McMaster University | ENGINEERING Computing & Software

MacSync

MacSync

Mahad Ahmed, Abyan Jaigirdar, Prerna Prabhu, Farhan Rahman, Ali Zia

Version 1, 2025-10-25

Table of Contents

Control Information **2**
(G) Goals **3**
(G.1) Context and Overall Objectives *3*
(G.2) Current situation *4*
(G.3) Expected Benefits *4*
(G.4) Functionality overview *5*
Functional Requirements 6
Key Functional Requirements 6
Non-Functional Requirements 6
RACI Matrix 6
(G.5) High-level usage scenarios *7*
(G.6) Limitations and Exclusions *9*
(G.7) Stakeholders and requirements sources *9*
G.7.1 Stakeholder Table 9
G.7.2 Personas Table 10
G.7.3 Additional Requirements Sources 10
(E) Environment **12**
(E.1) Glossary *12*
(E.1.1) Vocabulary 12
(E.1.2) Domain Model 13
(E.2) Components *14*
(E.3) Constraints *15*
(E.4) Assumptions *16*
(E.5) Effects *16*
(E.6) Invariants *17*
(S) System **18**
(S.1) