SECP3204: Software Engineering WBL

System Design Descriptions (SDD)

KTDI EVENT MANAGEMENT SYSTEM

Version 2.0

Date: 10 June 2023

School of Computing, Faculty of Engineering

Prepared by: Group Curiousity

Table of Contents

3	**System Architectural Design**			**3**
	3.1	Architectural Style and Rationale		3
	3.2	Component Model		5
	3.3	Use Case Diagram		7
4	**Detailed Description of Components**			**10**
	4.1	Complete Package Diagram		10
	4.2	Detailed Description		11
		4.2.1	P001: Authentication Subsystem	11
		4.2.2	P002: Create Event Subsystem	25
		4.2.3	P003: Event Preparation Subsystem	37
		4.2.4	P004: Register Event Subsystem	52
		4.2.5	P005: Attendance and Feedback Subsystem	63
		4.2.6	P006: Active Quota Subsystem	74
5	**Data Design**			**79**
	5.1	Data Description		79
	5.2	Data Dictionary		81
6	**User Interface Design**			**85**
	6.1	Overview of User Interface (3 users)		85
	6.2	Screen Images		87

7.	**Requirement Matrix**	
	Appendices	–

3. System Architectural Design

3.1 Architecture Style and Rationale

The architecture style used for the KTDI Event Management System is the Model-View-Controller (MVC) pattern. The MVC pattern separates the presentation and interaction from the system data into three distinct components: the model, view, and controller. This architectural structure provides a clear separation of concerns and enhances the maintainability and scalability of the system.

The Model component in the MVC pattern handles the management of data and its associated operations. It represents classes or objects that store the attributes or properties of the described entities. The Model encapsulates the logic for retrieving, manipulating, and storing data, ensuring data integrity and consistency.

The View component manages the visual representation and user interface of the system. It is responsible for displaying the system's output to the users. The View renders the data from the Model and presents it in a user-friendly format, such as web pages, forms, or graphical interfaces.

The Controller acts as the bridge between the Model and View components. It manages the user interaction and orchestrates the flow of data and actions between the two. The Controller receives user input, processes it, and triggers appropriate actions in the Model or View. It ensures that the system responds to user requests and maintains the synchronization between the data and the user interface.

The development cycle is sped up using MVC architecture. Individual developers can collaborate on the same module in a development team at different levels, pipelining the development. The proposed system benefits from the categorized modular and loosely coupled components. Diverse resources and documentation of MVC for reference and learning become one of the factors for choosing MVC architecture as the architecture style.

Besides, MVC architecture is suitable for large-scale applications as it is easy to modify. With MVC, developers only need to modify a single component, and the changes will propagate

throughout the application. This simplifies the process of making updates or enhancements, providing efficiency and flexibility for developers working on the application.

Last,one of the most significant advantages of utilising MVC in a web-based application is its inherent flexibility, which results in faster development processes. Its flexibility allows the changes made will not entirely affect other part of the system, which provides more efficient and targeted modifications.

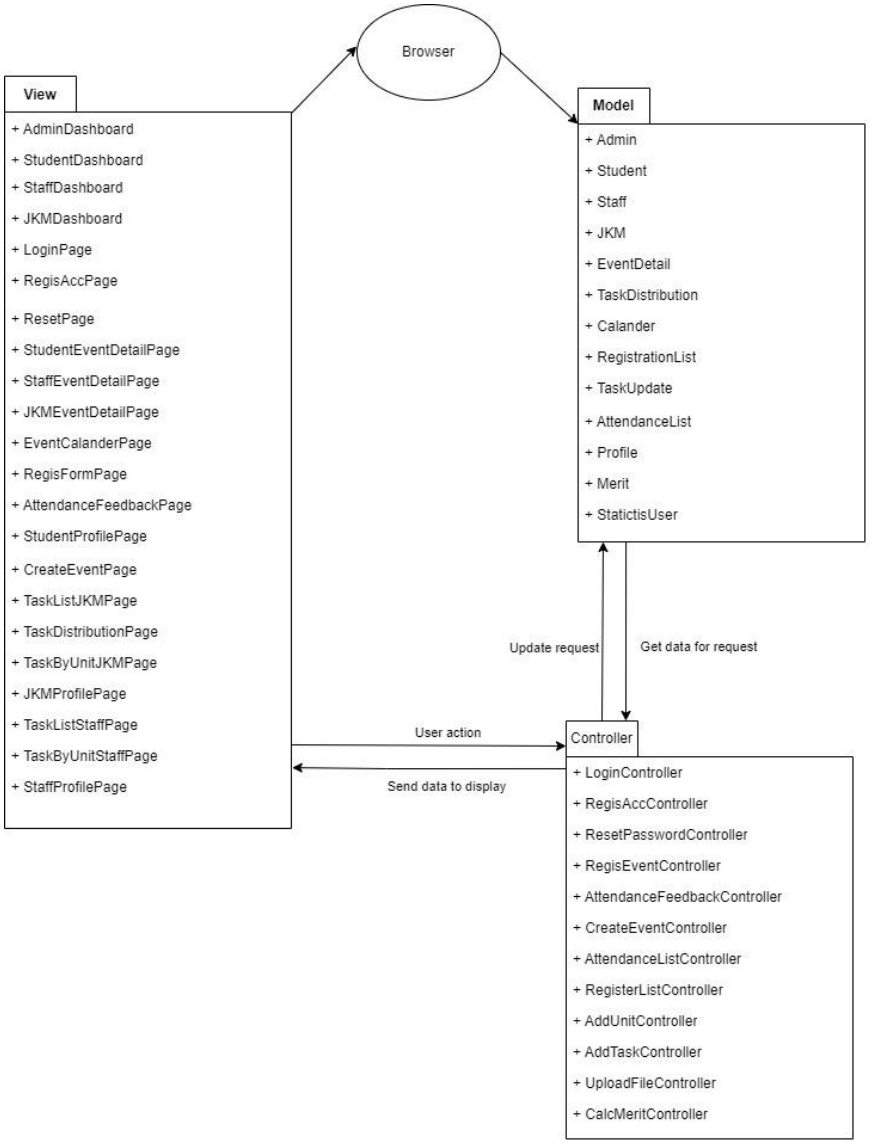

Figure 3.1.1: Architecture Model for KTDI Event Management System

3.2 Component Model

This system consists of six subsystems, each containing components responsible for specific functions. Components communicate with other components both within and outside their respective subsystems through interfaces, represented as ports. The subsystems include the authentication subsystem, create event subsystem, event preparation subsystem, register event subsystem, attendance and feedback subsystem, and active quota subsystem.

The Authentication subsystem comprises the following components: account registration, login, and password reset. Account registration allows new users to create their accounts. Login enables registered users to access the system based on their roles using their email address and password. Password reset allows users to reset their account password.

The Create Event subsystem consists of the following components: event calendar, create event, and event detail. The event calendar displays events with corresponding dates. When an event is created, it is marked on the event calendar. Create event allows JKM members to create and edit event details. Event detail stores all event information and presents it to users.

The Event Preparation subsystem facilitates event planning and organization. It includes components such as task distribution, document of task, and registration for event link. Task distribution provides folders for different units, allowing JKM members to update task progress and store relevant documents. The registration for event link generates a registration form template linked to the event detail.

The Register Event subsystem allows students to register for events using the registration form embedded in the event detail. Submitted forms are stored in the event's registration list.

The Active Quota subsystem tracks students' event attendance and assigns merit points. It comprises the profile and merit components. The profile component stores a list of events attended by students, along with their corresponding merits. The merit component calculates the total merit and determines whether a user is an active quota student.

The Attendance and Feedback subsystem includes components for attendance and feedback management, such as link, form, and list. This subsystem records participant

attendance, collects feedback, and assigns merits based on event attendance. It helps identify quota active students.

The component diagram of KTDI event management system is shown as Figure 3.2 below:

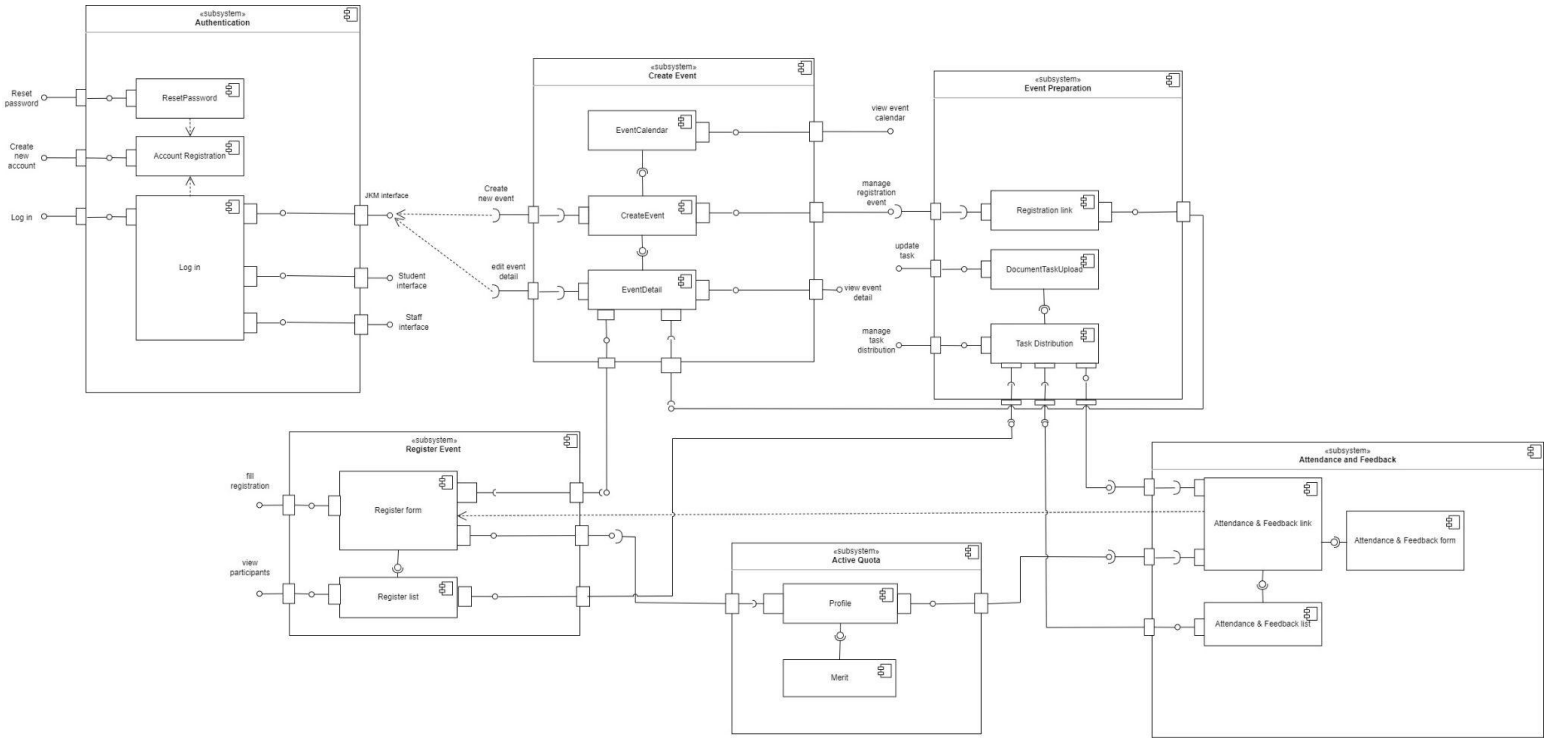

Figure 3.2.1: Component Diagram of KTDI Event Management System

3.3 Use Case Diagram

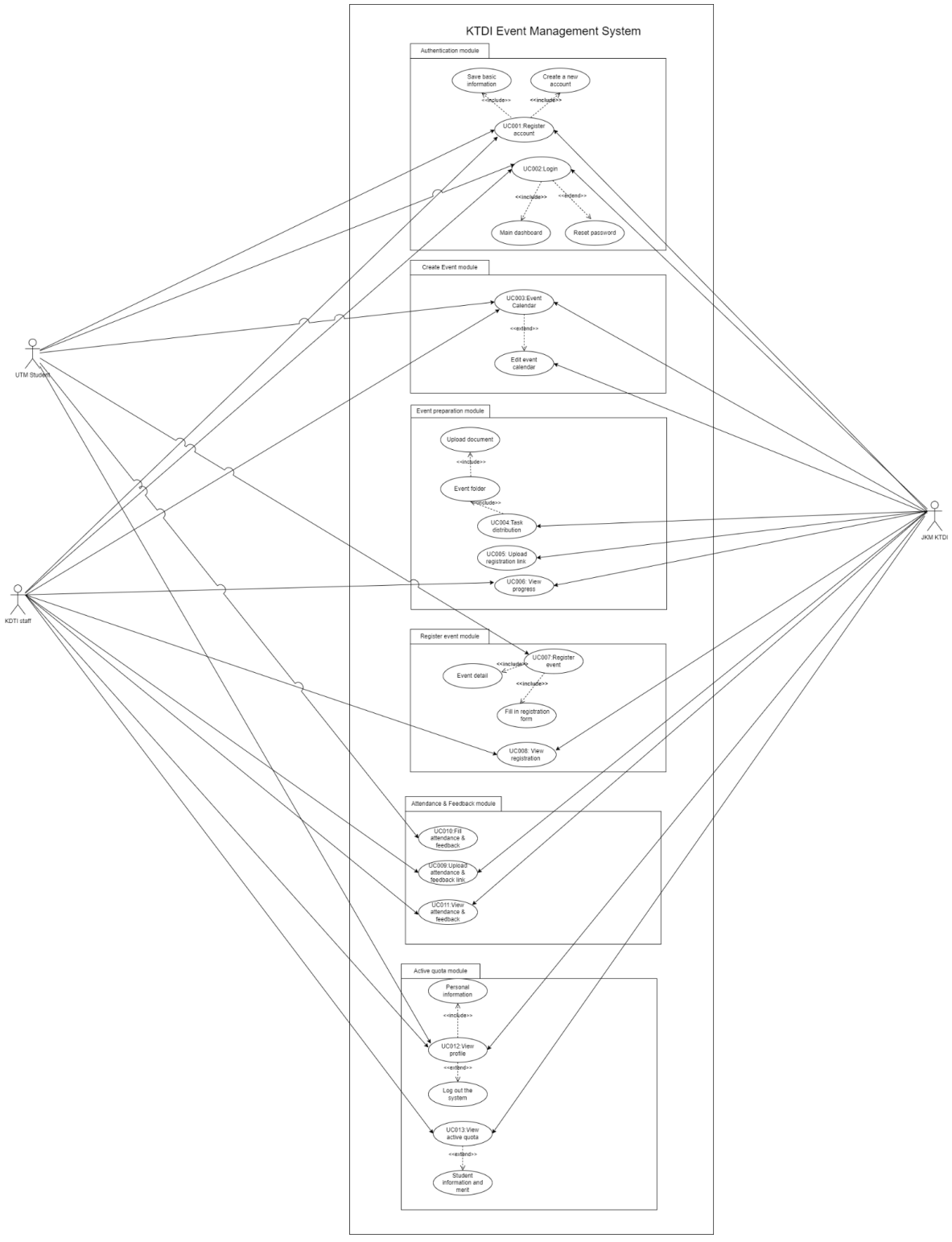

Figure 3.3.1: Use Case Diagram of KTDI Event Management System

The figure above shows all of the functions that exist in the KTDI Event Management System. Besides the use case and subsystems, there are three users that are directly involved in the system which are JKM members, KTDI staff and students. The arrow shows users can only access certain functions. There are 13 use cases that are divided into 6 subsystems. The description on what the actors and use case's role in the system will discussed in the table below:

Module	**Function**	**Description**
Authentication Module	UC001 – Register Account	This use case allows students in UTM to sign up as a user for the system.
	UC002 – Log in	This use case allows registered users to log in thee system.
Create Event Module	UC003 – Event Calendar	This use case allows JKM KTDI to upload event details on the respective date of the event calendar and allows students and KTDI staff to view more detailed information about the event.
Event Preparation Module	UC004 – Task Distribution	This use case allows JKM KTDI to assign tasks to JKM member according to the unit, upload documents and update their task progress.
	UC005 – Upload Registration Link	This use case allows JKM KTDI to upload the details of the event and the registration form link for students to register for the event..
	UC006 – View Progress	This use case allows JKM KTDI and KTDI staff to view the progress of the task distributed by viewing the content in the each unit's folder
Register Event Module	UC007 – Register Event	This use case allows students to register an event by filling their personal information in the registration form. The system will store the registered event in the student profile.

	UC008 – View Registration	This use case allows JKM KTDI and KTDI staff to view the users that have registered for an event.
Attendance & Feedback Module	UC009 – Upload Attendance & Feedback Link	This use case allows JKM KTDI to upload the attendance and feedback link to students to record participants' attendance and feedback for the calculation of quota active and improvement of the event.
	UC010 – Fill Attendance & Feedback	This use case allows students to fill in their attendance and feedback at the end of the event.
	UC011 – View Attendance & Feedback	This use case allows JKM KTDI and KTDI staff to view the attendance and feedback from students for an event in a list.
Active Quota Module	UC012 – View Profile	This use case will show the user's personal information and display the event joined in a list for JKM and students.
	UC013 – View Active Quota	This use case will show the list of all the students that participate in KTDI events with their KTDI merit score to identify quota-active students. When clicking on a student's name, it will display the list of activities and merit provided for each event that has been participated in by the student and the total KTDI merit earned by the student.

Table 3.3.2: Use Case of KTDI Event Management System

4. Detailed Description of Components

4.1 Complete Package Diagram

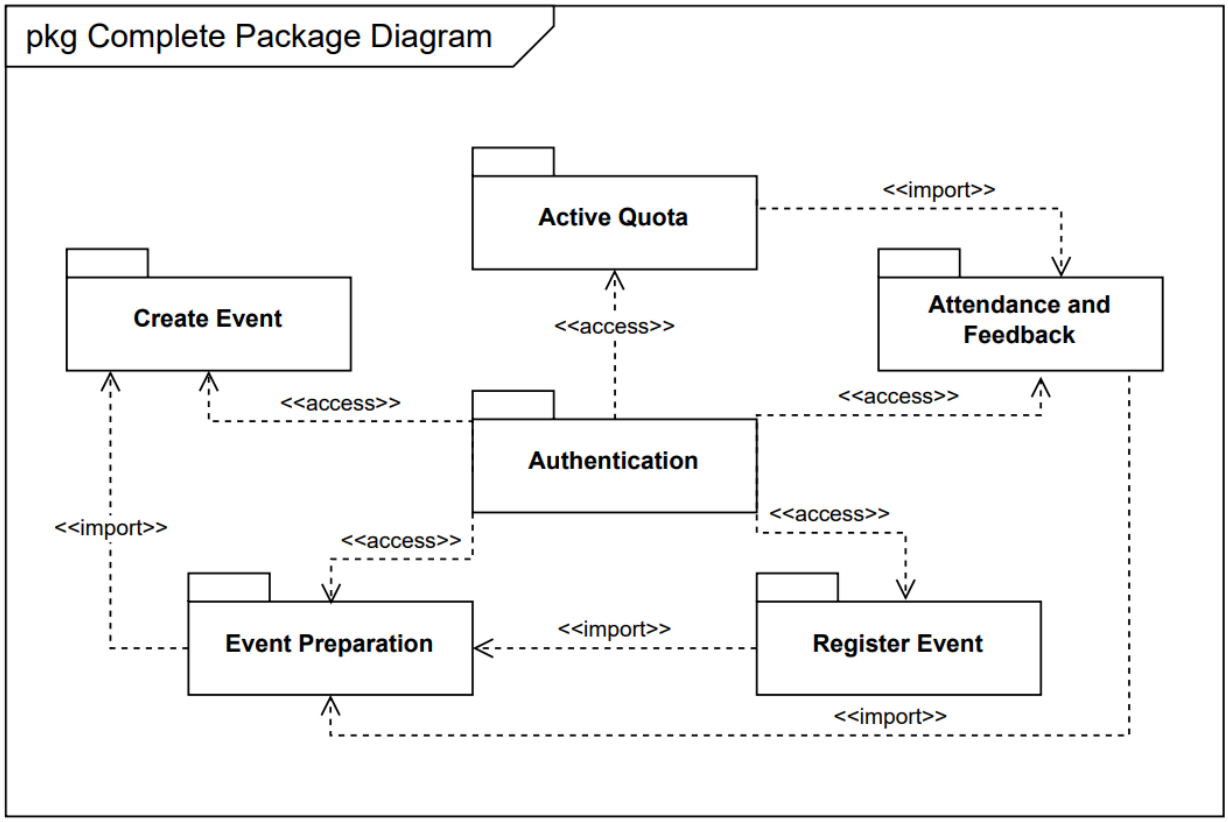

Figure 4.1.1: Complete Package Diagram for KTDI Event Management System

4.2 Detailed Description

4.2.1 P001: Authentication Subsystem

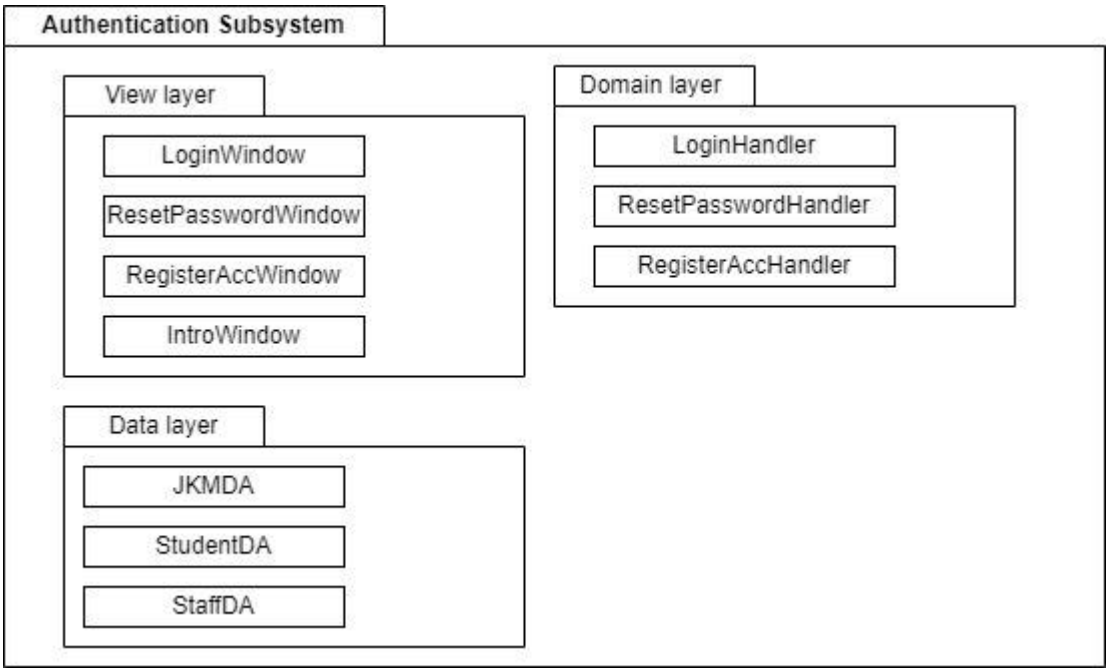

Figure 4.2.1.1: Package Diagram for Authentication Subsystem

4.2.1.1 Class Diagram

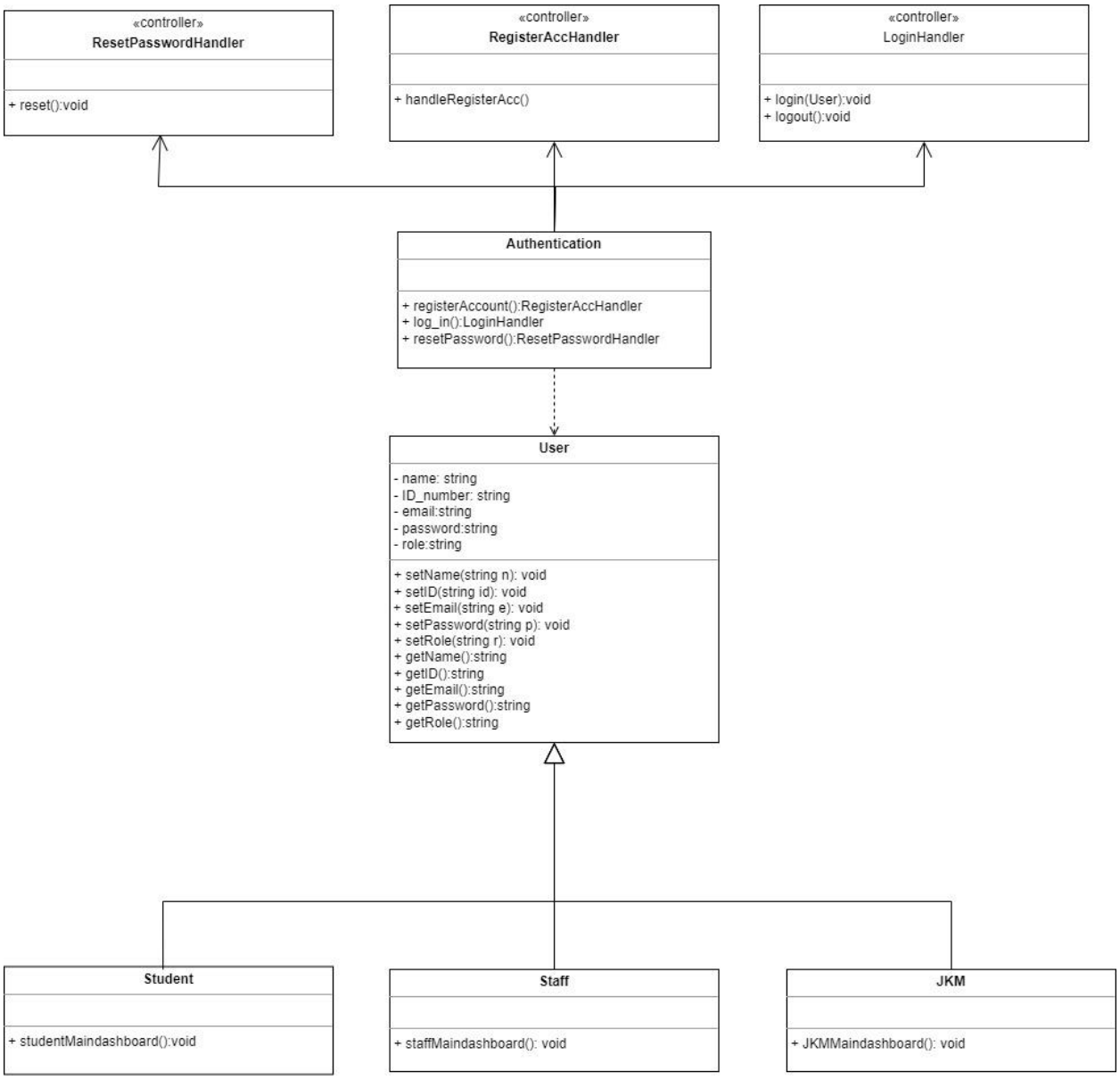

Figure 4.2.1.1.1: Class Diagram for Authentication Subsystem

Entity Name	ResetPasswordHandler
Method Name	reset
Input	email & password
Output	-
Algorithm	1. Start. 2. Accept email input from the user to identify the account. 3. Generate a link and send it to the user's registered email address for verification. 4. Allow the user to reset their password 5. Update the user's password in the database. 6. End.

Entity Name	RegisterAccHandler
Method Name	handleRegisterAcc
Input	name, ID_number, email, password, role
Output	-
Algorithm	1. Start. 2. Accept the user registration input, including name, password, role, email and ID_number. 3. Validate the input data to ensure it meets the required criteria (e.g.required field is filled correctly). 4. Create a new instance of the User class with the provided information and store it in the database.

	5. Store all user information to the database. 6. End.

Entity Name	LoginHandler
Method Name	login
Input	email & password & role
Output	-
Algorithm	1. Start. 2. Accept the email and password input from the user. 3. Retrieve the user data from the database based on the provided email. 4. Verify that the provided password matches the stored password for the corresponding user. 5. If the credentials are valid, call the method for the main dashboard according to the role. 6. Display main dashboard according to the role 7. End.

Entity Name	LoginHandler
Method Name	logout
Input	user action to log out
Output	-

Algorithm	1. Start. 2. Users select to log out. 3. Account is logged out 4. End.

Entity Name	Authentication
Method Name	registerAccount
Input	name, ID_number, email, password, role
Output	-
Algorithm	1. Start. 2. Accept the user registration input, including name, password, role, email and ID_number. 3. Invoke the RegisterAccHandler's handleRegisteAcc() method, passing the registration input as parameters. 4. Return the result of the RegisterAccHandler's handleRegisterAcc() method back to the Authentication. 5. End.

Entity Name	Authentication
Method Name	log_in
Input	name
Output	-

Algorithm	1. Start. 2. Accept the email and password input from the user. 3. Invoke the LoginHandler's login method, passing the email and password as parameters. 4. Return the result of the LoginHandler's login() method back to the Authentication. 5. End.

Entity Name	Authentication
Method Name	resetPassword
Input	email & password
Output	-
Algorithm	1. Start. 2. Accept the email input from the user to identify the account. 3. Invoke the RegisterAccHandler's reset() method, passing the email as a parameter. 4. Return the result of the RegisterAccHandler's reset() method back to the Authentication. 5. End.

Entity Name	User

Method Name	setName
Input	na9ifdme
Output	-
Algorithm	1. Start. 2. Set a name according to user input. 3. Add name to the database. 4. End.

Entity Name	User
Method Name	setID
Input	ID_number (matric no or staff ID)
Output	-
Algorithm	1. Start. 2. Set an ID according to user input. 3. Add ID to the database. 4. End.

Entity Name	User
Method Name	setEmail
Input	email

Output	-
Algorithm	1. Start. 2. Set an email address according to user input. 3. Add email address to the database. 4. End.

Entity Name	User
Method Name	setPassword
Input	password
Output	-
Algorithm	1. Start 2. Set a password according to user input. 3. Add password to the database. 4. End

Entity Name	User
Method Name	setRole
Input	role
Output	-
Algorithm	1. Start 2. Set a role according to user selection.

	3. Add a role to the database. 4. End

Entity Name	User
Method Name	getName
Input	-
Output	name
Algorithm	1. Start. 2. Fetch the name stored in memory 3. Return the name stored 4. End.

Entity Name	User
Method Name	getID
Input	-
Output	ID_number
Algorithm	1. Start. 2. Fetch the ID stored in memory 3. Return the ID stored 4. End.

Entity Name	User
Method Name	getEmail
Input	-
Output	email
Algorithm	1. Start. 2. Fetch the email address stored in memory 5. Return the email address stored 6. End.

Entity Name	User
Method Name	getPassword
Input	-
Output	password
Algorithm	1. Start. 2. Fetch the password stored in memory 3. Return the password stored 4. End.

Entity Name	User
Method Name	getRole

Input	-
Output	role
Algorithm	1. Start. 2. Fetch the role stored in memory 5. Return the role stored 6. End.

Entity Name	Student
Method Name	studentMaindashboard
Input	-
Output	Student Main Dashboard
Algorithm	1. Start 2. Display student main dashboard 3. End

Entity Name	Staff
Method Name	staffMaindashboard
Input	-
Output	Staff Main Dashboard

Algorithm	1. Start 2. Display staff main dashboard 3. End

Entity Name	JKM
Method Name	JKMMaindashboard
Input	
Output	JKM Main Dashboard
Algorithm	1. Start 2. Display JKM main dashboard 3. End

4.2.1.2 Sequence Diagram

a) SD001: Sequence diagram for Register Account

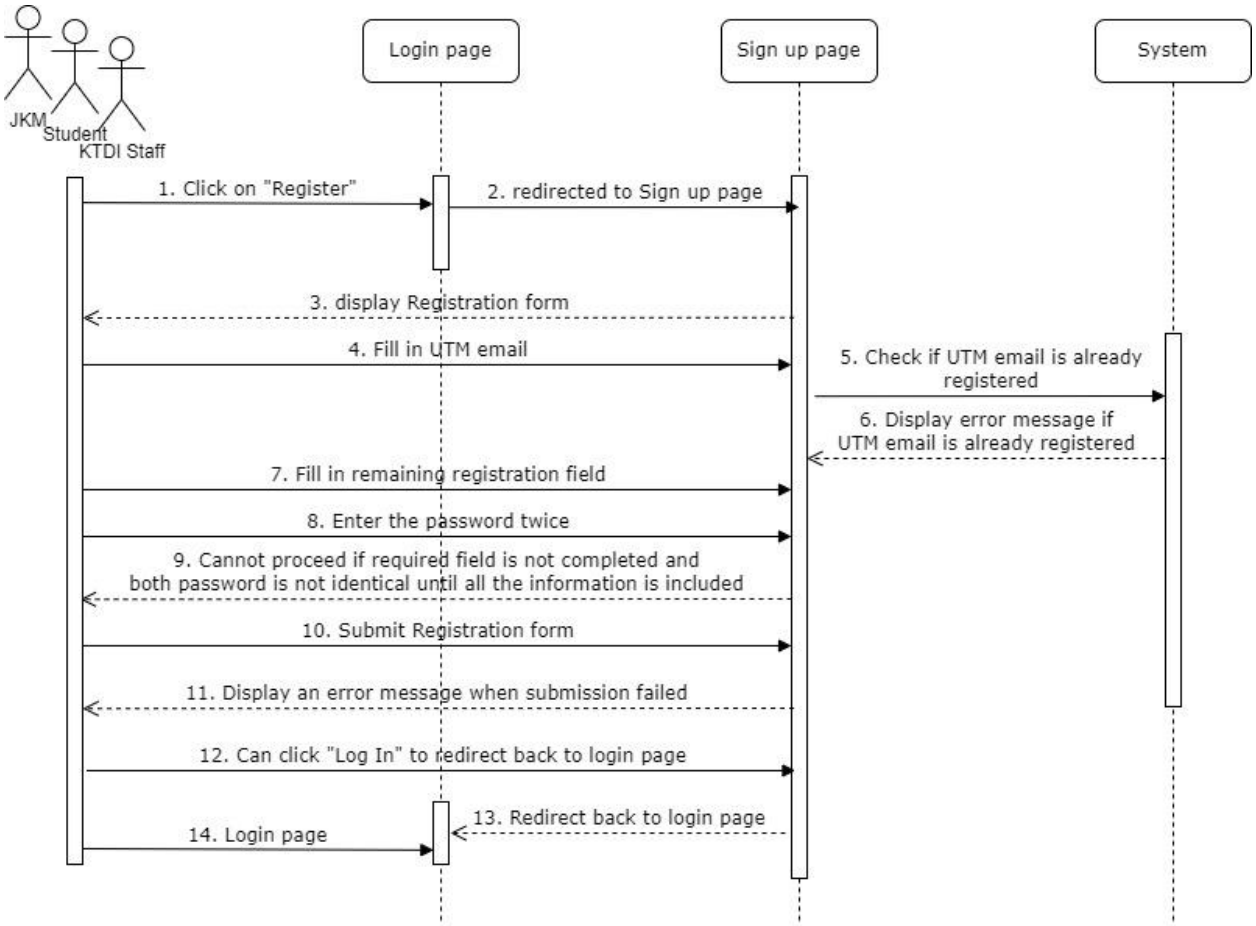

Figure 4.2.1.2.1: Sequence Diagram for Register Account

b) SD002: Sequence diagram for Login

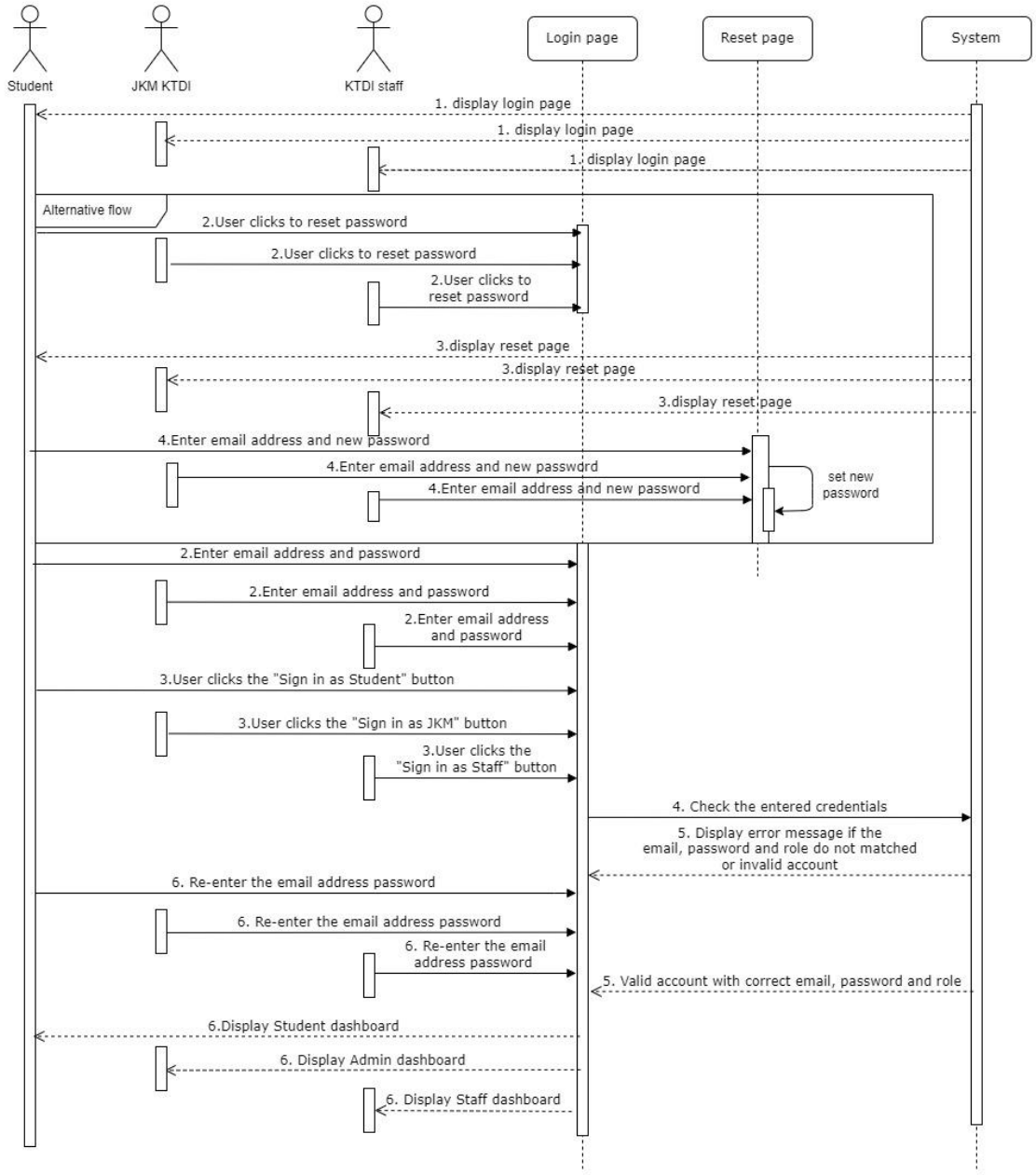

Figure 4.2.1.2.2: Sequence Diagram for Login

4.2.2 P002: <Create Event> Subsystem

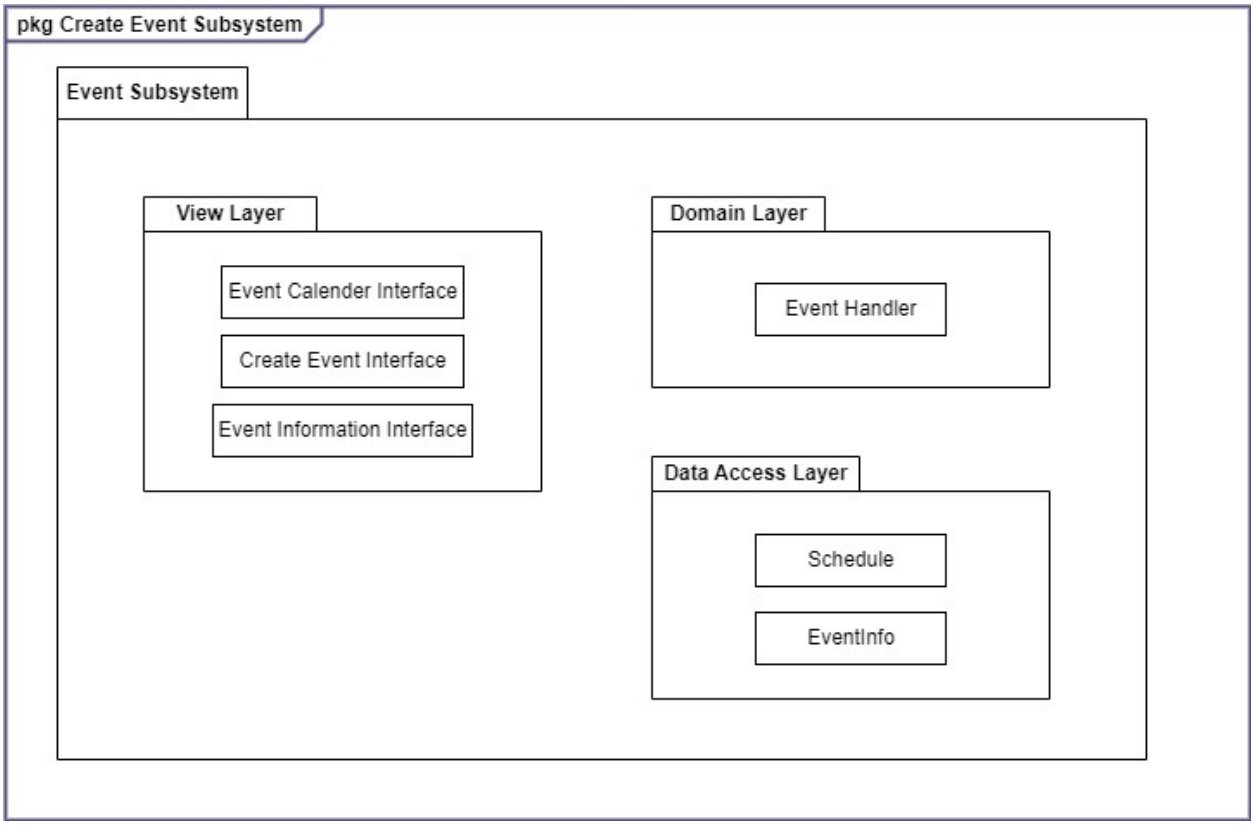

Figure 4.2.2.1: Package Diagram for Create Event subsystem

4.2.2.1 Class Diagram

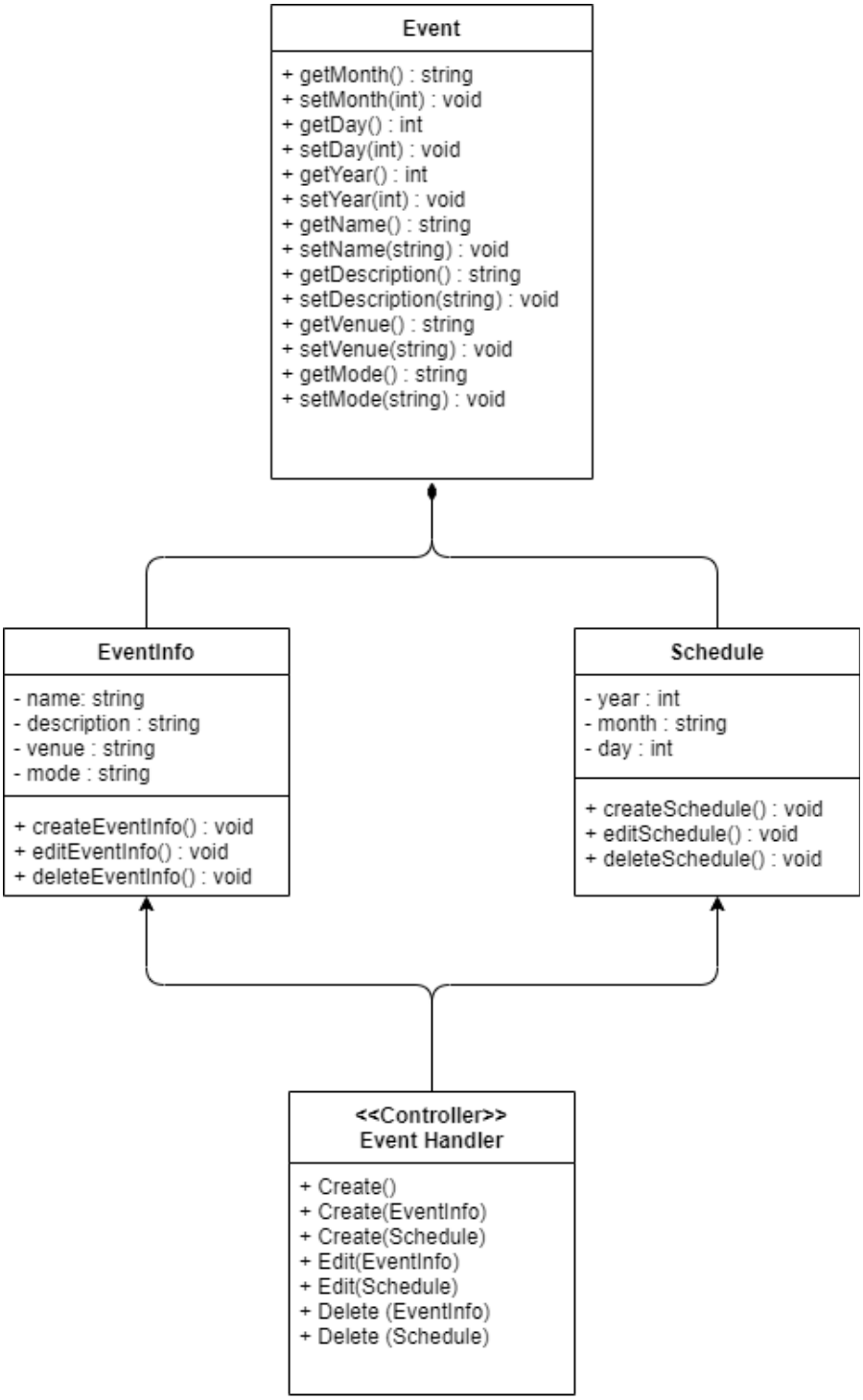

Figure 4.2.2.1.1: Class Diagram for Create Event Subsystem

Entity Name	Event
Method Name	getMonth
Input	-
Output	List of event month in the system
Algorithm	1. Start. 2. Fetch the information of event month in the system. 3. Display the event month on the event calendar and event information interface. 4. End.

Entity Name	Event
Method Name	getDay
Input	-
Output	List of event day in the system
Algorithm	1. Start. 2. Fetch the information of event days in the system. 3. Display the event day on the event calendar and event information interface. 4. End.

Entity Name	Event
Method Name	getYear

Input	-
Output	List of event year in the system
Algorithm	1. Start. 2. Fetch the information of the event year in the system. 3. Display the event year on the event calendar and event information interface. 4. End.

Entity Name	Event
Method Name	getName
Input	-
Output	List of event name in the system
Algorithm	1. Start. 2. Fetch the information of the event name in the system. 3. Display the event name in the event calendar and event information interface. 4. End.

Entity Name	Event
Method Name	getDescription
Input	-
Output	List of event description in the system

Algorithm	1. Start. 2. Fetch the information of event description in the system. 3. Display the event description on the event information interface. 4. End.

Entity Name	Event
Method Name	getVenue
Input	-
Output	List of event venue in the system
Algorithm	1. Start. 2. Fetch the information of the event venue in the system. 3. Display the event venue on the event information interface. 4. End.

Entity Name	Event
Method Name	getMode
Input	-
Output	List of event mode in the system
Algorithm	1. Start. 2. Fetch the information of event mode in the system. 3. Display the event mode on the event information

	interface. 4. End.

Entity Name	Event
Method Name	setMonth
Input	Event month to be created
Output	-
Algorithm	1. Start. 2. Set a new event month with provided information. 3. Add event month to the database. 4. End.

Entity Name	Event
Method Name	setDay
Input	Event days to be created
Output	-
Algorithm	1. Start. 2. Set new event days with provided information. 3. Add event days to the database. 4. End.

Entity Name	Event
Method Name	setYear
Input	Event year to be created
Output	-
Algorithm	1. Start. 2. Set a new event year with provided information. 3. Add event year to the database. 4. End.

Entity Name	Event
Method Name	setName
Input	Event name to be created
Output	-
Algorithm	1. Start 2. Set a new event name with provided information. 3. Add event name to the database. 4. End.

Entity Name	Event
Method Name	setVenue
Input	Event venue to be created

Output	-
Algorithm	1. Start. 2. Set a new event venue with provided information. 3. Asd event venue to the database. 4. End.

Entity Name	Event
Method Name	setMode
Input	Event mode to be created
Output	-
Algorithm	1. Start. 2. Set a new event mode with provided information. 3. Add event mode to the database. 4. End.

Entity Name	EventInfo
Method Name	createEventInfo
Input	Event name, Event description, Event venue, Event mode
Output	-
Algorithm	1. Start. 2. System displays a form for creating an event. 3. User fills in with the information of name, description,

	venue,mode for an event. 4. Submit the filled information. 5. End.

Entity Name	EventInfo
Method Name	editEventInfo
Input	Event name, Event description, Event venue, Event mode
Output	-
Algorithm	1. Start. 2. System will display a form that was filled in. 3. Edit the information for the selected event. 4. End.

Entity Name	EventInfo
Method Name	deleteEventInfo
Input	Event name, Event description, Event venue, Event mode
Output	-
Algorithm	1. Start. 2. Select the event that the user wants to delete. 3. Prompt user to confirm deletion 4. Delete the event information from the database. 5. The selected event will be directly deleted. 6. End.

Entity Name	Schedule
Method Name	createSchedule
Input	Event year, Event month, Event Day
Output	-
Algorithm	1. Start. 2. System displays a form for creating an event. 3. User fills in with the date information which include year, month and day for an event according with the event information. 4. Submit the filled information. 5. End.

Entity Name	Schedule
Method Name	editSchedule
Input	Event year, Event month, Event Day
Output	-
Algorithm	1. Start. 2. System will display a form that was filled in. 3. Edit the date information for the selected event. 4. End.

Entity Name	Schedule
Method Name	deleteSchedule
Input	Event year, Event month, Event Day
Output	-
Algorithm	1. Start. 2. Select the event that the user wants to delete. 3. Prompt user to confirm deletion 4. Delete the date information from the database. 5. The selected event will be directly deleted. 6. End.

4.2.2.2 Sequence Diagram

a) SD003 : Sequence diagram for Event Calendar

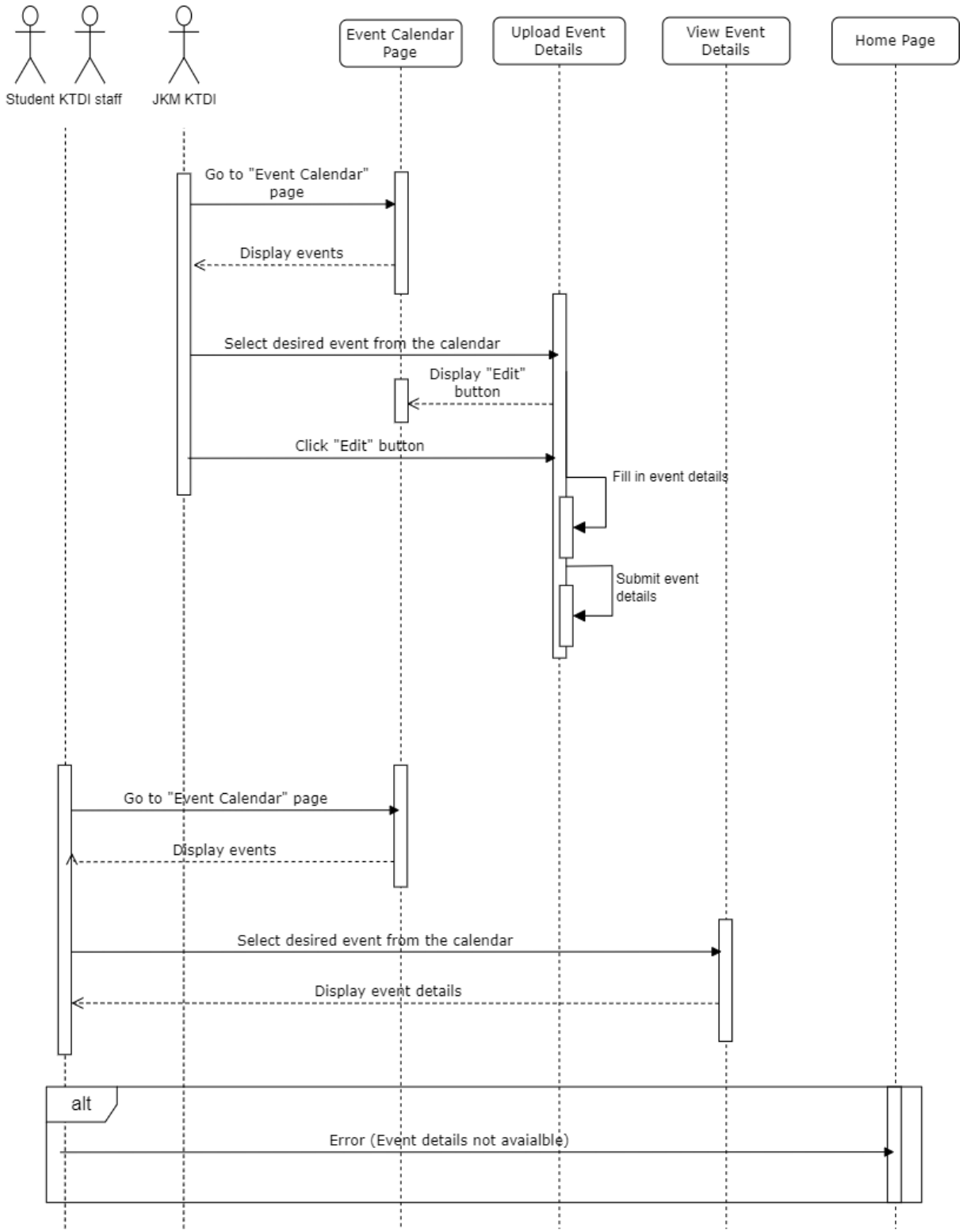

Figure 4.2.2.2.1: Sequence Diagram of <Event Calendar>

4.2.3 P003: <Event Preparation> Subsystem

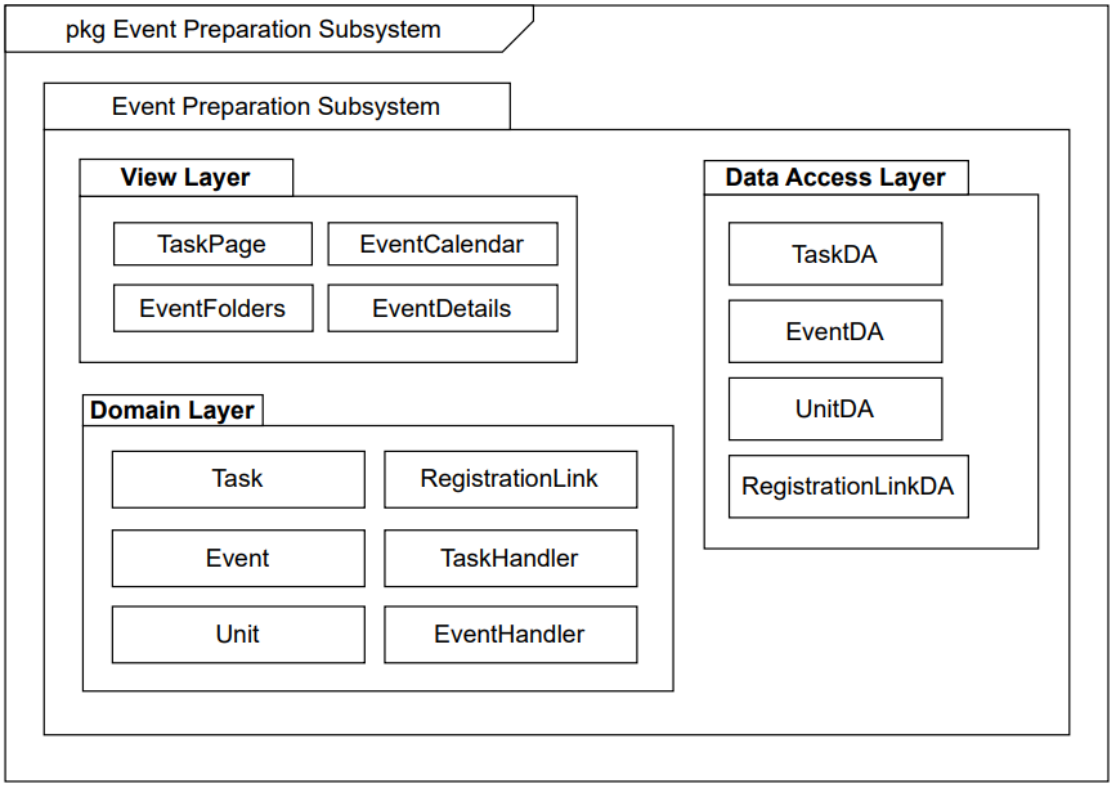

Figure 4.2.3.1: Package Diagram for Event Preparation Subsystem

4.2.3.1 Class Diagram

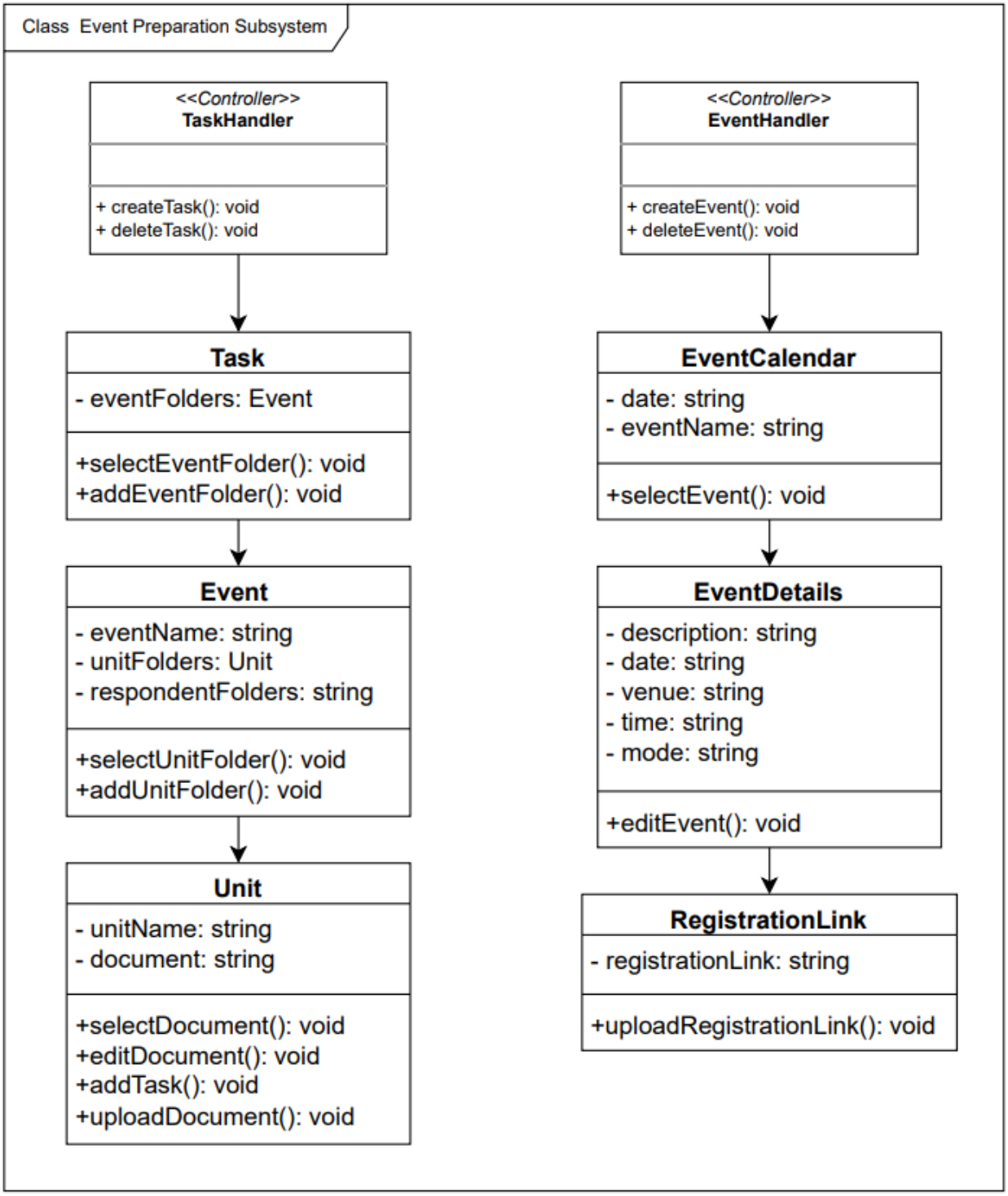

Figure 4.2.3.1.1: Class Diagram for Event Preparation Subsystem

Entity Name	Create Task
Method Name	createTask
Input	Task details
Output	None
Algorithm	1. Start 2. Input the task details. 3. Validate the task data. 4. If the task data is valid: 4.1. Perform any necessary additional operations or calculations related to the task. 4.2. Save the task in TaskDA. 5. Else if the task data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Delete Task
Method Name	deleteTask
Input	Task to be deleted
Output	None

Algorithm	1. Start 2. Input the task to be deleted. 3. Retrieve the task from the TaskDA. 4. If the task data is valid: 4.1. Delete the task from TaskDA. 5. Else if the task data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Select Event Folder
Method Name	selectEventFolder
Input	None
Output	Selected event folder to be used for further processing or display.
Algorithm	1. Start 2. Display a list of available event folders to the user. 3. The user selects an event folder. 4. Retrieve the selected event folder from the user's input. 5. Perform any necessary additional operations or calculations related to the selected event folder. 6. Use the selected event folder for further processing or display. 7. End

Entity Name	Add Event Folder

Method Name	addEventFolder
Input	Event folder details
Output	None
Algorithm	1. Start 2. Set the attributes of the event folder based on the user's input. 3. Validate the event folder data to ensure it meets any necessary criteria or constraints. 4. If the event folder data is valid: 5.1. Perform any necessary additional operations or calculations related to the event folder. 5.2. Save the event folder in EventDA. 5. If the event folder data is invalid: 6.1. Handle the validation error appropriately 6. End

Entity Name	Select Unit Folder
Method Name	selectUnitFolder
Input	None
Output	Selected unit folder to be used for further processing or display.

Algorithm	1. Start 2. Display a list of available unit folders for the selected event to the user. 3. The user selects a unit folder. 4. Retrieve the selected unit folder from the user's input. 5. Perform any necessary additional operations or calculations related to the selected unit folder. 6. Use the selected unit folder for further processing or display. 7. End.

Entity Name	Add Unit Folder
Method Name	addUnitFolder
Input	Unit Folder details
Output	None
Algorithm	1. Start 2. Set the attributes of the unit folder based on the user's input. 3. Validate the unit folder data to ensure it meets any necessary criteria or constraints. 4. If the unit folder data is valid: 4.1. Perform any necessary additional operations or calculations related to the unit folder. 4.2. Save the unit folder in UnitDA. 5. If the unit folder data is invalid: 5.1. Handle the validation error appropriately

	6. End

Entity Name	Select Document
Method Name	selectDocument
Input	None
Output	Selected document to be used for further processing or display.
Algorithm	1. Start 2. Display a list of available documents for the selected event to the user. 3. The user selects a document. 4. Retrieve the selected document from the user's input. 5. Perform any necessary additional operations or calculations related to the selected document. 6. Use the selected document for further processing or display. 7. End

Entity Name	Edit Document
Method Name	editDocument
Input	Document details
Output	None

Algorithm	1. Start 2. Select the document to be edited. 3. Validate the document to ensure it meets any necessary criteria or constraints. 4. If the document data is valid: 4.1. Perform any necessary additional operations or calculations related to the document. 4.2. Save the document. 5. If the document data is invalid: 5.1. Handle the validation error appropriately 6. End

Entity Name	Add Task
Method Name	addTask
Input	Task details
Output	None
Algorithm	1. Start 2. Input the task details. 3. Validate the task data to ensure it meets any necessary criteria or constraints. 4. If the task data is valid: 4.1. Perform any necessary additional operations or calculations related to the task. 4.2. Save the task data in TaskDA. 5. If the unit task data is invalid: 5.1. Handle the validation error appropriately

	6. End

Entity Name	Upload Document
Method Name	uploadDocument
Input	Document file
Output	None
Algorithm	1. Start 2. Input the document file. 3. Validate the document file to ensure it meets any necessary criteria or constraints. 4. If the document file is valid: 4.1. Perform any necessary additional operations or calculations related to the document. 4.2. Save the document. 5. If the document file is invalid: 5.1. Handle the validation error appropriately 6. End

Entity Name	Create Event
Method Name	createEvent
Input	Event details
Output	None

Algorithm	1. Start 2. Input the event details. 3. Validate the event data. 4. If the event data is valid: 4.1. Perform any necessary additional operations or calculations related to the task. 4.2. Save the event in EventDA. 5. Else if the event data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Delete Event
Method Name	deleteEvent
Input	Event to be deleted
Output	None
Algorithm	1. Start 2. Input the event to be deleted. 3. Retrieve the event from the EventDA. 4. If the event data is valid: 4.1. Delete the event from EventDA. 5. Else if the event data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Select Event
Method Name	selectEvent
Input	None
Output	Selected event to be used for further processing or display.
Algorithm	1. Start 2. Display a list of available events to the user. 3. The user selects an event. 4. Retrieve the selected event from the user's input. 5. Perform any necessary additional operations or calculations related to the selected event. 6. Use the selected event for further processing or display. 7. End

Entity Name	Edit Event
Method Name	editEvent
Input	Event details
Output	None
Algorithm	1. Start 2. Select the event to be edited. 3. Validate the event to ensure it meets any necessary criteria or constraints. 4. If the event data is valid:

	4.1. Perform any necessary additional operations or calculations related to the event. 5.2. Save the event. 5. If the event data is invalid: 5.1. Handle the validation error appropriately. 6. End

Entity Name	Upload Registration Link
Method Name	uploadRegistrationLink
Input	Registration link URL
Output	None
Algorithm	1. Start 2. Input the registration link. 3. Validate the registration link to ensure it meets any necessary criteria or constraints. 4. If the registration link is valid: 4.1. Update the event's registration link. 4.2. Save the updated event in RegistrationLinkDA. 5. If the registration link is invalid: 5.1. Handle the validation error appropriately. 6. End

4.2.3.2 Sequence Diagram

a) SD004: Sequence diagram for Task Distribution Event

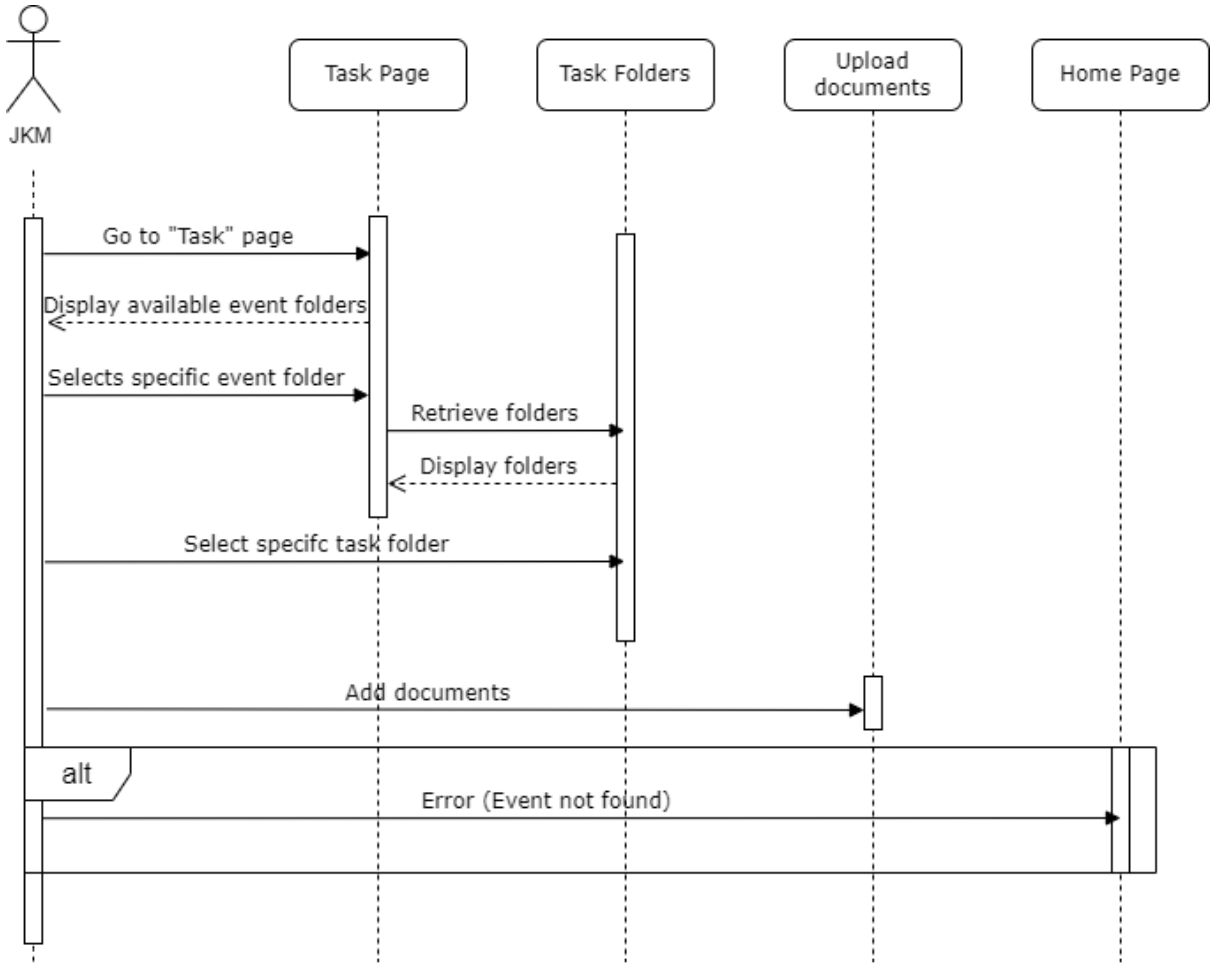

Figure 4.2.3.2.142.3.2.1: Sequence Diagram of Task Distribution Event

b) SD005: Sequence diagram for Upload Registration Link Event

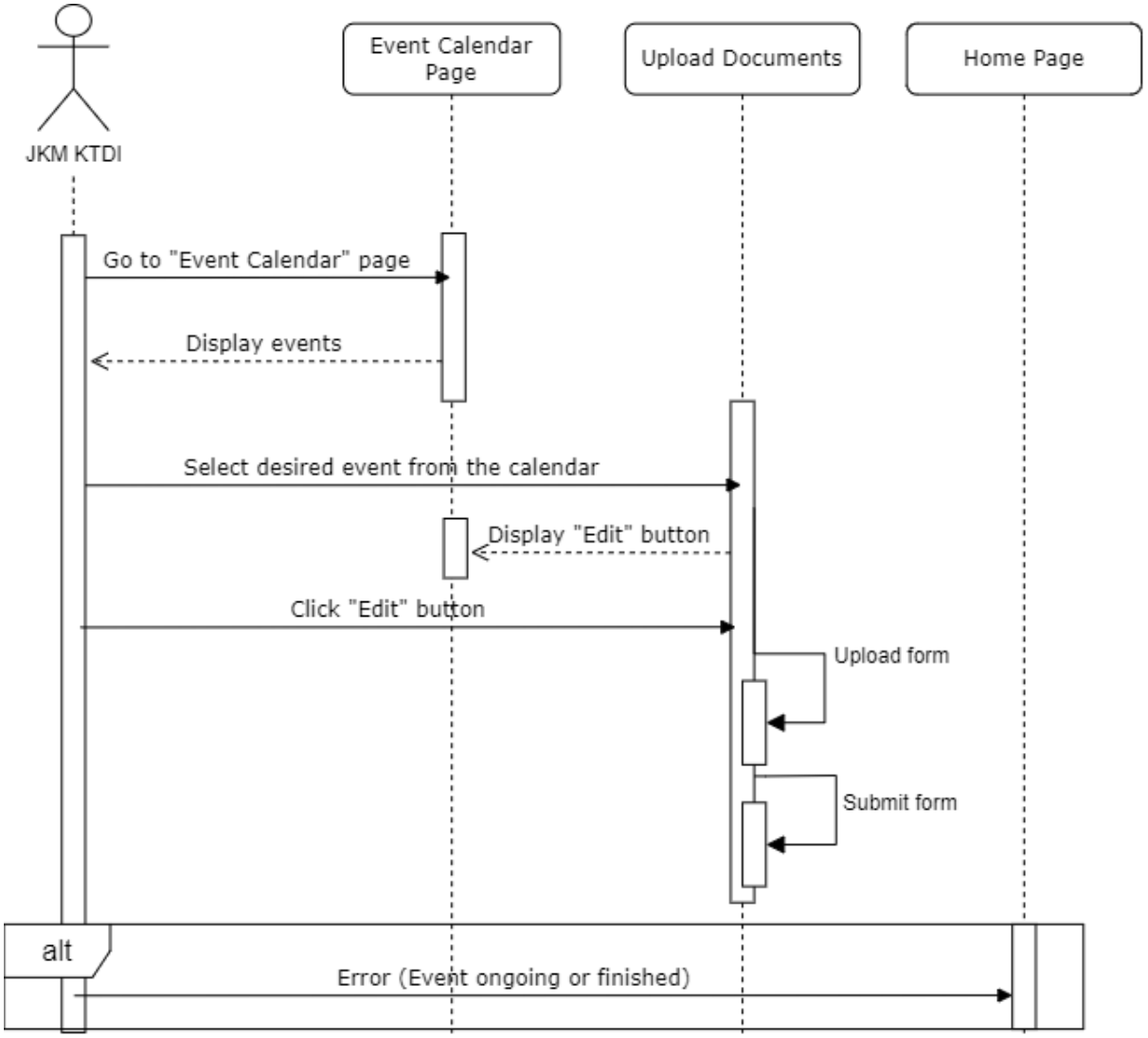

Figure 4.2.3.2.242.3.2.2: Sequence Diagram of Upload Registration Link Event

c) SD006: Sequence diagram for View Progress Event

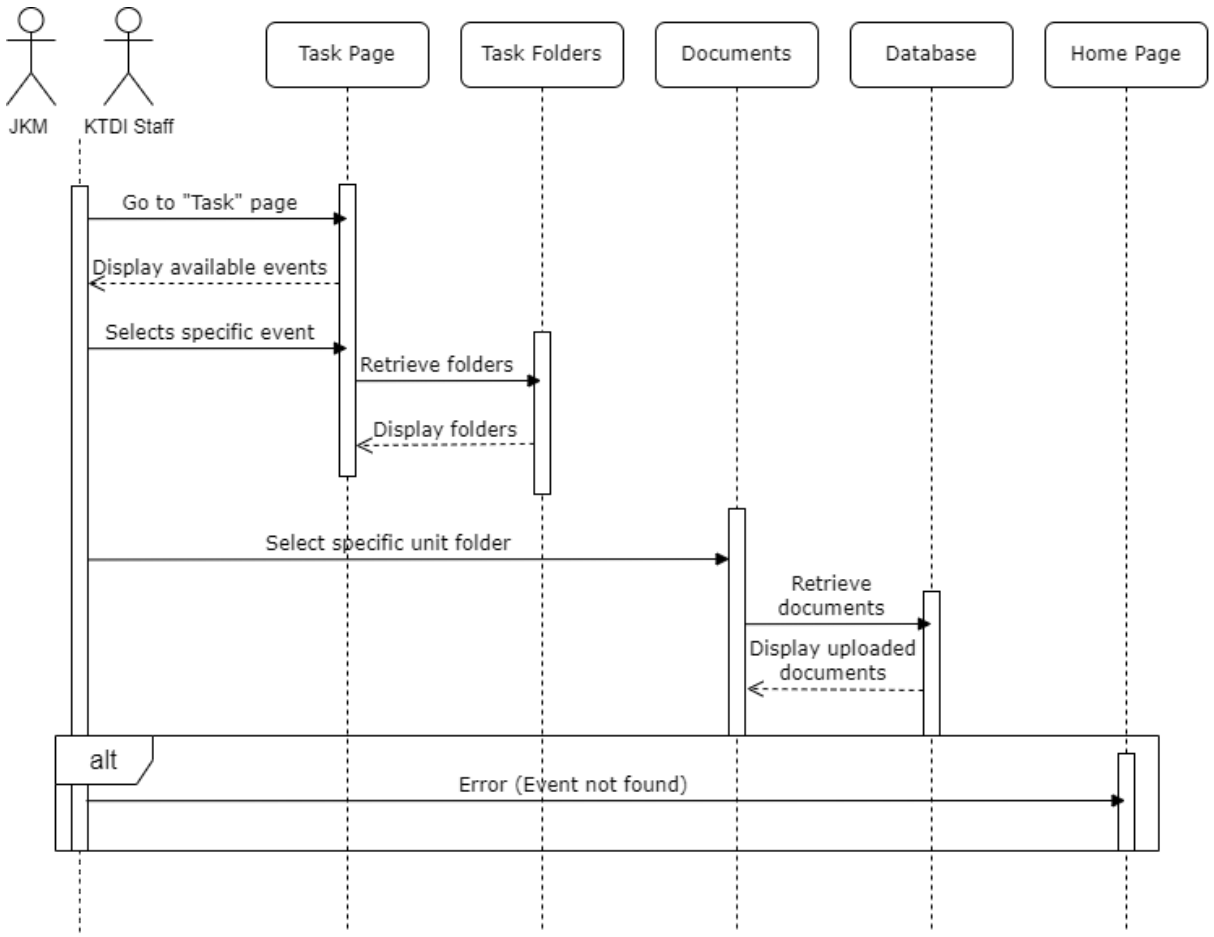

Figure 4.2.3.2.3: Sequence Diagram of View Progress Event

4.2.4 P004: <Register Event> Subsystem

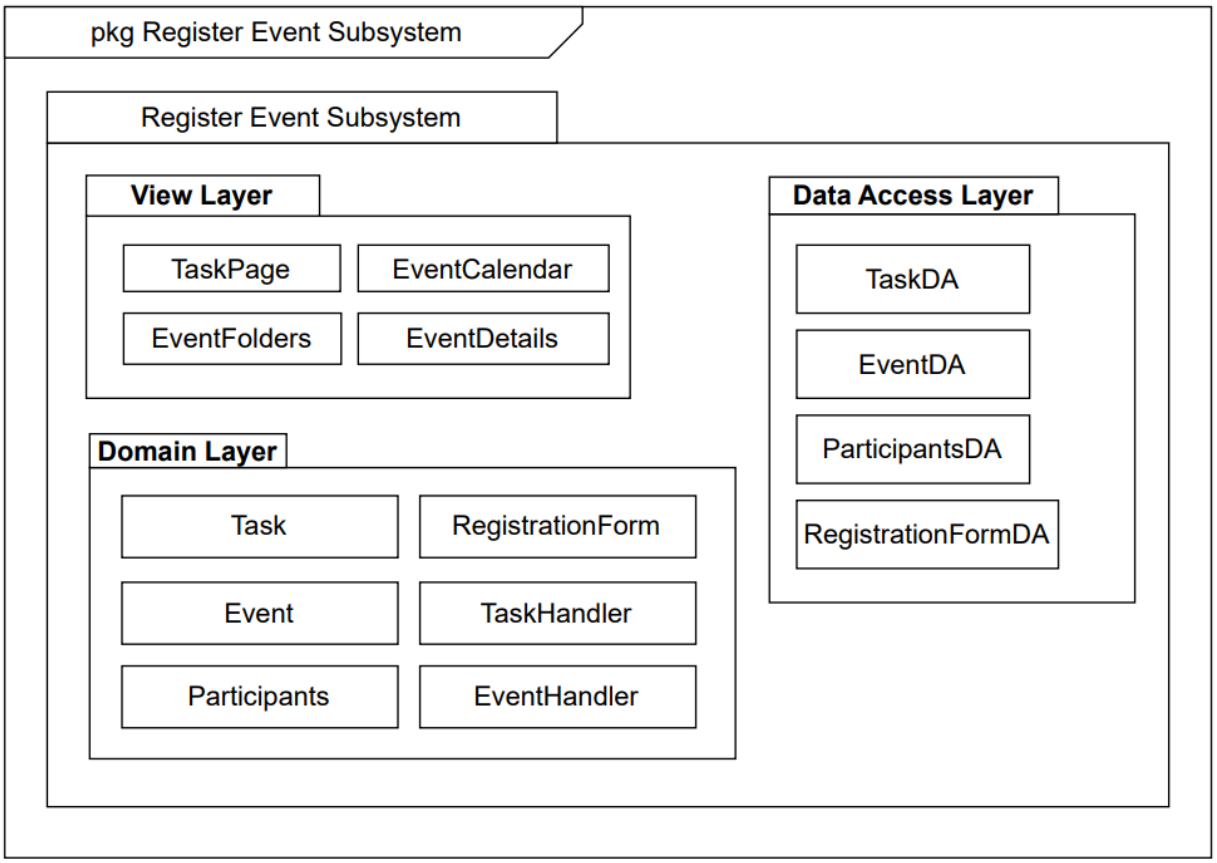

Figure 4.2.4.1: Package Diagram of Register Event

4.2.4.1 Class Diagram

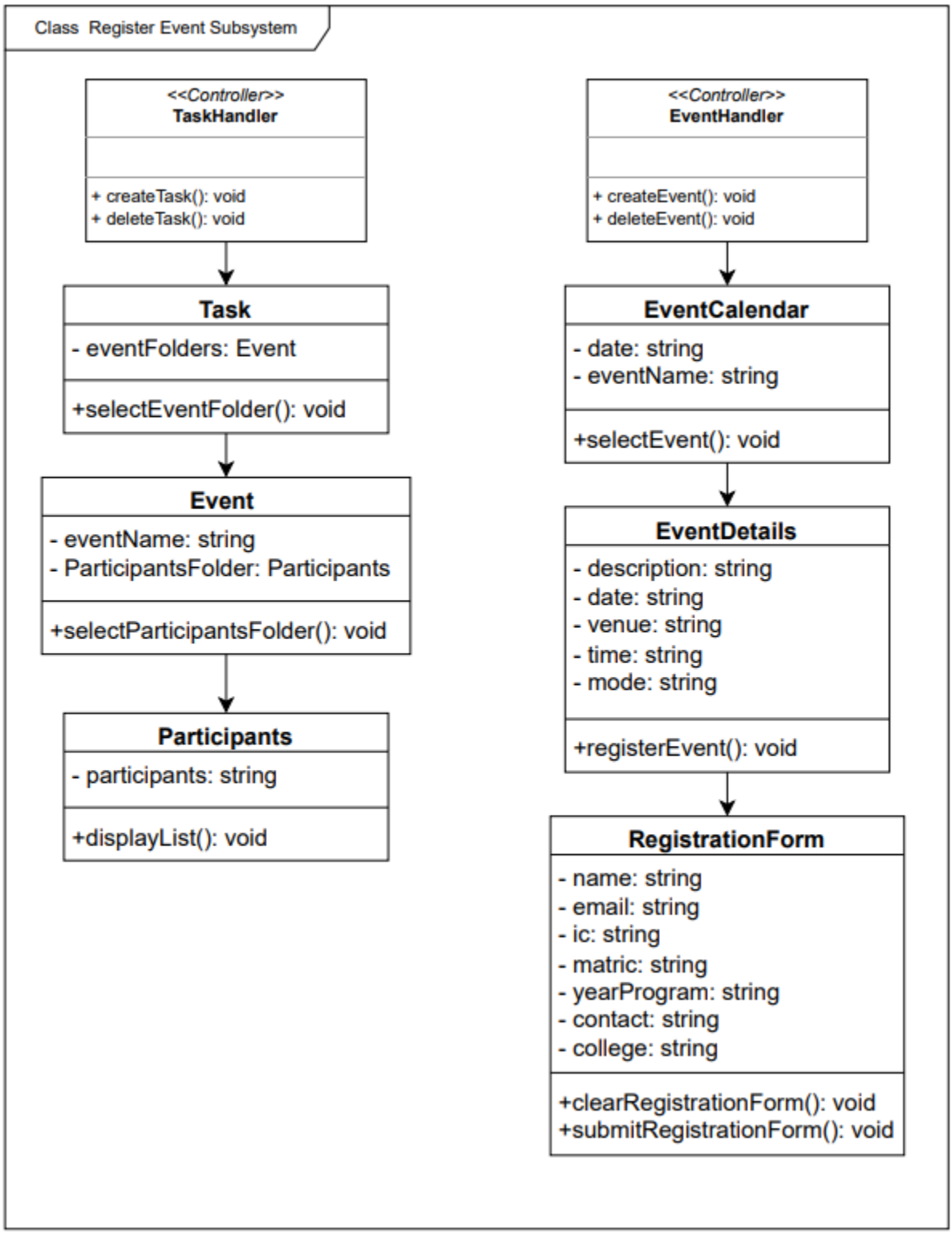

Figure 4.2.4.1.1: Class Diagram of Register Event

Entity Name	Create Task
Method Name	createTask
Input	Task details
Output	None
Algorithm	1. Start 2. Input the task details. 3. Validate the task data. 4. If the task data is valid: 4.1. Perform any necessary additional operations or calculations related to the task. 4.2. Save the task in TaskDA. 5. Else if the task data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Delete Task
Method Name	deleteTask
Input	Task to be deleted
Output	None

Algorithm	1. Start 2. Input the task to be deleted. 3. Retrieve the task from the TaskDA. 4. If the task data is valid: 4.1. Delete the task from TaskDA. 5. Else if the task data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Select Event Folder
Method Name	selectEventFolder
Input	None
Output	Selected event folder to be used for further processing or display.
Algorithm	1. Start 2. Display a list of available event folders to the user. 3. The user selects an event folder. 4. Retrieve the selected event folder from the user's input. 5. Perform any necessary additional operations or calculations related to the selected event folder. 6. Use the selected event folder for further processing or display. 7. End

Entity Name	Select Participants Folder

Method Name	selectParticipantsFolder
Input	None
Output	Selected participants folder to be used for further processing or display.
Algorithm	1. Start 2. Display participants folder for the selected event to the user. 3. The user selects the participant folder. 4. Retrieve the participant folder from the user's input. 5. Perform any necessary additional operations or calculations related to the participant folder. 6. Use the participant folder for further processing or display. 7. End.

Entity Name	Display List
Method Name	displayList
Input	None
Output	Participant list to be used for further processing or display.
Algorithm	1. Start 2. Retrieve the list of participants from ParticipantsDA. 3. Present the participant list to the user. 4. End

Entity Name	Create Event
Method Name	createEvent
Input	Event details
Output	None
Algorithm	1. Start 2. Input the event details. 3. Validate the event data. 4. If the event data is valid: 4.1. Perform any necessary additional operations or calculations related to the task. 4.2. Save the event in EventDA. 5. Else if the event data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Delete Event
Method Name	deleteEvent
Input	Event to be deleted
Output	None

Algorithm	1. Start 2. Input the event to be deleted. 3. Retrieve the event from the EventDA. 4. If the event data is valid: 4.1. Delete the event from EventDA. 5. Else if the event data is invalid: 5.1 Handle the validation error appropriately. 6. End

Entity Name	Select Event
Method Name	selectEvent
Input	None
Output	Selected event to be used for further processing or display.
Algorithm	1. Start 2. Display a list of available events to the user. 3. The user selects an event. 4. Retrieve the selected event from the user's input. 5. Perform any necessary additional operations or calculations related to the selected event. 6. Use the selected event for further processing or display. 7. End

Entity Name	Register Event

Method Name	registerEvent
Input	Personal details
Output	None
Algorithm	1. Start 2. Select the event to be registered for. 3. Validate the event to ensure it meets any necessary criteria or constraints. 4. If the event is open for registration: 4.1. Display the registration form. 4.2. Input the personal details. 5. If the event is closed for registration: 5.1. Handle the validation error appropriately. 6. End

Entity Name	Submit Registration Form
Method Name	submitRegistrationForm
Input	Input information
Output	None
Algorithm	1. Start 2. Validate the input information to ensure it meets any necessary criteria or constraints. 3. If the input information is valid: 4.1. Update the input information. 4.2. Save the input information in RegistrationFormDA.

	4. If the input information is invalid: 5.1. Handle the validation error appropriately. 5. End

Entity Name	Clear Registration Form
Method Name	clearRegistrationForm
Input	None
Output	Clear the registration form fields
Algorithm	1. Start 2. Retrieve the registration form fields from RegistrationFormDA. 3. Clear the values of each registration form field to their initial or default state. 4. Update related visual indicators to reflect the cleared state of the registration form. 5. End

4.2.4.2 Sequence Diagram

a) SD007: Sequence diagram for Register Event

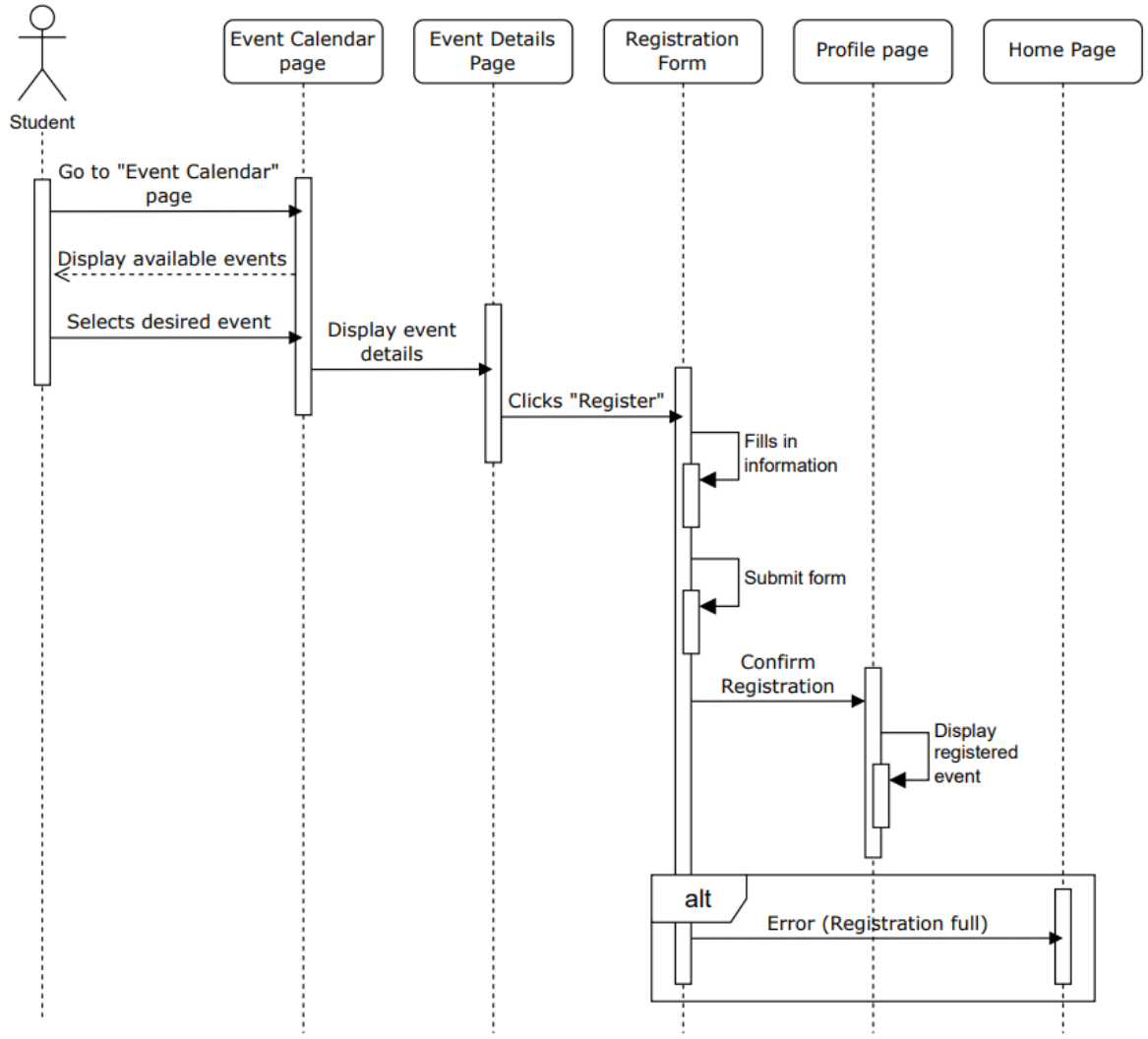

Figure 4.2.4.2.1: Sequence Diagram of Register Event

b) SD008: Sequence diagram for View Registration Event

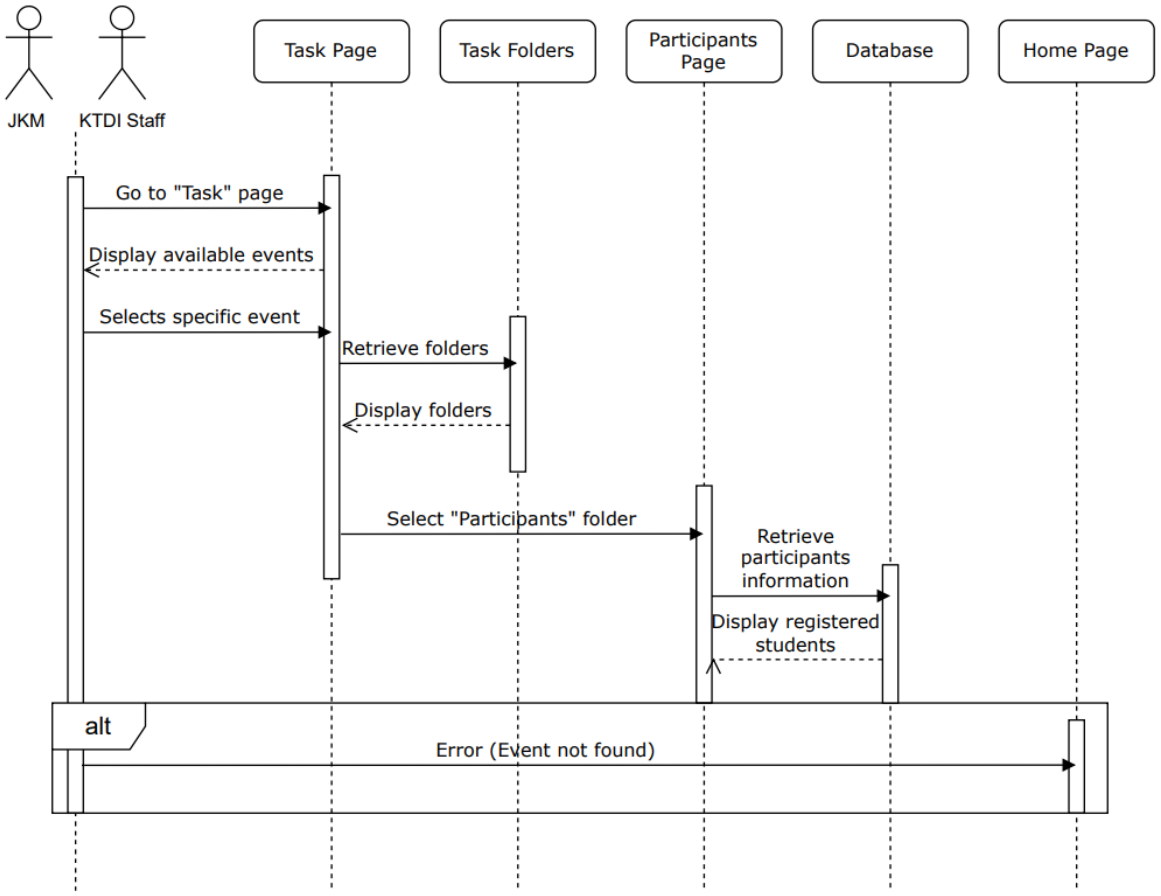

Figure 4.2.4.2.2: Sequence Diagram of View Registration Event

4.2.5 P005: <Attendance And Feedback> Subsystem

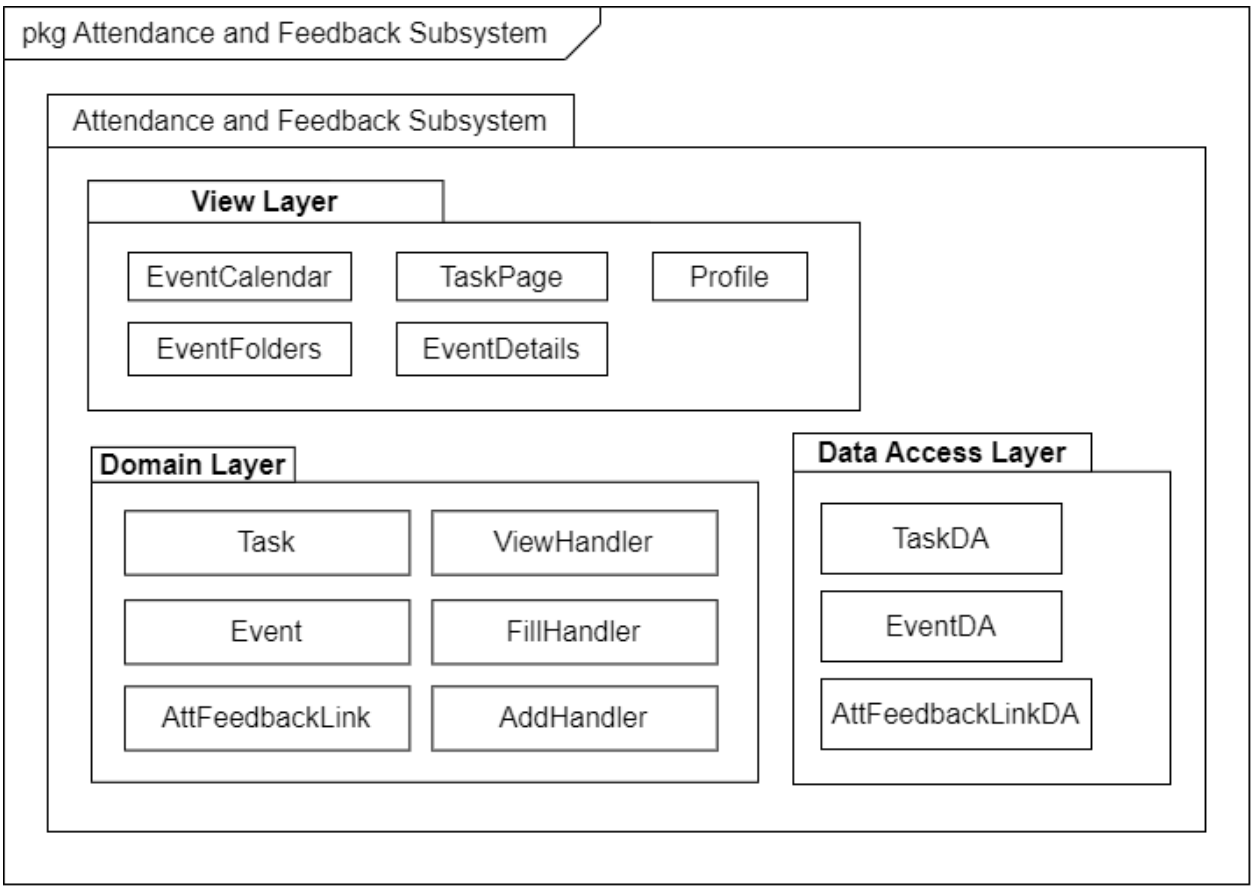

Figure 4.3.5.1: Package Diagram for Attendance and Feedback Subsystem

4.2.5.1 Class Diagram

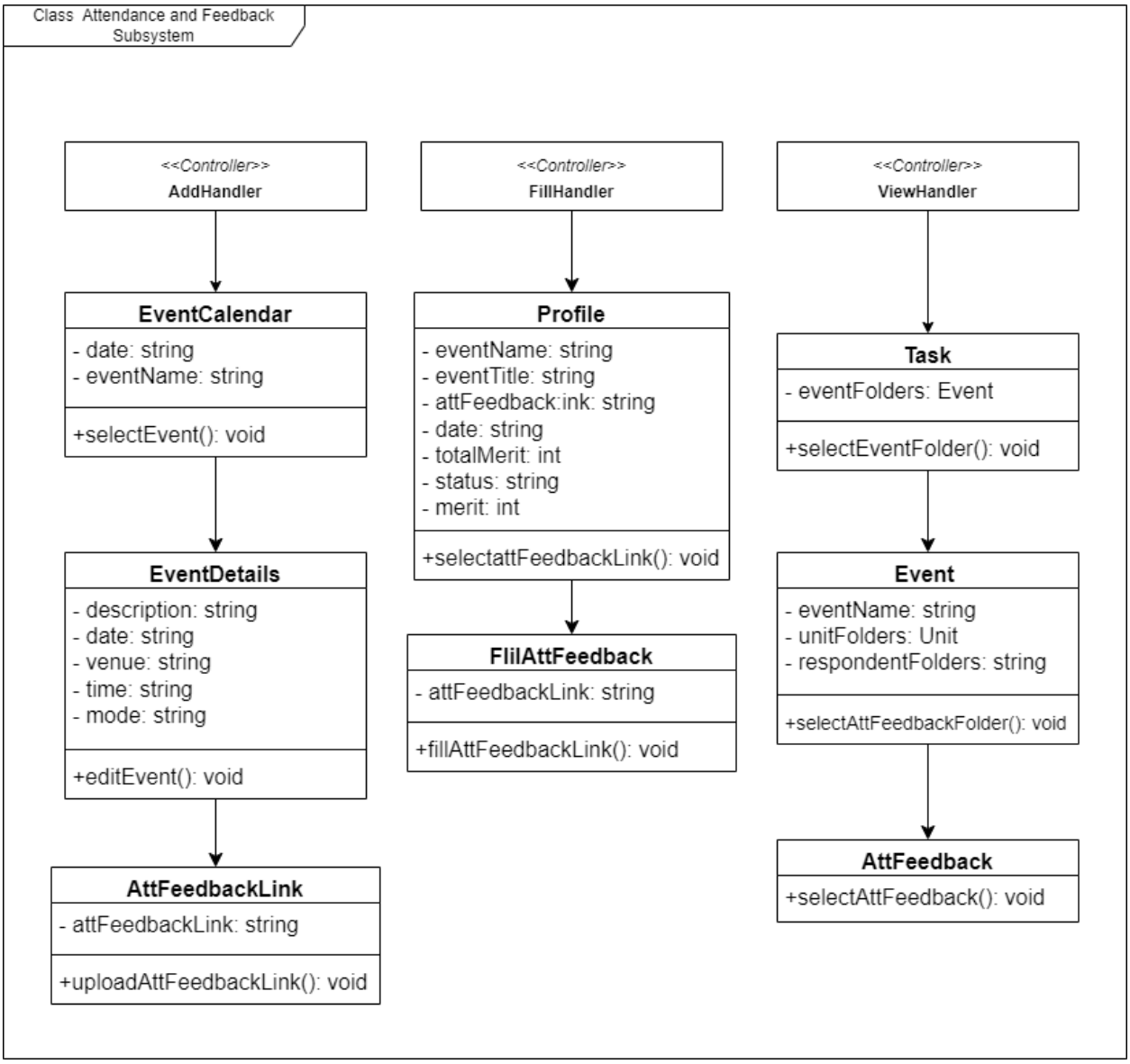

Figure 4.2.5.1.1: Class Diagram of Attendance and Feedback Event

Entity Name	Select Event
Method Name	selectEvent
Input	None
Output	Selected event to be used for further display.
Algorithm	1. Start 2. Display a list of available events to the user. 3. The user selects an event. 4. Retrieve the selected event from the user's input. 5. Perform any necessary additional operations or calculations related to the selected event. 6. Use the selected event for further processing or display. 7. End

Entity Name	Edit Event
Method Name	editEvent
Input	Event details
Output	None
Algorithm	1. Start 2. Select the event to be edited. 3. Validate the event to ensure it meets any necessary criteria or constraints. 4. If the event data is valid:

	4.1. Perform any necessary additional operations or calculations related to the event. 4.2. Save the event. 5. If the event data is invalid: 5.1. Handle the validation error appropriately. 6. End

Entity Name	Upload Attendance and Feedback Link
Method Name	uploadAttFeedbackLink
Input	Attendance and Feedback URL
Output	None
Algorithm	6. Start 7. Input the attendance and feedback link. 8. Validate the attendance and feedback link to ensure it meets any necessary criteria or constraints. 9. If the attendance and feedback link is valid: 4.1. Update the event's attendance and feedback link. 4.2. Save the updated attendance and feedback in AttFeedbackLinkDA. 10. If the attendance and feedback link is invalid: 5.1. Handle the validation error appropriately. 11. End

Entity Name	Select Attendance And Feedback Link

Method Name	selectattFeedbackLink
Input	Event details
Output	None
Algorithm	1. Start 2. Select the attendance and feedback link to be filled according to the event attended. 3. Validate the event details to ensure that it is available 4. If the attendance and feedback link data is available: 4.1. Display the attendance and feedback form details to the user 5. If the event data is unavailable: 5.1. Handle the validation error appropriately. 6. End

Entity Name	Fill Attendance and Feedback Link
Method Name	fillAttFeedbackLink
Input	Attendance and Feedback URL
Output	None

Algorithm	1. Start 2. Fill out the attendance and feedback link. 3. Validate the attendance and feedback link to ensure it meets any necessary criteria or constraints. 4. If all the blanks in the attendance and feedback link are filled in: 4.2. Save the updated attendance and feedback in AttFeedbackLinkDA. 5. If some / all of the blanks on the attendance and feedback link are not filled in: 5.1. Handle the validation error appropriately. 6. End

Entity Name	Select Event Folder
Method Name	selectEventFolder
Input	None
Output	Selected event folder to be used for further processing or display.
Algorithm	1. Start 2. Display a list of available event folders to the user. 3. The user selects an event folder. 4. Retrieve the selected event folder from the user's input. 5. Perform any necessary additional operations or calculations related to the selected event folder. 6. Use the selected event folder for further processing or display. 7. End

Entity Name	Select Attendance and Feedback Folder
Method Name	selectAttFeedbackFolder
Input	None
Output	Selected attendance and feedback folder to be used for further processing or display.
Algorithm	1. Start 2. Display the attendance and feedback folder for the selected event to the user. 3. The user selects the folder. 4. Retrieve the attendance and feedback folders from the user's input. 5. Perform any necessary additional operations or calculations related to the selected attendance and feedback folder. 6. Use the attendance and feedback folder for further processing or display. 7. End.

Entity Name	Select Specific Attendance and Feedback Form to View
Method Name	selectAttFeedback
Input	None
Output	Select a specific attendance and feedback folder to be used for further processing or display.

Algorithm	1. Start 2. Display a list of filled out attendance and feedback forms for the selected event to the user. 3. The user selects a specific attendance and feedback form. 4. Retrieve the attendance and feedback form from the user's input. 5. Perform any necessary additional operations or calculations related to the selected attendance and feedback forms. 6. Use the attendance and feedback forms for further processing or display. 7. End.

4.2.5.2 Sequence Diagram

a) SD009: Sequence diagram for Upload Attendance and Feedback Link Event

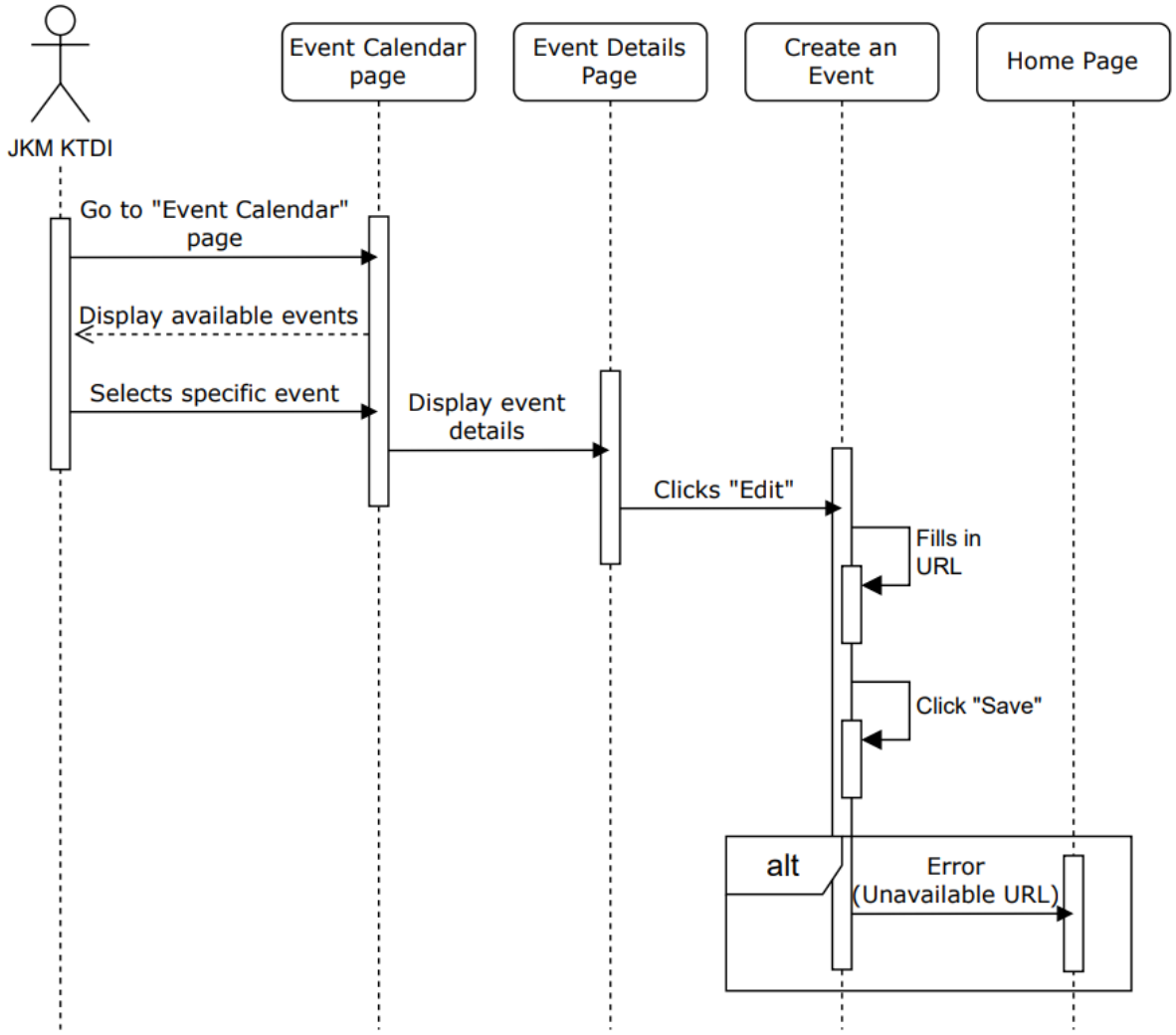

Figure 4.2.5.2.1: Sequence Diagram of Upload Attendance and Feedback Link Event

b) SD010: Sequence diagram for Fill Attendance and Feedback Link Event

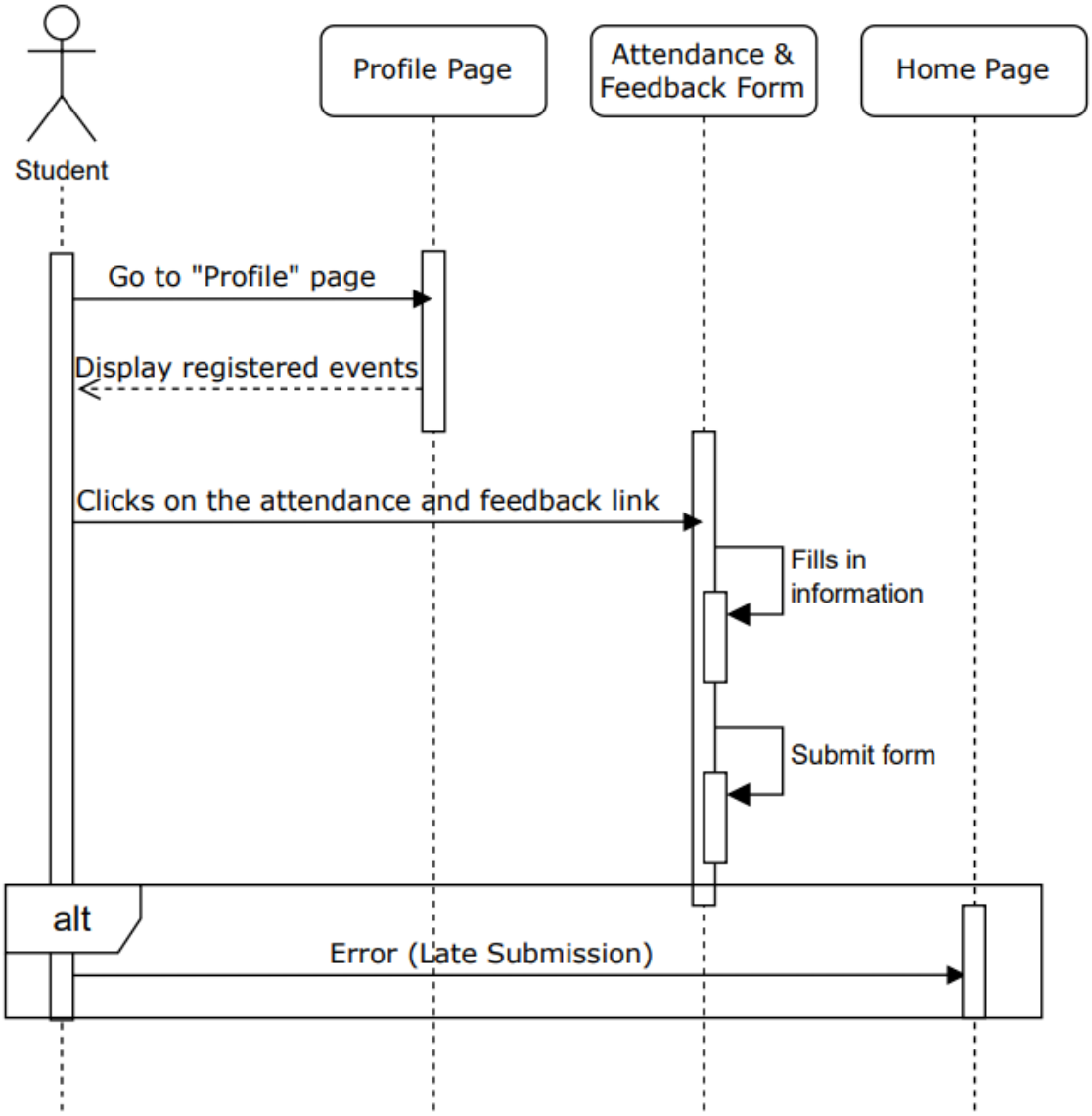

Figure 4.2.5.2.2: Sequence Diagram of Fill Attendance And Feedback Link Event

c) SD011: Sequence diagram for View Attendance And Feedback Link Event

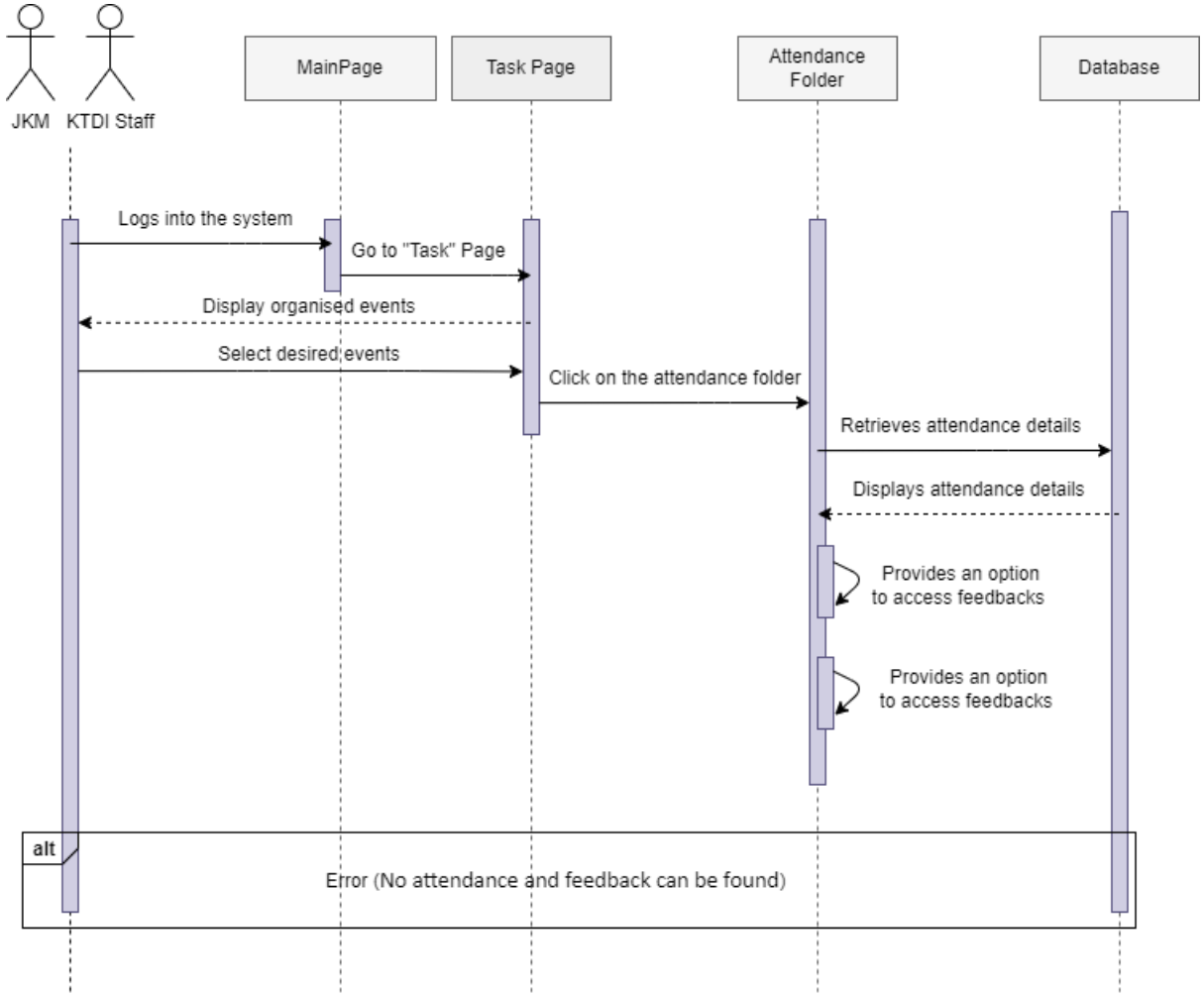

Figure 4.2.5.2.3: Sequence Diagram of View Attendance And Feedback Link Event

4.2.6 P006: <Active Quota> Subsystem

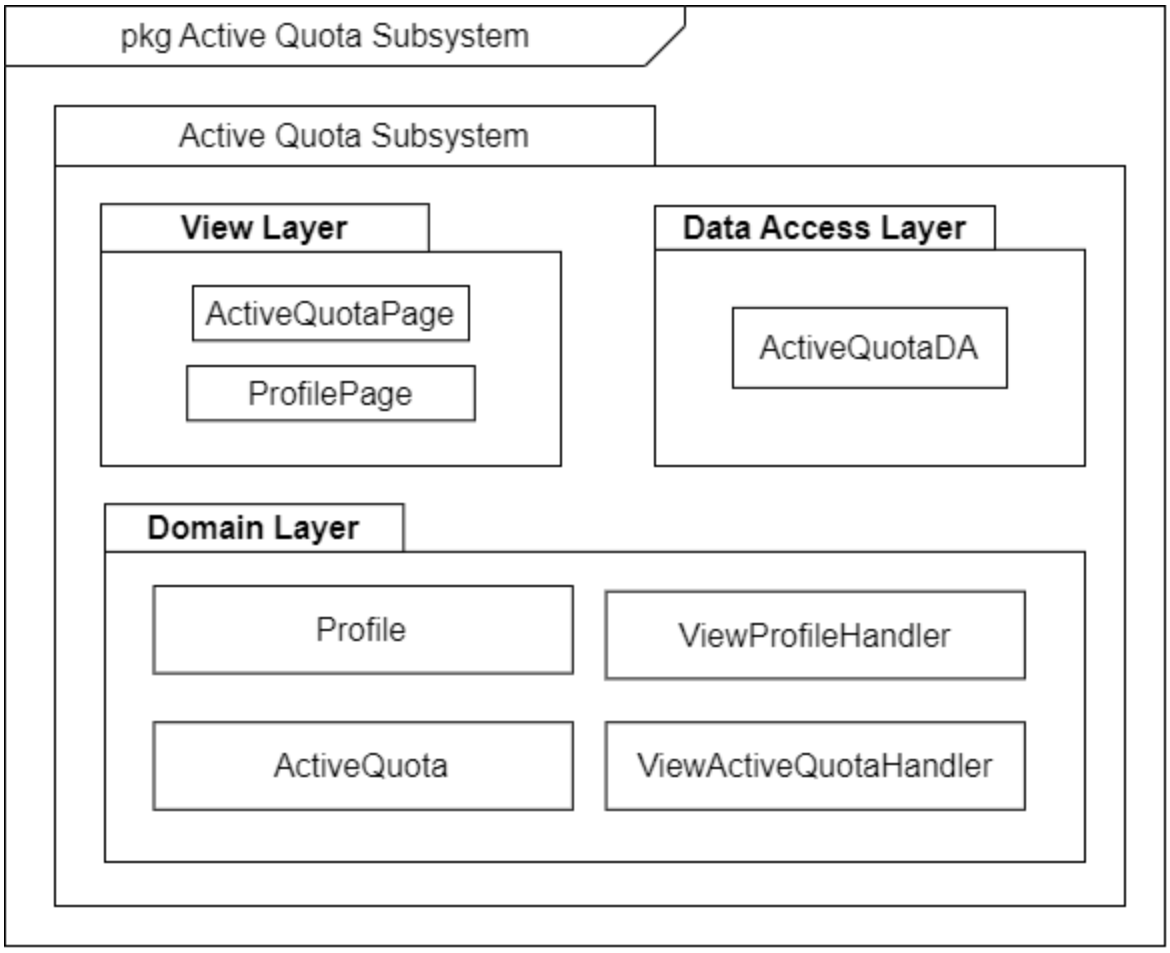

Figure 4.2.6.1: Package Diagram for Active Quota Subsystem

4.2.6.1 Class Diagram

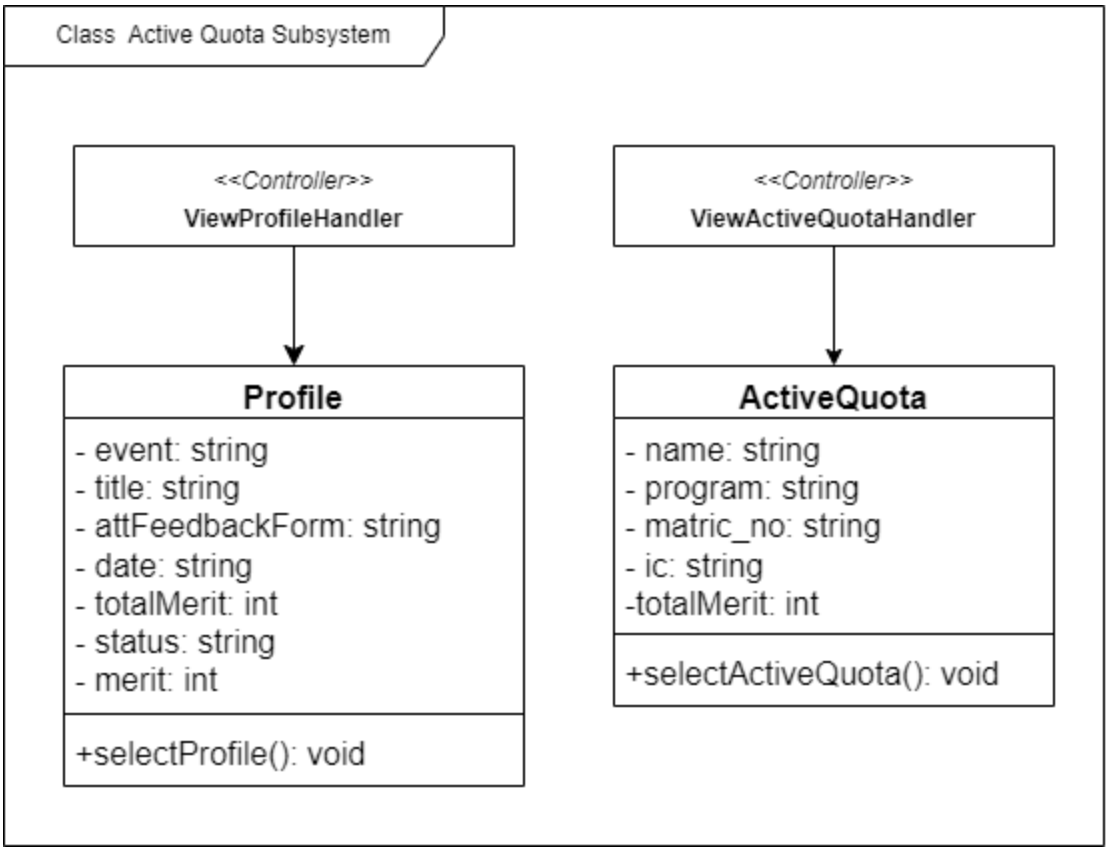

Figure 4.2.6.1.1: Class Diagram of Active Quota Event

Entity Name	Select Profile
Method Name	selectProfile
Input	None
Output	Selected profile to be used for further display.
Algorithm	1. Start 2. Display user's profile. . 3. Retrieve the profile details from the user's input.

	4. View merit earned for each event and active quota status. 5. End

Entity Name	Select Active Quota
Method Name	selectActiveQuota
Input	None
Output	Selected active quota tab to be used for further display.
Algorithm	1. Start 2. Display a list of students' information and total merit earned. 3. View students' total merit earned. 4. End

4.2.6.2 Sequence Diagram

a) SD012: Sequence diagram for View Profile Event

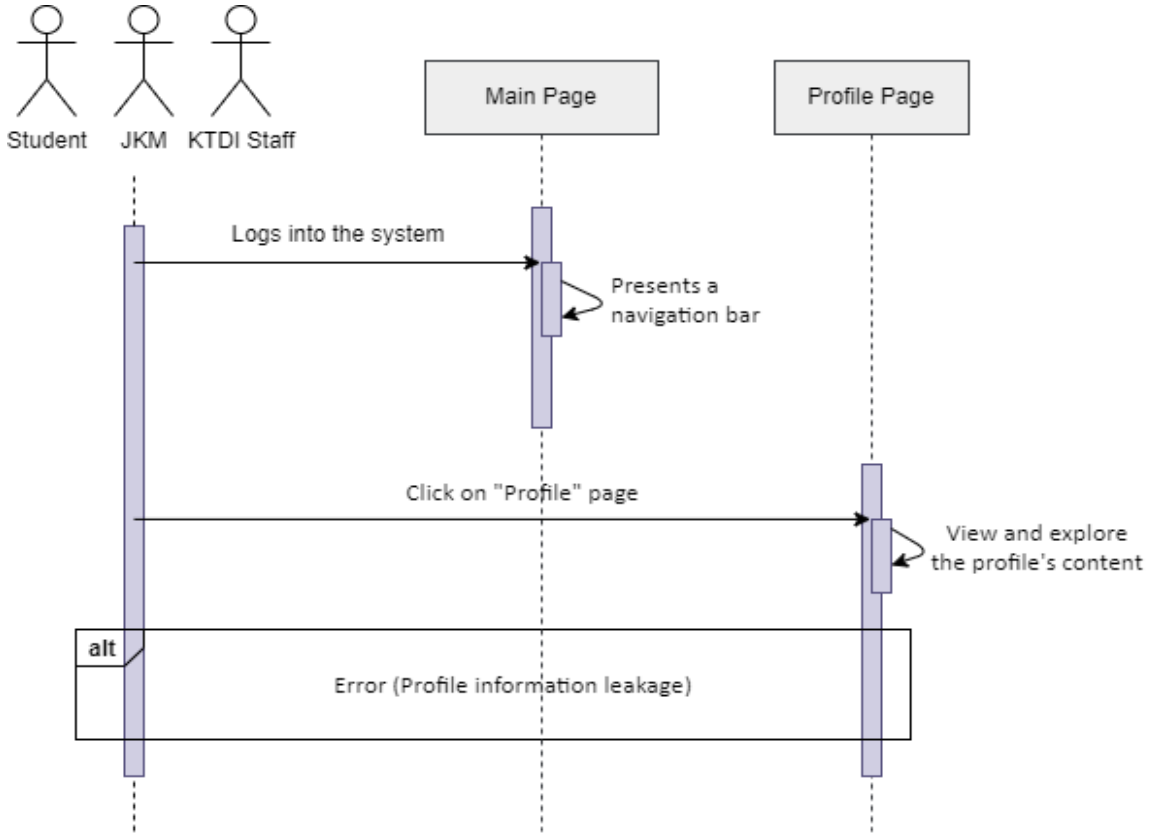

Figure 4.2.6.2.1: Sequence Diagram of View Profile Event

b) SD013: Sequence diagram for View Active Quota Event

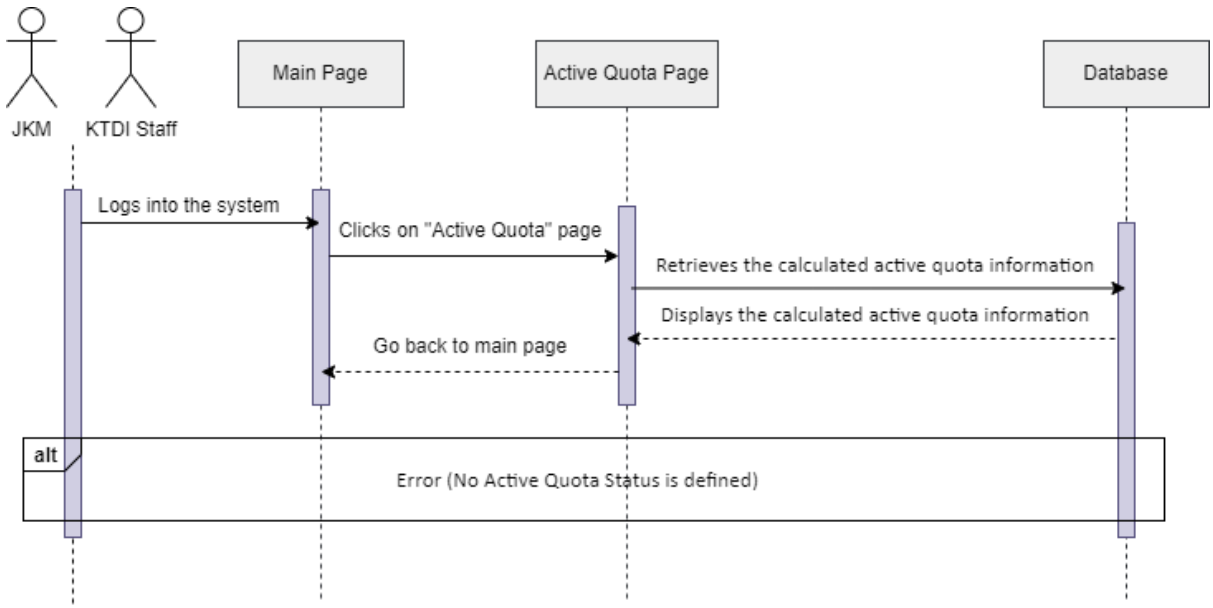

Figure 4.2.6.2.2: Sequence Diagram of View Active Quota Event

5. Data Design

5.1 Data Description

The major data or systems entities are stored into a relational database named as KtdieventDA, processed and organised into *n* entities as listed in Table 5.1.

Table 5.1: Description of Entities in the Database

No.	Entity Name	Description
1.	EventInfo	Used to store events information which can be created, edited, deleted by JKM KTDI or viewed by the students or KTDI staff members.
2.	Schedule	Used to store schedule information which is linked to the event information.
3.	TaskDA	Used to store folder information about the different events created by JKM KTDI.
4.	EventDA	Used to store folder information about the relevant units for a created event.
5.	UnitDA	Used to store the document information which consists of detailed information for a respective event.
6.	RegistrationLinkDA	Used to store the registration link information according to the updated event.

7.	ParticipantsDA	Used to store the information of participants list for a respective event.
8.	RegistrationFormDA	Used to store the personal information of each participant in registering an event.
9.	AttFeedbackLinkDA	Used to store attendance and feedback information which are filled by each participant through a link for a respective event.
10.	ActiveQuotaDA	Used to store the calculated active quota information for each student.

5.2 Data Dictionary

5.2.1 Entity: <EventInfo>

Attribute Name	Type	Description
name	string	Name of event
description	string	Description of event
venue	string	Venue of event
mode	string	Mode of event

5.2.2 Entity: <Schedule>

Attribute Name	Type	Description
year	int	Organised event year
month	int	Organised event month
day	int	Organised event day

5.2.3 Entity: <TaskDA>

Attribute Name	Type	Description
eventFolders	Event	Event folder

5.2.4 Entity: <EventDA>

Attribute Name	Type	Description

eventName	string	Name of event
unitFolders	Unit	Folder for different units to do their tasks
respondentFolders	string	Folder which consists of attendance and feedback from the participants

5.2.5 Entity: <UnitDA>

Attribute Name	Type	Description
unitName	string	Name of units
document	string	Document uploaded by each respective unit

5.2.6 Entity: <RegistrationLinkDA>

Attribute Name	Type	Description
registrationLink	string	Link of registration for the students who want to register for an event

5.2.7 Entity: <ParticipantsDA>

Attribute Name	Type	Description
participants	string	List of participant information for a respective event.

5.2.8 Entity: <RegistrationFormDA>

Attribute Name	Type	Description
name	string	Name of the registrant
email	string	Email of the registrant
ic	string	Identity card number of the registrant
matric	string	Matric number of the registrant
yearProgram	string	Study year and programme of the registrant
contact	string	Contact number of the registrant
college	string	College stayed by the registrant

5.2.9 Entity: <AttFeedbackLinkDA>

Attribute Name	Type	Description
attFeedbackLink	string	Link of attendance and feedback for the participants who have attended in a respective event

5.2.10 Entity: <ActiveQuotaDA>

Attribute Name	Type	Description

name	string	Name of the student
program	string	Study programme of the student
matric_no	string	Matric number of the student
ic	string	Identity card number of the student
totalMerit	int	Total merit points earned by the student

6. User Interface Design

6.1 Overview of User Interface

The interface includes a general design for every user and a specified interface type for the dedicated users: JKM KTDI, KTDI staff members, and students. Firstly, the interface that is displayed to every user is the intro page, which displays the title of our system, and users can choose whether they want to sign up or log in. Typically, sign-up is for new users, and login is for users who already have an account. If the user chooses to sign up, they are required to fill in their personal information and choose their appropriate roles. As for logging in, the users will dive into using the system with their respective credentials. Subsequently, there is a forgot password and reset password interface in case the user has forgotten their password and would like to reset it in order to use the system.

As for JKM KTDI, there will be a list of segments that will be displayed for them to navigate and utilise in their dashboard. They are also able to view graphs and statistics of the current number of residents working or living in KTDI, as well as the number of participants in the events that were recently held. Next, by clicking into the "Event Calendar" tab, JKM KTDI will be able to view events that have taken place, are ongoing, or are going to take place. With this feature, they will be able to plan their events in advance so that no events clash with one another. By clicking on an existing event that is already on the calendar, more information about the event will be displayed. They can edit the information, which consists of title, date, venue, and mode, as well as upload registration, attendance, and feedback links. Besides, if a student registers for a particular event, they can view the list in the "Task" tab. This is the same for attendance and feedback filled in by the students who participated in the events. In the same tab as well, JKM KTDI will be able to distribute their tasks according to their roles in the event, such as the director, secretary, treasurer, and exco units. JKM KTDI can add folders as required for the events. By clicking into one of the folders, each member of the respective group is able to upload and edit their related documents. In addition, the "Active Quota" tab allows the JKM KTDI members to ease their jobs by arranging the total merits of the students for priority in booking the hostel rooms for the next semester. This interface displays the personal information of the students and their total merit for the semester.

Moving on to the KTDI staff members' interface, it consists of similar tasks to JKM KTDI. In the main dashboard, KTDI staff members are also able to view the graphs and statistics of residents and events. The information about the event posted by JKM KTDI can then be seen in the "Event Calendar" tab, where they can also keep up with JKM KTDI's work in progress. On top of that, in the "Task" tab, staff members can only view updates and documents uploaded by JKM KTDI. This allows them to keep track of what the JKM KTDI has done in order to make the event successful. They are able to access any event folder and the documents freely; however, they are unable to edit them. Next, the KTDI staff members also have the "Active Quota" tab, which displays the total merits of each student, helping them to easily calculate and document whoever has priority in booking hostel rooms next semester.

Last but not least, the student interface has two main segments: the event calendar and their profile. Students can view events that have already been added to the event calendar by the JKM KTDI. Upon pressing into the event, they are able to view the poster as well as the event descriptions. If they are interested, they may register for the event. On their profile page, they are able to view their participation in events. Once they have registered for the event, an attendance and feedback form will also appear here. To ensure that the student attended the event, they must enter a secret code that is displayed during the event held by JKM KTDI. Besides, they are also able to view the merit points earned for each event they participated in, as well as their total merit and active quota status, so they will be able to know in advance if they can get priority for hostel booking.

