Software Requirements Specification

for

Student Club Management System with Budget and Venue Integration

Version <1.0>

Table of contents

Contents

1 Introduction

1.1 Purpose

The purpose of the Student Club Management System with Budget and Venue Integration is to streamline and automate the management of student clubs within a university setting. This system aims to centralize key club activities such as event planning, budget tracking, and venue booking into a single, user-friendly platform.

The system will serve as a bridge between student organizations, faculty coordinators, and administrative staff, ensuring smooth communication and accountability. Ultimately, it is designed to enhance organizational efficiency, promote student engagement, and support the growth of extracurricular activities on campus.

1.2 Scope

The proposed software is a web-based Student Club Management System designed for use within a university environment. The goal of the system is to provide a centralized platform for managing student clubs, memberships, budgets, and venue reservations efficiently. The application is to be developed over a 3-month period by a team of 4 developers.

The system will support the following scope:

- The platform must support a minimum of 30 clubs.
- The system must be scalable to accommodate at least 10,000 students.
- The system will involve 3 internal user roles:
    - System Administrator – managing the overall system, users, and configurations.
    - Club Administrator – manages specific club activities, members, events, voting, and requests.
    - Student – can view/join clubs, view/join events and voting

Also, the system will integrate with two external systems:

- University Financial Management System – for tracking fund transparency, submitting budget, approving, and tracking club budgets.
- Campus Space Reservation Database – for checking availability and booking university venues for club activities.

Key features of the system will include:

- Role-Based Access Control with clearly defined permissions, UI access, and features based on user roles.
- Real-time integration with external systems to reflect the latest data for financials and venue availability.
- Responsive and accessible web interface for users with a stable internet connection.

1.3 Product Overview

The Student Club Management System is a web-based application designed for management of student clubs within a university. The system is integrated with the university's University Financial Management System and Campus Space Reservation Database. Allow club admin and members to manage club's detail, checking financial flow and booking of venue.

1.3.1 Product Perspective

The Student Club Management System is a web-based application designed for the management of student clubs within the university. The platform is for handling club's membership, event planning, budget tracking, and venue booking for a minimum of 30 clubs and 10,000 students of the traffic. The system interacts with external systems and users to streamline club operations while maintaining clear boundaries. It focuses on club-related activities.

The system interfaces with two external systems which are the University Financial Management System and the Campus Space Reservation Database. The Financial Management System enables clubs to submit budgets, receive approvals, and track funds, ensuring transparency and compliance with university financial policies. The Campus Space Reservation Database allows club administrators to view real-time venue availability and submit booking requests to university for club's events, simplifying the event scheduling. These interactions assume a stable internet connection and API connection with the corresponding external system to provide real-time updates to users.

The system supports three internal actors which are System Admin, Club Admin, and Student. These actors are accessing the platform via a web interface with role-based access control to perform their respective features. System Admins can manage the platform's configuration and user accounts, Club Admins manage club-specific operations like membership, budgets, events, and event voting, Students engage with

clubs through membership, event voting and event participation. External actors include the Financial Management System and Campus Space Reservation Database, which connecting data with the system to support its functionalities.

A context diagram, provided below, illustrates the system's interactions. The diagram depicts the Student Club Management System as the central entity, with bidirectional data flows to the Financial Management System (fund tracing, budget submissions and approvals), the Campus Space Reservation Database (venue availability, venue booking requests and confirmations), and the three user actors (login, data input, and output). This visual representation clarifies the system's scope and external dependencies.

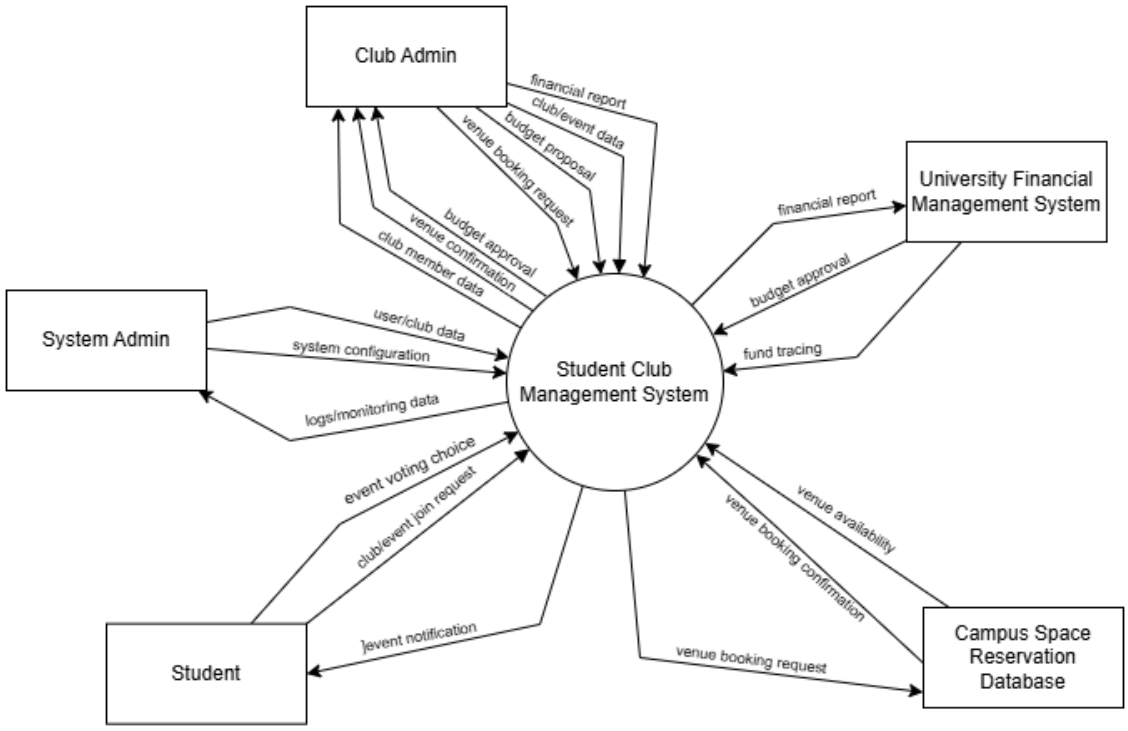

Figure 1.0 Context Diagram - Student Club Management with Budget & Venue Integration

1.3.2 Product Function

The Student Club Management System provides the following primary functions:

- **Membership Management**: Allow adding, viewing, and removing members and managing membership status.
- **Event Management**: Allow creating, proposing, announcing, voting, and joining events.
- **Venue Booking**: Allow viewing available venue, booking venue for club's events through university's API.
- **Financial Management:** Allow tracing fund, requesting budget, allocate club budget and view financial report.
- **System Management**: Allow system admin to manage user accounts, monitor system data, create, edit, and remove clubs.

1.3.3 User Characteristics

The Student Club Management System is designed for:

- **System Admin**: University staff with advanced technical skills, know about the system and computer state, responsible for system configuration and user management, monitor the system daily and provide help to users with troubles.
- **Club Admin**: Student or lecturer who has basic computer skills, manage club operations, manage members, manage events and manage event voting.
- **Student**: University student with minimal computer skills, joining clubs and events, using the system with stable internet access, interesting in explorer clubs.

1.3.4 Limitations

The Student Club Management System has the following limitations:

- User must be university staff/student, also comply with university policies
- User must have minimum with a device that can access web browser (computer / smart phone)
- Some function like booking venue, financial tracking relies on University Financial Management System and Campus Space Reservation Database, which may impose integration constraints.
- Booking venue and budget request are limited by sequential process as its depend on the API of University Financial Management System and Campus Space Reservation Database.

- Basic logging on user login, perform action and external call to API.
- Lack of direct control on financial approval and venue booking confirmation because its depend on the external system.
- Constrained to web-compatible languages like JavaScript, HTML and CSS, limiting the backend complexity.
- Support around 30 clubs and 10000 students to use
- Vulnerable to external system and misuse of admin role.
- Limited to university staff to become an admin role to avoid leak to student data to other party
- Require user to have minimal physical effort and assume user can handle web navigation.
- Subject to real-time update delay from University Financial Management System and Campus Space Reservation Database, affecting incorrect data/outdate data
- Support only for English language UI
- Lack of automated backup systems, require manual backup
- Require online internet connectivity, no offline functionality provided

1.4 Definitions

Term	Definition
Student Club Management System	A web-based application to manage student's club
Web-based Application	An application running on browser like google chrome, Firefox or Microsoft edge
Student	University students that use the program for joining clubs
Club Admin	University student/lecturer with basic computer skills to manage the club operations
System Admin	University staff with advance computer skill for monitor/configure the system, manage user account and manage clubs.
Member Status	The status of a student on specific club (active, pending, quit, suspended)
Role-Based Access Control	A security mechanism to restrict user access the unauthorized page/features.

Event Proposal	An event suggestion for club member to approve, joining and voting.
Venue Booking	The proses to reserve a venue from university campus for event purpose.
University Financial Management System	An external system which process the financial related request to university.
Campus Space Reservation Database	An external database providing university venue availability and booking service.

2 References

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

Requirements

3 Requirements

3.1 Functions

The **Use Case Diagram** for the Student Club Management System illustrates the interactions between **actors** and the **core functionalities** of the system. It provides a visual overview of how different roles interact with the system to achieve specific goals.

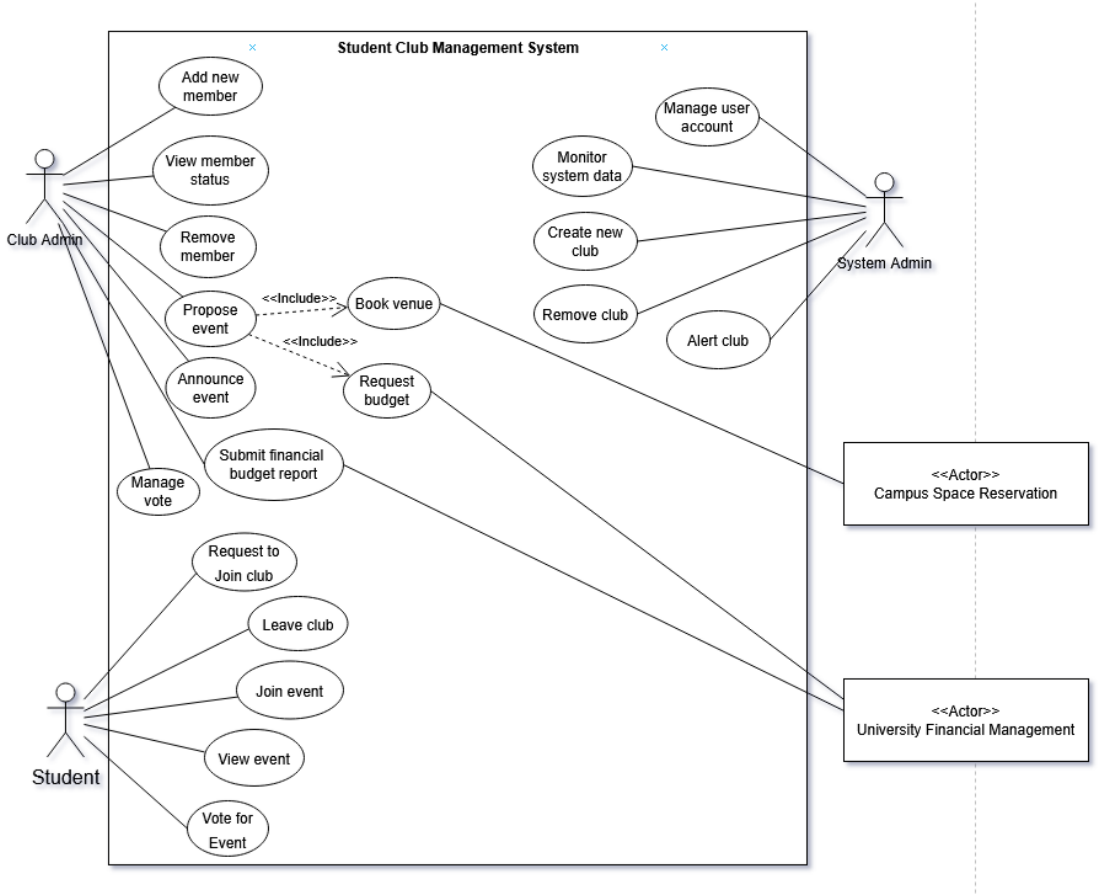

Figure 2.0 Use Case for Student Club Management with Budget & Venue Integration

1) Student

F001 Request to Join a club

Use Case ID	UC001	Version	1.0
Purpose		To allow students to send a request to join a club.	
Feature		F001 Request to Join a club	
Actor		Student	
Trigger		Student selects "Browse Clubs" and chooses a club	
Pre-Conditions		• - Student is logged in. • - Club exists and is accepting members. • - Student is not already a member.	
Scenario Name		Step	Action
Main Flow		1	Student selects Browse Clubs to view available clubs.
		2	Student Selects Club from the list.
		3	System prompts Confirm Send Request to Join Club (options: No/Yes).
		4	Student confirms (Yes). System sends Approval Request to Club Admin.
Alternate Flow – Confirmation (No)		3.1	Student selects No. System returns to club list.
Rules		- Membership requests require admin approval.	
Author		Mohanad Hassan	

Table 1.0 Use Case Specification – Request to Join a Club

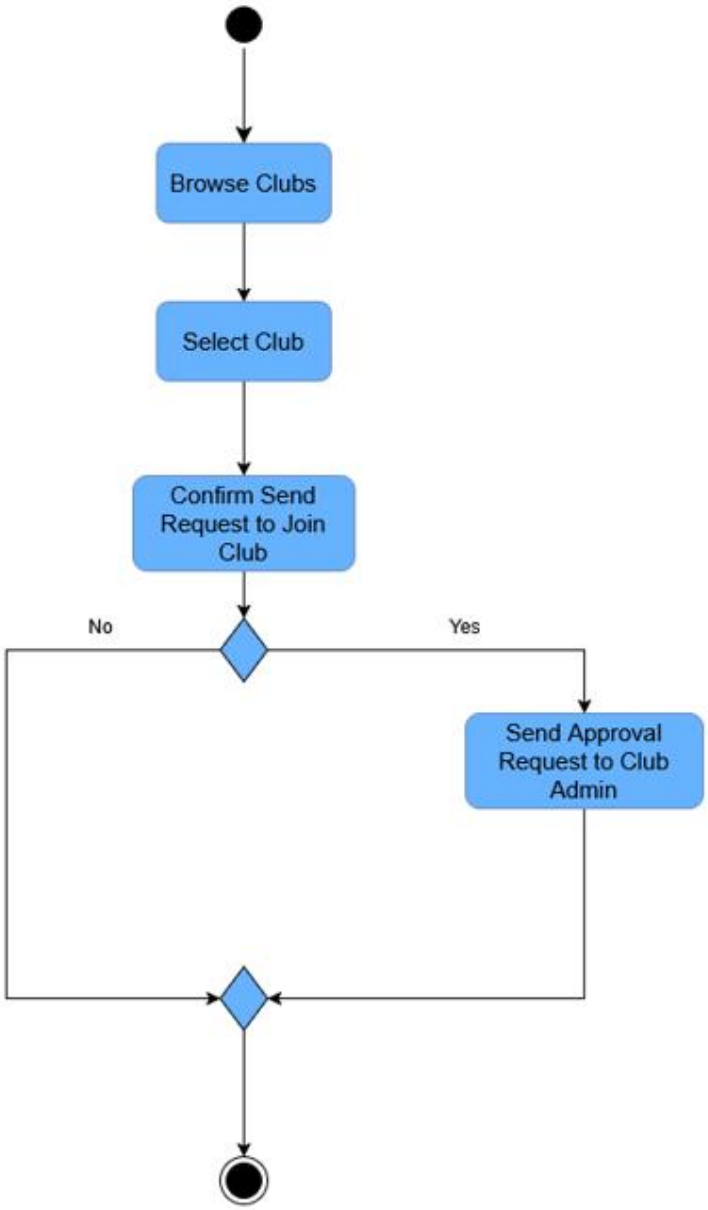

Figure 3.0 Activity Diagram – Request to Join a Club

F002 Leave Club

Use Case ID	UC002	Version	1.0
Purpose		Student shall be able to leave club that he is part of.	
Feature		F002 Leave Club	
Actor		Student	
Trigger		Student selects "View Student's Clubs" and chooses a club to leave	
Pre-Conditions		• The student is logged into the system. • The student is an active member of the selected club.	
Scenario Name		Step	Action
Main Flow		1	Student selects View Student's Clubs to see their current clubs.
		2	Student Selects Club they wish to leave.
		3	System prompts Confirm Leave Club (options: No/Yes).
		4	Upon confirmation (Yes): a) System Removes Student from Club Members. b) Automatically Sends Notification to Club Admin.
Alternate Flow – Confirmation (No)		3.1	Student selects No. System returns to club list.
Rules		- Immediate removal upon confirmation. - Notifications are mandatory.	
Author		Mohanad Hassan	

Table 2.0 Use Case Specification – Leave Club

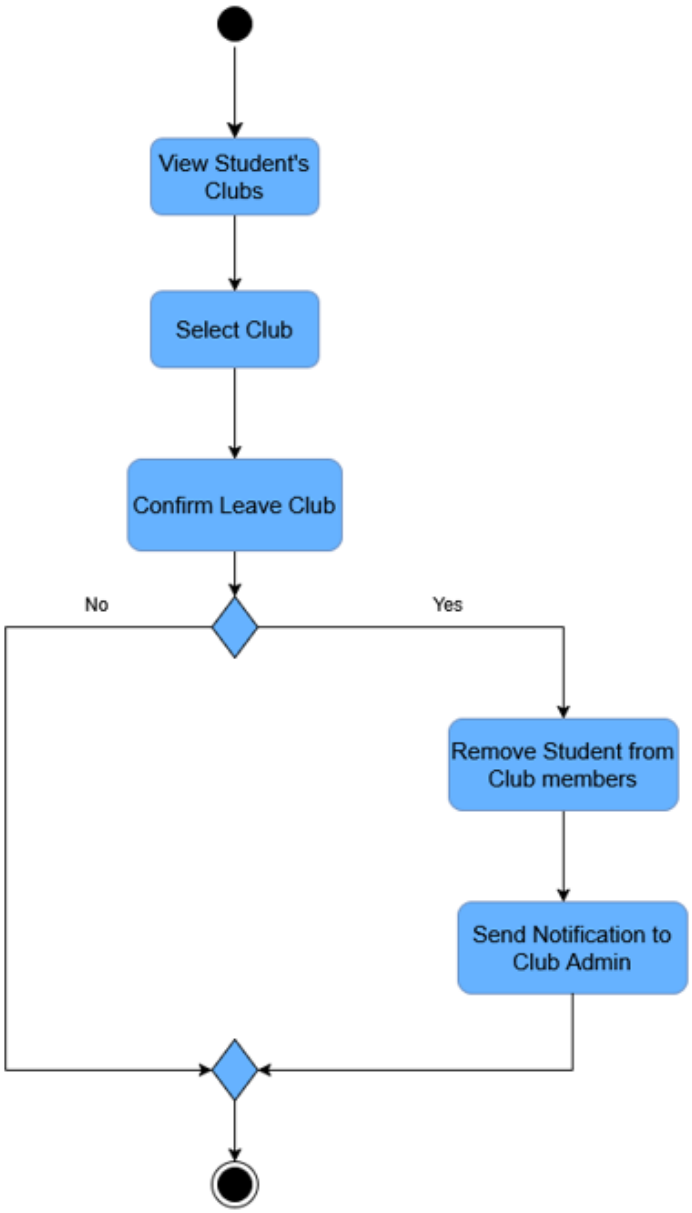

Figure 4.0 Activity Diagram – Leave Club

F003 Register for Event

Use Case ID	UC003	Version	1.0
Purpose		To enable students to register for club events.	
Feature		F003 Register for Event	
Actor		Student	
Trigger		Student selects an event from the "View Events" list.	
Pre-Conditions		• - Student is logged in. • - Event is open for registration. • - Venue data is accessible.	
Scenario Name		Step	Action
Main Flow		1	Student selects View Events.
		2	Student Selects Event to register.
		3	System checks Event Capacity (via venue integration).
		4	If Not Full, system registers student.
Alternate Flow – Full Capacity		3.1	If it is full, System displays a message saying that capacity is full.
Rules		- Capacity checks are real-time. - Eligibility criteria	
Author		Mohanad Hassan	

Table 3.0 Use Case Specification - Register for Event

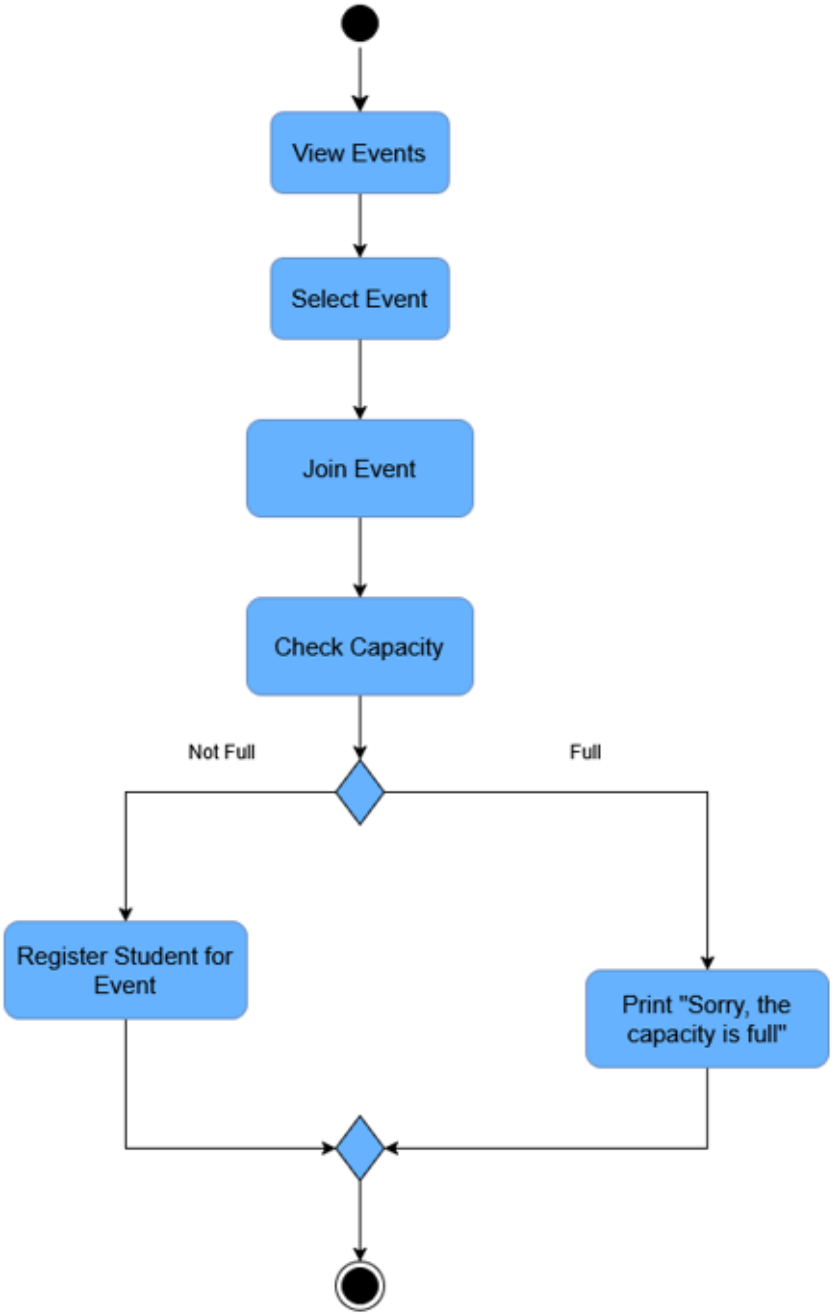

Figure 5.0 Activity Diagram – Register for Event

F004 View Event Details

Use Case ID	UC004	Version	1.0
Purpose		To display detailed information about an event.	
Feature		F004 View Event Details	
Actor		Student	
Trigger		Student selects an event from the "View Events" list.	
Pre-Conditions		• - Student is logged in. • - Event exists in the system.	
Scenario Name		Step	Action
Main Flow		1	Student selects View Events.
		2	Student Selects Event from the list.
		3	System Fetches Event Details
Rules		- Event details must be up-to-date. - data is read-only.	
Author		Mohanad Hassan	

