Software Requirements Specification

for

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

Version 1.0

Group No.: 08

Haizatul Nazirah Nizam binti Hairunizam	1231303504
Amirah Aisyah binti Azman	1221305806
Puteri Iman Balqis binti Muhammad Zulkey	1211306277
Heng Javenn	1221305928

Date: 25 May 2025

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this project is to create a simple and efficient digital system that allows students to check in to campus events using their student ID. This system will connect with the university's student database to confirm each student's identity and with the payment system to handle ticket purchases. By bringing these features together, the system will help event organizers manage registrations more easily, keep accurate records of who attended, and reduce long lines or confusion during events. It also makes the check-in process faster and more convenient for students, improving their overall experience at campus events.

1.2 Scope

The Campus Event Check-In System is a centralized digital platform designed to streamline the registration, payment, and attendance process for campus events. The system serves three main user roles which are Student, Event Organizer, and University Admin, with an integrated Payment Gateway to manage all financial transactions securely. Each role has distinct functions and access privileges to ensure a smooth and organized event experience.

Core Functionalities:

- **User Management:** Enables account registration, login, and role-based access. Profiles are securely maintained for each user.
- **Event Registration and Attendance:** Students can view upcoming events, register for them, and check in using their student id. Real-time identity verification is done using university records.
- **Payment Integration:** Students can make payments for event tickets or on-site purchases. The system verifies and processes payments securely via a payment gateway, and University Admin can resolve any payment issues.

- **Event Creation and Approval:** Event organizers can submit event requests, set event details, and manage event logistics. Events must be reviewed and approved by University Admins before going online.
- **Communication and Notification:** Event Organizers and University Admins can send announcements or system-generated notifications (e.g., registration confirmation, payment success, and reminders) to students.
- **Attendance Management:** Event Organizers can view and manage the list of attendees and generate attendance reports after each event.
- **System Monitoring and Reporting:** University Admins can monitor overall system activity, generate system-wide reports, and manage user accounts and roles.

1.3 Product overview

This project involves creating a digital check-in system for campus events that integrates with both the university's student identification database and payment processing system. The platform streamlines event attendance tracking while handling ticket verification and on-site purchases.

1.3.1 Product perspective

The **Campus Event Check-In System** is a standalone web-based application that integrates with existing university systems to improve event management. It uses the **Student ID Database** to verify student identities and the **University Payment System** to process payments. It also interacts with users (students, event organizers, and university admins) through a browser interface and optionally connects with the **University Access Control System** to manage physical entry to event venues. This system supports real-time attendance tracking, reporting, and payment verification, making it a vital part of the broader campus technology ecosystem.

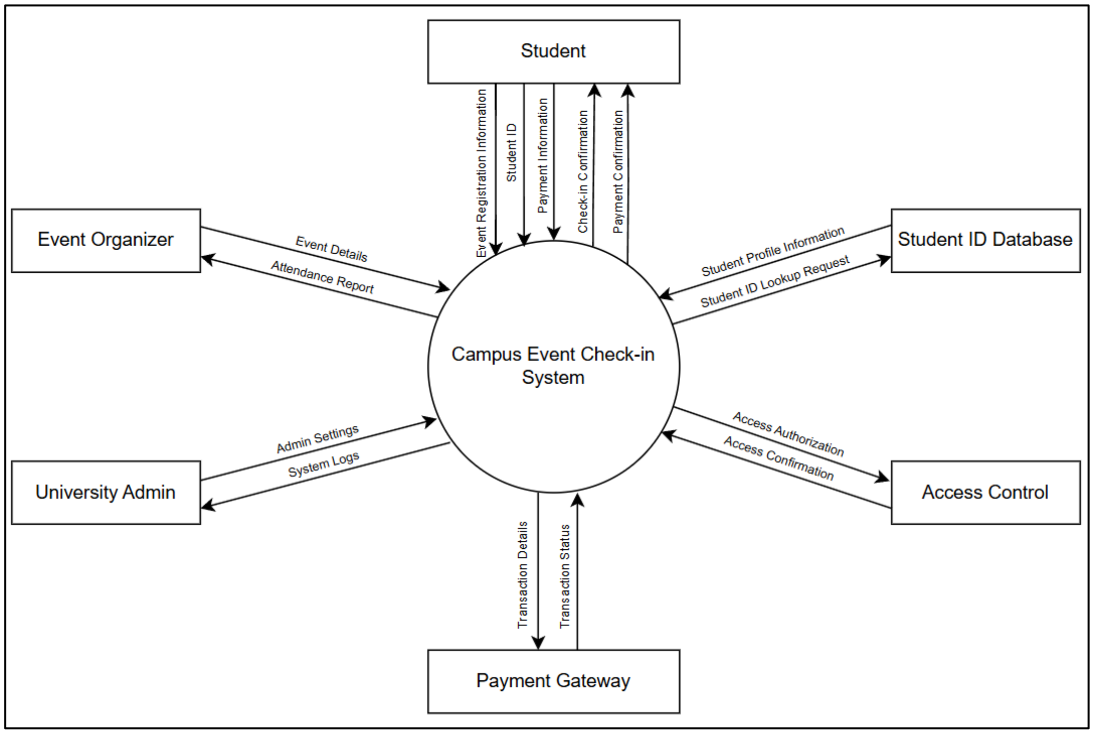


Figure 1.3.1.1: Campus Event Check-in System Context Diagram

1.3.2 Product functions

The Campus Event Check-In System is designed to simplify and digitalize the management of campus events. It integrates event registration, identity verification, and payment processing into one streamlined platform. The primary functions of the software are listed below:

i. **Account Registration:** Allows students and event organizers to create a new account in the system.
ii. **User Login:** Enables all users to log securely using their credentials.
iii. **Event Discovery:** Let students view a list of upcoming campus events.
iv. **Event Registration:** Allows students to register for available events through the platform.
v. **Payment Processing:** Handles online payments via the university's payment gateway.

vi. **Ticket and Confirmation:** Delivers registration confirmation and virtual tickets to students.
vii. **Event Check-in:** Verify student ID for entry into events.
viii. **Notification Delivery:** Send system messages, updates, and announcements to users.
ix. **Event Management:** Allows organizers to create, edit, cancel, and monitor events.
x. **Attendance Management:** Enables organizers to manage attendee lists and track participation.
xi. **Report Generation:** Provide detailed attendance and system activity reports to authorized users.
xii. **User and Access Control:** Allows university admin to manage user accounts and access permissions.
xiii. **System Monitoring:** Admin can oversee platform activity and ensure policy compliance.
xiv. **Payment Verification:** Admin can review and verify payment details and resolve issues.
xv. **Identity Verification:** Validate student identities using the university database.

1.3.3 User characteristics

The Campus Event Check-In System includes a variety of core functions that support event registration, identity verification, payment processing, and administrative controls. The table below outlines the primary functions the software is expected to perform:

Table 1.1.3.1 User Characteristics

Role	**Description**	**Required Knowledge**
Student	Use the system to register for events, check in with student ID, and make payments.	Basic computer or mobile usage, access to student ID, and using payment methods.
Event Organizer	Create and manage events, monitor attendance, verify check-ins, and handle on-site registration.	Event planning basics, system dashboard usage, and basic reporting.
University Admin	Oversee system usage, manage user roles, access reports, and ensure policy compliance.	System management, university policies, and data privacy knowledge.
Payment Gateway	Process transactions securely during registration or on-site payments.	Knowledge of payment processing, PCI compliance, and secure APIs.

1.3.4 Limitations

While the Campus Event Check-In System is designed to streamline event management and attendance tracking, there are several limitations that may affect its functionality or performance. These limitations should be considered during development and deployment:

i. The system is limited to managing campus-related events organized by official university departments or student bodies.
ii. Only students with a valid university ID can check in through the system.
iii. The system relies on stable internet connectivity to process real-time check-ins and payments.
iv. Payments can only be processed through the university's approved payment gateway. Third-party wallets or cash transactions are not supported.
v. The system does not support offline functionality for check-ins or ticket validation.
vi. Reports generated by the system are limited to data captured within the platform and do not include manual attendance records.
vii. User roles and permissions are predefined and cannot be customized beyond the four main roles (students, event organizers, university admin, payment gateway).
viii. The system may not support events held off-campus or in venues without access control or internet infrastructure.

1.4 Definitions

This section highlights the definitions for key terms used in the Software Requirements Specification (SRS) for the Campus Event Check-in System:

i. **Application**: A set of software programs designed to perform a specific function for the user. In this context, it refers to the Campus Event Check-in System that helps manage event registrations and attendance.

ii. **Attendance Management**: The functionality that allows event organizers to track and manage the list of attendees for each event, including generating reports on participation.

iii. **Campus Event Check-in System**: A digital platform designed to streamline the registration, payment, and attendance process for campus events, allowing students to check in using their student ID and facilitating event management for organizers and university administrators.

iv. **Data Privacy**: The principle of protecting personal information collected from users, ensuring that data is handled in compliance with relevant regulations, such as the Personal Data Protection Act (PDPA).

v. **Digital Ticket**: A virtual representation of a student's registration for an event, often in the form of a QR code, which is used for check-in purposes.

vi. **Event Creation**: The functionality that allows event organizers to submit new event requests, including details such as date, time, location, and capacity.

vii. **Event Organizer**: A user role responsible for creating, managing, and overseeing campus events, including submitting event requests and monitoring attendance.

viii. **Notification**: Automated messages sent to users regarding event confirmations, reminders, or updates, ensuring that students are informed about their registrations and event details.

ix. **Payment Gateway**: An external service integrated into the system that securely processes financial transactions for event ticket purchases and on-site payments.

x. **Registration**: The process by which students sign up for events, which includes providing necessary information and confirming their participation.

xi. **Student ID**: A unique identification number assigned to each student by the university, used for verifying identity and accessing various services within the Campus Event Check-in System.

xii. **University Admin**: A user role that oversees the overall operation of the Campus Event Check-in System, managing user accounts, event approvals, and ensuring compliance with university policies.

xiii. **User Management**: The process of creating, updating, and managing user accounts within the system, including role assignments and access permissions.

xiv. **User Roles**: Distinct categories of users within the system, including Students, Event Organizers, University Admins, and Payment Gateway, each with specific functions and access privileges.

xv. **Usability**: The ease with which users can navigate and interact with the Campus Event Check-in System, emphasizing user-friendly design and accessibility.

xvi. **Verify Student Identity**: Validates student details using the university student ID database to ensure only eligible students can register for events.

2. References

This Document is prepared in reference to the following documents:

I. ISO/IEC/IEEE 29148:2018 – International Standards

II. Chaturvedi, A., Singh, A., & Yadav, A. (2024). CU-EVENTS: A comprehensive event management system for university. International Journal for Research in Applied Science and Engineering Technology (IJRASET). https://www.ijraset.com/research-paper/cu-events-a-comprehensive-event-management-system-for-university

III. Mohd Akin, N. I. A., Othman, N., Bakar, S. A., & Saad, M. N. M. (2024). The development of mobile application for college event attendance system. Journal of Mathematics and Computing Science, 10(2). https://journal.uitm.edu.my/ojs/index.php/JMCS/article/view/4265

IV. Wang, Y., & Xu, H. (2021). Research on campus card and virtual card information system. In Proceedings of the 2021 International Conference on Wireless Communications, Networking and Applications (pp. 1047–1053). Springer. https://link.springer.com/chapter/10.1007/978-981-19-2456-9_117

V. Yang, S., & Wen, L. (2020). Design and research of virtual payment system in colleges and universities. Open Journal of Social Sciences, 8(6), 232–241. https://www.scirp.org/html/35-1763541_101114.htm

3. Requirements

3.1 Functions

This system's functions are categorized using the **Kano Model** into **Basic**, **Performance**, and **Attractive** functions. This approach helps identify which features are essential, which improve satisfaction, and which delight users.

3.1.1 Basics Functions

These are fundamental features that users expect as a minimum requirement for the system to function properly. Their absence will lead to immediate dissatisfaction and usability issues. They are considered essential for the system to be acceptable and usable.

Table 3.1.1.1 Register Account

Field	Details
Function	Register Account
Purpose	To allow students and event organizers to create accounts in the system.
Pre-condition	The user is not yet registered in the system.
Post-condition	A new user account is created and stored in the system.
Main Flow	1. User clicks "Create Account". 2. Enter required details. 3. Submit registration form. 4. System validates credentials. 5. System creates account and sends confirmation.
Alternate Scenario	If student ID is already registered, the system rejects the request with an error message.

Table 3.1.1.2 Login

Field	Details
Function	Login
Purpose	To allow users securely access the system using valid credentials.
Pre-condition	The user has already registered and has a valid username and password.
Post-condition	The user is successfully logged in and redirected to their dashboard.
Main Flow	1. User navigates to login page. 2. Enters username and password. 3. Clicks “Login” button. 4. System verifies credentials. 5. System grants access.
Alternate Scenario	If login credentials are invalid, an error message is displayed and access id denied.

Table 3.1.1.3 View Upcoming Event

Field	Details
Function	View Upcoming Event
Purpose	To allow students to browse upcoming events on campus.
Pre-condition	Student is logged in to the system.
Post-condition	A list of upcoming events is displayed to the student.
Main Flow	1. Student selects "Upcoming Events". 2. System fetches event data. 3. Events are listed with details.
Alternate Scenario	If no events are available, the system displays a "No events found" message.

Table 3.1.1.4 Register Event

Field	Details
Function	Register Event
Purpose	To allow students to register for events hosted on campus.
Pre-condition	Student is logged in and event is open for registration.
Post-condition	The student is marked as registered for the selected event.
Main Flow	1. Student browses event list. 2. Selects an event. 3. Clicks "Register". 4. System adds the student to the event's attendance list.
Alternate Scenario	If the event is full, the system shows "Registration Closed" message.

Table 3.1.1.5 Check-In to Event

Field	Details
Function	Check In to Event
Purpose	Allow students to verify attendance by scanning the student ID.
Pre-condition	Student is registered for the event.
Post-condition	Student is marked as “Checked-In”.
Main Flow	1. Student presents ID. 2. Event organizer verifies and updates attendance.
Alternate Scenario	Invalid or duplicate ID shows an error message.

3.1.2 Performance Functions

These functions directly influence user satisfaction. The better they are implemented and the more efficiently they perform, the more satisfied users will be. These are not strictly required for basic operation but are important for delivering a high-quality user experience.

Table 3.1.2.1 Make Payment

Field	Details
Function	Make Payment
Purpose	To allow students to pay for event tickets or services using a secure method.
Pre-condition	The student is logged in and has registered for an event that requires payment.
Post-condition	Payment is processed and recorded.
Main Flow	1. Student selects payment option. 2. Enters card/bank details. 3. Clicks "Pay". 4. Payment gateway processes transaction. 5. System confirms and stores result.
Alternate Scenario	If payment fails, the system notifies the student and prompts to retry.

Table 3.1.2.2 Receive Virtual Ticket

Field	Details
Function	Receive Virtual Ticket
Purpose	Provide students with a digital ticket after payment.
Pre-condition	Student has successfully paid.
Post-condition	Ticket is sent to student email or dashboard.
Main Flow	1. System confirms payment. 2. System generate ticket. 3. System send ticket to student.
Alternate Scenario	If ticket sending fails, system retries or stores ticket in user account.

Table 3.1.2.3 Manage Event

Field	Details
Function	Manage Event
Purpose	To let event organizers control event details.
Pre-condition	Event organizer is logged in.
Post-condition	Event is created, updated, or deleted.
Main Flow	1. Organizer opens dashboard. 2. Adds/edits/deletes event. 3. System updates event data.
Alternate Scenario	If validation fails, system prevents save and shows error.

