FACULTY OF COMPUTING

UNIVERSITI TEKNOLOGI MALAYSIA

YOUTH VENTURE DATA MANAGEMENT SYSTEM

INNOVATIO

AA WBL PROJECT: GROUP REPORT

NO.	NAME	MATRIC NUMBER
1	WAN NUR SOFEA BINTI MOHD HASBULLAH	A22EC0115
2	LOW YING XI	A22EC0187
3	MUHAMMAD DANIAL BIN AHMAD SYAHIR	A22EC0206
4	THEVAN RAJU A/L JEGANATH	A22EC0286

COURSE CODE : SECP3723

COURSE TITLE : SYSTEM DEVELOPMENT TECHNOLOGY

SECTION : 02

LECTURER : DR. AHMAD NAJMI BIN AMERHAIDER NUAR

Table of Contents

Company/Client Background

The client involved in this project is called Youth Venture Asia. Youth Venture is known as a company that helps youth develop their entrepreneurial ideas and prepares them for their careers. Youth Venture is a profit-for-purpose educational and marketing organisation that provides event management services through a technologically-driven ecosystem dedicated to youth development.

Guided by a mission to empower and nurture the youth, Youth Ventures Asia is dedicated to fostering personal and career development among young individuals. Thus, their pursuit involves leveraging technology to track youth development effectively and in a more systematic manner. The rationale behind this approach is the recognition that effective data management is essential for organisations to retrieve, organise, and utilise all information resources efficiently. Less organised data across multiple databases and a lack of data arrangement can be time-consuming, requiring significant manual effort to merge, organise, and retrieve the desired information, making the process challenging and labour-intensive. Facing challenges such as repetitive data storage, manual data merging, human error during manual processes, and time-consuming data sharing with their clients, Youth Venture Asia recognises the need to adapt to the rapidly increasing workload and the growing number of youth involved in their events. They view the web development project as a strategic opportunity to address their issues and effectively track youth progress, aligning with their long-term vision for the growth of youth in society.

The current system used by Youth Venture for handling events and youth data involves different platforms such as Google Forms, Google Sheets, Airtable, WhatsApp, and so on. This raises an issue as the database is not centralised and is less organised in handling all the data. Even though the platforms involved are mature in functionality, availability, and cost-effective. However, the current data management system using a cross-platform approach is time-consuming. One relevant inconvenience is the challenge of maintaining consistency and synchronisation across multiple platforms, leading to potential data discrepancies and inefficiencies in the overall management process.

The target users of the new data management system are students, Youth Venture administrator and their clients. The proposed system aims to reduce the workload for both administrator and students with effective data compilation features and a time-saving registration and feedback process. By centralizing operations, the system aims to optimise the registration and feedback processes, ultimately saving valuable time for all stakeholders involved. A centralised system allows all stakeholders to operate within a single system, reducing the complexities of cross-platform management. It facilitates easier data sharing and compilation, streamlining processes, and enhancing overall efficiency. Thus, it can significantly enhance efficiency and reduce workload by providing the ability to adapt to the rapid increase in youth data. As a result, Youth Venture has the potential to stay ahead of the competition and ensure sustained longevity with a new system that can adapt to its changing needs.

Problem Background

Youth Venture seeks an effective system to track students' progress and development effectively to benefit its evolving community. The organisation requested to develop a data management system to track the students' progress and development who join their events. The need for such a system arises from the need for more efficient methods for storing and tracking students' development. Some data management processes, like merging data, might require manual effort and be time-consuming. The repeated filling of registration forms by the same students burdens the organisation, as the organisation has already obtained their information. Moreover, data sharing with Youth Venture’s clients and their partners becomes a challenge as manual methods are used to extract the data needed.

The system aims to reduce the workload for both admins and students with the features of effective data compilation and a time-saving registration and feedback process. This is vital because Youth Venture currently faces problems with the current system due to an increasing number of users. Manpower to maintain the system is also limited. Hence, the current system will increase the workload and cannot keep up with the growing need for better efficiency, ease and productivity. This not only diminishes Youth Venture's competitiveness, but it also makes it difficult for the company to meet customers' needs accurately.

To ensure Youth Venture stays ahead of the competition and has sustained longevity, it must proactively develop a new system that can adapt to its changing needs.

Proposed Solution

Mission Statement
Develop a system for Youth Venture to track progress and development of student

Overview
To overcome the challenges presented in the problem statement and improve the data management process for Youth Venture, we propose the development of a robust and user-friendly system known as the "Youth Venture Data Management System."

The key features of this proposed system include:

1. Centralized Data Management:

- All student data will be stored in a centralized database with organized tables for each aspect, eliminating the need for manual compilation from multiple sources. This database will include comprehensive information such as names, telephone numbers, current institutions, courses taken, email addresses, talents, academic certifications,experiences and resumes.

2. Efficient Data Entry:

- Students will conveniently input their information during event registration, guaranteeing that all essential details are captured firsthand. Additionally, after each event, students can provide feedback effortlessly through a link provided by event organizers. Administrators and clients can take a look into student participation.

3. Incentive System:

- A point-based incentive program will be introduced to motivate students to engage in the data management process actively. Students will earn badges for events they participate in. These badges can be redeemed for rewards as students reach a specific amount of badges.

4. Feedback Forms:

- The system will integrate feedback forms using external links provided by the event organizer. This streamlines the data collection and program evaluation processes. This will streamline the data collection and program evaluation processes.

5. Auto-Generated Resumes:

- Students can generate resumes from the data recorded in their profiles, information entered during event registration, and certifications and experiences added in the system. This feature will prepare them for future career development and recognition.

6. Mobile-Friendly Interface:

- The system will offer a user-friendly and mobile-responsive web-based interface, making it accessible via smartphones, ensuring students can conveniently access and engage with the platform.

7. User Account Management :

- The system will facilitate the creation and management of user accounts, serving both students and partner/clients. Administrators will have one account for efficient management of events. Meanwhile, Youth Ventures clients and partners can access participation's data to figure out potential students to accept their offers for internship and so on.

8. Registration and Event Participation :

- The system will support event registration and participation, simplifying the process for students, clients and administrators.

With the implementation of this system, Youth Venture will benefit from streamlined data management, reduced administrative burdens, improved data visibility, and enhanced student engagement. The system's comprehensive features will address the challenges posed by the existing system and empower Youth Venture to fulfill its mission effectively.

Development Approach

The system development life cycle, known as SDLC, is the basic process of developing a system. The main objective is to produce high-quality software within a reasonable timeframe. There are 7 phases in the SDLC cycle, including identifying problems,determining human information requirements,analyzing system needs,designing system needs,developing and documenting software,testing and maintaining the system, as well as implementing and evaluating the system.

Firstly, SDLC requires us to identify the problems. The team attended Industry Day 1 on 24 October 2023 and met the clients from Youth Venture Asia, Mr. Aniq Amrez and Mr. Hanif Marzuki as representatives of Youth Venture. Our clients shared their requirements and needs for the Youth Venture Data Management System during the talk. The basic requirements of the system from Youth Venture are to have feedback and evaluation forms for all programs/events, Youth Venture as administrator, functionality to generate resumes, and a badge feature. The team conducted a discussion and summarised the requirements collected from the talk.

The second phase involves determining human information requirements. In this phase, we have a briefing regarding the existing system of Youth Venture by Mr. Hanif Marzuki. The team prepared an online meeting session with Youth Venture to ask questions regarding the flow of the system and get more details about their expected work flow for the new system. Mr. Hanif requested to have a design similar to the mobile application called Setel and emphasise the reward and badge features should be included in the new system based on our creativity.

After the meeting, the team proceeded to phase 3, which is analysing system needs. The team had a discussion on how to expand the ideas to create the new system. First of all, the team started to prepare a proposal of the system to propose the early ideas on how the system works out. Moreover,the team proceeded to create Entity Relational Diagram (ERD) and Data Dictionary for the system. The team had meetings during the class hours to discuss and design ERD and Data Dictionary for the database.

In addition, phase 4 focused on designing the system, which includes the user interface (UI) design and the user experience (UX) design. The design is based on verified ERD and the final normalisation database design. The team used the Model View Controller (MVC) approach to design the system with some minor changes in the ERD, such as the addition attributes. In the initial stages of the design process, the team established a framework for the functionality of the actual system. The subsequent steps involve code modification to align with the identified logic and incorporate pertinent features. This approach allows for the early implementation of essential functionalities, facilitating thorough testing and refinement of the system.

Next, in phase 5, which is developing and documenting software, the team developed the complete system based on the modules assigned to each member. The team started the coding for the development of the system and connected it to the localhost setup. The team members have been given a time frame to finish their respective modules to ensure the system can be completed on time. During the development of the system, the team is under the guidance of Dr. Ahmad Najmi. The development phase took around 5 weeks. After finishing all the modules, the team committed the codes to GitHub for testing, synchronisation, and version control.

Next, the team proceeded with the next SDLC phase, which is the 6th phase: testing and maintaining the system. After the coding phase was completed, the team applied testing to the functionality of the system. The team checked all the functionality that we provide in the system one by one, such as making a registration, logging in to the system, resetting passwords, editing profiles, updating profiles, creating events, editing events, deleting events, displaying a list of events, event details, managing registration of events, event registration and list of participants with the details, feedback, a resume, and also a reward & badge. The team fixed the error when inspecting any bugs or errors that occurred during the testing phase. The team prepared a user manual for the clients as their reference throughout the overview of the system and its functionalities.

Lastly,the project reached the last phase of the SDLC cycle, which is implementing and evaluating the system. During this phase, Industry Day 2 was held for pitching and demonstrating the system to the clients. The team introduced the system with its main functionalities and extra features that the team developed during the presentation.

System Design

An overview of the Youth Venture Data Management System will be provided in this section. To introduce the system's features and show how the end-user can interact with the system,a use case diagram, entity relationship diagram, and sequence diagram of the overall system will be provided in this section. The developer team formulated it into five subsystems, each of which has several use cases. The Youth Venture Data Management System consists of five different subsystems: Authentication subsystem, Profile subsystem, Event subsystem, Resume subsystem, Reward and Badge subsystem. Youth Venture, students, and client/partner are the target users of this system.

This use-case diagram shows the modules presented in the system.

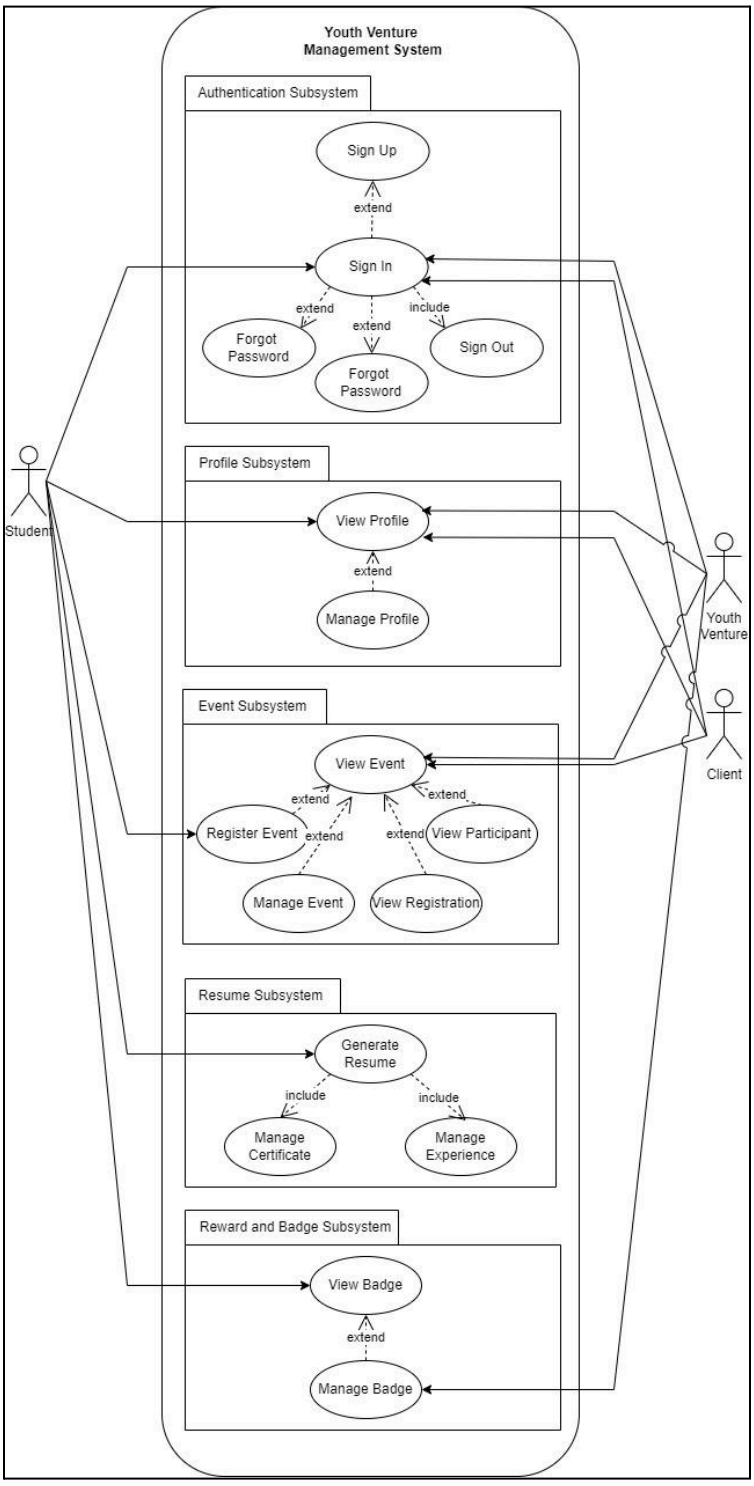


Figure 1. Use Case Diagram of Youth Venture Management System

This entity relationship diagram shows how the entities in the database interact with each other.

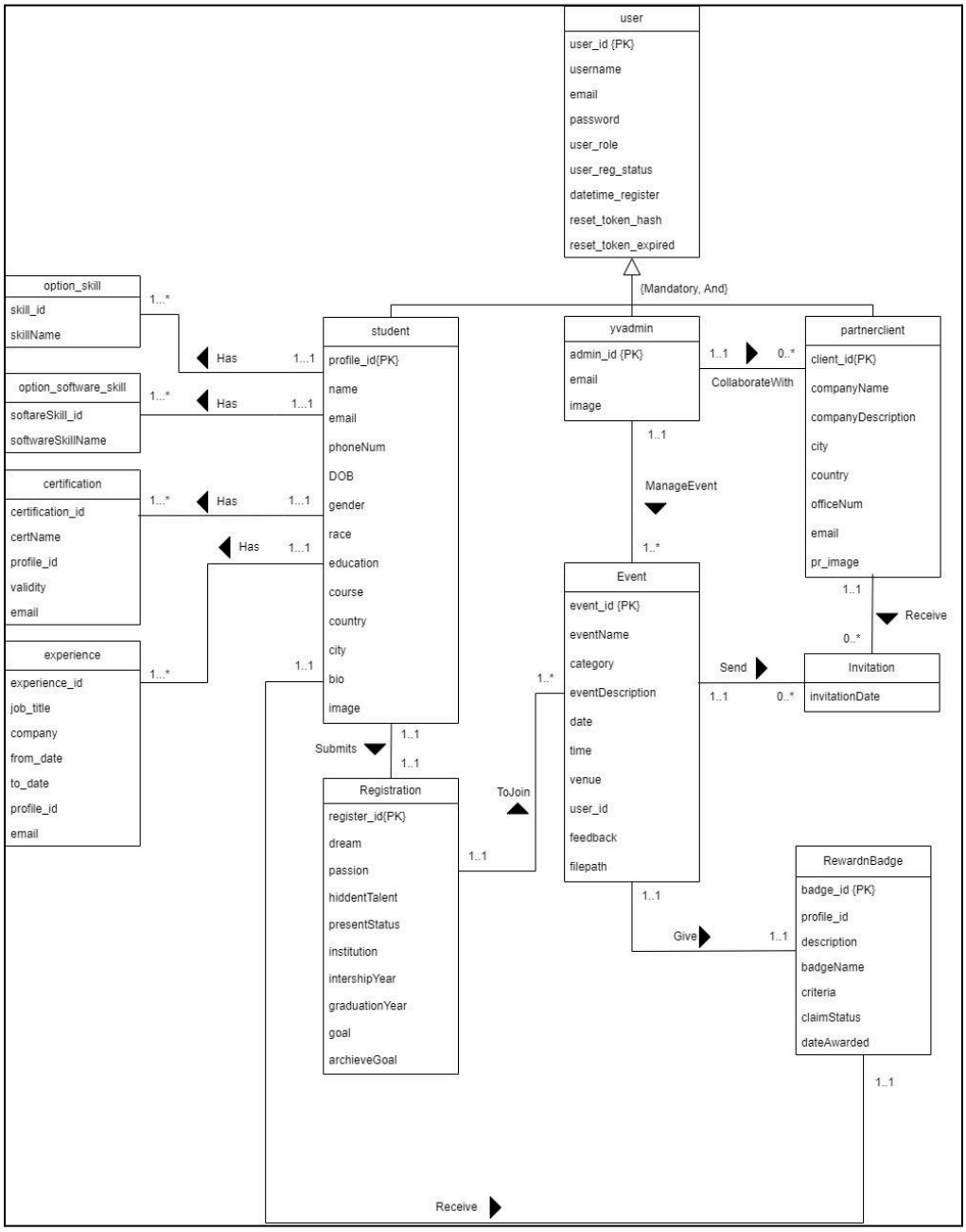


Figure 2. Entity Relationship Diagram

This diagram shows the flow of systems for all users.

