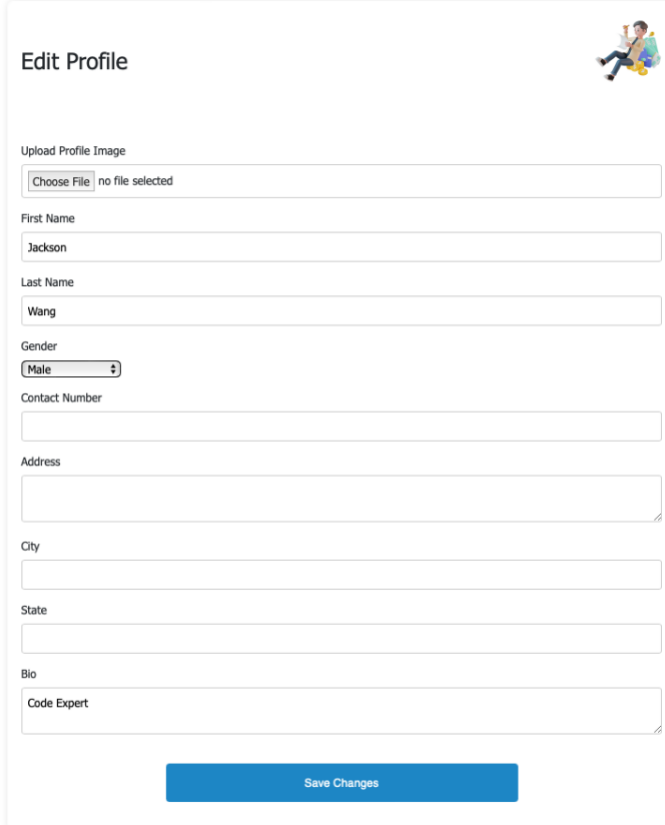


Figure 4.25: User Profile

Figure 2.5 shows the User profile with User Details such as Username, the Club they joined, Registered events and Organized Event. User are also able to edit their personal details by pressing the “Edit Profile” Button.

The image shows a web form titled "Edit Profile" with a small cartoon icon of a person sitting on a chair in the top right corner. The form contains several input fields: "Upload Profile Image" with a "Choose File" button and "no file selected" text; "First Name" with the value "Jackson"; "Last Name" with the value "Wang"; "Gender" with a dropdown menu showing "Male"; "Contact Number"; "Address"; "City"; "State"; and "Bio" with the value "Code Expert". A blue "Save Changes" button is at the bottom center.

Edit Profile

Upload Profile Image

Choose File no file selected

First Name

Jackson

Last Name

Wang

Gender

Male

Contact Number

Address

City

State

Bio

Code Expert

Save Changes

Figure 4.26: Edit Personal Details

Figure 4.26 shows the form for user to edit their details. User are able to Upload their profile picture, set their first name and last name, gender, Address, Contact number and Bio. When user pressed “Save Changes”, the details will be updated in the database.

Jackson Wang

Bio

Code Expert

Edit Profile

Joined Clubs

- Coding Club

Figure 4.27: User Details

Figure 4.27 shows the User Profile after user edit their personal details. If the user has set their first name and last name, the user profile will show their name instead of username. The bio of the user will also be updated. Private personal information such as address, contact will not be shown in the public.

4.4.10 Register and Login



Figure 4.28: Register Page

Figure 4.28 shows the Register Page with Email, Username, Password, Confirm Password.

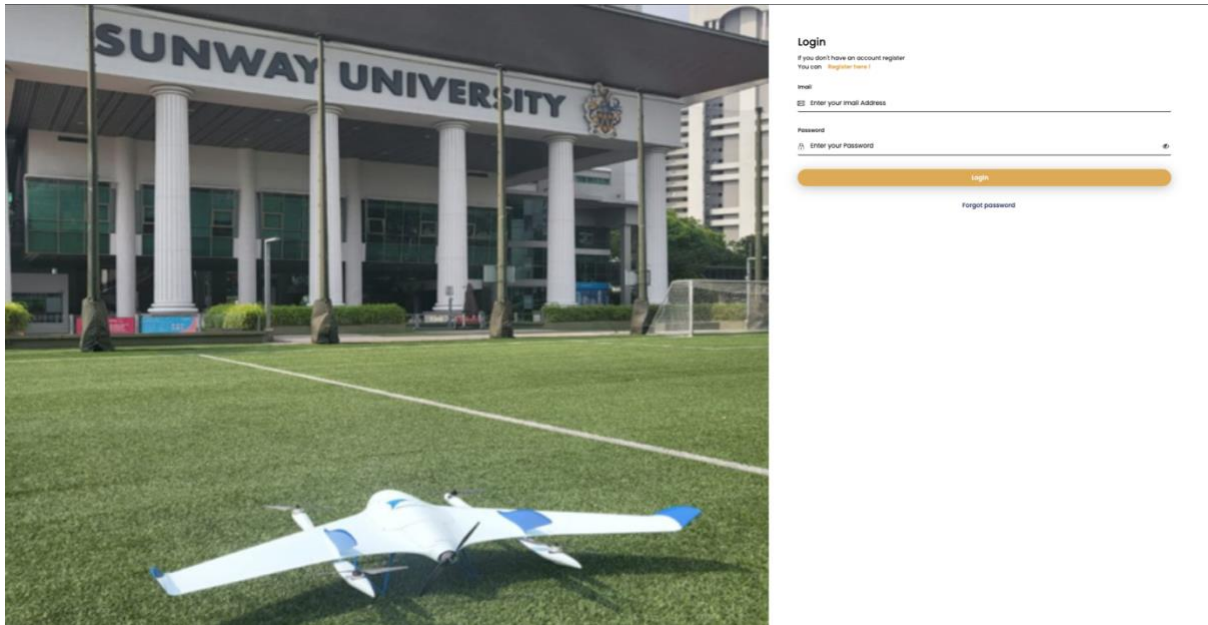


Figure 4.29: Login Page

Figure 4.29 shows the Login Page of the system. User need to key in the registered Email and Password in order to login into the system. Once user logged in successfully, user will be redirect to the home page.

4.4.11 Email Verification

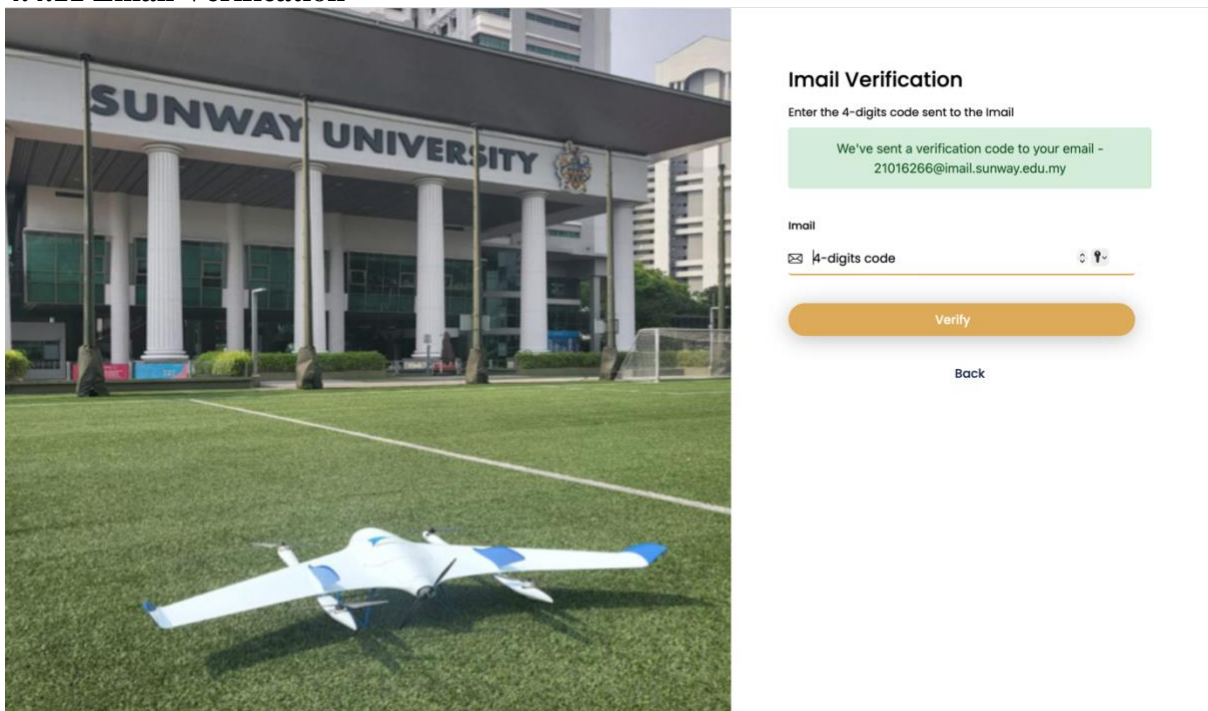


Figure 4.30: Email Verification Page

Figure 4.30 shows the Email Verification Page of the system. User need to key in the correct code which is sent to their email account in order to proceed with Login to verified the email account. If user register and haven't verified, when they login, they will be directed to this page and a message will show up letting user know that they haven't verified the email account. A new code will be generate and send to their email account, after the email is verified only user can be able to proceed to Login Page.

```
$mail = new PHPMailer();

// Set mailer configurations
$mail->isSMTP();
$mail->Host = "smtp.gmail.com";
$mail->SMTPAuth = true;
$mail->SMTPSecure = "tls";
$mail->Port = "587";
$mail->Username = "get2gethersunway@gmail.com";
$mail->Password = "rnovlmfyksctydjz";
$mail->Subject = "Iemail Verification Code";
$mail->setFrom('get2gethersunway@gmail.com');
$mail->isHTML(true);

//if user click verification code submit button
if(isset($_POST['check'])){
    $_SESSION['info'] = "";
    $otp_code = mysqli_real_escape_string($conn, $_POST['otp']);
    $check_code = "SELECT * FROM users WHERE code = $otp_code";
    $code_res = mysqli_query($conn, $check_code);
    if(mysqli_num_rows($code_res) > 0){
        $fetch_data = mysqli_fetch_assoc($code_res);
        $fetch_code = $fetch_data['code'];
        $email = $fetch_data['email'];
        $code = 0;
        $status = 'verified';
        $update_otp = "UPDATE users SET code = '$code', status = '$status' WHERE code = '$fetch_code'";
        $update_res = mysqli_query($conn, $update_otp);
        if($update_res){
            $_SESSION['name'] = $name;
            $_SESSION['email'] = $email;
            header('location: login.php');
            exit();
        }else{
            $errors['otp-error'] = "Failed while updating code!";
        }
    }else{
        $errors['otp-error'] = "You've entered incorrect code!";
    }
}
```

Figure 4.31: Code Snippet – PHP Mailer

As shown in Figure 4.31, the verification process is utilizing PHP Mailer to create the send verification code emails. In order to allow user to done the email verification before login, function is create to check the correct code.

4.4.12 Email Notification

A web form titled "Create Event". It contains several input fields: "Event Title" (text), "Event Venue" (text), "Select Club" (dropdown menu with "Coding Club" selected), "Start Time" (time picker), "End Time" (time picker), "Start Date" (date picker), and "End Date" (date picker). Below these is an "Event Description" section with an "Upload Event Image" button (labeled "Choose file" and "No file chosen") and a large text area for the description. A blue "Submit" button is at the bottom.

Figure 4.32: Create Event Page

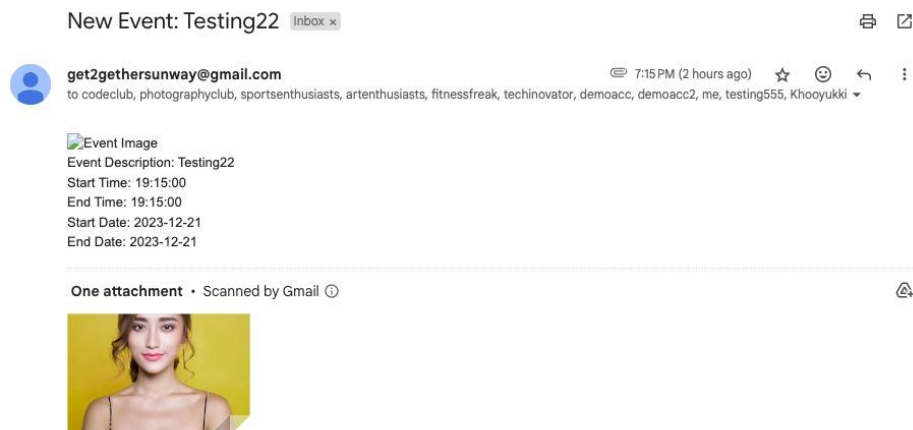


Figure 4.33: Email Notification

Figure 4.33 shows the Email Notification function of the system. As shown in Figure 4.32, as the Person-in-Charge (PIC) or committee member of the club, the user is only able to create events. Once the user has inputted all the events, they can select 'submit.' Afterward, all users will receive a notification regarding the newly created event.

```

if ($emailResult) {
    while ($row = mysqli_fetch_assoc($emailResult)) {
        // Add each email address to the recipient list
        $mail->addAddress($row['email']);
    }

    $mail->isHTML(true);

    // Fetch additional information from the events table
    $eventInfoSql = "SELECT start_time, end_time, start_date, end_date, event_image_path, event_description FROM events WHERE event_title = '$eventTitle'";
    $eventInfoResult = mysqli_query($conn, $eventInfoSql);

    if ($eventInfoResult) {
        $eventInfo = mysqli_fetch_assoc($eventInfoResult);

        // Include additional information in the email body
        $mail->Body .= "<br><img src='" . $eventInfo['event_image_path'] . "' alt='Event Image'>";
        $mail->Body .= "<br>Event Description: " . $eventInfo['event_description'];
        $mail->Body .= "<br>Start Time: " . $eventInfo['start_time'];
        $mail->Body .= "<br>End Time: " . $eventInfo['end_time'];
        $mail->Body .= "<br>Start Date: " . $eventInfo['start_date'];
        $mail->Body .= "<br>End Date: " . $eventInfo['end_date'];

        $mail->AddEmbeddedImage($eventInfo['event_image_path'], 'event_image', 'event_image.png');
    } else {
        echo "Error fetching event information from the database.";
    }

    $mail->Subject = "New Event: " . $eventTitle;

    $mail->send();
}

```

Figure 4.34: Code Snippet – PHP Email Notification

As shown in Figure 4.34, the sending process utilizes PHP to generate the code for sending email notifications about newly created events.

4.5 Testing and Quality Assurance

A comprehensive testing strategy, including unit testing, integration testing, and user acceptance testing, was employed to ensure the reliability and robustness of the developed features. Quality assurance processes were integrated throughout the development cycle to identify and rectify issues promptly.

Test Case ID	TC-001
Use Case ID	UC-001
Objective	Test if the user can register an account successfully with valid information.
Condition	1. An existing user account must not exist. 2. User is not currently logged in. 3. Server hosting the website and database must be online.
Test Steps	1. The user clicks on the “Register” button. 2. The user inputs their personal details and clicks “Register”.
Input Data	User’s details
Expected Result	Create a new account and navigate the user into the homepage.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-002
--------------	--------

Use Case ID	UC-002
Objective	Test if the user can receive OTP verification during registration.
Condition	1. An existing user account must not exist. 2. User is not currently logged in. 3. Server hosting the website and database must be online.
Test Steps	1. The user filled up their personal details and clicks “Register”. 2. The user will receive an email notification
Input Data	6 integers
Expected Result	A notification sent to users.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-003
Use Case ID	UC-003
Objective	Test if the user can login with an account successfully with an existing email account.
Condition	1. The user account must exist. 2. Login credentials must be valid. 3. User is not currently logged in. 4. Server hosting the website and database must be online.
Test Steps	1. The user clicks on the “Sign In” button. 2. User enters their email and password and clicks the “Login” button.
Input Data	User’s details
Expected Result	Navigate the user into the homepage.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-004
Use Case ID	UC-004
Objective	Test if the user can search for club name as search keyword.
Condition	1. User has logged into an existing customer account. 2. Server hosting the website and database must be online. 3. There exists a club with title containing the search keyword.
Test Steps	1. User enter the search keyword for the desired club. 2. System search club(s) based on the user keyword with the system database.
Input Data	Search keyword: CodeClub

Expected Result	System displays a list of clubs with their titles containing the search keyword.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-005
Use Case ID	UC-005
Objective	Test if the user can search for event name as search keyword.
Condition	<ol style="list-style-type: none"> 1. User has logged into an existing customer account. 2. Server hosting the website and database must be online. 3. There exists an event with title containing the search keyword.
Test Steps	<ol style="list-style-type: none"> 1. User enter the search keyword for the desired event. 2. System search event(s) based on the user keyword with the system database.
Input Data	Search keyword: Annual Charity Gala
Expected Result	System displays a list of events with their titles containing the search keyword.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-006
Use Case ID	UC-006
Objective	Test if the user can join club.
Condition	<ol style="list-style-type: none"> 1. The user has logged into an existing customer account. 2. The server hosting the website and database must be online. 3. The selected club is not in the joined club list.
Test Steps	<ol style="list-style-type: none"> 1. The user clicks on a club card. 2. The system navigates the user to the club details page. 3. The user clicks on the “Join Club” button.
Input Data	-
Expected Result	The system displayed a success message: “Joined the club successfully!”
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-007
Use Case ID	UC-007
Objective	Test if the user can view the event details information.

Condition	Server hosting the website and database must be online.
Test Steps	1. The user clicks on an event card. 2. The system navigates the user to the event details page.
Input Data	-
Expected Result	The system displays detailed information about the club.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-008
Use Case ID	UC-008
Objective	Test if the user can register event
Condition	Server hosting the website and database must be online.
Test Steps	1. The user clicks on an event card. 2. The system navigates the user to the event details page and clicks on the “Register” button. 3. The user clicks on the “Confirm Registration” button
Input Data	-
Expected Result	The system displayed a success message: “Event registration successful!”
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-009
Use Case ID	UC-009
Objective	Test if the user can cancel registration
Condition	Server hosting the website and database must be online. An event is registered
Test Steps	The system navigates the user to the event details page and clicks on the “Cancel Registration” button.
Input Data	-
Expected Result	The system displayed a success message: Cancelled Confirmed
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-010
Use Case ID	UC-010
Objective	Test if the user can view Club Details under Event
Condition	Server hosting the website and database must be online.

Test Steps	The system navigates the user to the event details page and clicks on the “View Club Details” button.
Input Data	-
Expected Result	The system displays detailed information about the club.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-011
Use Case ID	UC-011
Objective	Test if the user can view the events hosted by the club user joined
Condition	Server hosting the website and database must be online.
Test Steps	User clicks on the “subscription” button.
Input Data	-
Expected Result	The system displays events hosted by the club user joined.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-012
Use Case ID	UC-012
Objective	Test if the user can view Profile
Condition	Server hosting the website and database must be online.
Test Steps	User clicks on the “Profile” button.
Input Data	-
Expected Result	The system displays user’s profile.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-013
Use Case ID	UC-013
Objective	Test if the user can edit profile
Condition	Server hosting the website and database must be online.
Test Steps	<ol style="list-style-type: none"> 1. User clicks on the “Profile” button. 2. User clicks on the “Edit Profile” button. 3. User update the info and clicks on the “Save Changes” button.
Input Data	User’s personal details
Expected Result	The system updates user’s profile.
Actual Result	Matches the expected results.

Test Result	Pass
-------------	------

Test Case ID	TC-014
Use Case ID	UC-014
Objective	Test if the committee can edit club details
Condition	Server hosting the website and database must be online.
Test Steps	<ol style="list-style-type: none"> 1. Login as committee 2. User clicks on the “Profile” button. 3. User clicks on the “Edit Profile” button. 4. User update the info and clicks on the “Save Changes” button.
Input Data	Club details
Expected Result	The system updates user’s profile.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-015
Use Case ID	UC-015
Objective	Test if the committee can edit member position
Condition	Server hosting the website and database must be online.
Test Steps	<ol style="list-style-type: none"> 1. Login as committee 2. User clicks on the its club card. 3. User clicks on the “Edit Members” button. 4. User update the info and clicks on the “Save Changes” button.
Input Data	-
Expected Result	The system updates user’s role.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-016
Use Case ID	UC-016
Objective	Test if the committee can create event
Condition	Server hosting the website and database must be online.
Test Steps	<ol style="list-style-type: none"> 1. Login as committee 2. User clicks on the “Create Events” button. 3. User filled up the event details and clicks on the “Submit” button.
Input Data	Event details
Expected Result	The system created an event.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-017
--------------	--------

Use Case ID	UC-017
Objective	Test if the committee can cancel event
Condition	Server hosting the website and database must be online.
Test Steps	<ol style="list-style-type: none"> 1. Login as committee 2. User clicks on the “Profile” button. 3. User clicks on the event card. 4. User clicks on the “Cancel Event” button.
Input Data	-
Expected Result	The system removed an event.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-018
Use Case ID	UC-018
Objective	Test if the committee can edit event
Condition	Server hosting the website and database must be online.
Test Steps	<ol style="list-style-type: none"> 1. Login as committee 2. User clicks on the “Profile” button. 3. User clicks on the event card. 4. User clicks on the “Edit Event” button. 5. User filled up the event details and clicks on the “Update Details” button.
Input Data	Event Details
Expected Result	The system updated the event details.
Actual Result	Matches the expected results.
Test Result	Pass

Test Case ID	TC-019
Use Case ID	UC-019
Objective	Test if the committee view registered participant for the event
Condition	Server hosting the website and database must be online.
Test Steps	<ol style="list-style-type: none"> 1. Login as committee 2. User clicks on the “Profile” button. 3. User clicks on the event card.
Input Data	-
Expected Result	The system displays the registered participant for the event.
Actual Result	Matches the expected results.
Test Result	Pass

5.0 Application of Software Engineering Knowledge in Software System Development

This section should highlight THREE chosen SE knowledge applied in this project. Each knowledge should be linked to respective theories and applications and the decision should be justified. Marks will be given based on relevance and elaboration that demonstrate level of understanding.

5.1 Software Architecture

Model-View-Controller

The Model-View-Controller (MVC) architecture was chosen for Get2Gether, the event management website, because of its methodical division of duties. MVC neatly separates the data storage layer (Model), the presentation layer (View), and the application layer (Controller), allowing for greater clarity and versatility in the development process. This option is especially appropriate for an event management system since it ensures effective management of club details, event schedules, and user profiles (Model), while also providing an interesting user experience through the design of visual elements and the user interface (View). The Controller processes user input, executes business logic, and serves as a vital link between the Model and the View. This type of organisation not only allows modular development and maintainability, but also effortlessly accommodates future improvements. In essence, MVC provides the necessary basis for building a comprehensive and scalable student event management platform adapted to the project's specific requirements.

Client-Server

Due to the fact that it provides a framework that is both well-structured and efficient, the client-server architecture was chosen to be implemented. This design allows clients, which can represent either students or event organisers, to connect without any difficulty to a centralised server that houses the web application, databases, and application logic. Using this paradigm, data management and resource sharing are optimised, and it ensures that information pertaining to the event, including as event details and participant registrations, is easily accessible to both the people who create the event and the people who attend it. On top of that, the architecture is designed to suit the dual roles of Event Organisers and Event Participants. Users are able to switch between these positions with ease, which encourages flexibility and collaboration inside the Get2Gether event management system.

Publish-Subscribe

An event management website benefits from a Publish-Subscribe architecture, which loosely couples system components through event publication without knowing the subscribers. This architecture allows the creation of specific components, such as event organiser interfaces, while other components can subscribe to these events and present them to end-users in various ways, such as news feeds, personalised recommendations, and search results. This architecture is beautiful because the publishing component is agnostic about how the subscribing components interpret or display events as long as they follow a standard event format. New subscriber components can be added to introduce new pages or features without publisher adjustments. New partner or mobile app publishing interfaces can be developed without affecting display and subscription components. This loose coupling and adaptability are essential for a big event management platform that supports several event sources and consumption techniques. It streamlines development, promotes scalability, and allows non-intrusive integration of cross-cutting concerns like analytics and rate-limiting, improving platform efficiency and feature development velocity.

