A blue and black logo

AI-generated content may be incorrect.

CSE6224-SOFTWARE REQUIREMENTS ENG Assignment (35%)

Title: Campus Event Check-in System

Prepared By:

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Contribution** |
| 1211305566 | ALKATHERI, SULAIMAN ALI MAHDI | 25% |
| 1221101391 | LIM AI NEE | 25% |
| 1221100800 | MUHAMMAD AZHAR BIN AHMAD ROSEHAIZAT | 25% |
| 1211103488 | YOUSEF ELSAYED EZZAT ELSAYED | 25% |

Group: 1 TT1L

Supervision By:

Dr. Zarina Binti Che Embi ([zarina.embi@mmu.edu.my](mailto:wan.noorshahida.isa@mmu.edu.my))

**Table of Contents**

[List of Figures 5](#_Toc199082787)

[List of Tables 6](#_Toc199082788)

[1. Introduction 11](#_Toc199082789)

[1.1 Purpose 11](#_Toc199082790)

[1.2 Scope 11](#_Toc199082791)

[1.3 Product Perspective 11](#_Toc199082792)

[1.3.1 System Interface 12](#_Toc199082793)

[1.3.2 User Interface 13](#_Toc199082794)

[1.3.3 Hardware Interface 14](#_Toc199082795)

[1.3.4 Software Interface 15](#_Toc199082796)

[1.3.5 Communication Interface 16](#_Toc199082797)

[1.4 Product Functions 17](#_Toc199082798)

[1.5 User Characteristics 19](#_Toc199082799)

[1.6 Limitations 20](#_Toc199082800)

[1.7 Assumptions and Dependencies 21](#_Toc199082801)

[1.8 Definitions 21](#_Toc199082802)

[2. Requirements 23](#_Toc199082803)

[2.1 External Interfaces 23](#_Toc199082804)

[2.1.1 Login User Interface 23](#_Toc199082805)

[2.1.2 View Upcoming Events Interface 24](#_Toc199082806)

[2.1.3 Event details 25](#_Toc199082807)

[2.1.4 Register for Event 27](#_Toc199082808)

[2.1.5 Make Payment 29](#_Toc199082809)

[2.1.6 Receive e-ticket/QR code 30](#_Toc199082810)

[2.1.7 View Registration History 31](#_Toc199082811)

[2.1.8 Show QR Attendance 32](#_Toc199082812)

[2.1.9 Set Event Details 34](#_Toc199082813)

[2.1.10 View Events (Organizer) 36](#_Toc199082814)

[2.1.11 View registrations and attendance status 37](#_Toc199082815)

[2.2 Functions 38](#_Toc199082816)

[2.2.1 F00 View Upcoming Events 38](#_Toc199082817)

[2.2.2 F01 View Event Details 39](#_Toc199082818)

[2.2.3 F02 Register For Event 41](#_Toc199082819)

[2.2.4 F03 Make Payment 44](#_Toc199082820)

[2.2.5 F04 Receive e-Ticket/QR Code 47](#_Toc199082821)

[2.2.6 F05 View Registration History 48](#_Toc199082822)

[2.2.7 F06 Show QR Attendance 51](#_Toc199082823)

[2.2.8 F07 Create a New Event & F08 Set Event Details 52](#_Toc199082824)

[2.2.9 F09 View Events 54](#_Toc199082825)

[2.2.10 F10 View Registrations & Attendance Status 56](#_Toc199082826)

[2.2.11 F11 Check-in Upon Arrival 58](#_Toc199082827)

[2.2.12 F12 Login 59](#_Toc199082828)

[2.3 State Transition Diagrams 61](#_Toc199082829)

[2.4 Performance Requirements 62](#_Toc199082830)

[2.5 Security Requirements 63](#_Toc199082831)

[2.6 Usability Requirements 64](#_Toc199082832)

[2.7 Logical Database Requirements 65](#_Toc199082833)

[2.7.1 Event Table 66](#_Toc199082834)

[2.7.2 Registration Table 67](#_Toc199082835)

[2.7.3 Payment Table 67](#_Toc199082836)

[2.8 Design Constraints 68](#_Toc199082837)

[2.9 Software System Attributes 69](#_Toc199082838)

[2.10 Supporting Information 70](#_Toc199082839)

[2.10.1 Interview 71](#_Toc199082840)

[2.10.2 Questionnaires 72](#_Toc199082841)

[2.10.3 Observation 73](#_Toc199082842)

[3. Verification 74](#_Toc199082843)

[3.1 Inspection 74](#_Toc199082844)

[3.2 Walkthroughs 75](#_Toc199082845)

[3.3 Prototyping 75](#_Toc199082846)

[4. Appendices 76](#_Toc199082847)

[5. References 78](#_Toc199082848)

# List of Figures

[Figure 2.2.1. 1 F00 View Upcoming Events Sequence Diagram 39](#_Toc199082712)

[Figure 2.2.2. 1 F01 View Event Details Sequence Diagram 41](#_Toc199082720)

[Figure 2.2.3. 1 F02 Register for Event Sequence Diagram 44](#_Toc199062393)

[Figure 2.2.4. 1 F03 Make Payment Sequence Diagram Version 1 46](#_Toc199062402)

[Figure 2.2.4. 2 F03 Make Payment Sequence Diagram Version 2 47](#_Toc199062403)

[Figure 2.2.5. 1 F04 Receive e-Ticket/QR Code Sequence Diagram F04 Receive e-Ticket/QR Code Sequence Diagram 48](#_Toc199062603)

[Figure 2.2.6. 1 F05 View Registration History Sequence Diagram 50](#_Toc199062630)

[Figure 2.2.7. 1 F06 Show QR Attendance Sequence Diagram 52](#_Toc199062642)

[Figure 2.2.8. 1 F07 Set Event Details Sequence Diagram 54](#_Toc199062654)

[Figure 2.2.9. 1 F09 View Events Sequence Diagram 56](#_Toc199062665)

[Figure 2.2.10. 1 F10 View Registrations & Attendance Status Sequence Diagram 57](#_Toc199062718)

[Figure 2.2.11. 1 F11 Scan Attendance QR Code & Update Attendance Status Sequence Diagram 59](#_Toc199062725)

[Figure 2.2.12. 1 F12 Login Sequence Diagram 60](#_Toc199062734)

[Figure 2.3. 1 High level State Transition Diagram 61](#_Toc199062753)

[Figure 2.3. 2 Event Organizer State Transition Diagram 62](#_Toc199062754)

[Figure 2.3. 3 Student State Transition Diagram 62](#_Toc199062755)

[Figure 2.7. 1 System ERD 66](#_Toc199062764)

# List of Tables

[Table 1.3.2. 1 User Interface Requirements 13](#_Toc199062830)

[Table 1.3.3. 1 Hardware Interface Requirements 14](#_Toc199062836)

[Table 1.3.4. 1 Software Interface Requirements 15](#_Toc199062843)

[Table 1.3.5. 1 Communication Interface Requirements 16](#_Toc199062850)

[Table 1.4. 1 Campus Event Check-in System Function Table 17](#_Toc199082881)

[Table 1.5. 1 Intended Users and Expected Knowledge 20](#_Toc199082895)

[Table 1.8. 1 Glossary of Terms for the Campus Event Check-in System 21](#_Toc199082917)

[Table 2.1.1. 1 REQ\_IO0001 Table 23](#_Toc199062968)

[Table 2.1.1. 2 REQ\_IO0002 Table 24](#_Toc199062969)

[Table 2.1.1. 3 REQ\_IO0003 Table 24](#_Toc199062970)

[Table 2.1.1. 4 REQ\_IO0004 Table 24](#_Toc199062971)

[Table 2.1.2. 1 REQ\_IO0101 Version 1 Table 25](#_Toc199062978)

[Table 2.1.2. 2 REQ\_IO0101 Version 2 Table 25](#_Toc199062979)

[Table 2.1.3. 1 REQ\_IO0201 Table 25](#_Toc199063001)

[Table 2.1.3. 2 REQ\_IO0202 Table 25](#_Toc199063002)

[Table 2.1.3. 3 REQ\_IO0203 Table 26](#_Toc199063003)

[Table 2.1.3. 4 REQ\_IO0204 Table 26](#_Toc199063004)

[Table 2.1.3. 5 REQ\_IO0205 Table 26](#_Toc199063005)

[Table 2.1.3. 6 REQ\_IO0206Table 26](#_Toc199063006)

[Table 2.1.3. 7 REQ\_IO0207 Table 27](#_Toc199063007)

[Table 2.1.3. 8 REQ\_IO0208 Table 27](#_Toc199063008)

[Table 2.1.4. 1 REQ\_IO0301 Version 1 Table 27](#_Toc199063009)

[Table 2.1.4. 2 REQ\_IO0302 Version 1 Table 28](#_Toc199063010)

[Table 2.1.4. 3 REQ\_IO0303 Table 28](#_Toc199063011)

[Table 2.1.4. 4 REQ\_IO0304 Table 28](#_Toc199063012)

[Table 2.1.4. 5 REQ\_IO0305 Table 28](#_Toc199063013)

[Table 2.1.4. 6 REQ\_IO0301 Version 2 Table 29](#_Toc199063014)

[Table 2.1.4. 7 REQ\_IO0302 Version 2 Table 29](#_Toc199063015)

[Table 2.1.5. 1 REQ\_IO0401 Table 29](#_Toc199063048)

[Table 2.1.5. 2 REQ\_IO0402 Table 30](#_Toc199063049)

[Table 2.1.5. 3 REQ\_IO0403 Table 30](#_Toc199063050)

[Table 2.1.5. 4 REQ\_IO0404 Table 30](#_Toc199063051)

[Table 2.1.6. 1 REQ\_IO0501 Table 30](#_Toc199063067)

[Table 2.1.6. 2 REQ\_IO0502 Table 31](#_Toc199063068)

[Table 2.1.6. 3 REQ\_IO0503 Table 31](#_Toc199063069)

[Table 2.1.7. 1 REQ\_IO0601 Table 31](#_Toc199063078)

[Table 2.1.7. 2 REQ\_IO0602 Table 32](#_Toc199063079)

[Table 2.1.7. 3 REQ\_IO0603 Table 32](#_Toc199063080)

[Table 2.1.7. 4 REQ\_IO0604 Table 32](#_Toc199063081)

[Table 2.1.8. 1 REQ\_IO0701 Version 1 Table 32](#_Toc199063090)

[Table 2.1.8. 2 REQ\_IO0701 Version 2 Table 33](#_Toc199063091)

[Table 2.1.8. 3 REQ\_IO0702 Version 1 Table 33](#_Toc199063092)

[Table 2.1.8. 4 REQ\_IO0702 Version 2 Table 33](#_Toc199063093)

[Table 2.1.8. 5 REQ\_IO0703 Version 1 Table 33](#_Toc199063094)

[Table 2.1.8. 6 REQ\_IO0703 Version 2 Table 34](#_Toc199063095)

[Table 2.1.9. 1 REQ\_IO0801 Table 34](#_Toc199063103)

[Table 2.1.9. 2 REQ\_IO0802 Table 34](#_Toc199063104)

[Table 2.1.9. 3 REQ\_IO0803 Table 35](#_Toc199063105)

[Table 2.1.9. 4 REQ\_IO0804 Table 35](#_Toc199063106)

[Table 2.1.9. 5 REQ\_IO0805 Table 35](#_Toc199063107)

[Table 2.1.9. 6 REQ\_IO0806 Table 35](#_Toc199063108)

[Table 2.1.9. 7 REQ\_IO0807 Table 36](#_Toc199063109)

[Table 2.1.9. 8 REQ\_IO0808 Table 36](#_Toc199063110)

[Table 2.1.10. 1 REQ\_IO0901 Table 36](#_Toc199063130)

[Table 2.1.10. 2 REQ\_IO0902 Table 36](#_Toc199063131)

[Table 2.1.10. 3 REQ\_IO0903 Table 37](#_Toc199063132)

[Table 2.1.10. 4 REQ\_IO0904 Table 37](#_Toc199063133)

[Table 2.1.11. 1 REQ\_IO1001 Table 37](#_Toc199063143)

[Table 2.1.11. 2 REQ\_IO1002 Table 38](#_Toc199063144)

[Table 2.1.11. 3 REQ\_IO1003 Table 38](#_Toc199063145)

[Table 2.2.1. 1 REQ\_F0001 Table 38](#_Toc199063172)

[Table 2.2.1. 2 F00 View Available Events Use Case Specification Table 39](#_Toc199063173)

[Table 2.2.2. 1 REQ\_F0101 Table 39](#_Toc199063181)

[Table 2.2.2. 2 REQ\_F0102 Table 40](#_Toc199063182)

[Table 2.2.2. 3 F01 View Event Details Use Case Specification Table 40](#_Toc199063183)

[Table 2.2.3. 1 REQ\_F0201 Version 1 Table 41](#_Toc199063208)

[Table 2.2.3. 2 REQ\_F0202 Table 41](#_Toc199063209)

[Table 2.2.3. 3 REQ\_F0203 Table 42](#_Toc199063210)

[Table 2.2.3. 4 REQ\_F0204 Table 42](#_Toc199063211)

[Table 2.2.3. 5 REQ\_F0205 Version 1 Table 42](#_Toc199063212)

[Table 2.2.3. 6 REQ\_F0206 Version 1 Table 42](#_Toc199063213)

[Table 2.2.3. 7 REQ\_F0201 Version 2 Table 42](#_Toc199063214)

[Table 2.2.3. 8 REQ\_F0205 Version 2 Table 42](#_Toc199063215)

[Table 2.2.3. 9 REQ\_F0206 Version 2 Table 43](#_Toc199063216)

[Table 2.2.3. 10 F02 Register for Event Use Case Specification Table Version 1 43](#_Toc199063217)

[Table 2.2.3. 11 F02 Register for Event Use Case Specification Table Version 2 43](#_Toc199063218)

[Table 2.2.4. 1 REQ\_F0301 Table 45](#_Toc199063225)

[Table 2.2.4. 2 REQ\_F0302 Table 45](#_Toc199063226)

[Table 2.2.4. 3 REQ\_F0303 Table 45](#_Toc199063227)

[Table 2.2.4. 4 F03 Make Payment Use Case Specification Table Version 1 45](#_Toc199063228)

[Table 2.2.4. 5 F03 Make Payment Use Case Specification Table Version 2 46](#_Toc199063229)

[Table 2.2.5. 1 REQ\_F0401 Tale 47](#_Toc199063237)

[Table 2.2.5. 2 F04 Receive e-Ticket/QR Code Use Case Specification Table 47](#_Toc199063238)

[Table 2.2.6. 1 REQ\_F0501 Table 49](#_Toc199063260)

[Table 2.2.6. 2 REQ\_F0502 Table 49](#_Toc199063261)

[Table 2.2.6. 3 REQ\_F0503 Table 49](#_Toc199063262)

[Table 2.2.6. 4 F05 View Registration History Use Case Specification Table Version 1 49](#_Toc199063263)

[Table 2.2.6. 5 F05 View Registration History Use Case Specification Table Version 2 50](#_Toc199063264)

[Table 2.2.7. 1 REQ\_F0601 Table 51](#_Toc199063355)

[Table 2.2.7. 2 REQ\_F0602 Table 51](#_Toc199063356)

[Table 2.2.7. 3 REQ\_F0603 Table 51](#_Toc199063357)

[Table 2.2.7. 4 REQ\_F0604 Table 51](#_Toc199063358)

[Table 2.2.7. 5 F06 Show QR Attendance Use Case Specification Table 51](#_Toc199063359)

[Table 2.2.8. 1 REQ\_F0701 Table 53](#_Toc199063369)

[Table 2.2.8. 2 REQ\_F0702 Table 53](#_Toc199063370)

[Table 2.2.8. 3 REQ\_F0703 Table 53](#_Toc199063371)

[Table 2.2.8. 4 REQ\_F0704 Table 53](#_Toc199063372)

[Table 2.2.8. 5 F07 Set Event Details Use Case Specification Table 53](#_Toc199063373)

[Table 2.2.9. 1 REQ\_F0901 Table 54](#_Toc199063399)

[Table 2.2.9. 2 REQ\_F0902 Table 55](#_Toc199063400)

[Table 2.2.9. 3 REQ\_F0903 Table 55](#_Toc199063401)

[Table 2.2.9. 4 REQ\_F0904 Table 55](#_Toc199063402)

[Table 2.2.9. 5 REQ\_F0905 Table 55](#_Toc199063403)

[Table 2.2.9. 6 F09 View Events Use Case Specification Table 55](#_Toc199063404)

[Table 2.2.10. 1 REQ\_F1001 Table 56](#_Toc199063422)

[Table 2.2.10. 2 REQ\_F1002 Table 56](#_Toc199063423)

[Table 2.2.10. 3 F10 View Registrations & Attendance Status Use Case Specification Table 57](#_Toc199063424)

[Table 2.2.11. 1 REQ\_F1101 Table 58](#_Toc199063433)

[Table 2.2.11. 2 REQ\_F1102 Table 58](#_Toc199063434)

[Table 2.2.11. 3 F11 Scan Attendance QR Code & Update Attendance Status Use Case Specification Table 58](#_Toc199063435)

[Table 2.2.12. 1 REQ\_F1201 Table 59](#_Toc199063463)

[Table 2.2.12. 2 REQ\_F1202 Table 59](#_Toc199063464)

[Table 2.2.12. 3 REQ\_F1203 Table 59](#_Toc199063465)

[Table 2.2.12. 4 F012 Login Use Case Specification Table 60](#_Toc199063466)

[Table 2.4. 1 Performance Requirements Table 63](#_Toc199063491)

[Table 2.5. 1 Security Requirements Table 63](#_Toc199063510)

[Table 2.6. 1 Usability Requirements Table 64](#_Toc199063520)

[Table 2.7. 1 Event Table Data Dictionary 67](#_Toc199063528)

[Table 2.7. 2 Registration Table Data Dictionary 67](#_Toc199063529)

[Table 2.7. 3 Payment Table Data Dictionary 67](#_Toc199063530)

[Table 2.8. 1 Design Constraints Table 68](#_Toc199063556)

[Table 2.9. 1 Software System Attributes Table 69](#_Toc199063564)

[Table 2.10.1. 1 Obtained Requirements From Interview Table 71](#_Toc199063667)

[Table 2.10.2. 1 Obtained Requirements From Questionnaires Table 72](#_Toc199063679)

[Table 2.10.3. 1 Obtained Requirements From Observation Table 73](#_Toc199063690)

# Introduction

## Purpose

The purpose of the project is to develop a digital check-in platform for campus events that seamlessly integrates with the university's student identification database and payment system.  
The system aims to automate and streamline the process of event attendance tracking, ticket verification, and on-site transactions, ensuring an efficient, secure, and user-friendly experience for students and event organizers.

Integration of these components will enable real-time monitoring of attendance records and payments, reducing administrative burdens and enhancing overall event management capabilities within the campus environment.

## Scope

Campus Event Check-in System with Student ID and Payment Integration is created in order to streamline the process of campus event attendance tracking, ticket verification, and payment processing for students and event organizers.

Campus Event Check-in System shall facilitate the following operations:

1. Check-in to campus events using student ID verification.
2. Verify event tickets digitally.
3. Process on-site payments through integrated payment gateways.
4. Provide real-time attendance and transaction records to event organizers.

## Product Perspective

The Campus Event Check-in System is an integrated digital platform that operates as an extension of the university's event management and student information infrastructure. It interacts with the university's student database to authenticate users and with a payment gateway to process event-related transactions. The system supports two main user roles: students and event organizers. Students use the system to browse available events, register, make payments, and check in using QR code. Organizers create events, manage attendee data, and monitor attendance status. **Figure 1.3. 1** shows the context diagram of the system

A diagram of a diagram

AI-generated content may be incorrect.

Figure 1.3. System Context Diagram

### System Interface

The system interface requirements describe how the Campus Event Check-in System interacts with external systems such as the university's student database, payment gateway, event management modules, and notification services **Table 1.3.1. 1**.

Table 1.3.1. System Interfaces Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Priority** | **Author** |
| **REQ\_SI001** | The system shall integrate with the university student database to authenticate student identities and retrieve relevant information during registration and check-in. | High | Azhar |
| **REQ\_SI002** | The system shall connect to a third-party payment gateway to process event payments securely, providing real-time confirmation of transaction status. | High | Ainee |
| **REQ\_SI003** | The system shall interface with device cameras or scanners to scan student IDs or QR codes for efficient check-in and attendance tracking. | Medium | Sulaiman |
| **REQ\_SI004** | The system shall be developed as a mobile application and must be fully compatible with the latest stable versions of **Android** and **iOS** to ensure broad accessibility and usability for mobile users. | High | Yousef |
| **REQ\_SI005** | The system may optionally integrate with an email/SMS service to send notifications such as registration confirmations or event reminders to students. | Low | Yousef |

### User Interface

The user interface (UI) requirements define the visual and interactive characteristics of the Campus Event Check-in System. These specifications aim to ensure consistency, clarity, and accessibility for end users across various devices **Table 1.3.2. 1.**

Table 1.3.2. User Interface Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface ID** | **Description** | **Priority** | **Author** |
| REQ\_UI001 | The GUI for the Campus Event Check-in System will use three background colors: White (RGB Hex: #FFFFFF) as the primary color, Navy Blue (RGB Hex: #001F3F) as the secondary color, and Emerald Green (RGB Hex: #2ECC71) for emphasis elements such as confirmation banners or successful check-in indicators. | High | Yousef |
| REQ\_UI002 | The system will use two main font colors: Black (RGB Hex: #000000) on light backgrounds and White (RGB Hex: #FFFFFF) on dark backgrounds to ensure readability and accessibility for all user types. | High | Yousef |
| REQ\_UI003 | The font family used will be "Poppins" for all headings and titles, and "Roboto" for body text across both desktop and mobile versions of the application, to ensure modern and clean visual presentation. | Medium | Yousef |
| REQ\_UI004 | The minimum base font size will be set to 16pt to ensure clarity on both mobile and desktop devices. Font resizing through accessibility settings will also be supported. | High | Yousef |

### Hardware Interface

The Campus Check-in application will be compatible with any desktop or mobile device with the following specifications **Table 1.3.3. 1**.

Table 1.3.3. Hardware Interface Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface ID** | **Description** | **Priority** | **Author** |
| REQ\_H1001 | The device must be equipped with a mobile-compatible processor (e.g., ARM-based). | High | Azhar |
| REQ\_H1002 | The device must have at least 2GB of RAM. | High | Azhar |
| REQ\_H1003 | The device must include at least 100MB of free internal storage. | High | Azhar |
| REQ\_H1004 | The device must support Wi-Fi or cellular data for internet access. | High | Azhar |
| REQ\_H1005 | The device must include a touch-enabled screen with at least 720p resolution. | High | Azhar |
| REQ\_H1006 | The device must be equipped with a functional rear-facing camera for QR code scanning. | High | Azhar |

### Software Interface

The Campus Check-in application requires other software products to function properly. The interfaces between The Campus Check-in application and other software products are described in **Table 1.3.4. 1**.

Table 1.3.4. Software Interface Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Name** | **Version Number** | **Purpose** | **Reference** |
| **Database** | phpMyAdmin | 5.2.1 | Used to create, store, modify, and delete event, user, and registration data. | phpMyAdmin official page |
| **Operating System** | Microsoft Windows | Windows XP or later | Runs browser-based application for admin and student users. | Chrome Browser system requirement |
| macOS | High Sierra 10.13 or later | Supports browser access on Apple devices. | macOS official site |
| Linux | Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, Fedora 32+ | Runs the system via browser on Linux environments. | Linux distributions’ sites |
| Android | Android 7.0 Nougat or later | Accesses system via mobile browsers. | Android system requirements |
| iOS | iOS 14.0 or later | Accesses system via Safari or Chrome on iPhones/iPads. | iOS system requirements |
| **Browser** | Google Chrome | 113.0.5672.64 | Primary browser interface used to access the system. | Chrome official page |
| Microsoft Edge | 112.0.1722.48 | Alternative browser to access the system. | Microsoft Edge official page |
| Safari | 16.4.1 | For iOS/macOS users to access the system. | Safari official page |
| Opera | 105.0.4970.63 | Optional browser supported for accessing the platform. | Opera official page |
| **Screen Reader** | Speechify | 9.35.0 | Assists visually impaired users by reading on-screen content aloud. | Speechify official page |

### Communication Interface

**Table 1.3.5. 1** shows an overview of the communication interfaces used in the Campus Event Check-in System. It lists each interface or protocol along with its purpose, type of communication, and the users involved.

Table 1.3.5. Communication Interface Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement ID** | **Interface/Protocol** | **Purpose** | **Type** | **Users** |
| **REQ\_CI001** | HTTPS | Ensures secure communication between the mobile app and the backend server. | System-to-User | Students, Event Organizers |
| **REQ\_CI002** | RESTful API (JSON) | Enables interaction between the mobile app and backend for login, registration, and payment. | System-to-System | Mobile App, Backend Server |
| **REQ\_CI003** | Camera API / QR Scanner SDK | Allows mobile devices to access the camera for scanning QR codes during event check-in. | System-to-Device | Students, Event Organizers |
| **REQ\_CI004** | Email API (e.g., SMTP via SendGrid/Firebase) | Sends registration confirmations and event reminders via email. | System-to-User | Students |
| **REQ\_CI005** | TLS/SSL | Encrypts all data transferred between the mobile app and the backend to ensure security. | Security (System-to-User) | All System Users |
| **REQ\_CI006** | TCP/IP over Wi-Fi or Cellular Network | Enables the mobile app to communicate with the backend over the internet. | Network Connectivity | All System Users |

## Product Functions

**Table 1.4. 1** shows an overview of the core features to be implemented in the Campus Event Check-in System. Each feature is identified by a unique ID and includes a brief description along with the user roles that can access it.

