**Software Requirements Specifications (SRS)**

**for**

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

**Part 1**

**Tutorial Section: TT1L**

**Group No.: Group 6**

| **HAZIQ IZZUDDIN BIN AHMAD TARMIDZI** | **1211112293** |  |
| --- | --- | --- |
| **WAN AMIRUL AMIR BIN WAN ROMZI** | **1211112289** |  |
| **MUHAMMAD ARIF BIN ABDUL JALEEL** | **1231303334** |  |
| **LIM XIAO QI** | **1211108212** |  |

**Date:** 23 May 2025

# **Contents**

[**Contents 2**](#_ndz2yj3a7ni)

[**Revisions 4**](#_vm5rmn44n0wd)

[**1 Introduction 5**](#_v6pbpnpw4s5x)

[1.1 Purpose 5](#_omvjdfp6pbx2)

[1.2 Scope 6](#_kwx60bq2u6v3)

[1.3 Product overview 7](#_f1u7bwfxl70d)

[1.3.1 Product perspective 7](#_67uvxemer39m)

[1.3.2 User characteristics 8](#_d3j127q8wfs)

[1.3.3 Limitations 9](#_dpmk951dboeq)

[1.4 Definitions 10](#_4itz51g07d8h)

[**2 References 11**](#_xzxupwgz6cnx)

[**3 Requirements 12**](#_db2ejflxld3r)

[3.1 Functions 12](#_26ji3gddj1sd)

[3.1.1 Product functions 12](#_p46ucnfzq7jz)

[3.1.2 Student 15](#_xv305t5evmz1)

[3.1.3 Event Organizer 17](#_2j4arh9bhbmf)

[3.1.4 Administrator 21](#_l479f75txhwh)

[3.1.5 Summary Table 22](#_s24dgekg3jbb)

[3.2 Performance requirements 23](#_s3dahksvhzfg)

[3.2.1 Response Time Requirements 23](#_tqwzf2wz9uki)

[3.2.2 Throughput Requirements 24](#_1ek9u35q18xc)

[3.2.3 Concurrency Requirements 24](#_8w23io2cadjo)

[3.2.4 Scalability Requirements 25](#_2cusfath023e)

[3.2.5 Availability Requirements 25](#_8sl4aolikxo5)

[3.3 Usability requirements 26](#_nkvrdfp25kmf)

[3.3.1 Ease of Use 26](#_g9hktea769a3)

[3.3.2 Learnability 27](#_o93ge1e0b5he)

[3.3.3 Efficiency 27](#_x0suji2vw1hw)

[3.3.4 User Satisfaction 28](#_47l85xklfe83)

[3.3.5 Accessibility 28](#_o6s35hyr6h6)

[3.4 Interface requirements 29](#_b0f3lgsw5frq)

[3.4.1 System Interfaces 29](#_jlmxudt7sik7)

[3.4.2 User Interfaces 29](#_2exckr895chn)

[3.4.3 Hardware Interfaces 30](#_wugmjrw9jqga)

[3.4.4 Software Interfaces 31](#_ulpat1361gq2)

[3.4.5 Communications Interfaces 32](#_xvyci4t42j3h)

[3.5 Logical database requirements 33](#_ab1rb9jbvlvj)

[3.5.1 Key Data Entities and Attributes 33](#_ax01oyhzn471)

[3.5.2 Relationships 34](#_xbw81othoayb)

[3.5.3 Constraints 35](#_28o3i3vl5mg4)

[3.5.4 Entity Relationship Diagram (ERD) 36](#_dcz5vahlk1at)

[3.6 Design constraints 37](#_bt3hw17j7v1f)

[3.6.1 Compliance with University Policies 37](#_hmpv5x3rvl18)

[3.6.2 Integration with Existing Infrastructure 37](#_dc6sizkkiy3q)

[3.6.3 Regulatory and Privacy Constraints 37](#_lo1va7i5mhgj)

[3.6.4 Technical Limitations 38](#_gnnirndbe3yu)

[3.6.5 Operational Constraints 38](#_pc4nyxfbx1hk)

[3.7 Software system attributes 39](#_jhaz2og8kiv1)

[3.7.1 Reliability 39](#_4jnkf9iiueqa)

[3.7.2 Availability 39](#_or9w2912zwn2)

[3.7.3 Security 40](#_9trxl0fpbyy8)

[3.7.4 Maintainability 40](#_v84cxfa4d30w)

[3.7.5 Portability 40](#_oy4j3jo6k7tt)

[3.8 Supporting information 40](#_x2ujq4ryyc58)

[3.8.1 Sample Input/Output Formats 41](#_k1fvg32k3h4u)

[3.8.2 Elicitation and Research Sources 42](#_p9pj3055mkeb)

[3.8.3 Problem Description 42](#_qo4ptnkzy8y1)

[3.8.4 Packaging and Deployment Instructions 43](#_h3nftlxzv18r)

[3.8.5 Validation Session 43](#_vxpmr98wqboh)

[3.8.6 Defect Summary 44](#_ifptn68hnvtf)

[3.8.7 Conflict Analysis 46](#_rfe206djtfrs)

[3.8.8 Conflict Analysis and Resolution 46](#_mlagmroatgzl)

[3.8.9 Change Log 47](#_epsd8pl98c2x)

[3.8.10 Requirements Traceability Matrix 47](#_k7ph86yvs4gi)

[3.8.11 Role in Requirements Validation, Negotiation & Management 48](#_am77l26d9wdm)

[3.8.12 Version Control & Configuration Summary 49](#_dlc8d1nt6xyz)

[4 Verification 50](#_fpb38h4br2xn)

[4.1 Verification Approach 50](#_1vrkshtmmu2z)

[4.2 Verification Criteria 50](#_erdabshbplhs)

[4.3 Verification of Elicitation Methods 51](#_y31zoeu099v8)

[4.4 Proof of Execution 53](#_n47jfsvn74bl)

[4.4.1 Techniques 1: Interview 53](#_xbpr2yi3adoj)

[4.4.2 Techniques 2: Observation 56](#_su0ze9qf7br9)

[4.4.3 Techniques 3: Brainstorming 64](#_ainkaea4p5un)

[4.5 Overall Elicitation 67](#_4ivhjj3lemr)

[5 Appendices 68](#_bi420xde747r)

[5.1 Assumptions and dependencies 68](#_s75duijofd66)

[5.2 Acronyms and abbreviations 69](#_uti2azkwlvzz)

[5.3 Glossary **70**](#_jhrp27v4570n)

# **Revisions**

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| SRS Part 1 | Haziq Izzuddin,  Wan Amirul Amir,  Muhaammad Arif,  Lim Xiao Qi | Part 1 | 25/05/2025 |
| SRS Part 2 | Tang Wei Xiong, Liew Wei Hong, Ng Kean Ping, Chan Mei Yi | Part 2 | 23/06/2025 |

# **Introduction**

## Purpose

The purpose of the Campus Event Check-in System is to streamline the process of student event registration and attendance by integrating with the university’s Student Information Database and Payment Gateway. The system aims to automate check-ins, support secure payments, and provide real-time attendance tracking, improving convenience for students and efficiency for event organizers.

## Scope

The Campus Event Check-in System will be used by students and event organizers within the university to manage event registration, attendance, and payments in a streamlined and integrated manner.

The system provides core functionalities such as student check-in using Student ID, event creation, payment processing, and real-time attendance tracking. It will integrate with the university’s Student Information Database for authentication and the university's Payment Gateway for processing transactions.

**The benefits of using this system include:**

* Simplified event check-in process for students
* Accurate and real-time attendance tracking
* Seamless payment experience for paid events
* Improved management tools for organizers
* Enhanced data reporting and event monitoring

**The objectives of the system are to:**

* Enable students to check in to events using their Student ID
* Facilitate secure, cashless payments for event participation
* Allow event organizers to create and manage events
* Track and report attendance data automatically
* Provide admin tools for user access and system monitoring

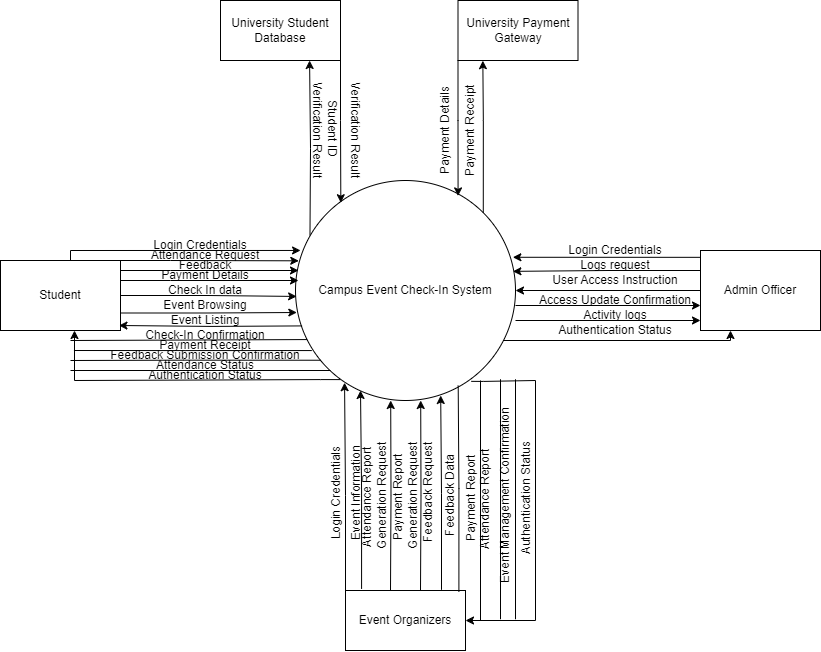
**The goals of the system are to:**

* Improve the efficiency and accuracy of event attendance tracking
* Minimize administrative overhead for event management
* Enhance the user experience for both students and organizers

## Product overview

### Product perspective

The Campus Event Check-in System is a web-based application that operates through an internet connection. When users access the system, the server authenticates their identity using the University Student Database. Based on their role as a student, event organizer, or admin officer, users are given access to specific features. The system also connects to the University Payment Gateway to process event-related transactions. All data is fetched and managed through the web server and presented to the user in real time.



**1.3.1.1 Context Diagram**

### User characteristics

The Campus Event Check-in System is designed to serve several distinct user groups, each with specific characteristics and needs:

* **Students:**  
  The primary users of the system, students vary widely in technical expertise but generally possess basic digital literacy. They require a straightforward, intuitive interface to quickly check in to events and make payments using their Student ID. The system must be easy to use on multiple devices, including smartphones and campus kiosks, to accommodate diverse user preferences.
* **Event Organizers:**  
  These users are responsible for creating and managing events, monitoring attendance, and handling ticketing. They typically have moderate technical skills and need access to real-time data and reporting features. Their tasks involve scheduling events, verifying attendance, and overseeing payment statuses, so the system should provide efficient management tools with clear dashboards.
* **Administrative Staff:**  
  Administrative users oversee the overall operation and maintenance of the system. They manage user roles, system configurations, and ensure data integrity. These users are expected to have a higher level of technical proficiency and require access to advanced administrative functions and security controls.

By understanding these user groups and their characteristics, the system can be designed to meet their specific usability needs, ensuring a positive user experience for all stakeholders.

### Limitations

The Campus Event Check-in System has several limitations that define its operational boundaries:

* The system is designed exclusively for campus events that are officially approved and managed by the university. It does not support external or off-campus events.
* It depends on the continuous availability and proper functioning of the university’s student information system for authenticating Student IDs. Any downtime or issues with this external system will impact the check-in process.
* Payment processing relies entirely on the university’s integrated payment gateway. The system does not support third-party or external payment methods outside this gateway.
* The system is limited to supporting users who have basic to moderate technical skills. It assumes students and staff will use standard web-enabled devices for access; specialized hardware or offline functionality is not provided.
* The platform must comply with university policies regarding data privacy and security, which may impose restrictions on data sharing and retention.

These limitations should be considered when deploying and maintaining the system to ensure realistic expectations and proper resource allocation.

## Definitions

* **Student ID:** A unique identifier assigned to each student by the university for authentication and identification purposes.
* **Check-in:** The process by which a student verifies their attendance at a campus event through the system.
* **Payment Gateway:** A secure external service used to process financial transactions related to event fees.
* **Event Organizer:** A university staff member or authorized student responsible for creating, managing, and overseeing campus events.
* **Administrative Staff:** Personnel responsible for maintaining the system, managing user roles, and ensuring smooth operation.
* **Attendance Report:** A document or data set generated by the system that details the number of attendees and their check-in status for an event.
* **Real-time:** Data or actions that are processed immediately as they occur, with minimal delay.
* **Middleware:** Software that connects different systems, enabling them to communicate and exchange data seamlessly.

# **References**

* IEEE. (2018). *ISO/IEC/IEEE 29148:2018 Systems and software engineering—Life cycle processes—Requirements engineering*. <https://www.iso.org/standard/72089.html>
* Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer.

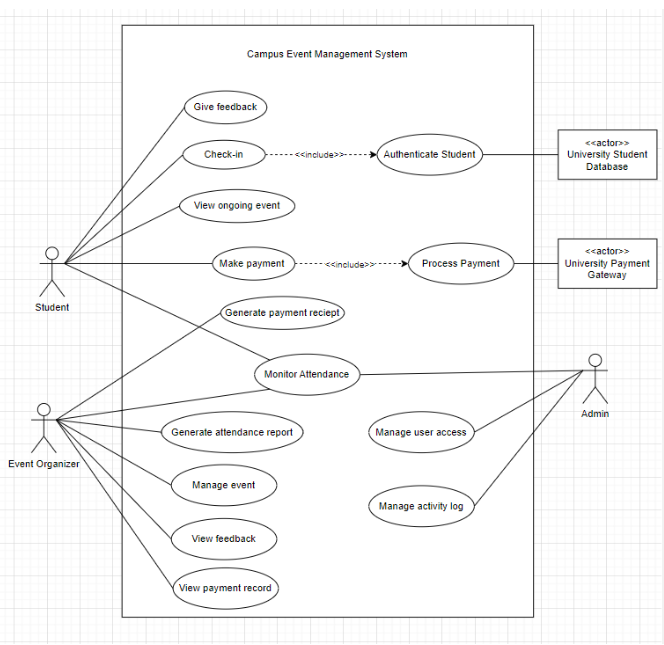
# **Requirements**

## Functions

### 3.1.1 Product functions

The Campus Event Check-in System provides the following primary functions:

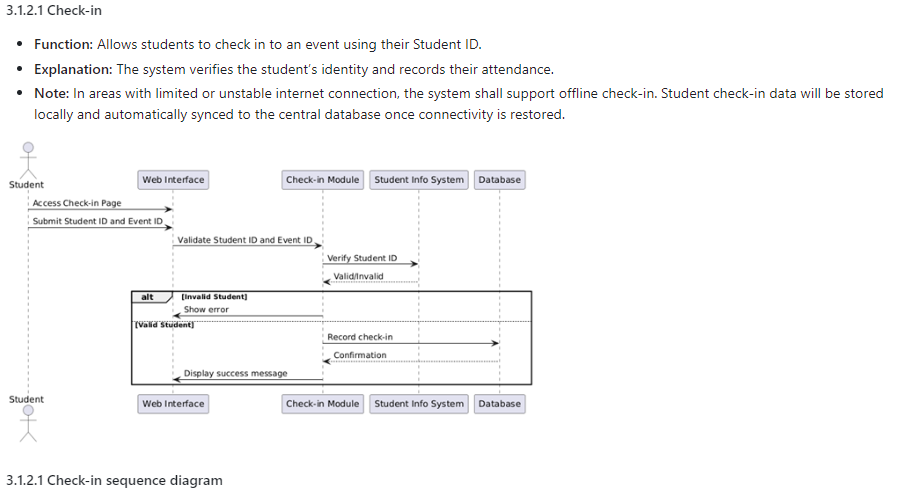
1. **Give feedback**: Students can provide feedback after events, helping improve future events.
2. **Check-in**: Students can check in to campus events by scanning or entering their Student ID. The system authenticates the student using the University Student Database before recording attendance in real time.
3. **View ongoing events**: Students can view details of upcoming or ongoing events to stay informed.
4. **Make payment**: Students pay for event fees through the system. This use case includes Process Payment, which is handled by the external University Payment Gateway.
5. **Monitor attendance**: Event Organizers and Admins can monitor real-time attendance data during events. This includes live updates on all participants who have checked in. Students can also use this feature to view their own attendance status, such as confirming whether they’ve successfully checked in or reviewing their attendance history for past events.
6. **Generate payment receipt**: The system generates a digital receipt after each successful payment. While this process is automatic, Event Organizers can access receipts for verification or financial reporting purposes.
7. **Generate attendance report**: Event Organizers can generate reports that summarize event attendance records. These reports help identify participation trends and overall event turnout.
8. **Manage events**: Organizers create, edit, and delete events. This includes managing event details, tickets, schedules, and attendance rules.
9. **View feedback**: Organizers can access feedback submitted by students to evaluate event success and identify areas for improvement.
10. **View payment record**: Organizers can view payment records for their events. This includes transaction history, payment status, and summary per participant.
11. **Authenticate Student**: Verifies the identity of students using the University Student Database before allowing them to check in to events.
12. **Process Payment**: Processes payment transactions through the University Payment Gateway and returns confirmation to the system.
13. **Manage user access**: Admins manage all system users by adding, editing, or removing user accounts and assigning appropriate roles (e.g., Student, Event Organizer).
14. **Manage activity logs**: Admins can track user actions within the system for security and audit purposes.

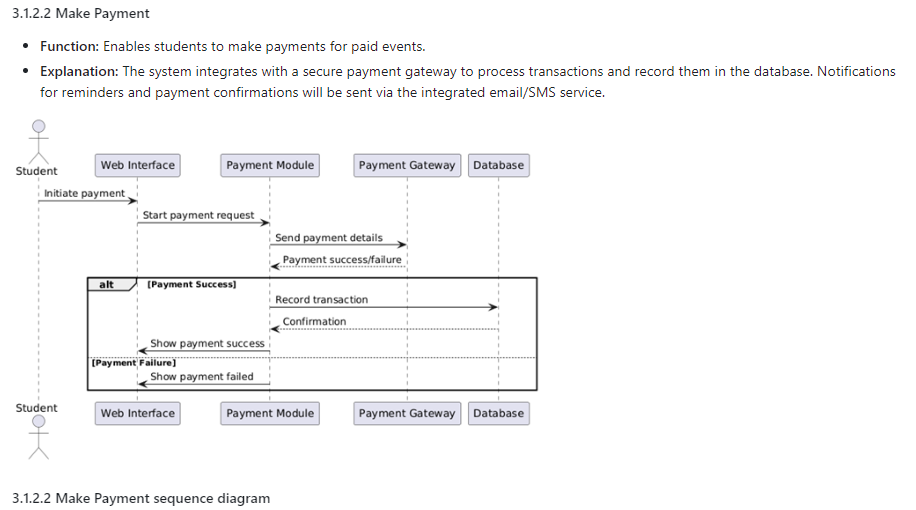


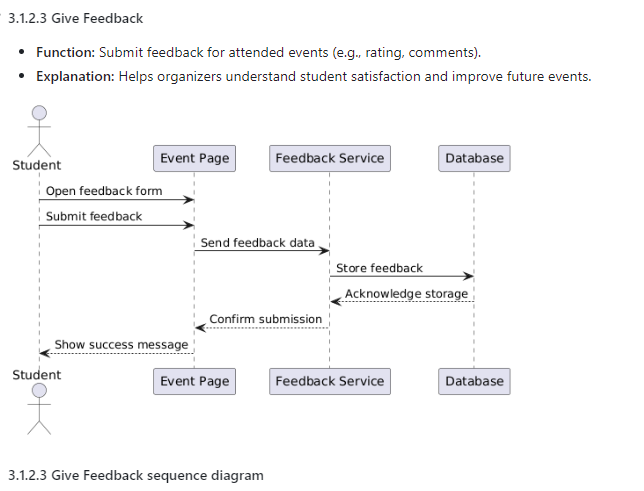
3.1.1.1 Use Case Diagram

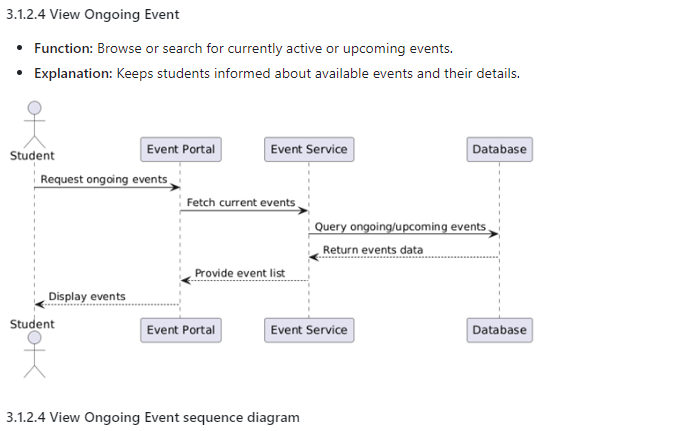
### 3.1.2 Student

Students are the primary users who attend events, check in, and make payments when necessary. Students may also give feed and view ongoing events.

****

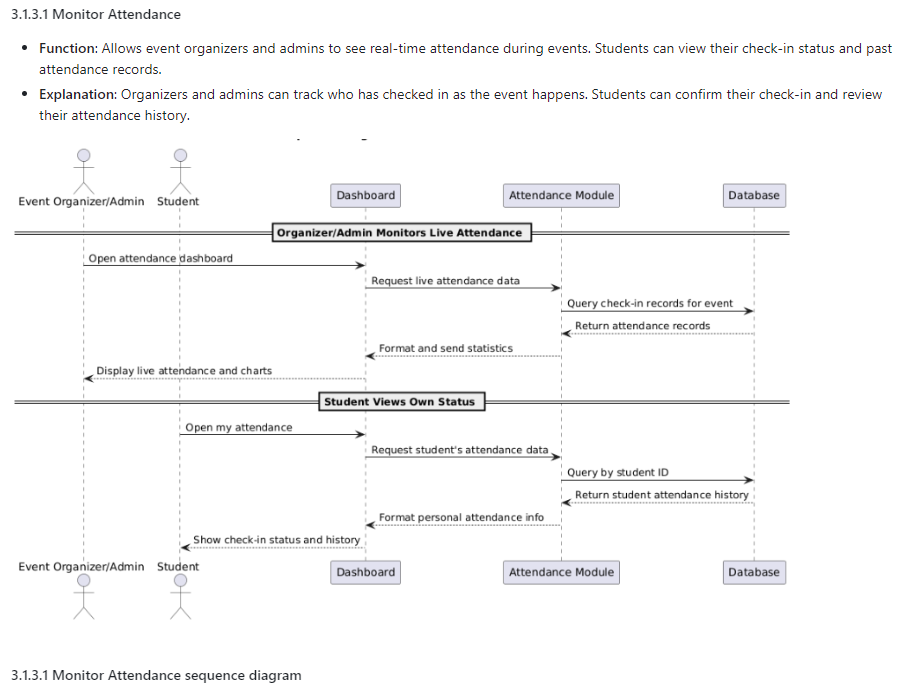


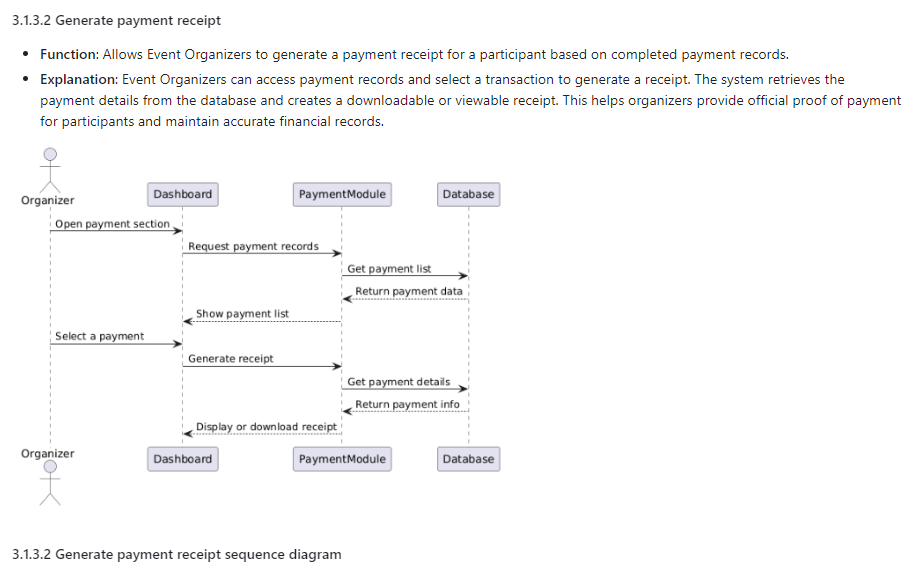


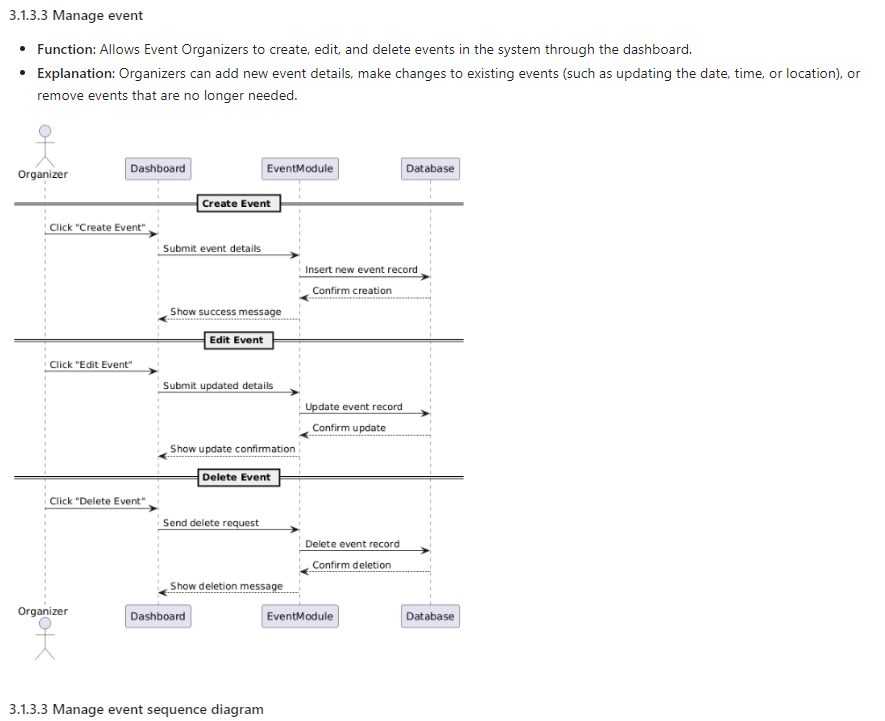


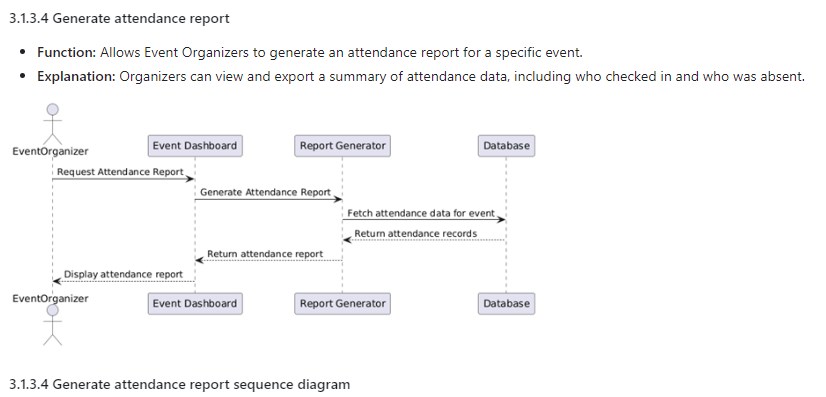
### 3.1.3 Event Organizer

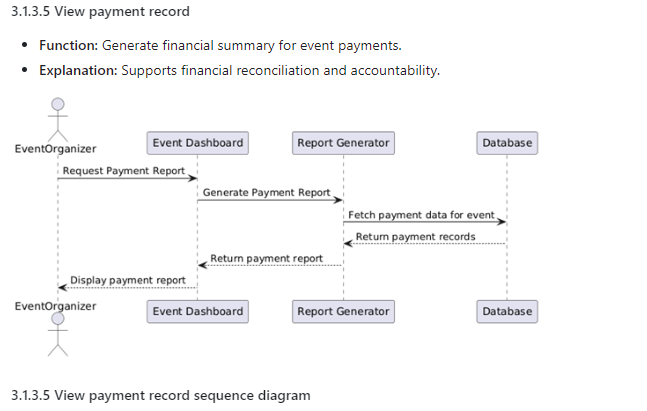
Event Organizers are responsible for creating events, monitoring attendance, and managing event logistics. Event Organizers are responsible for making attendance reports and payment reports when necessary.

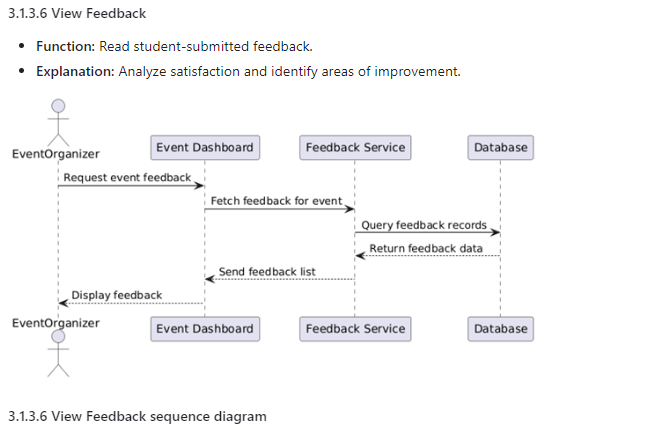








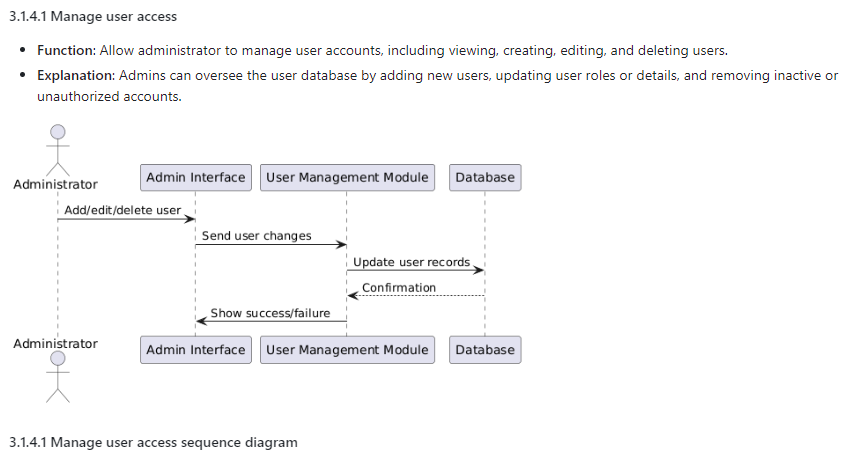


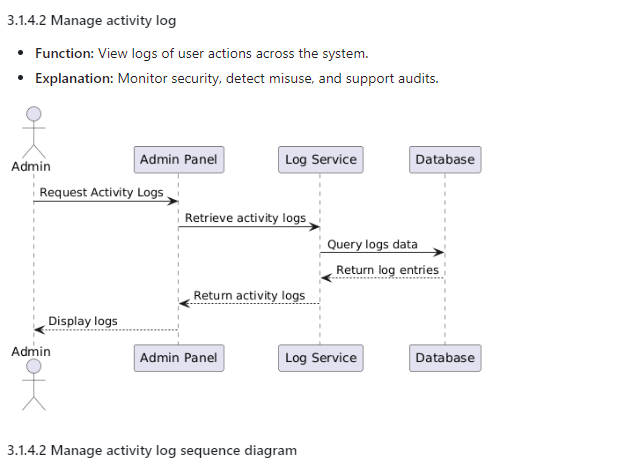


### 

### 3.1.4 Administrator

Administrators manage the system backend, reporting, and user roles. Administrators also manage the activity logs for user actions.





### 3.1.5 Summary Table

| Actor | Function | Description | Diagram |
| --- | --- | --- | --- |
| Student | Check-in | Check into events with Student ID | Check-in |
| Make Payment | Make payments for paid events | Make Payment |
| Give Feedback | Submit feedback for attended events | Give Feedback |
| View Ongoing Events | Browse or search for current/upcoming events. | View Ongoing Events |
| Event Organizer | Monitor Attendance | View real-time check-ins | Monitor Attendance |
| Generate Payment Receipt | Create payment receipts based on completed transactions. | Generate Payment Receipt |
| Manage events | Create and manage event details | Manage events |
| Generate Attendance Report | Export attendance summaries for events. | Generate attendance report |
| View Payment Record | Generate financial summary for event payments | View Payment Record |
| View Feedback | Read feedback submitted by students. | View Feedback |
| Administrator | Manage User Access | View, add, edit, or remove user accounts and assign roles. | Manage User Access |
| Manage activity logs | View logs of user actions for security and audit. | Manage activity logs |

## Performance requirements

The Campus Event Check-in System shall meet the following performance requirements to ensure efficient operation under both normal and peak conditions. These requirements are critical to maintaining usability, responsiveness, and reliability for all user groups, including students, event organizers, and administrative staff.

### Response Time Requirements

Response time refers to how quickly the system reacts to user input or actions. Fast response times are essential for user satisfaction, especially during check-in and payment operations. Delays in response could lead to bottlenecks during event entry or payment confirmation, negatively affecting user experience.

| Requirement ID | Description |
| --- | --- |
| PR-01 | The system shall respond to student check-in requests within **2 seconds** under normal load (≤100 concurrent users). |
| PR-02 | The system shall respond to payment confirmation operations within **3 seconds**, including interactions with the university's payment gateway. |
| PR-03 | The dashboard used by event organizers shall load attendance statistics within **5 seconds** under normal network conditions. |
| PR-04 | Administrative reporting queries shall generate summary reports within **10 seconds** for events with up to 1,000 attendees. |

### Throughput Requirements

Throughput measures the number of operations the system can handle per unit of time. These requirements ensure the system can accommodate high volumes of check-ins and payment transactions without degradation in performance, particularly during large events.

| Requirement ID | Description |
| --- | --- |
| PR-05 | The system shall support a minimum of **10 check-ins per second** during peak periods (e.g., large-scale events). |
| PR-06 | The system shall process at least **5 payment transactions per second** concurrently without error under peak load. |

### Concurrency Requirements

Concurrency requirements define how many users can use the system simultaneously without performance degradation. These ensure that students, organizers, and administrators can operate concurrently, especially during overlapping events or system-wide reporting tasks.

| Requirement ID | Description |
| --- | --- |
| PR-07 | The system shall support at least **500 simultaneous users** (students and staff) performing various operations, including check-in, payment, and reporting. |
| PR-08 | The event dashboard shall update in real-time for up to **10 concurrent organizers** monitoring the same event. |

### Scalability Requirements

Scalability defines how well the system can grow to handle increasing loads over time. These requirements ensure that the system remains functional and performant as user numbers and event complexity increase, especially across academic years.

| Requirement ID | Description |
| --- | --- |
| PR-09 | The system architecture shall support horizontal scaling to handle increased load, enabling support for **up to 5,000 concurrent users** during university-wide events. |
| PR-10 | The system shall maintain performance levels (as specified in PR-01 to PR-06) with a **10% increase in user traffic per semester**, up to a maximum of 10,000 registered users. |

### Availability Requirements

Availability requirements ensure that the system is reliably accessible and operational when needed. High availability is crucial, particularly for time-sensitive event check-ins and real-time reporting.

| Requirement ID | Description |
| --- | --- |
| PR-11 | The system shall have a minimum uptime of **99.5% per semester**, excluding scheduled maintenance windows. |

## Usability requirements

The Campus Event Check-in System is designed to support a wide range of users, including students with basic digital literacy, and staff with moderate to advanced technical skills. The system shall prioritize ease of use, learnability, and user satisfaction across all roles. Usability objectives are defined below and are intended to ensure that users can interact with the system efficiently and intuitively.

### Ease of Use

This subsection defines how intuitively users can perform their tasks. The goal is to minimize the number of steps or clicks required to complete frequent operations such as checking in, making a payment, or accessing dashboards.

| Requirement ID | Description |
| --- | --- |
| UR-01 | The interface shall allow students to complete a check-in process in **no more than 3 steps or clicks**. |
| UR-02 | Payment for an event shall be completed in **under 5 steps**, including confirmation through the payment gateway. |
| UR-03 | All user interfaces shall provide tooltips and contextual help for key actions to minimize user confusion. |

### Learnability

Learnability reflects how quickly a new user can begin using the system effectively. This is crucial for students and event organizers who may not receive formal training.

| Requirement ID | Description |
| --- | --- |
| UR-04 | New users (students and staff) shall be able to successfully complete core tasks (check-in, event creation, report generation) with **no prior training**, using only on-screen guidance. |
| UR-05 | The system shall include a **user onboarding guide or tutorial** accessible via the help section, with step-by-step walkthroughs for each user role. |

### Efficiency

Efficiency refers to how quickly frequent users can perform tasks once they are familiar with the system. This supports both student convenience and staff productivity.

| Requirement ID | Description |
| --- | --- |
| UR-06 | Experienced users shall be able to perform routine tasks (e.g., checking in, generating reports) in **under 1 minute** for each operation. |
| UR-07 | The system shall support keyboard navigation and accessibility shortcuts for all major actions, improving efficiency for power users. |

### User Satisfaction

User satisfaction measures how well the system meets user expectations and comfort. Feedback collection and monitoring are included to ensure continuous improvement.

| Requirement ID | Description |
| --- | --- |
| UR-08 | At least **85% of users** surveyed after system launch shall rate their experience as **"satisfactory" or better** in usability evaluations conducted each semester. |
| UR-09 | The system shall include a **feedback mechanism** (e.g., thumbs-up/down or comment box) on key pages to allow users to report satisfaction or usability concerns. |

### Accessibility

Accessibility ensures the system is usable by individuals with a wide range of abilities, complying with international standards for inclusivity and device compatibility.

| Requirement ID | Description |
| --- | --- |
| UR-10 | The user interface shall comply with **WCAG 2.1 Level AA** standards to support users with disabilities. |
| UR-11 | The system shall offer **responsive design** that works effectively on desktops, tablets, and mobile phones. |

## Interface requirements

This section defines the required interfaces between the Campus Event Check-in System and external systems, users, hardware devices, software modules, and communication protocols. Each interface is essential to ensure seamless integration, usability, and operational efficiency.

### System Interfaces

System interfaces describe how the application interacts with other systems in the university ecosystem.

| Requirement ID | Description |
| --- | --- |
| SI-01 | The system shall interface with the **university’s student information system (SIS)** to authenticate and validate student IDs during event check-in. |
| SI-02 | The system shall integrate with the **university’s official payment gateway** (e.g., FPX or Maybank2U) for processing payments. |
| SI-03 | The system shall connect to the university’s **event calendar system** to synchronize scheduled events, ensuring consistent event data across platforms. |
| SI-04 | The system shall support local offline data storage and later synchronization for check-in records. |
| SI-05 | The system shall integrate with an external email or SMS gateway (e.g., SMTP server or Twilio API) to send event reminders, payment confirmations, and check-in notifications to users. |

### User Interfaces

User interfaces define how users interact with the system through a web-based platform.

| Requirement ID | Description |
| --- | --- |
| UI-01 | The system shall provide a **responsive web interface** accessible on desktops, tablets, and smartphones. |
| UI-02 | The interface shall include a **fixed top navigation bar** with links to Dashboard, Events, Reports, Payments, and Settings (based on user role). |
| UI-03 | Forms shall include **input validation**, tooltips, and clear error messages to support user data entry. |
| UI-04 | Students shall have a **Check-In screen** with a text field for Student ID input or a QR scanner widget for fast access. |
| UI-05 | Event organizers shall have access to an **Event Management Dashboard** with tabs for event creation, edit, attendance overview, and financial summary. |
| UI-06 | Administrators shall have an **Admin Panel** with options for user role management, system settings, and audit logs. |

### Hardware Interfaces

Hardware interfaces describe the supported devices and peripherals needed for the system's functionality.

| Requirement ID | | Description |
| --- | --- | --- |
| HI-01 | The system shall support standard **QR code scanners** connected via USB or integrated into mobile devices for student check-in. | |
| HI-02 | The system shall be compatible with **touchscreen kiosks** for self-check-in stations located at event venues. | |
| HI-03 | The system shall not require any specialized hardware beyond existing university-issued devices (e.g., computers, mobile phones, scanners). | |

### Software Interfaces

Software interfaces describe how this system interacts with other software components or third-party APIs.

| Requirement ID | Description |
| --- | --- |
| SWI-01 | The system shall interact with the **university’s authentication API** to verify the validity of Student IDs and retrieve basic student profiles. |
| SWI-02 | The payment module shall consume the **REST API provided by the university’s payment gateway**, using secure token-based authentication. |
| SWI-03 | The reporting module shall generate reports in **PDF and CSV formats** using a third-party open-source reporting library (e.g., JasperReports). |

### Communications Interfaces

Communication interfaces define the protocols, data formats, and network security features used to exchange information.

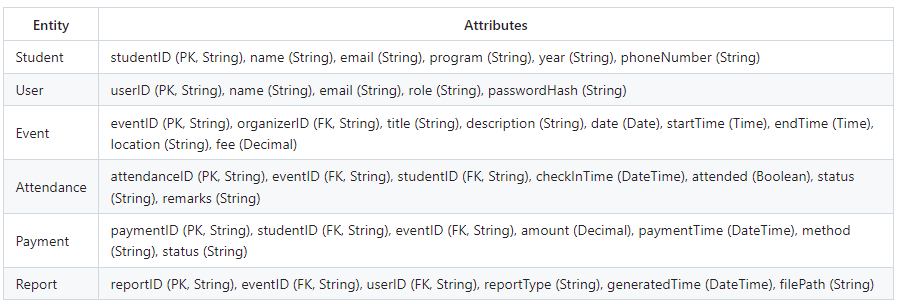
| Requirement ID | Description |
| --- | --- |
| CI-01 | All communication between the client interface and server shall use **HTTPS (TLS 1.2 or higher)** to ensure end-to-end encryption. |
| CI-02 | Data exchanged with external APIs (e.g., SIS, payment gateway) shall follow **RESTful architecture** using **JSON message format**. |
| CI-03 | The system shall support **WebSocket connections** for real-time updates (e.g., live attendance dashboards for event organizers). |
| CI-04 | The system shall be hosted within the university’s intranet with optional **VPN access** for external administration. |

## Logical database requirements

This section outlines the logical database requirements of the Campus Event Check-in System. It describes the essential data entities, their key attributes, relationships between the entities, and the constraints that ensure data integrity. These elements form the foundation for handling student authentication, event management, payment processing, and attendance reporting.

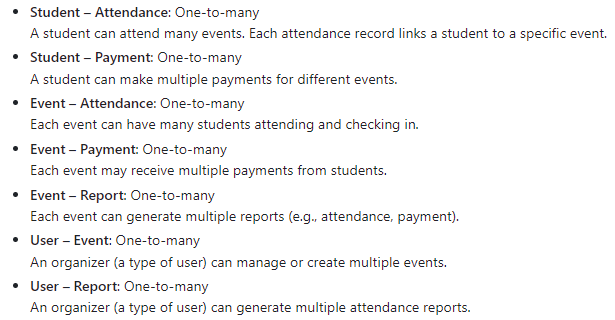
### Key Data Entities and Attributes

The core data entities used in the system along with their primary attributes. Each entity represents a real-world object (e.g., student, event, payment) necessary to perform the system's main operations



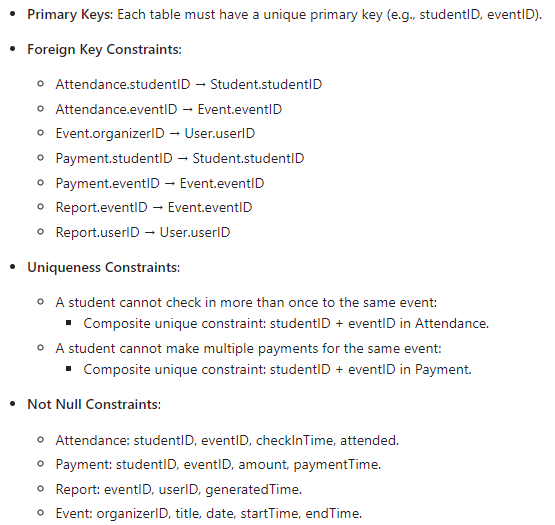
### Relationships

The following relationships define how entities in the system are connected. These connections reflect business rules and ensure the system supports real-world workflows.

****

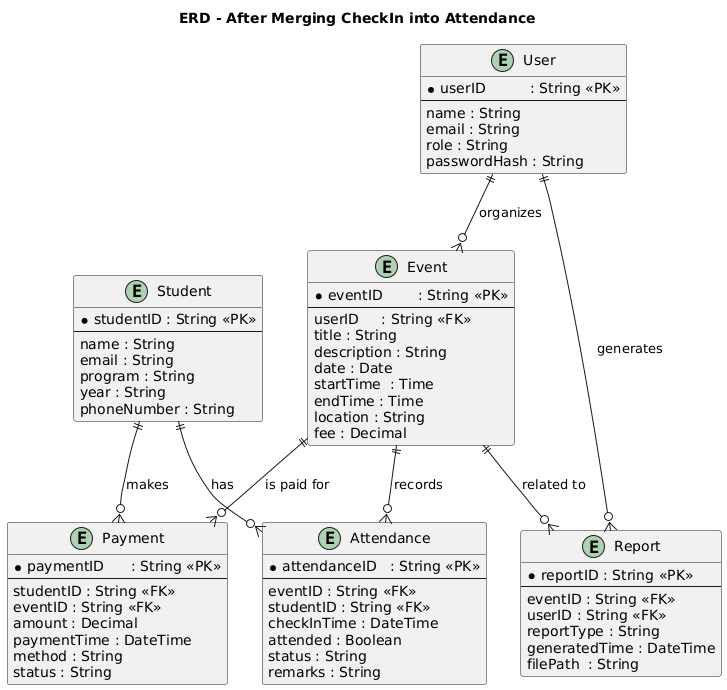
### Constraints

Rules that ensure the accuracy and consistency of the data. These constraints include primary keys, foreign keys, uniqueness rules, and not-null conditions that enforce data integrity.

****

### Entity Relationship Diagram (ERD)

a visual representation of the data entities and their relationships. The ERD simplifies understanding of how data flows between the system components.

****

## Design constraints

This section outlines the restrictions and limitations that affect the design of the Campus Event Check-in System. These constraints may originate from institutional policies, regulatory requirements, integration dependencies, or technical limitations of the university’s IT infrastructure. All design decisions must adhere to these constraints to ensure system compatibility, maintainability, and compliance with external standards.

### Compliance with University Policies

* The user interface must align with the university’s official branding and style guidelines, including color schemes, fonts, and logos.
* The system must adhere to the university’s IT security policies, including role-based access control and data encryption standards.

### Integration with Existing Infrastructure

* The system must be **compatible with the university’s existing student information system**, which will be used for authenticating Student IDs.
* The payment module must integrate exclusively with the university's **approved payment gateway**; third-party gateways are not supported.
* The system must **host all data on university-approved servers** to comply with institutional data governance policies.

### Regulatory and Privacy Constraints

* The system must **comply with Malaysia’s Personal Data Protection Act (PDPA)** in handling student and user data.
* Any storage or processing of personal data must ensure **data encryption at rest and in transit**.
* Access logs and user activities must be recorded in accordance with university audit and compliance requirements.

### Technical Limitations

* The system must be **web-based and mobile-friendly**, functioning on common browsers and devices used by students and staff.
* The design must **minimize external dependencies**, avoiding reliance on cloud services or APIs not approved by the university’s IT department.
* The system must function within the **resource constraints of existing campus hardware**, including shared network infrastructure and kiosk devices.

### Operational Constraints

* The system must allow **maintenance and updates without significant downtime**, ideally using a modular deployment strategy.
* All components must be designed for **interoperability within the university network**, with support for single sign-on (SSO) if adopted in the future.

## Software system attributes

This section outlines the key software quality attributes required to ensure the Campus Event Check-in System is reliable, secure, maintainable, and performs efficiently in its operational environment. These attributes directly impact the system’s usability, robustness, and long-term sustainability.

### Reliability

The system must operate continuously and correctly in the intended environment with minimal disruptions.

* The system shall be able to **recover from a crash or unexpected shutdown within 1 minute** without data loss.
* All critical operations (e.g., check-in, payment) shall include transaction logging and automatic retry mechanisms in case of temporary failure.
* The system shall maintain data integrity in the event of network interruptions or database outages.

### Availability

Availability is critical for events occurring during working hours and weekends.

* The system shall be **available at least 99.9% of the time during working hours**, defined as Monday to Friday, 8:00 AM to 6:00 PM.
* Scheduled maintenance must be limited to non-peak hours and communicated in advance to users.
* In the event of downtime, users should receive a clear message indicating the issue and estimated recovery time.

### Security

Security is essential to protect student data and financial transactions.

* The system shall implement **Role-Based Access Control (RBAC)**, restricting system functions based on user roles (Student, Organizer, Administrator).
* **All sensitive user data**, such as Student IDs and payment details, must be **encrypted at rest and in transit** using industry-standard encryption (e.g., AES-256 and HTTPS/TLS 1.2+).
* User sessions must expire after a period of inactivity and require re-authentication.

### Maintainability

Maintainability ensures the system can be updated and improved over time without major disruptions.

* The system shall be **developed using modular architecture**, allowing components (e.g., check-in, payment) to be updated independently.
* All source code shall follow **standardized coding conventions and documentation practices** (e.g., PEP 8 for Python or PSR-12 for PHP).
* A version control system (e.g., Git) shall be used for all codebase management and deployment tracking.

### Portability

The system should be easily deployable across different environments with minimal configuration.

* The software must run on both **Linux (e.g., Ubuntu Server)** and **Windows Server** operating systems **without requiring additional configuration**.
* Configuration files must use platform-independent formats (e.g., .env, JSON, YAML).
* The system should support containerization (e.g., Docker) to simplify cross-environment deployment.

## Supporting information

This section provides supplementary materials and contextual information that support the interpretation and implementation of the Campus Event Check-in System requirements. These details aid developers, testers, and stakeholders in understanding the project scope, functional behavior, and any special conditions related to deployment or usage.

### Sample Input/Output Formats

The following are sample formats for key system functions. These formats are for reference and **should be considered part of the system requirements**.

*Note: all the below code is json*

**a) Student Check-In Input via QR code scan or manual entry):**

{

"studentID": "1231303334",  
"eventID": "SRE-240501",  
"timestamp": "2025-05-20T09:02:34Z"

}

**b)** **Payment Transaction Output:**

{

"transactionID": "TXN789456123",  
"studentID": "1231303334",  
"eventID": "SRE-240501",  
"amount": 20.00,  
"paymentStatus": "Success",  
"paymentMethod": "Online FPX",  
"timestamp": "2025-05-20T09:03:00Z"

}

**c)** **Attendance Report Export Format (CSV):**

Event ID, Event Name, Student ID, Name, Check-in Time, Status

EVT-240501, Career Fair 2025, A2023000193, Sarah Lim, 2025-05-20 09:02, Present

### Elicitation and Research Sources

These sources **are not formal requirements**, but they provide context for the development team and stakeholders.

* A **questionnaire survey** was conducted among 80 university students, with 92% stating they prefer mobile check-in over manual methods.
* A **Kano Model** study was used during requirements elicitation to differentiate must-have vs. attractive features.
* A **cost-benefit analysis** indicates that reducing manual check-in overhead will save approximately 120 man-hours per semester.

### Problem Description

The software is designed to solve the following challenges in university event management:

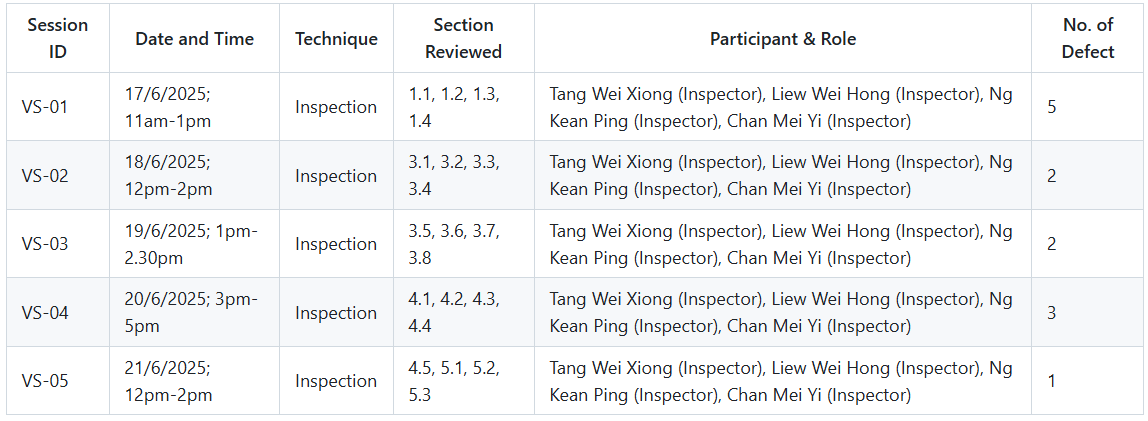
* **Inefficient manual attendance tracking**, leading to inaccurate records and long queues.
* **Limited payment transparency**, where event fees are collected without clear digital records.
* **Lack of real-time data**, preventing organizers from making timely decisions during event execution.
* **Cumbersome report generation**, which delays reconciliation and departmental reporting.

### Packaging and Deployment Instructions

The following packaging and security instructions **are part of the deployment requirements**:

* The source code will be packaged in a **Docker container** with environment variables securely injected via CI/CD pipeline.
* Database credentials and API keys will be stored using **encrypted environment secrets** (e.g., GitHub Actions Secrets, Azure Key Vault).
* Initial deployment requires setting up connections to:
  + **Student Information System (SIS)** using RESTful API tokens.
  + **University Payment Gateway** with approved client certificates.
* Export restrictions do not apply. The codebase is licensed under the university’s internal software policy and must not be shared with third-party vendors without prior authorization.

### 3.8.5 Validation Session



### 3.8.6 Defect Summary

#### Content Defect

### 

#### Document Defect



#### Aggrement Defect

### 

### 3.8.7 Conflict Analysis

### 

### 3.8.8 Conflict Analysis and Resolution

### 

### 

### 3.8.9 Change Log

### 

### 

### 3.8.10 Requirements Traceability Matrix

### 

### 3.8.11 Role in Requirements Validation, Negotiation & Management

### 

### 

### 3.8.12 Version Control & Configuration Summary

Repository Branch: project-part-2

Key Files:

SRS.md: Working version of updated SRS

TT1L\_G9\_SRS.doc: Final version

changelog.md: Record of all requirement-related changes

Commits Made by: Tang Wei Xiong

Pull Requests Merged by: Tang Wei Xiong

Change Log Entries Made by: Tang Wei Xiong

# **Verification**

## Verification Approach

* **How**: The system will be verified using functional testing, unit testing, and system integration testing to ensure it performs as required.
* **Who**: The product development team and the quality assurance (QA) department will be responsible for carrying out the verification activities.
* **When**: Verification will take place at key points during the development cycle, such as after the completion of each sprint.
* **Where**: All verification activities will be conducted within the QA testing environment.

## Verification Criteria

Verification ensures that all requirements gathered through various elicitation techniques are accurate, relevant, and feasible for implementation. It validates the authenticity of user needs, assesses the practicality of implementing features, and ensures alignment with system objectives. This step is critical for preventing costly rework and ensuring that the end product meets stakeholder expectations.

* **Completeness**: The set of requirements covers all essential aspects of a university event management system, including login, check-in, payment, reminders, and analytics.
* **Consistency**: No contradictions or overlaps were found among requirements obtained from interviews, observations, and brainstorming sessions.
* **Feasibility**: All proposed requirements are implementable using current technologies such as student ID systems, QR scanning, RFID, and cloud computing.
* **Traceability**: Requirements are properly recorded, categorized by the Kano model, and stored in the GitHub repository for easy access and version control.
* **User Validation**: Direct interaction with students through interviews and user surveys confirmed the practical value and relevance of the proposed features.

## Verification of Elicitation Methods

To verify the elicited requirements, each item was assessed based on three criteria: its source (interview, observation, or brainstorming), the method used to verify it (feedback, existing systems, or system simulation), and documented evidence (articles, transcripts, system design). The table below summarizes the verification process:

| **Requirement** | **Source** | **Verification Method** | **Verified Evidence** | **Kano Category** |
| --- | --- | --- | --- | --- |
| Student ID Login (SSO) | Interview | User feedback confirms preference for university credentials | Interview & GitHub | Must-Have |
| QR Code Check-In | Interview, Observation | | Verified by student responses and implemented in CU-Events system | | --- |  |  | | --- | | Interview, Article 01 | Performance |
| Email Confirmation | Interview | Cited as essential for transparency and trust | Interview | Must-Have |
| Calendar Sync/ Reminders | Interview | Students reported better attendance with reminders | Interview | Delighter |
| Autofill via SIS | Interview | Reduced data entry fatigue; enhances usability | Interview | Must-Have |
| Digital Certificate Vault | Interview | Seen as useful for resumes and portfolios | Interview | Delighter |
| Integrated Payment Gateway | Interview, Articles 01 & 02 | Validated through peer systems and preferred payment methods | Interview , Articles 01 & 02 | Performance |
| Event History Log | Interview | Students want visibility into past participation | Interview | Delighter |
| Offline Check-In Mode | Interview, Article 03 | Local sync ensure reliability in poor network areas | Interview,Article03 | Must-Have |
| Personalized Event Recommendations | Interview | Not required but boosts engagement | Interview | Delighter |

## Proof of Execution

### Techniques 1: Interview

Interviewer: Melvin Woo

Roles: Student

| **Interview Questions** | **Feature** | **Insight Gained** | **Kano Category** |
| --- | --- | --- | --- |
| 1. How important is it for you to log in using your Student ID instead of creating a new account? | Student Authentication (SSO via SIS) | Students prefer official university login for ease and security. | Must-Have |
| 2. Would you prefer scanning a QR code to check in at events over manual sign-ins? | QR Code-Based Check-In | QR codes are favored for speed and convenience. | Performance |
| 3. Is receiving a confirmation email after registration and payment important to you? | Email Notification & Payment Receipt | Confirmation builds trust and ensures transaction transparency. | Must-Have |
| 4. How helpful would it be to get automatic reminders for events you've signed up for? | Calendar Sync / Notifications | Students often forget events; reminders improve attendance. | Delighter |
| 5. Would you like your personal details to autofill during event registration? | SIS Integration for Autofill | Avoids repetitive entry, improves user experience. | Must-Have |
| 6. Would you find it useful to store and download participation certificates in the system? | Digital Certificate Vault | Certificates are useful for resumes and portfolios. | Delighter |
| 7. Do you prefer paying for events through UPI, credit/debit card, or e-wallets? | Integrated Payment Gateway | Mobile-friendly payments are essential for students. | Performance |
| 8. Would it be useful to view your past event attendance in your profile? | Event History Log | Tracking participation helps students manage involvement. | Delighter |
| 9. How would you feel if check-in failed due to internet issues at the venue? | Offline Check-In Mode | Offline functionality is expected for reliability. | Must-Have |
| 10. Would personalize event suggestions based on your interests improve your experience? | Personalized Event Recommendations | Not expected but would increase engagement. | Delighter |

**Outcomes:**

| **Feature** | **Kano Category** | **Justification** |
| --- | --- | --- |
| Student ID Login (SSO) | Must-Have | Students expect seamless login using university credentials; builds trust. |
| QR Code-Based Check-In | Performance | Significantly speeds up the entry process; preferred over manual sign-in. |
| Email Confirmation & Receipts | Must-Have | Essential for user confidence in payment and registration tracking. |
| Calendar Sync / Event Reminders | Delighter | Unexpected but appreciated; improves attendance and time management. |
| Autofill Student Details via SIS | Must-Have | Repetition is frustrating; students expect pre-filled forms for speed. |
| Digital Certificate Vault | Delighter | Storing downloadable proof of participation adds long-term value. |
| Integrated Payment Gateway | Performance | Students value flexible and mobile payment methods like UPI and e-wallets. |
| Attendance History Log | Delighter | Helps students track involvement; adds transparency and reflection value. |
| Offline Mode for Check-In | Must-Have | Reliability is expected regardless of internet conditions at the event site. |
| Personalized Event Recommendations | Delighter | Enhances engagement; not required, but positively influences event discovery. |

### Techniques 2: Observation

**Article 01**

Authors: Aditi Chaturvedi, Krishna Sharma, Akshat Dua, Aastha Gupta

Publisher: IJRASET Journal for Research in Applied Science and Engineering Technology

Link:<https://www.ijraset.com/best-journal/cuevents-a-comprehensive-event-management-system-for-university>

**Objective:**

The primary goal of this study is to design and implement CU-Events, a centralized event management platform tailored for universities. This system aims to address common challenges in campus event management such as inefficient manual registration processes, poor attendance tracking, and lack of centralized data for event organizers. CU-Events seeks to streamline event creation, participant registration, and real-time check-in using student IDs, all while incorporating features to handle payment integration for paid events.

**Methodology:**

The system is developed based on a modular, three-tier architecture including the presentation layer (front-end), business logic layer (back-end), and data layer (database). The front-end utilizes React.js for creating an interactive user interface that supports easy navigation for both students and organizers. The back-end is built on Node.js and Express.js, ensuring efficient processing of requests and robust API services. A MySQL database manages event details, user profiles, attendance, and payment records. The study involved iterative development with continuous feedback from students and campus event managers to refine usability and performance.

**Findings:**

CU-Events demonstrated marked improvements in administrative efficiency by automating registration and attendance tracking. The integration of student ID scanning reduced entry bottlenecks during events. Payment integration allowed for smooth transactions for paid events without the need for separate payment portals. User surveys indicated increased satisfaction due to the system's ease of use and quick access to event information. The system also facilitated detailed analytics for event attendance and revenue, assisting campus officials in making data-driven decisions for future event planning.

**Article 02**

Authors: Sun Yang, Lixia Wen

Publisher: Open Journal of Social Sciences

Link: <https://www.scirp.org/journal/paperinformation?paperid=101114>

**Objective:**

This research aims to develop a unified virtual payment system tailored for campus environments that consolidates various payment services, including event fee collection, dining, transportation, and other campus-related payments. The objective is to improve campus financial operations by creating an intelligent, secure, and convenient payment platform that can integrate seamlessly with student ID cards and mobile applications.

**Methodology:**

The study employed a three-layer architectural design: an infrastructure layer to provide the foundational hardware and network environment; a core framework layer to handle payment processing, authentication, and security; and a service platform layer delivering user-facing payment applications. The implementation leverages microservices architecture deployed via Docker containers and Kubernetes for scalability. Spring Cloud supports service discovery and fault tolerance. The research included prototype development and simulated testing with mock payment transactions under high concurrency to evaluate system stability and response time.

**Findings:**

The virtual payment system effectively supports multiple payment methods, including QR codes, NFC, and direct integration with student ID cards. The system shows high scalability and fault tolerance, handling thousands of transactions concurrently without downtime. It also provides robust security measures such as encrypted communication and two-factor authentication. Campus administration reported a significant reduction in manual billing errors and improved financial tracking. Students benefited from a one-stop payment interface, reducing the need to carry multiple cards or cash, thereby enhancing convenience and safety.

**Article 03**

Authors: Ibad Ali, Bhushanwar, Atulbahi Vaghela, Khodifad, Dayaram Patil, Pandwal

Publisher: ResearchGate

Link:<https://www.researchgate.net/publication/389201749_The_Campus_Pay_Offline_Payment_System_Using_RFID_Technology_for_Campus_Transactions>

**Objective:**

This study focuses on the development of an offline campus payment system that leverages RFID-enabled student ID cards to facilitate fast, secure, and reliable transactions in areas with limited or unstable internet connectivity. The system targets campus facilities such as event venues, cafeterias, and bookstores, allowing students to make payments without the delays or disruptions caused by network failures.

**Methodology:**

The system architecture integrates RFID readers at transaction points and a local database that records payments temporarily when offline. Transactions are encrypted and stored securely, then synchronized with a centralized cloud database once internet connectivity is restored. The team developed a prototype system using Arduino microcontrollers, RFID tags embedded in student IDs, and a custom-built POS terminal. Field testing was conducted on a university campus to monitor transaction speed, error rates, and user acceptance under varying network conditions.

**Findings:**

Results showed the offline RFID system reduced transaction times significantly compared to traditional cash or card payments, especially in high-traffic areas during event check-ins. Users reported high satisfaction due to ease of use and reliability. The system maintained data integrity with no reported loss of transaction records after synchronization. Campus administrators highlighted improved operational continuity even during network outages, making the system ideal for remote or infrastructure-limited environments. The study concluded that RFID-based offline payments offer a scalable and cost-effective solution for campus financial transactions.

**Observation Technique Tables**

| **Article Title** | **Observation Type** | **Observation Context** | **Tools/Tech Used** | **Parameters Observed** | **Findings/Outcomes** |
| --- | --- | --- | --- | --- | --- |
| **CU-Events: A Comprehensive Event Management System for University** | System Usage & User Behavior | Observing system interaction during event registration and check-in phases using student ID and payment modules | React.js (UI), Node.js, Express.js, MySQL, Student ID Scanner | Registration time, check-in speed, user satisfaction, payment success rates | Reduced entry bottlenecks; user satisfaction increased; smooth payment experience; enabled event data analytics for planners |
| **Development of Unified Virtual Payment System in Campus Environment** | Simulated System Performance | Observing virtual payment platform under stress conditions with mock transactions | Spring Cloud, Docker, Kubernetes, Student ID/NFC integration | Transaction throughput, system response under load, authentication success rate | High scalability; supported high concurrency; secure transaction handling; improved financial tracking; users appreciated simplified digital payments |
| **The Campus Pay: Offline Payment System Using RFID Technology** | Field-Based Functional Test | Observing real-world transactions in offline conditions across campus (cafeterias, bookstores, events) | RFID Tags, Arduino, Local DB, POS Terminal, Sync to Cloud | Transaction time, data synchronization accuracy, network dependency, error rates | Reliable in low-connectivity zones; fast transaction times; no data loss post-sync; high user and admin satisfaction; ideal for infrastructure-limited areas |

**Key Observation**

| **Aspect** | **Common Trend Across Articles** | **Kano Category** |
| --- | --- | --- |
| **Student ID Integration** | All systems utilized student ID cards (with RFID or digital scan) for check-ins and payments, streamlining access and increasing accuracy. | Must-Be |
| **Check-in Speed & Efficiency** | Real-time or offline systems reduced bottlenecks at event entries and service points, enhancing the overall user experience. | One-Dimensional |
| **Payment Integration** | Systems offered digital or offline payment integration, reducing dependency on cash and external systems. | Must-Be |
| **Offline Functionality (RFID Systems)** | RFID-based systems continued to function during internet outages, syncing data when back online—crucial in unstable network environments. | Attractive |
| **System Usability (UI/UX Design)** | User interfaces were designed for ease of use; users expressed satisfaction with system navigation and functionality. | One-Dimensional |
| **Analytics & Reporting Tools** | Event organizers and admins used analytics for attendance tracking and revenue analysis, improving data-driven decision-making. | Attractive |
| **Multi-Payment Options (NFC, QR, ID)** | Support for various payment modes (e.g., QR code, NFC, student ID) gave users flexibility and choice. | Attractive |
| **Scalability & Performance** | Systems handled high transaction volumes smoothly due to modular and scalable architectures (e.g., microservices, cloud computing). | One-Dimensional |
| **Security & Data Integrity** | All systems used encryption and secure authentication; no data loss was reported, even during offline syncing. | Must-Be |
| **User Satisfaction & Adoption** | Students and admins showed high satisfaction, especially with simplified processes and faster transactions, leading to higher system adoption. | One-Dimensional |

### Techniques 3: Brainstorming

**Questions:**

| **Prompt (Question)** | **Goal** |
| --- | --- |
| What are the problems with current campus event check-in methods? | Identify pain points and justify the need for a new system. |
| How can student IDs be used to simplify event check-ins? | Explore methods to automate and speed up the check-in process. |
| What types of payment should be integrated? | Define payment flexibility and cater to user preferences. |
| Should the system operate offline? | Determine the necessity of offline capabilities. |
| What data is needed at check-in and how can privacy be ensured? | Define data requirements and ensure privacy compliance. |
| What benefits can this system provide to event organizers? | Identify administrative advantages and added value. |
| What kind of user interface would work best for students and staff? | Improve usability for both user groups. |
| How should users receive payment confirmations? | Clarify how payment status and confirmations are communicated. |
| What platforms should the system support? | Define technical scope and accessibility across devices. |
| How can the system be tested before launch? | Plan for reliability testing and user feedback collection. |

**Outcomes:**

| **Outcome (Idea)** | **Kano Category** | **Notes** |
| --- | --- | --- |
| Replace manual check-ins with automated scanning via student IDs. | Must-Be | Manual processes are slow, error-prone, and frustrating for users. |
| Use RFID/QR/NFC-enabled student IDs for real-time scanning and attendance tracking. | One-Dimensional | Links attendance to student profiles and speeds up entry. |
| Support debit/credit cards, mobile wallets, and prepaid campus accounts. | One-Dimensional | Multiple options improve convenience and centralized tracking. |
| Use offline-capable RFID readers with local storage and sync when online. | Attractive | Essential for events with poor network connectivity. |
| Collect minimal data with encryption and access control. | Must-Be | Privacy compliance is critical; minimize and protect data. |
| Provide an admin dashboard for attendance, payments, and analytics. | Attractive | Facilitates data-driven decisions and reduces manual effort. |
| Mobile-first UI for students; web dashboard for staff. | One-Dimensional | Intuitive design boosts adoption and reduces training. |
| Auto-generate digital receipts and QR check-in tickets sent by email/app. | Must-Be | Confirmation reassures users and speeds entry validation. |
| Support web (admin), mobile app (students), and optional kiosks. | Attractive | Cross-platform increases accessibility and convenience. |
| Pilot test at campus events; stress test concurrency; collect user feedback. | Must-Be | Identifies bugs and improves UX before full rollout. |

## Overall Elicitation

The execution of the elicitation phase was carefully documented and cross-referenced for accuracy and relevance:

* **Interviews** were conducted with students to collect direct feedback on desired system features. Transcripts and question mappings are recorded in the elicitation document.
* **Observation findings** were drawn from peer-reviewed academic articles that describe existing systems implemented in university environments. This external validation demonstrates both the feasibility and success of similar solutions.
* **Brainstorming sessions** allowed the team to creatively identify technical solutions to user needs. These sessions generated a list of implementable ideas aligned with Kano categorization.

Each technique reinforced the reliability of the requirements. For instance, observing CU-Events and RFID systems validated the practicality of check-in and payment features. Interview responses highlighted the importance of convenience, speed, and system trust. Brainstorming brought together perspectives from various stakeholders to envision a user-centric, technically sound system.

All documentation—interview notes, article reviews, and brainstorming outcomes—is maintained in the project’s GitHub repository, ensuring transparency, accessibility, and version control for future use: <https://github.com/Rwppy/SRE_G6>

# **Appendices**

## Assumptions and dependencies

This section outlines the key assumptions made during the design and development of the Campus Event Check-in System, along with the dependencies that must be in place for the successful implementation and functioning of the system.

**Assumptions**

1. **Availability of Student Information System (SIS):**  
   It is assumed that the university has an existing and consistently operational Student Information System (SIS) that stores up-to-date student data, including Student ID numbers and authentication credentials.
2. **Access to Web-Enabled Devices:**  
   All intended users (students, organizers, administrators) will have access to at least one web-enabled device (e.g., smartphone, laptop, or desktop) with a modern web browser to interact with the system’s web interface.
3. **Basic Digital Literacy:**  
   Users, especially students and event organizers, are expected to possess a basic level of digital literacy, including the ability to log in, navigate web forms, scan QR codes, and complete online transactions.
4. **Authorized Event Management:**  
   Only university-approved personnel (staff or authorized students) are permitted to create and manage events within the system, ensuring that all listed events are legitimate and relevant to the university community.
5. **Data Security and Privacy Compliance:**  
   The university’s IT infrastructure and policies will support compliance with data protection regulations and best practices for handling sensitive information, including student attendance records and payment details.
6. **Stable Internet Connectivity:**  
   It is assumed that users will have consistent access to internet services for real-time operations such as check-ins, data synchronization, and payment processing.
7. **User Cooperation and Honesty:**  
   The system presumes that users will behave in accordance with university regulations and will not attempt to circumvent the check-in process or misuse their privileges.

**Dependencies**

1. **Student Information System (SIS):**  
   The system depends on seamless integration with the university’s SIS for student identity verification and autofill features. Any disruption in this system may impact check-in accuracy and user data integrity.
2. **Payment Gateway Integration:**  
   The ability to process payments for paid events relies on the proper functioning of the university’s official payment gateway. This includes support for online and point-of-sale transactions.
3. **Email Service Provider:**  
   The system requires access to a reliable email service to send notifications, payment confirmations, event reminders, and digital attendance certificates to users.
4. **Hosting Infrastructure:**  
   The web application must be hosted on reliable university servers or cloud platforms with sufficient storage, bandwidth, and uptime to handle peak traffic, especially during large-scale events.
5. **Barcode or QR Code Scanner Support:**  
   To facilitate rapid check-ins, the system requires compatibility with barcode or QR code scanning tools, either through students' mobile devices or on-site kiosk scanners.
6. **Data Backup and Disaster Recovery Systems:**  
   The system’s dependability is contingent on the availability of secure data backup procedures and disaster recovery plans managed by the university’s IT department.

## 

## Acronyms and abbreviations

The following table lists the acronyms and abbreviations used throughout this Software Requirements Specification (SRS) document, along with their full terms and explanations:

| Acronym / Abbreviation | Full Term | Description |
| --- | --- | --- |
| **SRS** | Software Requirements Specification | A formal document that outlines the functional and non-functional requirements of the system. |
| **SIS** | Student Information System | A university database system used to manage and store student records and academic data. |
| **ID** | Identification | A unique identifier (e.g., Student ID) used for user authentication and tracking. |
| **UI** | User Interface | The graphical layout and interactive components through which users interact with the system. |
| **UX** | User Experience | The overall experience of users when using the system, including ease of use and satisfaction. |
| **SSO** | Single Sign-On | A centralized login system allowing users to access multiple services using one set of credentials. |
| **Qr Code** | Quick Response Code | A machine-readable barcode that stores information, often used for check-ins or linking to URLs. |
| **API** | Application Programming Interface | A set of functions and protocols that allow different software systems to communicate. |
| **DB** | Database | A structured collection of data stored electronically and accessed by the system’s backend. |
| **HTTP** | Hypertext Transfer Protocol | A protocol used for transmitting data over the web, essential for the system’s web interface. |
| **SSL** | Secure Socket Layer | A security protocol that provides encrypted communication between the client and the server. |
| **CRUD** | Create,Read,Update,Delete | Basic operations performed on data stored in databases, essential for event and user management. |

## Glossary

* **Check-in**: The process of registering presence at an event using the system.
* **Student ID**: A unique identifier assigned to each student by the university.
* **Payment Gateway**: A service that authorizes and processes payments securely.
* **Event Organizer**: A staff or student responsible for managing campus events.
* **Authentication**: The process of verifying user identity before granting access.