Table 4.0 Use Case Specification- View Event Details

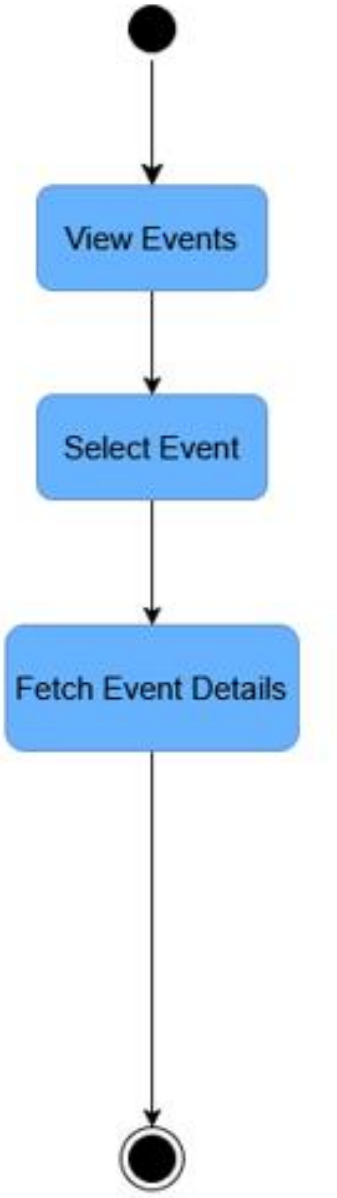

Figure 6.0 Activity Diagram – View Event Details

F005 Submit a Vote

Use Case ID	UC005	Version	1.0
Purpose		To allow club members to vote in polls.	
Feature		F005 Submit a Vote	
Actor		Student	
Trigger		Student selects a vote from the "Vote List.".	
Pre-Conditions		• - Student is logged in. • - Student is a club member. • - Vote is active.	
Scenario Name		Step	Action
Main Flow		1	Student opens Vote List.
		2	Student Selects Vote.
		3	Student Chooses Option and confirms
		4	If confirmation is yes, System submits vote and records it.
Alternate Flow – Confirmation is No		3.1	If confirmation is No, Student returns to vote selection.
Rules		- One vote per student per poll.	
Author		Mohanad Hassan	

Table 5.0 Use Case Specification – Submit a Vote

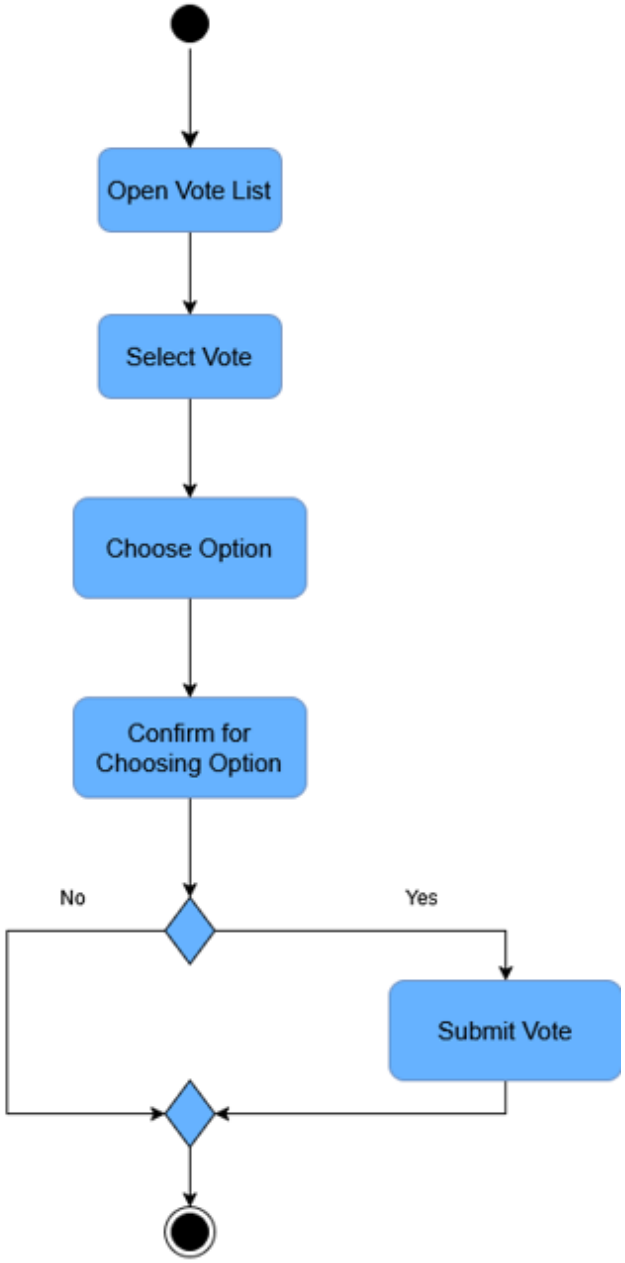

Figure 7.0 Activity Diagram - Submit a Vote

2) Club Admin

F006 Submit Financial Budget Report

Use Case ID	UC006	Version	1.0
Purpose		To enable club admins to submit financial budget reports.	
Feature		F006 Submit Financial Budget Report	
Actor		Club Admin	
Trigger		Admin selects "Open Financial Budget Report".	
Pre-Conditions		• - Admin is logged in. • - Budget module is accessible. • - Report period is open.	
Scenario Name		Step	Action
Main Flow		1	Admin selects Open Financial Budget Report.
		2	Admin Fills Report with required data.
		3	Admin Submits Report to Financial Management
		4	System Checks Submission Success.
		5	If submission is success, System displays "Report Submitted".
Alternate Flow – Submission Failed		4.1	If submission failed, System displays "Submission Failed" and retries.
Rules		- Reports must align with allocated budgets. - Submissions are timestamped.	
Author		Mohanad Hassan	

Table 6.0 Use Case Specification – Submit Financial Budget Report

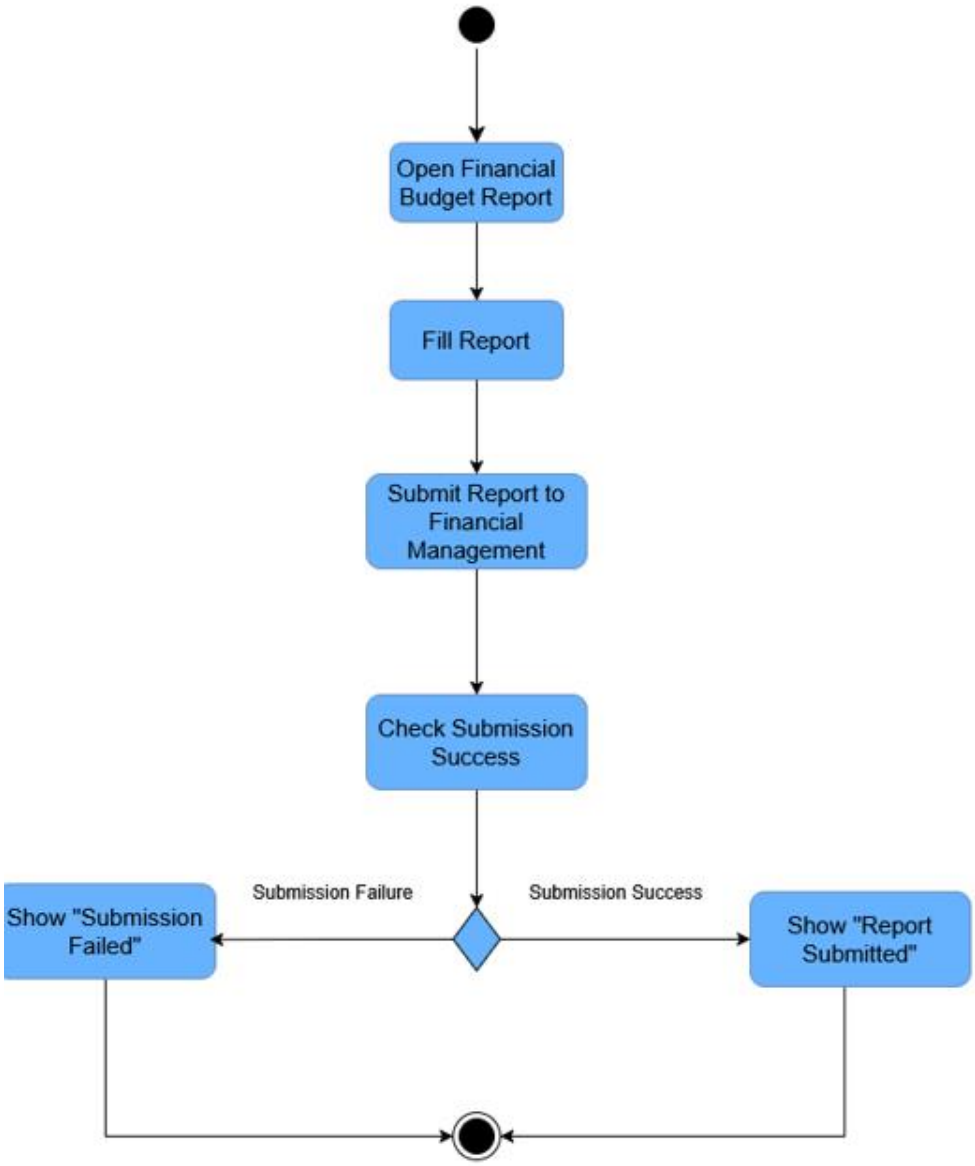

Figure 8.0 Activity Diagram – Submit Financial Budget Report

F007 Propose Event

Use Case ID	UC007	Version	1.0
Purpose		To allow club admins to propose and plan events.	
Feature		F007 Propose Event	
Actor		Club Admin	
Trigger		Admin selects "Propose Event".	
Pre-Conditions		• - Admin is logged in. • - Venue booking module is available.	
Scenario Name		Step	Action
Main Flow		1	Admin selects Propose Event.
		2	Admin Enters Event Details
		3	Admin Books Venue (integration with venue system).
		4	Admin Submits Proposal
		5	System Requests Budget Approval from Financial Management.
		6	Admin Sends Notification to Club Members.
Alternate Flow – Venue Unavailable		3.1	If Selected venue is booked, display message that says selected venue is already booked
Alternate Flow – Request is rejected		5.1	If request is rejected, display message that says request is rejected
Rules		- Budget requests must not exceed club limits. - Venue booking requires confirmation.	
Author		Mohanad Hassan	

Table 7.0 Use Case Specification – Propose Event

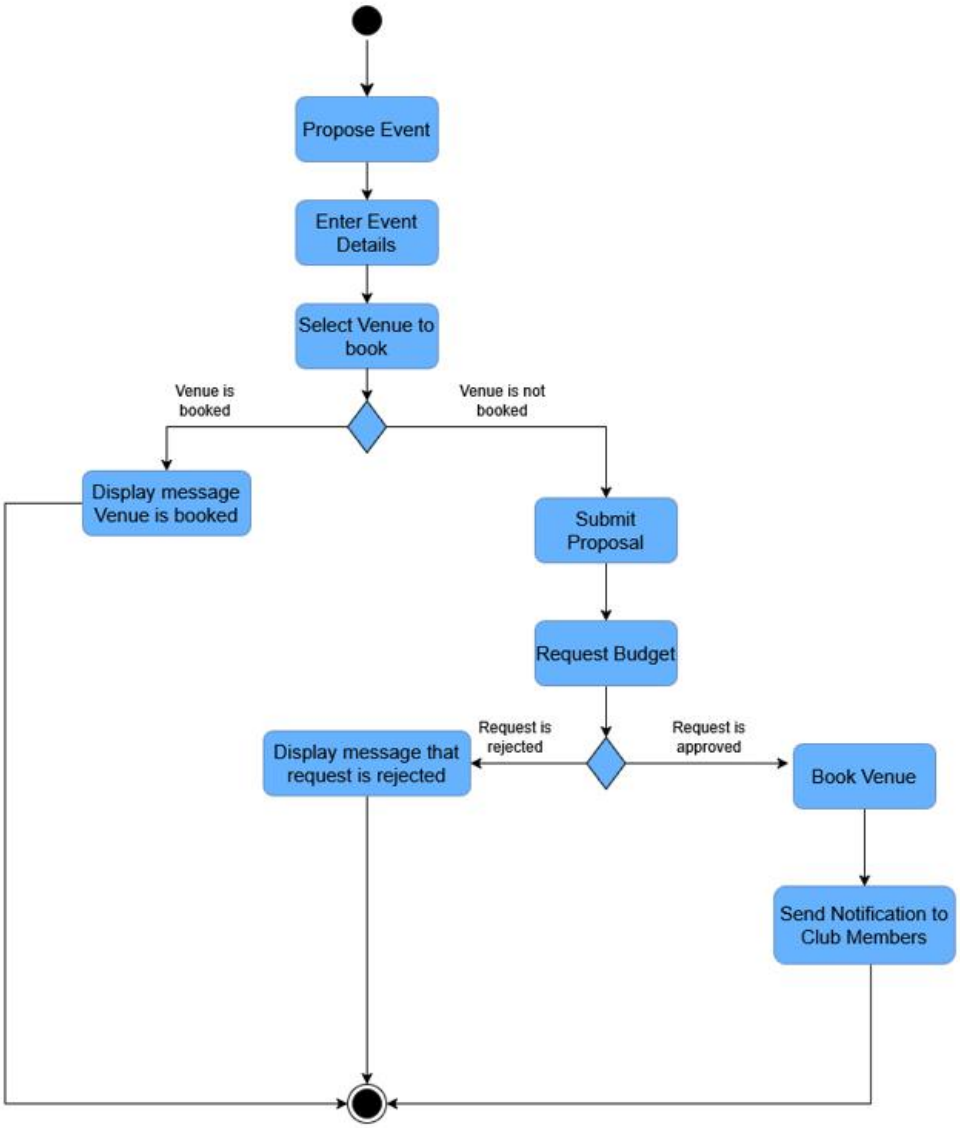

Figure 9.0 Activity Diagram – Propose Event

F008 Announce Event

Use Case ID	UC008	Version	1.0
Purpose		To notify club members about upcoming events.	
Feature		F008 Announce Event	
Actor		Club Admin	
Trigger		Admin selects "Announce Event".	
Pre-Conditions		• - Admin is logged in. • - Event is approved and scheduled.	
Scenario Name		Step	Action
Main Flow		1	Admin selects Announce Event.
		2	Admin Enters Announcement Details
		3	Admin Sends Notification to Club Members
Rules		- Announcements require event approval.	
Author		Mohanad Hassan	

Table 8.0 Use Case Specification – Announce Event

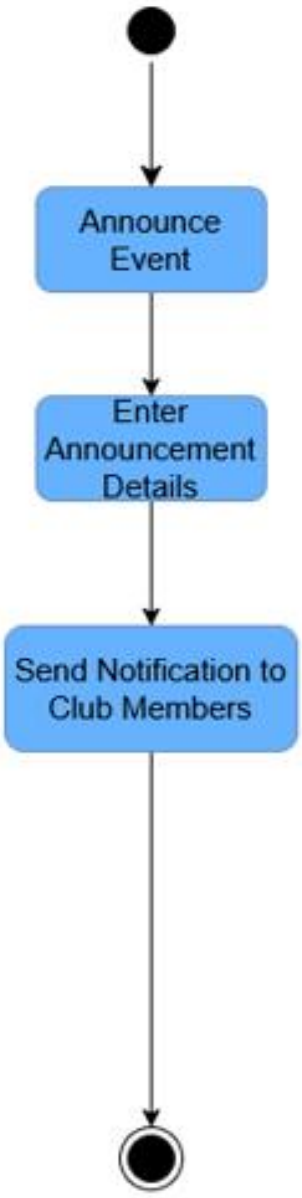

Figure 10.0 Activity Diagram – Announce Event

F009 Manage Votes

Use Case ID	UC009	Version	1.0
Purpose		To allow admins to create or remove polls	
Feature		F009 Manage Votes	
Actor		Club Admin	
Trigger		Admin selects "Open Vote List"	
Pre-Conditions		• - Admin is logged in. • - Voting module is accessible.	
Scenario Name		Step	Action
Main Flow		1	Admin opens Vote List.
		2	Admin selects Make New Vote
		3	Admin Enters Details/Options
		4	System Notifies Club Members about the new vote
Alternate Flow – Delete Vote		1.1	Admin Removes the Vote
Rules		- Admins can delete votes before they go live.	
Author		Mohanad Hassan	

Table 9.0 Use Case Specification – Manage Votes

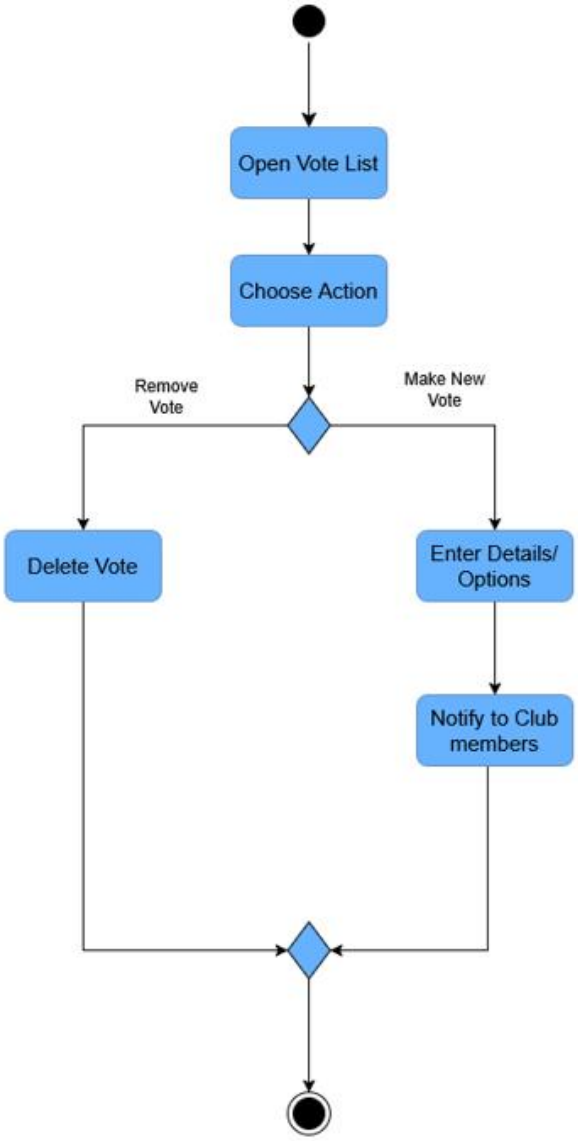

Figure 11.0 Activity Diagram – Manage Votes

F010 Add Member to Club

Use Case ID	UC010	Version	1.0
Purpose		To allow admins to manually add students to a club.	
Feature		F010 Add Member to Club	
Actor		Club Admin	
Trigger		Admin selects "View Member List" > "Add New Member".	
Pre-Conditions		• - Admin is logged in. • - Student is registered in the system.	
Scenario Name		Step	Action
Main Flow		1	Admin selects View Member List.
		2	Admin selects Add New Member and Searches for Students.
		3	Admin Selects Student from results.
		4	System prompts confirmation (No/Yes).
		5	Admin confirms (Yes). System Adds Student to Club and Sends Notification.
Alternate Flow – Confirmation is No		4.1	If confirmation is no, System returns to Member list.
Rules		- Students must accept membership if approval is required	
Author		Mohanad Hassan	

Table 10.0 Use Case Specification – Add Member to Club

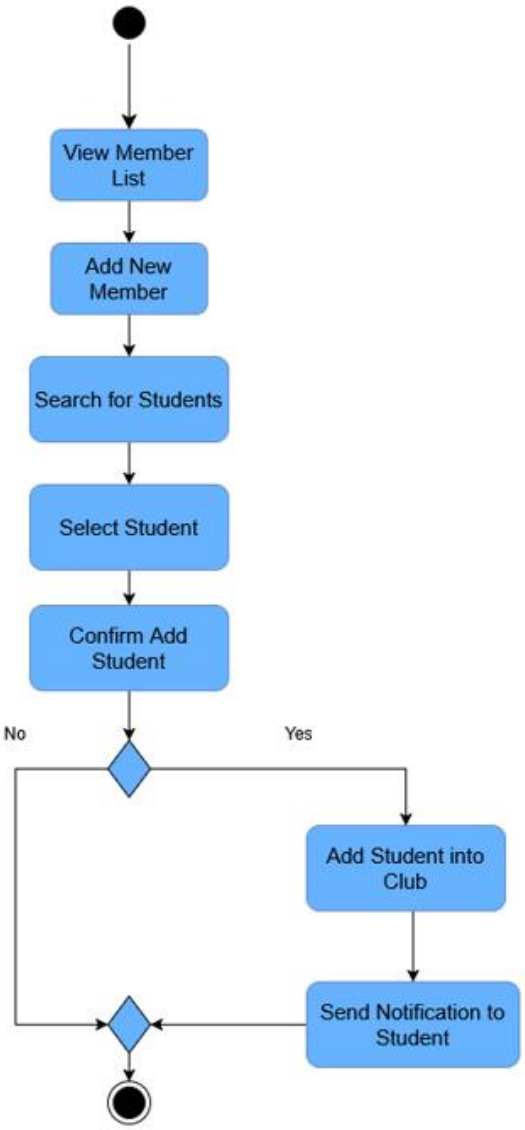

Figure 12.0 Activity Diagram – Add Member to Club

F011 Fetch Member Status

Use Case ID	UC011	Version	1.0
Purpose		To view detailed status of a club member.	
Feature		F011 Fetch Member Status	
Actor		Club Admin	
Trigger		Admin selects a member from "View Member List".	
Pre-Conditions		• - Admin is logged in. • - Member exists in the club.	
Scenario Name		Step	Action
Main Flow		1	Admin selects View Member List.
		2	Admin Selects Member from the list.
		3	System Fetches Member Status
Rules		- Status data is read-only	
Author		Mohanad Hassan	

Table 11.0 Use Case Specification – Fetch Member Status

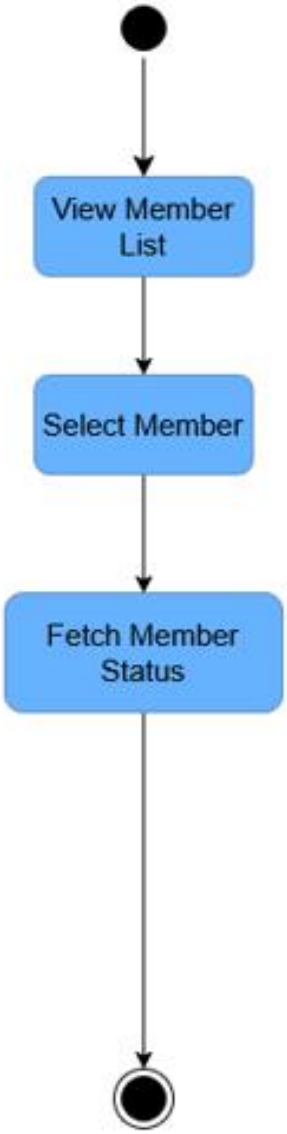

Figure 13.0 Activity Diagram – Fetch Member Status

F012 Remove Member from Club

Use Case ID	UC012	Version	1.0
Purpose		To allow admins to remove members from a club.	
Feature		F012 Remove Member from Club	
Actor		Club Admin	
Trigger		Admin selects "View Member List" > "Remove Member".	
Pre-Conditions		• - Admin is logged in. • - Member is part of the club.	
Scenario Name		Step	Action
Main Flow		1	Admin selects View Member List.
		2	Admin Selects Member and chooses Remove Member.
		3	System prompts confirmation (No/Yes).
		4	Admin confirms (Yes). System removes member and Sends Notification
Alternate Flow – Confirmation is No		3.1	Admin selects No. System returns to member list.
Rules		- Removed members lose access to club resources immediately.	
Author		Mohanad Hassan	

Table 12.0 Use Case Specification - Remove Member from Club

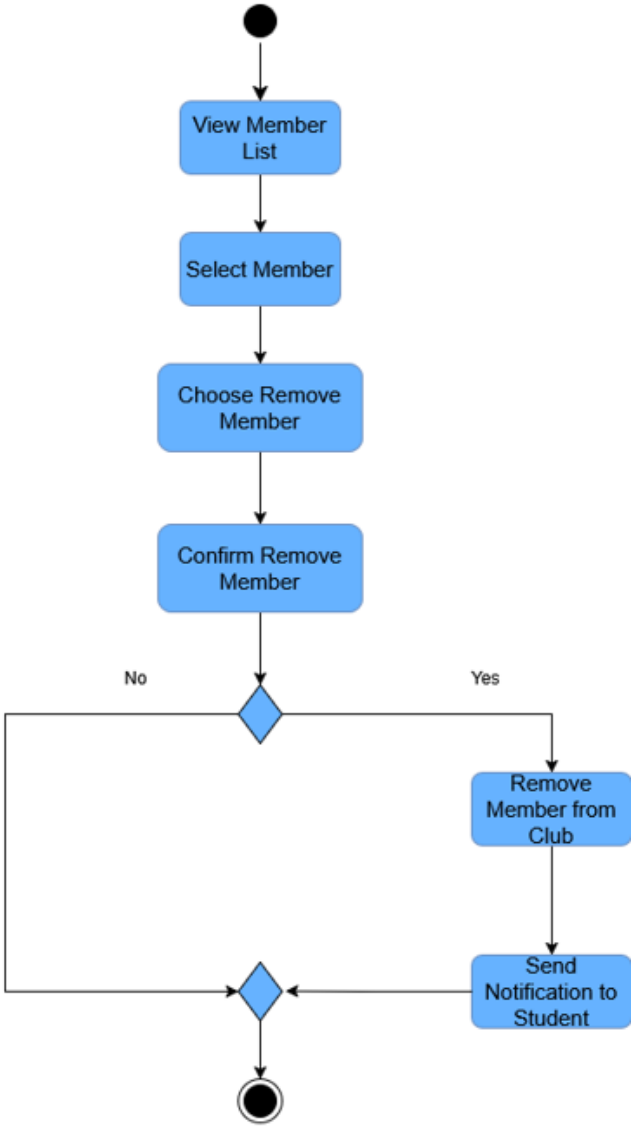

Figure 14.0 Activity Diagram – Remove Member from Club

3) System Admin

F013 Trigger Club Alert

Use Case ID	UC013	Version	1.0
Purpose		To allow System Admins to send urgent alerts to members of a specific club.	
Feature		F013 Trigger Club Alert	
Actor		System Admin	
Trigger		Admin selects "Trigger Alert" > "Select Club to Alert".	
Pre-Conditions		• - Admin is logged in with elevated privileges. • - Club exists in the system.	
Scenario Name		Step	Action
Main Flow		1	Admin selects Trigger Alert and chooses a club.
		2	Admin Enters Alert Details
		3	Admin Sends Alert to all club members.
Rules		- Alerts are logged for auditing	
Author		Mohanad Hassan	

Table 13.0 Use Case Specification – Trigger Club Alert

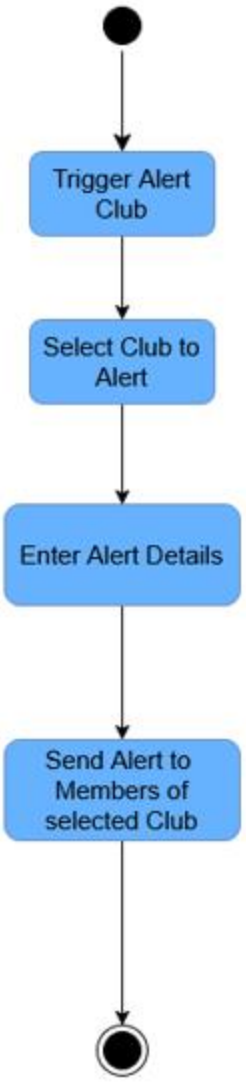

Figure 15 Activity Diagram – Trigger Alert Club

F014 Monitor System Metrics

Use Case ID	UC014	Version	1.0
Purpose		To enable System Admins to view real-time system performance data.	
Feature		F014 Monitor System Metrics	
Actor		System Admin	
Trigger		Admin selects "Open Monitoring".	
Pre-Conditions		• - Admin is logged in. • - Monitoring module is active.	
Scenario Name		Step	Action
Main Flow		1	Admin selects Open Monitoring.
		2	System Fetches Metrics/Logs
		3	System Displays Charts/Tables for visualization
Rules		- Metrics are read-only and refresh every second	
Author		Mohanad Hassan	

Table 14.0 Use Case Specification – Monitor System Metrics

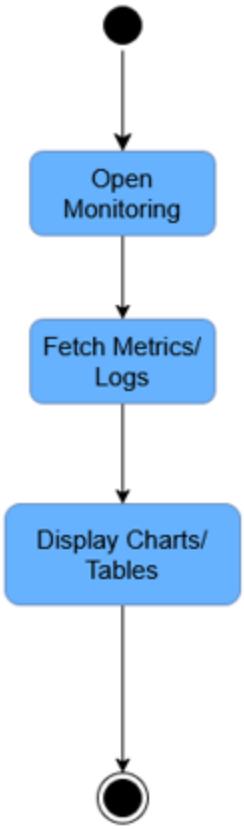

Figure 16.0 Activity Diagram – Monitor System Metrics

F015 Create New Club

Use Case ID	UC015	Version	1.0
Purpose		To allow System Admins to create new clubs in the system.	
Feature		F015 Create New Club	
Actor		System Admin	
Trigger		Admin selects "Open Create Club UI".	
Pre-Conditions		• - Admin is logged in. • - Club name is unique.	
Scenario Name		Step	Action
Main Flow		1	Admin selects Open Create Club UI.
		2	Admin Enters Details
		3	Admin Submits the form
		4	System creates the club and confirms "Club Created."
Rules		- Metrics are read-only and refresh every second	
Author		Mohanad Hassan	

Table 15.0 Use Case Specification – Create New Club

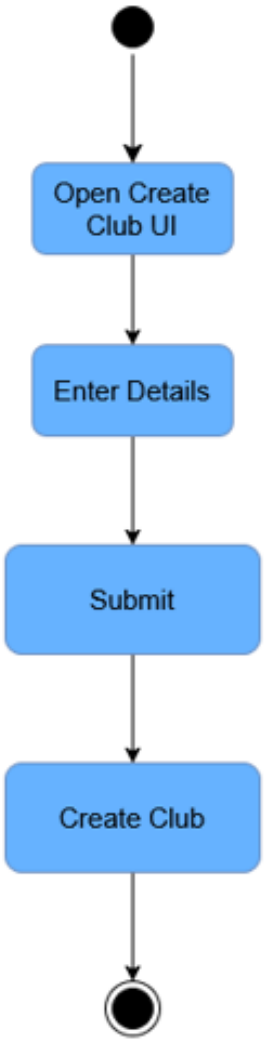

Figure 17.0 Activity Diagram – Create New Club

F016 Manage User Accounts

Use Case ID	UC016	Version	1.0
Purpose		To enable System Admins to create, edit, or delete user accounts	
Feature		F016 Manage User Accounts	
Actor		Club Admin	
Trigger		Admin selects "Open User Management".	
Pre-Conditions		• - Admin is logged in with full privileges.	
Scenario Name		Step	Action
Main Flow		1	Admin selects Open User Management
		2	Admin Searches User or selects Create New User.
Sub Flow – Create New User		2.1	Admin selects Create New User > Insert Details
		2.2	System validates and creates the account
Sub Flow – Edit User		2.1	Admin selects user > Edit User's Information
		2.2	Admin Modifies User Information
		2.3	System updates and confirms "User Updated."
Sub Flow – Remove User		2.1	Admin selects user > Delete User Record
		2.2	System removes user
Rules		- User deletions are irreversible - Role changes require re-authentication.	
Author		Mohanad Hassan	

Table 16.0 Use Case Specification – Manage User Accounts

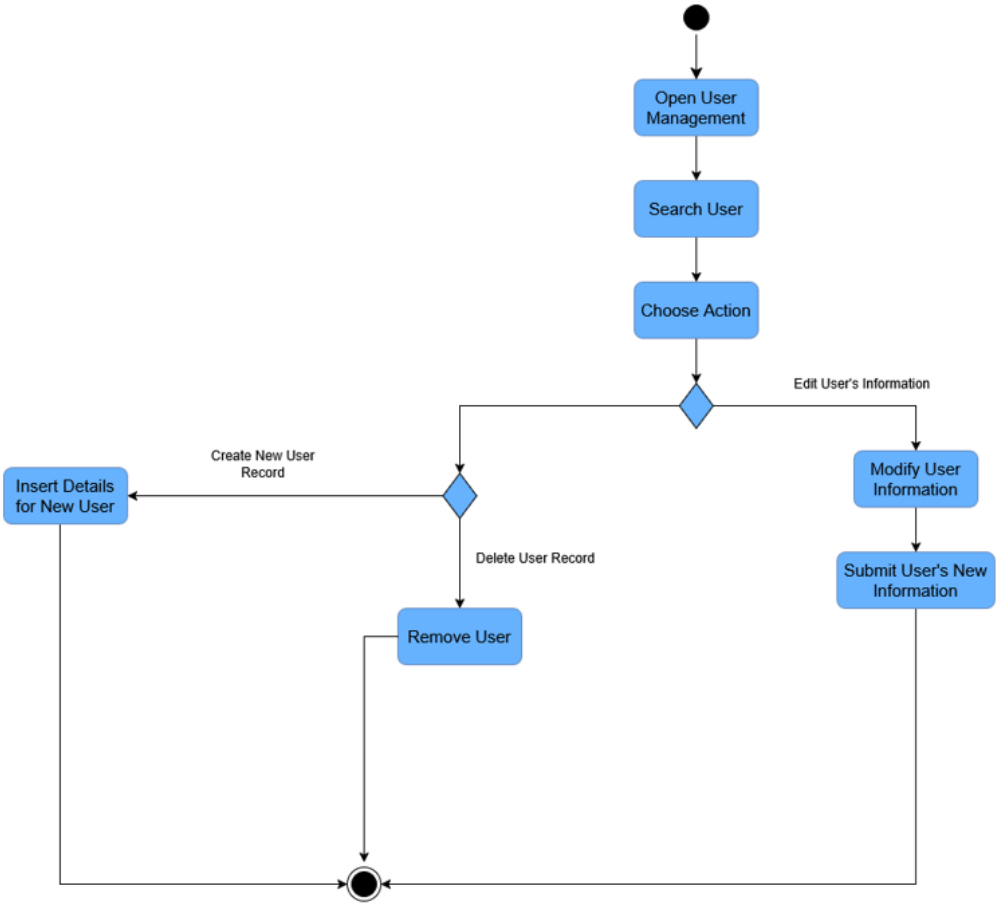

Figure 18.0 Activity Diagram – Manage User Accounts

F017 Delete Club

Use Case ID	UC017	Version	1.0
Purpose		To allow System Admins to permanently remove a club from the system.	
Feature		F017 Delete Club	
Actor		System Admin	
Trigger		Admin selects "View Club List" > "Delete Club".	
Pre-Conditions		• - Admin is logged in. • - Club must exist in the system.	
Scenario Name		Step	Action
Main Flow		1	Admin selects View Club List
		2	Admin Selects Club and chooses Delete Option
		3	System prompts confirmation (No/Yes).
		4	Admin confirms (Yes). System Removes Club and all associated data
Alternative Flow – Confirmation is No		3.1	Admin confirms (No). System returns back to view club list.
Rules		- Club deletions are irreversible	
Author		Mohanad Hassan	

Table 17.0 Use Case Specification - Delete Club

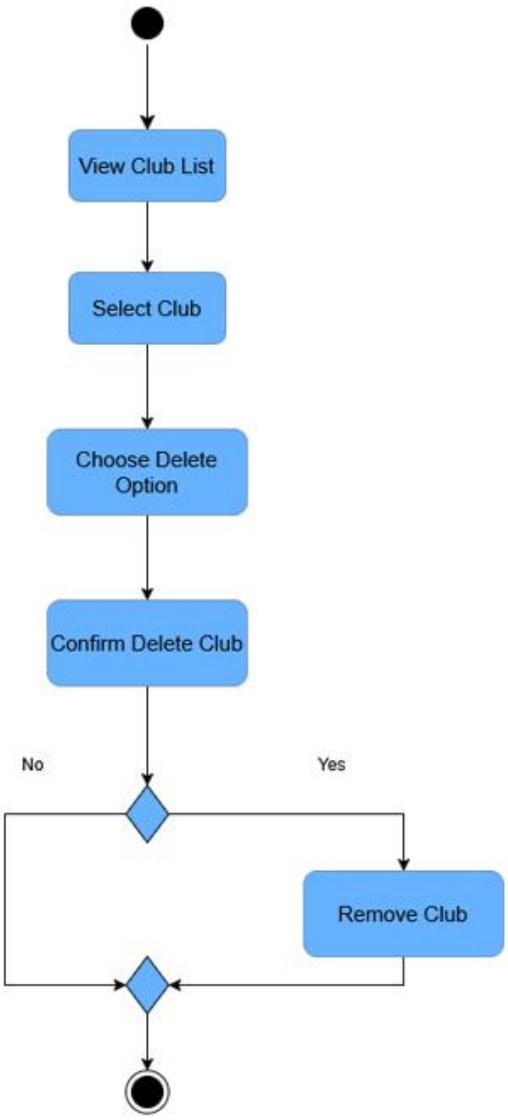

Figure 19.0 Activity Diagram – Delete Club

4) Financial Management System

F018 Integrate with University Financial Management

Use Case ID	UC018	Version	1.0
Purpose		To validate and approve budget requests via the university's financial system	
Feature		F018 Integrate with University Financial Management	
Actor		University Financial Management System	
Trigger		Club Admin submits a budget request from the Financial Dashboard	
Pre-Conditions		• - Financial system API is accessible. • - Budget request is pending approval.	
Scenario Name		Step	Action
Main Flow		1	Admin opens Financial Dashboard
		2	Admin selects View Pending Requests and chooses a request.
		3	System Fetches Request Details
		4	Admin Enters Budget Amount and triggers validation
		5	Financial system Validates Amounts against university policies
		6	If Valid, system allows Submit Approval
Alternate Flow – Amount is invalid		6.1	If Invalid, system will display error message that says amount exceeds limits.
Rules		- Budgets must comply with university fiscal policies. - - Approvals are logged in both systems.	
Author		Mohanad Hassan	

Table 18.0 Use Case Specification – Integrate with University Financial Management

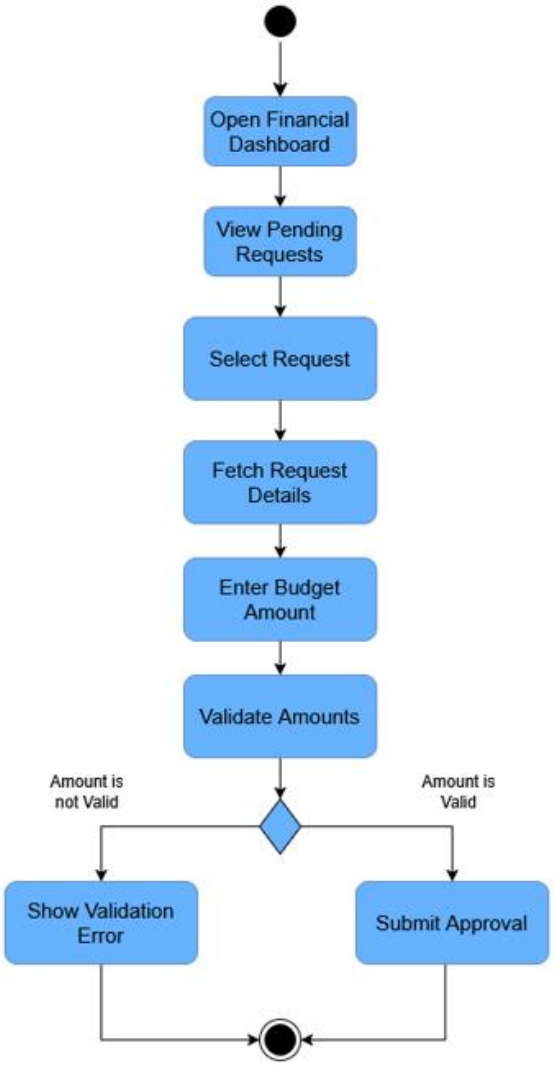

Figure 20.0 Activity Diagram – Integrate with University Financial Management

5) Campus Space Reservation

F019 Campus Space Reservation Integration

Use Case ID	UC019	Version	1.0
Purpose		To manage venue bookings via the campus reservation system	
Feature		F019 Book Venue	
Actor		Campus Space Reservation System	
Trigger		Club Admin initiates a venue booking for an event.	
Pre-Conditions		• - Campus API is operational. • - Venue availability data is synced.	
Scenario Name		Step	Action
Main Flow		1	Admin Accesses Campus API from the system
		2	Admin Views Pending Bookings for the club
		3	Admin Selects Booking and chooses Approve/Reject
		4	If Approved, System will confirm booking
		5	System will notify club admin that booking is approved
Alternate Flow – Rejection		3.1	If Rejected, system will reject booking.
		3.2	System will notify club admin that booking is rejected
Rules		- Approving or Rejecting is irreversible	
Author		Mohanad Hassan	

Table 19.0 Use Case Specification – Book Venue

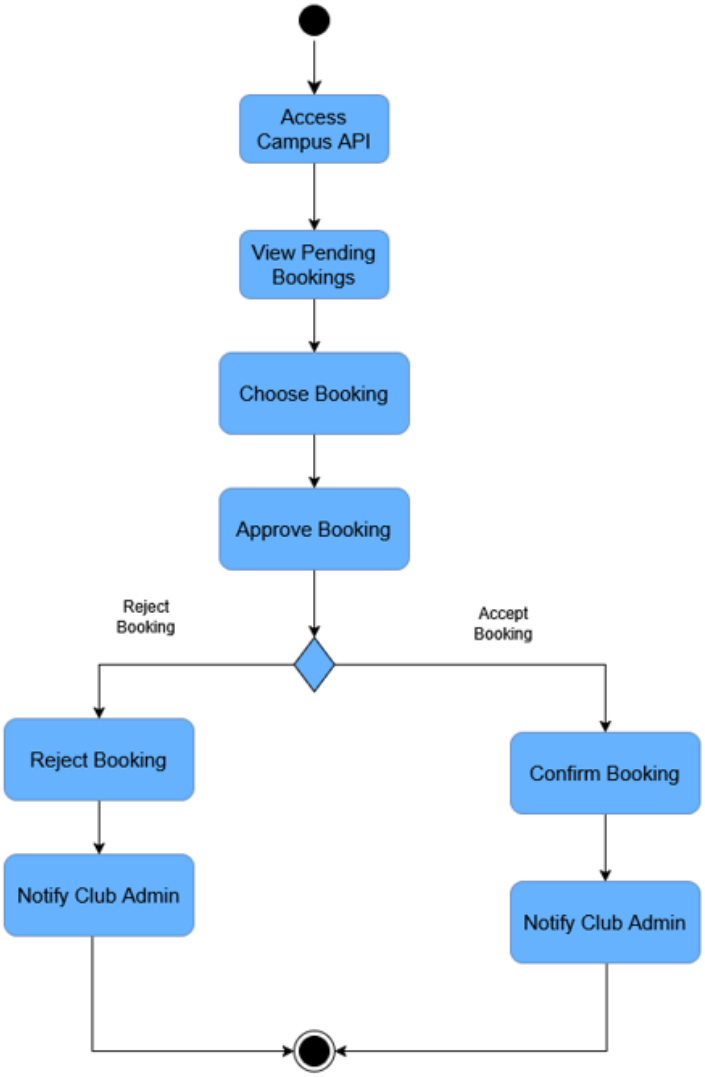

Figure 21.0 Activity Diagram – Book Venue

3.2 Performance Requirements

The performance requirements for Student Club Management System are:

Requirement ID	Description	Priority
REQ_P001	The average respond time of system shall less than 2 seconds.	High
REQ_P002	The system shall support at least 500 users performing action at same time.	High
REQ_P003	System should be at least 90% of uptime every day.	High
REQ_P004	The average responds time of system on external interface (eg. query venue availability) shall less than 5 seconds.	High
REQ_P005	The average responds time of system on peak load scenarios (eg. deadline of voting) shall less than 10 seconds.	High

Table 20.0 Performance Requirements

3.3 Usability Requirements

Requirement	Description	Measurable Criteria
Ease of navigation	Core function can be done in <= 3 click.	100% of core functions (e.g., register for club, submit report) testable within 3 clicks from the homepage.
Learnability	Complete core task in <= 10 minutes for new user.	90% of new users complete at least 3 core tasks (e.g., join club, vote) within 10 minutes, as tested in a controlled environment.
Consistency	Uniform layout and navigation.	90% of screens adhere to a consistent layout and navigation structure, verified against a predefined style guide (e.g., university branding).
Efficiency	Done frequence task in <= 30 seconds.	95% of experienced users complete frequent tasks (e.g., view event, submit vote) in ≤ 30 seconds.
Error Prevention and Recovery	Display clear error message and inline validation to prevent error, allow user to correct error in single action (re-enter input) within 3 clicks.	100% of errors on input fields (e.g., budget amount) have inline validation, and 90% of users can correct errors within 3 clicks.
User feedback	System must provide feedback for all actions (success message/loading indicator)	100% of user actions (e.g., submit report, book venue) trigger feedback (e.g., success message, loading indicator) within 2 seconds.
Device Accessibility	The interface should fully functional and readable on all device (phone, pc, tablet)	100% of core functionalities are accessible on iOS, Android, and desktop browsers.
Help and Documentation	A help feature must be available, providing step-by-step guidance for at least 80% of the system's use cases	Help feature covers step-by-step guidance for ≥ 80% of use cases (e.g., 16 out of 20 functions).
User Satisfaction	In user acceptance testing with a sample of 20 students and 5 club admins, at least 90% must report a satisfy of satisfaction.	≥ 90% of 25 participants (20 students, 5 club admins) rate satisfaction ≥ 4 on a 5-point scale during user acceptance testing.

