BMIT3173 Integrative Programming

ASSIGNMENT 202509

Student Name	:	Lim Jun Wei
Student ID	:	24WMR09078
Programme	:	RSD3
Tutorial Group	:	G5

System Title	:	TARUMT Event Management System
Modules	:	Event & Helper Module

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University of Management and technology's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

Plagiarism Statement Form

I, Name (Block Capitals) ____LIM JUN WEI_____Student ID____24WMR09078____ Programme __RSD3___Tutorial Group ____5_____confirm that the submitted work are all my own work and is in my own words.

I LIM JUN WEI acknowledge the use of AI generative technology.

Signature: _____

Date: _____19/12/2025_____

Table of Contents

1. Introduction to the System

TARUMT Event Management System is a web-based application developed for realizing the planning, organization and participation of events within Tunku Abdul Rahman University of Management and Technology (TARUMT). This system aims to **replace the manual and fragmented event management processes with a centralized, efficient and role-based digital platform**.

Via the application of **Laravel framework** in this system, it can support multiple user roles including administrators, organizers, members and event helpers while each user role will be assigned with different responsibilities and access privileges. By integrating the event creation feature, the approval workflows, helper recruitment, participant applications and membership management into a single system, the platform improves transparency, accountability and operational efficiency across the entire event lifecycle.

The system allows the organizers to **create and manage the events**, **open for helper recruitments** and **review the helper applications**. Meanwhile, the members can also **browse the approved events**, **apply for helper roles** and **track their application status**. On the other hand, the administrators will **oversee the overall system operations**, including event approvals, organizer verification, user management, and enforcement of penalties or membership rules. This structural role-based approach can make sure that all stakeholders interact with the system in a secure and controlled manner.

Moreover, the TARUMT Event Management System also integrates the features such as **event status tracking**, **helper role availability control**, **email notifications**, **dashboard summary cards** and **membership credit management**. These features can effectively help the users to make informed decisions, receive timely updates and maintain the engagement during the event process. The system also supports filtering, searching and sorting functions of events to enhance usability and user experience.

In conclusion, the TARUMT Event Management System acts as a comprehensive solution for managing the university-level events, enhancing the coordination between organizers and helpers and ensuring the consistent administrative oversight. Via leveraging the modern web technologies and modular system architecture, the platform can be built with high scalability, maintainability and adaptability for future enhancements.

2. Module Description

The Event Module is the core component of the TARUMT Event Management System. It is responsible for managing the complete lifecycle of the events within the platform. It allows the organizers to create and manage events while enabling administrators to review and approve event submissions. Not only that, the members are also able to view the approved events and apply for helper roles. The module will enforce the role-based access and event status control to ensure that each user can only perform the actions which are suitable to their role. By implementing this structured workflow, the Event Module can ensure that all the events will be properly reviewed, published and managed. Eventually, it provides a consistent and organized event management process across the system.

Admin Event Management

Create Events

The Create Events function allows administrators to **register a new event** within the system by filling in the essential info such as the event name, description, category and assigned organizer. They may also **check the deposit requirement box** to automatically create a deposit payment record for that event upon event creation. This event creation function also **supports the upload of proposal documents and event images** for providing additional event details. In order to ensure a valid event scheduling, the system **enforces the date constraints** where the event start date must be at least one month after the current date and the event end date must be later than event start date.

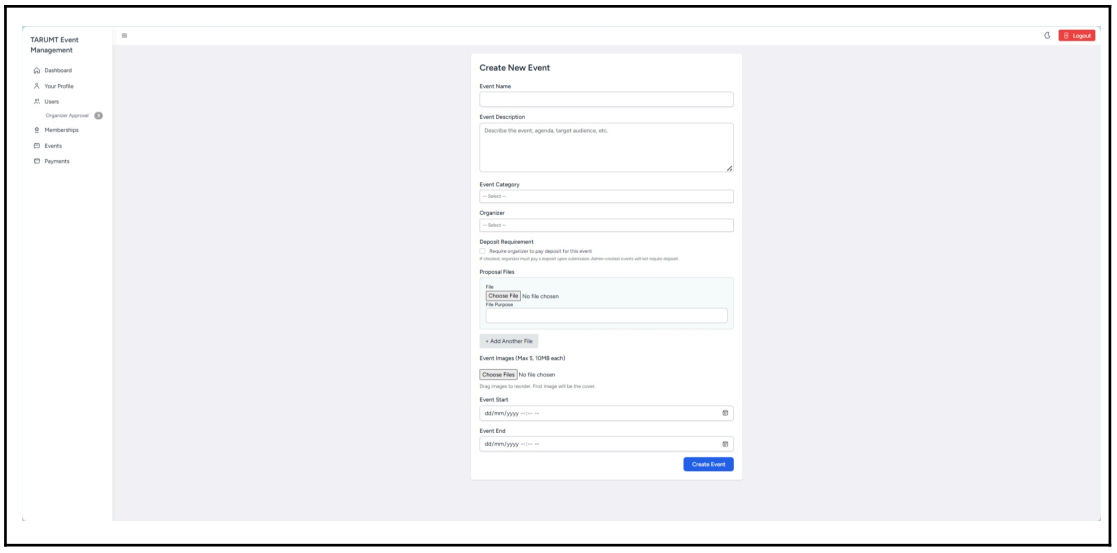

Figure 2.1.1: Admin Site Create Event Form

Class Path: *App\Livewire\Admin\CreateEvent*

View Event Listing

The View Event Information function allows administrators to **view and manage all events** in the system through a centralized table view. The table provides the key event information including event name, organizer name, event start date, event end

date and current event status. In order to improve the usability of the event listing, the table also supports the **sorting for each column** which enables the administrators to arrange events in ascending or descending order. The **filtering options are provided** for narrowing down the events based on their status such as pending, approved, ongoing, cancelled, rejected or completed. Not only that, the **event date range filtering option** is also offered within this function. A **search bar** is available to quickly locate the events by their name when supporting pagination for large amounts of event records.

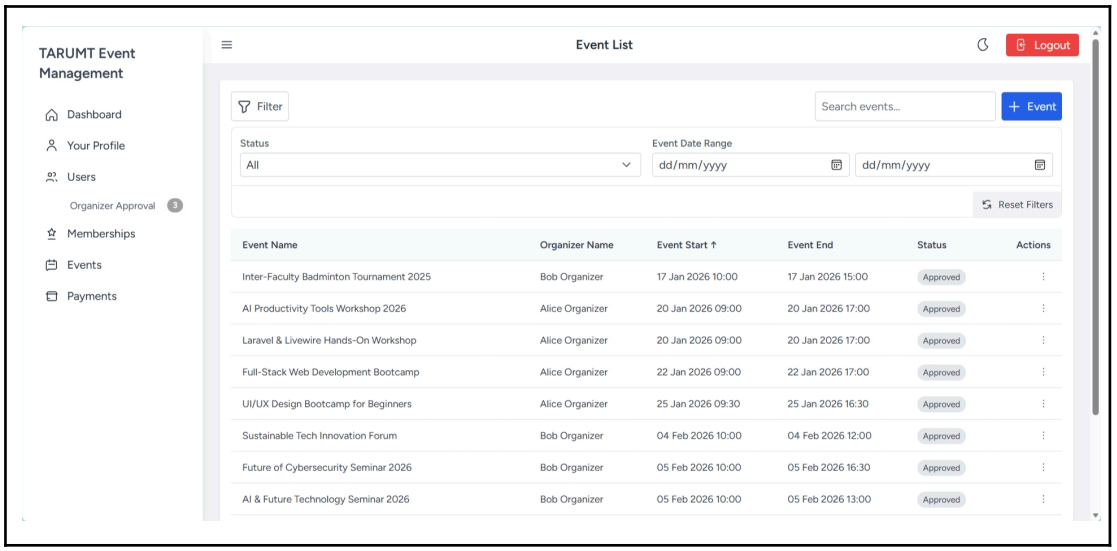

Figure 2.1.2: Admin Site View Event Listing

Class Path: *App\Livewire\Admin\EventTable*

View Event Details

The View Event Details function allows the administrator to access the complete information for a selected event. It **shows the essential event details** such as category, status, start and end dates and description along with the proposal documents and event images that can be previewed directly. Meanwhile, the event organizer's company name, type and contact details will also be shown as well. If the event requires a **deposit**, the related **payment information** such as payment status, amount, method and payment data will be displayed. Besides that, this page also provides an **overview of helper recruitment details** to allow admins to view the available helper roles, number of vacancies and total applicants.

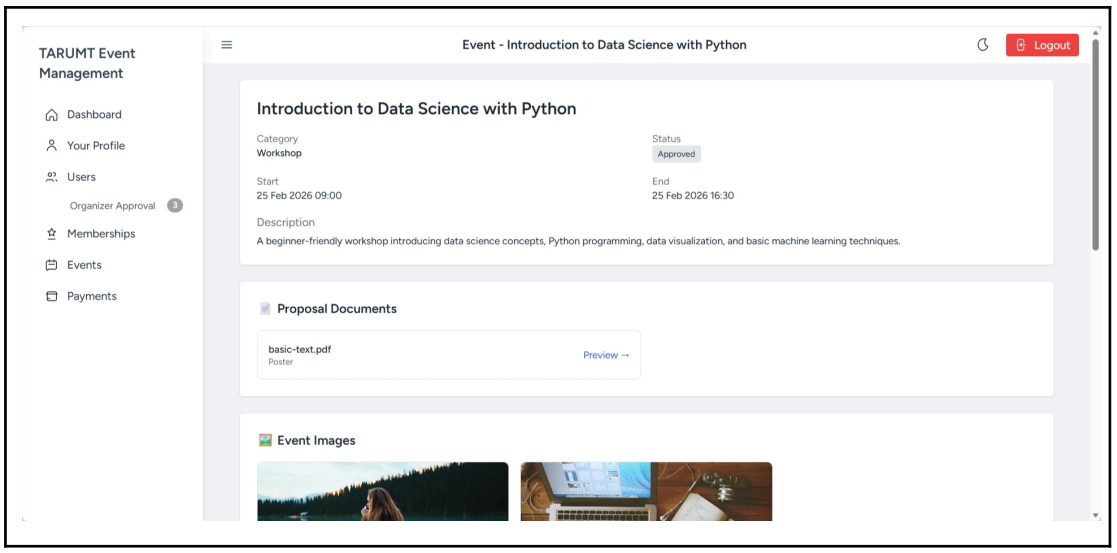

Figure 2.1.3: Admin Site View Event Details - Section 1

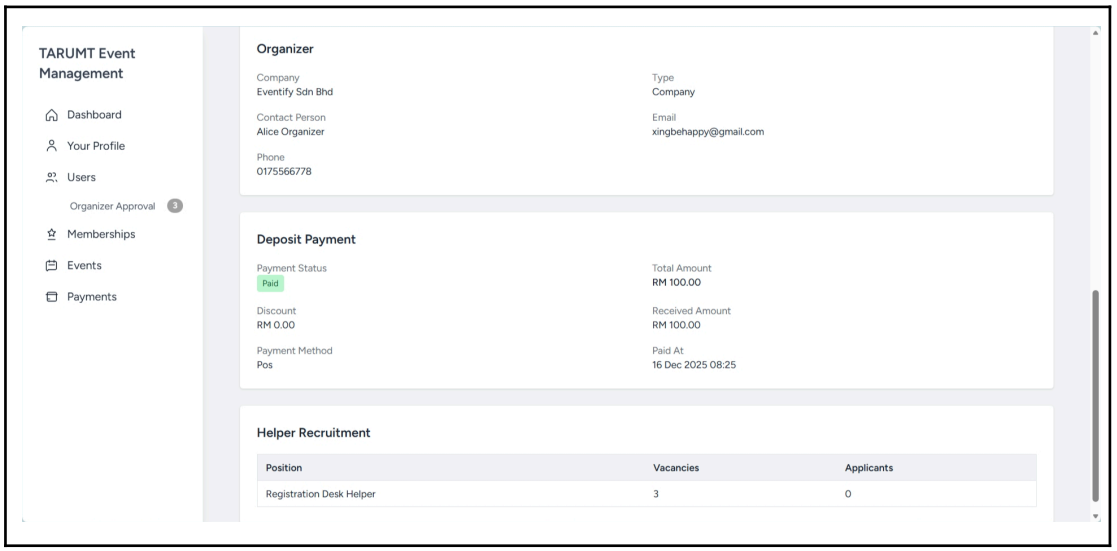

Figure 2.1.4: Admin Site View Event Details - Section 2

Class Path: *App\Http\Controllers\EventController@show*

Edit Existing Events

The Edit Existing Event function allows the administrators to **update the selected event information** after an event has been created as long as it is in pending or approved status. They are permitted to **modify the key event details such as event name, event description, event category, proposal documents, event images and event start and end dates**. There are certain fields such as **assigned organizer and deposit requirement that are displayed as read-only** and cannot be modified once the event is created. This restriction helps for preserving the data integrity, particularly for organizer accountability and payment-related records. By providing the controlled editing capabilities, this function allows administrators to correct or update the event info while ensuring consistency across the related system modules.

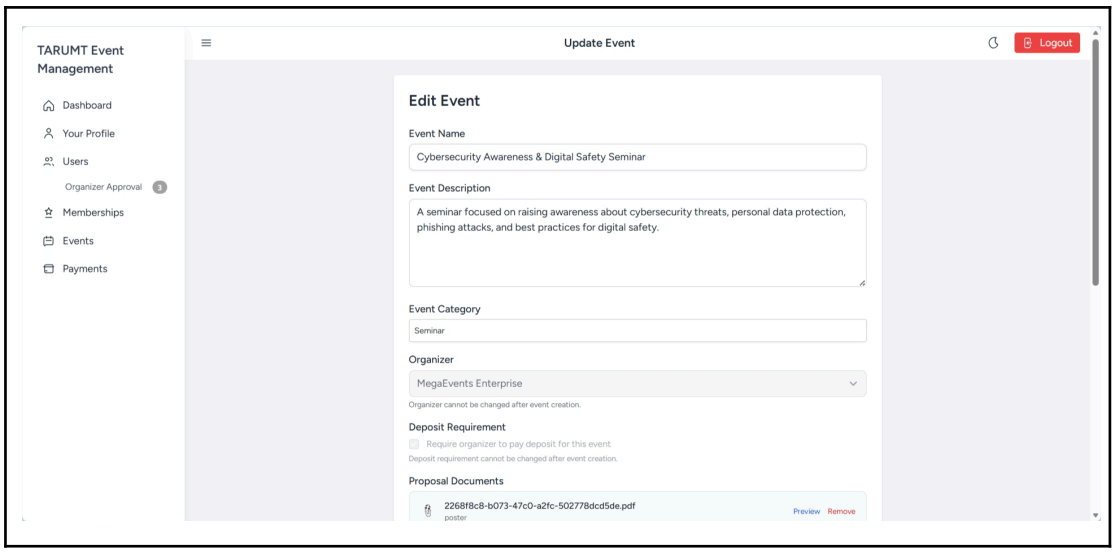

Figure 2.1.5: Admin Site Edit Existing Events - Section 1

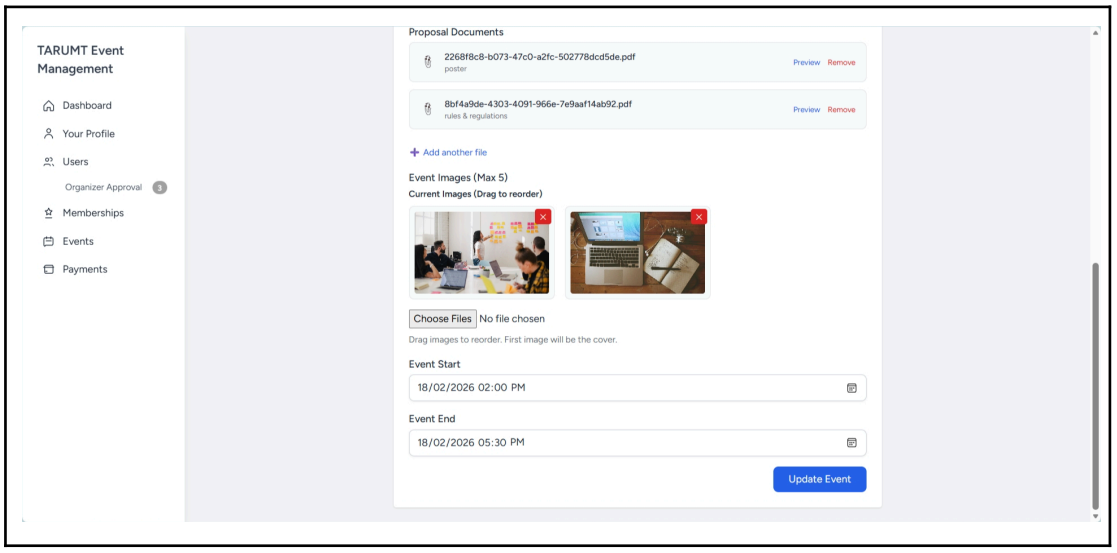

Figure 2.1.6: Admin Site Edit Existing Events - Section 2

Class Path: *App\Livewire\Admin\EditEvent*

Approve or Reject Events

In the event listing table action column, administrators are given with management options such as **approving or rejecting event submissions** and **cancelling events** based on the current event status. Once the event submission has been approved, this event will be accessible by member and public site.

