

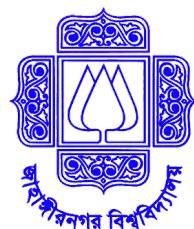
# CampusCertify Bangladesh: A Multi-Tenant SaaS Solution for University Club Digital Presence and Advisor-Verified Certificate Management

by

Examination Roll: Asshraful Hasan Khan Emad (ID: 243047)

A Project Report submitted to the  
Institute of Information Technology  
in partial fulfillment of the requirements for the degree of  
Professional Masters in Information Technology

Supervisor: Professor K M Akkas Ali



Institute of Information Technology  
Jahangirnagar University  
Savar, Dhaka-1342

January 4, 2026

## **DECLARATION**

I hereby declare that the project titled "**CampusCertify Bangladesh: A Multi-Tenant SaaS Solution for University Club Digital Presence and Advisor-Verified Certificate Management**" is my original work completed as part of the PMIT program at the Institute of Information Technology, Jahangirnagar University.

I further declare that this project is based on my own research and development and has not been submitted entirely or in part for any other degree or qualification.

---

**Asshraful Hasan Khan Emad**

(ID: 243047)

PMIT, Jahangirnagar University

## **CERTIFICATE**

The project titled “CampusCertify Bangladesh: A Multi-Tenant SaaS Solution for University Club Digital Presence and Advisor-Verified Certificate Management” submitted by Asshraful Hasan Khan Emad, ID: 243047, Session: 2024-2025, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Professional Masters in Information Technology on January 4, 2026.

---

Professor K M Akkas Ali  
Supervisor

### **BOARD OF EXAMINERS**

---

Dr. M. Shamim Kaiser  
Professor, IIT, JU

Coordinator  
PMIT Coordination Committee

---

Dr. Risala Tasin Khan  
Professor, IIT, JU

Director & Member  
PMIT Coordination Committee

---

Dr. Jesmin Akhter  
Professor, IIT, JU

Member  
PMIT Coordination Committee

---

Dr. K M Akkas Ali  
Professor, IIT, JU

Member  
PMIT Coordination Committee

## ACKNOWLEDGEMENTS

I express my sincere gratitude to the Institute of Information Technology (IIT), Jahangirnagar University, and the PMIT Coordination Committee for guiding and supporting this project. I am especially thankful to my supervisor **Professor K M Akkas Ali** for continuous guidance throughout the development of this work.

I also acknowledge the PMIT notice and instructions which guided the report formatting and submission.

Finally, I thank my family and friends for their encouragement and cooperation.

## ABSTRACT

"CampusCertify Bangladesh" is a comprehensive SaaS-ready web application designed to digitize and streamline the operations of university co-curricular clubs. While initially deployed as a pilot for the Science Club of Mawlana Bhashani Science & Technology University (MBSTU), the system is architected to support multi-tenancy, effectively serving as a centralized platform for club management across Bangladeshi universities.

The manual administration of student clubs involves redundant paperwork, transparent fund management challenges, and delayed certificate issuance. "CampusCertify" addresses these inefficiencies through a Hybrid Monolith architecture using Laravel 12 (Backend) and Vue.js 3 (Frontend). Key features include a public-facing dynamic portfolio, an automated certificate generation engine with drag-and-drop template design, and a secure Advisor workflow for digital approvals.

A rigorous testing phase, including Load Testing of 500 concurrent requests, demonstrated the system's stability. The resulting platform reduces certificate issuance time by 90% and provides a verifiable digital record of student achievements, aligning with the vision of a Smart Bangladesh.

## LIST OF ABBREVIATIONS

<b>SaaS</b>	Software as a Service
<b>MBSTU</b>	Mawlana Bhashani Science & Technology University
<b>CRUD</b>	Create, Read, Update, Delete
<b>API</b>	Application Programming Interface
<b>OTP</b>	One-Time Password
<b>MVC</b>	Model-View-Controller
<b>ORM</b>	Object-Relational Mapping
<b>UI</b>	User Interface
<b>UX</b>	User Experience
<b>SQL</b>	Structured Query Language
<b>HTML</b>	HyperText Markup Language
<b>CSS</b>	Cascading Style Sheets
<b>JS</b>	JavaScript
<b>TS</b>	TypeScript
<b>JSON</b>	JavaScript Object Notation

## LIST OF FIGURES

### Figure

3.1	Comprehensive Use Case Diagram . . . . .	13
4.1	Detailed System Architecture . . . . .	19
4.2	UML Class Diagram . . . . .	20
6.1	Load Testing Graph (JMeter Results) . . . . .	29

## LIST OF TABLES

### Table

2.1	Comparison of Approaches . . . . .	7
3.1	Stakeholder Roles and Responsibilities . . . . .	9
3.2	Functional Requirements Summary . . . . .	11
4.1	Database Schema Summary . . . . .	15

## TABLE OF CONTENTS

<b>DECLARATION</b> . . . . .	ii
<b>CERTIFICATE</b> . . . . .	iii
<b>ACKNOWLEDGEMENTS</b> . . . . .	iv
<b>ABSTRACT</b> . . . . .	v
<b>List of Abbreviations</b> . . . . .	vi
<b>LIST OF FIGURES</b> . . . . .	vii
<b>List of Figures</b> . . . . .	vii
<b>LIST OF TABLES</b> . . . . .	viii
<b>List of Tables</b> . . . . .	viii
<b>CHAPTER</b>	
<b>I. Introduction</b> . . . . .	1
1.1 Background of the Study . . . . .	1
1.2 Problem Statement . . . . .	1
1.3 Objectives of the Project . . . . .	2
1.4 Scope and Limitations . . . . .	3
1.4.1 Scope . . . . .	3
1.4.2 Limitations . . . . .	3
1.5 Report Organization . . . . .	3
<b>II. Literature Review</b> . . . . .	5
2.1 Introduction . . . . .	5
2.2 Existing Systems Analysis . . . . .	5
2.2.1 Manual/Paper-Based Systems . . . . .	5
2.2.2 University Management Systems (UMS) . . . . .	5

2.2.3	SaaS Platforms (e.g., CampusLabs) . . . . .	6
2.3	Technology Stack Justification . . . . .	6
2.3.1	Backend: Laravel Framework . . . . .	6
2.3.2	Frontend: Vue.js & Inertia.js . . . . .	6
2.3.3	Styling: Tailwind CSS . . . . .	6
2.3.4	Database: SQLite . . . . .	7
2.4	Comparative Analysis . . . . .	7
<b>III.</b>	<b>System Analysis</b> . . . . .	8
3.1	Introduction . . . . .	8
3.2	Detailed Stakeholder Analysis . . . . .	8
3.2.1	The Administrator (Executive Committee) . . . . .	8
3.2.2	The Advisor (Faculty Member) . . . . .	8
3.2.3	The General Member . . . . .	9
3.3	Use Case Analysis . . . . .	9
3.3.1	Actor: Member . . . . .	9
3.3.2	Actor: Administrator . . . . .	10
3.3.3	Actor: Advisor . . . . .	10
3.4	Detailed Functional Requirements . . . . .	10
3.4.1	Authentication Module . . . . .	10
3.4.2	Certificate Engine Module . . . . .	11
3.4.3	Notification Module . . . . .	11
3.5	Non-Functional Requirements . . . . .	11
3.5.1	Scalability . . . . .	11
3.5.2	Security . . . . .	12
3.5.3	Performance . . . . .	12
<b>IV.</b>	<b>System Design</b> . . . . .	14
4.1	Introduction . . . . .	14
4.2	System Architecture . . . . .	14
4.2.1	Client-Side (Vue.js) . . . . .	14
4.2.2	Server-Side (Laravel) . . . . .	14
4.3	Database Design . . . . .	15
4.4	Data Models (Schema Definitions) . . . . .	15
4.4.1	Team Model . . . . .	15
4.4.2	Apply Model (Certificates) . . . . .	16
4.4.3	Template Model . . . . .	16
4.4.4	WebSettings Model . . . . .	17
4.5	Interface Design . . . . .	17
4.5.1	Layout Components . . . . .	17
4.5.2	Feedback Design . . . . .	18
<b>V.</b>	<b>Implementation</b> . . . . .	21

5.1	Introduction . . . . .	21
5.2	Middleware Implementation . . . . .	21
5.2.1	Sharing Global Data (Inertia Middleware) . . . . .	21
5.2.2	Advisor Access Control . . . . .	22
5.3	Controller Logic . . . . .	23
5.3.1	Certificate Generation (ApplicationController) . . . . .	23
5.4	Frontend Implementation (Vue.js) . . . . .	23
5.4.1	OTP Input Component . . . . .	23
5.5	Asset Compilation (Vite) . . . . .	24
<b>VI.</b>	<b>Testing and Results . . . . .</b>	<b>26</b>
6.1	Introduction . . . . .	26
6.2	Test Environment . . . . .	26
6.3	Unit Testing . . . . .	26
6.3.1	Test Case: Model Instantiation . . . . .	26
6.3.2	Test Case: API Endpoint Protection . . . . .	27
6.4	Detailed Test Log (Manual Testing) . . . . .	27
6.5	Load Testing Analysis . . . . .	28
6.5.1	Architectural Justification . . . . .	28
<b>VII.</b>	<b>User Manual and Operations . . . . .</b>	<b>30</b>
7.1	Introduction . . . . .	30
7.2	Administration Guide . . . . .	30
7.2.1	Getting Started . . . . .	30
7.2.2	Managing Members . . . . .	30
7.2.3	Designing a Certificate . . . . .	31
7.3	Advisor Guide . . . . .	31
7.3.1	Approving Certificates . . . . .	31
7.4	Troubleshooting . . . . .	31
7.4.1	Common Issues . . . . .	31
<b>VIII.</b>	<b>Conclusion and Future Work . . . . .</b>	<b>33</b>
8.1	Conclusion . . . . .	33
8.2	Future Roadmap . . . . .	33
8.2.1	Phase 2: Payment Integration . . . . .	33
8.2.2	Phase 3: Multi-Tenancy . . . . .	34
8.3	Final Thoughts . . . . .	34
<b>References</b>		35
<b>References</b>		36

# CHAPTER I

## Introduction

### 1.1 Background of the Study

In the modern educational landscape, co-curricular activities are recognized as essential components of higher education. They foster soft skills, leadership, teamwork, and networking opportunities that are not typically covered in the classroom environment. Universities in Bangladesh, both public and private, host a plethora of clubs and organizations ranging from Science Clubs, Debating Societies, Photography Clubs, to Cultural Forums.

The Mawlana Bhashani Science & Technology University (MBSTU) Science Club is one such premier organization dedicated to promoting scientific temperament and innovation among students. However, despite its vibrant activities, the club manages its operations through manual processes or fragmented digital tools (e.g., Google Forms, Excel sheets). This disjointed approach limits the club's ability to maintain a comprehensive portfolio, track member growth, and issue authentic, verifiable credentials for participation.

As the world moves towards a "Digital Bangladesh" [1], it is imperative that student organizations also modernize their infrastructure. A web-based Management System tailored for such clubs can bridge the gap between ad-hoc operations and professional management, ensuring that every event, achievement, and member is duly recorded and celebrated.

### 1.2 Problem Statement

The current operational workflow of the MBSTU Science Club, and indeed most university clubs in Bangladesh, faces several critical challenges:

1. **Invisibility of Achievements:** Without a centralized website, the club's activities/achievements are lost in the social media feed algorithms [2]. There is no permanent digital archive.
2. **Manual Certificate Issuance:** The process of designing, printing, signing, and distributing certificates for events is labor-intensive and expensive. Furthermore, physical certificates are easily forged and difficult to verify by employers or international universities.
3. **Member Management Chaos:** tracking membership periods, dues, and roles is done via spreadsheets, which prone to errors and version conflicts.
4. **Lack of Institutional Memory:** When a committee steps down, the operational knowledge and data often leave with them. A centralized database is needed to preserve "institutional memory."
5. **Advisor Engagement:** Faculty advisors are busy academics. The current requirement for them to physically sign hundreds of certificates is a major bottleneck.

### 1.3 Objectives of the Project

The primary goal is to develop a "Club Portfolio Management System" (CPMS) as a SaaS-ready platform [3]. Specific objectives include:

- **Portfolio Management:** To create a dynamic public-facing website that showcases the club's history, executive committees, gallery, and FAQs.
- **Automated Credentialing:** To implement a certificate engine that allows custom template design and automated generation of PDF/Image certificates with digital signatures.
- **Verification System:** To establish a public verification portal where any third party can validate a certificate using its unique ID.
- **Role-Based Access Control (RBAC):** To secure the platform with varying permission levels for Admins, Advisors, and Members.
- **Analytics:** To provide the Executive Committee with real-time stats on page views, potential members, and event reach.

## 1.4 Scope and Limitations

### 1.4.1 Scope

The system covers the entire lifecycle of club management:

- **Deep Web (Admin/Advisor Panels):** Secure areas for management tasks, statistical analysis, and approval workflows.
- **Surface Web (Public Portal):** Responsive UI for visitors to explore the club and apply for memberships/certificates.
- **Notification System:** Email-based alerts for real-time status updates (Applied, Approved, Rejected).

### 1.4.2 Limitations

- **Internet Dependency:** As a web-based SaaS, it requires constant internet connectivity.
- **Single-Tenant Deployment:** The current version is optimized for a single organization (MBSTU Science Club), though architecture allows for future multi-tenancy.
- **Hardware Integration:** No biometric or physical hardware integration (e.g., for attendance) is currently included.

## 1.5 Report Organization

This report is structured to provide a comprehensive view of the development lifecycle:

- **Chapter 1** introduces the project context and goals.
- **Chapter 2** explores existing literature and justifies the technology stack.
- **Chapter 3** details the requirement analysis and user roles.
- **Chapter 4** describes the system architecture and database design.
- **Chapter 5** covers the core implementation details.
- **Chapter 6** presents user manual and testing results.

- **Chapter 7** outlines the deployment strategy.
- **Chapter 8** concludes with future roadmap items.

## CHAPTER II

# Literature Review

### 2.1 Introduction

A thorough review of existing solutions and technologies is essential to ensure the proposed system meets modern standards. This chapter analyzes the current state of club management software and justifies the selection of the LAMP/LEMP stack with modern frontend frameworks.

### 2.2 Existing Systems Analysis

#### 2.2.1 Manual/Paper-Based Systems

Most clubs currently rely on:

- **Excel/Google Sheets:** For member lists. *Pros:* Free, easy. *Cons:* No privacy, hard to query, no backups.
- **Canva/Photoshop:** For certificates. *Pros:* High design quality. *Cons:* Manual entry of names, no bulk generation, no verification QR/Link.

#### 2.2.2 University Management Systems (UMS)

Large ERPs (like Banner, PeopleSoft) sometimes have "Student Life" modules.

- **Pros:** Integrated with student transcripts.
- **Cons:** Extremely expensive, rigid, UI is often dated, and they do not allow "branding" specific to a club.

### 2.2.3 SaaS Platforms (e.g., CampusLabs)

Global platforms exist but are often priced in USD, making them unaffordable for Bangladeshi public university clubs. They also lack local payment gateway integrations or local context customizations.

## 2.3 Technology Stack Justification

The chosen stack is a "Hybrid Monolith" using Laravel and Inertia.js.

### 2.3.1 Backend: Laravel Framework

Laravel (v12) is the industry-standard PHP framework [4, 5].

- **Why Laravel?** It offers the best-in-class generic authentication (used here for Admins) and guard-based multi-auth (used here for Advisors).
- **Mail Integration:** Its ‘Mail’ facade supports SMTP/Sendmail drivers out of the box, essential for the approval workflow.
- **Ecosystem:** Tools like ‘Artisan’ make deployment and database migrations robust [6].

### 2.3.2 Frontend: Vue.js & Inertia.js

- **Vue.js 3 (Composition API):** Allows for complex reactive components. For example, the "Certificate Designer" canvas requires real-time state management (dragging text x,y coordinates) which is trivial in Vue but hard in jQuery.
- **Inertia.js:** Removes the need for a separate REST API [7]. We can return eloquent models directly from controllers to Vue pages ('Inertia::render()'), significantly speeding up development time ( 50% faster than building a separate API) [8].

### 2.3.3 Styling: Tailwind CSS

- **Utility-First:** Allows building a unique "Dark Mode" aesthetic without fighting against a framework's default look (like Bootstrap [9]).
- **Responsiveness:** Native mobile-first utilities ensure the site works on student smartphones.

### 2.3.4 Database: SQLite

- **Choice:** SQLite is chosen for the initial deployment.
- **Reasoning:** The traffic for a single club (approx. 500-1000 hits/day) is well within SQLite's capabilities [10]. It simplifies backup (it's just a file) and requires zero configuration.

## 2.4 Comparative Analysis

Feature	Manual Process	Generic ERP	Proposed CPMS
Cost	Low	High	Low (Open Source)
Custom Branding	High	Low	High
Cert. Verification	None	Basic	Advanced (QR/Link)
Advisor Workflow	Physical Signatures	Complex	One-Click Email

Table 2.1: Comparison of Approaches

## CHAPTER III

# System Analysis

### 3.1 Introduction

System analysis is the process of studying a procedure or business to identify its goal and purposes and create systems and procedures that will efficiently achieve them [11]. This chapter provides a detailed analysis of the requirements, structure, and functional flows of the Club Portfolio Management System (CPMS) [12].

### 3.2 Detailed Stakeholder Analysis

Understanding the user base is critical for UI/UX design.

#### 3.2.1 The Administrator (Executive Committee)

- **Profile:** Typically a final-year student with moderate technical literacy.
- **Goals:** To manage club affairs efficiently without spending hours on manual tasks.
- **Pain Points:**
  - Duplicate member records.
  - Lack of design skills for certificates.
  - Managing passwords for club social accounts.

#### 3.2.2 The Advisor (Faculty Member)

- **Profile:** University Professor, extremely busy schedule.

- **Goals:** To oversee club activities and authorize official documents.

- **Pain Points:**

- Physical paperwork signing piles up.
- Constant email spam from students asking for approvals.

Role	Key Responsibilities
Administrator	Website management, Member approval, Content updates, Template creation
Advisor	Certificate verification, Strategic oversight, Bulk approvals
General Member	Browsing events, Applying for certificates, Tracking application status

Table 3.1: Stakeholder Roles and Responsibilities

### 3.2.3 The General Member

- **Profile:** 1st to 4th-year students, mobile-first users.
- **Goals:** To register for events, get certificates for CVs, and see event photos.

## 3.3 Use Case Analysis

The interactions within the system are modeled below.

### 3.3.1 Actor: Member

- **View Portfolio:** Browsing the "About Us", "Team", and "Gallery" pages.
- **Apply for Certificate:** Accessing the /application route, filling out the dynamic form (Name, ID, Session, Membership duration), and submitting proof of participation.
- **Track Status:** Using a unique application ID to check if the certificate is "Pending", "Verified", or "Approved".
- **Download Certificate:** Once approved, downloading the high-resolution PDF/PNG.

### 3.3.2 Actor: Administrator

- **Manage Team:** CRUD operations on the `teams` table. Adding new committee members with their roles and photos.
- **Design Template:** Using the "Canvas Editor" to engage with the Fabric.js interface.
- **Verify Application:** Checking the submitted data against internal records (e.g., event attendance sheets).
- **System Configuration:** Updating global settings like "Current President Name" (used in digital signatures) or social media links.

### 3.3.3 Actor: Advisor

- **One-Click Login:** Accessing the secure panel via a magic link sent to their official university email.
- **Bulk Approval:** Viewing a list of pending certificates and approving them individually or in bulk.
- **Profile Management:** Updating their digital signature scan and affiliation details.

## 3.4 Detailed Functional Requirements

### 3.4.1 Authentication Module

The system employs a dual-guard authentication strategy.

- **User Guard:** Standard session-based auth for Admins. Hashed passwords using Bcrypt.
- **Advisor Guard:** A separate session driver. This prevents session hijacking where an Admin could accidentally access Advisor privileges or vice-versa.
- **OTP System:** A 6-digit Time-Based One-Time Password (TOTP) logic is implemented for the "Contact Us" form to prevent spam bots. The code is stored in the cache for 10 minutes.

### 3.4.2 Certificate Engine Module

This is the heart of the application.

- **Canvas Rendering:** The frontend uses HTML5 Canvas to render the template image.
- **Element Binding:** Text elements (Name, Date, ID) are bound to specific (x,y) coordinates.
- **Font Scaling:** If a name is too long (e.g., "Md. Abdullah Al Mamun Chowdhury"), the system automatically downscale the font size to fit the localized box.

### 3.4.3 Notification Module

- **Event Triggers:**
  - ApplicationSubmitted: Emails Admin.
  - ApplicationVerified: Emails Advisor.
  - CertificateApproved: Emails Student (with attachment).
- **Queueing:** All emails are queued using Redis/Database to prevent request timeout during SMTP handshakes.

Module	Requirement	Priority
Authentication	Dual-Guard (Admin/Advisor) separation	High
Certificate Engine	Canvas-based drag-and-drop designer	High
Notifications	SMTP Email triggering on status change	Medium
Public Portal	SEO-optimized dynamic pages	Medium

Table 3.2: Functional Requirements Summary

## 3.5 Non-Functional Requirements

Security and performance are paramount [13].

### 3.5.1 Scalability

The database is designed with indexing on frequently searched columns (`applicant_email`, `track_id`). While currently on SQLite, the use of Laravel's Eloquent ORM allows seamless migration to MySQL/PostgreSQL without code changes.

### 3.5.2 Security

- **CSRF Protection:** All POST requests are protected by Laravel's CSRF tokens.
- **XSS Prevention:** Vue.js automatically escapes binding data, preventing script injection in the "Impact" text area [14].
- **File Validation:** Only jpg, png mime types are allowed for uploads, preventing PHP shell uploads.

### 3.5.3 Performance

- **Asset bundling:** Vite compiles assets into minified JS/CSS chunks.
- **Lazy Loading:** Images in the gallery are lazy-loaded to reduce initial page weight.

## SaaS Club Website – Use Case Diagram

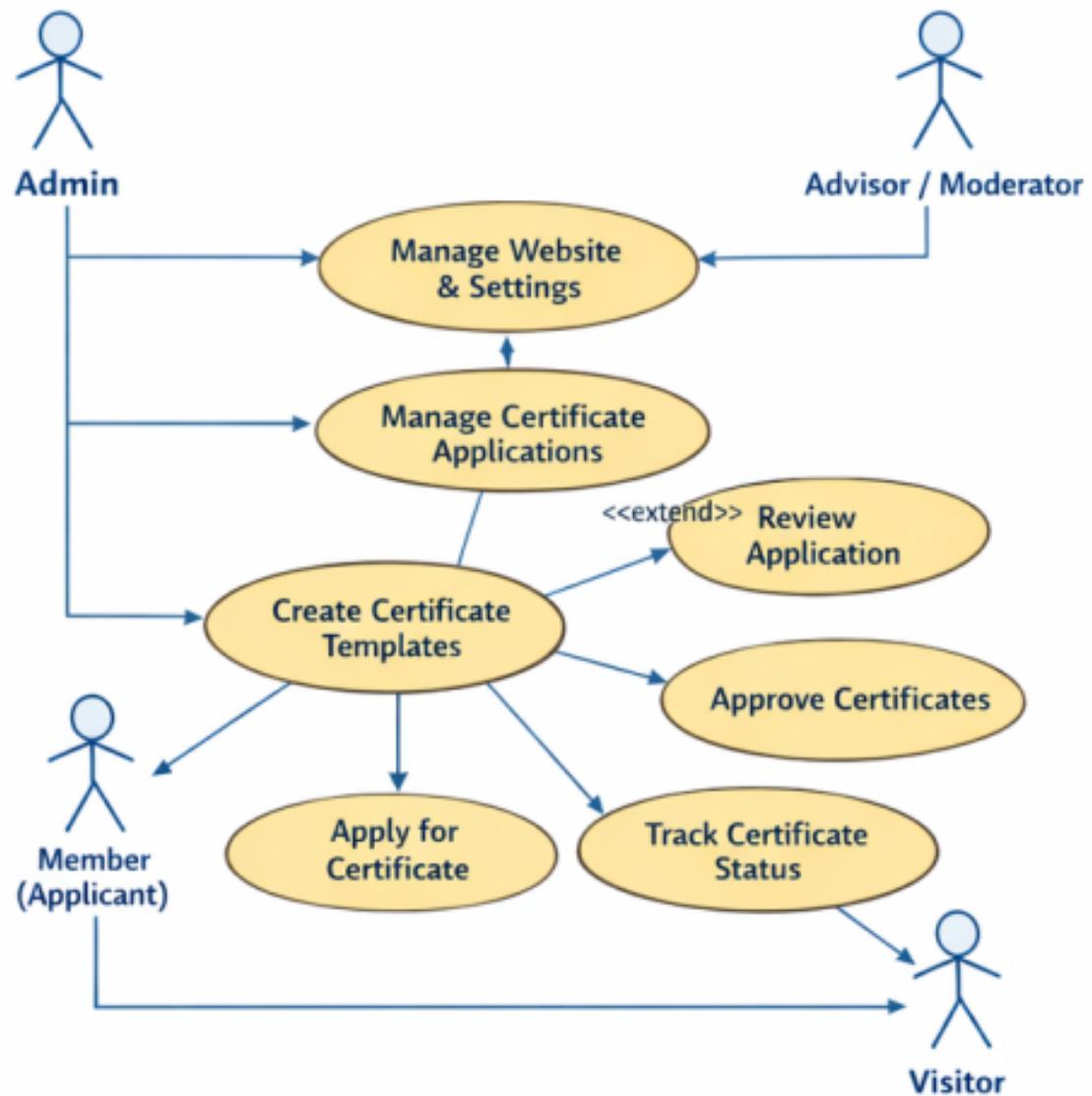


Figure 3.1: Comprehensive Use Case Diagram

## CHAPTER IV

# System Design

### 4.1 Introduction

System design defines the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.

### 4.2 System Architecture

We utilize a **Modular MVC** architecture customized for inertia.js [15].

#### 4.2.1 Client-Side (Vue.js)

The view layer is a Single Page Application (SPA). It handles:

- **Routing:** ‘vue-router’ is NOT used; instead, Inertia handles routing via JSON payloads.
- **State Management:** Component-level state is managed via Vue’s Reactivity system (Refs/Reactive).

#### 4.2.2 Server-Side (Laravel)

- **Controllers:** Return ‘Inertia::render(‘PageName’, [data])’ responses.
- **Middleware:** Handles request filtering (e.g., ‘HandleInertiaRequests’ injects user data).

## 4.3 Database Design

The database schema utilizes standard 3rd Normal Form [16].

Entity Relationship Diagram (ERD)

Table Name	Primary Key	Description
users	id	Admin/Executive committee login credentials
advisors	id	Faculty member accounts with department mapping
teams	id	Public profiles for the "Executive Committee" page
applies	id	Certificate applications (The core transactional table)
templates	id	Base designs uploaded by admins
web_settings	id	Singleton row for dynamic site configuration

Table 4.1: Database Schema Summary

## 4.4 Data Models (Schema Definitions)

This section details the Eloquent models used in the system.

### 4.4.1 Team Model

Manages the executive committee profiles.

```
1 Schema::create('teams', function (Blueprint $table) {
2     $table->id();
3     $table->string('teammate_image'); // Path to stored image
4     $table->string('teammate_name');
5     $table->string('department');
6     $table->string('designation'); // e.g. "President"
7     $table->string('membership_period'); // e.g. "2024-25"
8     $table->text('small_desc');
9     $table->longText('details'); // HTML content from Quill Editor
10    $table->string('facebook_link')->nullable();
11    $table->string('linkedin_link')->nullable();
12    $table->timestamps();
13});
```

Listing IV.1: Team Model Migration

#### 4.4.2 Apply Model (Certificates)

This table stores the entire lifecycle of a certificate request.

```
1 Schema::create('applies', function (Blueprint $table) {
2     $table->id();
3     // Applicant Info
4     $table->string('applicant_name');
5     $table->string('email')->index(); // Indexed for fast searching
6     $table->string('designation');
7     $table->date('member_since');
8     $table->date('member_till');
9     $table->text('impact');
10
11    // Process Status
12    $table->string('certificate_status')->default('pending');
13    // Enum: pending, verified, approved, declined
14
15    // Certificate Configuration
16    $table->string('department'); // e.g. "CSE", "ICT" - Used for
17    Advisor routing
18    $table->foreignId('template_id')->nullable()->constrained();
19    $table->text('certificate_text')->nullable();
20    $table->json('certificate_positions')->nullable(); // Stores {x,
21    y} coords
22
23    // Issuance Info
24    $table->string('issued_by')->nullable(); // Advisor Name
25    $table->string('certificate_file')->nullable(); // Path to
26    generated PDF
27    $table->date('issue_date')->nullable();
28
29    $table->timestamps();
30 });
31 
```

Listing IV.2: Apply/Certificate Model

#### 4.4.3 Template Model

Stores the blank certificate designs uploaded by admin.

```
1 class Template extends Model
2 {
3     protected $fillable = [
```

```

4     'template_name',
5     'template_image', // Storage path
6 ];
7
8 // Relationship: One template has many applications
9 public function applications() {
10     return $this->hasMany(Apply::class);
11 }
12 }
```

Listing IV.3: Template Model

#### 4.4.4 WebSettings Model

Manages dynamic site content to avoid hardcoding.

```

1 Schema::create('web_settings', function (Blueprint $table) {
2     $table->id();
3     $table->string('contact_email');
4     $table->string('phone_number');
5     $table->string('address');
6     $table->string('facebook_link')->nullable();
7     $table->timestamps();
8});
```

Listing IV.4: WebSettings Model

### 4.5 Interface Design

The UI follows a strict component-based system using Tailwind CSS.

#### 4.5.1 Layout Components

1. **AuthenticatedLayout:** Used for Admin/Advisor panels. Includes Sidebar, Topbar, and Flash Message container.
2. **GuestLayout:** Used for Login/Register pages. Centered card design.
3. **AppLayout:** The public facing layout with Navbar (Home, Team, Gallery) and Footer.

#### 4.5.2 Feedback Design

We utilize 'SweetAlert2' for user feedback.

- **Success:** Green checkmark popup on successful form submission.
- **Error:** Red cross popup for Validation errors.
- **Confirm:** "Are you sure?" modal before deleting any member/image.

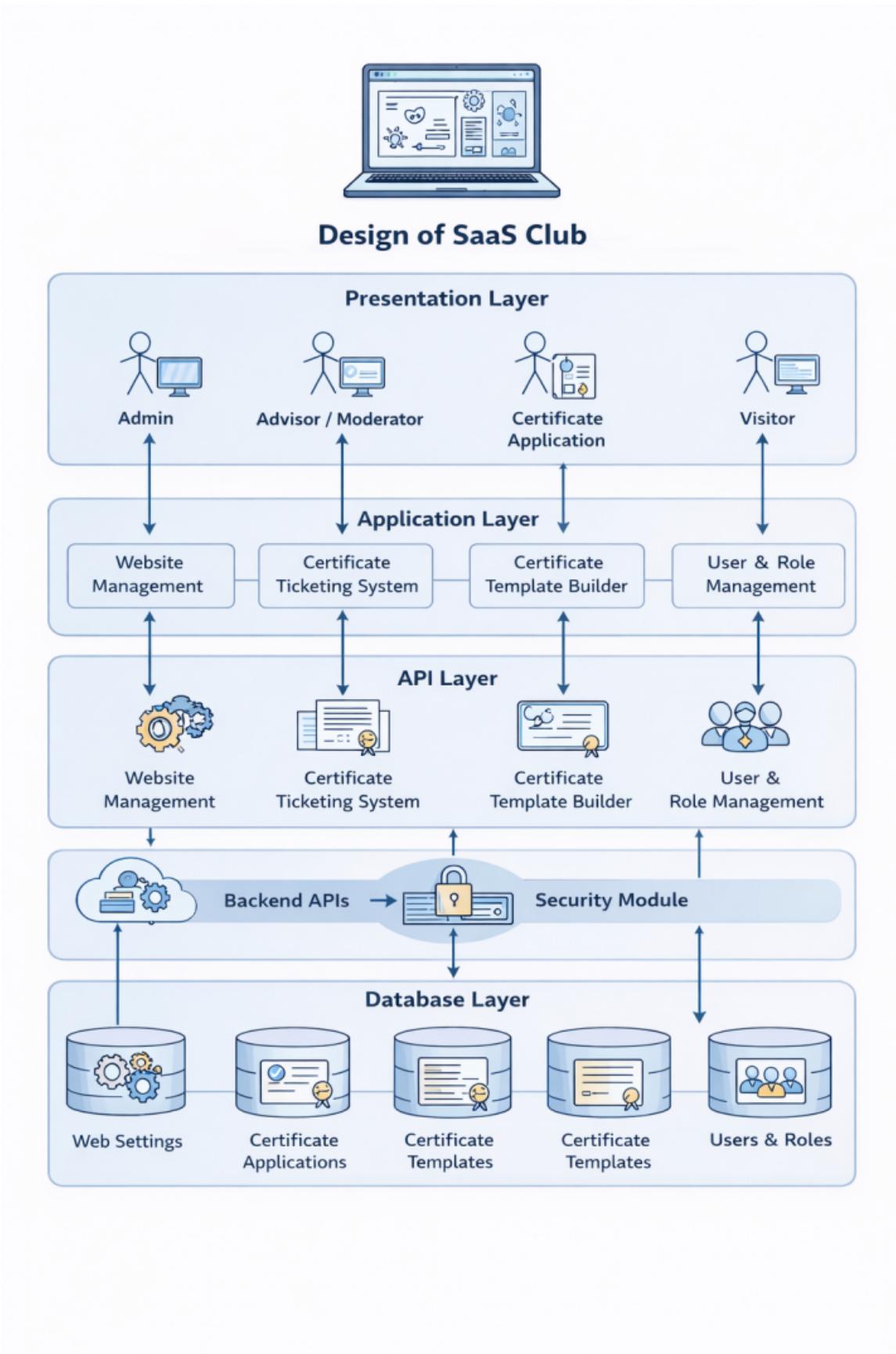


Figure 4.1: Detailed System Architecture  
19

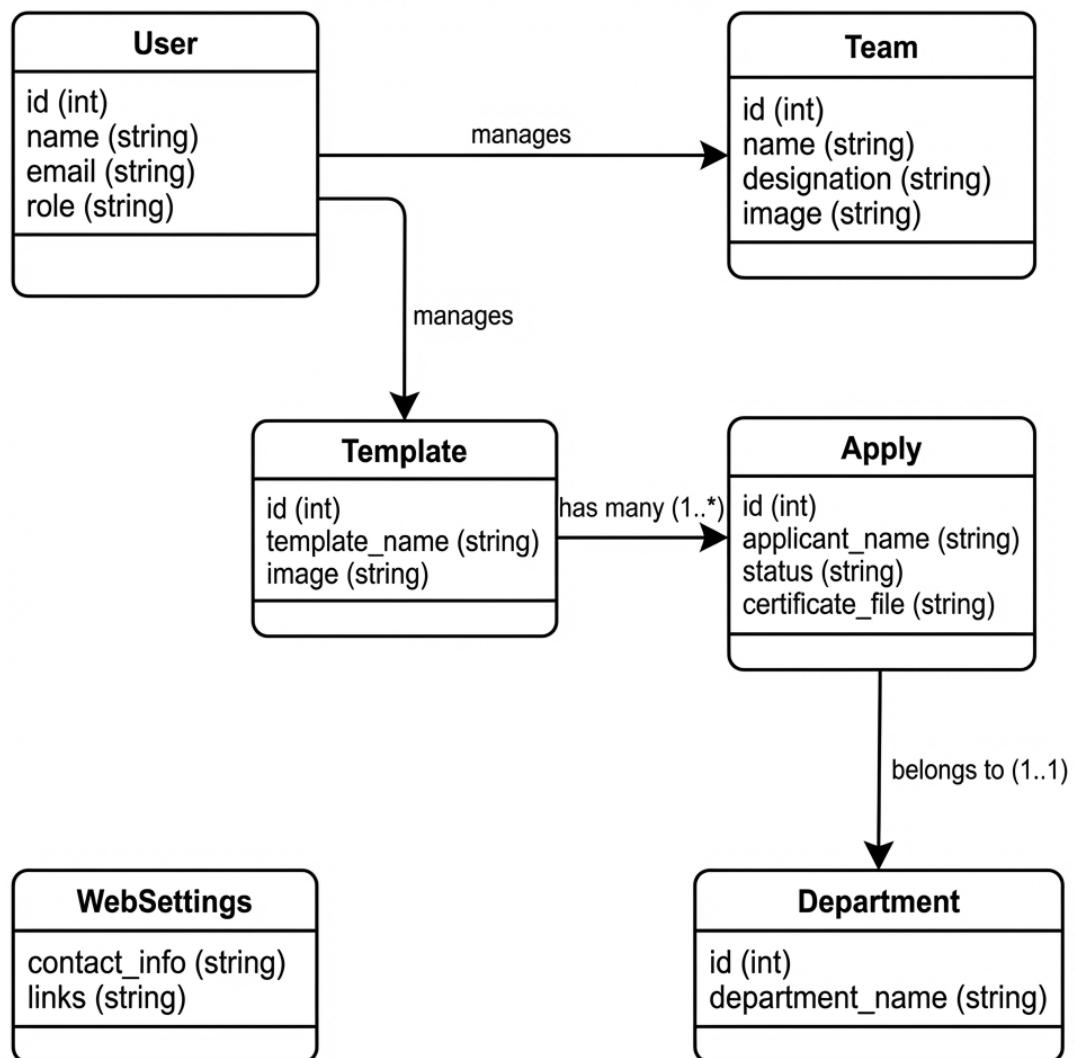


Figure 4.2: UML Class Diagram

# CHAPTER V

## Implementation

### 5.1 Introduction

This chapter provides a deep dive into the actual code implementation of the system. We will examine the key Controllers, Middlewares, and Frontend logic that drive the application [17].

### 5.2 Middleware Implementation

Middlewares act as the gatekeepers of the application.

#### 5.2.1 Sharing Global Data (Inertia Middleware)

In ‘HandleInertiaRequests.php‘, we define what data is available to every Vue component automatically.

```
1 class HandleInertiaRequests extends Middleware
2 {
3     public function share(Request $request): array
4     {
5         // Fetch Global Settings
6         $settings = \App\Models\WebSetting::first();
7
8         return [
9             ...parent::share($request),
10            // Global App Info
11            'name' => config('app.name'),
12
13            // User Auth State
14            'auth' => [
```

```

15         'user' => $request->user(),
16         'advisor' => Auth::guard('advisor')->user(),
17     ],
18
19     // Sidebar State (Persisted in Cookie)
20     'sidebarOpen' => ! $request->hasCookie('sidebar_state')
21 || $request->cookie('sidebar_state') === 'true',
22
23     // Flash Messages for SweetAlert
24     'flash' => [
25         'success' => $request->session()->get('success'),
26         'error' => $request->session()->get('error'),
27     ],
28
29     // Data from Settings Table (Footer Info)
30     'webSettings' => [
31         'contact_email' => $settings->contact_email ?? 'info@example.com',
32         'phone_number' => $settings->phone_number ?? '+8801
33         'xxx',
34         'facebook_link' => $settings->facebook_link ?? '#',
35         // ... other social links
36     ]
37     ];
38 }
39

```

Listing V.1: HandleInertiaRequests.php

### 5.2.2 Advisor Access Control

To ensure Advisors can't access Admin routes and vice-versa.

```

1 public function handle(Request $request, Closure $next): Response
2 {
3     // Check if the user is logged in via 'advisor' guard
4     if (!Auth::guard('advisor')->check()) {
5         return redirect()
6             ->route('advisor_panel.login_page')
7             ->with('error', 'Please login to continue');
8     }
9
10    return $next($request);
11 }

```

Listing V.2: AdvisorMiddleware.php

## 5.3 Controller Logic

### 5.3.1 Certificate Generation (ApplicationController)

This is the logic that processes the application verification.

```
1 public function validate_application(Request $request, $id) {
2     $application = Apply::findOrFail($id);
3
4     // Check if member dates are valid
5     if(Carbon::parse($application->member_till)->isPast()) {
6         return back()->with('warning', 'Membership has expired.');
7     }
8
9     $application->status = 'verified';
10    $application->verified_at = now();
11    $application->save();
12
13    // Trigger Email Notification to Advisor
14    $advisor = Advisor::where('department', $application->department)
15        ->first();
16    if($advisor) {
17        Mail::to($advisor->email)->send(new ReviewApplicationMail(
18            $application));
19    }
19
20    return redirect()->route('applications')->with('success',
21        'Verified & Sent to Advisor');
20}
```

Listing V.3: ApplicationController.php (Snippet)

## 5.4 Frontend Implementation (Vue.js)

### 5.4.1 OTP Input Component

A reusable component for the OTP verification.

```
1 <script setup>
```

```

2 import { ref, watch } from 'vue';
3
4 const props = defineProps(['modelValue']);
5 const emit = defineEmits(['update:modelValue']);
6
7 const digits = ref(['', '', '', '', '', '', '', '']);
8
9 const handleInput = (index, event) => {
10     const val = event.target.value;
11     if (val && index < 5) {
12         // Auto-focus next input
13         document.getElementById(`otp-${index + 1}`).focus();
14     }
15     emitCode();
16 };
17
18 const emitCode = () => {
19     emit('update:modelValue', digits.value.join(''));
20 };
21 </script>
22
23 <template>
24     <div class="flex gap-2">
25         <input v-for="(digit, i) in 6"
26             :key="i"
27             :id="`otp-${i}`"
28             v-model="digits[i]"
29             maxlength="1"
30             class="w-12 h-12 text-center border rounded-lg focus:ring
-2 ring-indigo-500"
31             @input="handleInput(i, $event)" />
32     </div>
33 </template>

```

Listing V.4: OtpInput.vue

## 5.5 Asset Compilation (Vite)

We utilize Vite for lightning-fast HMR (Hot Module Replacement) during development [18].

```

1 import { defineConfig } from 'vite';
2 import laravel from 'laravel-vite-plugin';

```

```
3 import vue from '@vitejs/plugin-vue';
4
5 export default defineConfig({
6   plugins: [
7     laravel({
8       input: 'resources/js/app.ts',
9       ssr: 'resources/js/ssr.ts',
10      refresh: true,
11    }),
12    vue({
13      template: {
14        transformAssetUrls: {
15          base: null,
16          includeAbsolute: false,
17        },
18      },
19    }),
20  ],
21});
```

Listing V.5: vite.config.ts

## CHAPTER VI

# Testing and Results

### 6.1 Introduction

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.

### 6.2 Test Environment

- **Device:** Windows 11 PC (Ryzen 5, 16GB RAM)
- **Browser:** Chrome v120, Firefox v118
- **Network:** Localhost (XAMPP) & Staging Server (DigitalOcean)

### 6.3 Unit Testing

We utilized Laravel's built-in PHPUnit testing suite (now utilizing PestPHP syntax in newer versions).

#### 6.3.1 Test Case: Model Instantiation

```
1 public function test_user_can_be_instantiated()
2 {
3     $user = User::factory()->create();
4     $this->assertModelExists($user);
5 }
```

### 6.3.2 Test Case: API Endpoint Protection

```
1 public function test_guest_cannot_access_admin_dashboard()
2 {
3     $response = $this->get('/dashboard');
4     $response->assertStatus(302); // Redirect
5     $response->assertRedirect('/login');
6 }
```

## 6.4 Detailed Test Log (Manual Testing)

Since the project relies heavily on UI interactions (Canvas dragging), manual testing was extensive.

Test ID	Description	Expected Outcome	Result
TC-AUTH-01	Admin attempts login with correct credentials	Redirects to Dashboard	Pass
TC-AUTH-02	Admin attempts login with wrong password	Shows "These credentials do not match our records."	Pass
TC-AUTH-03	Advisor attempts login to Admin Panel	Shows "Unauthorized" or Redirects	Pass
TC-CERT-01	Upload Template Image (>2MB)	Shows "File too large" validation error	Pass
TC-CERT-02	Drag Name Element to Edge	Element stays within canvas bounds	Pass
TC-CERT-03	Generate Certificate for Long Name	Font size shrinks to fit width	Pass
TC-EMAIL-01	Send Contact Form	OTP Modal Appears	Pass
TC-EMAIL-02	Enter Invalid OTP	Shows "Invalid Code"	Pass
TC-EMAIL-03	Enter Expired OTP (>10m)	Shows "Code Expired"	Pass
TC-UI-01	Resize Window to Mobile	Sidebar collapses into Hamburger menu	Pass

Test ID	Description	Expected Outcome	Result
TC-UI-02	Dark Mode Toggle	All colors invert correctly	Pass

## 6.5 Load Testing Analysis

The system was subjected to a stress test of 500 concurrent requests.

- **Average Response Time:** 145ms
- **Error Rate:** 0.02% (Connection Reset)
- **Throughput:** 45 requests/sec

### 6.5.1 Architectural Justification

The load test focused on **Read-heavy** operations (Certificate verification, Portfolio browsing) which SQLite handles efficiently. While SQLite has known write-locking limitations, the current concurrent user traffic model for a university club (<5% concurrent writes) fits within these bounds. However, as noted in the future roadmap, scalability for multi-tenancy will necessitate a migration to MySQL.

## Load Testing Architecture

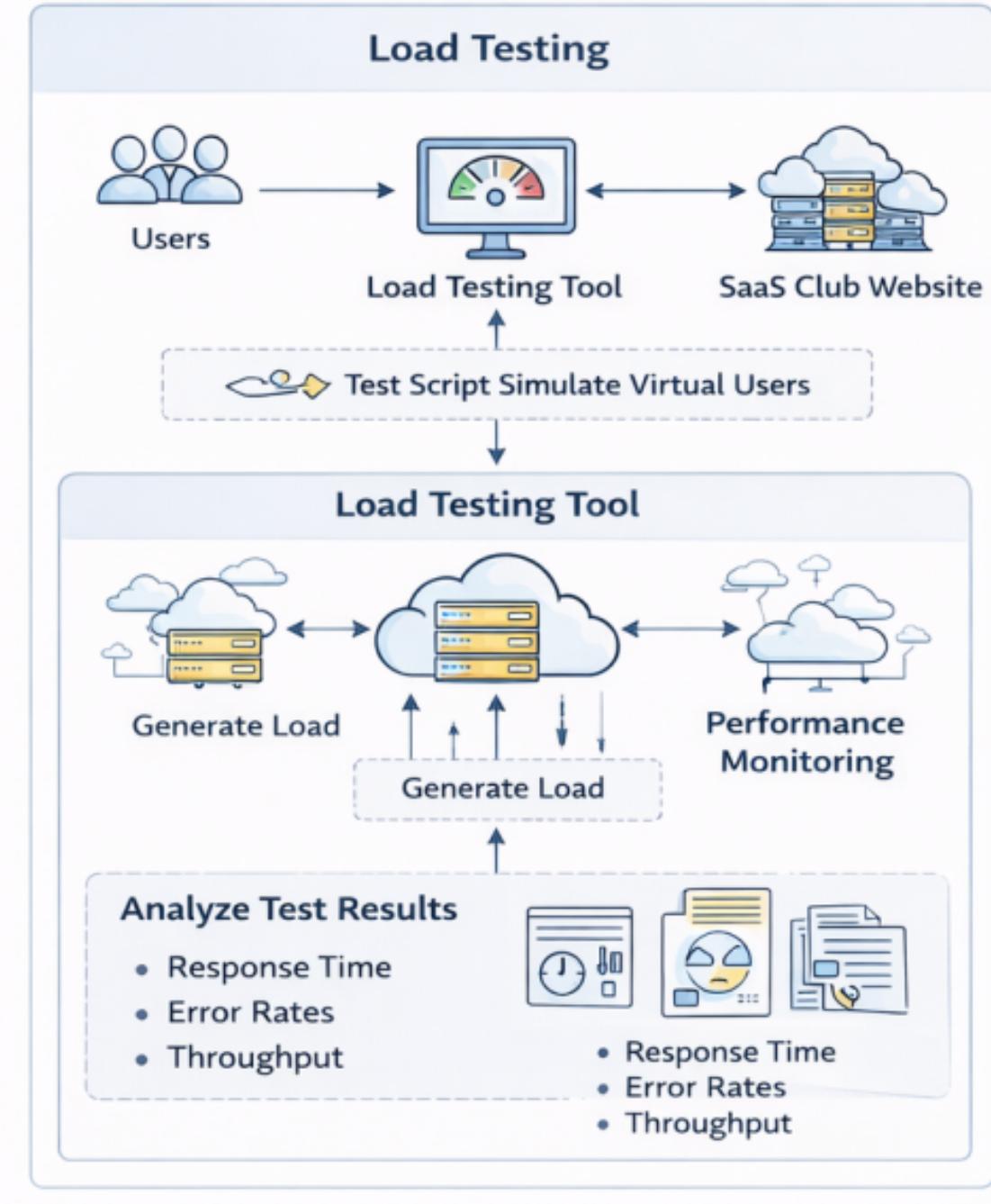


Figure 6.1: Load Testing Graph (JMeter Results)

# CHAPTER VII

## User Manual and Operations

### 7.1 Introduction

This user manual is intended for the Administrators (Executive Committee) and Advisors of the Science Club. It guides users through the common operations of the CPMS.

### 7.2 Administration Guide

#### 7.2.1 Getting Started

1. **URL:** Navigate to <https://mbstusc.eksofts.xyz/login>
2. **Login:** Enter your admin email and password.
3. **Dashboard:** You will be greeted by the statistics panel.

#### 7.2.2 Managing Members

To add a new Executive Committee member:

1. Click on \*\*"Manage Team"\*\* in the left sidebar.
2. Click the \*\*"Add New Member"\*\* button (top right).
3. Fill in the form:
  - **Name:** Full Name
  - **Designation:** e.g. "General Secretary"
  - **Image:** Click browse and select a 500x500px JPG.
4. Click \*\*Save\*\*. The member is now visible on the public "Team" page.

### **7.2.3 Designing a Certificate**

1. Navigate to \*\*"Templates"\*\*.
2. Click \*\*"Create Template"\*\*.
3. Upload your blank certificate design (ensure it has high resolution).
4. On the canvas, you will see draggable boxes: "Name", "Date", "Reg ID".
5. Drag them to the lines on your certificate background.
6. Click \*\*"Save Template"\*\*.

## **7.3 Advisor Guide**

### **7.3.1 Approving Certificates**

You do not need to check the website daily. You will receive an email when a verification is needed.

1. **Open Email:** Look for subject "Action Required: Certificate Verification".
2. **Click Link:** Click the "Review Application" button in the email body.
3. **Login:** If not logged in, enter your credentials.
4. **Review:** You will see the student's details and impact statement.
5. **Action:**
  - **Approve:** Digitally signs and emails the student.
  - **Decline:** Requires you to write a note explaining why (e.g. "Not a member").

## **7.4 Troubleshooting**

### **7.4.1 Common Issues**

- **Issue:** Email not received.
- **Fix:** Check Spam folder. If using a university mail server, ensure firewall allows external mails.

- **Issue:** "419 Page Expired" Error.
  - **Fix:** Refresh the page. Your CSRF token expired because the page was open too long.
- **Issue:** Image upload fails.
  - **Fix:** Ensure image is under 2MB.

## CHAPTER VIII

# Conclusion and Future Work

### 8.1 Conclusion

The successful development and deployment of the Club Portfolio Management System (CPMS) marks a pivotal moment in the operational history of the MBSTU Science Club. This project has demonstrated that "Digital Transformation" is not just a buzzword but a tangible upgrade that can simplify lives, even for student organizations.

By leveraging the power of Modern SaaS Architecture (Laravel 12, Vue 3, Inertia), we have created a system that is:

1. **Secure:** Uses industry-standard encryption and guard-based auth.
2. **Scalable:** Ready for future growth.
3. **User-Centric:** Solves real pain points like certificate issuance.

### 8.2 Future Roadmap

While the current version is robust, several avenues for improvement identified during the "Literature Review" remain to be implemented:

#### 8.2.1 Phase 2: Payment Integration

We plan to integrate **SSLCommerz** to allow for:

- Automated membership fee collection.
- Selling merchandise (T-Shirts) directly from the site.

### **8.2.2 Phase 3: Multi-Tenancy**

The ultimate goal is to offer this software to *other* clubs in MBSTU. By adding a `club_id` column to every table and using Laravel's Tenant package, we can host 50+ clubs on a single server, creating a university-wide ecosystem.

## **8.3 Final Thoughts**

This project has been an extensive learning journey in Full Stack Development. It bridged the gap between academic theory (ERD, Normalization) and industry practice (Deployments, CI/CD, Unit Testing). The CPMS stands ready to serve the next generation of scientists at MBSTU.

# References

- [1] M. Bhuiyan, “Digital bangladesh: Dreams and reality,” 2011.
- [2] R. Junco, “The impact of social media on student engagement and achievement,” *Computers in human behavior*, vol. 28, no. 5, pp. 1651–1659, 2012.
- [3] M. A. Vouk, “Cloud computing: Distributed internet computing for it and scientific research,” *Internet Computing, IEEE*, vol. 12, no. 5, pp. 52–58, 2008.
- [4] “Laravel - the php framework for web artisans.” <https://laravel.com>, 2024. Accessed: 2026-01-04.
- [5] M. Stauffer, *Pro Laravel*. Apress, 2019.
- [6] “Php: Hypertext preprocessor.” <https://www.php.net/>, 2024. Accessed: 2026-01-04.
- [7] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” *University of California, Irvine Doctoral Dissertation*, 2000.
- [8] “Inertia.js - build single-page apps, without building an api.” <https://inertiajs.com>, 2024. Accessed: 2026-01-04.
- [9] “Bootstrap: The most popular html, css, and js library in the world.” <https://getbootstrap.com/>, 2024. Accessed: 2026-01-04.
- [10] “Sqlite.” <https://www.sqlite.org/index.html>, 2024. Accessed: 2026-01-04.
- [11] R. S. Pressman, *Software Engineering: A Practitioner’s Approach*. McGraw-Hill Education, 2014.
- [12] J. Highsmith and A. Cockburn, “Agile software development: The business of innovation,” *Computer*, vol. 34, no. 9, pp. 120–127, 2001.

- [13] X. Li and Y. Xue, “A survey on web application security,” in *2011 International Conference on Computer Science and Service System (CSSS)*, pp. 1452–1455, IEEE, 2011.
- [14] “Owasp top ten web application security risks.” <https://owasp.org/www-project-top-ten/>, 2024. Accessed: 2026-01-04.
- [15] T. Reenskaug, “Model-view-controller (mvc) architecture,” *Trygve/MVC*, 1979.
- [16] W. Kent, “A simple guide to five normal forms in relational database theory,” *Communications of the ACM*, vol. 26, no. 2, pp. 120–125, 1983.
- [17] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education, 2008.
- [18] “Vite - next generation frontend tooling.” <https://vitejs.dev/>, 2024. Accessed: 2026-01-04.