Table 3.1.2.4 Generate Attendance Report

Field	Details
Function	Generate Attendance Report
Purpose	To generate event participation reports.
Pre-condition	Event has participants.
Post-condition	Report is generated.
Main Flow	1. Event organizer selects report option. 2. System compiles data. 3. Report is shown/exported.
Alternate Scenario	If data is missing, system shows empty or partial report.

Table 3.1.2.5 View Registered Event

Field	Details
Function	View Registered Event
Purpose	To let students see which events they've signed up for.
Pre-condition	User is logged in.
Post-condition	User sees list of registered events.
Main Flow	1. User goes to "My Events". 2. System shows list based on student ID
Alternate Scenario	If user hasn’t registered for anything, system shows empty message.

Table 3.1.2.6 Manage User Account

Field	Details
Function	Manage User Account
Purpose	To allow admins to update or deactivate accounts.
Pre-condition	Admin is logged in.
Post-condition	User account is updated.
Main Flow	1. Admin searches for user. 2. Updates or deactivates account. 3. System saves changes.
Alternate Scenario	If user not found, system shows error message.

3.1.3 Attractive Functions

These are unexpected or innovative features that delight users when present. While not essential for the core functionality of the system, they enhance user engagement and satisfaction. Their absence does not cause dissatisfaction, but their presence adds significant value.

Table 3.1.3.1 Send Notification

Field	Details
Function	Send Notification
Purpose	To keep users informed with updates or reminders.
Pre-condition	Event or system update exists.
Post-condition	Notification is sent.
Main Flow	1. System triggers notification. 2. Notification is sent via in-app notification.
Alternate Scenario	If notification delivery fails, system retries or logs error.

Table 3.1.3.2 Submit/ Cancel Event Request

Field	Details
Function	Submit/Cancel Event Request
Purpose	To let organizers propose or cancel events.
Pre-condition	Event organizer is logged in.
Post-condition	Request is submitted or cancelled.
Main Flow	1. Event organizer fills request form. 2. System submits to admin. 3. Admin notified
Alternate Scenario	If validation fails, system blocks submission.

Table 3.1.3.3 Resolve Payment Issue

Field	Details
Function	Resolve Payment Issue
Purpose	To let admins handle payment disputes.
Pre-condition	Payment issue exists and admin is logged in.
Post-condition	Payment issue resolved.
Main Flow	1. Admin review issue. 2. Confirm payment status. 3. System updates record.
Alternate Scenario	If issue is unresolved, system flags for follow-up.

Table 3.1.3.4 Send Announcement

Field	Details
Function	Send Announcement
Purpose	To allow event organizers to send custom messages to attendees.
Pre-condition	Event exists and event organizer is logged in.
Post-condition	Message is delivered to student.
Main Flow	1. Event organizer composes message. 2. Event organizer clicks “Send”. 3. System sends message.
Alternate Scenario	If student not found or invalid, system prevents send and alerts user.

3.2 Performance Requirements

The Campus Event Check-in System is expected to deliver exceptional performance, especially during large-scale campus events where hundreds or thousands of students interact with the platform simultaneously. The system must:

i. Guarantee an average response time of under 2 seconds for all key functions such as login, check-in, registration, and payment.
ii. Support up to 10,000 concurrent users without lag, crashes, or performance drops.
iii. Perform real-time synchronization of student check-in status and payment verification, ensuring the system reflects up-to-date information for event organizers and administrators.
iv. Provide instant QR code scanning results and digital ticket validation for seamless entry at event venues.
v. Maintain high performance even under varied network conditions (e.g., mobile data, campus Wi-Fi).

3.3 Usability Requirements

User-friendliness is a top priority to promote engagement and ease of use for all stakeholders. The system shall:

i. Be designed with a mobile-first responsive interface, enabling smooth navigation on smartphones, tablets, and desktops.
ii. Ensure accessibility for students with disabilities, including visual and motor impairments, in compliance with WCAG 2.1 standards.
iii. Offer simplified registration and one-click check-in processes to minimize user effort.
iv. Present information using clear visual hierarchy and feedback indicators (e.g., loading animations, confirmations).
v. Include multi-language options if feasible, accommodating international users.
vi. Provide contextual help tips, tooltips, and FAQ access to support users without technical knowledge.

3.4 Interface Requirements

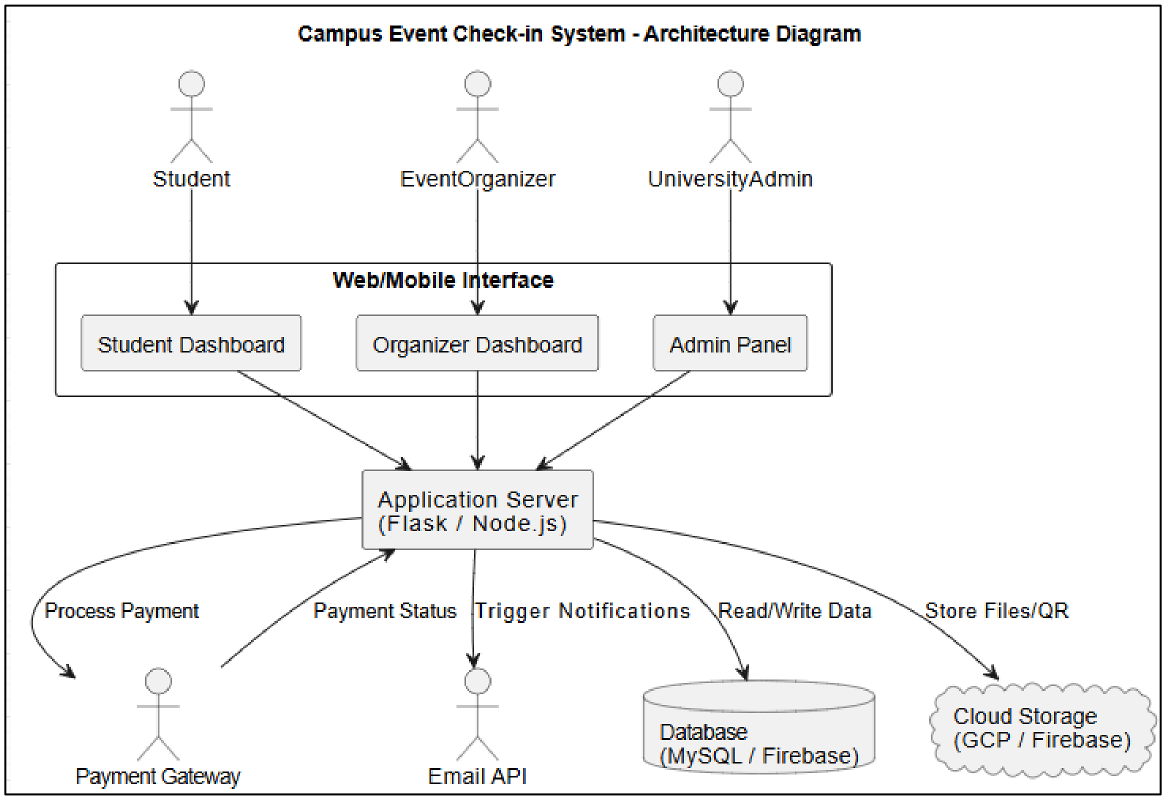


Figure 3.4.1: System Architecture Diagram

The system interface must be modular and consistent across user roles. Key components include:

i. Student Dashboard: Displays event listings, personal check-in history, QR code passes, and notifications.
ii. Organizer Dashboard: Includes tools for event creation, approval submission, participant tracking, and communication features.
iii. Admin Panel: Grants control over user accounts, event approvals, payment issue resolution, and system activity monitoring.
iv. Payment Gateway API: Integrated for real-time transaction processing and refund handling.
v. Student ID Authentication API: Validates identity through university records for secure logins and event access.

3.5 Logical Database Requirements

The system shall implement a relational database structure to support secure, normalized storage of all core information:

i. Users Table: Contains user credentials, roles, and profile details.
ii. Events Table: Stores event metadata including organizers, venue, capacity, and timing.
iii. Registrations Table: Links users to events with status (registered, checked in, canceled).
iv. Payments Table: Logs payment IDs, status, amount, and timestamp for tracking and reconciliation.
v. Feedback Table: Captures post-event user ratings, comments, and satisfaction scores.
vi. Badges/Rewards Table: Manages loyalty achievements and redemption tracking.

Data Flow Overview:

i. Student registers → payment initiated → confirmation issued → QR code generated.
ii. Check-in scanned → student status updated → visible in dashboard analytics.
iii. Post-event → feedback submitted → stored and made accessible to organizers/admins.

All relationships must be enforced with foreign key constraints to ensure referential integrity and prevent orphan records.

3.6 Design Constraints

The development and design of the system shall adhere to specific branding and deployment limitations:

i. All visual elements (buttons, colors, fonts) shall follow the university's visual identity guidelines.
ii. The platform must be mobile-first, prioritizing usability on small-screen devices commonly used by students.
iii. The system shall function entirely via web browser without needing external app installations.

iv. Use of open-source technologies is preferred to maintain cost-effectiveness and flexibility.
v. Backend must be scalable to support future modules such as public event listings or third-party integrations.

3.7 Software System Attributes

To provide a dependable experience, the software shall fulfill the following quality attributes:

i. Security: All data transmitted and stored must be encrypted (TLS/HTTPS). Authentication tokens and access rights must be enforced across all endpoints.
ii. Privacy & PDPA Compliance: The system must request explicit consent before collecting personal data. Only essential user data should be stored, and data access must be role-based.
iii. Data Protection: Daily backups must be automated. Data should be stored redundantly across multiple servers or cloud regions.
iv. High Availability: The system should achieve 99.9% uptime, particularly during active events.
v. Scalability: Backend infrastructure must accommodate spikes in traffic, especially during peak hours or high-profile events.
vi. Maintainability: Source code must be documented and modular, enabling easy updates or bug fixes. CI/CD pipelines are encouraged for efficient deployment.
vii. Auditability: All administrative actions (e.g., approval, deletion, refund processing) should be logged for transparency and security audits.

These attributes align with university policies, student expectations, and national legal frameworks, forming the basis of a trustworthy and efficient event check-in system.

3.8 Supporting Information

3.8.1 Use Case Diagram

The following use case diagram illustrates the interactions between the three main actors—Student, Event Organizer, and Admin—with the Campus Event Check-in System. It provides a visual summary of the functional requirements specified in Section 3.1.

This diagram was created based on detailed system functionality analysis and reflects the core features elicited through surveys, observations, PBR, and brainstorming. It is accessible at the following link for viewing:

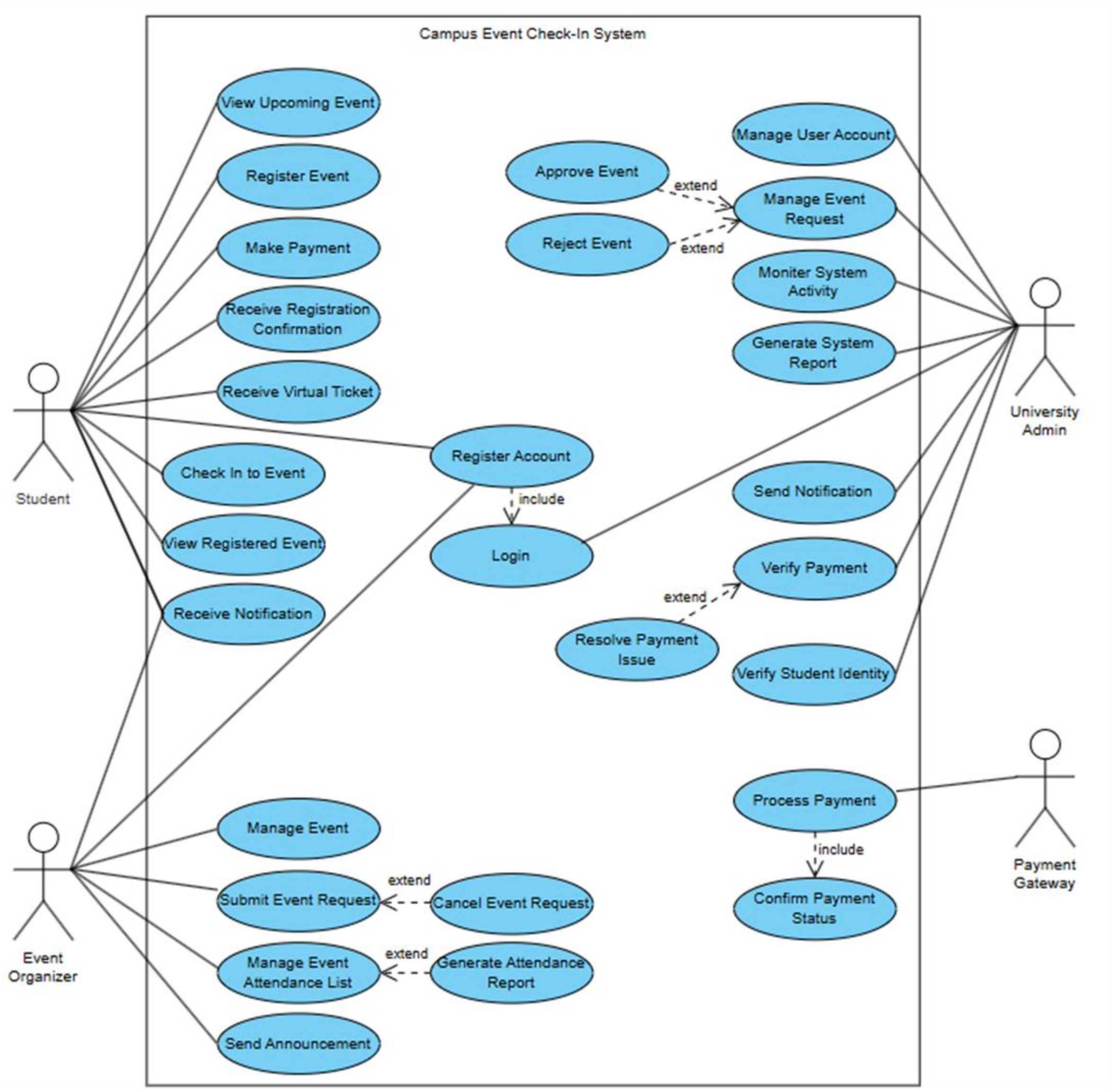


Figure 3.8.1: Use Case Diagram – Campus Event Check-In System

Primary Actors and Associated Use Cases:

1. **Student**

   Students are the end-users who interact with the system primarily to register for and attend events.

Table 3.8.1.1 Student Use Case

Use Case	**Description**
View Upcoming Events	Allows students to browse and view upcoming campus events listed in the system.
Register for Events	Enables students to enroll in selected events. This may trigger the payment workflow if the event is not free.
Make Payment	Facilitates online payment via a secure payment gateway for events requiring a ticket purchase.
Receive Registration Confirmation	Upon successful registration and payment, students receive automated confirmation via system notifications.
Receive Digital Ticket	Issues a QR code or virtual ticket to the student for event check-in purposes.
Check In to Event	Enables students to check in at the event location using their Student ID or virtual ticket.
View Registered Events	Displays a list of events the student has registered for, along with event details.
Receive Notifications	Sends system-generated alerts regarding upcoming events, confirmations, and reminders.

2. **Event Organizer**

Event organizers are responsible for event creation, monitoring, and management.