5.2 Software Testing

Testing is a vital phase of any software development lifecycle. Following two of the seven testing principles like "testing shows presence of defects, not their absence" and "early testing saves time and money" guided our testing strategy. We prioritized testing critical functionalities based on risk to the user and the clubs & societies.

We utilized decision tables to define and test key login workflows, and state transition diagrams to model registration system states and transitions. These techniques uncovered defects early, reducing cost. Test-driven development enabled continuous, iterative testing throughout development lifecycles. Each iteration concluded by reviewing results to improve subsequent testing. We leveraged both white-box (focuses on code) and black-box (focuses on functionality) methodologies.

Component/Unit testing isolated components to validate integrations points. We built test cases aligned to requirements specifications, executing them before/during/after development to

catch issues early. Regression testing after fixes ensured no new issues were introduced. For user-facing functions, black box testing evaluated software behaviour against requirements. Testing focused on usability from an end-user perspective. Lastly, user acceptance testing verified the software works as intended for users. Adopting a structured, requirements-based testing strategy following established software development and testing principles enabled us to build higher quality software and embed quality earlier in development lifecycles.

Decision Table Testing

Test Condition ID	Test Condition
1	Email Valid
2	Email Invalid
3	Password Valid
4	Password Invalid
5	Login to Homepage
6	Display Error Message

Condition	1	2	3	4	5	6
Email Valid	Y	N	N	N	Y	N
Password Valid	N	N	Y	N	Y	N
Actions						
Login to Homepage	N	N	N	N	Y	N
Display Error Message	Y	Y	Y	Y	N	Y

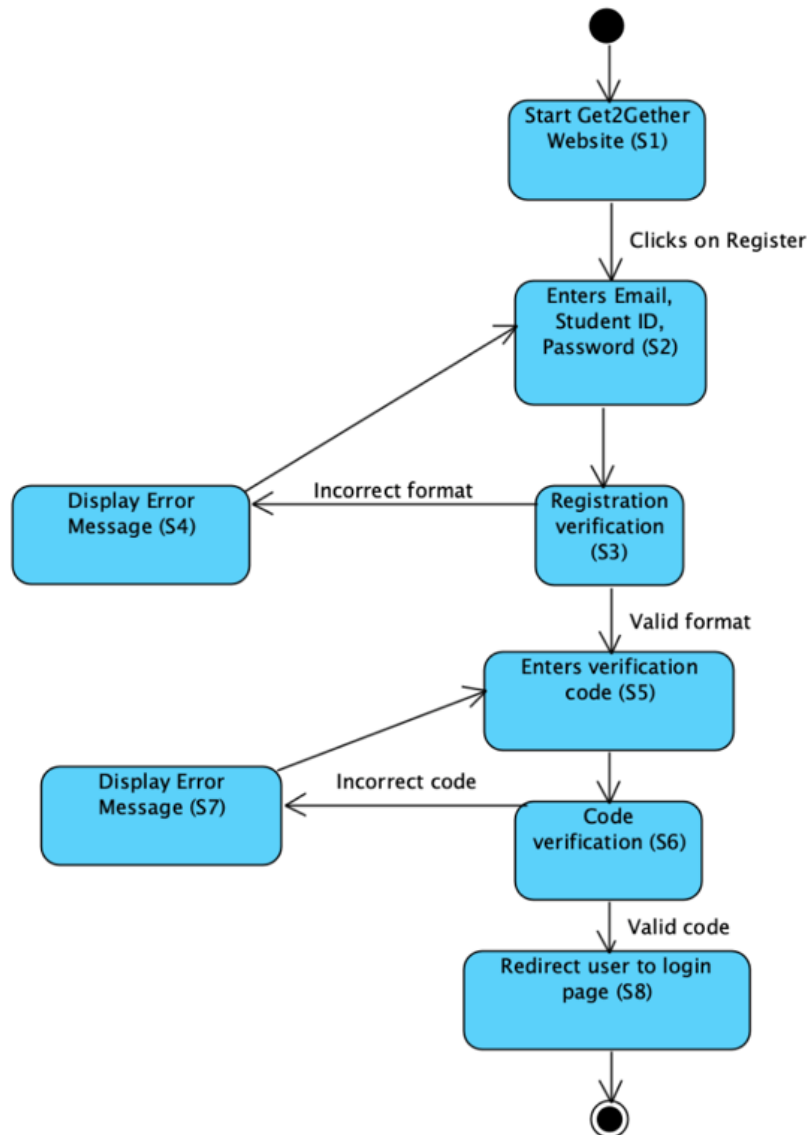


Figure 5.1: State transition diagram for registration function

5.3 Requirement Engineering

ID	Name	Description	Roles
FR-01	Create an account	Users register for an account using their university email addresses, either as a participants or organisers.	User
FR-02	Login	Users like participants and organisers are allowed to login into their account.	User
FR-03	Create events	Registered accounts will be able to create events	User
FR-04	View created events	Registered accounts will be able to view created events.	User
FR-05	Manage created events	Registered accounts will be able to manage created events.	User
FR-06	Cancel created events	Registered accounts will be able to cancel their created events.	User
FR-07	Search event availability	Users can browse events by event type, club society and date.	User
FR-08	View available events	Users will be shown the available event details and information	User
FR-09	Register event	User should be able to register an event.	User
FR-10	Manage user profile	Users can manage profiles with personalized details.	User
FR-11	View club details	Users can view club details.	User
FR-12	Subscribe interested club	Users will be able to subscribe to interested club.	User
FR-13	Receive notification	Users will receive notifications for club event updates.	User

Figure 5.2: Functional Requirements

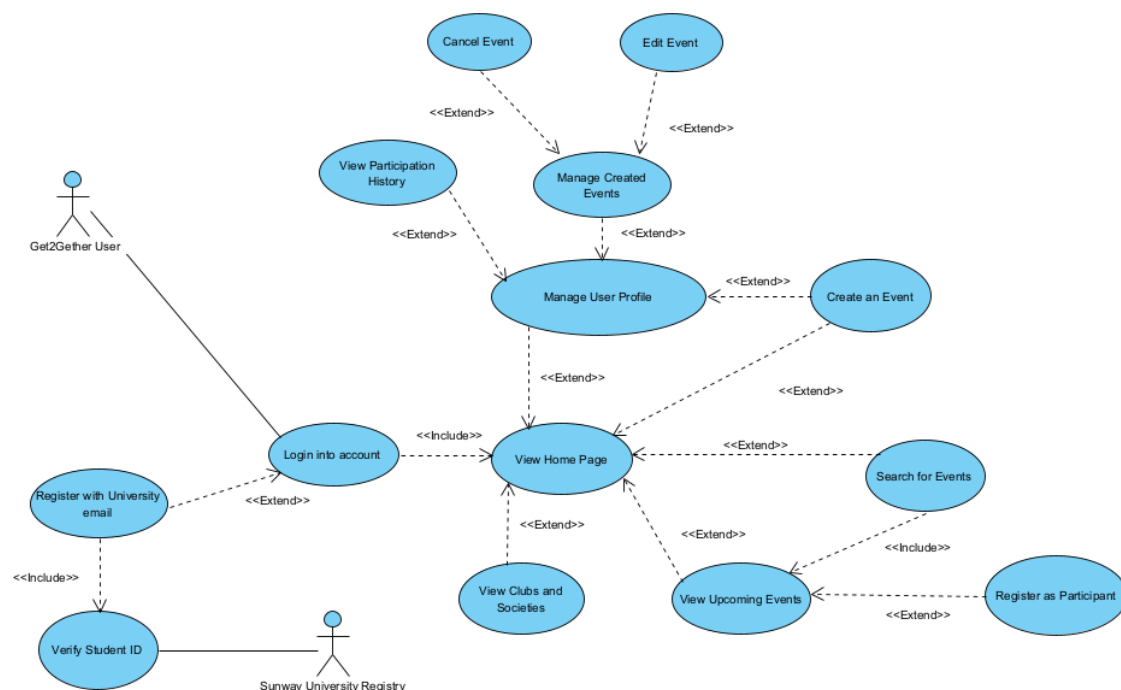
ID	Name	Description	Roles
01	Performance	Users using the system to register event during normal operation and the system should process the user's registration with a latency average of three seconds.	User
02	Security	An unauthorized user with an invalid email address tries to login the website during operation. The system should not allow user from accessing the website within 2 minutes of detection.	User
03	Usability	A regular user with no technical background visits the event management website during runtime and is able to use each function easily within 2 minutes of usage.	User
04		An organizer would like to make changes to the event details during normal operation. The organizer can make changes to the event within a minute with no issues.	User
05	Modifiability	During the system evolution phase, the developer wishes to add a notification feature to the system. Within a day, the feature is added to the system and tested.	Developer, Architect
06	Testability	The system tester awaits the completion of the error validation code during development and proceeds to perform a test by entering invalid inputs. The results capture those errors with 100% rate of detection within 10 seconds.	Tester

Figure 5.3: Quality Attributes

It was critical to comprehend and implement the various forms of software engineering requirements during the development of our website application. The aforementioned requirements serve a critical function in the identification of system flaws and opportunities for enhancement. They guarantee that the documentation conforms to established guidelines, the content is thorough, and a consensus is reached among stakeholders from a unified standpoint. The requirements play a critical role in guaranteeing that the system satisfies the stakeholders' expectations. Each of the two primary classes of requirements that dominated our project—functional requirements and quality attributes was described in detail in the previous section of this document.

Our strategy was predicated on the distinct differentiation of these categories of requirements. Functional requirements delineate the actions that the system ought to execute, providing intricate detail regarding the services, functions, and operations that the website ought to execute. In contrast, quality attributes establish the benchmarks that the system ought to satisfy, encompassing aspects such as usability, performance, and dependability.

By classifying requirements into two discrete categories, we acquired a more profound comprehension of the website's workflow and detected any omissions that warranted enhancement. This method also assisted us in determining which features the system ought to exclude in order to prevent functional overlap. Therefore, due diligence and implementation of these categories of requirements were critical to the successful development of our website.



Some of the activities that were conducted when defining the requirements for the website included negotiation, elicitation, and documentation. The requirements underwent a process of negotiation among stakeholders to ensure alignment and consensus. This negotiation process involved ongoing face-to-face meetings held weekly, where our team met physically to discuss

and refine the requirements. During these in-person meetings, every stakeholder had the opportunity to express their thoughts and opinions openly, fostering a collaborative environment and preventing potential miscommunications among team members.

As such, Figure 5.5 shows the initial prototype for Get2Gether Event Management in creating. This prototype was reference multiple times in future meetings to elicit more requirements that might've been missed and enhance the design to more user-friendly.

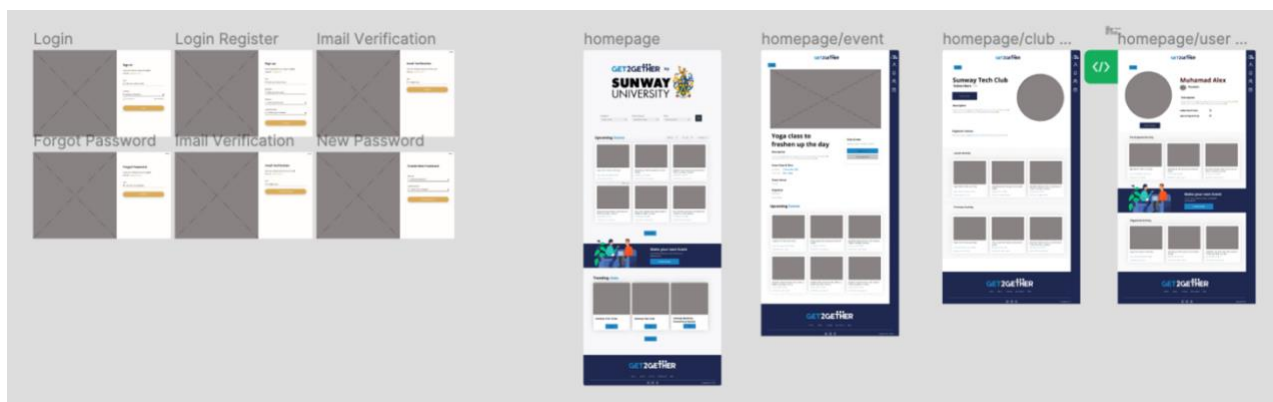


Figure 5.5: Wireframe

In order to address uncertainties in the documentation of requirements, we placed emphasis on the importance of clarity and precision. Linguistic, syntactic, and semantic ambiguities were identified. We ensured unambiguous communication and reduced the likelihood of misunderstandings by employing strategies such as maintaining a glossary for potentially ambiguous terms and structuring requirements using syntactic patterns when required.

In addition, elicitation of requirements was utilised in the process of determining the criteria for our event management website. We examined existing systems and documents, as well as conducted interviews with the target audience of our website. We even compared our performance to that of other websites and applications that are also involved in event management. This facilitated our comprehension of the essential features that our website ought to possess. We also reviewed regulatory documents and policies to ensure that our website was appropriate for our target audience.

Engaging in discussions with various individuals who would utilise our website proved to be exceedingly beneficial. Imitating one another's thoughts, we collectively devised additional

criteria through discussion. When we first launched our website, for instance, only students could create events. However, following discussion and ideation, we also added features for club management. Software engineering requirements are analogous to a structure's substructure. A greater likelihood of success exists for the undertaking if the requirements are precise. As a result, we devoted considerable effort to ensuring that our website satisfied the requirements of our consumers.

6.0 Conclusion

In summary, the implementation of the University Student Event Management Website called Get2Gether signifies a substantial advancement in the improvement of the event ecosystem within the university. By virtue of its all-encompassing and intuitive database, it enables pupils to conveniently retrieve data pertaining to diverse student organisations and their activities, thereby promoting increased participation among members of the campus community. The current features of the website provide a strong basis for future improvements, allaying the groundwork for a paradigm shift in event organisation, discovery, and participation. Nevertheless, a number of unresolved matters and prospects for further investigation remain. To begin with, the implementation of ongoing user feedback and iterative enhancements will be indispensable in the process of perfecting the user experience and maximising the efficacy of the platform. Furthermore, to amplify its influence, the website could be equipped with sophisticated algorithmic event recommendation systems, instantaneous event updates, and seamless integration with social media platforms. In addition, ongoing concerns include the improvement of accessibility features to accommodate a wide range of users and the implementation of strong security measures to safeguard user data. In essence, the University Student Event Management Website is in its nascent stages, and its development and evolution hold tremendous promise for enhancing its capacity to cater to the needs of the university community in subsequent periods.