Software Requirements Specification

for

Student Club Management System with Budget and Venue Integration

Tutorial Section: TT5L

Group Name: TT5L_G1

Student Name	**Student ID**
TAI ZHI XUAN	1211109038
HAZIM ELAMIN MOHAMED ALI MUSA	241UC2400P
IZZA NELLY BINTI MOHD NASIR	1211111583
SAY SI TING	1211108625

Contents

1. Introduction

1.1 Purpose

The purpose of the Student Club Management System is to streamline and automate essential processes involved in student club management within the university. This software system is

designed to address significant operational inefficiencies related to club registration, event management, budget tracking, and communications among club members and administrative staff. Currently, manual handling of these processes leads to excessive paperwork, delayed approvals, lack of real-time budget visibility, and difficulties in managing event attendance and venue reservations.

By digitizing these processes, the Student Club Management System aims to enhance student engagement, ensure accurate and timely communications, and provide efficient oversight of club activities and financial management. It aligns closely with the university's administrative goals of promoting active student participation, reducing paperwork burdens, improving transparency, and fostering a collaborative community environment.

1.2 Scope

The Student Club Management System is intended to automate and support the following key functionalities:

- Club Registration: Enables secure online submission, review, and approval of new club registrations.
- Event Management: Provides a structured workflow for event creation, planning, and management, including an integrated guided event creation wizard.
- Budget Management: Facilitates real-time submission, tracking, approval, and reporting of budget requests and expenditures, including integration with the university's financial system.
- Role Assignment: Clearly defines and manages roles for club officers, such as President, Treasurer, Secretary, and committee members, enforcing role-based access control.
- Member Registration: Allows students to join clubs, register for events, and manage personal preferences through an intuitive online interface.

- Venue Booking Integration: Integrates with the university's Venue Reservation System to streamline venue search, availability checking, and reservation confirmations.
- Notifications and Alerts: Implements an automated notification service for approvals, rejections, event reminders, and budget updates via email, push notifications, and other digital communication methods.

Out of scope for the initial release:

- Management of alumni club memberships.
- Full-scale social media integration beyond basic notifications.
- Comprehensive integration with third-party external payment gateways.

The defined scope ensures clarity in expectations, efficient project management, and targeted software delivery, directly addressing the primary needs identified through the requirements elicitation and analysis phases.

1.3 Product Overview

The Student Club Management System (SCMS) is a secure, web-based application that unifies all day-to-day club activities—registration, event planning, budget control, communication, and analytics—within a single digital workspace. Users interact with the system through role-based dashboards (Club Member, Club Officer, Student Affairs Officer, Finance Officer, Venue Manager, and System Administrator), while the system exchanges data with several university-wide services to ensure end-to-end automation.

SCMS supports the university's administrative objectives by:

- Reducing paperwork and approval delays through online forms, automated workflows, and electronic signatures.
- Increasing transparency and accountability with real-time budget balances, audit trails, and role-based access control.
- Enhancing student engagement via personalised dashboards, mobile notifications, and gamification badges that recognise active participation.
- Synchronising disparate services (finance, venue booking, notifications) into a cohesive experience, eliminating double-entry and data silos.

Key external integrations include:

- University Authentication Service – single sign-on (SSO) and multi-factor authentication for all users.
- University Financial System – live ledger queries, budget request posting, and transaction recording.
- Venue Reservation System – real-time venue discovery, availability checks, and reservation confirmations.
- Notification Service – email, SMS, and push notifications for approvals, reminders, and critical updates.

1.3.1 Product Perspective

SCMS is positioned as a specialised subsystem within the university's Digital Campus Platform. Figure 1 (Context Diagram) illustrates the high-level interactions between SCMS, its human actors, and three enterprise services (finance, venue, notifications).

- Actors / Front-end Interfaces
  - Students (Club Members) access SCMS via browser or mobile app.
  - Club Officers, Student Affairs Officers, Finance Officers, Venue Managers, and System Administrators access advanced dashboards aligned with their responsibilities.
- Core SCMS Services
  - Identity & Access Management layer (built-in credential store and role based access control).
  - Club & Event Module (handles registration, role assignment, event workflow).
  - Finance Module (handles budget requests, approvals, live balance retrieval).
  - Integration Gateway (REST / message bus adapters for external systems).
  - Notification Engine (dispatches messages via the Notification Service).
  - Reporting & Analytics Module (dashboards, exports, gamification metrics).
- External Systems
  - University Financial System: SCMS sends approved budget transactions and retrieves real-time balances.
  - Venue Reservation System: SCMS queries for venue availability and submits confirmed bookings.
  - Notification Service: SCMS publishes notification events; the service delivers them via e-mail, SMS, or push.

This architectural positioning allows SCMS to reuse existing enterprise services (finance, messaging) while providing a tailored feature set for student-club operations. The loose coupling via RESTful APIs and message queues supports future scalability—additional modules (e.g., alumni management) can be added without disrupting existing integrations.

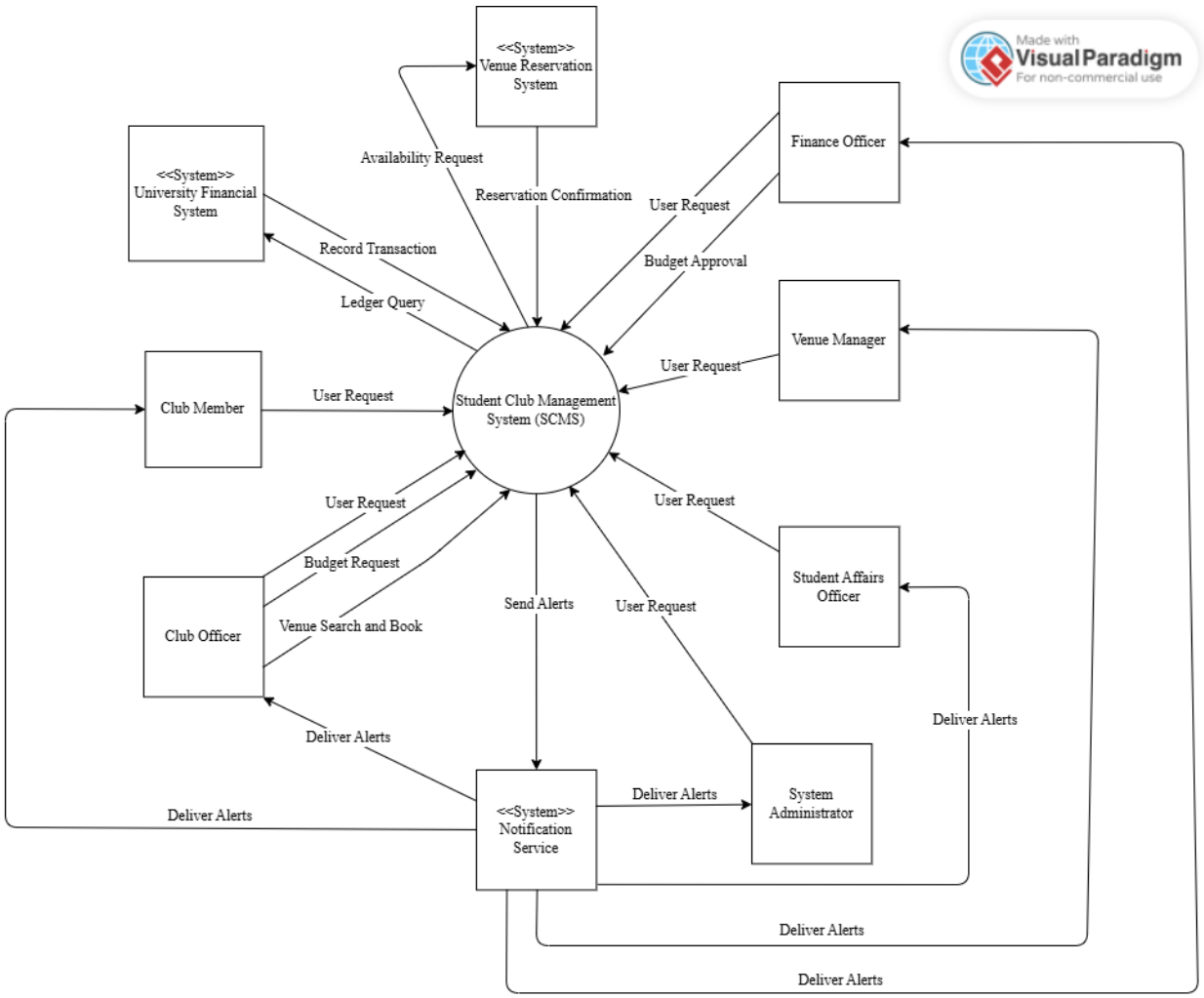


Figure 1 Context Diagram

Figure 1 presents the Level 0 Context Diagram for SCMS. It places SCMS at the centre of its environment, shows six human actors and three external enterprise services, and labels all major data flows.

1.3.2 Product Functions

The SCMS provides comprehensive functionalities to support effective club management and administrative workflows. The key functions include:

- Club Registration: Secure and automated online registration, submission, review, and approval workflows.
- Role Assignment: Assignment and management of clearly defined club officer roles (e.g., President, Treasurer, Secretary).
- Event Creation and Management: An intuitive, guided event creation wizard integrated with venue reservation and attendee tracking.
- Budget Tracking and Management: Real-time tracking, submission, approval, and reporting of club financial activities.
- Approval Workflows: Structured and automated processes for approvals from relevant officers (Student Affairs, Finance) for events, budget requests, and club registrations.
- Venue Integration: Seamless real-time integration with the university's Venue Reservation System for booking venues.
- Member Engagement: Personalized dashboards, event RSVP capabilities, attendance tracking via QR/NFC, and member-to-member messaging.
- Notifications and Alerts: Automated notifications and alerts via email, SMS, and mobile push notifications for all major system events.
- Reporting and Analytics: Dashboard-based analytics and exportable reports for attendance, budget management, and overall club engagement metrics.

1.3.3 User Characteristics

The SCMS caters to various user groups within the university community, each with distinct roles and characteristics:

- Club Members: Primarily undergraduate and graduate students with basic computer literacy. The system features a highly intuitive, accessible user interface designed specifically for users without technical expertise.
- Club Officers (President, Treasurer, Secretary): Students with moderate computer skills who manage club activities and require slightly advanced functionalities such as budgeting and event management.

- Student Affairs Officer: Administrative staff with intermediate technical proficiency, responsible for approving club activities and ensuring compliance with university policies.
- Finance Officer: University finance administrators who possess strong familiarity with financial management systems and require detailed reporting and approval capabilities.
- Venue Manager: Facilities personnel with intermediate to advanced familiarity with digital booking systems, overseeing venue reservation processes.
- System Administrator: IT personnel with advanced technical expertise responsible for configuring, monitoring, and maintaining the SCMS.

Usability considerations in SCMS focus on ensuring simplicity and intuitiveness, allowing all users—from non-technical students to advanced administrative users—to interact efficiently and effectively.

1.3.4 Limitations

The following limitations and constraints define the operational boundaries of the SCMS:

- The system exclusively supports student clubs officially recognized and governed by the university's Student Affairs department.
- Initially, the SCMS will be accessible via web browsers only, with mobile app availability planned for subsequent releases.
- Financial functionalities are limited to interaction with the university's internal financial system, excluding integrations with external third-party payment services.
- Venue bookings are restricted to university-owned facilities, and the system does not currently accommodate external venue management.
- The initial release excludes alumni club memberships and broader social media integrations, except basic notification functionalities.

These constraints ensure a clearly defined operational scope for the initial deployment, facilitating streamlined delivery and robust initial implementation.

1.4 Definitions

Term/Acronym	Definition
Student Affairs	The department or division responsible for providing services that support student success and oversee student clubs and activities.
Expenditure	The amount of money spent by a club, tracked and managed through the system's financial module.
Audit Trail	A security-relevant chronological record or set of records that provide documentary evidence detailing the sequence of activities affecting a specific operation, procedure, or event.
Club Officers	Students granted administrative privileges within a club, responsible for managing club-related activities, events, and finances.
Venue Reservation System	An external university-managed system that handles the booking, management, and availability tracking of physical spaces used for events.
Notification Service	An external service used to send automated emails, SMS, and push notifications to users regarding system-generated alerts and updates.
Single Sign-On (SSO)	Authentication service allowing users to access multiple university services with a single set of login credentials.
System Administrator	IT personnel responsible for system configuration, monitoring, maintenance, and ensuring reliable operation of the SCMS.

2. References

IEEE. (2018). ISO/IEC/IEEE 29148:2018 Systems and software engineering—Life cycle processes—Requirements engineering. https://www.iso.org/standard/72089.html

Object Management Group. (2017). Unified Modeling Language (UML), Version 2.5.1. https://www.omg.org/spec/UML/2.5.1/

Pohl, K. (2010). Requirements engineering: Fundamentals, principles, and techniques. Springer.

World Wide Web Consortium. (2008). Web Content Accessibility Guidelines (WCAG) 2.0. Retrieved from https://www.w3.org/TR/WCAG20/

3. Requirements

3.1 Functions

This section enumerates all externally visible functions of the Student Club Management System (SCMS), organized by actor. For each function, detailed behavior is captured in a corresponding use-case specification and illustrated by sequence diagrams.

3.1.1 Function List

Actor / External System	**Function List**
Club Member	• Register • Log in • QR Sign-in via Mobile • RSVP to Event • Send Messages • Read Messages • Join Club Online • Scan QR Attendance • Manage Accessibility Preferences • Personal Dashboard • Reminder Alert • Mobile push opt-in
Club Officer	• Log in • Club Registration • Create Event • Assign Member Roles • Export Financial CSV • Submit Budget Request • View Budget Balance • Track Attendance

Student Affairs Officer	• Log in • Process Club Registration • Process Event Proposal • Monitor Compliance
Finance Officer	• Log in • View Budget Balance • Process Budget Request • Disburse Funds
Venue Manager	• Log in • Maintain Venue Catalogue • Process Booking
System Administrator	• Log in • Maintain User Access • Configure Integrations • Generate Audit Trail • Monitor System Health
University Financial System	• Record Approved Transactions • Provide Live Balances
Venue Reservation System	• Provide Availability Search • Reserve Venue
Notification Service	• Deliver Status Notifications • Send Reminder Alerts

3.1.2 Use-Case Diagram

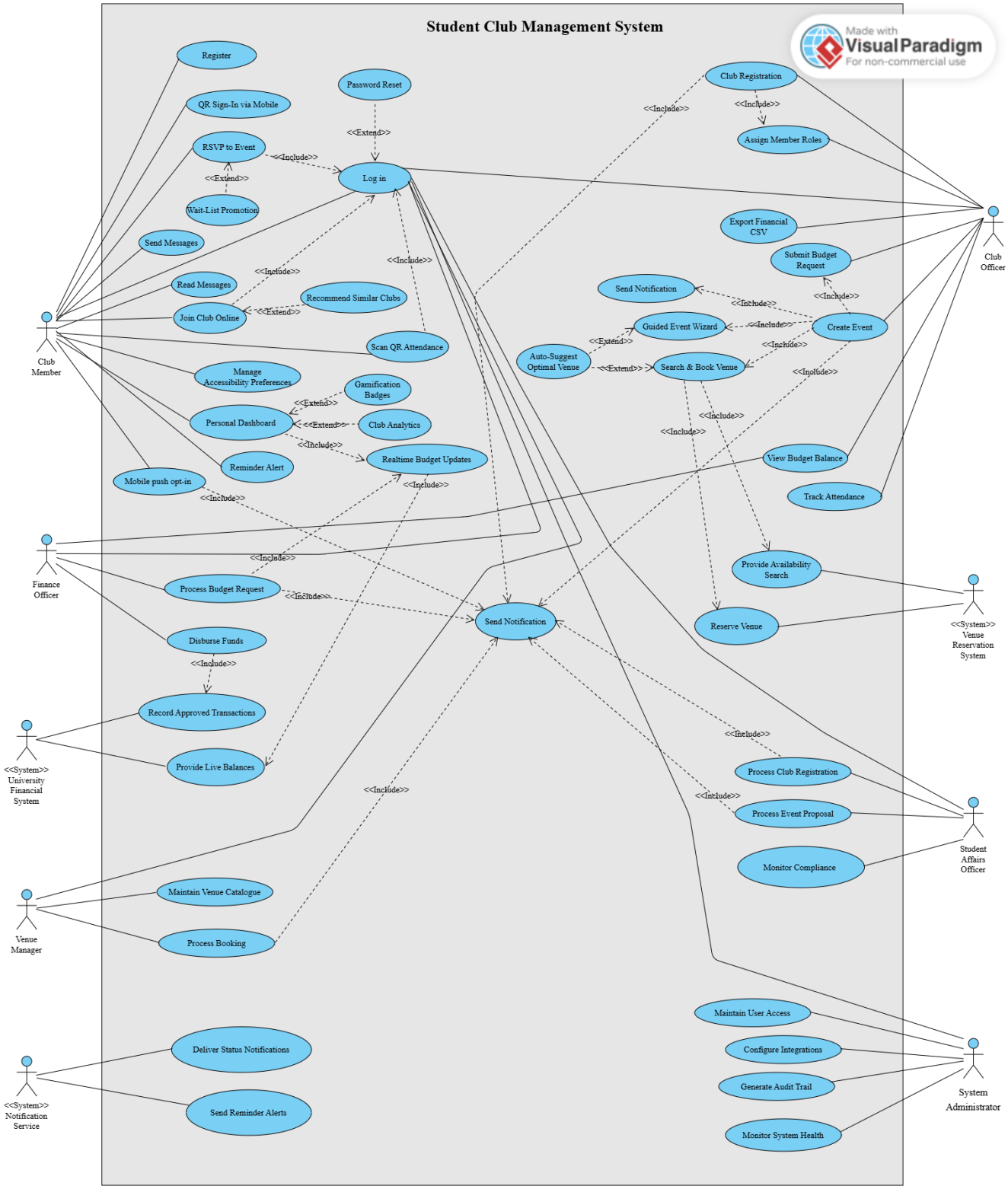


Figure 2 showing SCMS and all actor-to-usecase links.

3.1.3 Use-Case Specifications

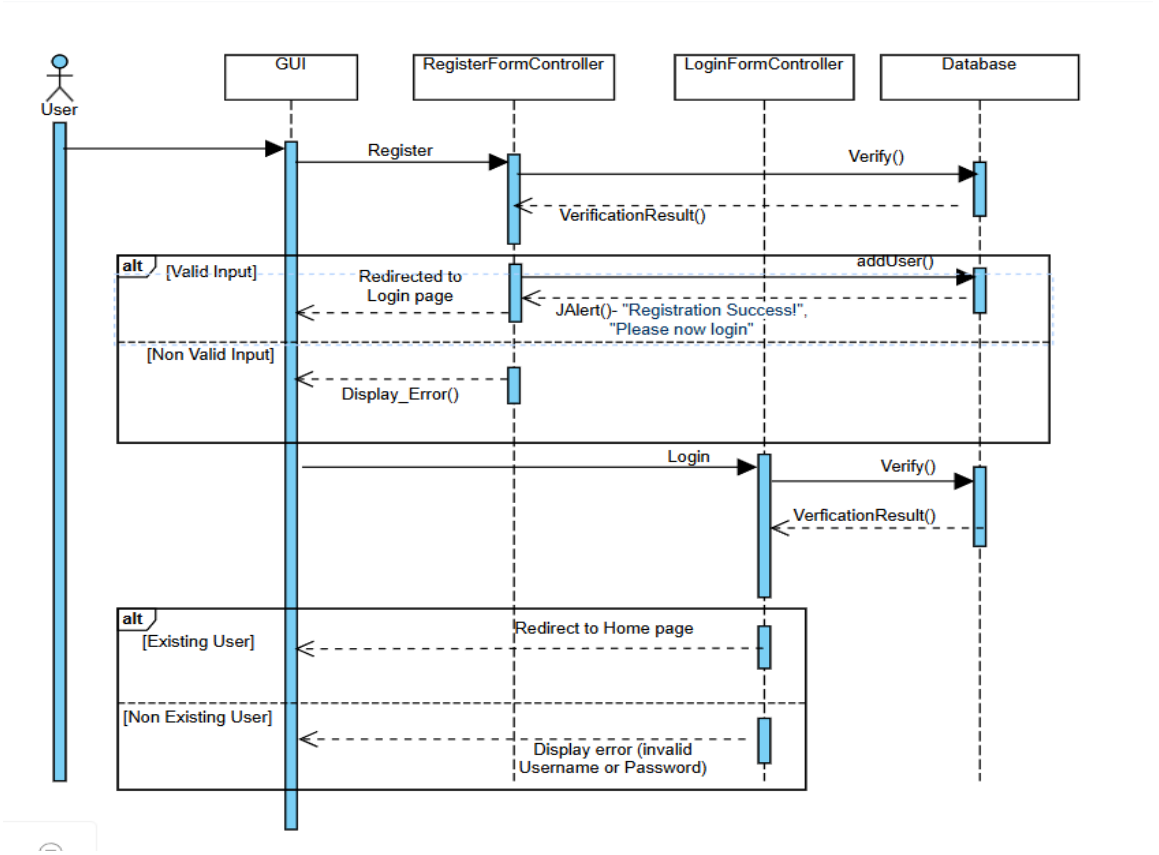
Below are subsections for each actor and external system. Each subsection contains use-case specifications and associated sequence diagrams.

3.1.3.1 Club Member

3.1.3.1.1 Use Case: Club Member Login

Use Case Element	Description
Primary Actor	Student (club member).
Purpose	Allow existing members to authenticate and access their club account
Preconditions	• Member has previously registered account • Member knows their login credentials (email/username and password) • System authentication service is operational
Postconditions	• Member is authenticated and logged into the system • Session is created and maintained • Member has access to authorized club features • Login activity is recorded for security purposes
Trigger	Members want to access their club account
Main Flow	1. Member navigates to login page 2. Member enters email/username and password 3. System validates credentials against database 4. System verifies member account status 5. System creates secure session token 6. System redirects member to dashboard or requested page 7. Member gains access to personalized club features
Alternative Flow – Additional Info Needed	• Invalid credentials: System displays error message and offers password reset • Account locked/suspended: System shows account status message

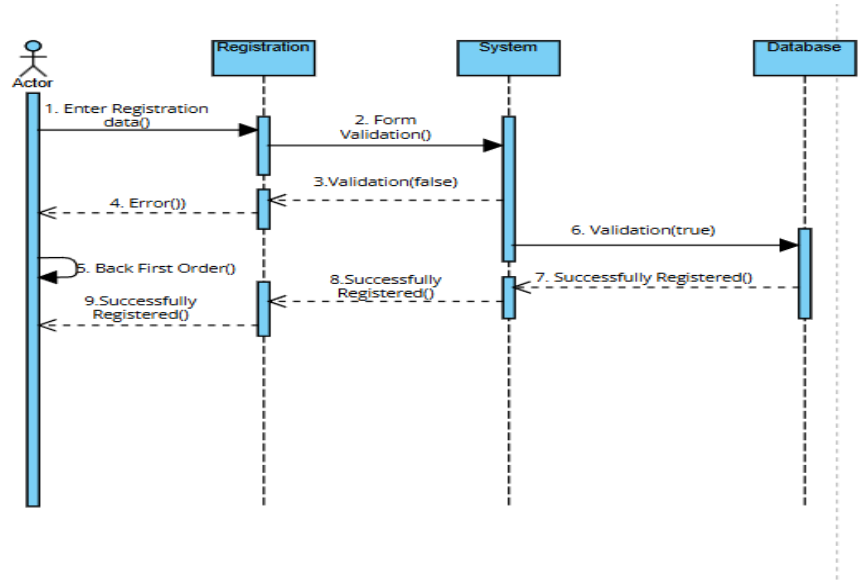
	• Forgot password: System provides password recovery option • Account not verified: System prompts to check email for verification link



3.1.3.1.2 Use Case: Club Member Register

Use Case Element	Description
Primary Actor	Student
Purpose	To enable new users to establish an account and gain access to club services
Preconditions	• Registration is open for new members • User has valid email address • Registration form and validation rules are configured
Postconditions	• New member account is created in the system • Verification email is sent to member • Member can login after email verification • Member profile is established with basic information
Trigger	New user wants to create a club account
Main Flow	1. User accesses registration form 2. User fills in required personal information and credentials 3. System validates form data for completeness and format 4. System checks if email/username is already registered 5. System creates new member account with pending status 6. System sends verification email to user's email address 7. System displays registration success message 8. User clicks verification link in email 9. System activates member account 10. User can now login with new credentials
Alternative Flow – Additional Info Needed	• Email already exists: System offers login option or password recovery • Invalid email format: System highlights field errors and requests correction • Weak password: System shows password strength requirements • Email verification not received: System offers to resend verification email

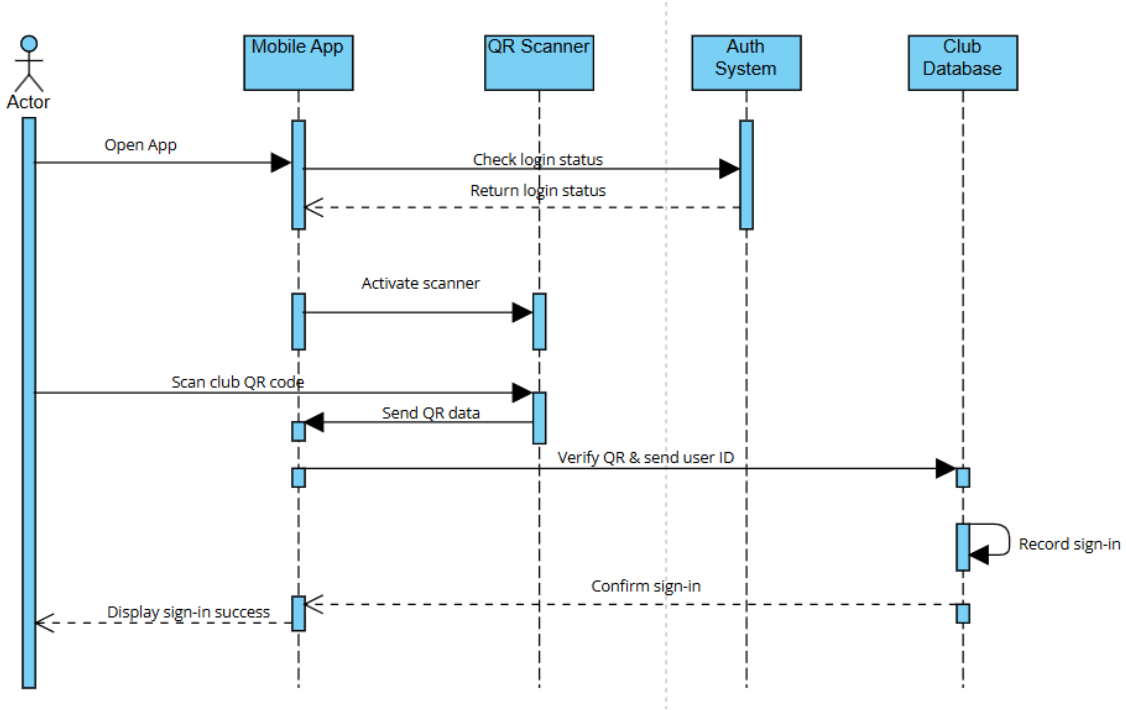
	• Verification link expired: System generates new verification link



3.1.3.1.3 Use Case: Sign In via mobile

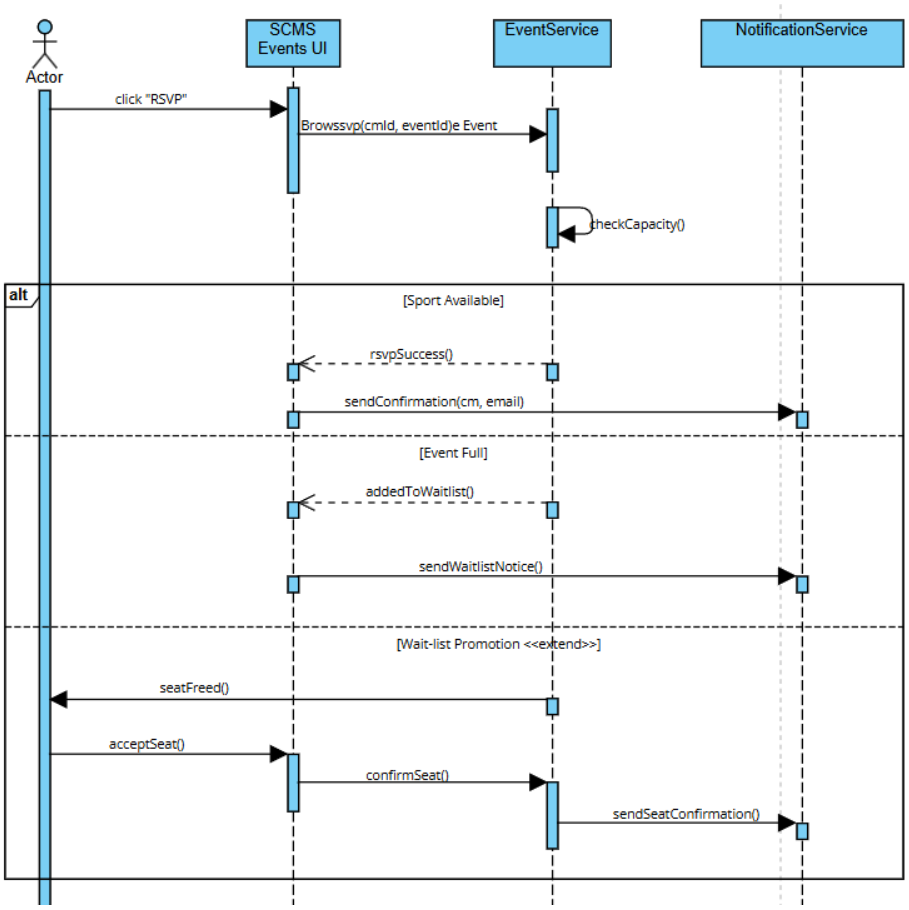
Use Case Element	Description
Primary Actor	Student (club member).
Purpose	To provide a quick and contactless way for members to check in when arriving at the club
Preconditions	• Member has mobile app installed and is logged in • Club has QR code displayed at entrance • Member has valid membership status
Postconditions	• Member's attendance is recorded in the system • Member receives confirmation of successful sign-in • Club staff can view real-time attendance data
Trigger	Member arrives at club and wants to sign in
Main Flow	1. Member opens mobile app 2. System checks member's login status 3. Member activates QR scanner feature 4. Member scans club's QR code 5. System verifies QR code and member credentials 6. System records sign-in with timestamp

	7. System displays success confirmation to member
Alternative Flow – Additional Info Needed	• QR code is damaged/unreadable: System prompts member to try manual check-in or contact staff • Member not logged in: System redirects to login screen • Invalid membership: System displays error message and directs to membership renewal



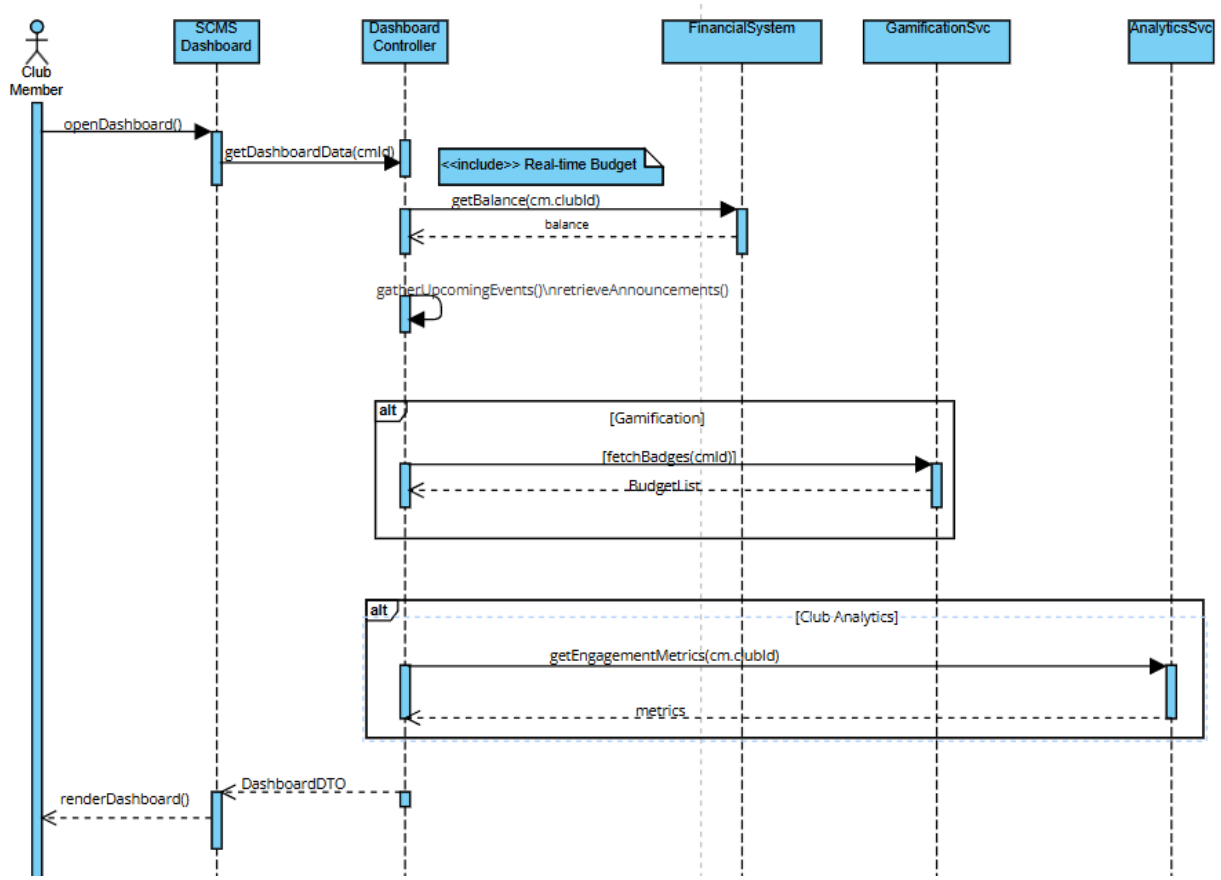
3.1.3.1.4 Use Case: RSVP Event

Use Case Element	Description
Primary Actor	Student (club member)
Purpose	To enable members to register for club events and help organizers plan accordingly
Preconditions	• Member has active account and is logged in • Events are published and available for RSVP • Member has permission to view/register for events
Postconditions	• Member's RSVP status is recorded for the event • Event organizer can view updated attendance list • Member receives RSVP confirmation • System may send reminder notifications before event
Trigger	Member wants to register for an upcoming club event
Main Flow	1. Member browses available events 2. System displays list of upcoming events 3. Member selects specific event 4. System shows event details and RSVP option 5. Member clicks RSVP button 6. System processes RSVP and updates attendance list 7. System sends confirmation notification to member
Alternative Flow – Additional Info Needed	• Event is full: System displays waitlist option or "event full" message • RSVP deadline passed: System shows "registration closed" message • Member already registered: System shows current RSVP status with option to cancel


Actor
SCMS Events UI
EventService
NotificationService
click "RSVP"
Browssvp(cmId, eventId)e Event
checkCapacity()
alt
[Sport Available]
rsvpSuccess()
sendConfirmation(cm, email)
[Event Full]
addedToWaitlist()
sendWaitlistNotice()
[Wait-list Promotion <<extend>>]
seatFreed()
acceptSeat()
confirmSeat()
sendSeatConfirmation()

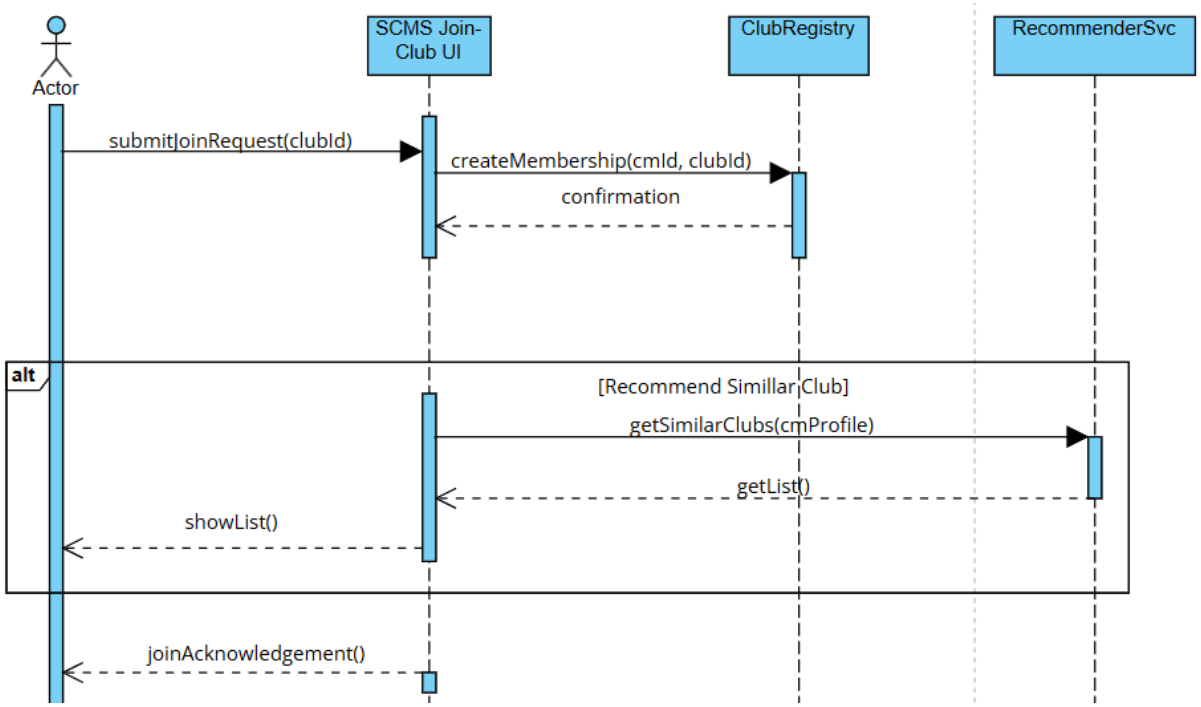
3.1.3.1.5 Use Case: Personal Dashboard

Use Case Element	Description
Primary Actor	Student
Purpose	To give members a centralized view of their club engagement and upcoming activities
Preconditions	• Member has active account and is logged in • Member has activity history or upcoming events • Dashboard components are configured
Postconditions	• Member sees personalized information summary • Dashboard reflects current and relevant data • Member can quickly access key functions • Activity metrics are updated
Trigger	Member logs into their account or navigates to dashboard
Main Flow	1. Member accesses personal dashboard 2. System retrieves member's activity data 3. System fetches upcoming events and deadlines 4. System compiles membership status information 5. System generates activity statistics 6. System displays personalized dashboard 7. Member interacts with dashboard elements 8. System updates display based on interactions
Alternative Flow – Additional Info Needed	• No recent activity: System displays welcome message and suggested actions • System maintenance: Dashboard shows limited information with notice • Slow data loading: System displays loading indicators and loads content progressively


Club Member
SCMS Dashboard
Dashboard Controller
FinancialSystem
GamificationSvc
AnalyticsSvc
openDashboard()
getDashboardData(cmId)
<<include>> Real-time Budget
getBalance(cm.clubId)
balance
gatherUpcomingEvents()\nretrieveAnnouncements()
alt
[Gamification]
[fetchBadges(cmId)]
BudgetList
alt
[Club Analytics]
getEngagementMetrics(cm.clubId)
metrics
DashboardDTO
renderDashboard()

3.1.3.1.6 Use Case: Student Join Club Member

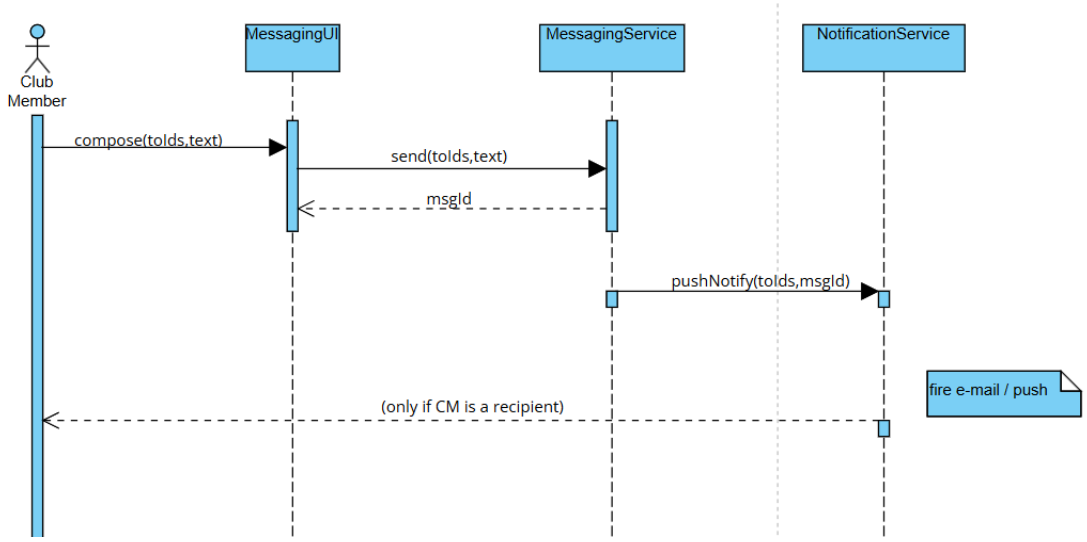
Use Case Element	Description
Primary Actor	Student
Purpose	To provide an easy way for new members to sign up and become part of the club
Preconditions	• Club has online registration enabled • Registration requirements are defined • Payment system is functional (if required)
Postconditions	• New member account is created • Member has access to club services • Membership fee is processed (if applicable) • Welcome materials are sent to new member • Member appears in club directory
Trigger	Person wants to become a club member
Main Flow	1. Prospective member browses available clubs 2. System displays club information and joining options 3. User selects club to join 4. System presents registration form 5. User completes personal information 6. System processes membership application 7. User completes payment (if required) 8. System creates member account 9. System sends welcome email and confirmation
Alternative Flow – Additional Info Needed	• Club membership is full: System offers waitlist option • Payment declined: System prompts for alternative payment method • Missing required information: System highlights incomplete fields • Duplicate membership: System checks for existing account


Actor
SCMS Join-Club UI
ClubRegistry
RecommenderSvc
submitJoinRequest(clubId)
createMembership(cmId, clubId)
confirmation
alt
[Recommend Simillar Club]
getSimilarClubs(cmProfile)
getList()
showList()
joinAcknowledgement()

3.1.3.1.7 Use Case: Club Member Send Messages

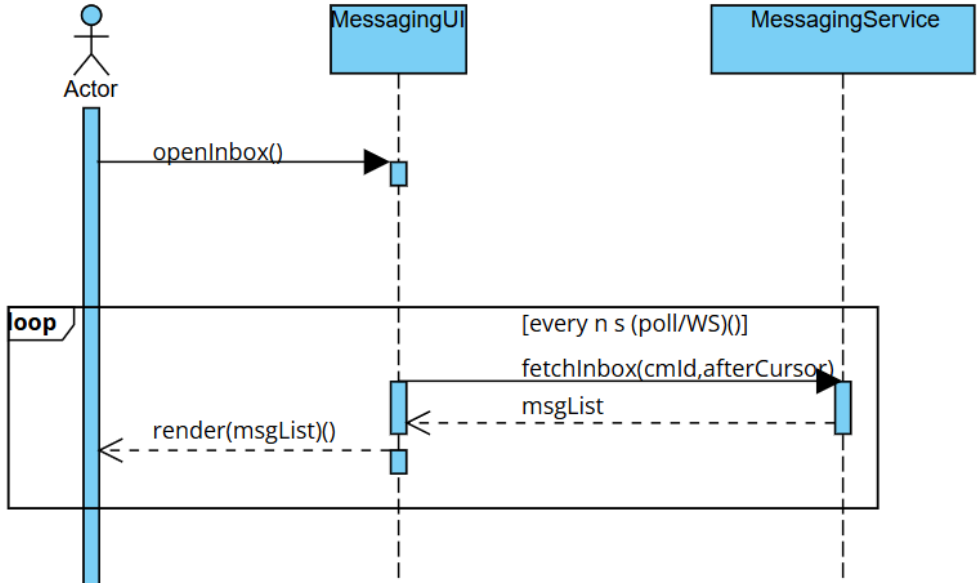
Use Case Element	Description
Primary Actor	Student (club member).
Purpose	To allow communication within the club community for announcements, discussions, and direct messaging
Preconditions	• User has active account and is logged in • User has messaging privileges • Recipients exist in the system and are accessible to the sender
Postconditions	• Message is stored in the system • Message is delivered to intended recipients • Recipients receive notifications based on their preferences • Message appears in sender's sent items • Message delivery status is tracked
Trigger	User wants to communicate with other club members
Main Flow	1. User opens messaging interface 2. User composes new message with subject and content 3. User selects recipients from contact list or enters recipient details 4. User reviews message content and recipients 5. User sends message 6. System validates recipients and message content 7. System stores message in database 8. System creates notifications for recipients 9. System delivers notifications based on recipient preferences 10. System confirms successful delivery to sender
Alternative Flow – Additional Info Needed	• Recipient not found: System displays error and suggests alternative recipients • Message contains inappropriate content: System flags for review or blocks • User lacks permission to message certain recipients: System shows access restriction

	• Network connectivity issues: System queues message for later delivery • Message exceeds size limit: System prompts user to reduce content size



3.1.3.1.8 Use Case: Club Member Read Messages

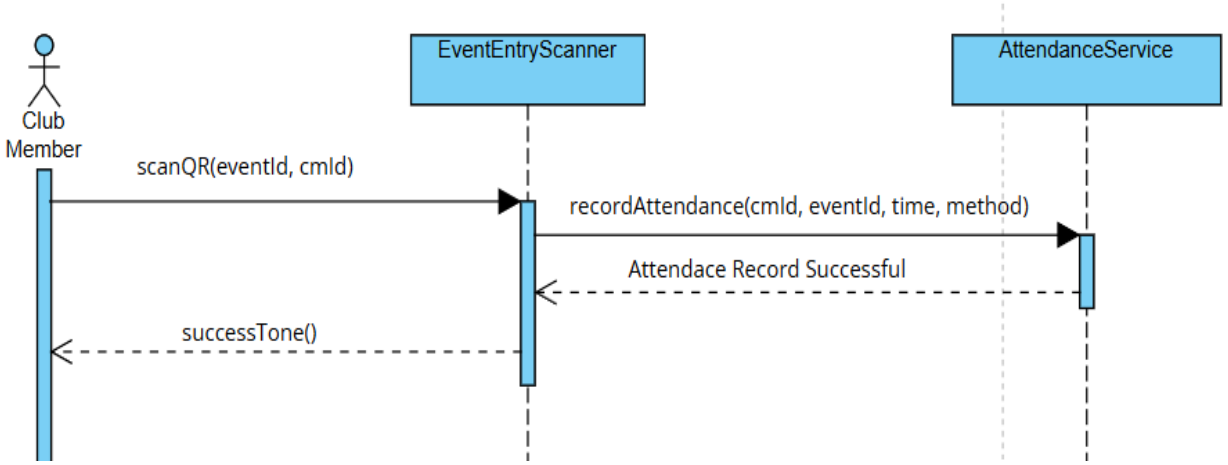
Use Case Element	Description
Primary Actor	Student (club member).
Purpose	To enable users to access and read communications from other club members
Preconditions	• User has active account and is logged in • User has received messages in their inbox • User has permission to access messaging system
Postconditions	• Message is marked as read in the system • Message content is displayed to user • Read status is updated for sender tracking • User can respond to or take action on the message
Trigger	User wants to check and read received messages
Main Flow	1. User opens inbox or message center 2. System retrieves all messages for the user 3. System displays list of messages with sender, subject, and timestamp 4. User browses message list (showing read/unread status) 5. User selects specific message to read 6. System marks message as read 7. System displays full message content 8. User can view message details, attachments, or related information
Alternative Flow – Additional Info Needed	• No messages available: System displays "no messages" notice • Message failed to load: System shows error and offers retry option • Message contains attachments: System provides download or preview options • Message is part of a conversation thread: System shows conversation history • User wants to reply: System provides reply interface with original message context


Actor
MessagingUI
MessagingService
openInbox()
loop
[every n s (poll/WS)()]
fetchInbox(cmId,afterCursor)
msgList
render(msgList)()

3.1.3.1.9 Use Case: Club Member Scan QR / NFC Attendance

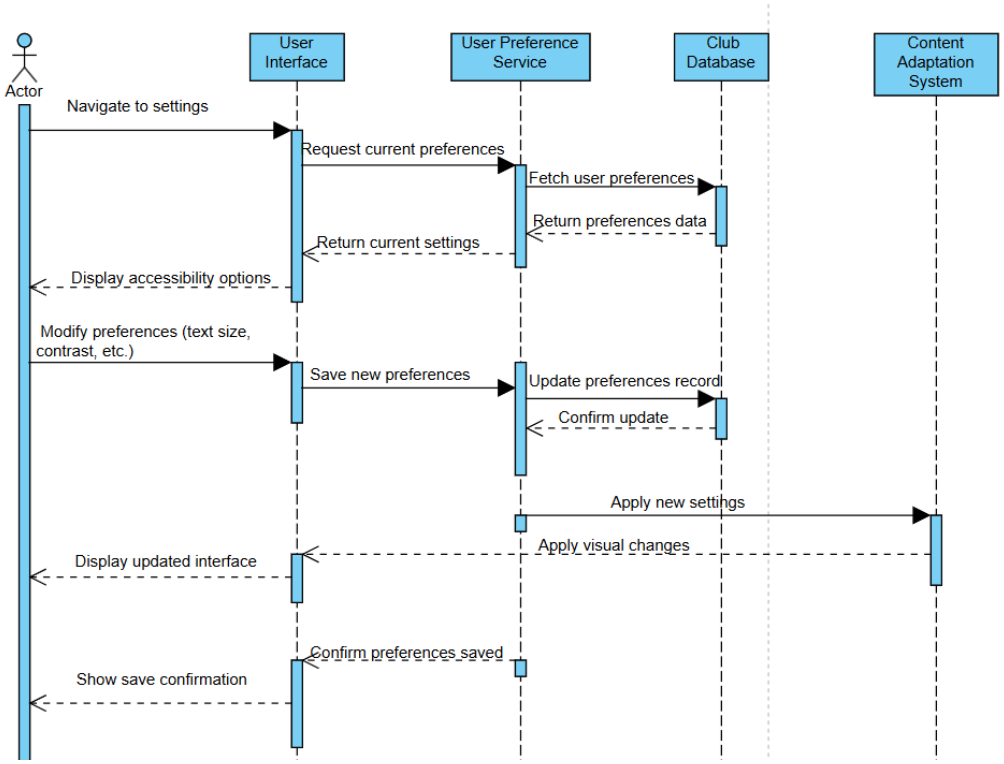
Use Case Element	Description
Primary Actor	Student (club member).
Purpose	To enable members to self-register their attendance at club activities, events, or facility visits
Preconditions	• Member using QR scanner to take attendance • QR codes or NFC tags are placed at event/location venues • Attendance tracking is enabled for the event/location • Member has valid membership status
Postconditions	• Member's attendance is recorded with timestamp and location • Attendance data is available for event organizers and reporting • Member receives confirmation of successful attendance recording • Member's participation history is updated
Trigger	Member arrives at club event or location and wants to record attendance
Main Flow	1. Member opens mobile app attendance feature 2. System activates QR/NFC scanner 3. Member scans QR code or taps NFC tag at venue 4. System captures attendance location and timestamp 5. System verifies member identity and event/location validity 6. System records attendance in database 7. System displays confirmation message to member 8. Member's attendance is logged for tracking purposes
Alternative Flow – Additional Info Needed	• QR code unreadable/NFC tag not working: System provides manual check-in option • Event/location not found: System displays error and suggests contacting organizer • Member already checked in: System shows previous check-in time • Network connectivity issues: System stores attendance locally and syncs when connection restored

	• Invalid event time: System notifies member if trying to check in outside event hours



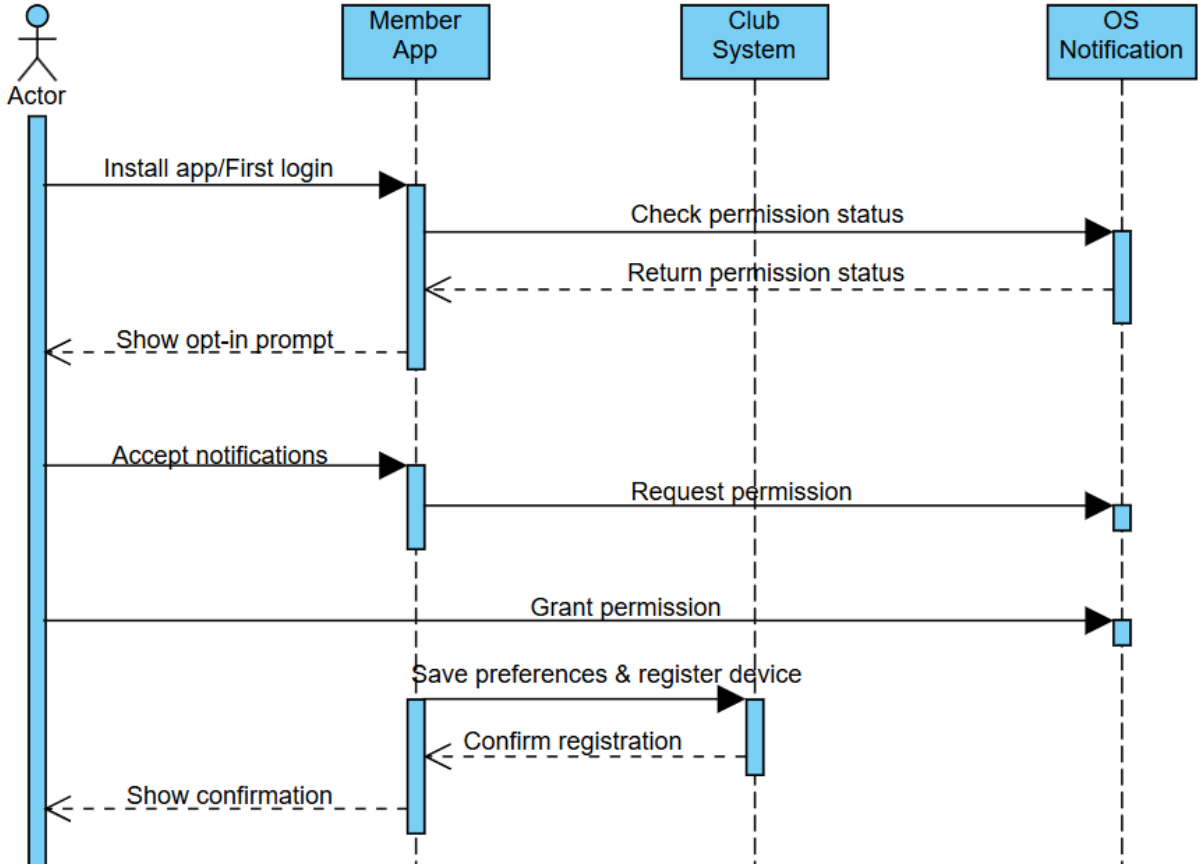
3.1.3.1.10 Use Case: Club Member Manage Accessibility Preferences

Use Case Element	Description
Primary Actor	Student (club member).
Purpose	To ensure the platform is usable by members with different accessibility requirements
Preconditions	• Member has active account and is logged in • Accessibility options are available in the system • User interface supports customization
Postconditions	• User preferences are saved and applied • Interface adapts to user's accessibility needs • Settings persist across sessions • User can easily modify preferences as needed
Trigger	Member wants to customize accessibility settings
Main Flow	1. Member navigates to settings/preferences 2. System displays current accessibility options 3. Member modifies preferences (text size, contrast, etc.) 4. System previews changes in real-time 5. Member saves new preferences 6. System applies settings across all interfaces 7. System confirms preferences saved
Alternative Flow – Additional Info Needed	• Conflicting settings: System suggests optimal combinations • Browser compatibility issues: System provides alternative options • Settings reset: System offers to restore previous preferences


Actor
User Interface
User Preference Service
Club Database
Content Adaptation System
Navigate to settings
Request current preferences
Fetch user preferences
Return preferences data
Return current settings
Display accessibility options
Modify preferences (text size, contrast, etc.)
Save new preferences
Update preferences record
Confirm update
Apply new settings
Apply visual changes
Display updated interface
Confirm preferences saved
Show save confirmation

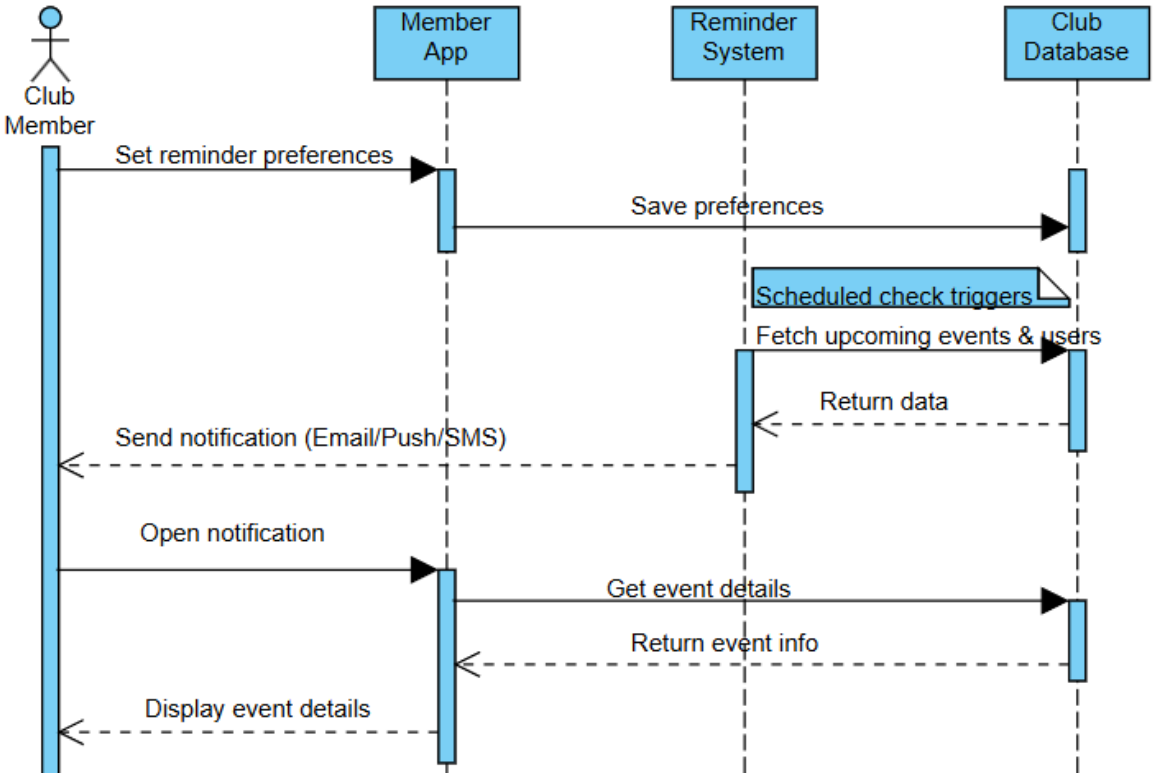
3.1.3.1.11 Use Case: Club Member Mobile Push Opt-in

Use Case Element	Description
Primary Actor	Student (club member).
Purpose	To obtain user permission for push notifications while respecting privacy preferences
Preconditions	• Member has mobile app installed • Device supports push notifications • App has necessary permissions framework • Member has not previously opted in
Postconditions	• Member's notification preference is recorded • Device is registered for push notifications (if opted in) • Member can receive relevant club notifications • Preferences can be changed later in settings
Trigger	Member installs app for the first time or accesses notification settings
Main Flow	1. Member installs app or logs in for first time 2. System checks current notification permission status 3. System displays opt-in prompt explaining benefits 4. Member accepts push notifications 5. System requests permission from device OS 6. Member grants permission through system dialog 7. System registers device and saves preferences 8. System confirms successful setup to member
Alternative Flow – Additional Info Needed	• Member declines: System respects choice and provides alternative notification methods • OS permission denied: System explains how to enable in device settings • Device not compatible: System offers alternative notification options • Member changes mind later: System provides easy access to modify preferences in settings


Actor
Member App
Club System
OS Notification
Install app/First login
Check permission status
Return permission status
Show opt-in prompt
Accept notifications
Request permission
Grant permission
Save preferences & register device
Confirm registration
Show confirmation

3.1.3.1.12 Use Case: Club Member Reminder Alert

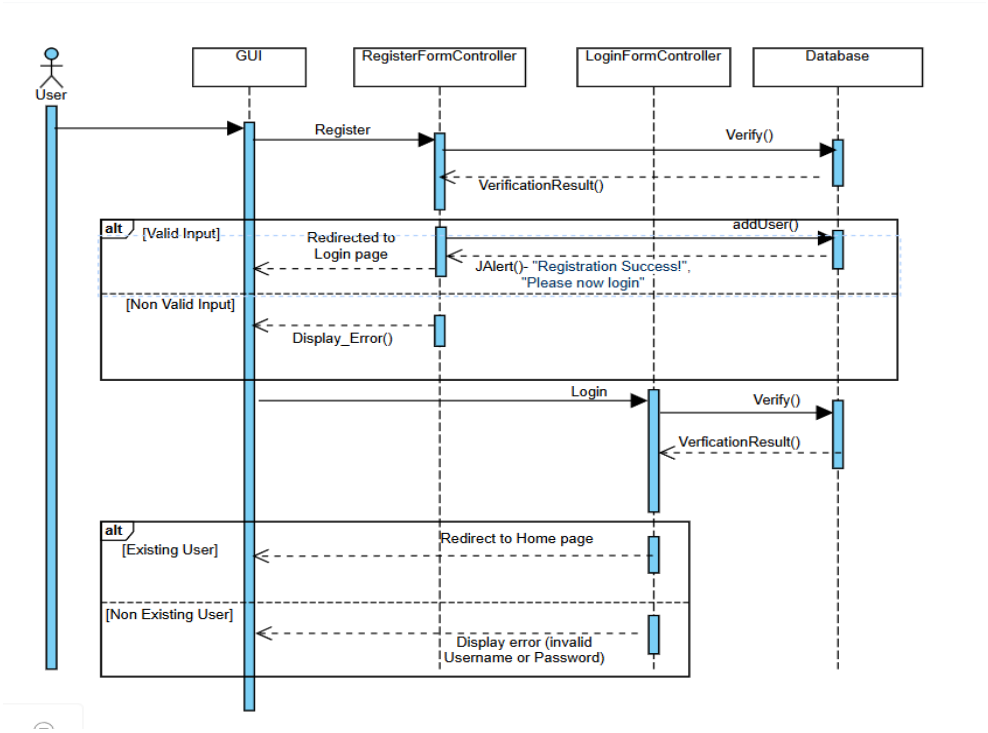
Use Case Element	Description
Primary Actor	Student (club member).
Purpose	To ensure members don't miss important events and stay engaged with club activities
Preconditions	• Member has opted in for reminder notifications • Events or deadlines are scheduled in the system • Member's contact preferences are configured • Reminder timing rules are set
Postconditions	• Relevant members receive timely reminders • Reminder delivery is logged in the system • Members can respond to reminders if needed • System tracks reminder effectiveness
Trigger	Scheduled time before an event or deadline
Main Flow	1. Member sets up reminder preferences 2. System stores preferences in database 3. Scheduled system process checks for upcoming events 4. System identifies members to notify based on preferences 5. System sends appropriate notifications (email/push/SMS) 6. Member receives and opens notification 7. System displays event details when member responds
Alternative Flow – Additional Info Needed	• Member has disabled notifications: System respects preference and skips • Event cancelled: System sends cancellation notice instead • Delivery failure: System attempts alternative notification method • Member responds with questions: System provides event organizer contact


Club Member
Member App
Reminder System
Club Database
Set reminder preferences
Save preferences
Scheduled check triggers
Fetch upcoming events & users
Return data
Send notification (Email/Push/SMS)
Open notification
Get event details
Return event info
Display event details

3.1.3.2 Club Officer

3.1.3.2.1 Use Case: Login

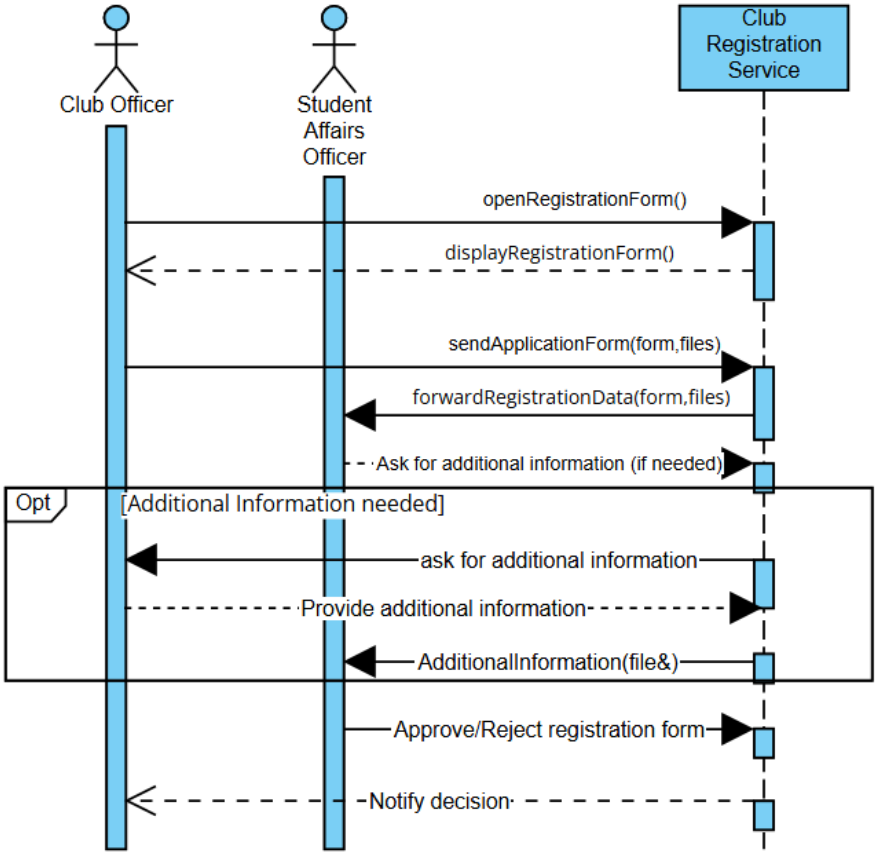
Use Case Element	Description
Primary Actor	Student (club officer).
Purpose	Allow existing officers to authenticate and access their club account
Preconditions	• Officer has previously registered account • Officer knows their login credentials (email/username and password) • System authentication service is operational
Postconditions	• Officer is authenticated and logged into the system • Session is created and maintained • Member has access to authorized club features • Login activity is recorded for security purposes
Trigger	Officer wants to access their club account
Main Flow	1. Officer navigates to login page 2. Officer enters email/username and password 3. System validates credentials against database 4. System verifies member account status 5. System creates secure session token 6. System redirects member to dashboard or requested page 7. Officer gains access to personalized club features
Alternative Flow – Additional Info Needed	• Invalid credentials: System displays error message and offers password reset • Account locked/suspended: System shows account status message • Forgot password: System provides password recovery option • Account not verified: System prompts to check email for verification link


User
GUI
RegisterFormController
LoginFormController
Database
Register
Verify()
VerificationResult()
alt
[Valid Input]
Redirected to Login page
addUser()
JAlert()- "Registration Success!", "Please now login"
[Non Valid Input]
Display_Error()
Login
Verify()
VerficationResult()
alt
[Existing User]
Redirect to Home page
[Non Existing User]
Display error (invalid Username or Password)

3.1.3.2.2 Use Case: Club Registration

Use Case Element	Description
Primary Actor	Student (club founder).
Secondary Actors	Student Affairs Officer
Purpose	To allow students to register a new club by submitting required documentation and forms for approval by student affairs.
Preconditions	• User is logged in as a verified student. • No existing club with the same name. • Club registration period is open.
Postconditions	• Club is either successfully registered and visible in the system or the registration is rejected or sent back for revision.
Trigger	• A student clicks on the "Register New Club" button from the dashboard.
Main Flow	1. The student clicks "Register New Club." 2. The system displays registration form (club name, objective, structure, logo, etc.). 3. The student fills out the form and uploads the required documents. 4. The student submits the registration request. 5. The system forwards the request to Student Affairs Officer for review. 6. The system sets status to "Under Review." 7. The student Affairs Office reviews the documents. 8. The system notifies the student when decision is made, and the status updated.
Alternative Flow – Additional Info Needed	• The student Affairs flags the application as "Incomplete." • The system sends notification to the applying student with requested additional information. • Student edits and resubmits the application.

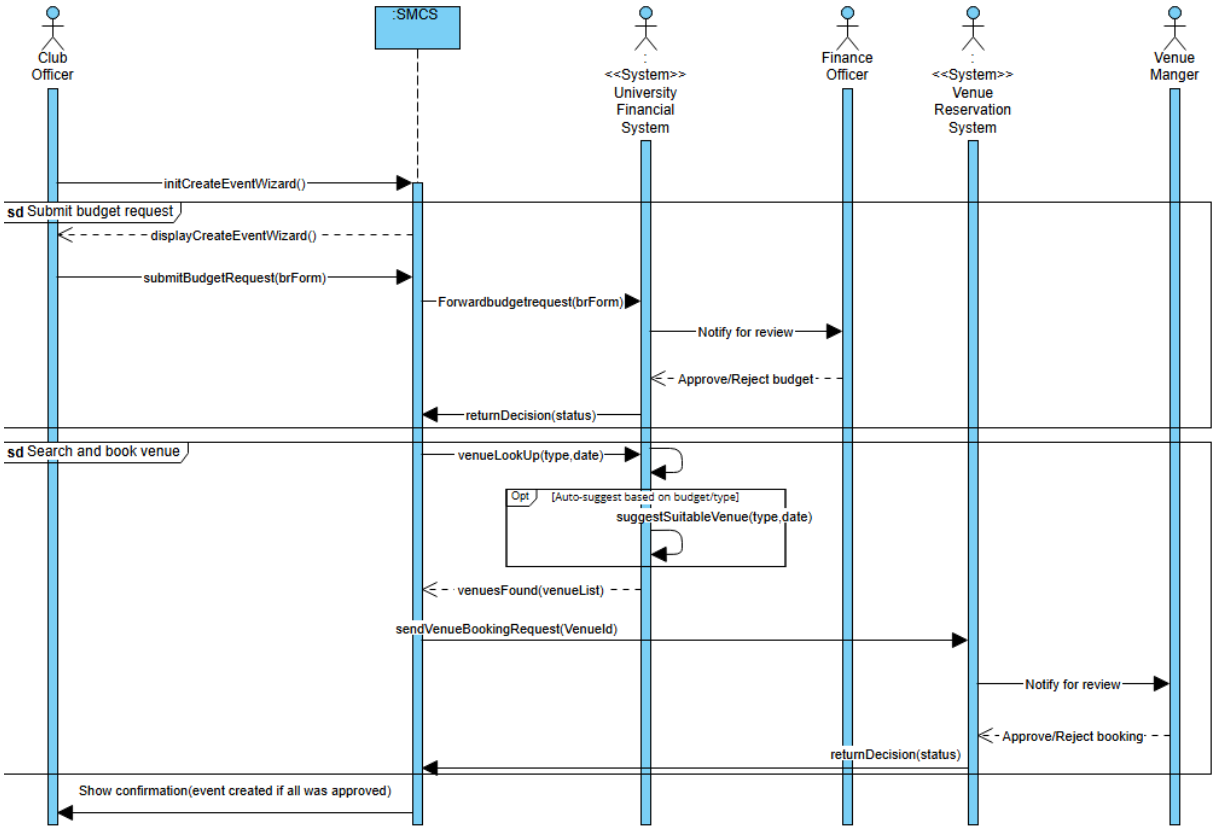
Alternative Flow – Application Rejected	• The student Affairs Officer rejects the registration request. • System notifies the student with rejection reason. • The student may revise and resubmit or cancel the request.
Alternative Flow – Application Approved	• The Student Affairs Officer approves the application. • Club status updated to "Active." • The system adds the club to the official directory and notifies all applicants.
Exception Flows	• System session times out, user is logged out. - Internet connection is lost. • Required fields are missing, the system highlights them and locks submission.



3.1.3.2.3 Use Case Specifications: Create Event

Use Case Element	Description
Primary Actor	Club Officer.
Secondary Actors	Venue Manager, University Financial System, Finance Officer, Venue Reservation System, Student Affair Officer.
Purpose	To allow the club officer to create and submit a new event proposal, including venue booking and budget requests, for administrative approval.
Preconditions	• User is registered as a club officer. • The club is active and registered. • The officer has permission to manage events.
Postconditions	• Event is created. • venue is booked. • Budget proposal is submitted. • Notifications are sent to club members.
Trigger	A Club officer clicks on the “create event” button on the system.
Main flow	1. Club officer clicks "Create Event". 2. The system displays event creation form. 3. Officer fills in event details (budget, date, type). 4. Officer searches for a venue using filters (size, location, availability). 5. The system returns a list of available venues. 6. Officer selects a venue → system reserves it tentatively. 7. Officer fills in budget items and submits the proposal. 8. System sends budget to Finance Officer for review. 9. System saves event as “Pending Approval”. 10. Notifications sent to relevant parties.
Alternative flow- No venue is available to book	• Officer receives a “no available venue" message. • The system suggests alternative dates or locations.
Alternative flow- Budget proposal was denied	• Finance Officer rejects the submitted proposal. • The System notifies the club officer with rejection reasons. • Officer may edit and resubmit or cancel the proposal.
Alternative flow- Event proposal was denied	• Student affair officer reviews and rejects the event proposal. • System notifies club officer of the decision and reason. • Officers can revise and resubmit or delete the proposed event.

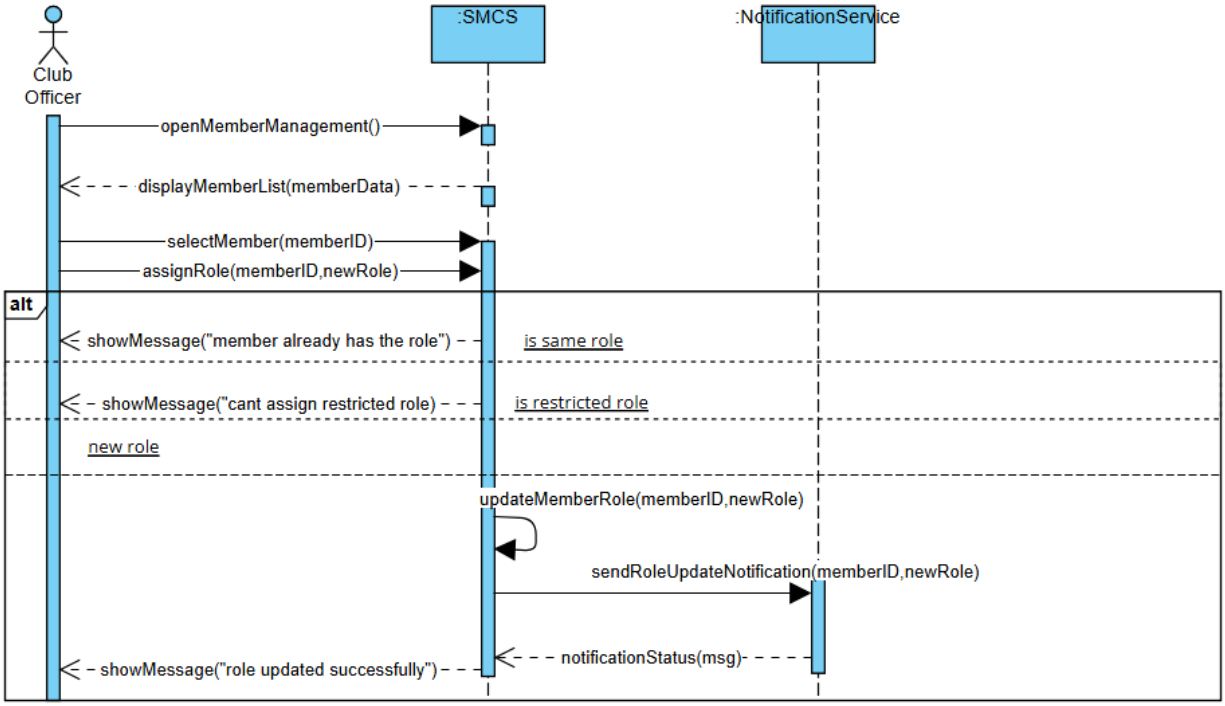
Exception flows	• System session times out, user is logged out and redirected to login page. • Internet connection is lost. • Required fields are missing, system points which fields are missing.



3.1.3.2.4 Use Case Specifications: Assign Member Roles

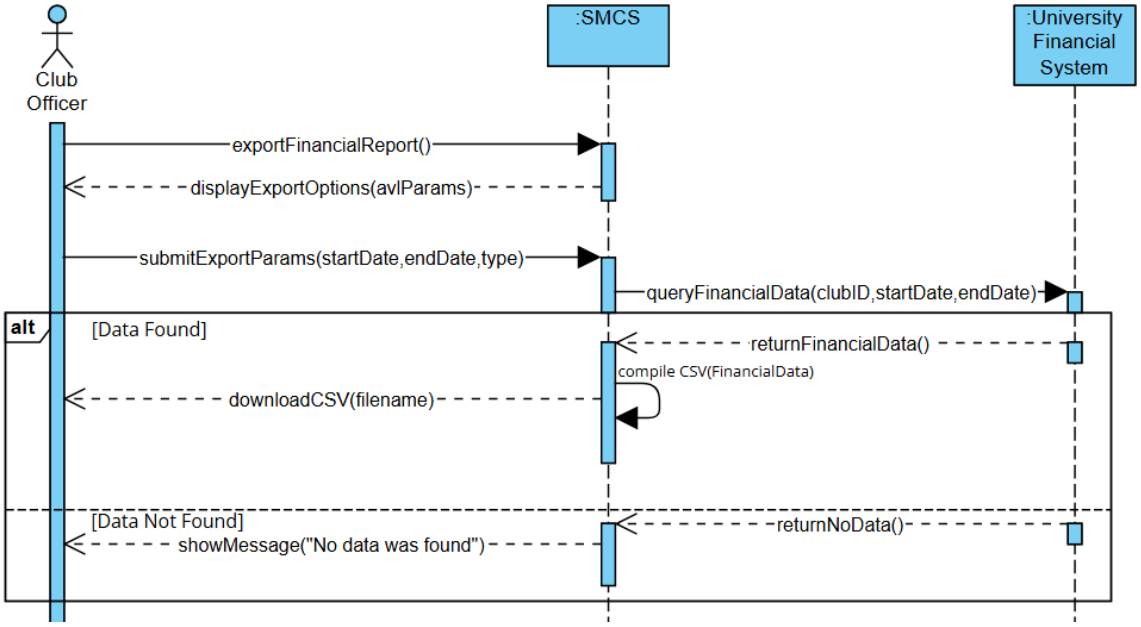
Use Case Element	Description
Primary Actor	Club Officer.
Secondary Actors	Club Members.
Purpose	To allow a club officer to assign or update roles (e.g., Treasurer, Secretary, Event Coordinator) to members within the club.
Preconditions	• User is registered as a club officer. • Club is active and registered. • Club members are listed and verified in the system.
Postconditions	• Selected member roles are updated in the system. • Affected members are notified of their new roles. • Role-based permissions are updated.
Trigger	A club officer clicks on the "Assign Roles" or "Edit Roles" option in the club dashboard.
Main Flow	1. Club officer accesses the member management panel. 2. The system displays a list of current members and their roles. 3. The officer selects a member. 4. Officer assigns or changes the selected member role from a dropdown menu (e.g., Treasurer, Secretary, etc.). 5. Officer confirms and submits changes. 6. System updates the role in the database. 7. Notification is sent to the selected member.
Alternative Flow – Member already has the selected role	• System prompts that the member currently holds the same role.
Alternative Flow – Role is restricted	• Officer attempts to assign a role restricted by the university rules (e.g., only one president per club). • System blocks assignment and shows message.
Exception Flows	• Session times out, user is redirected to login page. • Internet connection is lost. • Required fields (e.g., role selection) are missing, the system

	• highlights missing inputs and prevents submission.



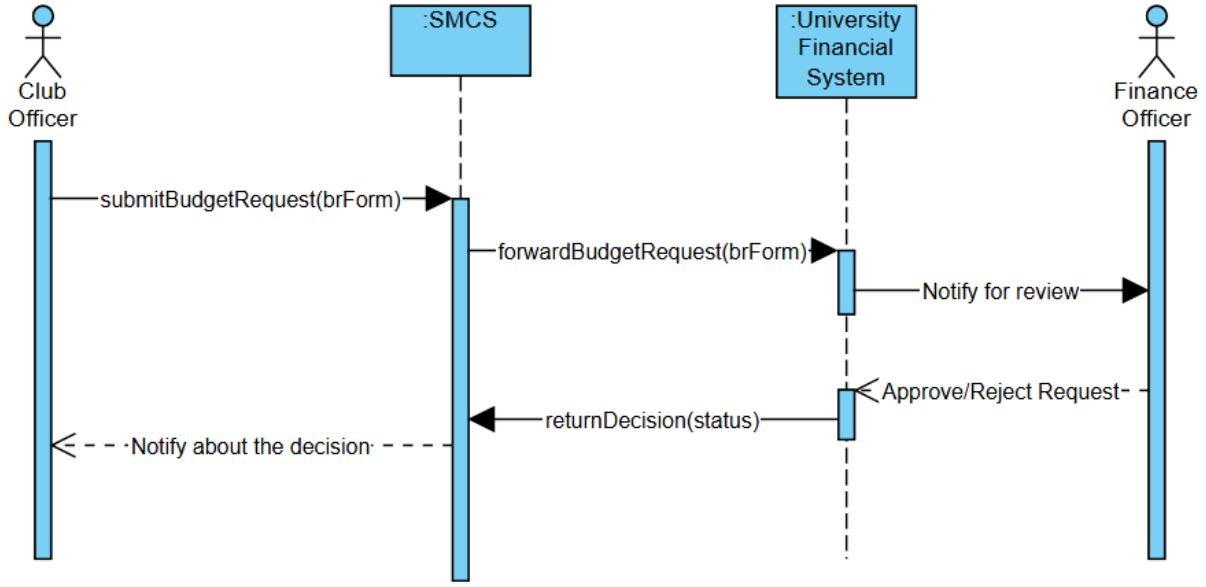
3.1.3.2.5 Use Case: Exporting Financial CSV

Use Case Element	Description
Primary Actor	Club Officer
Secondary Actors	University Financial System
Purpose	To allow a club officer to generate and download financial data (budget requests, spending records, balances) in CSV format for reporting, recordkeeping or reviewing.
Preconditions	• User is logged in with Officer permission. • Club is active and has existing financial data.
Postconditions	• CSV file is generated and downloaded.
Trigger	User clicks “Export Financial Report” .
Main Flow	1. User selects “Export Financial Report.” 2. System displays export options. 3. User selects parameters and confirms. 4. System queries the University Financial System for relevant data. 5. System compiles the data into CSV format. 6. CSV file is generated and download prompt appears.
Alternative Flow – No Data Available	• System finds no matching records for selected parameters. • Displays a message: “No financial data available for selected criteria.”
Exception Flows	• Financial system is unreachable; system displays error and retry option • Internet connection is lost during export, system aborts.


Club Officer
:SMCS
:University Financial System
exportFinancialReport()
displayExportOptions(avlParams)
submitExportParams(startDate,endDate,type)
queryFinancialData(clubID,startDate,endDate)
alt
[Data Found]
returnFinancialData()
compile CSV(FinancialData)
downloadCSV(filename)
[Data Not Found]
returnNoData()
showMessage("No data was found")

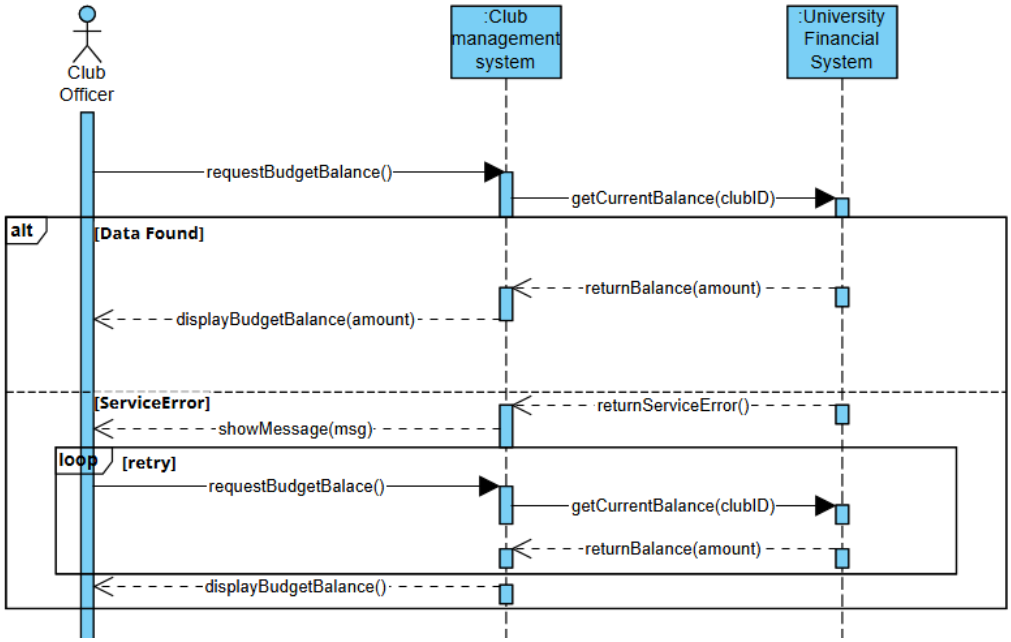
3.1.3.2.6 Use Case: Submit Budget Request

Use Case Element	Description
Primary Actor	Club Officer
Secondary Actors	Finance Officer, University Financial System
Purpose	To submit and process a budget request for approval or rejection.
Preconditions	• Club Officer filled a valid budget request form (brForm). • Student Club Management System and University Financial System are both operational.
Postconditions	• Budget request is approved or rejected. • Decision is communicated to the Club Officer.
Trigger	Club Officer initiates a budget request submission.
Main Flow	1. Club Officer submits the budget request. 2. System forwards the request to the university financial system. 3. The request then is forward to the finance officer to review the request. 4. Finance officer evaluates the request and approves/rejects it. 5. Club officer is notified of the decision made.
Alternative Flow – Data Unavailable	• If required data is missing, the system prompts the Club Officer to resubmit complete information. • Request remains pending until corrections are made.
Exception Flows	• If the University Financial System is unavailable, SMCS queues the request for retry. • If a system error occurs, the Finance Officer manually intervenes to process the request.


Club Officer
:SMCS
:University Financial System
Finance Officer
submitBudgetRequest(brForm)
forwardBudgetRequest(brForm)
Notify for review
Approve/Reject Request
returnDecision(status)
Notify about the decision

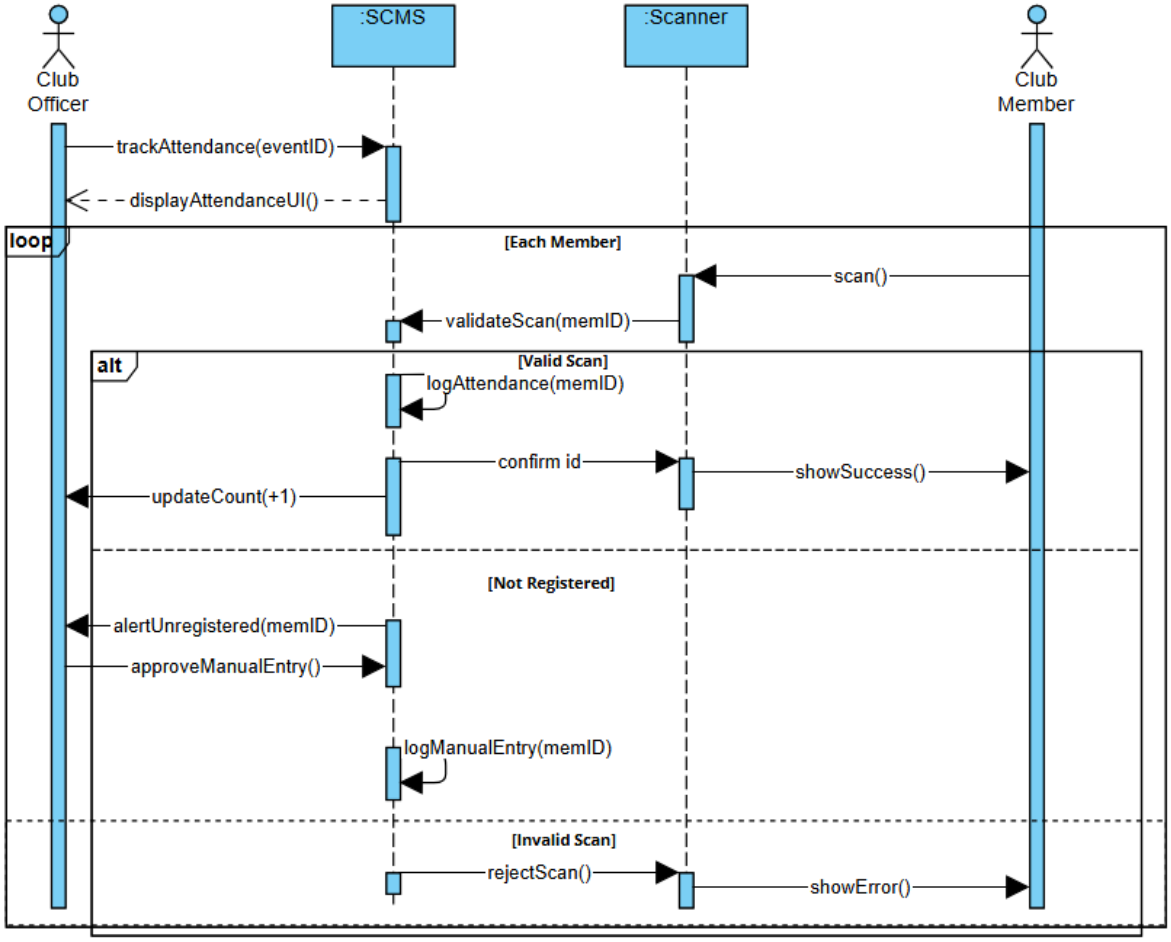
3.1.3.2.7 Use Case: View Budget Balance

Use Case Element	Description
Primary Actor	Club Officer
Secondary Actors	University Financial System
Purpose	To allow a club officer to assess the current balance of the club's budget in real time.
Preconditions	• User is logged in with Officer role. • Club is active and registered.
Postconditions	• Budget balance is displayed on screen.
Trigger	User clicks "View Budget Balance."
Main Flow	1. Officer clicks on "View Budget Balance". 2. System sends a request to University Financial System. 3. System receives and displays the current budget balance.
Alternative Flow – Data Unavailable	• University Financial System is down or unreachable. • System displays error message and retry option.
Exception Flows	• Session timeout, user is redirected to login. • Internet connection lost.



3.1.3.2.8 Use Case: Track Attendance

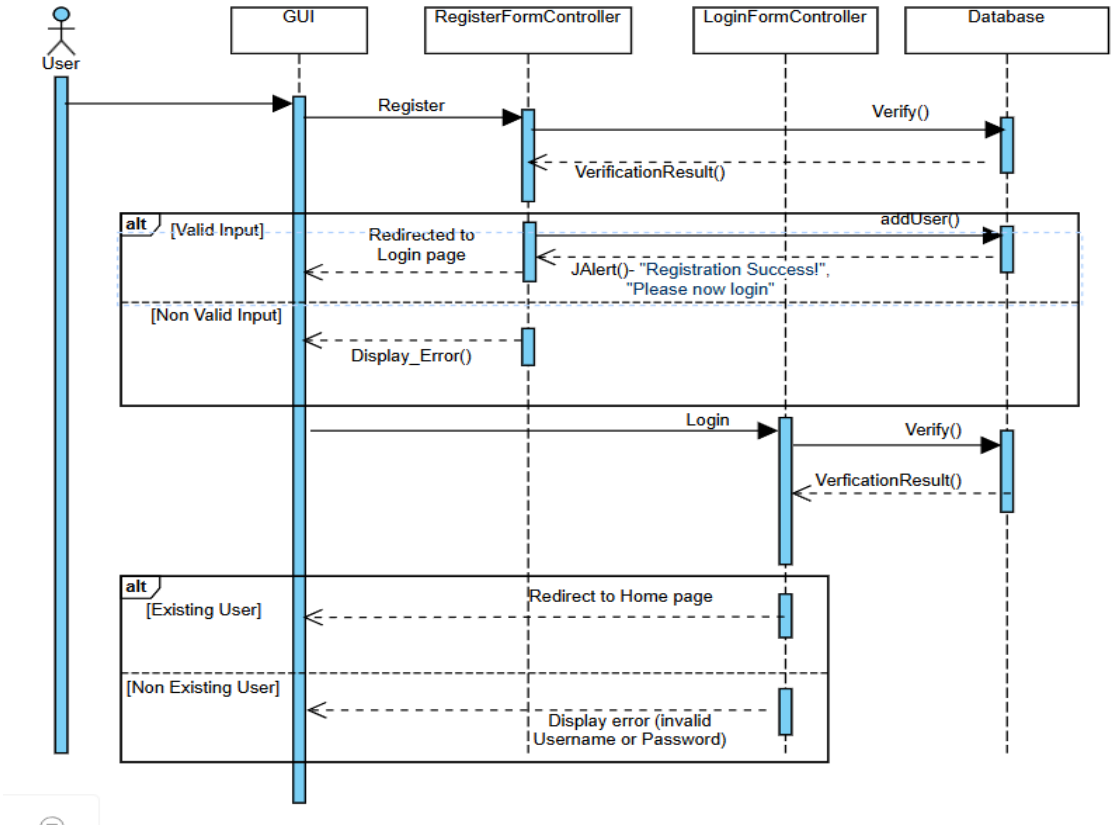
Use Case Element	Description
Primary Actor	Club Officer
Secondary Actors	Event Participants, QR/NFC Scanner Device
Purpose	To record the attendance of members at a club event using QR or NFC-based check-in.
Preconditions	• Event is active and scheduled. • QR/NFC check-in system is enabled and operational.
Postconditions	• Attendance records are saved. • Members marked as present.
Trigger	Club officer clicks on "Track Attendance".
Main Flow	1. Officer opens the attendance tracking module. 2. Participants scan QR/NFC tag at the check-in point. 3. System logs user ID. 4. Attendance list updates in real time.
Alternative Flow – Invalid Scan	• User scans an invalid QR/NFC tag. • System displays an error and prompts to retry.
Alternative Flow – Member Not Registered for Event	• System checks event RSVP list. • If not registered, prompts officer for manual override.


Club Officer
:SCMS
:Scanner
Club Member
trackAttendance(eventID)
displayAttendanceUI()
loop
[Each Member]
scan()
validateScan(memID)
alt
[Valid Scan]
logAttendance(memID)
confirm id
showSuccess()
updateCount(+1)
[Not Registered]
alertUnregistered(memID)
approveManualEntry()
logManualEntry(memID)
[Invalid Scan]
rejectScan()
showError()

3.1.3.3 Student Affairs Officer

3.1.3.3.1 Use Case Specification: Login

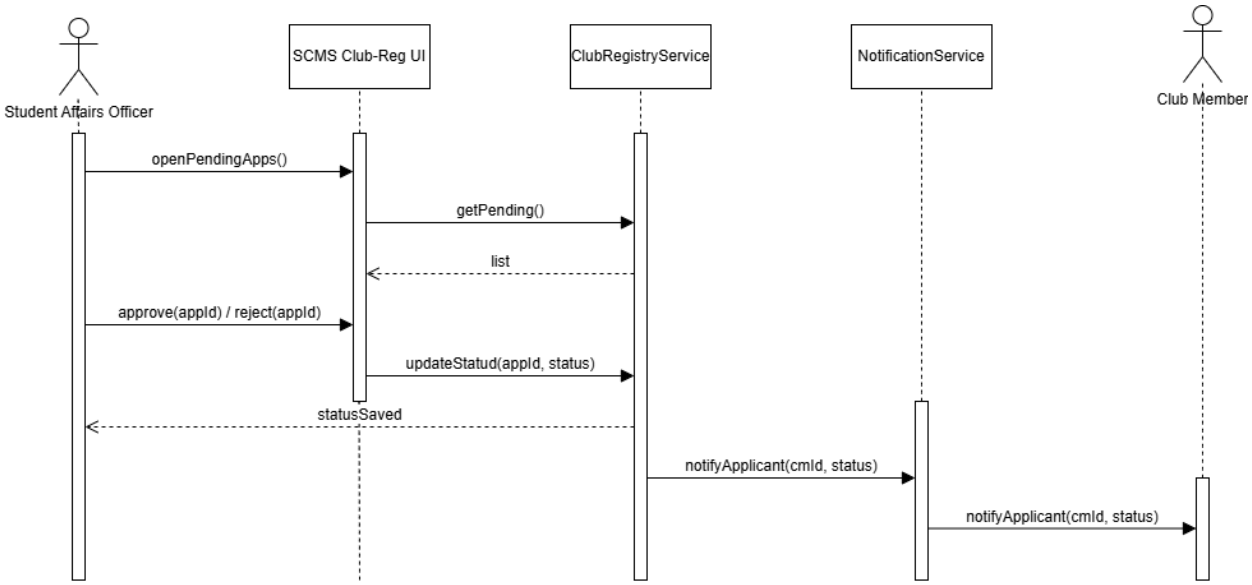
Use Case Element	Description
Primary Actor	Student Affairs Officer.
Purpose	Allow existing officers to authenticate and access their club account
Preconditions	• Officer has previously registered account • Officer knows their login credentials (email/username and password) • System authentication service is operational
Postconditions	• Officer is authenticated and logged into the system • Session is created and maintained • Member has access to authorized club features • Login activity is recorded for security purposes
Trigger	Officer wants to access their account
Main Flow	8. Officer navigates to login page 9. Officer enters email/username and password 10. System validates credentials against database 11. System verifies officer account status 12. System creates secure session token 13. System redirects member to dashboard or requested page 14. Officer gains access to personalized features
Alternative Flow – Additional Info Needed	• Invalid credentials: System displays error message and offers password reset • Account locked/suspended: System shows account status message • Forgot password: System provides password recovery option • Account not verified: System prompts to check email for verification link


User
GUI
RegisterFormController
LoginFormController
Database
Register
Verify()
VerificationResult()
alt
[Valid Input]
Redirected to
Login page
addUser()
JAlert()- "Registration Success!",
"Please now login"
[Non Valid Input]
Display_Error()
Login
Verify()
VerficationResult()
alt
[Existing User]
Redirect to Home page
[Non Existing User]
Display error (invalid
Username or Password)

3.1.3.3.2 Use Case Specification: Process Club Registration

Use Case Element	Description
Primary Actor	Student Affairs Officer
Secondary Actors	Student
Purpose	To review and decide on student-submitted club registration applications. The officer may approve or reject applicants.
Preconditions	• A student has submitted a complete club registration form. • The system has received all required documentation. • The Student Affairs Officer is logged in an authorized.
Postconditions	• The application is marked as **Approved** or **Rejected.** • The applicant is notified of the outcome. • If approved, the club is added to the official registry.
Trigger	A student submits a new club registration request.
Main Flow	1. Officer logs into SCMS and navigates to "Pending Club Registrations." 2. System displays a list of unprocessed applications. 3. Officer selects a submission and reviews the form, documents, and club name. **4.** Officer selects an action: **Approve** or **Reject.** 5. System updates the status accordingly. 6. System sends notification to the student with outcome.
Alternative Flow – Incomplete Submission	• Officer flags the application as "Incomplete." • System notifies the student with a message to revise and resubmit.
Alternative Flow – Application Rejected	• Officer rejects the request and provides a reason. • System notifies the student and archives the application.
Alternative Flow – Application Approves	• Officer approves the application. • System updates status to **Active**.

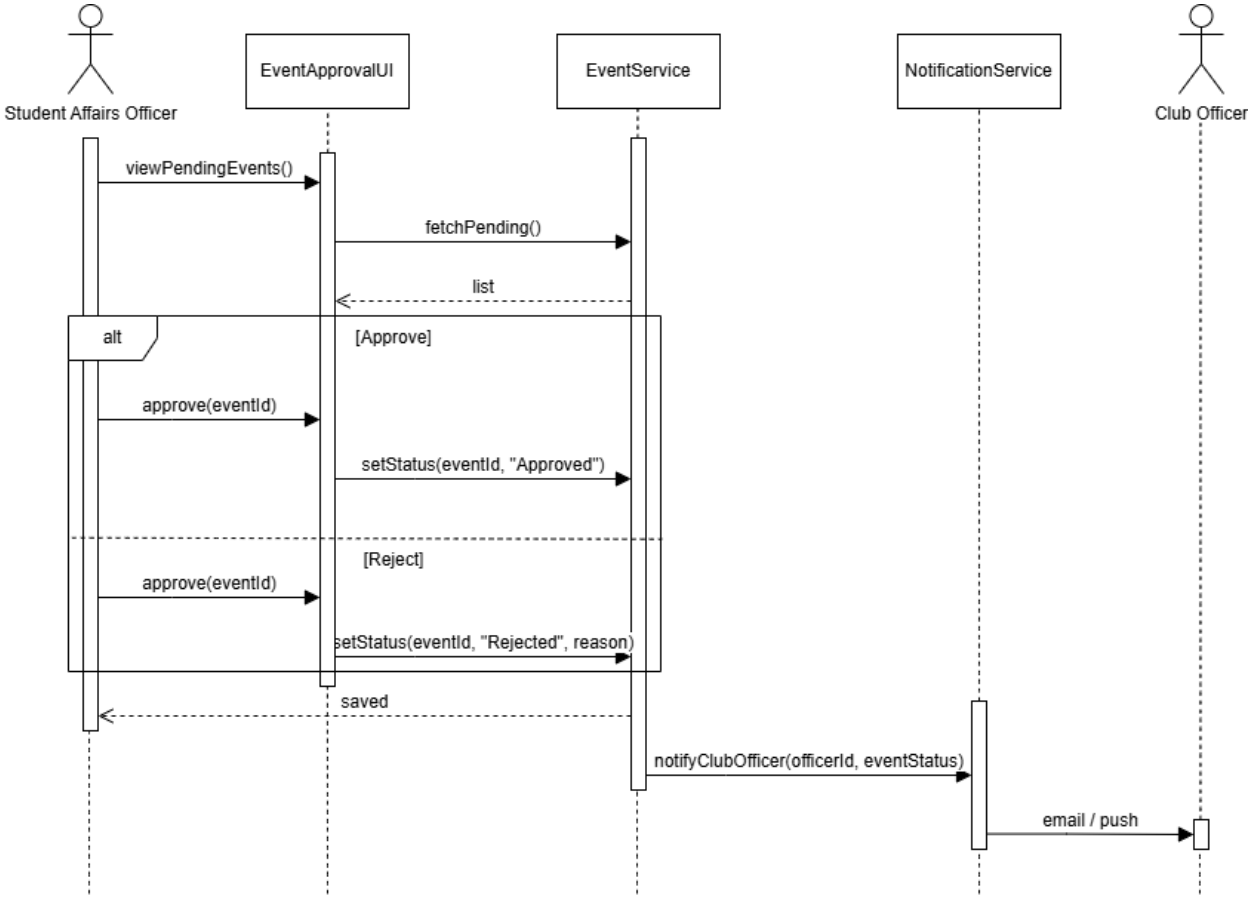
	• Club is added to the directory and visible to students.
Exception Flow	• System session times out: user is redirected to login. • Internet connection is lost: submission is held until reconnect. • Required fields are missing: system highlights errors and blocks action.



3.1.3.3.3 Use Case Specification: Process Event Proposal

Use Case Element	**Description**
Primary Actor	Student Affairs Officer
Secondary Actors	Club Officer
Purpose	To evaluate submitted club event proposals and either approve or reject those proposals.
Preconditions	• The event proposal is submitted by the Club Officer. • All required event details and supporting documents are present. • Officer is logged in and has access to pending proposals.
Postconditions	• The event is marked as **Approved** or **Rejected.** • Club Officer is notified of the outcome. • Approved events proceed to the calendar and dashboard
Trigger	A new event proposal is submitted and assigned to the Student Affairs Officer
Main Flow	1. Officer logs into system and selects "Pending Event Proposals." 2. System displays list of submitted proposals. 3. Officer selects a proposal and reviews event name, date, budget, and risk assessment details. 4. Officer selects an action: **Approve** or **Reject**. 5. System updates event status. 6. Notification is sent to the submitting Club Officer.
Alternative Flow – Proposal Rejected	• Officer rejects the event. • Provides reason • System updates status and notifies officer.
Alternative Flow – Proposal Approved	• Officer approves the proposal. • Event is added to the official schedule/calendar. • System triggers notifications to relevant users.
Exception Flow	• Session timeout: officer is logged out.

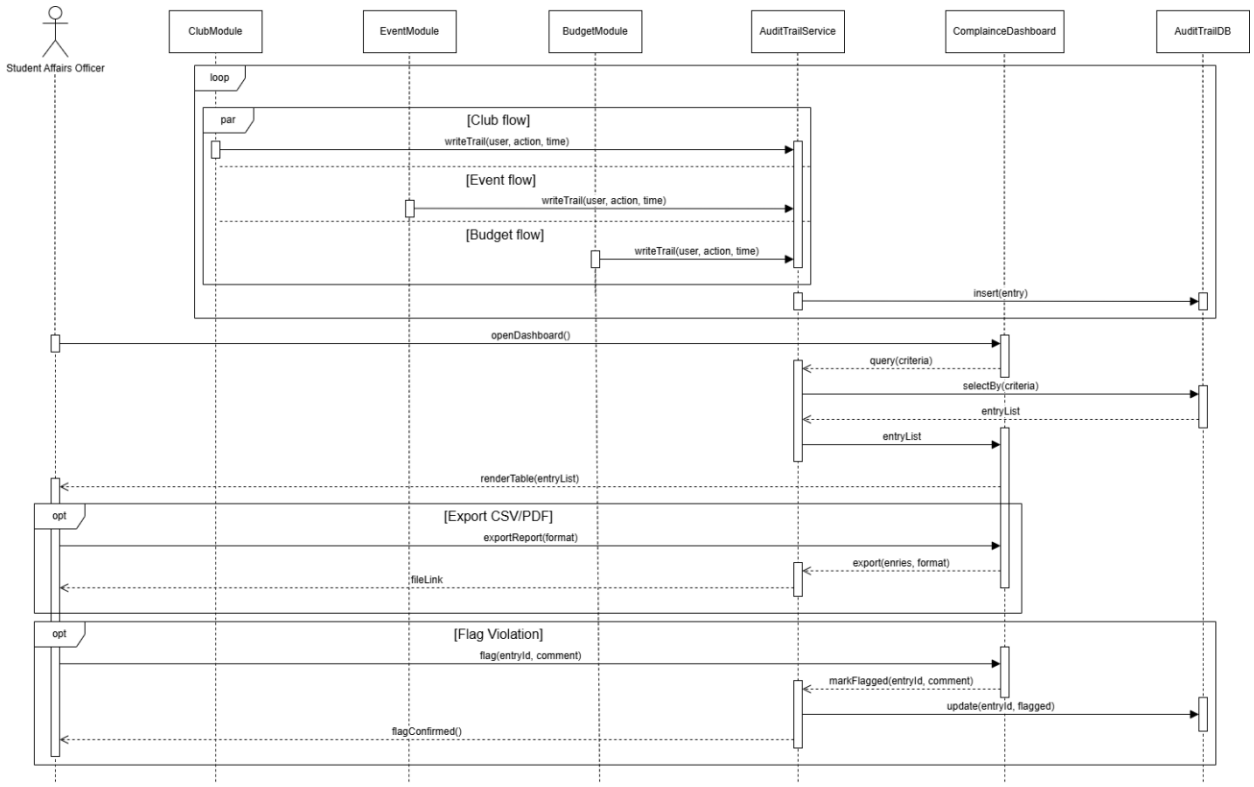
	• Missing fields: system block submission and highlight missing items. • Network failure: system saves progress and retries when back online.



3.1.3.3.4 Use Case Specification: Monitor Compliance

Use Case Element	Description
Primary Actor	Student Affairs Officer
Secondary Actors	Club Officer
Purpose	To monitor ongoing club activities, budgets, and events to ensure compliance with university policy and regulations.
Preconditions	• Club is active and registered. • Events and budget records exist in the system. • Officer is logged in with appropriate permissions.
Postconditions	• Any compliance issues are flagged and recorded. • Clubs are notified of required corrections (if any). • Compliance logs are updated.
Trigger	The Student Affairs Officer performs a scheduled or triggered audit of club activities.
Main Flow	1. Officer accesses the "Compliance Monitoring" dashboard in SCMS. 2. System displays all clubs with filters (date range, budget, events). 3. Officer reviews recent event history, financial transactions, and role assignments. 4. If all is valid, system logs the review as "Compliant." 5. If discrepancies are found, officer flags the club for follow-up. 6. Notifications are sent to the Club Officer(s).
Alternative Flow – Compliance Issues Found	• Officer flags non-compliance (e.g., missing reports, overspending). • System generates warning notice and sends it to club admin. • Follow-up deadline is recorded.
Exception Flow	• System logs are unavailable: displays error and prompts retry.

	• Officer lacks access: system shows permission error. • Network issues delay report loading: system caches results for later viewing.

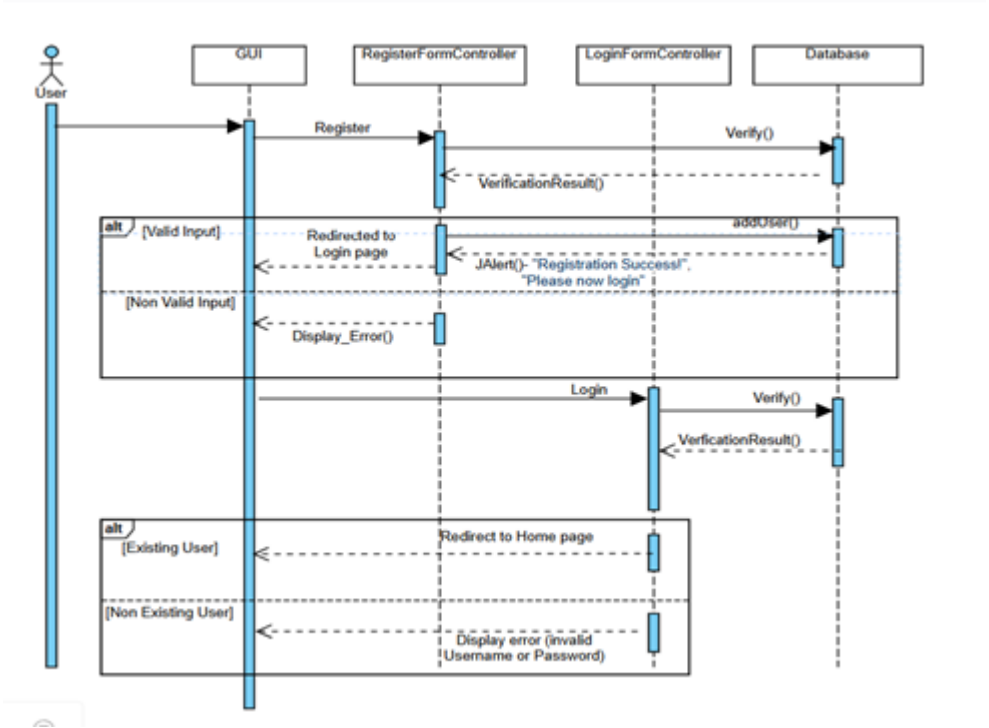


3.1.3.4 Finance Officer

3.1.3.4.1 Use Case Specification: Login

Use Case Element	Description
Primary Actor	Finance Officer.
Purpose	Allow existing officers to authenticate and access their club account
Preconditions	• Officer has previously registered account • Officer knows their login credentials (email/username and password) • System authentication service is operational
Postconditions	• Officer is authenticated and logged into the system • Session is created and maintained • Member has access to authorized club features • Login activity is recorded for security purposes
Trigger	Officer wants to access their account
Main Flow	1. Officer navigates to login page 2. Officer enters email/username and password 3. System validates credentials against database 4. System verifies member account status 5. System creates secure session token 6. System redirects member to dashboard or requested page 7. Officer gains access to personalized features

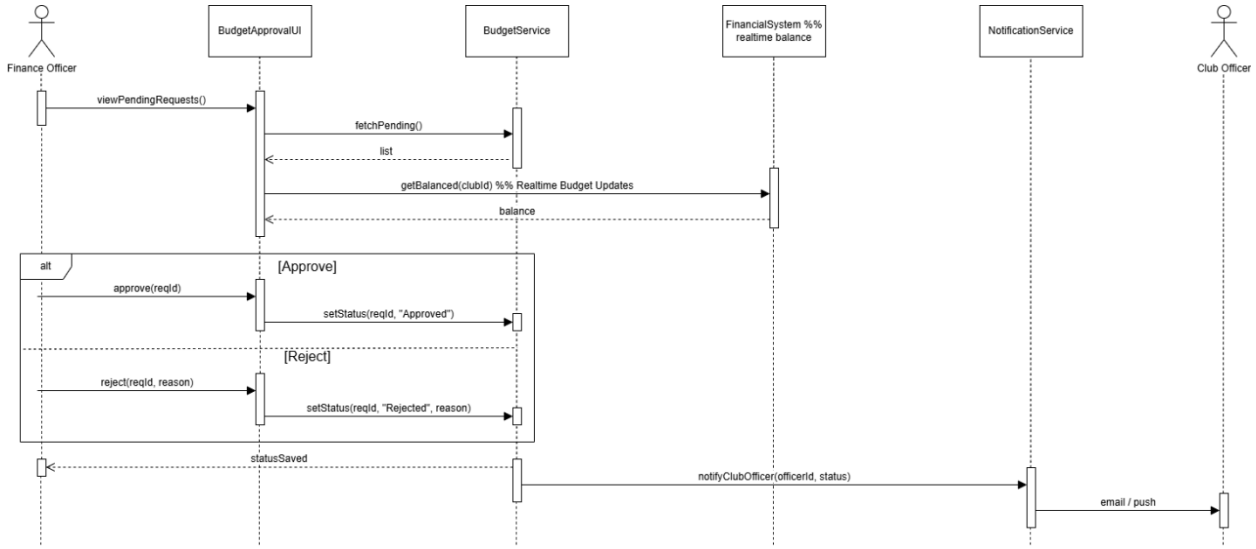
Alternative Flow – Additional Info Needed	• Invalid credentials: System displays error message and offers password reset • Account locked/suspended: System shows account status message • Forgot password: System provides password recovery option • Account not verified: System prompts to check email for verification link



3.1.3.4.2 Use Case Specification: Process Budget Request

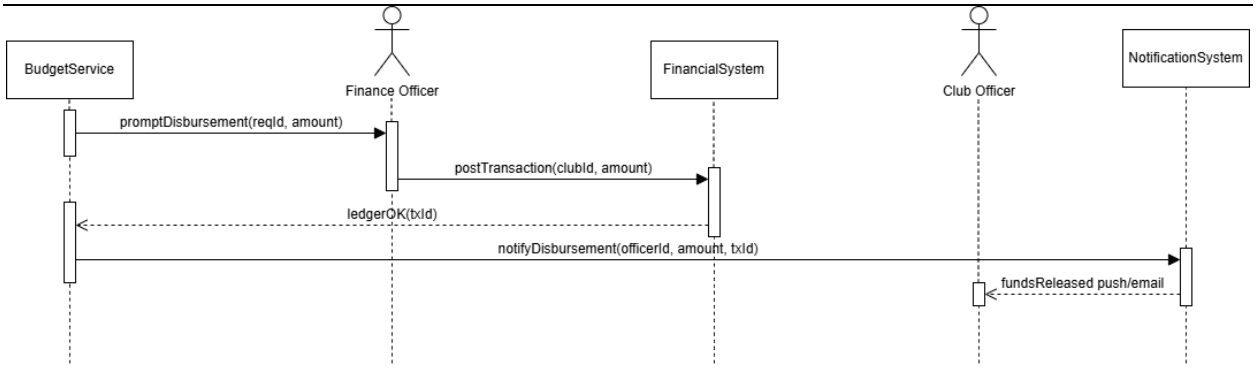
Use Case Element	Description
Primary Actor	Finance Officer
Secondary Actors	Club Officer, University Financial System
Purpose	To evaluate and make decisions on budget requests submitted by clubs for events or operations.
Preconditions	• Budget request is submitted. • Finance Officer is logged in. • University Financial System is operational.
Postconditions	• Budget request is either **Approved** or **Rejected** • Request status is updated. • The Club Officer is notified of the decision.
Trigger	A new budget request is submitted by a Club Officer.
Main Flow	1. Officer logs into SCMS and views "Pending Budget Requests." 2. Selects a request to review. 3. Verifies requested amount, justification, and supporting docs. 4. Chooses one of the following actions: a. Approve – forwards to financial system for recording. b. Reject – enters reason for rejection. 5. System updates status and sends notification.
Alternative Flow – Budget Exceeds Limits	• Officer rejects request due to budget limit violations. • System logs justification and notifies club.
Alternative Flow – Request Approved	• Officer approves request. • System logs approval and forwards to University Financial System for transaction posting.

Exception Flow	• Financial system offline: request is queued for retry. • Session timeout: officer is logged out. • Data conflict or duplication: system warns and halts submission.



3.1.3.4.3 Use Case Specification: Disburse Funds

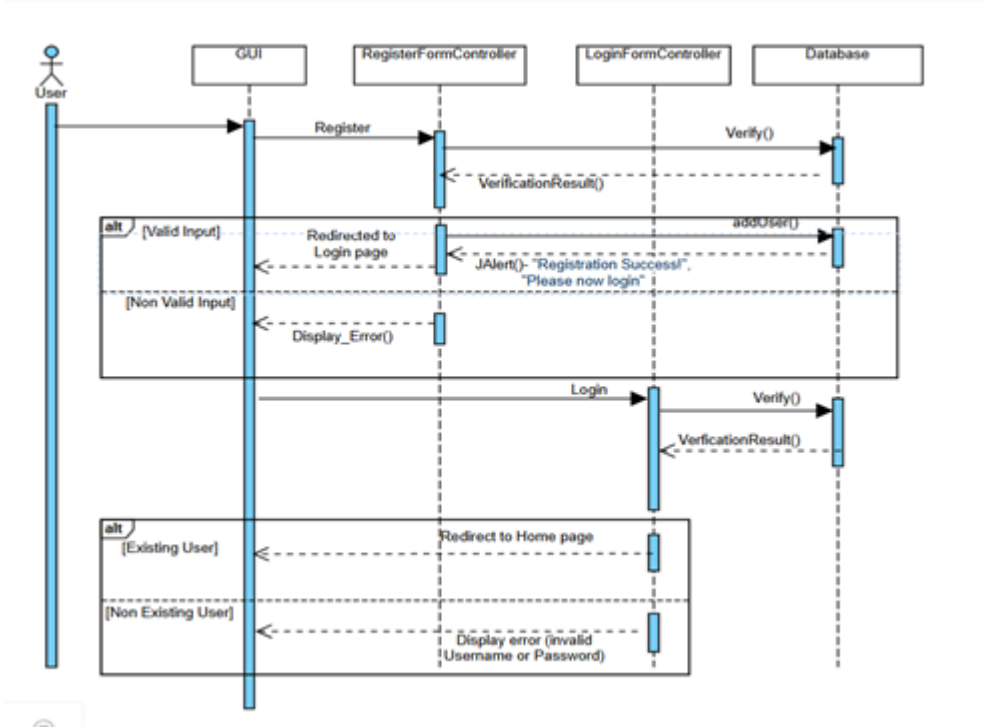
Use Case Element	Description
Primary Actor	Finance Officer
Secondary Actors	University Financial System, Club Officer
Purpose	To transfer approved budget amounts to the club's virtual account or ledger and log the disbursement in the system.
Preconditions	• Budget request is approved. • Finance Officer is logged in. • University Financial System is operational.
Postconditions	• Funds are successfully disbursed. • Transaction is recorded in SCMS and financial system. • Club Officer is notified of successful transfer.
Trigger	A budget request is approved and ready for disbursement.
Main Flow	1. Officer accesses "Approved Budget Requests" in SCMS. 2. Selects request marked for disbursement. 3. Verifies account info and budget amount. 4. Confirms fund release. 5. System updates request status to "Disbursed." 6. Financial transaction is recorded via University Financial System. 7. Notification is sent to the Club Officer
Exception Flow	• University Financial System is down: SCMS queues transaction for retry. • Data mismatch or invalid amount: system aborts and logs error. • Session timeout or logout: officer must re-authenticate.


BudgetService
Finance Officer
FinancialSystem
Club Officer
NotificationSystem
promptDisbursement(reqId, amount)
postTransaction(clubId, amount)
ledgerOK(txId)
notifyDisbursement(officerId, amount, txId)
fundsReleased push/email

3.1.3.5 Venue Manager

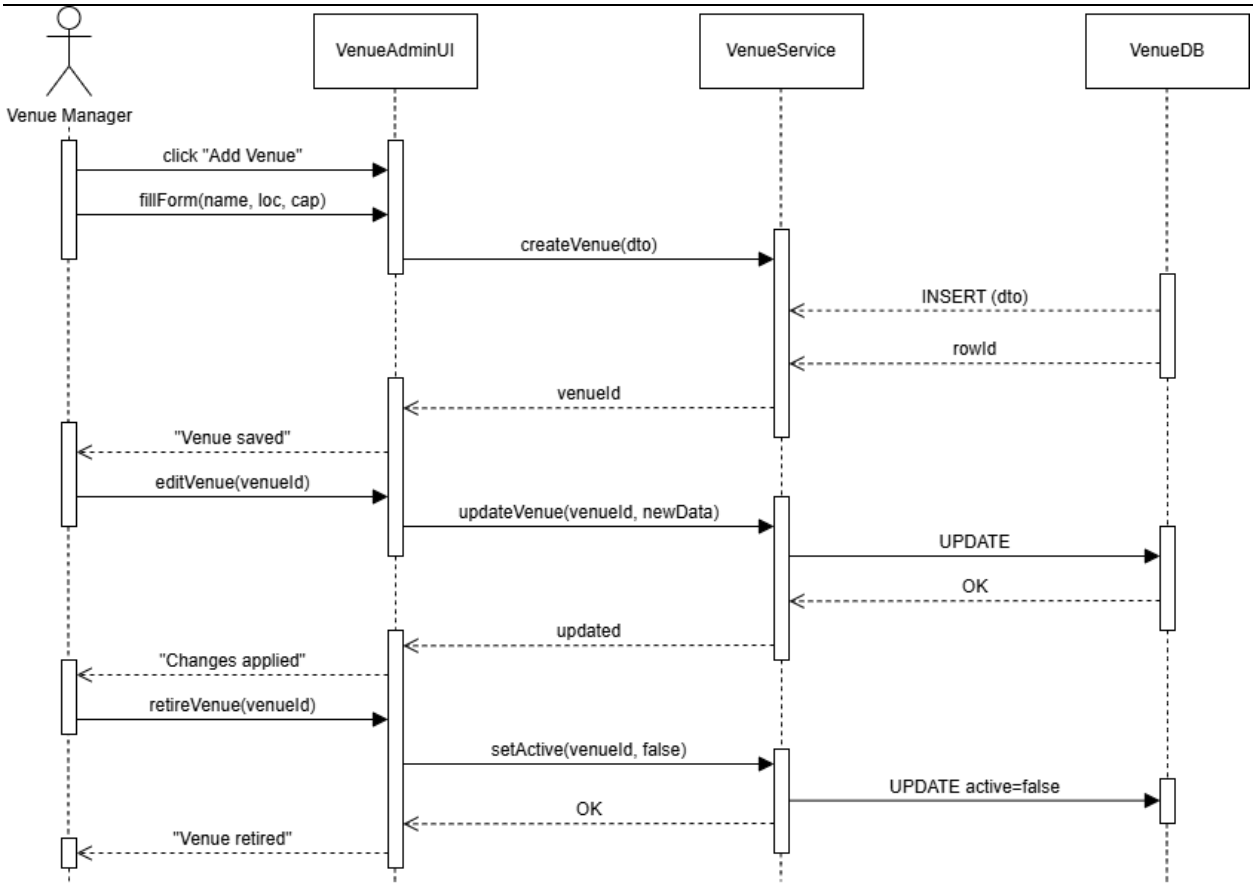
Use Case Element	Description
Primary Actor	Venue Manager.
Purpose	Allow venue Manager to authenticate and access their club account
Preconditions	• Manager has previously registered account • Manager knows their login credentials (email/username and password) • System authentication service is operational
Postconditions	• Manager is authenticated and logged into the system • Session is created and maintained • Manager has access to authorized features • Login activity is recorded for security purposes
Trigger	Manager wants to access their account
Main Flow	1. Manager navigates to login page 2. Manager enters email/username and password 3. System validates credentials against database 4. System verifies manager account status 5. System creates secure session token 6. System redirects member to dashboard or requested page 7. Manager gains access to personalized page

Alternative Flow – Additional Info Needed	• Invalid credentials: System displays error message and offers password reset • Account locked/suspended: System shows account status message • Forgot password: System provides password recovery option • Account not verified: System prompts to check email for verification link



3.1.3.5.2 Use Case Specification: Maintain Venue Catalogue

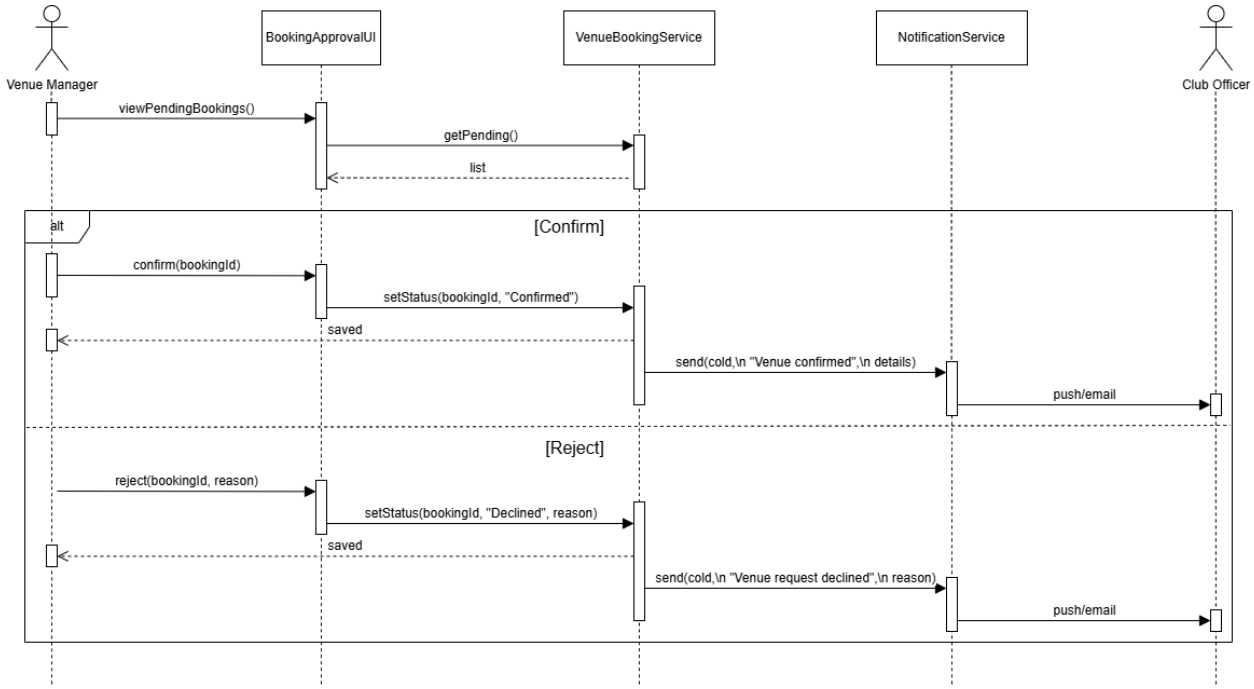
Use Case Element	Description
Primary Actor	Venue Manager
Purpose	To add, update, or deactivate venues in the system's venue catalogue for event booking and scheduling purposes.
Preconditions	• Venue Manager is logged in with proper permissions. • The venue catalogue module is accessible and active.
Postconditions	• Venue data is added, edited, or deactivated. • SCMS updates venue list for all users. • Changes are logged for auditing.
Trigger	Venue Manager performs an update via the SCMS admin panel.
Main Flow	1. Venue Manager navigates to "Venue Catalogue." 2. Views current list of active venues. 3. Chooses one of the following actions: a. **Add** new venue (name, location, capacity, equipment). b. **Edit** details of existing venue. c. **Deactivate** a venue that is no longer available. 4. System validates input and saves changes. 5. Changes become visible to club officers booking events.
Alternative Flow – Invalid Venue Data	• Manager inputs missing or invalid values. • System highlights errors and block submission.
Exception Flow	• Connection lost: system prompts to retry later. • Session timeout: user is logged out. • Data conflict: changes are rejected if venue is currently reserved.


Venue Manager
VenueAdminUI
VenueService
VenueDB
click "Add Venue"
fillForm(name, loc, cap)
createVenue(dto)
INSERT (dto)
rowId
venueId
"Venue saved"
editVenue(venueId)
updateVenue(venueId, newData)
UPDATE
OK
updated
"Changes applied"
retireVenue(venueId)
setActive(venueId, false)
UPDATE active=false
OK
"Venue retired"

3.1.3.5.3 Use Case Specification: Process Booking

Use Case Element	**Description**
Primary Actor	Venue Manager
Secondary Actors	Club Officer
Purpose	To approve or reject venue booking requests submitted by club officers based on availability, rules, and conflict checks.
Preconditions	• Booking request has been submitted through SCMS. • Venue is registered in the system. • Venue Manager is logged in with the appropriate role.
Postconditions	• Booking is either **Confirmed** or **Rejected**. • Club Officer is notified of the decision. • Venue status is updated to reflect the booking.
Trigger	A new venue booking request is received and pending action.
Main Flow	1. Venue Manager accesses "Pending Bookings." 2. Selects a request and reviews event info, venue, and timing. 3. Checks venue availability and rules (e.g., capacity, conflicts). 4. Selects action: a. **Approve** – marks venue as reserved. b. **Reject** – provides reason. 5. System updates booking status and notifies Club Officer.
Alternative Flow – Venue Conflict Detected	• System detects a clash with another booking. • Venue Manager rejects request and suggests alternative dates/times.
Alternative Flow – Venue Blocked or Under Maintenance	• The venue is temporarily blocked. • System prevents approval
Exception Flow	• Venue no longer exists: system flags error. • Session times out or user logs out: action cancelled.

	• Booking ID mismatch: system aborts and logs error.

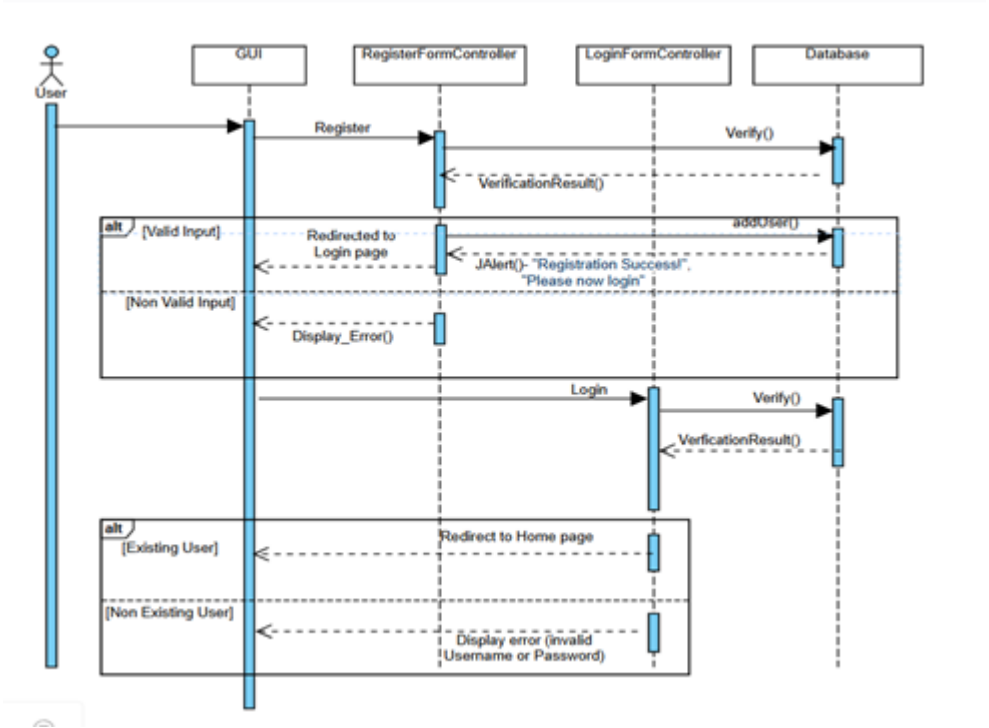


3.1.3.6 System Administrator

3.1.3.6.1 Use Case Specification: Login

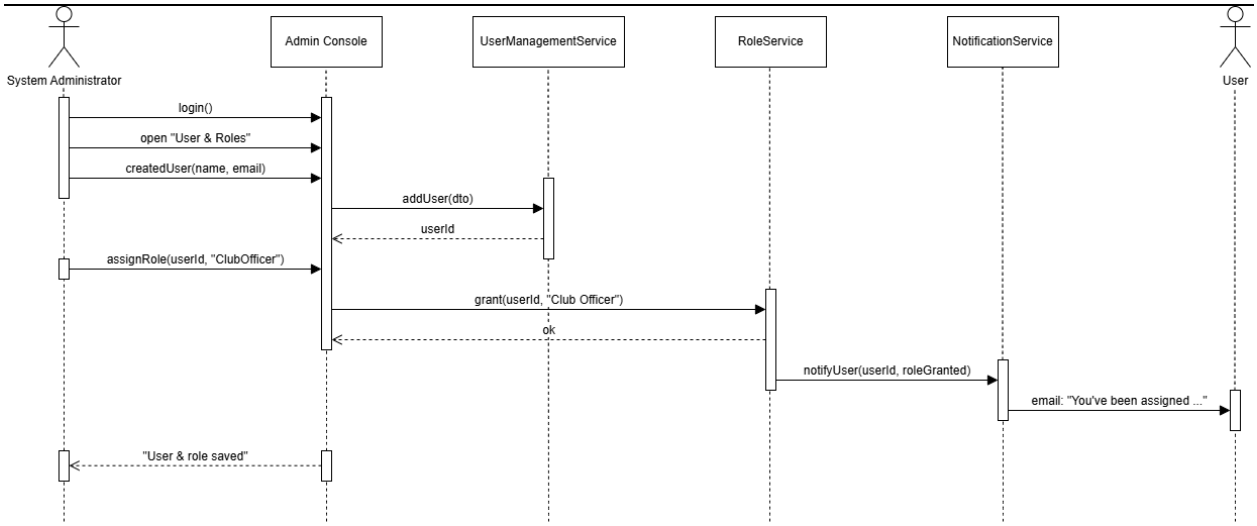
Use Case Element	Description
Primary Actor	System Administrator.
Purpose	Allow the System Administrator to authenticate and access their club account
Preconditions	• Administrator has previously registered account • Administrator knows their login credentials (email/username and password) • System authentication service is operational
Postconditions	• Administrator is authenticated and logged into the system • Session is created and maintained • Administrator has access to authorized club features • Login activity is recorded for security purposes
Trigger	Administrator wants to access their account
Main Flow	1. Administrator navigates to login page 2. Administrator enters email/username and password 3. System validates credentials against database 4. System verifies Administrator account status 5. System creates secure session token 6. System redirects member to dashboard or requested page 7. Administrator gains access to personalized page

Alternative Flow – Additional Info Needed	• Invalid credentials: System displays error message and offers password reset • Account locked/suspended: System shows account status message • Forgot password: System provides password recovery option • Account not verified: System prompts to check email for verification link



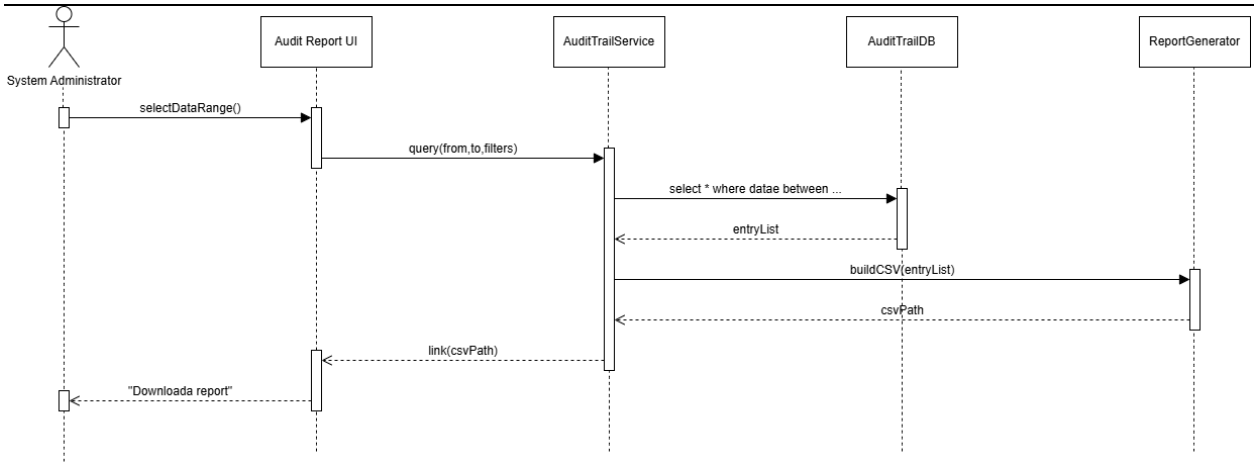
3.1.3.6.2 Use Case Specification: Manage Users

Use Case Element	Description
Primary Actor	System Administrator
Purpose	To manage user accounts and access privileges for all roles in the system.
Preconditions	• System Administrator is authenticated and authorized. • User management module is operational.
Postconditions	• User accounts are created, updated, suspended, or removed. • Role-based permissions are assigned or modified. • Logs are updated for audit tracking.
Trigger	Admin logs in and selects "User Management" from the admin dashboard.
Main Flow	1. Admin logs into SCMS and navigates to the User Management module. 2. System displays a list of all registered users, their roles, and statuses. 3. Admin can select edit user to update their role. 4. System validates input and applies changes to the database. 5. System confirms the action with a success message. 6. Audit Trail is updated to log the change.
Alternative Flow – Invalid Role or Input	• Admin selects invalid role or leaves required field blank. • System blocks action and shows validation error.
Alternative Flow – Attempt to Delete Admin	• System prevents deletion of root/admin account. • Displays warning message.
Exception Flow	• Session timeout: system logs admin out. • Database error: system displays failure and retries. • Permissions error: action restricted based on admin level.


System Administrator
Admin Console
UserManagementService
RoleService
NotificationService
User
login()
open "User & Roles"
createdUser(name, email)
addUser(dto)
userId
assignRole(userId, "ClubOfficer")
grant(userId, "Club Officer")
ok
notifyUser(userId, roleGranted)
email: "You've been assigned ..."
"User & role saved"

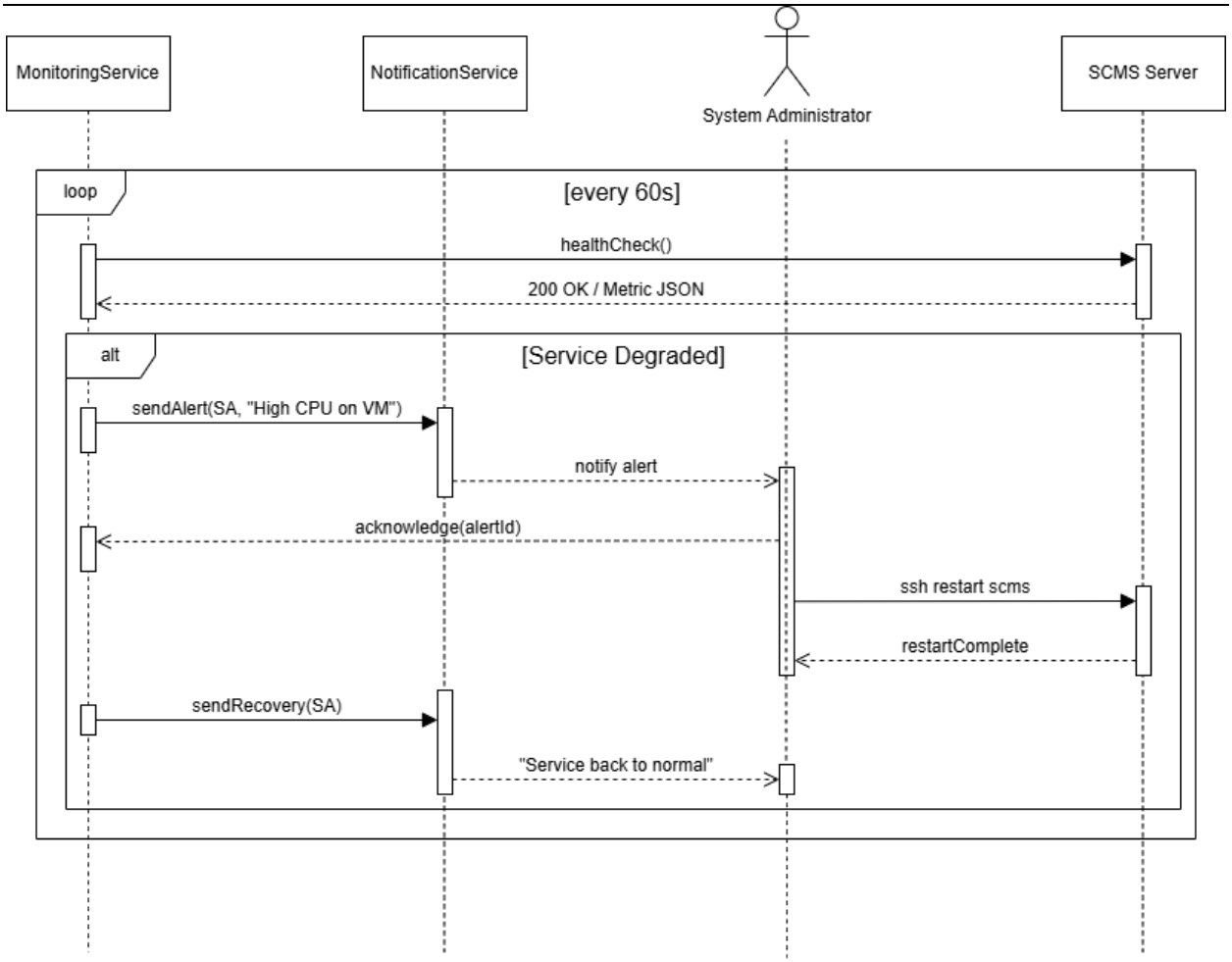
3.1.3.6.3 Use Case Specification: Generate Audit-Trail Report

Use Case Element	Description
Primary Actor	System Administrator
Purpose	To generate a report of system activity logs including user actions, approval, logins, changes to settings, and other auditable events.
Preconditions	• Admin is logged in with sufficient privileges. • Audit trail feature is enabled and has recorded data.
Postconditions	• Audit report is generated viewable. • Option to export the report. • System logs the reporting activity.
Trigger	Admin chooses to generate and audit trail from the admin panel.
Main Flow	1. Admin opens the Audit Trail module. 2. Selects date range, user role, or event filters. 3. Clicks "Generate Report." 4. System retrieves relevant logs from database. 5. Report is displayed in tabular format. 6. Admin can export or download the report.
Alternative Flow – Export Failed	• Export attempt fails due to format issue. • System displays error and retries option.
Exception Flow	• Log file corruption or missing data: system logs internal error and shows message. • Network timeout: system retries or aborts process. • Permission mismatch: admin is denied access if rights are insufficient.


System Administrator
Audit Report UI
AuditTrailService
AuditTrailDB
ReportGenerator
selectDataRange()
query(from,to,filters)
select * where datae between ...
entryList
buildCSV(entryList)
csvPath
link(csvPath)
"Downloada report"

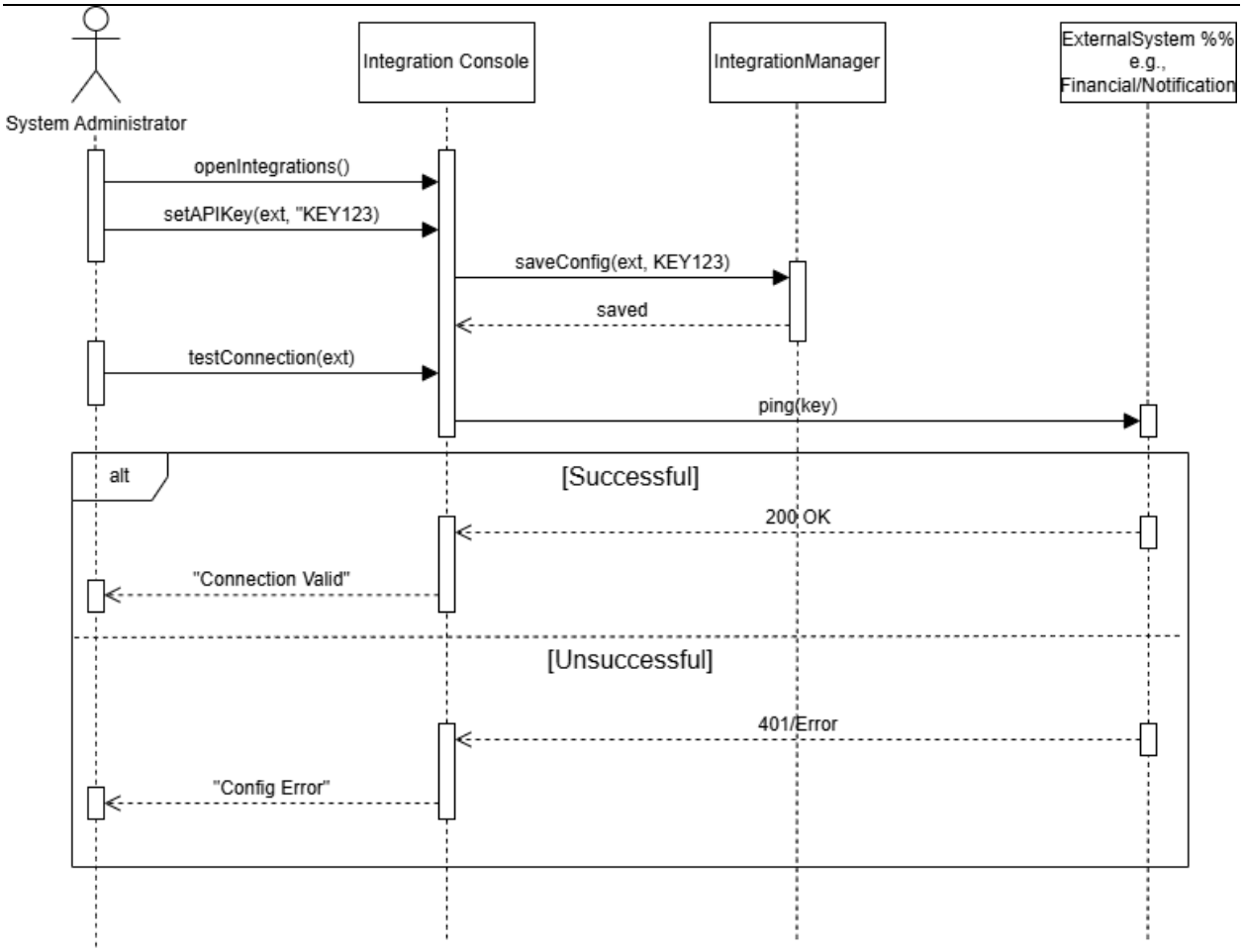
3.1.3.6.4 Use Case Specification: Monitor System Health

Use Case Element	Description
Primary Actor	System Administrator
Purpose	To track real-time system performance, uptime, resource usage, and detect errors or downtimes.
Preconditions	• Admin is logged in with system monitoring access. • Monitoring module is active and properly configured.
Postconditions	• Health status is updated. • Warnings or errors are flagged and logged. • Admin receives alerts for critical issues.
Trigger	Admin accesses "System Health" panel or a system event triggers auto-monitoring.
Main Flow	1. Admin opens the monitoring dashboard. 2. System loads metrics: CPU usage, memory, request rates, uptime. 3. Dashboard displays color-coded status indicators (e.g., healthy, warning, critical). 4. Admin reviews logs and performance history. 5. System auto-refreshes metrics every few seconds/minutes. 6. Admin may download logs or trigger diagnostics.
Exception Flow	• Connection lost to monitoring backend. • Admin receives fallback alert via Notification Service. - Session timeout or unauthorized access triggers logout or restriction message.


MonitoringService
NotificationService
System Administrator
SCMS Server
loop
[every 60s]
healthCheck()
200 OK / Metric JSON
alt
[Service Degraded]
sendAlert(SA, "High CPU on VM")
notify alert
acknowledge(alertId)
ssh restart scms
restartComplete
sendRecovery(SA)
"Service back to normal"

3.1.3.6.5 Use Case Specification: Configure Integrations

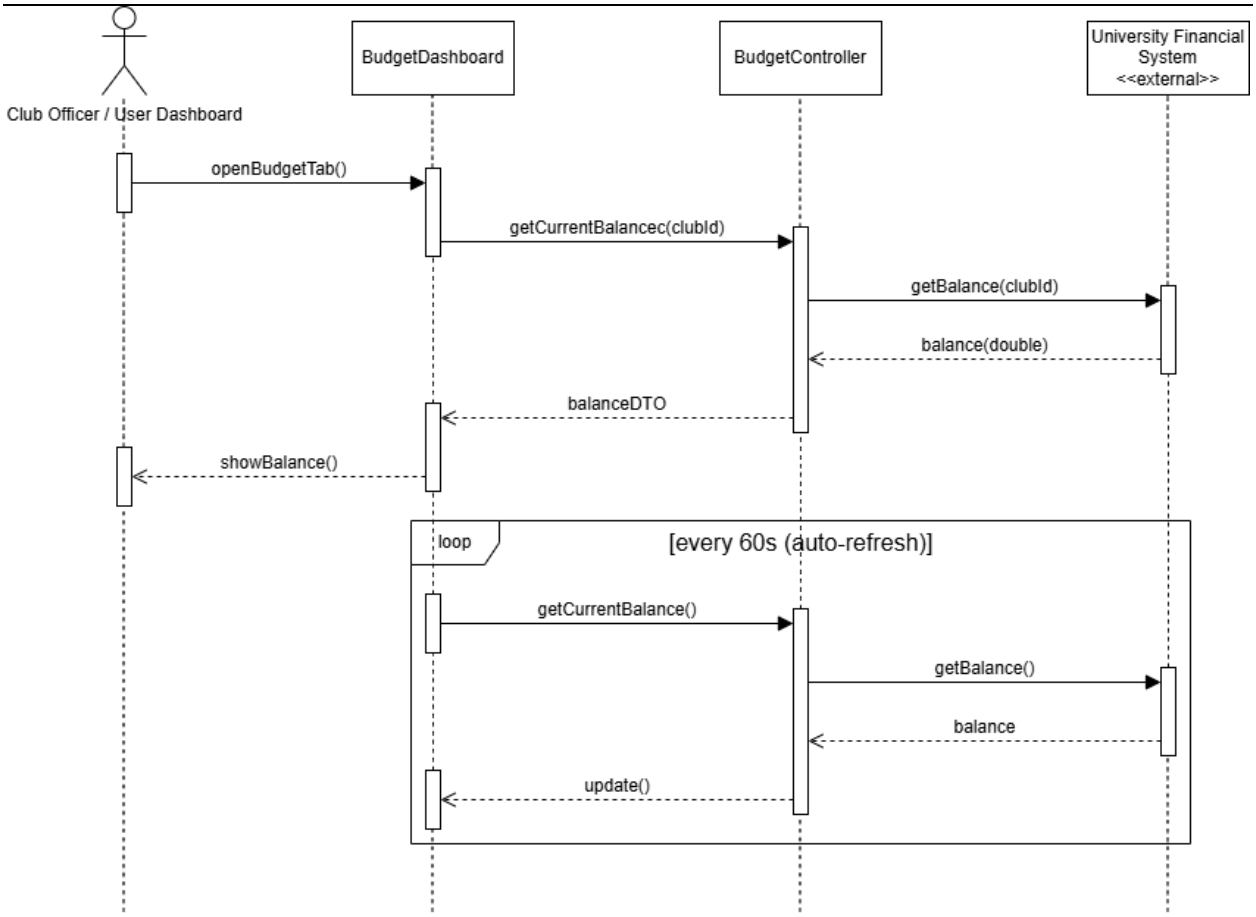
Use Case Element	Description
Primary Actor	System Administrator
Secondary Actors	External System
Purpose	To connect, configure, and manage the external services that the SCMS relies on.
Preconditions	• Admin is logged in with integration privileges. • External systems are reachable and have valid API credentials or connection settings.
Postconditions	• Integration settings are saved and active. • SCMS is successfully communicating with external systems. • Logs are updated to reflect changes.
Trigger	Admin initiates integration setup or modifies existing configuration.
Main Flow	1. Admin opens the "Integrations" panel in SCMS. 2. Selects the external system to configure (e.g., Venue Reservation, Financial System). 3. Enters required credentials, API endpoints, or tokens. 4. System validates input and attempts test connection. 5. If successful, the system enables integration and saves settings. 6. Admin sees confirmation and updated status indicators.
Alternative Flow – Test Connection Fails	• System displays connection error. • Admin is prompted to review credentials or check endpoint status/
Exception Flow	• External API is down: system logs failure and disables sync. • Unauthorized access attempt: system alerts and blocks change. • Session timeout: admin is logged out and redirected.


System Administrator
Integration Console
IntegrationManager
ExternalSystem %% e.g., Financial/Notification
openIntegrations()
setAPIKey(ext, "KEY123)
saveConfig(ext, KEY123)
saved
testConnection(ext)
ping(key)
alt
[Successful]
200 OK
"Connection Valid"
[Unsuccessful]
401/Error
"Config Error"

3.1.3.7 University Financial System

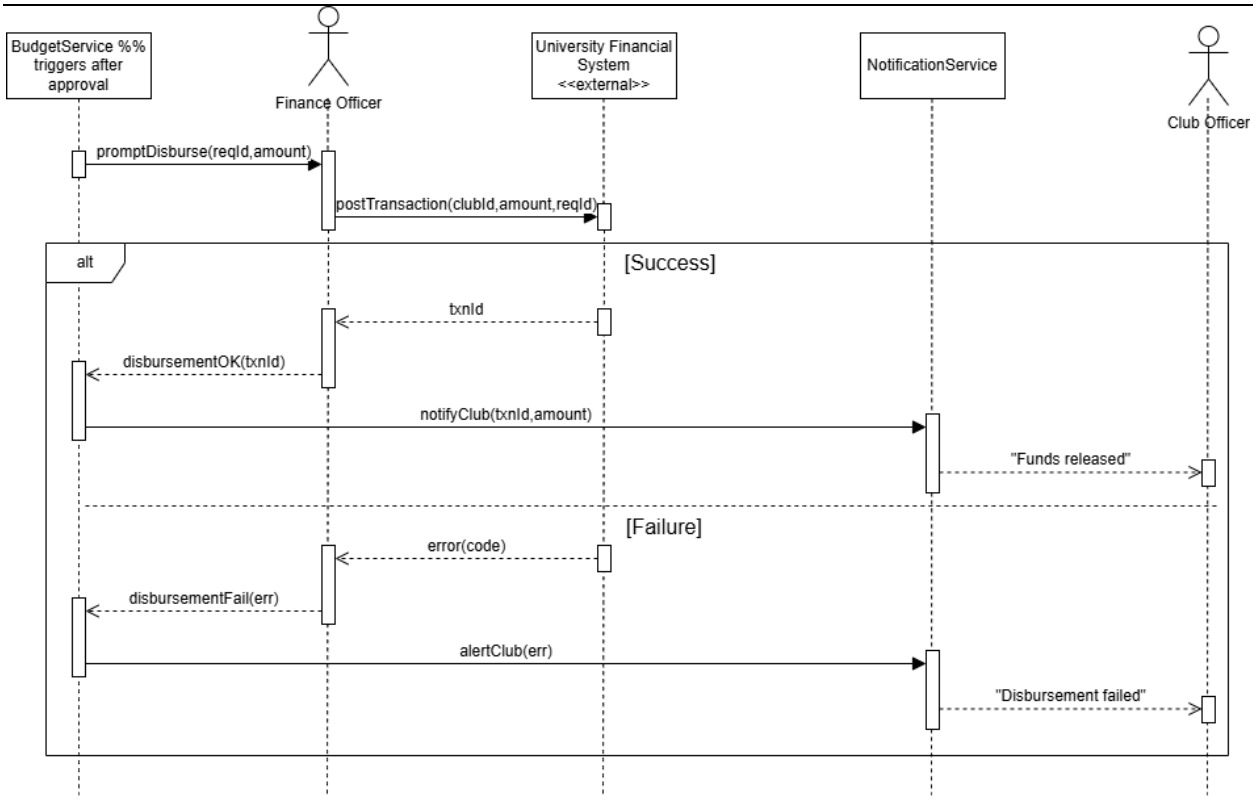
3.1.3.7.1 Use Case Specification: Provide Live Balance

Use Case Element	Description
Primary Actor	University Financial System
Purpose	To return the real-time balance of a specific club account upon request by the SCMS.
Preconditions	The club is registered and has a financial record. Financial System is online and accessible. A valid request is received from SCMS.
Postconditions	SCMS receives current budget balance. Balance is displayed in the user interface (e.g., for Club Officer or Finance Officer).
Trigger	SCMS sends a request to retrieve live balance for a specific club.
Main Flow	1. SCMS sends API request to the University Financial System with the club ID. 2. Financial system looks up the club's financial record. 3. System calculates and formats the current balance. 4. Balance is returned to SCMS. 5. SCMS displays the amount on screen.
Alternative Flow – Club Record Not Found	• Financial system returns "record not found" error. • SCMS displays error message to user.
Exception Flow	• Financial system is unreachable: SCMS shows "Unable to retrieve balance" with retry option. • Network timeout: request fails and is logged. • Invalid request format: system returns validation error.


Club Officer / User Dashboard
BudgetDashboard
BudgetController
University Financial System
<<external>>
openBudgetTab()
getCurrentBalancec(clubId)
getBalance(clubId)
balance(double)
balanceDTO
showBalance()
loop
[every 60s (auto-refresh)]
getCurrentBalance()
getBalance()
balance
update()

3.1.3.7.2 Use Case Specification: Record Approved Transactions

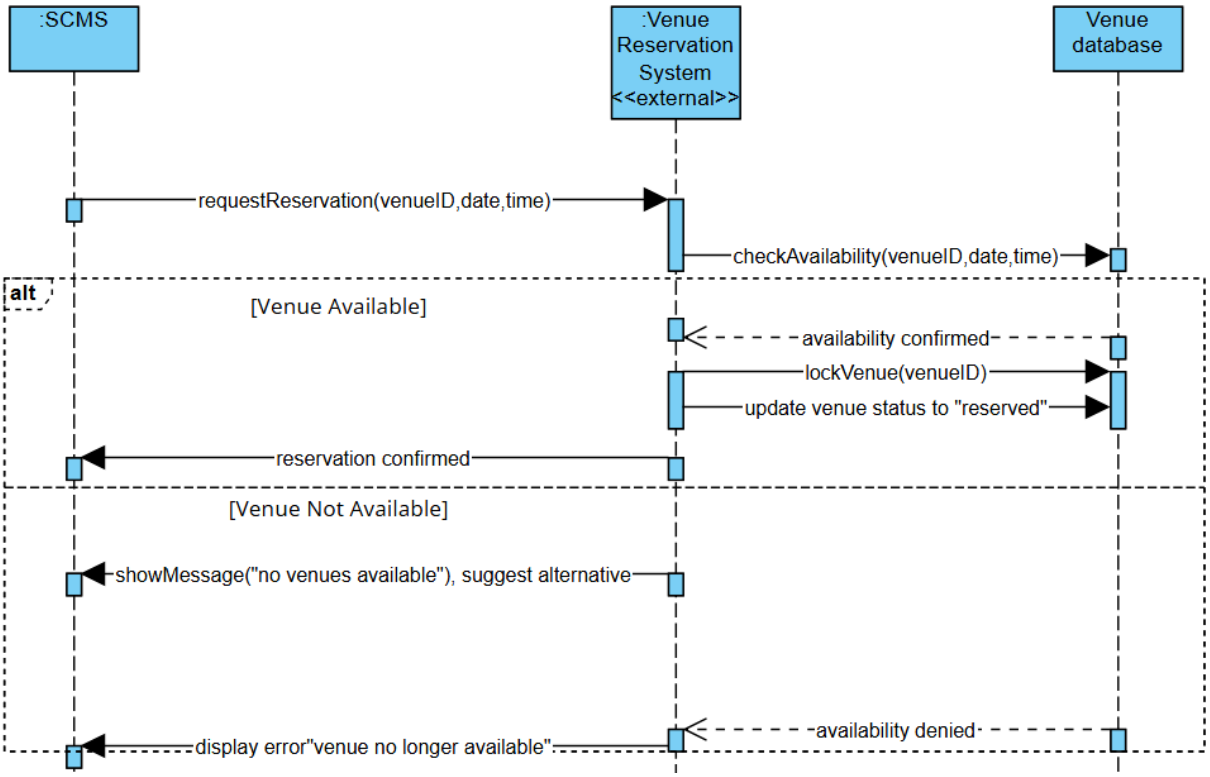
Use Case Element	Description
Primary Actor	University Financial System
Purpose	To record and confirm transactions that have been approved in SCMS, ensuring the club's ledger is up to date with disbursed amounts.
Preconditions	• Budget request is marked as "Approved." • SCMS sends transaction details in proper format. • University Financial System is operational.
Postconditions	• Transactions are recorded in the financial ledger. • Confirmation is sent back to SCMS. • System logs the transaction for auditing.
Trigger	SCMS sends a budget approval event that includes transaction data to be logged.
Main Flow	1. SCMS formats and sends transaction payload to the University Financial System. 2. Financial System verifies and logs the transaction in the appropriate club account. 3. Returns success message with transaction ID. 4. SCMS updates the budget request status to "Disbursed." 5. Notification sent to relevant parties.
Exception Flow	• Financial system down: SCMS retries automatically. • API error or timeout: transaction is queued and user is notified. • Data format mismatch: error logged, SCMS alerts admin.


BudgetService %% triggers after approval
Finance Officer
University Financial System <<external>>
NotificationService
Club Officer
promptDisburse(reqId,amount)
postTransaction(clubId,amount,reqId)
alt
[Success]
txnId
disbursementOK(txnId)
notifyClub(txnId,amount)
"Funds released"
[Failure]
error(code)
disbursementFail(err)
alertClub(err)
"Disbursement failed"

3.1.3.8 Venue Reservation System

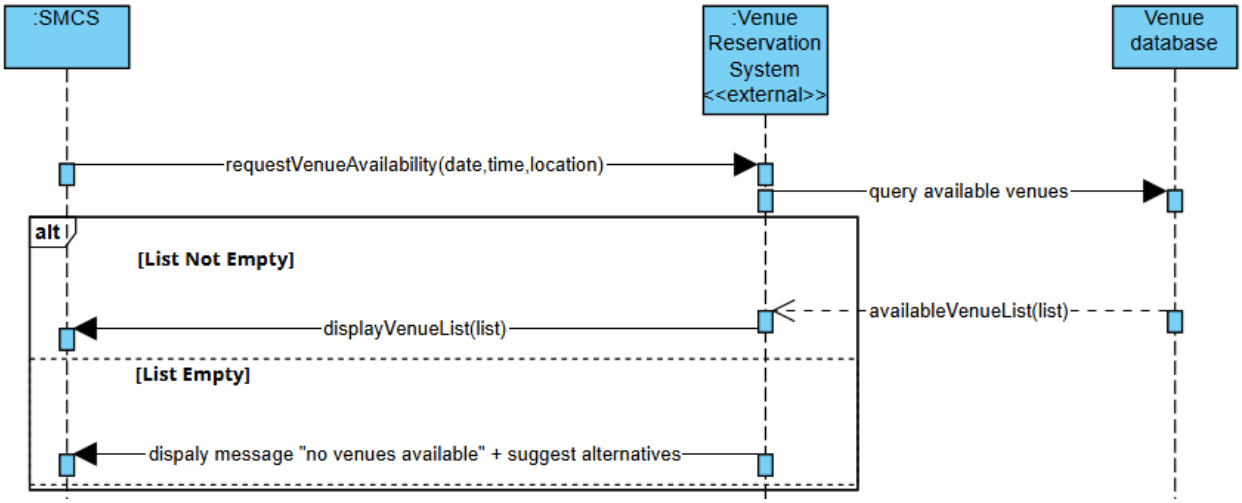
3.1.3.8.1 Use Case: Reserve Venue

Use Case Element	Description
Primary Actor	Venue Reservation System
Secondary Actors	Club Management System
Purpose	To reserve a venue for an event or meeting.
Preconditions	Venue is registered. Desired venue and date information is provided.
Postconditions	Venue is reserved and confirmation is saved. Reservation request is declined due to venue unavailability.
Trigger	A request to reserve a venue is received based on date, time, and venue.
Main flow	System receives prerequisites of the venue, date, and time. System sends reservation request (venueID, date, time) to Venue Database. Venue system checks for availability. If available, create reservation entry. System returns confirmation ID.
Alternative Flow – Venue Not Available	Venue Reservation System returns "unavailable".
Exception Flows	Database connection fails, system displays an error and retries or aborts.


:SCMS
:Venue Reservation System <<external>>
Venue database
requestReservation(venueID,date,time)
checkAvailability(venueID,date,time)
alt
[Venue Available]
availability confirmed
lockVenue(venueID)
update venue status to "reserved"
reservation confirmed
[Venue Not Available]
showMessage("no venues available"), suggest alternative
availability denied
display error"venue no longer available"

3.1.3.8.2 Use Case: Provide Availability Search

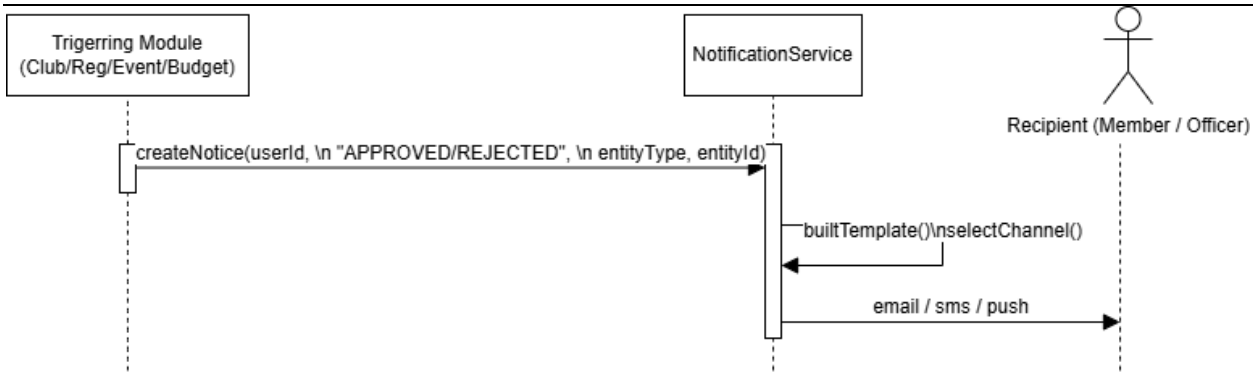
Use Case Element	Description
Primary Actor	Venue Reservation System
Secondary Actors	Club Management System, Venue Database
Purpose	To retrieve a list of venues available for a specified date, time, and location to support event planning.
Preconditions	• Venue is registered. • Date, time, and location inputs are provided.
Postconditions	• List of available venues is retrieved.
Trigger	• System receives a request for selected venue availability (date, time, location).
Main Flow	• Club Management System sends venue availability request to Venue Reservation System. • System queries the Venue Database. • Venue Database returns available venues. • Venue Reservation System sends the list back to the Club Management System.
Alternative Flow – No Venues Available	• Venue Database returns an empty list. • System displays a "no venues available" message and suggests alternate time or location.
Exception Flows	• Database connection fails, system displays an error and retries or aborts.


:SMCS
:Venue Reservation System <<external>>
Venue database
requestVenueAvailability(date,time,location)
query available venues
alt
[List Not Empty]
availableVenueList(list)
displayVenueList(list)
[List Empty]
dispaly message "no venues available" + suggest alternatives

3.1.3.9 Notification Service

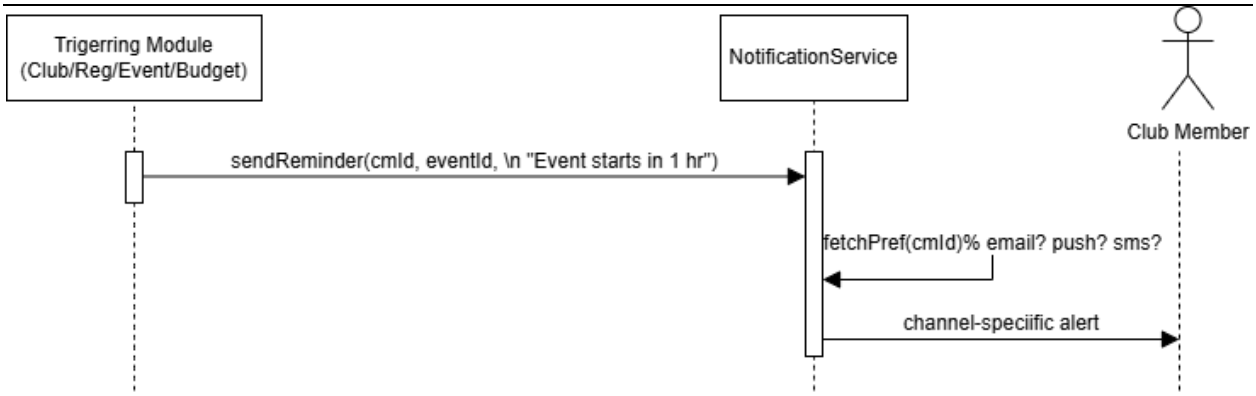
3.1.3.9.1 Use Case Specification: Deliver Status Notification

Use Case Element	Description
Primary Actor	Notification Service
Purpose	To deliver real-time status notifications to users through email, push, or SMS.
Preconditions	• A triggering event occurs in SCMS • Recipient has opted in and notification preferences are set. • Notification Service is operational.
Postconditions	• Message is sent and delivery is logged. • SCMS receives delivery status (success/failure).
Trigger	SCMS triggers a notification event.
Main Flow	1. SCMS sends notification payload (user ID, message, channel) to Notification Service. 2. Notification Service validates data and message type. 3. Dispatches message via selected channels (emails, push, SMS). 4. Logs delivery result and returns status to SCMS. 5. SCMS updates notification history for the users.
Alternative Flow – User Opted Out	• Notification is skipped for opted-out channels. • System logs that user declined delivery.
Exception Flow	• Service outage or API error: SCMS queues the notification for retry. • Malformed payload: Service rejects request and logs error. • Rate limit exceeded: SCMS delays further notifications and alerts admin.


Trigerring Module
(Club/Reg/Event/Budget)
NotificationService
Recipient (Member / Officer)
createNotice(userId, \n "APPROVED/REJECTED", \n entityType, entityId)
builtTemplate()\nselectChannel()
email / sms / push

3.1.3.9.2 Use Case Specification: Send Reminder Alert

Use Case Element	Description
Primary Actor	Notification Service
Purpose	To automatically send schedule reminder alerts to users (e.g., upcoming event, RSVP deadline, form resubmission).
Preconditions	• User has opted in to receive reminders. • A relevant upcoming event or deadline exists in the system. • Notification Service is active.
Postconditions	• Reminder is sent through preferred channel(s). • Delivery status is logged in the system.
Trigger	Scheduled time for a reminder is reached.
Main Flow	1. SCMS identifies upcoming reminders based on user preferences and event data. 2. Sends request to Notification Service with message and recipient list. 3. Notification Service dispatches reminders via email, push, or SMS. 4. Logs each delivery and returns status to SCMS. 5. SCMS records delivery history in user profile.
Alternative Flow – Reminder Skipped	If user opted out or disabled reminders: System respects preference and skips delivery. Logs “opt-out” status.
Exception Flow	• Service is temporarily down: SCMS queues alert. • Malformed message or missing data: system logs error and skips that alert. • Delivery service rate limit exceeded: system slows dispatch and notifies admin.


Trigerring Module
(Club/Reg/Event/Budget)
NotificationService
Club Member
sendReminder(cmId, eventId, \n "Event starts in 1 hr")
fetchPref(cmId)% email? push? sms?
channel-speciific alert

3.2 Performance Requirements

This table describes the performance requirements for the student club management system.

Requirement ID	Description	Priority
Core Functional Responsiveness		
REQ_P001	The system shall respond to event creation or updating operations within ≤2 seconds, ensuring a smooth user-friendly experience.	High
REQ_P002	The system shall complete the "Join Club" workflow, creating the membership in ≤ 5 seconds under normal load.	High
REQ_P003	The system shall record each member's presence via QR or NFC scan in ≤ 1 second per scan.	High
REQ_P004	Real-time attendance dashboard shall update in ≤1 second of each new scan.	High
REQ_P005	The system shall record approved financial transactions and update balances ≤ 2 seconds of transaction approval.	High
REQ_P006	Transitions between guided-wizard steps (e.g., from "Event Details" to "Date & Time") shall occur in ≤ 300 ms under normal load.	
Messaging		
REQ_P007	Once sent, messages shall be delivered to the recipient in ≤ 500 ms under normal load.	High
REQ_P008	Fetching and displaying a message thread shall be done in ≤ 1 second under normal load.	Medium
Event & Venue Management		
REQ_P009	If the event proposal is approved, the approved event must be displayed in the system within 24 hours of approval.	High
REQ_P010	The system shall retrieve the list of available venues from the Venue Reservation System in ≤ 5 seconds.	High
REQ_P011	Upon successful reservation, the system shall immediately tag the reserved venue as "reserved" and lock it to prevent double bookings in ≤ 1 second of confirmation.	High
REQ_P012	The system shall provide real-time venue auto-suggestions, displaying up to 5 matching venues that share the same criteria	High

	as the user types, with each suggestion list generated in ≤ 2 seconds.	
Security & Load Handling		
REQ_P013	The system's data, including personal information of students and Admins, finance information and club data, shall be protected from unauthorized access.	High
REQ_P014	The system shall successfully handle up to 100 simultaneous QR/NFC based login requests without errors or service degradation.	High
REQ_P015	The system shall record approved financial transactions and update balances ≤ 2 seconds of transaction approval.	High
Notifications & Alerts		
REQ_P016	Successful notification delivery rate shall be ≥ 95%.	High
REQ_P017	Failed reminder alert deliveries shall retry each alert up to 3 times within a ten-minute period.	High
REQ_P018	Upon successful opt-in, the system shall send a "Confirmation" push notification (test alert) within 30 seconds for ≥ 98% of new opt-ins.	Medium
Analytics & Reports		
REQ_P019	The system shall generate a club's monthly member attendance analytics report and display it on the "club analytics" dashboard in ≤ 10 seconds of request.	Medium
File Upload & Media		
REQ_P020	File uploads shall complete uploading in ≤ 4 seconds and make the file available for download immediately upon completion for review.	Medium
REQ_P021	Uploading and previewing an event banner image (up to 10 MB) in the wizard shall complete in ≤ 3 seconds, with the image displayed immediately upon successful upload.	Low
Gamification & Badges		
REQ_P022	Badge notifications shall be delivered via in-app notification or mobile push notifications in ≤ 3 seconds of badge assignment.	Medium

3.3 Usability Requirements

This table describes the usability requirements for the student club management system.

Requirement_ID	Description	Priority
REQ_UR001	The interface shall allow a club officer to complete the event creation process in fewer than 3±2 steps.	High
REQ_UR002	The event creation process shall provide inline tools to act as aid or guidance to minimize input error.	Medium
REQ_UR003	System notifications (e.g., budget denial, event rejection, etc.) shall be clear, contextual, and provided within 5 seconds of a decision.	High
REQ_UR004	If the venue search returns no venue, the system must suggest at least two venues as an alternative.	High
REQ_UR005	The system interface shall be responsive and fully usable across desktop and mobile devices.	High
REQ_UR006	The system shall support standard accessibility gestures and voice commands for mobile/touch-based devices.	High
REQ_UR007	Users shall be able to preview and remove (if needed) any uploaded content before final submission to reduce error.	High
REQ_UR008	The system shall prevent irreversible actions (e.g., club deletion, event cancellation, budget disbursement, etc.) by requiring confirmation prompts with a clear consequences dialogue box and “Undo” options where applicable.	High
REQ_UR009	The system shall restrict visibility of sensitive financial and personal data based on user roles.	High
REQ_UR010	System icons and labels shall follow standard UX patterns and practices (e.g., envelope for messages), to ensure clarity and familiarity.	Medium
REQ_UR011	The system shall require a confirmation step before sending messages to more than 10 recipients, clearly displaying the number of recipients and a preview of the message.	High
REQ_UR012	Club officer shall be shown a summary screen to review all event details before final event creation, with the ability to go back and edit any section.	High

REQ_UR013	Users attempting to leave a form with unsaved changes shall be prompted with a warning box to prevent accidental data loss.	High
REQ_UR014	Role-assignment pages shall display clear warnings before changing permissions of users with administrative privileges.	High
REQ_UR015	The system shall automatically log out users after 15 minutes of inactivity and prompt re-authentication, to prevent unauthorized access on shared devices.	High
REQ_UR016	Users shall be notified of session timeouts with a warning 2 minutes before auto-logout, giving them an option to extend the session.	Medium
REQ_UR017	Deactivating a club shall not remove its data immediately; instead, the system shall archive it for 30 days with a clear option to restore.	High
REQ_UR018	The system shall display a visual calendar or timeline view of all venue bookings to help users avoid overlapping or unavailable slots.	High
REQ_UR019	If a user selects a venue that is already booked, the system shall immediately show a conflict warning and suggest the nearest available alternatives.	High
REQ_UR020	Upon venue selection, the system shall display venue rules, capacity limits, and equipment restrictions to avoid accidental misuse.	Medium
REQ_UR021	A booking summary, including cost (if applicable), location, capacity, and time, shall be shown before final confirmation to prevent mis-bookings.	High
REQ_UR022	Upon submission, users shall be shown a complete budget request summary with a confirmation prompt before finalizing.	High
REQ_UR023	If a budget request is rejected, the system shall show the reason provided by the finance officer.	High
REQ_UR024	If duplicate scans are attempted, the system shall ignore the new scans and notify the user that they already did scan.	High

REQ_UR025	Club officers shall be able to export attendance data in CSV or Excel format with one click.	Medium
REQ_UR026	Club Members shall be able to view their personal attendance history.	Medium

3.4 Interface Requirements

This section describes the interfaces of the Student Club Management System (SCMS), including interactions with external systems, the user interface layout, hardware interfaces for attendance, APIs, and communications protocols.

3.4.1 System Interfaces:

I. University Authentication System (LDAP)

- Purpose: To authenticate students and staff using their institutional credentials.
- Users: Club Member, Club Officer, Finance Officer, Student Affairs Officer, System Administrator.
- Use Cases: Log in, Register, QR Sign-In via Mobile.
- Authentication Flow: Credentials entered in login or registration interface are validated via secure university authentication servers.

II. University Financial System

- Purpose: To manage budget requests, fund disbursements, and provide live balance tracking.
- Users: Finance Officer, Club Officer.
- Use Cases: Submit Budget Request, Disburse Funds, Record Approved Transactions, Provide Live Balances, Realtime Budget Updates.
- Integration:
    - System pulls approved transaction data and budget updates.
    - System submits budget requests to the financial backend.
    - Live balances are displayed in the Club Officer's and Finance Officer's dashboards.

III. Venue Reservation System

- Purpose: To provide venue booking capabilities for club events.
- Users: Club Officer, Student Affairs Officer, Venue Manager.
- Use Cases: Search & Book Venue, Reserve Venue, Auto-Suggest Optimal Venues, Process Booking.
- Integration: SCMS will consume APIs to check venue availability, auto-suggest optimal venues based on event data (capacity, type), and complete booking transactions.

IV. University Notification System

- Purpose: To send system-generated and custom notifications to users.
- Users: All roles indirectly via Send Notification, Send Reminder Alerts.
- Use Cases: Deliver Status Notifications, Send Messages, Reminder Alert, Mobile Push Opt-In.
- Integration: SCMS queues messages via Notification API; delivery handled by external system via push/email/SMS.

3.4.2 User Interfaces

SCMS will be accessible via responsive web and mobile interfaces, tailored to the user's role (e.g., Club Member, Club Officer, System Admin).

I. Navigation and Layout:

- A left-side navigation bar will give access to core modules (Dashboard, Events, Budgets, Messages).
- A fixed top navigation bar will display user info, alerts, and account settings.
- All interfaces will be mobile-responsive.

II. Login and Authentication:

- Login and registration screens will validate credentials via LDAP.
- Password reset and QR login support are provided.

III. Club Member Dashboard:

- Allows joining clubs, RSVPing to events, messaging, and managing preferences.

IV. Club Officer Tools:

- Access to club registration, member role management, budget submission, and event creation.

V. Finance Officer Interface:

- Modules for processing and disbursing funds, and accessing balance reports.

VI. Attendance System:

- NFC/QR Scan page for logging attendance at events using camera/NFC reader.

VII. Notification Panel:

- View and configure mobile push alerts, reminders, and admin notices.

3.4.3 Hardware Interfaces

I. NFC Reader Integration:

- Purpose: For event attendance logging using contactless cards or mobile devices.
- Use Cases: Scan QR/Attendance, Track Attendance.
- Support: USB or built-in NFC readers on tablets and kiosks.

II. Mobile Device Camera:

- Purpose: For scanning QR codes.
- Use Cases: Scan QR/Attendance, QR Sign-In via Mobile.
- Integration: Uses browser or mobile app camera APIs to read QR codes.

3.4.4 Software Interfaces

- Google Maps API:
  Used to display venue locations during event planning. Venue pins, directions, and maps will be embedded in the UI.
- Notification Delivery Services:
  The system may integrate with services like Firebase Cloud Messaging (FCM) or internal SMTP servers for push/email alerts.

- Data Export Utilities:
  Budget and attendance reports will be exportable in CSV format, compatible with Excel and other spreadsheet tools.

3.4.5 Communications Interfaces

I. Google Maps API:
   - Purpose: Display venue locations and suggestions during event creation.
   - Use Cases: Auto-Suggest Optimal Venues, Create Event.
   - Integration: Events module includes a map with location picker and route display.

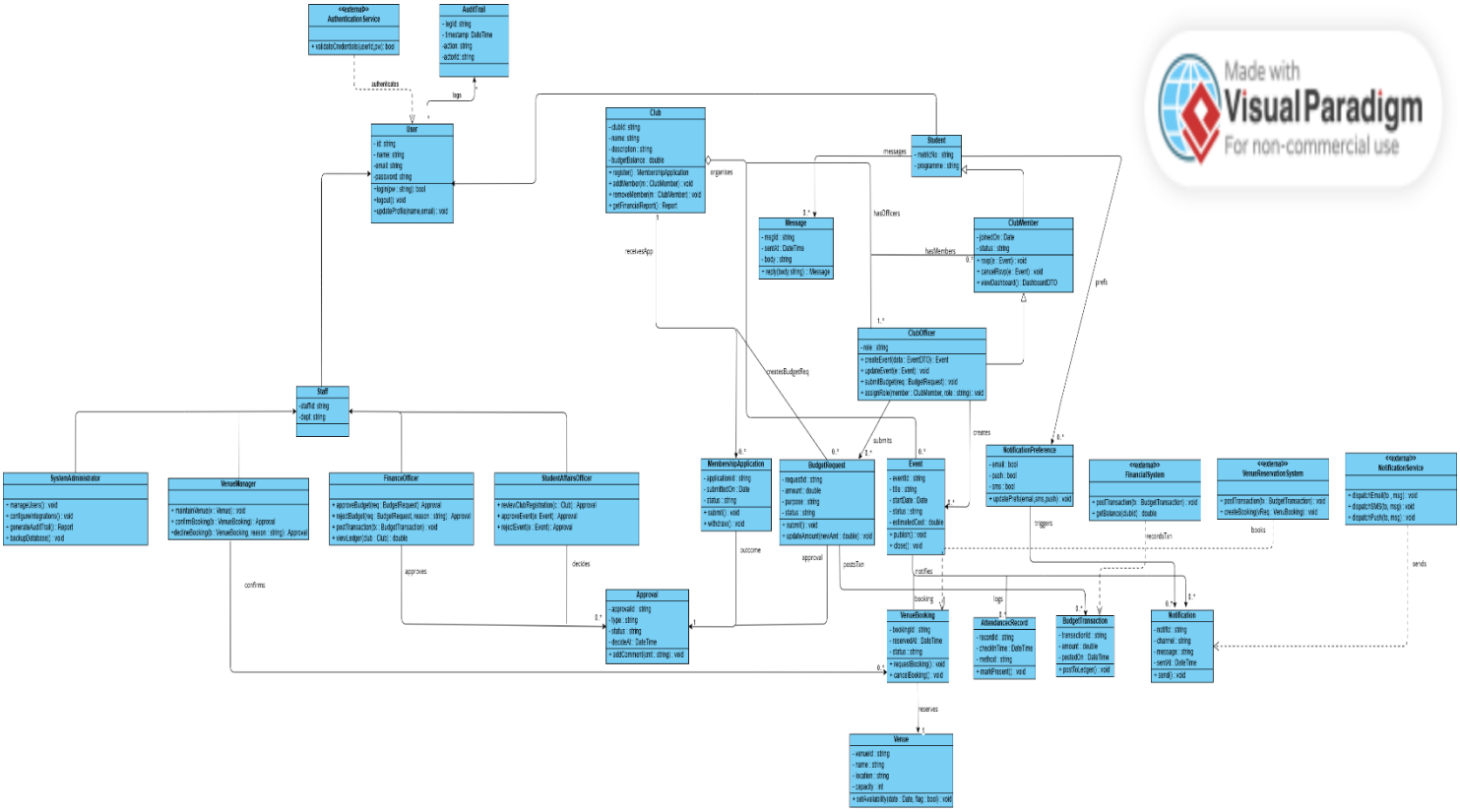
II. Notification Service Integration:
   - Purpose: Delivers real-time notifications and scheduled reminders.
   - Use Cases: Send Notification, Reminder Alert, Deliver Status Notifications.
   - Support: Email, SMS (via third-party providers like Twilio), or Push (via Firebase).

III. CSV Export (Finance):
   - Purpose: Allow officers to download budget and event data.
   - Use Cases: Export Financial CSV, Club Analysis.

IV. Audit & Logging Tools:
   - Purpose: Compliance tracking, activity monitoring.
   - Use Cases: Generate Audit Trail, Monitor Compliance.

3.5 Logical Database Requirements

The Student Club Management System (SCMS) database is designed to support a modular, role-based application that manages clubs, events, budgeting, and communications within a university ecosystem. The system leverages relational principles to ensure data integrity, consistency, and accessibility across modules. Key data entities and their relationships are outlined below:



link: Class Diagram

Key Data Entities:

User Management Entities

- User: Core entity with attributes including id, name, email, and password. Serves as the base entity for all system users with authentication capabilities.
- Student: Inherits from User, with additional attributes matricNo (matriculation number) and programme to identify academic program enrollment.
- Staff: Inherits from User, with attributes staffId and dept (department) for organizational identification.

Student Role Entities

- ClubMember: Represents students participating in clubs with attributes joinedOn (membership date) and status (Active/Inactive). Provides event RSVP functionality.
- ClubOfficer: Inherits from ClubMember with additional role attribute (President, Treasurer, etc.) and extended privileges for club management.

Staff Role Entities

- StudentAffairsOfficer: Manages club registrations and event approvals with specialized review capabilities.
- FinanceOfficer: Handles budget approvals and financial transactions with ledger management functions.
- VenueManager: Manages venue maintenance and booking confirmations.
- SystemAdministrator: Provides system-wide management including user administration and audit capabilities.

Core Domain Entities

- Club: Central entity with attributes clubId, name, description, and budgetBalance. Manages organizational structure and financial tracking.
- Event: Contains eventId, title, startDate, status, and estimatedCost for activity management.
- Venue: Physical location entity with venueId, name, location, and capacity attributes.
- MembershipApplication: Application tracking with applicationId, submittedOn, and status (Pending/Approved/Rejected).

Process Management Entities

- BudgetRequest: Financial request tracking with requestId, amount, purpose, and approval status.
- BudgetTransaction: Financial transaction recording with transactionId, amount, and postedOn timestamp.
- VenueBooking: Reservation management with bookingId, reservedAt, and booking status.
- Approval: Generic approval workflow entity with approvalId, type, status, and decision timestamp.

Supporting Entities

- AttendanceRecord: Event participation tracking with recordId, checkInTime, and attendance method.
- Notification: Communication entity with notifId, channel, message, and delivery timestamp.
- NotificationPreference: User communication preferences for email, SMS, and push notifications.
- Message: Internal messaging system with msgId, sentAt, and message body.
- AuditTrail: System activity logging with logId, timestamp, action, and actorId.

Key Relationships:

Inheritance Relationships

- User serves as the parent class for Student and Staff, establishing role-based access control.
- Student extends to ClubMember, which further extends to ClubOfficer, creating a hierarchy of student privileges.
- Staff extends to specialized roles (StudentAffairsOfficer, FinanceOfficer, VenueManager, SystemAdministrator).

Core Business Relationships

- Club-Member Association: One Club has zero to many ClubMembers, with ClubOfficers being a specialized subset.
- Club-Event Management: One Club organizes zero to many Events, with ClubOfficers creating and managing events.
- Event-Venue Booking: Each Event requires one VenueBooking, which reserves one specific Venue.
- Budget Management: Clubs create BudgetRequests, which generate BudgetTransactions upon approval.

Approval Workflows

- MembershipApplication → Approval: Each application has one approval outcome.

- BudgetRequest → Approval: Financial requests require approval before transaction processing.
- Event Planning: Events may require approval from StudentAffairsOfficers before publication.

Communication and Tracking

- Event → Notification: Events generate notifications to relevant stakeholders.
- User → NotificationPreference: Users configure their communication preferences.
- Event → AttendanceRecord: Events log attendance for participation tracking.
- System-wide Auditing: All User actions are logged in AuditTrail for compliance.

Key Constraints:

Data Integrity Constraints

- Unique Identifiers: All entities require unique primary keys (id, clubId, eventId, etc.)
- Email Uniqueness: User email addresses must be unique across the system.
- Matriculation Numbers: Student matricNo must be unique within the academic institution.
- Staff Identification: Staff staffId must be unique within the organization.

Business Rule Constraints

- Membership Status: ClubMember status must be either 'Active' or 'Inactive'.
- Application Status: MembershipApplication status limited to 'Pending', 'Approved', or 'Rejected'.
- Budget Balance: Club budgetBalance cannot be negative (non-negative constraint) .
- Event Capacity: VenueBooking cannot exceed Venue capacity limits.
- Date Consistency: Event startDate cannot be in the past, VenueBooking reservedAt must be future-dated.

Role-Based Constraints

- Officer Requirements: ClubOfficer must first be a ClubMember of the same club

- Approval Authority: Only designated staff roles can approve specific request types (StudentAffairsOfficer for events, FinanceOfficer for budgets)
- Venue Management: Only VenueManager can confirm or decline VenueBookings

External System Constraints

- Authentication Dependency: User login requires validation through AuthenticationService
- Financial Integration: BudgetTransactions must be recorded in external FinancialSystem
- Venue Synchronization: VenueBooking availability must be verified with VenueReservationSystem
- Notification Delivery: All Notifications must be processed through NotificationService

Temporal Constraints

- Event Lifecycle: Events must follow status progression (Draft → Published → Closed)
- Audit Requirements: All significant actions must generate AuditTrail entries with accurate timestamps
- Booking Timeframes: VenueBooking requests must be submitted within specified advance notice periods

3.6 Design Constraints

This section outlines the design constraints that the Student Club Management System must adhere to. These constraints include regulatory requirements, university policies, and technical limitations that impact the system's architecture, user interface, or functionality.

3.6.1 Branding and UI Guidelines

The user interface must adhere to the university's official branding guidelines. This includes the correct use of logos, color palettes, font styles, and layout standards as defined by the university's digital identity manual. Accessibility is also a priority; the interface must conform to WCAG standards to ensure usability for all students, including those with disabilities. Additionally, both light and dark mode support must be incorporated to enhance user experience across different environments.

3.6.2 Authentication and Security

Authentication must be handled through the university's centralized LDAP system. Under no circumstance should user credentials be stored or processed locally. All login and identity management functions must delegate to the university's secure authentication server. The system is required to implement role-based access control (RBAC) to ensure that users only access features and data appropriate to their assigned roles, such as Club Member, Club Officer, Finance Officer, or Administrator.

3.6.3 Offline Attendance Support

Attendance tracking features must be designed with offline capability in mind. In scenarios where internet connectivity is unavailable—such as outdoor events—the system must still allow QR and NFC-based attendance logging. Attendance data should be stored securely on the device and synchronized with the central server once the connection is re-established, ensuring no data is lost during offline use.

3.6.4 Integration Constraints

The SCMS must integrate with existing university systems, including the Financial System and the Venue Reservation System, without modifying their underlying data structures. These integrations must use predefined API contracts and respect the access permissions provided by the external systems. For example, while SCMS may read financial data or submit budget proposals, it cannot perform administrative functions on those external platforms.

3.6.5 Deployment and Hosting Constraints

Deployment of the SCMS is restricted to the university's approved hosting environments. This includes internal servers or institutionally-managed cloud platforms such as AWS or Azure. The system must comply with the university's IT and cybersecurity policies, including requirements for data encryption, backup, firewall protection, and routine software updates.

3.6.6 Data Retention and Privacy

Data handling within the SCMS must follow the university's data protection policies and any applicable legal frameworks such as the General Data Protection Regulation (GDPR). Audit logs, especially those related to financial transactions and attendance records, must be retained for at least 12 months for auditing and compliance purposes.

3.6.7 Mobile and Browser Compatibility

The system must be fully compatible with modern web browsers including Chrome, Firefox, Safari, and Edge. Mobile access is essential, particularly for attendance scanning and on-the-go notifications, and must support devices running Android version 10 or higher and iOS version 14 or higher.

3.6.8 Third-Party Component Restrictions

Only third-party libraries and APIs that are open-source or appropriately licensed for academic or institutional use may be used. Components with restrictive licenses, such as GPL-licensed frontend frameworks, must be avoided unless explicit approval is granted by the university's legal or IT department.

3.7 Software System Attributes

This section outlines the quality attributes expected of the Student Club Management System (SCMS), which contribute to its reliability, usability, and long-term performance.

Attribute	**Description**
Reliability	The system shall maintain consistent availability and data integrity. It must recover from crashes within 1 minute, and failed operations shall be logged and retried automatically where possible.
Availability	SCMS must maintain 99.9% uptime during university hours (Monday to Friday, 8 AM to 6 PM). Scheduled maintenance windows shall be announced in advance.
Security	The system shall implement role-based access control (RBAC), session timeout policies, and data encryption for sensitive fields (e.g., passwords, financial info). Audit trails must record all admin-level activities.
Maintainability	SCMS shall be developed using modular architecture, allowing components such as event, finance, and messaging modules to be updated independently. Code must follow university coding standards and documentation guidelines.
Portability	The application shall be compatible with both Windows and Linux server environments. Client interfaces must support modern web browsers (Chrome, Firefox, Edge) and adapt to different screen sizes.
Scalability	The system shall support up to 200 concurrent users without significant performance degradation. APIs must be designed to support horizontal scaling in the future.
Auditability	All critical actions must be logged in an audit trail that is accessible only to authorized admins for compliance review.

3.8 Supporting Information

During the requirements elicitation phase for the Student Club Management System (SCMS), several methods were employed to ensure that the needs of student user, club officers, and administrative staff were accurately constraints. The following techniques are used:

I. Surveys and Kano Model Analysis

An online survey was distributed to students and club members to gather feedback on current pain points and desired features. The responses were analyzed using the Kano Model to categorize features into must-haves, performance elements, and delighters.

II. Document Review

Relevant documents such as club management policies, event guidelines, and financial procedures were reviewed. This helped identify institutional rules that the system must comply with.

III. Brainstorming Sessions

Internal brainstorming sessions were held among the development team to refine feature ideas, generate user stories, and align technical feasibility with user expectations.

IV. Existing System Observation

Features from existing university systems were analyzed for functionality, UI patterns, and integration points. Observing these helped define how SCMS could interact with them effectively.

3.8.1 Surveys and Kano Model Analysis

The following table summarizes feature evaluations based on user responses to Kano Model-style survey system.

ID	Description	Feature/Request/Opinion
Q01	92% of users responded positively to the idea of a step-by-step wizard; 60% disliked having to manually create events.	Re1 01 – Guided event creation wizard.

Q02	Over half of users expect real-time updates. Delays in budget visibility were rated as frustrating or unacceptable.	Req 02 – Real-time budget updates
Q03	Users overwhelmingly want searchable venue listings. 80% disliked manual searching or relying on static lists.	Req 03 – Venue search and filter
Q04	94% want a digital RSVP system; 98% found no-RSVP systems frustrating.	Req 04 – Online RSVP system
Q05	90% prefer QR or NFC scanning over manual sign-in. Speed and accuracy were emphasized.	Req 05 – QR/NFC attendance capture
Q06	Majority of users want CSV export for transparency and reporting. 82% disliked not having this feature.	Req 06 – CSV export for financial reports
Q07	90% preferred joining clubs online. 92% found manual sign-ups inconvenient.	Req 07 – Online membership application
Q08	Users expect tailored dashboards. 90% found generic or cluttered layouts unhelpful.	Req 08 – Custom user dashboard
Q09	Most users want reminders before events. 96% found it frustrating when none were provided.	Req 09 – Reminder alerts for events
Q10	90% supported a built-in chat system. 92% disliked the lack of integrated communication.	Req 10 – In-system member messaging

3.8.2 Document Review

The following summarizes insights gathered from official documents that influenced SCMS feature and requirements.

ID	Description	Feature/Request/Opinion
D01	UPM's registration form (UPMSA/CR-2022) requires club name, objectives, and documents.	Req 1 – Enable secure online club registration with upload and status tracking.
D02	The budget request template outlines required sections (e.g., estimated cost, justification).	Req 2 – Strctured budget request form with validation and digital submission.
D03	Event proposals must go through multiple stages (Club Officer → Advisor → SAO).	Req 8 – Built-in workflow routing with approval status display.
D04	Clubs must assign roles like President, Secretary, Treasurer, etc., as per SAO guidelines.	Req 3 – Allow role assignment and editing within system.
D05	Clubs are required to submit post-event reports, often done via printed forms or email.	Req 15, Req 22 – Digital post-event report submission and CSV report export.

3.8.3 Brainstorming Sessions

The following insights were gathered during internal brainstorming sessions by the project team. These ideas were proposed based on known user frustrations, industry practices, and internal logic improvements.

ID	Description	Feature/Request/Opinion
B01	System must support secure, traceable digital submission of club registration forms and documents.	Req 1 – Transparent application and approval process.

B02	Account access should require username/password with optional multi-factor authentication.	Req 2 – Protect user data and access.
B03	Club officers must be able to assign formal positions (e.g., President, Treasurer) within the system.	Req 3 – Digital role management to avoid offline tracking.
B04	System must enforce access separation: Members, Officers, and Admins see only their relevant content.	Req 4 – Prevent data leakage or unauthorized actions.
B05	System must auto-notify users of changes to application or event statuses.	Req 8 – Real-time feedback loop.
B06	System should provide high-contrast mode, font scaling, and keyboard navigation.	Req 9 – Inclusive access for all users.
B07	Each user sees a dashboard with upcoming events, budget snapshots, and role-specific actions.	Req 17 – Clarity and fast navigation.
B08	System should send push/email alerts 24h before an event for RSVP'd users.	Req 18 – Boost event attendance.
B09	Members should be able to chat within the club system without using external apps.	Req 19 – Centralized communication.
B10	System suggests optimal venue based on event type, expected size, and availability.	Req 20 – Smart matching for logistics.
B11	Reward members for participation (e.g., attending events, hosting activities).	Req 21 – Motivate student engagement.
B12	Visualize trends in attendance, budget use, and event frequency.	Req 22 – Better decision-making for officers.
B13	Send urgent updates (e.g., cancellations) via mobile notifications.	Req 23 – Send urgent updates (e.g., cancellations) via mobile notifications.
B14	Suggest clubs based on a student's interests, past activity, or joined events.	Req 24 – Promote student discovery.

B15	Allow mobile users to scan a QR code to log into their SCMS web account.	Req 25 – Fast login experience.

3.8.4 Existing System Observation

The following table summarizes insights gathered from observing existing systems, which influenced specific feature requests for SCMS.

ID	Description	Feature/Request/Opinion
X1	The university's official venue reservation portal offeres a comprehensive catalogue of lecture halls and event spaces. It supports venue filtering by type and real-time clash detection before confirming bookings.	Req 12 – Searchable venue database with millisecond-level search, filter combinations, and hard stop on double bookings.
X2	PUTRA Presence is a UPM-wide attendance sustem using QR code and Bluetooth check-ins, authenticated via UPM ID. Live test show <2 second scan latency and offline caching for poor connectivity.	Req 14 – QR/Bluetooth attendance capture with sub-2-second response time and offline support.
X3	CampusGroups centralized club operations; budget requests, event RSVPs, all under one dashboard. Its UI and workflow design shows how multiple features can coexist cleanly.	Req 11 – Realtime budget updates, Req 13 – Online RSVP, Req 16 – Online membership application with unified navigation and role-based permissions.
X4	This app functions fully online, caches hundreds of check-ins, supports parallel scanning across deiced, and exports to CSV.	Req 14 – Offline fallback for attendance Req 15 – Bulk CSV export of attendance for post-event analytics.
X5	Eventbrite's smooth RSVP process includes auto e-ticketing and reminder	Req 13 – Single-click RSVP, Req 18 – Automated reminders via email/calendar invite.

	emails 24 hours pre-event. It sets expectations for event UX and turnout.	

4. Verification

4.1 Verification Approach

Verification of the Student Club Management System (SCMS) shall be conducted through a structured and multi-layered testing strategy to ensure full compliance with functional and non-functional requirements. The verification process will include unit testing, integration testing, system testing, functional testing, and user acceptance testing (UAT). Unit and integration testing will be performed during the development phase to validate individual components and the interactions between system modules. System and functional testing will be executed at the completion of each major development iteration to evaluate the overall performance and feature completeness of the system.

User acceptance testing will be carried out prior to deployment, involving actual end users such as club officers, finance officers, and administrative staff, to validate real-world usability and business rule compliance. These verification activities will be the joint responsibility of the software development team and the university's Quality Assurance (QA) department. The development team will focus on technical testing at the code and module levels, while the QA team will oversee comprehensive system testing and user-facing verification.

All verification procedures will be executed within a dedicated quality assurance testing environment that replicates the production infrastructure. User acceptance testing will occur in a secured staging environment accessible to authorized personnel, ensuring that testing outcomes are reflective of actual system conditions while maintaining data integrity and system stability.

4.2 Verification Criteria

The software will be verified against a set of functional and quality benchmarks derived from the system requirements. These criteria are essential to ensure that the system is robust, secure, user-friendly, and aligned with institutional goals.

- Authentication: All users must be authenticated via the university's LDAP system without delay or error. Unauthorized access attempts must be blocked.
- Response Time: Core user operations such as login, event creation, and attendance scanning must have a response time of less than 3 seconds under normal load conditions.
- Budget Submission Accuracy: Budget proposals submitted by club officers must be accurately recorded and reflected in the finance officer's dashboard without data loss or corruption.
- Event Booking Integration: Venue booking features must accurately reflect real-time availability from the university's Venue Reservation System, and duplicate bookings must be prevented.
- Offline Attendance Sync: When using the offline attendance feature, QR/NFC scan data must be stored locally and successfully synchronized with the central database within 5 minutes of reconnecting to the internet.
- Data Security: All personal and financial data must be transmitted over secure HTTPS protocols and stored with appropriate encryption.
- Role-Based Access: Users must only have access to the functionalities relevant to their assigned role, with no privilege escalation or data leakage between roles.
- Mobile Compatibility: The application must function correctly on the latest two major versions of Android and iOS, particularly for features like attendance scanning and notifications.
- Notification Delivery: System-generated notifications and reminders must be delivered within 60 seconds of being triggered.
- Error Handling: The system must display user-friendly error messages and log all critical system errors for further review.

5. Appendices

5.1 Assumptions and Dependencies

This section outlines the key assumptions made during system design and external dependencies that may affect the operation or delivery of the SCMS.

Assumption / Dependency	**Description**
A1: UPM SSO is always available	The SCMS relies on UPM's Single Sign-On (SSO) service for user authentication. The system assumes this service is operational.
A2: Financial and venue systems are integrated	The SCMS is dependent on successful integration with the University Financial System and Venue Reservation System.
A3: Notification Service is reliable	Timely delivery of alerts and reminders is dependent on the availability of the university's Notification Service.
A4: Student Affairs and Finance Officers are available to process requests	Workflow operations assume that staff users perform timely approvals.
A5: Clubs have internet access	Users must have a stable internet connection to interact with SCMS.
A6: All users understand basic system usage	Assumes that students and staff can navigate basic web applications and use provided instructions.
A7: Scheduled Maintenance is Communicated	System may experience downtime for updates; assumes maintenance schedules are communicated in advance.

5.2 Acronyms and Abbreviations

The table below lists acronyms and abbreviations used throughout this document, along with their full meanings for ease of reference.

Acronyms / Abbreviations	**Meaning**
SCMS	Student Club Management System
SSO	Single Sign-On
SRS	Software Requirements Specification
UI	User Interface
API	Application Programming Interface
DTO	Data Transfer Object
CSV	Comma-Separated Values
OTP	One-Time Password
RBAC	Role-Base Access Control
UPM	Universiti Putra Malaysia
UAT	User Acceptance Testing
CRUD	Create, Read, Update, Delete

5.3 Glossary

The following glossary defines key terms used throughout this Software Requirements Specification to ensure clarity and consistency.

Term	**Definition**
Club Member	A student who has successfully joined a registered club through the SCMS platform.
Club Officer	A member with special privileges such as creating events, managing roles, and submitting budget requests.
Event Proposal	A formal request submitted by a Club Officer detailing the plan, date, and budget of an upcoming event, pending approval from Student Affairs.
RSVP	A feature that allows Club Members to confirm their attendance for events.

Budget Request	A proposal submitted to request funding for a club event, subject to approval by Finance Officers.
Notification	A message delivered to users via email, push or SMS to inform them of changes, deadlines, or approvals.
Venue Booking	A reservation request made by a Club Officer to secure a venue for an event.
Audit Trail	A secure log of actions taken within the system for accountability and compliance.
System Administrator	A staff role responsible for managing user accounts, integrations, monitoring, and audit functions within SCMS
SCMS	Student Club Management System, the web-based platform for managing club operations, events, budgets, and communication.