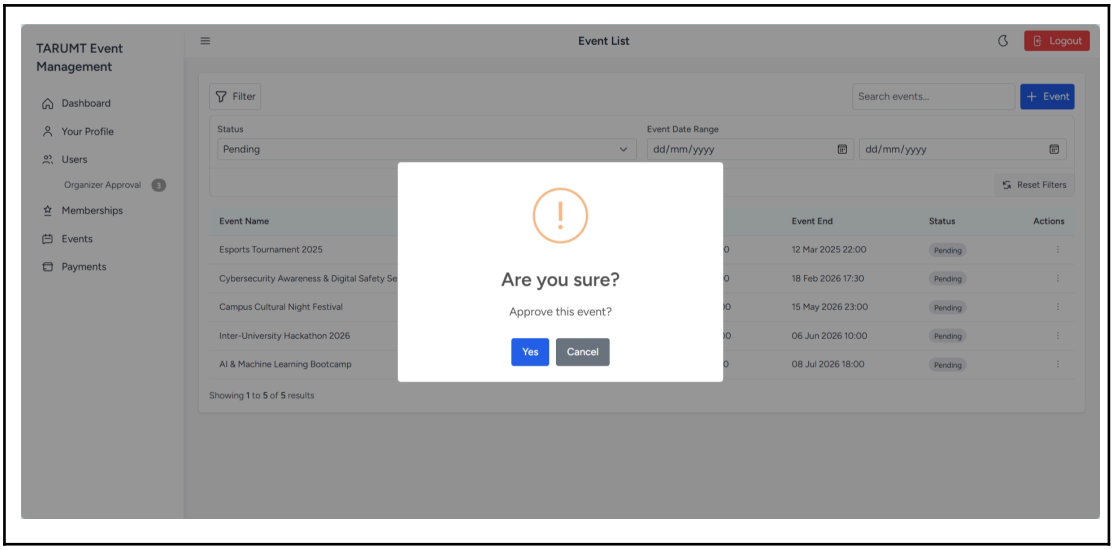

Figure 2.1.7: Admin Site Approve Pending Event Confirmation Dialog

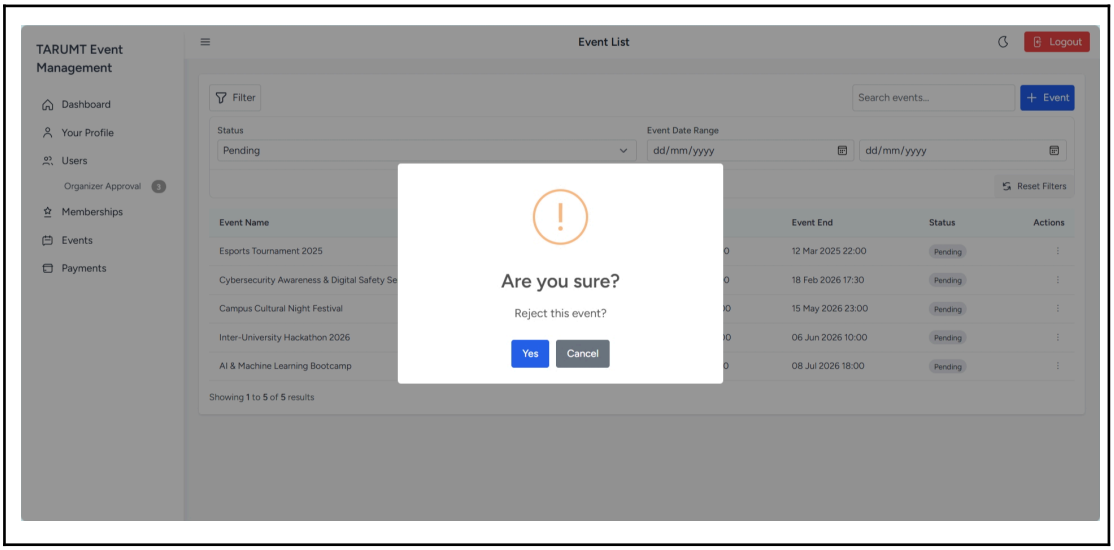

Figure 2.1.8: Admin Site Reject Pending Event Confirmation Dialog

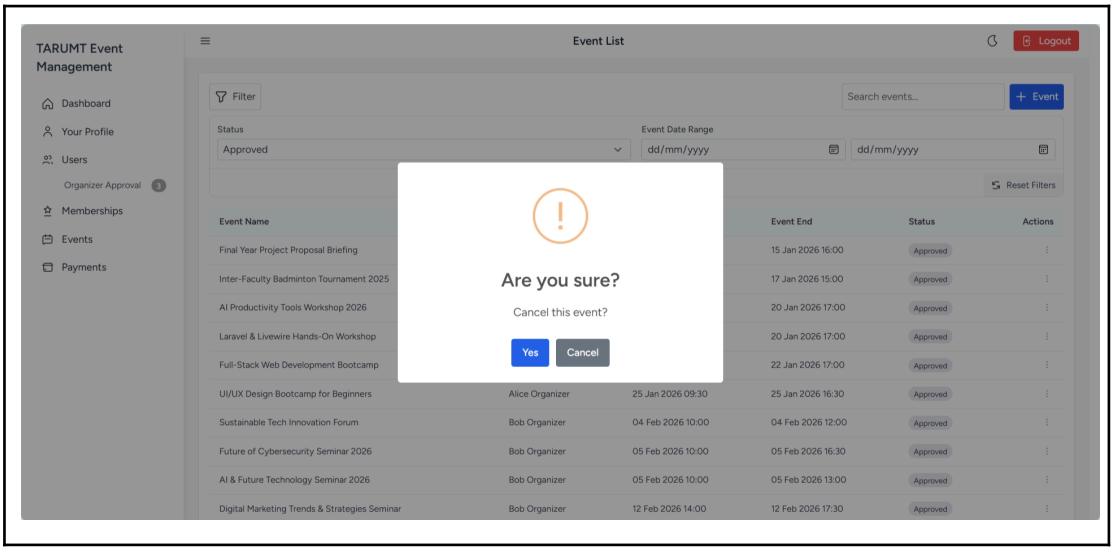

Figure 2.1.9: Admin Site Cancel Approved Event Confirmation Dialog

Class Path: *App\Livewire\Admin\EventTable*

Organizer Event Management

Create Event

The Create Event function allows the organizers to **submit a new event proposal** in the system organizer site. The organizers will provide the event information such as **event name**, **event description, event category and so on**. This function also supports **uploading proposal documents and event images** with limits applied for making sure that each event's upload documents are always within the controllable sizes. In context of time schedule, the system will enforce the **date validation rules** where the event start date must be at least one month after the current date and the event end date must be later than the event start date. Once the event has been created with pending status, the admin site will be able to update the event listing and allow administrators to review the event proposal submission.

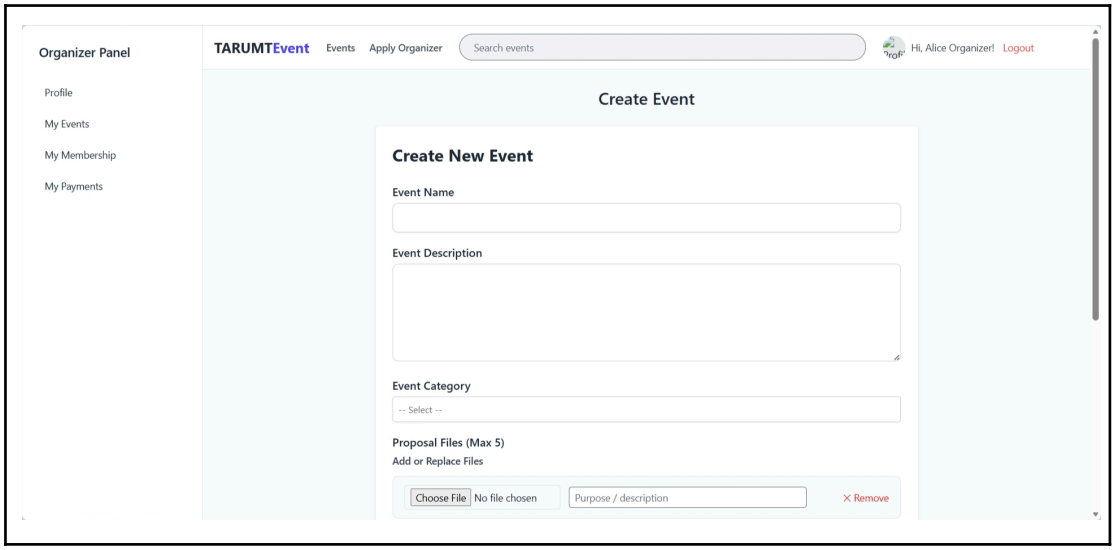

Figure 2.2.1: Organizer Site Create Event Form - Section 1

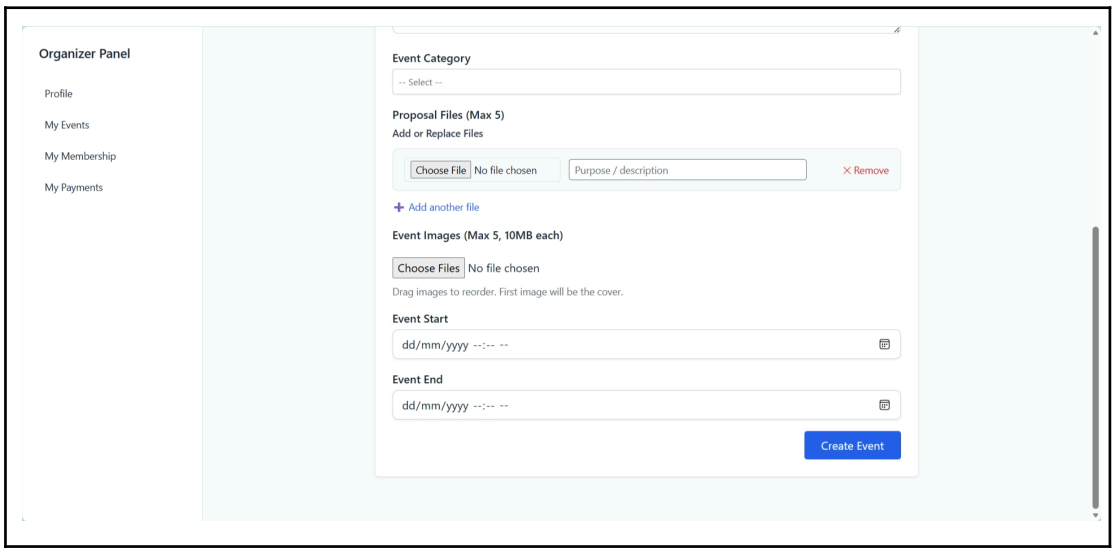

Figure 2.2.2: Organizer Site Create Event Form - Section 2

Class Path: *App\Livewire\Organizer\EventForm*

View Event Listing

The View Event Listing function allows the organizers to **view and manage their events**. The events are displayed in a card-based layout while each card will display the key event info such as event name, start date and current status. For quickly locating the specific events, this page has provided a **search function for event names** and **sorting options based on event name, start date and status**. The filtering features are also available for narrowing down the events by **event date range, category and status**. For each event card, the organizers can perform action by clicking on the **view or edit button** for viewing or editing event details. The editing button is only permitted when the event is in a pending or approved state.

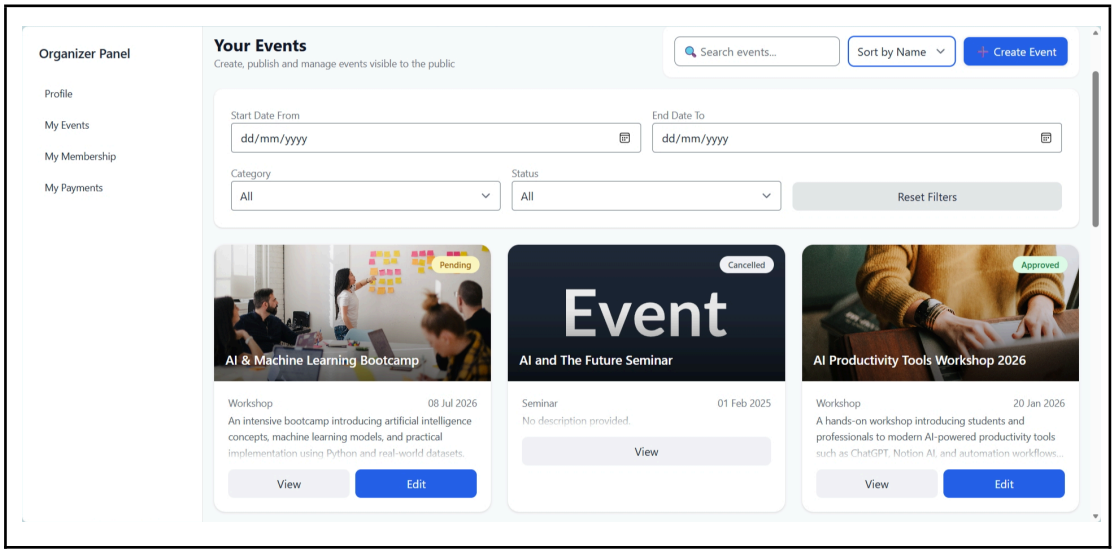

Figure 2.2.3: Organizer Site Event Listing

Class Path: *App\Livewire\Organizer\EventGrid*

Edit Event Information

The Edit Event Information function allows the organizers to **update the details of their existing events**. Organizers can modify the event information including the event name, event description, event category, proposal documents, event images or even event start and end dates. The same validation rules applied during event creation will also be applied here for ensuring data consistency and valid scheduling.

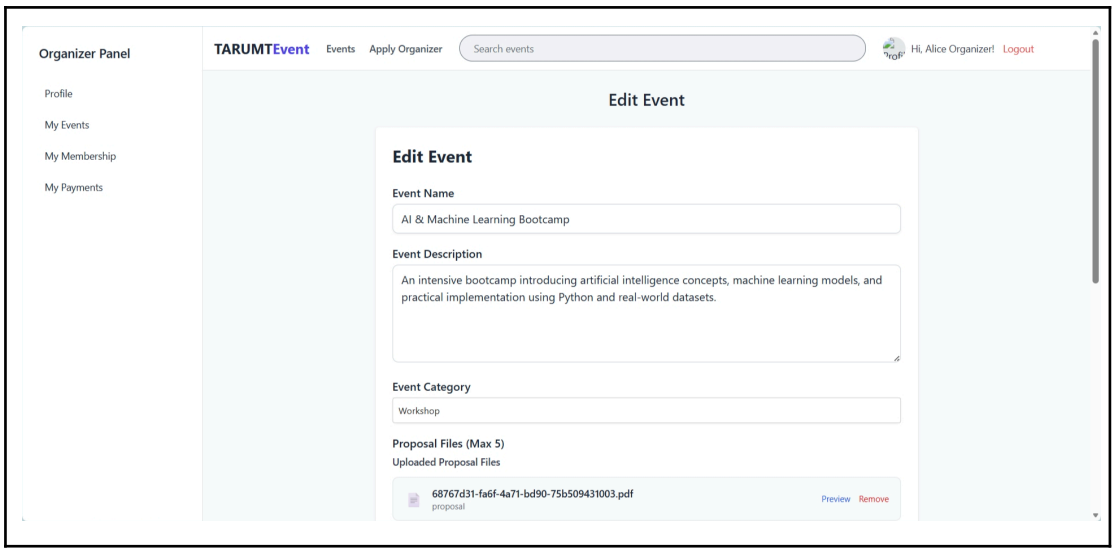

Figure 2.2.4: Organizer Site Event Editing Form - Section 1

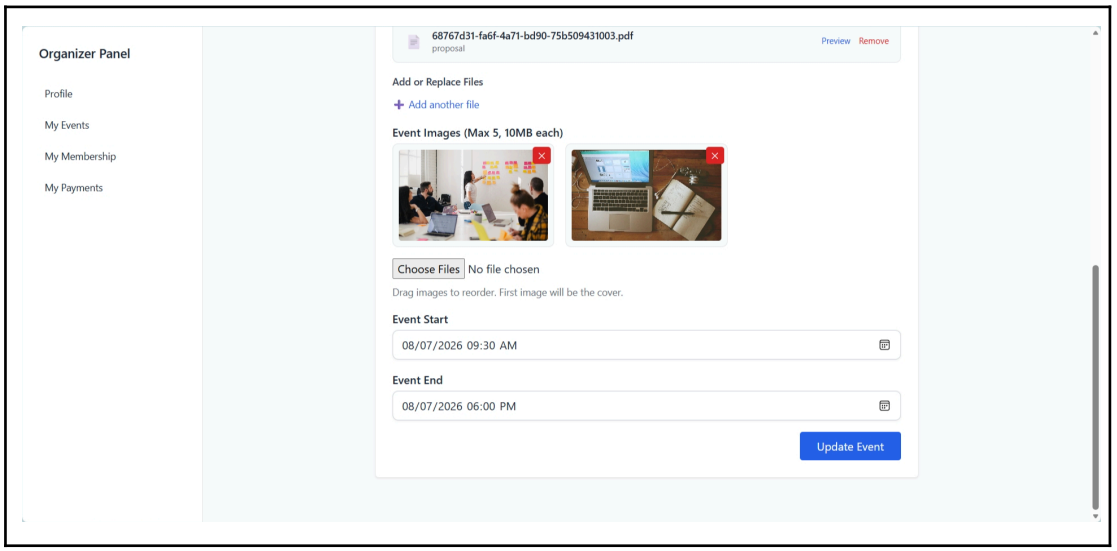

Figure 2.2.5: Organizer Site Event Editing Form - Section 2

Class Path: *App\Livewire\Organizer\EventForm*

View Event Details

The View Event Details function allows the organizers to **view comprehensive information for a selected event** through a tab-based interface. The **Overview tab displays the basic event information** such as the event name, event images, category, start and end dates and proposal documents. It also provided a label which indicates whether this event was submitted by an administrator or the organizer. The **Helpers tab shows the helper roles created for the event** in an expandable card form. It includes the role names, number of applicants and available vacancies with options to view details, edit or soft-delete existing roles. The organizers may also create new helper roles for additional recruitment from this tab. On the other hand, the **Payments tab will show all the payment records associated with the event** including the deposit and penalty payments.

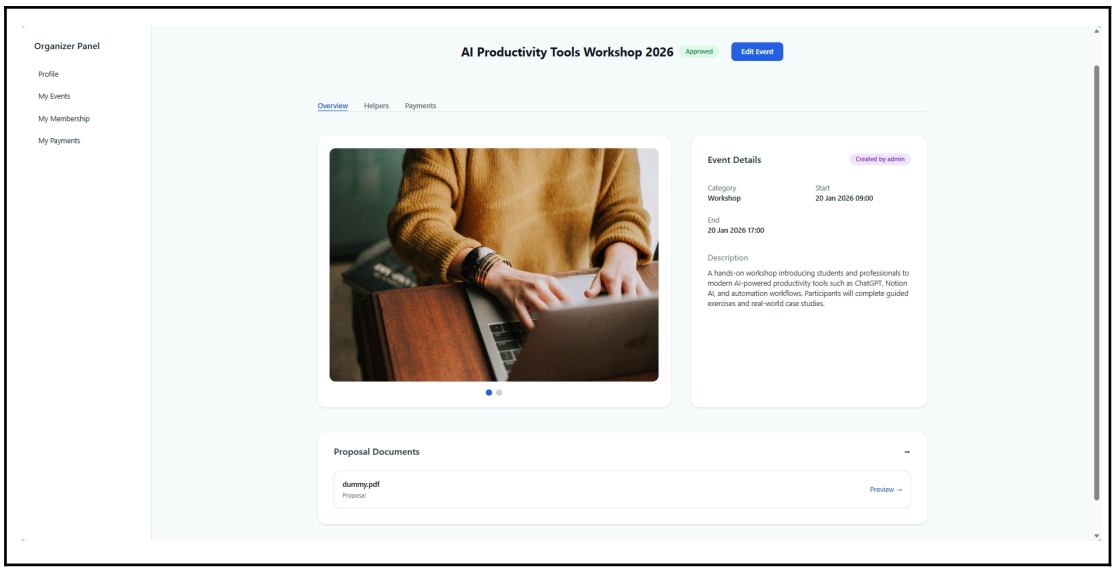

Figure 2.2.6: Organizer Site Event Details - Overview Tab

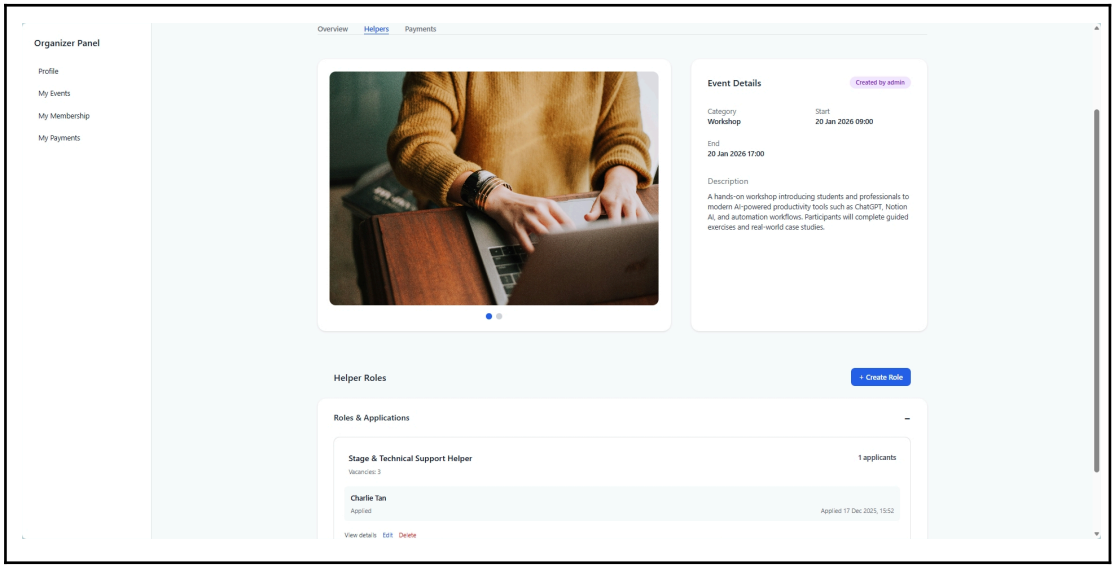

Figure 2.2.7: Organizer Site Event Details - Helpers Tab

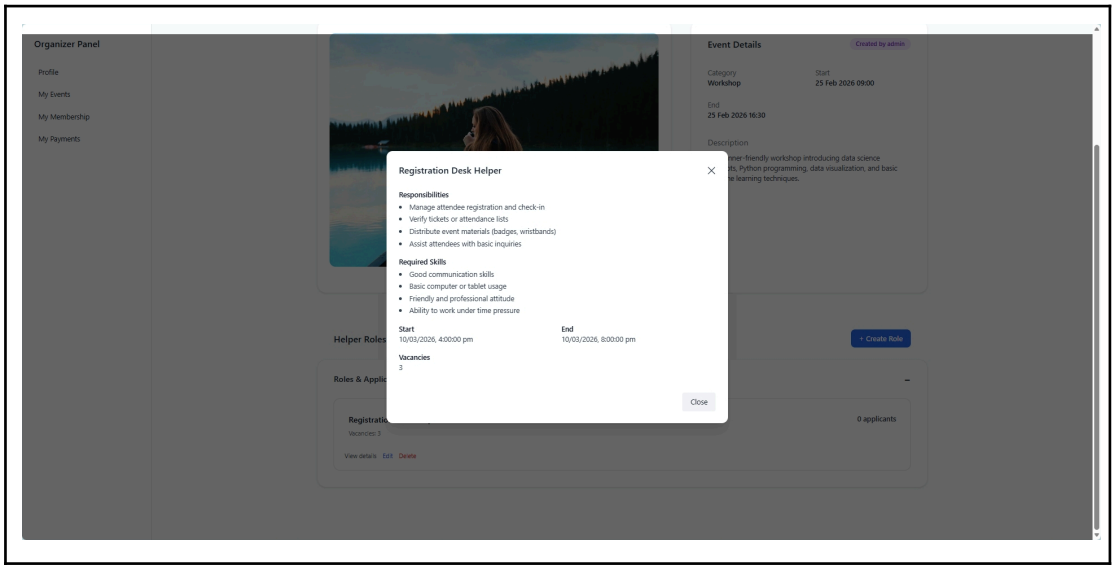

Figure 2.2.8: Organizer Site Event Details - Helper Role View Modal

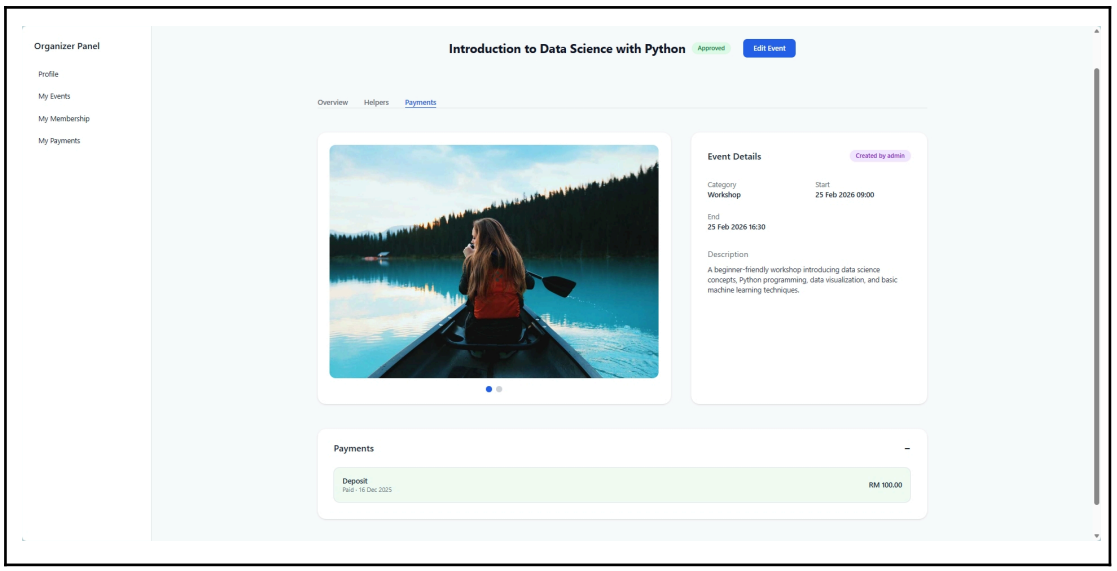

Figure 2.2.9: Organizer Site Event Details - Payments Tab

Class Path: *App\Http\Controllers\EventController@organizerShowEvent*

Create Helper Roles for Event Recruitment

The Create Helper Role function allows organizers to **define the helper positions for event recruitments**. Via this function, organizers can specify the **role name** and describe the **responsibilities** associated with the role. The system supports the expandable input fields for responsibilities and **required skills** while enabling organizers to add multiple items based on the needs of the event. Organizers are also required to **set the schedule for the helper role** by selecting the start and end date and time. Additionally, the number of vacancies can be specified to control the number of helpers required for the role.

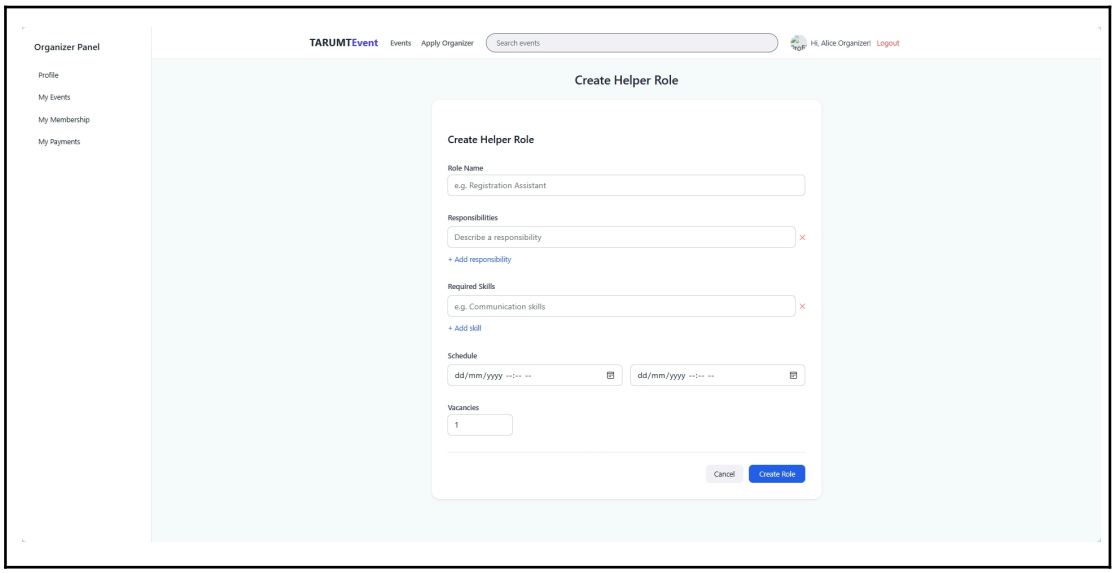

Figure 2.2.10: Organizer Site Helper Creation Form

Class Path: *App\Livewire\Organizer\EventHelperRoleForm*

Review Helper Applications

The Review Helper Applications function allows organizers to **review and manage their events' helper role applications submitted by the member**. From the Helpers tab on the event details page, the organizers can select a helper application card to open a **modal window which displays the applicant's details** such as name, email address, student ID, gender, phone number, application date, motivation, relevant experience and listed skills. After reviewing the application, the organizers can choose whether to **accept or reject the applicant**. If an application is rejected, the organizer will need to **provide a rejection reason**. Upon acceptance or rejection, the system will eventually **send out an email notification** to inform the applicant about the result of the helper role application.

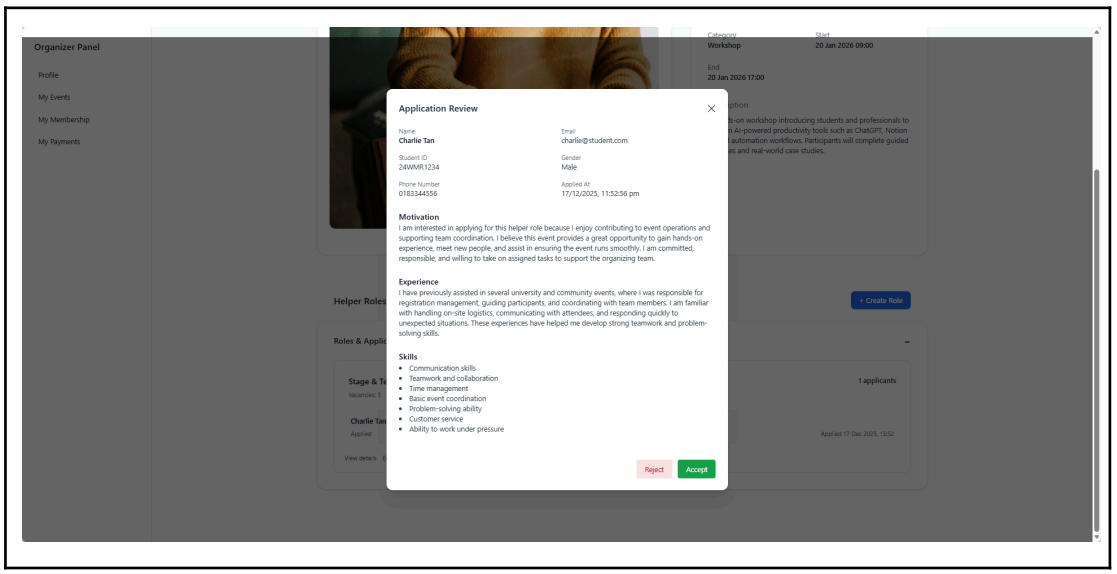

Figure 2.2.11: Organizer Site Review Application Modal

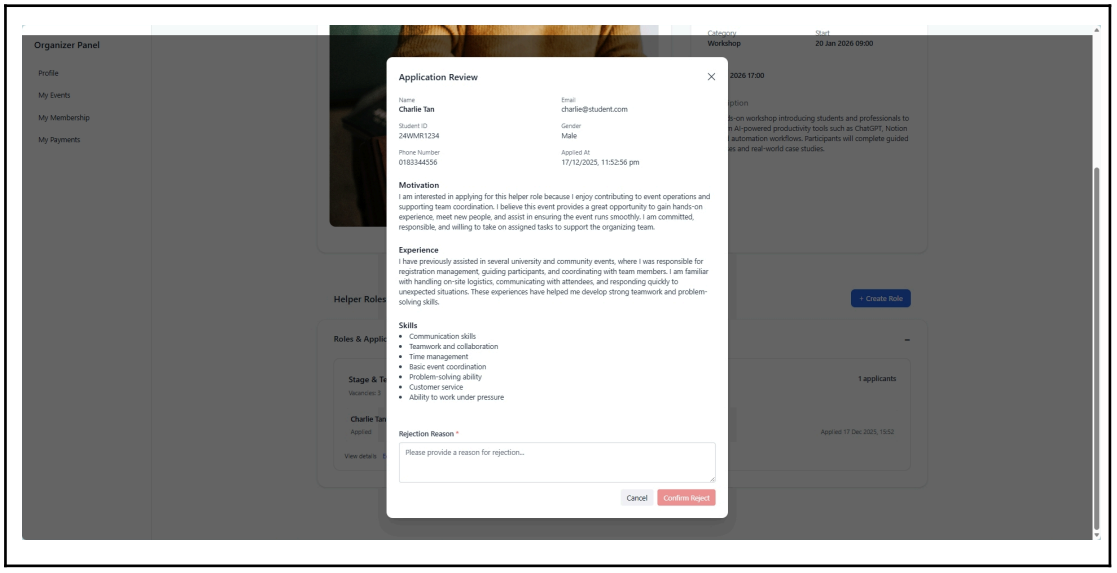

Figure 2.2.12: Organizer Site Reject Application Form

Class Path: *App\Http\Controllers\EventController@organizerShowEvent*

Member Event Participation

View Available Events

The View Available Events function allows members to **browse the events that are available within the system**. The events will be displayed in a card-based layout which displays the information such as event title, event image, category and current status. For helping the members to easily find out the relevant events, the page also provides a **search bar for event name searching**, as well as **sorting options based on event time, start date and event status in both ascending and descending order**. Additionally, the **filtering options are available to narrow down the events by category, status and selected date range**.

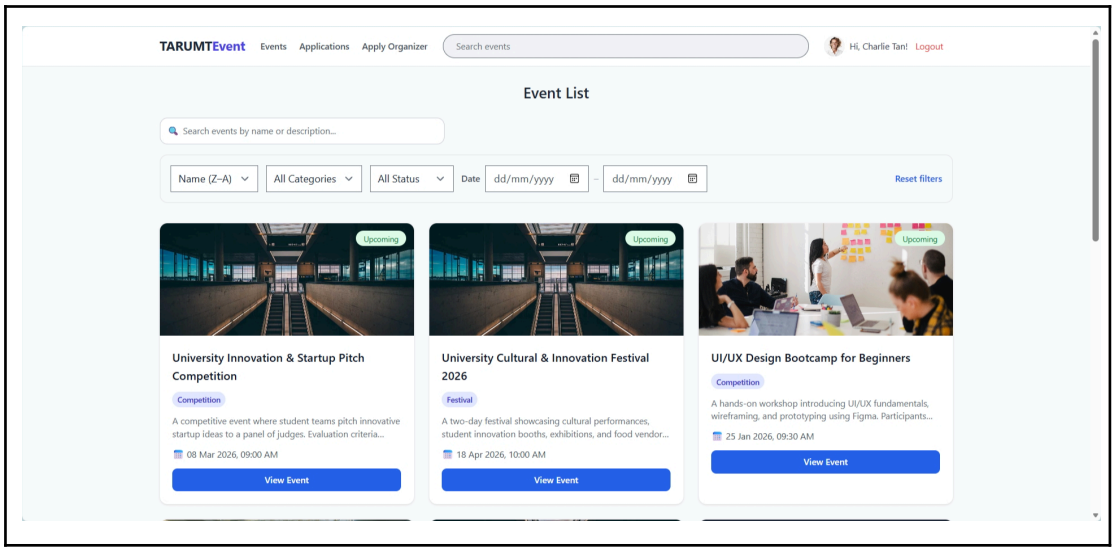

Figure 2.3.1: Member Site Event Listing

Class Path: *App\Livewire\Member\EventList*

View Event Details

The View Event Details function allows members to **view detailed information about the selected event** through a tab-based interface. The **Overview tab shows the essential event information** such as event images, event name, category, current status, description and so on. The **Recruitment tab displays all available helper role recruitments** which show the details such as role responsibilities, required skills, number of vacancies and scheduled start and end date time. The **Apply button** is also provided for each helper role to allow the members to submit applications for roles they are interested in.

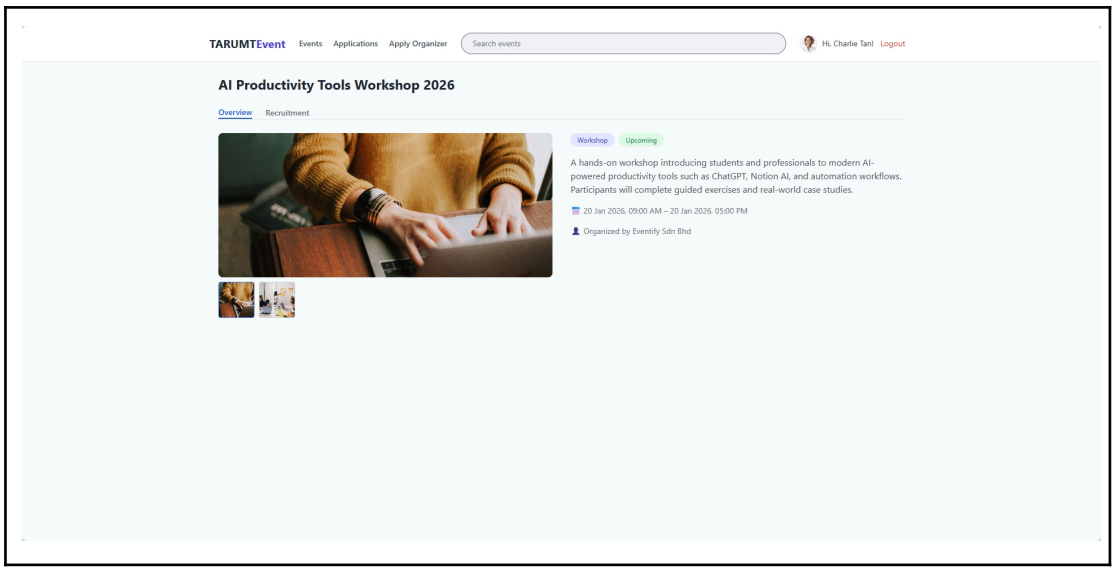

Figure 2.3.2: Member Site Event Detail - Overview Tab

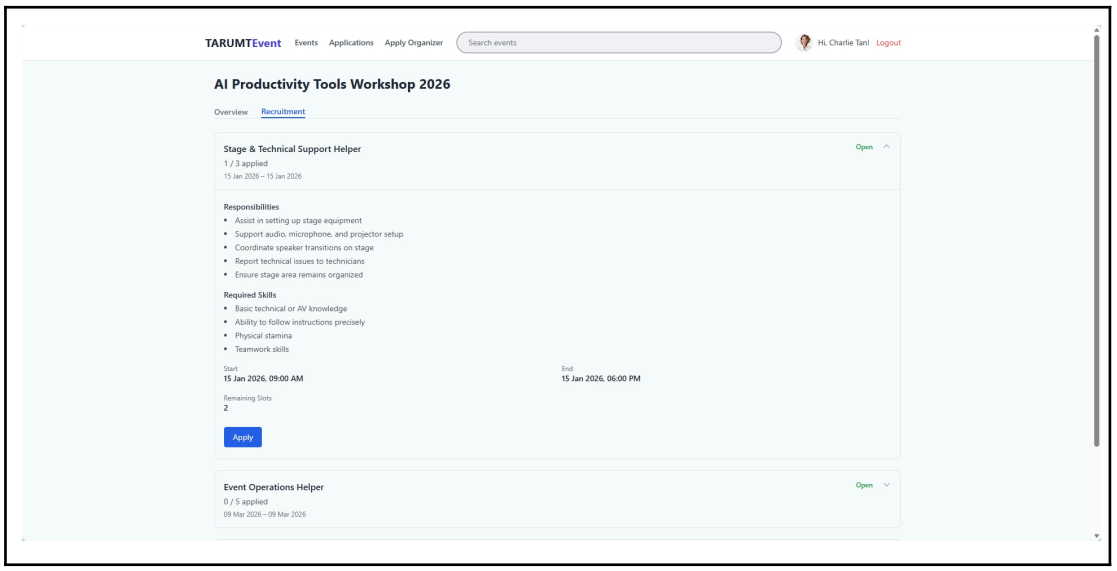

Figure 2.3.3: Member Site Event Detail - Recruitment Tab

Class Path: *App\Http\Controllers\EventController@memberOrGuestShowEvent*

Apply for Helper Roles

The Apply for Helper Roles function allows the members to **submit an application for a selected helper role**. Through this application form, the members are required to **provide their motivation and optionally provide the relevant experience and list out their skills** using expandable input fields. Once submitted, the system will record the application and automatically **send an email notification to the organizer's mailbox** to inform them there is a new application. The submitted application will also be updated in the organizer's event helper recruitment list.

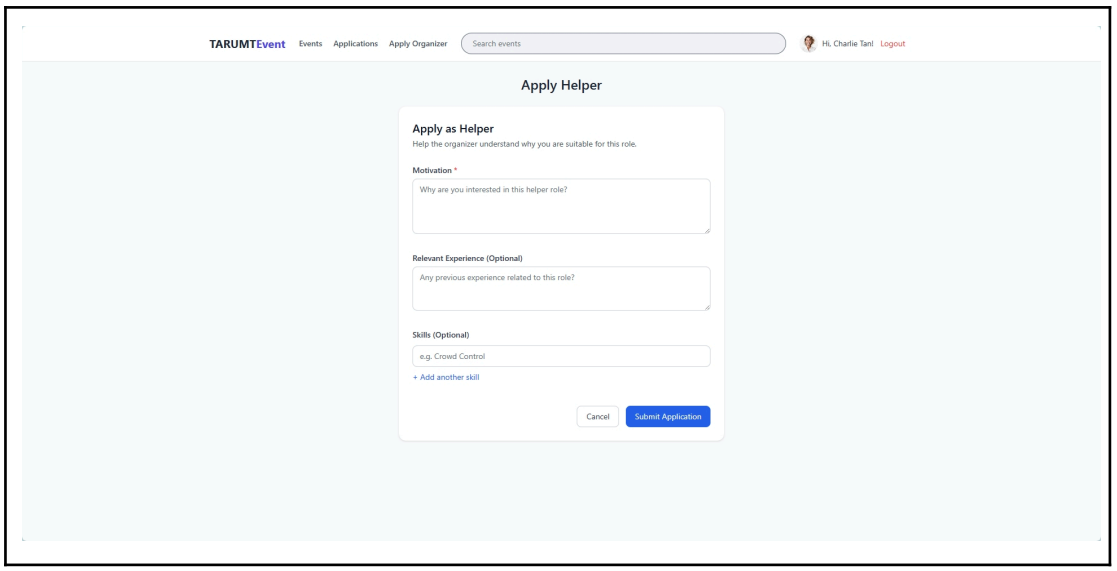

Figure 2.3.4: Member Site Apply Helper Form

Class Path: *App\Livewire\Organizer\EventHelperRoleForm*

View Helper Application History List

The Helper Application Listing function enables the members to **view all helper role applications they have submitted** in a table view. Each record displays the key information such as the event name, helper role name, role schedule (start and end date and time), application status (Applied, Accepted, Rejected, or Cancelled), application submission date and a view button. In order to enhance the usability and navigation, the table also **supports ascending and descending sorting** via clickable column headers. This can ease the members to organize applications based on their preferences. Additionally, members can **filter their applications history by status** and perform **keyword searches using event names or role names**. This can lead to an efficient retrieval of specific application records.

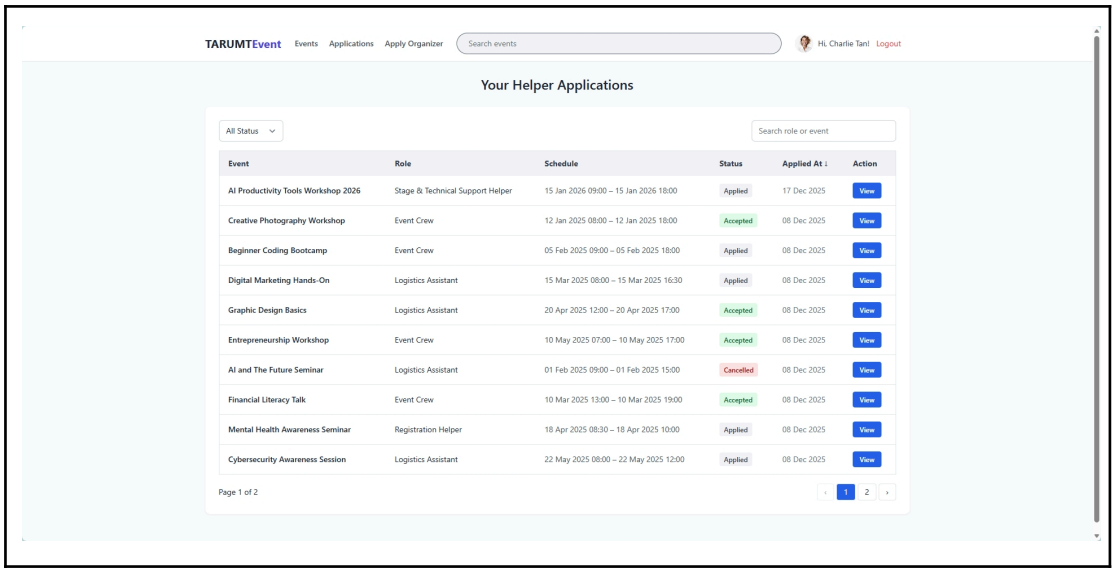

Figure 2.3.5: Member Site Helper Application History List

Class Path: *App\Livewire\Member\HelperApplicationList*

View Helper Application Detail

The Helper Application Detail function provides the members with a comprehensive view of an individual helper role application. Upon selecting the “View” action from the listing page, members are redirected to this page which displays **the detailed information related to the selected application**. For example, the associated event name, applied helper role, role schedule, application status and application submission date. Not only that, it also displays the information provided by the member during the application process. For instance, the motivation, relevant experience and listed skills.

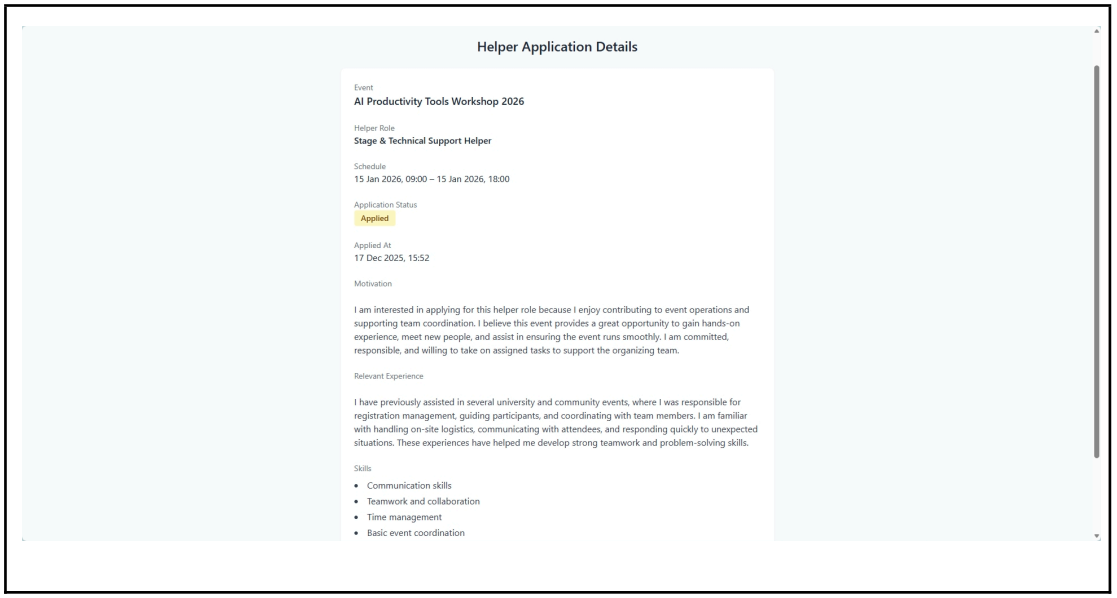

Figure 2.3.6: Member Site Helper Application Detail

Class Path: *App\Http\Controllers\EventController@memberHelperApplicationShow*

3. Entity Classes

Entity Class Diagram

Figure 3.1: Entity Class Diagram of Event Module

Entity Class Descriptions

HelperApplication

The **HelperApplication** entity acts as a volunteer or helper application submitted by a member for a specific role within an event. It stores the applicant's personal inputs such as motivation, experience and skills as well as the application status and review-related timestamps. This entity also records email notification timestamps to track communication events such as application submission, acceptance, rejection or

cancellation.

HelperRole

The **HelperRole** entity defines a specific helper position required for an event, such as registration desk staff or technical support. It includes role details such as responsibilities, required skills, schedule and the number of available vacancies. The entity also tracks remaining slots and role availability status. This allows the system to manage the helper recruitment effectively.

Event

The **Event** entity represents an organized activity created by an administrator or organizer. It contains the core event information including the name, description, schedule, images and current status. This entity also stores the event lifecycle data such as completion and reward timestamps. This allows the system to support automated status updates and organizer reward processing.

4. Design Pattern

4.1 Description of Design Pattern

Overview of the Decorator Design Pattern

The Decorator Design Pattern is a structural design pattern that allows us to dynamically add on the behavior to individual objects without modifying other objects of the same class (GeeksforGeeks, 2025). Instead of just using the inheritance to extend the functionality, the pattern will rely on the object composition which means that a decorator will wrap an existing object and enhance its behavior while maintaining the same interface.

One of the key characteristics of the Decorator pattern is that both base objects and the decorator can implement the same common interface. This can make sure that all the decorated objects remain interchangeable and flexible with the original object. This enables multiple decorators to be combined or layered together during runtime. Eventually, this pattern can offer a greater flexibility compared to subclassing, especially when there are multiple variations of functionality required.

When it comes to the Event Module, the Decorator pattern is well-suited for constructing the event-related data that varies based on different viewing requirements. Events may require additional information such as helper recruitment details or payment records to be displayed depending on the user role or usage scenario. Via the application of Decorator pattern, the event data can be extended dynamically without changing the core event structure. Meanwhile, this can also promote a flexible and scalable design.

Purpose and Benefits of the Decorator Pattern

The purpose of applying the Decorator Design Pattern in the Event Module is to achieve a flexible and dynamic event view data construction without modifying the base event structure. In the TARUMT Event Management System, the event information will be displayed in different forms based on the user role and the viewing context such as displaying basic event details, helper recruitment information or payment records. The Decorator pattern allows these variations to be added incrementally when ensuring the base event data is unchanged.

One key benefit of this approach is the avoidance of excessive subclass creation. Without the Decorator pattern, different combinations of event information will require multiple specialized event view classes. This may result in an increased complexity and reduced maintainability. After encapsulating each additional responsibility such as helper recruitment data or payment details, the system can remain modular and be easier to extend within its own decorator.

Besides that, the Decorator pattern offers the separation of concerns by isolating distinct event-related features into independent components. This can improve the code readability and maintainability since the changes made on one part of the event view data would not affect the others. In overall, the use of Decorator Design Pattern can improve the scalability, flexibility and long-term maintainability of the Event

Module. This helps make it perfectly suited for handling evolving event-related requirements.

Application of the Decorator Pattern in the Event Module

In the TARUMT Event Management System, the Decorator Design Pattern is applied within the Event Module to dynamically construct event detail views based on different user roles and usage scenarios. The event information in the system is not uniform since administrators, organizers, members and public users require different levels of event-related data when viewing the same event.

Via implementation of Decorator pattern, the system will be able to extend the base event data incrementally without modifying the core event structure. The admin event detail view requires base event information along with a summarized overview of helper recruitment, such as the number of helper roles, available vacancies and applicant counts, as well as the deposit-related payment information. On the other hand, the organizer event detail view will require more comprehensive data which involves the detailed helper role information such as responsibilities, required skills, schedules, helper application details and complete payment records covering both deposits and penalties.

In the context of a member site, the member event detail view only requires base event information and publicly visible helper recruitment details. Meanwhile, the public event view is limited to basic event information only. This role-based variation in data requirements can demonstrate how the Decorator pattern can realize the flexible composition of event views by selectively adding relevant information layers. Eventually, the Event Module will achieve a scalable and maintainable design which can easily support different presentation requirements without duplicating the code or structural complexity.

4.2 Implementation of Design Pattern

Class Diagram of the Decorator Design Pattern

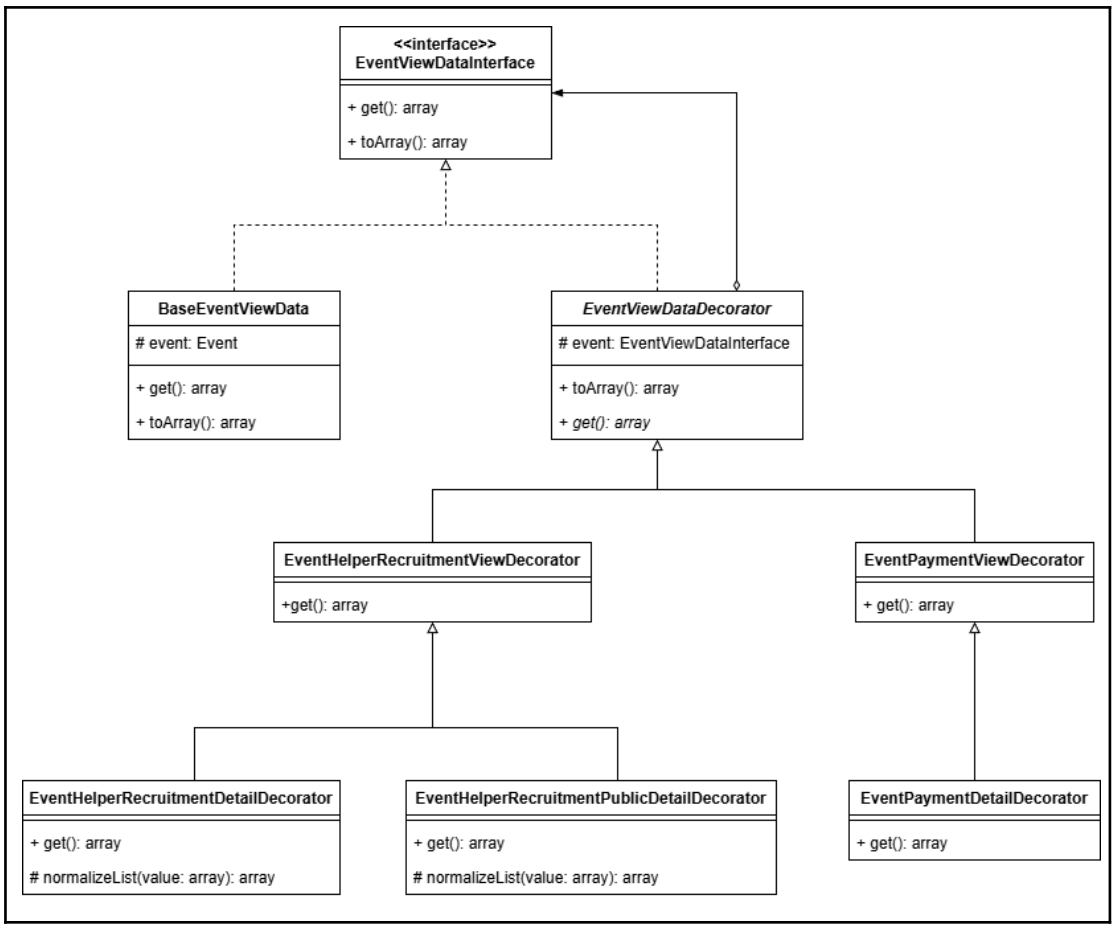

Figure 4.2: Class Diagram of Decorator Design Pattern

Figure 4.2 illustrates the class diagram of the Decorator Design Pattern as implemented in the Event Module. The core of the design is the interface class named **EventViewDataInterface**, it is responsible for defining the common contract to retrieve the event-related data. This interface can make sure that both base event data and all decorators can be treated uniformly.

The **BaseEventViewData** class serves as the concrete component and provides the fundamental event information. It implements the common interface and represents the core event data that is always required regardless of viewing context. In order to support extensibility, the **EventViewDataDecorator** class will act as an abstract decorator. It also implements the same interface and maintains a reference to another **EventViewDataInterface** object. This allows the additional data to be layered dynamically.

On the other hand, the concrete decorators will extend the abstract decorator to enrich the event data with specific responsibilities. The **EventHelperRecruitmentViewDecorator** and **EventPaymentViewDecorator** add helper recruitment and payment-related information respectively. These decorators are further specialized to support different levels of detail such as detailed or public-facing helper recruitment data and detailed payment records. Via this hierarchical structure, multiple decorators can be combined to construct the customized event views without modifying the base event data class.

Implementation in the Event Module

EventViewDataInterface

The **EventViewDataInterface** defines a common contract for all event view data components within the Event Module. It ensures that both the base event data class and all decorator classes expose a consistent interface for retrieving the event-related information. Via enforcement of shared structure, this interface will allow the decorated event data objects to remain interchangeable regardless of how many decorators are applied. This plays an important role in enabling a flexible composition of event data when maintaining the consistency across different event views.

BaseEventViewData

The **BaseEventViewData** class serves as the core concrete component in the Decorator Design Pattern implementation. It mainly provides the fundamental event information that is required across all viewing contexts, regardless of user role. This class represents the base event data and acts as the foundation upon which additional event-related information can be layered through decorators. All event views begin with this base component to ensure a consistent and reliable starting point for data construction.

EventViewDataDecorator

The **EventViewDataDecorator** class works as an abstract decorator that implements the same interface as the base event data component. It still maintain a reference to another **EventViewDataInterface** instance which allows it to wrap and extend existing event data dynamically. This class provides the structural backbone for all concrete decorators which enables the additional responsibilities to be added without modifying the base event data class. By centralizing the wrapping logic, it can support many clean and reusable extensions of event-related functionality.

EventHelperRecruitmentViewDecorator

The **EventHelperRecruitmentViewDecorator** extends the abstract decorator to introduce the helper recruitment-related information into the event view data. It enhances the base event data with an overview of helper recruitment activities associated with an event. This decorator can provide a summarized representation of helper recruitment to make it suitable for views that require awareness of recruitment status without exposing detailed application-level information.

EventHelperRecruitmentDetailDecorator

The **EventHelperRecruitmentDetailDecorator** further extends the helper recruitment decorator to provide comprehensive helper recruitment details. This includes in-depth information related to helper roles and applications, intended for users who require a full visibility into the recruitment process. Via isolation of detailed recruitment data within this decorator, the system can make sure that all the sensitive or role-specific information is only included in event views where it is appropriate.

EventHelperRecruitmentPublicDetailDecorator

The **EventHelperRecruitmentPublicDetailDecorator** is responsible for exposing helper recruitment information that is suitable for member-facing views. Unlike the detailed recruitment decorator, this class limits the scope of information to publicly visible helper role details. This can ensure that members users can view recruitment opportunities without accessing internal or sensitive application data to support controlled data exposure within the Event Module.

EventPaymentViewDecorator

The **EventPaymentViewDecorator** takes the role of extending the event view data to include the payment record which is associated with the specific event. It introduces payment awareness into the event view which allows certain roles to view payment status or requirements without accessing the detailed payment records. It provides a modular way to attach payment-related context to an event while keeping financial concerns separated from core event data.

EventPaymentDetailDecorator

The **EventPaymentDetailDecorator** will further enhance the payment decorator by providing the detailed payment records associated with an event. This includes comprehensive payment information required for organizer-level view. By encapsulating detailed financial data within this decorator, the system can ensure that sensitive payment information is only included in event views that require this access.

Summary of Event View Data Components and Data Scope

Component / Class	Purpose in Event Module	Data Provided
BaseEventViewData	Provides the core event information that serves as the foundation for all event views	Complete event data including event category information, as well as organizer details such as organizer identifier, company name, organizer type, address, registration number, linked user account, contact name, email address, telephone number and gender
EventHelperRecruitmentViewDecorator	Extends base event data with an overview of helper recruitment information	Base event data combined with helper recruitment overview data, including helper role identifier, role name, number of vacancies and total helper application count

EventHelperRecruitmentDetailDecorator	Adds comprehensive helper recruitment and application details for organizer-level views	Inherited helper recruitment overview data along with detailed helper role information such as responsibilities, required skills, schedule, status, and vacancy information, as well as detailed helper application data including applicant identifiers, application timestamps, motivation, experience, skills, review status and rejection information
EventHelperRecruitmentPublicDetailDecorator	Provides publicly visible helper recruitment information for member views	Inherited helper recruitment overview data combined with members accessible helper role details, including responsibilities, required skills, schedule, vacancy information, role status and helper availability indicators
EventPaymentViewDecorator	Adds basic payment-related information to the event view	Deposit-related payment data including payment identifier, payment type, discount amount, total amount, payment status, payment method, received amount, change amount and payment timestamp
EventPaymentDetailDecorator	Extends payment information with complete financial records	Inherited deposit payment data along with all additional payment records associated with the event, including both deposit and penalty-related payment information

Table 4.2: Summary of Event View Data Components and Data Scope

Justification for Design Pattern Selection

1. **Support for role-based data composition**

   The Event Module requires different combinations of event-related data to be presented to administrators, organizers, members and public users. The

Decorator pattern allows the event data to be dynamically extended based on the user role to ensure that each role receives only the information relevant to its responsibilities.

2. **Avoidance of class explosion**

   If using inheritance to support multiple variations of event views, it will undeniably require a large amount of subclasses to represent every possible combination of event data. Thus, the Decorator pattern can avoid this issue by encapsulating additional responsibilities such as helper recruitment and payment information within independent decorator classes that can be applied as needed.

3. **Improved maintainability and extensibility**

   By isolating event-related features into separate decorators, the changes or enhancements can be made to one aspect of the event data without affecting others. This can ease the Event Module to maintain and enable new event-related features to be added in the future with minimal impact on existing code.

4. **Clear separation of concerns**

   The Decorator pattern promotes separation of concerns by ensuring that core event data, helper recruitment logic and payment-related logic are handled independently. This can produce a cleaner code structure and better readability because each decorator is responsible for a specific aspect of the event data enrichment.

5. **Scalability for future requirements**

   Along with the involvement of TARUMT Event Management System, additional event-related data or new viewing requirements may be introduced. In order to handle this situation, the Decorator pattern can provide a scalable foundation that supports future expansion by allowing new decorators to be introduced without modifying the existing event data structure.

5. Software Security

5.1 Potential Threat/Attack

Threat 1: Unauthorized Access to Event and Helper Application Data

The Event Module might expose several sensitive functionalities which may involve helper applications view, helper roles applications, events creation and management. Without a proper access control mechanism, the attacker can easily bypass the user interface and directly access to the protected API endpoints by manipulating request URLs or even forging HTTP requests. For example, a non-member user can retrieve the member-only helper application data or an authenticated user can try to access the event management functions that should not be permitted to them.

This type of attack is categorized as Broken Access Control. It means that there are insufficient authentication or authorization checks to allow users to perform actions or access resources beyond their privileges (OWASP Foundation, 2021). If exploited, this type of attack will lead to an unauthorized data disclosure, privilege escalation or even unauthorized modification of event and helper application records.

Threat 2: Malicious File Upload and File-Based Attacks

The Event Module may allow administrators and organizers to upload proposal documents and event images as part of event creation and editing. Without secure file handling mechanisms, the attackers might exploit this vulnerability by uploading malicious files, oversized files or files with disguised extensions (OWASP Foundation, 2023). These files could be used to consume excessive server resources, compromise storage integrity or introduce malicious content into the system.

Additionally, if the user input were used directly in file path construction, attackers could also exploit the path traversal vulnerabilities to overwrite the existing files or store the files in unintended locations. These file-based attacks will pose some serious risks such as data corruption, denial-of-service attacks and potential system compromise.

5.2 Secure Coding Practice

Secure Coding Practice 1: Role-Based Access Control with Token-Based Authentication

In order to mitigate the unauthorized access to sensitive event and helper application data, the Event Module has enforced a role-based access control (RBAC) using a deny-by-default security model. The access to protected event-related functionalities will be restricted based on the user roles such as member, organizer and administrator. This can make sure that the users can only perform actions permitted by their assigned role (OWASP Foundation, 2023).

All the event-related API endpoints are protected at the routing level where the unauthorized requests are denied by default. Only the authenticated users are

allowed to access these endpoints for preventing the unauthorized users from invoking protected operations directly.

```
Route::middleware('auth:sanctum')->group(function () {
    Route::post( uri: '/events/helper-applications/apply', [EventController::class, 'applyHelperApplication']); /api/events/h
    Route::get( uri: '/events/member/helper-applications/list', [EventController::class, 'getMemberHelperApplications']);
});
```

Figure 5.2.1: Route protected with auth:sanctum

The token-based authentication is used for verifying the identity of the users when communicating with the Event API. The short-lived access tokens are generated and managed securely. This can ensure that API requests are authenticated while reducing the risk of token misuse. These tokens are scoped and time-limited to further enhance security.

```
public static function getApiToken(): string  1 usage  LIM JUN WEI
{
    $user = auth()->user();

    if (session()->has('sanctum_api_token')) {
        return session( key: 'sanctum_api_token');
    }

    $token = $user->createToken(
        name: 'livewire-api',
        ['api-access'],
        Carbon::now()->addHours( value: 4)
    )->plainTextToken;

    session(['sanctum_api_token' => $token]);

    return $token;
}
```

Figure 5.2.2: getApiToken() method

Before invoking the protected event-related API endpoints, the system will need to prepare authenticated requests by obtaining a valid access token for the current user. The token is attached to the request header and used to authenticate the API call. If the request fails authentication or returns an unexpected response, the operation is aborted immediately to prevent unauthorized or inconsistent access to event data.

```
$token = SanctumHelper::getApiToken();

$response = Http::withToken($token)
    ->acceptJson()
    ->get(
        url: config( key: 'services.event.url') . '/api/events/member/helper-applications/list',
        [
            'timestamp' => now()->toIso8601String(),
            'search' => $this->search,
            'status' => $this->status,
            'sort_by' => $this->sortField,
            'sort_dir' => $this->sortDirection,
            'page' => $this->page,
        ]
    );
```

Figure 5.2.3: Calling getApiToken() before API request

In addition to authentication, the authorization is also explicitly validated on every request. The controller-level role checks are enforced to ensure that only users with the appropriate role can access specific resources. For example, member-only endpoints validate that the authenticated user is a member before processing the request. Any unauthorized attempt to access restricted pages or API routes will be immediately denied.

```
$user = auth()->user();
abort_if(!$user->isMember(),  code: 403);
```

Figure 5.2.4: Http::withToken() API call

By combining the route-level protection, authenticated API access and role-based authorization checks, the Event Module will ensure that the access permissions are validated on every request. This layered access control strategy can effectively prevent the privileges escalation and unauthorized access issues even if the users attempt to bypass the user interface or directly invoke protected endpoints.

Secure Coding Practice 2: Secure File and Memory Management

In order to overcome the malicious file upload and file-based attacks within the Event Module, the secure file and memory management practices are implemented during event creation and editing. These practices ensure that all the uploaded proposal documents and event images are handled safely without exposing the system to risks. For example, the file system abuse, path traversal or excessive memory consumption.

Uploaded files are strictly controlled in terms of allowed file formats, file size limits and upload quantity before being processed. The proposal documents are restricted to approved document formats when the event images are limited to valid image files. The maximum file size and count constraints are enforced to reduce the risk of denial-of-service attacks caused by oversized or excessive uploads (OWASP Foundation, 2023).

```
'proposal_files' => 'required|array|min:1|max:' . self::MAX_FILES,
'proposal_files.*.file' =>
    'required|file|mimes:pdf,doc,docx,ppt,pptx|max:10240',
'event_images' => 'array|max:' . self::MAX_IMAGES,
'event_images.*' => 'image|max:10240',
```

Figure 5.2.5: Validation rules enforcing allowed file types, file size limits and upload quantity for event file uploads

On the other hand, the system will never use the user input as the way to construct the file storage path for preventing path traversal and unauthorized file placement. All the file paths will be generated internally by the system using unique identifiers. The original filenames provided by the users are preserved only as metadata and are not used to determine where files are stored. Thus, this methodology can effectively prevent the attackers from manipulating the file paths to overwrite the existing files or store files in unintended locations.

```
$path = 'proposals/tmp/' . Str::uuid() . '.' . $file->getClientOriginalExtension();

$path = 'events/images/' . Str::uuid() . '.' . $image->getClientOriginalExtension();
```

Figure 5.2.6: System generated file paths using unique identifiers

Meanwhile, the uploaded files will also be handled temporarily in memory and streamed directly to the external object storage. This can ensure that the raw file contents are not retained longer than necessary within the application. This can efficiently reduce the memory exposure and limit the risk of sensitive data leakage.

```
fopen($file->getRealPath(), mode: 'r'),
```

Figure 5.2.7: Streaming of uploaded files to external storage

Only the validated file metadata such as file paths and public URLs will be transmitted to the backend API. The backend will further validate this metadata before persisting it to ensure that all the malformed or malicious data is rejected.

```
'proposal_files.*.url' => 'required|url',
'event_images.*.url' => 'required|url',
```

Figure 5.2.8: Backend validation of uploaded file metadata before event creation

Lastly, the sensitive credentials used for file uploads are also securely stored in environment configuration files and are not exposed to client-side code or users. This may ensure that the access to the external storage service always remains protected and cannot be misused by unauthorized parties.

```php
private function uploadToSupabase($file, string $path): void  2 usages  LIM JUN WEI
{
    $response = Http::withHeaders([
        'Authorization' => 'Bearer ' . config( key: 'services.supabase.service_role_key'),
        'x-upsert' => 'true',
    ])->withBody(
        fopen($file->getRealPath(),  mode: 'r'),
        $file->getMimeType()
    )->put(
         url: config( key: 'services.supabase.url') . "/storage/v1/object/" .
        config( key: 'services.supabase.bucket') . "/{$path}"
    );

    if (!$response->successful()) {
        throw new \Exception( message: 'Supabase upload failed');
    }
}
```

Figure 5.2.9: Secure upload files to external storage using environment-based credentials

6. Web Services

1. Service Exposure

The Event Module exposes a **RESTful web service** for allowing other system modules to retrieve the event-related information in a standardized and loosely coupled manner. This service is implemented using a JSON-based REST API which provides a lightweight and flexible communication method. This method is also useful and suitable for modern web applications nowadays. The exposed service is designed to be consumed by other modules such as Finance Module. Meanwhile, all the exposed services are also complying with the Interface Assignment (IFA) standards by including the mandatory request timestamps and structured response formats to ensure traceability and consistency.

Example of Exposed Web Service

In order to demonstrate how the exposed web services are consumed by other modules, the following example shows how an external module invokes the Event Module's API to update an event's status.

Consumed Web Service Details

- Service Provider Module: Event Module
- Consuming Module: Finance Module Module
- API Route: /api/events/{event}/status
- HTTP Method: PATCH
- Purpose: Update the status for a specific event

```
Route::patch( uri: '/events/{event}/status', [EventController::class, 'updateStatus']);
```

Figure 6.1.1: REST API endpoint exposed by the Event Module for updating event status

```
protected function rejectEvent(int $eventId): bool  1 usage   LIM JUN WEI +1
{
    $response = Http::patch(
        url: config( key: 'services.event.url') . "/api/events/{$eventId}/status",
        [
            'timestamp' => now()->toIso8601String(),
            'action' => 'rejected',
        ]
    );

    return
        $response->successful()
        && $response->json( key: 'status') === 'success';
}
```

Figure 6.1.2: Example of Finance Module consuming the Event Module's web service to update event status

2. Webservice Mechanism

adminListEvents

	Description
Protocol	RESTFUL
Function Description	Retrieves a paginated list of events for administrators with support for searching, filtering by status, category and date range, and sorting by selected fields.
Source Module	Event Module
Target Module	Admin Sidebar, Admin Site Event Listing
URL	http://127.0.0.1:8002/api/events/admin/list
Function Name	adminListEvents

Table 6.2.1: Interface Agreement (IFA) - adminListEvents Web Service

organizerListEvents

	Description
Protocol	RESTFUL
Function Description	Retrieves a paginated list of events for an organizer with support for searching, filtering by status, category and date range, and sorting by selected fields.
Source Module	Event Module
Target Module	Organizer Dashboard, Organizer Site Event Listing
URL	http://127.0.0.1:8002/api/events/organizer/list
Function Name	organizerListEvents

Table 6.2.2: Interface Agreement (IFA) - organizerListEvents Web Service

memberListEvents

	Description
Protocol	RESTFUL
Function Description	Retrieves a paginated list of publicly available events for members, with support for searching by event name, filtering by category, status, and date range, and sorting by selected event fields.
Source Module	Event Module

Target Module	Member Site Event Listing
URL	http://127.0.0.1:8002/api/events/member/list
Function Name	memberListEvents

Table 6.2.3: Interface Agreement (IFA) - memberListEvents Web Service

listEventCategories

	Description
Protocol	RESTFUL
Function Description	Retrieves the list of all available event categories used for event classification and filtering.
Source Module	Event Module
Target Module	Event Creation and Editing Form, Event Listing Filtering
URL	http://127.0.0.1:8002/api/events/list/eventCategories
Function Name	listEventCategories

Table 6.2.4: Interface Agreement (IFA) - listEventCategories Web Service

bulk

	Description
Protocol	RESTFUL
Function Description	Retrieves multiple event records based on a list of event IDs and enriches each event with its associated organizer information.
Source Module	Event Module
Target Module	Finance Module, Event Listing Table
URL	http://127.0.0.1:8002/api/events/bulk
Function Name	bulk

Table 6.2.5: Interface Agreement (IFA) - bulk Web Service

getPenaltyEligibleEvents

	Description
Protocol	RESTFUL

Function Description	Retrieves a list of events that are eligible for penalty processing based on their status and event timeline, with associated organizer information included.
Source Module	Event Module
Target Module	Penalty Payment Creation Service
URL	http://127.0.0.1:8002/api/events/penalty-eligible
Function Name	getPenaltyEligibleEvents

Table 6.2.6: Interface Agreement (IFA) - getPenaltyEligibleEvents Web Service

store

	Description
Protocol	RESTFUL
Function Description	Creates a new event based on the submitted event details and initializes related data such as proposal documents and event images. If a deposit is required, the service automatically triggers the creation of a corresponding deposit payment through the Finance module.
Source Module	Event Module
Target Module	Admin Site Event Creation Service
URL	http://127.0.0.1:8002/api/events/store
Function Name	store

Table 6.2.7: Interface Agreement (IFA) - store Web Service

organizerStore

	Description
Protocol	RESTFUL
Function Description	Allows an organizer to create a new event by validating the organizer's identity and ownership, storing the event in a pending state and automatically initiating a required deposit payment through the Finance module.
Source Module	Event Module
Target Module	Organizer Site Event Creation Service
URL	http://127.0.0.1:8002/api/events/organizer/store

Function Name	organizerStore

Table 6.2.8: Interface Agreement (IFA) - organizerStore Web Service

storeEventHelperRole

	Description
Protocol	RESTFUL
Function Description	Creates a new helper role for a specified event by storing role details such as responsibilities, required skills, schedule and available vacancies to support event helper recruitment.
Source Module	Event Module
Target Module	Event Helper Role Creation Service
URL	http://127.0.0.1:8002/api/events/{event}/helper-roles
Function Name	storeEventHelperRole

Table 6.2.9: Interface Agreement (IFA) - storeEventHelperRole Web Service

acceptHelperApplication

	Description
Protocol	RESTFUL
Function Description	Accepts a helper application for an event by updating the application status, reducing the remaining vacancies for the corresponding helper role and triggering follow-up actions when the role reaches full capacity.
Source Module	Event Module
Target Module	Helper Application Review Service
URL	http://127.0.0.1:8002/api/events/helper-applications/{application}/accept
Function Name	acceptHelperApplication

Table 6.2.10: Interface Agreement (IFA) - acceptHelperApplication Web Service

rejectHelperApplication

	Description
Protocol	RESTFUL

Function Description	Rejects a helper application by updating its status with a rejection reason and recording the review timestamp while triggering notification-related follow-up actions.
Source Module	Event Module
Target Module	Helper Application Review Service
URL	http://127.0.0.1:8002/api/events/helper-applications/{application}/reject
Function Name	rejectHelperApplication

Table 6.2.11: Interface Agreement (IFA) - rejectHelperApplication Web Service

updateEventHelperRole

	Description
Protocol	RESTFUL
Function Description	Updates an existing helper role for a specific event by modifying role details and vacancy information while ensuring data consistency with already accepted helper applications.
Source Module	Event Module
Target Module	Event Helper Role Editing Service
URL	http://127.0.0.1:8002/api/events/{event}/helper-roles/{helperRole}
Function Name	updateEventHelperRole

Table 6.2.12: Interface Agreement (IFA) - updateEventHelperRole Web Service

destroyEventHelperRole

	Description
Protocol	RESTFUL
Function Description	Deletes a helper role for an event by deactivating the role and automatically cancelling any pending helper applications, subject to the event's current status.
Source Module	Event Module
Target Module	Event Helper Role Deletion Service
URL	http://127.0.0.1:8002/api/events/{event}/helper-roles/{helperRole}

Function Name	destroyEventHelperRole

Table 6.2.13: Interface Agreement (IFA) - destroyEventHelperRole Web Service

update

	Description
Protocol	RESTFUL
Function Description	Updates an existing event by modifying its core details, status, images and proposal documents while ensuring data consistency through transactional processing.
Source Module	Event Module
Target Module	Admin Site Event Editing Service
URL	http://127.0.0.1:8002/api/events/{event}
Function Name	update

Table 6.2.14: Interface Agreement (IFA) - update Web Service

organizerUpdate

	Description
Protocol	RESTFUL
Function Description	Allows an organizer to update an existing event by validating organizer identity and ownership, modifying event details and associated files and enforcing status rules to ensure proper event lifecycle control.
Source Module	Event Module
Target Module	Organizer Site Event Editing Service
URL	http://127.0.0.1:8002/api/events/organizer/{event}
Function Name	organizerUpdate

Table 6.2.15: Interface Agreement (IFA) - organizerUpdate Web Service

updateStatus

	Description
Protocol	RESTFUL
Function	Updates the status of an event based on administrative actions

Description	such as approval, rejection or cancellation while validating payment completion and coordinating with the Finance module to manage related deposit payments.
Source Module	Event Module
Target Module	Admin Site Event Status Updating Service
URL	http://127.0.0.1:8002/api/events/{event}/status
Function Name	updateStatus

Table 6.2.16: Interface Agreement (IFA) - updateStatus Web Service

getById

	Description
Protocol	RESTFUL
Function Description	Retrieves detailed information of a specific event based on the provided event identifier.
Source Module	Event Module
Target Module	Event Editing Form
URL	http://127.0.0.1:8002/api/events/{id}
Function Name	getById

Table 6.2.17: Interface Agreement (IFA) - getById Web Service

applyHelperApplication

	Description
Protocol	RESTFUL
Function Description	Submits a helper application for a selected event role by validating eligibility conditions and recording the applicant's details while preventing duplicate applications and triggering follow-up notification services.
Source Module	Event Module
Target Module	Member Site Helper Application Form
URL	http://127.0.0.1:8002/api/events/helper-applications/apply
Function Name	applyHelperApplication

Table 6.2.18: Interface Agreement (IFA) - applyHelperApplication Web Service

getMemberHelperApplications

	Description
Protocol	RESTFUL
Function Description	Retrieves a paginated list of helper applications submitted by a member, with support for searching by event or role name, filtering by application status and sorting by selected fields.
Source Module	Event Module
Target Module	Member Site Helper Application Listing Service
URL	http://127.0.0.1:8002/api/events/member/helper-applications/list
Function Name	getMemberHelperApplications

Table 6.2.19: Interface Agreement (IFA) - getMemberHelperApplications Web Service

Web Services Request Parameter (provide)

adminListEvents

Field Name	**Field Type**	**Mandatory / Optional**	**Description**	**Format**
timestamp	String	Mandatory	Time when the request was made	ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ)
search	String	Optional	Keyword search based on event name	Free text
status	String	Optional	Filter events by status (e.g., pending, approved, ongoing, completed, cancelled, rejected)	String value
startDateFrom	Date	Optional	Filter events that end on or after the specified date	YYYY-MM-DD

startDateTo	Date	Optional	Filter events that start on or before the specified date	YYYY-MM-DD
sortBy	String	Optional	Field used to sort the event list (e.g., event_name, status, event_start, event_end, organizer_name)	String value
sortDir	String	Optional	Sorting direction for the specified sort field	asc / desc
perPage	Integer	Optional	Number of events returned per page	Numeric value
page	Integer	Optional	Page number for paginated results	Numeric value

Table 6.2.20: Web Services Request Parameter (provide) - adminListEvents

organizerListEvents

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made	ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ)
organizer_id	Integer	Mandatory	Unique identifier of the organizer whose events are being retrieved	Numeric value

search	String	Optional	Keyword search based on event name	Free text
status	String	Optional	Filter events by status (e.g., draft, published, cancelled)	String value
category	Integer	Optional	Filter events by event category ID	Numeric value
start_from	Date	Optional	Filter events starting on or after this date	YYYY-MM-DD
start_to	Date	Optional	Filter events ending on or before this date	YYYY-MM-DD
sort_by	String	Optional	Field used to sort the event list (default: event_start)	Valid column name
sort_dir	String	Optional	Sorting direction	asc / desc
page	Integer	Optional	Page number for pagination	Numeric value (≥ 1)

Table 6.2.21: Web Services Request Parameter (provide) - organizerListEvents

memberListEvents

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made	ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ)
search	String	Optional	Keyword search based	Free text

			on event name	
status	String	Optional	Filter events by event status (approved, ongoing, completed)	String value
category	Integer	Optional	Filter events by event category ID	Numeric value
start_from	Date	Optional	Filter events ending on or after this date	YYYY-MM-DD
start_to	Date	Optional	Filter events starting on or before this date	YYYY-MM-DD
sort_by	String	Optional	Field used to sort the event list (event_name, event_start, event_end, status, created_at)	Valid column name
sort_dir	String	Optional	Sorting direction for the result set	asc / desc
page	Integer	Optional	Page number for pagination	Numeric value (≥ 1)
per_page	Integer	Optional	Number of records per page (maximum 30)	Numeric value (1–30)

Table 6.2.22: Web Services Request Parameter (provide) - memberListEvents

listEventCategories

Field Name	Field Type	Mandatory / Optional	Description	Format

timestamp	String	Mandatory	Time when the request was made	ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ)

Table 6.2.23: Web Services Request Parameter (provide) - listEventCategories

bulk

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made	ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ)
ids	String	Mandatory	Comma-separated list of event IDs to be retrieved in bulk	Numeric values separated by commas (e.g. 1,5,12)

Table 6.2.24: Web Services Request Parameter (provide) - bulk

getPenaltyEligibleEvents

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made	ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ)

Table 6.2.25: Web Services Request Parameter (provide) - getPenaltyEligibleEvents

store

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)

created_by_role	String	Mandatory	Role that created the event	admin / organizer
event_name	String	Mandatory	Name of the event	Text (5–255 characters)
description	String	Optional	Detailed description of the event	Text (max 5000 characters)
event_category_id	Integer	Mandatory	Identifier of the event category	Numeric value
organizer_id	Integer	Mandatory	Identifier of the event organizer	Numeric value
event_start	DateTime	Mandatory	Event start date and time	YYYY-MM-DD HH:MM:SS
event_end	DateTime	Mandatory	Event end date and time	YYYY-MM-DD HH:MM:SS
requires_deposit	Boolean	Optional	Indicates whether a deposit payment is required	true / false
proposal_files	Array	Mandatory	List of uploaded proposal documents	JSON array
proposal_files[].path	String	Mandatory	Storage path of the proposal file	File path string
proposal_files[].url	String	Mandatory	Public URL of the proposal file	Valid URL
proposal_files[].original_name	String	Mandatory	Original filename of the uploaded file	Filename string
proposal_files[].purpose	String	Mandatory	Purpose of the proposal	Text

			document	
event_images	Array	Optional	List of uploaded event images	JSON array
event_images[ ].path	String	Mandatory (if provided)	Storage path of the image	File path string
event_images[ ].url	String	Mandatory (if provided)	Public URL of the image	Valid URL
event_images[ ].position	Integer	Mandatory (if provided)	Display order of the image	Numeric value (0-based)

Table 6.2.26: Web Services Request Parameter (provide) - store

organizerStore

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
event_name	String	Mandatory	Name of the event	Text (5–255 characters)
description	String	Optional	Detailed description of the event	Text (max 5000 characters)
event_category_id	Integer	Mandatory	Identifier of the event category	Numeric value
event_start	DateTime	Mandatory	Event start date and time	YYYY-MM-DD HH:MM:SS
event_end	DateTime	Mandatory	Event end date and time	YYYY-MM-DD HH:MM:SS
proposal_files	Array	Mandatory	List of proposal documents submitted by the organizer	JSON array
proposal_files[	String	Mandatory	Storage path	File path string

].path			of the proposal file	
proposal_files[ ].url	String	Mandatory	Public URL of the proposal file	Valid URL
proposal_files[ ].original_nam e	String	Mandatory	Original name of the uploaded file	Filename string
proposal_files[ ].purpose	String	Mandatory	Purpose of the proposal document	Text
event_images	Array	Mandatory	List of event images	JSON array
event_images[ ].path	String	Mandatory	Storage path of the image	File path string
event_images[ ].url	String	Mandatory	Public URL of the image	Valid URL
event_images[ ].position	Integer	Mandatory	Display order of the image	Numeric value (0-based)
deleted_image s	Array	Optional	List of images removed during edit	JSON array
deleted_image s[].path	String	Mandatory (if provided)	Storage path of deleted image	File path string
deleted_propo sals	Array	Optional	List of proposal documents removed during edit	JSON array
deleted_propo sals[].path	String	Mandatory (if provided)	Storage path of deleted proposal	File path string

Table 6.2.27: Web Services Request Parameter (provide) - organizerStore

storeEventHelperRole

Field Name	Field Type	Mandatory /	Description	Format

		Optional		
event	Integer (Path Param)	Mandatory	Identifier of the event to which the helper role belongs	Numeric value
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
role_name	String	Mandatory	Name of the helper role	Text (max 255 characters)
responsibilities	Array	Mandatory	List of responsibilities assigned to the helper role	JSON array of strings
responsibilities[]	String	Mandatory	Description of a responsibility	Text (max 500 characters)
required_skills	Array	Mandatory	List of skills required for the helper role	JSON array of strings
required_skills[]	String	Mandatory	Description of a required skill	Text (max 255 characters)
start_datetime	DateTime	Mandatory	Start date and time of the helper role	YYYY-MM-DD HH:MM:SS
end_datetime	DateTime	Mandatory	End date and time of the helper role	YYYY-MM-DD HH:MM:SS
vacancies	Integer	Mandatory	Total number of helpers required for the role	Numeric value (≥ 1)

Table 6.2.28: Web Services Request Parameter (provide) - storeEventHelperRole

acceptHelperApplication

Field Name	**Field Type**	**Mandatory / Optional**	**Description**	**Format**

application	Integer (Path Parameter)	Mandatory	Unique identifier of the helper application to be accepted	Numeric value
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)

Table 6.2.29: Web Services Request Parameter (provide) - acceptHelperApplication

rejectHelperApplication

Field Name	Field Type	Mandatory / Optional	Description	Format
application	Integer (Path Parameter)	Mandatory	Unique identifier of the helper application to be rejected	Numeric value
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
reason	String	Mandatory	Reason provided by the organizer for rejecting the application	Text (3–500 characters)

Table 6.2.30: Web Services Request Parameter (provide) - rejectHelperApplication

updateEventHelperRole

Field Name	Field Type	Mandatory / Optional	Description	Format
event	Integer (Path Parameter)	Mandatory	Identifier of the event that owns the helper role	Numeric value
helperRole	Integer (Path Parameter)	Mandatory	Identifier of the helper role to be updated	Numeric value

timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
role_name	String	Mandatory	Name of the helper role	Text (max 255 characters)
responsibilities	Array	Mandatory	List of responsibilities assigned to the helper role	JSON array of strings
responsibilities[]	String	Mandatory	Description of a responsibility	Text (max 500 characters)
required_skills	Array	Mandatory	List of skills required for the helper role	JSON array of strings
required_skills[]	String	Mandatory	Description of a required skill	Text (max 255 characters)
start_datetime	DateTime	Mandatory	Start date and time of the helper role	YYYY-MM-DD HH:MM:SS
end_datetime	DateTime	Mandatory	End date and time of the helper role	YYYY-MM-DD HH:MM:SS
vacancies	Integer	Mandatory	Updated total number of helpers required	Numeric value (≥ 1)

Table 6.2.31: Web Services Request Parameter (provide) - updateEventHelperRole

destroyEventHelperRole

Field Name	Field Type	Mandatory / Optional	Description	Format
event	Integer (Path Parameter)	Mandatory	Identifier of the event that owns the helper role	Numeric value
helperRole	Integer (Path Parameter)	Mandatory	Identifier of the helper role to	Numeric value

			be deleted	
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)

Table 6.2.32: Web Services Request Parameter (provide) - destroyEventHelperRole

update

Field Name	Field Type	Mandatory / Optional	Description	Format
event	Integer (Path Parameter)	Mandatory	Unique identifier of the event to be updated	Numeric value
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
event_name	String	Mandatory	Name of the event	Text (5–255 characters)
description	String	Optional	Description of the event	Text (max 5000 characters)
event_category_id	Integer	Mandatory	Identifier of the event category	Numeric value
event_start	DateTime	Mandatory	Event start date and time	YYYY-MM-DD HH:MM:SS
event_end	DateTime	Mandatory	Event end date and time	YYYY-MM-DD HH:MM:SS
status	String	Optional	Updated event status (admin-controlled)	approved, rejected, cancelled
existing_images	Array	Optional	List of existing event images to retain	JSON array
existing_imag	String	Mandatory (if	Storage path	File path string

es[].path		provided)	of the existing image	
existing_images[].url	String	Mandatory (if provided)	Public URL of the existing image	Valid URL
existing_images[].position	Integer	Mandatory (if provided)	Display order of the image	Numeric value
new_images	Array	Optional	List of newly uploaded event images	JSON array
new_images[].path	String	Mandatory (if provided)	Storage path of the new image	File path string
new_images[].url	String	Mandatory (if provided)	Public URL of the new image	Valid URL
new_images[].position	Integer	Mandatory (if provided)	Display order of the image	Numeric value
deleted_images	Array	Optional	List of images removed from the event	JSON array
deleted_images[].path	String	Mandatory (if provided)	Storage path of the deleted image	File path string
new_proposals	Array	Optional	Newly uploaded proposal documents	JSON array
new_proposals[].path	String	Mandatory (if provided)	Storage path of the proposal file	File path string
new_proposals[].url	String	Mandatory (if provided)	Public URL of the proposal file	Valid URL
new_proposals[].original_name	String	Mandatory (if provided)	Original name of the uploaded file	Filename string
new_proposal	String	Mandatory (if	Purpose of the	Text

s[].purpose		provided)	proposal document	
deleted_proposals	Array	Optional	List of proposal documents removed	JSON array
deleted_proposals[].path	String	Mandatory (if provided)	Storage path of the deleted proposal	File path string

Table 6.2.33: Web Services Request Parameter (provide) - update

organizerUpdate

Field Name	Field Type	Mandatory / Optional	Description	Format
event	Integer (Path Parameter)	Mandatory	Unique identifier of the event to be updated	Numeric value
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
event_name	String	Mandatory	Name of the event	Text (5–255 characters)
description	String	Optional	Description of the event	Text (max 5000 characters)
event_category_id	Integer	Mandatory	Identifier of the event category	Numeric value
event_start	DateTime	Mandatory	Event start date and time	YYYY-MM-DD HH:MM:SS
event_end	DateTime	Mandatory	Event end date and time	YYYY-MM-DD HH:MM:SS
proposal_files	Array	Optional	Updated proposal documents (existing + new)	JSON array

proposal_files[].path	String	Mandatory (if provided)	Storage path of the proposal file	File path string
proposal_files[].url	String	Mandatory (if provided)	Public URL of the proposal file	Valid URL
proposal_files[].original_name	String	Mandatory (if provided)	Original filename	Filename string
proposal_files[].purpose	String	Mandatory (if provided)	Purpose of the proposal document	Text
event_images	Array	Optional	Updated list of event images	JSON array
event_images[].path	String	Mandatory (if provided)	Storage path of the image	File path string
event_images[].url	String	Mandatory (if provided)	Public URL of the image	Valid URL
event_images[].position	Integer	Mandatory (if provided)	Display order of the image	Numeric value
deleted_images	Array	Optional	Images removed by the organizer	JSON array
deleted_images[].path	String	Mandatory (if provided)	Storage path of deleted image	File path string
deleted_proposals	Array	Optional	Proposal files removed by the organizer	JSON array
deleted_proposals[].path	String	Mandatory (if provided)	Storage path of deleted proposal	File path string

Table 6.2.34: Web Services Request Parameter (provide) - organizerUpdate

updateStatus

Field Name	Field Type	Mandatory / Optional	Description	Format

event	Integer (Path Parameter)	Mandatory	Unique identifier of the event whose status is being updated	Numeric value
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
action	String	Mandatory	Action to be performed on the event status	approved, rejected, cancelled

Table 6.2.35: Web Services Request Parameter (provide) - updateStatus

getById

Field Name	Field Type	Mandatory / Optional	Description	Format
id	Integer (Path Parameter)	Mandatory	Unique identifier of the event to be retrieved	Numeric value
timestamp	String	Mandatory	Time when the request was made	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)

Table 6.2.36: Web Services Request Parameter (provide) - getById

applyHelperApplication

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made for IFA traceability	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
event_id	Integer	Mandatory	Unique identifier of the event being applied for	Numeric value

helper_role_id	Integer	Mandatory	Unique identifier of the helper role within the event	Numeric value
motivation	String	Mandatory	Applicant's motivation for applying to the helper role	Text (10–1000 characters)
experience	String	Optional	Applicant's prior experience related to the role	Text (up to 1000 characters)
skills	Array of String	Optional	List of skills relevant to the helper role	Max 10 items, each ≤ 100 characters

Table 6.2.37: Web Services Request Parameter (provide) - applyHelperApplication

getMemberHelperApplications

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	String	Mandatory	Time when the request was made for IFA traceability	ISO 8601 (YYYY-MM-DDTHH:MM:SSZ)
search	String	Optional	Search keyword for event name or helper role name	Text
status	String	Optional	Filter applications by application status	applied / accepted / rejected / cancelled
sort_by	String	Optional	Field used to sort the application list	created_at / status / role_name / start_datetime / event_name

sort_dir	String	Optional	Sorting direction	asc / desc
page	Integer	Optional	Page number for paginated results	Numeric value (≥ 1)

Table 6.2.38: Web Services Request Parameter (provide) - getMemberHelperApplications

Web Services Response Parameter (consume)

User Module

getEventOrganizers

Field Name	Field Type	Mandatory / Optional	Description	Format
status	String	Mandatory	Status of the API request	success / failed
timestamp	String	Mandatory	Time when the response was generated	ISO 8601 datetime
data	Array	Mandatory	List of available event organizers	JSON array
data[].id	Integer	Mandatory	Unique identifier of the organizer	Numeric
data[].company_name	String	Mandatory	Registered company name of the organizer	Text

Table 6.2.39: Web Services Response Parameter (consume) - getEventOrganizers

getByOrganizerIds

Field Name	Field Type	Mandatory / Optional	Description	Format
status	String	Mandatory	Status of the request	success / failed
timestamp	String	Mandatory	Time when the	ISO 8601

			response was generated	datetime
data	Object	Mandatory	Organizer–user mapping keyed by organizer ID	JSON object
data.{organizer_id}	Object	Mandatory	Organizer-related user information	JSON object
data.{organizer_id}.organizer_id	Integer	Mandatory	Unique identifier of the organizer	Numeric
data.{organizer_id}.user_id	Integer	Mandatory	Unique identifier of the user account	Numeric
data.{organizer_id}.name	String	Mandatory	Name of the user associated with the organizer	Alphabetic text
data.{organizer_id}.email	String	Mandatory	Email address of the user	Valid email format
data.{organizer_id}.blacklisted	Boolean	Mandatory	Indicates whether the organizer is blacklisted	true / false

Table 6.2.40: Web Services Response Parameter (consume) - getByOrganizerIds

getOrganizerDetailById

Field Name	**Field Type**	**Mandatory / Optional**	**Description**	**Format**
status	String	Mandatory	Status of the request	success / failed
timestamp	String	Mandatory	Time when the response was generated	ISO 8601 datetime
data	Object	Mandatory	Detailed organizer and user information	JSON object

data.organizer_id	Integer	Mandatory	Unique identifier of the organizer	Numeric
data.company_name	String	Mandatory	Registered company name of the organizer	Text
data.type	String	Mandatory	Type of organizer (e.g. internal, external)	Text
data.address	String	Mandatory	Registered business address of the organizer	Text
data.reg_no	String	Mandatory	Business registration number	Alphanumeric
data.user_id	Integer	Mandatory	Unique identifier of the associated user	Numeric
data.name	String	Mandatory	Name of the user linked to the organizer	Alphabetic text
data.email	String	Mandatory	Email address of the user	Valid email format
data.tel_no	String	Optional	Contact telephone number of the user	Numeric
data.gender	String	Optional	Gender of the user	Text

Table 6.2.41: Web Services Response Parameter (consume) - getOrganizerDetailById

isBlacklisted

Field Name	Field Type	Mandatory /	Description	Format

		Optional		
status	String	Mandatory	Status of the request	success / failed
timestamp	String	Mandatory	Time when the response was generated	ISO 8601 datetime
organizer_id	Integer	Mandatory	Unique identifier of the organizer	Numeric
blacklisted	Boolean	Mandatory	Indicates whether the organizer is blacklisted	true / false

Table 6.2.42: Web Services Response Parameter (consume) - isBlacklisted

getUserOrganizer

Field Name	**Field Type**	**Mandatory / Optional**	**Description**	**Format**
status	String	Mandatory	Status of the request	success / error
timestamp	String	Mandatory	Time when the response was generated	ISO 8601 datetime
data	Object	Mandatory	Organizer information associated with the user	JSON object
data.organizer_id	Integer	Mandatory	Unique identifier of the organizer	Numeric
data.company_name	String	Mandatory	Registered company name of the organizer	Text

Table 6.2.43: Web Services Response Parameter (consume) - getUserOrganizer

getUsersByIds

Field Name	Field Type	Mandatory / Optional	Description	Format
status	String	Mandatory	Status of the request	success / failed
timestamp	String	Mandatory	Time when the response was generated	ISO 8601 datetime
data	Array	Mandatory	List of user profiles	JSON array
data[].id	Integer	Mandatory	Unique identifier of the user	Numeric
data[].name	String	Mandatory	Full name of the user	Text
data[].email	String	Mandatory	Email address of the user	Valid email format
data[].gender	String	Optional	Gender of the user	Text
data[].tel_no	String	Optional	Contact telephone number	Numeric
data[].student_id	String	Optional	Student identification number	Alphanumeric

Table 6.2.44: Web Services Response Parameter (consume) - getUsersByIds

addOrDeductCreditScore

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	DateTime (ISO 8601)	Mandatory	Timestamp of the response returned by the Membership Module	ISO 8601 date-time string (YYYY-MM-DDTHH:MM:SSZ)
status	Boolean	Mandatory	Indicates whether the credit score update	true / false

			operation was successful	
message	String	Mandatory	Confirmation message after credit score update	String

Table 6.2.45: Web Services Response Parameter (consume) - addOrDeductCreditScore

Payment Module

getByFilter

Field Name	Field Type	Mandatory / Optional	Description	Format / Example
timestamp	DateTime (ISO 8601)	Mandatory	Timestamp of the response returned by the Payment Module. Used for request–response verification (IFA).	ISO 8601 date-time string (YYYY-MM-DDTHH:MM:SSZ)
status	String	Mandatory	Indicates the processing result of the request.	success / error
data	Array	Mandatory	List of payment records that match the filter criteria.	Array of payment objects
data[].id	Integer	Mandatory	Unique identifier of the payment record.	Positive integer
data[].organizer_id	Integer	Mandatory	Identifier of the organizer associated with the	Positive integer

			payment.	
data[].event_id	Integer	Mandatory	Identifier of the related event.	Positive integer
data[].type	String	Mandatory	Type of payment (e.g. deposit, rental, penalty).	deposit / rental / penalty
data[].status	String	Mandatory	Current payment status.	pending / paid / overdue / cancelled / refunded
data[].total_amt	Decimal	Mandatory	Total amount required for the payment.	Decimal number (2 decimal places)
data[].received_amt	Decimal	Mandatory	Amount successfully received.	Decimal number (2 decimal places)
data[].payment_method	String	Mandatory	Method used to make the payment.	cash / card / pos
data[].pay_at	DateTime (ISO 8601)	Optional	Date and time when the payment was made.	ISO 8601 date-time string (YYYY-MM-DDTHH:MM:SSZ)
data[].created_at	DateTime (ISO 8601)	Mandatory	Timestamp indicating when the payment record was created.	ISO 8601 date-time string (YYYY-MM-DDTHH:MM:SSZ)

Table 6.2.46: Web Services Response Parameter (consume) - getByFilter

getByEventId

Field Name	Field Type	Mandatory / Optional	Description	Format

timestamp	DateTime (ISO 8601)	Mandatory	Timestamp of the response returned by the Payment Module.	ISO 8601 date-time string (YYYY-MM-DDTHH:MM:SSZ)
status	String	Mandatory	Indicates the result of the request.	success / error
data	Array	Mandatory	List of payment records matching the filter criteria.	Array of Payment objects
data[].id	Integer	Mandatory	Unique identifier of the payment record.	Positive integer
data[].organizer_id	Integer	Mandatory	Identifier of the organizer related to the payment.	Positive integer
data[].event_id	Integer	Optional	Identifier of the related event.	Positive integer
data[].type	String	Mandatory	Type of payment.	deposit / penalty
data[].status	String	Mandatory	Current status of the payment.	pending / paid / overdue / cancelled / refunded
data[].discount_amt	Decimal	Optional	Discount amount applied to the payment.	Decimal (2 decimal places)
data[].total_amt	Decimal	Mandatory	Total amount required for the payment.	Decimal (2 decimal places)
data[].deposit_left_amt	Decimal	Optional	Remaining deposit	Decimal (2 decimal

			amount (if applicable).	places)
data[].received_amt	Decimal	Mandatory	Amount received from the payer.	Decimal (2 decimal places)
data[].change_amt	Decimal	Optional	Change returned to the payer.	Decimal (2 decimal places)
data[].payment_method	String	Mandatory	Payment method used.	cash / card / pos
data[].pay_at	DateTime (ISO 8601)	Optional	Date and time when the payment was made.	YYYY-MM-DD THH:MM:SSZ
data[].created_at	DateTime (ISO 8601)	Mandatory	Timestamp indicating when the payment record was created.	YYYY-MM-DD THH:MM:SSZ
data[].event	Object	Optional	Event details retrieved from the Event Module using the event ID.	Event object
data[].event.id	Integer	Mandatory	Unique identifier of the event.	Positive integer
data[].event.event_name	String	Mandatory	Name of the event.	String
data[].event.organizer	Object	Optional	Organizer information associated with the event.	Organizer object
data[].event.organizer.id	Integer	Mandatory	Organizer identifier.	Positive integer
data[].event.organizer.name	String	Mandatory	Organizer name.	String

Table 6.2.47: Web Services Response Parameter (consume) - getByEventId

getPaymentByEventId

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	DateTime (ISO 8601)	Mandatory	Timestamp of the response returned by the Payment Module (IFA timestamp).	ISO 8601 date-time string (YYYY-MM-DDTHH:MM:SSZ)
status	String	Mandatory	Indicates the result of the request.	success / error
data	Object	Mandatory	Deposit payment information associated with the specified event. Returns null if no deposit payment exists.	Payment object / null
data.id	Integer	Mandatory	Unique identifier of the payment record.	Positive integer
data.type	String	Mandatory	Type of payment for the event.	deposit
data.discount_amt	Decimal	Optional	Discount amount applied to the deposit payment.	Decimal (2 decimal places)
data.total_amt	Decimal	Mandatory	Total deposit amount required for the event.	Decimal (2 decimal places)

data.pay_at	DateTime (ISO 8601)	Optional	Date and time when the deposit payment was made.	YYYY-MM-DD THH:MM:SSZ
data.status	String	Mandatory	Current status of the deposit payment.	pending / paid / overdue / cancelled / refunded
data.payment_method	String	Mandatory	Payment method used for the deposit.	cash / card / pos
data.received_amt	Decimal	Mandatory	Amount received for the deposit payment.	Decimal (2 decimal places)
data.change_amt	Decimal	Optional	Change returned after payment, if applicable.	Decimal (2 decimal places)

Table 6.2.48: Web Services Response Parameter (consume) - getPaymentByEventId

isPaymentExistsByPaymentId

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	DateTime (ISO 8601)	Mandatory	Timestamp of the response returned by the Payment Module (IFA timestamp).	2025-03-21T10:25:30Z
status	String	Mandatory	Indicates the result of the request.	success
exists	Boolean	Mandatory	Indicates whether at least one payment record exists for the	true / false

			specified event.	

Table 6.2.49: Web Services Response Parameter (consume) - isPaymentExistsByPaymentId

apiUpdatePayment

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	DateTime (ISO 8601)	Mandatory	Timestamp indicating when the response was generated by the Payment Module (IFA timestamp).	2025-03-21T10:45:12Z
status	String	Mandatory	Indicates the outcome of the request processing.	success / error
message	String	Mandatory	Descriptive message indicating the result of the payment status update operation.	Payment status updated successfully

Table 6.2.50: Web Services Response Parameter (consume) - apiUpdatePayment

createEventDepositPayment

Field Name	Field Type	Mandatory / Optional	Description	Format
timestamp	DateTime (ISO 8601)	Mandatory	Timestamp indicating when the response was generated by the Payment Module (IFA timestamp).	YYYY-MM-DDTHH:MM:SSZ

status	String	Mandatory	Indicates the outcome of the request processing.	success / error
message	String	Mandatory	Descriptive message indicating the result of the deposit payment creation request.	Human-readable text
payment.id	Integer	Optional	Unique identifier of the created payment record. Returned only when the request is successful.	Positive integer
payment.event_id	Integer	Optional	Identifier of the event associated with the deposit payment.	Positive integer
payment.organizer_id	Integer	Optional	Identifier of the organizer associated with the deposit payment.	Positive integer
payment.type	String (Enum)	Optional	Type of payment created by the system.	deposit
payment.discount_amt	Decimal	Optional	Discount amount applied based on the organizer’s	Decimal (2 decimal places)

			membership privileges.	
payment.total_amt	Decimal	Optional	Final payable amount after discount deduction.	Decimal (2 decimal places)
payment.deposit_left_amt	Decimal	Optional	Remaining refundable deposit amount for the event.	Decimal (2 decimal places)
payment.status	String (Enum)	Optional	Current status of the payment record.	pending / paid / overdue / cancelled / refunded
payment.pay_at	DateTime	Optional	Timestamp indicating when the payment was completed. Null if unpaid.	YYYY-MM-DDTHH:MM:SSZ
payment.payment_method	String (Enum)	Optional	Method used to complete the payment transaction.	cash / card / pos
payment.received_amt	Decimal	Optional	Amount received from the organizer during payment.	Decimal (2 decimal places)
payment.change_amt	Decimal	Optional	Change returned to the organizer after payment.	Decimal (2 decimal places)
payment.created_at	DateTime	Optional	Timestamp indicating when the payment record was created.	YYYY-MM-DDTHH:MM:SSZ

payment.updated_at	DateTime	Optional	Timestamp indicating when the payment record was last updated.	YYYY-MM-DDTHH:MM:SSZ

Table 6.2.51: Web Services Response Parameter (consume) - createEventDepositPayment

3. Service Consumption

The Event Module also consumes the web services exposed by other system modules to support cross-module functionality and ensure the data consistency across the system. These services are accessed through RESTful JSON-based APIs which enables the Event Module to easily retrieve and update the external data without direct database dependency. All the consumed web services comply with the Interface Agreement (IFA) standards which includes the mandatory timestamp parameters and structured JSON responses. This can ensure a reliable communication and traceability between modules.

Example of Consumed Web Service

One key example of service consumption in the Event Module is the verification of deposit payment status before approving an event. The Event Module consumes a web service exposed by the Finance Module to determine whether the required deposit payment has been successfully completed.

Consumed Web Service Details

- Service Provider Module: Finance Module
- Consuming Module: Event Module
- API Route: /api/payments/{eventId}/depositPayment
- HTTP Method: GET
- Purpose: Retrieve the deposit payment status for a specific event

```
Route::get( uri: '/payments/{eventId}/depositPayment', [PaymentController::class, 'getPaymentByEventId']);
```

Figure 6.3.1: Finance Module API route for retrieving deposit payment information by event ID

```
$response = Http::withHeaders([
    'X-INTERNAL-KEY' => config( key: 'services.internal.key'),
    'X-INTERNAL-SECRET' => encrypt(config( key: 'services.internal.secret')),
])->get(
    url: config( key: 'services.finance.url') . "/api/payments/{$event->id}/depositPayment",
    [
        'timestamp' => $validated['timestamp'],
    ]
);
```

Figure 6.3.2: Event Module consuming the Finance Module web service to verify deposit payment status

7. Index

Figure No.	Title	View Path	Class Path
2.1.1	Admin Site Create Event Form	resources/views/livewire/admin/create-event.blade.php	App\Livewire\Admin\CreateEvent
2.1.2	Admin Site View Event Listing	resources/views/livewire/event-table.blade.php	App\Livewire\Admin\EventTable
2.1.3	Admin Site View Event Details - Section 1	resources/views/admin/events/show.blade.php	App\Http\Controllers\EventController@show
2.1.4	Admin Site View Event Details - Section 2	resources/views/admin/events/show.blade.php	App\Http\Controllers\EventController@show
2.1.5	Admin Site Edit Existing Events - Section 1	resources/views/livewire/admin/edit-event.blade.php	App\Livewire\Admin\EditEvent
2.1.6	Admin Site Edit Existing Events - Section 2	resources/views/livewire/admin/edit-event.blade.php	App\Livewire\Admin\EditEvent
2.1.7	Admin Site Approve Pending Event Confirmation Dialog	resources/views/livewire/event-table.blade.php	App\Livewire\Admin\EventTable
2.1.8	Admin Site Reject Pending Event Confirmation Dialog	resources/views/livewire/event-table.blade.php	App\Livewire\Admin\EventTable
2.1.9	Admin Site Cancel Approved Event Confirmation Dialog	resources/views/livewire/event-table.blade.php	App\Livewire\Admin\EventTable
2.2.1	Organizer Site Create Event Form - Section 1	resources/views/livewire/organizer/event-form.blade.php	App\Livewire\Organizer\EventForm
2.2.2	Organizer Site Create Event Form - Section 2	resources/views/livewire/organizer/event-form.blade.php	App\Livewire\Organizer\EventForm
2.2.3	Organizer Site Event Listing	resources/views/livewire/organizer/event-grid.blade.php	App\Livewire\Organizer\EventGrid

2.2.4	Organizer Site Event Editing Form - Section 1	resources/views/livewire/organizer/event-form.blade.php	App\Livewire\Organizer\EventForm
2.2.5	Organizer Site Event Editing Form - Section 2	resources/views/livewire/organizer/event-form.blade.php	App\Livewire\Organizer\EventForm
2.2.6	Organizer Site Event Details - Overview Tab	resources/views/organizer/events/tabs/overview.blade.php	App\Http\Controllers\EventController@organizerShowEvent
2.2.7	Organizer Site Event Details - Helpers Tab	resources/views/organizer/events/tabs/helpers.blade.php	App\Http\Controllers\EventController@organizerShowEvent
2.2.8	Organizer Site Event Details - Helper Role View Modal	resources/views/organizer/events/tabs/helpers.blade.php	App\Http\Controllers\EventController@organizerShowEvent
2.2.9	Organizer Site Event Details - Payments Tab	resources/views/organizer/events/tabs/payments.blade.php	App\Http\Controllers\EventController@organizerShowEvent
2.2.10	Organizer Site Helper Creation Form	resources/views/livewire/organizer/event-helper-role-form.blade.php	App\Livewire\Organizer\EventHelperRoleForm
2.2.11	Organizer Site Review Application Modal	resources/views/organizer/events/show.blade.php	App\Http\Controllers\EventController@organizerShowEvent
2.2.12	Organizer Site Reject Application Form	resources/views/organizer/events/show.blade.php	App\Http\Controllers\EventController@organizerShowEvent
2.3.1	Member Site Event Listing	resources/views/livewire/member/event-list.blade.php	App\Livewire\Member\EventList
2.3.2	Member Site Event Detail - Overview Tab	resources/views/member/events/partials/overview.blade.php	App\Http\Controllers\EventController@memberOrGuestShowEvent
2.3.3	Member Site Event Detail - Recruitment Tab	resources/views/member/events/partials/recruitment.blade.php	App\Http\Controllers\EventController@memberOrGuestShowEvent
2.3.4	Member Site Apply Helper Form	resources/views/livewire/organizer/event-helper-role-form.blade.php	App\Livewire\Organizer\EventHelperRoleForm
2.3.5	Member Site Helper Application History List	resources/views/livewire/member/helper-application-list.blade.php	App\Livewire\Member\HelperApplicationList

2.3.6	Member Site Helper Application Detail	resources/views/member/events/helper-applications/show.blade.php	App\Http\Controllers\EventController@memberHelperApplicationShow
3.1	Entity Class Diagram of Event Module	-	-
4.2	Class Diagram of Decorator Design Pattern	-	-
5.2.1	Route protected with auth:sanctum	-	-
5.2.2	getApiToken() method	-	App\Helpers\SanctumHelper@getApiToken
5.2.3	Calling getApiToken() before API request	-	App\Livewire\Member\HelperApplicationList@fetchApplications
5.2.4	Http::withToken() API call	-	App\Http\Controllers\EventController@getMemberHelperApplications
5.2.5	Validation rules enforcing allowed file types, file size limits and upload quantity for event file uploads	-	App\Livewire\Admin\CreateEvent@rules
5.2.6	System generated file paths using unique identifiers	-	App\Livewire\Admin\CreateEvent
5.2.7	Streaming of uploaded files to external storage	-	App\Livewire\Admin\CreateEvent@uploadToSupabase
5.2.8	Backend validation of uploaded file metadata before event creation	-	App\Http\Requests\Event\StoreEventRequest@rules
5.2.9	Secure upload files to external storage using environment-based credentials	-	App\Livewire\Admin\CreateEvent@uploadToSupabase
6.1.1	REST API endpoint exposed by the Event Module for updating event status	-	-

6.1.2	Example of Finance Module consuming the Event Module's web service to update event status	-	App\Services\PaymentService\PaymentService@rejectEvent
6.3.1	Finance Module API route for retrieving deposit payment information by event ID	-	-
6.3.2	Event Module consuming the Finance Module web service to verify deposit payment status	-	App\Http\Controllers\EventController@updateStatus

8. References

GeeksforGeeks. (2025, September 25). Decorator design pattern. GeeksforGeeks. https://www.geeksforgeeks.org/system-design/decorator-pattern

OWASP Foundation. (2021). A01:2021 – Broken access control. The OWASP Foundation. https://owasp.org/Top10/2021/A01_2021-Broken_Access_Control/index.html

OWASP Foundation. (n.d.). File upload cheat sheet. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

OWASP Foundation. (n.d.). Unrestricted file upload. The OWASP Foundation. https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload