Faculty of Computing and Informatics (FCI)

Multimedia University Cyberjaya

CSN6244 – Software Requirement Engineering

Trimester 2510

Group Number: G09

Campus Event Check-in System with Student ID and Payment Integration

Tutorial Session: TT1L

Group Members:

NAME	ID	EMAIL(MICROSOFT)
Tang Wei Xiong	1211112069	1211112069@student.mmu.edu.my
Liew Wei Hong	1211108896	1211108896@student.mmu.edu.my
Ng Kean Ping	242UC2451V	NG.KEAN.PING@student.mmu.edu.my
Chan Mei Yi	242UC2451U	CHAN.MEI.YI@student.mmu.edu.my

Table Contents

1.0 Introduction

1.1 Purpose

The project's purpose of this Software Requirements Specification (SRS) document is to clearly define and outline the complete functional and non-functional requirements for the Campus Event Check-in System with Student ID and Payment Integration. The system is designed to eliminate the inefficiencies of manual check-in methods by integrating with the Multimedia University's student identification database and online payment gateway.

Currently, event registration and check-in at MMU are mostly using physical attendance lists, manual verification and paper tickets, which might cause some issues such as time-consuming, human error, data loss and inefficiency. Furthermore, there is no integration between student authentication systems and payment methods. This might cause friction and delay in event participation, especially for those large-scale or ticketed events.

This project proposes a full solution on creating a platform where students can use their MMU ID to log in, browse upcoming events, register instantly and check in using a QR code generated by the system. Additionally, for paid events, the platform will include secure and convenient online payment functionality that integrates with trusted third-party gateways. Event organizers will also benefit from a management dashboard to create events, monitor registrations and validate check-ins in real time.

By developing this system, MMU is able to improve campus engagement, reduce the workload on administrative staff and modernize the way student events are conducted. This SRS serves as a contractual basis between the development team and stakeholders to ensure the system meets the defined expectations and is developed efficiently within scope.

1.2 Scope

The Campus Event Check-in System with Student ID and Payment Integration is a centralized digital platform, which is designed to streamline the management of campus events at MMU. This system will address the limitations of MMU's current manual processes such as physical attendance sheets, printed tickets and disconnected payment methods, by introducing a centralized digital solution. It will serve both event participants and event organizers by offering role-specific functionalities for ease of access and management. This platform will include the following key features and functions:

In-Scope Features:

- **User Registration and Authentication**:
    - Login using official MMU credentials with role-based access (Student, Organizer, Admin).
    - Role-based interfaces and permissions for streamlined workflows.
    - **Event Management Tools**:
- Dashboards for creating, editing, and scheduling events.
    - Capacity planning with automatic waitlist generation for full events.
    - Support for various ticketing models (free, paid, tiered, early bird).
- **Digital Ticketing System**:
    - Online reservation and ticket purchasing integrated into the platform.
    - Generation of QR or barcode-based digital tickets.
    - Real-time ticket validation using scanning devices at entry points.
- **Check-in and Attendance Tracking**:
    - Contactless check-in via QR codes, NFC tags, or student ID scans.
    - Real-time attendance logging and entry/exit timestamps.
    - Metrics available for participation tracking and analysis.
- **Student ID System Integration**:
    - Verification of student status, faculty, and program using MMU records.
    - Read-only access to ensure compliance with privacy policies.
- **Secure Payment Integration**:
    - Payments supported via credit/debit cards and mobile wallets.
    - Integration with university finance systems for reconciliation.
    - E-receipts and transaction history accessible to users.
- **Notifications and Communications**:
    - Automatic emails for confirmations, reminders, and event updates.
    - Optional post-event follow-ups with feedback forms or thank-you notes.
- **Administrative Dashboard**:
    - Real-time and historical views of check-in activity, attendance, and revenue.
    - Exportable reports for audits and internal use.
    - Aggregated metrics for event analytics and demographic insights.

Out-of-Scope Features:
The following functionalities are not part of the initial system release:

- **Mobile App (Native)**: No standalone Android or iOS app; system is web-responsive only.
- **Offline Support**: Internet connection is required for check-in and other real-time operations.
- **Facial Recognition**: No biometric verification; only student ID-based authentication is supported.
- **Physical Access Control**: No integration with hardware-based systems like RFID gates or turnstiles.
- **Advanced Analytics**: No AI-driven analytics or predictive models included in this version.
- **Multi-University Integration**: System is exclusive to MMU; no cross-institution access or support.
- **External Event Hosting**: Only events organized by MMU departments or approved student bodies are supported.
- **Public or Anonymous Attendance**: Participation is limited to authenticated MMU students only.
- **Third-party Marketing Tools**: No integrations with external promotional or social media platforms.
- **Refund or Chargeback Handling**: All refund processes are managed externally by the integrated payment service.
- **Student ID Record Editing**: The system has read-only access to student data; no direct modification is allowed.

1.3 Product Overview

The Campus Event Check-in System with Student ID and Payment Integration is a web platform developed to streamline event registration, ticketing and attendance tracking in Multimedia University. It has a centralised system that allows students to log in the platform using their student ID, register for events and check in using system-generated QR code. The platform can also integrate secure online payment options for the paid events.

Additionally, event organizers are able to create and manage events, track attendance in real time and generate reports through a dedicated dashboard. The system integrates with MMU's student database for trustworthy verification, maintains data security, and enforces role-based access. This solution modernizes the campus event experience while increasing efficiency and lowering administrative effort.

1.3.1 Product Perspective

The Campus Event Check-in System acts as a web-based platform under MMU’s student services infrastructure. The platform can also interact with various institutional components to provide a centralized and efficient event management experience.

The system relies on integration with the Student Information System (SIS) to authenticate users using MMU-issued student credentials, this ensures only verified users gain access. It also accesses read-only student data such as name, student ID, program and faculty, to support personalized user interaction and enforce access control.

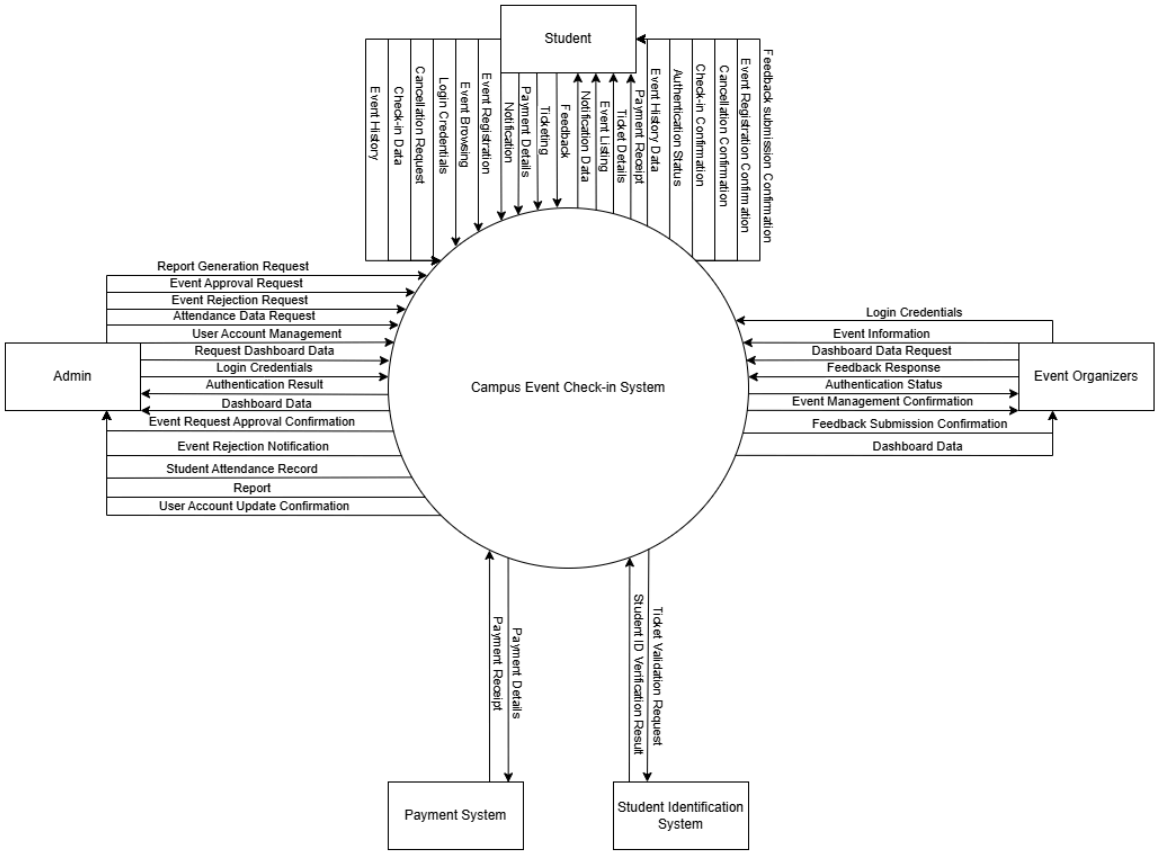
The system handles ticket purchases by establishing a connection with a secure third-party payment source. Without directly keeping sensitive financial information, the system will record transaction status and show receipts for user reference whenever any transactions take place using these services.

Additionally, the system supports email using notification services. This ensures that users receive real-time updates regarding registration, ticket confirmations and event changes.

Although the web-based platform is intended to be used independently, it adheres to defined protocols and APIs to provide scalability and possible future integration with other MMU platforms, such the attendance analytics tools or the campus mobile app.

In conclusion, the system provides a safe, effective and intuitive solution by filling in the gaps between user administration, payment processing and attendance monitoring. Thus, improving the university's event engagement procedure.

Below is the context diagram showing on how the system interacts with users and other system:



1.3.2 Product Function

The Campus Event Check-in System provides a set of core functionalities customized to simplify the event management within the university. The primary functions of the platform is listed below:

- **User Authentication**
  - Allows students, organizers and administrators to log in using official MMU credentials.
  - Role-based access ensures users only access features relevant to their role.
- **Event Discovery and Registration**
  - Students can browse a list of upcoming events with the description.
  - Users can register or purchase tickets directly through the system.
- **Digital Ticketing**
  - Automatically generates unique QR code tickets upon registration or payment.
  - Supports free and paid ticket models with tiered pricing options.
- **Secure Payment Integration**
  - Provides seamless payment via third-party gateways.
  - Generates digital receipts and logs transaction history.
- **Real-time Check-in**
  - Enables event check-in using QR code scanning, with real-time validation and attendance logging.
- **Event Management Dashboard (Organizers)**
  - Organizers can create, edit, and manage event details, set capacities and monitor ticket sales.
  - Real-time data on registration numbers and attendance rates can be shown.
- **Administrative Dashboard (Admin)**
  - Admins can monitor platform-wide activity, generate reports and manage user roles and permissions.
  - Provides analytics on event participation, ticket revenue and student engagement.
- **Notification and Communication System**
  - Sends automated email reminders, event updates, and post-event feedback requests.

1.3.3 User Characteristics

The users of this system are primarily members of the Multimedia University (MMU) community. They are grouped into the following categories:

Users	**Technical Skill Level**	**Usage Context**	**Frequency of Use**
Students	Moderate (Familiar with web-based platform and mobile technologies)	Students will use the system to log in using their MMU credentials, browse upcoming campus events, register or purchase tickets and check in using QR codes.	Occasional, depends on the event interest
Event Organizers	Moderate/Advanced (Familiar with event planning tools)	Organizers will use the dashboard to create, manage and monitor events, including registration tracking, ticketing and real-time check-in validation.	Regular, usually during the event planning and execution
Administrators	Advanced (Have access to the full system for CRUD activities)	Admins oversee all system activities, manage user roles, generate institutional reports and ensure system compliance with university policies.	Consistent, especially during the event sessions or the reporting period

1.3.4 Limitation

The Campus Event Check-in System is designed with certain constraints that may affect its functionality and performance under specific conditions:

- **Web-Only Access**: The system is limited to a responsive web application. There are no native mobile applications for Android or iOS platforms in the current version.
- **Internet Dependency**: An active internet connection is required for most operations, including user authentication, event registration, check-ins and payment processing. The system does not support offline check-in functionality.
- **Single-Institution Use**: The platform is built exclusively for MMU and does not support multi-university or cross-institution integrations.
- **No Biometric Verification**: The system uses student ID-based login for identity verification. Advanced features like facial recognition or fingerprint scanning are not supported.
- **Limited Analytics**: The current version provides only basic reporting and metrics. It does not include AI-powered analytics or predictive attendance models.
- **Third-Party Dependency for Payments**: All payment transactions rely on third-party gateways. The system does not directly process refunds or chargebacks, and such matters must be handled through the payment provider.

1.4 Definition

Here are some commonly used terms and language units with explanations:

Terms	Definition
Graphical User Interface	A visual part of the application that users interact with
User	Any individual that interacts with, accesses or utilizes a product, service, application or system to perform tasks or achieve specific outcomes.
Admin	Administrator – a high-privilege system user with oversight and control over system-wide settings and data.
QR Code	Quick Response Code – a type of matrix barcode used to verify event tickets at check-in.
Organizer	A user role responsible for creating and managing campus events within the system.
Student ID	Official identification credentials issued by MMU to students, used for authentication.
Payment Gateway	A third-party service provider used to process online transactions.

2. Reference

Google Firebase Docs. *Firebase Authentication*. Retrieved from **https://firebase.google.com/docs/auth**

Kano model example - A case study in customer satisfaction. (2025). Kanosurveys.com. **https://www.kanosurveys.com/articles/kano-model-example**

TouchNet. (2024, February 13). *How ID management systems enhance campus security*. Retrieved from **https://www.touchnet.com/trends/blog/2024/02/13/how-id-management-systems-enhance-campus-security**

Rahman, M., & Chowdhury, A. (2024). *Enhancing security and efficiency in mobile payment systems: An integrated approach utilizing advanced technologies*. Retrieved from **https://www.researchgate.net/publication/384245337**

Liew, W. H. (2025). Campus Event Check-in System with Student ID and Payment Integration (Brainstorming and Interview). Retrieved from **https://drive.google.com/drive/folders/1unHF53jnDAT37DPvJ3QvnnNbKCASs9b4**

Liew, W. H. (2025). Campus Event Check-in System with Student ID and Payment Integration (Questionnaire). Retrieved from **https://docs.google.com/forms/d/e/1FAIpQLScmchzmOKNcu6oJ41ZxJPR6xqKcNBB4_xHSFDU7hRQXyMaQ_g/viewform?usp=dialog**

3. Requirements

3.1 Functions

3.1.1 Use case diagram

This use case diagram shows the main functions of the Campus Event Check-In System and how users interact with it. There are three actors: Student, Event Organizer, and Admin. Each has specific roles such as managing events, registering for events, and handling user accounts. The system also connects with two external systems: the Payment System for processing payments and the Student Identification System for validating identities and recording attendance.

Actor	Use Case
Student	Login account, Browse event, Register for event, Check-in to event, Cancel registration for event, View notification, View ticket, View event history, Submit feedback
Event Organizer	Login account, View dashboard, Respond to feedback, Manage event(Create, Edit, Cancel)
Admin	Login account, View dashboard, Manage user account, View student attendance, Generate report, Manage event request(Reject, Approve)
Payment System	Make Payment online
Student Identification System	Record attendance, Validate ticket

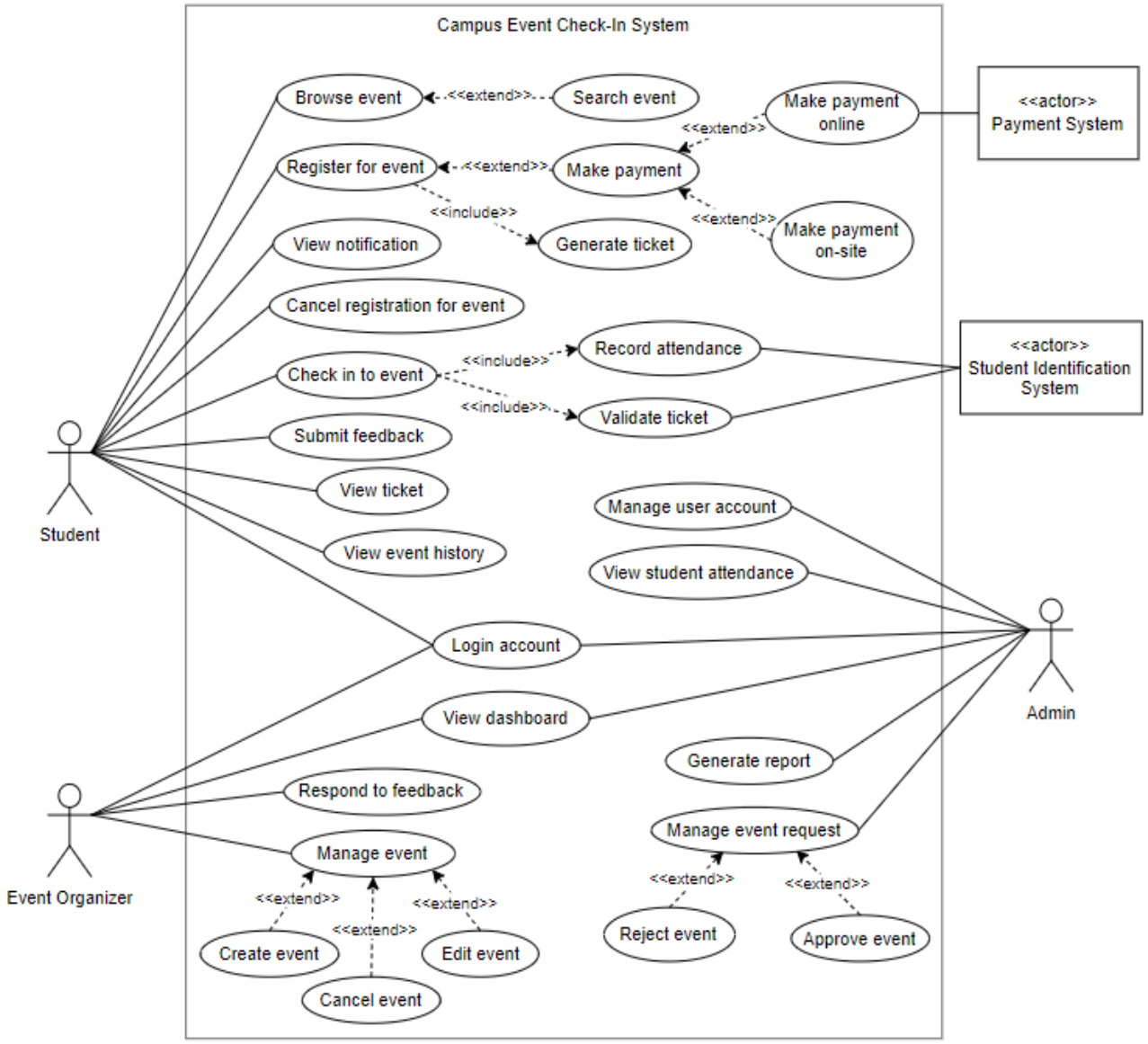


Figure 3.1: Use Case Diagram

3.1.2 Sequence Diagram

3.1.2.1 Login account

Use Case Name:	Login account
Description:	This use case allows a registered user (Student, Event Organizer, or Admin) to log in using valid credentials. The system authenticates the credentials and redirects the user to their respective dashboard based on their role.
Primary Actor:	User includes (Student, Event Organizer, Admin)
Precondition:	The user must already have a registered account.
Postcondition:	If the login is successful, the user is redirected to their respective dashboard. If the login fails, an error message is displayed and the user remains on the login page.
Main Flow:	1.User navigates to the login page. 2.User enter their username and password. 3.Login Page sends the login details to the Authentication System. 4.Authentication System checks credentials with the User Database. 5.If credentials are valid, the system returns the user's role and grants access.
Alternate Flow:	**Invalid Login:** 1. User enters incorrect username or password. 2. Authentication fails and system returns an error message. 3. User is prompted to re-enter credentials.

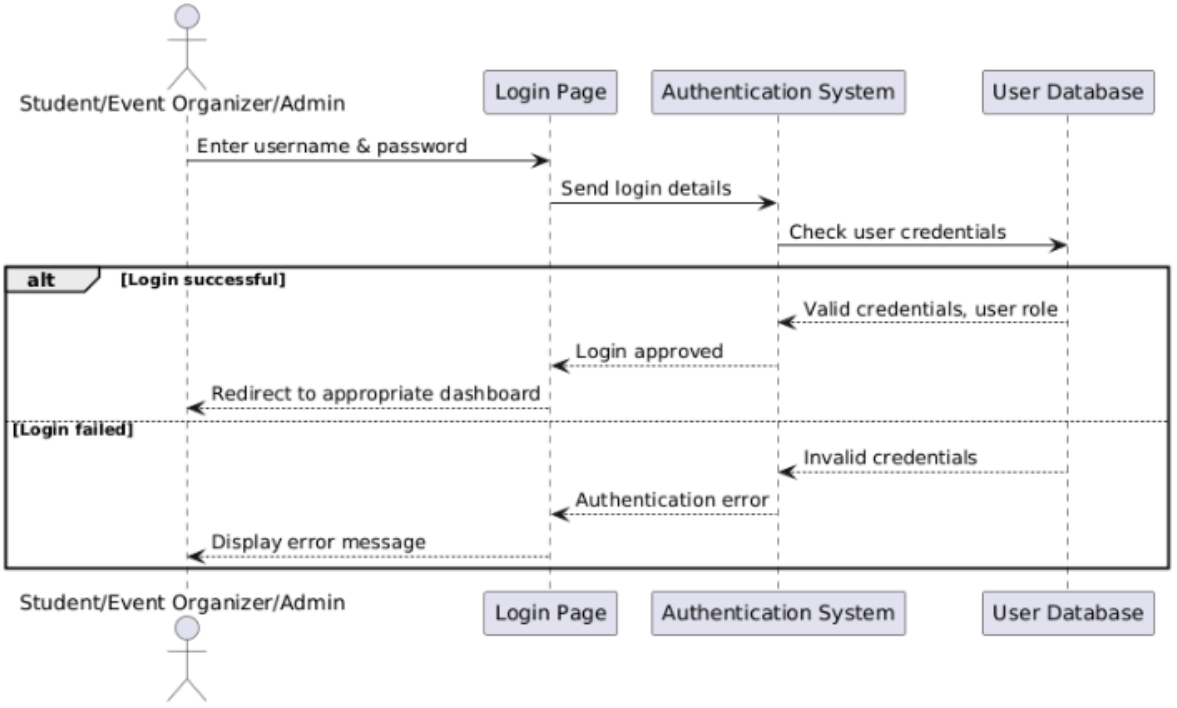


Figure 3.1.2.1: Login account (sequence diagram)

3.1.2.2 Register for event

Use Case Name:	Register for event
Description:	This use case allows a student to register for an event. The system checks event availability. If available, the student submits a registration form.If the event requires payment, the system extends the process to handle online payment via an external payment system. Upon successful payment (if required), the system records the registration and generates a ticket.
Primary Actor:	Student
External Actor:	Payment system
Precondition:	The student is logged into the system. The event is open for registration.
Postcondition:	The student is successfully registered, and a ticket is generated. If payment is required, it must be completed before confirmation.
Main Flow:	1. Student selects an event and clicks “Register.” 2. System checks event availability. 3. If available, the system shows the registration form. 4. Student submits the form. 5.If payment is required, the system redirects the student to the external Payment System. 6. Upon successful payment, registration is recorded. 7. Ticket is generated and sent to student.
Alternate Flow:	1.If the event is full or unavailable, the system displays an error message and prevents registration. 2.If payment fails, the system prompts the student to retry or select another payment method.

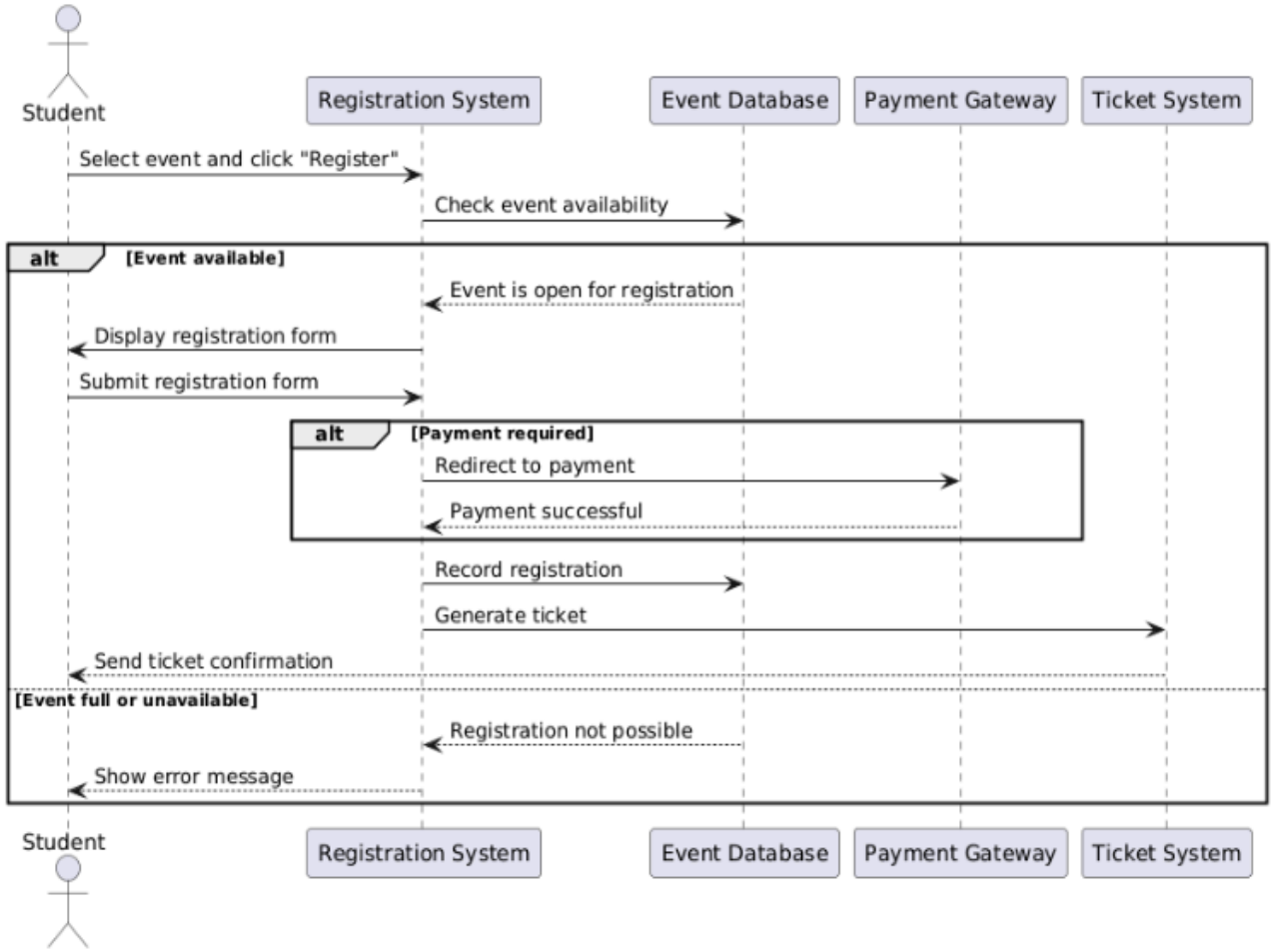


Figure 3.1.2.2: Register for event (sequence diagram)

3.1.2.3 Browse event

Use Case Name:	Browse Event
Description:	This use case allows a student to browse all available events. The student can optionally refine the list by searching events using keywords or filters.
Primary Actor:	Student
Precondition:	Student is logged into the system. Events are available in the system.
Postcondition:	Student successfully views the event list or the filtered event list after searching.
Main Flow:	1. Student selects "Browse Event". 2. System displays the list of all available events. 3. Student optionally selects "Search Event" and enters keywords or filters. 4. System filters the event list according to search criteria. 5. System displays filtered event results.
Alternate Flow:	1. If no events are available, system displays a message indicating no events found. 2. If search yields no matching events, system displays a "no results found" message.

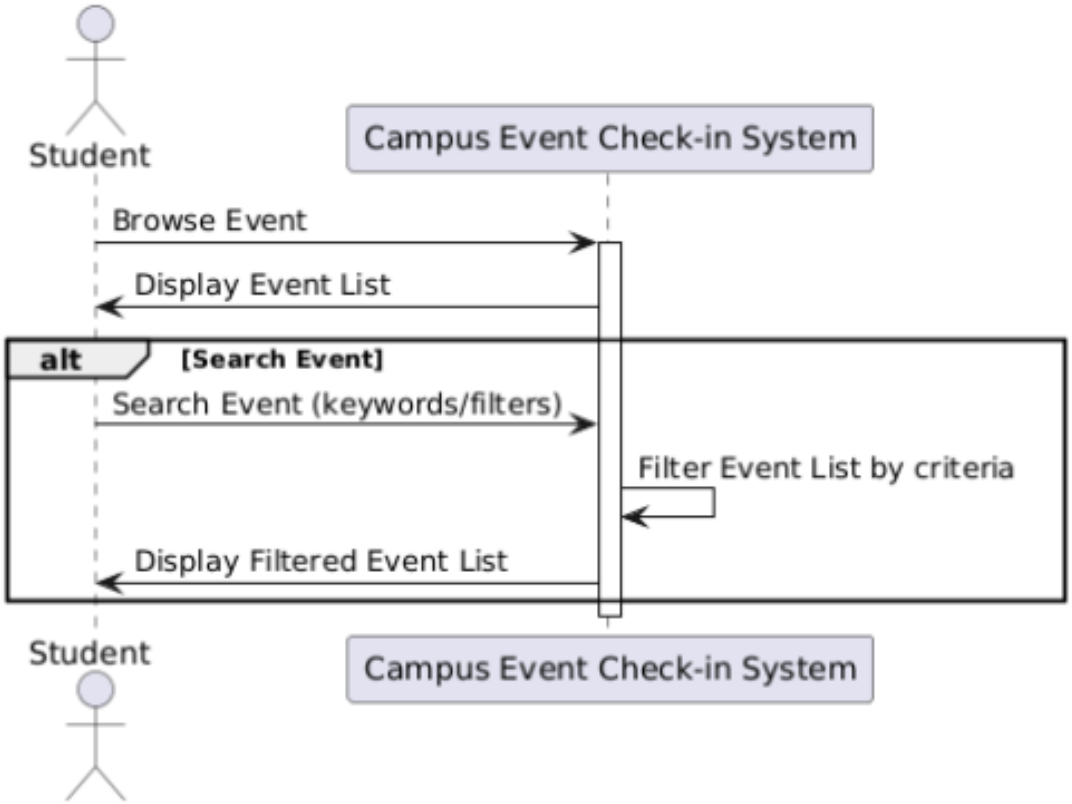


Figure 3.1.2.3: Browse Event (sequence diagram)

3.1.2.4 View notification

Use Case Name:	View notification
Description:	Student views notifications sent by the system, such as event updates or messages.
Primary Actor:	Student
Precondition:	Student is logged into the system. Notifications exist for the student.
Postcondition:	Student has seen notifications or a message indicating no notifications are available.
Main Flow:	1. Student selects the "View Notifications" option. 2. System shows the list of notifications. 3. Student reads selected notification.
Alternate Flow:	1. If there are no notifications, system displays a message indicating "No notification available."

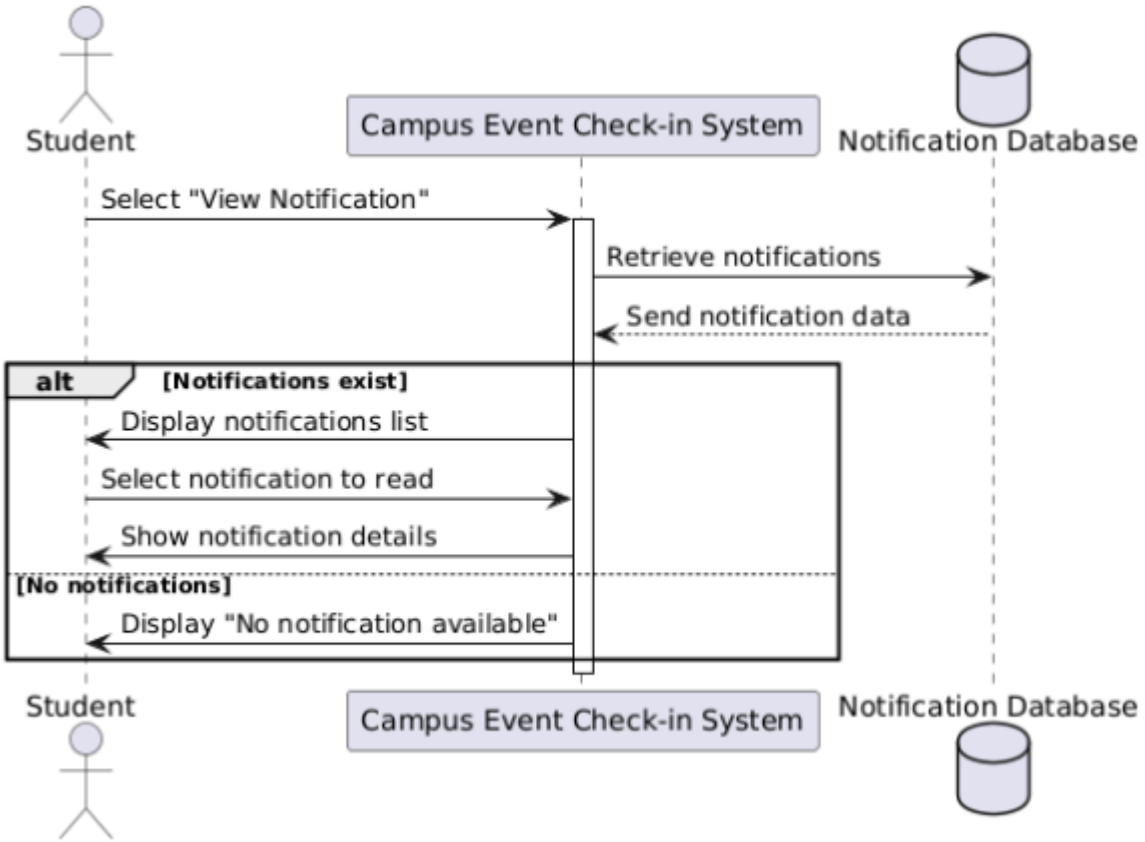


Figure 3.1.2.4: View notification (sequence diagram)

3.1.2.5 Cancel registration for event

Use Case Name:	Cancel registration for event
Description:	The student requests to cancel their event registration. The system verifies if the registration exists, asks for confirmation, and updates the database. If successful, the student is notified.
Primary Actor:	Student
Precondition:	The student must be logged in. The event registration must exist in the system.
Postcondition:	The registration is successfully canceled, and the database is updated.
Main Flow:	1.Student selects "Cancel Registration" for an event. 2.System checks if the registration exists in the database. 3.If registration is found, the system prompts the student for confirmation. 4.Student confirms cancellation. 5.System updates the database to cancel registration. 6.System notifies the student of the cancellation success.
Alternate Flow:	If no registration is found, the system displays an error message: "No registration found."

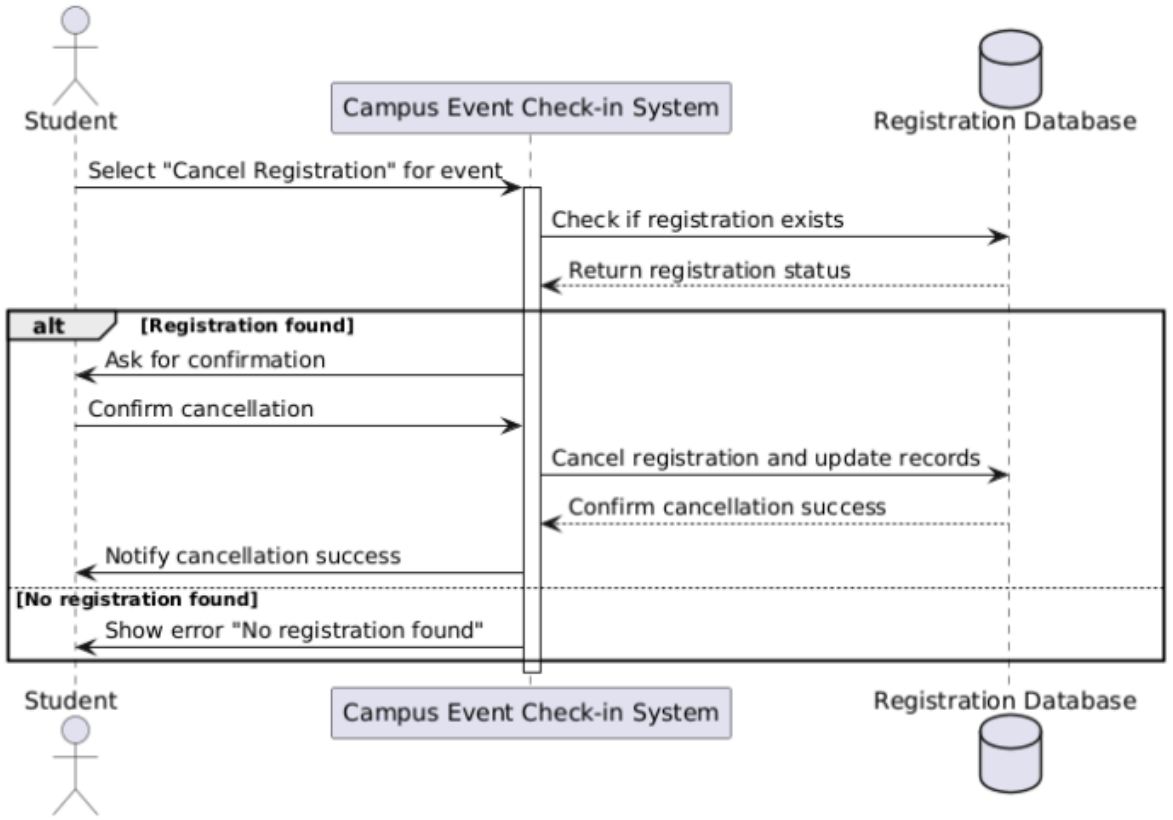


Figure 3.1.2.5: Cancel registration for event (sequence diagram)

3.1.2.6 Check in to event

Use Case Name:	Check in to event
Description:	This use case allows a student to check in to an event using contactless methods such as QR code scan, NFC tag, or student ID. The system validates the ticket using the Student Identification System. If valid, it logs the attendance with real-time timestamp.
Primary Actor:	Student
External Actor:	Student Identification System
Precondition:	1.Student has a valid event ticket. 2.The event is currently open for check-in. 3.The student is logged into the system or has their ID/ticket ready.
Postcondition:	1.Student is marked as present in the attendance system. 2.Entry timestamp is logged. 3.Ticket is validated successfully.
Main Flow:	1. Student scans a QR code, taps NFC, or presents student ID at the event entrance. 2. System sends the ticket or ID data to the Student Identification System for validation. 3. Student Identification System verifies ticket validity. 4. If valid, the system logs attendance with current timestamp. 5. System displays a check-in confirmation to the student.
Alternate Flow:	**Invalid ticket or ID** - The Student Identification System returns an invalid status. - The system denies check-in and displays an error message to the student.

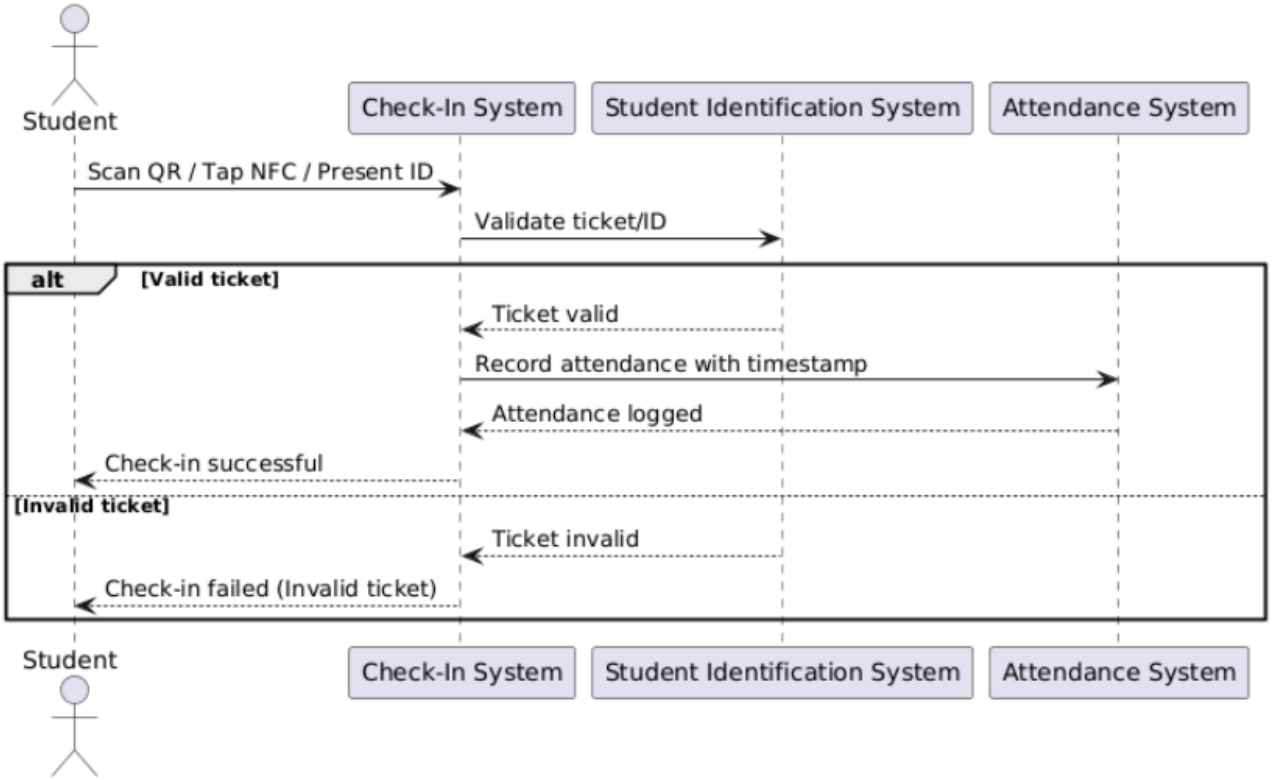


Figure 3.1.2.6: Check in to event (sequence diagram)

3.1.2.7 Submit feedback

Use Case Name:	Submit feedback
Description:	This use case allows a student to submit feedback for a past event they attended. The system stores the feedback and makes it viewable by the event organizer.
Primary Actor:	Student
Precondition:	The student must be logged in and have attended the event..
Postcondition:	Feedback is successfully recorded and stored in the system.
Main Flow:	1. Student selects a past event. 2. Student writes and submits feedback. 3. System saves the feedback.
Alternate Flow:	If student has not attended the event, the system prevents feedback submission.

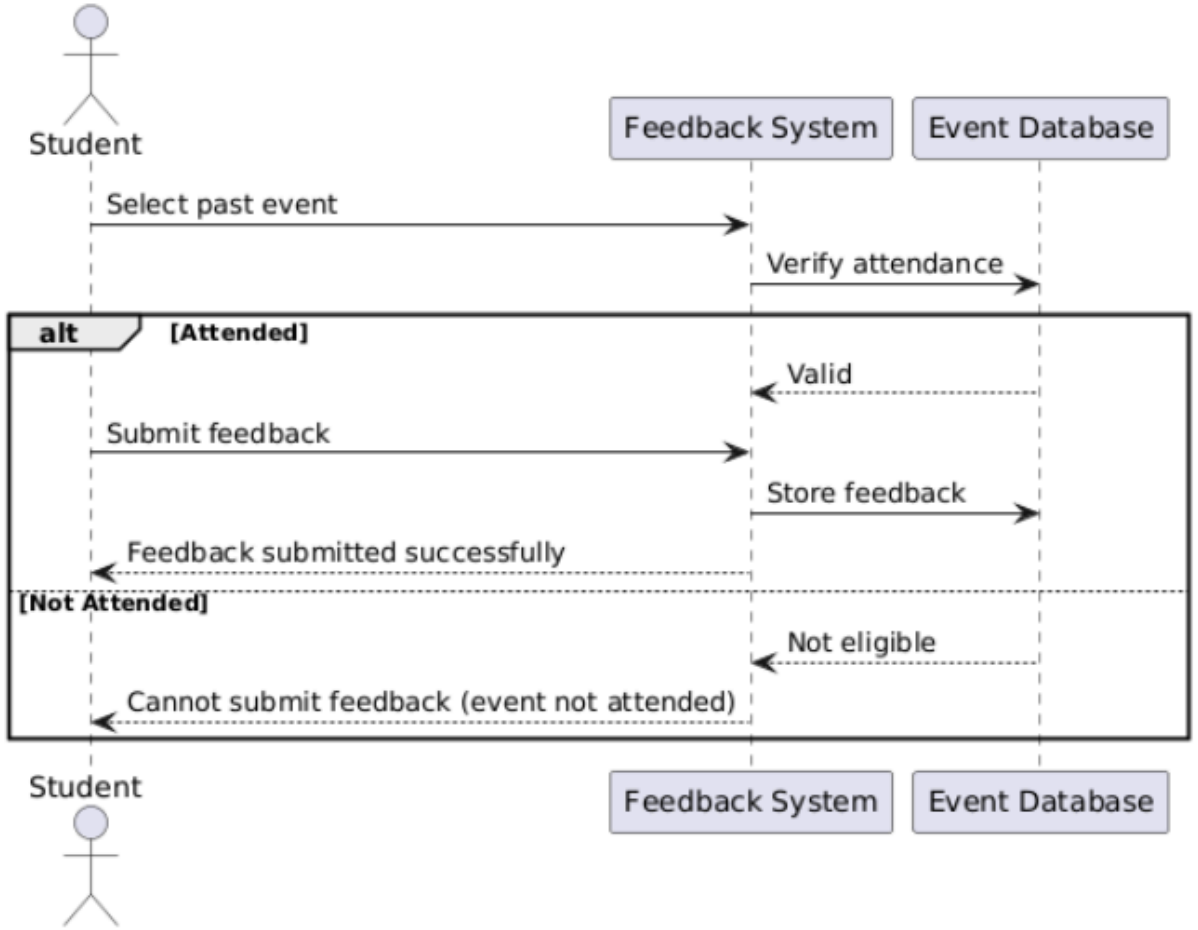


Figure 3.1.2.7: Submit feedback (sequence diagram)

3.1.2.8 View ticket

Use Case Name:	View ticket
Description:	Allows a student to view their event ticket after successful registration.
Primary Actor:	Student
Precondition:	Student must be logged in and have a valid ticket generated for an event.
Postcondition:	The ticket details are displayed to the student.
Main Flow:	1.The student selects the "View Ticket" option. 2.The system verifies the student's identity and retrieves the ticket information. 3.The system displays the event ticket details to the student.
Alternate Flow:	If no ticket is found, the system displays a message: "No ticket available for this event."

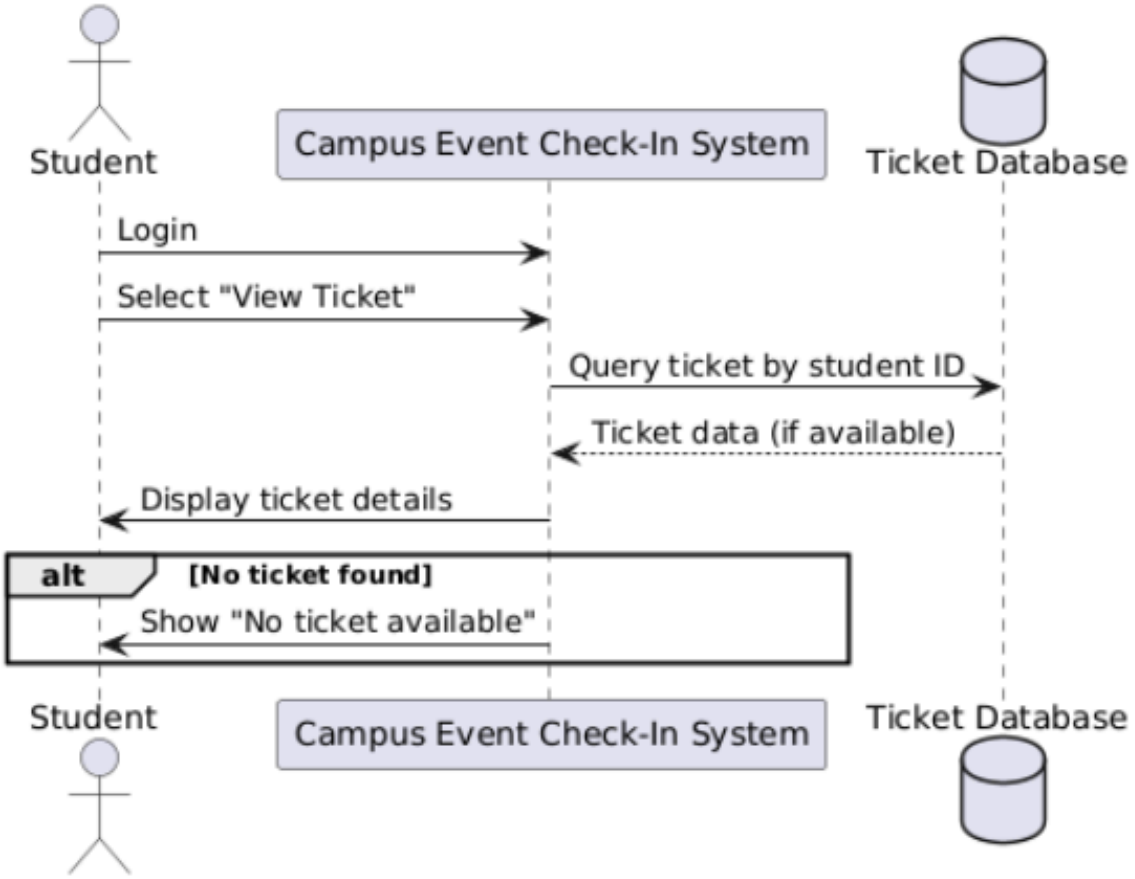


Figure 3.1.2.8: View ticket (sequence diagram)

3.1.2.9 View event history

Use Case Name:	View event history
Description:	This use case allows a student to access and review a list of all events they have previously registered for or attended. It helps students keep track of their event participation history within the system.
Primary Actor:	Student
Precondition:	The student must be logged in to the system with a valid account and have previously participated in at least one event.
Postcondition:	The system displays a list of previously attended or registered events.
Main Flow:	1.The student logs into the system. 2.The student selects the "View Event History" option. 3.The system queries the database for all past events associated with the student's ID. 4.The system displays a list of previous events with relevant details.
Alternate Flow:	If the student has not participated in any events, the system shows a message stating: "No event history available."

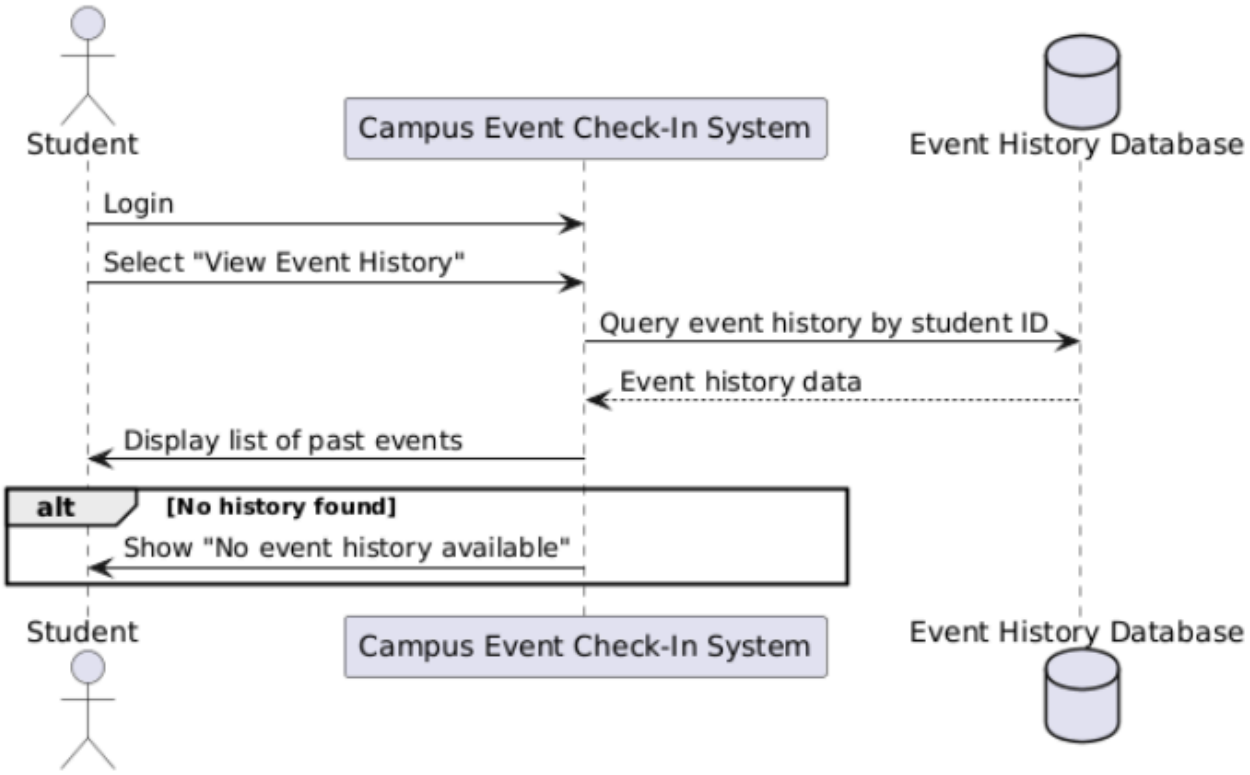


Figure 3.1.2.9: View event history (sequence diagram)

3.1.2.10 View dashboard

Use Case Name:	View dashboard
Description:	This use case allows Event Organizers and Admins to view a dashboard summarizing key system data. Event Organizers can view metrics such as the number of events created, upcoming event dates, and total registrations. Admins can access broader system summaries like total users, system activity logs, and event submission trends.
Primary Actor:	Event Organizer, Admin
Precondition:	The actor must be logged in with the appropriate role (Event Organizer or Admin).
Postcondition:	The system displays a dashboard with role-specific summaries and statistics
Main Flow:	1.The user logs into the system. 2.The user selects "View Dashboard" from the main menu. 3.The system checks the user's role (Organizer or Admin). 4.The system queries relevant summary data from the database (e.g., event count, user activity). 5.The system displays the dashboard with graphs, tables, or KPIs tailored to the user’s role.
Alternate Flow:	If the database fails to fetch summary data, the system displays a message indicating unavailable or missing information.

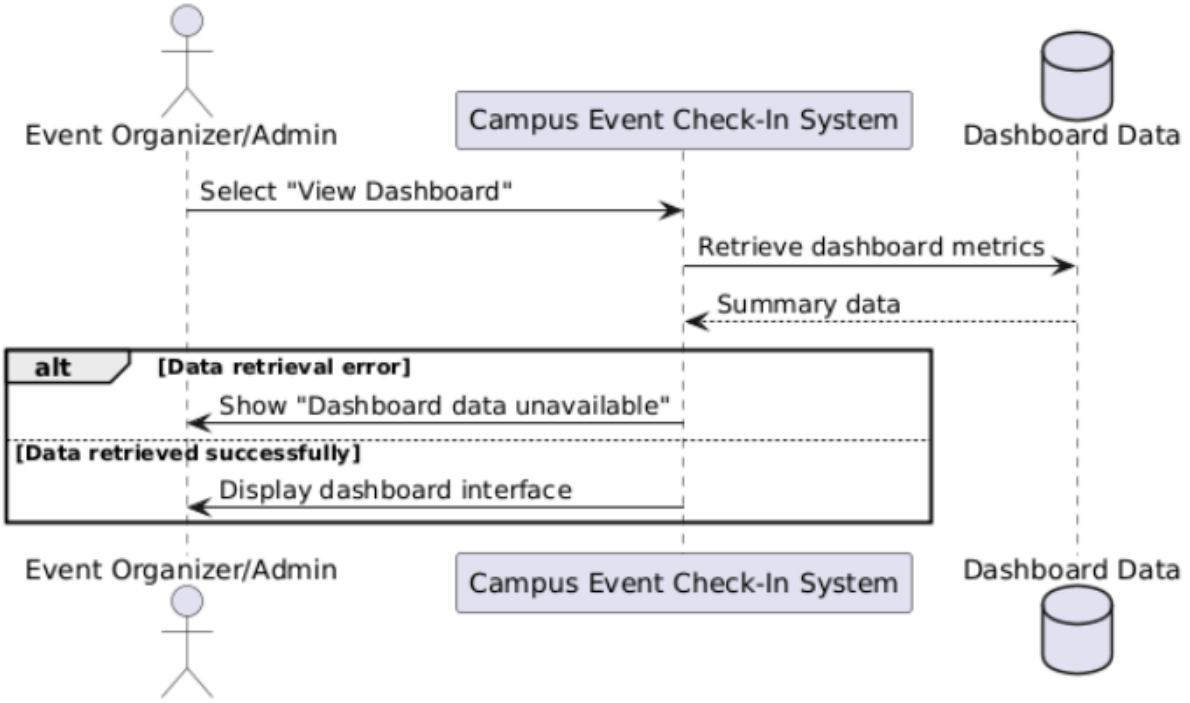


Figure 3.1.2.10: View dashboard (sequence diagram)

3.1.2.11 Respond to feedback

Use Case Name:	Respond to feedback
Description:	The Event Organizer reviews feedback submitted by students about campus events and responds to them.
Primary Actor:	Event Organizer
Precondition:	Students have submitted feedback.
Postcondition:	A response from the Event Organizer is recorded and sent to the student.
Main Flow:	1. Event Organizer select specific feedback. 2. Event Organizer writes and submit a response. 3. System saves response and notifies student.
Alternate Flow:	If the database fails to retrieve student feedback, the system displays an error message and prompts the organizer to retry later.

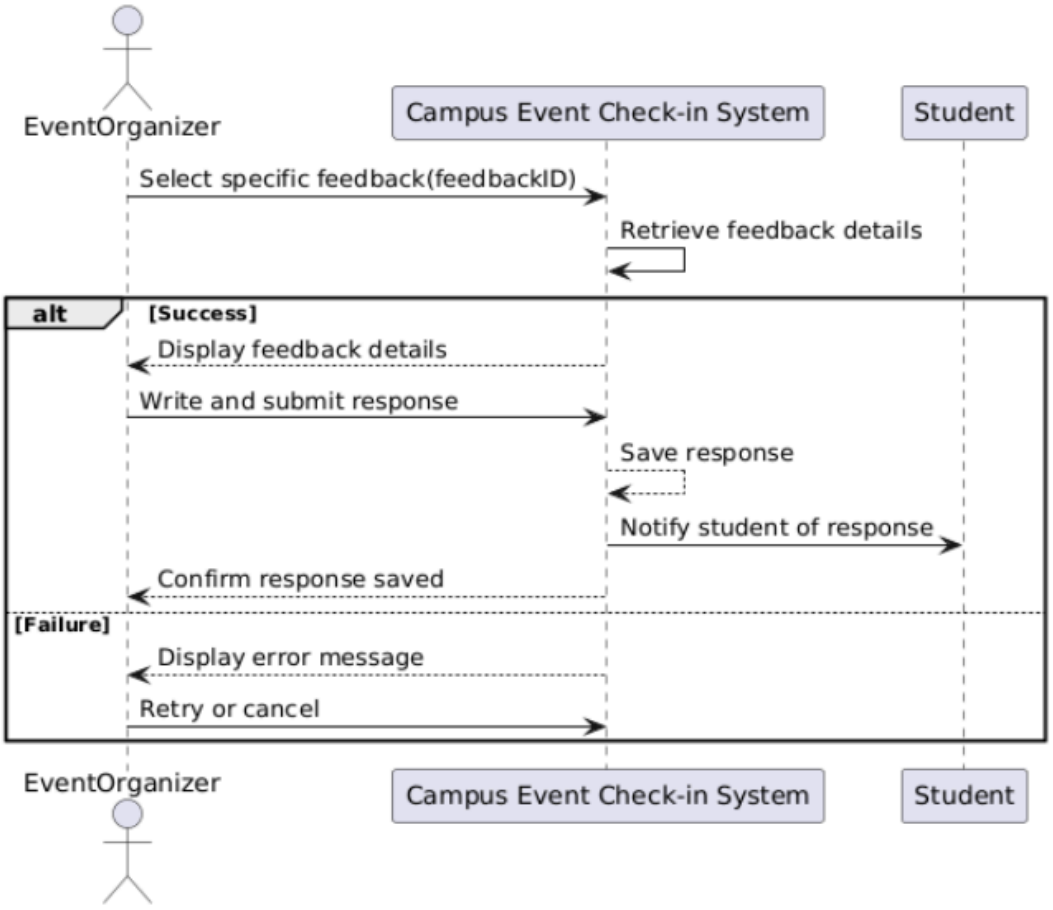


Figure 3.1.2.11: Respond to feedback (sequence diagram)

3.1.2.12 Manage event

Use Case Name:	Manage event
Description:	The Event Organizer creates, updates, or deletes events within the system to manage campus event details.
Primary Actor:	Event Organizer
Precondition:	Event Organizer is logged into the system.
Postcondition:	Event information is created, updated, or deleted accordingly in the system.
Main Flow:	1. Event Organizer logs into the system and navigates to event management. 2. Selects action: create, update, or delete event. 3. Inputs or modifies event details and submit changes. 4. System saves changes and confirms.
Alternate Flow:	If input validation fails, system shows error and prompts re-entry.

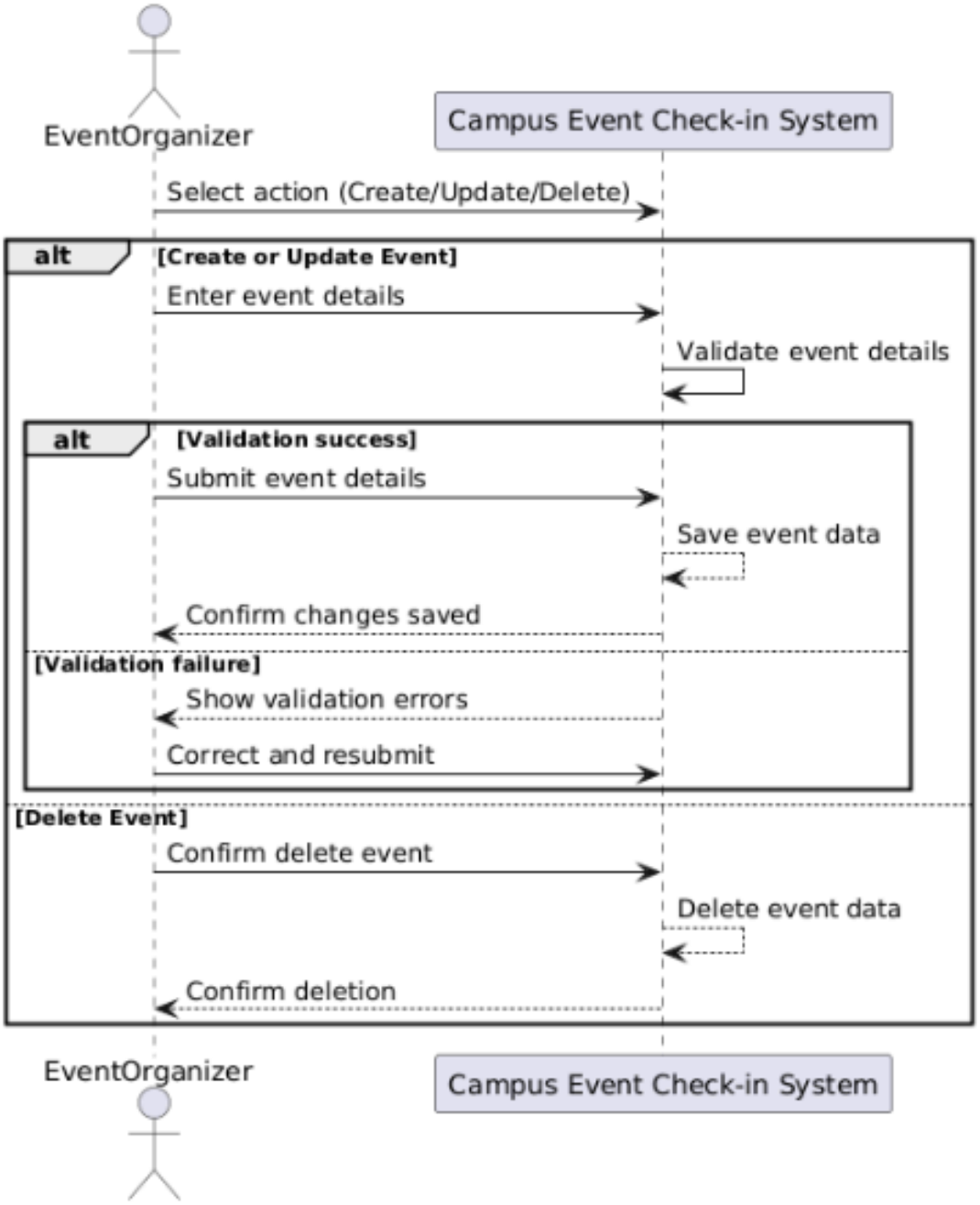


Figure 3.1.2.12: Manage event (sequence diagram)

3.1.2.13 Manage User Account

Use Case Name:	Manage User Account
Description:	The Admin blocks or unblocks user accounts to control access to the event system.
Primary Actor:	Admin
Precondition:	Admin is logged into the system.
Postcondition:	User account access is blocked or restored accordingly.
Main Flow:	1. Admin selects user account management. 2. Admin views list of users. 3. Admin selects a user and chooses to block or unblock the account. 4. System updates account status and confirms.
Alternate Flow:	If the update fails, the system displays an error message and prompts the admin to retry or cancel the operation.

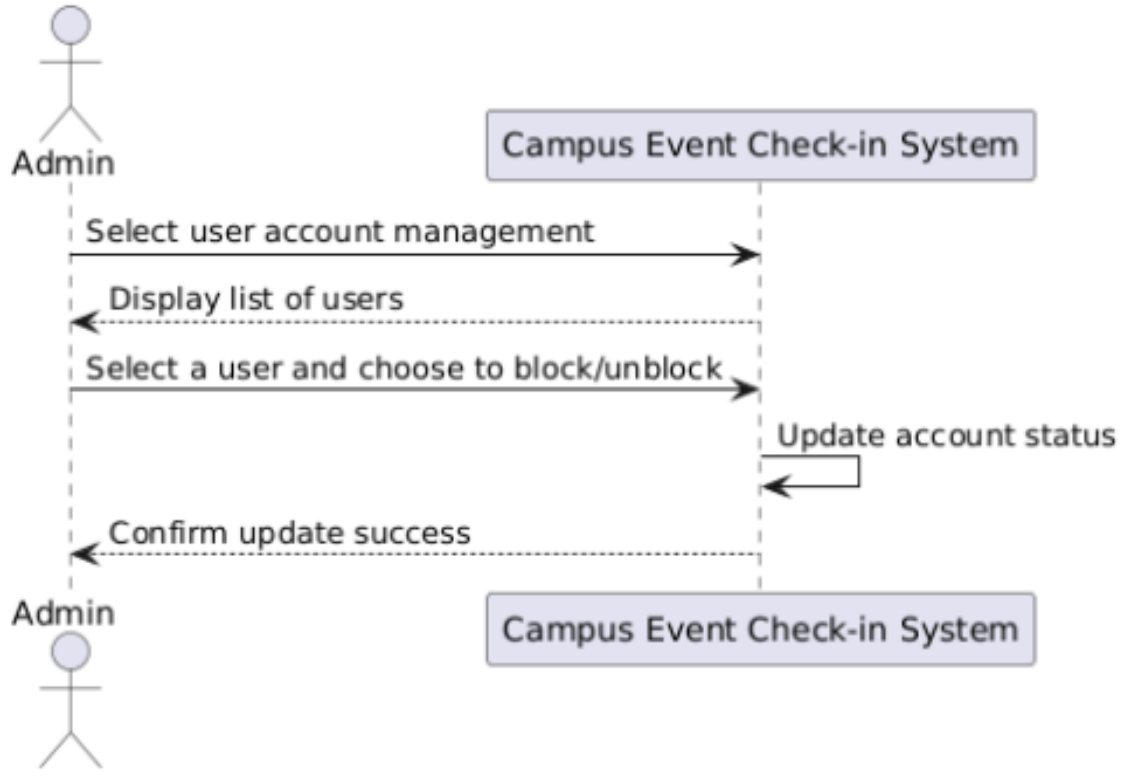


Figure 3.1.2.13: Manage user account (sequence diagram)

3.1.2.14 View Student Attendance

Use Case Name:	View Student Attendance
Description:	The Admin views attendance records of students for a specific campus event.
Primary Actor:	Admin
Precondition:	Admin is logged into the system.
Postcondition:	Attendance records of students for the selected event are displayed to the Admin.
Main Flow:	1. Admin selects attendance management. 2. System displays list of campus events. 3. Admin selects a specific event. 4. System retrieves and displays student attendance records for the selected event.
Alternate Flow:	If attendance data retrieval fails, system displays an error message and prompts admin to retry or cancel.

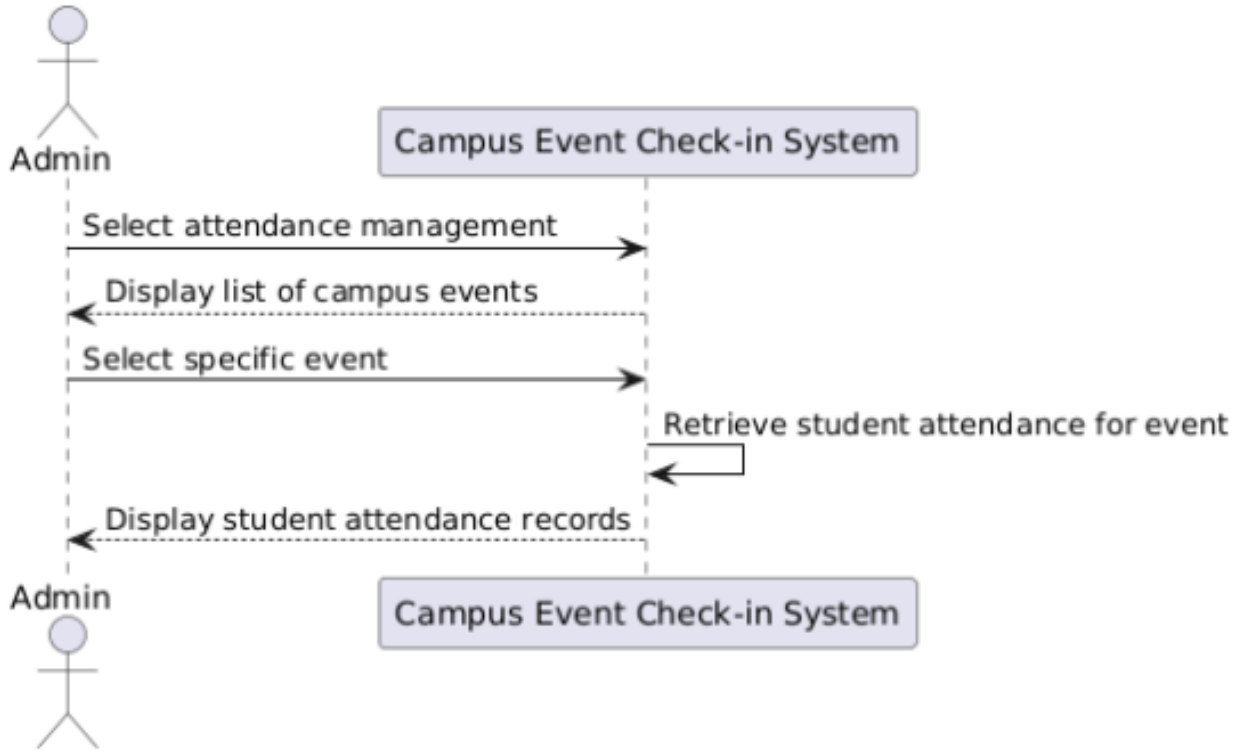


Figure 3.1.2.14: View Student Attendance (sequence diagram)

3.1.2.15 Generate report

Use Case Name:	Generate Report
Description:	The Admin generates exportable reports on event attendance, check-in activity, revenue, and analytics for internal and audit purposes.
Primary Actor:	Admin
Precondition:	Admin is logged into the system.
Postcondition:	A detailed report is generated and available for download in a standard format (e.g., PDF, Excel).
Main Flow:	1. Admin selects “Generate Report” option. 2. System shows report types (real-time, historical, revenue, analytics). 3. Admin selects a report type and applies filters (date, event, demographics). 4. System processes and compiles report. 5. System presents download/export link.
Alternate Flow:	If the system fails to generate the report (e.g., due to a server error or no data available), it displays an error message and prompts the Admin to retry or modify the filters.

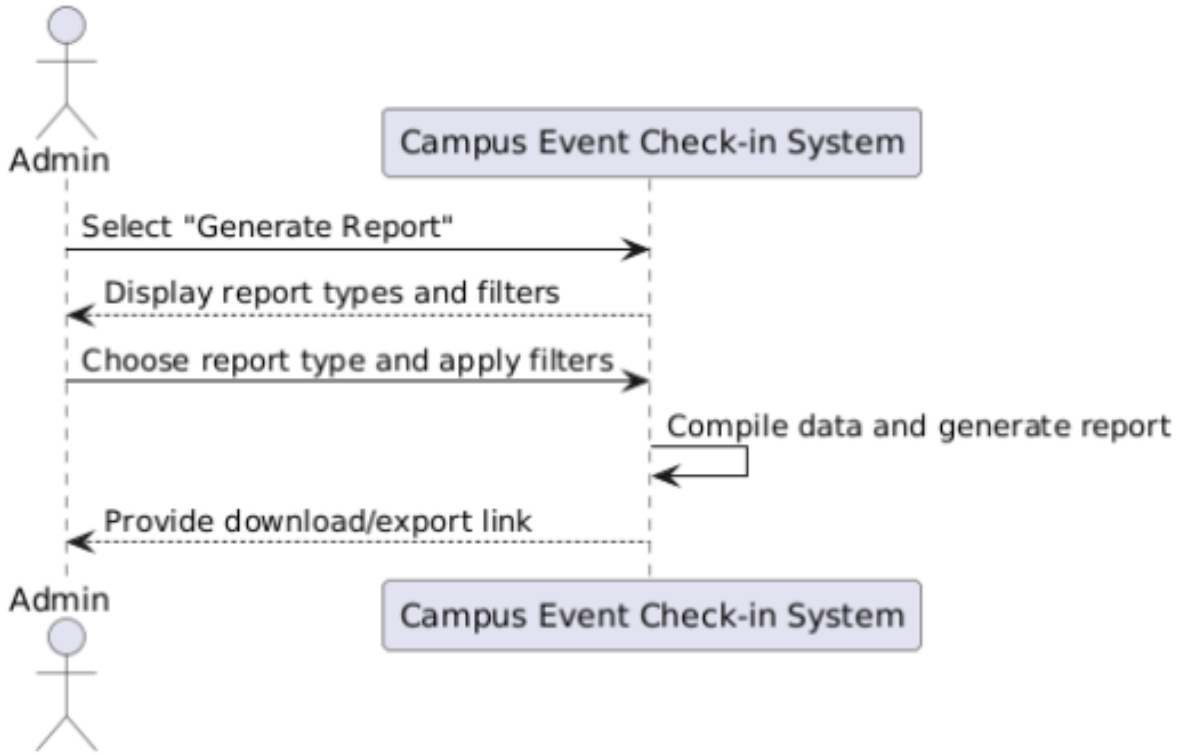


Figure 3.1.2.15: Generate Report (sequence diagram)

3.1.2.16 Manage Event Request

Use Case Name:	Manage Event Request
Description:	The Admin reviews submitted event requests and approves or rejects them through the system.
Primary Actor:	Admin
Precondition:	Admin is logged into the system.
Postcondition:	The selected event request is either approved or rejected and its status is updated.
Main Flow:	1. Admin selects “Manage Event Request” option. 2. System retrieves list of pending requests from the database and displays them to Admin. 3. Admin selects a request to approve or reject. 4. System updates the request status.
Alternate Flow:	If status update fails, the system shows an error message and prompts the Admin to retry.

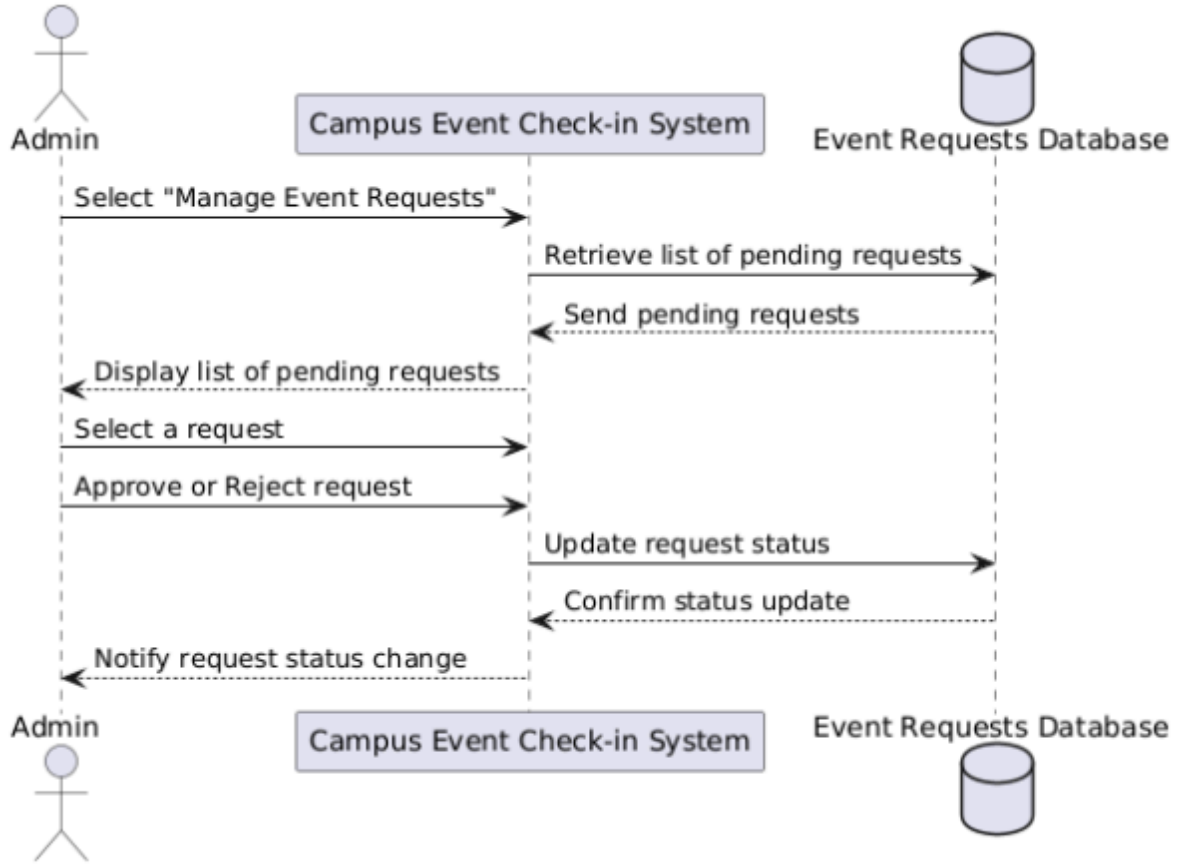


Figure 3.1.2.16: Manage event request (sequence diagram)

3.2 Functional Requirements

This section specifies the core features and capabilities the system must deliver to fulfill its intended use. These requirements describe what the system shall do, focusing on user authentication, event handling, ticketing, administrative tasks, notifications, and feedback functionalities. Each functional requirement is uniquely identified and assigned a priority level based on its importance to the system's success.

Requirement ID	Functional Requirement Name	Description	Priority
REQ_F001	User Authentication	The system shall authenticate users via MMU credentials and grant role-based access (Student, Organizer, Admin).	High
REQ_F002	Event Registration and Ticketing	1. Students shall register for events and purchase tickets via the platform. Paid events shall redirect to third-party payment gateways. 2. The system shall generate unique QR/barcode tickets upon successful registration/payment.	High
REQ_F003	Real-Time Check-In	The system shall validate tickets via QR code/NFC/student ID scans and log attendance with timestamps in real time.	High
REQ_F004	Event Management	Organizers shall create, edit, or delete events, set capacities, and monitor registrations/attendance via a dashboard.	High
REQ_F005	Administrative Controls	Admins shall manage user accounts, generate reports, approve/reject event requests, and view attendance records.	High
REQ_F006	Notifications	The system shall send automated emails for event confirmations, reminders, and updates.	Medium
REQ_F007	Feedback	Students shall submit feedback for attended events, and organizers shall respond via the platform.	Medium

Table 3.2: Functional requirements table

3.3 Performance Requirements

This section outlines the performance-related expectations for the system to ensure it operates efficiently, reliably, and responsively under various conditions. These requirements cover aspects such as speed, scalability, system availability, and the ability to handle concurrent users and transactions. Each performance requirement is assigned a unique identifier and a priority level to guide development and testing efforts.

Requirement ID	Performance Requirement Name	Description	Priority
REQ_P001	User Authentication Speed	The system shall process user authentication within 3 seconds under normal load.	High
REQ_P002	Real-Time Check-In Validation	Real-time check-in validation shall occur within 3 second of ticket scan.	High
REQ_P003	Event Dashboard Load Time	The system shall load the event dashboard for Event Organizers/Admins within 3 seconds under normal load.	High
REQ_P004	Event Report Generation Time	The system shall generate and deliver event reports within 5 seconds for reports covering ≤ 1000 records.	Medium
REQ_P005	Feedback Submission Time	The feedback submission feature shall store and confirm user feedback within 2 seconds.	Medium
REQ_P006	Concurrent Users	The system shall support up to 2,000 concurrent users during peak event registrations/check-ins.	High
REQ_P007	Availability	The platform shall maintain 99.9% uptime during event periods, excluding scheduled maintenance.	High
REQ_P008	Transaction Throughput	Payment processing via third-party gateways shall handle 50 transactions per minute without degradation.	High
REQ_P009	Event Data Retrieval	Event browsing/search results shall load within 3 seconds for up to 30 active events.	High
REQ_P0010	Scalability	The system shall scale horizontally to support a 50% increase in user base (e.g., from 10,000 to 15,000 users) without redesigning core architecture.	High

Table 3.3: Performance requirements table

3.4 Usability Requirements

This section defines the usability goals and requirements that ensure the system is user-friendly, efficient, and visually consistent. These requirements focus on delivering a smooth user experience for all roles (Students, Organizers, Admins) by considering aspects such as interface design, ease of use, responsiveness, and feedback mechanisms. Each requirement is uniquely identified and prioritized based on its importance to system usability.

Requirement ID	Usability Requirement Name	Description	Priority
REQ_UR001	Role-Based Interface	The system shall provide tailored dashboards for Students, Organizers, and Admins, with intuitive navigation and minimal training required.	High
REQ_UR002	Error Handling	User-facing error messages shall be clear and actionable (e.g., “Payment failed. Please retry or select another method”).	Medium
REQ_UR003	Ease of Use	Icons and labels shall be clearly defined and consistently placed across all pages to reduce user confusion.	High
REQ_UR004	Efficiency	Auto-fill and input validation shall be implemented to reduce form entry time by 30% for repeat users.	Medium
REQ_UR005	User Satisfaction	The system shall use modern, responsive UI design that adjusts to various screen sizes to ensure consistent satisfaction across devices.	High
REQ_UR006	Consistent Design	All pages shall adhere to predefined color schemes, font families, and button styles to ensure visual consistency. Deviations require approval from the design team.	High
REQ_UR007	Feedback Mechanism	Users shall report bugs or suggest improvements via a dedicated portal. The system shall acknowledge submissions with a confirmation message and ticket number for tracking.	Low
REQ_UR008	Real-Time User Feedback	During operations such as form submission, file upload, or long processing tasks, the system shall display real-time feedback (e.g., loading spinners, success confirmations, error pop-ups).	Medium

Table 3.4: Usability requirements table

3.5 Interface Requirements

3.5.1 System Interface

The campus Event Check-in System and Student ID and Payment Integration will interface with various other systems to ensure seamless data exchange, communication, and functionality. Below is a detailed list of system interfaces:

Interface ID	System Name	Description	Details
REQ_SI001	Student Identification System	Interfaces with MMU's official student records for authentication and identity verification	Read-only access via secure API to verify student ID, faculty, program, and enrollment status
REQ_SI002	Payment Gateway	Integrates with trusted third-party payment providers for secure online transactions	RESTful API for ticket purchases, e-wallet/FPX integration, transaction logging, and e-receipt generation

3.5.2 User Interface

The Campus Event Check-in System is a responsive web app that scales to phones, tablets, and desktops. A fixed top bar shows the logo on the left and a role-based menu on the right; after sign-in, each user lands on a tailored dashboard and can reach any page with one click. Key actions appear as clear buttons, while data is displayed in sortable tables or cards. Forms use consistent labels above fields, mark required entries with a red "*", and place the "Save/Submit" button at the lower-right. Success, error, or confirmation messages pop up in a coloured banner and can be dismissed with a close icon.

3.5.2.1 Student Interface

1. Login

The student meets a centred sign-in card containing two fields (Student ID / e-mail and password) and one blue "Log in" button. After a successful log-in the system redirects the

student to the personal dashboard; after a failure it shows a red banner that says “Invalid credentials. Please try again.”

2. Dashboard

The dashboard greets the student by name and displays three quick-access cards: “Upcoming Events”, “My Tickets”, and “Latest Notifications”. Each card includes a “View” button that leads to the related page.

3. Browse and Search Events

When the student selects “Events” in the top bar, the system shows event cards in a masonry grid. Every card carries the title, date, venue, and a green “Register” button. A search field and filter drop-downs for category and date range run across the top of the page. Typing a keyword and pressing **Enter** instantly refreshes the list.

4. Register for Event

After the student presses “Register”, a pop-up summarises the event details and asks for confirmation. If the event is free the student simply presses “Confirm” and the system creates the ticket. If payment is required the pop-up adds a drop-down for “Payment method” and a “Pay now” button. Pressing that button opens the payment gateway in a modal window; when the gateway replies “paid”, the modal closes and the system shows “Registration successful” together with a link to the ticket.

5. Make Payment

If the student chooses “**Pay online**”, the system opens the payment gateway in a modal window; when the payment is confirmed, the ticket is issued as paid, and the student receives a confirmation message and ticket link.

If the student chooses “**Pay on-site**”, the pop-up records the choice and shows “Payment pending”. The ticket is still issued but marked unpaid until the cashier scans its barcode; after scanning, the status switches to paid and the student receives an email receipt.

6. Generate Ticket

When registration is finished, the system automatically creates a ticket that contains a QR code. The student can see the ticket immediately or later by opening “My Tickets”.

7. View Ticket

The “My Tickets” page lists every registered event in a two-column table. An “Open” button beside each row shows the ticket in full size with the QR code at the top, event details in the middle, and a grey “Download PDF” link at the bottom.

8. Cancel Registration

Inside “My Tickets” each paid ticket has a red “Cancel” button that stays active until the organiser’s cut-off time. Pressing the button launches a confirmation dialog; on acceptance the system cancels the ticket, reverses the payment, and shows “Registration cancelled”.

9. View Notification

The bell icon in the top bar carries a small badge that displays the number of unread items. Clicking the bell opens a dropdown containing the latest messages. If the student selects “See all notifications” the system navigates to a full-screen list ordered by date. When the list is empty the screen states “No notifications available” in centred grey text.

10. Check-in to Event

At the venue entrance the student opens the ticket and shows the QR code to the gate staff. The staff member’s scanner sends the code to the system; if the code is valid the interface plays a short “success” sound and flashes green. The student sees a green banner on the mobile screen that says “Checked-in at 09:05”.

11. Submit Feedback

After the event a grey “Leave feedback” button appears on the ticket page. Clicking it opens a simple form that contains five empty stars and a multi-line comment box. When the student presses “Submit”, the system thanks them and empties the form for a possible second comment.

12. View Event History

The “History” tab shows every past event in a chronological table with columns for date, event title, ticket status, and given rating. Rows are expandable to reveal the original feedback text.

3.5.2.2 Event Organizer Interface

1. Login

The event organizer meets a centred sign-in card containing two fields (event organizer ID / e-mail and password) and one blue “Log in” button. After a successful log-in the system redirects the event organizer to the personal dashboard; after a failure it shows a red banner that says “Invalid credentials. Please try again.”

2. Dashboard

After login, the event organizer is directed to a personalized dashboard. The dashboard displays summary cards such as “Events Created”, “Registrations Today”, and “Pending Feedback”. A sidebar on the left provides quick links to “My Events” and “Feedback”.

3. Manage Event

Selecting “My Events” opens a table listing the organizer’s events, each with action links: “Edit”, “Cancel”, or “View”. Clicking “Create Event” opens a step-by-step wizard to enter event details like title, description, date, time, venue, capacity, and pricing. A final “Publish” button adds the event to the student catalogue. Choosing “Edit” reopens the wizard with existing details filled in, and a blue “Save changes” button activates when edits are made. Selecting “Cancel” prompts a confirmation and allows the organizer to enter an optional explanation, which is emailed to all registered students.

4. Respond to Feedback

The “Feedback” page shows a list of comments sorted by star rating. Pressing “Reply” under a comment reveals a one-line input field. When the organiser sends the reply, the student sees it under their original comment.

3.5.2.3 Admin Interface

1. Login

The event admin meets a centred sign-in card containing two fields (admin ID / e-mail and password) and one blue “Log in” button. After a successful log-in the system redirects the admin to the personal dashboard; after a failure it shows a red banner that says “Invalid credentials. Please try again.”

2. Dashboard

After successful login, the admin is directed to a dashboard showing key metrics such as total number of students, number of events this month, tickets sold, and refund rate. A collapsible sidebar gives access to various administrative tools for managing users, events, and system settings.

3. Manage User Account

Selecting “Users” opens a paginated table with a search bar at the top-right. Each user row includes action buttons for “Block” and “Unblock”. Pressing “Block” disables the user’s access immediately and shows a confirmation message. Pressing “Unblock” restores access, also with a confirmation message. Only one of the two buttons is shown at a time based on the user’s current status.

4. View Student Attendance

The “Attendance” page lets the admin choose an event from a drop-down list. Once chosen, a table lists every registered student with columns for Student ID, name, faculty, and check-in time.

5. Generate Report

The “Reports” page displays a short form with start date, end date, and report type. After pressing “Generate”, the system shows a progress bar, then offers a PDF download named “Event-Report-2025-04-30.pdf”.

6. Manage Event Request

The “Event Requests” page displays all pending submissions from organisers in individual cards, each showing event details. Each card has an “Approve” button and a “Reject” button. Pressing “Approve” opens a short confirmation dialog; confirming changes the request status to “Approved” and highlights the card with a green border. Pressing “Reject” opens a dialog for entering a rejection reason; once confirmed, the card disappears and an email is sent to the organiser.

3.5.3 Hardware Interface

The Campus Event Check-in System and Student ID and Payment Integration will be compatible on any desktop or mobile devices with the specification described below. The Campus Event Check-in System may not be able to access all features or even use the system at all if the devices do not meet the recommended specifications that described below:

Interface ID	Description
REQ_HI001	The processor for the devices shall be at least a 32- bit, x86 processor, depending on the brand of the processor
REQ_HI002	The Random Access Memory (RAM) required for the devices shall be at least 4GB
REQ_HI003	The storage required for the devices shall be at least a minimum of 1GB free space on primary storage
REQ_HI006	The network adapter shall be either an ethernet connection, wireless network, or mobile data

3.5.4 Software Interface

The Campus Event Check-in System also requires other software to function properly. The interfaces between Campus Event Check-in System and other software are described below:

ID	Category	Name	Version Number	Purpose	Reference
REQ_SI001	Operating System	Microsoft Windows	Windows 10 or later	Software platform that manages user device hardware and software resources and facilitates the execution of computer programs that are required to execute DMS.	Opera Browser Requirement
		macOS	macOS Mojave 10.14 or later		Opera Browser Requirement
		Linux	Ubuntu 20.04+, Debian 11+, openSUSE 15.5+, Fedora Linux 40+		Opera Browser Requirement
		iOS	iOS 15.0 or later		Opera Browser Requirement
		Android	Android 10.0 Quince Tart or later		Opera Browser Requirement
REQ_SI002	Browser	Google Chrome	124.0.6367.159	The application end users used to make requests to our web server to get the relevant data.	Chrome Browser Official Page
		Microsoft Edge	124.0.2478.80		Microsoft Edge Official Page
		Safari	16.6.1		Safari Official Page
REQ_SI003	Database	MySQL	8.4 or later	Used to store, create, modify, and delete data used in the software	MySQL Official Page

3.5.5 Communication Interface

This section outlines the various interfaces to communications, specifying the protocols and methods the system will use to interact with other systems, networks, and services.

Interface ID	Description	Protocols	Priority
REQ_CI001	The platform shall support HTTP/HTTPS protocols for secure web communication	HTTP/HTTPS	High
REQ_CI002	The platform shall integrate with social media APIs (e.g., Facebook Graph API, Twitter API, LinkedIn API) for data exchange	RESTful API, OAuth 2.0	High
REQ_CI003	The platform shall use WebSocket protocol for real-time communication and notifications.	WebSocket	Medium
REQ_CI004	The platform shall support SMTP for sending emails to users for notifications, updates, and password resets.	SMTP	Medium
REQ_CI005	The platform shall utilize LDAP/Active Directory for authentication and user management within organizational networks	LDAP/Active Directory	High
REQ_CI006	The platform shall support SFTP for secure file transfers between the platform and other systems	SFTP	Medium
REQ_CI007	The platform shall ensure that shared content to social media is posted within 1 second	MQTT	Low
REQ_CI008	The platform shall enable data exchange with external databases through JDBC/ODBC connections	JDBC/ODBC	Medium
REQ_CI009	The platform shall support JSON and XML formats for data interchange between services and clients	JSON, XML	High

3.5.6 Memory Constraints

The Memory Constrains are described below:

Constrains ID	Description
REQ_MC001	The primary memory (RAM) of the devices must be at least 4GB to ensure smooth operation
REQ_MC002	The secondary storage (e.g., hard drive or SSD) should have a minimum of 1GB free space available

3.6 Logical Database Requirements

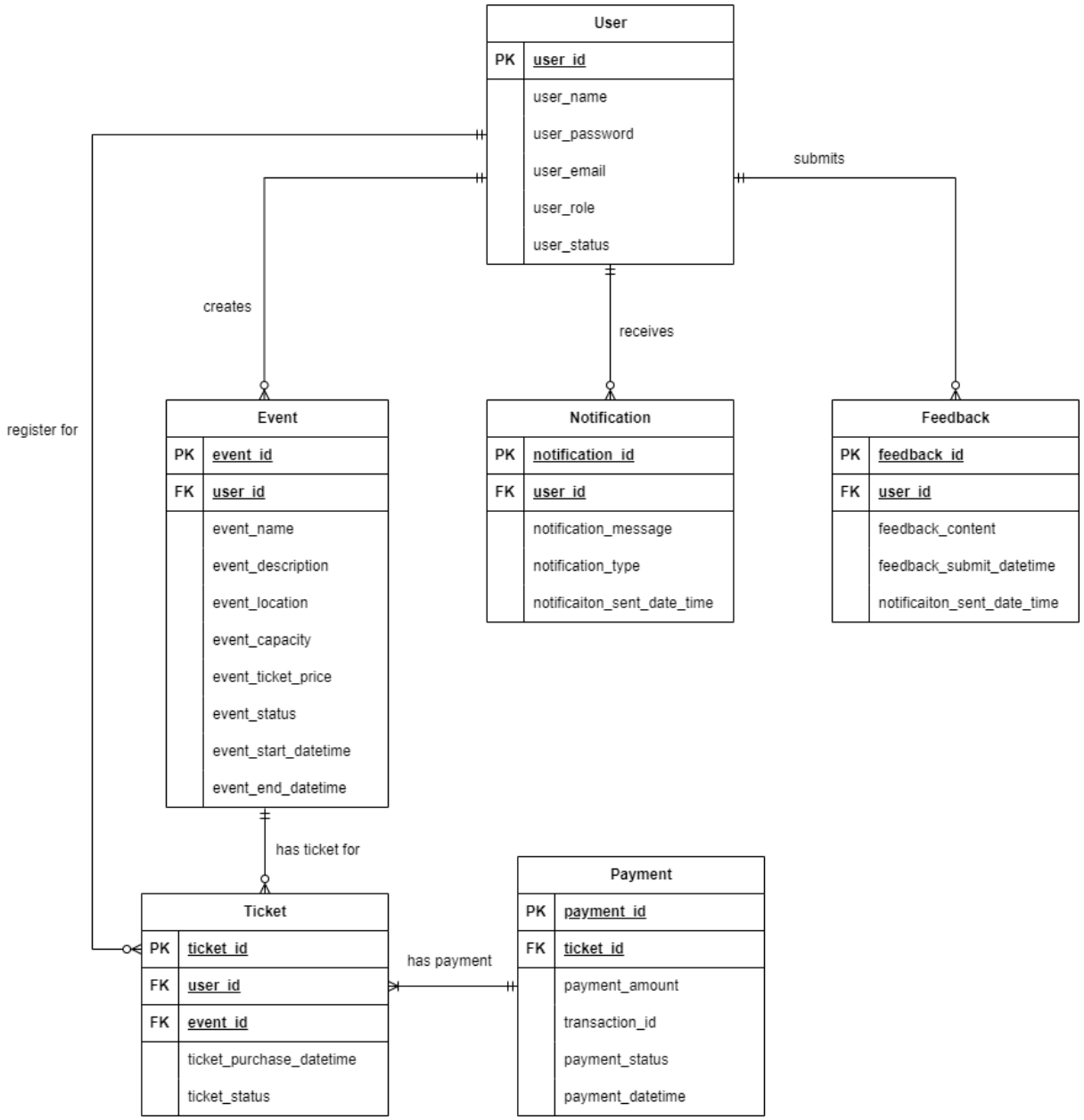


Figure 3.6: Entity-Relationship Diagram

The Entity-Relationship Diagram (ERD) models the core data structure of the Campus Event Check-in System, capturing key entities, their attributes, relationships, and constraints. It ensures efficient management of user roles, event registrations, payments, notifications, and feedback.

Below is a detailed breakdown:

3.6.1 User Data Dictionary

Field Name	Description	Data Type	Constraints	Extra Notes
user_id	Unique user identifier	INT	PRIMARY KEY, AUTO_INCREMENT	MMU ID
user_name	Full name of the user	VARCHAR(255)	NOT NULL	-
user_password	Hashed password for authentication	VARCHAR(255)	NOT NULL	-
user_email	User's email address	VARCHAR(255)	NOT NULL, UNIQUE	-
user_role	Role (Student/Organizer/ Admin)	ENUM	NOT NULL, ENUM('Student','Organizer','Admin')	Role-based access control
user_status	Account status (Active/Blocked)	ENUM	DEFAULT 'Active', ENUM('Active','Blocked')	Admins can block/unblock accounts

3.6.2 Event Data Dictionary

Field Name	Description	Data Type	Constraints	Extra Notes
event_id	Unique event identifier	INT	PRIMARY KEY, AUTO_INCREMENT	-
user_id	Organizer who created the event	INT	FOREIGN KEY REFERENCES User(user_id)	Only Organizers can create events
event_name	Name of the event	VARCHAR(255)	NOT NULL	-
event_description	Event details	TEXT	NOT NULL	-
event_location	Physical or virtual location	VARCHAR(255)	NOT NULL	-

event_capacity	Maximum attendees	INT	NOT NULL	-
event_ticket_price	Ticket cost	DECIMAL(10,2)	NOT NULL, >= 0	Represents the price per ticket for an event
event_status	Event status	ENUM	ENUM('Active','Cancelled','Completed')	-
event_start_datetime	Event start time	DATETIME	NOT NULL	-
event_end_datetime	Event end time	DATETIME	NOT NULL	CHECK (event_end_datetime > event_start_datetime)

3.6.3 Ticket Data Dictionary

Field Name	Description	Data Type	Constraints	Extra Notes
ticket_id	Unique event identifier	INT	PRIMARY KEY, AUTO_INCREMENT	-
user_id	Student who registered	INT	FOREIGN KEY REFERENCES User(user_id)	Students can register for multiple events
event_id	Associated event	INT	FOREIGN KEY REFERENCES Event(event_id)	-
ticket_purchase_datetime	Timestamp of purchase	DATETIME	NOT NULL	-
ticket_status	Ticket validity	ENUM	ENUM('Valid','Used','Cancelled')	Used for check-in validation

3.6.4 Payment Data Dictionary

Field Name	Description	Data Type	Constraints	Extra Notes
payment_id	Unique payment identifier	INT	PRIMARY KEY, AUTO_INCREMENT	-
ticket_id	Linked ticket	INT	FOREIGN KEY REFERENCES Ticket(ticket_id)	One-to-one relationship with Ticket
payment_amount	Transaction amount	DECIMAL(10,2)	NOT NULL	-
transaction_id	Third-party gateway transaction ID	VARCHAR(255)	NOT NULL, UNIQUE	References Payment Gateway API
payment_status	Payment status	ENUM	ENUM('Pending','Completed','Failed')	-
payment_datetime	Transaction timestamp	DATETIME	NOT NULL	-

3.6.5 Notification Data Dictionary

Field Name	Description	Data Type	Constraints	Extra Notes
notification_id	Unique notification identifier	INT	PRIMARY KEY, AUTO_INCREMENT	-
user_id	Recipient	INT	FOREIGN KEY REFERENCES User(user_id)	-
notification_message	Content of the notification	TEXT	NOT NULL	-
notification_type	Category (EventUpdate/Reminder)	ENUM	ENUM('EventUpdate','Reminder')	-
notification_sent_date_time	Timestamp of notification	DATETIME	NOT NULL	-

3.6.6 Feedback Data Dictionary

Field Name	Description	Data Type	Constraints	Extra Notes
feedback_id	Unique feedback identifier	INT	PRIMARY KEY, AUTO_INCREMENT	-
user_id	Student who submitted feedback	INT	FOREIGN KEY REFERENCES User(user_id)	-
feedback_content	Feedback text	TEXT	NOT NULL	-
feedback_submit_datetime	Timestamp of submission	DATETIME	NOT NULL	-

3.7 Design Constraints

This section outlines the specific restrictions and limitations that must be considered during the design and development of the system. These constraints are derived from institutional policies, industry standards, legal requirements, and technical considerations. Adhering to these constraints ensures the system is secure, compliant, scalable, and aligned with the university's operational and branding requirements. Each constraint is assigned a unique identifier and a priority level to guide implementation efforts effectively.

Requirement ID	Design Constraints Name	Description	Priority
REQ_DC001	Branding Compliance	The user interface must comply with the university's branding guidelines, including color scheme, logo usage, typography, and layout.	Medium
REQ_DC002	Authentication Security	Passwords must be stored using a secure hash algorithm (e.g., bcrypt or SHA-256).	High
REQ_DC003	Device Compatibility	The system must be responsive and work on modern desktop browsers (e.g., Chrome, Firefox, Edge, Safari).	High
REQ_DC004	Data Privacy Compliance	The system must comply with university data protection policies and relevant privacy laws (e.g., GDPR or PDPA), including secure storage and limited access to personal data.	High
REQ_DC005	Language and Accessibility	The application must support English and follow basic accessibility standards (e.g., WCAG 2.1 Level AA) to support users with disabilities.	Medium
REQ_DC006	Deployment Restrictions	The software must be deployable on the university's on-premise server or designated cloud provider approved by the IT department.	High
REQ_DC007	Third-party API Limitation	Any use of third-party APIs (e.g.QR code generation) must be approved by the university IT department and comply with their data handling policies.	High
REQ_DC008	No Installation Requirement	The system must be web-based and accessible via browser without requiring users to install any software.	Medium
REQ_DC009	Time Zone Standardization	All date and time data must be stored and displayed in the university's official time zone	Medium

		(e.g., UTC+8).	
REQ_DC010	No Advertising Policy	No third-party marketing information or adverts may be found on the system.	High
REQ_DC011	Modular Architecture Requirement	Modularity in the system's design will facilitate future integration, scalability, and feature growth.	Medium

3.8 Software System Attributes

3.8.1 Reliability

Requirement ID	Description	Priority
REQ_REL001	The system shall maintain a minimum of 99.5% operational reliability during official MMU working hours and campus event periods.	High
REQ_REL002	The system shall log all check-in, registration, and payment operations with transactional integrity to avoid data loss.	High
REQ_REL003	The system shall be able to recover from unexpected failures using rollback mechanisms and backup recovery within 5 minutes.	Medium
REQ_REL004	All critical system components (e.g., authentication, ticket validation) shall include fail-safes to ensure uninterrupted user operation.	High
REQ_REL005	The system shall notify administrators automatically upon detection of any module failure or data inconsistency.	Medium

3.8.2 Availability

Requirement ID	Description	Priority
REQ_AVA001	The platform shall maintain at least 99.9% uptime during university event periods, excluding scheduled maintenance.	High
REQ_AVA002	The system shall utilize automatic failover and backup servers to ensure high availability	High
REQ_AVA003	Users shall receive a notification at least 24 hours in advance for planned system downtime or maintenance.	Medium
REQ_AVA004	The system shall provide real-time monitoring and alerting for service outages within 1 minute of detection.	High

3.8.3 Security

Requirement ID	Description	Priority
REQ_SRA001	The platform's data shall be protected against unauthorized access.	High
REQ_SRA002	The platform shall employ encryption and access control to safeguard data.	High
REQ_SRA003	The server API shall implement authentication checks to prevent unauthorized access.	High
REQ_SRA004	The system shall enforce strong password policies, including a minimum length and complexity requirements.	High
REQ_SRA005	The system shall conduct regular security vulnerability assessments and penetration testing.	Medium
REQ_SRA006	The platform shall have a built-in firewall to protect against unauthorized access and network attacks.	High

3.8.4 Maintainability

Requirement ID	Description	Priority
REQ_MAI001	The system shall adopt a modular architecture to support independent updates to components.	High
REQ_MAI002	All system source code shall be maintained in a version-controlled repository (e.g., Git).	High
REQ_MAI003	The system shall include detailed documentation for APIs, deployment, and configuration.	Medium
REQ_MAI004	The system shall generate logs for all major operations to facilitate troubleshooting.	High
REQ_MAI005	System updates or patches shall not cause downtime exceeding 10 minutes.	Medium

3.8.5 Portability

Requirement ID	Description	Priority
REQ_POR001	The system shall be accessible on modern browsers (Chrome, Firefox, Edge, Safari) on desktop and mobile devices.	High
REQ_POR002	The platform shall not require any installation; all features shall be available via a browser.	High
REQ_POR003	The system shall be deployable on both MMU's on-premise servers and cloud platforms with no major reconfiguration.	High
REQ_POR004	The system shall store configuration settings in environment-agnostic formats (e.g., JSON or YAML) for cross-platform compatibility.	Medium

3.9 Supporting Information

This section presents the supporting information for the system's requirements, derived from the requirement elicitation phase. To ensure that the **Campus Event Check-in System with Student ID and Payment Integration** meets the actual needs of its intended users, we carried out a structured elicitation process using three main techniques: **brainstorming, interviews, and questionnaires**.

- **Brainstorming** sessions were conducted within the internal project team to identify initial features, define user roles (student, organizer, admin), and document early assumptions. These sessions laid the groundwork for further validation with external stakeholders.

- **Interviews** were held with selected MMU students to explore real-world frustrations and expectations related to current event attendance practices. These semi-structured interviews provided qualitative insights and helped validate or refine the assumptions made during brainstorming.

- **Questionnaires** were then distributed to a wider student population to collect quantitative feedback on key system features. Using structured Likert-scale questions, we were able to validate interview findings at scale and identify clear preferences, such as support for MMU ID-based login, QR code check-in, and digital payment options.

The combination of these techniques ensured both depth and breadth in capturing stakeholder needs, providing a strong foundation for defining and prioritizing system requirements.

3.9.1 Brainstorming

3.9.1.1 Purpose

The brainstorming session was held to identify core features, understand user expectations, and evaluate potential functionalities for the Campus Event Check-in System with Student ID and Payment Integration. The aim was to streamline event check-ins, enhance user satisfaction and reduce manual workloads.

3.9.1.2 Outcomes

The brainstorming session resulted in a diverse and insightful set of outcomes that contributed significantly to shaping the feature set of the Campus Event Check-in System. Ideas, structured feature listing and Kano Model analysis are collected in this brainstorming session. This aided the team in coming up with feature ideas and ranking them according to perceived value and user expectations.

Participants proposed several innovative ideas, including:

- Real-time visibility of check-in status for event organizers to monitor attendance live.
- Automatic student check-in using student ID, reducing human intervention.
- Notification reminders sent before the event begins to improve attendance.
- A refund request feature within the system to enhance convenience for users.

These ideas addressed key user pain points such as time-consuming manual check-ins, forgotten event schedules, and inconvenient refund processes.

Each participant highlighted critical features they believed should be part of the system:

- Secure login system with different roles (Student, Organizer, Admin).
- Two-factor authentication for added security.
- Fast check-in process, ideally under 10 seconds per user.
- A user-friendly application tutorial for first-time users.
- Complete event information provided in the system interface.
- Reminder notifications prior to event start.

These were refined into essential and supporting functionalities that align with usability and performance requirements.

The team categorized three features in the form of Kano-style discussion:

- **Two-Factor Authentication** was seen as a *Must-be Feature*—something users expect by default. If the system doesn't include this feature, it might cause dissatisfaction.
- The **application tutorial** was assessed as a performance feature; users value it and are more satisfied when it is available, but its absence doesn't raise any significant issues.
- **Event Reminders** were considered a *Delighter Feature*—they pleasantly surprise users but aren't strictly expected.

This classification helped distinguish between basic necessities, performance boosters, and potential user experience enhancers.

All in all, the team has made out a conclusion that the main system should place a high priority on speed, dependability and usability while also integrating considerate addition that improves user experience. The brainstorming method helped the team establish development goals based on practical user needs and expectations while also encouraging creativity.

3.9.2 Interviews

The goal of the interview was to get firsthand information from a MMU student. The interview session helps to create a system that better meets user expectations by understanding their present events, registration, and payment-related behaviors, preferences, and pain points.

Kano

The Kano Model was used to evaluate the perceived value of the suggested features, validate them, and find any usability or trust issues early in the development phase.

Key Insight:

Feature Expectations:

- **Reminders:** Strongly supports automatic event reminders, citing forgetfulness as a common issue.
- **Trust in Payment System:** Would trust the system more if it offers:
    - A clear history of purchases
    - Digital invoices or receipts
    - Consistency in successful transactions

Kano Model Responses:

- MMU ID Login should be seen as a performance feature.
- Profile Saved in database is an indifferent category.
- Accessibility Features was considered as a performance feature.
- System Reliability is a must-be feature.
- Ticket Sharing might be seen as indifferent.

The selected student's feedback highlights some key design priorities:

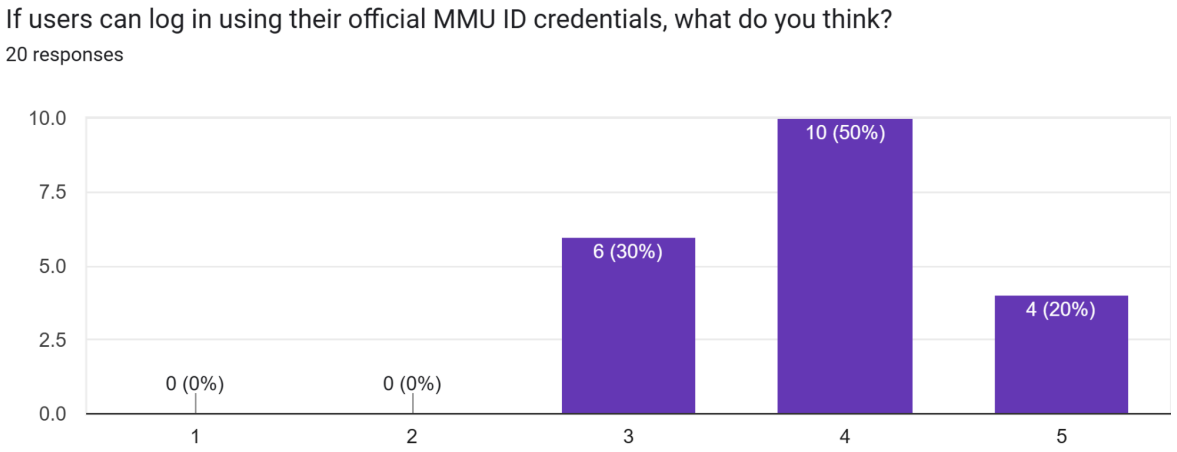
- The system should maintain a reliable and transparent transaction history.
- Reminder notifications are crucial to support engagement.
- Core features should be seamless but also intuitive for occasional users.
- Accessibility and database integration are welcomed, though not deal-breakers.

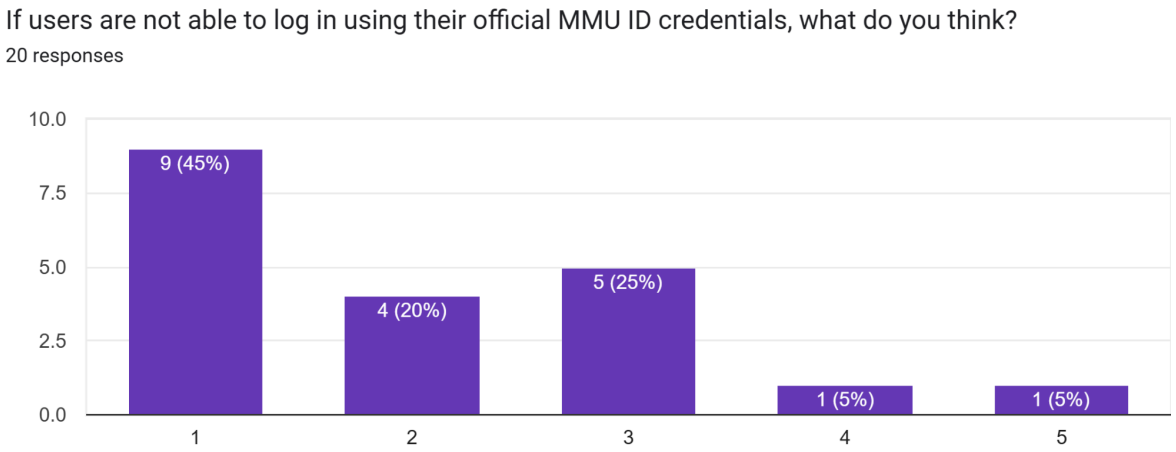
This interview session reinforced design decisions such as prioritizing reliability, transparency in payments, and timely communication with users.

3.9.3 Questionnaire

The purpose of this questionnaire is to gather valuable insights from students regarding their experiences and expectations when joining events in the campus. The following is the questions and responses from the students:

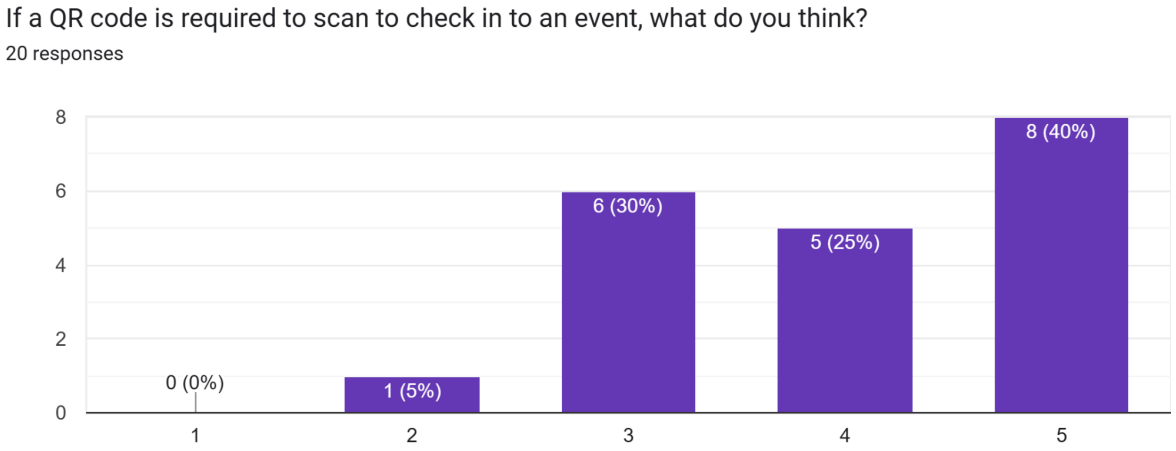
MMU Login Access:

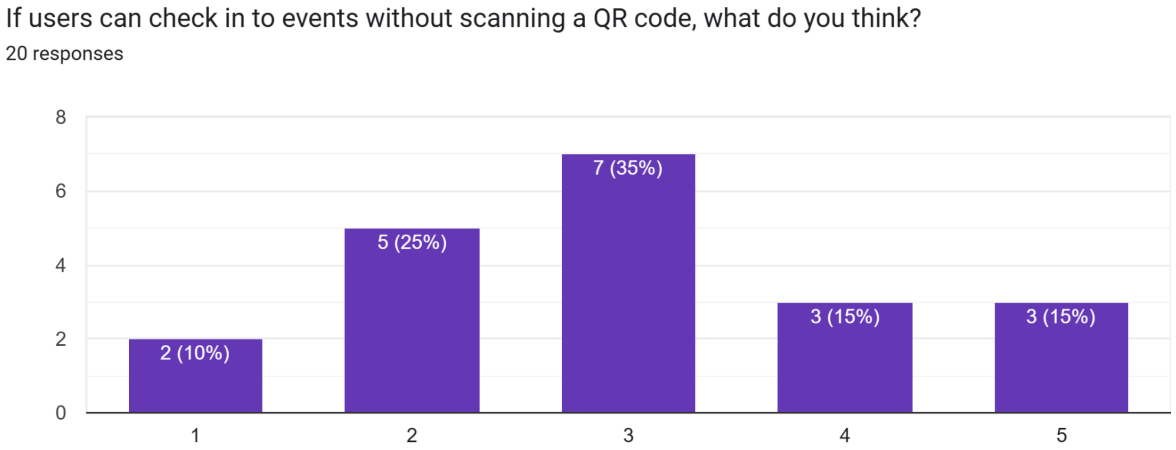




The questionnaire was designed to evaluate the importance of enabling users to log in using their official MMU ID credentials. Major users choose a higher rating that the system should request MMU ID for users to log in. Users expressed dissatisfaction with the absence of this functionality in the dysfunctional scenario. These results suggest a clear expectation among users that MMU ID login should be a core component of the system.

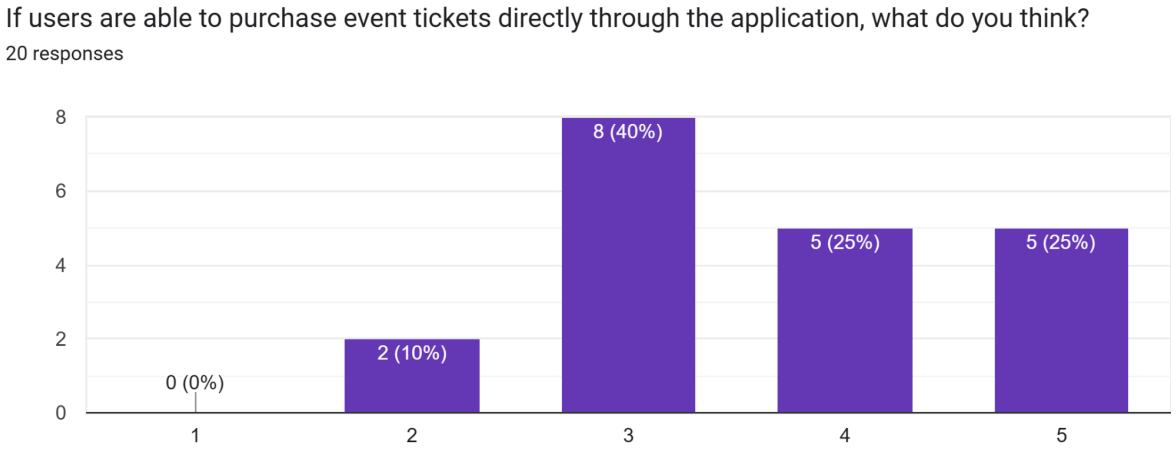
QR Code Check-in:

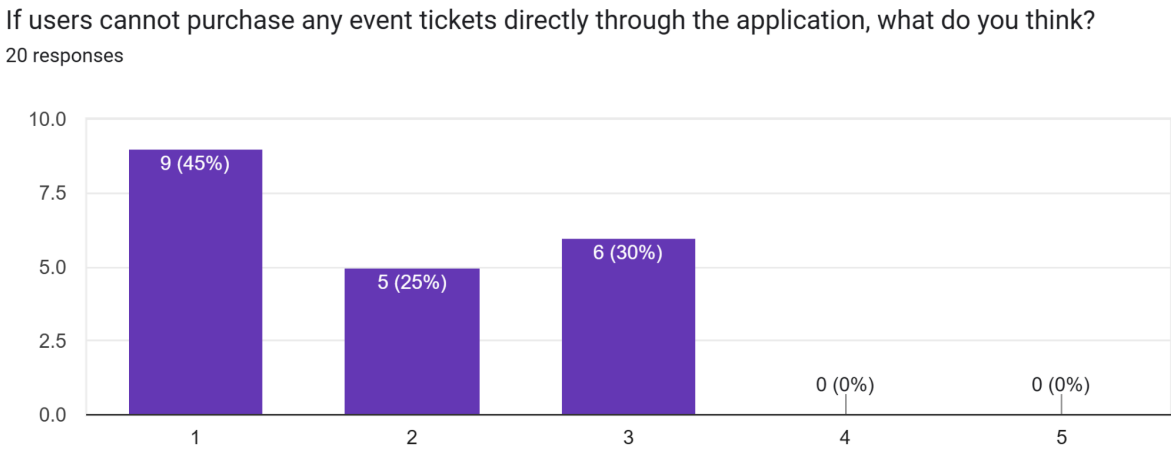




The questionnaire aimed to determine user preference for QR code-based event check-in. Respondents gave a high rating for requiring QR code scanning, suggesting it is seen as a valuable and efficient method. When presented with the alternative of not using a QR code, responses were neutral, indicating that while not mandatory, QR code functionality enhances user experience and is generally expected.

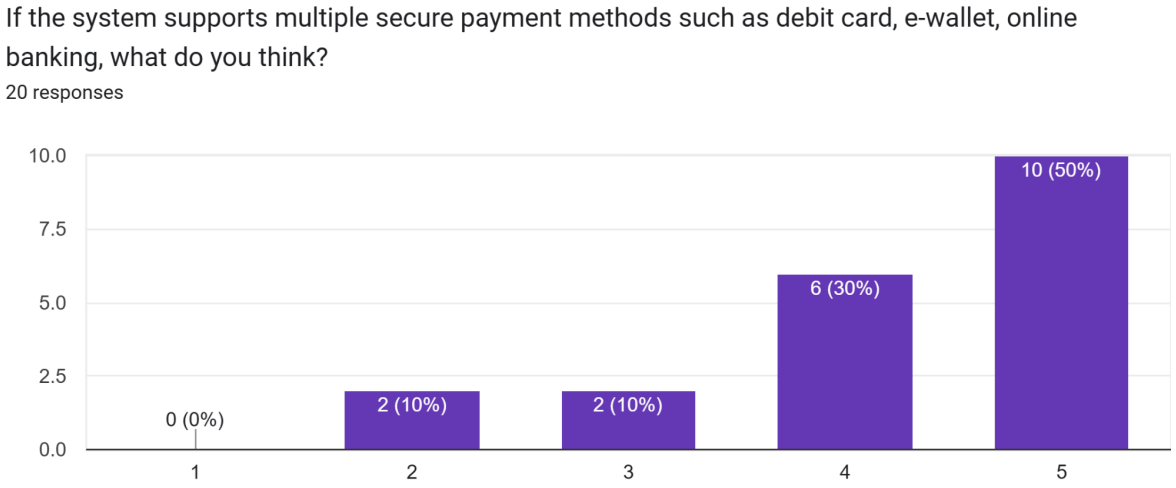
Ticket Purchasing:

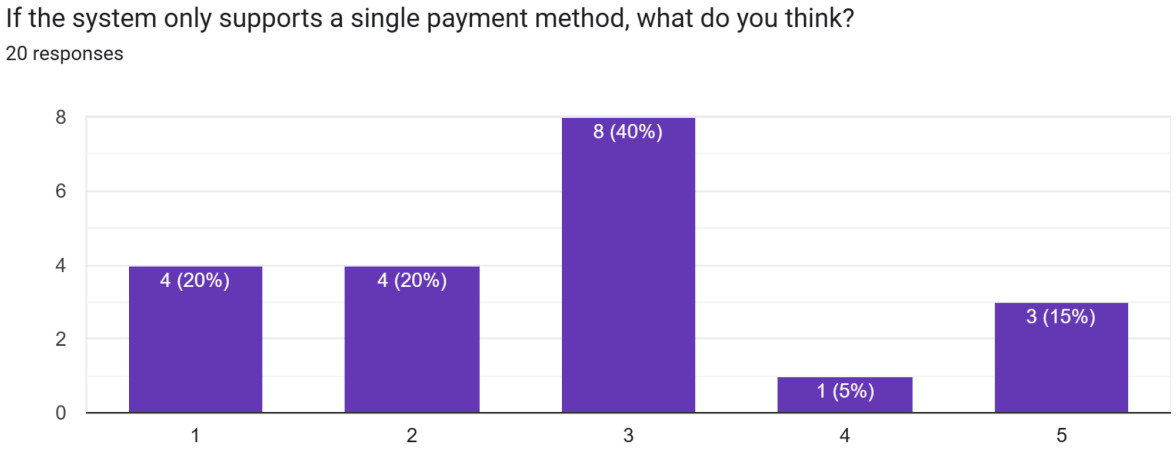




Responses showed moderate support for enabling ticket purchases directly through the application. The feature received a neutral average score, suggesting users are open to it but do not view it as essential. However, when the feature was excluded, users responded with a low rating, indicating that the inability to purchase tickets within the app would negatively affect their experience.

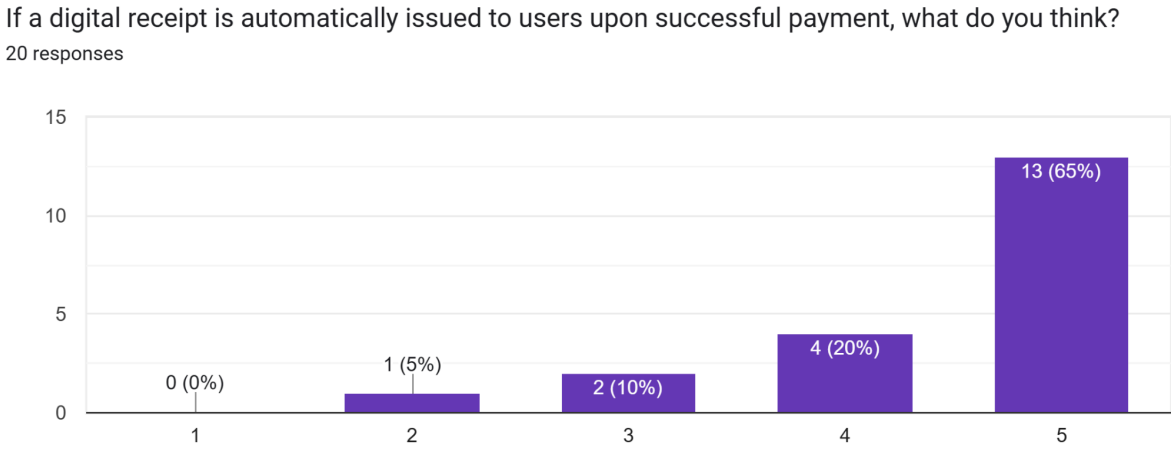
Payment Methods:





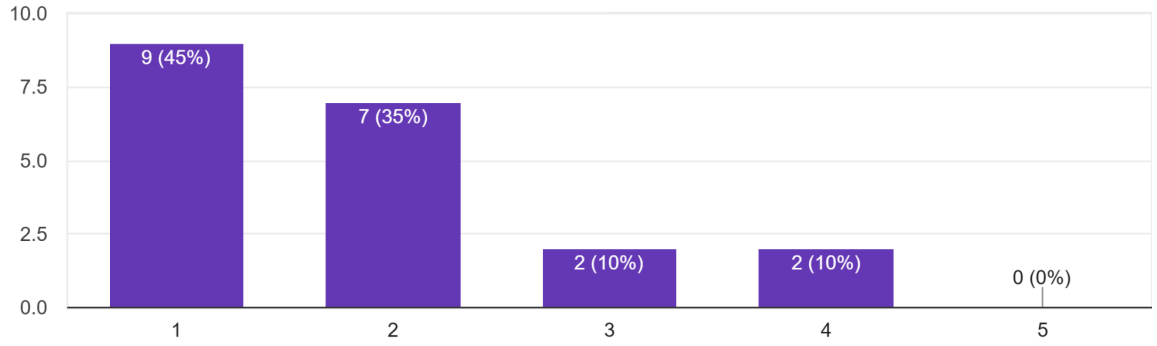
Participants gave a high rating to the inclusion of multiple secure payment options such as debit cards, e-wallets, and online banking. This suggests a strong expectation for flexibility and convenience in handling payments. The response to the scenario where only a single payment method is available was neutral, indicating it is acceptable but may not meet all users' preferences.

Payment Confirmation:



If there's no payment confirmation or receipt shown to users when payment completed, what do you think?

20 responses

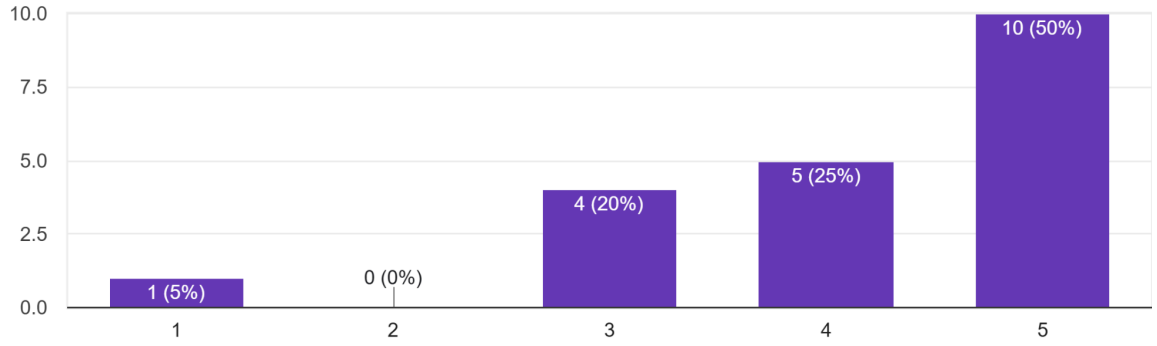


The issuance of digital receipts after payment was rated very highly by users. This reflects a strong desire for transparency and assurance in transactions. When the possibility of no payment confirmation was introduced, it received the lowest score, highlighting that the absence of receipts would significantly undermine user trust and satisfaction.

Real-Time Notifications:

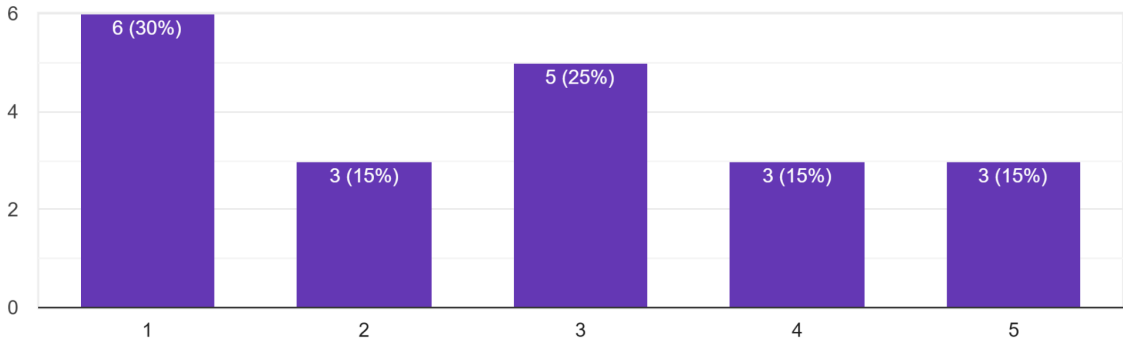
If users are able to receive real-time updates regarding changes to event schedule or location, what do you think?

20 responses



If the system does not provide any real-time notifications or updates of the event, what do you think?

20 responses

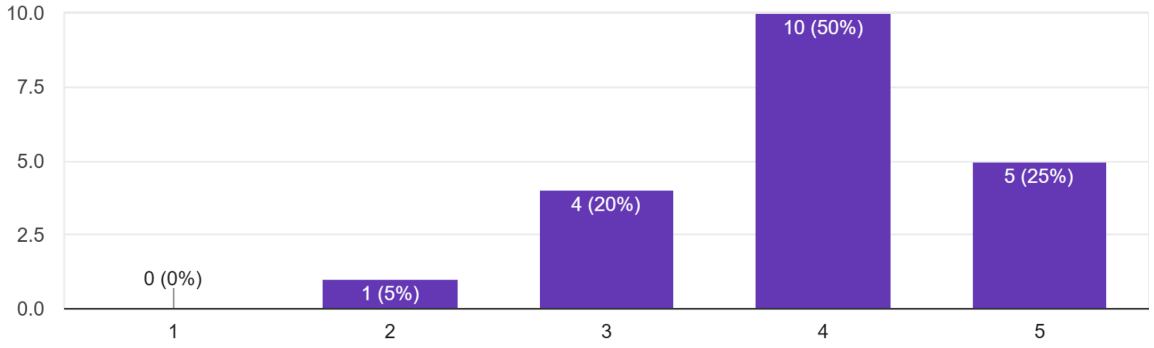


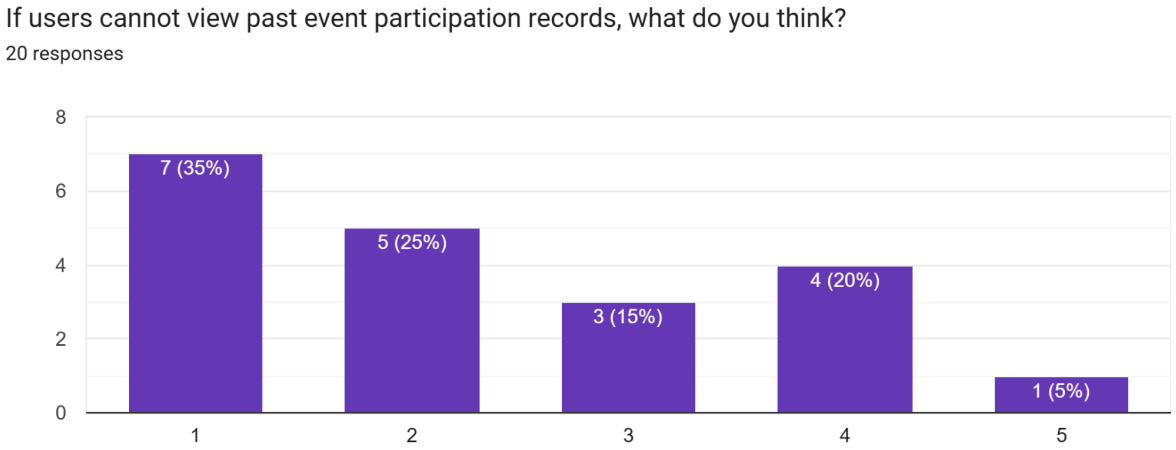
Real-time updates regarding changes to event schedules or locations were highly valued by respondents, with this feature receiving one of the highest average ratings. In contrast, the lack of notifications received the lowest rating, emphasizing that timely information is considered essential for user confidence and effective event participation.

Event History:

If users can view a personal history log of all previously attended events, what do you think?

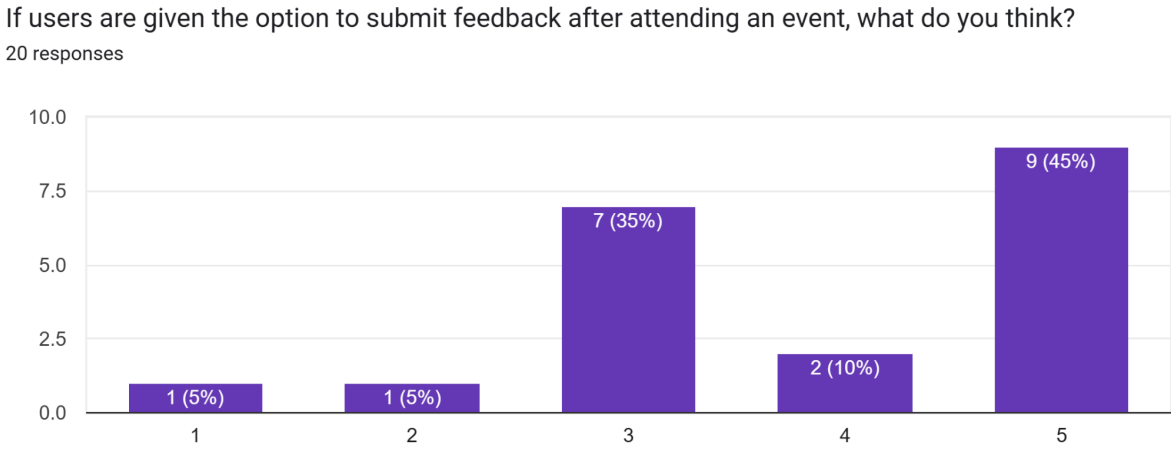
20 responses

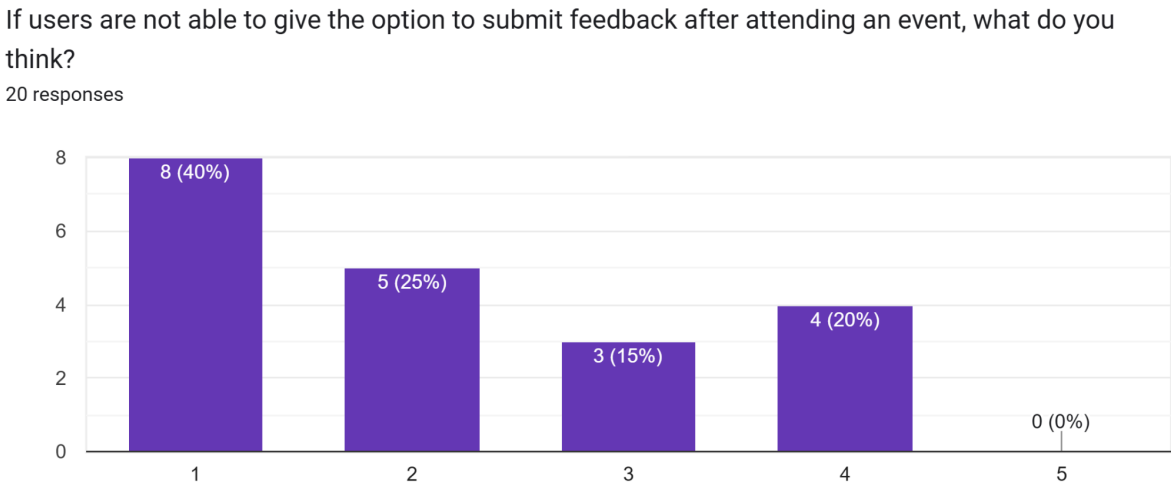




The option for users to view a history of previously attended events was positively received, with a generally high score. This suggests that users find value in having access to their participation records. When this feature was absent, it was rated poorly, indicating that users consider this capability beneficial for tracking involvement and managing future plans.

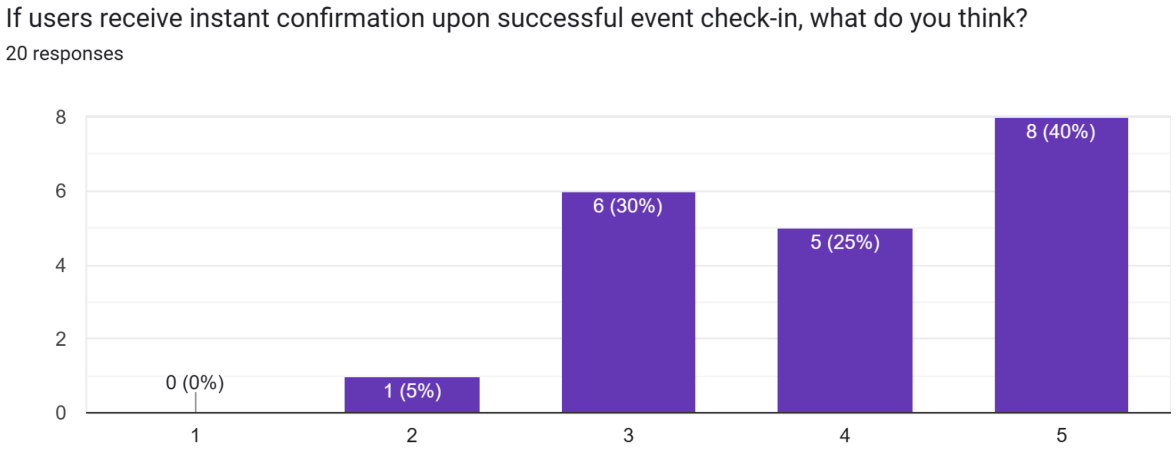
Feedback Submission:

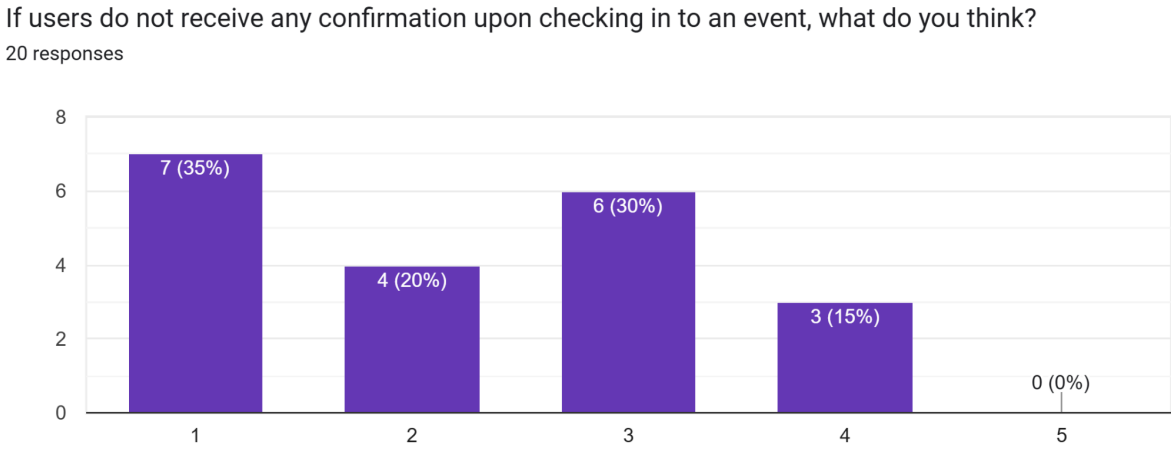




Allowing users to submit feedback after attending events received very strong support. Respondents view feedback mechanisms as a vital communication channel between users and event organizers. The feature's absence received the lowest possible score, reinforcing its importance in fostering engagement and continuous improvement.

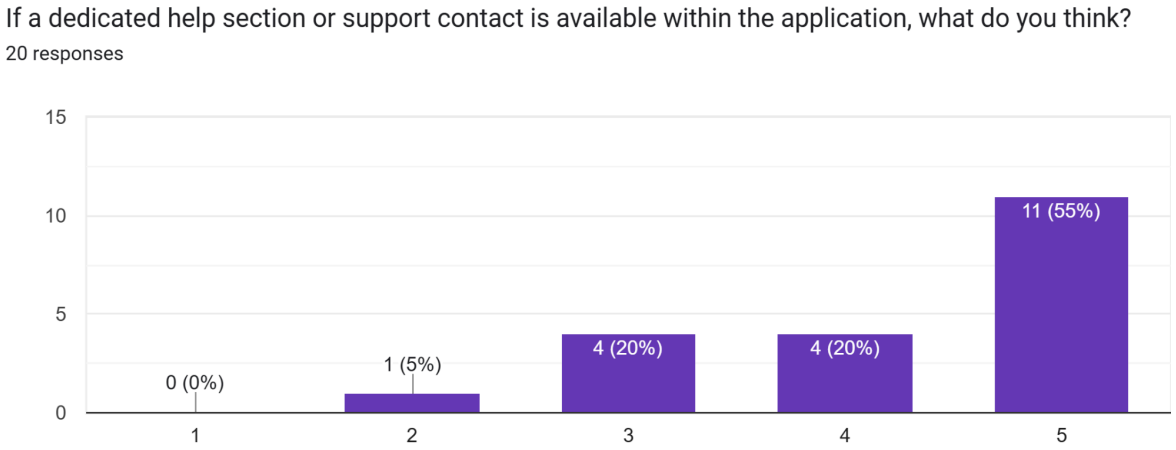
Check-in Confirmation:

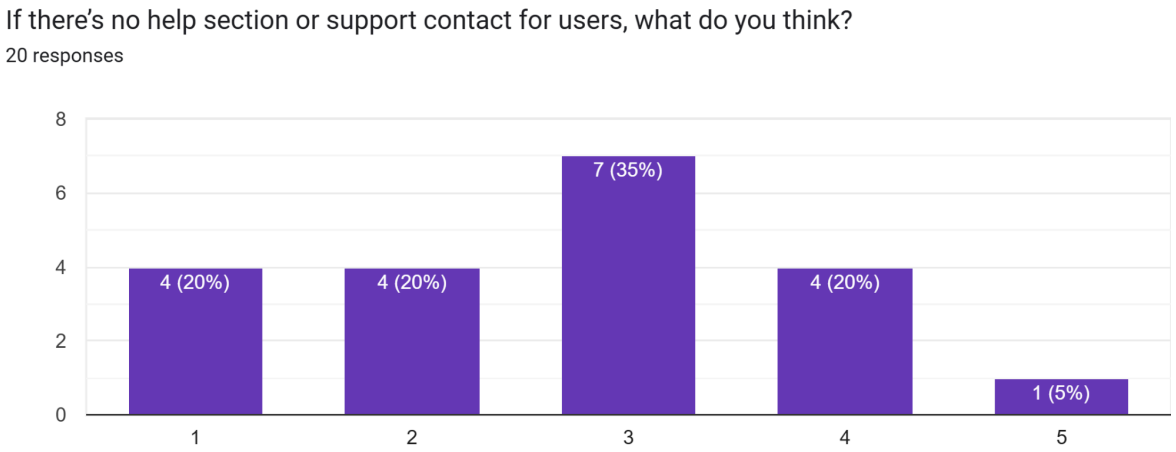




Instant confirmation upon successful event check-in was rated extremely highly by participants. This demonstrates a clear demand for immediate feedback and assurance when using the system. Conversely, the lack of confirmation was poorly received, underlining that such verification is a critical aspect of the user experience.

Help/Support Access:





A dedicated help section or support contact within the application was strongly supported by users. The feature was seen as essential for resolving issues and providing assistance when needed. The scenario without such support was rated neutrally, indicating that while users may manage without it, having accessible support is a significant value-add.

4.0 Verification

4.1 Verification Approach

The verification of the Campus Event Check-in System will follow a structured, multi-phase approach to ensure the system meets the functional and non-functional requirements. The action includes details on how, who, when and where is described below.

- **How:**
  Individual modules of the system will be tested using unit testing, and then component relationships will be confirmed through integration testing. For instance, student databases and payment processing logins. While system testing evaluates overall performance, usability, and specification compliance, functional testing will verify each use case.

- **Who:**
  The verification process will be managed by the development team in collaboration with the university's IT department. Selected students, event organizers, and administrative staff are involved in testing to ensure real-world usability.

- **When:**
  Verification will be carried out at the end of each major development phase. Unit and integration tests will be executed continuously during development, while system and user acceptance tests will be conducted during the final stages before production deployment.

- **Where:**
  All verification activities will be performed in the staging environment set up by the university's IT development team. This environment will mirror the production setting to ensure test results reflect on the real-world conditions. User testing will also be carried out in live campus scenarios with simulated events to stimulate an actual usage.

4.2 Verification Criteria

To make sure that the system meets its specified functional and non-functional requirements, the following standards will be applied:

- **Authentication Accuracy**: The system must correctly verify MMU student credentials against the university's student database.
- **Event Registration Functionality**: Students must be able to successfully browse, filter and register for events without any dysfunctionality.
- **QR Code Generation and Scanning**: Users will receive a unique QR code for each event every time when an event is registered. Scanners should be able to read and validate the QR codes within 1 second.
- **Payment Integration**: The system must safely handle payments through third-party gateways for events that are paid for. The user's ticket status must be updated instantly upon successful transactions.
- **Check-in Performance**: Check-in validation at event entrances should take no longer than 2 seconds per student under standard network conditions.
- **Data Accuracy**: Attendance records, payment confirmations and registration data must be stored and displayed without inconsistencies across dashboards and reports.
- **System Uptime**: The platform should be available 99.5% of the time during university operational hours.
- **Security Compliance**: In accordance with university data protection policies, all student data must be safely stored and accessible only through encrypted channels.
- **Error Handling**: The system must display clear and meaningful error messages when there are issues such as invalid input, failed logins or payment issues, without exposing technical details.

5.0 Appendices

5.1 Assumptions and Dependencies

This section outlines the assumptions and external dependencies that may influence the development or functionality of the Campus Event Check-in System with Student ID and Payment Integration.

ID	Assumption / Dependency
A01	The system depends on the availability of the university's **Student Information System (SIS)** for user authentication.
A02	A stable internet connection is assumed to be available for all users and for accessing cloud services and APIs.
A03	MMU will provide secure API access to student data, including names, IDs, faculties, and program codes.
A04	The system assumes the availability and reliability of a **third-party payment gateway** for processing payments.
A05	Devices used for QR code scanning (e.g., mobile phones or scanners) are assumed to be compatible with the system.
A06	The system assumes that all students and staff have valid MMU credentials for login.
A07	Project implementation depends on access to MMU's server infrastructure or an approved cloud provider.

5.2 Acronyms and Abbreviations

Acronym	Defination
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
GDPR	General Data Protection Regulation
MMU	Multimedia University
NFC	Near Field Communication
PDPA	Personal Data Protection Act (Malaysia)
QR Code	Quick Response Code
RBAC	Role-Based Access Control
SIS	Student Information System
SRS	Software Requirements Specification
UI	User Interface
UX	User Experience

5.3 Glossary

Term	Defination
Check-in	The process where a student confirms their attendance at an event using QR code or student ID.
Dashboard	A visual interface summarizing important metrics, such as registrations and attendance.
Digital Ticket	An electronic ticket issued upon event registration or payment, containing a scannable QR code.
Event Organizer	A user role responsible for creating, managing, and monitoring events.
Feedback	Comments or ratings submitted by students after attending an event.