Table 3.8.1.2 Event Organizer Use Case

Use Case	**Description**
Submit Event Request	Allows event organizers to propose new events, including necessary details, for administrative review.
Cancel Event Request *(extend)*	Enables organizers to withdraw a pending or approved event before it takes place.
Manage Event Details	Allows modification of event data post-approval, including updating the schedule, venue, or capacity.
Verify Student Identity	Ensures that students checking in at the event are legitimate participants using university records.
View Attendance List	Provides real-time or post-event access to the list of students who have checked in.
Generate Attendance Report *(extend)*	Compiles and exports attendance data for reporting or evaluation purposes.
Send Announcements	Allows the organizer to broadcast updates or reminders to students registered for the event.

3. **University Admin**

   University Admin users oversee and manage system-wide activities, ensuring events meet institutional requirements and resolving administrative issues.

Table 3.8.1.3 University Admin Use Case

Use Case	**Description**
Manage Event Requests	Provides administrative control to review, approve, or reject event proposals submitted by organizers.
Approve/Reject Event *(extend)*	Grants or denies approval to event requests based on institutional criteria.
Manage User Accounts	Enables account management tasks, including role assignment, user creation, suspension, or deletion.
Monitor System Activity	Tracks system logs, user behavior, and event participation across the platform.
Resolve Payment Issues *(extend)*	Facilitates administrative handling of payment failures, disputes, or refunds.
Generate System Reports	Produces comprehensive system-wide reports on events, attendance rates, and payment summaries.
Send Notifications	Distributes critical alerts, announcements, or instructions to targeted user groups.

4. **Payment Gateway**

   The Payment Gateway is an external system integrated for secure financial transactions.

Table 3.8.1.4 Payment Gateway Use Case

Use Case	**Description**
Process Payment	Executes the transaction between the student and the event system for ticketed events.
Confirm Payment Status *(include)*	Communicates the result of the transaction (success/failure) to the main system for confirmation and access control.

3.8.2 Sequence Diagram

Student Event Registration & Check-In Flow

The student views available events, registers, pays via an integrated gateway, and checks in on the event day. The attendance module logs the record only if validation passes.

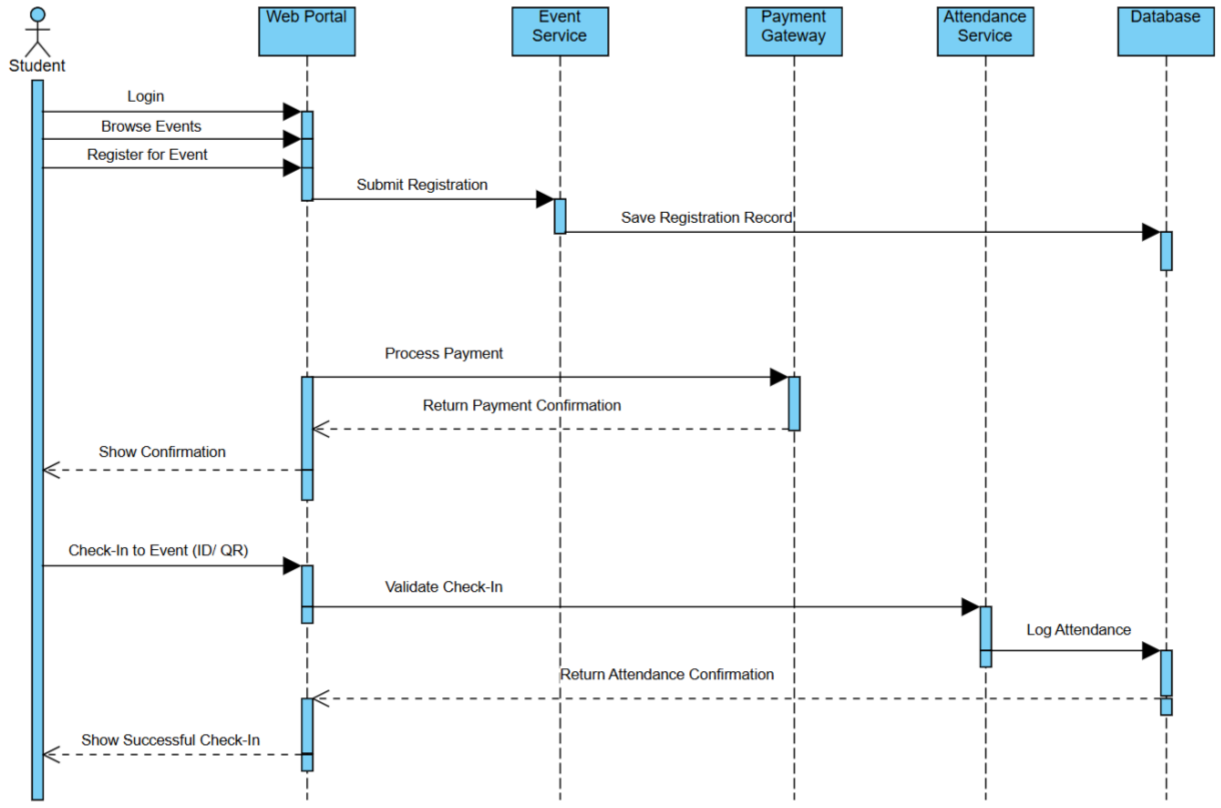


Figure 3.8.2.1: Student Event Registration & Check-In Flow

Event Organizer Dashboard Monitoring

This diagram shows how event organizers interact with the system to monitor attendance and manage event logistics. After logging in, organizers can view event details, fetch real-time attendance data, and export reports. This enhances operational efficiency and transparency for event tracking.

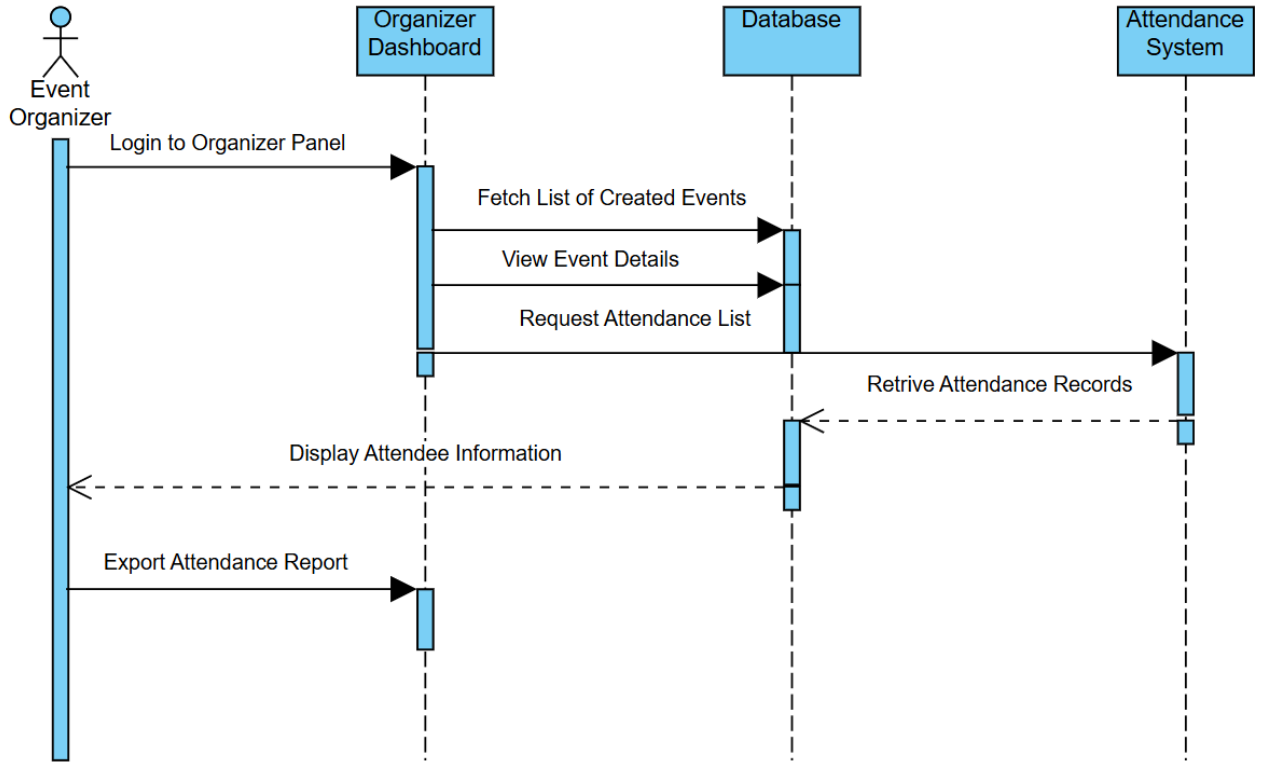


Figure 3.8.2.2: Event Organizer Dashboard Monitoring

Admin Event Approval Process

This sequence outlines how administrators manage event approvals. When an event is submitted, the system notifies the admin, who then logs in to review event details. Upon approval or rejection, the system updates the event status and notifies the organizer. This ensures institutional control and compliance before event publishing.

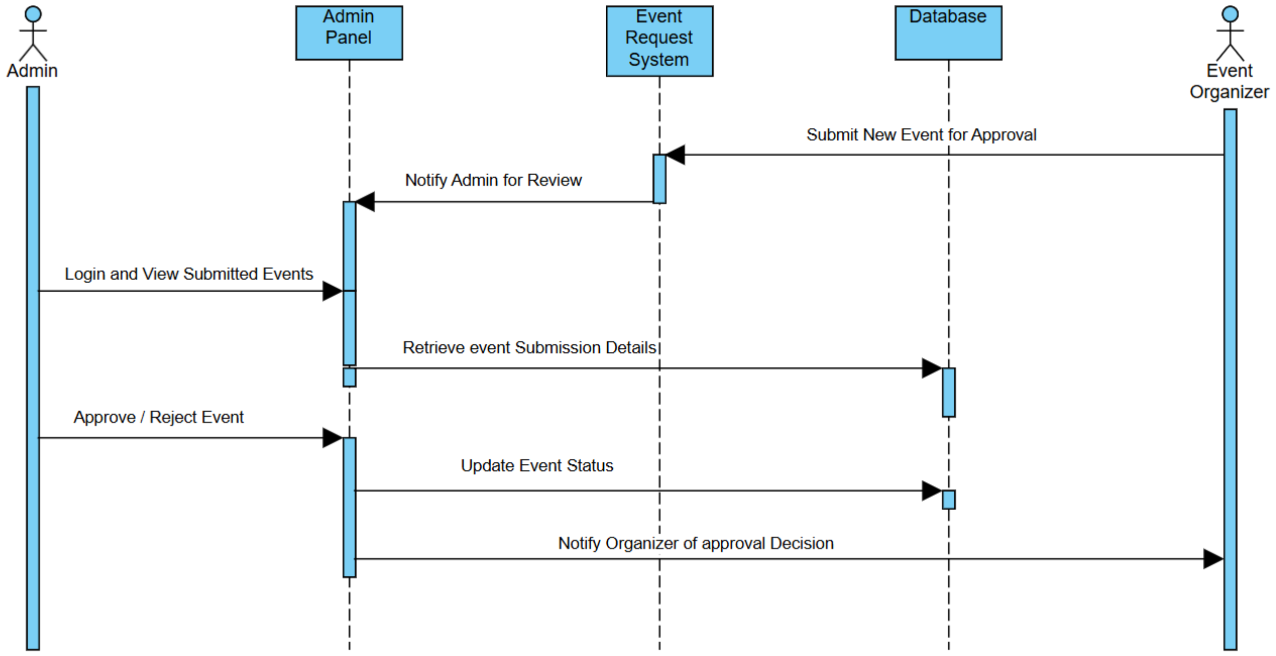

Admin
Admin Panel
Event Request System
Database
Event Organizer
Submit New Event for Approval
Notify Admin for Review
Login and View Submitted Events
Retrieve event Submission Details
Approve / Reject Event
Update Event Status
Notify Organizer of approval Decision

Figure 3.8.2.3: Admin Event Approval Process

Admin Resolves Payment Issues

This diagram outlines the process for resolving payment-related issues. When a student reports a problem, the admin retrieves the transaction record and checks it with the payment gateway. The admin then informs the student of the resolution or further steps.

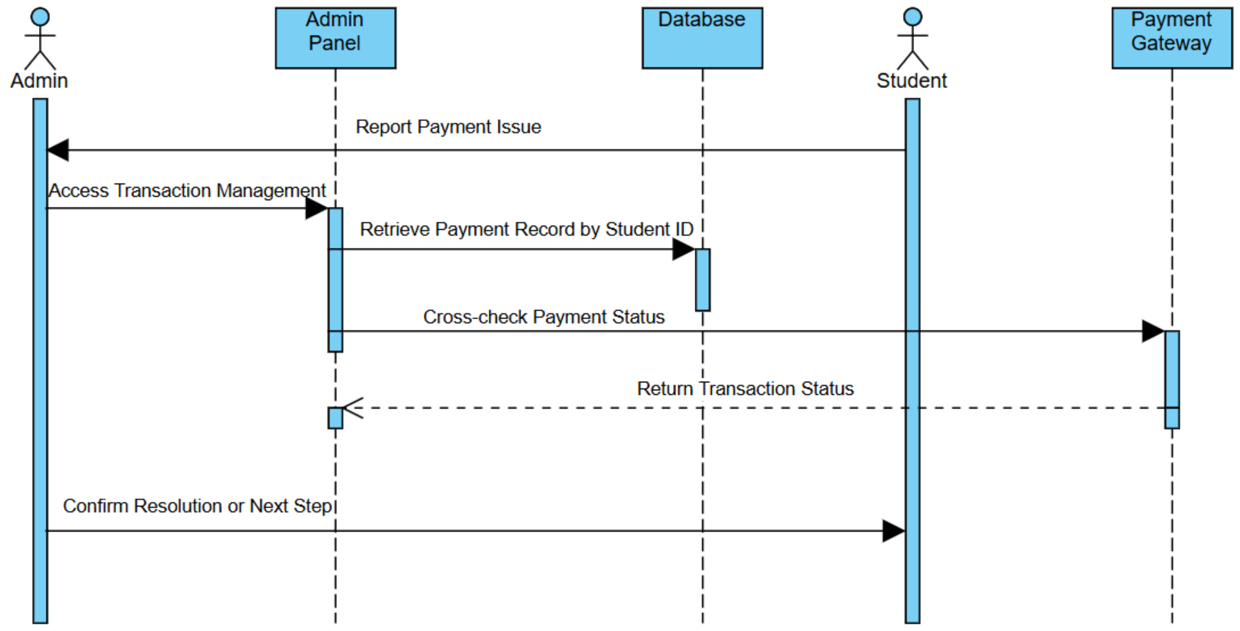

Admin
Admin Panel
Database
Student
Payment Gateway
Report Payment Issue
Access Transaction Management
Retrieve Payment Record by Student ID
Cross-check Payment Status
Return Transaction Status
Confirm Resolution or Next Step

Figure 3.8.2.4: Admin Resolves Payment Issues

Student Views Registered Events and Receives Notifications

This diagram models the process where a student logs in to view their registered events and receives system-generated reminders for upcoming ones. It supports the "View Registered Events" and "Receive Notifications" use cases described in the SRS and enhances usability by improving communication between the system and students.

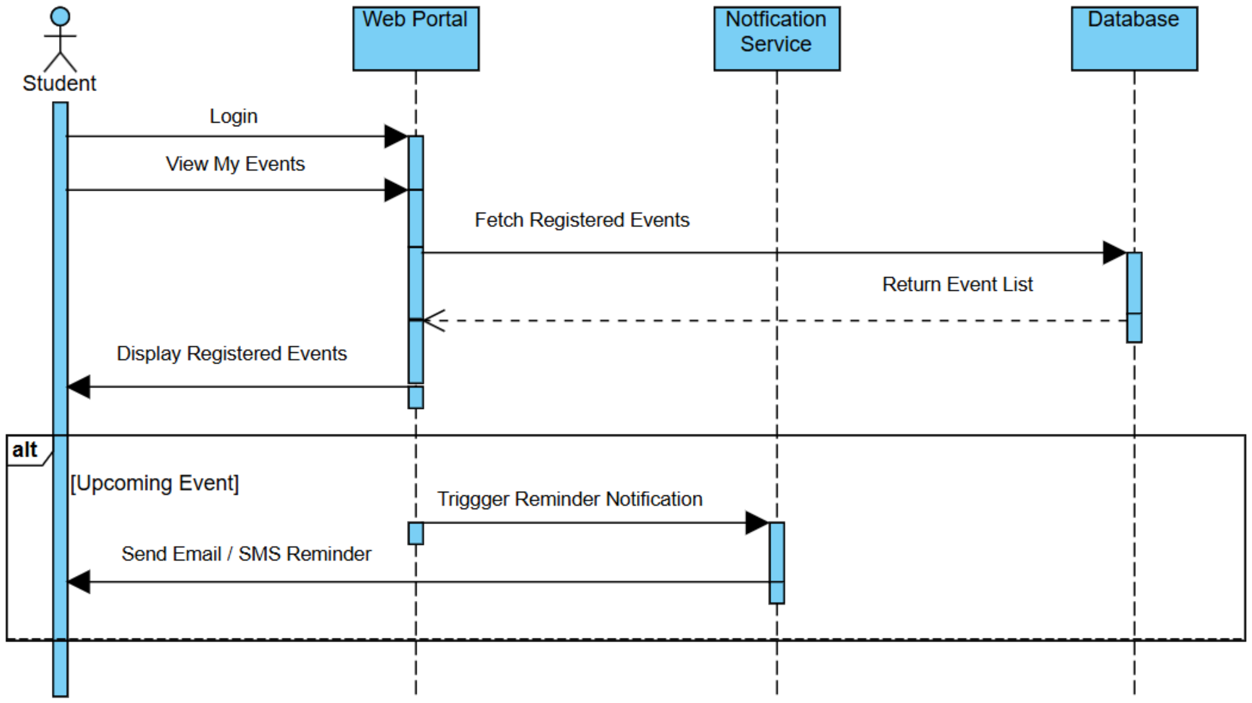


Figure 3.8.2.5: Student Views Registered Events and Receives Notifications

Student Receives Digital Ticket (QR Code)

After a student completes registration and payment, the system generates a unique digital ticket (QR code). This ticket is used during check-in and ensures quick verification, supporting the "Receive Digital Ticket" use case.

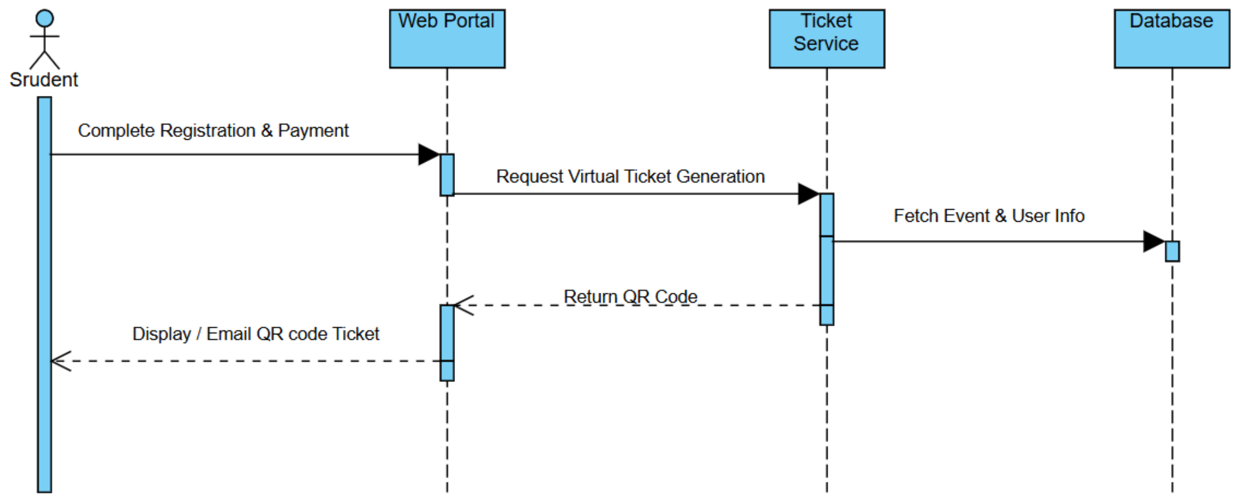

Srudent
Web Portal
Ticket Service
Database
Complete Registration & Payment
Request Virtual Ticket Generation
Fetch Event & User Info
Return QR Code
Display / Email QR code Ticket

Figure 3.8.2.6: Student Receives Digital Ticket (QR Code)

Event Organizer Sends Announcements

This diagram illustrates how event organizers send announcements or reminders to register students. It helps enhance communication and coordination before events

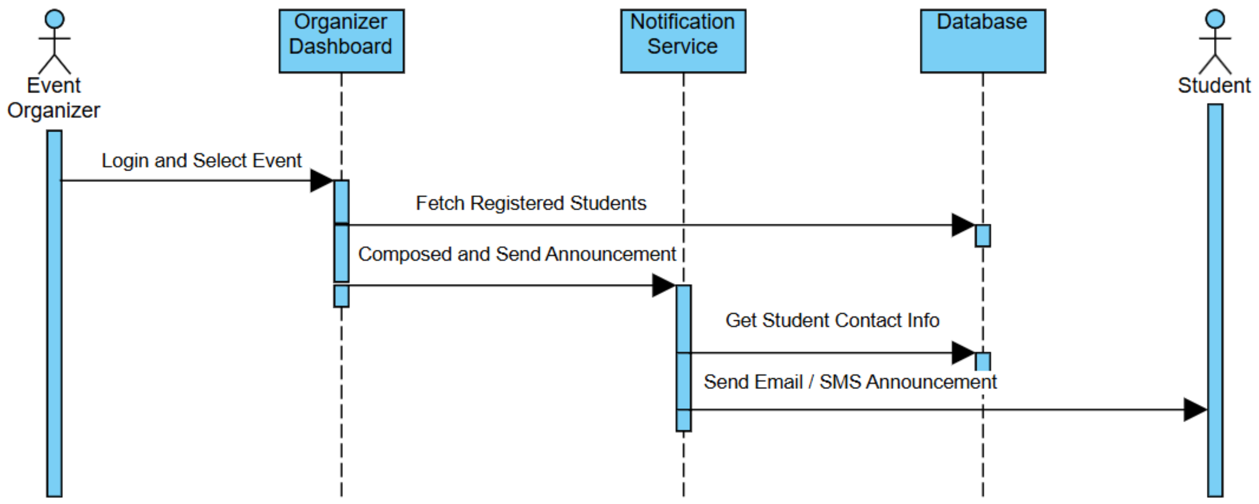


Figure 3.8.2.7: Event Organizer Sends Announcements

Student Views Upcoming Events

This diagram models the student flow for browsing upcoming events. After logging in, the student requests available event listings. The Event Service queries the database and returns a list, which is then displayed to the student. This supports the "View Upcoming Events" use case and ensures students are informed of opportunities on campus.

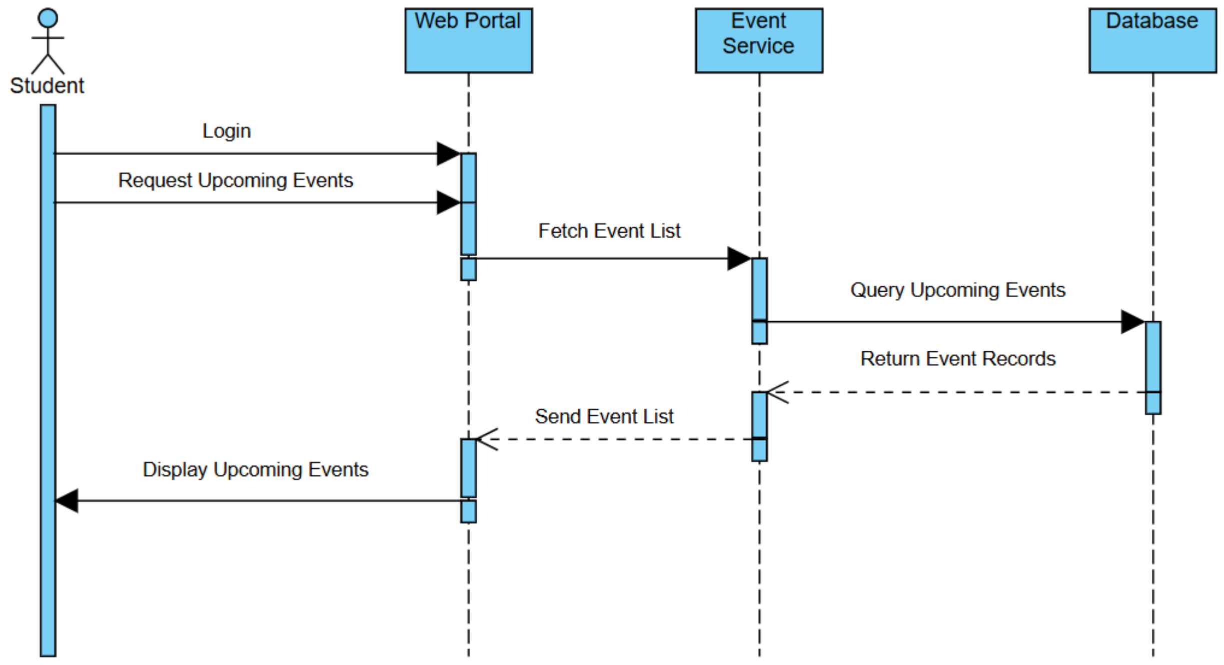


Figure 3.8.2.8: Student Views Upcoming Events

Event Organizer Cancels Event Request

This sequence shows how event organizers cancel a scheduled or pending event. The cancellation is routed through the dashboard and confirmed by the Event Service, which updates the event record in the database. This use case aligns with administrative control and event logistics management.

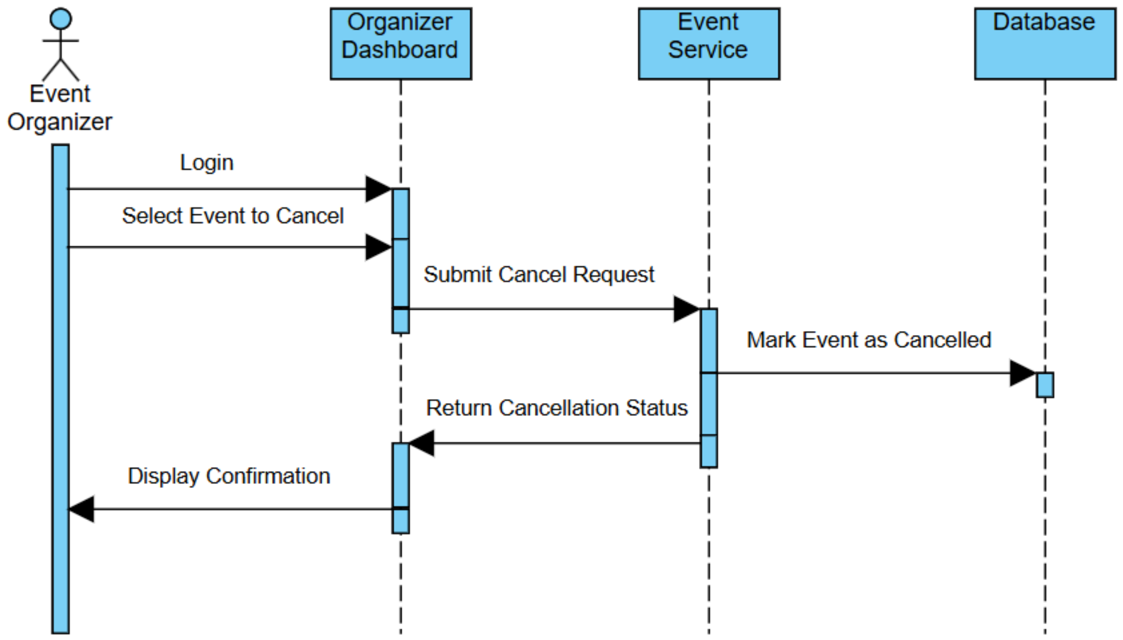


Figure 3.8.2.9: Event Organizer Cancels Event Request

Event Organizer Manages Event Details

This sequence diagram illustrates how an event organizer modifies event details such as time, venue, or capacity. The update request flows from the dashboard to the Event Service, which commits the changes to the database and provides confirmation.

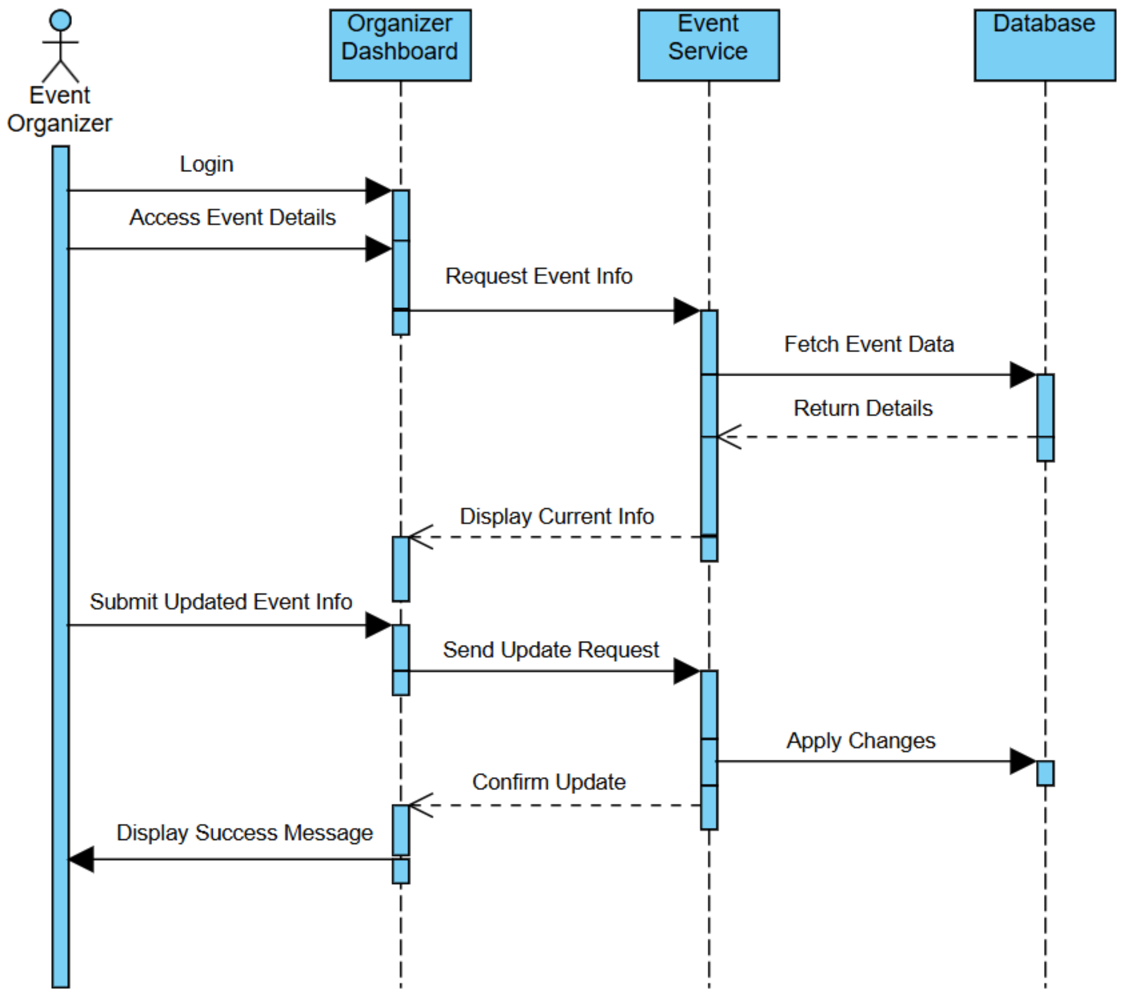


Figure 3.8.2.10: Event Organizer Manages Event Details

Admin Manages User Accounts

This diagram describes how an admin manages user accounts, including retrieving the user list, updating roles, or suspending access. The Admin Panel connects to the User Service which then communicates with the user database.

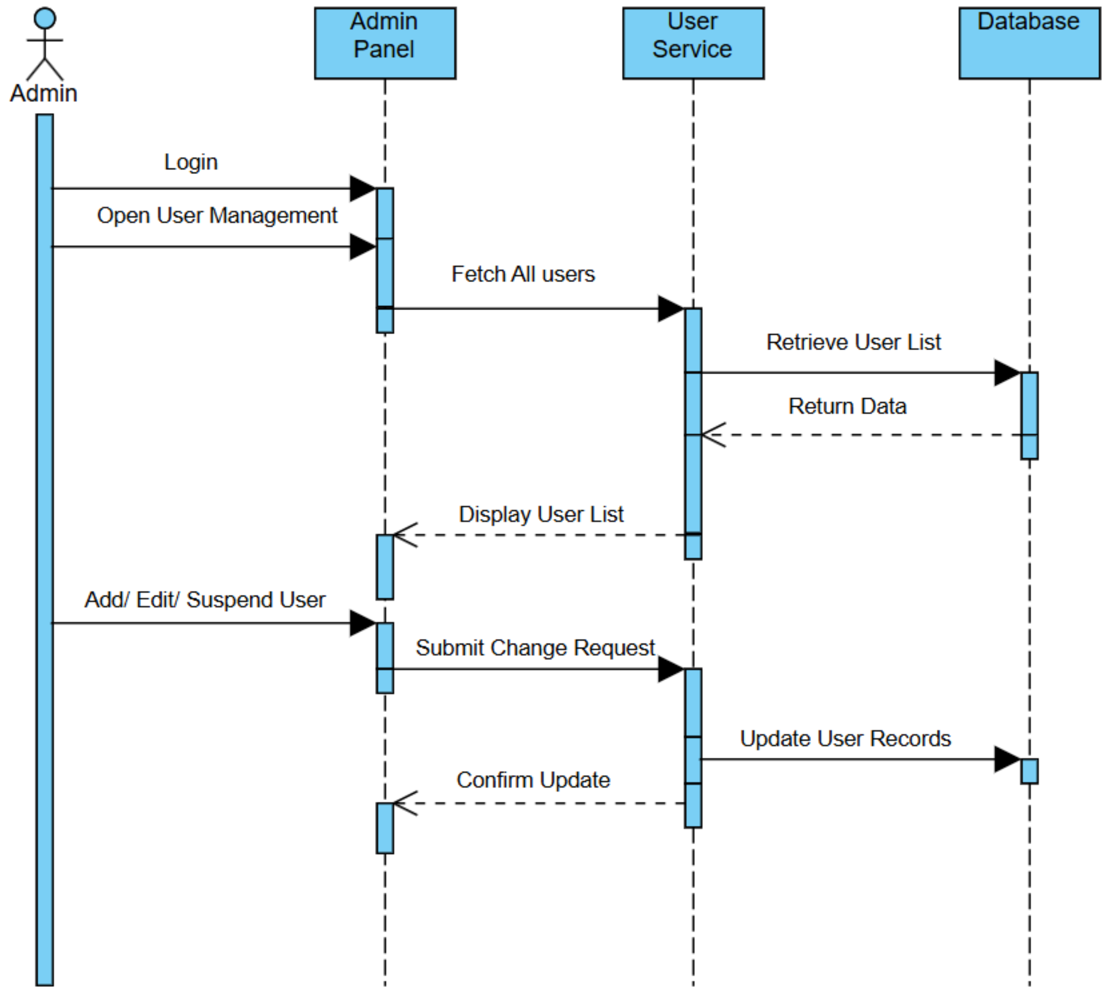


Figure 3.8.2.11: Admin Manages User Accounts

Admin Monitors System Activity

This diagram models how an admin reviews system-wide activity such as logins, transactions, and event registrations. The log data is stored centrally and presented via the Admin Panel for security auditing and operational visibility.

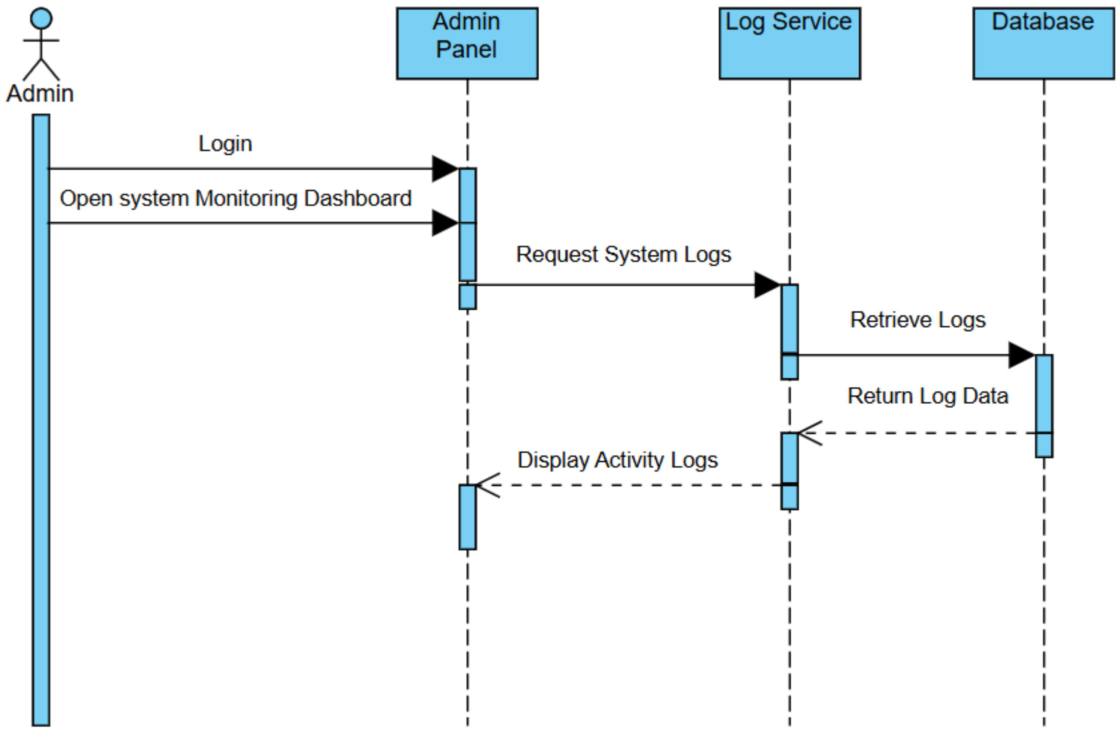


Figure 3.8.2.12: Admin Monitors System Activity

Admin Generates System Reports

This diagram represents the workflow of an admin generating reports about events, user activity, and payments. The Admin Panel triggers the Report Service to compile data from multiple modules and return a formatted report.

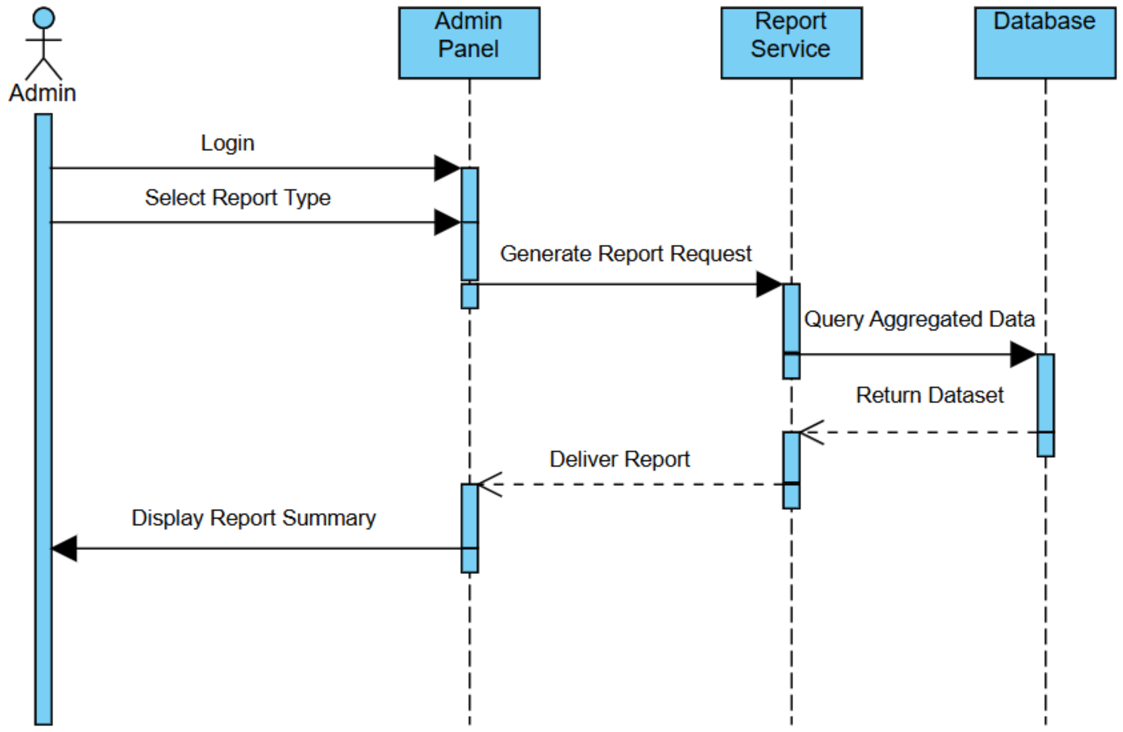


Figure 3.8.2.13: Admin Generates System Reports

3.8.3 Questionnaire Results and Charts

This questionnaire was distributed to students, organizers, and university administrators to gather functional and non-functional requirements for the Campus Event Check-In System. The form was created using Google Forms and received a total of 26 responses.

Link to the Google Form (preview):
https://docs.google.com/forms/d/1oau6dI1DVgZERLIKcSRsRTYqDozKpNOX7Jot2t4Rfbw/preview

Link to the Google Form (Responses):
https://docs.google.com/forms/d/1oau6dI1DVgZERLIKcSRsRTYqDozKpNOX7Jot2t4Rfbw/edit#responses

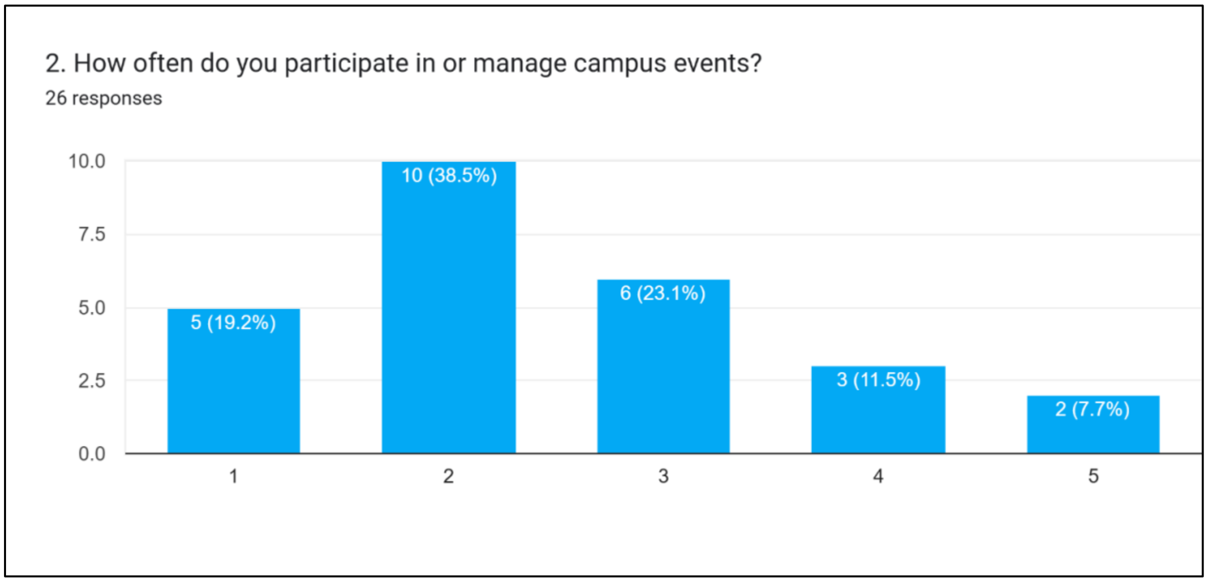


Figure 3.8.3.1: Frequency of Participation in Campus Events

Majority of respondents indicated moderate to frequent participation in campus events. This supports the system's relevance and justifies the need for a streamlined platform to handle recurring check-ins and registrations.

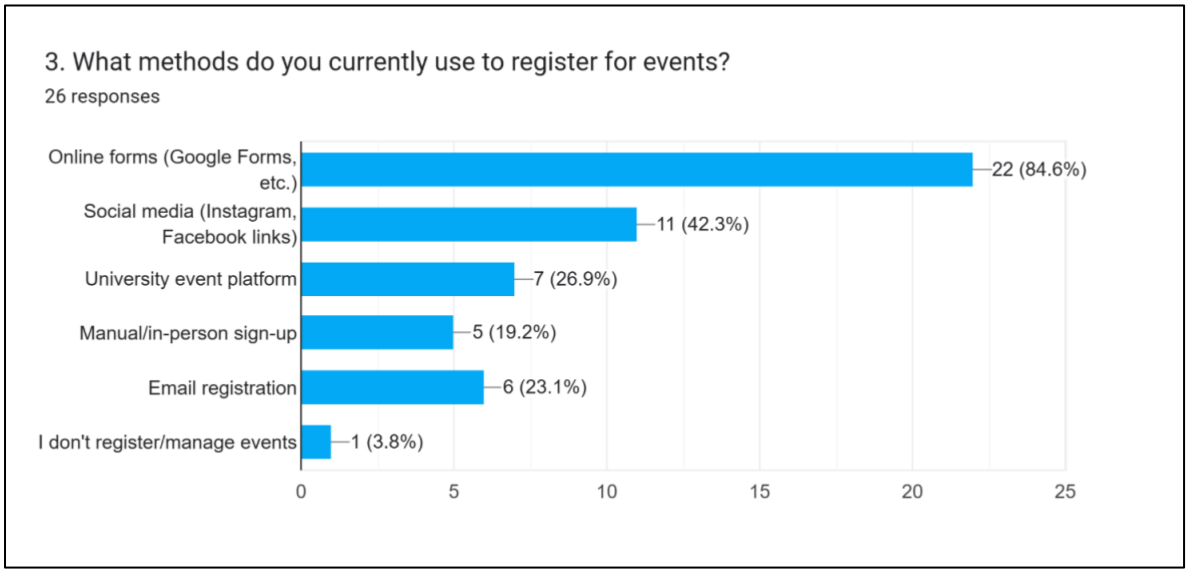


Figure 3.8.3.2: Current Registration Methods Used by Students

Many students currently rely on manual methods such as Google Forms or on social media. This highlights inefficiencies in the current system and validates the need for a centralized, digital registration platform.

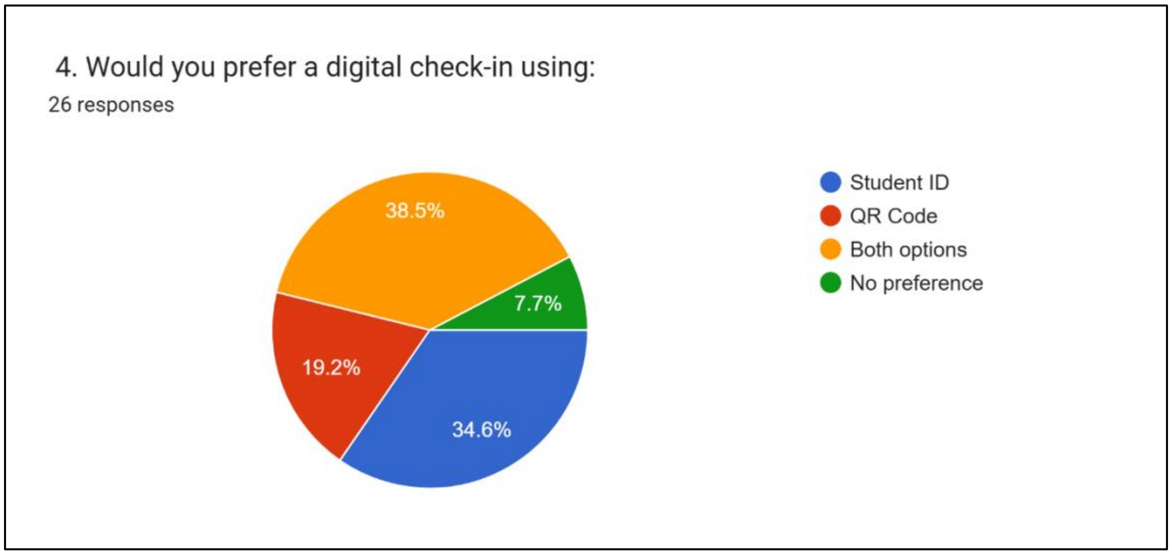


Figure 3.8.3.3: Preferred Digital Check-In Method

Respondents were fairly split, with 38.5% prefer using both QR code and Student ID, while 34.6% prefer only Student ID and 19.2% prefer only QR. This result justifies implementing dual check-in methods to support flexibility and reduce technical failure risk.

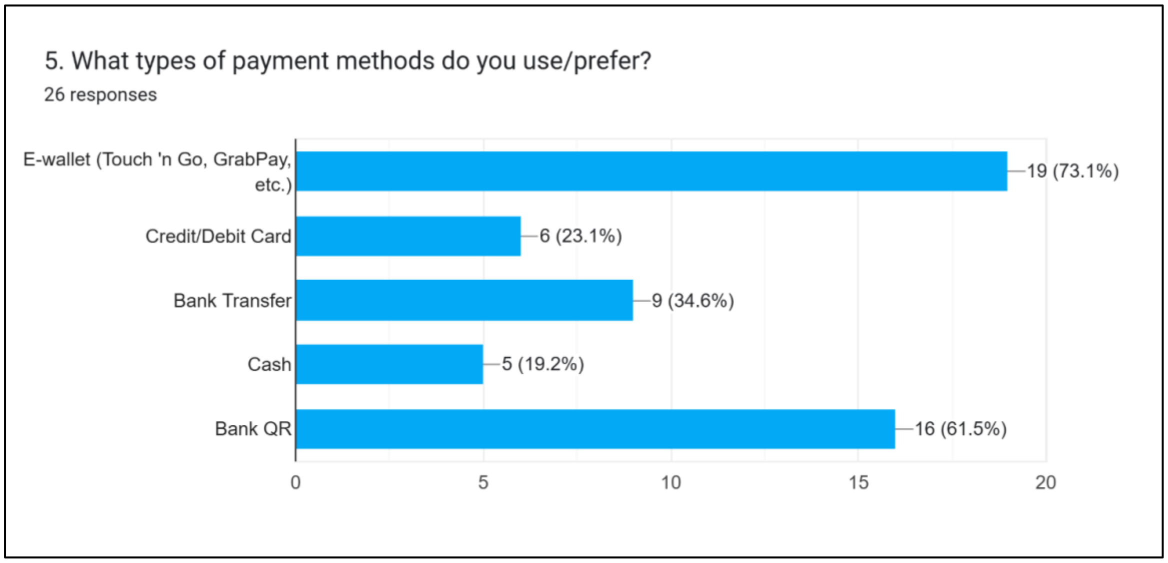


Figure 3.8.3.4: Preferred Payment Methods

A large majority, 73.1%, prefer e-wallets, followed by 61.5% who use bank QR. This supports integration with Malaysia's common online payment gateways for event ticket purchases.

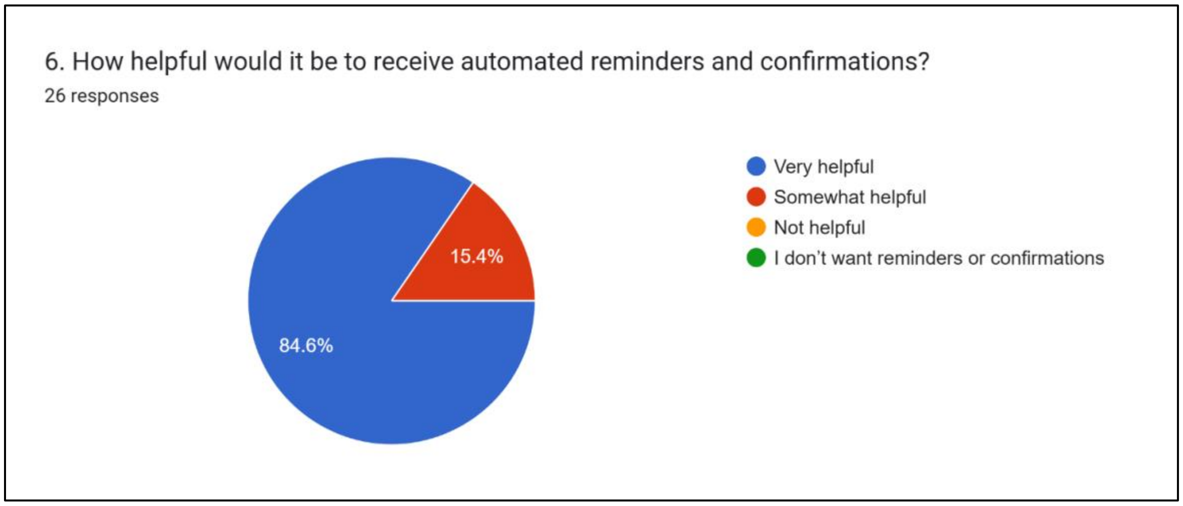


Figure 3.8.3.5: Importance of Automated Notifications

Overwhelmingly, users rated this feature as very helpful. This confirms the need for an automated notification system as a performance attribute in the system.

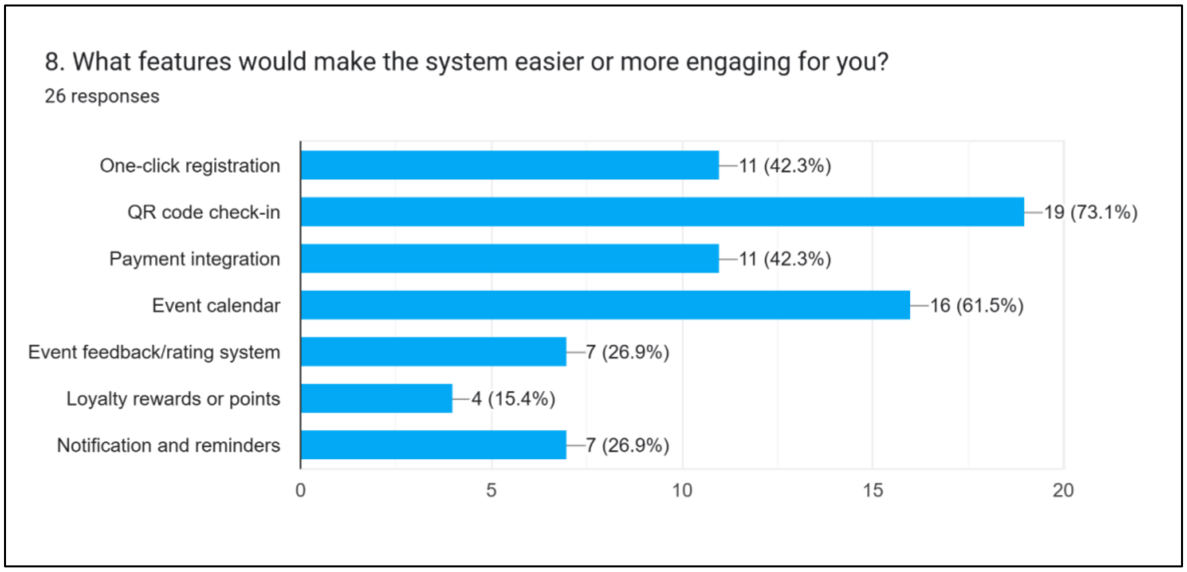


Figure 3.8.3.6: Most Desired System Features

Features like QR code check-in, one-click registration, payment integration, and event calendar were the most selected. This supports the inclusion of these functionalities in the system's requirement list and Kano Model classification.

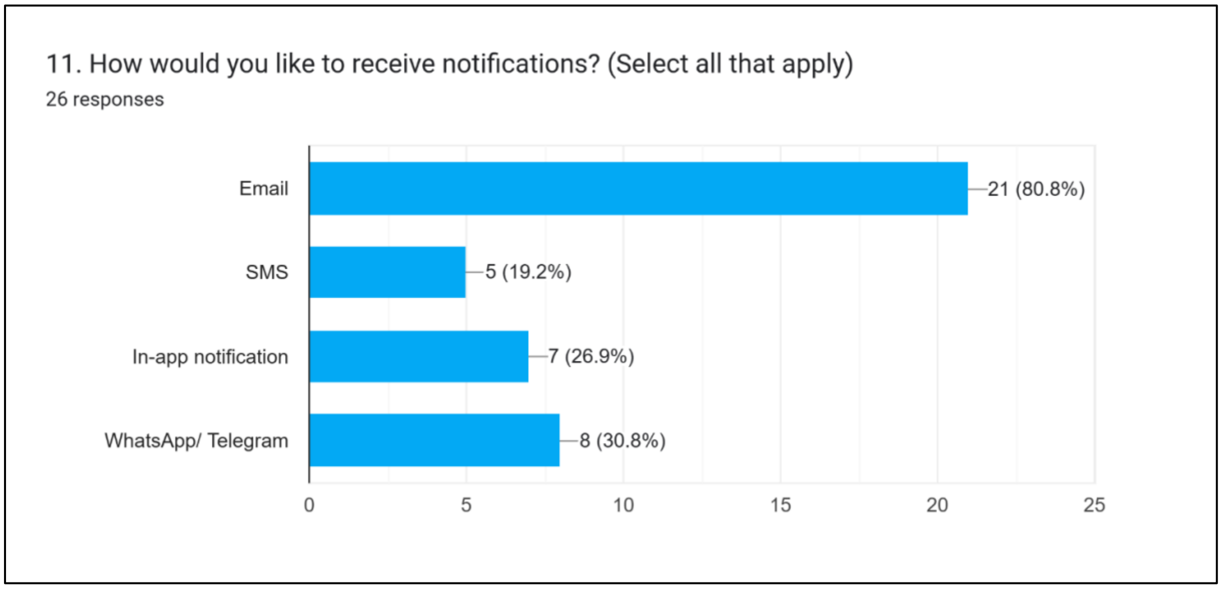


Figure 3.8.3.7: Preferred Notification Channels

80.8% of respondents prefer receiving notifications via email, and 30.8% via WhatsApp or Telegram. This confirms the need to interface with both email servers and WhatsApp or Telegram in the communication module.

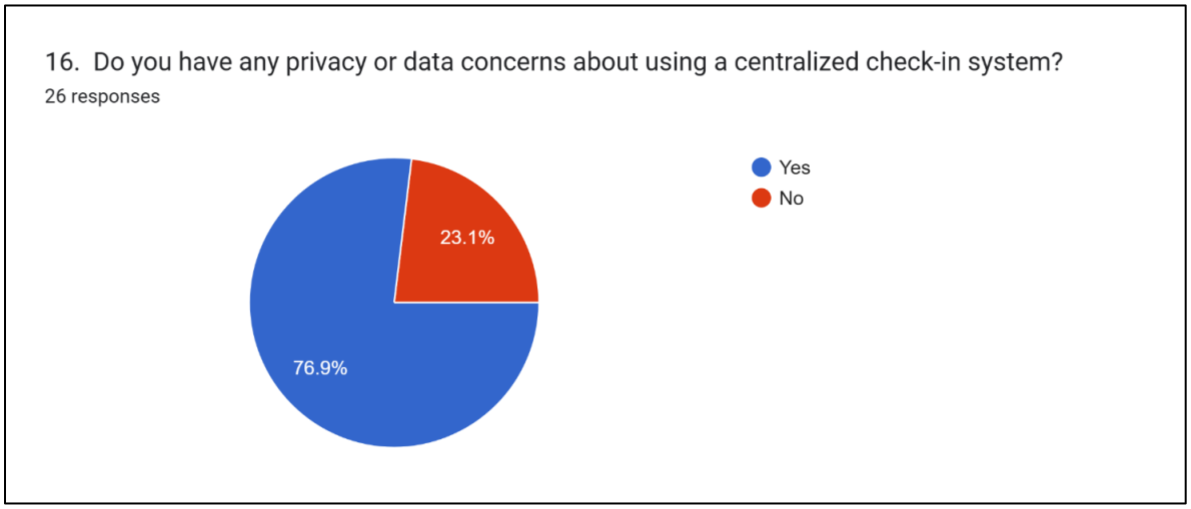


Figure 3.8.3.8: User Concerns about Privacy and Data Security

76.9% of users answered "Yes", citing concerns about data leaks and unauthorized access. This highlights the importance of implementing strict privacy controls, encryption, and PDPA compliance.

3.8.4 System Architecture Overview

The Campus Event Check-in System uses a modular and layered setup. It is built to grow easily, be simple to fix or update, and connect safely with other systems inside and outside the university. The system is meant to run in the cloud so that it can be used anytime and stay reliable. Main services use containers like Docker to help with running many small parts. Databases and file storage use cloud tools that can grow as needed.

Main Parts of the System:

1. **Client Layer**:
   This part has the web pages for students, event organizers, and admins. These pages work on phones and computers. They are web apps and are shared through a safe cloud connection.

2. **Application Layer**:

   This part runs the system's main features like event setup, attendance checks, sending messages, making tickets, and user login. Each part is a small service in its own container. This helps the system stay stable and easy to scale.

3. **Integration Layer**:

   This part talks to other systems like the student database and payment system. It uses APIs and token logins. It checks data, keeps things safe, and formats messages the right way.

4. **Data Layer**:

   This part uses cloud databases to store things like event info, user details, attendance, and payment records. The data is kept safe, backed up, and follows the rules of the university.

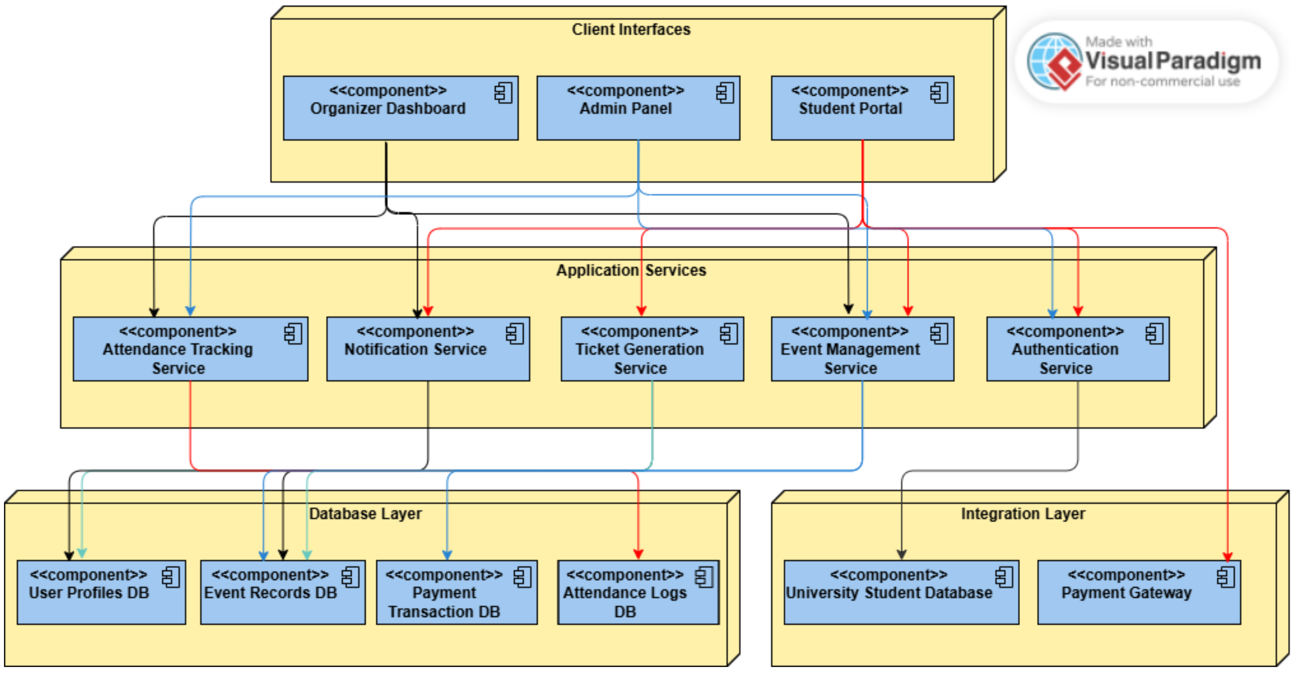

Client Interfaces
<<component>> Organizer Dashboard
<<component>> Admin Panel
<<component>> Student Portal
Made with Visual Paradigm For non-commercial use
Application Services
<<component>> Attendance Tracking Service
<<component>> Notification Service
<<component>> Ticket Generation Service
<<component>> Event Management Service
<<component>> Authentication Service
Database Layer
<<component>> User Profiles DB
<<component>> Event Records DB
<<component>> Payment Transaction DB
<<component>> Attendance Logs DB
Integration Layer
<<component>> University Student Database
<<component>> Payment Gateway

Figure 3.8.4.1 Deployment Diagram – System Architecture

3.8.5 Elicitation Evidence

Formal documentation and justification of the requirement elicitation activities conducted during the development of the Campus Event Check-In System are presented below. A combination of techniques was used to accurately capture stakeholder needs and expectations and translate them into clear system requirements. These techniques included the use of questionnaires, observation of existing systems, perspective-based analysis of scholarly articles, and internal brainstorming sessions.

Questionnaire

A structured questionnaire was developed using Google Forms to gather requirements from key target users of the system, namely students, event organizers, and university administrators. The questionnaire included a combination of multiple-choice and open-ended questions aimed at identifying user preferences, existing challenges in current systems, and desired system features. A total of 26 valid responses were collected.

Key insights gathered from the questionnaire include:

1. **Check-In Preference**: A majority of users (38.5%) prefer using both QR code and Student ID for check-in. This confirms the need to support multiple digital check-in methods for flexibility and reliability.
2. **Payment Preferences**: 73.1% of users favor e-wallets, while 61.5% also use Bank QR. The system must integrate with Malaysia’s most common digital payment options.
3. **Notification Preferences**: Most users (84.6%) find automated reminders very helpful, with 80.8% preferring email and 30.8% preferring WhatsApp or Telegram. Email and WhatsApp or Telegram support is essential for timely communication.

4. **Desired Features**: Top desired features include QR check-in (73.1%), event calendar (61.5%), and One-click registration and Payment Integration both are 42.3%. These should be prioritized as core or performance features.
5. **Privacy Concerns**: 76.9% of users are concerned about personal data privacy. Compliance with PDPA and implementation of encryption and secure login are critical.

Observation

Observational research was conducted on existing campus and commercial event systems to identify operational gaps and usability concerns.

Systems Observed:

- Manual Check-In (Google Forms) – Lacked automation and secure verification
- Eventbrite Platform – Effective but not linked to student databases
- RFID/Smart Card Systems – Fast and secure but required hardware infrastructure

Key Findings:

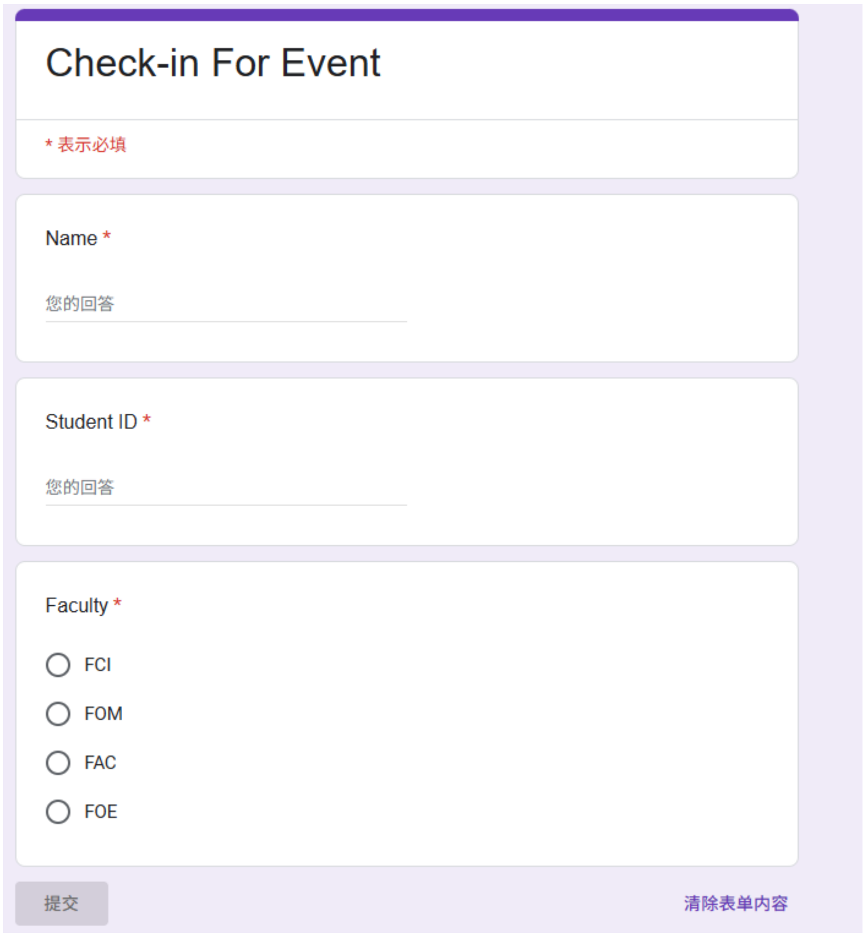
- Manual systems caused long queues and data inconsistencies
- QR-based systems significantly improved check-in speed
- RFID offered secure and efficient check-in but required backend integration

1. **Manual Google Form Check-In**

   This image captures a manual Google Form-based check-in system. The process, although simple, resulted in long queues and higher risk of data inconsistency, emphasizing the need for automation.

   Informed Requirements:

   - B-06: Duplicate check-in prevention
   - B-07: Manual check-in backup method


Check-in For Event

* 表示必填

Name *

您的回答

Student ID *

您的回答

Faculty *

- FCI
- FOM
- FAC
- FOE

Figure 3.8.5.2.1 Google Form for Check-in

2. Eventbrite

Eventbrite provides a digital check-in application that enables event organizers to scan QR codes from attendees' electronic tickets. The platform supports real-time attendee tracking and is integrated with event payment systems, enhancing both efficiency and accuracy. However, it lacks integration with university student databases, which limits its ability to perform secure identity verification within an academic context.

Informed Requirements:

- A-01: QR check-in
- P-01: Real-time attendance tracking
- P-05: Payment integration

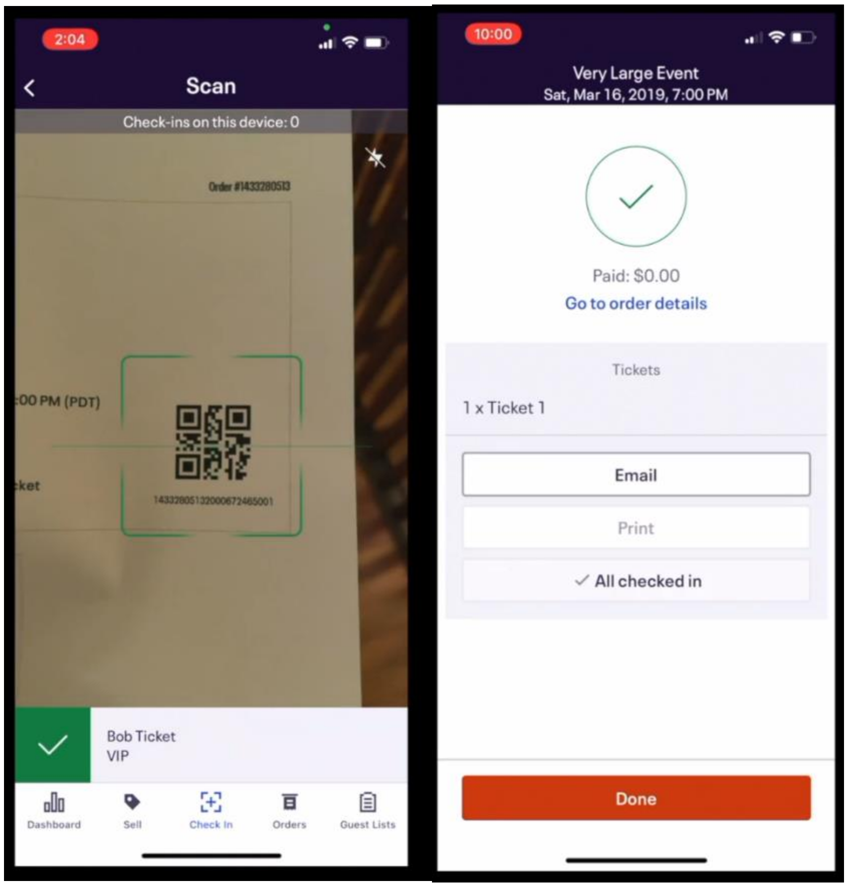


Figure 3.8.5.2.2: Eventbrite - Scan QR for Check-In and Event Payment System

3. **RFID Card Check-In**

   In this system, students check in to events by tapping their RFID-enabled student ID cards at designated entry points. The system is integrated with the university's backend infrastructure to authenticate student identities and update attendance records. While this method offers a fast, secure, and fully automated check-in process, it necessitates significant infrastructure investment and seamless integration with internal systems such as financial services and student information databases.

   Informed Requirements:

   - B-01: Identity verification using Student ID
   - P-03: High-speed check-in
   - A-03: Dual check-in modes (QR/ID)

Figure 3.8.5.2.3: RFID Check-In in University

Perspective-Based Reading

Perspective-Based Reading was used to analyze four academic articles published between 2020 and 2024. The documents were reviewed from three stakeholder viewpoints: Student, Event Organizer, and University Admin. This helped identify stakeholder-specific expectations, system constraints, and design patterns relevant to the project.

Articles Reviewed:

- Mohd Akin et al. (2024) – Mobile event attendance with QR code and OTP
- Chaturvedi et al. (2024) – CU-Events university event management system
- Yang & Wen (2020) – QR-based virtual payment system
- Wang & Xu (2021) – Campus card integration with backend systems

Key Insights:

- This excerpt from Chaturvedi et al. (2024) supports the inclusion of real-time dashboards for event organizers and QR-based registration mechanisms. It reflects features that align with both performance and basic expectations for an academic event system.
- “Students were dissatisfied with long queues and the lack of digital confirmation after check-in.” — This quote from Mohd Akin et al. (2024) justifies implementing automated email confirmations and mobile check-in processes.
- Yang & Wen (2020) emphasized seamless mobile transactions, validating the requirement for integrated e-wallet and QR-based payment options.
- Wang & Xu (2021) highlighted the importance of integration with student information systems and data protection, reinforcing the need for PDPA-compliant architecture.

Brainstorming

An internal brainstorming session was conducted among the development team via Microsoft Teams to generate creative ideas, especially delighter features under the Kano model. Prompts were used to guide idea generation.

Table 3.8.5.4.1 Brainstormed Features Summary

Feature Name	**Description**	**Stakeholder**	**Kano Category**
Loyalty Points System	Earn points for attending events, visible on profile	Student	Delighter
Achievement Badges	Award badges for milestone attendance (e.g., 5, 10, 20 events)	Student	Delighter
Attendance Leaderboard	Displays top attendees to encourage participation	Student	Delighter
Event Recommendation Engine	Recommends future events based on past attendance or categories	Student	Delighter
Calendar Integration	Sync event schedule with Google/Outlook calendar	Student	Delighter
Social Media Sharing	Share attendance, badges, or event on Instagram/Facebook	Student	Delighter
Feedback & Rating System	Allow post-event feedback and star rating	Student, Organizer	Satisfier
Exportable Reports	Download attendance data as PDF or Excel	Organizer, Admin	Satisfier

4. Verification

4.1 Functions Verification

4.1.1 Basic Functions Verification

B-01: Register Account

- **Test Method**: Unit and Integration Testing
- **Verification Criteria**: New user accounts created successfully with valid credentials, rejection of duplicate student IDs, required field validation, email confirmation process
- **Key Test Cases**: Valid registration, duplicate student ID attempt, missing required fields, invalid email format

B-02: Login

- **Test Method**: Security and Functional Testing
- **Verification Criteria**: Successful authentication with valid credentials, proper rejection of invalid credentials, session management and timeout, password encryption
- **Key Test Cases**: Valid/invalid login credentials, account lockout after failed attempts, session persistence

B-03: View Upcoming Events

- **Test Method**: Functional and Database Testing
- **Verification Criteria**: Events correctly retrieved from database, proper display of event details, filtering and sorting functionality
- **Key Test Cases**: Display events with various statuses, empty database handling, event sorting and filtering

B-04: Register Event

- **Test Method**: Integration and Business Logic Testing
- **Verification Criteria**: Students can register for available events, registration blocked when capacity reached, database updates correctly
- **Key Test Cases**: Registration for available/full events, duplicate registrations, past event registration attempts

B-05: Check In to Event

- **Test Method**: Integration and Identity Verification Testing
- **Verification Criteria**: Student ID validation, attendance status updates, duplicate check-in prevention, real-time status reflection
- **Key Test Cases**: Valid/invalid student ID check-in, duplicate attempts, unregistered student check-in

4.1.2 Performance Functions Verification

P-01: Make Payment

- **Test Method**: Payment Gateway Integration Testing
- **Verification Criteria**: Successful payment processing, proper handling of failures, transaction logging, security compliance
- **Key Test Cases**: Successful payment, payment failures, timeout handling, refund processing

P-02: Receive Virtual Ticket

- **Test Method**: System Integration Testing
- **Verification Criteria**: QR code generation after payment, ticket delivery via email and dashboard, unique ticket generation
- **Key Test Cases**: Ticket generation after payment, email delivery failures, QR code validation

P-03: Manage Event

- **Test Method**: CRUD Operations Testing
- **Verification Criteria**: Event creation/update/deletion functionality, input validation, authorization checks
- **Key Test Cases**: Create/update/delete events, unauthorized access attempts

P-04: Generate Attendance Report

- **Test Method**: Data Processing and Export Testing
- **Verification Criteria**: Accurate attendance data compilation, multiple export formats, real-time and historical reports
- **Key Test Cases**: Report generation for completed events, different export formats, large dataset performance

P-05: View Registered Events

- **Test Method**: Data Retrieval Testing
- **Verification Criteria**: Accurate display of user's registered events, proper event status indication, chronological sorting
- **Key Test Cases**: Display events for active user, different event statuses, pagination handling

P-06: Manage User Account

- **Test Method**: Administrative Function Testing
- **Verification Criteria**: User search and retrieval, account status modification, role assignment, audit trail
- **Key Test Cases**: User search by criteria, activate/deactivate accounts, modify roles and permissions

4.1.3 Attractive Functions Verification

A-01: Send Notification

- **Test Method**: Communication System Testing
- **Verification Criteria**: Multi-channel notification delivery, message queuing and retry mechanisms, notification preferences management
- **Key Test Cases**: Send notifications via different channels, handle delivery failures, test notification preferences

A-02: Submit/Cancel Event Request

- **Test Method**: Workflow Testing
- **Verification Criteria**: Event request submission process, request cancellation functionality, admin notification for pending requests
- **Key Test Cases**: Submit valid event request, cancel pending request, admin approval/rejection workflow

A-03: Resolve Payment Issue

- **Test Method**: Administrative Process Testing
- **Verification Criteria**: Payment dispute investigation tools, transaction status modification, issue resolution tracking
- **Key Test Cases**: Investigate payment discrepancies, process refunds, update transaction statuses

A-04: Send Announcement

- **Test Method**: Communication Feature Testing
- **Verification Criteria**: Targeted message broadcasting, message composition and formatting, delivery tracking
- **Key Test Cases**: Send announcements to event attendees, target specific user groups, handle delivery failures

4.2 Performance Requirement Verification

4.2.1 Response Time Testing

- **Test Method**: Load Testing using JMeter or LoadRunner
- **Verification Criteria**: Average response time < 2 seconds for all key functions, 95th percentile response time < 3 seconds
- **Test Scenarios**: Login process under normal load, event registration during peak hours, payment processing, check-in process during large events

4.2.2 Concurrent User Testing

- **Test Method**: Stress Testing and Load Distribution
- **Verification Criteria**: System supports 10,000 concurrent users, no performance degradation under maximum load
- **Test Scenarios**: Gradual user load increase from 1,000 to 10,000, simultaneous check-ins during major events, peak registration periods

4.2.3 Real-time Synchronization Testing

- **Test Method**: Integration and Data Consistency Testing
- **Verification Criteria**: Check-in status updates within 1 second, payment verification synchronization, event capacity updates in real-time
- **Test Scenarios**: Multiple simultaneous check-ins, payment processing during high traffic, event capacity near-full scenarios

4.2.4 QR Code and Ticket Validation Testing

- **Test Method**: Functional and Performance Testing
- **Verification Criteria**: QR code scanning response < 1 second, ticket validation accuracy 99.9%
- **Test Scenarios**: Rapid consecutive QR code scans, various QR code formats, damaged codes, network connectivity variations

4.3 Usability Requirement Verification

4.3.1 Mobile-First Responsive Design Testing

- **Test Method**: Cross-device and Browser Testing
- **Verification Criteria**: Responsive design across screen sizes (320px to 1920px), touch-friendly interface elements
- **Test Scenarios**: Testing on smartphones (iOS, Android), tablet compatibility, desktop browser compatibility

4.3.2 Accessibility Compliance Testing

- **Test Method**: WCAG 2.1 Compliance Testing
- **Verification Criteria**: Screen reader compatibility, keyboard navigation support, colour contrast ratio compliance
- **Test Tools**: WAVE Web Accessibility Evaluation Tool, axe accessibility checker, screen reader testing

4.3.3 User Experience Testing

- **Test Method**: Usability Testing with Target Users
- **Verification Criteria**: Task completion rate > 90%, user satisfaction score > 4/5
- **Test Scenarios**: New user registration process, event discovery and registration, check-in process simulation

4.3.4 Multi-language Support Testing

- **Test Method**: Localization Testing
- **Verification Criteria**: Language switching functionality, text translation accuracy, UI layout adaptation
- **Test Scenarios**: English and Malay language support, text expansion/contraction handling, date and number format localization

4.4 Interface Requirement Verification

4.4.1 User Interface Component Testing

- **Test Method**: Component and Integration Testing
- **Verification Criteria**: Consistent UI elements across user roles, dashboard functionality for each user type
- **Test Scenarios**: Student dashboard navigation, event organizer tools, admin panel functionality, cross-role permission verification

4.4.2 API Integration Testing

- **Test Method**: API Testing and System Integration
- **Verification Criteria**: Payment Gateway API integration, Student ID Authentication API connectivity
- **Test Scenarios**: Payment processing API calls, student verification API responses, error handling for API failures

4.4.3 System Interface Testing

- **Test Method**: End-to-end Integration Testing
- **Verification Criteria**: Seamless data flow between system components, proper error propagation and handling
- **Test Scenarios**: Complete user journey testing, cross-system data validation, failure recovery mechanisms

4.5 Logical Database Requirements Verification

4.5.1 Database Schema Validation

- **Test Method**: Database Design and Integrity Testing
- **Verification Criteria**: Proper table relationships and foreign key constraints, data normalization compliance, index optimization
- **Test Scenarios**: Referential integrity enforcement, cascade delete operations, query performance optimization

4.5.2 Data Flow Testing

- **Test Method**: Data Processing and Workflow Testing
- **Verification Criteria**: Complete user registration to check-in data flow, payment processing data integrity, attendance tracking accuracy
- **Test Scenarios**: User registration to event participation flow, payment processing to ticket generation, data backup and recovery procedures

4.5.3 Database Performance Testing

- **Test Method**: Database Load and Performance Testing
- **Verification Criteria**: Query response times under load, database scalability verification, concurrent access handling
- **Test Scenarios**: High-volume data insertion during peak registration, complex query performance with large datasets

4.6 Design Constraints Verification

4.6.1 University Branding Compliance

- **Test Method**: Visual Design Review and Testing
- **Verification Criteria**: Adherence to university visual identity guidelines, consistent colour scheme and typography
- **Verification Process**: Design review by university brand committee, cross-page consistency checking

4.6.2 Mobile-First Design Verification

- **Test Method**: Responsive Design Testing
- **Verification Criteria**: Mobile-optimized user interfaces, touch-friendly interaction elements, performance on mobile devices
- **Test Scenarios**: Small screen layout verification, touch gesture functionality, mobile network performance

4.6.3 Browser Compatibility Testing

- **Test Method**: Cross-browser Testing
- **Verification Criteria**: Functionality across major browsers, no external app installation required
- **Test Browsers**: Chrome, Firefox, Safari, Edge, mobile browsers

4.6.4 Technology Stack Verification

- **Test Method**: Architecture and Technology Review
- **Verification Criteria**: Open-source technology preference, scalable backend architecture, cost-effectiveness analysis
- **Verification Areas**: Technology licensing compliance, scalability architecture validation, performance benchmarking

4.7 System Attribute Verification

4.7.1 Security Testing

- **Test Method**: Security Penetration Testing and Code Review
- **Verification Criteria**: Data encryption (TLS/HTTPS) implementation, authentication and authorization mechanisms, input validation
- **Test Scenarios**: Penetration testing for common vulnerabilities, authentication bypass attempts, session management security

4.7.2 Privacy and PDPA Compliance Testing

- **Test Method**: Privacy Audit and Compliance Review
- **Verification Criteria**: Explicit consent mechanisms, data minimization practices, role-based access controls
- **Verification Areas**: Data collection consent processes, user data access rights, data retention policies

4.7.3 Data Protection and Backup Testing

- **Test Method**: Data Recovery and Backup Testing
- **Verification Criteria**: Automated daily backup functionality, data redundancy across multiple locations
- **Test Scenarios**: Backup creation and verification, data recovery simulation, disaster recovery procedures

4.7.4 High Availability Testing

- **Test Method**: System Reliability and Uptime Testing
- **Verification Criteria**: 99.9% uptime achievement, failover mechanisms functionality, load balancing effectiveness
- **Test Scenarios**: System component failure simulation, network interruption recovery, database failover testing

4.7.5 Scalability Testing

- **Test Method**: Performance and Load Scalability Testing
- **Verification Criteria**: Traffic spike handling capability, auto-scaling mechanisms, resource utilization optimization
- **Test Scenarios**: Gradual load increase testing, peak event traffic simulation, resource scaling verification

4.7.6 Maintainability Testing

- **Test Method**: Code Quality and Documentation Review
- **Verification Criteria**: Code documentation standards, modular architecture implementation, CI/CD pipeline functionality
- **Verification Areas**: Code review and quality metrics, documentation completeness, deployment automation testing

4.7.7 Auditability Testing

- **Test Method**: Audit Trail and Logging Verification
- **Verification Criteria**: Administrative action logging, security event monitoring, compliance reporting capabilities
- **Test Scenarios**: Administrative action tracking, security incident logging, audit report generation

4.8 Supporting Information Verification

4.8.1 Traceability Matrix Verification

- **Verification Method**: Requirements Traceability Analysis
- **Verification Criteria**: All requirements trace back to elicitation evidence, use cases align with functional requirements
- **Traceability Areas**: Questionnaire responses to system features, observation findings to usability requirements

4.8.2 Use Case Diagram Validation

- **Verification Method**: Stakeholder Review and Validation
- **Verification Criteria**: All user roles and interactions represented, use case completeness and accuracy
- **Validation Process**: Stakeholder review sessions, use case walkthrough testing, scenario-based validation

4.8.3 Sequence Diagram Verification

- **Verification Method**: System Flow and Logic Validation
- **Verification Criteria**: Accurate system interaction representation, complete message flow documentation
- **Verification Areas**: System component interaction accuracy, message sequence correctness, exception handling completeness

4.8.4 Questionnaire Results Validation

- **Verification Method**: Statistical Analysis and Interpretation Review
- **Verification Criteria**: Statistical significance of responses, proper interpretation of survey data
- **Validation Process**: Response data accuracy verification, statistical analysis review, conclusion validity assessment

5. Appendices

5.1 Assumptions and Dependencies

5.1.1 System Assumptions

Technical Assumptions:

- University maintains current student database with enrolment information
- Reliable internet connectivity available at event venues
- Users have access to smartphones/computers with modern browsers
- Payment gateway services maintain 99.5% uptime
- University IT infrastructure supports system integration

User Assumptions:

- Students possess basic digital literacy for system usage
- Event organizers have fundamental event planning knowledge
- University administrators understand system management
- Users willing to provide necessary registration information
- Students carry university ID cards during events

Business Assumptions:

- University policy supports digital transformation initiatives
- Sufficient budget allocation for development and maintenance
- Stakeholder commitment to system adoption
- Legal compliance framework remains stable
- University events continue requiring attendance management

5.1.2 External Dependencies

University Systems:

- Student Information System (SIS) for identity verification
- University Payment Gateway for transaction processing
- Email System for notification delivery
- Network Infrastructure for system access
- Access Control Systems for venue integration

Third-Party Services:

- Payment Processing Services (FPX, e-wallet providers)
- SMS/WhatsApp Services for notifications
- Cloud Infrastructure (AWS, Azure, Google Cloud)
- SSL Certificate Providers for secure communications
- Backup and Recovery Services

Technology Dependencies:

- Web Browser Compatibility with modern standards
- Mobile Operating Systems (iOS/Android) updates
- Database Management Systems support
- Development Framework maintenance
- Security Libraries updates

5.1.3 Regulatory and Compliance Dependencies

- Personal Data Protection Act (PDPA) 2010 compliance
- University policies for student data and event management
- Financial transaction regulations compliance
- Web accessibility guidelines (WCAG 2.1)
- Cybersecurity frameworks and best practices

5.1.4 Operational Dependencies

Human Resources:

- Technical support staff for system maintenance
- Training personnel for user onboarding
- System administrators for monitoring
- Help desk support for troubleshooting

Organizational:

- Change management for system adoption
- Communication channels for announcements
- Event management processes
- Budget approvals for operation

5.1.5 Risk Mitigation Strategies

Technical Risk Mitigation:

- Backup systems and failover mechanisms
- Regular data recovery procedures
- Continuous security monitoring
- Real-time performance monitoring

Operational Risk Mitigation:

- Comprehensive user training programs
- Detailed documentation and manuals
- Established support processes
- Structured change management approach

5.2 Acronyms and Abbreviations

5.2.1 Technical Acronyms

API - Application Programming Interface
HTTPS - Hypertext Transfer Protocol Secure
JSON - JavaScript Object Notation
PDF - Portable Document Format
QR - Quick Response (Code)
REST - Representational State Transfer
SMS - Short Message Service
SQL - Structured Query Language
TLS - Transport Layer Security
UI - User Interface
URL - Uniform Resource Locator
UUID - Universally Unique Identifier
XML - Extensible Markup Language

5.2.2 Business and Organizational Acronyms

PDPA - Personal Data Protection Act
SIS - Student Information System
SRS - Software Requirements Specification
FAQ - Frequently Asked Questions
IT - Information Technology
HR - Human Resources
QA - Quality Assurance
UAT - User Acceptance Testing
ROI - Return on Investment
SLA - Service Level Agreement

5.2.3 System-Specific Acronyms

CECS - Campus Event Check-in System
EO - Event Organizer
UA - University Admin
PG - Payment Gateway
RTO - Recovery Time Objective
RPO - Recovery Point Objective

5.2.4 Testing and Quality Acronyms

WCAG - Web Content Accessibility Guidelines
PCI DSS - Payment Card Industry Data Security Standard
ISO - International Organization for Standardization
IEEE - Institute of Electrical and Electronics Engineers
OWASP - Open Web Application Security Project
GDPR - General Data Protection Regulation (for reference)

5.2.5 Technology Platform Acronyms

AWS - Amazon Web Services
CSS - Cascading Style Sheets
HTML - HyperText Markup Language
HTTP - HyperText Transfer Protocol
JS - JavaScript
MVC - Model-View-Controller
ORM - Object-Relational Mapping
SDK - Software Development Kit
CDN - Content Delivery Network
DNS - Domain Name System

5.2.6 Communication and Integration Acronyms

SMTP - Simple Mail Transfer Protocol
FTP - File Transfer Protocol
SSH - Secure Shell
VPN - Virtual Private Network
LDAP - Lightweight Directory Access Protocol
SAML - Security Assertion Markup Language
OAuth - Open Authorization
JWT - JSON Web Token

5.2.7 Database and Storage Acronyms

RDBMS - Relational Database Management System
NoSQL - Not Only SQL
CRUD - Create, Read, Update, Delete
ACID - Atomicity, Consistency, Isolation, Durability
ETL - Extract, Transform, Load
DML - Data Manipulation Language
DDL - Data Definition Language

5.2.8 Performance and Monitoring Acronyms

KPI - Key Performance Indicator
SLI - Service Level Indicator
SLO - Service Level Objective
APM - Application Performance Monitoring
CPU - Central Processing Unit
RAM – Random Access Memory
I/O - Input/Output
QPS - Queries Per Second