Table of Contents

1. Introduction to the System

The **TARUMT Society and Event Management System** is a **web-based platform** built to simplify and centralize the management of student societies and campus events at TARUMT. It provides students with an easy way to **explore societies, join activities, and participate in events**, including both **free and paid** events. At the same time, society leaders (President and Committee) are supported with tools to manage society operations such as **membership handling, announcements, event organization, and facility booking processes**, ensuring that events are properly arranged and coordinated across different societies.

To strengthen engagement and communication, the system includes interactive features that allow society leaders to publish updates while members can respond and participate through comments. For paid events, the platform supports **secure ticketing and payment handling**, and it also provides **event notifications and reminders** to keep participants informed. Through integration of core modules such as **Event Management, Society Management, User Management, Ticketing & Payment, and Facility Management**, the system ensures a smooth end-to-end workflow covering event creation, approval processes, facility allocation, and user participation while helping the university prevent scheduling conflicts and improve campus event operations overall.

2. Module Description

The Facility Management Module handles the full workflow of reserving campus facilities for society activities and events. It provides a centralized booking system where society leaders can request facilities, the administration can review and approve/reject requests, and the system automatically helps prevent timetable clashes by tracking booking dates and times. This ensures every event has a confirmed venue arrangement while improving coordination across multiple societies.

2.1 Facility Records (Admin)

View Facilities List

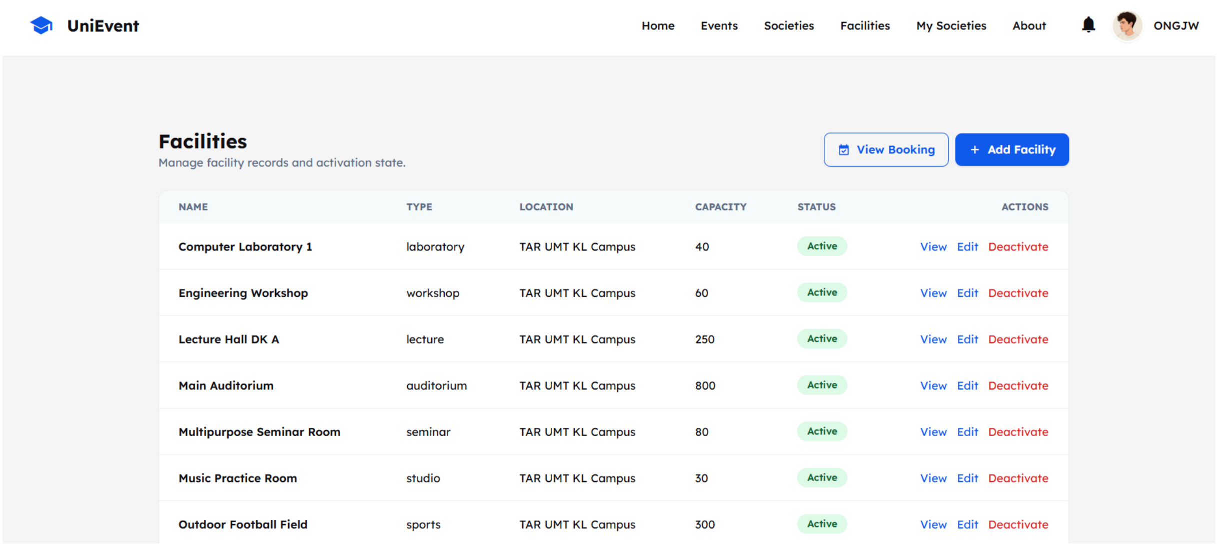
This function displays a complete list of facility groups in the system, including their **name, type, location, capacity, and active status**. The active status indicates whether a facility is currently available for booking.

Administrators can perform the following actions from this page:

- **View** the facility timetable and venue availability
- **Edit** facility details
- **Deactivate** a facility to prevent future bookings

Source Path: resources\views\admin\facilities\index.blade.php

URL: http://127.0.0.1:8000/admin/facilities



Facilities

Manage facility records and activation state.

NAME	TYPE	LOCATION	CAPACITY	STATUS	ACTIONS
Computer Laboratory 1	laboratory	TAR UMT KL Campus	40	Active	View Edit Deactivate
Engineering Workshop	workshop	TAR UMT KL Campus	60	Active	View Edit Deactivate
Lecture Hall DK A	lecture	TAR UMT KL Campus	250	Active	View Edit Deactivate
Main Auditorium	auditorium	TAR UMT KL Campus	800	Active	View Edit Deactivate
Multipurpose Seminar Room	seminar	TAR UMT KL Campus	80	Active	View Edit Deactivate
Music Practice Room	studio	TAR UMT KL Campus	30	Active	View Edit Deactivate
Outdoor Football Field	sports	TAR UMT KL Campus	300	Active	View Edit Deactivate

Figure 2.1.1

Add Facility

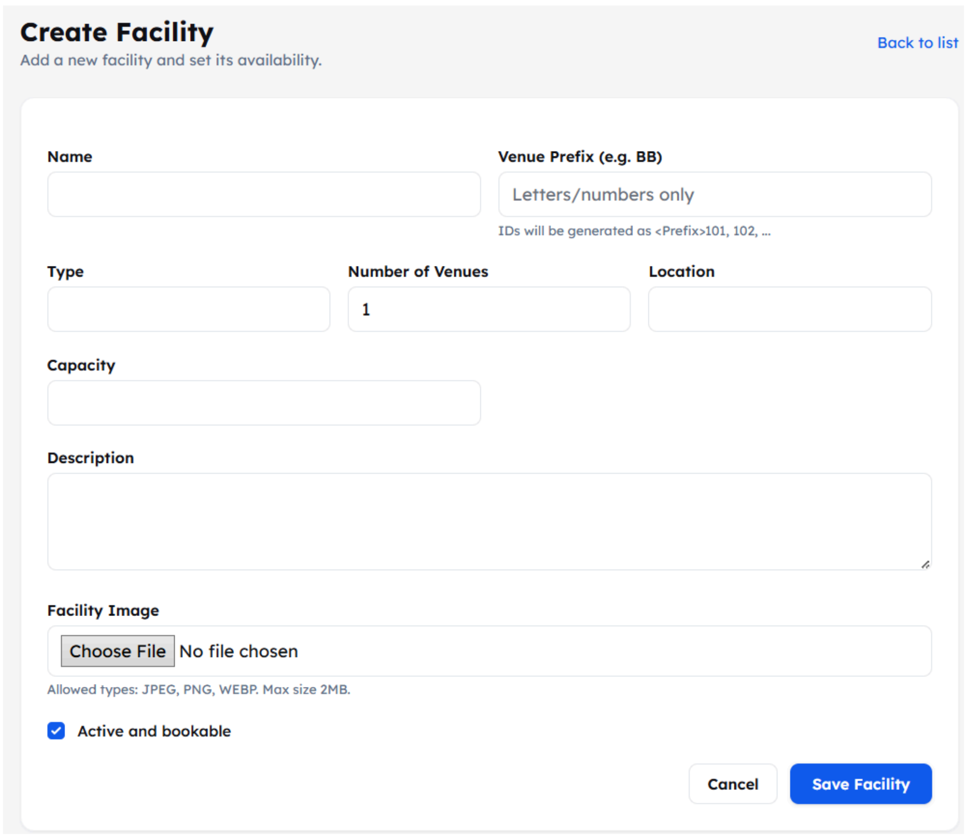
This function allows administrators to register a new facility group into the system. A facility group represents a category of venues (e.g., Computer Lab, Lecture Hall) and can automatically generate multiple venues using a **venue prefix and running number** (e.g., C401, C402).

During creation, the system performs input validation to ensure:

- Required fields (name, type, location, capacity) are provided
- The venue prefix follows valid character rules
- The number of venues is a positive integer
- Uploaded facility images are valid image files

Source Path: resources\views\admin\facilities\create.blade.php

URL: http://127.0.0.1:8000/admin/facilities/create



Create Facility

Add a new facility and set its availability.

Name:

Venue Prefix (e.g. BB): Letters/numbers only

IDs will be generated as <Prefix>101, 102, ...

Type:

Number of Venues: 1

Location:

Capacity:

Description:

Facility Image: No file chosen

Allowed types: JPEG, PNG, WEBP. Max size 2MB.

☑ Active and bookable

Figure 2.1.2

Edit Facility

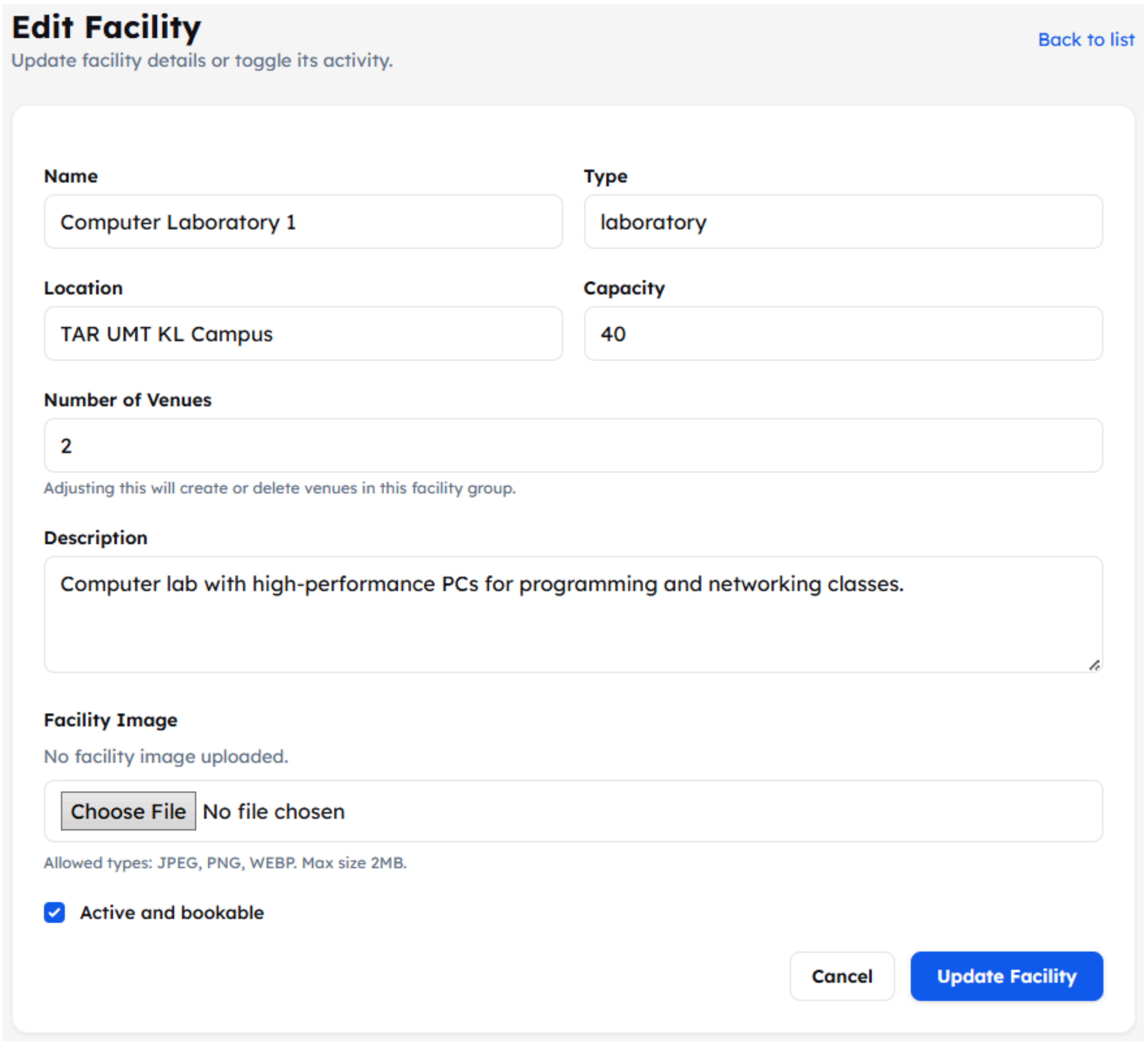
This function allows administrators to update existing facility details, including **name, type, location, capacity, description, image, and active status**.

If the **number of venues** is modified:

- Increasing the number will automatically generate new venues
- Decreasing the number will soft-delete excess venues to preserve booking history

Source Path: resources\views\admin\facilities\edit.blade.php

URL: http://127.0.0.1:8000/admin/facilities/4/edit



Edit Facility

Update facility details or toggle its activity.

Name	Computer Laboratory 1
Type	laboratory
Location	TAR UMT KL Campus
Capacity	40
Number of Venues	2

Adjusting this will create or delete venues in this facility group.

Description

Computer lab with high-performance PCs for programming and networking classes.

Facility Image

No facility image uploaded.

Choose File No file chosen

Allowed types: JPEG, PNG, WEBP. Max size 2MB.

[✓] Active and bookable

Figure 2.1.3

Deactivate Facility

This function allows administrators to deactivate a facility or facility group without permanently deleting it. Deactivated facilities:

- Remain stored in the system for reference and reporting
- Are excluded from availability checks and booking operations

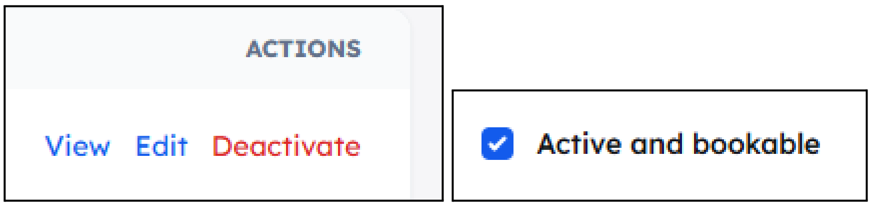


Figure 2.1.4

2.2 Facility Timetable / Availability View

View Facility Timetable

This function displays a **daily timetable from 08:00 to 22:00**, divided into **30-minute time slots**, for all venues under a selected facility group.

Each time slot is clearly labeled using different statuses:

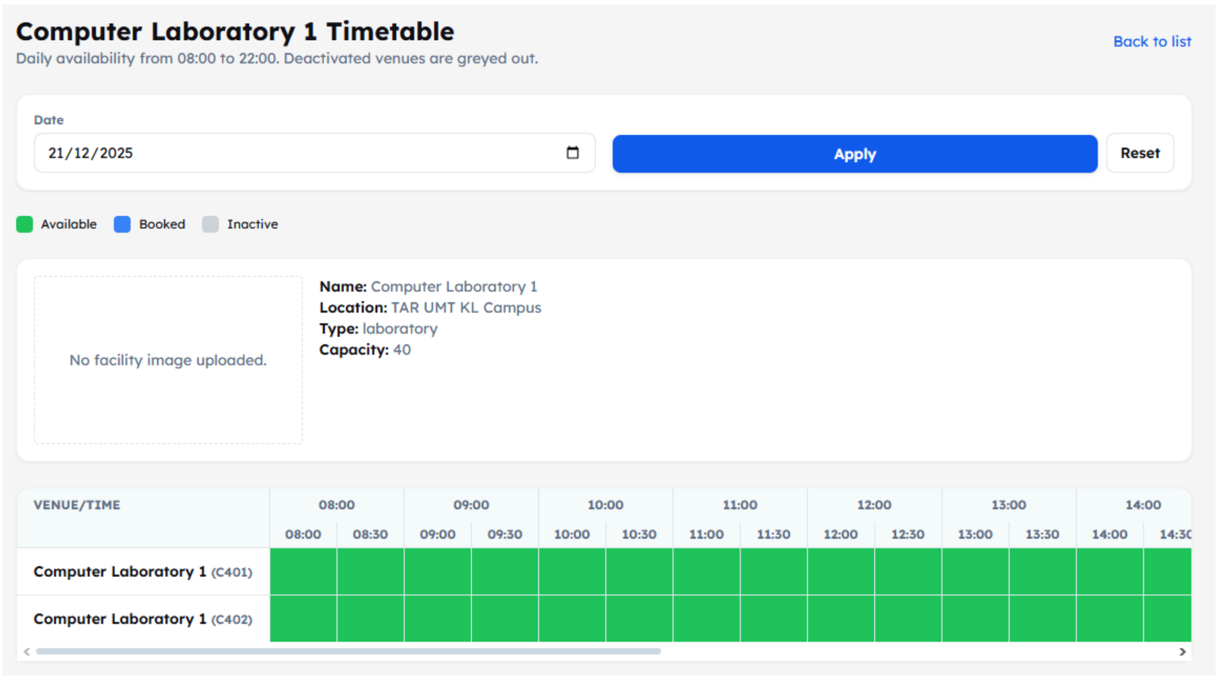
- **Available** – The venue is free and can be booked
- **Booked** – The venue is reserved by an existing booking with blocking status (PENDING or APPROVED)
- **Inactive** – The venue or facility is not active and cannot be booked

Bookings with blocking statuses are treated as occupied to prevent overlapping reservations. This visual approach allows administrators and users to identify conflicts easily before creating or approving bookings.

The timetable view improves transparency and supports informed decision-making when managing facility bookings.

Source Path: resources\views\admin\facilities\show.blade.php

URL: http://127.0.0.1:8000/admin/facilities/4



Computer Laboratory 1 Timetable

Daily availability from 08:00 to 22:00. Deactivated venues are greyed out.

Back to list

Date: 21/12/2025

Apply Reset

🟩 Available 🟦 Booked ⬜ Inactive

No facility image uploaded.

Name: Computer Laboratory 1
Location: TAR UMT KL Campus
Type: laboratory
Capacity: 40

VENUE/TIME	08:00		09:00		10:00		11:00		12:00		13:00		14:00	
	08:00	08:30	09:00	09:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:3
Computer Laboratory 1 (C401)	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Computer Laboratory 1 (C402)	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Figure 2.2.1

2.3 Facility Booking Management

View Bookings List + Filter by Date

This function displays a list of facility bookings that overlap with a selected date. Each record shows:

- Facility and venue
- Booking start and end time

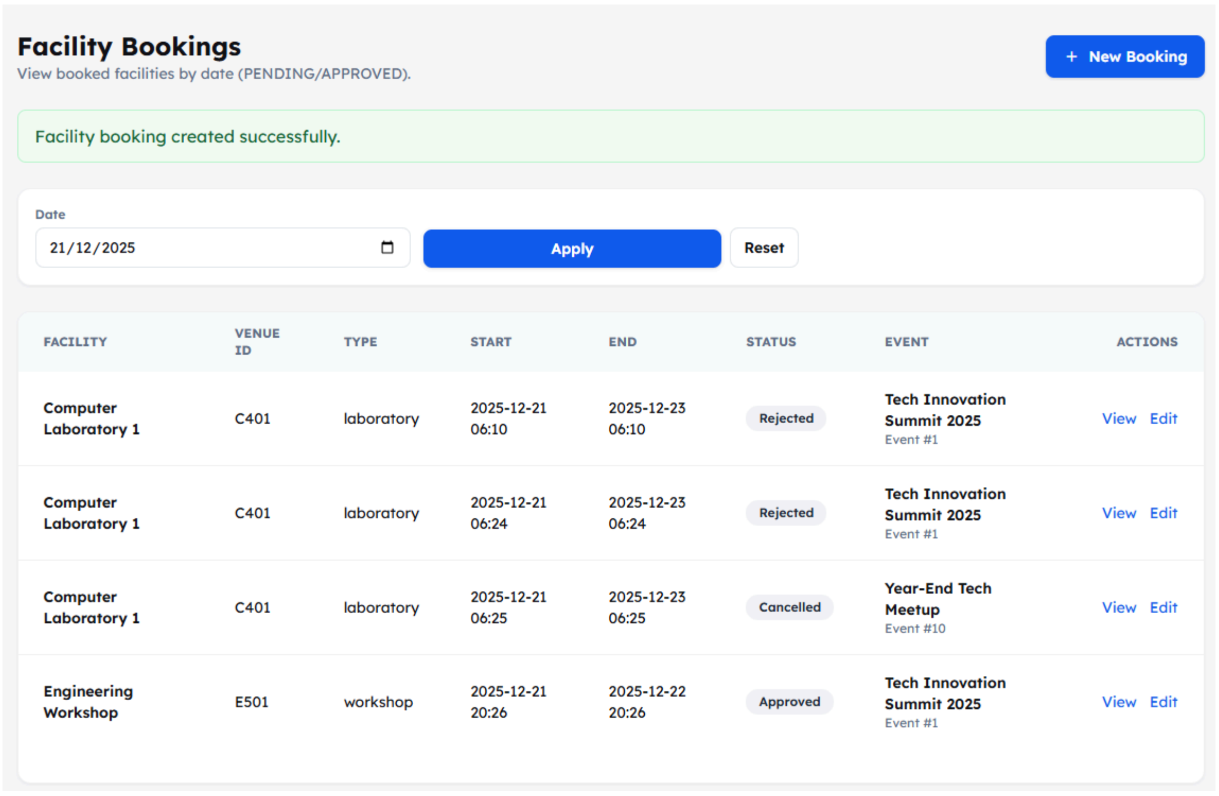
- Booking status
- Associated event

Administrators can perform actions such as **viewing details, editing bookings, or creating new manual bookings**. Non-administrative users are typically restricted to view-only access.

Filtering by date helps administrators focus on relevant bookings and manage daily operations more efficiently.

Source Path: resources\views\admin\bookings\index.blade.php

URL: http://127.0.0.1:8000/admin/bookings



Facility Bookings

View booked facilities by date (PENDING/APPROVED).

Facility booking created successfully.

Date: 21/12/2025

FACILITY	VENUE ID	TYPE	START	END	STATUS	EVENT	ACTIONS
Computer Laboratory 1	C401	laboratory	2025-12-21 06:10	2025-12-23 06:10	Rejected	Tech Innovation Summit 2025 Event #1	View Edit
Computer Laboratory 1	C401	laboratory	2025-12-21 06:24	2025-12-23 06:24	Rejected	Tech Innovation Summit 2025 Event #1	View Edit
Computer Laboratory 1	C401	laboratory	2025-12-21 06:25	2025-12-23 06:25	Cancelled	Year-End Tech Meetup Event #10	View Edit
Engineering Workshop	E501	workshop	2025-12-21 20:26	2025-12-22 20:26	Approved	Tech Innovation Summit 2025 Event #1	View Edit

Figure 2.3.1

View Booking Details

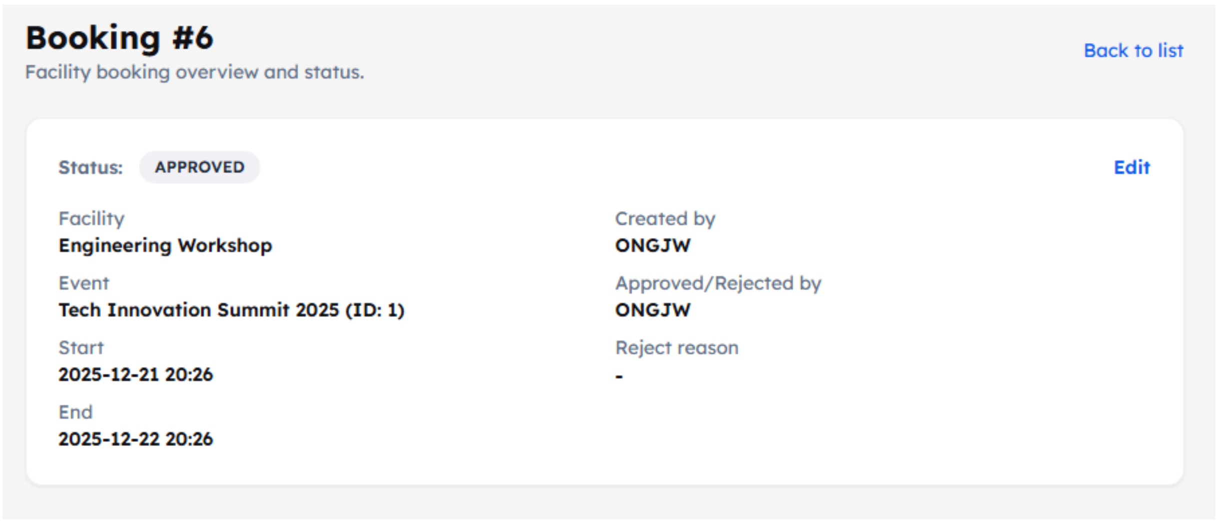
This function displays complete information for a selected booking, including:

- Booking status
- Facility and venue
- Linked event
- Time range
- Booking creator
- Approver (if approved)
- Rejection reason (if rejected)

It is mainly used for **audit, verification, and decision-making** before approving or rejecting a booking.

Source Path: resources\views\admin\bookings\show.blade.php

URL: http://127.0.0.1:8000/admin/bookings/6



Booking #6
Facility booking overview and status.

Back to list

Status: APPROVED

Edit

Facility	Created by
Engineering Workshop	**ONGJW**
Event	Approved/Rejected by
Tech Innovation Summit 2025 (ID: 1)	**ONGJW**
Start	Reject reason
2025-12-21 20:26	**-**
End	
2025-12-22 20:26	

Figure 2.3.2

Create Manual Booking (Admin) + Dynamic Venue Loading

This function allows administrators to manually create a facility booking, typically used when:

- An event booking requires administrative intervention
- A booking needs to be created or adjusted outside the standard workflow

The admin selects:

1. An event
2. A facility group
3. An available venue (loaded dynamically via API based on the selected facility)

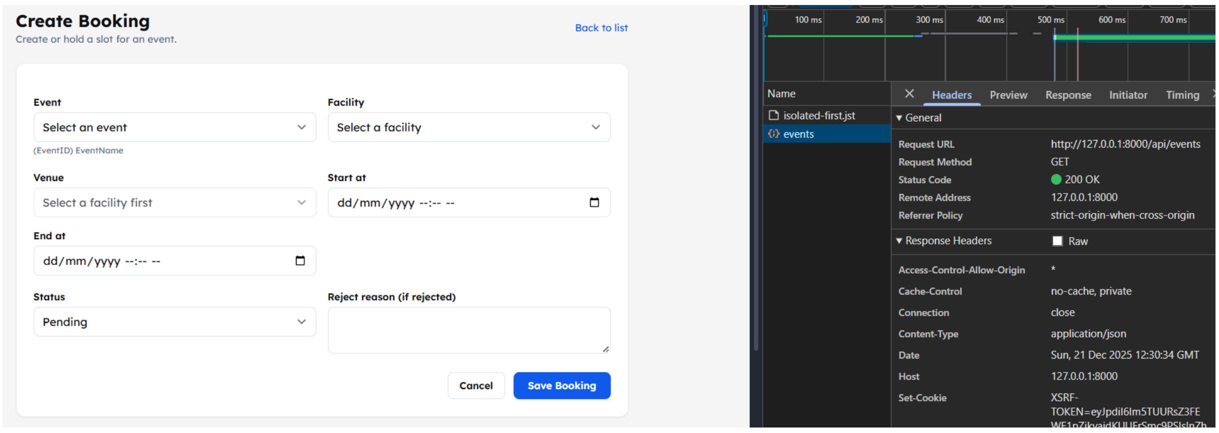
When saving the booking, the system validates that:

- All required fields are provided
- The selected venue belongs to the chosen facility
- The booking time does not overlap existing blocking bookings (PENDING or APPROVED)

This ensures only valid and conflict-free bookings are created.

Source Path: resources\views\admin\bookings\create.blade.php

URL: http://127.0.0.1:8000/admin/bookings/create



Create Booking
Create or hold a slot for an event.

Back to list

Event	Facility
Select an event	Select a facility
(EventID) EventName	
Venue	Start at
Select a facility first	dd/mm/yyyy --:-- --
End at	
dd/mm/yyyy --:-- --	
Status	Reject reason (if rejected)
Pending	

Cancel Save Booking

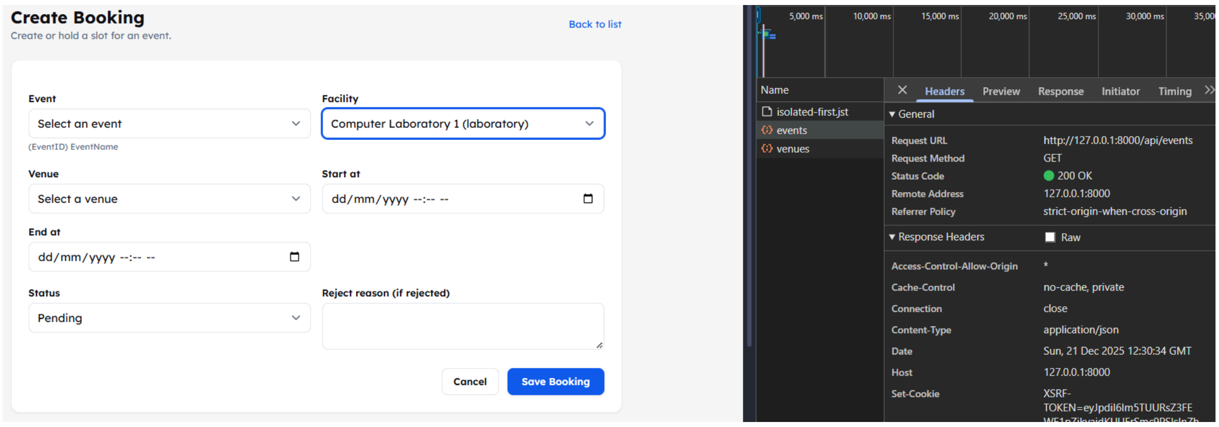
▾ General

Request URL	http://127.0.0.1:8000/api/events
Request Method	GET
Status Code	● 200 OK
Remote Address	127.0.0.1:8000
Referrer Policy	strict-origin-when-cross-origin

▾ Response Headers ☐ Raw

Access-Control-Allow-Origin	*
Cache-Control	no-cache, private
Connection	close
Content-Type	application/json
Date	Sun, 21 Dec 2025 12:30:34 GMT
Host	127.0.0.1:8000
Set-Cookie	XSRF-TOKEN=eyJpdiI6Im5TUURsZ3FE

Figure 2.3.3


Create Booking
Create or hold a slot for an event.
Back to list

Event	Select an event
	(EventID) EventName
Facility	Computer Laboratory 1 (laboratory)
Venue	Select a venue
Start at	dd/mm/yyyy --:-- --
End at	dd/mm/yyyy --:-- --
Status	Pending
Reject reason (if rejected)	

Cancel · Save Booking

Name: isolated-first.jst, events, venues

Headers · Preview · Response · Initiator · Timing

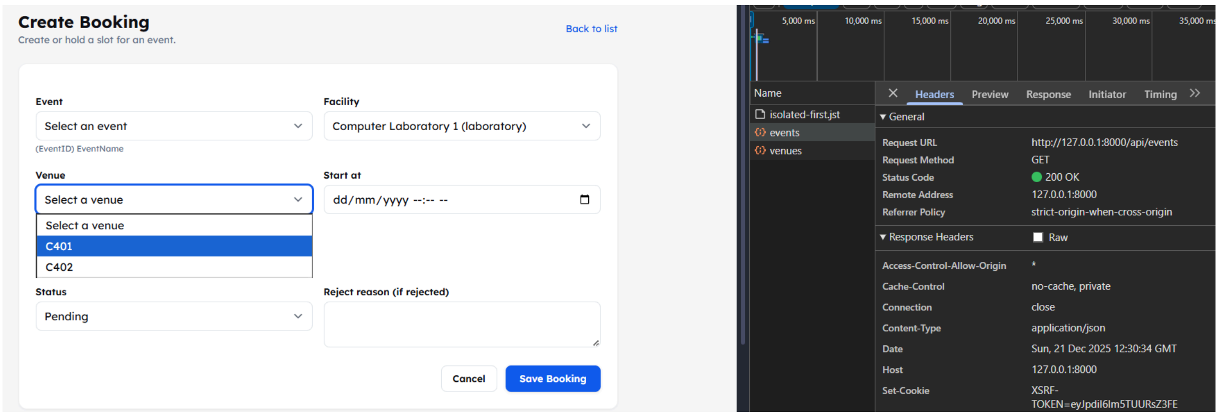
▾ General

Request URL	http://127.0.0.1:8000/api/events
Request Method	GET
Status Code	● 200 OK
Remote Address	127.0.0.1:8000
Referrer Policy	strict-origin-when-cross-origin

▾ Response Headers ☐ Raw

Access-Control-Allow-Origin	*
Cache-Control	no-cache, private
Connection	close
Content-Type	application/json
Date	Sun, 21 Dec 2025 12:30:34 GMT
Host	127.0.0.1:8000
Set-Cookie	XSRF-TOKEN=eyJpdil6Im5TUURsZ3FE

Figure 2.3.4


Create Booking
Create or hold a slot for an event.
Back to list

Event	Select an event
	(EventID) EventName
Facility	Computer Laboratory 1 (laboratory)
Venue	Select a venue
	Select a venue / C401 / C402
Start at	dd/mm/yyyy --:-- --
Status	Pending
Reject reason (if rejected)	

Cancel · Save Booking

Name: isolated-first.jst, events, venues

Headers · Preview · Response · Initiator · Timing

▾ General

Request URL	http://127.0.0.1:8000/api/events
Request Method	GET
Status Code	● 200 OK
Remote Address	127.0.0.1:8000
Referrer Policy	strict-origin-when-cross-origin

▾ Response Headers ☐ Raw

Access-Control-Allow-Origin	*
Cache-Control	no-cache, private
Connection	close
Content-Type	application/json
Date	Sun, 21 Dec 2025 12:30:34 GMT
Host	127.0.0.1:8000
Set-Cookie	XSRF-TOKEN=eyJpdil6Im5TUURsZ3FE

Figure 2.3.5

Edit Booking

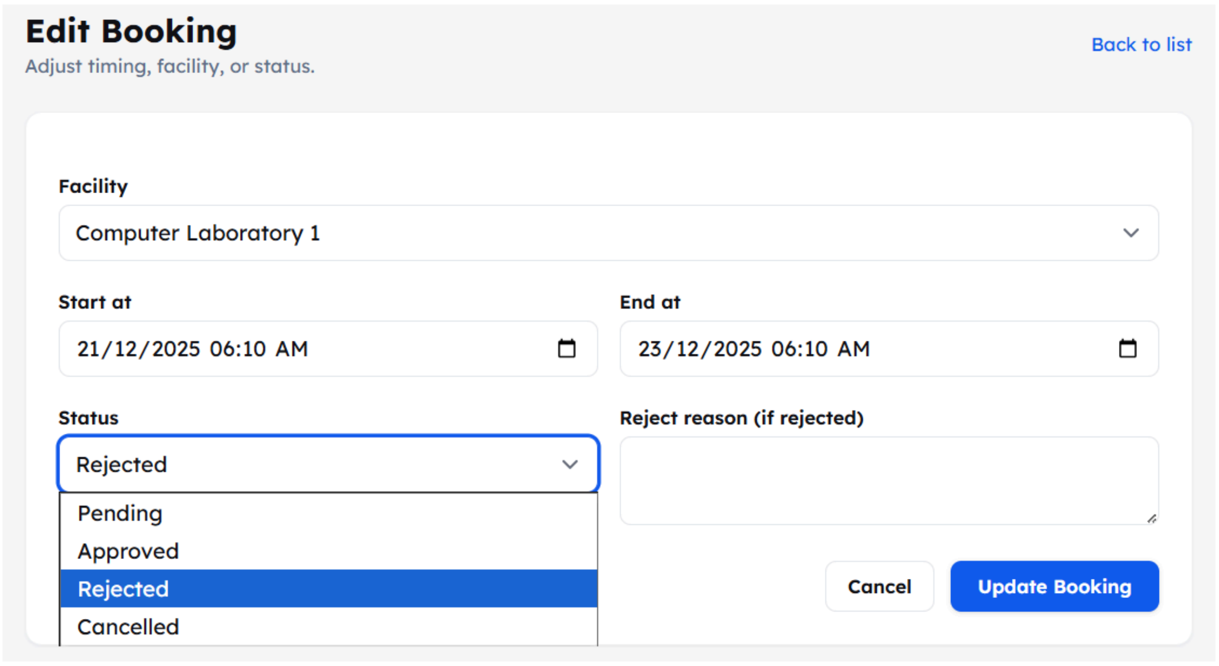
This function allows administrators to modify existing bookings, including:

- Venue
- Start and end time
- Booking status
- Rejection reason

If the updated booking status is **PENDING or APPROVED**, the system rechecks availability to prevent time overlaps. If the status is changed to a non-rejected state, any previous rejection reason is automatically cleared to maintain data consistency.

Source Path: resources\views\admin\bookings\edit.blade.php

URL: http://127.0.0.1:8000/admin/bookings/4/edit


Edit Booking

Adjust timing, facility, or status.

Back to list

Facility: Computer Laboratory 1

Start at: 21/12/2025 06:10 AM

End at: 23/12/2025 06:10 AM

Status: Rejected

Reject reason (if rejected):

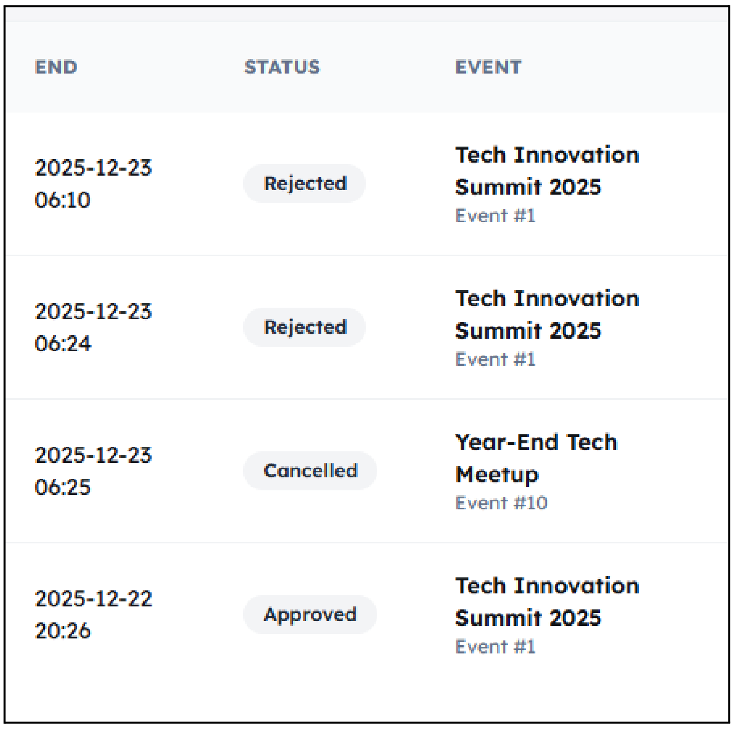
Figure 2.3.6

Approve/Reject Booking

This function finalizes the booking decision:

- **Approve** – The booking is confirmed only if the time slot is still available, and the approver is recorded
- **Reject** – The booking is marked as rejected, and a rejection reason is stored

This step ensures that facilities are reserved only after proper administrative review.



END	STATUS	EVENT
2025-12-23 06:10	Rejected	**Tech Innovation Summit 2025** Event #1
2025-12-23 06:24	Rejected	**Tech Innovation Summit 2025** Event #1
2025-12-23 06:25	Cancelled	**Year-End Tech Meetup** Event #10
2025-12-22 20:26	Approved	**Tech Innovation Summit 2025** Event #1

Figure 2.3.7

2.4 Availability Check + Event Integration

Real-time Availability Check (Admin Event Create)

When an administrator creates or edits an event and selects a facility with a proposed time range, the system automatically triggers an API call to check availability.

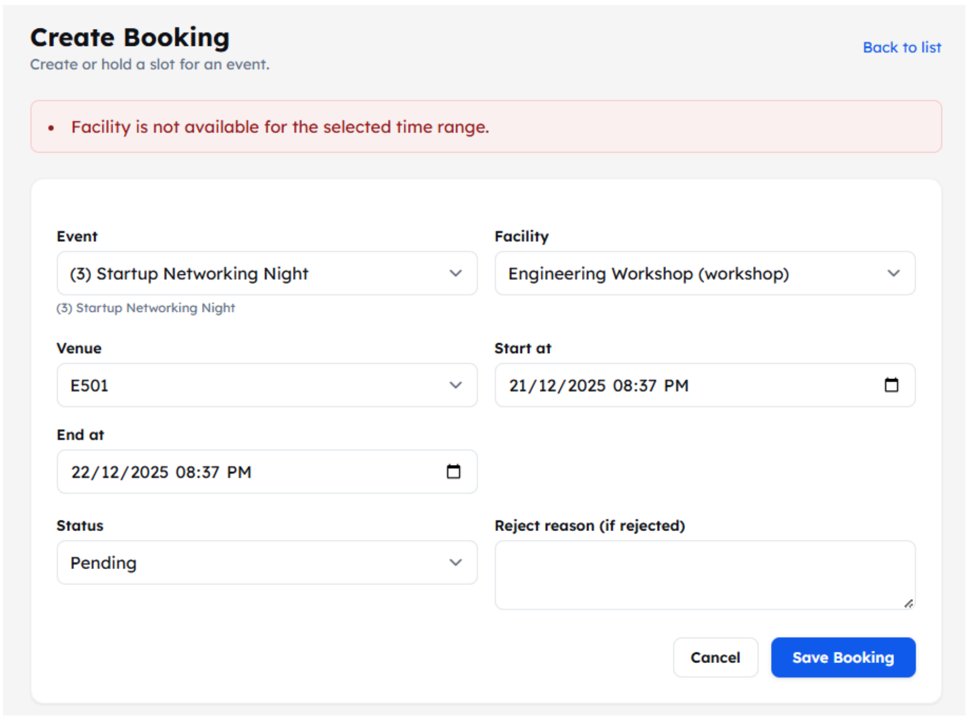
The availability service:

- Detects overlapping bookings
- Returns availability status and conflict details (if any)
- Allows the user to adjust time or facility before submitting the event

This real-time validation prevents scheduling conflicts at the **event creation stage**, ensuring that only events with valid facility arrangements proceed for approval.

Source Path: resources\views\admin\events\create.blade.php

URL: http://127.0.0.1:8000/admin/bookings/4/edit


Create Booking

Create or hold a slot for an event.