6.2 Screen Images

6.2.1 General

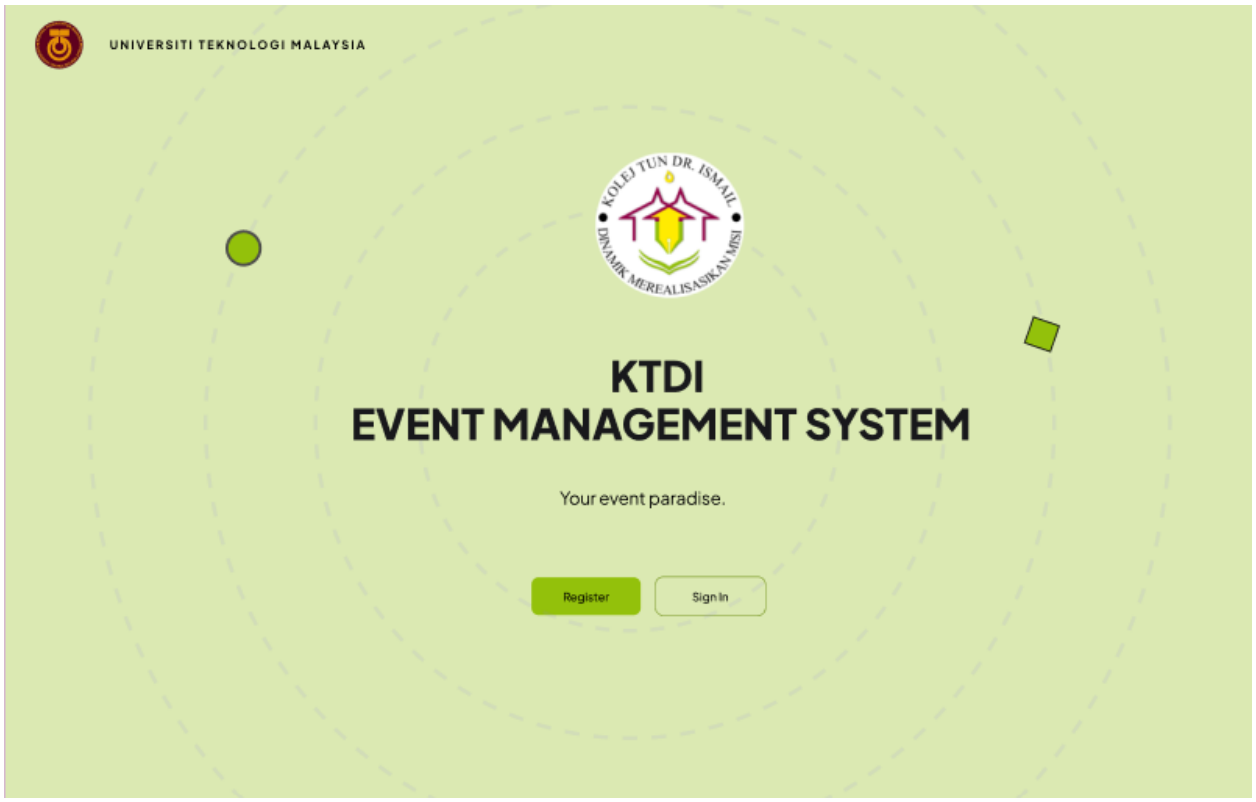

Figure 6.2.1.1: Interface for <Intro Page>

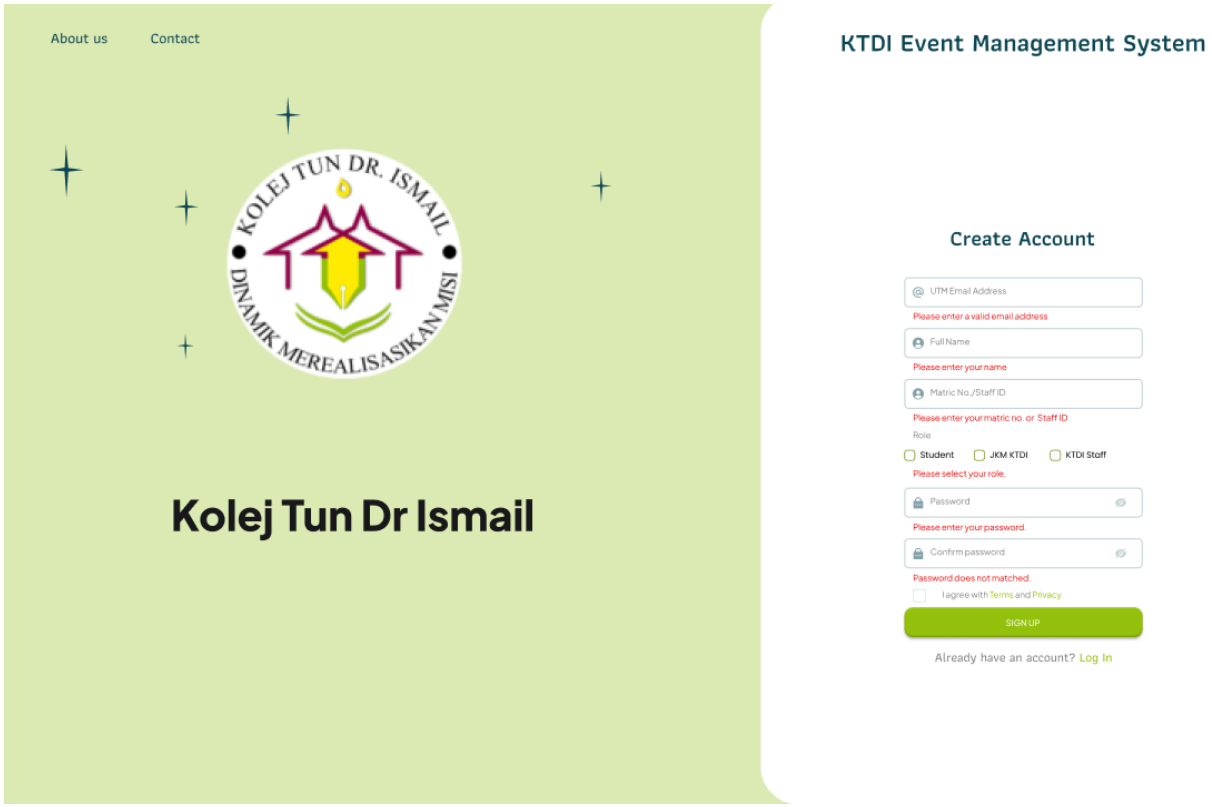

Figure 6.2.1.2: Interface for <Register Account>

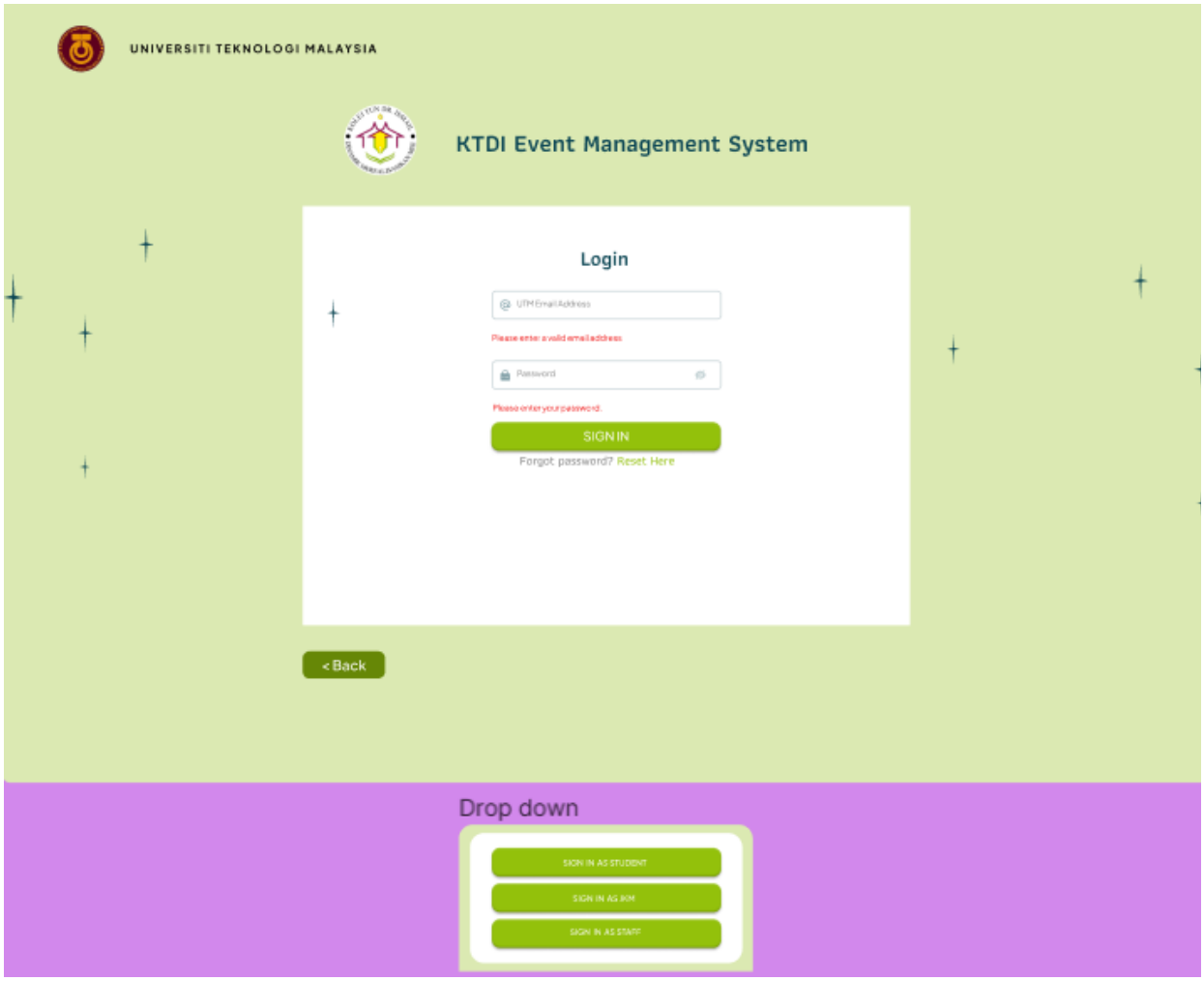

Figure 6.2.1.3: Interface for <Sign In>

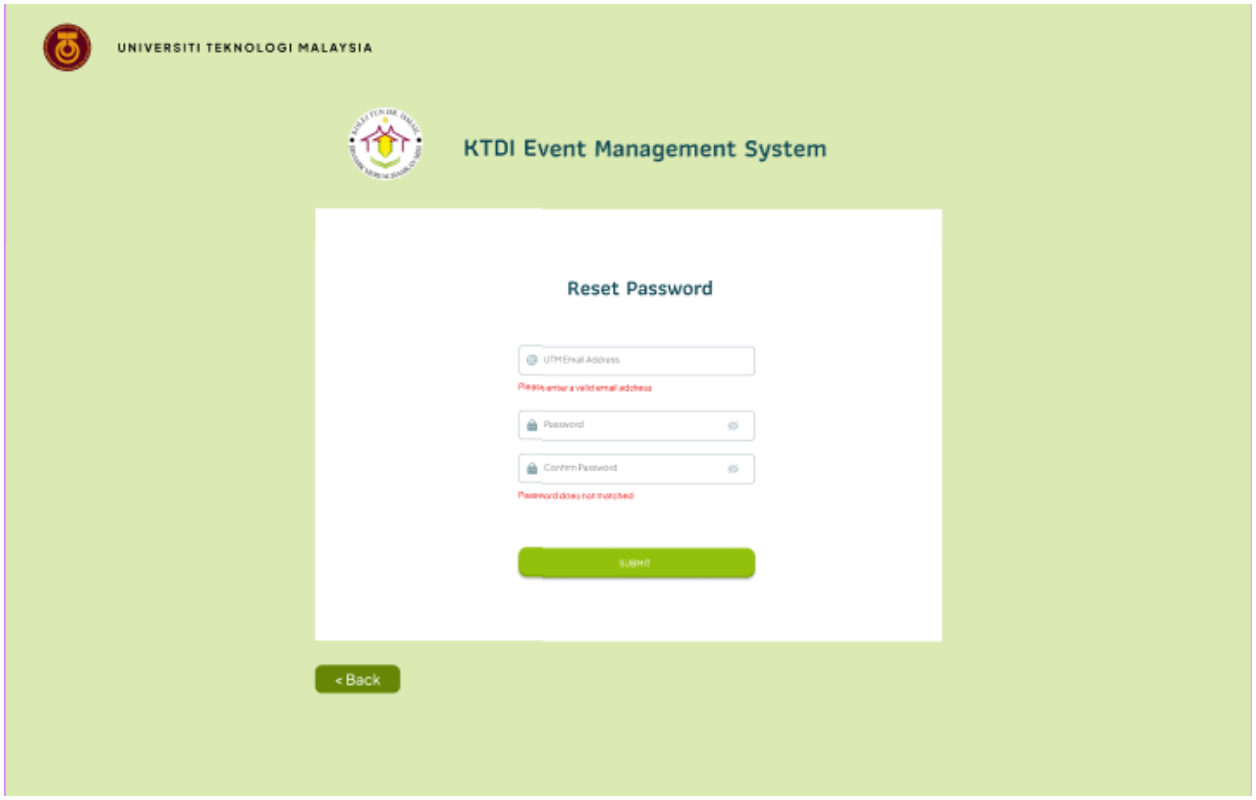

Figure 6.2.1.4: Interface for <Reset Password>

6.2.2 JKM KTDI

Figure 6.2.2.1: Interface for <Home Page>

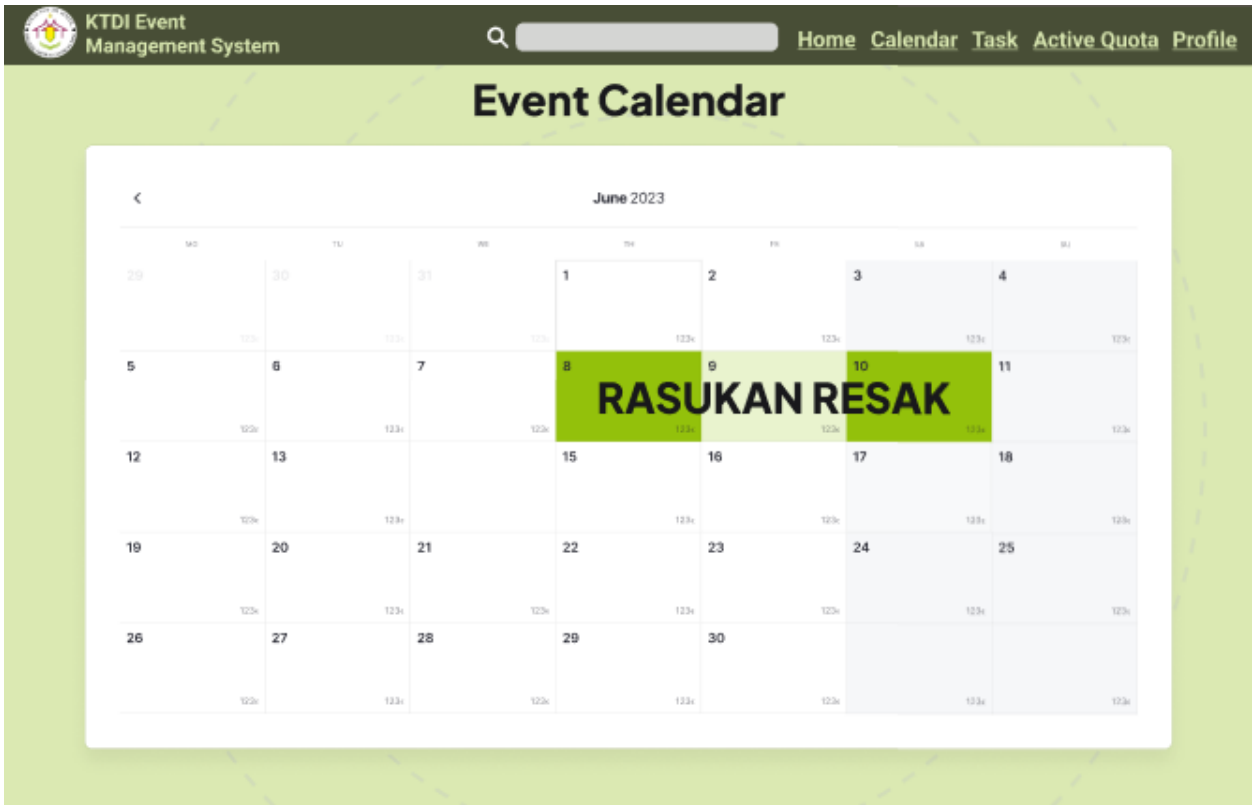

Figure 6.2.2.2: Interface for <Event Calendar>

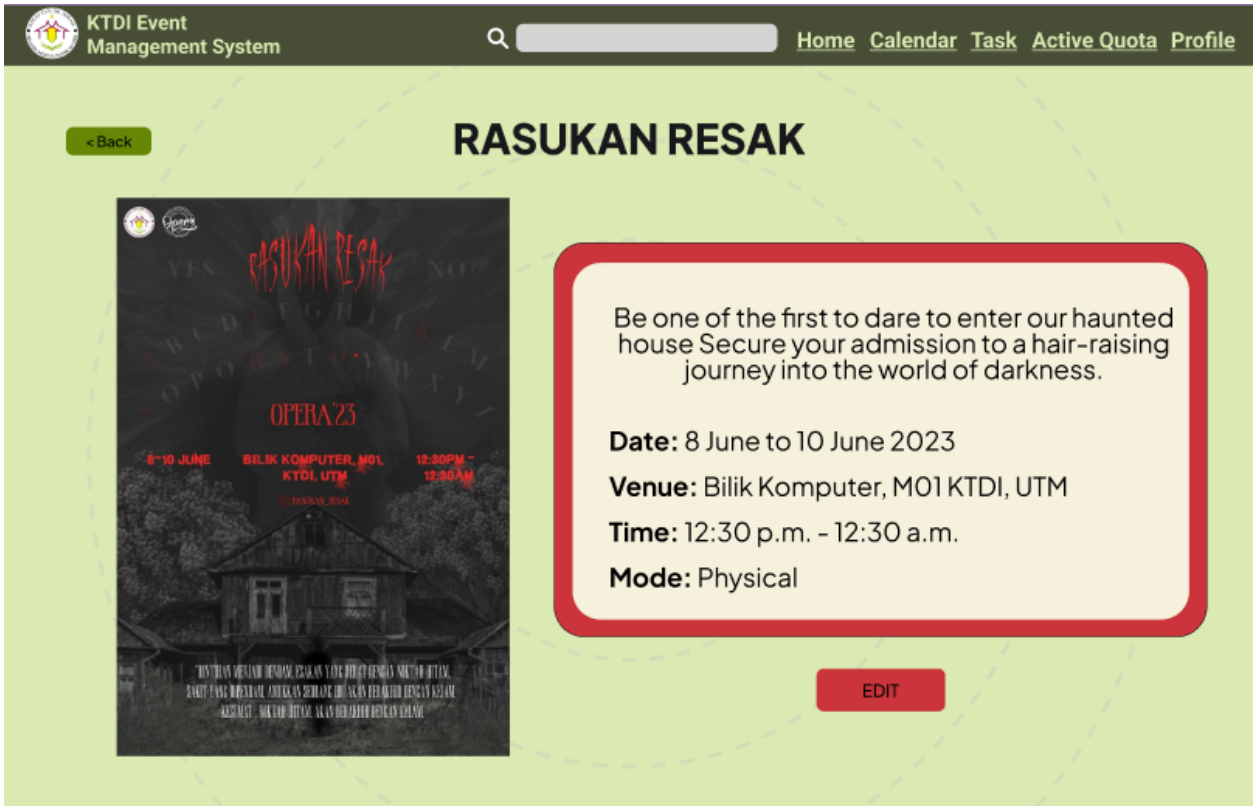

Figure 6.2.2.3: Interface for <Event Calendar Content>

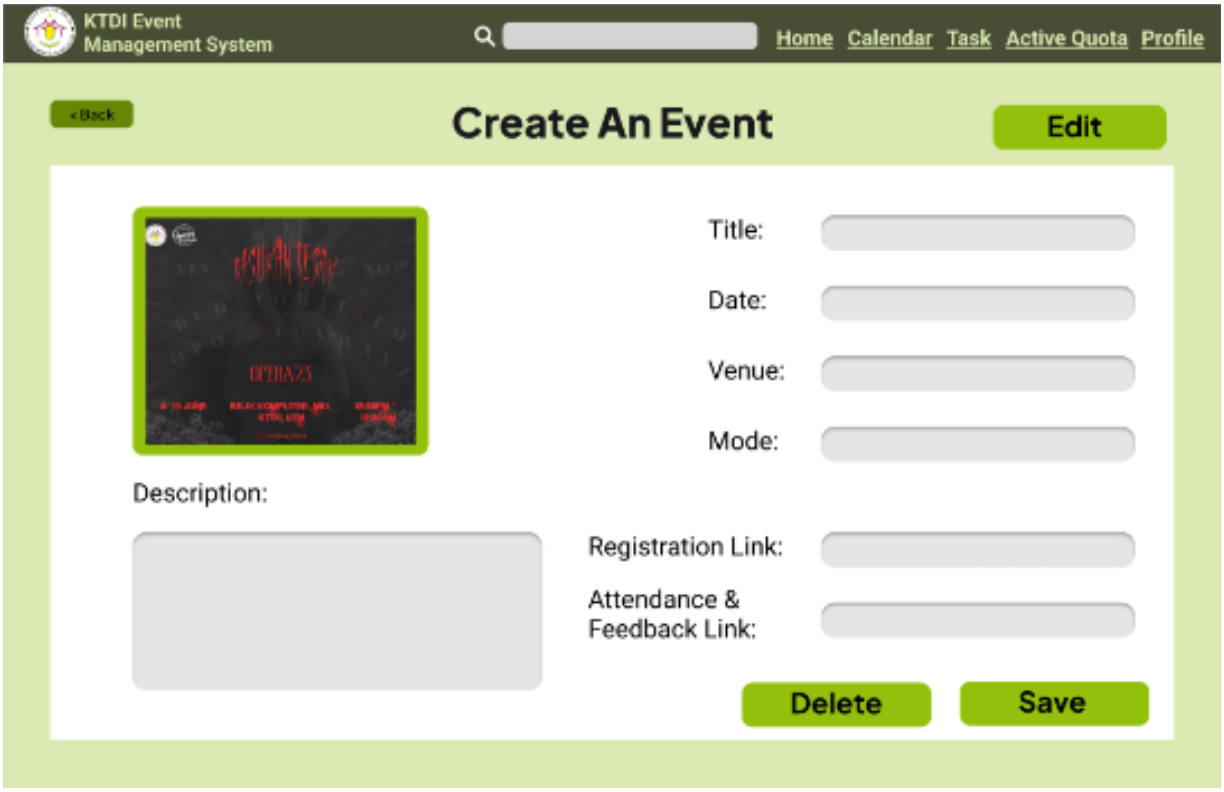

Figure 6.2.2.4: Interface for <Edit / Create Event>

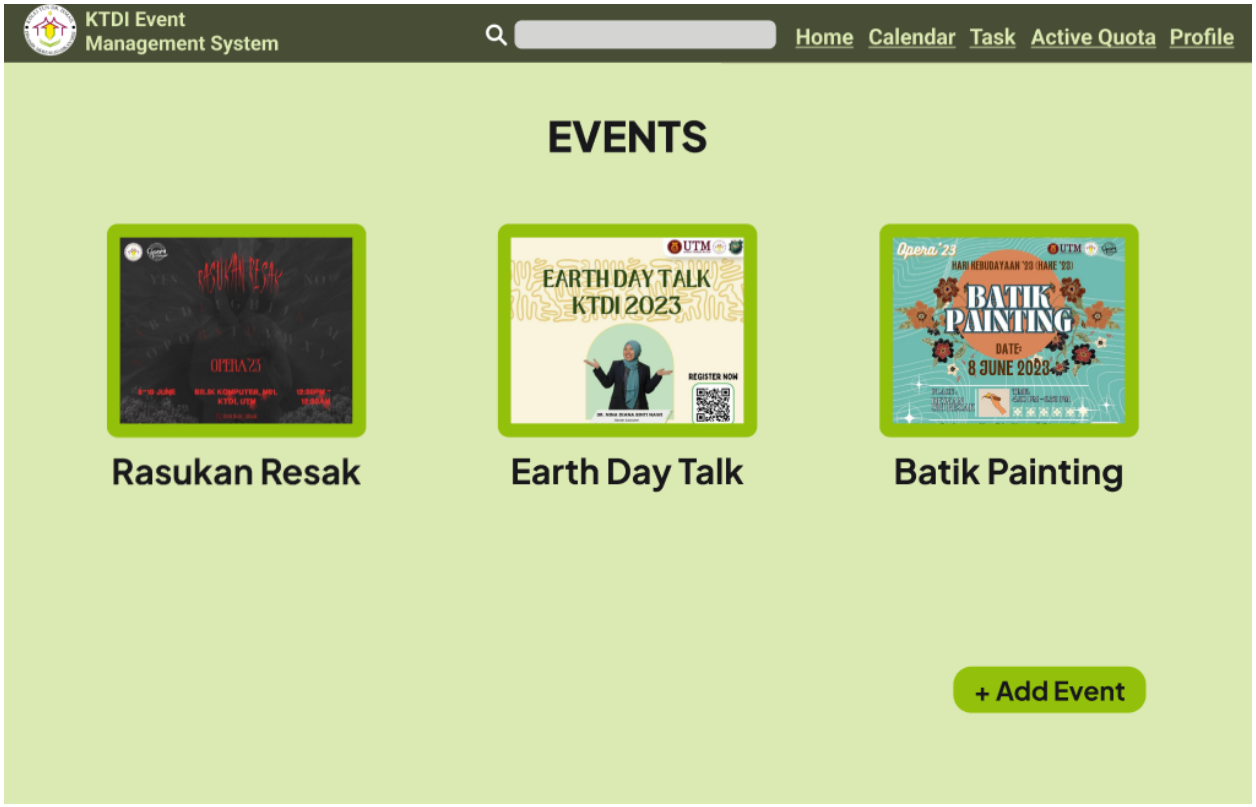

Figure 6.2.2.5: Interface for <Task Page>

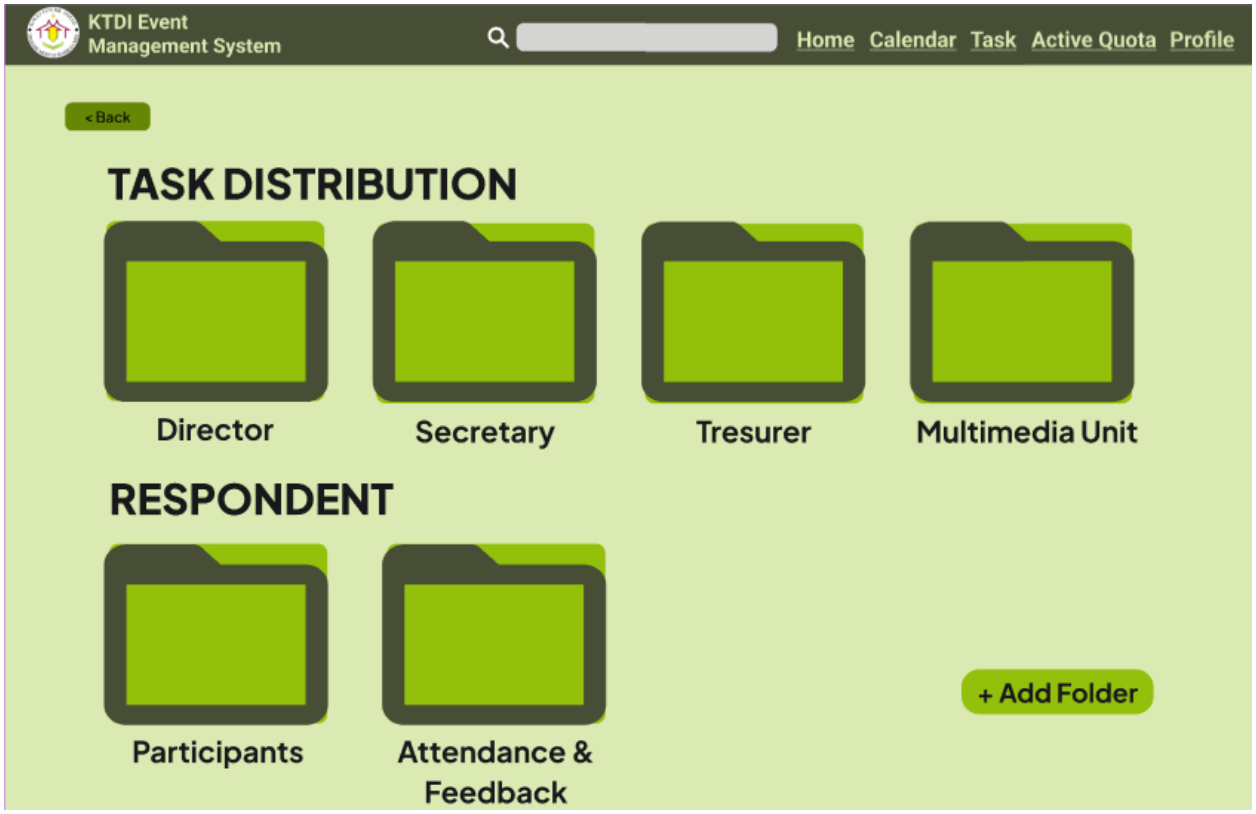

Figure 6.2.2.6: Interface for <Task Distribution>

Figure 6.2.2.7: Interface for <Task Tile>

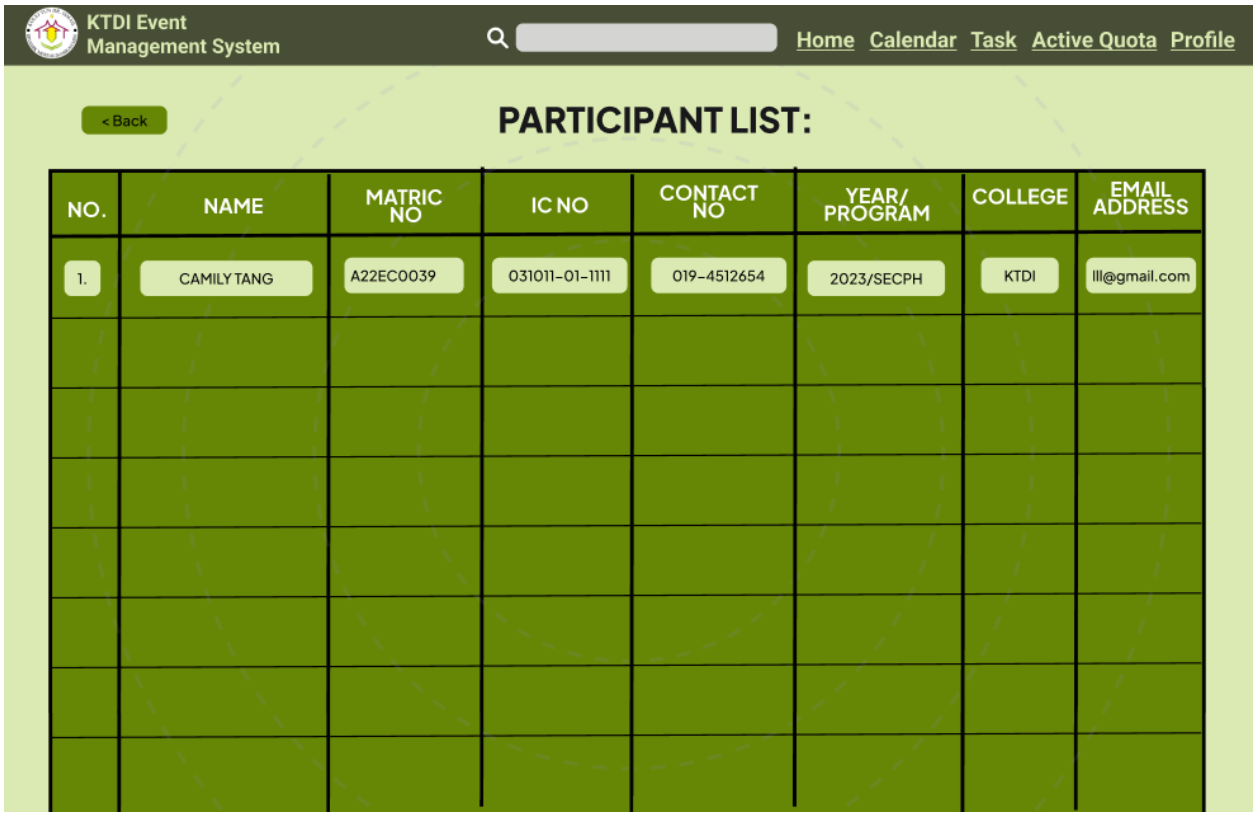

Figure 6.2.2.8: Interface for <Participant List>

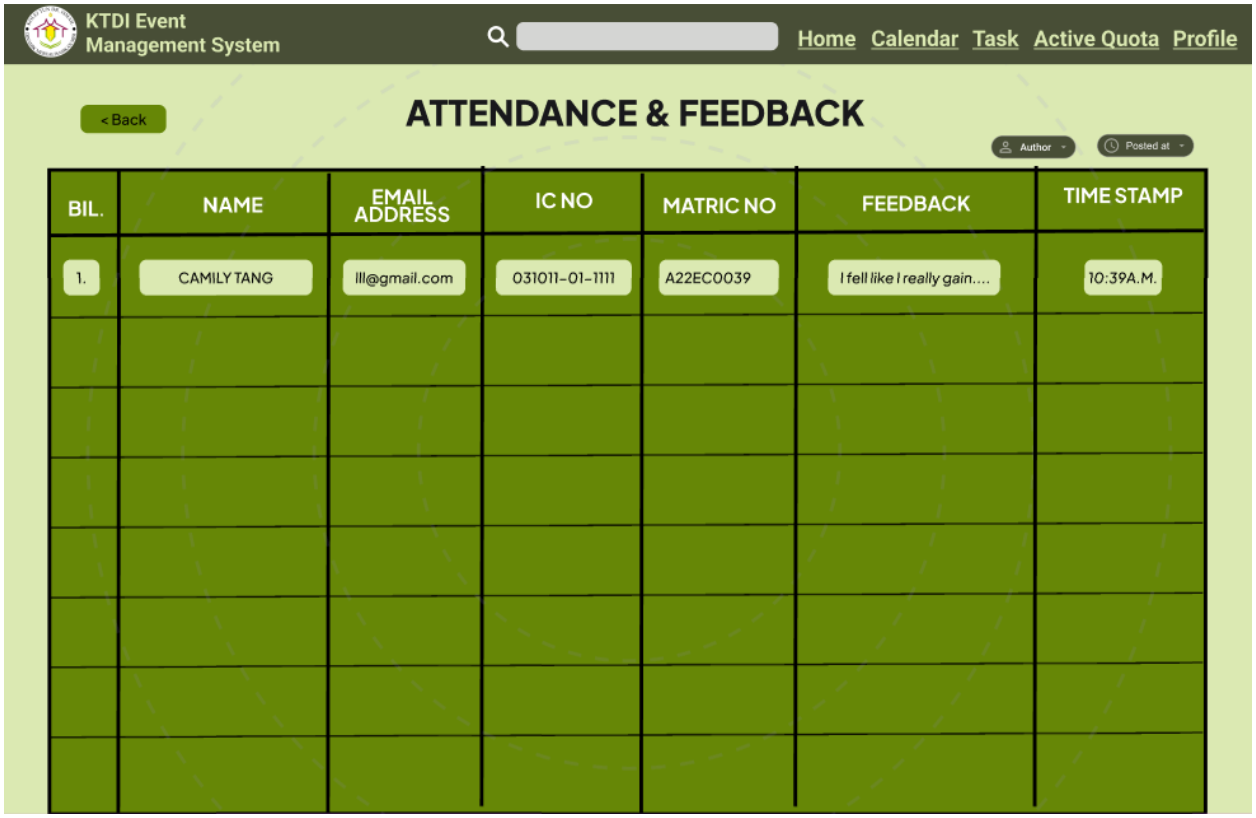

Figure 6.2.2.9: Interface for <Attendance And Feedback>

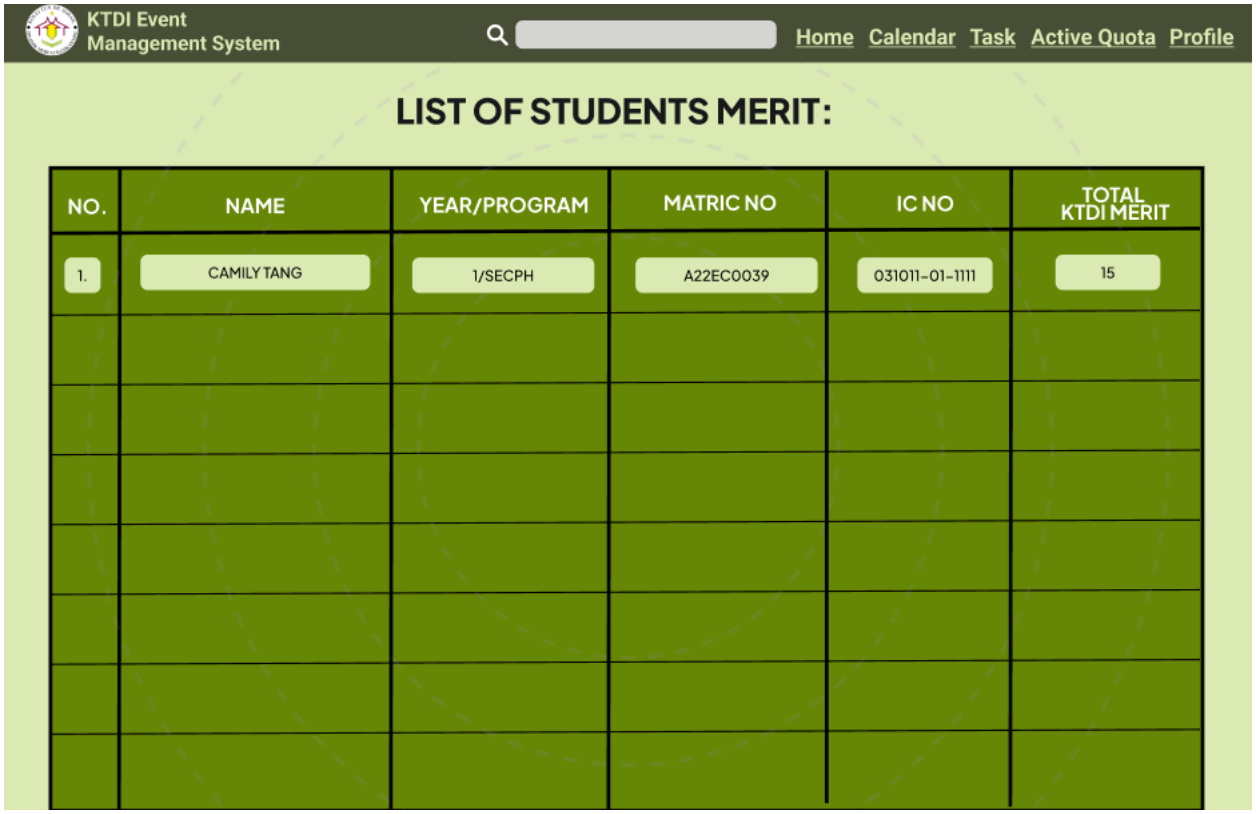

Figure 6.2.2.10: Interface for <Active Quota Page>

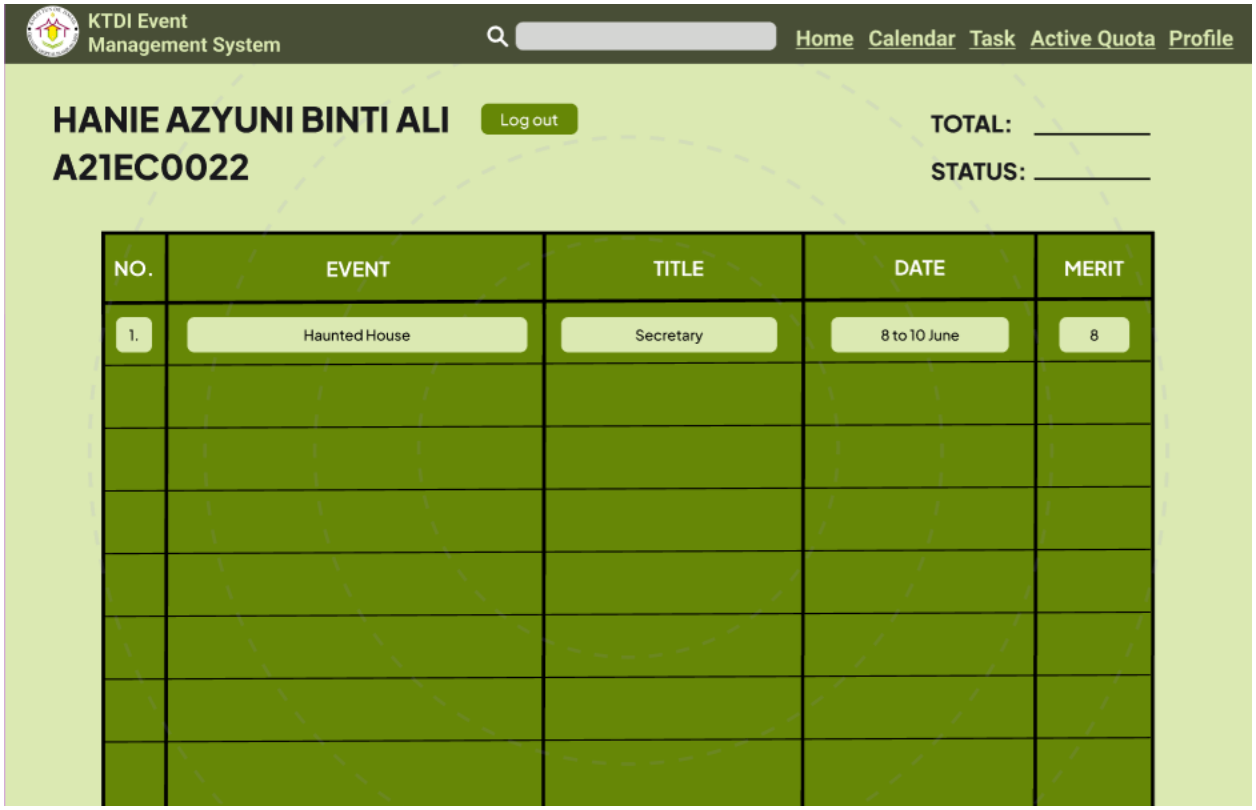

Figure 6.2.2.11: Interface for <Profile Page>

6.2.3 KTDI Staff Members

Figure 6.2.3.1: Interface for <Home Page>

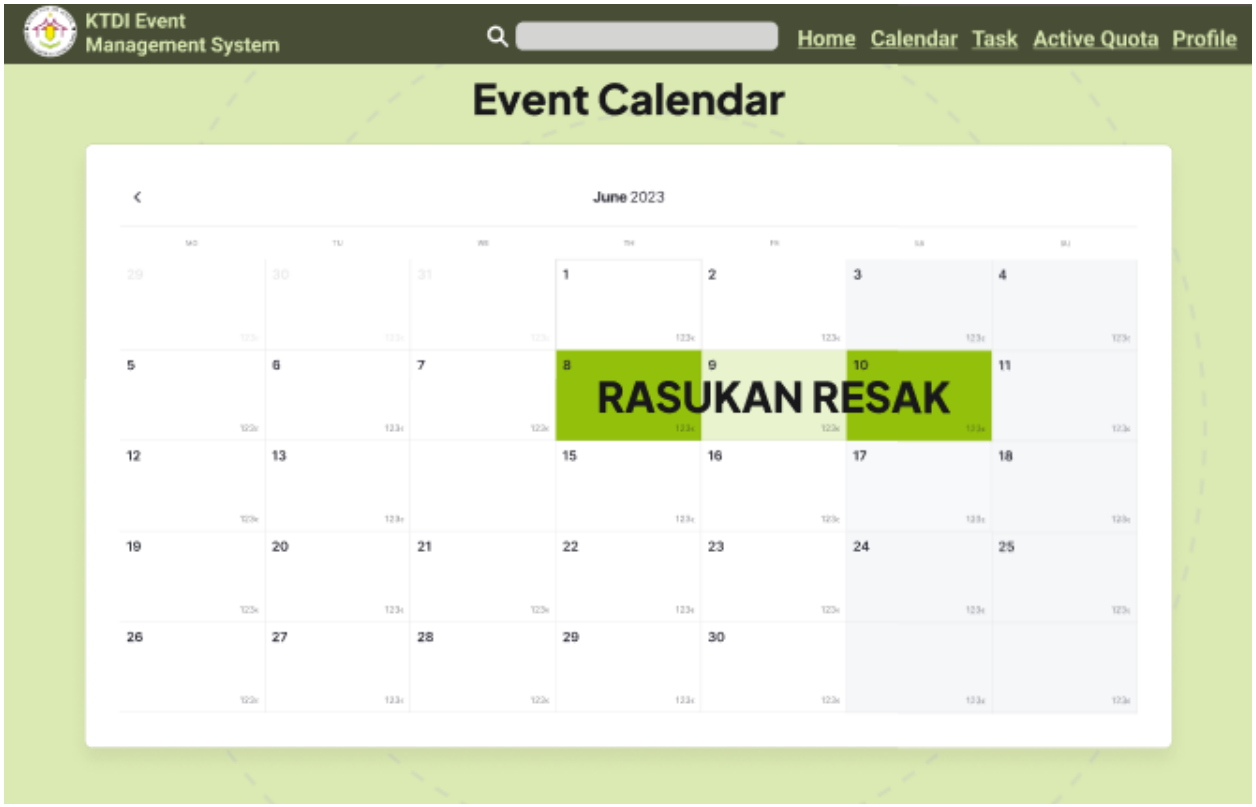

Figure 6.2.3.2: Interface for <Event Calendar>

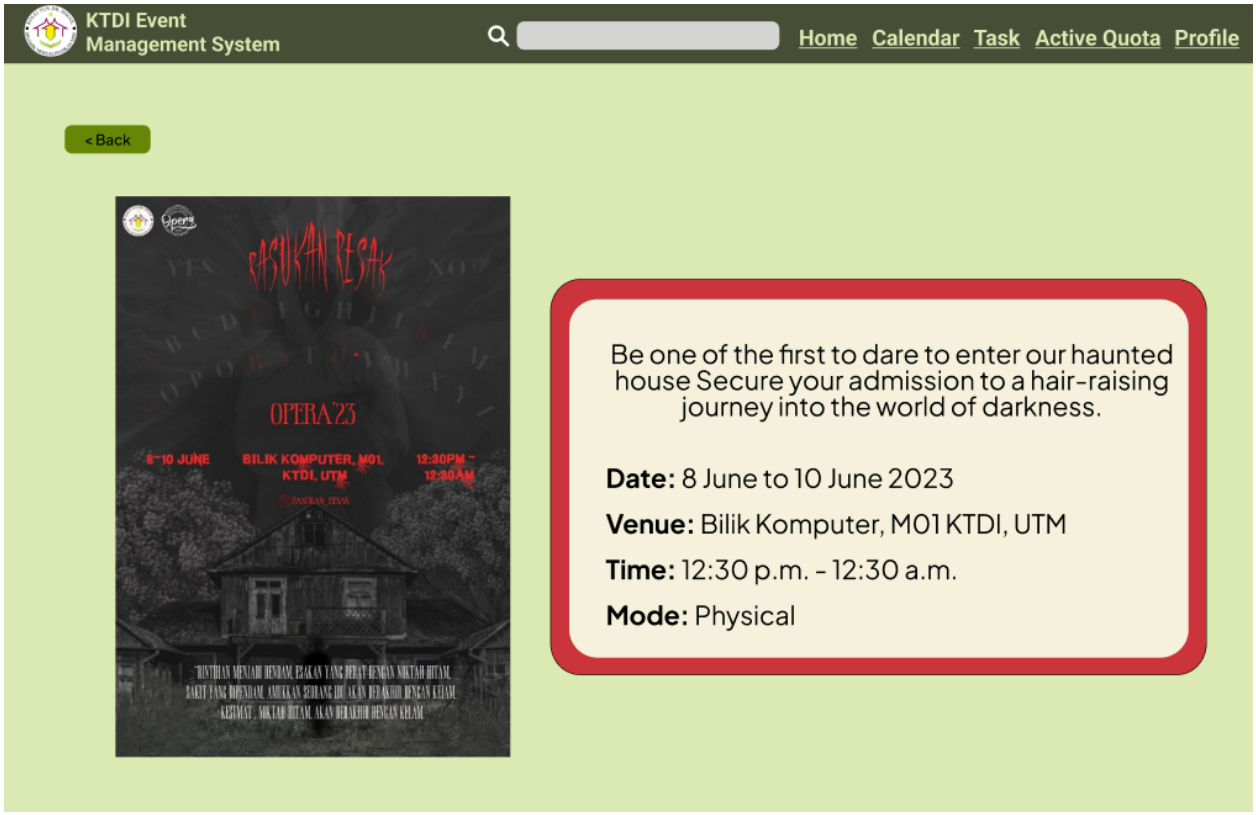

Figure 6.2.3.3: Interface for <Event Calendar Content>

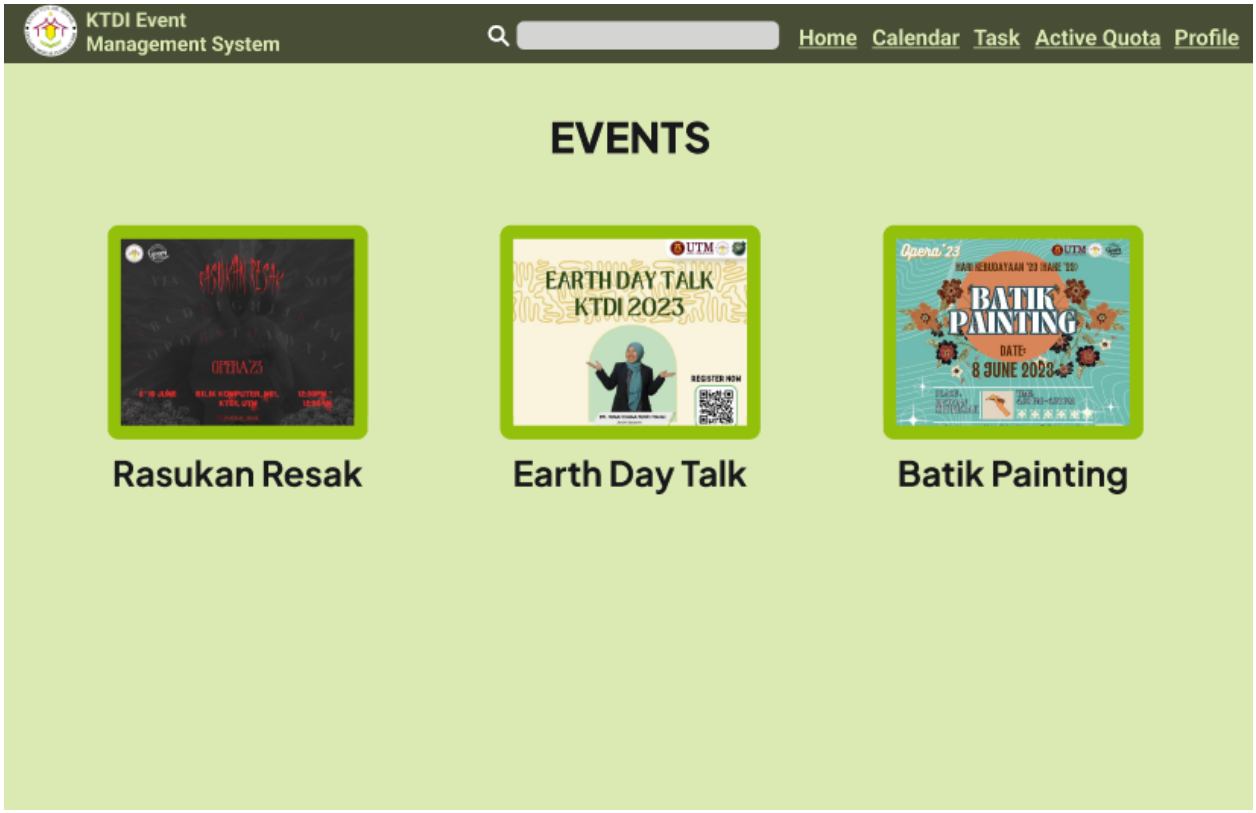

Figure 6.2.3.4: Interface for <Task Page>

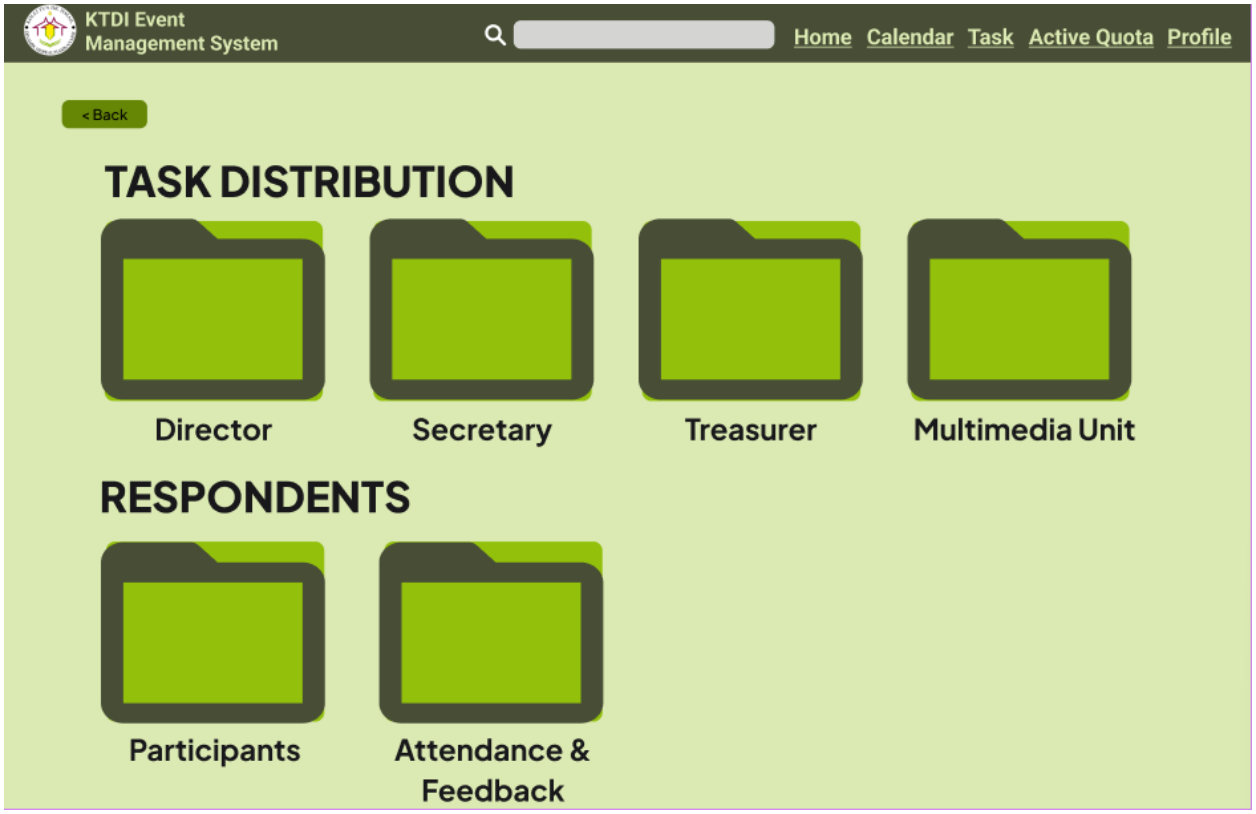

Figure 6.2.3.5: Interface for <Task Distribution>

Figure 6.2.3.6: Interface for <Task Tile>

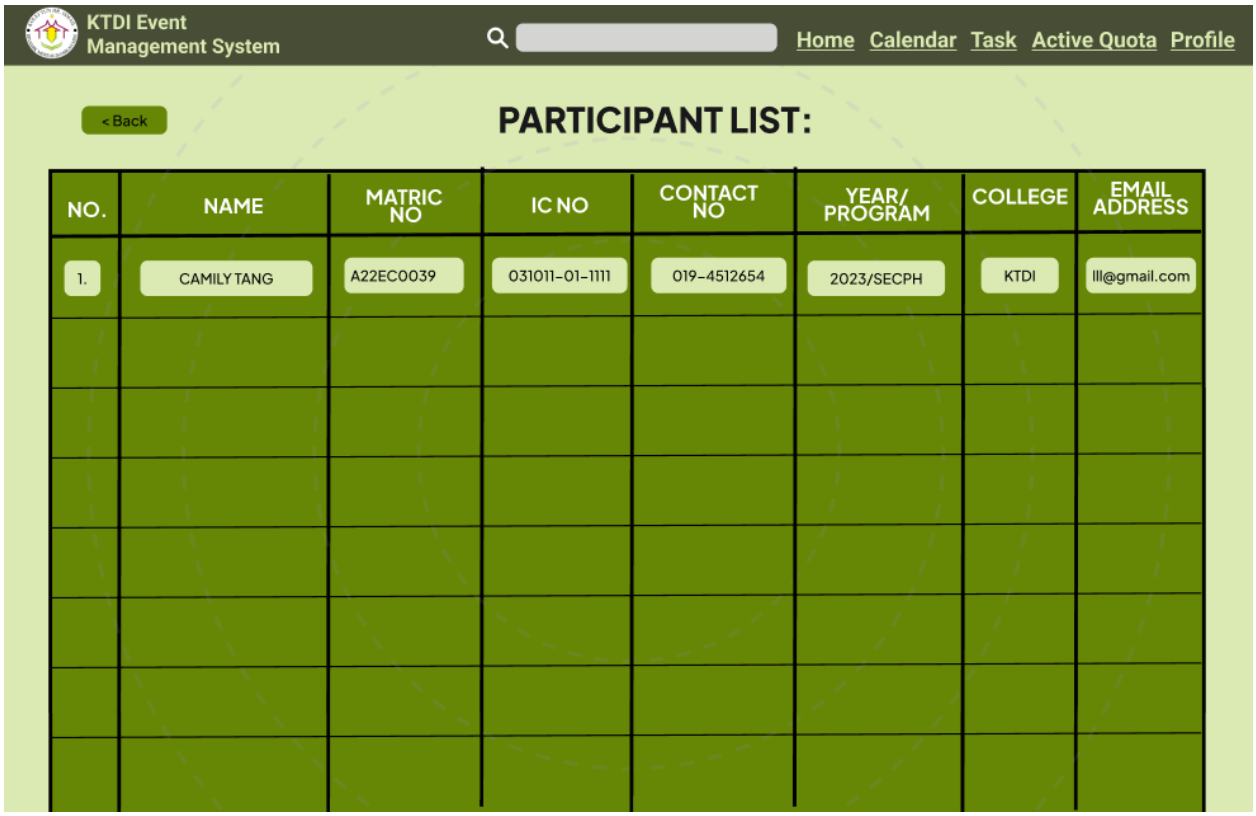

Figure 6.2.3.7: Interface for <Participant List>

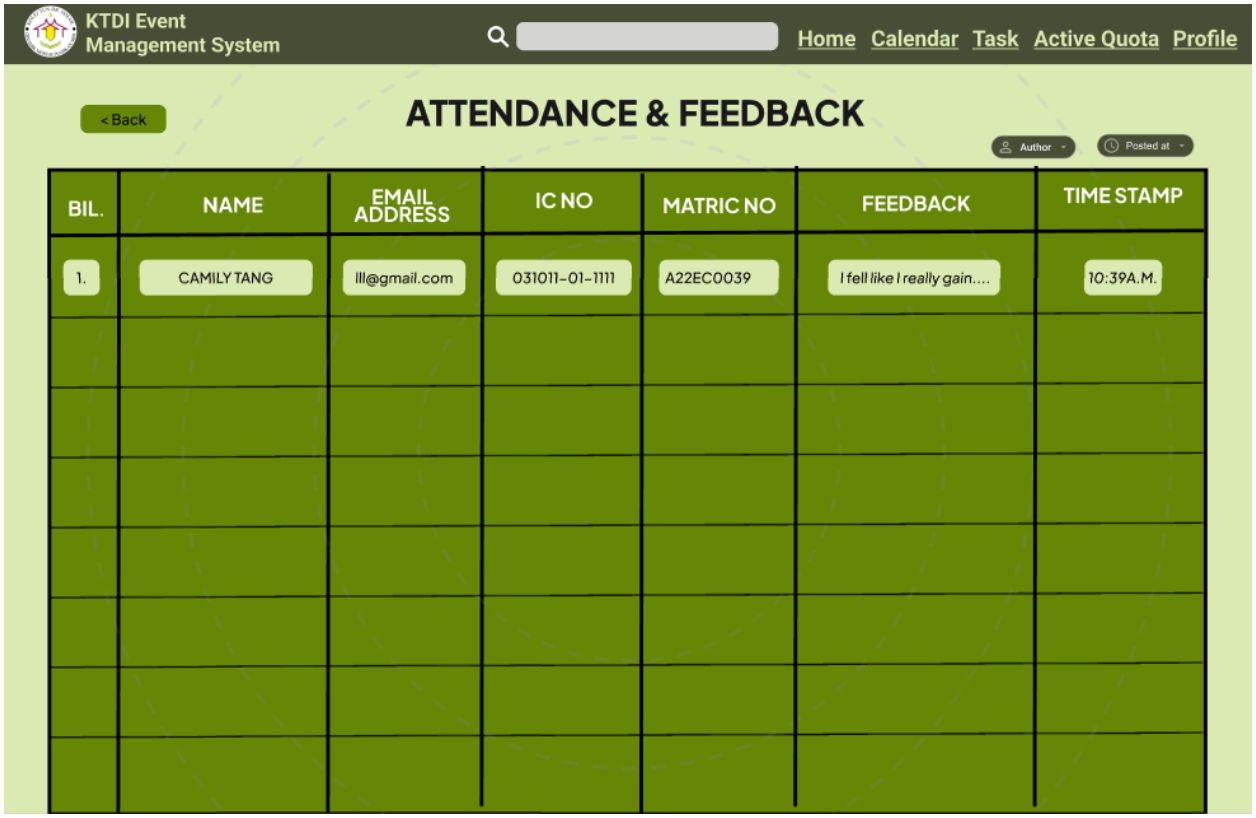

Figure 6.2.3.8: Interface for <Attendance And Feedback>

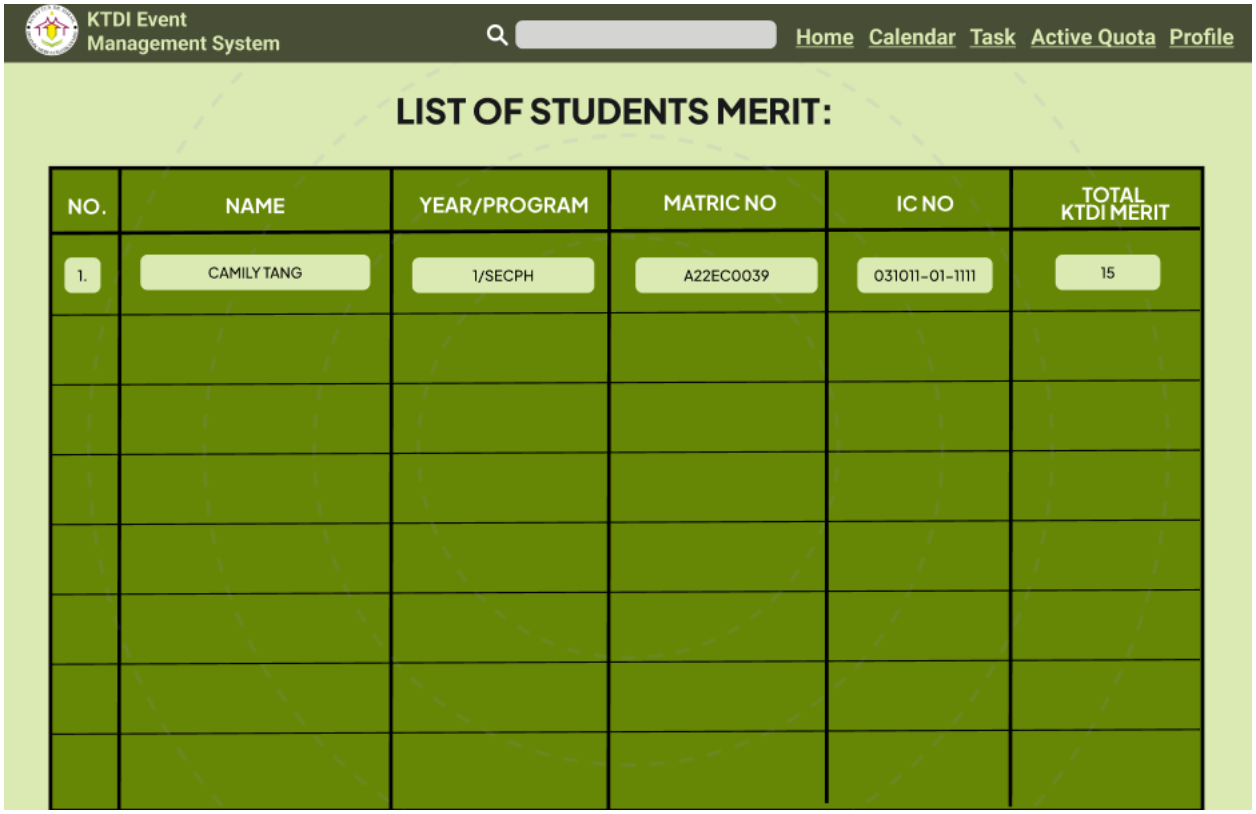

Figure 6.2.3.9: Interface for <Active Quota Page>

Figure 6.2.3.10: Interface for <Profile Page>

6.2.4 Students

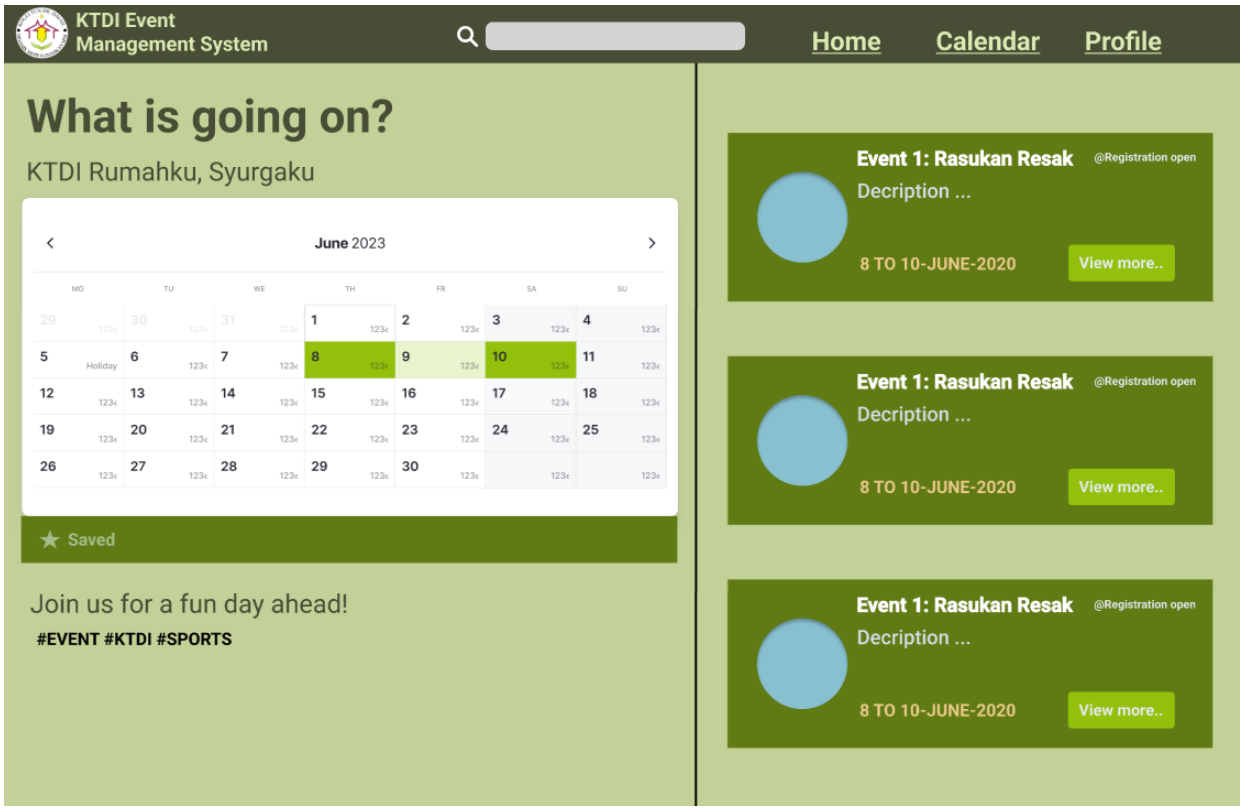

Figure 6.2.4.1: Interface for <Home Page>

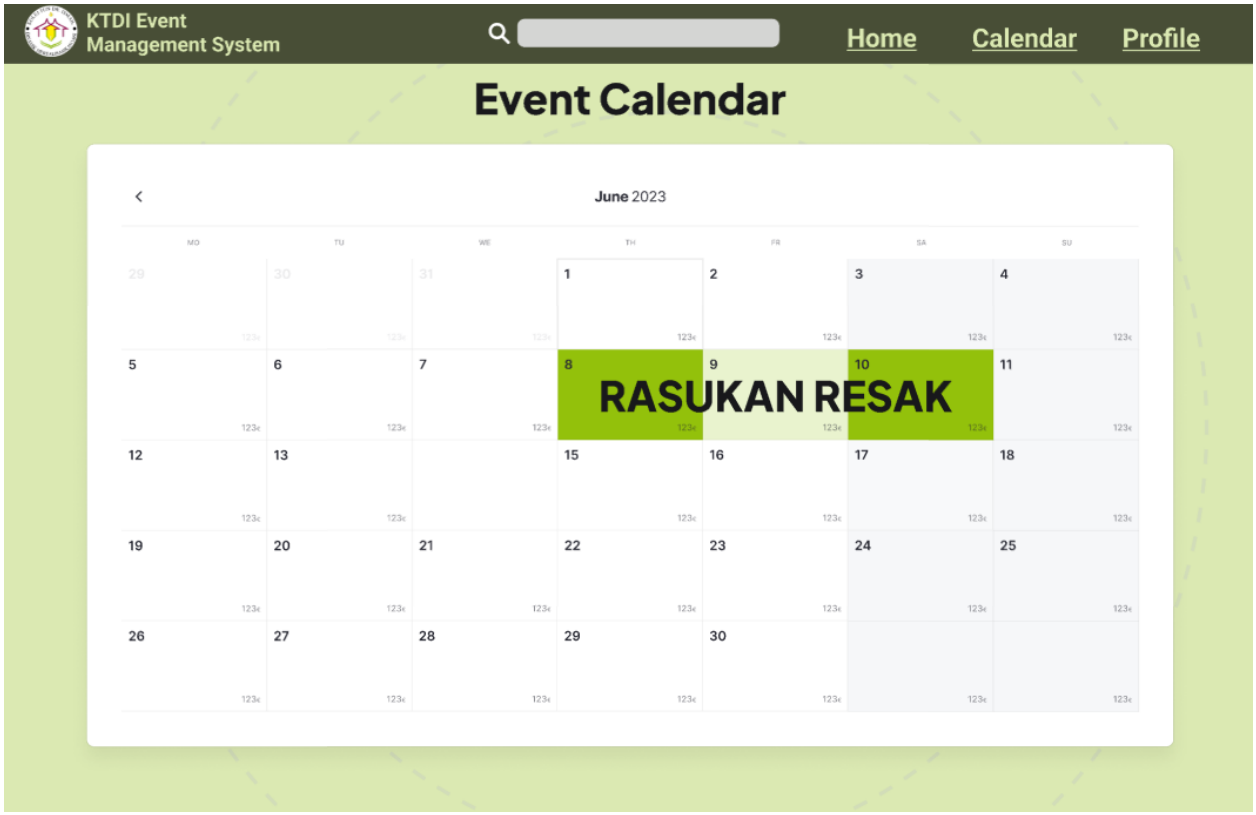

Figure 6.2.4.2: Interface for <Event Calendar>

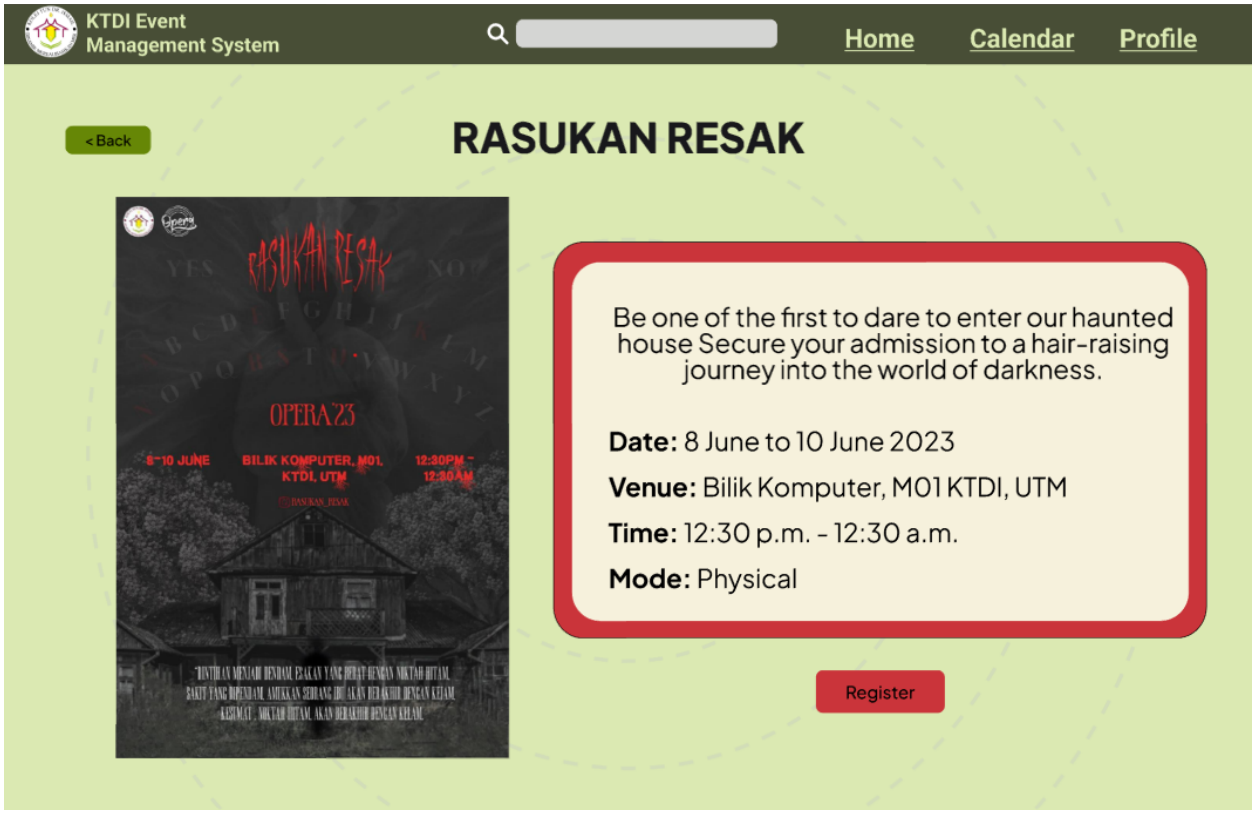

Figure 6.2.4.3: Interface for <Event Calendar Content>

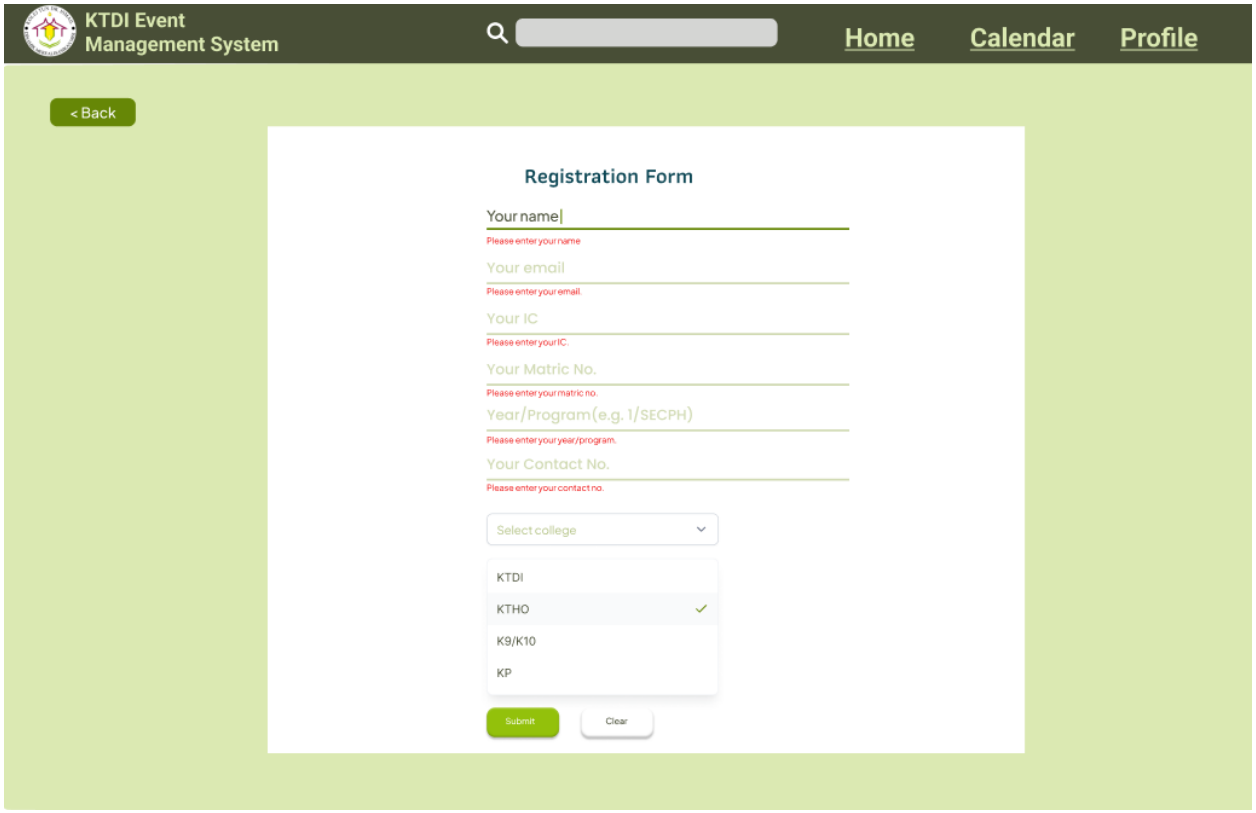

Figure 6.2.4.4: Interface for <Registration Form>

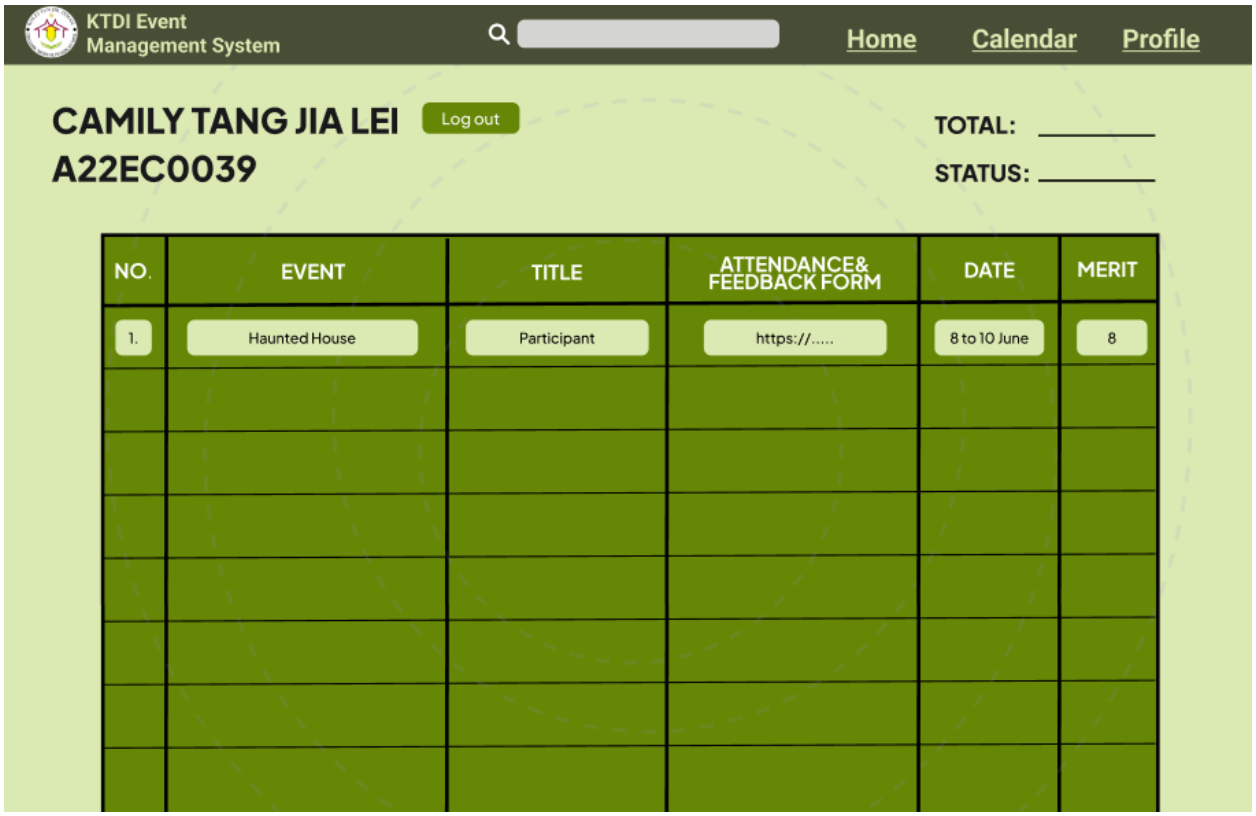

Figure 6.2.4.5: Interface for <Profile Page>

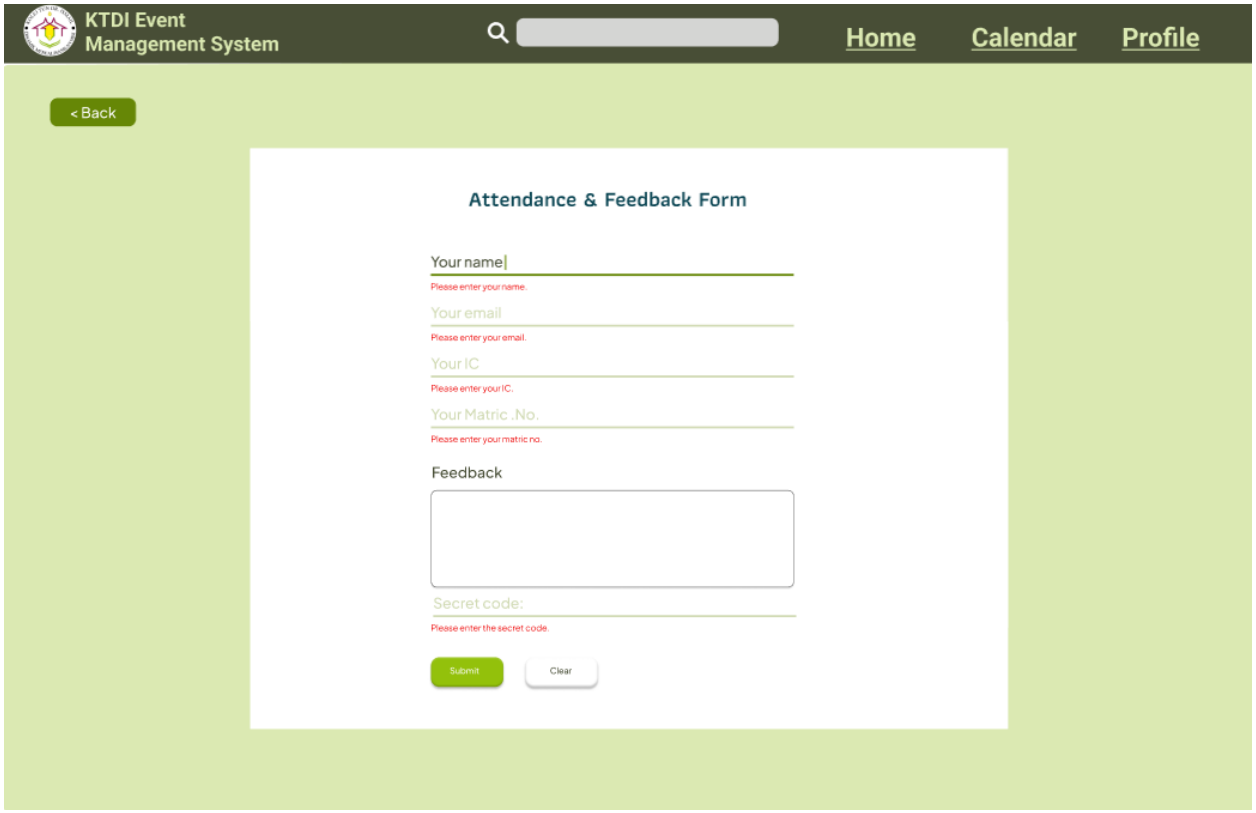

Figure 6.2.4.6: Interface for <Attendance and Feedback Form>