

CSE 6334 Software Requirements Engineering

Project Part 1: SRS

Group :7

|  |  |
| --- | --- |
| Name | Student ID |
| Akid Syazwan bin Nor Azman Shah | 1211111238 |
| Teng Chay Xuan | 1231300802 |
| Kishen Kumar A/L Kumaran | 1211111528 |
| Shannon Fernandez | 1211106748 |

[1. Introduction 2](#_Toc593712604)

[1.1. Purpose 2](#_Toc547128626)

[1.2. Scope 2](#_Toc1846523540)

[1.3. Product overview 2](#_Toc974279657)

[1.3.1. Product perspective 2](#_Toc220913615)

[1.3.2. System Interfaces 3](#_Toc921746270)

[1.3.2.1. User Interfaces 4](#_Toc1219612713)

[1.3.2.2. Hardware Interfaces 4](#_Toc1578971537)

[1.3.2.3. Software Interfaces 5](#_Toc422776210)

[1.3.2.4. Communications Interfaces 5](#_Toc1794806024)

[1.3.2.5. Memory Constraints 6](#_Toc766968505)

[1.3.2.6. Operations 6](#_Toc1218351796)

[1.3.2.7. Site Adaptation Requirements 7](#_Toc1315596726)

[1.3.3. Product function 8](#_Toc1508122033)

[1.3.4. User characteristics 9](#_Toc97163019)

[1.3.5. Limitations 10](#_Toc732388437)

[1.3.5.1. Definitions 11](#_Toc1781823731)

[2. References 11](#_Toc2013459632)

[3. Requirements 12](#_Toc165841433)

[3.1. Functions 12](#_Toc1025203263)

[3.1.1. Functions Activity/Sequence Diagrams 14](#_Toc792298593)

[3.1.1.1. Login Activity Diagram 14](#_Toc576772180)

[3.1.1.2. Register to System Activity Diagram 14](#_Toc1117896575)

[3.1.1.3. Join Cub Activity Diagram 15](#_Toc2020229716)

[3.1.1.4. View Upcoming Events Activity Diagram 15](#_Toc787497534)

[3.1.1.5. RSVP for Events Activity Diagram 16](#_Toc1799167962)

[3.1.1.6. Manage Club Members Activity Diagram 16](#_Toc1767927195)

[3.1.1.7. Create Event Proposal Sequence Diagram 17](#_Toc125103530)

[3.1.1.8. Submit Budget Proposal Activity Diagram 17](#_Toc495967733)

[3.1.1.9. Submit Venue Booking Proposal Activity Diagram 18](#_Toc1707492181)

[3.1.1.10. Cancel Event Sequence Diagram 18](#_Toc1132308361)

[3.1.1.11. View RSVP for Event Activity Diagram 19](#_Toc1317472047)

[3.1.1.12. Submit Event Report Activity Diagram 19](#_Toc1170895946)

[3.1.1.13. Monitor Club and Event Activity Activity Diagram 20](#_Toc2013758345)

[3.1.1.14. Generate Report Activity Diagram 21](#_Toc1624562411)

[3.2. Performance requirements 22](#_Toc1012693057)

[3.3. Usability requirements 22](#_Toc1782475299)

[3.4. Interface requirements 23](#_Toc8236276)

[3.4.1. System interface 23](#_Toc1541040277)

[3.4.2. User interface 23](#_Toc1855723155)

[3.4.3. Hardware interface 24](#_Toc707094393)

[3.4.4. Software interface 24](#_Toc853502440)

[3.4.5. Communication interface 25](#_Toc937686120)

[3.5. Logical database requirements 25](#_Toc2115898021)

[3.6. Design constraint 26](#_Toc251244530)

[3.7. Software system attributes 27](#_Toc736232357)

[3.7.1. Reliability 28](#_Toc427644232)

[3.7.2. Availability 28](#_Toc188139751)

[3.7.3. Security 28](#_Toc1766317026)

[3.7.4. Maintainability 29](#_Toc185700913)

[3.7.5. Portability 29](#_Toc2104143062)

[3.8. Supporting information 29](#_Toc2046845173)

[3.8.1. Sample Input/Output Formats 29](#_Toc1787554231)

[3.8.2. Supporting or Background Information 30](#_Toc1124719369)

[3.8.3. Problem Statement (What the Software Solves) 30](#_Toc903410943)

[4. Verification 31](#_Toc505186512)

[4.1. Verification approach. 31](#_Toc668461648)

[4.2. Verification criteria 32](#_Toc1021136910)

[4.2.1. Unit Testing Verification Criteria 32](#_Toc1821856440)

[4.2.2. Integration testing verification criteria 33](#_Toc2039206351)

[4.2.3. Functional testing verification criteria 34](#_Toc1840943998)

[4.2.4. User acceptance testing verification criteria 35](#_Toc1708496785)

[4.2.5. Performance verification criteria 36](#_Toc1770289614)

[4.2.6. Usability verification criteria 37](#_Toc2068382693)

[5. Appendices 37](#_Toc317993945)

[5.1. Assumptions and Dependencies 37](#_Toc1401719131)

# Introduction

## Purpose

The purpose of this document is to define the software requirements for the Student Club Management System. This document outlines the intended functionality, user interfaces, constraints, and system interactions for developers, testers, and stakeholders.

## Scope

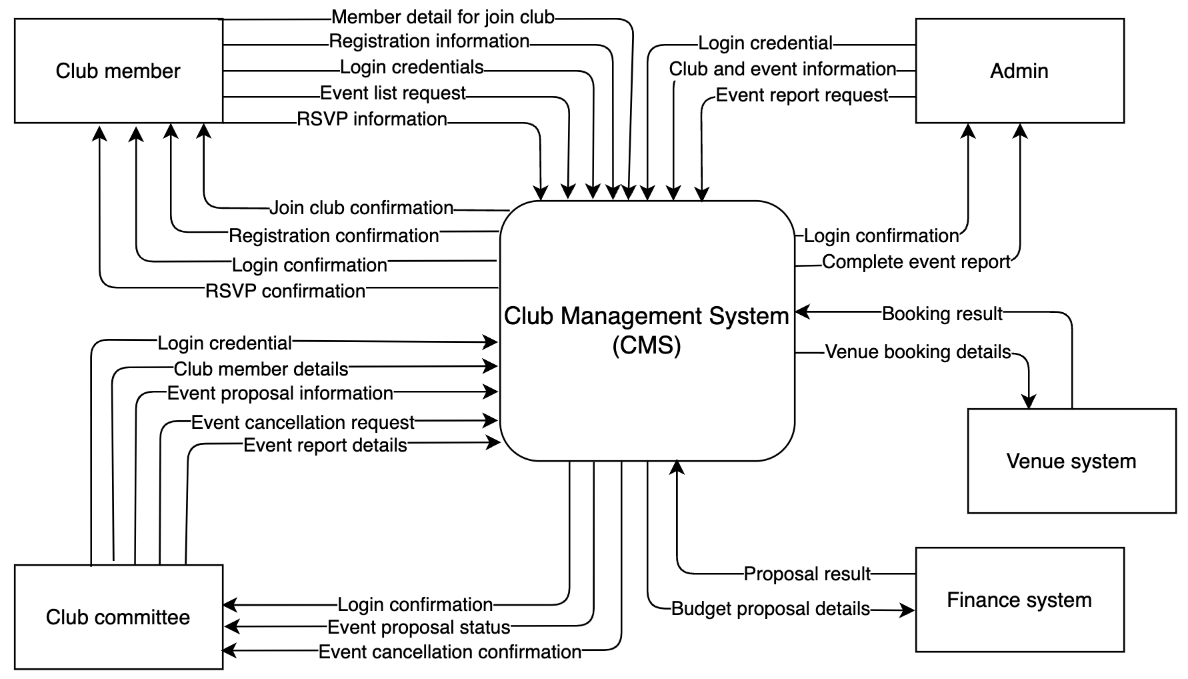
The Student Club Management System will serve as a centralized digital platform that facilitates the administration of student clubs, allowing for effective member management, event coordination, financial tracking, and venue booking. It aims to increase efficiency, transparency, and accessibility for students and university administrators.

## Product overview

The Student Club Management System (CMS) with Budget and Venue integration is designed to serve as a centralized platform for managing the various activities of student club within the university. The aim of this system is to make necessary task like club membership registration, event planning, budget submission and approval and venue booking more efficient. The platform will also integrate with the university space reservation and finance management system to facilitate easier departmental cooperation

### Product perspective

The Club Management System (CMS) is a web-based solution that function as a centralized platform for overseeing student club activities at the university. It aims to facilitate smooth communication among different parties including student, club committee, external system like financial and venue system. This system is a part of the university ecosystem and connect with the institutional platform to enhance administrative workflows and enrich user experience. Figure 1.3.1 below show the context diagram of the system.



*Figure 1.3.1 CMS context diagram*

### System Interfaces

External Systems and Interactions:

|  |  |
| --- | --- |
| **System** | **Functionality** |
| University Financial System | Submit and retrieve status of budget proposals. |
| Venue Booking System | Submit venue booking requests and get confirmation or denial. |
| University Authentication | Authenticate users and assign roles (e.g., student, committee, admin). |

#### User Interfaces

The Student Club Management System will include intuitive user interfaces customized by role:

|  |  |
| --- | --- |
| **User Role** | **Interface Description** |
| Club Member | Dashboard to register for clubs, view events, and RSVP. |
| Club Committee | Panel for managing members, submitting proposals, and tracking events. |
| Administrator | Admin portal for monitoring activity, generating reports, and approving requests. |

**UI Characteristics:**

* Mobile-responsive design making it easy for students to access on the go.
* Compliant with the university’s digital accessibility and branding standards.
* Easy navigation with a top menu, dashboard tiles, and modal forms for quick interactions.

#### Hardware Interfaces

The Student Club Management System will run on cloud infrastructure and does **not** directly interface with hardware devices. However, it must be compatible with:

|  |  |
| --- | --- |
| **Hardware Type** | **Requirement** |
| Desktop / Laptop | Browsers: Chrome, Firefox, Safari, Edge (latest 2 versions). |
| Mobile Devices (iOS/Android) | Responsive UI must work on all screen sizes |

#### Software Interfaces

The Student Club Management System requires integration with the following external and internal software systems:

|  |  |  |
| --- | --- | --- |
| **Software Product** | **Purpose** | **Source** |
| University Financial System | Budget request processing and approval workflows | Finance Department |
| Venue Booking System | Room reservation and calendar sync | Campus Facilities Unit |
| University Login System | Secure login and session management | IT Department |
| PostgreSQL Database | Backend data persistence | Open Source |
| Node.js (Backend Runtime) | Server-side logic | Open Source |

#### Communications Interfaces

The Student Club Management System will utilize the following communication interfaces:

* **Protocol**: HTTPS over TLS 1.3
* **Internal Communication**: REST APIs between frontend and backend components
* **External Communication**:
  + **University Authentication Service** via OAuth2 SSO
  + **Financial System API** via secure REST API endpoints
  + **Venue Reservation System** via authenticated REST API
* **Email Notifications**: SMTP protocol with university mail server (student.mmu.edu.my) for alerts and confirmations (event approval, RSVP status)

**Security Notes**:

* All communications are encrypted end-to-end.
* APIs require token-based authentication using OAuth2 bearer tokens.

#### Memory Constraints

The Student Club Management System will adhere to the following memory constraints:

* **Primary Memory (RAM)**:
  + Minimum server RAM: 4 GB
  + Recommended: 8 GB for production environment with caching enabled
* **Secondary Memory (Storage)**:
  + Minimum disk space: 50 GB for application, logs, and file uploads
  + Backup: minimum of 30 days
* **Database**:
  + PostgreSQL data volume expected to grow with number of clubs and members. This will require periodic archiving to be implemented in the system.

#### Operations

The Student Club Management System is designed to support both **interactive** and **automated** operations, including:

**a) Modes of Operation:**

* **User-Initiated**:
  + Club committee submits event and budget proposals.
  + Admin approves or rejects requests.
  + Students RSVP to events.
* **Scheduled/Automated**:
  + Daily sync with venue and finance systems.
  + Email Notification (daily or real-time).

**b) Operation Timing:**

* **Interactive Use**: 24/7 user access from web interface.
* **Unattended Use**:
  + Nightly backups
  + Log cleanup
  + API-based data sync during off-peak hours

**c) Support Functions:**

* Data validation
* Duplicate checks for event names, venues, and schedules
* Conflict detection for venue booking

**d) Backup & Recovery:**

* Full database backup every 24 hours.
* Recovery procedures documented and tested quarterly.
* Incremental backups every 6 hours.
* Logs retained for 30 days minimum.

#### Site Adaptation Requirements

The Student Club Management includes some site-specific customization requirements:

**a) Initialization Sequences:**

* Institution-specific data:
  + Club categories
  + Venue types
  + Approval workflows
* Safety constraints for budget caps and event duration rules are configurable per institution.

**b) Configurable Features:**

* Branding (logo, colour scheme, footer text)
* Notification rules (e.g., when to send reminders)
* Admin roles and permissions can be adapted based on the university’s organizational hierarchy

### Product function

|  |  |
| --- | --- |
| **Stakeholder** | **Function** |
| **Club Member** | Register to system |
| Login to system |
| Join the club |
| View upcoming events |
|  | RSVP for events |
| **Club Committee** | Login to system |
| Manage club members |
| Create event proposal |
| Submit budget proposal |
| Submit venue booking proposal |
| Cancel event |
| Submit event report |
| **Admin** | Login to system |
| Monitor club and event activity |
| Generate reports |
| **Financial Department Staff** | View, approve or decline budget proposal |
| **Venue Booking Staff** | View, approve or decline venue booking proposals |

### User characteristics

There are several key user roles in Club Management System (CMS), including club member, club committee, admin, finance and venue officer. The following table shows the expected level of knowledge for each user.

System internal user:

|  |  |  |
| --- | --- | --- |
| Role | Description | Required knowledge |
| Club member | A student who participates in the university club | Basic knowledge of using the CMS to register, view events and RSVP in the events. |
| Club committee | A leader in the club who will manage the club activities and submission | Knowledge on how to manage club information, submit the event proposal include venue and finance request, and submit event report. |
| Admin | University staff that responsible for managing and oversee the overall club activities in the university. | Knowledge in user management, activity monitoring, generating report in the system |

System external user:

|  |  |  |
| --- | --- | --- |
| Role | Description | Knowledge |
| Finance staff | University staff who handle the financial matters | Ability to track budget, review and approve/reject club application, and update financial status within the CMS. |
| Venue staff | University staff who manage venue booking and scheduling. | Knowledge in venue booking and scheduling and maintaining venue usage record in CMS. |

### Limitations

Club Management System (CMS) provide an efficient platform for managing student club activities, there are several limitations as well that may affect its functionality and performance, these limitations are outlined in the table below:

|  |  |
| --- | --- |
| Limitations | Description |
| Limited to university club only | The system developed specifically for university club only, external organization cannot use the platform. |
| Venue booking constraints | The venue viewing/booking depend on the integration with the university venue system, if the university system is offline, the information will be inaccurate. |
| No mobile application | The CMS is a web-based application and may not be fully optimized for mobile device. A dedicated mobile application is not available at this moment. |
| Cannot use without internet | The CMS is a web-based application which needed internet connection to use. Location with weak or no internet connection difficult to use it. |

#### Definitions

|  |  |
| --- | --- |
| Definition | Description |
| CMS | Club Management System (CMS) is a software application design to manage and support university club activities. It allows user to register, submit proposal, manage membership, and track event participation. |
| Club member | A registered university student who joined the club through the official enrolment process. |
| Club Committee | A member elected or appointed to a leadership position in the club. |
| Event | An organized activity planned by a club that requires submission of proposal including the financial resource, venue booking and university approval. |
| Admin | University staff that responsible for managing and oversee the overall club in the university. |
| Finance Module | An integrated part of the CMS that handle financial processes. |
| Venue Module | An integrated part of the CMS that handle venue booking processes. |

# References

This document is prepared in reference to the following documents:

1. IEEE. (2018). ISO/IEC/IEEE 29148:2018 Systems and software engineering— Life cycle processes— Requirements engineering.

<https://www.iso.org/standard/72089.html>

1. M. Hariprasad, N. N, N. Dey, P. D and R. Kumar P, "College Club Activity Management System," 2023 7th International Conference on Computation

System and Information Technology for Sustainable Solutions (CSITSS),

Bangalore, India, 2023, pp. 1-5, doi: 10.1109/CSITSS60515.2023.10334208.

1. Wiegers, K., & Beatty, J. (2013). Software requirements (3rd ed.). Microsoft Press.

# Requirements

## Functions

In this section, the key function of the Club Management System (CMS) is outlined. This function will represent the core activities that user can perform in the system. The diagram 3.1 below is the use case diagram which show overview of how different user can interact with the system and table 3.1 show the detailed of each function.

A diagram of a student club management system

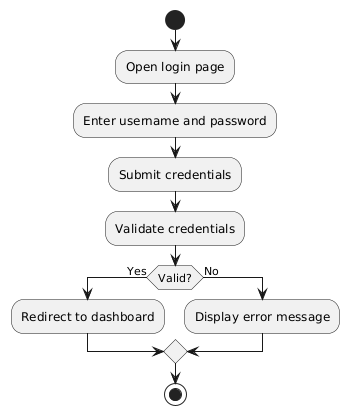
AI-generated content may be incorrect.*Figure 3.1 Use case diagram*

|  |  |  |
| --- | --- | --- |
| **Function Code** | **Function Name** | **Description** |
| F001 | Login to system | Allows club members, committee and system admin to access the system using their credentials. |
| F002 | Register to system | Enables students to register in the system. |
| F003 | Join club | Enable student to join the available club in university |
| F004 | View Upcoming Events | Member can request to displays a list of upcoming events organized by clubs. |
| F005 | RSVP for Events | Allows members to RSVP for events they are interested in attending. |
| F006 | Manage Club Members | Enables committee members to manage member lists. |
| F007 | Create Event Proposal | Committee can draft a proposal for a new event. |
| F008 | Submit Budget Proposal | Committee submits budget requests to the finance system for approval. |
| F009 | Submit Venue Booking Proposal | Committee sends a booking request to reserve a venue through the venue system. |
| F010 | Cancel Event | Committee can cancel planned events due to various reasons. |
| F011 | View RSVP for event | Committee can view the RSVP result in the system for the event |
| F012 | Submit Event Report | After an event, committee submits a report with results and feedback. |
| F013 | Monitor Club and Event Activity | Admin can track all club and event activities through the system. |
| F014 | Generate Reports | Admin generates event and activity reports for recordkeeping or analysis. |

*Table 3.1 List of function*

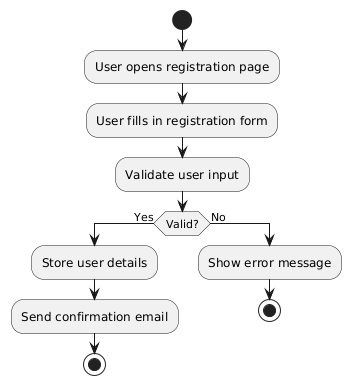
### Functions Activity/Sequence Diagrams

#### Login Activity Diagram



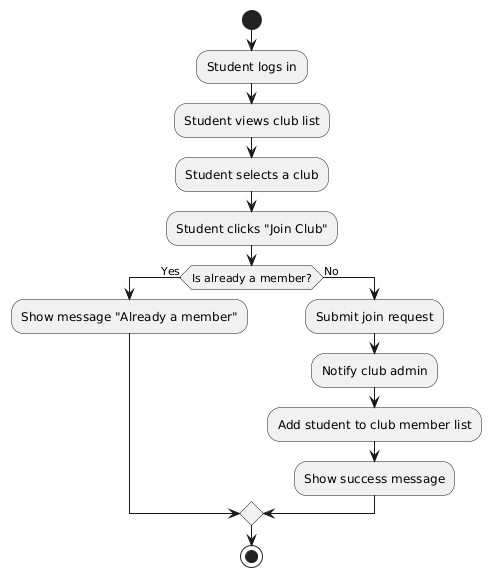
*Figure 3.1.1.1 Login Activity Diagram*

#### Register to System Activity Diagram



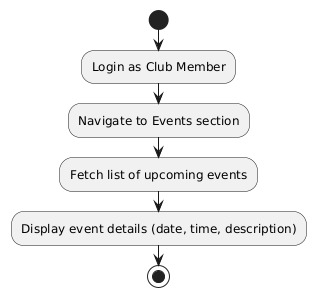
*Figure 3.1.1.2 Register to System Activity Diagram*

#### Join Cub Activity Diagram



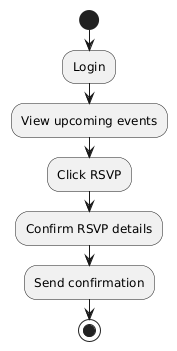
*Figure 3.1.1.3 Join Club to System Activity Diagram*

#### View Upcoming Events Activity Diagram



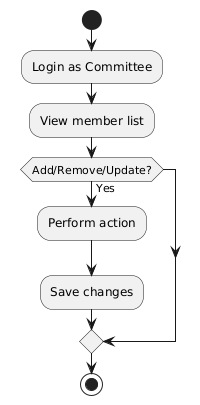
*Figure 3.1.1.4 View Upcoming Events Activity Diagram*

#### RSVP for Events Activity Diagram



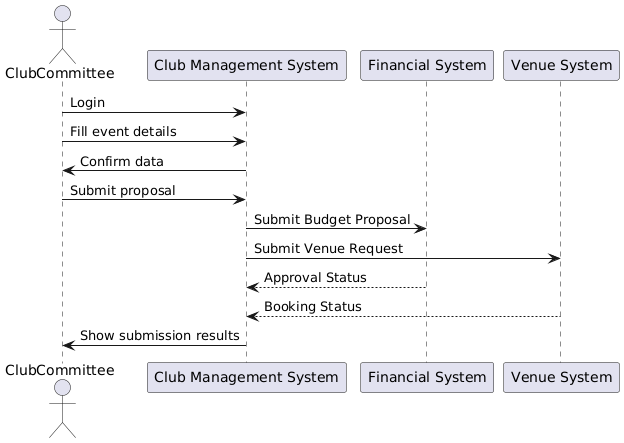
*Figure 3.1.1.5 RSVP for Events Activity Diagram*

#### Manage Club Members Activity Diagram



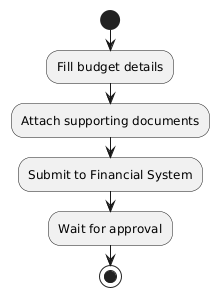
*Figure 3.1.1.6 Manage Club Members Activity Diagram*

#### Create Event Proposal Sequence Diagram



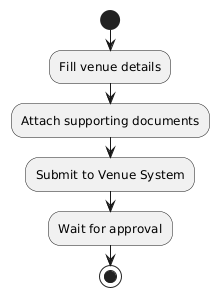
*Figure 3.1.1.7 Create Event Proposal Sequence Diagram*

#### Submit Budget Proposal Activity Diagram



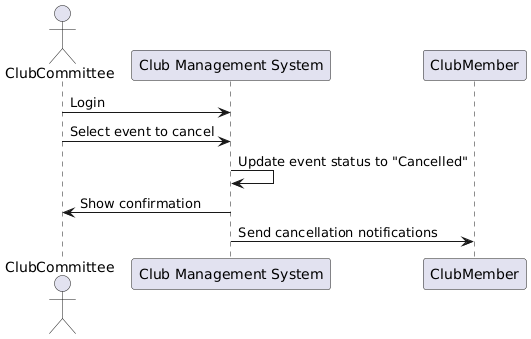
*Figure 3.1.1.8 Submit Budget Proposal Activity Diagram*

#### Submit Venue Booking Proposal Activity Diagram



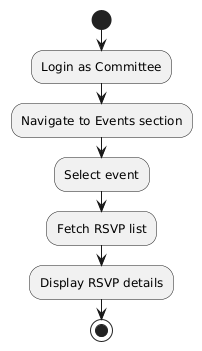
*Figure 3.1.1.9 Submit Venue Booking Proposal Activity Diagram*

#### Cancel Event Sequence Diagram



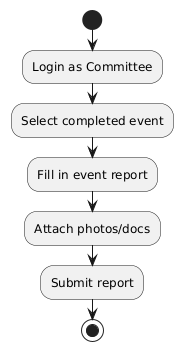
*Figure 3.1.1.10 Cancel Event Sequence Diagram*

#### View RSVP for Event Activity Diagram



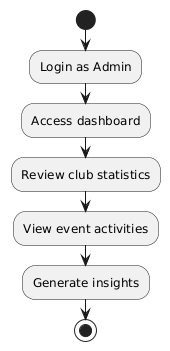
*Figure 3.1.1.11 View RSVP for Event Activity Diagram*

#### Submit Event Report Activity Diagram



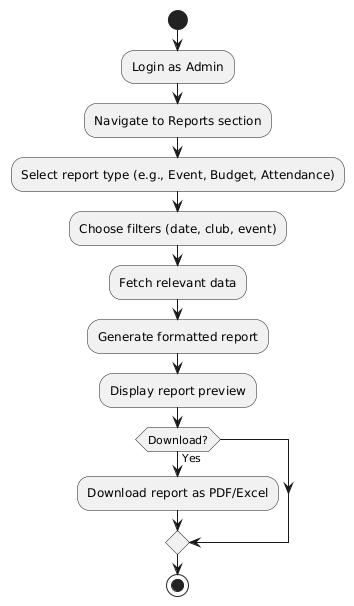
*Figure 3.1.1.12 Submit Event Report Activity Diagram*

#### Monitor Club and Event Activity Activity Diagram



*Figure 3.1.1.13 Monitor Club and Event Activity* *Activity Diagram*

#### Generate Report Activity Diagram



*Figure 3.1.1.14 Generate Report Activity Diagram*

## Performance requirements

|  |  |  |
| --- | --- | --- |
| Requirement ID | Description | Priority |
| REQ\_P001 | The system should load all dashboard components in 3 seconds for 90% of users | High |
| REQ\_P002 | Event RSVP submission shall process within 2 seconds during peak usage. | High |
| REQ\_P003 | Budget proposal submission to finance system should receive response within 5 seconds | Medium |
| REQ\_P004 | Venue booking function shall synchronize with external venue system within 5 second. | Medium |
| REQ\_P005 | Report for up to 1000 record shall generate within 10 second. | Low |

## Usability requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Description | Metric | Priority |
| REQ\_U001 | User should complete registration/login within 3 clicks from the homepage | ≤ 3 clicks | High |
| REQ\_U002 | Club committee shall submit event proposal within 3 minutes (average) | ≤ 3 minutes | High |
| REQ\_U003 | Member should complete RSVP to the event within 5 clicks | ≤ 3 clicks | Medium |
| REQ\_U004 | 90% of users shall rate the system’s ease of use as ≥4/5 on a post-training survey. | Survey score | High |

## Interface requirements

### System interface

|  |  |  |
| --- | --- | --- |
| **System** | **Interface Type** | **Benefits** |
| **University Authentication System** | OAuth2 / SAML | Ensures secure access with Single Sign-On (SSO), reducing login friction and improving user trust. |
| **Financial Management System** | REST API | Enables automated submission and real-time tracking of budget proposals, reducing manual paperwork. |
| **Venue Reservation System** | REST API | Facilitates efficient venue booking by checking availability and automating approvals. |

### User interface

Key UI Features:

* Top Navigation Bar: Provide access to the Dashboard, Club, Upcoming Events and Profile.
* **Role-based Dashboards**:
  + **Members**: See available clubs, RSVP to events.
  + **Committee**: Submit event proposals, manage members, track budgets.
  + **Admins**: Approve proposals, monitor reports.
* **Forms**:
  + Event proposal form (fields for title, date, description, budget, venue)
  + Member registration form
* **Interactive Components**:
  + Event calendar
  + RSVP buttons
  + Notifications panel

The interface will follow modern UI/UX principles and be accessible on desktop and mobile devices.

### Hardware interface

The system should support the following hardware-related considerations:

* **Client Devices**:
  + Laptops and desktops (Windows, macOS, Linux)
  + Mobile devices (iOS, Android) with modern browsers
* **Server Infrastructure**:
  + Deployed on cloud or university-hosted virtual machines
  + Supports standard networking, monitoring, and backup operations

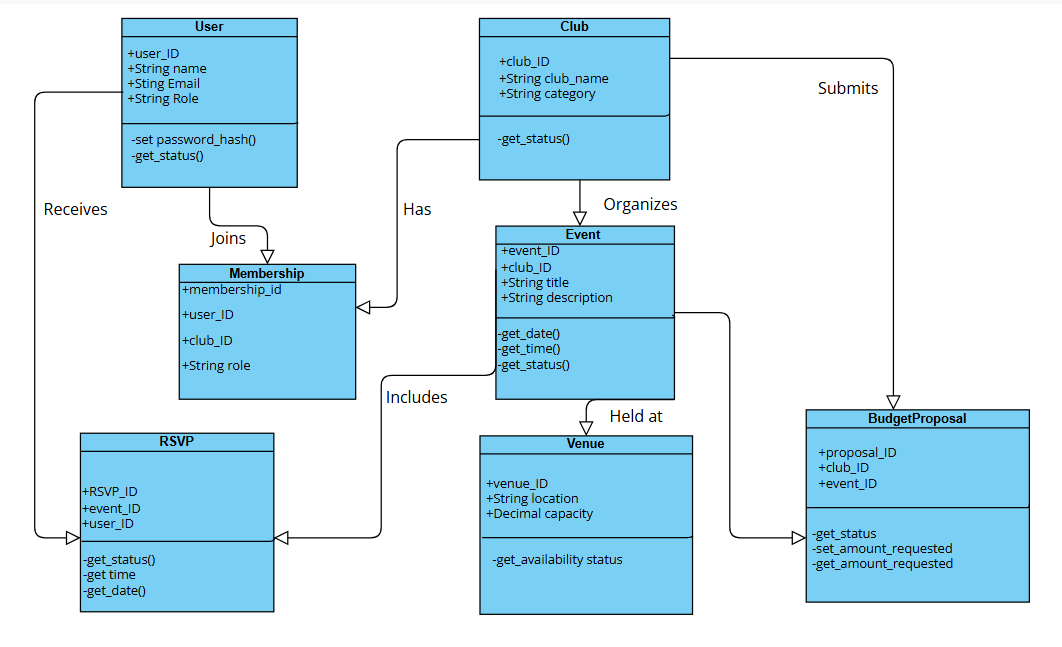
### Software interface

|  |  |  |
| --- | --- | --- |
| **Software Product** | **Version/Spec** | **Benefits** |
| University Financial System | v3.2+ (API) | Enables efficient, automated processing of budget requests with real-time status tracking. |
| Venue Booking System | v2.5 (API) | Reduces scheduling conflicts and manual coordination by providing up-to-date room availability and booking integration. |
| University Login System | OAuth2, SAML | Enhances security and user convenience through centralized authentication and role-based access control. |
| PostgreSQL Database | 13+ | Provides reliable, scalable, and performant storage of system data including users, events, and proposals. |
| Node.js (Backend Runtime) | 18.x | Offers efficient handling of asynchronous tasks and API requests, contributing to fast and responsive system performance. |

### Communication interface

|  |  |  |
| --- | --- | --- |
| **Interface** | **Protocol** | **Use** |
| Web communication | HTTPS (TLS 1.3) | For secure API communication and frontend/backend interaction |
| API endpoints | REST (JSON) | For financial system, venue system, and internal APIs |
| Notifications | SMTP (TLS) | For sending email alerts on budget/venue decisions |

## Logical database requirements



Relationships:

* A **User** can be a member of many **Clubs**; a **Club** has many **Users** → *many-to-many* via **Membership**.
* A **Club** can organize multiple **Events**; each **Event** belongs to one **Club** → *one-to-many*.
* Each **Event** can have multiple **RSVPs** by **Users** → *one-to-many*.
* Each **Budget Proposal** is linked to one **Club** and optionally to one **Event**.
* An **Event** is held at one **Venue**; a **Venue** may host multiple **Events** over time.

Constraints:

* **Unique Constraints** on email (User), club name (Club), and event name per date (Event).
* **Foreign Keys** between related entities to ensure referential integrity.
* **Check Constraints** for status fields (get\_status ('pending', 'approved', 'rejected')).

## Design constraint

The design of the Student Club Management System is subject to several constraints arising from institutional policies, technical environments, and regulatory compliance.

Technical Constraints:

* **Authentication** must use the university’s existing **OAuth2 based SSO system**.
* The system must be developed using **open-source technologies** (e.g., Node.js, PostgreSQL).
* **RESTful API architecture** must be followed for external integrations (Financial and Venue systems).
* The web interface must be **responsive** and usable on standard browsers (Chrome, Firefox, Safari, Edge).

Institutional Constraints:

* The system must comply with **university IT governance policies**, including data privacy and audit requirements.
* Only **authorized personnel** (admins, committee members) may access approval or modification features.

Regulatory/Standards Constraints:

* Must comply with **data protection regulations** (e.g., FERPA or GDPR equivalents in your jurisdiction).
* All student records and event data must be **encrypted in transit (TLS 1.3)** and at rest.

UI/UX Constraints:

* Interface design must follow the **university branding guidelines** including logo, colours, and font standards.
* Navigation must be simple and consistent across all modules.

## Software system attributes

This section specifies the key quality attributes expected from the **Student Club Management System (SCMS)** and their implementation-related requirements.

### Reliability

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** | **Author** |
| REQ\_SCMS\_001 | SCMS shall recover from unexpected crashes within 60 seconds without data loss. | High | Project Team |

### Availability

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** | **Author** |
| REQ\_SCMS\_002 | SCMS shall maintain at least 99.9% uptime during weekdays from 8 AM to 6 PM. | High | Project Team |

### Security

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** | **Author** |
| REQ\_SCMS\_003 | SCMS shall implement role-based access control for all user operations. | High | Project Team |
| REQ\_SCMS\_004 | SCMS shall encrypt all sensitive data using TLS 1.3 during transmission and AES-256 at rest. | High | Project Team |

### Maintainability

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** | **Author** |
| REQ\_SCMS\_005 | SCMS code shall follow standard coding guidelines and include inline documentation. | Medium | Project Team |
| REQ\_SCMS\_006 | SCMS shall achieve at least 80%-unit test coverage for critical modules. | Medium | Project Team |

### Portability

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** | **Author** |
| REQ\_SCMS\_007 | SCMS shall run on both Linux and Windows environments using Docker containers. | High | Project Team |
| REQ\_SCMS\_008 | SCMS shall not include any OS-specific file paths or dependencies. | Medium | Project Team |

## Supporting information

### Sample Input/Output Formats

|  |  |  |
| --- | --- | --- |
| **Function** | **Sample Input** | **Expected Output** |
| Club Registration | Name: Robotics Club  Advisor: Prof. A. Kumar | Club created successfully. Status: “Pending" |
| Event Proposal | Title: Hackathon 2025  Date: 2025-11-10  Venue: Auditorium | Event created. Budget submission link generated. Status: Draft |
| RSVP Submission | Event ID: 121112526  User ID: 801 | "RSVP confirmed. Check your university email for the event invite." |
| Budget Request | Club: AI Society  Amount: $450  Justification: Workshop equipment | "Budget proposal submitted. Awaiting finance approval." |

### Supporting or Background Information

* The university currently manages student clubs using a combination of email, Google Forms, and paper-based budget forms.
* There is no existing unified system for event, budget, and member tracking — causing delays, miscommunication, and duplication.
* The SCMS will serve as the first centralized platform for these tasks, integrating with the university's finance and venue APIs.
* Reference Materials:
  + - Student Affairs Handbook
    - University Financial Workflow Documentation
    - Venue Booking Policy Manual

### Problem Statement (What the Software Solves)

The current method of managing student club activities is fragmented and inefficient:

* Event proposals are submitted manually via forms and emails, leading to inconsistent approvals and lost requests.
* Budget submissions are often delayed due to missing paperwork or manual routing.
* Club membership rosters are not centralized, making it hard to validate participation or assign privileges.
* Venue booking conflicts occur due to lack of real-time visibility.

The SCMS addresses these issues by:

* Automating workflows for proposals, approvals, and bookings.
* Maintaining a real-time dashboard for all stakeholders.
* Centralizing membership and role management per club.
* Reducing administrative load through streamlined integrations and audit trails.

# Verification

This section will describe the approach and criteria used to verify the University Club Management System (CMS) meets its functional and non-functional requirements. Verification ensures that the developed system will meet the stakeholder's expectation, complete and perform according to the specification

## Verification approach.

The verification of CMS will be conducted using various software testing technique throughout the development lifecycle. The table below outlines the verification method, responsible parties, timeline and location for each of the technique. Table 4.1 below show the approach will be used.

Table 4.1 Verification approach list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Verification technique** | **Description** | **Responsible parties** | **Timeline** | **Location** |
| **Unit Testing** | Verifies individual modules function as intended. | Developer | Week 4 – Week 7 | Developer environment |
| **Integration Testing** | Ensures interaction between modules works as expected. | Developers, QA Tester | Week 7 – Week 9 | Staging/test environment |
| **Functional Testing** | Validate that each function (F001–F013) meets the requirements. | QA team, project supervisor | Week 9 – Week 10 | University test lab |
| **System Testing** | Assesses the entire system behaviour, including performance and security. | QA team | Week 10 – Week 11 | Controlled lab/test server |
| **User Acceptance Testing (UAT)** | Real users will test the system to confirm it meets real-world need. | End-users, QA team | Week 11 – Week 12 | Campus lab or virtual environment |

## Verification criteria

### Unit Testing Verification Criteria

|  |  |  |
| --- | --- | --- |
| **Module** | **Test Case** | **Expected Output** |
| Login | Validate correct username/password | Return true, user logged in |
|  | Validate incorrect password | Return false, login fails |
| Registration | Register new user with valid inputs | Return success, user stored in DB |
|  | Register with missing student ID | Throw validation error |
| Event module | Create event with valid data | Event object created and stored |
| RSVP module | Submit RSVP with valid member ID and event ID | RSVP saved; confirmation returned |
| Report module | Generate report for 1000 records | Returns report object, size ≤ 10s |

### Integration testing verification criteria

|  |  |  |
| --- | --- | --- |
| **Integrated Modules** | **Test Case** | **Expected Outcome** |
| Registration + login | New user registers and logs in immediately | Login succeeds post-registration |
| Event creation + RSVP | User RSVP to an event created minutes earlier | RSVP linked to correct event |
| Budget proposal + finance system | Submit budget, receive acknowledgment from external API | Response = "Proposal Received" in external finance system |
| Venue booking + venue system | Book venue via UCMS, view confirmation in external system | Booking marked “Reserved” externally within 5 second |
| Event report + report generator | Submit report and generate monthly summary | System summarizes and includes the latest event reports |

### Functional testing verification criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Code** | **Feature** | **Test Case** | **Expected Outcome** |
| F001 | Login | Log in as a member with valid credentials | Redirected to member dashboard |
| F002 | Registration | Register with valid student ID and MMU email | Account created successfully |
| F003 | Join Club | Join a club for student role | Successfully join the available club |
| F004 | View Upcoming Events | Member views upcoming events | List shows all upcoming events |
| F005 | RSVP for Events | RSVP to an event as a logged-in user | RSVP recorded and confirmation shown |
| F006 | Manage Club Members | Committee removes a member from the club list | Member removed from list |
| F007 | Create Event Proposal | Committee submits event title, date, objective | Event proposal stored in system |
| F008 | Submit Budget Proposal | Submit budget request of RM500 for approval | Status shown as "Pending Approval" |
| F009 | Submit Venue Booking Proposal | Send venue request for May 30 in Lecture Hall A | Venue status = “Requested” |
| F010 | Cancel Event | Committee cancels an event scheduled for next week | Event marked as cancelled; participants notified |
| F011 | View RSVP for event | Committee checks RSVP for an upcoming event | List of attendees displayed |
| F012 | Submit Event Report | Submit post-event report including feedback | Report stored and status = "Submitted" |
| F013 | Monitor Club and Event Activity | Admin views current event and activity logs | Dashboard shows updated info |
| F014 | Generate Reports | Admin generates yearly club activity report | Downloaded PDF includes summary for the year |

### User acceptance testing verification criteria

|  |  |  |
| --- | --- | --- |
| **User Role** | **UAT Scenario** | **Acceptance Criteria** |
| Club Member | Login, RSVP for event, view upcoming events | All actions complete without errors, easy to navigate |
| Club Committee | Register, create event, submit budget and venue request, submit report | Workflow can be completed in under 10 minutes; responses within SLA |
| Admin | Monitor club activity, generate reports, cancel an event | Data accurate, reports correctly generated |
| All Users | Use system for 30 minutes and rate usability | ≥90% users rate usability ≥4/5 |

### Performance verification criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Verification Criteria** | **Priority** |
| REQ\_P001 | Load dashboard within 3 seconds for 90% of users | 90% of test users must experience ≤3s dashboard loading time. | High |
| REQ\_P002 | Process RSVP submissions within 2 seconds during peak usage | RSVP action must be confirmed in ≤2s with 100 concurrent users. | High |
| REQ\_P003 | Submit budget proposal and receive response in ≤5 seconds | Budget system responds to proposal submission within 5 seconds. | Medium |
| REQ\_P004 | Venue booking synchronizes with external system in ≤5 seconds | Venue booking request is acknowledged by external system within 5 seconds. | Medium |
| REQ\_P005 | Generate reports for up to 1000 records in ≤10 seconds | Report generation for 1000 records must not exceed 10 seconds. | Low |

### Usability verification criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Verification Criteria** | **Priority** |
| REQ\_U001 | Complete registration/login in ≤3 clicks from homepage | Test users must register/login with ≤3 clicks from homepage. | High |
| REQ\_U002 | Submit event proposal in ≤3 minutes (average) | Committee users should complete event proposal form within 3 minutes. | High |
| REQ\_U003 | RSVP submission requires ≤5 clicks | Test members must complete RSVP in ≤5 clicks. | Medium |
| REQ\_U004 | 90% of users rate ease of use ≥4/5 in post-training survey | Survey results show 90% of users give ≥4/5 for usability. | High |

# Appendices

## Assumptions and Dependencies

This section outlines all necessary conditions assumed to be true during the design and development of the Student Club Management System (CMS), as well as the technical, organizational, and environmental dependencies the system relies on. These factors are critical to ensure the CMS functions as intended and meets user expectations.  
  
A.1 Assumptions

1. **User Authentication and Access**
2. All end-users (students, committee members, admins, staff) possess valid and active university-issued credentials for access via the university's Single Sign-On (SSO) system.
3. Role-based access rights (student, committee, admin, finance staff, venue staff) are properly pre-assigned by the university's identity management system.
4. University staff are responsible for managing and updating user roles and permissions within the authentication system.
5. **Network and Accessibility**
6. All users have regular access to the internet and modern web browsers (latest 2 versions of Chrome, Firefox, Safari, or Edge).
7. The system will be hosted on a cloud-based infrastructure or a university-provided virtual machine with guaranteed uptime of at least 99.5%.
8. Users will interact with the system via web-based interfaces; mobile responsiveness is expected but no native mobile app will be provided during the initial release.
9. **System Environment and Compatibility**
10. All external APIs (financial system, venue booking system, authentication system) are compatible with RESTful communication and return predictable, validated JSON responses.
11. External systems (finance, venue, university SSO) remain stable, maintained, and accessible throughout the lifecycle of CMS.
12. External system outages are rare and will be handled through fallback mechanisms or error messages.
13. **Operational and Policy Assumptions**
14. University regulations on event proposal approval, budget limits, venue usage, and report submissions are clearly documented and will be provided to the development team.
15. Each club has a designated advisor and at least one active committee member responsible for administrative actions (e.g., proposals, reports).
16. All clubs are formally recognized by the university and must follow standardized procedures for funding and venue requests.
17. **Communication and Notification**
18. The university provides access to an SMTP-compatible email server for outbound communications (e.g., RSVP confirmations, budget/venue decisions, password recovery).
19. Notifications and alerts are enabled for all major workflows: event creation, approval, budget response, venue response, RSVP confirmations, and report submissions.
20. **Data and Security Assumptions**
21. The university has an acceptable use policy and privacy guidelines governing the storage and access of student information.
22. All data transmitted between CMS and external/internal systems will use TLS 1.3 encryption.
23. User data at rest (within the database) will be encrypted using industry standards (e.g., AES-256).
24. **Development and Testing Environment**
25. The development team will have sandbox access to test instances of the financial and venue systems.
26. Testing teams (QA, committee volunteers) are available for User Acceptance Testing (UAT) before system deployment.
27. The university will provide or approve test accounts representing all user roles for end-to-end testing.
28. **Backup and Recovery**
29. Scheduled backups (full and incremental) are implemented and managed by the hosting provider or university IT.
30. System recovery plans are validated at least quarterly and include procedures for data restoration in under 2 hours.
31. **Localization and Branding**
32. All interface text will be delivered in English (default) unless requested otherwise.
33. University branding (logos, colours, font styles) will be provided to maintain consistent UI design.

A. **Dependencies**

1. **University Authentication System**

* Type: OAuth2-based Single Sign-On (SSO)
* Role: Secure user login and access control
* Dependency: CMS depends on this system to validate users and assign permissions; any downtime in SSO affects all user access.

1. **Financial Management System**

* Type: REST API-based budget processing tool
* Role: Receive and respond to club budget proposals
* Dependency: Budget submission workflow in CMS relies on this system for real-time approval/rejection and tracking.

1. **Venue Booking System**

* Type: REST API integration
* Role: Submit venue reservation requests, receive confirmation or rejection
* Dependency: CMS depends on this system to process and reflect accurate venue availability; outdated or unavailable venue data can result in double bookings or failed reservations.

1. **University Email Server (SMTP)**

* Type: SMTP server (e.g., smtp.student.mmu.edu.my)
* Role: Send all automated emails (event updates, proposal status, confirmation)
* Dependency: Without this, users will not receive system-generated notifications, alerts, or confirmation emails.

1. **PostgreSQL Database**

* Type: Relational Database
* Role: Store all persistent data (users, clubs, events, proposals, reports)
* Dependency: All system modules depend on this database; failure or misconfiguration will halt CMS operations.

1. **Node.js Runtime**

* Type: Server-side JavaScript execution engine
* Role: Executes backend logic and API endpoints
* Dependency: CMS relies on Node.js to process data, communicate with other systems, and execute business logic.

1. **University Governance Policies**

* Type: Organizational/Regulatory
* Role: Define data handling, user access, event regulations
* Dependency: CMS workflows (e.g., proposal approvals, event limits) must align with university rules; policy changes can affect functional logic.

1. Deployment Infrastructure

* Type: Cloud platform or university VM hosting
* Role: Host and run CMS with required resources (CPU, RAM, storage)
* Dependency: Hosting platform must support TLS, scheduled tasks (cron jobs), API gateway routing, and sufficient performance.

1. Browser Standards

* Type: Client-Side Software
* Role: Interface through which users access CMS
* Dependency: CMS assumes compliance with ECMAScript and HTML5 standards; unsupported browsers may cause UI/UX issues.

1. Stakeholder Availability

* Type: Organizational
* Role: Provide input, feedback, UAT participation
* Dependency: Without active participation from students, committees, and staff, requirements validation and usability testing are at risk.

**B. Acronyms and Abbreviations (Exhaustive List)**

|  |  |
| --- | --- |
| Acronym / Abbreviation | Full Form / Description |
| CMS | Club Management System |
| UI | User Interface |
| UX | User Experience |
| SSO | Single Sign-On |
| API | Application Programming Interface |
| HTTPS | Hypertext Transfer Protocol Secure |
| TLS | Transport Layer Security |
| OAuth2 | Open Authorization 2.0 (authentication protocol) |
| SMTP | Simple Mail Transfer Protocol |
| REST | Representational State Transfer (web service style) |
| JSON | JavaScript Object Notation |
| SQL | Structured Query Language |
| DB | Database |
| RAM | Random Access Memory |
| VM | Virtual Machine |
| QA | Quality Assurance |
| UAT | User Acceptance Testing |
| FERPA | Family Educational Rights and Privacy Act (or equivalent local privacy legislation) |
| GDPR | General Data Protection Regulation |
| REQ | Requirement (prefix used in ID numbers, e.g., REQ\_U001) |
| Fxxx | Function Code (e.g., F001: Login, F002: Register) |
| MMU | Multimedia University |
| CRUD | Create, Read, Update, Delete (common database operations) |
| SRS | Software Requirements Specification |
| SRS ID | Unique identifier used to track requirement in SRS (e.g., REQ\_P001) |
| HTTPS | Hypertext Transfer Protocol Secure |
| AES-256 | Advanced Encryption Standard – 256-bit encryption |
| UX Survey | Usability Experience Survey |
| UI/UX | Combined term for user interface and user experience |
| DevOps | Development and Operations (software delivery practice) |
| ESLint | JavaScript code quality tool (used in development, if applicable) |
| SMTP | Simple Mail Transfer Protocol |
| CRUD | Create, Read, Update, Delete (standard operations on data) |
| OAuth | Open Authorization protocol used for secure token-based user authentication |
| JWT | JSON Web Token (used for session management, if applicable) |
| CI/CD | Continuous Integration / Continuous Deployment |
| MVC | Model-View-Controller (software design pattern) |