

**CSE6224 - SOFTWARE REQUIREMENTS ENGINEERING**

**Student Club Management System with Budget and Venue Integration**

**TT5L**

**Group 2**

|  | **ID** | **Name** |
| --- | --- | --- |
| Leader | 1231302923 | Teow Wei Ting |
| Member | 1211112187 | Eng Zi Ying |
| Member | 1211110469 | See Jie Sheng |
| Member | 1221101288 | Sua Wei Khong |

# 

# **Content**

[Content 2](#_yxtjsqs5ctws)

[1.1 Purpose 4](#_5b61cdd6zz3s)

[1.2 Scope 5](#_xlb1i7btsal4)

[1.3 Product Overview 6](#_6xr4i5mb1im3)

[1.3.1 Product Perspective 8](#_wtsiixrw2wj6)

[1.3.2 Product Functions 11](#_llh0wdo7pnvd)

[1.3.3 User Characteristics 14](#_khtp6mbqiccf)

[1.3.4 Limitations 16](#_99dggt3tcom7)

[1.4 Definitions 18](#_lo3lbnrlii0h)

[2.0 References 20](#_a8b2mjrbl7lu)

[3.0 Requirements 21](#_9dbz7zo4buwb)

[3.1 Functions 21](#_6gu66wtwj4k4)

[3.1.1 Register 22](#_he5hngpx5pqw)

[3.1.2 Login 24](#_xo1zqnei0oiy)

[3.1.3 Log Out 26](#_fn9ldzdmzg0w)

[3.1.4 Reset Password 28](#_5u41s9lcdfuh)

[3.1.5 View Profile 31](#_1kqz90f65zd9)

[3.1.6 Update Profile 33](#_oe6sbt5fjcqq)

[3.1.7 Request Create Club 35](#_6apzy88ssz1x)

[3.1.8 Approve Club Request 37](#_zdzt88rtl7kd)

[3.1.9 Request Join Event Committee 40](#_xgpj2w9tuu5x)

[3.1.10 Approve Join Committee Request 42](#_7fdwq5wlrqsh)

[3.1.11. Update Club Details 44](#_6umdds4a6lpa)

[3.1.12. Delete Club 46](#_7wz4hlr7u0t)

[3.1.13. Join Club 48](#_bqrqd5drpxwj)

[3.1.14. View Club List 50](#_c8q9ib89pe5r)

[3.1.15. Request Create Event 52](#_drfli95rd3as)

[3.1.16. Update Event Info 54](#_633p9s5dm6ri)

[3.1.17. Cancel Event 56](#_umodb7gh0en8)

[3.1.18. Renew Membership 58](#_d1jp9qyaqll8)

[3.1.19. View Announcement 60](#_mn34yotql0uf)

[3.1.20 Approve Event Request 62](#_cypkv1bsdbhi)

[3.1.21 Request Venue Booking 64](#_45z1dtl023r6)

[3.1.22 Approve Venue Request 66](#_xwwf6inmkpex)

[3.1.23 View Event Calendar 68](#_jdsp8xne0rpo)

[3.1.24 Submit Budget Proposal 70](#_5xk0g6yrkkjk)

[3.1.25 Approve Budget Proposal 72](#_rv23a5x37ggb)

[3.1.26 View Club Budget 74](#_ihcqw0up006l)

[3.1.27 Generate Financial Report 76](#_qymvirpbd12e)

[3.1.28 View Financial Report 78](#_e7uzuelxgivm)

[3.1.29 Send Club Announcement 80](#_k3c2lwfwobgy)

[3.1.30 Notify User 82](#_e638erufyda9)

[3.1.31 View All Club 84](#_i01mnp5mrkmq)

[3.1.32 Search Club 86](#_9u2339yrd7k5)

[3.1.33 Filter Event 88](#_w7pdlgish98)

[3.1.34 Filter Budget Request 90](#_9tjjamtuqbto)

[3.1.35 Log Activity 92](#_x1pi5c31mvf0)

[3.1.36 View Audit Log 94](#_e18braubpyug)

[3.1.37 Set Permission 96](#_599mrmot5b2f)

[3.1.38 Generate Venue Status Report 98](#_g87h1723b7pd)

[3.1.39 View Venue Status 100](#_hra86bzfhkdz)

[3.2 Performance Requirements 102](#_utblx5vvod6w)

[3.2.1. Dynamic Actions or Changes: 102](#_v6npvbij7n54)

[3.2.2. Quantitative Criteria: 102](#_fuve4wg4a8ic)

[3.2.3. Performance Requirements for Operational Phases and Modes: 103](#_95mzrp2k9291)

[3.3 Usability Requirements 104](#_mdcp6qtvxt3l)

[3.3.1 Performance 104](#_gfv1ugja5aui)

[3.3.2 Security 104](#_injasiw2xy3u)

[3.3.3 Reliability 105](#_vwcios3z3twd)

[3.3.4 Availability 105](#_i3gh2buq6343)

[3.3.5 Maintainability 105](#_nfq3gmfcdrux)

[3.4 Interface Requirements 106](#_2s6m598p83r7)

[3.4.1 System Interfaces 106](#_f64bxbfwm6q)

[3.4.2 User Interfaces 106](#_yb5uopu6see)

[3.4.3 Hardware Interfaces 108](#_5k1jupqoxjp)

[3.4.4 Software Interfaces 108](#_oalnkq5ksfpa)

[3.4.5 Communication Interfaces 109](#_137dmnablxja)

[3.5 Logical Database Requirements 111](#_gadlnuwlksi1)

[3.5.1 User Class 112](#_qfsb3s7lmhrs)

[3.5.2 Club Class 112](#_ghkhfsdclg98)

[3.5.3 Event Class 113](#_iregeho2ca7p)

[3.5.4 BudgetRequest Class 113](#_udd4a0c4lhwf)

[3.5.5 VenueRequest Class 114](#_yos3n3fajzgf)

[3.5.6 Announcement Class 115](#_ojsvuk7bck60)

[3.5.7 Membership Class 115](#_2395znilttmp)

[3.5.8 ClubFinance Class 1](#_bv75xgksz2kb)

[3.6 Design Constraints 1](#_5lzab4wsawfi)

[3.6.1 External Standards and Regulatory Requirements 1](#_xj319pr7afja)

[3.6.2 Project Limitations 1](#_r2hvkzjh8y9u)

[3.6.3 Technical Constraints 1](#_m2ushb3wa2as)

[3.6.4 Platform Constraints 1](#_exqbet7o9u51)

[3.6.5 Environmental Constraints 1](#_ovkeyyho82it)

[3.7 Software System Attributes 1](#_309w2z27r21a)

[3.8 Supporting Information 1](#_lm1d441hqnf9)

[3.8.1 Brainstorming 1](#_2j2qeuymvh72)

[3.8.2 Perspective-based Reading 1](#_p2vejumwrj13)

[3.8.3 Questionnaire 1](#_qv91at82375j)

[4. Verification 1](#_mvjgo09pat8n)

[5. Appendixes 1](#_ca4kdnlm61lp)

[5.1 Assumptions and Dependencies 1](#_3raf62xnvpv3)

[5.2 Acronyms and Abbreviations 1](#_lwqna9u4opt)

**1.0 Introduction**

Student clubs improve campus life by encouraging students to be leaders, collaborate, and interact in the community. However, coordinating club activities, funds, and event logistics may be difficult and time-consuming, frequently necessitating manual processes and communication breakdowns. As institutions grow and student participation increases, there is a greater demand for a centralised digital solution to enable efficient club administration. The Student Club Management System with Budget and Venue Integration addresses this demand by delivering a modern platform adapted to the specific needs of student organisations.

## **1.1 Purpose**

The purpose of the "Student Club Management System with Budget and Venue Integration" is to provide a comprehensive digital solution that centralizes and simplifies the management of student clubs within the university. The goal of this system is to solve common problems that student organisations face, like inadequate communication, manual membership and financial tracking, and trouble organising event logistics and venue reservations. The software integrates with current university financial systems and space reservation tools, allowing clubs to manage their budgets, submit funding requests, and reserve spaces with minimal administrative friction. Along with offering a user-friendly interface for students to take part in club activities, attend events, and stay informed, it also promotes transparency in financial management and decision-making processes. Lastly, the system aims to promote a more active and organised student life environment, lessen administrative burdens for university staff and student leaders, and support the development and success of student-driven projects.

## 

## 

## **1.2 Scope**

The Student Club Management System with Budget and Venue Integration is a comprehensive web-based platform designed to help, automate, and improve the management of student organisations at a university. It strives to centralise different administrative and operational functions that are currently performed manually or across disparate systems. This system will offer a single, unified interface for managing club activities, allowing for effective communication and collaboration among student leaders, members, and university authorities.

Furthermore, the system's core features include club registration and membership administration, which allows administrators to manage member databases, assign positions, and monitor organisational structures.Then the website also facilitates event planning, allowing groups to propose and organise activities through scheduling tools and approval protocols.Moreover, event-related features will include notification systems, and calendar integration to assist decrease schedule conflicts and increase planning accuracy.

Besides, the budget management and money request module, which is integrated with the university's financial system, is an important part of the system. This enables clubs to submit financing proposals, track approvals, monitor expenditures, and ensure transparency in the use of allotted funds. The tool will guarantee that financial management corresponds to university standards, thereby easing contacts between student organisations and the finance office.Additionally, the technology also interfaces with the university venue booking systems, enabling real-time availability checks, space reservation requests, and status tracking. This ensures that event locations are booked in a timely and conflict-free way, reducing unnecessary communication and human coordination. The interface gives both student leaders and administration complete visibility over venue usage on campus.

From the member's perspective, the system provides quick access to club information, event lists, and online resources. Members can explore active clubs, register for events, and receive notifications via personalised dashboards.Therefore , this promotes greater participation and a deeper sense of community among the student group. Other than that, Release 1.0 of the system focuses on delivering the core features that provide immediate value by reducing paperwork, improving coordination, and ensuring compliance. While this release provides the groundwork, future additions could include mobile access, advanced reporting and analytics, deeper third-party connectivity and improved security through customisable role-based permissions. For instance, learning platforms and social media.

Finally, this technology contributes to the university's overall goals of digital transformation, greater operational efficiency, and more student engagement. It is intended to evolve alongside student requirements and technology improvements, establishing itself as a critical instrument in the modern university ecosystem.

## **1.3 Product Overview**

The Student Club Management System (SCMS) is a unified platform designed to simplify and enhance the operations of student clubs and organizations. By integrating with the university's financial management system and campus space reservation database, SCMS provides a centralized solution for membership management, event planning, budget tracking, and venue booking. This system eliminates administrative inefficiencies, ensuring seamless coordination for student-led activities.

Core Functionalities:

1. **Membership Management:** SCMS platform implemented with a centralized database for club members, including roles, permissions, and club checking tools. It automated the membership approvals and renewals for efficiency.
2. **Budget Tracking & Financial Integration:** The platform also has real-time synchronization with the university's financial system for transparent fund allocation and past expense history tracking. This automated the budget requests, approvals, and expenditure reports.
3. **Event Planning & Venue Booking:** The direct integration with the campus space reservation system provides instant venue availability checks and bookings. This provides efficient tracking on venue users with logged requests and approvals on the SCMS platform database.
4. **Unified Communication:** SCMS platform implemented with several networking features for strengthening students' bonding. For example, announcement dashboards, club event committee team, and email or SMS notifications. These collaboration tools shall further enhance the event planning and task delegation within the student club.

Relationship to Existing Systems:

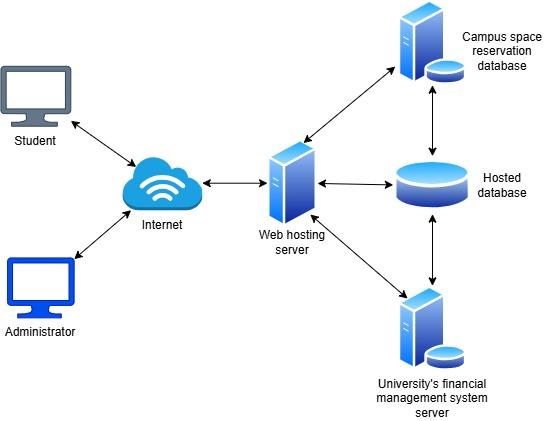
SCMS acts as an intermediary layer between student clubs, university's financial and facility management systems. It retrieves real-time budget data from the financial system and checks venue availability from the space reservation database, ensuring accuracy and efficiency. The platform enhances existing processes without replacing core university systems.

User Interfaces:

* Administrator Interface: University staff can oversee club registrations, budget requests and approvals, and event compliance. Administrator could utilizes the dashboard for tracking club engagement and resources utilization.
* Student Interface: Students can further be promoted into Member, Committee and Club President respectively. Students can register into the SCMS, join clubs and request to create new clubs. Once students join a club, they will be promoted into Member, which has more features such as RSVP for events, view club budgets, and participate in club events via club exclusive announcements. On the other hand, the student will be promoted into Club President once the new club creation request is approved. The Club President can manage members, submit budget requests, book venues, and request to organize events via the exclusive features on their dashboard provided on SCMS. Furthermore, Member can be promoted into Committee once they applied to join committee team in the club event. They are given limited access to help Club President further enhance the management of the club event. For instance, send club announcements, update event info, and manage financial reports.

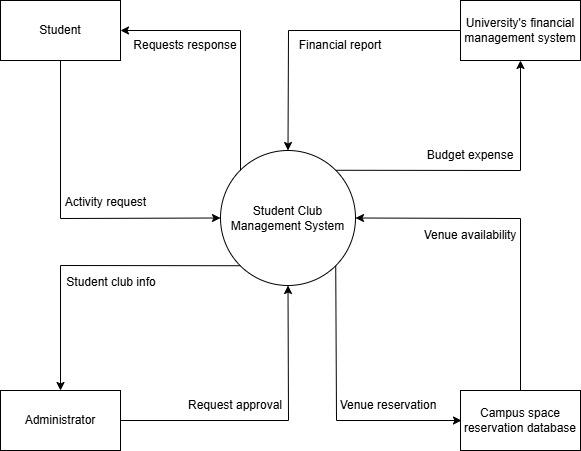
Overall, the Student Club Management System empowers student organizations by automating administrative tasks, improving financial transparency, and optimizing event planning. It helps to foster a more active and well-managed campus community.

### **1.3.1 Product Perspective**

*****Figure 1: Student Club Management System*

As shown in Figure 1 above, the Student Club Management System (SCMS) is a centralized platform designed for three primary users: Student, Administrator, and university backend systems. The SCMS platform will be hosted on an internet-accessible web server. It works by collaborating with the university's existing financial management system server and campus space reservation database.

The hosted database acts as an intermediary for real-time data synchronization between the web server and university's backend systems. By integrating with the university's financial management system, SCMS can rapidly retrieve oldest to latest budget data from the system and summarizes a best view for administrators. On the other hand, campus space reservation databases can store every venue reservation data for efficient tracking purposes.



*Figure 2: Context Diagram of Student Club Management System*

As shown in Figure 2 above, the context diagram illustrates the interactions between the Student Club Management System (SCMS) and its key external entities: Student, Administrator, the University's Financial Management System, and the Campus Space Reservation Database.

Students can submit various activity requests (such as club creation, event proposals, venue bookings, and budget submissions) and receive corresponding responses. Administrators act as system overseers by reviewing and approving these requests, supported by the student club information provided through SCMS.

Additionally, SCMS integrates with the University’s Financial Management System to handle budget tracking, expense reporting, and approval workflows. It also connects to the Campus Space Reservation Database to check venue availability and manage event space bookings in real time.

SCMS serves as a bridge between student organizations and institutional resources, enabling comprehensive management of student clubs. Its core functionalities include membership tracking, event planning, financial oversight, and facility reservation management. The system involves seven primary actors: Student, Member, Committee Member, Club President, Administrator, the University’s Financial Management System, and the Campus Space Reservation Database. The use case diagram below illustrates all function and feature available in SCMS.

### **1.3.2 Product Functions**

The following table features an overview of the list of features in the system along with a brief description, categorized by accessible actors.

| **Function ID** | **Function Name** | **Description** | **Accessible Actor(s)** |
| --- | --- | --- | --- |
| F-001 | Register | Allows users (e.g., students) to register an account | Student |
| F-002 | Login | Authenticates user credentials to access the system | All users |
| F-003 | Log out | Ends the current user session | All users |
| F-004 | Reset Password | Sends reset link and allows users to update password | All users |
| F-005 | View Profile | Allow the user to view their profile information | All users |
| F-006 | Update Profile | Users can update their profile information | All users |
| F-007 | Request Create Club | Allow students to request the creation of a new club | Student |
| F-008 | Approve Club Request | Review and approve club creation requests | Admin |
| F-009 | Request Join Event Committee | Allow a club member to request participation in committee | Member |
| F-010 | Approve Join Committee Request | Allow the president to approve a club member’s request as committee | Club President |
| F-011 | Update Club Details | Edit information about an existing club | Club President |
| F-012 | Delete Club | Remove an existing club from the system | Admin |
| F-013 | Join Club | Allow the student to join the club | Student |
| F-014 | View Club List | Allow the student to view existing club in the system | Student |
| F-015 | Request Create Event | Allow club president to request for create club event | Club President |
| F-016 | Update Event Info | Allow user to update event information | Club President, Committee |
| F-017 | Cancel Event | Cancel a planned or ongoing event | Club President, Committee |
| F-018 | Renew Membership | Allow member to renew club membership | Member |
| F-019 | View Announcement | Allow member to view exclusive announcement of the club | Member |
| F-020 | Approve Event Request | Approve or reject proposed event requests | Admin |
| F-021 | Request Venue Booking | Submit request for venue use | Club President |
| F-022 | Approve Venue Booking | Review and approve venue booking requests | Admin, Campus Space Reservation System |
| F-023 | View Event Calendar | Browse upcoming and scheduled events | Student |
| F-024 | Submit Budget Proposal | Propose a budget for an event or club activity | Club President |
| F-025 | Approve Budget Proposal | Review and approve or reject budget requests | Admin, Financial Management System |
| F-026 | View Club Budget | Monitor club budget, usage history, and balances | Member, Admin |
| F-027 | Generate Financial Report | Produce detailed budget reports in PDF/CSV format | Financial Management System |
| F-028 | View Financial Report | Review previously generated club financial report | Member |
| F-029 | Send Club Announcement | Notify club members with announcements or updates | Club President, Committee |
| F-030 | Notify User | Send alerts or notifications within the system | System (automated), Admin |
| F-031 | View All Club | View a list of all clubs in the system | Admin |
| F-032 | Search Club | Search clubs by name, category, or tags | Student, Admin |
| F-033 | Filter Event | Filter events by date, venue, or club | Student |
| F-034 | Filter Budget Request | Organize budget requests based on certain criteria | Admin |
| F-035 | Log Activity | Automatically log important user/system actions | System (automated), Admin |
| F-036 | View Audit Log | Review logged system activities | Admin |
| F-037 | Set Permission | Assign or modify access rights for each user role | Admin |
| F-038 | Generate Venue Status Report | Generates the current status of all venues for specified time slots | Campus Space Reservation System |
| F-039 | View Venue Status | View real-time venue availability | Admin |

*Table 1: Overview of Features separated by Actor*

### **1.3.3 User Characteristics**

The general characteristics of the intended groups of users are as follows:

| **User Group** | **Expected Role in System** | **Educational Level** | **Technical Expertise** | **Notes on Usability Needs** |
| --- | --- | --- | --- | --- |
| Student | Register, join/view/search/filter clubs, view/post announcements, view events, RSVP, request club creation, view budgets | Undergraduate/Diploma | Basic (form filling, navigation) | Needs intuitive UI, simple navigation |
| Member  (Student who joined club) | Renew membership, request to join committee, view club events/announcements, view budget reports | Undergraduate/Diploma | Basic to Moderate | Needs accessible dashboards for announcements and event details |
| Committee | Help manage events, update/cancel events, send announcements | Undergraduate/Diploma | Moderate | Needs clear access to event editing tools and notification panels |
| Club President | Create/update/delete club, request events, book venues, manage members, submit budget proposals, send announcements | Undergraduate | Moderate to Advanced | Requires a full-featured dashboard with event, budget, and venue management tools |
| Admin | Approve/reject club creation, budget requests, event requests, manage users/permissions, view logs, generate reports | University Staff | Advanced | Needs full control panel with logging, permission settings, and advanced search/filter capabilities |
| Financial Management System  (External System) | Receive and process budget approvals, generate reports | N/A | System Integration | Requires secure API or data sync for budget handling |
| Campus Space Reservation System  (External System) | Provide real-time venue status, approve venue requests | N/A | System Integration | Needs reliable integration for real-time venue management and status reporting |

*Table 2: User Characteristics Description*

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **1.3.4 Limitations**

1. **Scope of Features:**

The system gives priority to essential features including user identification, club management, event scheduling, and budget approval workflows because of its short development period which is one academic semester. Intelligent budget optimization tools, machine learning for event recommendations, and comprehensive analytics dashboards are examples of advanced technologies that have been left out. Although this choice guarantees on-time delivery, it restricts future-proofing and intelligent capabilities.

1. **Budget Constraints:**

The development is limited to open-source and free tools like Bootstrap/Tailwind CSS, Django, and PostgreSQL. Funding is not available for enterprise services, premium plugins, or commercial APIs. Options like premium alerting systems, powerful cloud services, or complex calendar integrations are thus not available. This could restrict performance improvements, scalability, and some user conveniences.

1. **User Base Access**:

The application only supports authenticated users from the university domain. Students, club president and administrators are all included in this. Even if they participate in club activities, guest users, and outside collaborators are unable to use the portal. This restricts collaborations and community outreach that could improve club involvement.

1. **Internet Dependency:**

Since the program is web-based and hosted on-campus or on a cloud server, users need to have a steady and reliable internet connection. As a result, tasks like creating an event, submitting a budget, or editing a profile cannot be completed without an offline connection. This might be problematic in isolated locations or when there are network failures.

1. **Mobile Limitations:**

Even though the system has responsive design support for mobile use on tablets and smartphones, it is not a native smartphone application. Its performance might be slower, and the loading process could be slower compared to the totally optimized smartphone applications. Further, mobile-related features like native calendar syncing, push notifications, or biometric login are unsupported.

1. **Technical Integration:**

All integrations must comply with university IT policy and security procedures. For example, Single Sign-On (SSO) integration must integrate via OAuth2 or LDAP, and external APIs like Google Calendar must comply with campus data governance. That constrains integration flexibility with third-party applications and will prevent or delay some features when APIs do not fit institution criteria.

1. **Resource Availability:**

The project is largely sustained and created by students, who may possess varying availability and technical abilities. This can cause delayed fixes of bugs, decreased testing coverage, and uneven development cycles. Feature requests or enhancements may be postponed due to school responsibilities.

1. **Hosting Environment:**

The deployment must take place on Linux-based servers from the institution or authorized cloud infrastructure. This limits the use of some server technologies like Windows hosting and the potential for deployment automation. Creating environments with restricted access or compliance requirements can also present administrative challenges for developers.

## **1.4 Definitions**

| **Term** | **Definition** |
| --- | --- |
| Application | The web-based Club and Event Management System designed to support university students and staff in organizing, managing, and approving club events. |
| User | Any individual interacting with the system, including students, club leaders, faculty advisors, and administrators. |
| Club Leader | A student with administrative rights over a club, responsible for event planning, budget submission, and activity management. |
| Administrator | A university staff user with full access to manage clubs, events, budgets, and system-wide settings. |
| Event Proposal | A structured digital request submitted by a club leader to organize an event, including logistics and financial needs. |
| Budget Approval Workflow | A step-by-step process for reviewing and approving club-submitted budgets before funds are allocated. |
| Single Sign-On (SSO) | A secure login method allowing users to access the system using existing university credentials. |
| Dashboard | The main interface displays a summary of club activities, pending tasks, and budget status for the user. |
| Responsive Design | UI/UX design approach ensuring the application functions well on desktops, tablets, and smartphones. |
| Django | A high-level, open-source Python web framework used to build the backend of the application. |
| PostgreSQL | An open-source relational database management system used to store and retrieve the application's data. |
| API (Application Programming Interface) | A set of protocols that allow software components to communicate, such as with Google Calendar or email notifications. |
| Open Source | Software with publicly accessible code that can be freely used, modified, and distributed. |
| Authentication | The process of verifying a user’s identity through credentials like usernames and passwords. |
| Authorization | The process of granting a user specific system permissions based on their role (e.g., student, advisor, admin). |
| Hosting Server | The server environment where the system is deployed, typically a Linux-based or cloud infrastructure approved by the university. |
| Kano Model | A method for prioritizing system features by categorizing them as Must-be, Performance, or Delighter based on user satisfaction and expectations. |
| Must-be Feature | A basic feature that users expect; its absence causes dissatisfaction, but its presence does not increase satisfaction. |
| Performance Feature | A feature where user satisfaction increases or decreases proportionally with the level of functionality or quality. |
| Delighter Feature | An unexpected feature that pleasantly surprises users and increases satisfaction, though its absence is not typically noticed. |
| Use Case | A specific interaction or scenario describing how a user performs a task using the system to achieve a goal. |
| Functional Requirement | A requirement that defines what the system should do, such as features, workflows, and data handling. |
| Non-Functional Requirement | A requirement that defines how the system performs tasks, covering aspects like performance, reliability, and usability. |
| Interview Transcript | A documented conversation between project members and stakeholders, used to gather insights, preferences, and requirements. |
| Stakeholder | Any individual or group with an interest or influence in the system, including students, staff, club leaders, and administrators. |
| Feedback System | A feature that allows users to provide ratings or comments on events to support continuous improvement and reporting. |
| Role-Based Access Control (RBAC) | A security approach that limits access to features and data based on user roles (e.g., student, president, admin). |
| Event Calendar | A visual tool in the system displaying upcoming club events and bookings, often integrated with external calendars. |
| Progressive Web App (PWA) | A web application that behaves like a native mobile app, offering offline access, push notifications, and mobile responsiveness. |
| ORM (Object-Relational Mapping) | A programming technique that connects database tables to application code, used for easier data management in Django or other frameworks. |

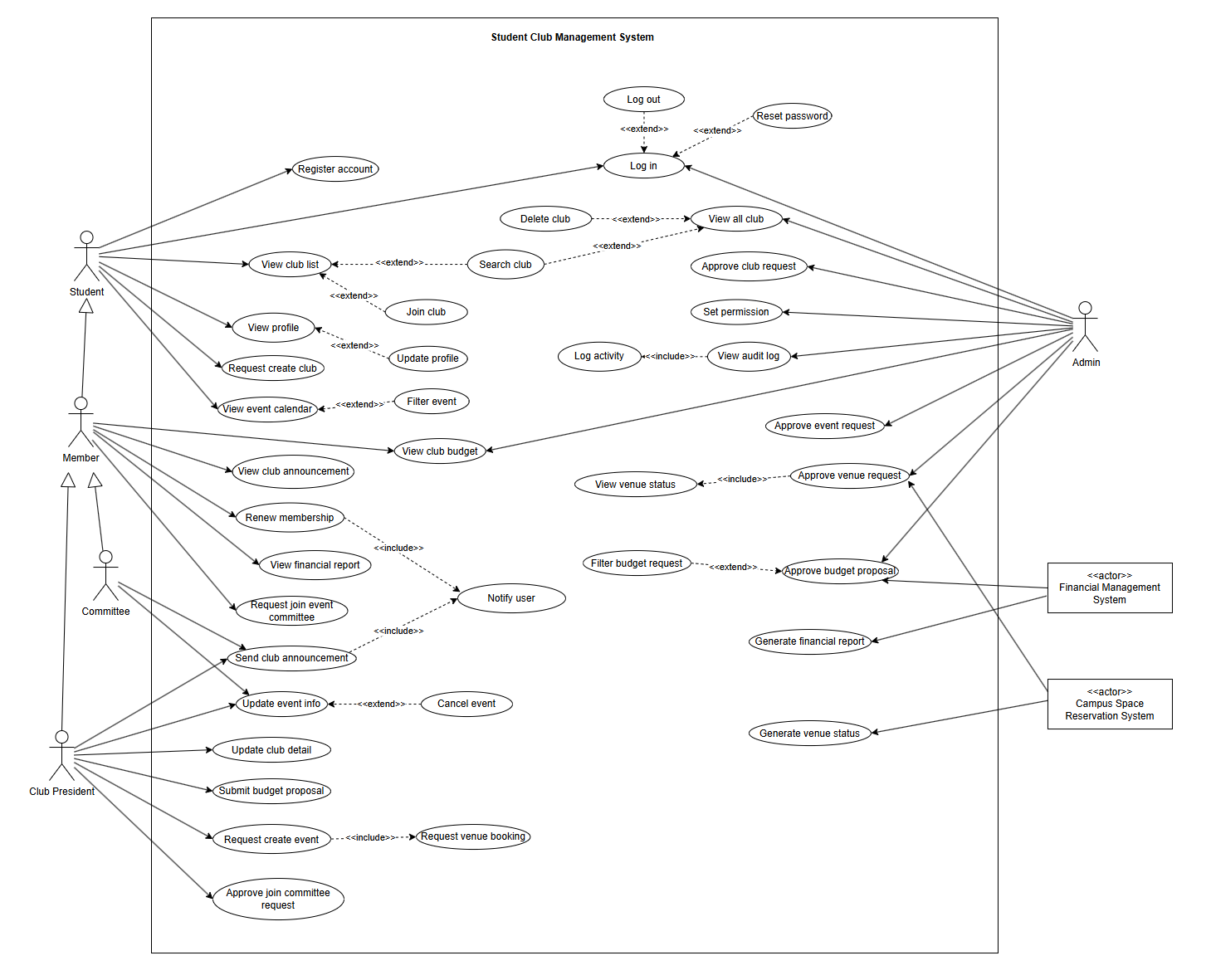
*Table 3: Terms and Definitions*

# **2.0 References**

1. Chenjeri, A., Sai, Satya, T., Santhan Mourya, K., Murali, K., Muddada, & Student. (2024). Issue 3 [www.jetir.org](http://www.jetir.org/) (ISSN-2349-5162). JETIR2403033 Journal of Emerging Technologies and Innovative Research, 11.
2. Et. al., A. A. A. (2021). Design Architecture Of An Integrated Student Activities Management System For Higher Education. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(5), 1676–1683. <https://doi.org/10.17762/turcomat.v12i5.2158>
3. Hariprasad, M., Neha N, Dey, N., Pratiba D, & Ramakanth Kumar P. (2023). College Club Activity Management System. <https://doi.org/10.1109/csitss60515.2023.10334208>
4. Khatri, I. A., Ghonge, M., Mirajkar, R., Shinde, S., Hon, R., & Yenkikar, A. (2024). Enhancing Campus Engagement with A Secure and Comprehensive Platform for Club Management and Student Participation. 2024 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), 1–6. <https://doi.org/10.1109/icbds61829.2024.10837416>
5. Mukthashree, B., Chinmayee K G, Manjuprasad, B., & Student. (n.d.). Campus Club Management System Application.

# **3.0 Requirements**

## **3.1 Functions**

 ***Figure 3.1: Use Case Diagram of Student Management System***

#### 

#### 

#### 

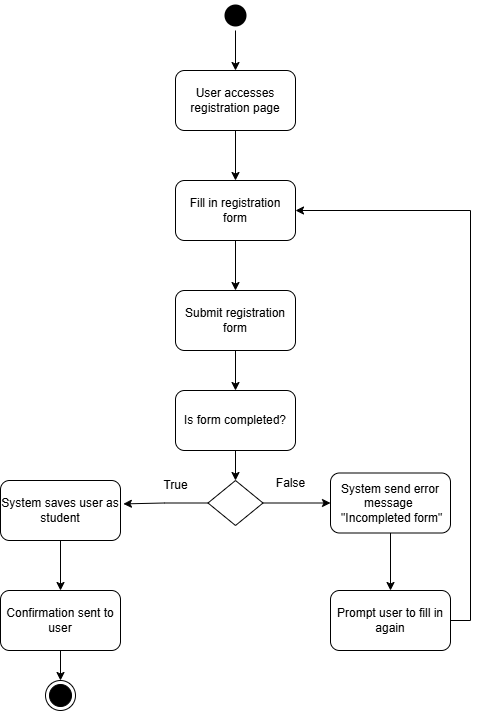
#### 

#### 

### **3.1.1 Register**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-001 |
| **Feature** | User Registration with Role Selection |
| **Purpose** | To register a user as a Student, and ensure appropriate follow-up actions based on the role. |
| **Actor(s)** | User |
| **Precondition** | - User accesses the registration page. |
| **Postcondition** | - User is registered as Student.  - Confirmation or notification is sent accordingly. |
| **Main Flow** | 1. User accesses the registration page.  2. System registers the user as Student and sends confirmation. |
| **Alternate Scenario** | If a user submits an incomplete form, the system prompts for correction. |

*Table 4: Register Use Case Specification*

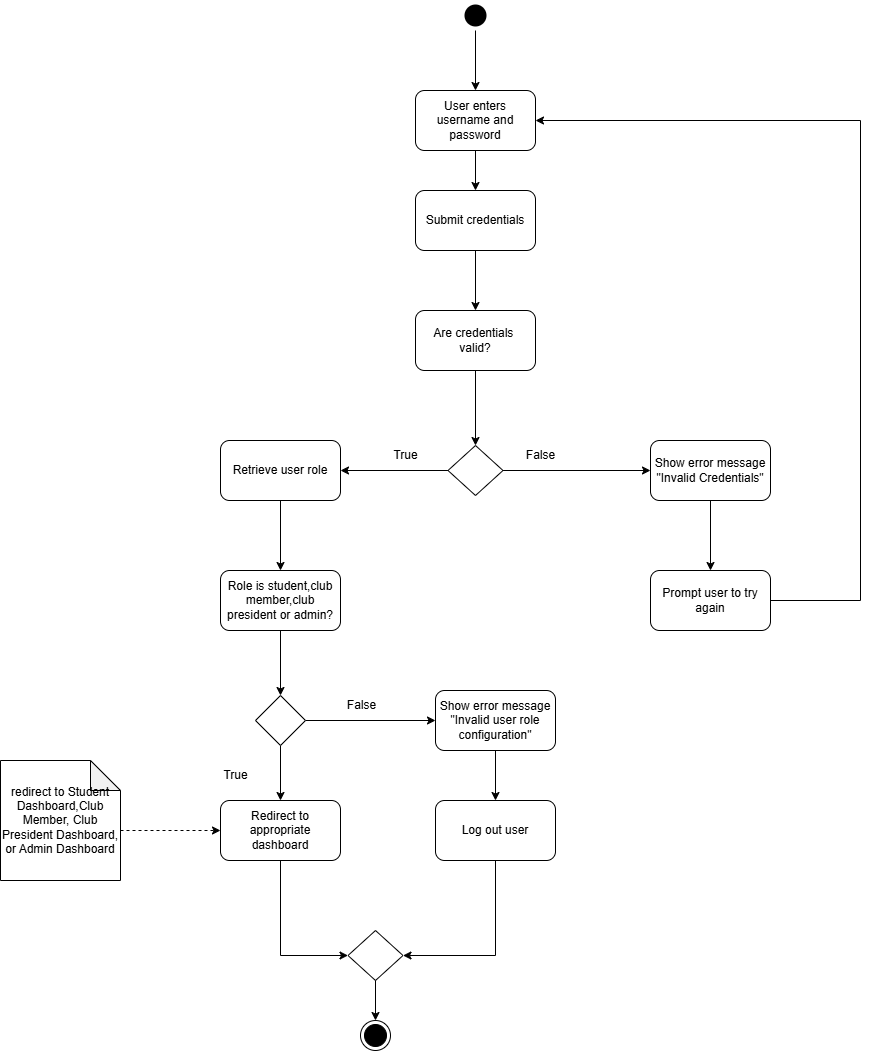


*Figure 3.1.1 Register*

### **3.1.2 Login**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-002 |
| **Feature** | Login User |
| **Purpose** | To authenticate users and redirect them to the appropriate dashboard based on their role (Student, Member, Club President or Admin). |
| **Actor(s)** | User |
| **Precondition** | - User accesses the login page. |
| **Postcondition** | - If login is successful and the user role is valid, the user is redirected to the correct dashboard.  - If invalid, the user sees an error or is logged out. |
| **Main Flow** | 1. User enters username and password.  2. User submits credentials.  3. System checks if credentials are valid.  4. If valid: system retrieves user role.  5. System checks if the user role is student, president, or admin.  6. If valid, the system redirects to the respective dashboard. |
| **Alternate Scenario** | 1. If credentials are invalid, show "Invalid Credentials" error message and prompt to try again.  2. If the user role is misconfigured or invalid, show "Invalid user role configuration" and log out. |

*Table 5: Login Use Case Specification*

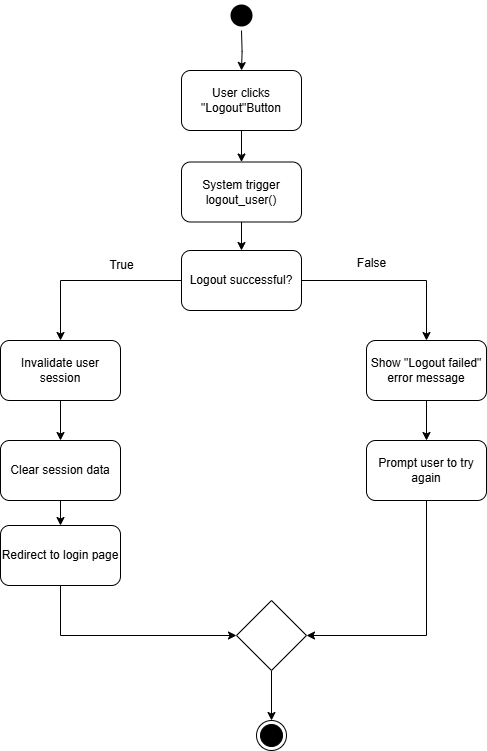


*Figure 3.1.2 Login*

### **3.1.3 Log Out**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-003 |
| **Feature** | User Logout |
| **Purpose** | To log the user out of the system, invalidate the session, clear session data, and redirect them to the login page. |
| **Actor(s)** | User |
| **Precondition** | User is logged in. |
| **Postcondition** | - User session is invalidated.  - Session data is cleared.  - User is redirected to the login page. |
| **Main Flow** | 1. User clicks the "Logout" button.  2. System triggers logout\_user() function.  3. System invalidates the user session.  4. System clears session data.  5. System redirects the user to the login page. |
| **Alternate Scenario** | If logout fails, show "Logout failed" error message and prompt the user to try again. |

*Table 6: Log Out Use Case Specification*

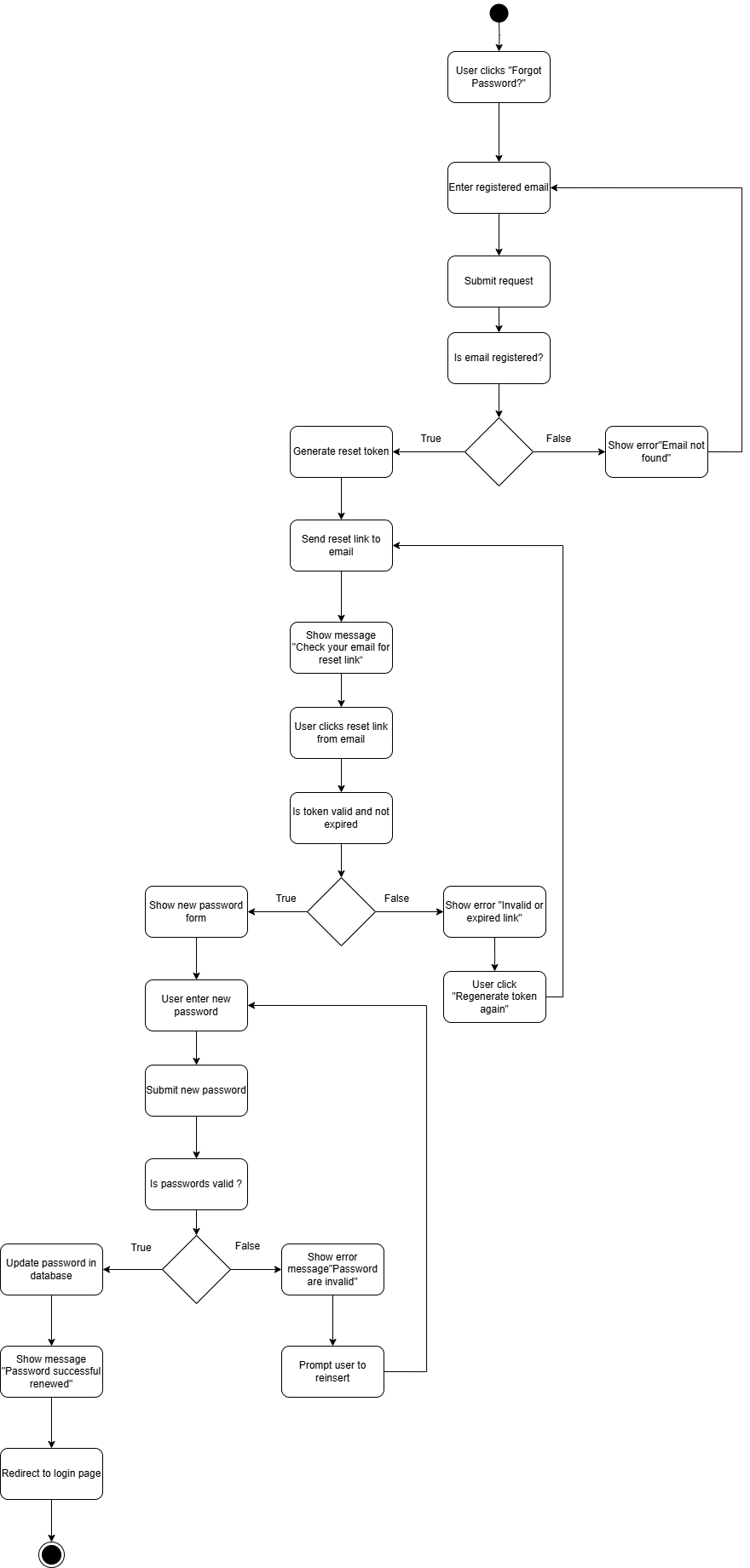


*Figure 3.1.3 Log Out*

### **3.1.4 Reset Password**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-004 |
| **Feature** | Reset Password |
| **Purpose** | To allow users to recover access to their account by securely resetting a forgotten password via a reset link sent to their registered email |
| **Actor(s)** | User |
| **Precondition** | User has a registered account and initiates a password reset request via the "Forgot Password?" link |
| **Postcondition** | User either receives a reset link email and successfully resets the password, or is shown an appropriate error message |
| **Main Flow** | 1. User clicks "Forgot Password?"  2. System prompts for registered email  3. User submits email  4. System verifies if email is registered  5. If valid, system generates reset token and sends email  6. User clicks reset link from email  7. System validates token  8. If valid, system prompts for new password  9. User submits new password  10. System validates and updates password  11. User is redirected to login with success message |
| **Alternate Scenario** | 1. If email not found, system shows error message  2. The token is expired or invalid, system shows error and user clicks “Regenerate token again”  3. The password doesn't meet criteria, system shows error and prompts re-entry |

*Table 7: Reset Password Use Case Specification*

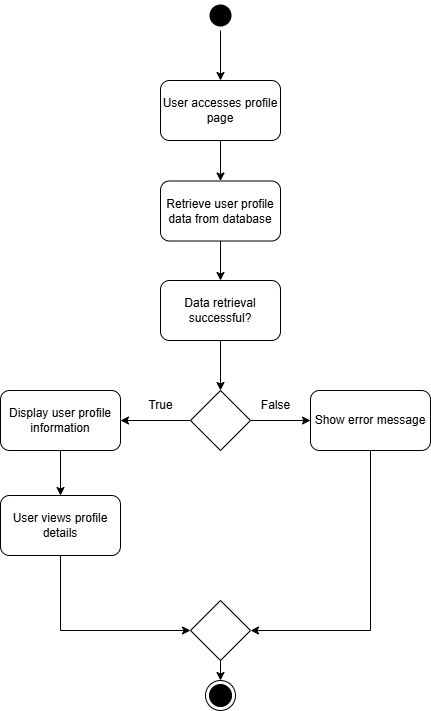


*Figure 3.1.4 Reset Password*

### **3.1.5 View Profile**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-005 |
| **Feature** | User Profile View |
| **Purpose** | Allow the user to view their profile information. |
| **Actor(s)** | User |
| **Precondition** | User is logged in and navigated to the profile section or dashboard. |
| **Postcondition** | User’s profile information is displayed on the screen. |
| **Main Flow** | 1. User accesses the profile section.  2. System retrieves the user profile data from the database.  3. System displays the user profile information.  4. User views profile details. |
| **Alternate Scenario** | Profile data retrieval fails, system shows an error message. |

*Table 8: View Profile Use Case Specification*

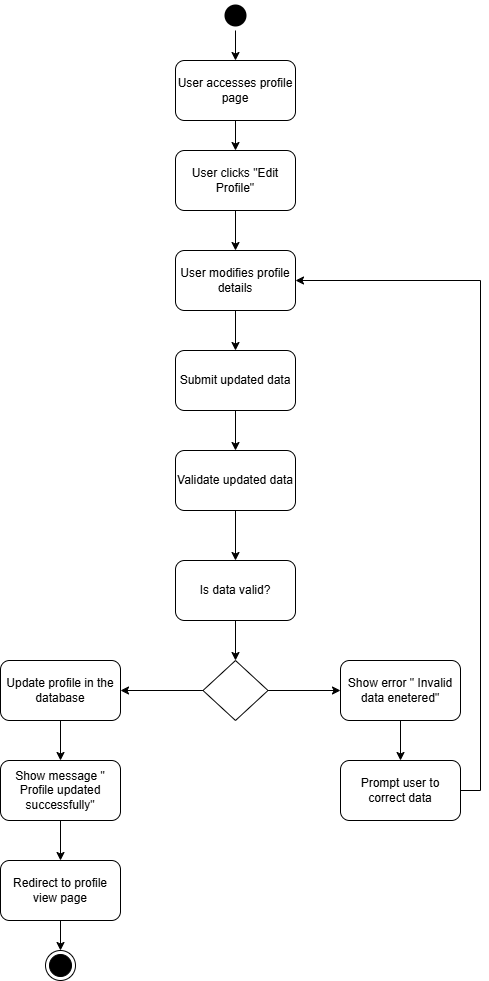


*Figure 3.1.5 View Profile*

### **3.1.6 Update Profile**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-006 |
| **Feature** | User Profile Update |
| **Purpose** | Allow the user to access and update their profile information. |
| **Actor(s)** | User |
| **Precondition** | User is logged in and on the profile page. |
| **Postcondition** | User’s profile information is updated in the database, or an error message is shown if invalid data. |
| **Main Flow** | 1. User accesses the profile page.  2. User clicks "Edit Profile".  3. User modifies profile details.  4. User submits the updated data.  5. System validates the data.  6. If data is valid, it is updated in the database.  7. System shows a success message and redirects to the profile view page. |
| **Alternate Scenario** | Invalid data entered, system prompts user to correct data. |

*Table 9: Update Profile Use Case Specification*

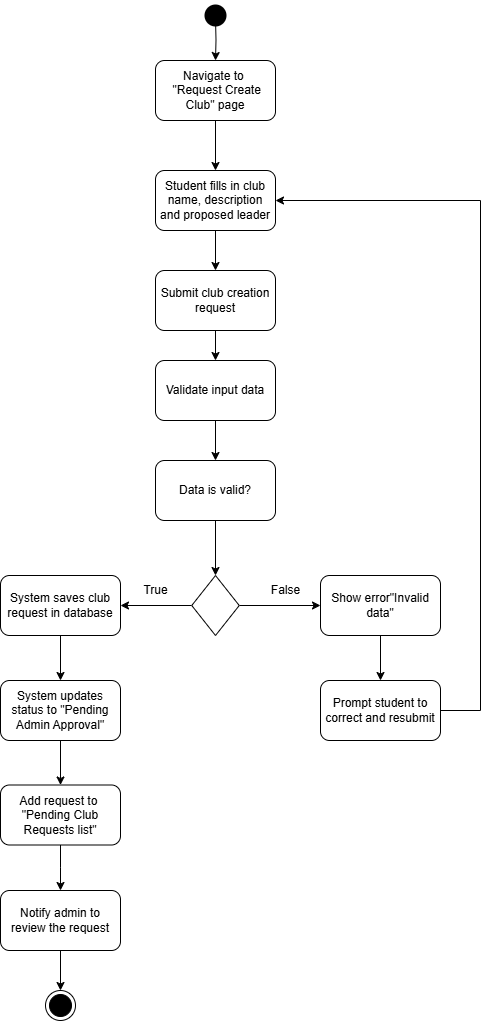


*Figure 3.1.6 Update Profile*

### **3.1.7 Request Create Club**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-007 |
| **Feature** | Request Create Club |
| **Purpose** | Allow students to request the creation of a new club. |
| **Actor(s)** | Student |
| **Precondition** | Student is logged in and navigates to the "Request Create Club" page. |
| **Postcondition** | Club creation requests are saved in the database, and the student is notified of the request status. |
| **Main Flow** | 1. Students access the "Request Create Club" page.  2. Students fill in the club name, description, and proposed leader.  3. Students submit the club creation request.  4. System validates the submitted data.  5. If data is valid, it is saved in the database and updated to "Pending Admin Approval".  6. System notifies the admin to review the request. |
| **Alternate Scenario** | Invalid data entered, system prompts students to correct and resubmit. |

*Table 10: Request Create Club Use Case Specification*

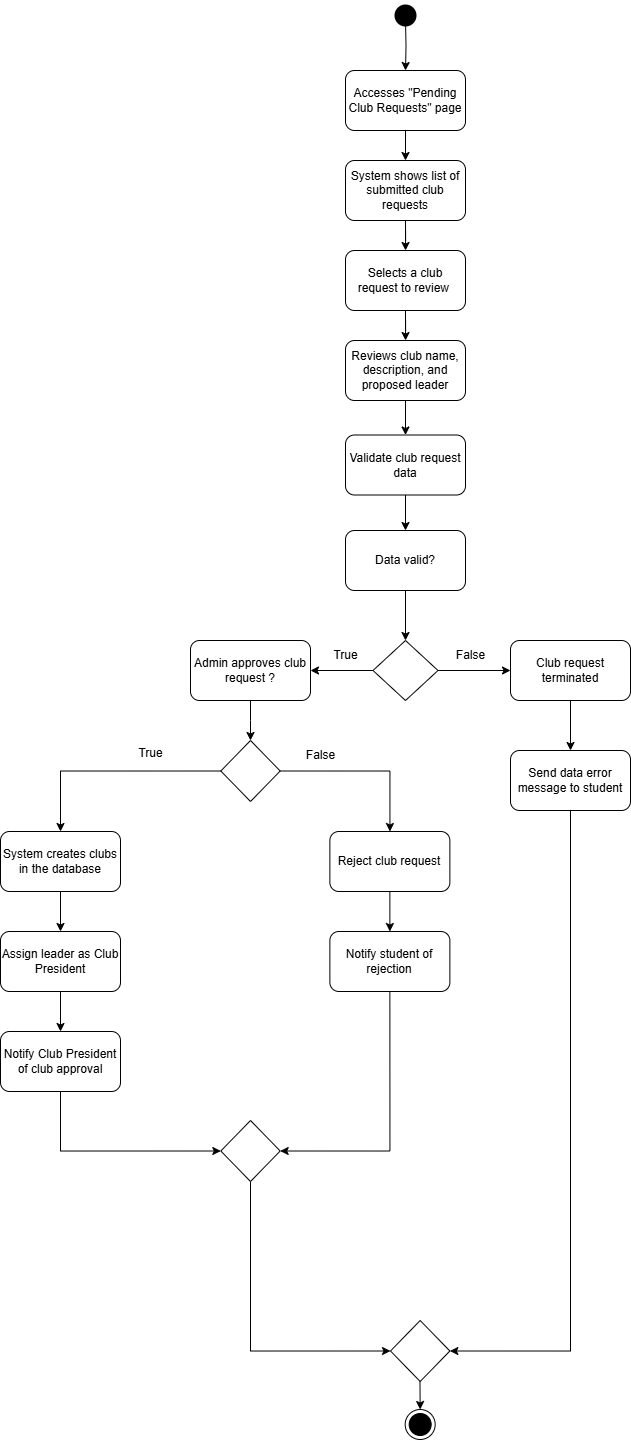


*Figure 3.1.7 Request Create Club*

### **3.1.8 Approve Club Request**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-008 |
| **Feature** | Club Approval by Admin |
| **Purpose** | Allow the admin to review and approve or reject club creation requests submitted by the president. |
| **Actor(s)** | Admin |
| **Precondition** | Admin is logged in and accesses the "Pending Club Requests" page. |
| **Postcondition** | Club requests are either approved or rejected, and the system notifies the user accordingly. |
| **Main Flow** | 1. Admin accesses the "Pending Club Requests" page.  2. System displays a list of submitted club requests.  3. Admin selects a club request to review.  4. Admin reviews the club name, description, and proposed leader.  5. Admin validates the data of the club request.  6. Admin approves or rejects the club request.  7. If approved, the system creates the club in the database.  8. If approved, the leader is assigned as the Club President.  9. System notifies the club president of club approval. |
| **Alternate Scenario** | 1. If data is invalid, the system terminates the club request, and sends a data error message to the student.  2. If approval is denied,the admin rejects the club request, and the system notifies the student of rejection. |

*Table 11: Approve Club Request Use Case Specification*

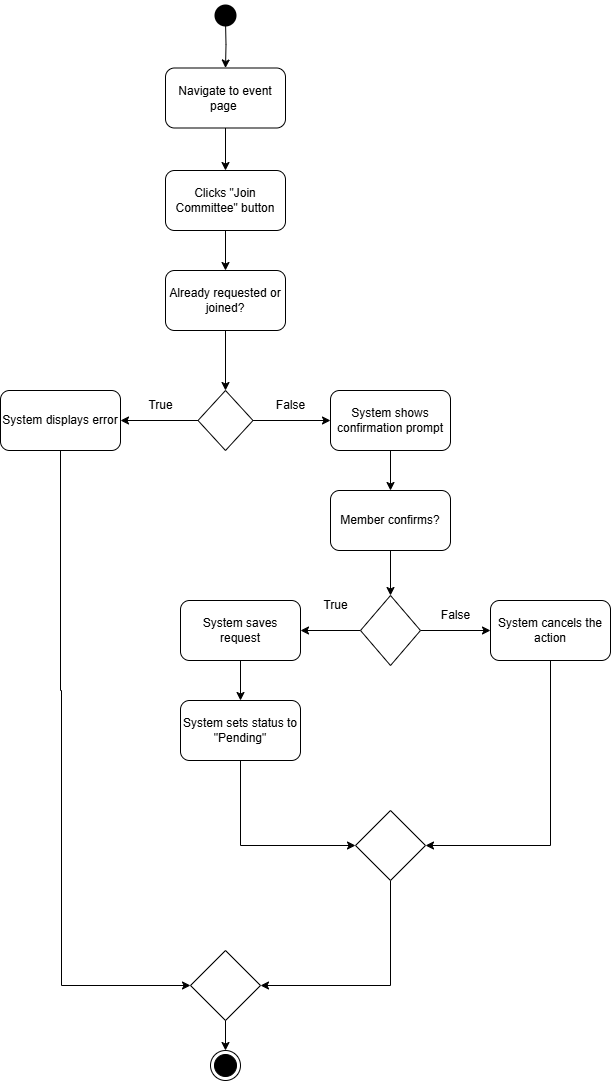


*Figure 3.1.8 Approve Club Request*

### **3.1.9 Request Join Event Committee**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-009 |
| **Feature** | Request Join Event Committee |
| **Purpose** | Allow a member to request participation in an event's organizing committee. |
| **Actor(s)** | Member |
| **Precondition** | The club member is logged in and is a valid member of the club hosting the event. |
| **Postcondition** | The request to join the committee is submitted and awaits approval by the event organizer or system. |
| **Main Flow** | 1. Club members navigate to the event page.  2. Members click the "Join Committee" button.  3. System displays a confirmation dialog.  4. Member confirms the request.  5. System records the request and notifies the event organizer.  6. Request status is set to "Pending". |
| **Alternate Scenario** | 1. If the member is not eligible (e.g. already part of the committee), the system blocks the request and shows an error.  2. Member cancels the request at the confirmation dialog. |

*Table 12: Request Join Event Committee Use Case Specification*

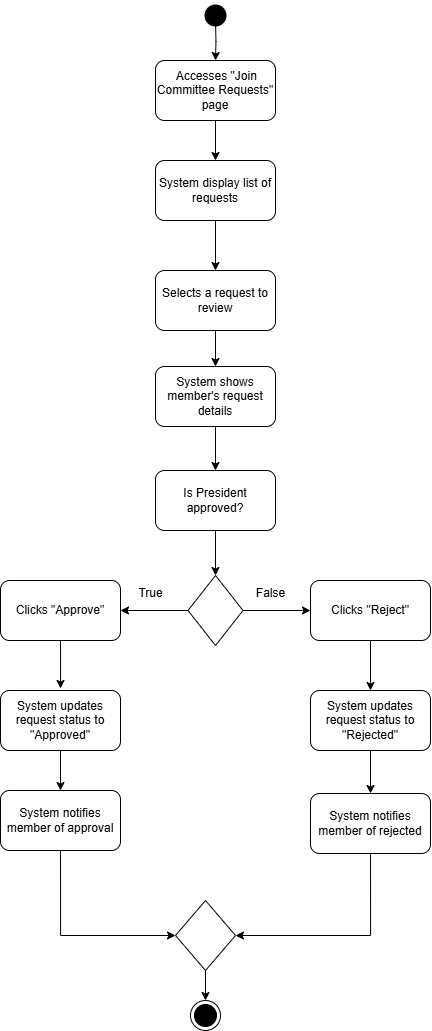


*Figure 3.1.9 Request Join Event Committee*

### **3.1.10 Approve Join Committee Request**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-010 |
| **Feature** | Approve Join Committee Request |
| **Purpose** | Allow the Club President to approve a club member’s request to join an event’s organizing committee. |
| **Actor(s)** | Club President |
| **Precondition** | Club President is logged in and viewing pending join committee requests for an event. |
| **Postcondition** | Selected club members are approved and assigned as part of the event committee. |
| **Main Flow** | 1. Club President views the list of join committee requests.  2. Club President selects a request to review.  3. The Club President clicks "Approve".  4. System updates the request status to "Approved".  5. System notifies the member of the approval. |
| **Alternate Scenario** | 1. If the Club President clicks “Rejected”,system updates request status to "Rejected" then notify the member. |

*Table 13: Approve Join Committee Request Use Case Specification*

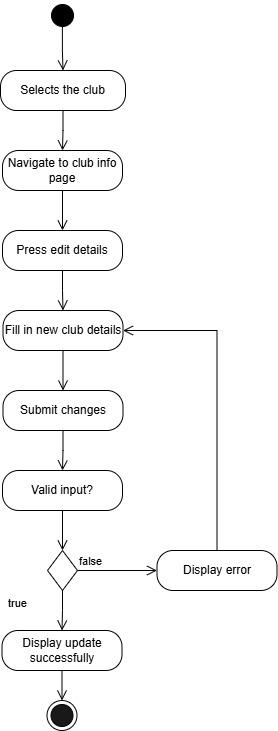


*Figure 3.1.10 Approve Join Committee Request*

### **3.1.11. Update Club Details**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-011 |
| **Feature** | Update club details |
| **Purpose** | Allow the user to modify the information of the club. |
| **Actor(s)** | Club President |
| **Precondition** | 1. Club exists.  2. Club President is authenticated.  3. Club President is logged in. |
| **Postcondition** | New club details updated in the system. |
| **Main Flow** | 1. Selects the club.  2. Navigate to the club info page.  3. Press on edit details.  4. Update new information.  5. Submits changes. |
| **Alternate Scenario** | Required field is missing or invalid input, show error and prompt correction. |

*Table 14: Update Club Details Use Case Specification*

  
*Figure 3.1.11: Update Club Details*

#### 

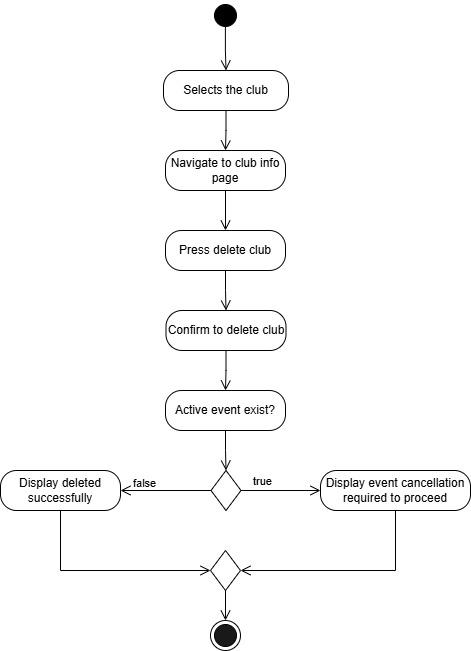
#### 

#### 

### **3.1.12. Delete Club**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-012 |
| **Feature** | Delete club |
| **Purpose** | Allow the user to delete the club from the system. |
| **Actor(s)** | Admin |
| **Precondition** | 1. Club exists.  2. Admin is authenticated.  3. Admin is logged in. |
| **Postcondition** | The club is deleted from the system. |
| **Main Flow** | 1. Selects the club.  2. Navigate to the club info page.  3. Press delete club.  4. Confirm to delete the club.  5. System display success message |
| **Alternate Scenario** | The club has an active event ongoing, system displays event cancellation required to delete the club. |

*Table 15: Delete Club Use Case Specification*

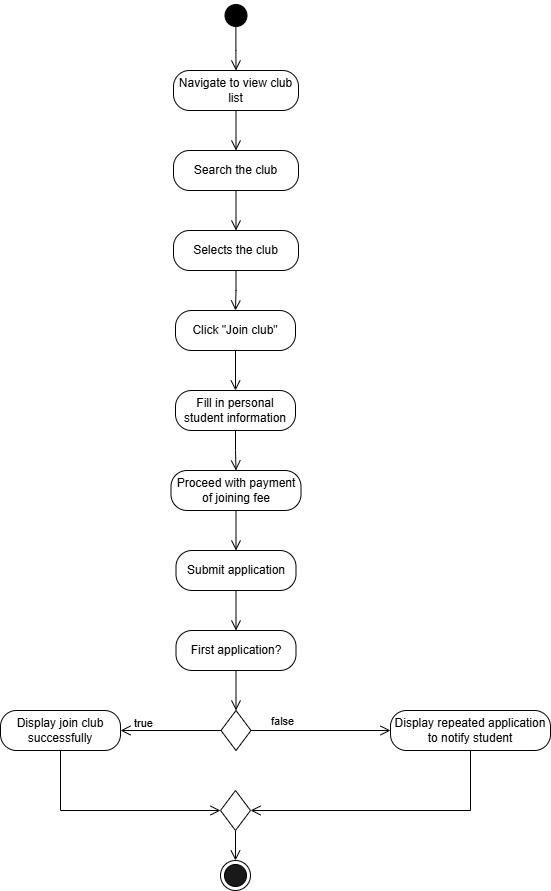
  
*Figure 3.1.12: Delete Club*

#### 

### **3.1.13. Join Club**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-013 |
| **Feature** | Join club |
| **Purpose** | Allow the student to join the club. |
| **Actor(s)** | Student |
| **Precondition** | 1. Club exists.  2. The student is logged into the system.  3. The student never joined the club. |
| **Postcondition** | The student joined the club successfully and was promoted into club member. |
| **Main Flow** | 1. Navigate to view the club list.  2. Search the club.  3. Select the club.  4. Click "Join club".  5. Fill in personal student information.  6. Proceed with payment of joining fee.  7. Submit application.  8. System displays a success message. |
| **Alternate Scenario** | Students have duplicated application, system pop up notification windows. |

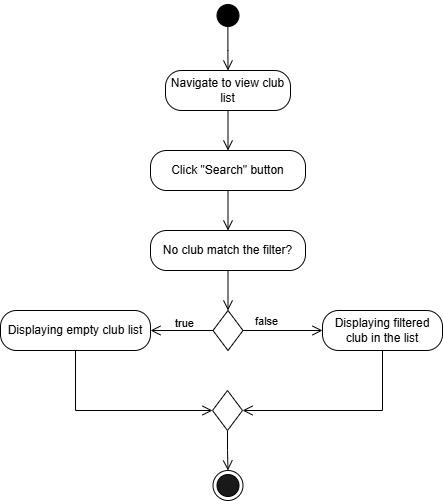
*Table 16: Join Club Use Case Specification*

  
*Figure 3.1.13: Join Club*

### **3.1.14. View Club List**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-014 |
| **Feature** | View club list |
| **Purpose** | Allow the student to view existing clubs in the system. |
| **Actor(s)** | Student |
| **Precondition** | Students are logged in the system. |
| **Postcondition** | Existing clubs in the list are displayed. |
| **Main Flow** | 1. Navigate to view the club list.  2. All existing clubs are displayed in the list. |
| **Alternate Scenario** | No club matches the applied filter, the club list is empty. |

*Table 17: View Club List Use Case Specification*

  
*Figure 3.1.14: View Club List*

#### 

#### 

#### 

#### 

#### 

#### 

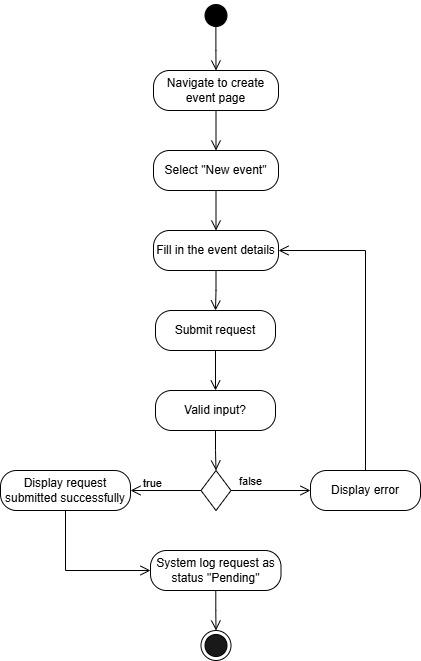
#### 

#### 

### **3.1.15. Request Create Event**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-015 |
| **Feature** | Request create event |
| **Purpose** | Allow the club president to request for a club event. |
| **Actor(s)** | Club president |
| **Precondition** | 1. Club exists.  2. Club president is authenticated.  3. Club president is logged in. |
| **Postcondition** | Club event request is submitted successfully, pending for admin approval. |
| **Main Flow** | 1. Navigate to request create event page.  2. Select "New event".  3. Fill in event details.  4. Submit request. |
| **Alternate Scenario** | Required field is missing or invalid input, show error and prompt correction. |

*Table 18: Request Create Event Use Case Specification*

  
*Figure 3.1.15: Request Create Event*

#### 

#### 

#### 

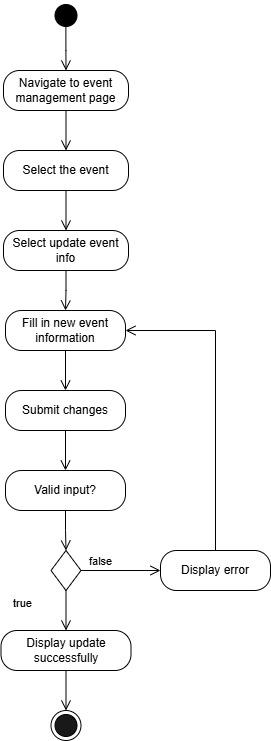
#### 

#### 

### **3.1.16. Update Event Info**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-016 |
| **Feature** | Update event info |
| **Purpose** | Allow users to update event information. |
| **Actor(s)** | 1. Club President  2. Committee |
| **Precondition** | 1. Event exists.  2. User is authenticated.  3. User is logged in. |
| **Postcondition** | Event new information updated successfully in the system. |
| **Main Flow** | 1. Navigate to the event management page.  2. Select the event.  3. Select update event info.  4. Fill in new event information.  5. Submit changes. |
| **Alternate Scenario** | Required field is missing or invalid input, show error and prompt correction. |

*Table 19: Update Event Info Use Case Specification*

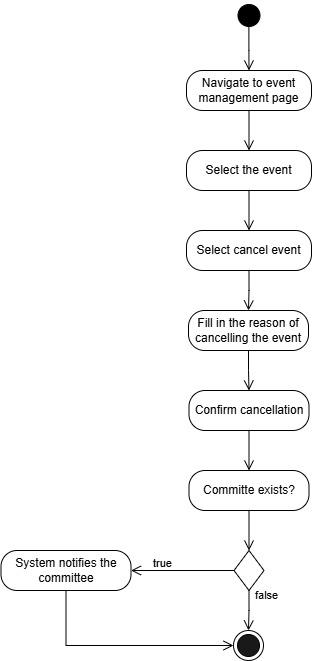
*****Figure 3.1.16: Update Event Info*

#### 

### **3.1.17. Cancel Event**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-017 |
| **Feature** | Cancel event |
| **Purpose** | Allow users to cancel the event. |
| **Actor(s)** | 1. Club president  2. Committee |
| **Precondition** | 1. Event exists.  2. User is authenticated.  3. User is logged in. |
| **Postcondition** | The event is cancelled and deleted from the system. |
| **Main Flow** | 1. Navigate to the event management page.  2. Select the event.  3. Select cancel event.  4. Fill in the reason for cancelling the event.  5. Confirm cancel. |
| **Alternate Scenario** | If the event has a committee, send notification to all committees. |

*Table 20: Cancel Event Use Case Specification*

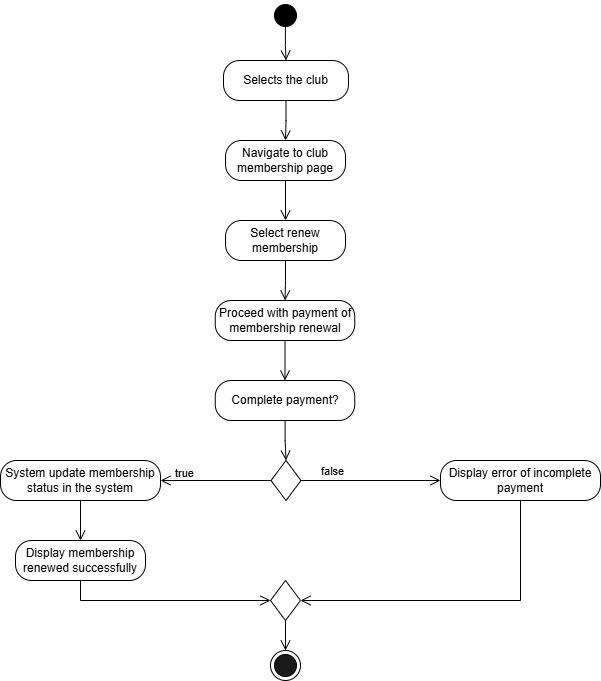
  
*Figure 3.1.17: Cancel Event*

#### 

### **3.1.18. Renew Membership**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-018 |
| **Feature** | Renew club membership |
| **Purpose** | Allow members to renew club membership. |
| **Actor(s)** | Member |
| **Precondition** | 1. Club exists.  2. Members are authenticated.  3. Members are logged in. |
| **Postcondition** | Members successfully renew the membership and update in the system. |
| **Main Flow** | 1. Select the club.  2. Navigate to the club membership page.  3. Select renew membership.  4. Proceed with the payment of membership renewal.  5. System update membership status in the system. |
| **Alternate Scenario** | Member did not complete the payment, system display error and notify member to try again to complete the payment. |

*Table 20: Renew Membership Use Case Specification*

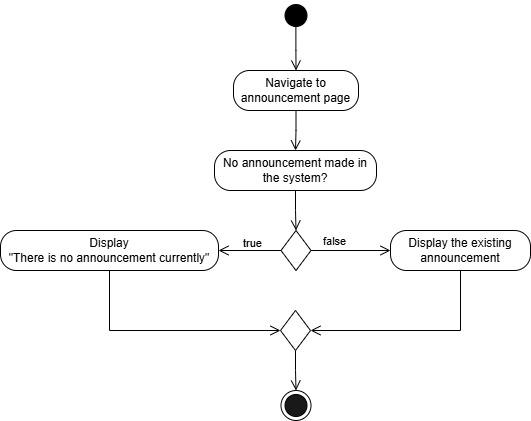
  
*Figure 3.1.18: Renew Membership*

#### 

### **3.1.19. View Announcement**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-019 |
| **Feature** | View announcement |
| **Purpose** | Allow members to view exclusive announcements of the club. |
| **Actor(s)** | Member |
| **Precondition** | 1. Club exists.  2. Members are authenticated.  3. Members are logged in. |
| **Postcondition** | All existing announcements are displayed. |
| **Main Flow** | 1. Select the club.  2. Navigate to the announcement page.  3. All existing club announcements are displayed. |
| **Alternate Scenario** | No announcement is made in the system, announcement page displaying "There is no announcement currently". |

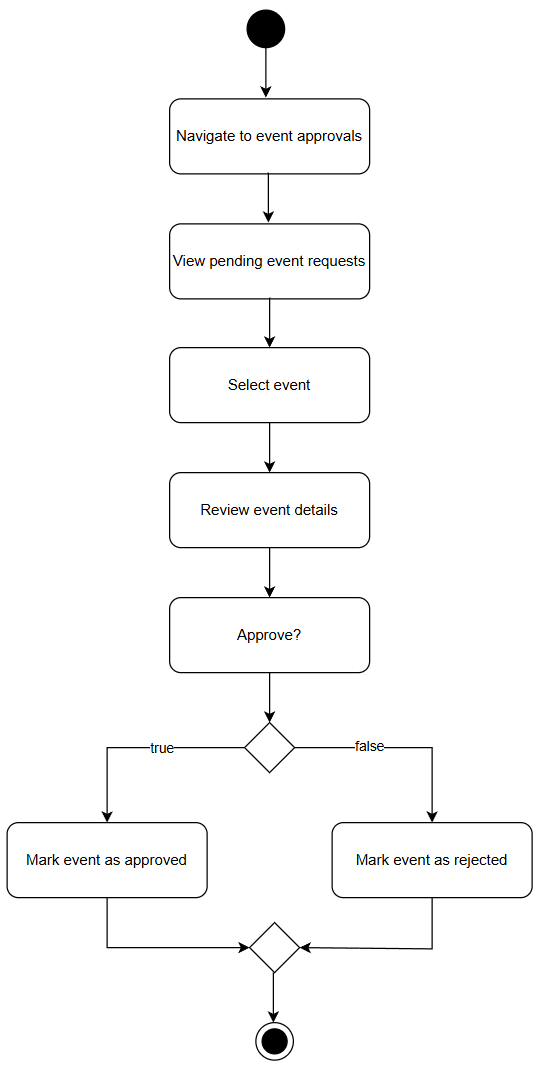
*Table 21: View Announcement Use Case Specification*

  
*Figure 3.1.19: View Announcement*

### **3.1.20 Approve Event Request**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-020 |
| **Feature** | Approve Event |
| **Purpose** | To allow administrators to review and approve club event proposals. |
| **Actor(s)** | Admin |
| **Precondition** | Admin is logged in and the event is in a "Pending Approval" status. |
| **Postcondition** | Event status is updated by the system. |
| **Main Flow** | 1. Navigates to event approvals.  2. View pending event requests  3. Select event.  4. Review details.  5. Admin chooses to approve or reject requests.  6. Mark event as "Approved" or "Rejected". |
| **Alternate Scenario** | If event details are incomplete, the admin may reject the request. |

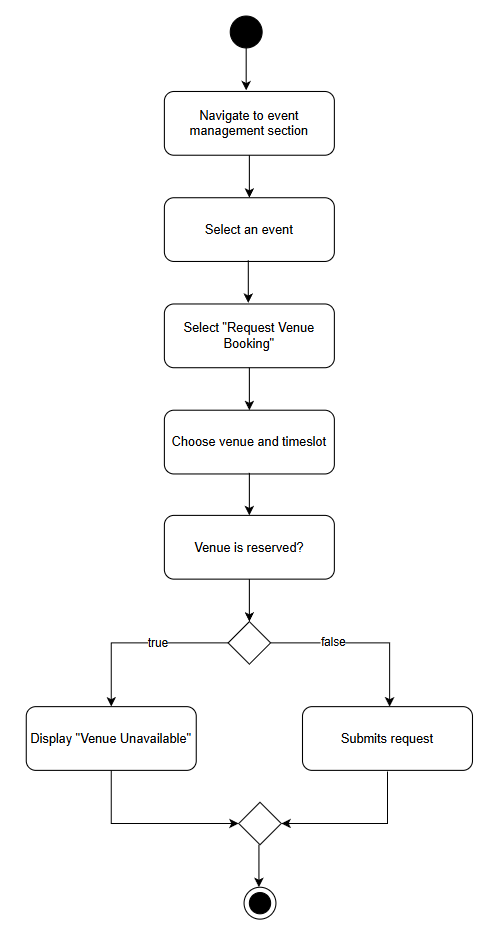
*Table 22: Approve Event Request Use Case Specification*

  
*Figure 3.1.20 Approve Event Request*

### **3.1.21 Request Venue Booking**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-021 |
| **Feature** | Request Venue Booking |
| **Purpose** | To allow club presidents to request available venues for events. |
| **Actor(s)** | Club President |
| **Precondition** | - Event must be created.  - Club President is logged in. |
| **Postcondition** | Venue booking request is submitted and pending admin approval. |
| **Main Flow** | 1. Navigate to the event management section.  2. Selects an event.  3. Select "Request Venue Booking".  4. Chooses venue and time slot.  5. Submits request. |
| **Alternate Scenario** | If the venue is already reserved, the system notifies unavailability or suggests alternatives. |

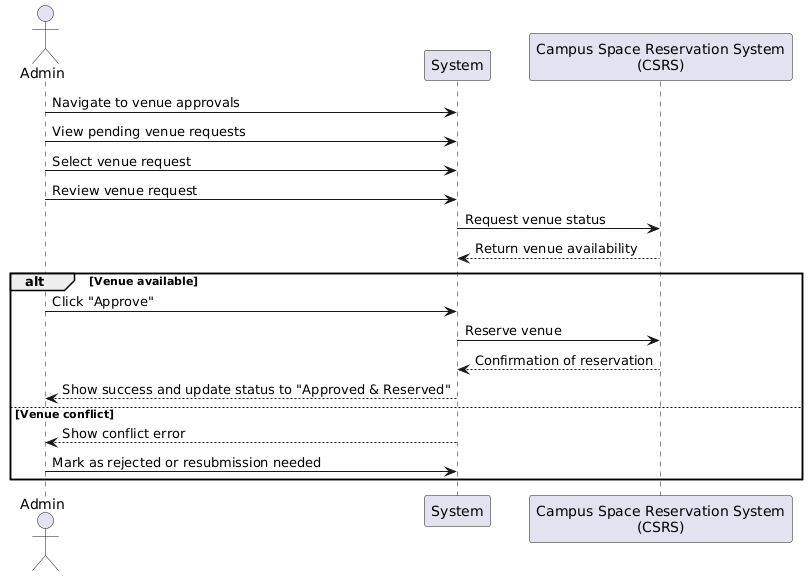
*Table 23: Request Venue Booking Use Case Specification*

  
*Figure 3.1.21 Request Venue Booking*

### **3.1.22 Approve Venue Request**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-022 |
| **Feature** | Approve Venue Request |
| **Purpose** | To enable admin to approve or reject venue booking requests, and trigger booking in the Campus Space Reservation System. |
| **Actor(s)** | 1. Admin  2. Campus Space Reservation System (external) |
| **Precondition** | - Admin is logged in.  - Event must be created.  - Venue request exists with "Pending Approval" status.  - Venue status must be viewed. |
| **Postcondition** | The booking status is updated in the system and the venue is reserved in the Campus Space Reservation System. |
| **Main Flow** | 1. Navigate to venue approvals.  2. View pending venue requests.  3. Select venue request.  4. Review venue request.  5. View venue status generated by Campus Space Reservation System (CSRS).  6. If the venue is available, Admin clicks "Approve".  7. System reserves venue via CSRS. |
| **Alternate Scenario** | If the schedule conflicts during final booking, the system returns an error and status is set to rejected or marked for resubmission with a comment. |

*Table 24: Approve Venue Request Use Case Specification*

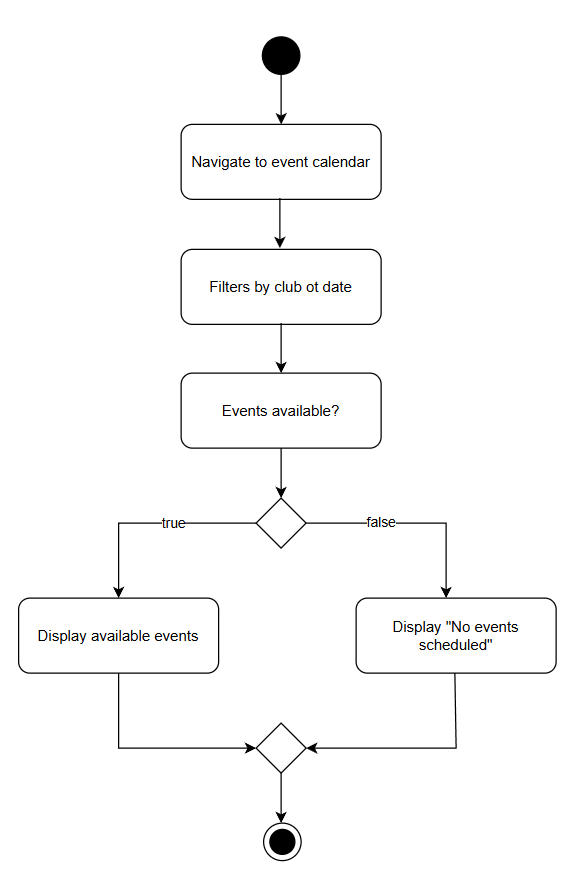
*Figure 3.1.22 Approve Venue Request*

#### 

### **3.1.23 View Event Calendar**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-023 |
| **Feature** | View Event Calendar |
| **Purpose** | To allow students to browse upcoming events on campus or by club. |
| **Actor(s)** | Student |
| **Precondition** | Student is logged in. |
| **Postcondition** | Events are viewed by the students in the calendar. |
| **Main Flow** | 1. Navigates to the event calendar.  2. Optionally filters by club/date.  3. Display available events. |
| **Alternate Scenario** | If no events are available, display "No events scheduled". |

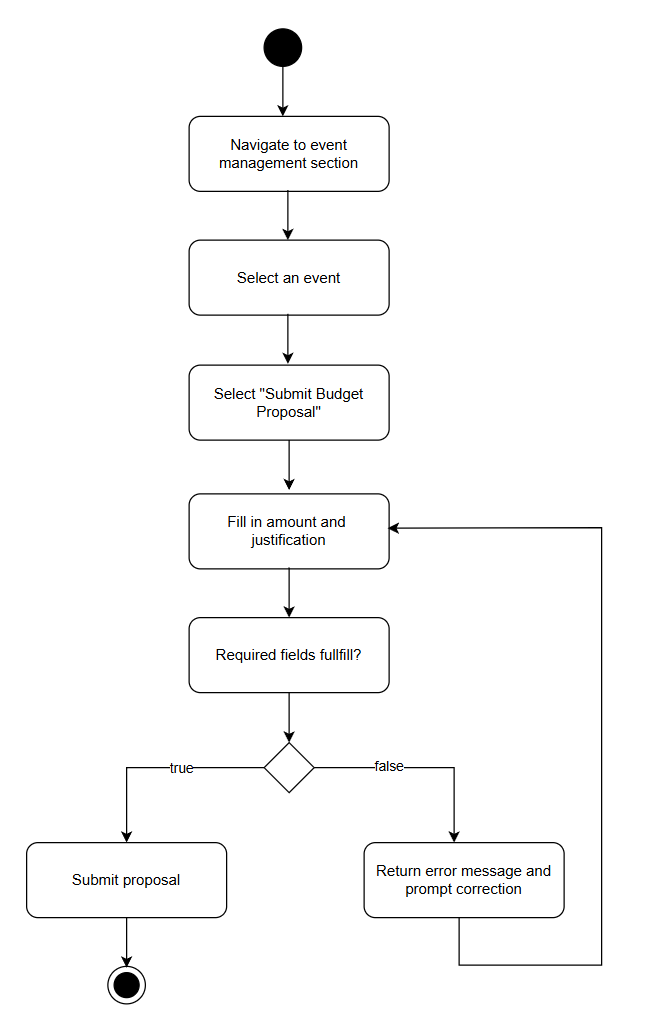
*Table 25: View Event Calendar Use Case Specification*

  
*Figure 3.1.23 View Event Calendar*

### **3.1.24 Submit Budget Proposal**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-024 |
| **Feature** | Submit Budget Proposal |
| **Purpose** | To allow presidents to request funds for event or club operation. |
| **Actor(s)** | Club President |
| **Precondition** | - Event must be created.  - Club President is logged in. |
| **Postcondition** | Budget proposal is submitted for review. |
| **Main Flow** | 1. Navigates to the event management section.  2. Selects an event  3. Select "Submit Budget Proposal".  4. Fills in amount and justification.  5. Submits proposal. |
| **Alternate Scenario** | If required fields are missing, show error and prompt correction. |

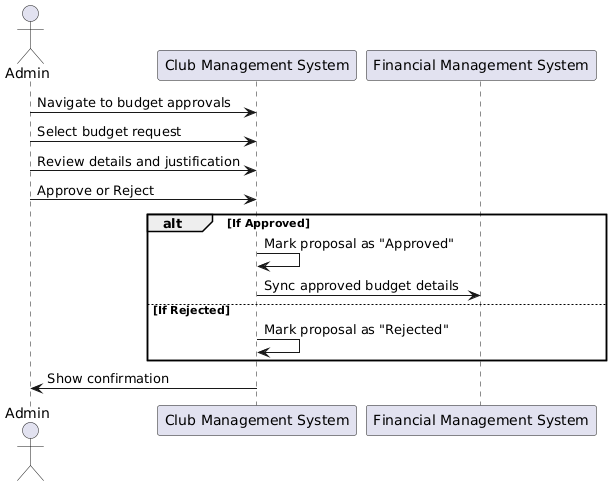
*Table 26: Submit Budget Proposal Use Case Specification*

  
*Figure 3.1.24 Submit Budget Proposal*

### **3.1.25 Approve Budget Proposal**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-025 |
| **Feature** | Approve Budget Proposal |
| **Purpose** | To enable the admin to approve or reject submitted budget requests. |
| **Actor(s)** | 1. Admin  2. Financial Management System (external) |
| **Precondition** | - Event must be created.  - Budget proposal is submitted and pending approval.  - Admin is logged in. |
| **Postcondition** | Budget proposal status is updated by the system |
| **Main Flow** | 1. Navigate to budget approvals.  2. Selects requests.  3. Review the details and justification.  4. Admin chooses to approve or reject proposals.  5. Mark proposal as"Approved" or "Rejected".  6. If approved, the system syncs approval and budget details to the Financial Management System. |
| **Alternate Scenario** | If data is unclear, the admin may request clarification from the club. |

*Table 27: Approve Budget Proposal Use Case Specification*

  
*Figure 3.1.25 Approve Budget Proposal*

#### 

#### 

#### 

#### 

#### 

#### 

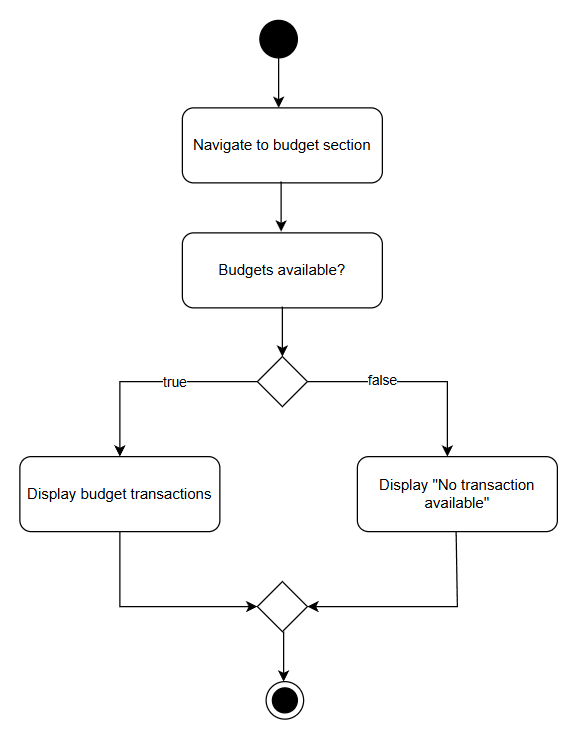
#### 

#### 

### **3.1.26 View Club Budget**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-026 |
| **Feature** | View Club Budget |
| **Purpose** | To allow club members and admin to monitor financial balance and transaction logs. |
| **Actor(s)** | 1.Member  2. Admin |
| **Precondition** | Members or administrators are logged in. |
| **Postcondition** | Budget transactions summary and details are viewed by members or admin. |
| **Main Flow** | 1. Navigates to the budget section.  2. Display budget transactions data. |
| **Alternate Scenario** | If there is no data, show “No transactions available”. |

*Table 28: View Club Budget Use Case Specification*

  
*Figure 3.1.26 View Club Budget*

#### 

#### 

#### 

#### 

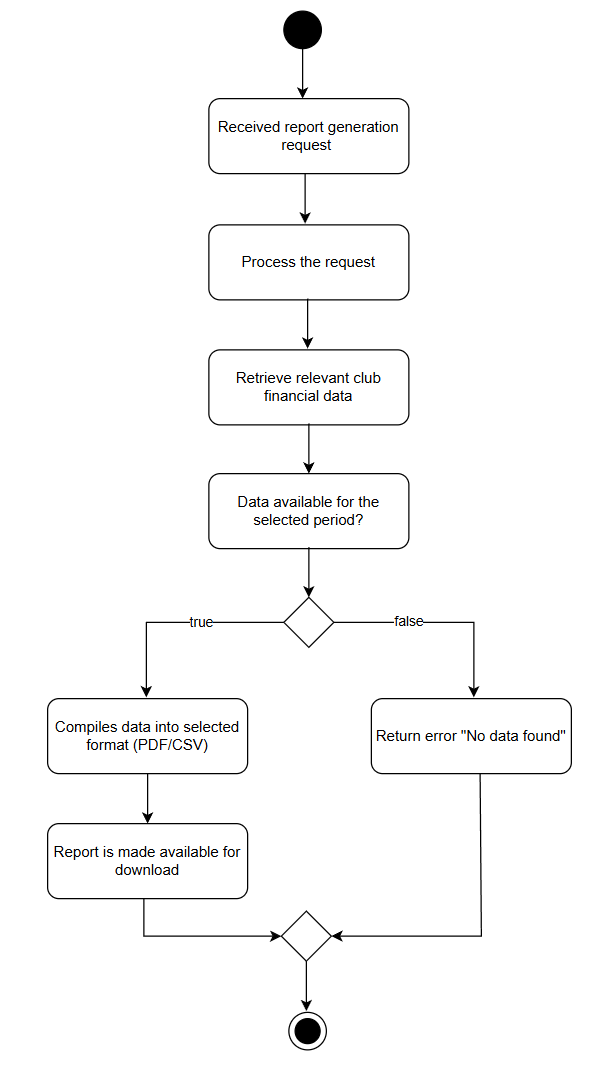
#### 

#### 

### **3.1.27 Generate Financial Report**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-027 |
| **Feature** | Generate Financial Report |
| **Purpose** | To allow the Financial Management System to generate downloadable reports of club financial activity. |
| **Actor(s)** | Financial Management System (external) |
| **Precondition** | Club financial data exists and report generation is requested by an authorized module/user. |
| **Postcondition** | Report file (PDF/CSV) is generated and downloadable. |
| **Main Flow** | 1. A report generation request is received.  2. The Financial Management System (FMS) processes the request.  3. FMS retrieves relevant club financial data.  4. FMS compiles data into selected formats (PDF/CSV).  5. Report is made available for download. |
| **Alternate Scenario** | If no data is available for the selected period, FMS returns an error or "No data found" message. |

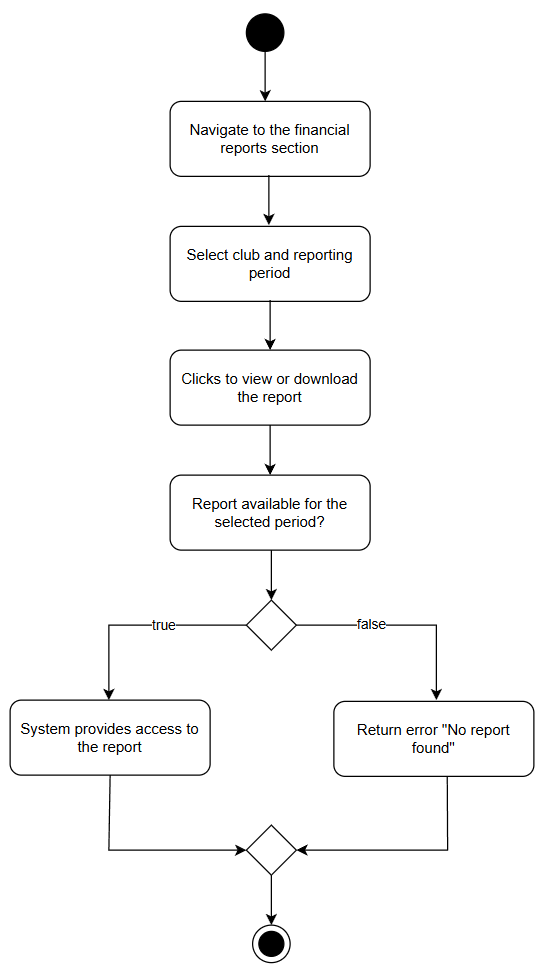
*Table 29: Generate Financial Report Use Case Specification*

  
*Figure 3.1.27 Generate Financial Report*

### **3.1.28 View Financial Report**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-028 |
| **Feature** | View Financial Report |
| **Purpose** | To allow members to view or download previously generated financial reports for their club. |
| **Actor(s)** | Member |
| **Precondition** | - Members are logged in.  - Club financial reports have been generated and stored. |
| **Postcondition** | Member views or downloads the selected financial report. |
| **Main Flow** | 1. Navigates to the financial reports section.  2. Selects club and reporting period.  3. Clicks to view or download the report.  4. System provides access to the report. |
| **Alternate Scenario** | If no report is available for the selected period, the system displays a "No reports found" message. |

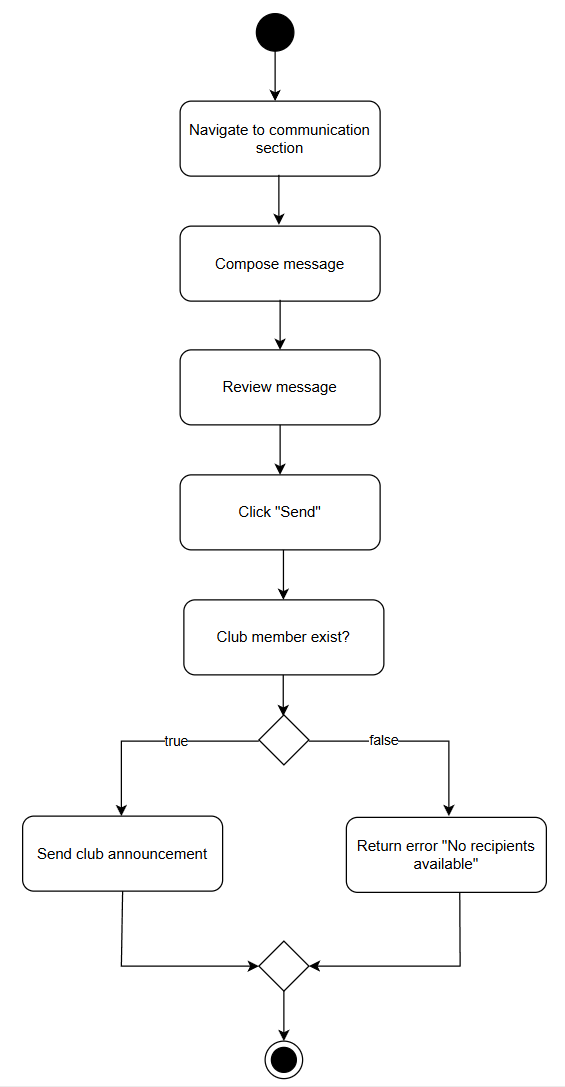
*Table 30: View Financial Report Use Case Specification*

  
*Figure 3.1.28 View Financial Report*

### **3.1.29 Send Club Announcement**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-029 |
| **Feature** | Send Club Announcement |
| **Purpose** | To allow the president or committee to notify club members of updates or events. |
| **Actor(s)** | 1.Club President  2. Committee |
| **Precondition** | Club President or committee is logged in. |
| **Postcondition** | Message is delivered to all club members via system notification or email. |
| **Main Flow** | 1. Navigate to the communication section.  2. Composed message.  3. Review message.  4. Click "Send". |
| **Alternate Scenario** | If no members exist, the system warns “No recipients available”. |

*Table 31: Send Club Announcement Use Case Specification*

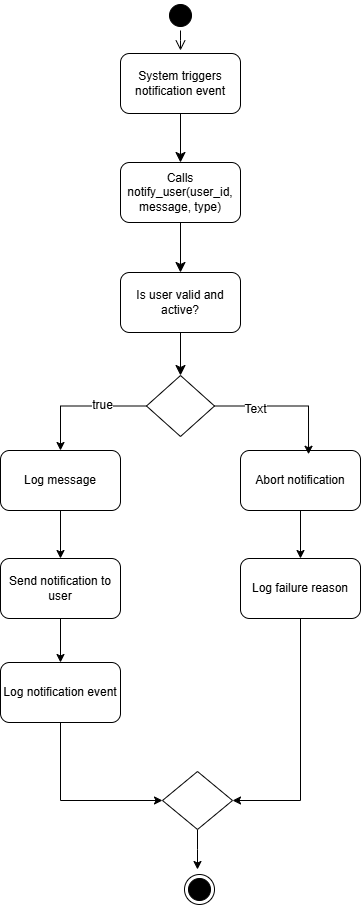
  
*Figure 3.1.29 Send Club Announcement*

#### 

### **3.1.30 Notify User**

| **Field** | **Description** |
| --- | --- |
| **ID** | F-030 |
| **Feature** | Communication & Notifications |
| **Purpose** | To send automated system notifications to specific users when key events occur |
| **Actor(s)** | 1. System (automated trigger)  2.Admin |
| **Precondition** | - User exists in the system  - User is active and eligible to receive notifications  - System services are operational |
| **Postcondition** | - Notification is stored in the system  - User is alerted through email depending on configuration |
| **Main Flow** | 1. System detects event requiring notification  2. Calls notify\_user(user\_id, message, type)  3. Validates user ID  4. Logs message  5. Sends notification to user  6. Log notification event |
| **Alternate Scenario** | If it is invalid user ID , system will abort notification and log failure reason |

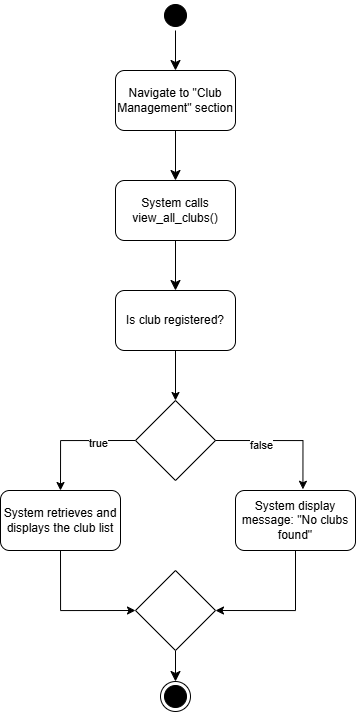
*Table 32: Notify User Use Case Specification*



*Figure 3.1.30 Notify User*

### **3.1.31 View All Club**

| **Field** | **Description** |
| --- | --- |
| **ID** | F-031 |
| **Feature** | View All Club |
| **Purpose** | To allow system admin view a list of all registered clubs in the system. |
| **Actor(s)** | Admin |
| **Precondition** | - Admin user is authenticated and logged in  - User has appropriate admin privileges  - Clubs exist in the system |
| **Postcondition** | - List of all clubs is displayed to the admin  - Admin can view the club's info |
| **Main Flow** | 1. Admin navigates to the club management section  2. System calls view\_all\_clubs()  3. System retrieves and displays club list |
| **Alternate Scenario** | No clubs are registered, system show message: "No clubs found" |

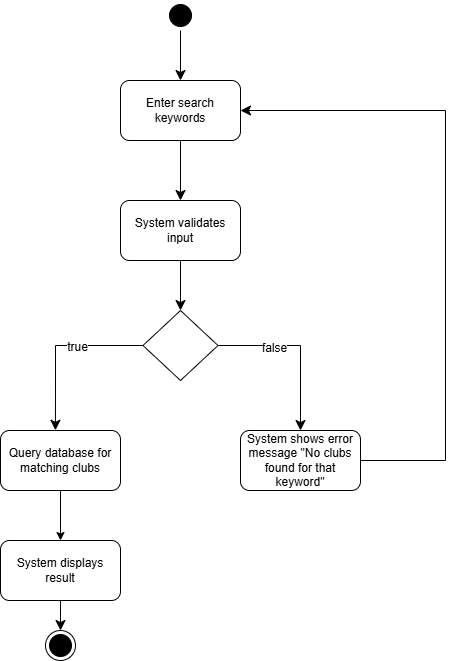


*Figure 3.1.31 View all club*

### **3.1.32 Search Club**

| **Field** | **Description** |
| --- | --- |
| **ID** | F-032 |
| **Feature** | Search Club |
| **Purpose** | To allow all users to search for clubs using keywords that match names, categories, or tags. |
| **Actor(s)** | 1.Student  2. Admin |
| **Precondition** | - Clubs exist in the system  - User has access to the search interface |
| **Postcondition** | - A filtered list of clubs matching the keyword is shown  - User can view details or take further action like join club |
| **Main Flow** | 1. User (student or admin) enters a keyword into the search bar  2. System calls search\_clubs(keyword)  3. Matches club name, category, and tags  4. Displays results list |
| **Alternate Scenario** | No matches found, system show message: “No clubs found for that keyword” and fall back to the previous page. |

*Table 33: Search Club Use Case Specification*

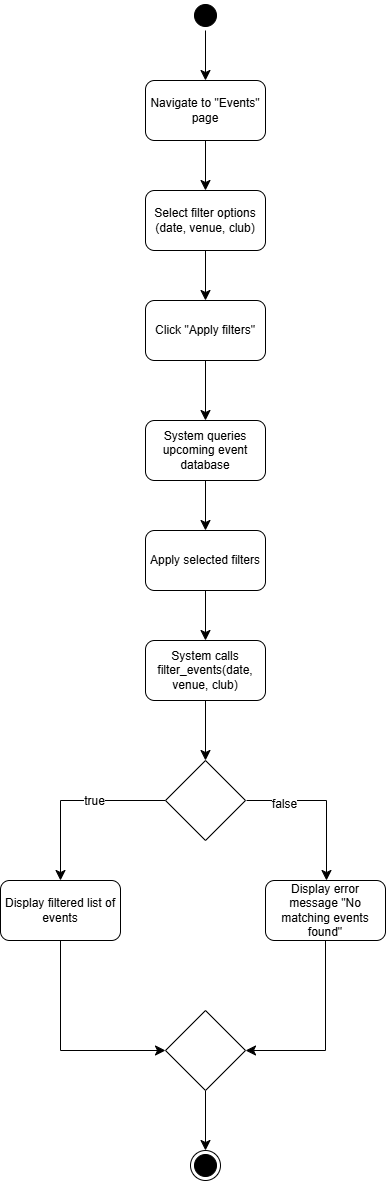


*Figure 3.1.32 Search Club*

### **3.1.33 Filter Event**

| **Field** | **Description** |
| --- | --- |
| **ID** | F-033 |
| **Feature** | Filter event |
| **Purpose** | To allow users to filter upcoming events based on selected criteria such as date, venue, or hosting club. |
| **Actor(s)** | Student |
| **Precondition** | - Events are available in the system  -User is logged in  - User is on the event listing page or has access to filters |
| **Postcondition** | - A filtered list of events is displayed to the user based on chosen criteria |
| **Main Flow** | 1. Navigate to the “Events” page  2. User selects filters: date, venue, or club  3. User clicks “Apply filters”  4. System queries the upcoming event database  5. System calls filter\_events(date, venue, club)  6. Events matching the criteria are retrieved  7. Display filtered list of event |
| **Alternate Scenario** | If there is no events match criteria, system show message: “No matching events found” |

*Table 34: Filter Event Use Case Specification*

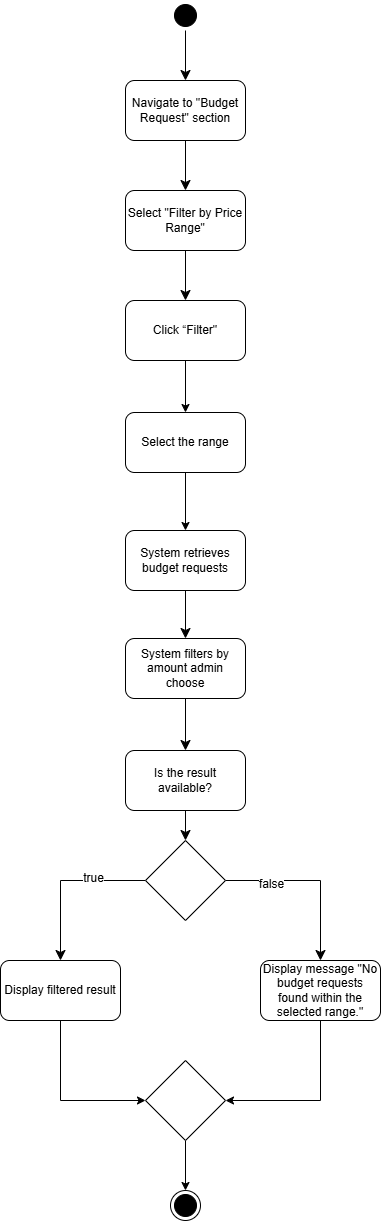


*Figure 3.1.33 Filter Event*

### **3.1.34 Filter Budget Request**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-034 |
| **Feature** | Filter categories of the budget request for the club using price range |
| **Purpose** | Allows admin to efficiently filter and organize budget requests based on specified price ranges for faster decision-making and approval workflows. |
| **Actor(s)** | Admin |
| **Precondition** | The user must be logged in as an Admin and budget requests must exist in the system. |
| **Postcondition** | Budget requests are displayed according to the selected price range, improving readability and enabling focused review. |
| **Main Flow** | 1. Navigates to the Budget Requests section.  2.. Selects "Filter by Price Range".  3. Admin click “Filter”.  4. Inputs or selects a range (e.g., $0–$500).  5. System retrieves budget requests.  6. System filters the amount chosen by the admin.  7. System displays only the matching requests. |
| **Alternate Scenario** | No Matching Requests: System shows “No budget requests found within the selected range.” |

*Table 35: Filter Budget Request Use Case Specification*



*Figure 3.1.34 Filter Budget Request*

### **3.1.35 Log Activity**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-035 |
| **Feature** | Log activity |
| **Purpose** | To automatically track and record significant actions performed by users in the system for audit and security purposes. |
| **Actor(s)** | 1.System  2. Admin |
| **Precondition** | A user performs an action that is deemed auditable such as login and budget approval. |
| **Postcondition** | A log entry is stored in the audit trail with the user ID, action details, and timestamp. |
| **Main Flow** | 1. User performs a key action in the system.  2. System event listener triggers.  3. System captures user ID and action.  4. Records a timestamp of the action.  5. Stores the entry in the audit log. |
| **Alternate Scenario** | If the action has no need to log, the system will ignore it. |

*Table 36: Log Activity Use Case Specification*

#### 

*Figure 3.1.35 Log Activity*

### **3.1.36 View Audit Log**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-036 |
| **Feature** | View audit log |
| **Purpose** | To allow system administrators to access and review system activity logs for monitoring, security audits, and issue investigation. |
| **Actor(s)** | Admin |
| **Precondition** | Admin is authenticated and logged in and has appropriate permissions to access system logs. |
| **Postcondition** | The system displays a list of recorded user activities with details such as user ID, action, and timestamp. |
| **Main Flow** | 1.. Navigates to the “audit log” section.  2.. System retrieves the data.  3.. System displays activity logs. |
| **Alternate Scenario** | If no logs are available, a message is shown indicating “No audit logs found.” |

*Table 37: View Audit Log Use Case Specification*

#### 

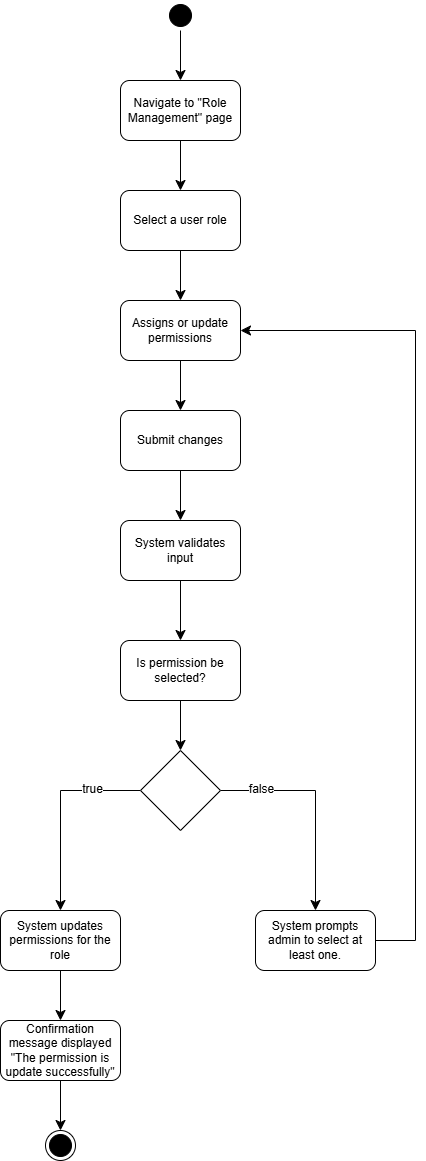
#### 

*Figure 3.1.36 View Audit Log*

### **3.1.37 Set Permission**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-037 |
| **Feature** | Set permission |
| **Purpose** | To allow system administrators to define or update the actions that each user role is authorized to perform within the system. |
| **Actor(s)** | Admin |
| **Precondition** | Admin is authenticated and logged in and has access to the role management module. |
| **Postcondition** | Permissions for the specified role are updated and enforced across the system. |
| **Main Flow** | 1. Navigates to the role/permission settings.  2. Selects a role.  3. Assigns or updates permissions.  4. System saves the changes. |
| **Alternate Scenario** | If no permissions are selected, the system prompts the admin to select at least one and fall back to the section. |

*Table 38: Set Permission Use Case Specification*

**

*Figure 3.1.37 Set permission*

### **3.1.38 Generate Venue Status Report**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-038 |
| **Feature** | Generate Venue Status Report |
| **Purpose** | Automatically generates the current status of all venues for specified time slots for Admin review and approval. |
| **Actor(s)** | Campus Space Reservation System |
| **Precondition** | The Admin must be logged in, and there must be existing venue requests or bookings in the system. Also, event content is approved. |
| **Postcondition** | A list of venue statuses (available, pending, booked) is presented to the Admin, enabling follow-up actions like approval. |
| **Main Flow** | 1. Admin navigates to the Venue Management page.  2. System sends a request to the Campus Space Reservation System.  3. Facility Management System checks venue bookings.  4. Compile the available and unavailable time slots.  5. Generate venue status report. |
| **Alternate Scenario** | If an external API fails, an error message “System unavailable or data corrupt is displayed. |

*Table 39: Generate Venue Status Report Use Case Specification*

#### 

#### 

*Figure 3.1.38 Generate Venue Status Report*

### **3.1.39 View Venue Status**

| **Field** | **Details** |
| --- | --- |
| **ID** | F-039 |
| **Feature** | View Venue Status |
| **Purpose** | Allows the Admin to view real-time venue availability before making decisions. |
| **Actor(s)** | Admin |
| **Precondition** | - Admin must be logged in.  - Venue booking requests must exist.  - Venue status report is available |
| **Postcondition** | Admin sees updated venue availability |
| **Main Flow** | 1. Admin navigates to the Venue Status section.  2. Selects a request to review.  3. Select date and time slots.  4. System generates a venue status report.  5. View generated status report. |
| **Alternate Scenario** | If there is no data found, the system will display an error message “No venue requests found”. |

*Table 40: View Venue Status Report Use Case Specification*

### 

*Figure 3.1.39 View Venue Status*

## **3.2 Performance Requirements**

### **3.2.1. Dynamic Actions or Changes:**

**a. Data Synchronization Rate:**

* The system must synchronise data between the student club management platform, the university's money system, and the campus venue booking system in real or near real time. This includes funding request updates, budget approvals, and venue reservation confirmations.
* The maximum allowable latency for synchronisation between systems was less than 5 seconds.

**b. User Interaction Velocity:**

* The platform must be able to handle heavy concurrent usage by students, club leaders, and university administration without causing visible performance reduction. The goal is to support up to 5,000 concurrent users with sub-second response times.

**c. Content Loading Speed:**

* All dashboards, event pages, member lists, and administrative interfaces must load within 2 seconds for 95% of user interactions, ensuring a seamless and responsive user experience.

**d. Notification Delivery Rate:**

* System-triggered messages, such as event reminders, budget status updates, and venue booking alerts, must be given to the appropriate users within 10 seconds of initiation.

### **3.2.2. Quantitative Criteria:**

**a. Endurance Capabilities:**

* The system must be operational 24/7 with an uptime of 99.9%, excluding scheduled maintenance.
* Scheduled maintenance windows should not exceed one hour per month and must take place during off-peak hours.
* The platform must have a minimum operational life of 5 years with regular updates, as well as the ability to seamlessly upgrade software and perform routine maintenance without disturbing operations.

**b. Session Duration and Utilization Rate:**

* Average user session duration is expected to be around 20 minutes, particularly during event planning or budgeting activities.
* The system should handle peak loads where up to 50% of all registered students and 100% of active club leaders and administrative users are simultaneously active, without impacting system performance.

### **3.2.3. Performance Requirements for Operational Phases and Modes:**

**a. Operational Phases:**

* Normal Use: The platform must efficiently enable actions such as member registration, event creation, and financial tracking with low latency and consistent responsiveness.
* Peak Operation: During times of high activity (e.g., semester starts, budget submission deadlines, campus-wide events), the system must dynamically scale to handle up to double the regular demand while maintaining performance criteria for all services.
* Maintenance Mode: Users should be advised of maintenance windows at least 24 hours in advance. The system must switch to read-only mode, allowing access to key information such as event calendars, club contacts, and policy documents while blocking content creation and editing capabilities.

**b. Modes:**

* Read Mode: Users must be able to explore events, club profiles, member directories, and historical data with a response time of less than two seconds.
* Write Mode: Actions including submitting funding requests, updating member information, and establishing events must be completed within 3 seconds.
* Analytical Mode: Administrators and club leaders should receive analytics dashboards and downloadable reports such as participation metrics, budget utilisation, and booking statistics within 5 seconds of a request.

## **3.3 Usability Requirements**

The Student Club Management System with Budget and Venue Integration is expected to provide excellent usability, dependability, and performance, ensuring a smooth and consistent user experience for all user groups, including students, club leaders, and administrators. The platform must be fast and responsive, with all pages and user activities loading in less than two seconds under normal usage settings. Special emphasis is placed on availability during important periods, such as event registration phases and budget submission deadlines, when performance and stability are essential.

In addition to performance, the system must maintain data correctness and integrity, particularly for important processes like financial tracking and venue reservations. Real-time data validation for transactions and updates is required to avoid data discrepancies and reduce administrative follow-ups. The interface should be clear, intuitive, and accessible to users with diverse technical backgrounds, allowing them to easily use the site.Then future scalability, both in terms of user base and functional modules, must be planned to support growing usage and new feature integrations. Besides, strict security, privacy, and role-based access controls are required to protect confidentiality and abide with university policies.

### **3.3.1 Performance**

The system must provide responsive performance in all operational conditions. Under typical circumstances, all user operations, such as website navigation, form submissions, and report production, should take no more than two seconds. During peak periods, such as the start of a semester or a university-wide event week, the system must grow to accommodate up to 5,000 concurrent users while maintaining performance. This includes the timely display of club dashboards, budget progress updates, and venue availability. Real-time notifications, calendar synchronisation, and financial approvals must all reply in less than five seconds to provide a smooth and user-friendly experience. These requirements are critical for maintaining user engagement and lowering the likelihood of job abandonment or repeat submissions.

### **3.3.2 Security**

Security is a primary issue for the system, especially given the processing of sensitive user information and financial records.Then. the platform must include strong authentication mechanisms, such as university single sign-on (SSO) integration and multi-factor authentication (MFA) for high-priority positions like administrators and club treasurers. All data transmission must be protected with HTTPS (SSL/TLS), and sensitive data such as personal information, budgetary details, and club money must be securely encrypted. Role-based access control must be carefully implemented to prevent unauthorised access to sensitive processes like fund request approvals and member management. The system must also adhere to relevant institutional IT rules and, when necessary, external standards such as ISO/IEC 27001 for information security.

### **3.3.3 Reliability**

The platform must maintain reliability with all functions functioning properly under normal and expected situations. To avoid cascade failures, system components should be designed using fault tolerance concepts.The platform must go through extensive testing, including unit, integration, and stress testing, to assure its reliability before deployment. All user inputs must be verified, and error handling systems must gracefully handle unexpected behaviour without crashing or corrupting data.Moreover, logging and alerting methods must be implemented to discover, report, and escalate system anomalies or failures in real time.Thus, the reliability criteria at the time of delivery is less than one error per 10,000 transactions.

### **3.3.4 Availability**

The system must have at least 99.9% uptime, excluding scheduled maintenance. This is equivalent to no more than 43 minutes of unexpected downtime every month. To fulfil this availability goal, the platform should have checkpoint and recovery methods that allow it to promptly restart operations after an unexpected shutdown.Besides, regular backups of both system data and configuration settings should be performed, with rollback possibilities in case of unsuccessful upgrades or system failures. During maintenance periods, the system must transition to a read-only mode, allowing users to view important data such as event details and club information. Therefore, notifications must be sent ahead of any planned outage to ensure transparency and minimise user inconvenience.

### **3.3.5 Maintainability**

The system must be highly maintainable in order to provide efficient updates, bug corrections, and feature upgrades with minimal disturbance. Furthermore, a modular architecture should be used to ensure that modifications to one component, such as budgeting or event management, have no impact on others. Then, source code must adhere to clean coding standards, be well-commented, and be backed up by current documentation.Moreover, automated testing and version control should be implemented to ensure the safe deployment of changes.Lastly, rollback mechanisms and change logs must be established to enable for speedy recovery from failed updates while preserving the system's long-term adaptability and stability.

## **3.4 Interface Requirements**

This section shall be organized into the following subsections to define the requirements imposed on various system components, such that external systems, user-facing modules, and manual operations in order to establish and maintain the necessary interfaces for the SCMS platform. Each requirement is assigned with an unique identifier to support verification, testing, and traceability throughout the development lifecycle.

### **3.4.1 System Interfaces**

| **Interface ID** | **Interface Description** | **Priority** |
| --- | --- | --- |
| IR\_SYSI001 | The system shall integrate with the university’s financial management system to retrieve and update student club budget data. | High |
| IR\_SYSI002 | The system shall connect to the campus venue reservation system for venue availability checks, submit bookings, and display confirmation details. | High |

*Table 41: System Interfaces Requirements*

#### 

### **3.4.2 User Interfaces**

| **Interface ID** | **Interface Description** | **Priority** |
| --- | --- | --- |
| IR\_UI001 | The web interface shall use a responsive design with a fixed top navigation bar including links to Dashboard, Clubs, Events, Budget, and Venue modules. | High |
| IR\_UI002 | The user interface shall include a collapsible side panel for module navigation and access to personalized settings. | Medium |
| IR\_UI003 | Forms used for data input shall include field validation, date pickers, dropdown menus, and tooltips for ease of use. | High |
| IR\_UI004 | The interface shall dynamically change based on user roles such as Student, Member, Committee, Club President, and Admin. | High |
| IR\_UI005 | Notifications and alerts shall be displayed using modal pop-ups and dashboard summaries. | Medium |
| IR\_UI006 | The UI shall comply with WCAG 2.1 accessibility standards such as contrast ratio, screen reader compatibility. | Medium |
| IR\_UI007 | The system shall provide a login and registration page accessible from the landing page, with entry fields for the username and password. | High |
| IR\_UI008 | The user interface shall support both light and dark GUI themes, allowing users to toggle themes from their profile settings. | Medium |

*Table 42: User Interfaces Requirements*

### **3.4.3 Hardware Interfaces**

| **Interface ID** | **Interface Description** | **Priority** |
| --- | --- | --- |
| IR\_HWI001 | The system shall be accessible from devices such as desktops, laptops, tablets, and smartphones. | High |
| IR\_HWI002 | Devices shall have minimum specs: 64-bit CPU, 2 GB RAM, and 1024×768 resolution. | High |
| IR\_HWI003 | The system shall support full-screen and touch interactions on mobile devices. | Medium |

*Table 43: Hardware Interfaces Requirements*

### **3.4.4 Software Interfaces**

| **Interface ID** | **Software** | **Interface Description** | **Priority** |
| --- | --- | --- | --- |
| IR\_SWI001 | LDAP Authentication Service | Used to authenticate users and assign roles via secure LDAP or SSO login mechanisms. | High |
| IR\_SWI002 | University Financial API | A REST API for submitting and tracking budget requests and approvals. Uses JSON format with OAuth2 authentication. | High |
| IR\_SWI003 | Campus Venue Reservation API | REST or SOAP API for managing venue availability and reservations. Supports JSON or XML formats. | High |
| IR\_SWI004 | PostgreSQL Database | Internal database for storing club information, transactions, members, and event logs. Accessed through JDBC. | High |
| IR\_SWI005 | Web Browsers (Chrome, Edge, Safari) | Supported platforms must have JavaScript and cookie support enabled. | High |

*Table 44: Software Interfaces Requirements*

### **3.4.5 Communication Interfaces**

| **Interface ID** | **Interface Description** | **Priority** |
| --- | --- | --- |
| IR\_COMI001 | All system communication shall occur over HTTPS using TLS 1.2 or higher to ensure secure data transfer. | High |
| IR\_COMI002 | All API messages exchanged with external systems, such as finance and venue shall use RESTful protocols in JSON format. | High |
| IR\_COMI003 | The system shall support OAuth 2.0 or SAML-based authentication for secure API access to university systems. | High |
| IR\_COMI004 | The system shall use SMTP over TLS for sending email notifications to users. | Medium |

*Table 45: Communication Interfaces Requirements*

### 

## **3.5 Logical Database Requirements**

<https://drive.google.com/file/d/1b3WAfg13vIQFF9Mfp7gwQTXTkrZIQRoa/view?usp=sharing>

#### 

*Figure 3.5 Class Diagram*

The main classes in the Student Club Management System include User, Club, Event, Announcement, VenueRequest, BudgetRequest, and ClubFinance. The User class serves as the base for system roles and is extended by Student, Member, Committee, President, and Admin, allowing role-specific functionalities while maintaining shared user attributes.

The Club class manages club-level details and maintains relationships with its members, events, and finances. Each Event is associated with a venue request and can have announcements and committees managing it. BudgetRequest handles financial proposals submitted by club presidents and reviewed by admins, while ClubFinance tracks the club’s fund balance and transaction records.

The system integrates with external entities—CampusSpaceReservation for venue availability and booking, and FinancialManagementSystem for budget synchronization. These are modeled as external classes to represent communication interfaces, not internal data storage.

Relationships across the system are mostly associations, with composition or aggregation used to reflect data dependencies (e.g., an event belongs to a club). This class structure ensures modularity, supports user-role expansion, and enables automated handling of core student club processes like membership management, budget tracking, and venue booking.

### **3.5.1 User Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| userID | Unique identifier for each user | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| username | Login username | VARCHAR(50) | NOT NULL, UNIQUE | Login Credentials |
| email | User email address | VARCHAR(100) | NOT NULL, UNIQUE |  |
| password | Hashed password | VARCHAR(255) | NOT NULL | Login Credentials |
| role | Role of the user in system | ENUM('Student', 'Member', 'Committee', 'President', 'Admin') | NOT NULL | Controls access and permissions |

*Table 46: User Class Data Dictionary*

### **3.5.2 Club Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| clubID | Unique identifier for each club | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| clubName | Name of the club | VARCHAR(100) | NOT NULL, UNIQUE |  |
| description | Club description | TEXT |  |  |
| createdDate | Date of club creation | DATE | NOT NULL |  |
| status | Approval status | ENUM('Pending', 'Approved', 'Rejected') | DEFAULT 'Pending' |  |
| createdBy | User ID of the user who created the club | UUID | FOREIGN KEY REFERENCES User(userID) | Usually a President |

*Table 47: Club Class Data Dictionary*

### **3.5.3 Event Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| eventID | Unique identifier for each event | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| clubID | ID of the club organizing the event | UUID | FOREIGN KEY REFERENCES Club(clubID) |  |
| title | Name of the event | VARCHAR(100) | NOT NULL |  |
| description | Description of the event | TEXT |  |  |
| date | Date and time of the event | DATETIME | NOT NULL |  |
| status | Approval status | ENUM('Pending', 'Approved', 'Rejected') | DEFAULT 'Pending' |  |
| venueID | Requested venue ID | UUID | FOREIGN KEY → VenueRequest  (venueRequestID) | Linked to venue requests |

*Table 48: Event Class Data Dictionary*

### **3.5.4 BudgetRequest Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| budgetID | Unique identifier for the budget request | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| clubID | ID of the club submitting request | UUID | FOREIGN KEY REFERENCES Club(clubID) |  |
| eventID | ID of the event for the budget request | UUID | FOREIGN KEY REFERENCES Event(eventID) |  |
| amountRequsted | Requested budget amount | DECIMAL(10,2) | NOT NULL |  |
| purpose | Purpose of the budget | TEXT |  |  |
| status | Approval status | ENUM('Pending', 'Approved', 'Rejected') | DEFAULT 'Pending' |  |
| submittedDate | Date of the request submission | DATE | NOT NULL |  |

*Table 49: BudgetRequest Class Data Dictionary*

### **3.5.5 VenueRequest Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| venueRequestID | Unique identifier for venue request | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| eventID | Associated event ID | UUID | FOREIGN KEY REFERENCES Event(evenID) |  |
| venueName | Requested venue | VARCHAR(100) | NOT NULL |  |
| requestedDate | Requested date | DATE | NOT NULL |  |
| status | Request status | ENUM('Pending', 'Approved', 'Rejected') | DEFAULT 'Pending' |  |

*Table 50: VenueRequest Class Data Dictionary*

### **3.5.6 Announcement Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| announcementID | Unique identifier | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| title | Title of the announcement | VARCHAR(150) | NOT NULL |  |
| message | Full announcement text | TEXT | NOT NULL |  |
| clubID | Club to which the announcement belongs | UUID | FOREIGN KEY → Club(clubID) |  |
| postedDate | Timestamp of creation | DATE | NOT NULL |  |

*Table 51: Announcement Class Data Dictionary*

### **3.5.7 Membership Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| membershipID | Unique identifier | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| userID | User ID | UUID | FOREIGN KEY → User(userID) |  |
| clubID | Club ID | UUID | FOREIGN KEY → Club(clubID) |  |
| roleInClub | Role in the club | ENUM | NOT NULL | 'Member', 'Committee', 'President' |
| joinedDate | Date the user joined the club | DATE | NOT NULL |  |

*Table 52: Membership Class Data Dictionary*

### **3.5.8 ClubFinance Class**

| **Field Name** | **Description** | **Data Type** | **Constraints** | **Extra Notes** |
| --- | --- | --- | --- | --- |
| financeID | Unique identifier | UUID | PRIMARY KEY, AUTO\_INCREMENT |  |
| clubID | Associated club ID | UUID | FOREIGN KEY → Club(clubID) |  |
| balance | Balance amount of money of the club | FLOAT | NOT NULL |  |
| lastUpdated | Date of the latest update on the finance report | DATE | NOT NULL |  |

*Table 53: ClubFinance Class Data Dictionary*

## **3.6 Design Constraints**

### **3.6.1 External Standards and Regulatory Requirements**

1. **Data Privacy and Security:**
2. **FERPA Compliance:** The platform must comply with the Family Educational Rights and Privacy Act to protect students' educational and activity records.
3. **University Data Governance**: The system must follow internal policies regarding the collection, storage, retention, and access of student data.
4. **Role-Based Access:** Data visibility must be restricted according to user roles (e.g., students, club presidents, admins).
5. **Audit Logging:** All access to sensitive personal data must be logged for accountability and review.
6. **Security Standards:**
7. **OWASP Top 10 Compliance:** The system must be protected against common vulnerabilities such as SQL injection, XSS, CSRF, and broken authentication.
8. **HTTPS Protocol:** All data exchanges between clients and servers must be encrypted using HTTPS with valid SSL/TLS certificates.
9. **Secure Coding Practices:** Input validation, session management, and secure authentication mechanisms must be implemented to reduce risk.
10. **HSTS Enforcement:** HTTP Strict Transport Security headers should be used to ensure all communication is securely routed.
11. **Accessibility Standards:**
12. **WCAG 2.1 Level AA Compliance:** The platform must be accessible to users with disabilities by adhering to Web Content Accessibility Guidelines.
13. **Keyboard Navigation:** All interactive elements must be operable without a mouse.
14. **Screen Reader Support:** Pages must use semantic HTML and ARIA roles for compatibility with assistive technologies.
15. **Responsive Design:** The interface must adapt fluidly to different screen sizes and orientations.

### **3.6.2 Project Limitations**

1. **Time Constraint**
2. The platform must be completed within one academic semester which is approximately 4–5 months.
3. Due to limited time, complex features such as analytics, AI, or predictive dashboards are not in scope.
4. **Budget Constraint**
5. The project must utilize free and open-source tools and platforms.
6. No commercial licensing or paid APIs are allowed due to limited funding.
7. **Resource Availability**
8. The development team is likely composed of students or part-time contributors with varying skill levels.
9. Testing feedback from real stakeholders (students, admins) may be limited due to academic scheduling.
10. **Stakeholder Input**
11. Availability for feedback and user testing from actual stakeholders (e.g., student club members, event approvers, venue managers) may be constrained due to conflicting academic and administrative responsibilities.

### **3.6.3 Technical Constraints**

1. **Technology Stack**
2. The backend must use the Django (Python) framework as per university guidelines.
3. Frontend technologies must be lightweight and compatible with Django templates.
4. **Database Technology**
5. Only PostgreSQL and SQLite are approved for database implementation due to internal IT restrictions.
6. **Integration Requirement**
7. The system must integrate with the university’s Single Sign-On (SSO) solution via OAuth2 or LDAP for secure user authentication.
8. **API Usage**
9. Any integration with third-party APIs such as Google Calendar, email services must adhere to the university’s data privacy and security policies.

### 

### **3.6.4 Platform Constraints**

1. **Hosting Environment**
2. The platform must be deployable on the university’s Linux-based infrastructure (on-premise or cloud-hosted), limiting the operating system and server stack configurations.
3. **Client Compatibility**
4. The system must be accessible through all major modern browsers, including Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge, without relying on any browser-specific features.
5. **Mobile Access**
6. A fully native mobile application is not in scope. The platform must use responsive web design to ensure usability across smartphones and tablets.
7. **Access Restrictions**
8. Access to the system must be restricted to authenticated university members (students, faculty, staff), with features and permissions tailored to their assigned roles.

### **3.6.5 Environmental Constraints**

1. **Network Dependency:**
2. The platform will require reliable internet access to function; offline features are not a requirement.
3. **Data Privacy Compliance:**
4. Data handling must align with FERPA and university-specific privacy policies, including data retention and access control rules.
5. **User Access Context:**
6. The system must perform reliably across both on-campus and remote environments, accounting for variable network speeds and firewall conditions.
7. **Maintenance Scheduling:**
8. System updates must be planned during non-peak hours (e.g., evenings or weekends) to avoid disrupting ongoing student or administrative activities.

## **3.7 Software System Attributes**

**I. Scalability**

1. **Horizontal Growth Support**
2. The system must be designed to support an increasing number of users, clubs, events, and transactions without requiring major architectural changes.
3. **Load Distribution**
4. System components such as authentication, event scheduling, and budget management should be independently scalable to handle peak load periods (e.g., club registration weeks).
5. **API and Data Scalability**
6. APIs and database queries must be optimized to prevent bottlenecks during high-volume interactions such as bulk approvals or calendar views.

**II. Responsiveness**

1. **Fast User Interaction**
2. UI elements should respond within 500 milliseconds for most actions like button clicks, tab navigation, and form validation.
3. **Page Load Performance**
4. All primary pages (dashboard, event list, profile) must load within 3 seconds under standard conditions.
5. **Lightweight Frontend**
6. The frontend should minimize resource usage using lazy loading, AJAX updates, and asynchronous form submissions.

**III. Security**

1. Session Management
2. Sessions must expire after inactivity and use secure session tokens. Re-authentication should be required for critical actions.
3. Role Isolation
4. The system must strictly separate user roles (student, committee, admin) to prevent privilege escalation.
5. Secure Data Transactions
6. Budget submissions, event approvals, and user profile edits must be protected with input sanitization and transaction validation.

**IV. Usability**

1. Intuitive Navigation
2. The UI must support easy navigation with clear labels, minimal clicks to reach key features, and consistent layout across pages.
3. Accessible Form Design
4. All forms must include helper text, real-time validation feedback, and tooltips to improve user experience.
5. Feedback and Notifications
6. Users must receive real-time system feedback (e.g., success/error messages) and relevant in-app/email notifications.

**V. Interoperability**

1. External System Integration
2. The system must support API-based integration with existing university systems such as financial software, facility booking tools, or attendance tracking.
3. Open Standards Compliance
4. Data exchange should use standard formats (e.g., JSON, CSV) and RESTful APIs for compatibility with third-party tools.
5. Calendar and Email Integration
6. Seamless syncing with services like Google Calendar, Outlook, and campus email systems is required to coordinate events and notify users.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **3.8 Supporting Information**

Throughout the elicitation phase, three elicitation methods were used to understand user needs and requirements for the Student Club Management System (SCMS). Below are the elicitation techniques that were used, along with the corresponding results and analyses.

### **3.8.1 Brainstorming**

A brainstorming session is conducted with stakeholders, including student and software developer, to identify core functionalities and pain points in Student Club Management System (SCMS). The figure below is the screenshot indicating when the session is conducted.

|  |
| --- |

***Figure 3.8.1: Brainstorming Session***

Session Information:

| Date & Time: 2/5/2025, 10:00pm  Participants: Teow Wei Ting (Note-taker)  Eng Zi Ying (Note-taker)  See Jie Sheng (Facilitator)  Sua Wei Khong (Time-keeper) |
| --- |

The flow of the brainstorming session is as follows:

1.Introduction (0–5 min)

* Facilitator explains the session’s purpose.

2.Scope Recap (5–10 min)

* Quick review of project modules. For example, event management and budget tracking.
* Highlight key constraints.

3.Silent Idea Generation (10–30 min)

* Each member writes ideas independently in their Google Doc tab.
* Focus on quantity; no discussion yet.

4.Group Sharing & Discussion (30–50 min)

* Members present ideas one by one.
* Group clusters have similar ideas.
* Discuss feasibility and dependencies briefly.

5.Prioritization (50–60 min)

* Dot voting or team ranking to select top ideas.

6.Feasibility Check (60–70 min)

* Identify technical/logistical challenges for top ideas.
* Note down the dependencies.

7.Wrap-up (70–75 min)

* Summarize key ideas and decisions.
* Document down the prioritized features.

The figure below is the analysis and result elicited from brainstorming session about general features on Student Club Management System (SCMS).

| **Feature** | **Description** | **Requirement ID** |
| --- | --- | --- |
| User Account Management | Basic user account features, including registration, login, logout, and recovery | F-001, F-002, F-003, F-004 |
| Club Management | Allows users to create, modify, and delete clubs; includes approval workflows | F-007, F-008, F-011, F-012 |
| Event Management | Requesting, updating, and canceling events initiated by users or clubs | F-015, F-016, F-017 |
| System Integration | Integration with university finance and venue systems | F-022, F-027 |

***Figure 3.8.1.1: Requirements elicited from Brainstorming***

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **3.8.2 Perspective-based Reading**

A perspective-based reading session is conducted to gather and refine software requirements by having team members research related articles or case studies on student club management systems and analyze them from assigned stakeholder perspectives. Each team member is assigned a stakeholder perspective and conducts article-based research. The figure below is the screenshot indicating when the session is conducted.

Session Information:

| Article Reviewed:   * Design architecture of an Integrated Student Activities Management System for higher education * Campus Club Management System Application   Date & Time: 10/5/2025, 10:00pm  Participants: Teow Wei Ting (Student)  Eng Zi Ying (Admin)  See Jie Sheng (Finance Officer)  Sua Wei Khong (Venue Manager) |
| --- |

| Article Reviewed:   * An Enhanced Django Framework for effective * Club Connect Management System * College Club activity management system   Date & Time: 15/5/2025, 8:00pm  Participants: Teow Wei Ting (Student)  Eng Zi Ying (Admin)  See Jie Sheng (Finance Officer)  Sua Wei Khong (Venue Manager) |
| --- |

The figure below is the analysis and result elicited from a perspective-based reading session about some advanced features and hardware configuration on Student Club Management System (SCMS).

| **Feature** | **Description** | **Requirement ID** |
| --- | --- | --- |
| Search & Filtering | Enables users and admins to search and filter clubs, events, and budgets | UC-SEARCH-01, UC-SEARCH-02, UC-FIN-005 |
| Technical Communication | Security protocols and authentication for secure API/data access | IR\_COMI001, IR\_COMI002, IR\_COMI003, IR\_COMI004 |
| Hardware & Compatibility | Minimum device specs and supported platforms | IR\_HWI001, IR\_HWI002, IR\_HWI003 |
| System Integration | Use of LDAP and APIs for financial and venue services | IR\_SWI001, IR\_SWI002, IR\_SWI003, IR\_SWI004, IR\_SWI005 |
| User Interface | Interface preferences including fixed navbar, collapsible side panel, and theming | IR\_UI001, IR\_UI002, IR\_UI008, IR\_UI007 |

***Figure 3.8.2.1: Requirements elicited from Perspective-based Reading***

### 

### 

### 

### 

### 

### 

### 

### **3.8.3 Questionnaire**

A questionnaire of 12 questions is created and distributed to 30 respondents. The group of respondents could be university Student, Member, Committee, Club President, Admin Staff and Student Affairs Department. The questionnaire focuses on collecting structured feedback and requirements from key stakeholders regarding the desired features of Student Club Management System from the respondents.

Below are the result and demographic for the questionnaire:

|  |
| --- |

***Figure 3.8.3.1: Question 1 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.2: Question 2 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.3: Question 3 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.4: Question 4 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.5: Question 5 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.6: Question 6 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.7: Question 7 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.8: Question 8 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.9: Question 9 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.10: Question 10 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.11: Question 11 in Questionnaire and its Results***

|  |
| --- |

***Figure 3.8.3.12: Question 12 in Questionnaire and its Results***

The figure below is the analysis and result elicited from the questionnaire distributed to the targeted respondents. It is to gather the usability preferences and identify critical user-experience features on Student Club Management System (SCMS).

| **Feature** | **Description** | **Requirement ID** |
| --- | --- | --- |
| Event & Venue Workflow | Event and venue request and approval process | UC-EVENT-04, UC-EVENT-05, UC-EVENT-06 |
| Event Viewing | Viewable calendar of upcoming events | UC-EVENT-07 |
| Communication | Sending announcements and system-generated user notifications | UC-COMM-01, UC-COMM-02 |
| Budget Management | Full budget workflow from submission to reporting | UC-BUDGET-01, UC-BUDGET-02, UC-BUDGET-03, UC-BUDGET-04, UC-BUDGET-05 |
| Audit & Permission | System activity tracking, logs, and role-based access control | UC-AUDIT-01, UC-AUDIT-02, UC-AUDIT-03 |

***Figure 3.8.3.13: Requirements elicited from Questionnaire***

### 

### 

### 

### 

### 

### 

### 

### **3.8.4 Interview**

On 22nd May 2025, 3:00pm, we conducted an interview with Wong Jun Jie, an experienced student club leader actively involved in organizing campus events. This interview focuses on understanding user satisfaction and expectations across three categories: Must-be (Dissatisfiers), Performance (Satisfiers), and Delighters (Exciters).

| Date & Time: 22/5/2025, 3:00pm  Location: FCI  Interviewed result:   * Users expect basic event creation and management to be part of the system. * Real-time venue booking is highly useful and would reduce planning delays. * Notifications for deadlines and approvals help club leaders stay organized. * Role-based access (e.g., president, committee) is important for managing responsibilities. * AI-powered suggestions for event timing and budgeting would be a great addition. * Peer review system for rating past events is seen as helpful but not essential. * Budget alerts (when close to spending limits) are appreciated but not expected. * Membership renewal reminders are often overlooked and would be useful. * Current manual processes (email, forms) are time-consuming and frustrating. * Efficient, centralized access to club data and event tools is a major need. |
| --- |

# 

# **4. Verification**

**Requirements Verification**

To make sure the Software Requirements Specification (SRS) is accurate, comprehensive, and in line with stakeholder expectations, a rigorous requirements verification procedure will be conducted before the design phase begins. Software developers, QA engineers, administrative stakeholders, and student reps are all involved in this evaluation process. Clarity, testability, consistency, and traceability will all be assessed for both functional and non-functional requirements. Every requirement will have matching test cases and design components if a Requirement Traceability Matrix (RTM) is used. The project must be cleared of any ambiguous, contradictory, or unnecessary requirements found during this stage before moving on to design.

All comments and modifications brought about by requirement evaluations will be officially recorded in order to maintain accountability and openness. To record requirements-related issues, assign action items, and monitor their resolution, issue-tracking platforms like GitHub Issues, Trello, or JIRA might be utilized. Stakeholder input will be documented during meetings and included into the revised SRS via an iterative review procedure. A collaborative and traceable development workflow will be facilitated by version control of the requirements document, which will enable all stakeholders to view past changes. This methodical feedback loop guarantees that the validated requirements provide a solid basis for all subsequent development operations.

**Design Verification**

The design verification process's goal is to verify that the system architecture and component-level designs satisfy industry standards for modularity, scalability, and maintainability as well as the validated requirements. The product team, which consists of developers, software architects, and quality assurance staff, will carry out formal reviews. To guarantee logical coherence and efficient modular separation, design documentation including UML class diagrams, ER diagrams, and data flow diagrams will be examined throughout these evaluations. The focus will be on confirming that the system design accurately models important operations, including role-based access control, event registration, and budget approval.

Before proceeding with the technical implementation, stakeholders will validate design prototypes, such as low-fidelity wireframes and workflow mockups. The user interface and system interactions will be improved based on input gathered during these sessions. All review results, including requests for clarification or changes, will be recorded in shared documentation and followed through on. The design is kept in line with project limitations, technical viability, and stakeholder needs thanks to the iterative review process. In the end, the validated design needs to function as an understandable and practical development roadmap.

**Code Verification**

The code will undergo human code reviews and static code analysis throughout the development stage. To automatically find any problems like code smells, formatting errors, and security flaws, static analysis tools like Pylint, ESLint, or SonarQube will be used. Platforms such as GitHub will be used to conduct peer code reviews concurrently. Before code is merged into the main branch, developers must examine and approve each other's contributions. Code verification will guarantee that the codebase is maintainable, that the system architecture is followed, and that project coding standards are followed. Integration must come after all important issues have been resolved.

**Unit Testing**

To confirm the accuracy of each component separately, unit testing will be done. Using frameworks like Pytest or the integrated test runner in Django, developers will create automated test cases. Core features like club formation, event registration, budget submission, and login authentication will be the focus of these testing. Where required, mocking will be utilized to separate components. Before code is deemed stable for integration testing, all tests must pass and a minimum of 80% code coverage will be the goal.

**Integration Testing**

The proper operation of associated components will be confirmed by integration testing. For instance, it will confirm that students may sign up for events after registering and joining a club, and that these interactions result in the proper permissions and notifications. Tools like Selenium, Postman, or Pytest with Django's LiveServer will be used for testing. This guarantees that business rules are appropriately enforced across components, data flows between modules, and APIs operate as intended.

**Regression Testing**

Using testing frameworks compatible with Django, like Pytest or Django's TestCase, automated regression test suites will be created to ensure consistency and dependability between releases. These test suites will cover a variety of important use scenarios, such as workflows with different user roles (e.g., a committee member revising a request after it has been submitted by a student and approved by an administrator). As a defense against regressions brought about by code changes, the tests will run automatically prior to each deployment or merging into the main development branch.

The test library will be able to develop with the system since all regression test cases will be version-controlled using the same Git repository as the application code. When a test fails, developers will receive feedback right away, and they have to fix the problems before releasing the product. The regression suite will expand over time to incorporate new capabilities, guaranteeing continuous protection and coverage of the system's essential operations. In addition to lowering post-deployment problems and enhancing long-term maintainability, this proactive approach to regression testing will greatly increase system stability.

**System Testing**

In a controlled staging environment, system testing will verify that the complete system satisfies the validated functional and non-functional requirements. This covers all of the main user roles (students, club leaders, event administrators, and system administrators) carrying out standard tasks including budget requests, club approval, and event planning. Error handling, edge cases, and boundary conditions will all be extensively examined. The QA team will be in charge of carrying out thorough test cases and documenting any errors so they can be fixed.

**Non-Functional Testing**

Additionally, the system will be put through a rigorous non-functional testing process to assess its compatibility, security, performance, and usability. The system's ability to support 500 users at once while keeping response times under three seconds will be tested through performance testing. Security testing will guarantee that data is safely stored and communicated and validate protection against typical threats like SQL injection, CSRF, and XSS. In order to verify accessibility, logical navigation, and ease of use, usability testing will entail assessing the interface from the viewpoints of administrators and students. Testing for compatibility will guarantee that the system operates correctly on all supported platforms, including PCs, tablets, and smartphones, as well as on all supported browsers (Chrome, Firefox, Safari, and Edge).

**User Acceptance Testing (UAT)**

The system will undergo User Acceptance Testing (UAT) to confirm that it meets the expectations and real-world requirements of its target users. The testing method will involve a carefully chosen group of student users, club representatives, event coordinators, and administrative personnel. Through realistic tasks including club creation, budget proposal submission and approval, event scheduling, and announcement viewing, these users will engage with the system in a staging or UAT environment. Verifying that the system supports their daily operations, is user-friendly, and operates dependably under normal circumstances is the aim. In order to replicate independent usage, UAT scenarios will be prepared and testers will be led through the procedure with little help.

Following every UAT session, comprehensive input will be gathered via surveys, structured interviews, or methods for reporting issues. The QA team and product managers will examine this input to find any lingering bugs, functional gaps, or usability problems. Before the system is deployed in production, all important problems must be fixed. The majority of test participants must formally approve the system before it can proceed, and acceptance criteria will be established in cooperation with stakeholders. The last test to determine whether the system is prepared for live use in an actual university setting is this last stage of verification.

**Deployment Verification**

The system will go through smoke testing after it is deployed to the production environment to ensure that essential features like user login, event registration, club listing, and administrative dashboards work as intended. To guarantee system stability, the QA and DevOps teams will do this testing as soon as the system is deployed. Real-time tracking of system problems and uptime will also be done with monitoring tools. The platform's stability, usability, and readiness for the larger university community to access are the main objectives.

## 

# **5. Appendixes**

## **5.1 Assumptions and Dependencies**

Some assumptions are made for this system. The assumptions are as below:

1. **Authentication System Availability** The system depends on the availability of the university's central student database or authentication system (e.g., Single Sign-On) to validate users upon login.
2. **Email and Notification Services** It is assumed that reliable email and/or push notification services are available and properly configured to deliver announcements and system alerts to users.
3. **External System Integration**
   * The Financial Management System must be operational to handle budget approval workflows and generate financial reports.
   * The Campus Space Reservation System must be accessible for venue booking status and availability.
4. **Stable Internet Connection** A stable internet connection is required for users to access the system, especially for real-time features such as event calendars and venue booking.
5. **Role Assignment Accuracy** It is assumed that user roles (e.g., Student, Committee, President, Admin) are correctly assigned and maintained either manually or through integration with a university system.
6. **Browser Compatibility** The system is assumed to be used primarily on modern browsers (Chrome, Firefox, Edge) with JavaScript enabled.
7. **User Consent and Email Verification** Users are expected to verify their email addresses and consent to receive notifications upon registration.
8. **Data Backup and Recovery** It is assumed that server infrastructure includes routine data backup and recovery processes to prevent data loss.

## **5.2 Acronyms and Abbreviations**

| **Acronym / Abbreviation** | **Meaning** |
| --- | --- |
| **ID** | Identifier |
| **CRUD** | Create, Read, Update, Delete |
| **SSO** | Single Sign-On |
| **CSV** | Comma-Separated Values |
| **PDF** | Portable Document Format |
| **API** | Application Programming Interface |
| **HTML** | HyperText Markup Language |
| **CSS** | Cascading Style Sheets |
| **JS** | JavaScript |
| **FMS** | Financial Management System |
| **CSR** | Campus Space Reservation System |
| **OTP** | One-Time Password |
| **Auth** | Authentication |
| **Admin** | Administrator |
| **UI/UX** | User Interface / User Experience |
| **SMS** | Student Management System |
| **SIS** | Student Information System |
| **LMS** | Learning Management System |
| **SaaS** | Software as a Service |
| **SSO** | Single Sign-On |
| **API** | Application Programming Interface |
| **DBMS** | Database Management System |
| **HTTP** | HyperText Transfer Protocol |
| **HTTPS** | HyperText Transfer Protocol Secure |
| **FTP** | File Transfer Protocol |
| **IP** | Internet Protocol |
| **DNS** | Domain Name System |
| **GUI** | Graphical User Interface |
| **CLI** | Command Line Interface |
| **SDK** | Software Development Kit |
| **IDE** | Integrated Development Environment |
| **QA** | Quality Assurance |
| **UAT** | User Acceptance Testing |
| **ERP** | Enterprise Resource Planning |
| **CMS** | Content Management System |
| **KPI** | Key Performance Indicator |
| **ROI** | Return on Investment |
| **SLA** | Service Level Agreement |
| **T&C** | Terms and Conditions |

*Table 54: Acronyms and Abbreviations Used*