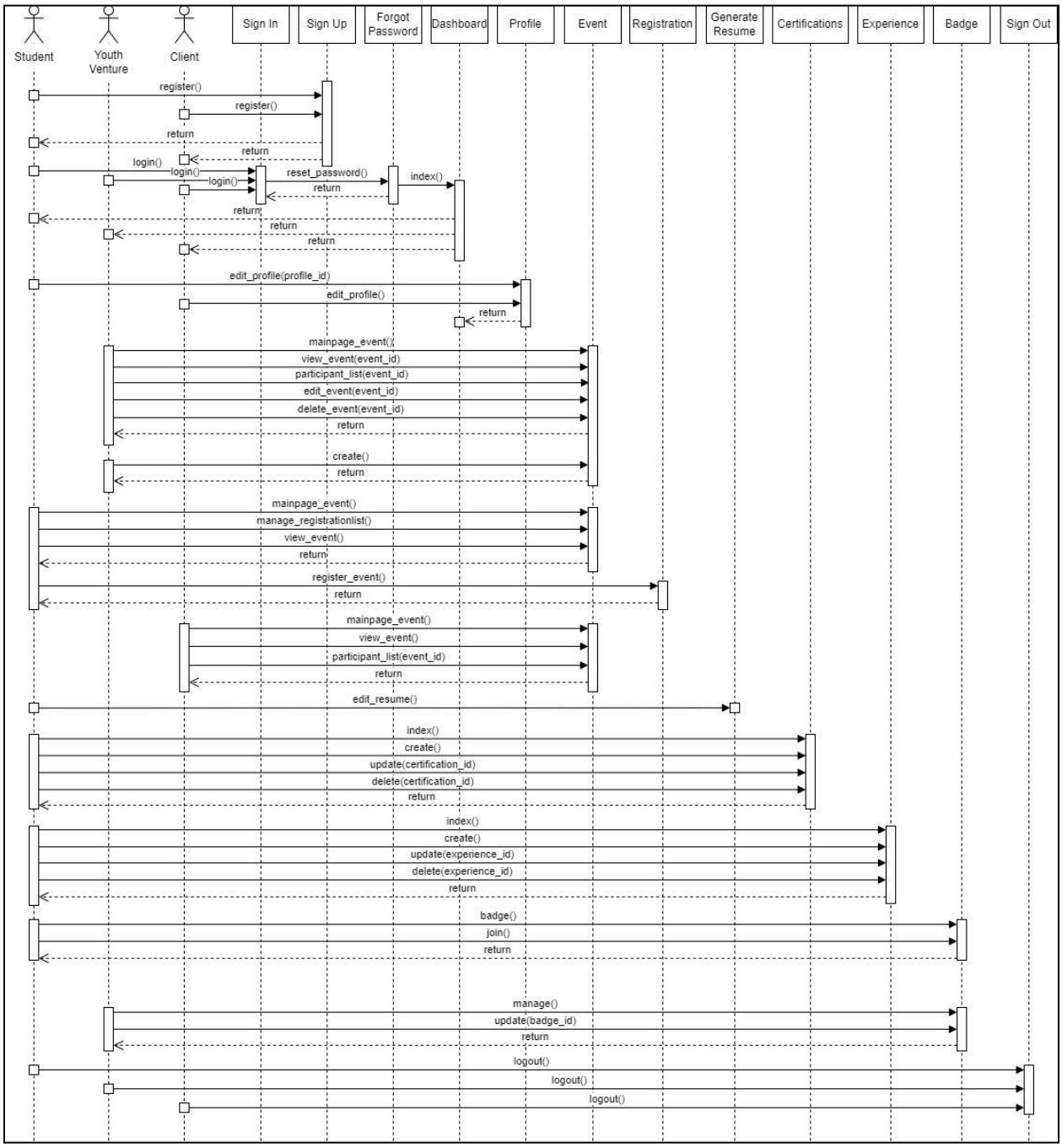


Figure 3. Sequence Diagram of Youth Venture Management System

Development Software, Language, Technology, API and Tools

1. Development Software

   1.1. Visual Studio Code

   This is the platform as a code editor that provides an environment for the coding tasks of the system.

2. Programming Language

   2.1. HTML (Hypertext Markup Language)

   The standard markup language is used to create and structure the content on web pages. It provides the basic building blocks for web development, especially the interface, and also defines elements such as headings, paragraphs, links, images, and more.

   2.2. PHP (Hypertext Preprocessor)

   The server-side scripting language used for web development. It is used for creating dynamic webpages and handling form data. In this project, PHP is used for interacting with databases, making it an integral part of server-side programming. PHP is used for executing SQL queries for managing and manipulating relational databases. This allows the developer to integrate database operations into web applications, such as retrieve, insert, update or delete data in databases.

   2.3. JavaScript

   Programming language that is mainly used for client-side web development. It enables interactive and dynamic features on web pages, such as asynchronous communication with servers.

3. Technology

   3.1. Browser

   Web browsers are fundamental technologies in the web development process. They interpret and render HTML, CSS, and JavaScript, allowing users to access and interact with web applications. Developers use browsers for testing, debugging, and optimising web pages.

   3.2. PHP Mailer

   PHP Mailer is an open source PHP library used for sending emails from a PHP application. It provides a flexible way to incorporate email functionality into web

applications and simplifies the process of composing and sending emails. In this project, it is used to send emails to users who would like to reset their account password.

4. Tools

4.1. XAMPP

In this project, XAMPP is used as a cross-platform software package that includes Apache, MySQL and PHP that provides a local development environment for the purpose of setting up and testing web applications on the local machines.

4.2. PhpMyAdmin

Web-based administration tool for managing MySQL databases and can perform database operations such as creating, modifying, and querying databases.

4.3. GitHub

GitHub is a web-based platform that provides version control and collaboration for software development projects. In this project, GitHub is used to track changes in source code during system development. The development team will use it to manage and collaborate on code repositories and coordinate work on the project.

4.4. Trello

Trello is a web-based project management tool that uses boards, lists, and cards to organise team missions and prioritise important tasks. In this project, it is used for project management purposes with the involvement of the development team, lecturers, and the client to track the progress of system development. Trello is the platform used by the team to store completed documentation.

Function/Module Explanation

1.1 Module Authentication

Person in charge: Low Ying Xi

1. Log In

Youth Venture Data Management System will require all users, including administrator, students, and clients, to log in to enter the system. Users need to provide the correct email address and password that have been registered to log in to the system. When an invalid/wrong email or password is entered, error message will be displayed and users cannot log in.

Controllers: Users.php>login() & createUserSession($user)

1) login()

   Attribute: $data, $loggedInUser

   Function used:

   i) $this->userModel->findUserByEmail($data[‘email’])

   ii)$this->userModel->login($data['email'], $data['password'])

   iii)$this->createUserSession($loggedInUser)

Explanation:

This login() function is a public function that will declare $data that includes ‘title’, ‘email’, ‘password’, ‘emailError’, ‘passError’, It will check if the request method is POST, and the function will process the form data; otherwise, $data will maintain the default values. The function will sanitise the post data and prevent malicious input, then assign the sanitised post data to $data. The email and password will be validated. If it's empty, it sets an error message. Using ‘findUserByEmail’, it checks the email to ensure it exists in the database. If the email and password have no errors, it will call the ‘login’ function to log in to the system. If login is successful, it will create a session using the ‘createUserSession’ function. If there is an error, it will render the login view with $data.

createUserSession($user)

Attribute: $_SESSION

Function used:
i)$this->userModel->getUserImage($user->email, $user->user_role)

Explanation:
This function sets various user-related information in the $_SESSION such as ‘user_id’, ‘username’, ‘email’, 'user_role', and 'user_image', using the data in $user. The function ‘getUserImage’ is used to fetch the user’s image. After setting it, the function will redirect the user to the home page of the system.

Models: User.php

1) findUserByEmail($email)
This function is used to check the existence of a user in the database based on the email address. The method eventually returns the result of ‘rowCount’. If the count is greater than zero, it suggests that a user with the specified email already exists in the database. Conversely, if the count is zero, it implies that no user with that email is currently registered. It plays a crucial role in the user registration or login processes, preventing duplicate registrations with the same email and ensuring the uniqueness of user accounts.

2) login($email, $password)
This function is used to verify user credentials during the login process. The function proceeds to check if a user record was successfully retrieved. If a user is found with the specified email, the function then retrieves the hashed password from the user record and employs the password_verify function to compare it with the provided plaintext password. If the password verification is successful, indicating that the provided password matches the stored hashed password, the function returns the user row, signifying successful authentication. If no user is found with the provided email or if the password verification fails, the function returns false. This indicates that the authentication process is unsuccessful and the login attempt is denied.

3) getUserImage($email, $role)

This function is used to retrieve the profile image URL associated with a user, based on their email address and role. Checking the role of the user is necessary to construct a query to select an image from a specific table (student, partnerclient, or yvadmin). If the user's role is not identified as 'Student,' 'Partner,' or 'Admin,' the function returns null to signify that no image URL is available for the specified role.

Views: login.php

2. Log Out

Users can log out of the system. The system will end this user's session when this user selects to log out.

Controllers: Users.php>logout()

1) logout()

Explanation: This function will unset or clear the user-related information session variables, terminating the user's session and redirecting the user to the login page.

Models: -

Views: head_wrapper.php (button logout)

3. Register Account

For students and clients, they are required to register their own account to log in to the system. In the "Register Account" page, the user can select the role: 'Student' or 'Client'. There are mandatory fields for both roles, such as username, email, password, and confirm password. Other fields are required to be filled in but are different based on the role. For example,

Student needs to fill in:

i) Full name

ii) Gender

iii) Race

iv) Contact number

v) Date of Birth

vi) Course

These fields must be filled out, as they are the basic information needed by students to register for an event. By setting the fields as required, this can ensure that the student's information can be fully retrieved and directly displayed on the event registration form.

Meanwhile, Client needs to fill in:

i) Company Name

ii) Office Number

Controllers: Users.php>register()

1) register()

Attribute: $data

Function used:

i) $this->userModel->findUserByEmail($data['email'])

ii) $this->userModel->register($data)

Explanation:

This function will handle user account registration requests, process the submitted form data, validate the input, and register new users into the system. The function initiates $data, which contains various user-related information and corresponding error messages. If the collected data passes all validation checks, the function proceeds to hash the password using the password_hash function for security. The registration is then handled through the register method of the user model. If the registration is successful, the user is redirected to the login page. Otherwise, an error message is displayed.

Models: User.php

1) findUserByEmail($email)

Refer to the explanation at No. 1.

2) register($data)

This function is used to insert general user information and role-specific details into the database, facilitating the user registration process with data integrity and

security. The function executes an SQL query to insert user-related information (username, email, password, user role, datetime of registration, and registration status) into the user table. For students, the function will insert profile details into the ‘student’ table associated with the obtained user ID. For partners, the function will insert partner-specific details into the ‘partnerclient’ table with the obtained user ID. The user ID is retrieved from the function lastInsertId() in the database.php.

Views: register.php

4. Reset Password

Users can reset their password using their email address. User will be required to key in the email address they used when registering their account. If it is a valid email address and exists in the database, the system will send an email with a special link that will redirect them to the web page for setting a new password. The link will only valid for 30 minutes. If the link has already expired, the new password will not be stored in the database. User needs to request a new, valid reset password link. The new password must be more than 8 characters with a mix of letters and numbers. Both the password and the confirmation password must match each other. After password is successfully reset, user can use the new password to log in.

Controllers: Users.php>reset_password() & new_password()

1) reset_password()
   Attribute: $token, $token_hash, $expiry, $data

   Function used:
   i) $this->userModel->updateResetToken($data)

Explanation:
This function generates a reset token, sets its expiry, handles user input for initiating the reset process, updates the token information in the database, and renders the appropriate views based on the process's success or failure.

2) new_password()

Attribute: $error_msg, $data, $token, $token_hash, $this_user

Function used:
i) $this->userModel->findByResetToken($token_hash)
ii) $this->userModel->updateNewPassword($data)

Explanation:
This function manages the user's input for creating a new password, validates the password, updates the password in the database, and either redirects the user to the login page on success or displays an error message on failure.

Models: User.php

1) updateResetToken($data)
This function updates the user's reset token hash and expiration time in the database. It uses a prepared SQL query to perform the update based on the user's email address. It includes the functionality to send a password reset email to the user. It uses an external mailer script (createMailerInstance) to send an email containing a link with the reset token. The link in the email points to the 'new_password' page with the reset token as a parameter.

2) findByResetToken($reset_token_hash)
This function retrieves a user from the database based on the provided reset token hash. If a user is found, it returns the user data; otherwise, it returns false.

3) updateNewPassword($data)
This function updates the user's password in the database after a successful password reset. The function returns true if the password update is successful and false otherwise.

Views: reset_password.php, new_password.php

5. View All Users

Administrator can view all the users that exist in the system.

Controllers: Users.php> view_all_user()

1) view_all_user()

   Attribute:$users, $data

   Function used:

   i) $this->userModel->manageUser()

Explanation: This function will retrieve all the data of users in the system and renders a view to display the list of all users.

Models: User.php

1) manageUser()

   This function will retrieve all user information from the database.

Views: view_all_user.php

1.2 Module Profile

Person in charge:Muhammad Danial

1. **Create Student Profile**

   This page contains the profile information of the student. Each student can have the to access their respective page.

   Controllers: Pages.php>edit_profile()

   1. edit_profile()

      Attribute: $allowed, $filename, $filetype, $filesize, $fileExt, $fileActualExt, $ext, $username, $maxsize, $location, $data, $studentProfile

      Function:

      i) $this- >pageModel->updateStudentProfile($data)

Explanation:

This function is used to set the format of profile picture whether it is in format jpg, jpeg, gif or png. Besides, it also creates a path for users to go to edit profile page to edit and update their profile. This function also retrieves email, and username from sign-in and sign-up process to ensure reliable data, and it is synchronised when editing the personal information of users. The limitation size of files also included in this function to prevent any larger size files from the image display. In addition, updating process in this function will update student information such as phonenumber, emailaddress, name, gender, race, education, city, country, course, date of birth, bio, and image of student profile.

Models: Page.php

1) $this- >pageModel->updateStudentProfile($data)

   This function will update the personal information of students also after clicking the update button it will be redirected to the edit profile again to see whether the information successfully updated or not.

Views: edit_profile.php

2. **Create Client Profile**

   This page contains the profile information of the client/partner. Each client/partner can have the access to their respective page.

   Controllers: Pages.php>edit_client()

   1. edit_client()

      Attribute: $allowed, $filename, $filetype, $filesize, $fileExt, $fileActualExt, $ext, $username, $maxsize, $location, $data, $studentProfile

Explanation:

This function contains the formatting of profile picture of client in jpg, jpeg, gif or png format. It is also the process of retrieving email and username from sign-in and sign-up process to ensure reliable and correct information and also synchronise when editing the personal information process of users. Attributes such as email, companyName, companyDescription, city, country, officerNum and pr_image will integrate with the personal information of client and be updated based on what the client inputs in the form.

Models: Page.php

1) $this->pageModel->updateClientProfile($data)

   Collection of client data will be updated by using this function and it will update the client's profile when the client clicks the update button on the edit profile page and they will see the changes when they are redirected to the page to verify the information they fill successfully or not.

Views: edit_client.php

1.3 Module Event

Person in charge: Low Ying Xi

1. **Main Page of Event**

   This page will show the data related to the event. All users can access this page.

   Controllers: -

   Models: Event.php

   1) countEvent()

   This function queries the 'events' table to retrieve all rows and then execute the query. It will obtain the row count using the ‘rowCount’ method and return the total number of events.

   2) countRegister()

   This function queries the 'registration' table to retrieve all rows, executes the query, and calculates the row count using rowCount. It then returns the total number of registrations.

   3) countEventByCategory(($category)

   This function is used to count events based on a specified category. The function retrieves the count of events in the specified category using the single method and returns this count.

   Views: mainpage_event.php

2. **Create Event**

   Administrator can create events in the system that are open for students to join. This will provide the event details for users to view, and the details will be stored in the database.

   Controllers:Events.php>create()

   1) create()

      Attribute: $data, $filepath, $invitationData, $uploadDir, $fileName, $targetFIlePath, $event_id

Function used:

i) $this->eventModel->createEvent($data)

ii) $this->eventModel->createInvitation($invitationData)

Explanation:

This create() function is a public function that will handle the creation of events in the Youth Venture Data Management System. The administrator can either upload or not upload the image for the event. All the event-related data must be filled out; this includes the event name, category, venue, date, time, event description, and feedback form link.

Models: Event.php

1) createEvent($data)

This function is used to insert a new event record into the 'events' table in the database. The function constructs a SQL query that includes various event-related attributes such as event name, category, description, date, time, venue, user ID, feedback, and file path. The query structure is designed to insert values into the respective columns of the 'events' table. The function then uses the bind method to associate the actual values from the $data with the placeholders in the SQL query. The function then executes the SQL query. If the execution is successful, the function returns the ID of the last inserted row using the ‘lastInsertId’ function in Database.php. This ID is essential for linking the event record with other related entities in the system. On the other hand, if there is an issue during the execution of the query, the function returns false to indicate that the event creation process failed.

2) createInvitation($invitationData)

This function is used to insert a collaborator invitation record into the database. The function will record the invitation of partners/clients associated with an event in the ‘invitation’ table. The data includes the event ID, client ID, and invitation date. If the query execution is successful, the function returns true, indicating that the invitation record was created. Conversely, if there is an issue during the execution of the query, the function returns false.

Views: create.php

3. **Edit Event**

Same interface design as ‘Create Event’ but with the previous data of the event fetched from the database for editing purposes. If there is any update on the details, the function will process and validate the data, then update the new data in the database.

Controllers: Events.php>edit_event($event_id)

1) edit_event($event_id)

   Attribute: $event, $data, $selected_clients, $updateCollaboratorSuccess, $invitationData[], $filepath, $uploadDir, $fileName, $targetFIlePath, $clientsToRemove, $newClients, $found

   Function used:

   i) $this->eventModel->findEventById($event_id)) *used multiple times

   ii) $this->eventModel->findSelectedClientsById($event_id)

   iii) $this->eventModel->deleteInvitations($event_id)

   iv) $this->eventModel->editEvent($data)

   v) $this->eventModel->createInvitation($invitation_Data)

   vi) $this->eventModel->deleteInvitationClient($event_id, $clientToRemove)

Explanation:

This function will obtain the existing event details and the selected collaborators associated with the event. These details are essential for pre-filling the edit form and handling collaborator-related operations. $data will initialise various attributes related to the event, including event details, collaborator information, and potential errors during form submission. Besides, $invitationData is defined to hold information about invitations. The function checks if the form has been submitted. If so, it processes the form data, including handling file uploads, validating form fields, and managing collaborators. The logic includes checking for changes in event attributes, validating form fields, and handling collaborators based on different scenarios:

A) If 'No collaborator' is selected and collaborators are chosen simultaneously, an error message is displayed.

B) If 'No collaborator' option is selected and no collaborators' option is chosen, existing collaborators are deleted, and the event is updated.
C) If 'No collaborator' option is not selected and collaborators are chosen, collaborators are added or removed based on changes.

Models: Event.php

1) findEventById($event_id)
The function will retrieve detailed information about a specific event based on the event ID.

2) findSelectedClientsById($event_id)
The function is used to retrieve the client ID associated with a specific event. This is to make sure that the interface will display the previous data stored accurately based on the client ID found.

3) createInvitation($invitation_Data)
Refer to the same function used in ‘Create Event’.

4) deleteInvitations($event_id)
This function will remove all the invitations to collaborators for a given event based on the event ID. If successful, the function will return true. Otherwise, it will return false.

5) deleteInvitationClient($event_id, $clientToRemove)
This function will remove an invitation to a specific client related to a particular event based on the event ID and client ID. If successful, the function will return true. Otherwise, it will return false.

6) editEvent($data)
This function is used to modify the details of an existing event in the database. If successful, the function will return true. Otherwise, it will return false.

Views: edit_event.php

4. Delete Event

Administrator can delete the event. The alert message will appear before deleting the message.

Controllers: Events.php>delete_event()

1) delete_event()

Attribute: $event, $data, $invitationData[]

Function used:

i) $this->eventModel->findEventById($event_id)

ii) $this->eventModel->findInvitationById($event_id)

iii)$this->eventModel->deleteEvent($event_id)

iv) $this->eventModel->deleteInvitations($event_id)

Explanation:

This function will handle the deletion of a specific event and its associated invitations within the application. The function will interact with two functions to obtain details about the event and its associated invitations. After receiving the request from the view, the function will delete the data from the ‘events’ and ‘invitation’ tables. If both deletion operations are successful, it redirects the user to the events page. Otherwise, it terminates the script with a "Something went wrong" message.

Models: Event.php

1) findEventById($event_id)

Refer to the same function used in ‘Edit Event'.

2) findInvitationById($event_id)

This function will retrieve all the invitations associated with a specific event based on the event ID.

3) deleteEvent($event_id)

This function is used to delete a specific event from the database based on the event ID. If successful, it will return true. Otherwise, it will return false.

4) deleteInvitations($event_id)

Refer to the same function used in ‘Edit Event’.

Views: manage_eventlist.php (button delete)

5. Event List

The event list will display the list of all the events in the system. This page allows administrator to access functionalities such as create, edit, delete, view, and view participants. Meanwhile, this page is accessible to clients/partners with functionalities such as view and view participant.

Controllers: Events.php>index()

1) index()

   Attribute: $data, $events

   Function used:

   i) $this->eventModel->manageAllEvents();

Explanation:

This function will retrieve a list of events and render them to the view to display the events on the user interface.

Models: Event.php

1) manageAllEvents()

   This function is used to retrieve a list of events from the database.

Views: mainpage_event.php

6. Event Details

This page displays more detailed information about a specific event that the event list does not include.

Controllers: Events.php>view_event($event_id)

1) view_event($event_id)

Attribute: $data, $event, $selected_clients

Function used:

i) $this->eventModel->findEventById($event_id)

ii) $this->eventModel->findSelectedClientsById($event_id)

Explanation:

This function will retrieve the details of an event and render them to the view to display the event details on the user interface.

Models: Event.php

1) findEventById($event_id)

Refer to the same function used in ‘Edit Event’.

2) findSelectedClientsById($event_id)

Refer to the same function used in ‘Edit Event’.

3) countEventByCategory(($category)

Refer to the same function used in ‘Main Page of Event’.

Views: view_event.php

7. Manage Registration List

This list will display all the events that have the functionality for students to view events, register events, and provide feedback on the event.

Controllers: Events.php>index()

Same as ‘Event List’.

Models: Event.php

Same as ‘Event list’

Views: manage_registrationlist.php

8. Event Registration

This is the function where students can register for an event. The form will be prefilled with the student's basic information that exists in the profile. For first-time registration students, there are some fields that they need to fill out. But for students who have registered for events before, their most recent data for registering an event will be retrieved and displayed on the registration form for the events they haven't joined. This is to reduce the time taken by students to fill out the form. Every time a student would like to join an event, they can still update on the prefilled data except the personal information. The form cannot be submitted when the student does not accept the agreement question. This can ensure students submit accurate data and confirm their participation to an event

Controllers: Events.php>register_event($event_id)

1) register_event($event_id)

   Attribute: $data, $user_id, $student, $profile_id, $event, $latest_data_register, $option_skill, $option_software_skill, $selectedSkill, $selectedSoftSkill, $skillData1, $skillData2, $skillsToRemove, $found, $softSkillsToRemove, $newSoftSkills, $newSkills

   Function used:

   i) $this->eventModel->fetchAllSkill()
   ii) $this->eventModel->fetchAllSoftwareSkill()
   iii) $this->eventModel->findEventById($event_id)
   iv) $this->eventModel->findStudentById($user_id)
   v) $this->eventModel->findRegisterInfo($profile_id)
   vi) $this->eventModel->findSelectedSkillById($profile_id)
   vii) $this->eventModel->findSelectedSoftSkillById($profile_id)
   viii) $this->eventModel->addSkills($skillData1)
   x) $this->eventModel->addSoftwareSkills($skillData2)
   xi) $this->eventModel->createRegisterInfo($data)
   xii) $this->eventModel->checkRegisterStatus($profile_id, $event_id)

Explanation:

This function is used to manage the registration process for the events. It handles form submissions, updates the data array with form values, and performs basic field validation. Besides, the function distinguishes between a first-time registration and updates to existing registrations. The function also handles the addition, removal, and update of skills and soft skills. The function will eventually prepare data to be passed to the view and then load the corresponding view with the prepared data.

Models: Event.php

1) fetchAllSkill()

This function is used to fetch all records from the 'option_skill' table in the database. It retrieves information related to different skills as options for the question about skills.

2) fetchAllSoftwareSkill()

This function is used to fetch all records from the 'option_software_skill' table in the database. It retrieves information related to different software skills as options for the question about software skills.

3) findEventById($event_id)

Refer to the same function used in ‘Edit Event’

4) findStudentById($user_id)

This function is used to fetch student profile information based on user ID. In this case, it is used to obtain the information of the user who is in session to prefill out the personal information part of the registration form.

5) findRegisterInfo($profile_id)

This function is used to fetch the latest registration details for a specific user profile. This can ensure every time a user can have a prefilled form with the latest information, it will reduce the time spent filling out the form again.

6) findSelectedSkillById($profile_id)

This function is used to retrieve the skill IDs associated with a specific user profile ID.

7) findSelectedSoftSkillById($profile_id)
This function is used to retrieve the software skill IDs associated with a specific user profile ID.

8) addSkills($skillData)
This function is used to add a new skill to the 'skill' table in the database. The skill ID and profile ID will be inserted into the table to record each skill is owned by which user based on their profile ID. If successfully added, the function will return true. Otherwise, it returns false.

9) addSoftwareSkills($skillData)
This function is used to add a new software skill to the 'softwareskill' table in the database. The software skill ID and profile ID will be inserted into the table to record each software skill is owned by which user based on their profile ID. If successfully added, the function will return true. Otherwise, it returns false.

10) createRegisterInfo($data)
This function is used to insert registration information into the 'registration' table. The function returns true if the execution of the query is successful, indicating that the registration information was inserted into the database. If the execution fails, it returns false.

11) checkRegisterStatus($profile_id, $event_id)
This function is used to check whether a user with a specific profile ID is registered for a particular event with a given event ID. If a row is found, it means that the user is registered for the specified event, and the function returns true. If no row is found, it means the user is not registered, and the function returns false.

Views: register_event.php

9. Participant List

The participant list will display all the participants in a specific event, along with some details about each participant. Administrator and clients can view the participant list and view more about the details of a specific participant from the page.

Controllers: Events.php>participant_list()

1) participant_list()

   Attribute: $data, $event, $registers

   Function used:

   i) $this->eventModel->findEventById($event_id)

   ii)$this->eventModel->findRegisterInfoByEventId($event_id);

Explanation:

This function is used to display a list of participants for a specific event. It retrieves relevant data, including the participant information, and then loads a view to display the participant list.

Models: Event.php

1) findEventById($event_id)

   Refer to the same function used in 'Edit Event'.

2) findRegisterInfoByEventId($event_id);

   This function is used to find and return all the registration information for a specific event based on the event ID.

3)findStudentByProfileId($profile_id)

   This function is used to retrieve a specific student from the 'student' table based on their profile ID. In this case, it is used in a loop to display each participant's personal information on the list.

Views: participant_list.php

10. Participant details

This page will display the participant details in a more detailed way compared to the participant list. It can be accessed through the view button in the participant list. Only administrator and clients can view this information.

Controllers: Events.php>view_more($event_id, $profile_id)

1) view_more($event_id, $profile_id)

   Attribute: $data, $student, $stuInEveRegister, $selectedSkill, $selectedSoftSkill

   Function used:
   i) $this->eventModel->findStudentByProfileId($profile_id)
   ii)$this->eventModel->checkRegisterByStuEve($profile_id, $event_id)
   iii)$this->eventModel->findSelectedSkillById($profile_id)
   $this->eventModel->findSelectedSoftSkillById($profile_id)

Explanation:
This function will retrieve the details of a participant and render them to the view to display the participant details on the user interface.

Models: Event.php

   1) findStudentByProfileId($profile_id)
   Refer to the same function used in ‘Participant List’.

   2) checkRegisterByStuEve($profile_id, $event_id)
   This function is used to check whether a registration exists for a specific student and event in the registration table. If a registration is found, it returns the registration information; otherwise, it returns false.

   3) findSelectedSkillById($profile_id)
   Refer to the same function used in ‘Event Registration’

   4) findSelectedSoftSkillById($profile_id)
   Refer to the same function used in ‘Event Registration’

Views: view_more.php

11. Feedback

Feedback of the event is collected by inserting a link during the creation of the event to allow the students to provide feedback through the link. Students only need to click on the button 'Feedback' at each event to provide feedback at the site accessed through the button.

12. View 'My Event'

Students can view the list of the events they have joined.

Controllers: Events.php>my_event()

1) my_event()

Attribute: $data, $user_id, $registers, $student, $profile_id

Function used:

i) $this->eventModel->findStudentById($user_id)

ii) $this->eventModel->findStuRegisterInfo($profile_id)

Explanation:

This function will retrieve the events joined by a participant and render them to the view to display the event details on the user interface.

Models: Event.php

1) findStudentById($user_id)

Refer to the same function used in 'Event Registration'.

2) findStuRegisterInfo($profile_id)

This function is used to retrieve all registration information for a specific student from the registration table based on their profile ID

Views: my_event.php

1.4 Module Resume

Person in charge: Thevan Raju

1. Manage Certification page

This page displays all certifications associated with the currently logged-in student and provides create, edit, and delete buttons for each certification shown.

Controllers:Certifications.php>index()

1) index()

Attribute:$student, $certifications, $data

Function used:

$this->certificationModel->findAllCertifications($_SESSION['email'])

Explanation:

This index() function will retrieve a list of certifications from the database that is associated with the currently logged-in student and render them to the view to display the certifications on the user interface.

Models: Certification.php

1. `findAllCertifications($userEmail])`: This function retrieves all certifications associated with a specific email stored in the session variable. It constructs a SQL query to select all columns from the `certification` table where the `email` matches the provided `$userEmail`. It binds the `$userEmail` parameter to the query using parameterized binding to ensure data integrity and security. Upon executing the query, it fetches all rows from the result set, representing the certifications associated with the given email. Finally, it returns these certifications as an array of certification records.

2. Create Certification

Students can create certifications in the system, which will be stored in the database.

Controllers:Certifications.php>create()

1) create()

Attribute: $student,$profile_id, $data

Function used:

i) $this->certificationModel->studentProfile()

ii) $this->certificationModel->addCertification($data)

Explanation:

This create() function is a public function that will handle the creation of certifications in the Youth Venture Data Management System. The student should fill out the certification name and valid date of certification.

Models: Certification.php

1) studentProfile() - This function retrieves student profile information based on the email stored in the session variable. The query selects all columns from the "student" table where the email matches a parameterized value. After preparing the query, it binds the parameter :email to the value stored in the $_SESSION['email'] variable. Parameter binding helps prevent SQL injection by separating SQL code from user input. The query is executed using $this->db->resultSet(), which likely fetches multiple rows of results from the database. The function returns the result of the query, which is an array containing all rows from the "student" table where the email matches the value stored in the session variable $_SESSION['email'].
2) addCertification($data)- This function is responsible for adding a new certification record to the database. It constructs an SQL query to insert data into the certification table, specifying the columns profile_id, certName, validity, and email. The function binds values to the placeholders in the SQL query using the bind() method to ensure that the data is properly sanitised and prevents SQL injection. It then executes the SQL query using the execute() method of the database class instance. If the query execution is successful (the certification is added to the database), the function returns true. If there's an error during the query execution, it returns false.

Views: create.php

3. Edit Certification

Same interface design as 'Create Certification' but with the previous data of the certification fetched from the database for editing purposes. If there is any update on the details, the function will process and validate the data, then update the new data in the database.

Controllers: Certifications.php>update($certification_id)

1) update($certification_id)

   Attribute: $certification, $data

   Function used:

   i) $this->certificationModel->findCertificationById($certification_id)

   ii) $this->certificationModel->updateCertification($data)

Explanation:

This function will obtain the existing certification details. These details are essential for pre-filling the edit form. $data will initialise various attributes related to the certification, including certification name, valid date and potential errors during form submission. The function checks if the form has been submitted. If so, it validates the input fields, and updates the data array with form input values. It checks for validation errors and ensures that at least one field is changed before updating the certification. If there are no errors, it updates the certification in the database. If the update is successful, it redirects the user to the certifications page. If there's an error during the update process, it displays an error message. If there are validation errors, it renders the certifications index view with the error messages.

Models: Certification.php

1) findCertificationById($certification_id) - The findCertificationById() function retrieves a single certification record from the database based on the provided certification ID. It prepares and executes a SQL query to select a certification by its ID. The certification ID is bound to the query using parameterized binding to prevent SQL injection. It fetches a single row from the result set using the single() method of the database class. Finally, it returns the fetched certification record.

2) updateCertification($data) - The updateCertification() function updates a certification record in the database with new data provided in the $data array. It prepares an SQL query to update the certName and validity fields of the certification table for a specific certification ID. The certification ID, certification name, and validity are bound to the query using parameterized binding to prevent SQL injection. The execute() method of the database class is called to execute the update query. If the update query is successful, the function returns true. Otherwise, it returns false.

Views: update.php

4. Delete Certification

Students can delete the certification. The alert message will appear before deleting the certification.

Controllers: Certifications.php>delete($certification_id)

1. delete($certification_id)
   Attribute: $certifications, $data

   Function used:
   i) $this->certificationModel->findCertificationById($certification_id)
   ii) $this->certificationModel->deleteCertification($certification_id)

Explanation:

This function will handle the deletion of a specific certification. It retrieves the certification details based on the provided certification ID using the findCertificationById() method. After receiving the request from the view, the function will delete the data from the ‘certification’ table.If the deletion is successful, the user is redirected to the certifications page. Otherwise, it terminates the script with a "Something went wrong" message.

Models: Certification.php

1) findCertificationById($certification_id)
Refer to the same function used in ‘Edit Certification'.

2) deleteCertification($certification_id)

This function is used to delete a specific certification from the database based on the certification ID. If successful, it will return true. Otherwise, it will return false.

Views: manage.php (button delete)

5. Manage Experience page

This page displays all experiences associated with the currently logged-in student and provides create, edit, and delete buttons for each certification shown.

Controllers:Experiences.php>index()

1) index()

Attribute: $student, $profile_id, $experiences, $data

Function used:

$this->experienceModel->findAllExperiences($_SESSION['email'])

Explanation:

This index() function will retrieve a list of experiences from the database that is associated with the currently logged-in student and render them to the view to display the experiences on the user interface.

Models: Experience.php

1. `findAllExperiences($userEmail)`: This function retrieves all experiences associated with a specific email stored in the session variable. It constructs a SQL query to select all columns from the `experience` table where the `email` matches the provided `$userEmail`. It binds the `$userEmail` parameter to the query using parameterized binding to ensure data integrity and security. Upon executing the query, it fetches all rows from the result set, representing the experiences associated with the given email. Finally, it returns these experiences as an array of experience records.

6. Create Experience

Students can create experiences in the system, which will be stored in the database.

Controllers:Experiences.php>create()

1) create()
   Attribute: $student,$profile_id, $data

   Function used:
   i) $this->experienceModel->studentProfile()
   ii) $this->experienceModel->addExperience($data)

Explanation:

This create() function is a public function that will handle the creation of experiences in the Youth Venture Data Management System. The student should fill out the job title, company from_date and to_date.

Models: Certification.php

1) studentProfile() - This function retrieves student profile information based on the email stored in the session variable. The query selects all columns from the "student" table where the email matches a parameterized value. After preparing the query, it binds the parameter :email to the value stored in the $_SESSION['email'] variable. Parameter binding helps prevent SQL injection by separating SQL code from user input. The query is executed using $this->db->resultSet(), which likely fetches multiple rows of results from the database. The function returns the result of the query, which is an array containing all rows from the "student" table where the email matches the value stored in the session variable $_SESSION['email'].

2) addExperience($data)- This function is responsible for adding a new experience record to the database. It constructs an SQL query to insert data into the experience table, specifying the columns profile_id, jobtitle, company, from_date, to_date and email. The function binds values to the placeholders in the SQL query using the bind() method to ensure that the

data is properly sanitized and prevents SQL injection. It then executes the SQL query using the execute() method of the database class instance. If the query execution is successful (the experience is added to the database), the function returns true. If there's an error during the query execution, it returns false.

Views: create.php

7. Edit Experience

Same interface design as 'Create Experience' but with the previous data of the experience fetched from the database for editing purposes. If there is any update on the details, the function will process and validate the data, then update the new data in the database.

Controllers: Experiences.php>update($experience_id)

1) update($experience_id)

   Attribute: $experience, $data

   Function used:

   i) $this->experienceModel->findExperienceById($experience_id)

   ii)$this->experienceModel->updateExperience($data)

Explanation:

This function will obtain the existing experience details. These details are essential for pre-filling the edit form. $data will initialize various attributes related to the experience, including job title, company name, from date, to date and potential errors during form submission. The function checks if the form has been submitted. If so, it validates the input fields, and updates the data array with form input values. It checks for validation errors and ensures that at least one field is changed before updating the experience. If there are no errors, it updates the experience in the database. If the update is successful, it redirects the user to the experiences page. If there's an error during the update process, it displays an error message. If there are validation errors, it renders the experiences index view with the error messages.

Models: Experience.php

3) findExperienceById($experience_id) - The findExperienceById() function retrieves a single experience record from the database based on the provided experience ID. It prepares and executes a SQL query to select an experience by its ID. The experience ID is bound to the query using parameterized binding to prevent SQL injection. It fetches a single row from the result set using the single() method of the database class. Finally, it returns the fetched experience record.
4) updateExperience($data) - The updateExperience() function updates an experience record in the database with new data provided in the $data array. It prepares an SQL query to update the jobtitle, company, from_date, to_date fields of the experience table for a specific experience ID. The experience ID, jobtitle, company, from_date, to_date are bound to the query using parameterized binding to prevent SQL injection. The execute() method of the database class is called to execute the update query. If the update query is successful, the function returns true. Otherwise, it returns false.

Views: update.php

8) Delete Experience

Students can delete the experience. The alert message will appear before deleting the experience.

Controllers: Experiences.php>delete($experience_id)

5) delete($experience_id)

Attribute: $experiences, $data

Function used:

i) $this->experienceModel->findExperienceById($experience_id)

ii) $this->experienceModel->deleteExperience($experience_id)

Explanation:

This function will handle the deletion of a specific experience. It retrieves the experience details based on the provided experience ID using the findExperienceById() method. After receiving the request from the view, the function will delete the data from the ‘experience’ table. If the deletion is successful, the user is redirected to the experiences page. Otherwise, it terminates the script with a "Something went wrong" message.

Models: Experience.php

1) findExperienceById($experience_id)
Refer to the same function used in ‘Edit Experience’'.

2) deleteExperience($experience_id)
This function is used to delete a specific certification from the database based on the certification ID. If successful, it will return true. Otherwise, it will return false.

Views: manage.php (button delete)

9) View Resume

Students can view their resume which is generated from various data in the system such as personal information, skills, certifications and experiences.

Controllers: Resume.php>edit_resume()

1) edit_resume()
Attribute: $student, $profile_id, $xskills, $softSkills, $certs, $exp, $skills[ ] , $sskills[ ]

Function used:
i)$this->resumeModel->studentProfile()
ii)$this->resumeModel->findSkillById($profile_id)
iii) $this->resumeModel->findSoftSkillById($profile_id)
iv) $this->resumeModel->findCertificationByProfileId($profile_id)
v)$this->resumeModel->findExperienceByProfileId($profile_id)
vi)$this->resume Model->findSelectedSkillOptionById($skill->skill_id)

vii) $this->resumeModel->findSelectedSkillOptionById ($sskill->skill_id)

Explanation:

The `edit_resume` function retrieves the student's profile information using the `studentProfile` method of the `resumeModel`. If the student profile exists, it fetches additional details such as skills, soft skills, certifications, and experiences. These details are organized into an associative array and passed to the view for rendering. The view file (`index.php` in the `resumes` directory) presents the edit form with pre-populated student details. If the student profile is not found, a view indicating that the user was not found is rendered. Overall, `edit_resume` ensures the display of the edit resume page with existing details, allowing students to make modifications as needed.

Models: Experience.php

1) studentProfile()
   Refer to the same function used in ‘Create Experience''.
2) findSkillById($profile_id)
   Retrieves all skills associated with a particular student profile identified by the $profile_id.
3) findSoftSkillById($profile_id)
   Fetches all soft skills associated with a specific student profile identified by the $profile_id.
4) findCertificationByProfileId($profile_id)
   Fetches all certifications belonging to a specific student profile identified by the $profile_id.
5) findExperienceByProfileId($profile_id)
   Retrieves all experiences recorded for a particular student profile identified by the $profile_id.
6) findSelectedSkillOptionById($skill_id)
   Retrieves the details of a selected skill option identified by the $skill_id.
7) findSelectedSoftSkillOptionById($softwareSkill_id)
   Fetches the details of a selected software skill option identified by the $softwareSkill_id.

Views: edit_resume.php

1.5 Module Reward and Badge

Person in charge: Wan Nur Sofea

1. View Badges Collected

This page shows the number of badges collected by students individually. This page can only be accessed by students.

Controllers: Rewards.php>badge()

1) badge()

   Attribute: $user_id, $profile_id, $data, $student, $badge.

   Function used:

   i) $this->rewardModel->findStudentByID($user_id)

   ii) $this->rewardModel->findStudentInReward($profile_id)

Explanation:

This function will retrieve the students' current number of badges collected based on the number of events and categories they are registered into. All three badges; gold, silver and bronze will be displayed individually with their respective numbers.

Models: Reward.php

1) findStudentById(&user_id)

   This function takes parameter of &user_id to find the individual student in the 'student' table. It queries to retrieves all data from 'student' table where the user_id is matched with &user_id. Then it return the respective row of student's information in the $row_profile defined beforehand.

2) findStudentInReward($profile_id)

   This function retrieves the student's badge records in the 'rewardnbadge' table including data of gold badge, silver badges, bronze badges, claimStatus and dateAwarded. It takes the parameter of &profile_id to find the specific row before returning it in a single row of data from the result set.

Views: badge.php

2. Join Badge Race

This page ask students to join the race of collecting badges from joining events hosted by Youth Venture.

Controllers: Rewards.php>join()

1) join()

   Attribute: $user_id, $profile_id, $student, $data

   Functions:

   i) $this->rewardModel->findStudentById($user_id)

   ii) $this->rewardModel->findExist($profile_id)

Explanation:

This function is to assign the initial value of badges to zero. It is also to assign the student info into the 'rewardnbadge' table for administrator use. Students' who already have joined do not have to re-join again.

Models: Reward.php

1) findStudentById(&user_id)

   This function takes parameter of &user_id to find the individual student in the 'student' table. It queries to retrieve all data from 'student' table where the user_id is matched with &user_id. Then it returns the respective row of student's information in the $row_profile defined beforehand.

2) findExist($profile_id)

   This function takes parameter $profile_id to check the existence of this student in the database 'rewardnbadge' table. It will return true if the student exists already and false otherwise.

Views: join.php

3. Manage All List of Badges

This page displays a list of students with the count of badges collected and the reward claimed. Only Youth Venture can access this page.

Controllers: Rewards.php>manage.php

1) manage.php

   Attributes: $badges, $event_ids, $goldBadge, $silverBadge, $bronzeBadge, $category, $update_data, $badge, $data

   Functions:

   i) $this->rewardModel->AmanageAllBadge()

   ii) $this->rewardModel->getRegisteredEvent($profile_id)

   iii) $this->rewardModel->getCategory($event_id->event_id)

   iv) $this->rewardModel->updateBadgeId($update_data, $profile_id)

Explanation:

This function is crucial for keeping track and displaying the most recent the current data from ‘rewardnbadge’ table to administrator. Youth Venture has the authority update the current reward status and date of awarded when the badges collected has reached into a certain threshold.

Models: Reward.php

1) AmanageAllBadge()

   This function will retrieves all data of all students from table ‘student’ and each corresponding badge from table ‘rewardnbadge’. The function performs an SQL query using JOIN operation where data from both tables are combined based in common value, profile_id. It return all result set that have corresponding data. The data that unsuccessfully combine due to lack of common value, will not be return. The attributes that are used are students’s name from ‘student’, and all attributes from ‘rewardnbadge’table

2) getRegisteredEvent($profile_id)

   This function retrieves all event_id from table ‘registration’ based on the matched the parameter, profile_id in the table. It return the array of results including the event_id registered by each respective profile_id.

3) getCategory($event_id->event_id)

   This function retrieves attributes category from ‘events’ table based on the matched parameter of event_id. It returns the a single row of data from the result set.

4) updateBadgeId($update_data, $profile_id)

   The function updates attributes from table ‘rewardnbadge’ based on the matched profile_id. The parameter update_data store the new data that will be inserted into each attributes. It takes the SQL UPDATE query to modify all values of goldBadge, silverBadge, bronzeBadge. Then execution of query is done, if the update is successful, it will return true. Otherwise, it return false.

Views: manage.php

4. Update the Badge

This page is to update the current number of all three badges and the reward for each student. Only Youth Venture can access this page and change any of the number badges accordingly.

Controllers: Rewards.php>update.php

1) update.php

   Attributes: $badge, $data

   Functions:

   i) $this->rewardModel->findRewardById($badge_id)

   $this->rewardModel->updateReward($data)

Explanation:

This function works to update the student’s current number of badges, reward and date awarded. This updation is made manually to give full authority to the Youth Venture to make any changes based on their regulations. They can always update students’ rewards continuously over time. Youth Venture can alter the number of badges from students’ data if there is any error or cases that makes the student unqualified to receive the badge.

Models: Reward.php

1) findRewardById($badge_id)

   This function is to find to retrieve all data from table ‘rewardnbadge’, and name from ‘student’ table where the profile_id in both tables are matched. The row of profile_id from ‘rewardnbadge’ table is based on the respective badge_id passed on in the parameter. The function returns a single row of data from both tables.

2) updateReward($data)

   This function uses query UPDATE to alter the values of goldBadge, silverBadge, bronzeBadge, claimStatus and dateAwarded attributes from table ‘rewardnbadge’. The row of data being updated is based on badge_id in the object parameter. If the execution is successful, the function will return otherwise, it will return false.

Views: update.php

System Credentials (Hosting)

This Youth Venture Data Management System has three users; Youth Venture as admin, partner/client and student.

Users	**Email**	**Password**
Youth Venture	YouthVenture@gmail.com	YVadmin123
Client/Partner	petronas@gmail.com	petronas12345
Student	ali@gmail.com	ali12345

This is the database credential for this system

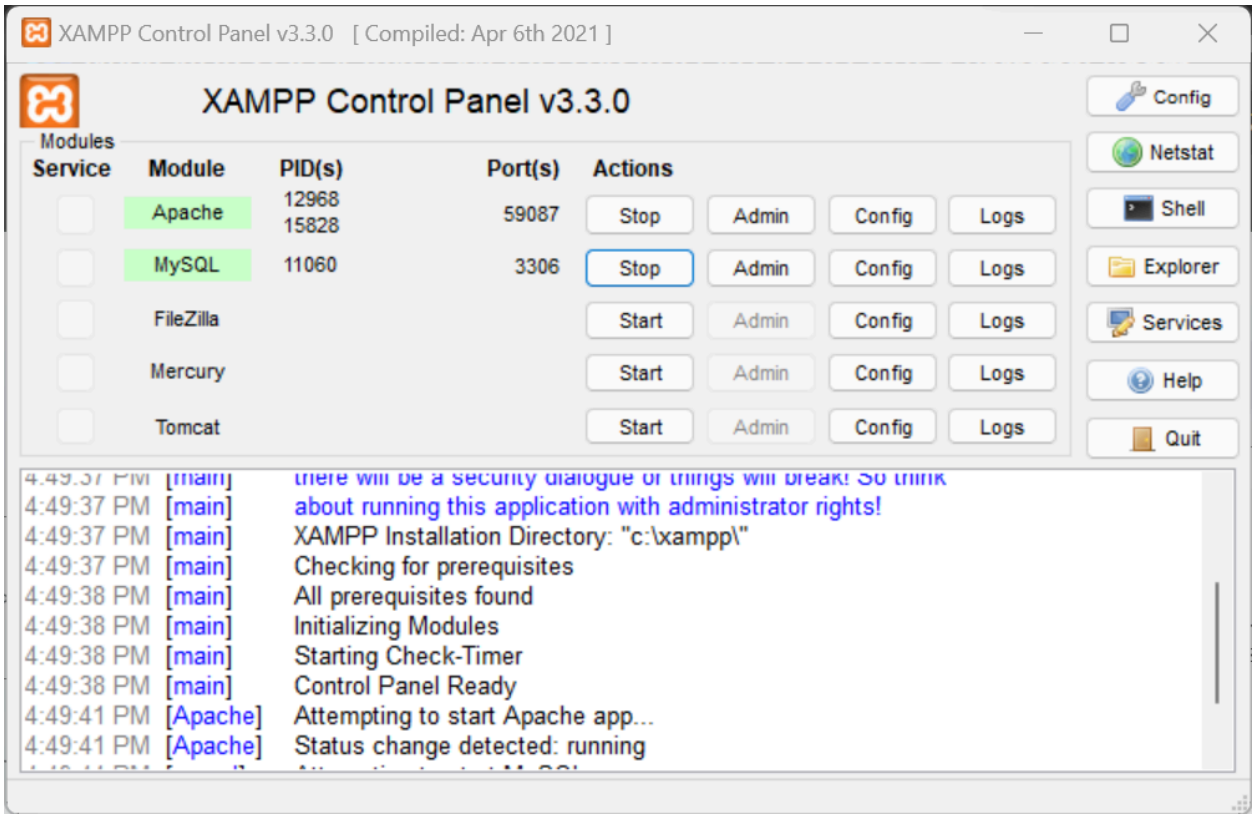
DB_Name	niagaped_Innovatio
DB_User	niagaped_Innovatio_user
DB_pass	Innovatio&**^%$#

Localhost Setup

Step 1: Download and Install XAMPP

Step 2: Start Apache and MySQL

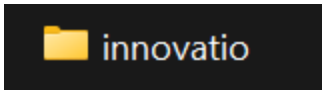
1. After installation, open the XAMPP Control Panel.
2. Start the Apache and MySQL services by clicking the "Start" button next to them.



Step 3: In the XAMPP Control Panel, click on the “Admin” button for MySQL.

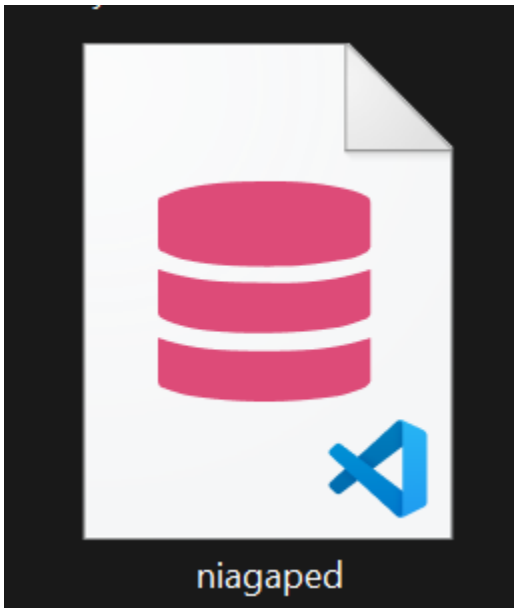
Step 4: Unzip the zip folder

1. After unzipping, move the innovatio folder into “htdocs” folder in the XAMPP installation directory.

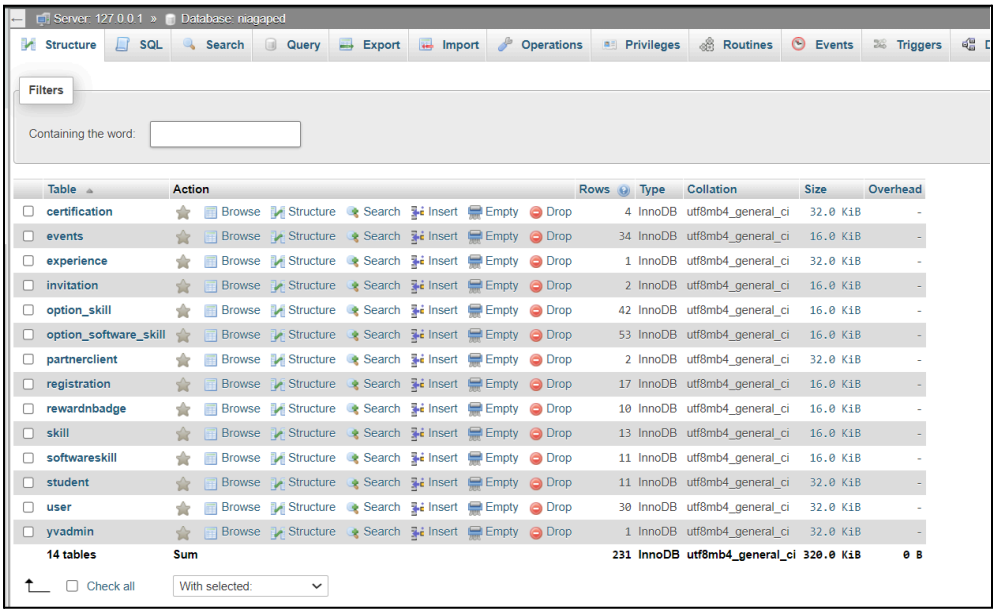


Step 5: Import database

1. In the phpmyadmin, create a database name “niagaped”.
2. Import this “niagaped” database into phpmyadmin.

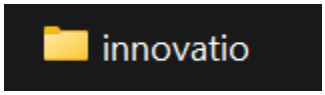


3. The database should look like this.



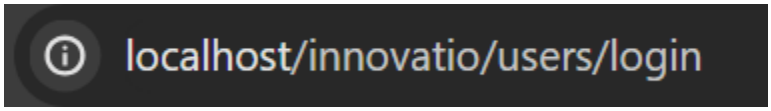
Step 6: Navigate to the ‘innovatio” Folder

1. Navigate to the XAMPP installation directory and find the “htdocs” folder.
2. Navigate to the “innovatio” directory.



Step 7: Navigate to the web browser

1. Type the link localhost/innovatio/users/login



2. The system now can be tested. Enter either the Youth Venture, client/partner or student account based on the credentials provided.

System Interface for All Users

There are three differents systems interfaces for all users:

Module Authentication:

i) Sign In

ii) Sign Up (Only for student and client/partner)

iii) Forgot Password

iv) Setup New Password

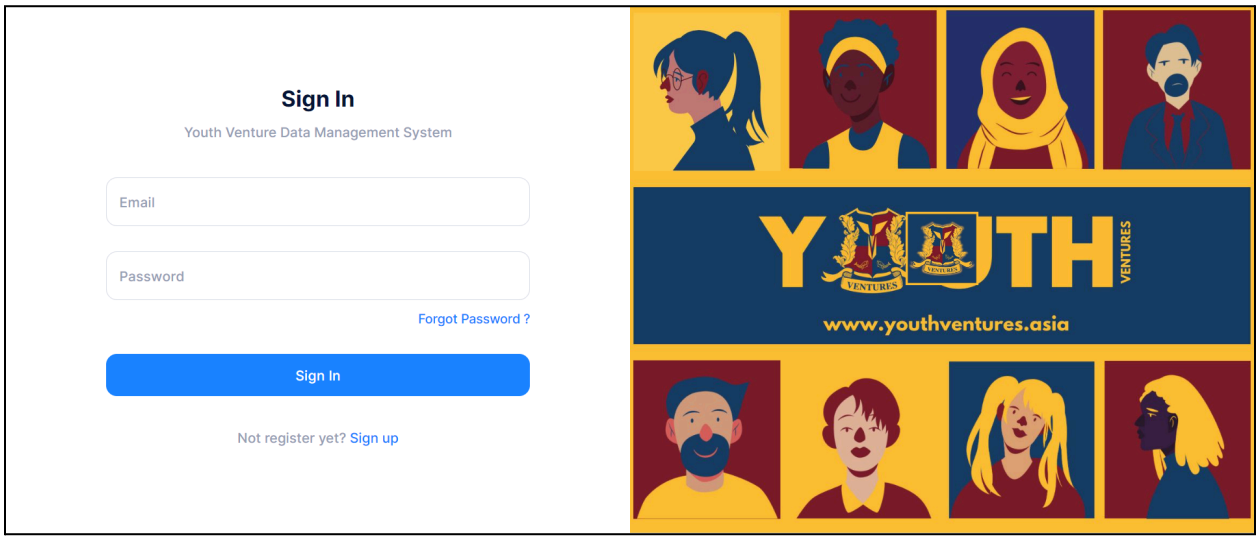


Figure 4.1: Sign In Page

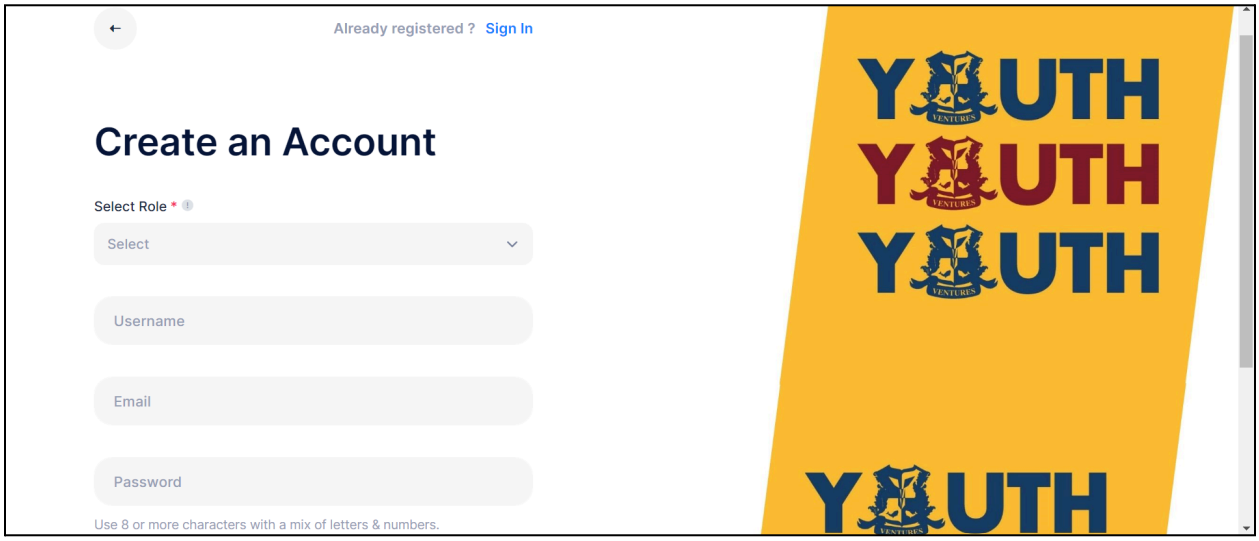


Figure 4.2: Sign In Page

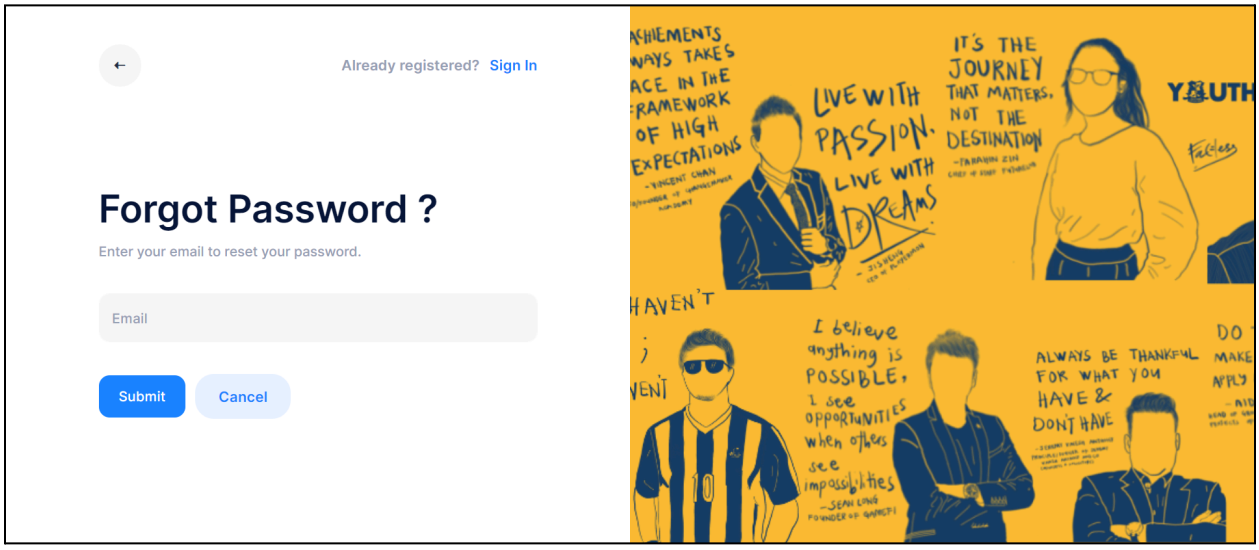


Figure 4.3: Forgot Password Page

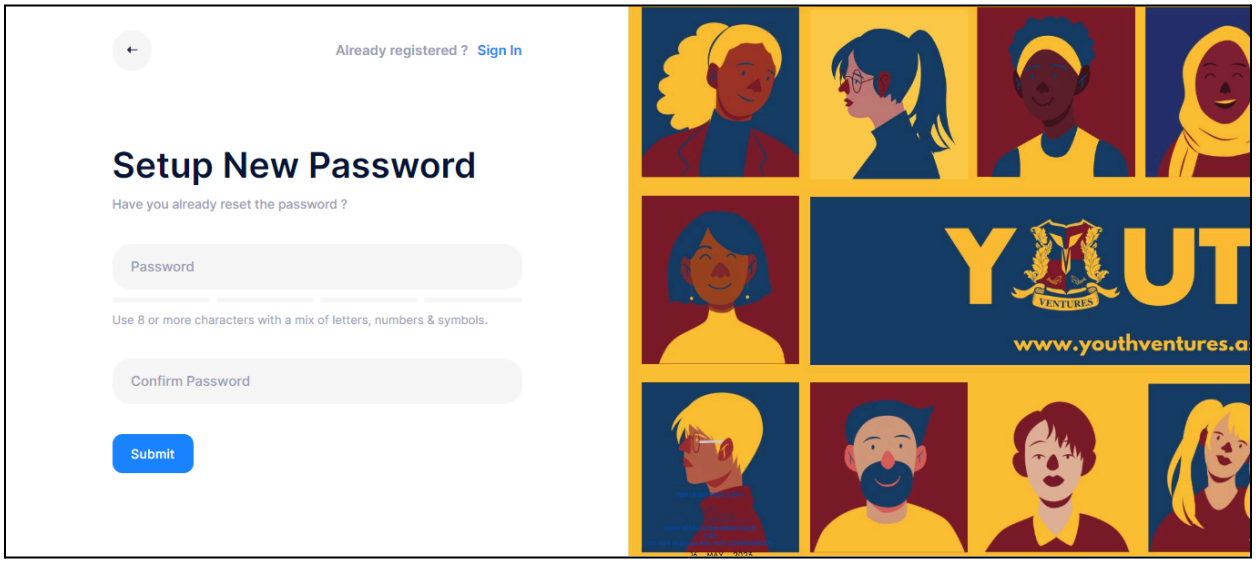


Figure 4.4: Setup New Password Page

Dashboard:

i) Youth Venture

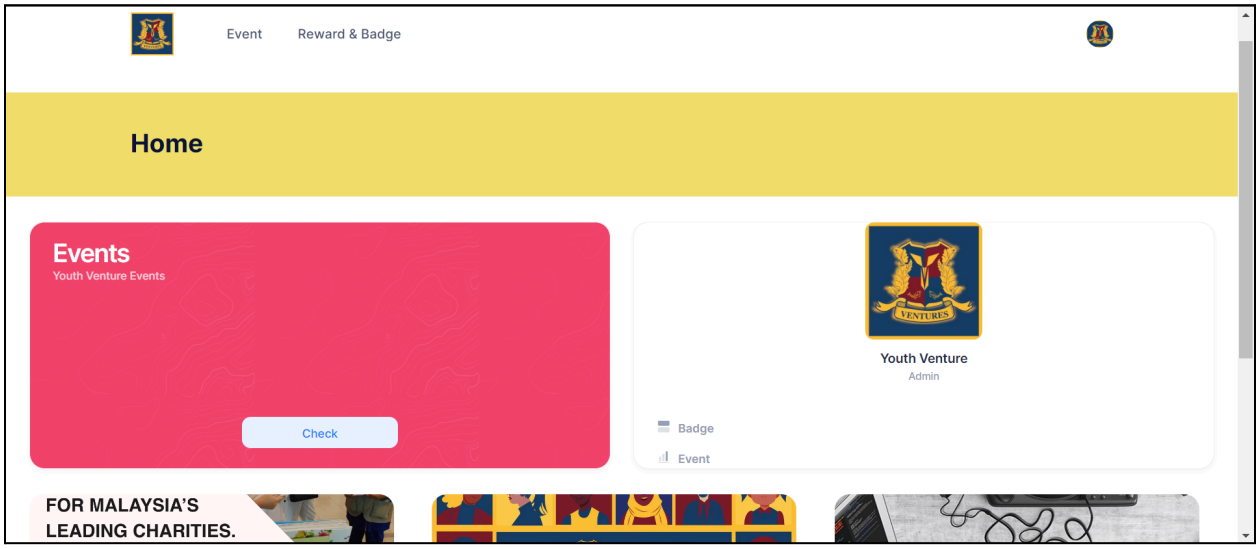


Figure 5.1: Dashboard Youth Venture Page

ii) Client/Partner

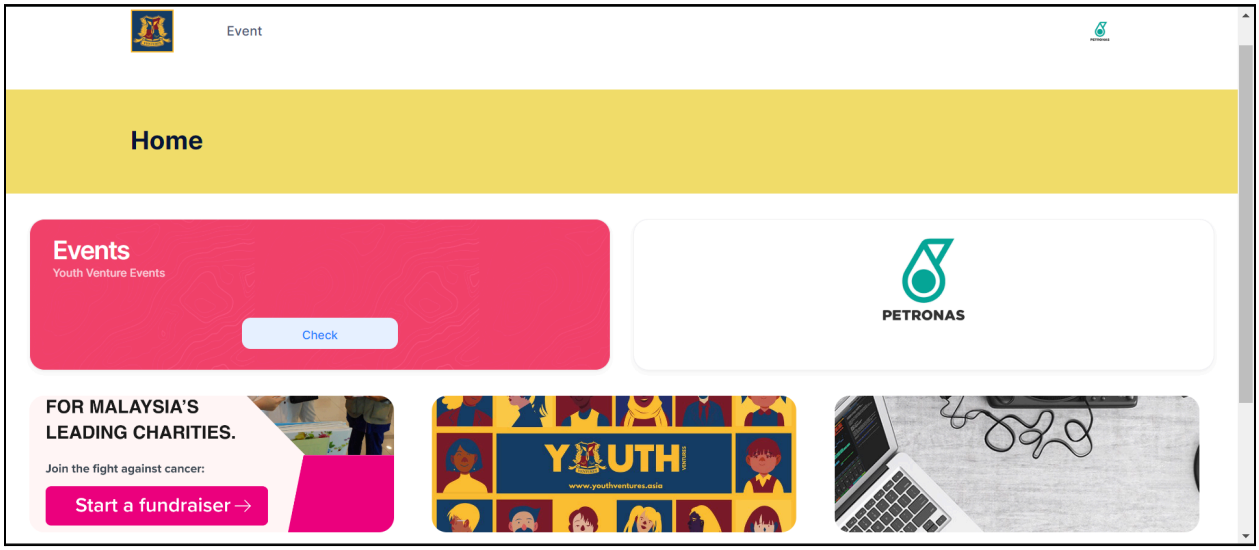


Figure 5.2: Dashboard Client/Partner Page

iii) Student

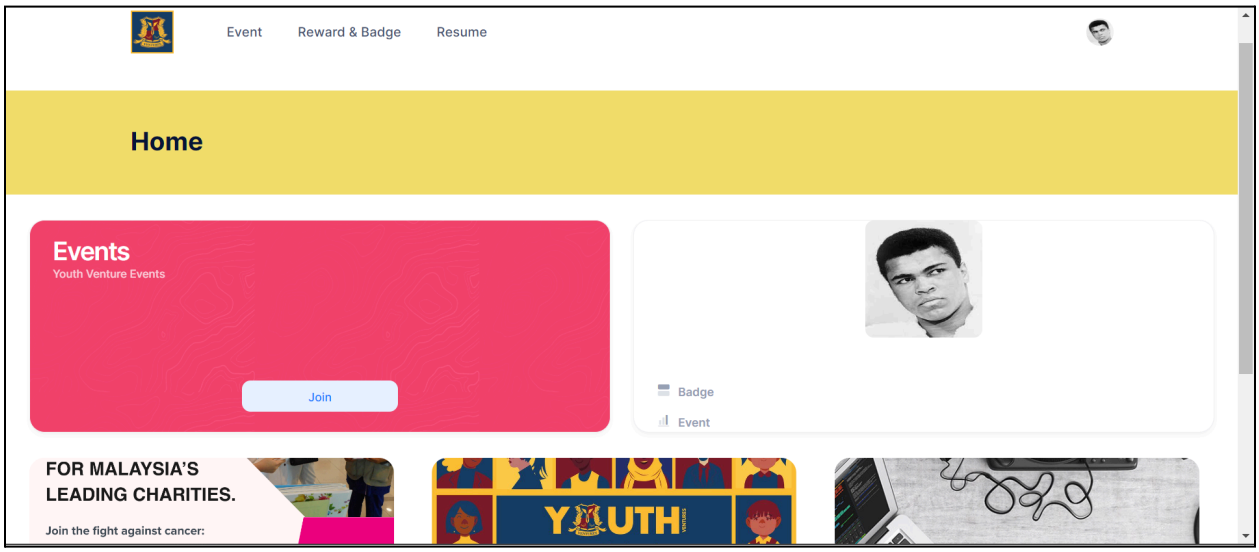


Figure 5.3: Dashboard Student Page

Module Profile:

i) Edit Profile (Only for student and client/partner)

ii) Manage User (Only for Youth Venture) - **Controller Users*

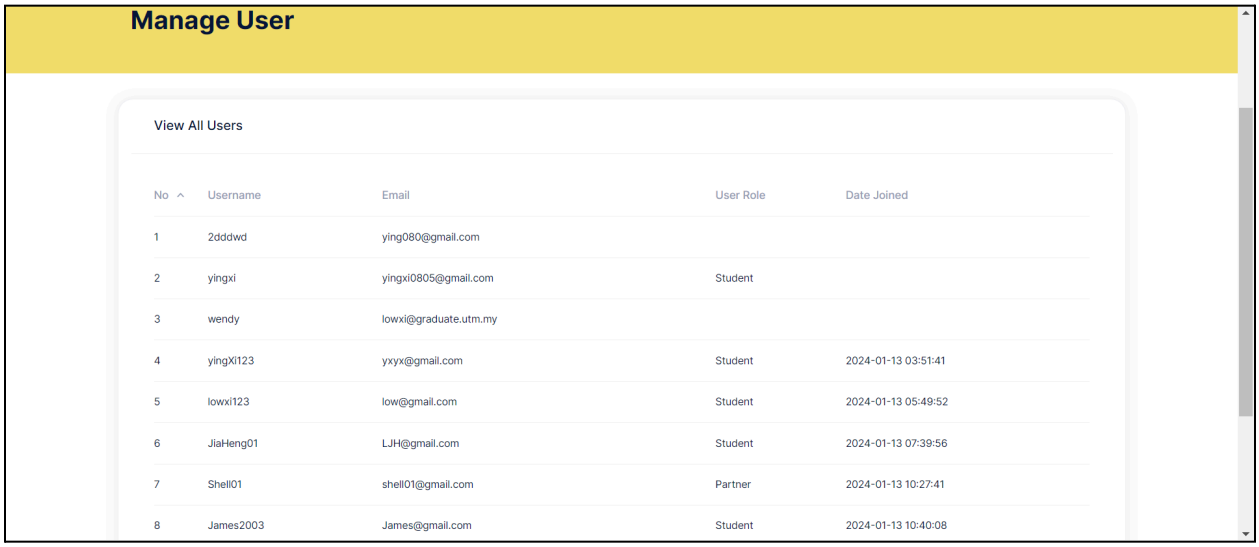


Figure 6.1: Manage User Page

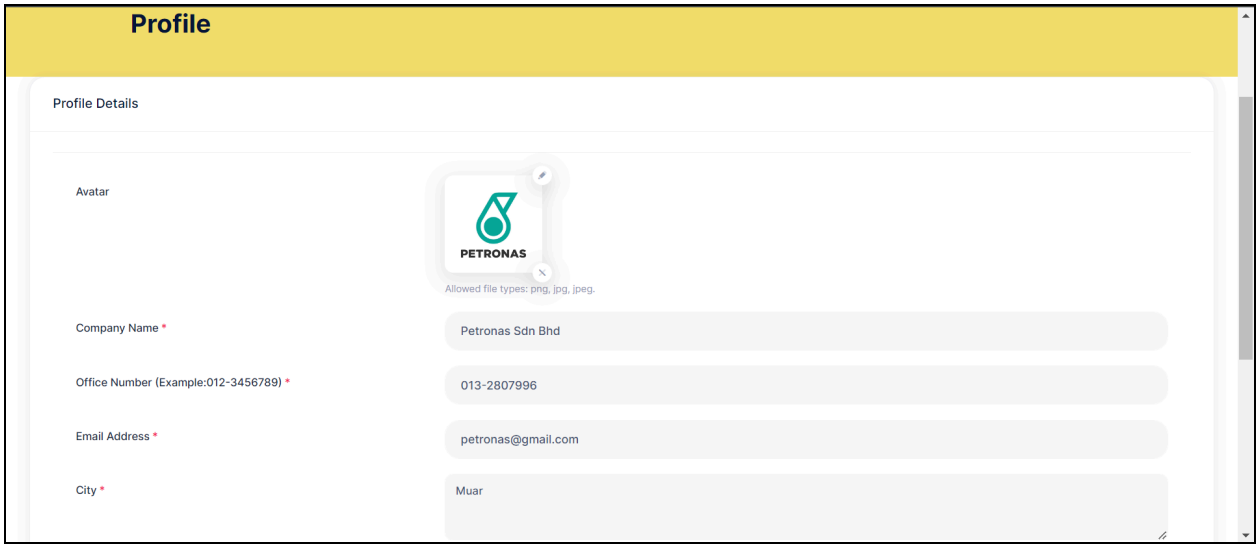


Figure 6.2: Edit Profile Client/Partner Page

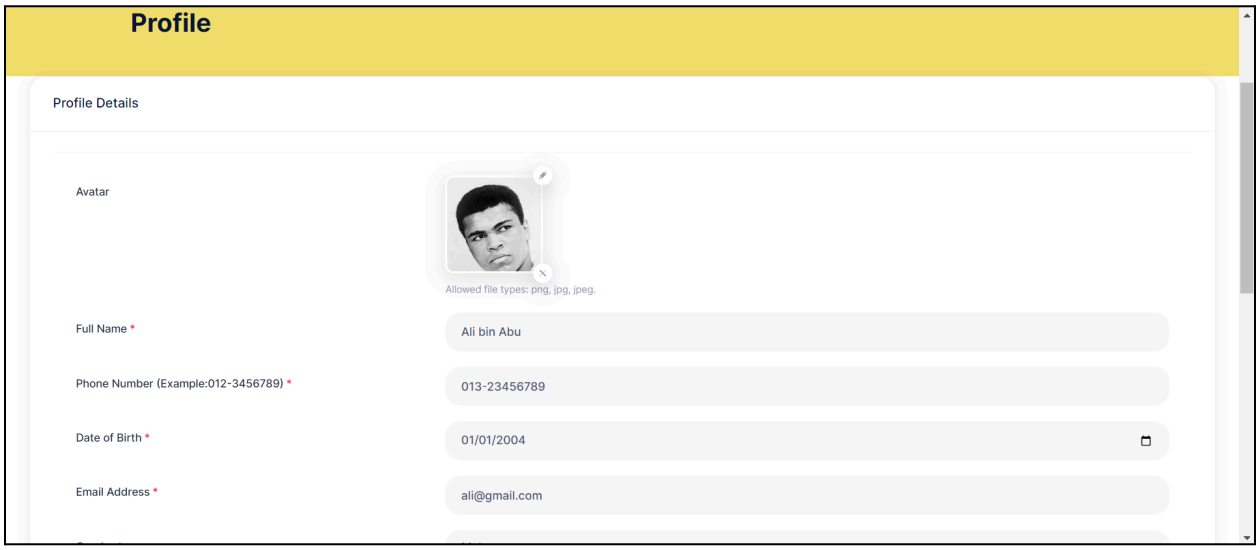


Figure 6.3: Edit Profile Student Page

Module Event:

i) Main Event Page

ii) Manage Registration List (Only for student)

iiii) View Event Details

iv) Registration Form (Only for student)

v) Manage Event List (Only for Youth Venture and client/partner)

vi) View Participant List (Only for Youth Venture and client/partner)

vii) View Participant Details

viii) Create Event (Only for Youth Venture)

x) Edit Event (Only for Youth Venture)

xii) View My Events

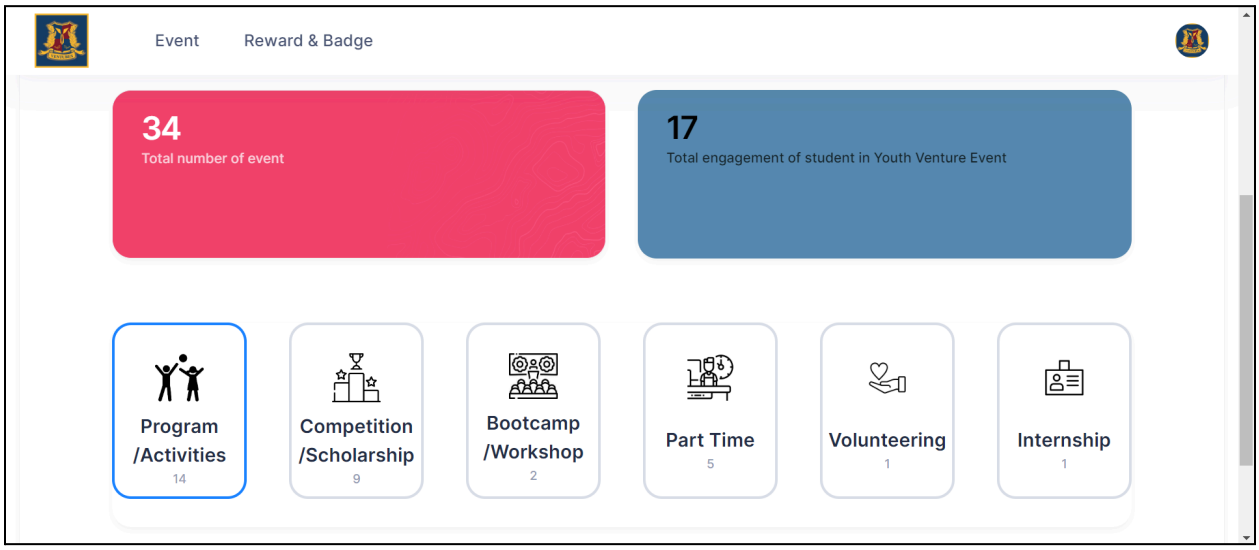


Figure 7.1: Main Event Page

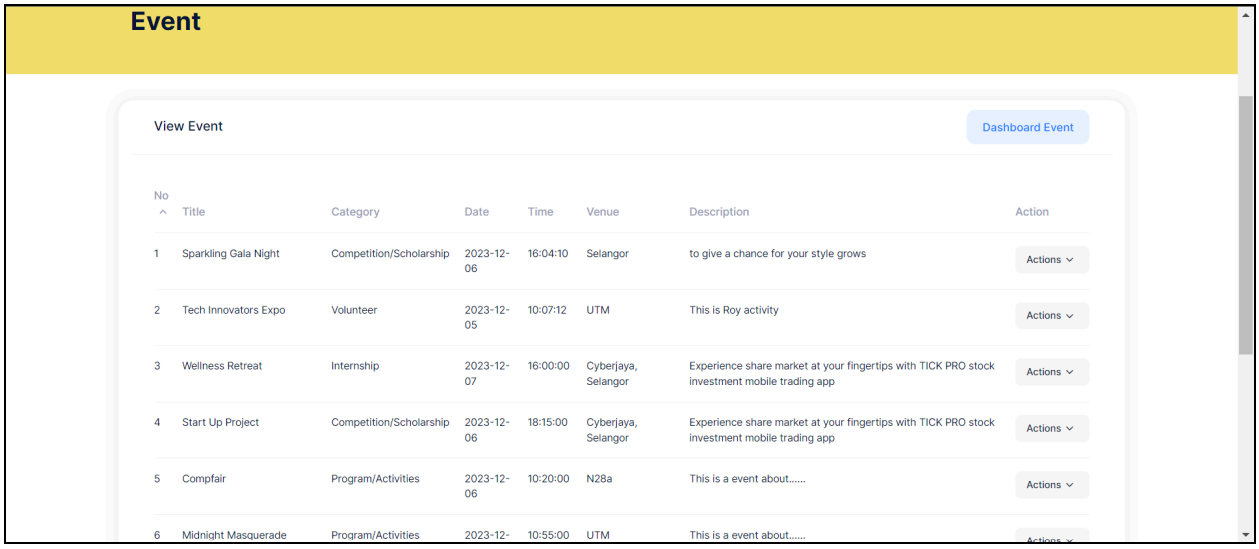


Figure 7.2: Manage Registration List Page

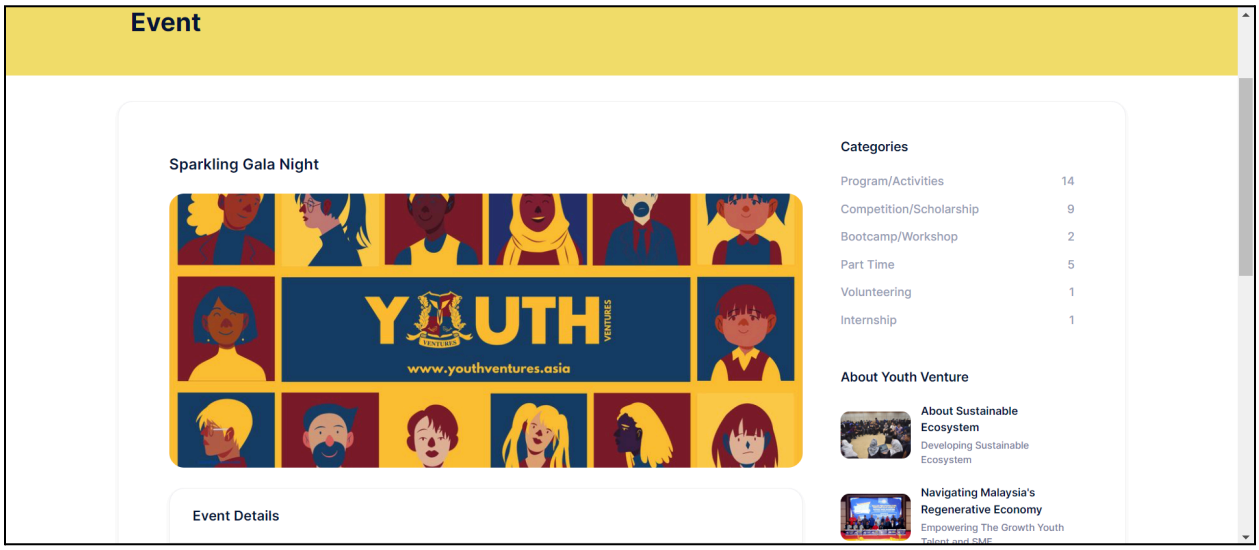


Figure 7.3: View Event Details Page

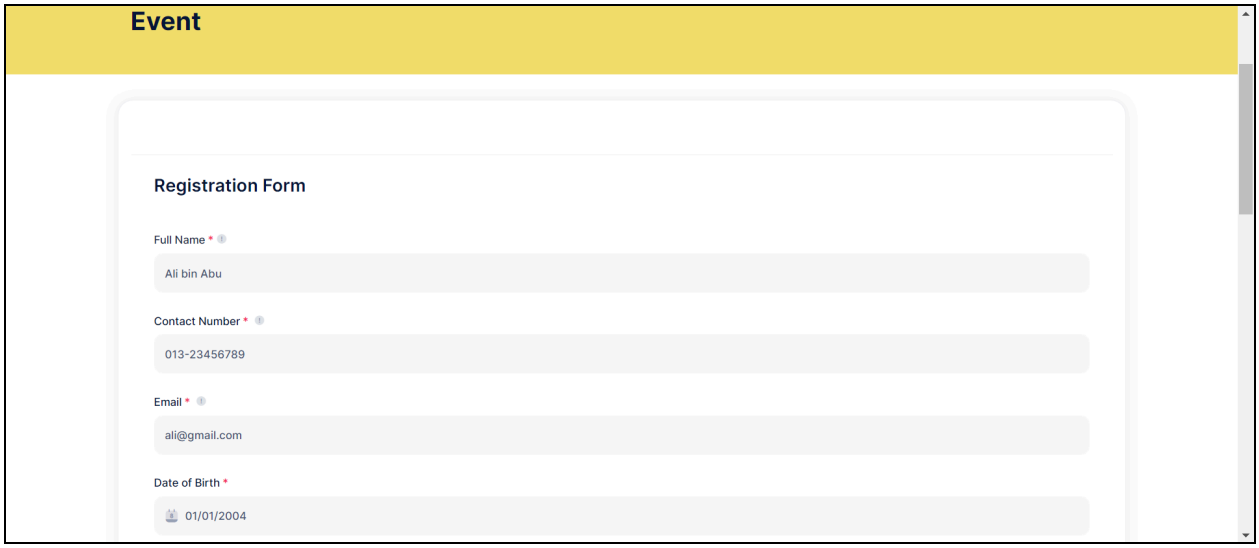


Figure 7.4: Registration Form Page

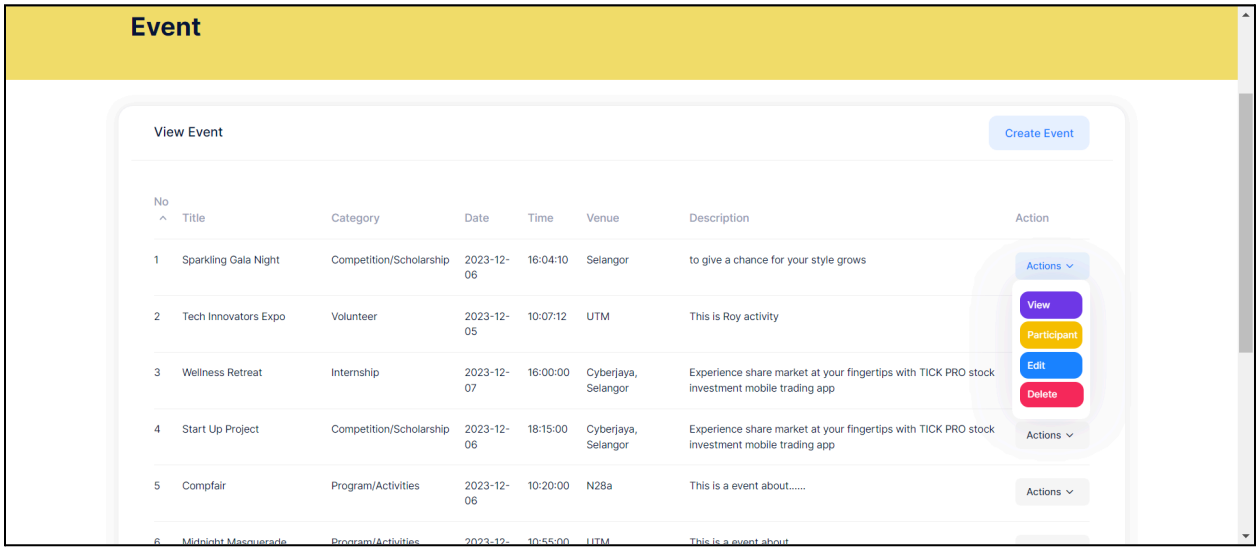


Figure 7.5.1: Manage Event List Youth Venture Page

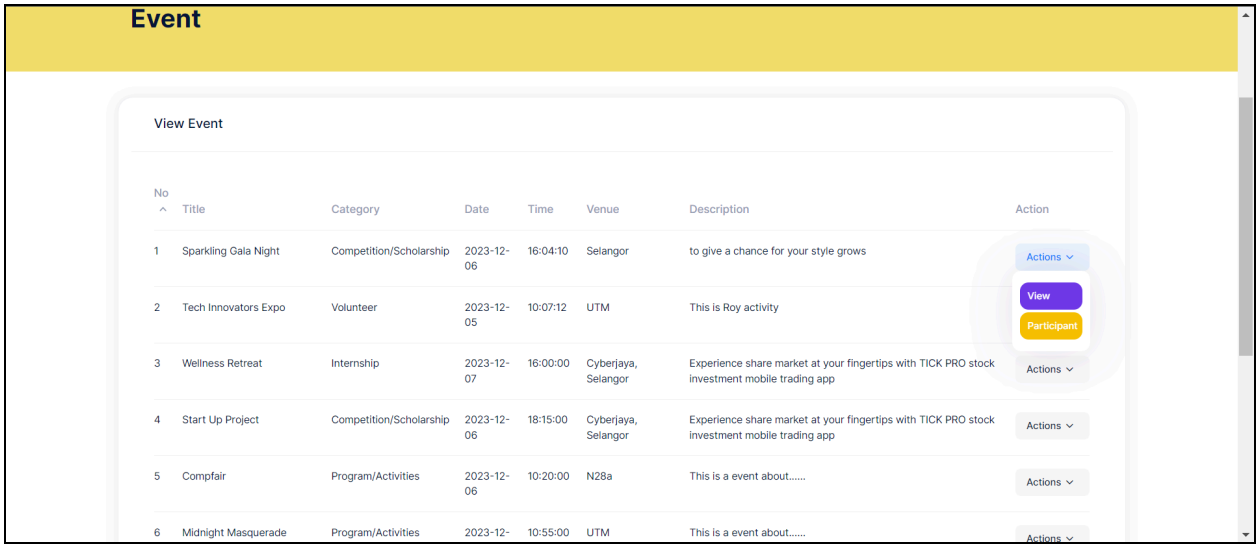


Figure 7.5.2: Manage Event List Client/Partner Page

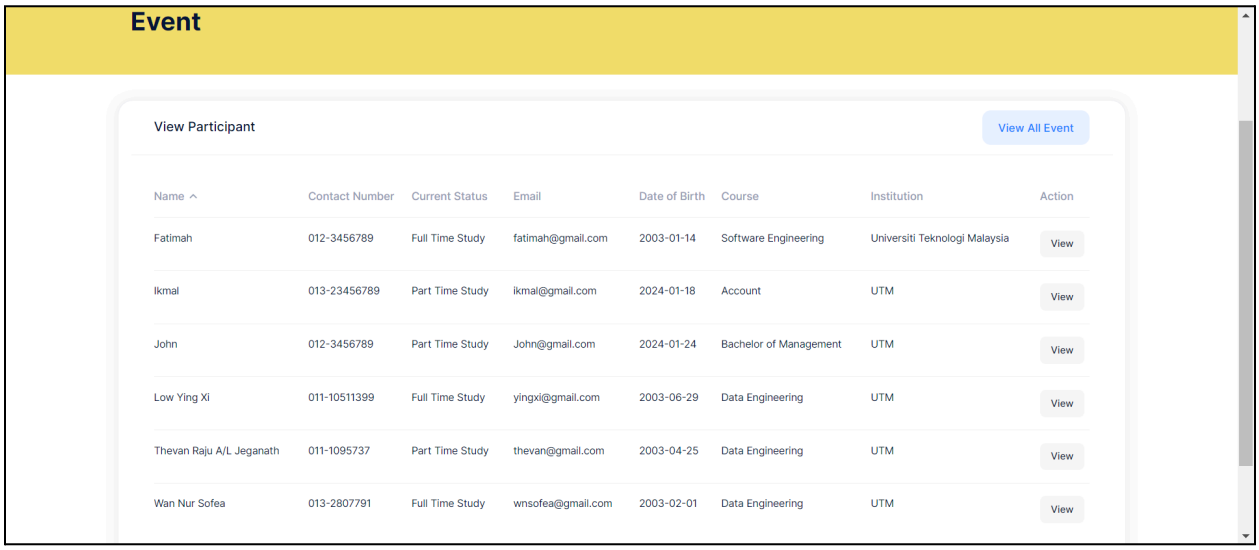


Figure 7.6: View Participant List Page

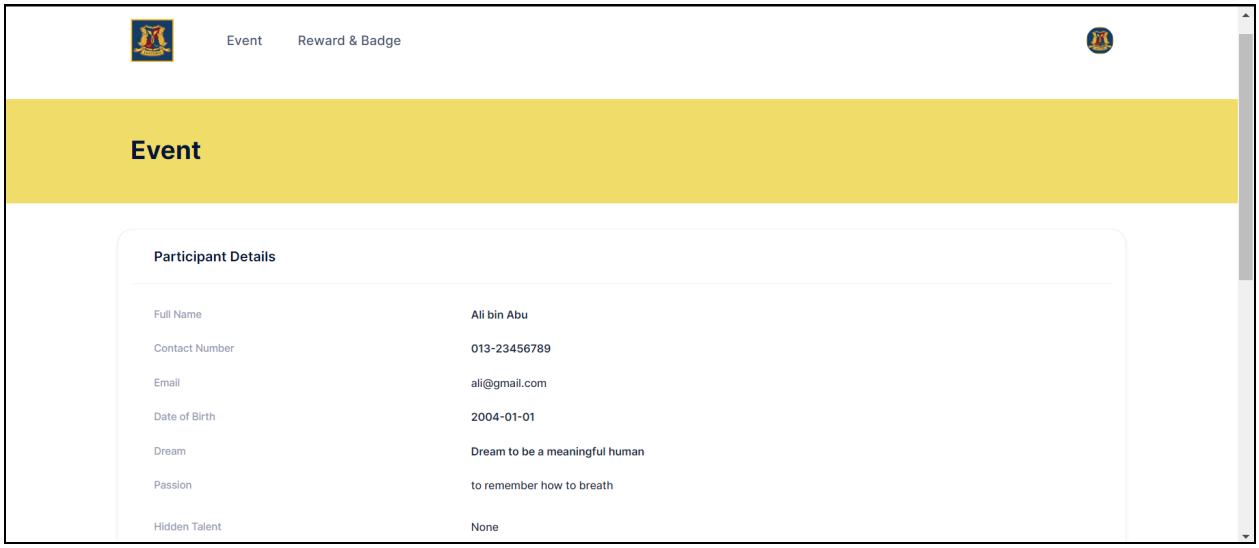


Figure 7.7: View Participant Details Page

Figure 7.8: Create Event Page

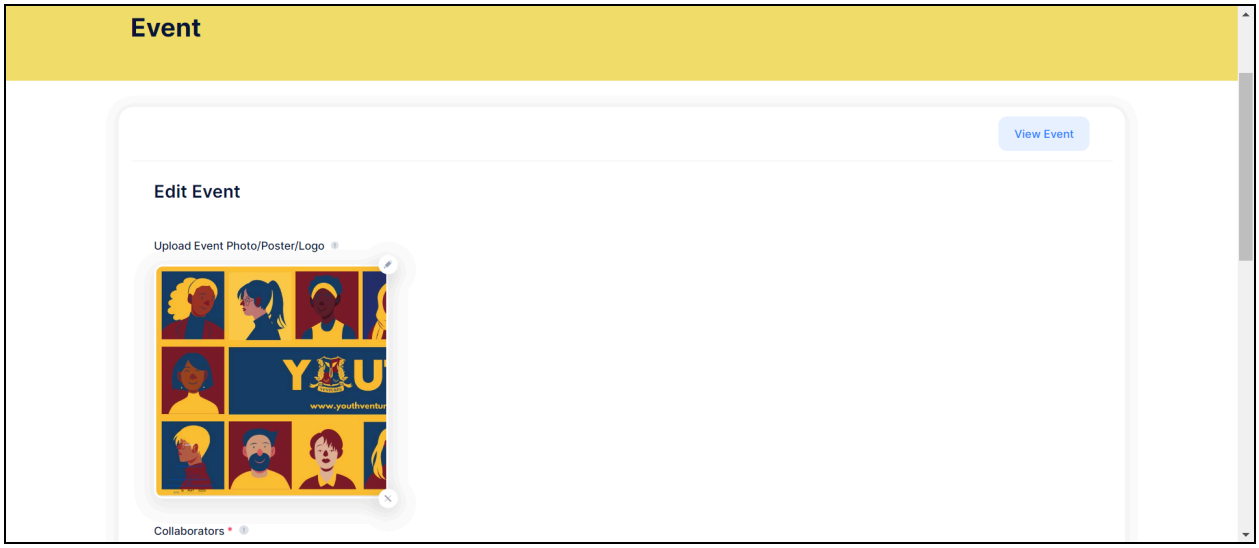


Figure 7.9: Edit Event Page

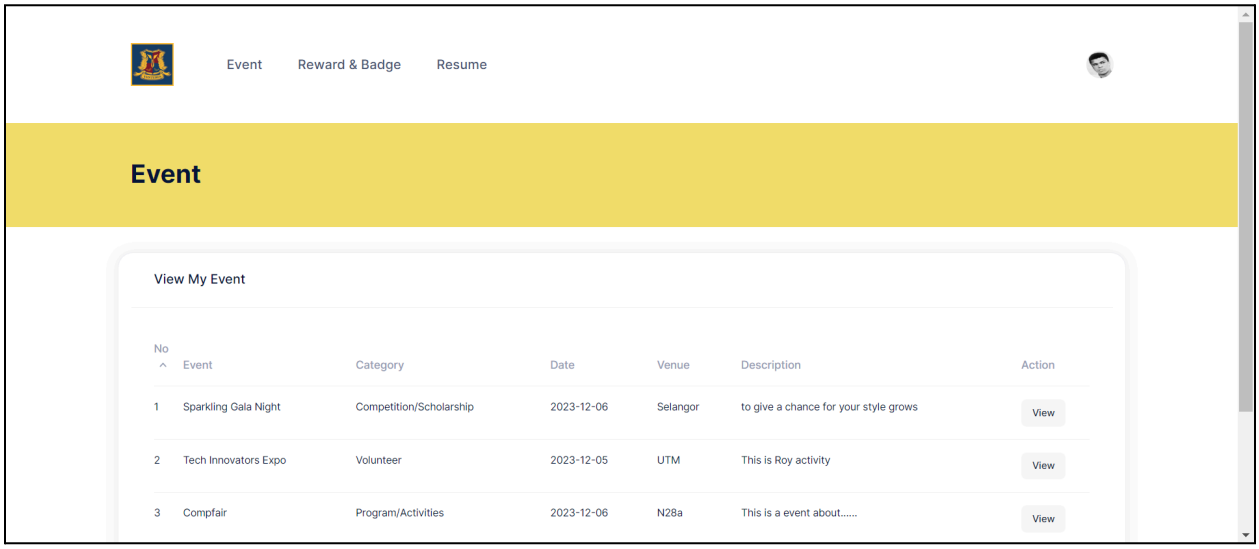


Figure 7.10: View My Event Page

Module Resume (Only for student)

i) View Resume

ii) Manage Certification

iii) Add Certification

iv) Update Certification

v) Manage Experience

vi) Add Experience

vii) Update Experience

Figure 8.1: View Resume Page

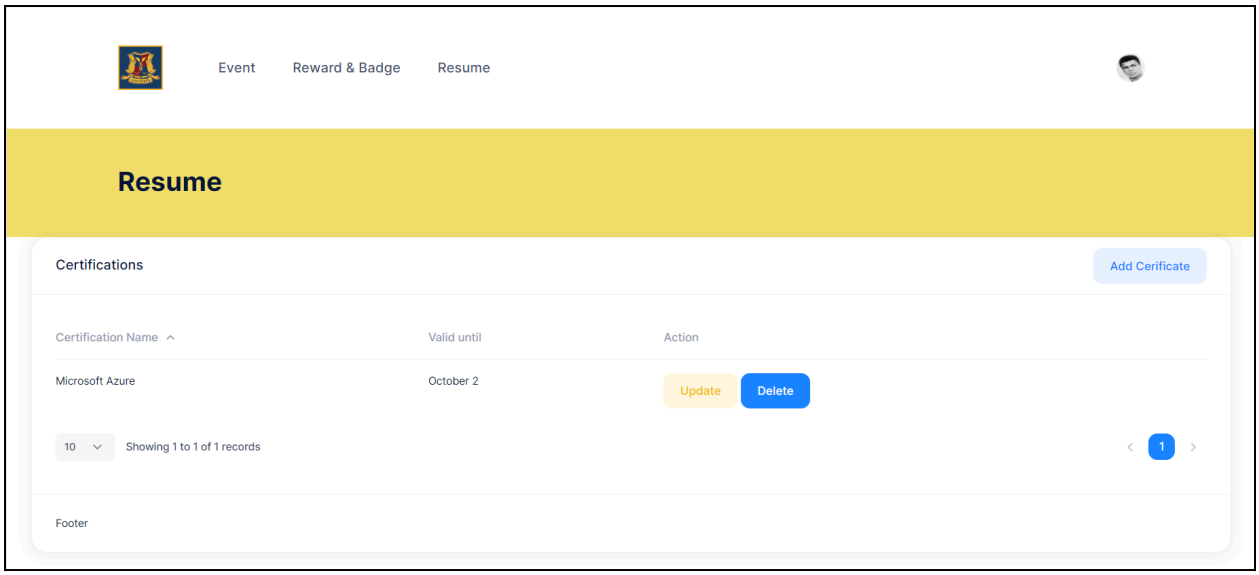


Figure 8.2: Manage Certification Page

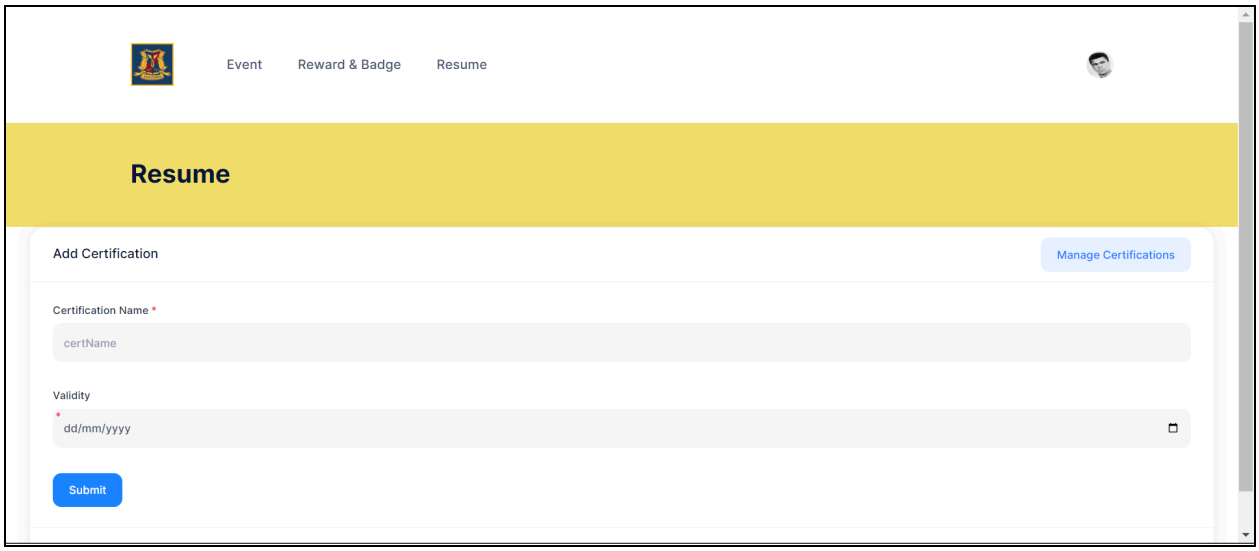


Figure 8.3: Add Certification Page

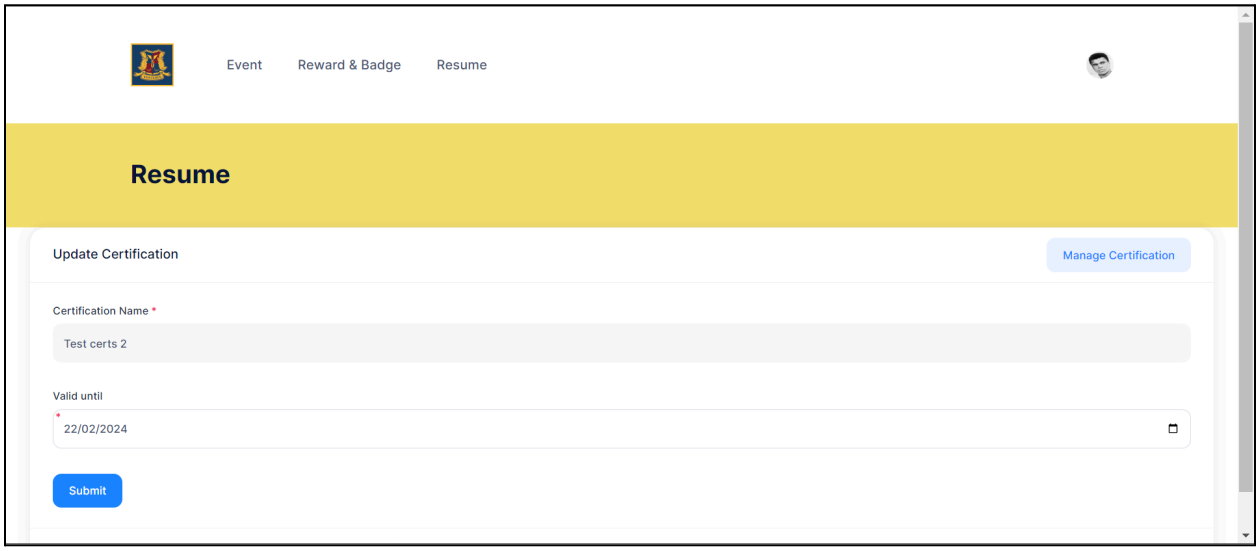


Figure 8.4: Update Certification Page

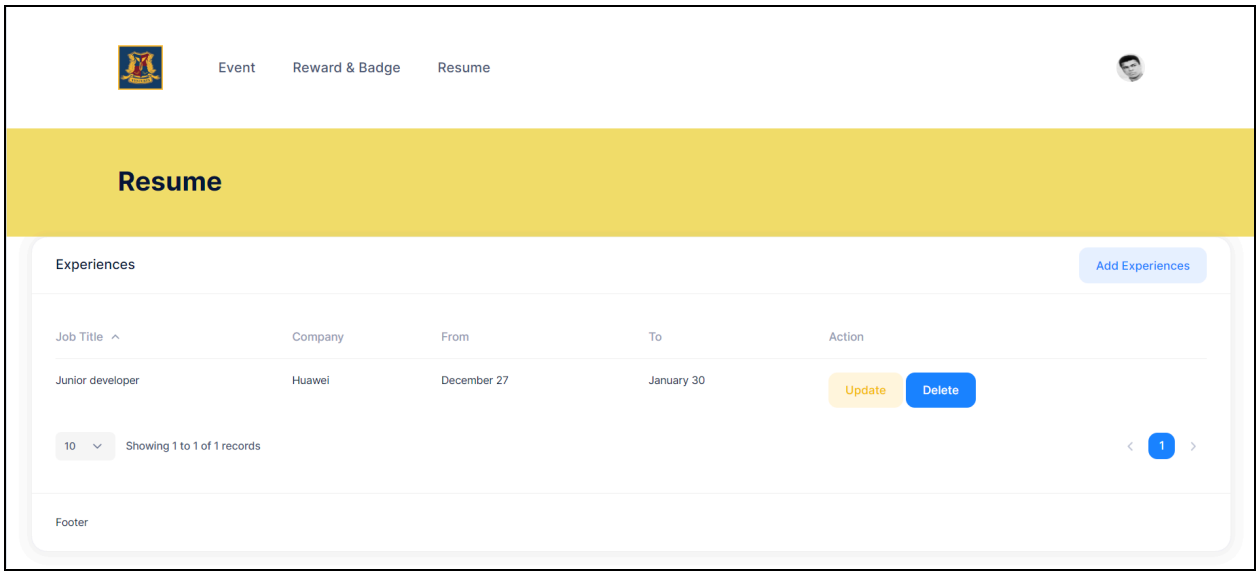


Figure 8.5: Manage Experience Page

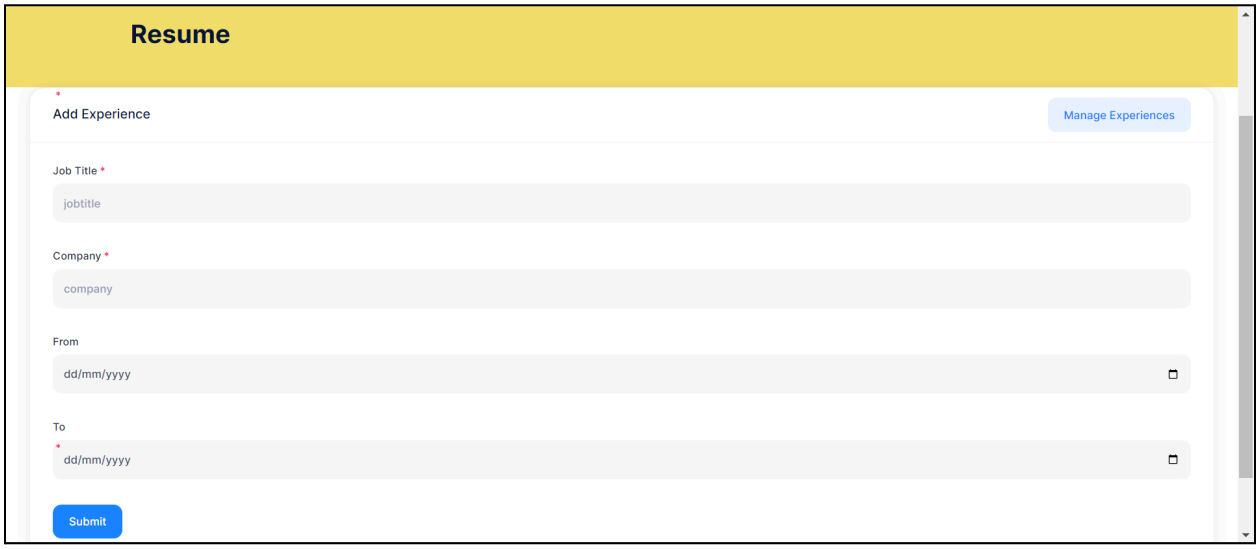


Figure 8.6: Add Experience Page

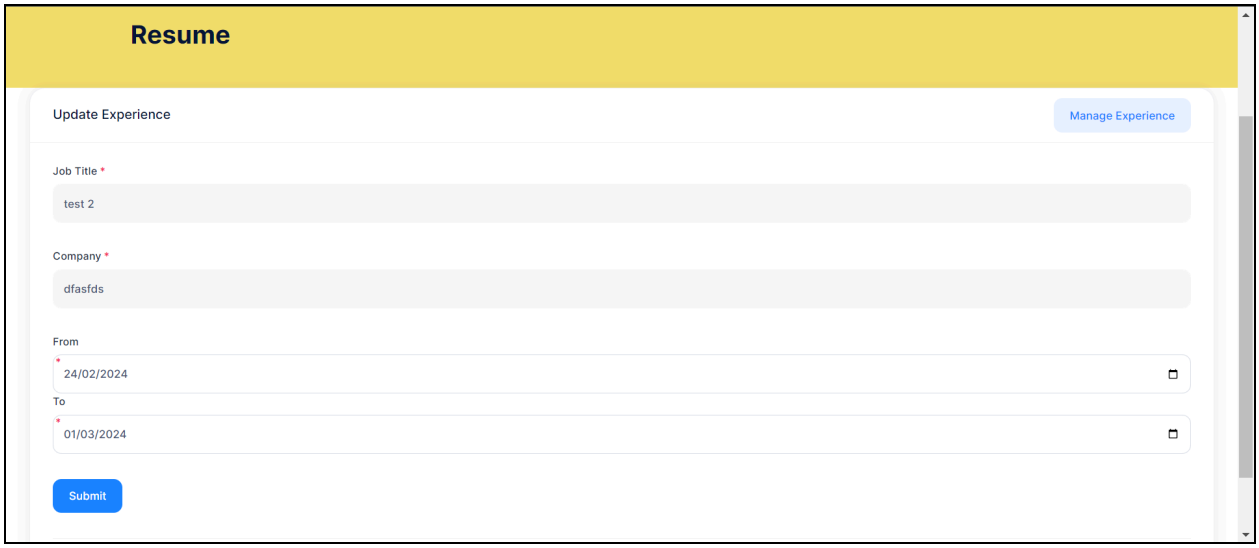


Figure 8.7: Update Experience Page

Module Reward and Badge

i) View Badge (Only for student)

ii) Register First Time Badge (Only for student)

iii) Manage Badge (Only for Youth Venture)

iv) Update Badge (Only for Youth Venture)

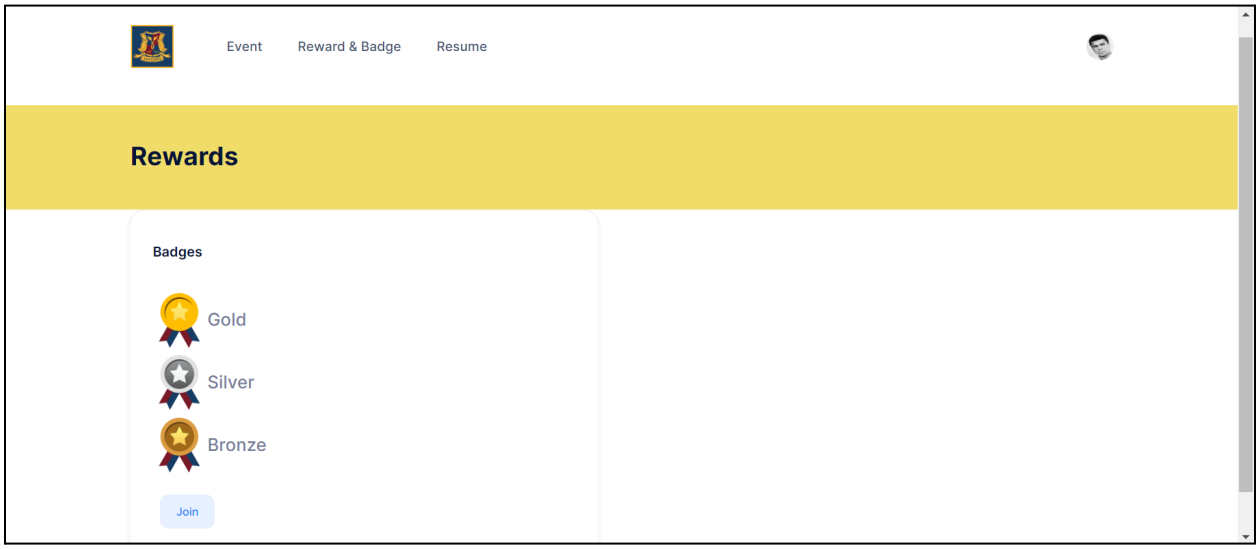


Figure 9.1: View Badge Page

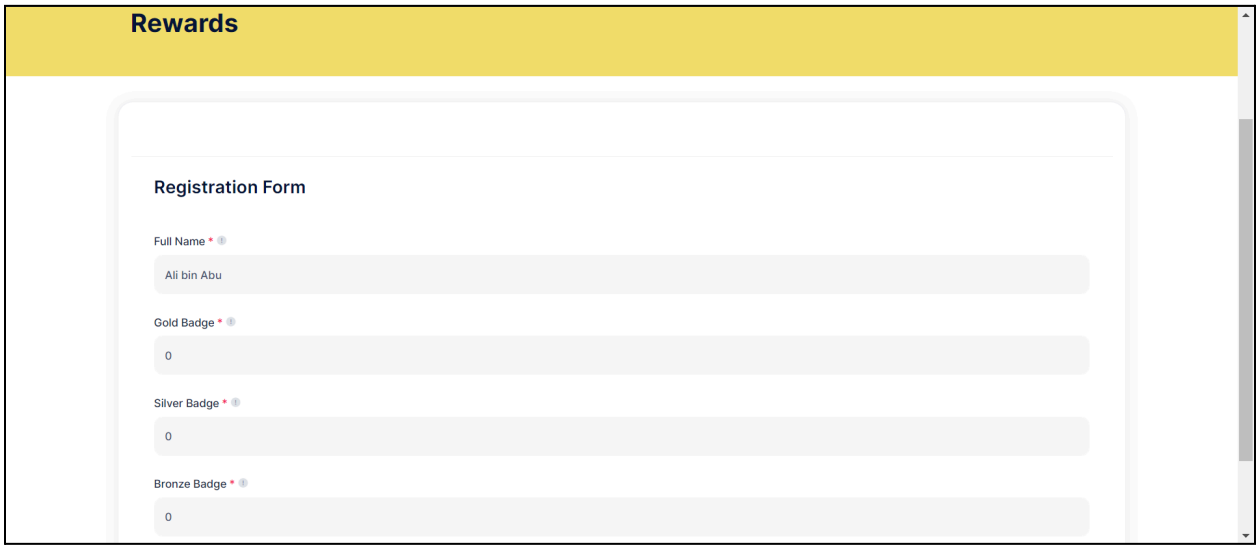


Figure 9.2: Register First Time Badge Page

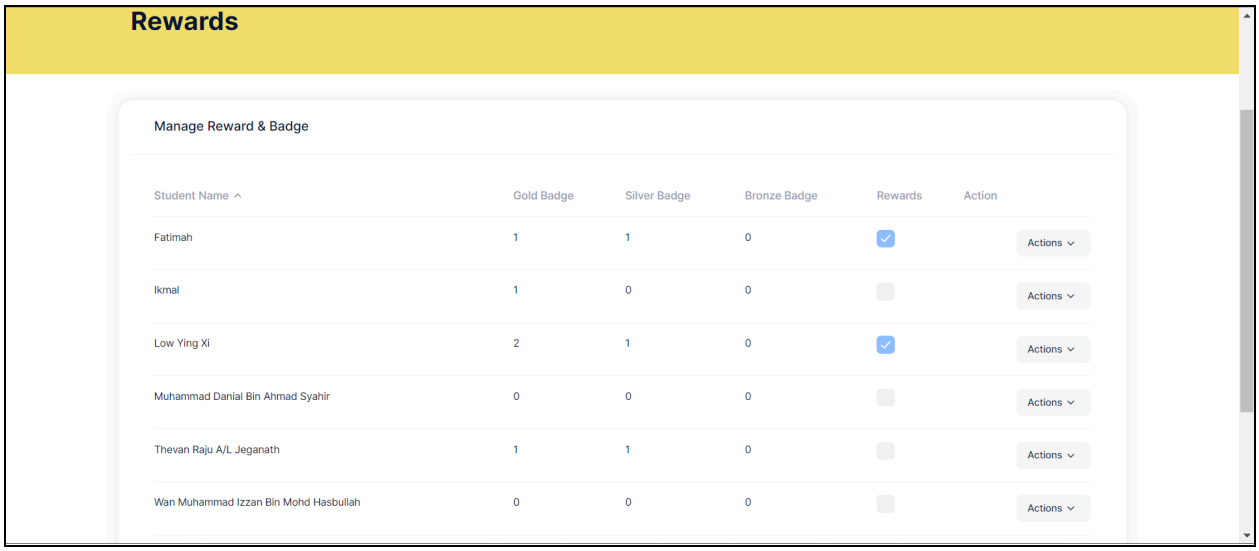


Figure 9.3: Manage Badge Page

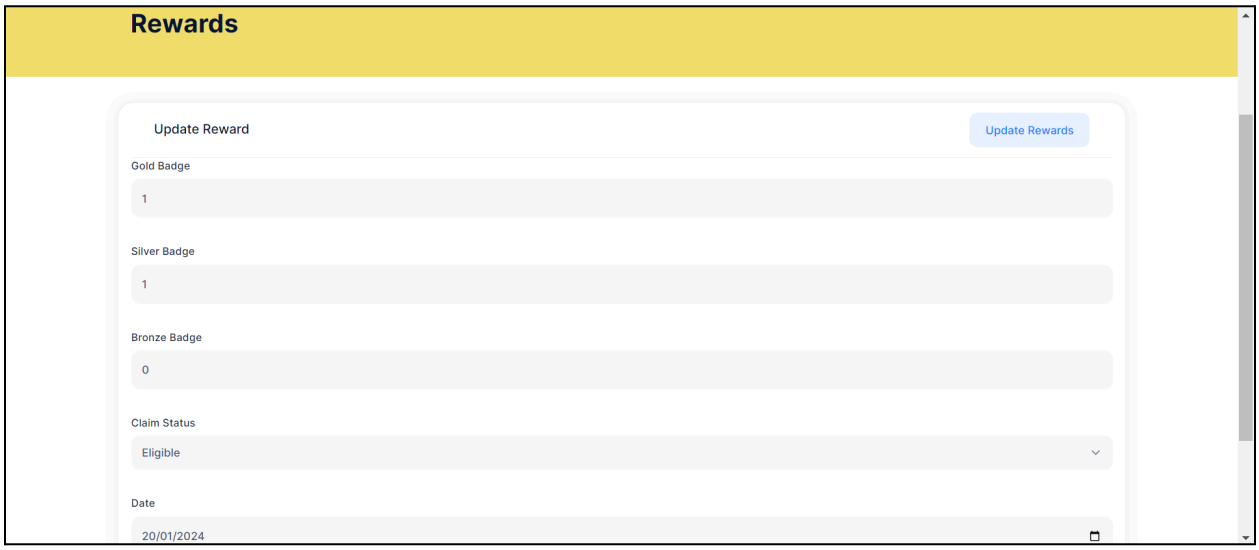


Figure 9.4: Update Badge Page

Alpha and Beta Testing Reports

Testing Activities

State	Test Description	Expected Outcome	Actual Outcome	Comments (any changes to be made)
Register	Student attempts to register into the Youth Venture Data Management System email:fatimah@gmail.com Password:fatimah 12345 User role:Student	User successfully register into the system	User successfully register into the system	N/A
Login	Student attempts to login into the Youth Venture Data Management System email:fatimah@gmail.com Password:fatimah 12345	User successfully login into the system	User successfully login into the system	N/A
Logout	Student attempts to logout into the Youth Venture Data Management System	User successfully logout into the system	User successfully logout into the system	N/A
Reset Password	User attempts to reset password using email address that integrate with user profile	The system redirects to reset password page and after input email it will notify reset password link in our email	The system redirects to reset password page and after input email it will notify reset password link in our email	N/A

View All users	Admin attempts see and edit the list of users in Youth Venture Data Management System	Admin redirects to the manage user page and have the power to view list of users in the system	Admin redirects to the manage user page and have the power to view list of users in the system	N/A
Create profile	User attempts to click my profile at the head wrapper section of our system	User directly been send to the page that display their profile and can input their personal information	User directly been send to the page that display their profile and can input their personal information	N/A
Update Profile	User attempts to update profile by clicking update button at below of edit profile page	User successfully update their profile	User successfully update their profile	N/A
Create Event	Admin attempts to create event that open to students to join	Admin successfully create and also send invitation to students to join the event	Admin successfully create and also send invitation to students to join the event	N/A
Edit Event	Admin attempts to edit event suitable with the requirement of what style they provide in the event	Admin successfully edit also connect this feature to the collaborators to edit their event following the target students	Admin successfully edit also connect this feature to the collaborators to edit their event following the target students	N/A
Delete Event	Admin attempts to delete event	Admin successfully delete event and display the event has been deleted from the	Admin successfully delete event and display the event has been deleted from the system also the	N/A

		system also the invitation of the deleted event	invitation of the deleted event	
Event List	Admin has power to display event list with actions of create,edit,delete, view details of event	Admin successfully display and make actions based on what they click also grant access to the partners/collaborators	Admin successfully display and make actions based on what they click also grant access to the partners/collaborators	N/A
Event Registration	Student attempts to register into an event	Student successfully redirects into registration form compulsory to the first timer and can updated pre filled data of different event	Student successfully redirects into registration form compulsory to the first timer and can updated pre filled data of different even	N/A
Participants list	Admin and Clients attempts to view list of participants	They successfully view the details of each participant	They successfully view the details of each participant	N/A
Participant details	Admin and client have power to view more detailed information of participants compared to participant list	They successfully can view much more detail information of registered participants	They successfully can view much more detail information of registered participants	N/A
Feedback	Link will be provide after student complete the event	Link contains feedback form that compulsory for student to fill before status of event	Link contains feedback form that compulsory for student to fill before status of event change to complete	N/A

		change to complete		
View ‘My Event’	Student attempts to view their list of event	Student successfully view their event that they have join	Student successfully view their event that they have join	N/A
Manage Certification	Student attempts view,edit,delete the certification of themselves	Student successfully fulfill of important action towards their certification	Student successfully fulfill of important action towards their certification	N/A
Create Certification	Student attempts to create certification into the system	Student successfully create their certification	Student successfully create their certification	N/A
Edit Certification	Student attempts to edit their certification	Student successfully edit their certification	Student successfully edit their certification	N/A
Delete Certification	Student attempts to delete their certification	Student successfully delete their certification	Student successfully delete their certification	N/A
Manage Experience Page	Admin attempts to manage certification of each student	Admin successfully manage and click any action providing in the manage resume page	Admin successfully manage and click any action providing in the manage resume page	N/A
Create Experience	Student attempts to create experience into the system	Student successfully create their experience	Student successfully create their experience	N/A
Edit Experience	Student attempts to edit their experience	Student successfully edit their experience	Student successfully edit their experience	N/A

Delete Experience	Student attempts to delete their experience	Student successfully delete their experience	Student successfully delete their experience	N/A
View Resume	Student attempts to view their respective resume that has been generate	Student successfully generates resume contains personal information, skills, certifications and experiences.	Student successfully generates resume contains personal information, skills, certifications and experiences.	N/A
VIew Badges Collected	Student attempts to view total badges that they have collected	Student successfully can view their badges based on hierarchy which is gold,silver and bronze	Student successfully can view their badges based on hierarchy which is gold,silver and bronze	N/A
Join Badge Race	Youth Venture attempts to collaborate with students join race to claim badges	They successfully collaborates towards the race and automatically updating total of badges of students in the system	They successfully collaborates towards the race and automatically updating total of badges of students in the system	N/A
Manage Badges	Youth Venture attempts to manage list of students that collecting badges from event	Youth Venture successfully grant power to access page and make any action of each badge	Youth Venture successfully grant power to access page and make any action of each badge	N/A
Update Badge	Youth Venture attempts to update current badges	Youth Venture successfully updating the number of	Youth Venture successfully updating the number of	N/A

		badges they want to give to students	badges they want to give to students	

UAT Reports

User Acceptance Test (UAT) Report

Section C - SYSTEM ANALYSIS & DESIGN
4= Exemplary
3= Proficient
2= Basic
1= Not Demonstrated

No.	Requirement	1	2	3	4
1.	Identifying Problems,Opportunities & Objectives			X	
2.	Determining Requirements			X	
3.	Analyzing System Needs			X	
4.	Designing System			X	
5.	Developing and Documenting System				X
6.	Testing and Maintaining System			X	

4= Strong Agree
3= Agree
2= Disagree
1= Strongly Disagree

No.	Requirement	1	2	3	4
7.	Evaluating the System				
	(a) The system is complete			X	
	(b) The system is very easy to learn and use.			X	
8.	Others				
	(a) Team always communicates with industry representatives.			X	
	(b) Utilize both physical and virtual communication platforms.			X	

Section D - SYSTEM DEVELOPMENT TECHNOLOGY

4= Exemplary
3= Proficient
2= Basic
1= Not Demonstrated

No.	Requirement	1	2	3	4
1.	Creativity and design				X
2.	User interface		X		
3.	Usability and navigation of the system			X	
4.	Technical quality			X	
5.	User manual			X	
6.	Clarity of explanation and progress presentation during regular meeting and industry day presentation			X	

Section E - DATABASE

4= Exemplary
3= Proficient
2= Basic
1= Not Demonstrated

No.	Requirement	1	2	3	4
1.	System requirements			X	
2.	Input Forms			X	
3.	Table Data			X	
4.	System Returns Information From Database Correctly			X	
5.	Demonstration of the application produces the desired output, report and/or graphs				X
6.	Charts/report format output correct data relationships			X	
7.	Group Presentation - Effectiveness				X

8.	Group Communication Skill Throughout the Whole Development Cycle			X	

Section E - OTHERS FEEDBACK

Are you satisfied with the developed system?	Satisfied
Constructive feedback about the group / system	Resume design clearly reflected the industry needs and helps students to document their input well

Gantt Chart

The Gantt Chart presents the steps/phases taken by team Innovatio during the development of the Youth Venture Data Management System.

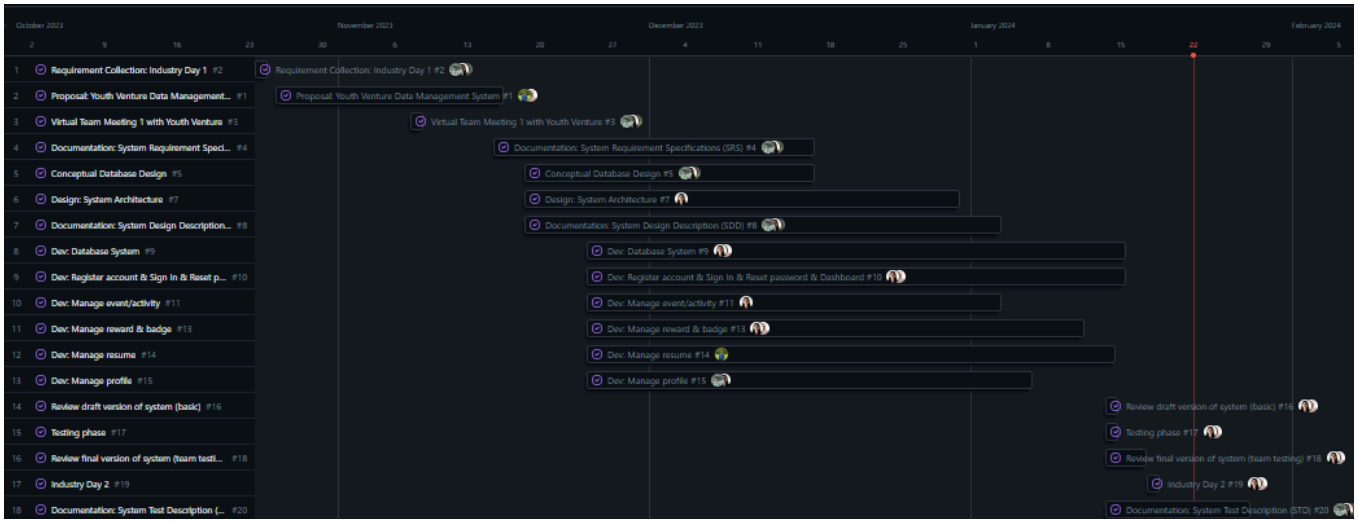


Figure 10: Innovatio Gantt Chart

Link: https://github.com/users/lowyingxi/projects/1/views/1