Table 1.4. Campus Event Check-in System Function Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature ID** | **Feature** | **Description** | **Accessible Roles** |
| REQ\_F00 | View Upcoming Events | Displays a list of upcoming events open for registration. | Student |
| REQ\_F01 | View Event Detail | Provides detailed information about a selected event. | Student |
| REQ\_F02 | Register for Event | Enables students to register for a selected event. | Student |
| REQ\_F03 | Make Payment | Allows students to pay for event tickets. | Student |
| REQ\_F04 | Receive e-Ticket/QR Code | Generates a QR code for the student upon successful registration and payment. | Student (System) |
| REQ\_F05 | View Registration History | Shows a history of past registered events. | Student |
| REQ\_F06 | Show QR Attendance for Event | Displays the QR code used for scanning during check-in. | Student |
| REQ\_F07 | Create New Event | Allows organizers to create a new event. | Event Organizer |
| REQ\_F08 | Set Event Detail | Enables organizers to enter event name, date, time, and location. | Event Organizer |
| REQ\_F09 | View Events | Displays a list of events created by the organizer. | Event Organizer |
| REQ\_F10 | View Registrations with Attendance Status | Allows organizers to view who has registered and their attendance status. | Event Organizer |
| REQ\_F11 | Check-in Upon Arrival | Enables organizers to scan QR codes or student IDs to verify attendance. | Event Organizer |
| REQ\_F12 | Login | Allows users to securely log into the system. | Student, Event Organizer |

In terms of diagrams, the use case diagram in **Figure 1.3.6. 1** shows the product functions but in visual format following UML Use Case Diagram notation.

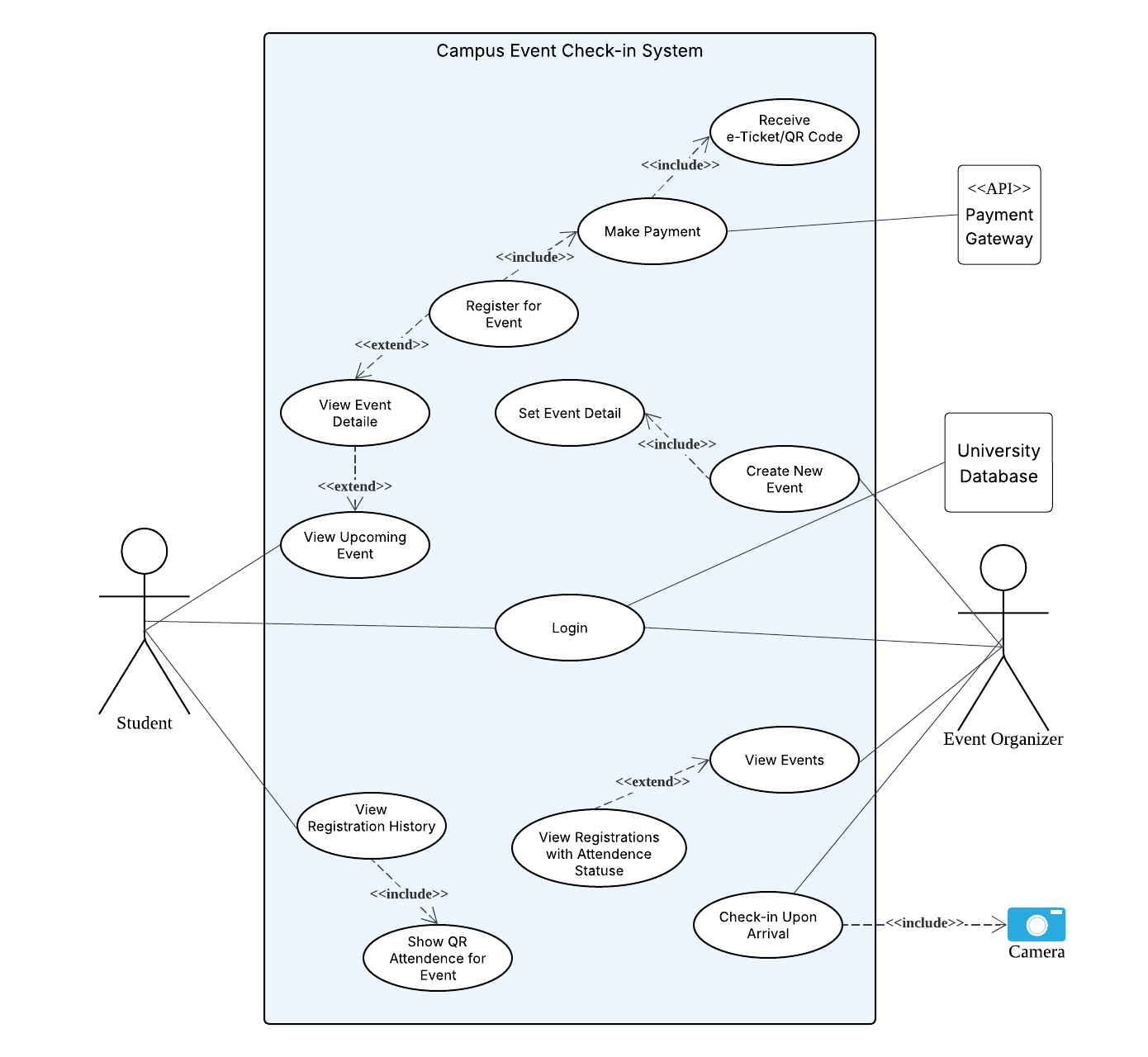


Figure 1.4. Campus Event Check-in System Use Case Diagram

## ****User Characteristics****

This section describes the end users of the Campus Event Check-in System and their expected level of knowledge. These characteristics help determine how the system should be designed for usability and accessibility. **Table 1.5. 1** summarizes the intended users and their required understanding of the system.

Table 1.5. Intended Users and Expected Knowledge

|  |  |  |
| --- | --- | --- |
| **Role** | **Description** | **Expected Knowledge** |
| **Student** | University students who will use the system to view events, register, make payments, and check in. | Basic understanding of web browsing, online form usage, and scanning QR codes. |
| **Event Organizer** | Authorized university staff or student committees managing events, registrations, and check-ins. | Familiarity with web forms, participant tracking, and scanning devices for check-in. |

## Limitations

Several limitations may affect the design, development, and usage of the system. These limitations must be considered to ensure that the system remains secure, scalable, and compliant.

* The system must comply with regional data protection laws to protect student privacy.
* Compatibility with assistive technologies (such as screen readers) is required for accessibility.
* Support for multiple languages may be needed in the future to cater to a diverse user base.
* The system must include secure login procedures and encryption for data in transit and at rest.
* User authentication must be robust to prevent unauthorized access.
* Measures must be in place to ensure data integrity and prevent corruption or loss.
* The system should be scalable to support growing numbers of users and data without performance loss.
* Key features must remain responsive, especially for time-sensitive operations like check-in.
* Regular system maintenance is required to address bugs, vulnerabilities, and browser compatibility.
* Compatibility issues may occur when integrated platforms are updated.
* Training resources and user documentation should be available to support users with limited technical expertise.

## Assumptions and Dependencies

The development and deployment of the system are based on the following assumptions and dependencies:

* The university will provide access to its student information database for identity verification.
* A secure and reliable internet connection is available at event locations.
* Third-party services like payment gateway APIs will remain stable and available.
* Users (students and organizers) will have access to compatible browsers and devices.
* The university will assign responsible personnel for ongoing maintenance and updates.
* External systems used for integration (student database, payment gateway) must maintain backward compatibility.

## Definitions

**Table 1.8. 1** describes are terminology, phrases, and words utilized in the document and their corresponding definitions as employed in this document.

Table 1.8. Glossary of Terms for the Campus Event Check-in System

|  |  |
| --- | --- |
| **Terms, Acronyms and Word** | **Definition** |
| Campus Event Check-in System | The proposed web-based application to manage student event participation, registration, and attendance. |
| Student | A university-registered individual who can register and check in to events. |
| Event Organizer | A staff or authorized user responsible for creating events and managing check-ins. |
| QR Code | A unique code generated after registration used to check in at the event venue. |
| Payment Gateway | A third-party service used to process online event payments securely. |
| Check-in | The process of confirming a student’s attendance using their ID or QR code. |
| REST API | An interface using HTTP methods for client-server communication in the system. |

# Requirements

Requirements are the functionalities, behaviors, qualities, and constraints that the final system must fulfill. This chapter outlines the functional and non-functional requirements necessary for the successful implementation of the Campus Event Check-in System. Each requirement is described in detail, and diagrams are provided when necessary to illustrate system behavior, user interactions, and data flows. This section also includes any supporting information that aids in understanding or validating the system’s expected operations and performance.

## External Interfaces

This section details the external interface requirements of the Campus Event Check-in System. It outlines the actions available in the user interfaces and specifies the expected inputs and outputs for each interaction. These interfaces are designed to ensure a smooth and intuitive experience for users while supporting secure and efficient communication with external systems such as student databases, payment gateways, and scanning devices. The interfaces described here serve as the bridge between users and system functionalities.

### Login User Interface

Since both the Student and the Event Organizer share the same login interface in the Campus Event Check-in System, the following tables **Table 2.1.1. 1 - Table 2.1.1. 4** describe the common interface elements used for authentication, applicable to both user types.

Table 2.1.1. REQ\_IO0001 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0001 | **Version** | 1.0 |
| **Item** | Login Button (Input) | | |
| **Item Description** | A button labelled “Login.” | | |
| **Item Purpose** | Initiates the submission of login credentials for authentication. | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0002, REQ\_IO0003 (Both fields must be filled for submission) | | |
| **Author** | Sulaiman | | |

Table 2.1.1. REQ\_IO0002 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0002 | **Version** | 1.0 |
| **Item** | Student ID Field (Input) | | |
| **Item Description** | A text field labelled “Student ID.” | | |
| **Item Purpose** | Allows the user to input their Student ID for submission. | | |
| **Input Format** | String | **Valid Input** | ASCII code from decimal 32 to 126 |
| **Related I/O** | None | | |
| **Author** | Sulaiman | | |

Table 2.1.1. REQ\_IO0003 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0003 | **Version** | 1.0 |
| **Item** | Password Field (Input) | | |
| **Item Description** | A text field labelled “Password.” | | |
| **Item Purpose** | Allows the user to input their password for submission. | | |
| **Input Format** | String | **Valid** **Input** | ASCII code from decimal 32 to 126 |
| **Related I/O** | None | | |
| **Author** | Sulaiman | | |

Table 2.1.1. REQ\_IO0004 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0004 | **Version** | **1.0** |
| **Item** | Failure Message (Output) | | |
| **Item Description** | A toast message displaying a failure message. | | |
| **Item Purpose** | To notify the end user that the login was unsuccessful. | | |
| **Input Format** | Not Applicable | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0001 (Only displayed after submitting input) | | |
| **Author** | Sulaiman | | |

### View Upcoming Events Interface

Since the feature to view available events is exclusive to the Student user in the Campus Event Check-in System, **Table 2.1.2. 1** describe the first version of the interface elements that enable students to access, browse, and explore the list of upcoming campus events. However, **Table 2.1.2. 2** shows the updated version of the table.

Table 2.1.2. REQ\_IO0101 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0101 | **Version** | **1.0** |
| **Item** | Event blocks (Input) | | |
| **Item Description** | A rectangle block with image and title of the event | | |
| **Item Purpose** | Have a quick visual look for available events | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | None | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.2. REQ\_IO0101 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0101 | **Version** | **2.0** |
| **Item** | Event blocks (Input) | | |
| **Item Description** | A rectangle block with the image and title of the event | | |
| **Item Purpose** | Have a quick visual look for upcoming available events | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | None | | |
| **Author (Updated By)** | Sulaiman | | |

### Event details

Once a student selects an event from the available list, the system displays detailed information about that specific event. The following tables **Table 2.1.3. 1** – **Table 2.1.3. 8** define the interface components used to present event details, enabling students to make informed decisions before registration or attendance.

Table 2.1.3. REQ\_IO0201 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0201 | **Version** | **1.0** |
| **Item** | Event’s image (Output) | | |
| **Item Description** | A visual representation of the event | | |
| **Item Purpose** | Let user easily recognize the event | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0101 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.3. REQ\_IO0202 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0202 | **Version** | **1.0** |
| **Item** | Event title (Output) | | |
| **Item Description** | A text field with event title | | |
| **Item Purpose** | Let user knows the event official name | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0101 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.3. REQ\_IO0203 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0203 | **Version** | **1.0** |
| **Item** | Event date (Output) | | |
| **Item Description** | A text field with starting date and time and ending date and time | | |
| **Item Purpose** | To let user know when is the event happening and make a schedule | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0101 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.3. REQ\_IO0204 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0204 | **Version** | **1.0** |
| **Item** | Event location (Output) | | |
| **Item Description** | A text field that shows where the event held and its address | | |
| **Item Purpose** | To let user know where to go for the event | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0101 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.3. REQ\_IO0205 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0205 | **Version** | **1.0** |
| **Item** | Event ticket price (Output) | | |
| **Item Description** | A text field that shows the price of ticket to enter the event | | |
| **Item Purpose** | To let user knows the cost required to attend the event | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0101 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.3. REQ\_IO0206Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0206 | **Version** | **1.0** |
| **Item** | Event capacity (output) | | |
| **Item Description** | A text field that shows how many slots for participants | | |
| **Item Purpose** | To let user know how limited slots of the events | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0101 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.3. REQ\_IO0207 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0207 | **Version** | **1.0** |
| **Item** | Event’s remark (Output) | | |
| **Item Description** | A text field that contains other detail about the event | | |
| **Item Purpose** | To let user knows the any additional information about the event such as stuff to bring, attire and others | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0101 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.3. REQ\_IO0208 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0208 | **Version** | **1.0** |
| **Item** | Register event (Input) | | |
| **Item Description** | A button with a text of “Register Event” on it | | |
| **Item Purpose** | Allow user to register in event | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0301 | | |
| **Author** | Lim Ai Nee | | |

### Register for Event

The registration functionality allows students to confirm their participation in a selected campus event. The following tables **Table 2.1.4. 1** – **Table 2.1.4. 5** outline the user interface elements that facilitate the event registration process, including input fields and confirmation messages relevant to successful enrollment. However, **Table 2.1.4. 6** and **Table 2.1.4. 7** show the new version of REQ\_IO0301 and REQ\_IO0302.

Table 2.1.4. REQ\_IO0301 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0301 | **Version** | **1.0** |
| **Item** | Name field (Input) | | |
| **Item Description** | A text field with user’s account name | | |
| **Item Purpose** | User can choose to re-enter the text field to desire input | | |
| **Input Format** | String | **Valid Input** | “{name} |
| **Related I/O** | REQ\_IO0208 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.4. REQ\_IO0302 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0302 | **Version** | **1.0** |
| **Item** | Student ID field (Input) | | |
| **Item Description** | A text field with user’s account student ID | | |
| **Item Purpose** | User can choose to re-enter the text field to desire input | | |
| **Input Format** | String | **Valid Input** | ASCII code from decimal 48 to 57 and 65 to 90 |
| **Related I/O** | REQ\_IO0208 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.4. REQ\_IO0303 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0303 | **Version** | **1.0** |
| **Item** | Mobile number field (Input) | | |
| **Item Description** | A text field labelled with “mobile number” | | |
| **Item Purpose** | To let organizer to contact the participant | | |
| **Input Format** | String | **Valid Input** | “{area}-{subscriber} |
| **Related I/O** | REQ\_IO0208 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.4. REQ\_IO0304 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0304 | **Version** | **1.0** |
| **Item** | Email address field (Input) | | |
| **Item Description** | A text field labelled with “email” | | |
| **Item Purpose** | To let organizer inform participant any event details and information | | |
| **Input Format** | String | **Valid Input** | “{name}@{domain} |
| **Related I/O** | REQ\_IO0208 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.4. REQ\_IO0305 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0305 | **Version** | **1.0** |
| **Item** | Ticket payment (Input) | | |
| **Item Description** | A button labelled with “Pay for Ticket” | | |
| **Item Purpose** | To let user proceed to payment process | | |
| **Input Format** | Button | **Valid Input** | Not applicable |
| **Related I/O** | REQ\_IO0208 is pressed | | |
| **Author** | Lim Ai Nee | | |

Table 2.1.4. REQ\_IO0301 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0301 | **Version** | **2.0** |
| **Item** | Name field (Output) | | |
| **Item Description** | A read only field contains the name of student retrieved from the university’s database. | | |
| **Item Purpose** | Display the student’s name | | |
| **Input Format** | None | **Valid Input** | “{name} |
| **Related I/O** | REQ\_IO0208 is pressed | | |
| **Author (Updated By)** | Sulaiman | | |

Table 2.1.4. REQ\_IO0302 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0302 | **Version** | **2.0** |
| **Item** | Student ID field (Output) | | |
| **Item Description** | A read only field contains the ID of student retrieved from the university’s database. | | |
| **Item Purpose** | Display the student’s ID | | |
| **Input Format** | None | **Valid Input** | ASCII code from decimal 48 to 57 and 65 to 90 |
| **Related I/O** | REQ\_IO0208 is pressed | | |
| **Author (Updated By)** | Sulaiman | | |

### Make Payment

For events that require a fee, the system provides an integrated payment interface to ensure a seamless transaction process. The following tables **Table 2.1.5. 1** – **Table 2.1.5. 4** specify the user interface components involved in the payment workflow, supporting secure input of payment details and confirmation of successful transactions.

Table 2.1.5. REQ\_IO0401 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0401 | **Version** | **1.0** |
| **Item** | Pay button (Input) | | |
| **Item Description** | A button with the text “Pay” | | |
| **Item Purpose** | To initiate submit and confirm payment information | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0402, REQ\_IO0403, REQ\_IO0404 , REQ\_IO0405 | | |
| **Author** | Azhar | | |

Table 2.1.5. REQ\_IO0402 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0402 | **Version** | **1.0** |
| **Item** | Credit Card Number Field (Input) | | |
| **Item Description** | Input field labelled “Credit Card Number” | | |
| **Item Purpose** | To input the credit card number | | |
| **Input Format** | String | **Valid Input** | ASCII Code 48 to 57 |
| **Related I/O** | REQ\_IO10401 | | |
| **Author** | Azhar | | |

Table 2.1.5. REQ\_IO0403 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0403 | **Version** | **1.0** |
| **Item** | Expiry Date Field (Input) | | |
| **Item Description** | Input field labelled “Expiry Date” | | |
| **Item Purpose** | To input the Expiry Date of the Credit Card | | |
| **Input Format** | String | **Valid Input** | Date in the format MM/YY |
| **Related I/O** | REQ\_IO10401 | | |
| **Author** | Azhar | | |

Table 2.1.5. REQ\_IO0404 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0404 | **Version** | **1.0** |
| **Item** | CVV/CVC Security Code Field (Input) | | |
| **Item Description** | Input field labelled “CVV/CVC” | | |
| **Item Purpose** | To input the CVV/CVC security code of the Credit Card | | |
| **Input Format** | Integer | **Valid Input** | Integer of length 3 |
| **Related I/O** | REQ\_IO10401 | | |
| **Author** | Azhar | | |

### Receive e-ticket/QR code

Upon successful registration and payment, the system generates a digital ticket in the form of a QR code. The following tables **Table 2.1.6. 1** – **Table 2.1.6. 3** describe the interface components responsible for delivering the e-ticket to the student, facilitating a secure and efficient check-in process at the event venue.

Table 2.1.6. REQ\_IO0501 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0501 | **Version** | **1.0** |
| **Item** | Event Name Label (Output) | | |
| **Item Description** | Label displaying the name of the event | | |
| **Item Purpose** | To display the name of the event for verification by user and event staff | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO604 (Selected an event from registration history) | | |
| **Author** | Azhar | | |

Table 2.1.6. REQ\_IO0502 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0502 | **Version** | **1.0** |
| **Item** | Details of events (Output) | | |
| **Item Description** | List of labels displaying the details of event, such as date and a brief description | | |
| **Item Purpose** | Allows user view details of past events | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQIO602 | | |
| **Author** | Azhar | | |

Table 2.1.6. REQ\_IO0503 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0503 | **Version** | **1.0** |
| **Item** | QR Code (Output) | | |
| **Item Description** | A QR Code generated specific to the user and event | | |
| **Item Purpose** | To serve as a ticket and for event staff to verify attendance | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | None | | |
| **Author** | Azhar | | |

### View Registration History

The system provides students with access to their past event registrations through a dedicated history interface. The following tables **Table 2.1.7. 1** – **Table 2.1.7. 4** define the interface elements that allow students to view, track, and verify their previous event participation records in an organized manner.

Table 2.1.7. REQ\_IO0601 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0601 | **Version** | **1.0** |
| **Item** | View Event History (Input) | | |
| **Item Description** | A button with the label “View Event History” | | |
| **Item Purpose** | Allows user to view past events that they have registered for | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | None | | |
| **Author** | Azhar | | |

Table 2.1.7. REQ\_IO0602 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0602 | **Version** | **1.0** |
| **Item** | Events (Output) | | |
| **Item Description** | List of labels displaying names of past events | | |
| **Item Purpose** | To display the event history of the user | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQIO603, REQIO604 | | |
| **Author** | Azhar | | |

Table 2.1.7. REQ\_IO0603 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0603 | **Version** | **1.0** |
| **Item** | View Details (Input) | | |
| **Item Description** | Button next to each “Event” label | | |
| **Item Purpose** | Allows users to view details of past events | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | REQIO602, REQIO 605 | | |
| **Author** | Azhar | | |

Table 2.1.7. REQ\_IO0604 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0604 | **Version** | **1.0** |
| **Item** | Generate QR Code (Input) | | |
| **Item Description** | A button with the label “Generate QR Code” | | |
| **Item Purpose** | Allows user to generate QR code to verify ticket and attendance | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | REQIO501 (goes to View e-Receipt/QR code) | | |
| **Author** | Azhar | | |

### Show QR Attendance

To support on-site verification, the system allows event organizers to scan and display student attendance using QR codes. The following tables **Table 2.1.8. 1** – **Table 2.1.8. 6** detail the initial and the updated versions for the interface elements involved in scanning, validating, and presenting QR-based attendance records during event check-in.

Table 2.1.8. REQ\_IO0701 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0701 | **Version** | **1.0** |
| **Item** | QR Code (Output) | | |
| **Item Description** | A unique QR code generated for the registered event | | |
| **Item Purpose** | Allows the event staff to scan and verify attendance | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | Triggered after successful registration | | |
| **Author** | Yousef | | |

Table 2.1.8. REQ\_IO0701 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0701 | **Version** | **2.0** |
| **Item** | QR Code (Output) | | |
| **Item Description** | A unique QR code generated for the registered event | | |
| **Item Purpose** | Allows the event staff to scan and verify attendance | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | Triggered after REQ\_IO0604 | | |
| **Author (Updated By)** | Sulaiman | | |

Table 2.1.8. REQ\_IO0702 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0702 | **Version** | **1.0** |
| **Item** | Event Name Label (Output) | | |
| **Item Description** | Displays the title of the event below the QR code | | |
| **Item Purpose** | Helps student identify which event the QR code belongs to | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | Triggered after successful registration | | |
| **Author** | Yousef | | |

Table 2.1.8. REQ\_IO0702 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0702 | **Version** | **2.0** |
| **Item** | QR Code (Output) | | |
| **Item Description** | A unique QR code generated for the registered event | | |
| **Item Purpose** | Allow the event staff to scan and verify attendance | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | None | | |
| **Author (Updated By)** | Sulaiman | | |

Table 2.1.8. REQ\_IO0703 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0703 | **Version** | **1.0** |
| **Item** | Back Button (Output) | | |
| **Item Description** | A button labeled “Back” to return to the dashboard | | |
| **Item Purpose** | Allows user to exit the QR screen | | |
| **Input Format** | Button | **Valid Input** | Button Click |
| **Related I/O** | Linked to user dashboard | | |
| **Author** | Yousef | | |

Table 2.1.8. REQ\_IO0703 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0703 | **Version** | **2.0** |
| **Item** | Back Button (Output) | | |
| **Item Description** | A button labeled “Back” to return to the dashboard | | |
| **Item Purpose** | Allows user to exit the QR screen | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | None | | |
| **Author (Updated By)** | Sulaiman | | |

### Set Event Details

Event organizers are responsible for defining the details of each event they create within the system. The following tables **Table 2.1.9. 1** – **Table 2.1.9. 8** outline the interface components that enable organizers to input and manage event-related information, such as title, description, date, time, venue, and capacity.