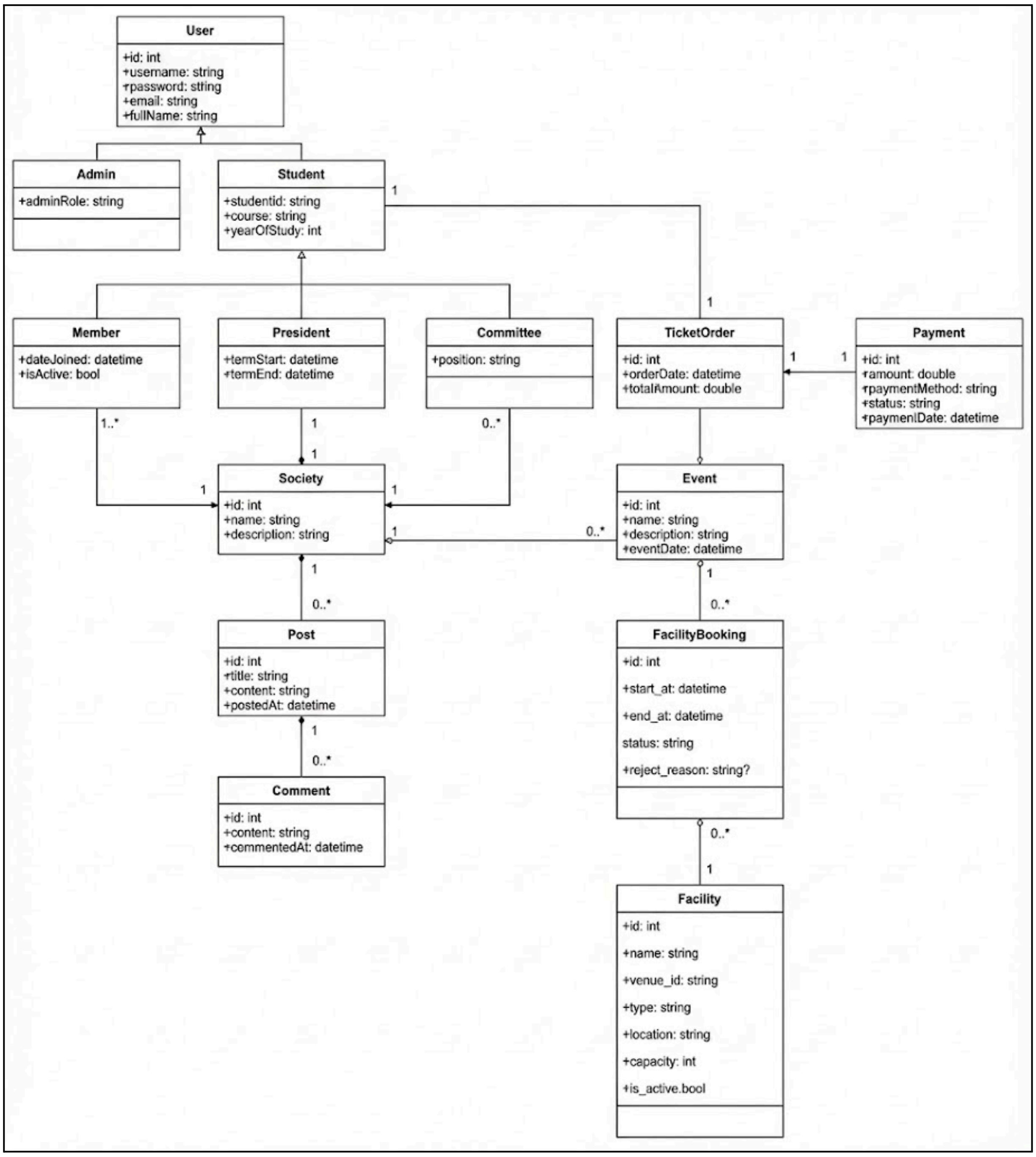
Back to list

- Facility is not available for the selected time range.

Event: (3) Startup Networking Night	Facility: Engineering Workshop (workshop)
(3) Startup Networking Night	
Venue: E501	Start at: 21/12/2025 08:37 PM
End at: 22/12/2025 08:37 PM	
Status: Pending	Reject reason (if rejected):

Figure 2.4.1

3. Entity Classes



4. Design Pattern

4.1 Description of Design Pattern

Proxy Pattern

The **Proxy Pattern** is a structural design pattern that introduces a surrogate object, known as a proxy, to control access to another object called the real subject. The proxy exposes the same interface as the real subject, allowing the client to interact with it transparently while the proxy performs additional processing before or after delegating the request. Common responsibilities of a proxy include access control, validation, logging, or request filtering, without altering the core functionality of the real object.

In the Facility Management Module, the Proxy Pattern is implemented as a **Protection Proxy** to enforce role-based access control. Controllers do not directly invoke the business logic services; instead, they communicate through proxy classes that act as gatekeepers. These proxies verify user authorization before allowing operations such as creating, updating, or cancelling facilities and bookings. Only when the access rules are satisfied will the proxy forward the request to the underlying service that handles the actual facility and booking logic. This design centralizes authorization logic, keeps business services focused on core rules such as availability checking and conflict prevention, and maintains a clean separation of concerns within the MVC architecture.

4.2 Implementation of Design Pattern

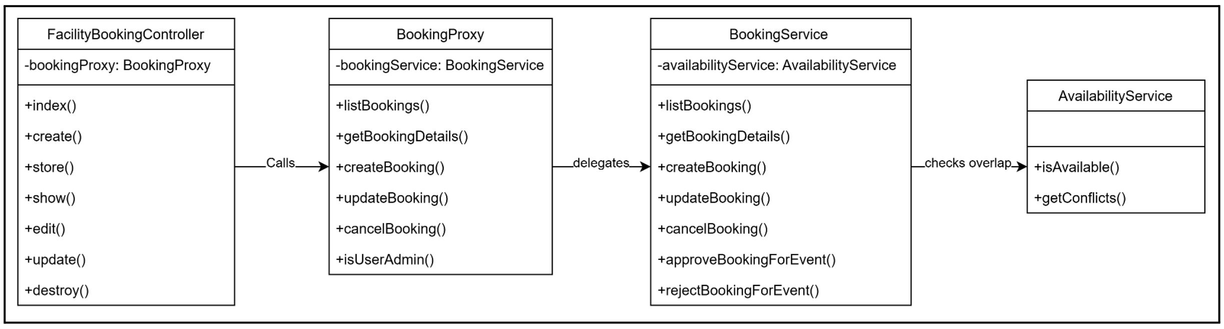


Figure 4.1.1: Proxy Pattern diagram

The Proxy Pattern was chosen for the Facility Management Module because enforcing access control is a core requirement of the module and must be applied consistently across multiple facility and booking operations. Different users have different permissions, where administrators are responsible for managing facilities and facility bookings, while students regardless of whether they hold society positions such as president or committee must be restricted from performing administrative actions. By implementing a Protection Proxy, authorization checks are centralized in a single gatekeeping layer rather than being repeatedly coded across many controller actions. This results in cleaner controllers, reduces the risk of missed permission checks, and keeps the underlying service layer focused on business rules such as timetable availability, overlap detection, and booking status updates. In addition, the proxy approach improves maintainability and extensibility because future permission changes or additional security features (e.g., audit logging or stricter restrictions) can be introduced within the proxy without affecting controllers or core booking logic.

4.3 Implementation of MVC Architecture

The Facility Management Module is developed using the **Model–View–Controller (MVC)** architectural pattern provided by the Laravel framework. MVC separates the system into three interconnected components—Model, View, and Controller—each with a distinct responsibility. This separation improves **maintainability, scalability, and code readability**, while supporting secure and structured request handling.

4.3.1 Model

The **Model** represents the system's data and business rules. It is responsible for interacting with the database, managing relationships between entities, and enforcing data integrity.
In this module, Eloquent ORM models are used to represent core entities such as facilities, venues, and bookings. The models handle:

- Database table mapping
- Relationships (e.g., facility to venues, bookings to events)
- Soft deletion to preserve historical data
- Query logic for availability and booking overlap checks

By encapsulating data logic inside models, the system ensures that database operations remain consistent and reusable across different controllers.

Model Path: app\Models\Facility.php

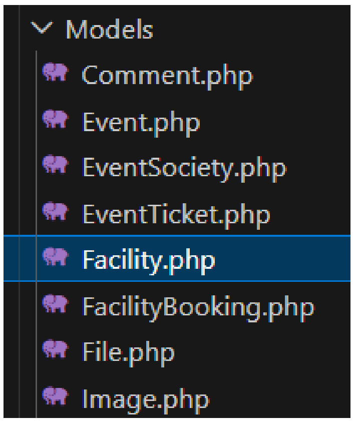


Figure 4.3.1.1

4.3.2 View

The **View** layer is responsible for presenting data to users and collecting user input. Views are implemented using **Blade templates**, which dynamically render content based on data provided by controllers.
In the Facility Management Module, views are used to:

- Display facility lists and details
- Render facility timetables and booking status
- Provide forms for creating and editing facilities and bookings

- Show validation errors and system feedback messages

The view layer does not contain business logic; it focuses solely on user interaction and presentation, ensuring a clean separation of concerns.

View Path: resources\views\admin\facilities\

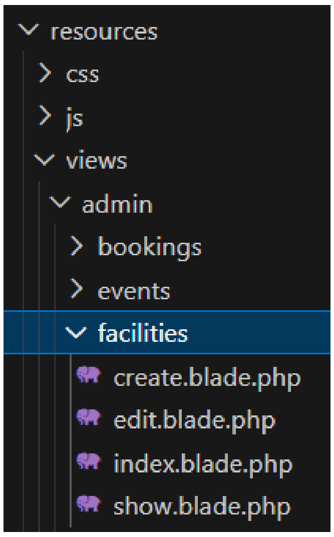


Figure 4.3.2.1

4.3.3 Controller

The **Controller** acts as the intermediary between the Model and View. It processes HTTP requests, validates user input, coordinates business logic, and returns appropriate views or JSON responses.

In this module, controllers:

- Handle CRUD operations for facilities and bookings
- Enforce access control using middleware, policies, and proxy classes
- Invoke availability checking logic before creating or approving bookings
- Pass validated data to views or API responses

Controllers do not directly contain complex business logic. Instead, they delegate responsibilities to models, services, or protection proxies, ensuring that controllers remain lightweight and focused on request flow.

Controller Path:

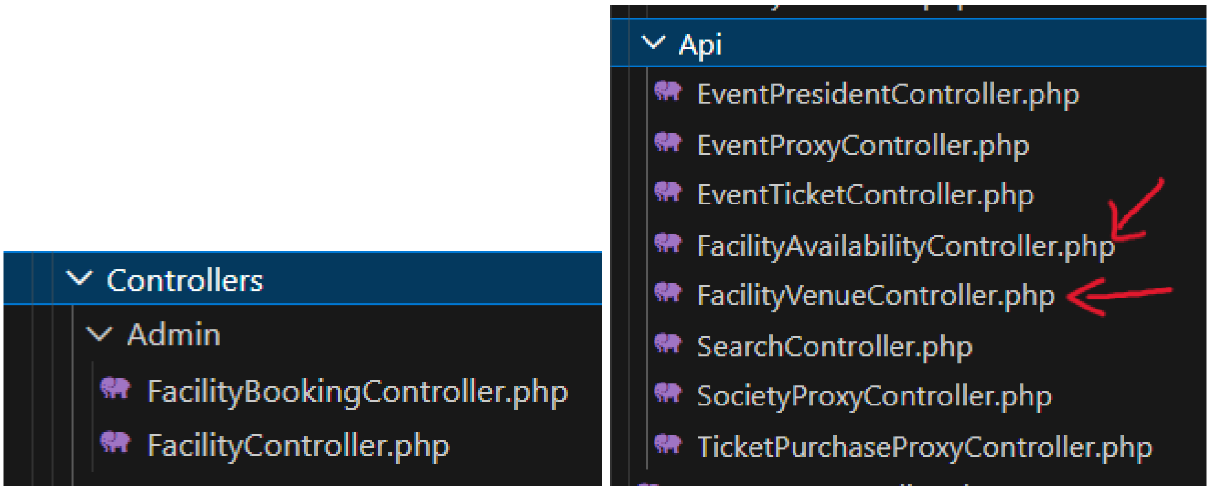
app\Http\Controllers\Admin\FacilityController.php

app\Http\Controllers\Admin\FacilityBookingController.php

API Path:

app\Http\Controllers\Api\FacilityAvailabilityController.php

app\Http\Controllers\Api\FacilityVenueController.php


```
Controllers
  Admin
    FacilityBookingController.php
    FacilityController.php
```

```
Api
  EventPresidentController.php
  EventProxyController.php
  EventTicketController.php
  FacilityAvailabilityController.php
  FacilityVenueController.php
  SearchController.php
  SocietyProxyController.php
  TicketPurchaseProxyController.php
```

Figure 4.3.3.1

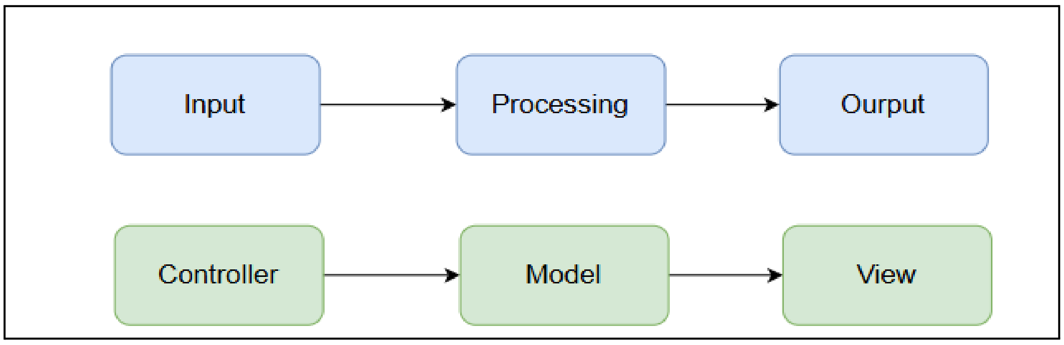


Figure 4.3.3.2

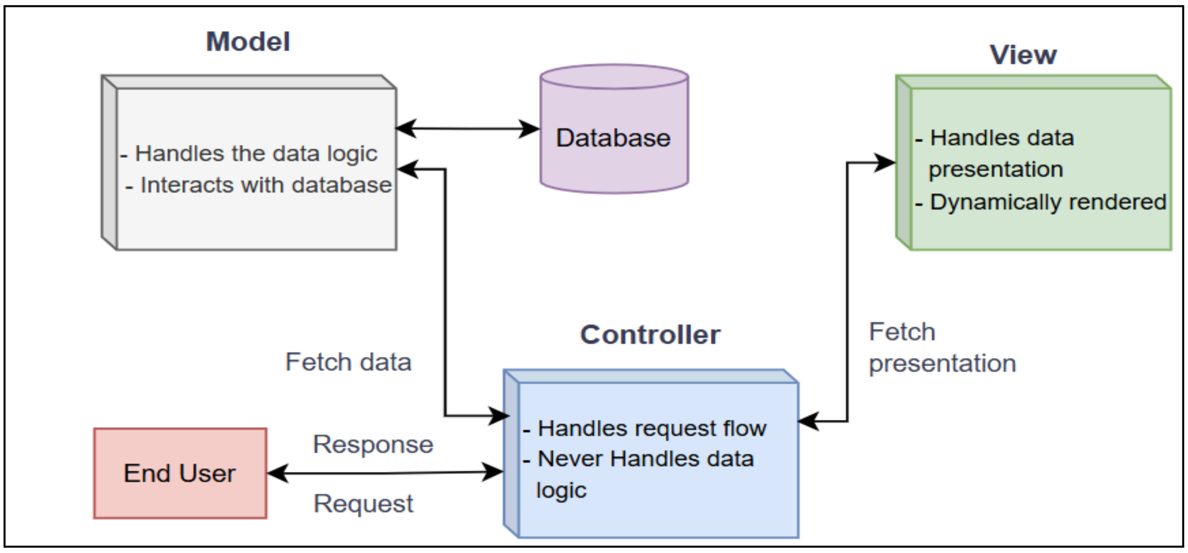


Figure 4.3.3.3

5. Software Security

5.1 Potential Threat/Attack

5.1.1 Broken Access Control / BOLA (Privilege escalation + booking abuse/DoS)

Broken Access Control occurs when users are able to perform actions beyond their authorized permissions. In the context of this system, a normal student user could attempt to directly access **administrator-only URLs or API endpoints** to create, update, or approve facility bookings.

If unauthorized users are allowed to create **PENDING or APPROVED bookings**, they could intentionally or unintentionally block facility time slots. Since the system prevents overlapping bookings, this misuse may lead to a **denial-of-service (DoS)** scenario where legitimate events are unable to reserve facilities.

This threat is especially critical in a shared university environment where multiple societies compete for limited facilities.

5.1.2 Malicious File Upload (Disguised Non-Image Files)

A malicious file upload attack occurs when an attacker uploads a file that is not a genuine image but is disguised using a valid image extension (e.g., `.png`, `.jpg`). Such files may contain executable code or scripts that could be executed or rendered later.

In this module, facility images uploaded by administrators are displayed on facility detail and timetable pages. If unsafe files are accepted, they may introduce **stored cross-site scripting (XSS)** or other security vulnerabilities, potentially compromising system integrity or user safety.

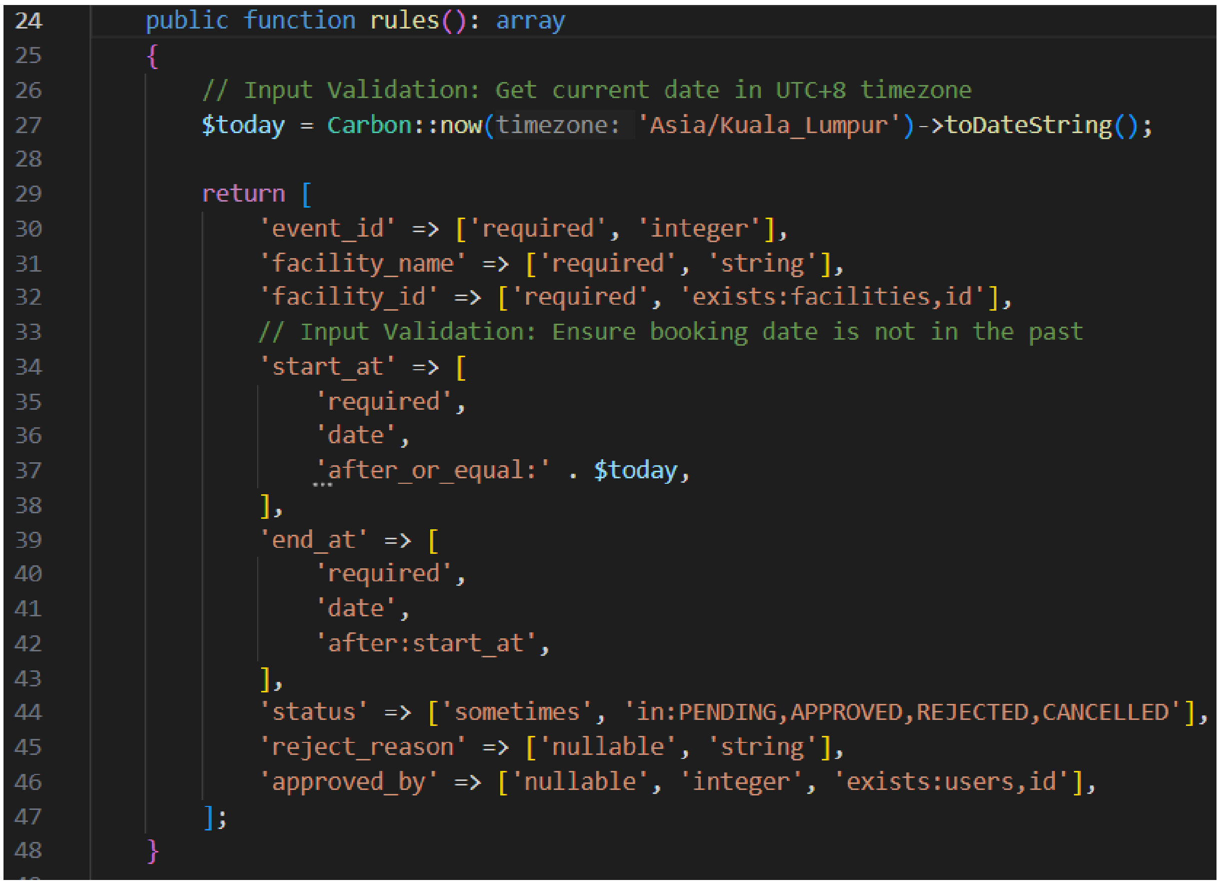
5.2 Secure Coding Practice

5.2.1 Input Validation - Validate for expected data types [11]

Input validation is implemented using **Laravel Form Request validation** and controller-level checks. These validations ensure that all incoming data conforms to expected formats and logical constraints before being processed.

Key validation rules include:

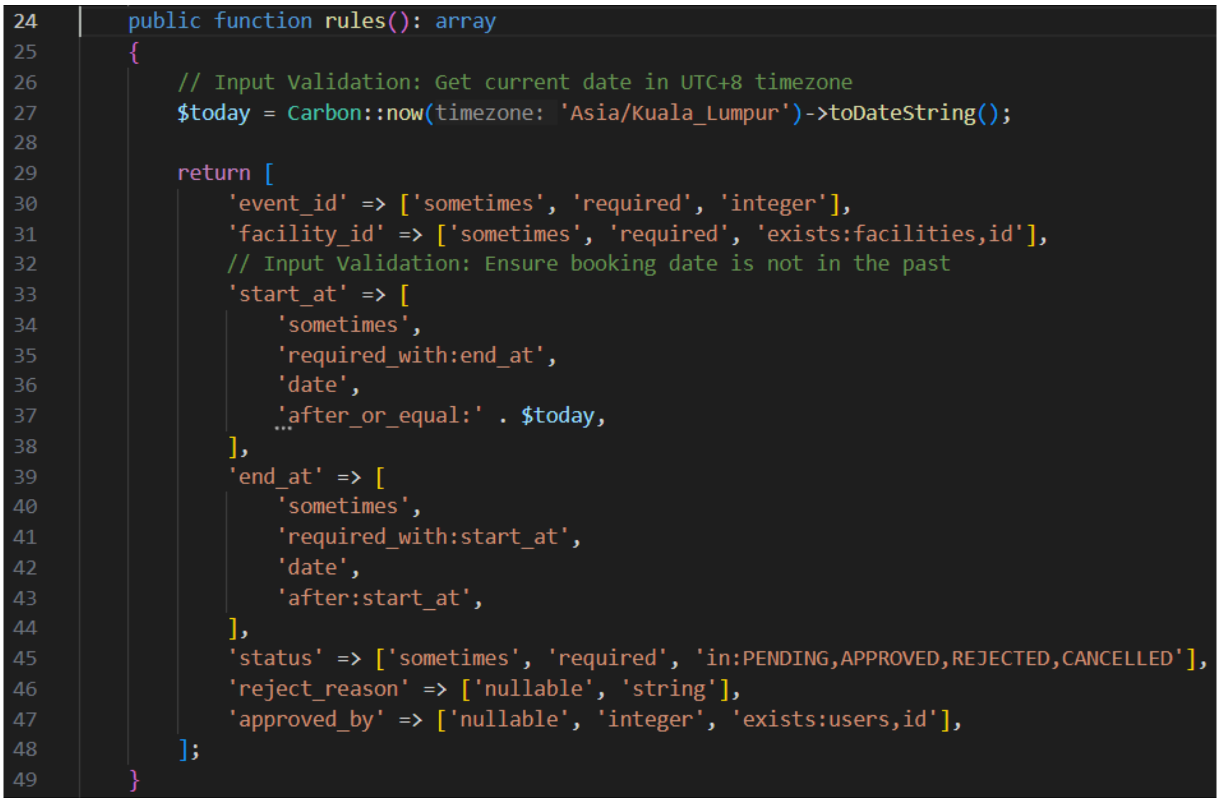
- Ensuring facility and event IDs exist in the database
- Verifying date and time formats
- Preventing bookings with start times later than end times
- Ensuring booking dates are not in the past
- Restricting booking status values to predefined enums



```php
    public function rules(): array
    {
        // Input Validation: Get current date in UTC+8 timezone
        $today = Carbon::now(timezone: 'Asia/Kuala_Lumpur')->toDateString();

        return [
            'event_id' => ['required', 'integer'],
            'facility_name' => ['required', 'string'],
            'facility_id' => ['required', 'exists:facilities,id'],
            // Input Validation: Ensure booking date is not in the past
            'start_at' => [
                'required',
                'date',
                'after_or_equal:' . $today,
            ],
            'end_at' => [
                'required',
                'date',
                'after:start_at',
            ],
            'status' => ['sometimes', 'in:PENDING,APPROVED,REJECTED,CANCELLED'],
            'reject_reason' => ['nullable', 'string'],
            'approved_by' => ['nullable', 'integer', 'exists:users,id'],
        ];
    }
```

Figure 5.2.1: BookingsStoreRequest



```php
    public function rules(): array
    {
        // Input Validation: Get current date in UTC+8 timezone
        $today = Carbon::now(timezone: 'Asia/Kuala_Lumpur')->toDateString();

        return [
            'event_id' => ['sometimes', 'required', 'integer'],
            'facility_id' => ['sometimes', 'required', 'exists:facilities,id'],
            // Input Validation: Ensure booking date is not in the past
            'start_at' => [
                'sometimes',
                'required_with:end_at',
                'date',
                'after_or_equal:' . $today,
            ],
            'end_at' => [
                'sometimes',
                'required_with:start_at',
                'date',
                'after:start_at',
            ],
            'status' => ['sometimes', 'required', 'in:PENDING,APPROVED,REJECTED,CANCELLED'],
            'reject_reason' => ['nullable', 'string'],
            'approved_by' => ['nullable', 'integer', 'exists:users,id'],
        ];
    }
```

Figure 5.2.2: BookingsUpdateRequest

5.2.2 Access Control - Restrict access to protected URLs to only authorized users [84]

Access control is enforced using a **multi-layered defense strategy**:

1. **Route Protection**

   Administrative routes are protected using authentication middleware, ensuring that only logged-in users can access them.

2. **Controller-Level Authorization**

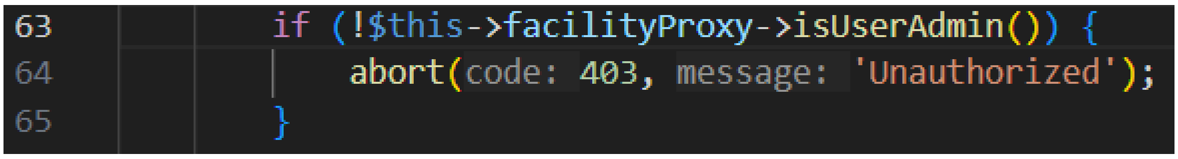
   Sensitive actions such as facility creation, booking approval, and booking modification are restricted to administrators within controllers.

3. **Proxy Pattern (Protection Proxy)**

   Protection proxies act as centralized gatekeepers. Before executing any sensitive operation, the proxy verifies whether the current user has sufficient privileges. Unauthorized requests are rejected before reaching core business logic.

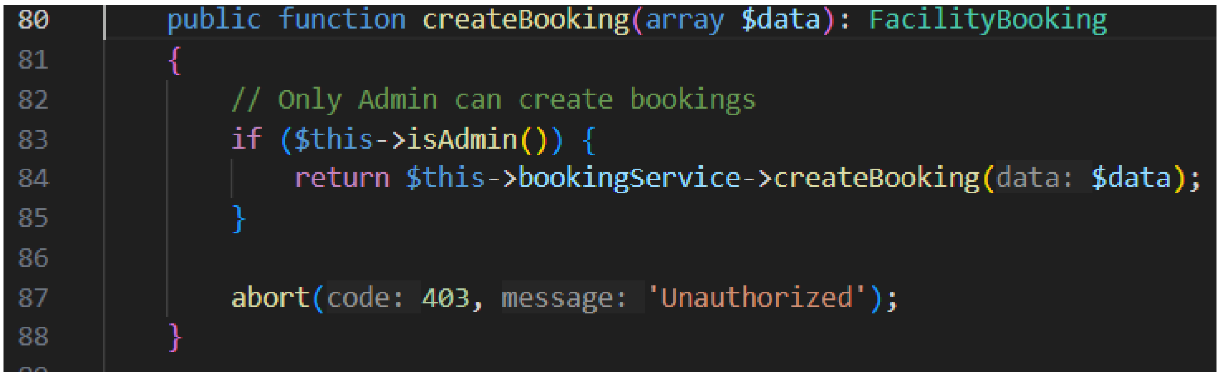
4. **Policy-Based Authorization**

   Laravel policies are used to enforce role-based permissions, ensuring that only authorized users can perform critical actions.



```
        if (!$this->facilityProxy->isUserAdmin()) {
            abort(code: 403, message: 'Unauthorized');
        }
```

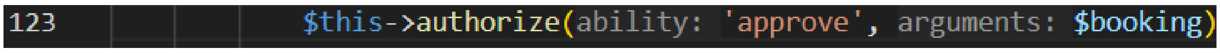
Figure 5.2.3



```
    public function createBooking(array $data): FacilityBooking
    {
        // Only Admin can create bookings
        if ($this->isAdmin()) {
            return $this->bookingService->createBooking(data: $data);
        }

        abort(code: 403, message: 'Unauthorized');
    }
```

Figure 5.2.4



```
            $this->authorize(ability: 'approve', arguments: $booking)
```

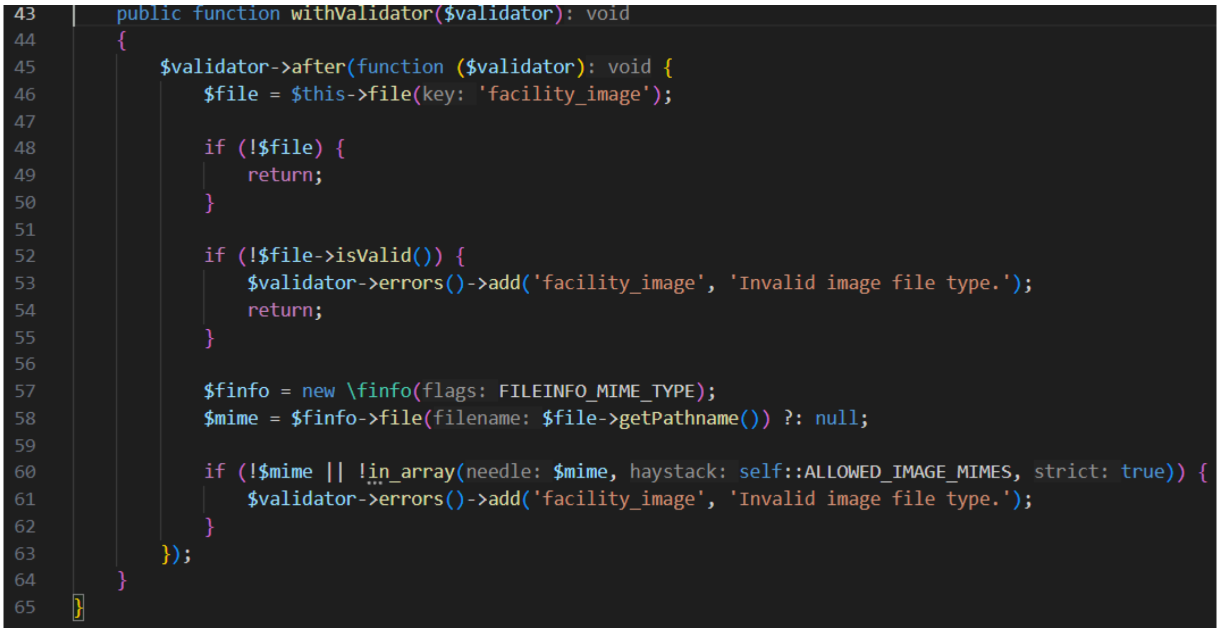
Figure 5.2.5

5.2.3 File Management - Validate uploaded files are the expected type by checking file headers [183]

Facility image uploads are secured using **file header (magic byte) inspection** in addition to standard extension and size checks.

The system verifies:

- File size limits
- Allowed extensions
- Actual MIME type using server-side file inspection



```php
    public function withValidator($validator): void
    {
        $validator->after(function ($validator): void {
            $file = $this->file(key: 'facility_image');

            if (!$file) {
                return;
            }

            if (!$file->isValid()) {
                $validator->errors()->add('facility_image', 'Invalid image file type.');
                return;
            }

            $finfo = new \finfo(flags: FILEINFO_MIME_TYPE);
            $mime = $finfo->file(filename: $file->getPathname()) ?: null;

            if (!$mime || !in_array(needle: $mime, haystack: self::ALLOWED_IMAGE_MIMES, strict: true)) {
                $validator->errors()->add('facility_image', 'Invalid image file type.');
            }
        });
    }
}
```

Figure 5.2.6

6. Web Services

The Facility Management Module integrates web services to enable **cross-module communication** with the Event Management Module. RESTful web services using **JSON over HTTP** are implemented to support real-time facility availability checking and event data retrieval. All services follow the **Interface Agreement (IFA)** requirements by including request identifiers and timestamps to ensure traceability and reliability.

1. Service Exposure

The Facility Management Module exposes a RESTful API that allows other modules, specifically the Event Management Module, to check whether a facility is available for a requested time range before an event is created or submitted for approval. This service helps prevent timetable clashes and ensures that only valid facility bookings are made.

2. The format for the Interface Agreement (IFA) is as below:

Webservice Mechanism

	Description
Protocol	RESTFUL
Function	Check whether a facility is available for a selected time range

Description	
Source Module	Facility Management Module
Target Module	Event Management Module
URL	POST http://127.0.0.1:8000/api/facilities/availability
Function Name	`checkAvailability()`

Table 6.1

Web Services Request Parameter (provide):

Field Name	**Field Type**	**Mandatory / Optional**	**Description**	**Format**
requestID	String	Mandatory	Unique request identifier for tracking	UUID / String
facilityId	Integer	Mandatory	Facility ID to be checked	Existing facilities.id
startAt	String	Mandatory	Booking start date and time	YYYY-MM-DD HH:MM:SS
endAt	String	Mandatory	Booking end date and time	YYYY-MM-DD HH:MM:SS
timeStamp	String	Mandatory	Request creation timestamp	YYYY-MM-DD HH:MM:SS

Table 6.2

Web Services Response Parameter (consume):

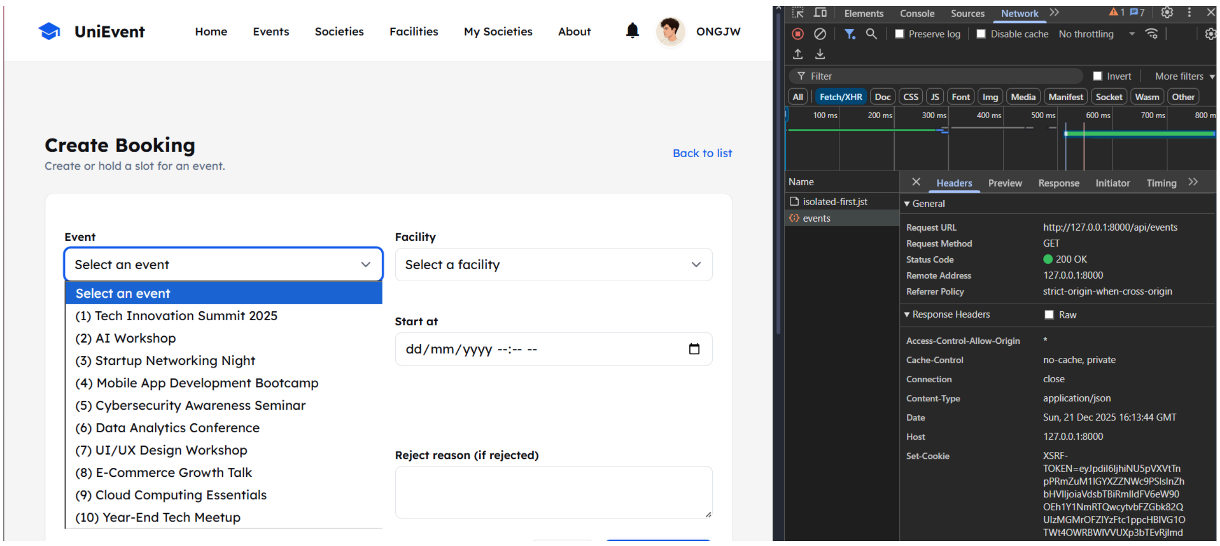
Field Name	**Field Type**	**Mandatory / Optional**	**Description**	**Format**
requestID	String	Mandatory	Unique request identifier for tracking	UUID / String
facilityId	Integer	Mandatory	Facility ID to be checked	Existing facilities.id
startAt	String	Mandatory	Booking start date and time	YYYY-MM-DD HH:MM:SS
endAt	String	Mandatory	Booking end date and time	YYYY-MM-DD HH:MM:SS

timeStamp	String	Mandatory	Request creation timestamp	YYYY-MM-DD HH:MM:SS

Table 6.3

3. Service Consumption

6.3.1 Get Event List



UniEvent — Home · Events · Societies · Facilities · My Societies · About · ONGJW

Create Booking

Create or hold a slot for an event.

Back to list

Event	Facility
Select an event	Select a facility

Event options:

- Select an event
- (1) Tech Innovation Summit 2025
- (2) AI Workshop
- (3) Startup Networking Night
- (4) Mobile App Development Bootcamp
- (5) Cybersecurity Awareness Seminar
- (6) Data Analytics Conference
- (7) UI/UX Design Workshop
- (8) E-Commerce Growth Talk
- (9) Cloud Computing Essentials
- (10) Year-End Tech Meetup

Start at: dd/mm/yyyy --:-- --

Reject reason (if rejected)

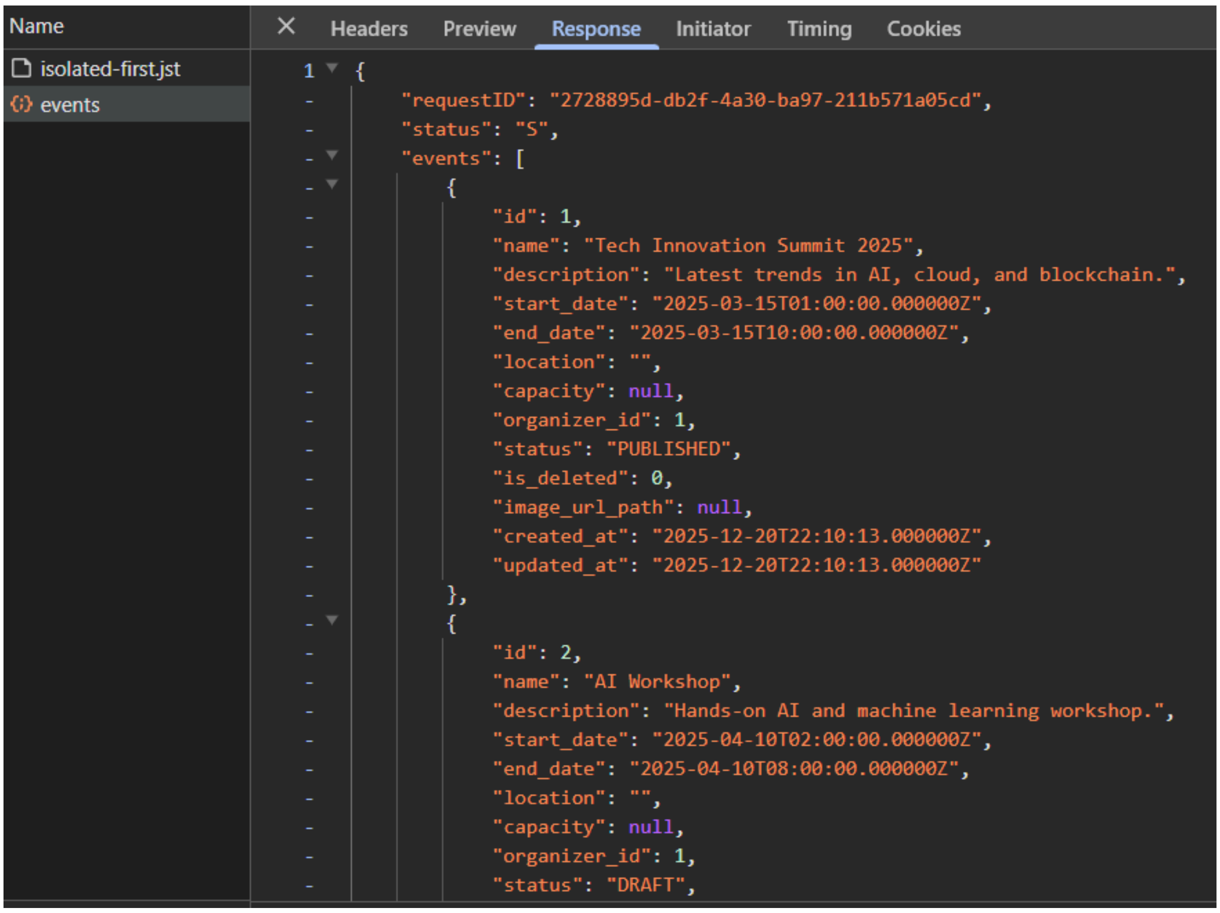
General

Request URL	http://127.0.0.1:8000/api/events
Request Method	GET
Status Code	● 200 OK
Remote Address	127.0.0.1:8000
Referrer Policy	strict-origin-when-cross-origin

Response Headers

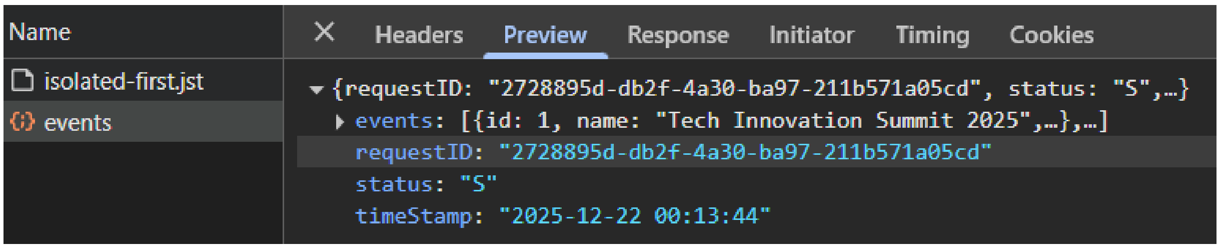
Access-Control-Allow-Origin	*
Cache-Control	no-cache, private
Connection	close
Content-Type	application/json
Date	Sun, 21 Dec 2025 16:13:44 GMT
Host	127.0.0.1:8000
Set-Cookie	XSRF-TOKEN=eyJpdil6IjhiNU5pVXVtTnpPRmZuM1lGYXZZNWc9PSlsInZhbHVlIjoiaVdsbTBiRmlldFV6eW90OEh1Y1NmRTQwcytvbFZGbk82QUlzMGMrOFZIYzFtc1ppcHBIVG1OTWt4OWRBWlVVUXp3bTEvRjlmd

Figure 6.1.1



```
{
    "requestID": "2728895d-db2f-4a30-ba97-211b571a05cd",
    "status": "S",
    "events": [
        {
            "id": 1,
            "name": "Tech Innovation Summit 2025",
            "description": "Latest trends in AI, cloud, and blockchain.",
            "start_date": "2025-03-15T01:00:00.000000Z",
            "end_date": "2025-03-15T10:00:00.000000Z",
            "location": "",
            "capacity": null,
            "organizer_id": 1,
            "status": "PUBLISHED",
            "is_deleted": 0,
            "image_url_path": null,
            "created_at": "2025-12-20T22:10:13.000000Z",
            "updated_at": "2025-12-20T22:10:13.000000Z"
        },
        {
            "id": 2,
            "name": "AI Workshop",
            "description": "Hands-on AI and machine learning workshop.",
            "start_date": "2025-04-10T02:00:00.000000Z",
            "end_date": "2025-04-10T08:00:00.000000Z",
            "location": "",
            "capacity": null,
            "organizer_id": 1,
            "status": "DRAFT",
```

Figure 6.1.2



```
▾ {requestID: "2728895d-db2f-4a30-ba97-211b571a05cd", status: "S",…}
  ▸ events: [{id: 1, name: "Tech Innovation Summit 2025",…},…]
    requestID: "2728895d-db2f-4a30-ba97-211b571a05cd"
    status: "S"
    timeStamp: "2025-12-22 00:13:44"
```

Figure 6.1.3

7. References

Concept && Coding - by Shrayansh. (2023, August 27). *32. All Structural Design Patterns | Decorator, Proxy, Composite, Adapter, Bridge, facade, FlyWeight* [Video]. YouTube. https://www.youtube.com/watch?v=WxGtmIBZszk

Refactoring.Guru. (2025, January 1). *Proxy in PHP*. https://refactoring.guru/design-patterns/proxy/php/example

Zhuk, M. (2024, June 23). *Design Patterns in PHP 8: Proxy*. DEV Community. https://dev.to/zhukmax/design-patterns-in-php-8-proxy-58ab

Team, B. (n.d.). *The PHP framework for web Artisans*. Binarcode. https://www.binarcode.com/services/laravel-development

What is MVC in PHP and how does it work? (2025, September 2). Ozzu®. https://www.ozzu.com/questions/610473/what-is-mvc-in-php-and-how-does-it-work

Quick Programming. (2022, November 4). *PHP MVC Framework from scratch | Source code included | Quick programming tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=q0JhJBYi4sw

Programster. (2018, March 18). *PHP - Creating your own ORM for a MySQL database* [Video]. YouTube. https://www.youtube.com/watch?v=nsy-5cJcPdA