Table 21.0 Usability Requirements

3.4 Interface Requirements

3.4.1 System Interfaces

Name	University Authentication Interface	**Description of Purpose**	To authenticate users based on their role, ensuring secure access to the system.
Source of Input or Destination of Output	Input: User credentials (username, password) Output: Authentication response (success/failure)		
Valid range, accuracy and/or tolerance	Username: 3-18 characters Password: 8-20 characters, include at least 1 letter and 1 number Accuracy: 100% match required for successful authentication; tolerance for errors is 0%.		
Units of measure	N/A	**Timing**	Authentication response must be received within 2 seconds under normal load (1,00 concurrent users).
Relationships to other inputs/outputs	Input trigger a session creation on system. Failure response trigger an error message on user interface		
Data formats	LDAP query or JSON data		
Command formats	POST request with JSON data as body	**Data items**	Input: username and password Output: status and errorMessage

Table 22.0 System Interfaces

3.4.2 User Interfaces

Name	Web Dashboard	**Description of Purpose**	To provide a user-friendly and device compatible interface for all user roles to access system features .
Source of Input or Destination of Output	Input: User interactions (clicks) via browser. Output: Visual feedback (success messages, loading indicator) on the browser.		
Valid range, accuracy and/or tolerance	Form fields (e.g., event date): Valid input (YYYY-MM-DD). Accuracy: 100% of inputs must be validated; tolerance for invalid inputs is 0%.		

Units of measure	Pixels (e.g., button size ≥ 48x48 pixels).	**Timing**	Interface updates (e.g., loading new data) must done within 2 second after user action.
Relationships to other inputs/outputs	Input trigger API call. Output display data from other entities like Club, Event.		
Data formats	HTML, CSS, JavaScript for UI; JSON for interaction with backend.		
Command formats	POST/GET request with optional JSON data as body	**Data items**	Input: eventName, date, voteOption Output: eventList, memberList, successMessage

Table 23.0 User Interfaces

3.4.3 Hardware Interfaces

Name	User Device Compatibility	**Description of Purpose**	To ensure the system is usable on a variety of user devices (PC, tablet, smartphone) for all user roles, supporting the system's multi-platform deployment.
Source of Input or Destination of Output	Input: User interactions (e.g. click, tap, keyboard input) from devices such as desktop, laptop, tablet, and smartphone. Output: Visual feedback (e.g. updated interface, success messages) displayed on the device screen.		
Valid range, accuracy and/or tolerance	Screen resolution: Minimum 320x480 pixels (smartphone) to 1920x1080 pixels (pc). Accuracy: 95% of interactions must shown correctly across multiple devices; tolerance of error is 5%.		
Units of measure	Pixels (for screen resolution).	**Timing**	Response to user inputs (e.g., button press, form submission) must occur within 1 second on all supported devices.
Relationships to other inputs/outputs	Input triggers the same actions as defined in the user interface (e.g., form submission, navigation). Output mirrors web dashboard outputs (e.g. eventList, successMessage) across all devices.		
Data formats	N/A		
Command formats	N/A	**Data items**	Input: Device-specific input data (e.g., mouse coordinates, keyboard strokes).

Table 24.0 Hardware Interfaces

3.4.4 Software Interfaces

Name	Email Notification Interface	**Description of Purpose**	To send email notifications to user for event, budget update, and system alert.
Source of Input or Destination of Output	Input: Notification request from the system Output: Email sent to user's email address via a third-party service (e.g., Mailersend).		
Valid range, accuracy and/or tolerance	Email address: Valid format (e.g. 1221111111@student.mmu.edu.my). Accuracy: 100% of valid emails must be sent; tolerance for delivery failure is 1%.		
Units of measure	N/A	**Timing**	Email must be sent within 1 minute
Relationships to other inputs/outputs	Input is triggered by events like Event creation or Budget approval. Output is logged in the system for further analysis.		
Data formats	JSON for API requests to third party (e.g. Mailersend).		
Command formats	POST request to API with JSON data as body	**Data items**	Input: to, subject, content Output: status, errorMessage

Table 25.0 Software Interfaces

3.4.5 Communications Interfaces

Name	HTTPS Protocol	**Description of Purpose**	To ensure secure communication between the client and server for all data exchanges.
Source of Input or Destination of Output	Input: HTTP requests from client (e.g. form submissions, API calls). Output: HTTP responses to client (e.g. JSON data, HTML pages).		
Valid range, accuracy and/or tolerance	Payload size: 0 - 10 MB per request. Accuracy: 100% data integrity; tolerance for packet loss is 0%.		
Units of measure	Bytes (payload size)	**Timing**	Response must be delivered within 2 seconds under normal load.

Relationships to other inputs/outputs	Input triggers backend processing (e.g. database queries). Output is consumed by user interface.		
Data formats	JSON or HTML for payloads;		
Command formats	POST/GET request to API with optional JSON data as body	**Data items**	Input: HTTPS GET for "/home" Output: HTML for Home page

Table 26.0 Communications Interfaces

3.5 Logical Database Requirements

The Student Club Management System shall utilize a relational database to logically organize and manage data related to users, clubs, events, votes, budgets, reports, and external systems. The database will support core operations and maintain relationships between the entities as represented in below and the class diagram below.

1. **User:**
   - Attribute: userID, name, email, password
   - Description: Base class for system admin, club admin and student.

2. **Student:**
   - Attribute: userID, name, email, password (inherit from user)
   - Description: Aggregation to Club but can exist independently

3. **SystemAdmin:**
   - Attribute: userID, name, email, password (inherit from user)

4. **ClubAdmin:**
   - Attribute: userID, name, email, password (inherit from user)
   - Description: Aggregation to Club but can exist independently

5. **Club:**
    - Attribute: clubID, name, description, members, events, votes
    - Description: Composes multiple Event, aggregates multiple Budget, FinancialReport, ClubAdmin, Student

6. **Event:**
    - Attribute: eventID, title, description, date, status, club, venue, participants
    - Description: - Composition to Club. Composes multiple votes, aggregates multiple venue

7. **Venue:**
    - Attribute: venueID, name, location, capacity, isAvailable
    - Description: Aggregation to Event but can exist independently.

8. **Vote:**
    - Attribute: voteID, voters, event, votesForYes, votesForNo
    - Description: Composition to Event.

9. **Budget:**
    - Attribute: budgetID, amount, purpose, status, requestDate
    - Description: Aggregation to Club but can exist independently.

10. **FinancialReport:**
    - Attribute: reportID, club, budget, detail
    - Description: Aggregation to Club but can exist independently.
11. **UniversityFinancialManagement:**
    - Description: External system for financial tracking and budget submission.

12. **CampusSpaceReservation:**
    - Description: External system for venue availability checking and venue booking.

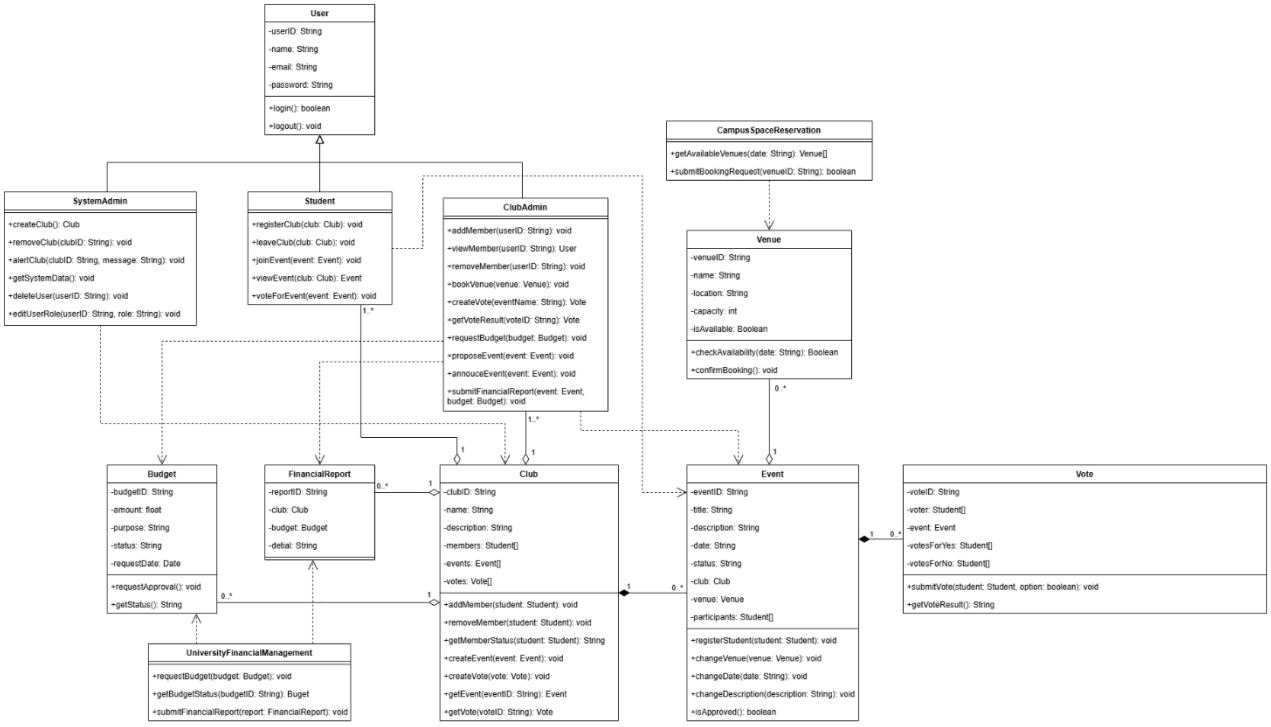

Figure 22.0 Class Diagram for Student Club Management with Budget & Venue Integration

3.6 Design Constraints

1. **University Branding Compliance**: The user interface design must follow the university's branding guidelines, including color schemes and logo placement.

2. **Web-Based Interface**: The platform must be accessible via web browsers and must be implemented to support both desktop and mobile views.

3. **Scalability Limit**: The system must be designed to handle a minimum of 30 clubs and 10,000 students, with database and server architecture constrained to scale efficiently within these limits without requiring a complete re-development of the system.

4. **Data Privacy Regulation**: The system must comply with applicable data privacy laws, ensuring that student and club admin data (e.g., names, emails) is encrypted and access is restricted to authorized users only.

5. **Performance Constraint**: The system must maintain average response times under 2 seconds, limiting the design to lightweight frameworks and optimizing database queries to meet this requirement.

6. **Authentication Security**: The system must use secure authentication mechanisms (e.g., password hashing, session management) to protect user accounts.

7. **Integration with University System**: The system must integrate with existing university systems (e.g., UniversityFinancialManagement and CampusSpaceReservation), constraining the design to use compatible APIs.

8. **Language and Localization**: The system must support the main language(s) of university and be designed to allow future localization (e.g., for international students).

9. **Hosting Constraint:** The system must be hosted on the university's network, and all data must store on servers located within the university's network for security.

3.7 Software System Attributes

Reliability:

- The system shall aim for 99.9% uptime**,** excluding planned maintenance and downtime of external integrated systems.
- Daily backups of the primary database shall be performed
- Backup success and integrity shall be monitored and logged.

Availability:

- The web server shall have a maximum downtime of 2 hours per 24-hour period.
- External APIs (Financial, Venue) are expected to have a maximum downtime of 2 hours per 24-hour period each.
- The system shall be accessible via modern browsers and responsive across desktop and mobile devices.

Security:

- The system shall support 2-Factor Authentication.
- User sessions shall expire after 30 minutes to prevent unauthorized access.
- The system shall implement **Role-Based Access Control (RBAC)** to ensure users can only access features and data appropriate to their role.
- API integrations shall use **API key authentication** with key rotation policies.
- All communication with the web server shall occur over **HTTPS** (TLS 1.2 or higher)
- Student data privacy shall comply with **GDPR/FERPA** and be continuously monitored by **Microsoft Defender for Cloud**

Maintainability:

- The system shall be developed using well-documented code and a modular design facilitated by the Django framework.
- Configuration settings should be manageable by System Admins where appropriate.

Portability:

- Being a web-based application, the client-side is portable across modern web browsers and operating systems (Windows, macOS, Android, iOS)
- Source code shall be maintained in **GitHub** with version control and GitHub Actions for CI/CD, enabling environment-independent builds.

3.8 Supporting Information

The following supporting information is provided to assist with understanding, developing, and maintaining the Student Club Management System. Unless explicitly stated, these items are not considered part of the formal system requirements, but serve to supplement the specification:

- **Sample Input/Output Format:**

```
{
  "eventID": "EVT001",
  "title": "Club Meeting",
  "description": "Monthly club discussion",
  "date": "2025-06-01T14:00:00Z",
  "clubID": "CLB001",
  "venueID": "VEN001"
}
```

Data format is JSON. Used by ClubAdmin to create an event via the RESTful API

- **Result from Questionnaires:**

Type	Requirements
Delighter	The system will be available 99.9% of the time
	Only the authorized users can access certain features
	The system can be accessed via desktop and mobile browsers
	The system can handle large numbers of users and do not crash
Type	Requirements
Satisfier	Ability to view list of clubs and their descriptions (Student)
	Ability to join or leave a club (Student)
	Ability to add or remove club members (Student)
	Ability to request venue reservation for events (Club Admin)
	Ability to view venue availability for the event (Club Admin)
	The response time for user actions is under 3 seconds
	The data is backed up daily

Type	Requirements
Dissatisfier	-
Type	Requirements
Indifferent (Need to Further Discuss)	Ability to view upcoming club events (Student)
	Ability to vote for club events (Student)
	Ability to filter or search for clubs/events (Student)
	Ability to submit event proposals (Club Admin)
	Ability to initiate budget requests (Club Admin)

- **Background Context:**

The system is designed to manage student club activities at a university with a student population exceeding 10000, requiring scalability and integration with existing university systems.

- **Technical Assumptions:**
  The system assumes access to a reliable internet connection and compatibility with standard browsers (e.g., Chrome, Firefox) and mobile OS versions (iOS 15+, Android 11+).

- **Problem1-Inefficient Club Management:**
  Current manual processes for managing club memberships and events are time-consuming, affecting over 30 clubs and their administrators. The system will automate these tasks, reducing administrative overhead by at least 50%.

- **Problem2-Lack of Financial Oversight:**
  Clubs struggle to track and submit budgets, leading to delays in funding approvals. The system will provide a structured process for budget requests and financial reporting, ensuring compliance with university financial policies.

- **Problem3-Poor Event Coordination:**
  Students and admins face challenges in scheduling and voting on events due to lack of centralized information. The system will offer real-time event updates and voting mechanisms, improving participation rates by an estimated 30%.

- **Problem4-Limited Accessibility:**
  Current tools are not optimized for mobile use, limiting access for students and admin. The system will provide a responsive interface across devices, enhancing usability.

- **Security & Packaging instructions:**
  Implement Two-Factor Authentication (2FA) for all administrative users, enhancing protection against unauthorized access. Source code and

configuration files will be version-controlled and packaged via GitHub repositories, with access restricted through organization-based role permissions.

4 Verification

The verification process ensures that the Student Club Management System with Budget and Venue Integration meets the specified functional, performance, usability, interface, logical database, design constraint, and software system attribute requirements outlined in Section 3 of this document. Verification will be conducted throughout the development lifecycle, ensuring that each component and the system as a whole perform as intended.

4.1 Verification Approach

The verification of the Student Club Management System will be conducted through a combination of methods, involving various team members and stakeholders at different stages and environments. This approach ensures that all specified requirements are adequately tested.

How:

- **Unit Testing:** Developers will conduct unit tests on individual modules and functions to verify that each component behaves as expected. This will focus on the logic of each product function and internal workings of software attributes.
- **Integration Testing:** Testing will be performed to verify the interfaces and interactions between different system components, including the crucial integration with external systems like the University Financial Management System and Campus Space Reservation Database. This will also cover the interactions between different user roles and their respective functionalities.
- **Functional Testing:** End-to-end functional testing will validate that the system performs all the functions specified in SRS Section 3.1 from the user's perspective, ensuring all use cases are correctly implemented according to the defined roles.
- **Performance Testing:** Performance tests, including load and stress testing, will be conducted to ensure the system meets the performance requirements outlined in SRS Section 3.2 (e.g., response times, concurrent user support for 10,000+ students, scalability for 50 clubs).
- **Usability Testing:** Representative end-users (Students, Club Admins, System Admins) will participate in usability testing to evaluate the ease of use, intuitiveness of the UI, and overall user experience against requirements in SRS Section 3.3.

- **Security Testing:** Security testing, potentially including vulnerability assessments and checks against university IT policies, will be performed to verify the security attributes defined in SRS Section 3.7 (e.g., Role-Based Access Control via Microsoft Entra ID, 2FA, HTTPS, API key usage, session timeouts, and data privacy compliance with GDPR/FERPA via Microsoft Defender for Cloud).
- **User Acceptance Testing:** A select group of end-users (System Administrators, Club Administrators, and Students), and potentially university staff overseeing club activities, will conduct UAT in a staging environment to confirm the system meets their needs and is fit for purpose before deployment.
- **Documentation Review:** The development team and key stakeholders will review all supporting documentation (as referenced in SRS Section 3.8), including user manuals and API documentation, for accuracy and completeness.

Who:

- **Development Team** Responsible for unit testing, initial integration testing, and supporting other testing phases.
- **Project Lead/Coordinator (from Development Team or University):** Responsible for overseeing the verification process, reviewing test results, and ensuring requirements are met. Involved in documentation review.
- **End-Users (System Admins, Club Admins, Students):** Responsible for participating in usability testing and UAT, providing critical feedback.
- **University IT/Financial Staff (as needed):** May be involved in UAT or consultation, particularly for workflows involving external system integrations (University Financial Management System, Campus Space Reservation Database).

When:

- **Unit Testing:** Continuously during the development of each module/feature.
- **Integration Testing**: After unit testing of individual components and as major features involving integrations are completed.
- **Functional Testing:** Iteratively as features or sets of features are developed and integrated.
- **Performance Testing:** Conducted at key milestones, especially before UAT, to validate scalability and response times under load.

- **Usability Testing:** Conducted iteratively with prototypes or early versions and more formally once a stable version of the UI is available.
- **Security Testing:** Conducted periodically and comprehensively before UAT and deployment.
- **User Acceptance Testing (UAT):** Conducted after the system has passed internal testing and is considered feature-complete for the 3-month development cycle.
- **Documentation Review:** Ongoing as documentation is produced and finalized before system release.

Where:

- **Development Environment:** For unit testing and initial developer-led integration testing.
- **Staging Environment:** An environment as identical to production as possible for UAT and final pre-deployment checks.
- **Production Environment:** Post-deployment monitoring and smoke testing will occur here.

4.2 Verification Criteria

The system will be considered verified when all the following criteria are met:

4.2.1 Functional Requirements

- All use-case scenarios (F001–F019) must pass their corresponding functional test cases with 100% success rate.
- No critical or major defects may remain open; any remaining defects must be classified as low severity and approved for deferral by the Project Lead.

4.2.2 Performance Requirements

- Average system response time under normal load shall be less than 2 seconds for 95% of transactions.
- Under peak load (500 concurrent users), response time shall remain below 10 seconds for 95% of transactions.
- The system shall demonstrate at least 90% uptime during performance test window.

4.2.3 Usability Requirements

- Core tasks (e.g., join club, submit vote) must be completed in no more than 3 clicks.
- New users must complete a core scenario in under 10 minutes in 90% of test sessions.
- User satisfaction score (via SUS or similar) must average at least 80% across a representative sample of 20 students and 5 club admins.

4.2.4 Interface Requirements

- All API calls to external systems (financial and venue) must have a success rate of at least 98% during integration testing.
- UI shall render correctly without errors on the latest versions of Chrome, Firefox, Edge, Safari, on both desktop and mobile form factors.

4.2.5 Logical Database Requirements

- Data integrity tests (foreign-key constraints, unique indices) must pass with zero violations.
- All CRUD operations on each entity (“User”, “Club”, “Event”, etc.) must be completed successfully in automated schema-validation tests.

4.2.6 Design Constraints

- Compliance checks (branding, encryption, authentication) must yield no violations in a governance audit.
- All code modules must conform to style guidelines with zero lint errors.

4.2.7 Software System Attributes

- Security scans (static analysis, vulnerability assessment) must report zero high or critical vulnerabilities.
- Automated backups must complete daily with 100% success and be restorable in a sandbox environment.
- System uptime measured over a one-week production-like test must meet or exceed 99.9%.

5 Appendices

5.1 Assumptions and Dependencies

Assumptions:

Dependency	Purpose	Risk if Unavailable
University Financial Management System	For real-time budget tracking, approval workflows, and fund disbursement.	Budget features will not function if the API fails.
Campus Space Reservation Database (CSRD)	For checking venue availability and submitting booking requests.	Venue booking functionality becomes unusable.
Microsoft Entra ID	Used for secure authentication and role-based access control (RBAC).	Unauthorized access or denial of access to legitimate users.
Azure Cloud Services	Includes hosting (App Service), storage (Azure SQL), and auto-scaling (Load Balancer).	Extended downtime in case of Azure service outages.
IBM DB2	Production database for core entities like member lists, budgets, and bookings.	Risk of data loss, corruption, or system-wide disruption.
Development Timeline	The project assumes a 3-month completion timeline with no critical delays in API availability or testing.	Risk of feature cutbacks or reduced quality if deadlines slip.
Microsoft Outlook API	Sends booking confirmations and event notifications.	Users may miss critical updates or reminders.

5.2 Acronyms and Abbreviations

SRS: Software Requirements Specifications

UI: User Interface

API: Application Programming Interface

CSRD: Campus Space Reservation Database

2FA: Two-Factor Authentication

RBAC: Role-Based Access Control

GDPR: General Data Protection Regulation

FERPA: Family Educational Rights and Privacy Act

SLA: Service-Level Agreement

LDAP: Lightweight Directory Access Protocol