Table 2.1.9. REQ\_IO0801 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0801 | **Version** | **1.0** |
| **Item** | Event Title Field (Input) | | |
| **Item Description** | A text field labeled “Event Title” | | |
| **Item Purpose** | Allows organizer to name the event | | |
| **Input Format** | String | **Valid Input** | Min. 3 characters |
| **Related I/O** | Linked to create event process | | |
| **Author** | Yousef | | |

Table 2.1.9. REQ\_IO0802 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0802 | **Version** | **1.0** |
| **Item** | Event Date Picker (Input) | | |
| **Item Description** | A calendar picker for choosing the event date | | |
| **Item Purpose** | Ensures valid and future event date is selected | | |
| **Input Format** | Date | **Valid Input** | Format: YYYY-MM-DD |
| **Related I/O** | REQ\_IO0808 (submit button) | | |
| **Author** | Yousef | | |

Table 2.1.9. REQ\_IO0803 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0803 | **Version** | **1.0** |
| **Item** | Event Time Field (Input) | | |
| **Item Description** | Input field for event start and end time | | |
| **Item Purpose** | Specifies when the event begins and ends | | |
| **Input Format** | Time | **Valid Input** | HH:MM (24-hour format) |
| **Related I/O** | REQ\_IO0808 | | |
| **Author** | Yousef | | |

Table 2.1.9. REQ\_IO0804 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0804 | **Version** | **1.0** |
| **Item** | Location Field (Input) | | |
| **Item Description** | A text field for entering event location | | |
| **Item Purpose** | Specifies where the event will be held | | |
| **Input Format** | String | **Valid Input** | Valid campus address |
| **Related I/O** | REQ\_IO0808 | | |
| **Author** | Yousef | | |

Table 2.1.9. REQ\_IO0805 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0805 | **Version** | **1.0** |
| **Item** | Capacity Field (Input) | | |
| **Item Description** | Field to define number of participant slots | | |
| **Item Purpose** | Limits how many users can register | | |
| **Input Format** | Integer | **Valid Input** | 1 to 5000 |
| **Related I/O** | REQ\_IO0808 | | |
| **Author** | Yousef | | |

Table 2.1.9. REQ\_IO0806 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0806 | **Version** | **1.0** |
| **Item** | Ticket Price Field (Input) | | |
| **Item Description** | Field to input the price of the event ticket | | |
| **Item Purpose** | Determines event cost for participants | | |
| **Input Format** | Float | **Valid Input** | e.g., 0.00 to 999.99 |
| **Related I/O** | REQ\_IO0808 | | |
| **Author** | Yousef | | |

Table 2.1.9. REQ\_IO0807 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0807 | **Version** | **1.0** |
| **Item** | Description Field (Input) | | |
| **Item Description** | Optional text field for additional notes | | |
| **Item Purpose** | Adds special instructions (e.g., dress code) | | |
| **Input Format** | String | **Valid Input** | Max. 255 characters |
| **Related I/O** | REQ\_IO0808 | | |
| **Author** | Yousef | | |

Table 2.1.9. REQ\_IO0808 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0808 | **Version** | **1.0** |
| **Item** | Create Event Button (Input) | | |
| **Item Description** | A button labeled “Create Event” | | |
| **Item Purpose** | Finalizes and submits event details | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | Submits data from REQ\_IO0801-0807 | | |
| **Author** | Yousef | | |

### View Events (Organizer)

To facilitate event management, the system provides organizers with an interface to view a list of events they have created. The following tables **Table 2.1.10. 1** – **Table 2.1.10. 4** define the interface elements that support the organizer in monitoring, updating, or managing event entries efficiently.

Table 2.1.10. REQ\_IO0901 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0901 | **Version** | **1.0** |
| **Item** | Events List (Output) | | |
| **Item Description** | A scrollable list or grid showing all created events | | |
| **Item Purpose** | Allows organizer to view and manage existing events | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0902-0904 | | |
| **Author** | Yousef | | |

Table 2.1.10. REQ\_IO0902 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0902 | **Version** | **1.0** |
| **Item** | Event Status Indicator (Output) | | |
| **Item Description** | Badge showing event status (e.g., "Upcoming", "Completed") | | |
| **Item Purpose** | Helps organizer identify ongoing or past events | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0901 | | |
| **Author** | Yousef | | |

Table 2.1.10. REQ\_IO0903 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0903 | **Version** | **1.0** |
| **Item** | Edit Button (Input) | | |
| **Item Description** | Button labeled “Edit” for each listed event | | |
| **Item Purpose** | Allows organizer to make changes to event details | | |
| **Input Format** | Button | **Valid Input** | Not Applicable |
| **Related I/O** | Links to Set Event Details form | | |
| **Author** | Yousef | | |

Table 2.1.10. REQ\_IO0904 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO0901 | **Version** | **1.0** |
| **Item** | Delete Button (Input) | | |
| **Item Description** | A button labeled “Delete” shown for each event | | |
| **Item Purpose** | Lets organizer remove an event permanently | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO0901 | | |
| **Author** | Yousef | | |

### View registrations and attendance status

To support effective event oversight, the system offers event organizers an interface to monitor participant registrations and their corresponding attendance status. The following tables **Table 2.1.11. 1** – **Table 2.1.11. 3** define the user interface components that facilitate filtering, tracking, and exporting attendance-related data for each event.

Table 2.1.11. REQ\_IO1001 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO1001 | **Version** | **1.0** |
| **Item** | Registrations List (Output) | | |
| **Item Description** | A table displaying a list of students registered for a selected event | | |
| **Item Purpose** | Enables the organizer to view all registered students for a particular event | | |
| **Input Format** | None | **Valid Input** | Not Applicable |
| **Related I/O** | REQ\_IO1005 (Event Selector) | | |
| **Author** | Sulaiman | | |

Table 2.1.11. REQ\_IO1002 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO1002 | **Version** | **1.0** |
| **Item** | Attendance Status Indicator (Output) | | |
| **Item Description** | A status label showing whether the student has checked in | | |
| **Item Purpose** | Helps organizers monitor and verify attendance | | |
| **Input Format** | None | **Valid Input** | “Checked In” / “Not Checked In” |
| **Related I/O** | REQ\_IO1001 | | |
| **Author** | Sulaiman | | |

Table 2.1.11. REQ\_IO1003 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_IO1003 | **Version** | **1.0** |
| **Item** | Event Selector Dropdown (Input) | | |
| **Item Description** | A dropdown menu listing all events created by the organizer | | |
| **Item Purpose** | Lets the organizer select an event to view its registrations | | |
| **Input Format** | Dropdown | **Valid Input** | Selected Event ID |
| **Related I/O** | REQ\_IO1001 | | |
| **Author** | Sulaiman | | |

## Functions

This section outlines the functional requirements that the system is expected to fulfill upon completion. Each requirement will be explained in detail and illustrated with a sequence diagram to enhance understanding.

### F00 View Upcoming Events

**Table 2.2.1. 1** describes the view upcoming events function and its description. While **Table 2.2.1. 2** shows the view upcoming events use case specification table. And **Figure 2.2.1. 1** depicts the sequence diagram for the view upcoming events function.

Table 2.2.1. REQ\_F0001 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0001 | **Version** | **1.0** |
| **Item** | When user clicks on the button, the system shall display the relevant event’s details | | |
| **Item Description** | Lim Ai Nee | | |

Table 2.2.1. F00 View Available Events Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F00 View Upcoming Events | **Version** | 1.0 |
| **Purpose** | To allow user to have a quick glance and check out for upcoming and happening events | | |
| **Actor** | Student | | |
| **Precondition** | An event is created in the system | | |
| **Postcondition** | The details of selected event will be displayed | | |
| **Main Flow** | 1. The System display a list of events 2. The Student scrolls to view multiple events | | |
| **Alternate Scenario** | None | | |
| **Author** | Lim Ai Nee | | |

A black background with white text

AI-generated content may be incorrect.

Figure 2.2.1. F00 View Upcoming Events Sequence Diagram

### F01 View Event Details

**Table 2.2.2. 1** & **Table 2.2.2. 2** describe the view event details function and its description. While **Table 2.2.2. 3** shows the view event details use case specification table. And **Figure 2.2.2. 1** depicts the sequence diagram for the view event details function.

Table 2.2.2. REQ\_F0101 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0101 | **Version** | **1.0** |
| **Item** | The system shall display all the event details entered by the event organizer | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.2. REQ\_F0102 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0102 | **Version** | **1.0** |
| **Item** | When user clicks on the “Register” button, the system shall display the registration page that requires user to fill up all the fields | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.2. F01 View Event Details Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F01 View Event Details | **Version** | 1.0 |
| **Purpose** | To allow the user to view the event details like location, time, ticket price and others and to allow the user to register in the event | | |
| **Actor** | Student | | |
| **Precondition** | An event is selected | | |
| **Postcondition** | User will need to provide necessary information to register for attending the event | | |
| **Main Flow** | 1. The student selects an event to view its details from the event list 2. The system will display relevant event details 3. The student can choose to click on the “Register” button to register for the event | | |
| **Alternate Scenario** | If the Student does not want to register for the event, they can choose to click on the “<” button to redirect to the view upcoming events page | | |
| **Author** | Lim Ai Nee | | |

A diagram of a system

AI-generated content may be incorrect.

Figure 2.2.2. F01 View Event Details Sequence Diagram

### F02 Register For Event

**Table 2.2.3. 1** – **Table 2.2.3. 9** describe the register for event function and its description. While **Table 2.2.3. 10** shows the first version of the register for event use case specification table, while **Table 2.2.3. 11** shows the second version of the register for event use case specification table. And **Figure 2.2.3. 1** depicts the sequence diagram for the register for event function.

Table 2.2.3. REQ\_F0201 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0201 | **Version** | **1.0** |
| **Item** | The system shall retrieve from the user account to get their name and display on this field and it is editable | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.3. REQ\_F0202 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0202 | **Version** | **1.0** |
| **Item** | The system shall retrieve from the user account to get their student ID and display on this field and it is editable | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.3. REQ\_F0203 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0203 | **Version** | **1.0** |
| **Item** | The system shall display this field with the text of “Mobile Number” and shall let user to enter their mobile number | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.3. REQ\_F0204 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0204 | **Version** | **1.0** |
| **Item** | The system shall display this field with the text of “Email” and shall let user to enter their email | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.3. REQ\_F0205 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0205 | **Version** | **1.0** |
| **Item** | The system shall direct user to the page to make payment and save the data into a temporary database | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.3. REQ\_F0206 Version 1 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0206 | **Version** | **1.0** |
| **Item** | When user clicks on the button, the system shall display the view event detail page | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.3. REQ\_F0201 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0201 | **Version** | **2.0** |
| **Item** | The system shall retrieve the student data from the university’s database and auto fill in the name and the ID field and give the student the ability to update the phone number and the personal email address. | | |
| **Author (Updated By)** | Sulaiman | | |

Table 2.2.3. REQ\_F0205 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0205 | **Version** | **2.0** |
| **Item** | The system shall direct the student to the payment page | | |
| **Author (Updated By)** | Sulaiman | | |

Table 2.2.3. REQ\_F0206 Version 2 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0206 | **Version** | **2.0** |
| **Item** | When user clicks on the “Back” button, the system shall display the event detail page | | |
| **Author (Updated By)** | Sulaiman | | |

Table 2.2.3. F02 Register for Event Use Case Specification Table Version 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F02 Register for Event | **Version** | 1.0 |
| **Purpose** | Let the user register and book a slot for the event they wanted to participate in | | |
| **Actor** | Student | | |
| **Precondition** | The “Register” button is clicked by the user | | |
| **Postcondition** | The user shall proceed to make payment for the event ticket after clicked on the button of “Pay for Ticket” | | |
| **Main Flow** | 1. The Student had click on “Register” button 2. The System shall display text fields for Student to filled up necessary information 3. The Student can choose to click on the “Pay for Ticket” button to proceed make payment for the ticket | | |
| **Alternate Scenario** | If the Student decide not to pay for the event, they can choose to click on the “<” button to redirect to the view event details page | | |
| **Author** | Lim Ai Nee | | |

Table 2.2.3. F02 Register for Event Use Case Specification Table Version 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F02 Register for Event | **Version** | 2.0 |
| **Purpose** | Allow the user to register and book a slot for the event they want to attend. | | |
| **Actor** | Student | | |
| **Precondition** | The “Register” button is clicked by the user | | |
| **Postcondition** | The user shall proceed to make payment for the event ticket after clicked on the button of “Pay for Ticket” | | |
| **Main Flow** | 1. The student had clicked on “Register” button 2. The system shall display the input fields 3. The student fills the input fields 4. The student clicks on the “Pay for Ticket” button 5. The system displays the payment page | | |
| **Alternate Scenario** | If the Student decide not to pay for the event, they can choose to click on the “< Back” button to back again to the event details page | | |
| **Author (Updated By)** | Sulaiman | | |

A diagram of a system

AI-generated content may be incorrect.

Figure 2.2.3. F02 Register for Event Sequence Diagram

### F03 Make Payment

**Table 2.2.4. 1** – **Table 2.2.4. 3** describe the make payment function and its description. While **Table 2.2.4. 4** shows the first version of the make payment use case specification table, while **Table 2.2.4. 5** shows the second version of the make payment use case specification table. And **Figure 2.2.4. 1** depicts the first version of the sequence diagram for the make payment function, while **Figure 2.2.4. 2** shows the second version of the sequence diagram for the make payment function.

Table 2.2.4. REQ\_F0301 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0301 | **Version** | **1.0** |
| **Item** | When the user clicks the “Pay” button, the system shall verify the inputs | | |
| **Author** | Azhar | | |

Table 2.2.4. REQ\_F0302 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0302 | **Version** | **1.0** |
| **Item** | If any of the inputs are invalid, the system shall display an error message above the invalid input | | |
| **Author** | Azhar | | |

Table 2.2.4. REQ\_F0303 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0303 | **Version** | **1.0** |
| **Item** | If all inputs are valid, the system shall proceed with the payment and save the ticket into the database | | |
| **Author** | Azhar | | |

Table 2.2.4. F03 Make Payment Use Case Specification Table Version 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F03 Make Payment | **Version** | 1.0 |
| **Purpose** | To allow users to pay for a ticket | | |
| **Actor** | Student | | |
| **Precondition** | Student has selected an event to purchase a ticket for | | |
| **Postcondition** | Student acquires a ticket for the event which is saved into the database | | |
| **Main Flow** | 1. The student shall select an event to purchase a ticket 2. The system shall redirect the student to the payment screen 3. The system shall display the input forms prompting the user to input payment details such as credit card number, CCV/CVC security code and expiry date 4. The student shall input the required details into the input forms 5. The student shall click the pay button 6. The system shall verify all inputs 7. The system shall display success message and save ticket into the database | | |
| **Alternate Scenario** | If any of the inputs are invalid, the system shall display an error message above the invalid input | | |
| **Author** | Azhar | | |

Table 2.2.4. F03 Make Payment Use Case Specification Table Version 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F03 Make Payment | **Version** | 2.0 |
| **Purpose** | To allow users to pay for a ticket | | |
| **Actor** | Student | | |
| **Precondition** | The student has selected an event to purchase a ticket for | | |
| **Postcondition** | The student acquires a ticket for the event which is saved into the database | | |
| **Main Flow** | 1. The system displays the payment page. 2. The student fills in the inputs in the payment form. 3. The system shall validate all inputs. 4. The system shall communicate with the payment gateway. 5. The system shall display success message and save ticket’s into the database if the payment gateway returned success payment status | | |
| **Alternate Scenario** | If any of the inputs are invalid, the system shall display an error message above the invalid input | | |
| **Author (Updated By)** | Sulaiman | | |

A diagram of a system

AI-generated content may be incorrect.

Figure 2.2.4. F03 Make Payment Sequence Diagram Version 1

A black screen with white text

AI-generated content may be incorrect.

Figure 2.2.4. F03 Make Payment Sequence Diagram Version 2

### F04 Receive e-Ticket/QR Code

**Table 2.2.5. 1** describes the receive e-ticket/QR code function and its description. While **Table 2.2.5. 2** shows the receive e-ticket/QR code use case specification table. And **Figure 2.2.5. 1** depicts the sequence diagram for the receive e-ticket/QR code function.

Table 2.2.5. REQ\_F0401 Tale

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0401 | **Version** | **1.0** |
| **Item** | When the user navigates to the show QR screen, the system shall display the name of the event and generate a QR code specific to the user and event | | |
| **Author** | Azhar | | |

Table 2.2.5. F04 Receive e-Ticket/QR Code Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F04 Receive e-Ticket/QR Code | **Version** | 1.0 |
| **Purpose** | To allow a user to view a QR-code to serve as a ticket for event staff to verify attendance | | |
| **Actor** | Student | | |
| **Precondition** | The student has purchased a ticket for an event. | | |
| **Postcondition** | None | | |
| **Main Flow** | 1. The user shall navigate to the generate QR Code screen for the chosen event. 2. The system shall retrieve the ticket from the database. 3. The system shall display the name of the event and a QR code encoded with the student’s details | | |
| **Alternate Scenario** | None | | |
| **Author** | Azhar | | |

A diagram of a system

AI-generated content may be incorrect.

Figure 2.2.5. F04 Receive e-Ticket/QR Code Sequence Diagram F04 Receive e-Ticket/QR Code Sequence Diagram

### F05 View Registration History

**Table 2.2.6. 1** – **Table 2.2.6. 3** describe the view registration history function and its description. While **Table 2.2.6. 4** shows the first version of the view registration history use case specification table, and **Table 2.2.6. 5** shows the second version of the view registration history use case specification table. And **Figure 2.2.6. 1** depicts the sequence diagram for the view registration history function.

Table 2.2.6. REQ\_F0501 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0501 | **Version** | **1.0** |
| **Item** | When the user navigates to the show QR screen, the system shall display the name of the event and generate a QR code specific to the user and event | | |
| **Author** | Azhar | | |

Table 2.2.6. REQ\_F0502 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0502 | **Version** | **1.0** |
| **Item** | If no events are retrieved from the database, the system shall display an error message. | | |
| **Author** | Azhar | | |

Table 2.2.6. REQ\_F0503 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0503 | **Version** | **1.0** |
| **Item** | If at least one event is retrieved, the system shall display all events in a list, with the name, event description and a “Generate QR Code” button | | |
| **Author** | Azhar | | |

Table 2.2.6. F05 View Registration History Use Case Specification Table Version 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F05 View Registration History | **Version** | 1.0 |
| **Purpose** | To allow users to view past events that they have registered | | |
| **Actor** | Student | | |
| **Precondition** | Student has registered to an event | | |
| **Postcondition** | None | | |
| **Main Flow** | 1. The user shall navigate to the “View Registration History” screen. 2. The system shall retrieve all registered events to the user from the database 3. The system shall display all registered events to the user in list form. 4. The system shall display two buttons, “View details” and “Generate QR code” | | |
| **Alternate Scenario** | None | | |
| **Author** | Azhar | | |

Table 2.2.6. F05 View Registration History Use Case Specification Table Version 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F05 View Registration History | **Version** | 2.0 |
| **Purpose** | To allow users to view past events that they have registered | | |
| **Actor** | Student | | |
| **Precondition** | Student had registered to an event | | |
| **Postcondition** | None | | |
| **Main Flow** | 1. The user navigates to the “Registration History” screen. 2. The system retrieves all registered events to the user from the database 3. The system displays all registered events to the user in list format. 4. The system shall display two buttons, “View details” and “Generate QR code” | | |
| **Alternate Scenario** | None | | |
| **Author (Updated By)** | Sulaiman | | |

A diagram of a system

AI-generated content may be incorrect.

Figure 2.2.6. F05 View Registration History Sequence Diagram

### F06 Show QR Attendance

**Table 2.2.7. 1** – **Table 2.2.7. 4** describe the show QR attendance function and its description. While **Table 2.2.7. 5** shows the show QR attendance use case specification table. And **Figure 2.2.7. 1** depicts the sequence diagram for the show QR attendance function.

Table 2.2.7. REQ\_F0601 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0601 | **Version** | **1.0** |
| **Item** | When the student navigates to the QR Attendance screen, the system shall retrieve the registered event ticket from the database. | | |
| **Author** | Yousef | | |

Table 2.2.7. REQ\_F0602 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0602 | **Version** | **1.0** |
| **Item** | The system shall generate a unique QR code containing encoded student and event data. | | |
| **Author** | Yousef | | |

Table 2.2.7. REQ\_F0603 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0603 | **Version** | **1.0** |
| **Item** | The system shall display the event title below the QR code. | | |
| **Author** | Yousef | | |

Table 2.2.7. REQ\_F0604 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0604 | **Version** | **1.0** |
| **Item** | If the student is not registered in any event, an error message is displayed. | | |
| **Author** | Yousef | | |

Table 2.2.7. F06 Show QR Attendance Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F06 Show QR Attendance | **Version** | 1.0 |
| **Purpose** | To allow a student to display a QR code as a ticket for check-in at the event venue | | |
| **Actor** | Student | | |
| **Precondition** | Student has successfully registered and paid for an event | | |
| **Postcondition** | QR Code is displayed and ready to be scanned | | |
| **Main Flow** | 1. Student navigates to registrations history 2. The system displays all event bookings. 3. Student clicks on “Ticket QR” button besides the event. 4. System retrieves event and ticket info 5. The system displays QR code and event name | | |
| **Alternate Scenario** | If the student is not registered in any event, an error message is displayed | | |
| **Author** | Yousef | | |

A black screen with white text

AI-generated content may be incorrect.

Figure 2.2.7. F06 Show QR Attendance Sequence Diagram

### F07 Create a New Event & F08 Set Event Details

**Table 2.2.8. 1** – **Table 2.2.8. 4** describe the set event details function and its description. While **Table 2.2.8. 5** shows the set event details use case specification table. And **Figure 2.2.8. 1** depicts the sequence diagram for the set event details function.

Table 2.2.8. REQ\_F0701 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0701 | **Version** | **1.0** |
| **Item** | When the organizer accesses the Create Event page, the system shall display input fields for event title, date, time, location, capacity, ticket price, and description. | | |
| **Author** | Yousef | | |

Table 2.2.8. REQ\_F0702 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0702 | **Version** | **1.0** |
| **Item** | When the organizer submits the form, the system shall validate all required fields. | | |
| **Author** | Yousef | | |

Table 2.2.8. REQ\_F0703 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0703 | **Version** | **1.0** |
| **Item** | If all inputs are valid, the system shall save the event to the database. | | |
| **Author** | Yousef | | |

Table 2.2.8. REQ\_F0704 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0704 | **Version** | **1.0** |
| **Item** | If any input is invalid, the system shall display an error message next to the field. | | |
| **Author** | Yousef | | |

Table 2.2.8. F07 Set Event Details Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F07 Set Event Details | **Version** | 1.0 |
| **Purpose** | To allow event organizers to input and save event information | | |
| **Actor** | Event Organizer | | |
| **Precondition** | Organizer has logged into the system | | |
| **Postcondition** | New event is stored in the database | | |
| **Main Flow** | 1. Event organizer navigate to Create Event form 2. System displays input fields 3. Event organizer fills fields (title, date, time, location, etc.) 4. Event organizer clicks "Create Event" 5. System validates and saves event | | |
| **Alternate Scenario** | If any field is invalid or empty, the system displays an error and prevents submission | | |
| **Author** | Yousef | | |

A black and white screen with white text

AI-generated content may be incorrect.

Figure 2.2.8. F07 Set Event Details Sequence Diagram

### F09 View Events

**Table 2.2.9. 1** – **Table 2.2.9. 5** describe the view events function and its description. While **Table 2.2.9. 6** shows the view events use case specification table. And **Figure 2.2.9. 1** depicts the sequence diagram for the view events function.

Table 2.2.9. REQ\_F0901 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0901 | **Version** | **1.0** |
| **Item** | When the organizer navigates to the My Events page, the system shall retrieve all events associated with that user. | | |
| **Author** | Yousef | | |

Table 2.2.9. REQ\_F0902 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0902 | **Version** | **1.0** |
| **Item** | The system shall display each event with its name, date, and current status ("Upcoming", "Completed"). | | |
| **Author** | Yousef | | |

Table 2.2.9. REQ\_F0903 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0903 | **Version** | **1.0** |
| **Item** | The system shall provide an "Edit" button beside each event that redirects to the event form. | | |
| **Author** | Yousef | | |

Table 2.2.9. REQ\_F0904 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0904 | **Version** | **1.0** |
| **Item** | The system shall provide a "Delete" button beside each event and confirm deletion via a popup. | | |
| **Author** | Yousef | | |

Table 2.2.9. REQ\_F0905 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F0905 | **Version** | **1.0** |
| **Item** | If no events exist, the system shall display the message: "No events found." | | |
| **Author** | Yousef | | |

Table 2.2.9. F09 View Events Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F09 View Events | **Version** | 1.0 |
| **Purpose** | To allow the organizer to view a list of events they have created | | |
| **Actor** | Event Organizer | | |
| **Precondition** | Organizer has created at least one event | | |
| **Postcondition** | A list of created events is shown | | |
| **Main Flow** | 1. Organizer opens "My Events" page 2. System retrieves events from database 3. System displays each event with status | | |
| **Alternate Scenario** | If no events exist, system displays a "No events found" message | | |
| **Author** | Yousef | | |

A black background with white text

AI-generated content may be incorrect.

Figure 2.2.9. F09 View Events Sequence Diagram

### F10 View Registrations & Attendance Status

**Table 2.2.10. 1** & **Table 2.2.10. 2** describe the view registrations and attendance status function and its description. While **Table 2.2.10. 3** shows the view registrations and attendance status use case specification table. And **Figure 2.2.10. 1** depicts the sequence diagram for the view registrations and attendance status function.

Table 2.2.10. REQ\_F1001 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F1001 | **Version** | **1.0** |
| **Item** | If no events exist, the system shall display the message: "No events found." | | |
| **Author** | Sulaiman | | |

Table 2.2.10. REQ\_F1002 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F1002 | **Version** | **1.0** |
| **Item** | If no student registered student in the event, the system shall display “No Registered Student Yet” message instead of the empty table. | | |
| **Author** | Sulaiman | | |

Table 2.2.10. F10 View Registrations & Attendance Status Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F10 View Registrations & Attendance Status | **Version** | 1.0 |
| **Purpose** | To allow the organizer to view the details of the registrations and attendance list | | |
| **Actor** | Event Organizer | | |
| **Precondition** | Organizer has created at least one event | | |
| **Postcondition** | A table that contains registered students in the event and their attendance status shall be displayed. | | |
| **Main Flow** | 1. Organizer opens "My Events" page 2. System retrieves events from database 3. System displays each event with status 4. Organizer may click "View Details" 5. System displays the registrations and attendance details | | |
| **Alternate Scenario** | If no events exist, the system displays a "No events found" message. | | |
| **Author** | Sulaiman | | |

A black background with white text

AI-generated content may be incorrect.

Figure 2.2.10. F10 View Registrations & Attendance Status Sequence Diagram

### F11 Check-in Upon Arrival

**Table 2.2.11. 1** & **Table 2.2.11. 2** describe the check-in upon arrival function and its description. While **Table 2.2.11. 3** shows the check-in upon arrival use case specification table. And **Figure 2.2.11. 1** depicts the sequence diagram for the check-in upon arrival function.

Table 2.2.11. REQ\_F1101 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F1101 | **Version** | **1.0** |
| **Item** | The system shall update the attendance status of the user in the event after validating the QR. | | |
| **Author** | Sulaiman | | |

Table 2.2.11. REQ\_F1102 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F1102 | **Version** | **1.0** |
| **Item** | The system shall display an error message on the event organizer device if the QR is invalid/non-related QR. | | |
| **Author** | Sulaiman | | |

Table 2.2.11. F11 Scan Attendance QR Code & Update Attendance Status Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F11 Check-in Upon Arrival | **Version** | 1.0 |
| **Purpose** | To allow the organizer to update the attendance list automatically | | |
| **Actor** | Event Organizer | | |
| **Precondition** | None | | |
| **Postcondition** | Attendance status of the student updated in the database | | |
| **Main Flow** | 1. Event organizer navigates to “Scan QR” screen 2. The system uses the camera of the phone allowing organizers to scan. 3. Event organizer aligns the camera on the QR code of the student 4. The system captures and validates the scanned QR. 5. The system updated the attendance status of the student on the event | | |
| **Alternate Scenario** | The system shall display an error message on the event organizer device if the QR is invalid/non-related QR. | | |
| **Author** | Sulaiman | | |

A black background with white rectangles

AI-generated content may be incorrect.

Figure 2.2.11. F11 Scan Attendance QR Code & Update Attendance Status Sequence Diagram

### F12 Login

**Table 2.2.12. 1** – **Table 2.2.12. 3** describe the login function and its description. While **Table 2.2.12. 4** shows the login use case specification table. And **Figure 2.2.12. 1** depicts the sequence diagram for the login function.

Table 2.2.12. REQ\_F1201 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F1201 | **Version** | **1.0** |
| **Item** | The system shall validate user’s inputs before sending a login request to the university’s database. | | |
| **Author** | Sulaiman | | |

Table 2.2.12. REQ\_F1202 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F1202 | **Version** | **1.0** |
| **Item** | The system shall display a descriptive error message if the form inputs are invalid or if any input field is empty. | | |
| **Author** | Sulaiman | | |

Table 2.2.12. REQ\_F1203 Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | REQ\_F1203 | **Version** | **1.0** |
| **Item** | The system shall not give access to the system if the user is not recognized in the university’s database. | | |
| **Author** | Sulaiman | | |

Table 2.2.12. F012 Login Use Case Specification Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | F12 Login | **Version** | 1.0 |
| **Purpose** | To give access to users to the system | | |
| **Actor** | Event Organizer / Student | | |
| **Precondition** | None | | |
| **Postcondition** | Access given to the user based on the user’s role. | | |
| **Main Flow** | 1. User Navigates to Login Page 2. System Displays the Login Page 3. User Fills in and Submits their login credentials 4. System Verify the submitted information 5. System authenticates the user into the system 6. System displays a success message and redirects to respective home pages | | |
| **Alternate Scenario** | 1. When the login credentials do not match anything available in the database, the system will display an error message. 2. When the fields are empty, the system will display an error message. | | |
| **Author** | Sulaiman | | |

A black screen with white text

AI-generated content may be incorrect.

Figure 2.2.12. F12 Login Sequence Diagram

## State Transition Diagrams

**Figure 2.3. 1** shows the high-level state transition diagram for the Campus Event Check-in System. The states illustrated are general and not specific to any user role. After successfully logging into the system, users can transition between different states depending on their actions and interactions within the application.

A diagram of a login page

AI-generated content may be incorrect.

Figure 2.3. High level State Transition Diagram

**Figure 2.3. 2** below shows the states available for the event organizer in the system. Once the event organizer successfully logs into the system, they will be directed to the 'Event Organizer Home Page' state. This state serves as the central hub for accessing all other states available to the event organizer.

A diagram of a event organization

AI-generated content may be incorrect.

Figure 2.3. Event Organizer State Transition Diagram

**Figure 2.3. 3** shows the states available for the students in the system. Once the student successfully logs into the system, they will first be directed to the 'Student Home Page' state. This state serves as the central hub for accessing all other states available to the students.

A diagram of a computer

AI-generated content may be incorrect.

Figure 2.3. Student State Transition Diagram

## Performance Requirements

**Table 2.4. 1** describes the performance requirements and their descriptions for the Campus Event Check-in System.

Table 2.4. Performance Requirements Table

|  |  |  |  |
| --- | --- | --- | --- |
| *Requirement ID* | *Description* | *Priority* | *Author* |
| REQ\_P001 | The system shall respond to end user requests within 1 to 5 seconds, ensuring a responsive user experience. | High | Sulaiman |
| REQ\_P002 | The system shall be able to support up to 1000 end user connections simultaneously, accommodating the diverse user base. | High | Sulaiman |
| REQ\_P003 | The system shall support integration via standard APIs with external services (e.g., payment gateway, student database), ensuring secure and seamless data exchange. | High | Sulaiman |
| REQ\_P004 | The system shall be designed to support different display sizes of various devices, promoting accessibility across different platforms for students and lecturers. | Medium | Sulaiman |
| REQ\_P005 | The system shall query data from the database within 2 seconds, ensuring quick access to information for the students. | Medium | Sulaiman |

## Security Requirements

**Table 2.5. 1** describes the security requirements and their descriptions for the Campus Event Check-in System.

Table 2.5. Security Requirements Table

|  |  |  |  |
| --- | --- | --- | --- |
| *Requirement ID* | *Description* | *Priority* | *Author* |
| REQ\_SR001 | The system's data, including personal and payment information of students, shall be protected from unauthorized access, ensuring the highest level of security. | High | Sulaiman |
| REQ\_SR002 | The system shall implement encryption, backup, and recovery mechanisms for data, ensuring data integrity and security for the students and event organizers. | High | Sulaiman |

## Usability Requirements

**Table 2.6. 1** describes the usability requirements and their descriptions for the Campus Event Check-in System.

Table 2.6. Usability Requirements Table

|  |  |  |  |
| --- | --- | --- | --- |
| *Requirement ID* | *Description* | *Priority* | *Author* |
| REQ\_UR001 | The system shall provide a clean and intuitive user interface that guides users (students and event organizers) through event registration, payment, and check-in workflows without requiring prior training. | High | Sulaiman |
| REQ\_UR002 | All user-facing text and interface labels shall be written in simple, concise, and consistent language, with optional multilingual support in future versions. | Medium | Sulaiman |
| REQ\_UR003 | The system shall provide feedback messages (e.g., success, error, loading indicators) in response to user actions such as login, payment, or check-in. | High | Sulaiman |
| REQ\_UR004 | The system shall complete key operations (e.g., event registration, QR code generation) within 2 seconds under normal network conditions. | High | Sulaiman |
| REQ\_UR005 | The system shall provide a mobile-friendly interface with responsive layouts for smartphones and tablets. | High | Sulaiman |
| REQ\_UR006 | The system shall use color contrast ratios that meet WCAG 2.1 AA standards to ensure readability for users with visual impairments. | High | Sulaiman |
| REQ\_UR007 | The system shall allow students to view and manage their registered events in a centralized dashboard with clear status indicators (e.g., “Paid”, “Pending”, “Checked-In”). | Medium | Sulaiman |
| REQ\_UR008 | The system shall guide first-time users with contextual tooltips or brief onboarding hints for critical actions like event registration or check-in. | Low | Sulaiman |

## Logical Database Requirements

The system is designed to store and manage a variety of data securely. All information is stored in an **encrypted online database**, which can only be accessed through **authorized requests from the web server** or by the **system administrator**, ensuring strong data protection and integrity.

**Figure 2.7. 1** presents the system's **Entity Relationship Diagram (ERD)**, which includes three main tables: **Event**, **Registration**, and **Payment**—representing the core components of the system’s data model.

The **Event** table includes a Created\_By column, which records the identity of the organizer who created the event, enabling traceability and accountability. The **Registration** table links students to their respective events, while also storing their ticket and attendance status.

The **Payment** table is carefully designed to follow **security best practices**. Instead of storing sensitive card details, it stores only non-sensitive data such as a Payment\_Token, Transaction\_ID, Paid\_At timestamp, and Payment\_Status. This design ensures compliance with industry standards like **PCI-DSS**, reducing security risks and supporting secure integration with third-party payment gateways.

Although the system also involves **Student** and **Event\_Organizer** data, these tables are **not included in the ERD**. This is because their data is retrieved directly from the **university’s central database**. This decision aligns with **database design best practices**, which emphasize maintaining a **single source of truth** to avoid data duplication, inconsistency, and synchronization issues.

The database structure adheres to **normalization principles**, eliminating redundancy and promoting data consistency across the system. Additionally, the ERD is designed with **scalability in mind**, allowing for the seamless integration of future features such as event categories, participant feedback, or attendance analytics, without disrupting the existing schema.

Overall, the ERD reflects a well-normalized, secure, and maintainable structure aligned with both academic standards and real-world best practices in database design.

A diagram of a function

AI-generated content may be incorrect.

Figure 2.7. System ERD

### Event Table

**Table 2.7. 1** shows the data dictionary of the Event table.

Table 2.7. Event Table Data Dictionary

|  |  |  |
| --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** |
| Event\_ID | int (PK) | Unique identifier for the event |
| Name | varchar(255) | Title of the event |
| Description | text | Detailed description of the event |
| Location | varchar(255) | Physical or online location of the event |
| Date | date | Scheduled date of the event |
| Time | time | Scheduled time of the event |
| Capacity | int | Maximum number of attendees allowed |
| Ticket\_Link | varchar(255) | URL to download/view the ticket |
| Created\_By | int | ID of the student or university’s staff (from university DB) |

### Registration Table

**Table 2.7. 2** shows the data dictionary of the Registration table.

Table 2.7. Registration Table Data Dictionary

|  |  |  |
| --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** |
| Registration\_ID | int (PK) | Unique identifier for the registration |
| Student\_ID | int (FK) | ID of the student registering (from university DB) |
| Event\_ID | int (FK) | ID of the event being registered for |
| Ticket\_Path | varchar(255) | Path or URL to the generated ticket |
| Attendance\_Status | varchar(50) | Status of attendance (e.g., Not Checked-In, Present) |

### Payment Table

**Table 2.7. 3** shows the data dictionary of the Payment table.

Table 2.7. Payment Table Data Dictionary

|  |  |  |
| --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Description*** |
| Payment\_ID | int (PK) | Unique identifier for the payment |
| Registration\_ID | int (FK) | ID of the associated registration |
| Payment\_Token | varchar(255) | Token or intent ID from the payment gateway |
| Transaction\_ID | varchar(100) | Unique transaction ID from the gateway |
| Paid\_At | datetime | Date and time when the payment was completed |
| Payment\_Status | varchar(50) | Status of the payment (e.g., Success, Failed) |

## Design Constraints

**Table 2.8. 1** describes the design constraints and their descriptions for the Campus Event Check-in System.

Table 2.8. Design Constraints Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Requirement ID*** | ***Description*** | ***Priority*** | ***Author*** |
| REQ\_C001 | The system must integrate with the university’s database for retrieving student and event organizer data, avoiding duplication and maintaining data integrity. | High | Sulaiman |
| REQ\_C002 | All data must be stored in an encrypted online database, accessible only through authenticated server or admin requests. | High | Sulaiman |
| REQ\_C003 | Sensitive payment data (e.g., card number, CVV) must not be stored. Only secure tokens and transaction IDs from payment gateways should be saved. | High | Sulaiman |
| REQ\_C004 | The system must be accessible as a cross-platform mobile application with a responsive and user-friendly interface. | High | Sulaiman |
| REQ\_C005 | The system must be developed using technologies compatible with mobile platforms (e.g., React Native, Flutter). | High | Sulaiman |
| REQ\_C006 | All user inputs must be validated both client-side (in-app) and server-side, with real-time feedback where applicable. | High | Sulaiman |
| REQ\_C007 | The system must support future scalability, such as adding new event features, roles, or analytics modules without disrupting the core architecture. | Medium | Sulaiman |
| REQ\_C008 | Role-based access must be enforced, ensuring each user type (Student, Organizer, Admin) has access only to relevant features. | High | Sulaiman |
| REQ\_C009 | The database must enforce referential integrity using primary and foreign keys to maintain consistent relationships between data. | High | Sulaiman |
| REQ\_C010 | All date and time values must be stored in UTC and displayed to users in their local timezones. | High | Sulaiman |

## Software System Attributes

**Table 2.9. 1** describes the software system attributes and their descriptions for the Campus Event Check-in System.

Table 2.9. Software System Attributes Table

|  |  |  |  |
| --- | --- | --- | --- |
| ***Requirement ID*** | ***Description*** | ***Priority*** | ***Author*** |
| REQ\_ST001 | Reliability – The system shall be available and functional with at least 99% uptime during operational hours. | High | Sulaiman |
| REQ\_ST002 | Availability – The system shall be accessible 24/7, except during scheduled maintenance, with minimal downtime for updates. | High | Sulaiman |
| REQ\_ST003 | Security – All sensitive data shall be encrypted in storage and during transmission. Role-based access control must be enforced throughout. | High | Sulaiman |
| REQ\_ST004 | Maintainability – The codebase shall be modular and documented to allow easy updates and debugging. | Medium | Sulaiman |
| REQ\_ST005 | Portability – The system shall support cross-platform deployment (e.g., Android and iOS) using a single codebase (e.g., React Native). | High | Sulaiman |
| REQ\_ST006 | Scalability – The system architecture shall allow future expansion to support more users, roles, and features without major structural changes. | Medium | Sulaiman |
| REQ\_ST007 | Usability – The system shall provide an intuitive and responsive user interface for students and event organizers. | High | Sulaiman |

## Supporting Information

During the requirements elicitation phase for the mobile-based event registration system for university students and organizers, various techniques were employed to ensure a clear and comprehensive understanding of the system's requirements. The following elicitation methods were utilized:

1. Interviews

An interview was conducted with a university club committee member who is actively involved in organizing student programs. The purpose of the interview was to understand the challenges faced when announcing events and managing the attendance of registered students. The insights gained highlighted issues such as limited reach of event announcements, difficulties in tracking attendance accurately, and the lack of a centralized, user-friendly system for both organizers and participants. These findings informed the system’s emphasis on features like event visibility, registration tracking, and QR-based attendance management.

2. Questionnaires

A structured questionnaire was developed and distributed to students using the **Kano Model** to categorize and prioritize system features based on user satisfaction. The Kano Model allowed the team to distinguish between **basic needs**, **performance needs**, and **excitement features**. Respondents were asked to evaluate potential functionalities—such as QR code-based attendance, and registration history tracking—based on their perceived importance and expected satisfaction.

This method enabled a data-driven understanding of user priorities and helped the team focus on features that would maximize user satisfaction while ensuring the system remained efficient and relevant to student needs.

3. Observation

Observational studies were conducted on existing mobile event and ticketing applications to understand user interaction patterns, interface design trends, and pain points. This helped in identifying successful design elements and functionality that could be adapted or improved for the university context. Observations also provided practical guidance for implementing features like QR scanning, dynamic event listing, and payment integration in a user-friendly manner.

### Interview

**Table 2.10.1. 1** details the features, requests, and feedback gathered through interviews with relevant stakeholders, which have been translated into requirements for the Campus Event Check-in System. However, for the full interview it can be accessed from this [Link](https://mmuedumy-my.sharepoint.com/:v:/g/personal/1211305566_student_mmu_edu_my/EbTbqncMpIxKuQ9B9HCNzcUB7MimpUCG5LOyt2PbeqzbZg?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=KdH7I4).

Table 2.10.1. Obtained Requirements From Interview Table

|  |  |  |
| --- | --- | --- |
| ***Feature / Request / Opinion*** | ***Description*** | ***Requirement ID*** |
| We want students to log in using their university accounts | The system must integrate with the university’s authentication system to allow secure login for all users. | REQ\_F011 |
| Students should be able to see all upcoming events and register easily | The system should provide students with access to browse available events and register with ease. | REQ\_F00, REQ\_F01 |
| After registering, students should get a QR code for attendance | Upon successful registration and payment, students should receive a QR code for event attendance check-in. | REQ\_F04 |
| We need a way to create new events and enter all the event details | Organizers must be able to create new events by setting titles, descriptions, dates, locations, and capacity. | REQ\_F07 |
| It should be easy to see who registered and who actually showed up | Organizers should have access to view the list of registered students and their attendance status. | REQ\_F09 |
| We'd like to scan the student's QR code when they arrive | The system should enable organizers to scan QR codes for quick and accurate student check-in on event day. | REQ\_F10 |
| The system should be secure so only authorized users can access it | All system functions and data must be protected through secure access control and encrypted communication. | REQ\_SR001 |

### Questionnaires

**Table 2.10.2. 1** details the features, requests, and feedback obtained through questionnaires and surveys distributed to relevant stakeholders, which have been translated into requirements for the Campus Event Check-in System.

Table 2.10.2. Obtained Requirements From Questionnaires Table

|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **Requirement ID** |
| Login with Student ID | The system should allow students to log in using their university-issued student ID and password. | REQ\_F11 |
| View Upcoming Events | Students should be able to browse a list of upcoming campus events online. | REQ\_F00 |
| View Event Details | Students should be able to see detailed event info such as image, date, venue, and capacity. | REQ\_F01 |
| Register for Events | Students should be able to register for events quickly and easily through the system. | REQ\_F02 |
| Make Online Payment | The system should allow students to pay online for paid events through a secure gateway. | REQ\_F03 |
| Receive e-Ticket/QR Code | After registering and paying, students should receive a QR code to use as their event entry ticket. | REQ\_F04 |
| View Registration History | Students should be able to view a history of all their past event registrations. | REQ\_F05 |
| QR Code Check-In | Students should be able to check in to events using the QR code provided by the system. | REQ\_F10 |

### Observation

**Table 2.10.3. 1** details the features, requests, and insights obtained through observing the behaviors and actions within existing systems, which have been translated into requirements for the Campus Event Check-in System.

Table 2.10.3. Obtained Requirements From Observation Table

|  |  |  |
| --- | --- | --- |
| **Feature / Characteristic** | **Description** | **Requirement ID** |
| Login | Allows both students and event organizers to access the system using their university credentials. | REQ\_F11 |
| View Event Details | Displays full details of a selected event, including date, location, and description. | REQ\_F01 |
| Reliability | The system should consistently perform its functions under defined conditions without failure. | REQ\_ST001 |
| Availability | The system should be accessible at any time with minimal downtime or interruptions. | REQ\_ST002 |
| Security | Data should be encrypted in transit and at rest, with strict access control for all users. | REQ\_ST003 |
| Maintainability | The system should have modular, well-documented code that supports updates and bug fixing with minimal effort. | REQ\_ST004 |
| Portability | The system should run on both Android and iOS devices using a cross-platform framework (e.g., React Native). | REQ\_ST005 |
| Scalability | The system architecture should support future growth such as more features, users, and integrations. | REQ\_ST006 |
| Usability | The system should have an intuitive and accessible interface for students and organizers. | REQ\_ST007 |

# Verification

To ensure the quality, correctness, and alignment of the system with user expectations, we will implement a multi-layered verification process throughout the development of the Campus Event Check-in System. These verification activities are critical in ensuring that the system functions as intended, is user-friendly, and fulfills its objectives for both students and event organizers.

This section outlines the verification methods used to validate the software requirements and their associated criteria. These practices will help confirm that the developed system meets stakeholder needs while maintaining robustness, reliability, and usability.

## Inspection

Inspection is a structured verification technique that involves reviewing the software requirements and design documents to identify defects, missing elements, or inconsistencies. This activity will involve the project team and relevant university stakeholders to ensure that the requirements are sound and implementable.

Inspections will focus on the following criteria:

1. **Compliance** – Ensure all requirements conform to institutional policies, technical standards, and software engineering best practices.
2. **Completeness** – Verify that all functionalities (e.g., login, event registration, QR check-in, history viewing) are fully and clearly documented.
3. **Consistency** – Confirm that the requirements do not conflict with each other or repeat information unnecessarily.
4. **Clarity** – Ensure that each requirement is written in a precise, understandable, and unambiguous manner.
5. **Feasibility** – Determine whether the proposed features can realistically be implemented within the project scope and constraints.

## Walkthroughs

Walkthroughs will complement inspections by offering a more collaborative and exploratory approach to verification. These involve the author of the requirements (typically the analyst or developer) presenting the document step-by-step to stakeholders such as students, club committee members, and system developers.

Walkthrough participants will review the logic and practicality of the requirements based on their domain experience and will provide feedback on potential improvements or missing details.

Key walkthrough criteria include:

1. **Compliance** – Confirm alignment with standards and expectations from university systems.
2. **Completeness** – Ensure no required features or stakeholder concerns are omitted.
3. **Consistency** – Identify and eliminate any contradictions or ambiguous statements.
4. **Clarity** – Validate that each feature is explained in a user-friendly and developer-ready format.
5. **Feasibility** – Assess whether each function (e.g., QR scanning or integration with university accounts) can be technically delivered.
6. **Accuracy** – Confirm that described behaviors align with what users need and how they will interact with the system.
7. **Adequacy** – Evaluate whether the documented requirements address the real-world problems identified through interviews and observation.

## Prototyping

A visual prototype of the mobile application will be developed to validate how well the proposed system satisfies the user requirements in a practical, hands-on form. The prototype will simulate key features such as:

* Viewing events
* Registering and paying
* Receiving a QR code
* Checking in via camera

Stakeholders will interact with the prototype and assess whether the system is intuitive, visually clear, and meets functional expectations.

Prototyping criteria:

1. **User Interface (UI)** – Validate that the interface is intuitive, mobile-friendly, and aligned with expected interaction flows.
2. **Correctness** – Verify that simulated actions match the intended functionality (e.g., QR check-in actually triggers status updates).
3. **Quality** – Ensure the prototype reflects design consistency, responsiveness, and accessibility where possible.

# ****Appendices****

**Assumptions and Dependencies**

The successful operation of the Campus Event Check-in mobile application depends on several assumptions and external factors. These assumptions outline the expected environment and user readiness, while dependencies identify critical systems or tools required for the system to function as intended.

1. It is assumed that all users (students and event organizers) have access to a functioning **mobile device** running Android or iOS with the required hardware specifications (camera, storage, network).
2. It is assumed that users will have **stable internet access** via Wi-Fi or mobile data to enable communication with the backend system (e.g., for registration, payments, and QR scanning).
3. The mobile devices used must **grant necessary permissions**, such as camera access, for QR code functionality to work correctly.
4. The system depends on the **availability and proper functioning** of the integrated **university authentication system**, which will be used for secure login via student credentials.
5. The system depends on third-party services such as the **payment gateway API** which must remain available and responsive.
6. It is assumed that users have basic familiarity with using mobile apps and can follow intuitive interfaces for tasks like viewing events, registering, and scanning QR codes.
7. The backend database and server infrastructure must remain operational and **compatible with the mobile frontend** at all times.

# References

* <https://buildfire.com/top-mobile-development-practices/>
* Requirements Engineering: Fundamentals, Principles, and Techniques Author: Klaus Pohl
* Requirements Engineering: Processes and Techniques Authors: Gerald Kotonya, Ian Sommerville
* <https://www.toptal.com/designers/mobile-ui/mobile-ux-design-principles>