Centennial College

# Software Requirements Engineering (COMP 225 - 001) Group 2

Kaylie Moskal - 301494702

Contents

[Software Requirements Engineering (COMP 225 - 001) Group 2 1](#_Toc205734072)

[Project Overview 3](#_Toc205734073)

[CampusConnect 3](#_Toc205734074)

[1.1 Purpose: 4](#_Toc205734075)

[1.2 Document Conventions: 4](#_Toc205734076)

[1.3 Intended Audience & Reading Suggestions: 4](#_Toc205734077)

[1.4 Project Scope: 5](#_Toc205734078)

[1.5 References: 5](#_Toc205734079)

[Section 2 “Overall description” and sub sections: 5](#_Toc205734080)

[2.1 Product Perspective: 5](#_Toc205734081)

[2.2 Product Features (Functions): 6](#_Toc205734082)

[2.3 User classes and Characteristics: 6](#_Toc205734083)

[2.4 Operating Environment: 6](#_Toc205734084)

[2.5 Assumptions and Dependencies: 7](#_Toc205734085)

[Section 3: External Interface Requirements 7](#_Toc205734086)

[3.1 Frontend (User Interfaces): 7](#_Toc205734087)

[3.2 Hardware Interfaces: 8](#_Toc205734088)

[3.3 Software Interfaces 9](#_Toc205734089)

[Section 4. Analysis Modeling 10](#_Toc205734090)

[4.1 Overview Of Modeling 10](#_Toc205734091)

[4.2 Formal Use Case Description – Register Event 10](#_Toc205734092)

[4.3 Swimlane Diagram – Register Event 12](#_Toc205734093)

[4.4 Use Case Table -Subsystem: Event Management 13](#_Toc205734094)

[Section 5 - Requirements Lists 15](#_Toc205734095)

[5.1 Functional Requirements 15](#_Toc205734096)

[5.2 Non-Functional Requirements 18](#_Toc205734097)

[Appendices 20](#_Toc205734098)

[Appendix C - Stakeholders Register 20](#_Toc205734099)

[Appendix D: Interview Questions and Answers 20](#_Toc205734100)

[Appendix E: 22](#_Toc205734101)

[Event States (proposed): 30](#_Toc205734102)

[Triggers (System/User Actions): 30](#_Toc205734103)

[State Transitions: 30](#_Toc205734104)

**Group Project**

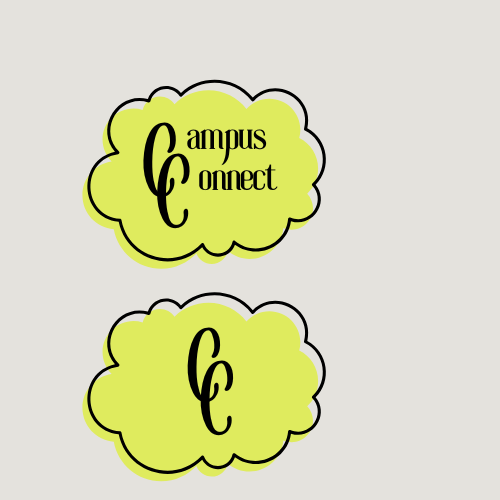
*Kaylie Moskal, Cabdirahmaan Ibrahim, Athavan Yokanathan, Kerrick Shiu, Ismaeel Abdus-Samad*

## Project Overview

**The Problem:** Many students at our school are unaware of the various events taking place on campus. Despite a wide range of activities happening regularly, the lack of awareness leads to low participation and missed opportunities for community engagement.

**Our Solution: CampusConnect***CampusConnect* is a campus-focused platform designed to keep students and faculty informed and engaged. The platform will allow users to discover events aligned with their interests, receive personalized notifications, and even create and promote their own events. Think of it as a blend between MeetUP and Eventbrite, tailored specifically for our campus community. With *CampusConnect*, we aim to foster a more connected, vibrant, and involved student life.

### CampusConnect

****

Main Logo and App Logo

### 1.1 Purpose:

Will address the issue of managing campus events by allowing students and faculty to create, discover, and register for events through a centralized platform.

### 1.2 Document Conventions:

|  |  |
| --- | --- |
| **Acronyms** | **Description** |
| RESTful | Representational State Transfer |
| APIs | Application programming interface(s) |
| SQL | Structured Query Language |
| JS | JavaScript |

### 1.3 Intended Audience & Reading Suggestions:

Developers:

Developers will use this document to understand the system requirements and features needed to build the Activity Finder on Campus application. They should focus on sections 2 and 3 which outline the system features, user interface, and technical details.

Testers/Engineers:

They validate that the application meets all defined requirements and functions correctly under different conditions. Creates testing cases/scenarios based on the documented use cases, functional requirements and user interface.

Marketing teams :

The marketing team will use this document to learn about the app’s purpose and key features so they can promote it effectively to students and faculty. They should read sections 1.1 to 1.4 to understand the project’s goals, audience, and scope.

End users:

Students/faculty and club leaders will use this document to understand what the app does and how it benefits them. They are encouraged to review sections 1.1 and 1.4 to see how the platform helps manage and join campus events. Also to provide feedback during testing phases to evaluate whether the application meets their needs for campus engagement.

### 1.4 Project Scope:

Many students and faculty members struggle to stay informed about events, clubs, and activities happening on campus. *Campus Connect* is a centralized digital platform that allows users to create, discover, and register for campus-related events. By signing in through their student or faculty credentials, users can access a personalized feed of upcoming activities based on their interests, program, or past participation.

The platform allows students and faculty to post events or club activities they are affiliated with, while also offering a shared calendar to keep track of what’s happening across the campus. To maintain a professional and safe environment, all posts are monitored by the school's IT department to ensure they align with the institution’s image and policies.

The software integrates with the school’s Blackboard or Luminate system, making it easier to sync academic schedules, manage user access, and streamline communication.

In addition to boosting campus engagement, the platform provides students with the opportunity to build their leadership and organizational skills. Students who create or manage events and clubs through the platform can include these experiences on their resumes or co-op applications, helping them stand out in future job opportunities by demonstrating initiative, teamwork, and campus involvement.

By improving event visibility and student participation, the system supports the school’s goal of promoting a more connected, active, and career-ready campus community.

### 1.5 References:

Meetup:<https://www.meetup.com/>

Kommunity:<https://kommunity.com/>

Blackboard Learn Documentation:<https://help.blackboard.com/>

React Native Documentation:<https://reactnative.dev/>

Node.js Documentation:<https://nodejs.org/>

Firebase Documentation:<https://firebase.google.com/docs>

## Section 2 “Overall description” and sub sections:

### 2.1 Product Perspective:

* CampusConnect is a new, self-contained system that will interface with existing post-secondary educational systems, specifically Centennial College's Learning Management System (Blackboard/Luminate). The system will operate as a standalone mobile and web application while maintaining secure integration points with the college's existing infrastructure for authentication and academic calendar synchronization.

### 2.2 Product Features (Functions):

The CampusConnect platform will provide the following core functionalities:

* **User Authentication:** Secure login through school credentials (@my.centennialcollege.ca)
* **Event Management:** Create, edit, delete, and manage events or club activities
* **Personalized Event Feed:** Customized content based on user interests and program
* **LMS Integration:** Connects to college Learning Management System
* **Academic Schedule Sync:** Integration with school's academic calendar
* **Content Moderation:** IT/administrative oversight and monitoring
* **User Preferences:** Update interests and manage notification settings
* **Multi-Platform Access:** WiFi and mobile data connectivity
* **Calendar Integration:** Sync with external calendar applications
* **Attendance Tracking:** Digital check-in and participation monitoring

### 2.3 User classes and Characteristics:

**Students:** Primary end users who will discover, register for, and participate in campus events and activities. They seek easy navigation and personalized content recommendations.

**Activity Organizers/Club Leaders:** Users responsible for creating and managing events, tracking attendance, and promoting activities to the campus community.

Admin: Admins will oversee the software and will check for legitimacy of activities and manage the accounts.

**Faculty Members:** Educational staff who may organize academic events, workshops, or departmental activities through the platform.

**System Administrators/IT Staff:** Technical personnel who oversee platform security, content moderation, user account management, and system maintenance.

**Administrative Staff:** Dean of Student Affairs, Co-op advisors, and other staff who monitor engagement metrics and support institutional goals.

### 2.4 Operating Environment:

* Will be designed for mobile use, Android and IOS.
* Can be accessed from personal computers and laptops to allow flexibility.
* Will have interaction with map based softwares, i.e, Google maps & Apple maps

### 2.5 Assumptions and Dependencies:

**Assumptions:**

* Centennial College will provide necessary permissions and API access for LMS integration
* Students and faculty have access to mobile devices or computers
* Reliable internet connectivity is available on campus
* Users have valid institutional email addresses

**Dependencies:**

* Approval and cooperation from college administration
* Access to existing student/faculty databases
* Integration permissions with Blackboard/Luminate systems
* Third-party service availability (Google Maps, calendar services)
* Compliance with institutional privacy and security policies

## Section 3: External Interface Requirements

### 3.1 Frontend (User Interfaces):

**Web Application:**

* Cross-browser compatibility: Chrome, Safari, Firefox, Edge
* Responsive design supporting desktop, tablet, and mobile browsers

**Mobile Application:**

* Native iOS and Android applications
* Built with React Native for cross-platform development
* Touch-optimized interface with swipe gestures and mobile-specific features

**Backend (Server-Side Components):**

* **Server Logic:** Node.js with Express framework for handling authentication, data processing, and API requests
* **Database Management:** Firebase (cloud storage) or PostgreSQL for event data, user profiles, and system logs
* **Security:** Email domain restriction to @my.centennialcollege.ca addresses

**Key Interface Features:**

* Login screen with institutional credential authentication
* Personalized event feed with filtering and search capabilities
* Event creation and management forms
* User profile and preference settings
* Calendar integration interface
* Notification management panel

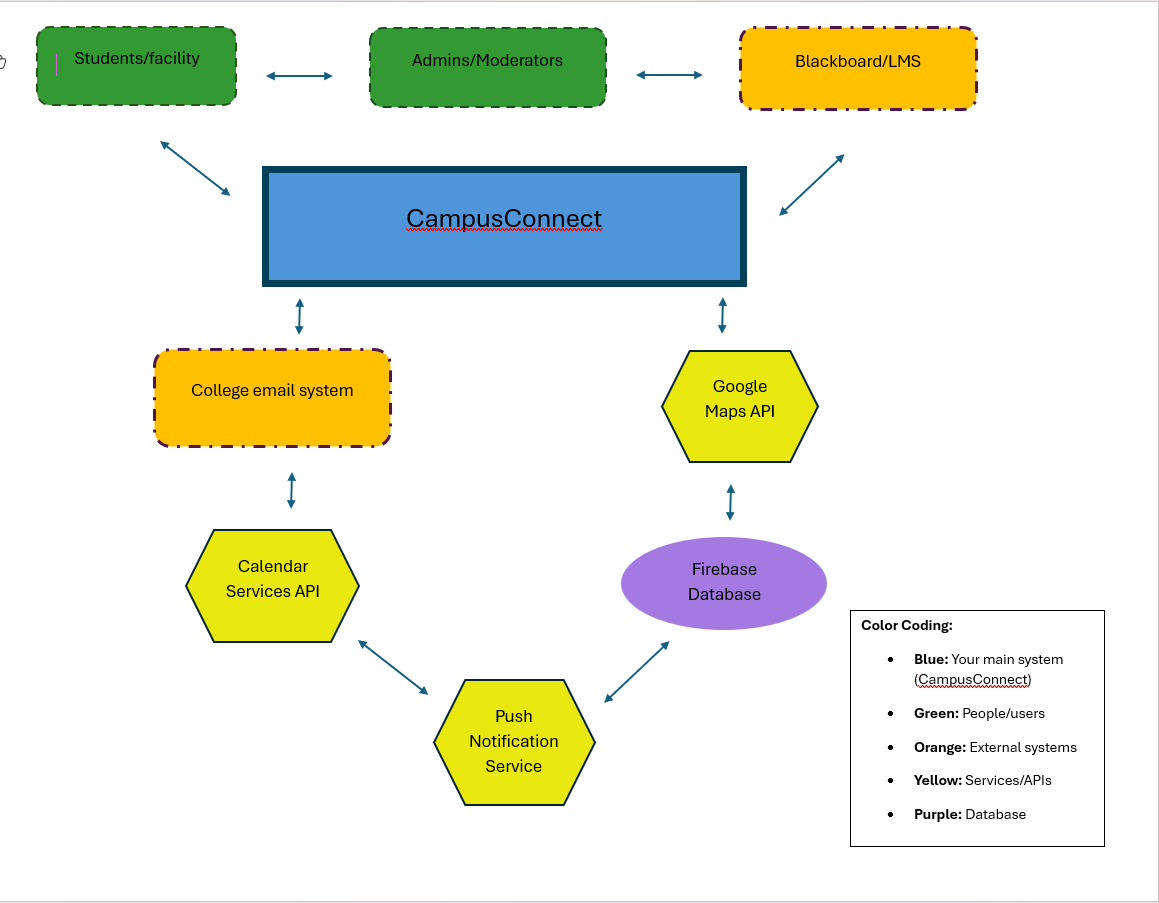
**Additional Features:**

* **Calendar Synchronization:** Automatic event addition to Outlook and Google Calendar
* **Push Notifications:** Event reminders and updates

Why this matters:

- Makes finding campus events and classes easy and fast.

- No more missed deadlines or paper sign-ups

****

### 3.2 Hardware Interfaces:

**Mobile Device Hardware:**

* **Camera:** For QR code scanning during event check-in and profile photo uploads
* **GPS/Location Services:** For location-based event recommendations and navigation
* **Push Notification Hardware:** Device notification systems for event alerts

**Campus Infrastructure:**

* **WiFi Access Points:** For internet connectivity throughout campus
* **QR Code Scanners:** For quick event check-in and attendance tracking
* **Digital Display Interfaces:** For displaying event information on campus screens

**Server Hardware:**

* **Cloud Infrastructure:** Scalable server resources for hosting the application
* **Database Servers:** Secure data storage and retrieval systems
* **Network Interface:** High-speed internet connectivity for real-time synchronization

### 3.3 Software Interfaces

The system will integrate with several third-party software applications and services:

**Calendar Integration:**

* **Google Calendar API:** For users with Google accounts to sync events automatically
* **Microsoft Outlook API:** Integration with college email systems and calendar services
* **Purpose:** Streamline event scheduling and prevent conflicts with academic schedules

**College Systems:**

* **Blackboard/Luminate LMS:** Secure API integration for user authentication and academic calendar synchronization
* **Student Information System:** Access to program information for personalized recommendations
* **College Email System:** Integration for automated notifications and communications

**Communication Services:**

* **Twilio API:** SMS and email notification services with enterprise-grade encryption
* **Purpose:** Optional event reminders and system notifications

**Payment Processing:**

* **Stripe API:** Secure payment gateway for events requiring fees or donations
* **PCI Compliance:** Ensuring secure financial transactions

**Analytics and Monitoring:**

* **Google Analytics:** User engagement tracking and platform usage statistics
* **Purpose:** Data-driven improvements and administrative reporting

**Mapping and Location:**

* **Google Maps API:** Campus navigation and event location services
* **Apple Maps Integration:** iOS-native mapping functionality

## Section 4. Analysis Modeling

### 4.1 Overview Of Modeling

Provides a detailed exploration of the system's functional behavior by modeling how key users interact with CampusConnect. Building on the use cases outlined in Section 3, this section focuses on how user goals are realized through the system's features. It includes a formalized use case description for one of the platform’s high-priority scenarios “*Register Event* “which allows students and faculty to confirm attendance for campus activities. This use case captures essential interactions such as registration, confirmation notifications, and calendar syncing. To complement this, a swimlane activity diagram illustrates the flow of responsibilities between the user and system. These models not only help validate system functionality but also serve as foundational artifacts for development and testing in later stages of the project.

\*We evaluated the Party Analysis Pattern (see Appendix E.5).

### 4.2 Formal Use Case Description – Register Event

This subsection provides a detailed formal description of the “*Register Event”* use case, one of the core functions of the CampusConnect platform. This process enables students and faculty to register for events hosted on campus, such as workshops, social gatherings, academic sessions, or club meetings. The description outlines the necessary conditions, user actions, and system responses involved in completing the event registration flow. It also specifies what happens in case of exceptions, such as when an event reaches full capacity or the user is not logged in. This level of detail is essential for developers, testers, and designers to understand exactly how this functionality should operate under various scenarios.

**Use Case Name:**  *Register for Event*

**ID:** UC01

**Actors:**  Student, Faculty

**Goal:**

To allow a student or faculty member to register for an event on campus through the CampusConnect system. Once registered, the event is added to their calendar, and a confirmation is sent.

**Preconditions:**

* The user is logged in with a verified institutional email (@my.centennialcollege.ca).
* The selected event is open for registration.

**Postconditions:**

* The user is marked as registered.
* A confirmation message is sent via email or push notification.
* The event is added to the user’s calendar.

**Trigger:**

The user selects an event and clicks “Register” on the event page.

**Main Flow:**

1. User logs into the CampusConnect app.
2. User navigates to the event list or searches for a specific event.
3. User clicks on an event to view its details.
4. User clicks the "Register" button.
5. System checks event capacity and registration deadline.
6. If space is available, the system registers the user's information.
7. The system adds the event to the user’s calendar.
8. The system sends a confirmation message to the user.

**Alternate Flow – Event Full:**

* If the event is at full capacity, the system displays a “Registration Closed” message and does not proceed with the registration.

**Exceptions:**

* If the user's session is inactive or expired, they are redirected to the login screen before registration can proceed.

**Priority:**

* High: critical to user engagement with campus events

**Frequency of use:**

* Occasional: based on individual event interest

**Channel to actor:**

* CampusConnect mobile or web app

**Secondary actors:**

* System database, calendar service, email notification system

**Channels to secondary actors**:

* Calendar service: API
* Email notification: internal messaging system

**Open issues:**

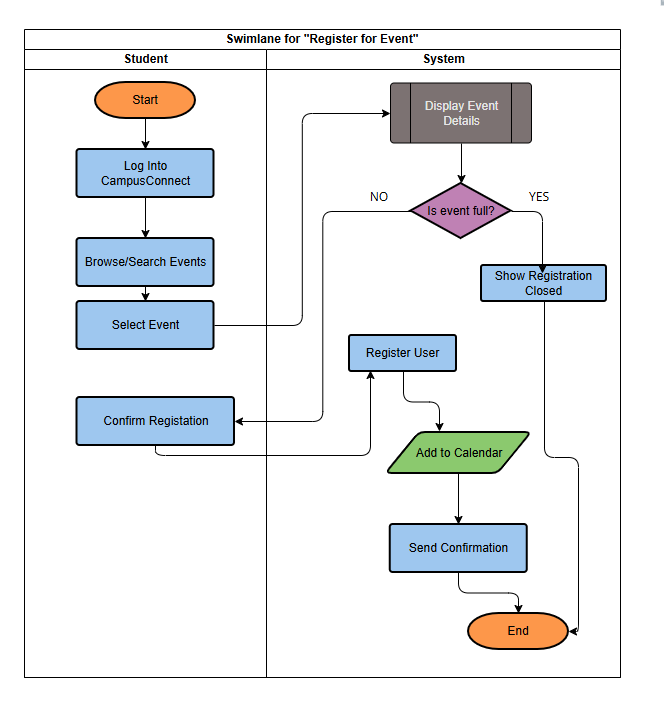
1. Should a waitlist option be available if the event is full?

2. Should students be able to edit their registration after confirmation?

3. Should the map directions be optional or automatic?

### 4.3 Swimlane Diagram – Register Event

The diagram below visually represents interaction between the Student and the System while registering for a campus event. It highlights the user decisions and system responses that take place from login to final registration confirmation or rejection due to full capacity.

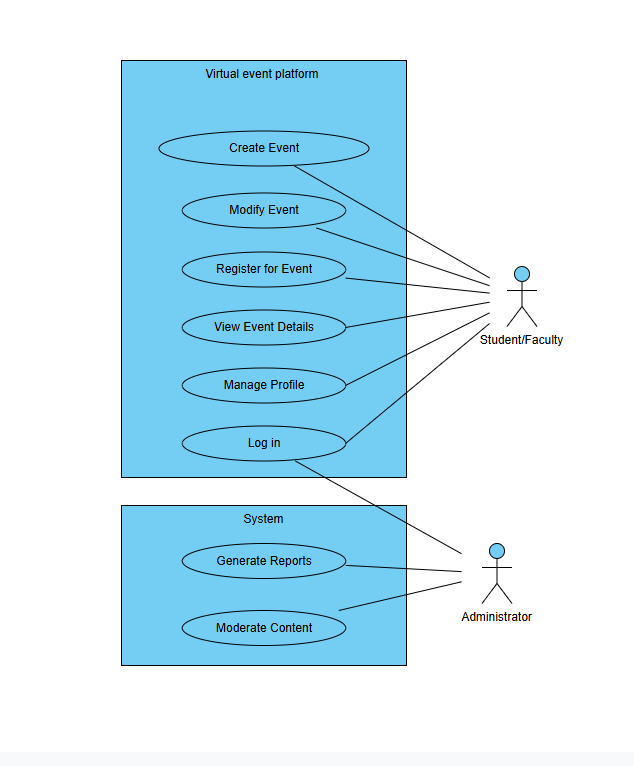


### 4.4 Use Case Table -Subsystem: Event Management

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **List of Related Requirements ID** | **Actor(s)** | **Brief Description** |
| Register for Event | FR04, FR07, FR08 | (Student or Faculty | Actor selects an event and clicks “Register.” System validates registration, sends confirmation, and syncs the calendar. |
| Create Event | FR02, FR08,FR12 | Student or Faculty | Actor enters event details such as title, date, and location. System stores the event and makes it visible to others. |
| Modify Event | FR12 | Student or Faculty | Actor edits an existing event to update date, time, or details. System applies changes and notifies registrants. |
| View Event Details | FR03 | Student or Faculty | Actor views event information like description, time, and location from the event list. |

**Subsystem: Account & System**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **List of Related Requirements ID** | **Actor(s)** | **Brief Description** |
| Generate Report | FR11, FR 12 | Administrator | Actors filter attendance or event data and receive participation reports. Useful for engagement tracking.. |
| Moderate Content | FR10 | Administrator | Admin reviews, approves, or removes events that do not meet school policy. |
| Manage Profile | FR09 | Student & Faculty | Update interests and notification preferences |



## Section 5 - Requirements Lists

### 5.1 Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement ID** | **Requirement Title** | **Short Description** | **Priority** | **Requester** |
| FR01 | User Registration & Authentication | The system should allow students and faculty to register and login using their @my.centennialcollege.ca email addresses with secure authentication. | High | IT Manager |
| FR02 | Event Creation | The system should allow authorized users (students, faculty, club leaders) to create new events with details like title, description, date, time, location, and capacity limits. | High | Club Presidents |
| FR03 | Event Discovery & Search | The system should provide a searchable event feed where users can browse, filter, and search for events based on categories, dates, interests, and keywords. | High | Students |
| FR04 | Event Registration/RSVP | The system should allow users to register for events, manage their RSVPs, and receive confirmation notifications. | High | Students |
| FR05 | Personal Event Feed | The system should display a personalized feed of recommended events based on user interests, program, past attendance, and preferences. | Medium | Students |
| FR06 | QR Code Check-in | The system should generate QR codes for events and allow attendees to check in by scanning the code with their mobile device camera. | Medium | Club Presidents |
| FR07 | Calendar Integration | The system should allow users to sync events with external calendar applications (Google Calendar, Outlook) and export event details. | Medium | Faculty |
| FR08 | Push Notifications | The system should send push notifications for event reminders, updates, cancellations, and new event recommendations. | High | Students |
| FR09 | User Profile Management | The system should allow users to create and update their profiles including interests, program information, and notification preferences. | Medium | Students |
| FR10 | Content Moderation Dashboard | The system should provide administrators with tools to review, approve, reject, or remove event posts and manage user reports. | High | IT Manager |
| FR11 | Attendance Tracking & Reports | The system should track event attendance and generate reports for event organizers showing participation statistics and attendee information. | Medium | Dean of Student Affairs |
| FR12 | Event Management Dashboard | The system should provide event organizers with a dashboard to edit event details, manage RSVPs, send updates to attendees, and cancel events. | High | Club Presidents |

### 5.2 Non-Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement ID** | **Requirement Title** | **Short Description** | **Priority** | **Requester** |
| NFR01 | Mobile Platform Compatibility | The system should be compatible with iOS (version 12+) and Android (version 8+) mobile devices with responsive design. | High | Students, IT Administrator |
| NFR02 | Web Browser Compatibility | The system should be compatible with Chrome, Firefox, Safari, and Edge browsers (latest 2 versions) for web access. | High | Faculty, IT Administrator |
| NFR03 | Performance & Load Handling | The system should handle at least 1000 concurrent users and respond to user requests within 3 seconds under normal load conditions. | High | IT Administrator |
| NFR04 | Data Security & Privacy | The system should implement SSL encryption, secure password storage, and comply with college privacy policies for student data protection. | High | IT Security Officer, IT Manager |
| NFR05 | System Integration Security | The system should securely integrate with Blackboard/Luminate using encrypted APIs without compromising existing system performance. | High | IT Manager, Integration Specialist |
| NFR06 | Offline Functionality | The mobile app should allow users to view previously loaded events and their personal calendar when offline, with data sync when connection is restored. | Medium | Students |
| NFR07 | User Interface Accessibility | The system should meet WCAG 2.1 accessibility standards for users with disabilities, including screen reader compatibility and keyboard navigation. | Medium | Students, Administrative Staff |
| NFR08 | Data Backup & Recovery | The system should automatically backup user data and events daily with ability to restore data within 24 hours in case of system failure. | High | IT Administrator |

## Appendices

### Appendix C - Stakeholders Register

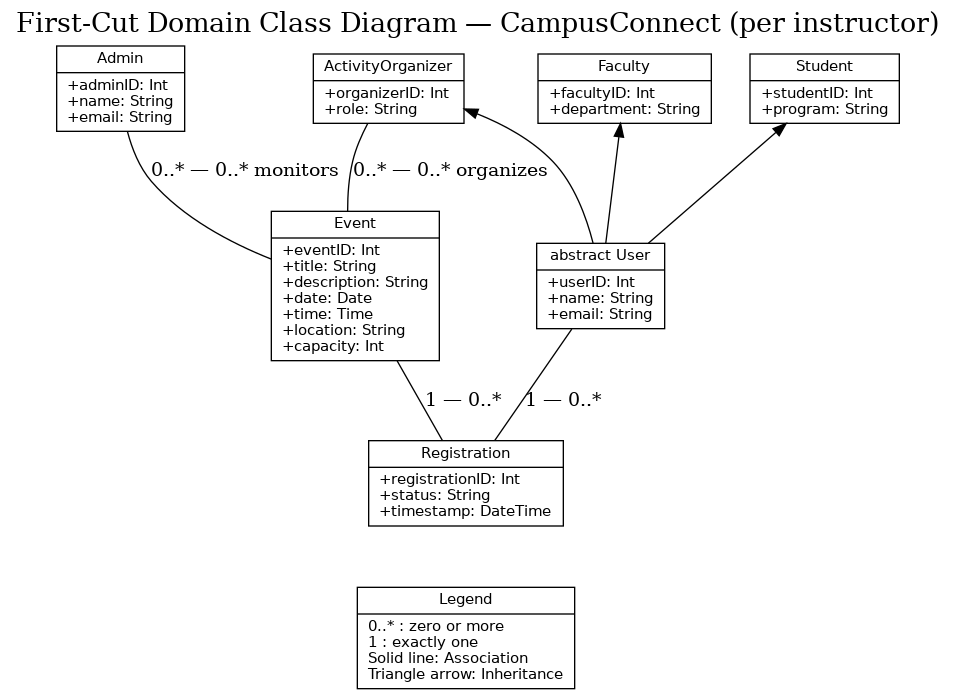
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stakeholders** | **Position** | **External/Internal** | **Contact Details** | **Op/Executive** | **Interest**  **(High, Med, Low)** |
| James Reilly | IT Department Manager | internal | james.reilly@centennialcollege.ca | OP | high |
| Sarah Porben | Faculty member | internal | sarah.porben@centennialcollege.ca | OP | med |
| Emily Thomas | Student | internal | emily.thomas@my.centennialcollege.ca | OP | high |
| Lucas Wright | Club/Event President | internal | lucas.wright@my.centennialcollege.ca | OP | high |
| Maria Lopez | Dean of Student Affairs | internal | maria.lopez@centennialcollege.ca | EX | high |
| Anika Sharma | Alumni Engagement Coordinator | internal | anika.sharma@centennialcollege.ca | OP | low |
| Mark Daniels | Blackboard/Luminate Integration Specialist | external | mark.daniels@blackboardpartners.com | OP | med |
| Priya Nair | Co-op/Career Service Advisor | internal | priya.nair@centennialcollege.ca | OP | med |

### Appendix D: Interview Questions and Answers

|  |  |  |
| --- | --- | --- |
| Stakeholder position | Question | Answer |
| Emily Thomas - Student | How do you prefer to receive updates about events? | Push notifications through the app are ideal because I check my phone more often than my email. |
| Emily Thomas - Student | What challenges do you face when finding campus events? | I usually find out about events too late or miss them entirely because they're not promoted well. |
| Sarah Porben - Faculty | What would help you manage your events more efficiently? | A centralized dashboard to create, edit, and track attendance for events would be very useful. |
| Sarah Porben - Faculty | Do you think syncing academic calendars with event schedules would be useful? | Yes, it would help avoid scheduling conflicts with classes or exams. |
| James Reilly - IT Manager | What are your biggest concerns regarding the platform’s implementation? | Security and content moderation—everything needs to align with school policies and privacy standards. |
| James Reilly - IT Manager | Are you comfortable with integration to Blackboard and Luminate? | Yes, provided it's done securely and doesn’t affect system performance. |
| Lucas Wright -Club | What features would help you promote and manage your club events? | Event creation, RSVP tracking, and automatic reminders would make managing events easier. |
| Lucas Wright - Club | How do you currently track attendance, and what would improve that process? | Manual headcounts. A QR check-in or digital attendance tracker would be better. |
| Maria Lopez - Dean of Student Affairs | How does this platform align with your department’s strategic goals? | It supports our goals for student engagement and co-curricular participation. |
| Maria Lopez - Dean of Student Affairs | What metrics would you want to see from this platform? | Student participation rates, event popularity, and club growth trends. |
| Priya Nair - Co-op & Careers | Would it benefit your office to track student involvement in clubs and events? | Definitely. It gives us data for resumes, co-op applications, and references. |
| Priya Nair - Co-op & Careers | Should students be able to export their involvement records? | Yes, a downloadable participation report would be very useful. |
| Ankia Sharma - Alumni Coordinator | Do you see potential for alumni involvement in the future? | Yes, especially for mentorship programs and alumni-led workshops or events. |
| Ankia Sharma - Alumni Coordinator | Would a separate alumni portal in the app be useful down the line? | Yes. It would keep them connected to campus life and support long-term engagement. |
| Mark Daniels - Integration Specialist | What are the technical requirements for integrating with Blackboard? | Use secure RESTful APIs, maintain authentication protocols, and protect user data. |
| Mark Daniels - Integration Specialist | How can we ensure smooth and secure syncing with academic data? | Establish a staging environment for testing, and follow data access best practices. |

### Appendix E:

#### E.1 First-Cut Domain Class

****

#### E.2 CRC Cards

|  |  |
| --- | --- |
| **Class: Student** | |
| **Description:** A student user who can discover, register for, and organize campus events. | |
| **Responsibility:** | **Collaborator:** |
| Manage own/personal Account | Event (for attending only) |
| Browse/search/register for events | Registration |
| Check into events |  |

|  |  |
| --- | --- |
| **Class: Faculty** | |
| **Description:** A faculty user who can attend or organize campus academic events. | |
| **Responsibility:** | **Collaborator:** |
| Manage own/personal Account | Event (for attending only) |
| View/browse and join Events | Registration |
| Check into events |  |

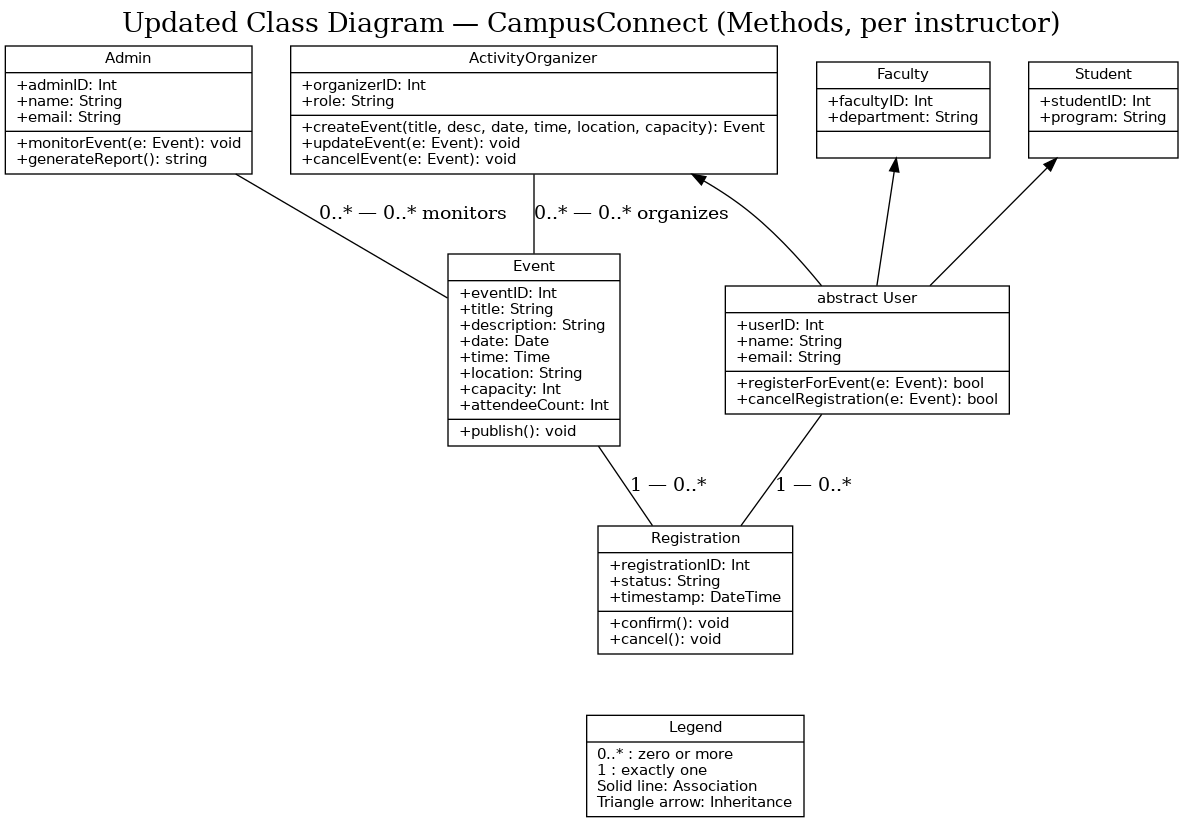
|  |  |
| --- | --- |
| **Class: Activity Organizer** | |
| **Description**: A user (student or faculty acting in organizer role) who creates/manages events | |
| **Responsibility:** | **Collaborator:** |
| Create Event | Event |
| Update event details |  |
| Publish event |  |
| Cancel Event |  |

| **Class: Event** | |
| --- | --- |
| **Description:** A campus event created and managed through CampusConnect. | |
| **Responsibility:** | **Collaborator:** |
| Contains event details and information | Registration |
| Track availability and participants | Faculty or Student |
| Generate a QR code for check-in | Admin |
| Log attendance records | Organizer |

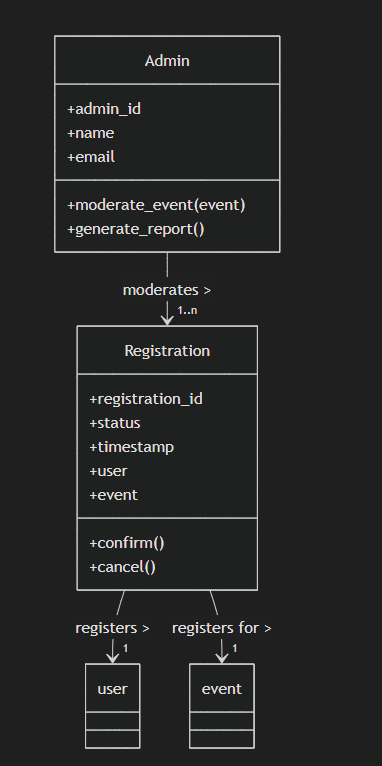
|  |  |
| --- | --- |
| **Class: Administrator** | |
| **Description:** System administrator or IT staff member who oversees content moderation and platform settings. | |
| **Responsibility:** | **Collaborator:** |
| Approve/reject event submissions |  |
| Configure notification rules | Event |
| Generate engagement and attendance reports |  |
| Maintain system integration settings |  |

|  |  |
| --- | --- |
| **Class: Registration** | |
| **Description:** Links one participant to one event and tracks registration status. | |
| **Responsibilities:** | **Collaborators:** |
| Create registration for event | student |
| Confirm/ waitlist/ reject | Event |
| Cancel registration | faculty |
| Mark Checked in or no show |  |

#### E.3 Updated Class Diagram (Methods)

****

#### E.4 Gen AI class diagram

****

**from datetime import datetime**

**class ActivityOrganizer:**

**def create\_event(self, title, desc, date, time, location, capacity):**

**return Event(title, desc, date, time, location, capacity)**

**def cancel\_event(self, event\_id):**

**return f"Event {event\_id} cancelled"**

**class Student(ActivityOrganizer):**

**def \_\_init\_\_(self, student\_id, name, email, program):**

**self.student\_id = student\_id**

**self.name = name**

**self.email = email**

**self.program = program**

**def register\_event(self, event):**

**return Registration(self, event)**

**def checkin(self, event\_id):**

**return f"Student {self.name} checked in to event {event\_id}"**

**class Faculty(ActivityOrganizer):**

**def \_\_init\_\_(self, faculty\_id, name, email, department):**

**self.faculty\_id = faculty\_id**

**self.name = name**

**self.email = email**

**self.department = department**

**def register\_event(self, event):**

**return Registration(self, event)**

**def checkin(self, event\_id):**

**return f"Faculty {self.name} checked in to event {event\_id}"**

**class Event:**

**def \_\_init\_\_(self, title, description, date, time, location, capacity):**

**self.event\_id = hash(title + str(date))**

**self.title = title**

**self.description = description**

**self.date = date**

**self.time = time**

**self.location = location**

**self.capacity = capacity**

**self.attendees = []**

**def get\_spot(self):**

**return self.capacity - len(self.attendees)**

**def add\_attendee(self, user):**

**if self.get\_spot() > 0:**

**self.attendees.append(user)**

**return True**

**return False**

**class Admin:**

**def \_\_init\_\_(self, admin\_id, name, email):**

**self.admin\_id = admin\_id**

**self.name = name**

**self.email = email**

**def moderate\_event(self, event):**

**return f"Event '{event.title}' moderated by {self.name}"**

**def generate\_report(self):**

**return "Engagement report generated"**

**class Registration:**

**def \_\_init\_\_(self, user, event):**

**self.registration\_id = f"{user.name}\_{event.event\_id}"**

**self.status = "Pending"**

**self.timestamp = datetime.now()**

**self.user = user**

**self.event = event**

**def confirm(self):**

**if self.event.add\_attendee(self.user):**

**self.status = "Confirmed"**

**else:**

**self.status = "Rejected"**

**def cancel(self):**

**self.status = "Cancelled"**

#### E.5 Party Analysis Pattern Integration

**About the Party Analysis Pattern:**

The Party Analysis Pattern, introduced by Martin Fowler (1996), provides a reusable object model for representing any person or organization involved in a system. Instead of modeling separate entities for each type of participant (e.g., Student, Faculty, Alumni, Club), the pattern introduces a single, generalized Party class with shared attributes such as partyID, name, and contactInformation.

Specific participant types become subtypes of Party (e.g., Person and Organization), allowing the system to handle all participants through one consistent structure while still enabling role-specific attributes.

**Why It Would Not Work for CampusConnect:**

After evaluating the Party Analysis Pattern for our CampusConnect system, we determined it would introduce more complications than benefits in our case:

1. **Different Authentication Systems**
   * In CampusConnect, students, faculty, and administrators log in using separate institutional email formats and authentication rules.
   * A single Party structure would require complex conditional logic for login, validation, and permissions. This would undermine the simplicity and security of having role-specific authentication models.
2. **Distinct Permissions and Roles**
   * Students, faculty, and administrators perform fundamentally different actions in the system:
     + Students primarily register for events.
     + Faculty may organize events or moderate content.
     + Administrators approve/reject events and manage platform settings.
   * With the Party pattern, permissions would be harder to manage because the base class would not naturally enforce these role-specific behaviors without excessive role-checking code.
3. **Unnecessary Complexity for Small, Fixed Role Sets**
   * The Party pattern shines when there are many participant types that can change over time (e.g., CRM systems, e-commerce platforms).
   * CampusConnect has only three fixed participant types (Student, Faculty, Admin), and these types are unlikely to change or require merging into a single shared workflow. A simple separate-class model is easier to implement and maintain.
4. **Risk of Over-Generalization**
   * Over-generalizing participants into one Party table/class could cause confusion in the database and code, especially for functions like attendance tracking, notifications, and event creation.
   * We would have to repeatedly filter and cast Party objects back to their specific type before performing actions, which increases the risk of errors.

**Conclusion:**

The Party Analysis Pattern is a powerful and reusable modeling approach for systems where participants share most attributes and behaviors. However, in the context of CampusConnect:

* The email domain and authentication system are role-specific, not shared.
* The permissions and workflows differ greatly between Students, Faculty, and Administrators.
* The role set is small and fixed, making the overhead of a generalized Party model unnecessary.

For these reasons, implementing the Party Analysis Pattern would reduce clarity, complicate authentication and permissions, and add avoidable complexity to the system’s design. We will retain separate Student, Faculty, and Administrator classes in our model.

#### E.6 State Diagram - Event

Represents campus events going through several life cycle stages.

### Event States (proposed):

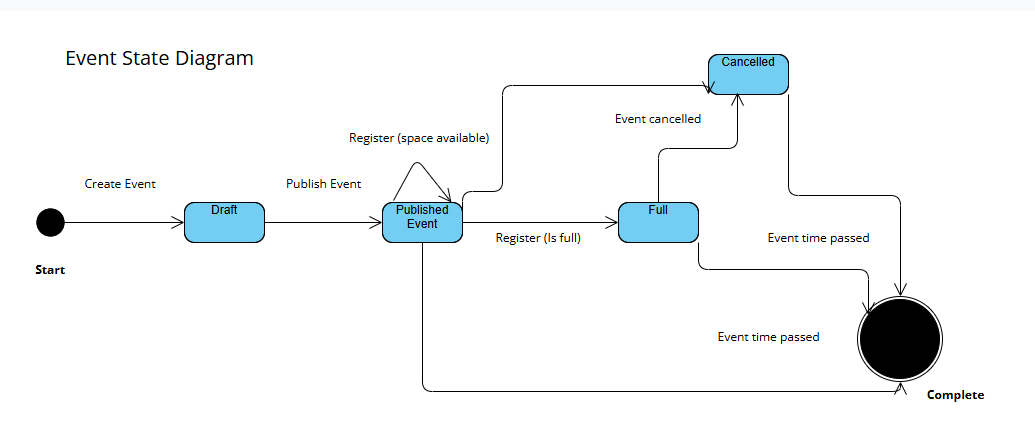
* Draft – Event is created but not yet visible to users.
* Published – Event has been approved and is now visible to students/faculty.
* Full – Event has reached its maximum capacity; registration is closed.
* Cancelled – Event is withdrawn by the organizer or an administrator.
* Completed – The event has taken place (past its scheduled date).

### Triggers (System/User Actions):

* Create Event – Moves event into Draft state.
* Publish Event – Moves event from Draft to Published (possibly pending admin approval).
* Registration Attempt (System Checks) – If the event reaches its capacity - transition to Full.
* Cancel Event – Can move an event from Published or Full to Cancelled.
* Event Date Passes – Time-based system transition - moves from Published or Full to Completed.

### State Transitions:

* Create Event → Draft
* Publish Event → Published
* [capacity reached] → Full
* Cancel Event → Cancelled
* [event date passes] → Completed



#### E.7 State Diagram - Registration

Captures the lifecycle of a user’s registration for an event.

Registration States (proposed):

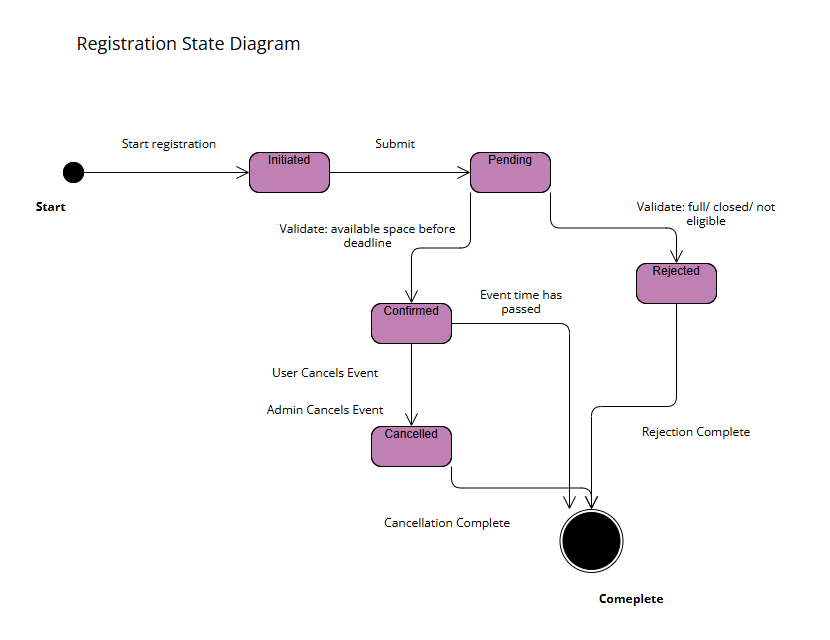
* Initiated – User starts the registration process (clicked *Register*).
* Pending – System is validating capacity, deadline, and user eligibility.
* Confirmed – Seat reserved; confirmation sent and calendar updated.
* Rejected – Registration denied (event full, past deadline, or not eligible).
* Cancelled – Registration revoked by user or admin (seat released if before deadline).
* Completed – Registration lifecycle ends (event time passed or cancellation finalized).

Triggers (System/User Actions):

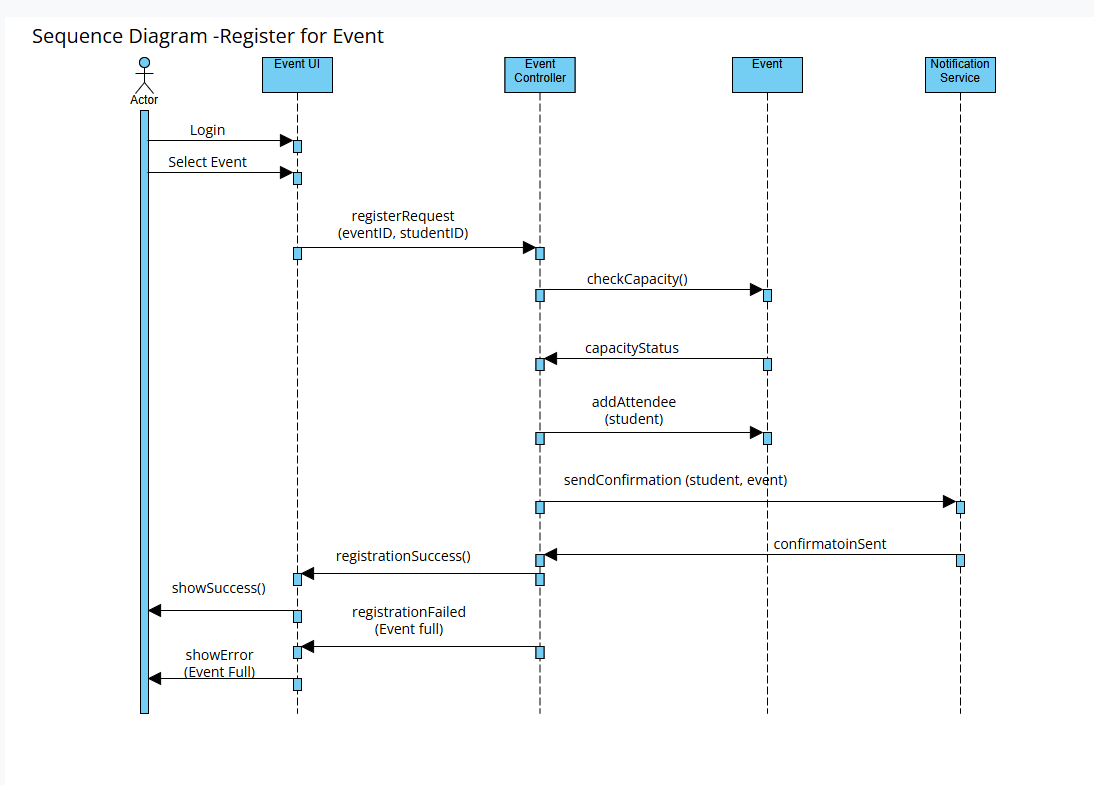
* Start Registration – Moves to *Initiated*.
* Submit – Moves from *Initiated* to *Pending*.
* Validate – From *Pending*, decides *Confirmed* (space available & before deadline) or *Rejected* (full/closed/not eligible).
* User Cancel – From *Confirmed* to *Cancelled* (before deadline).
* Admin Cancel Event – From *Confirmed* to *Cancelled*.
* Event Time Passes – Moves *Confirmed* to *Completed* (no further actions).

State Transitions:

* Start Registration → Initiated
* Initiated → Pending *(Submit)*
* Pending → Confirmed *(Validate: space available & before deadline)*
* Pending → Rejected *(Validate: full/closed/not eligible)*
* Confirmed → Cancelled *(User Cancel or Admin Cancel Event)*
* Confirmed → Completed *(Event time passes)*
* Cancelled → Completed *(cancellation finalized)*



#### E.8 Sequence Diagram - Register for Event



The sequence diagram above illustrates the interactions between the Student, Event UI, EventController, Event, and NotificationService during the event registration process.  
 The flow begins when the Student selects an event and clicks "Register." The Event UI sends a registration request to the EventController, which checks event capacity through the Event object.

If capacity is available, the EventController registers the student, updates the event attendance list, and calls the NotificationService to send a confirmation. If the event is full, the system returns a failure message to the Student. This diagram also shows an alternate flow for full capacity events.

#### E.9 Gen AI Sequence Diagram

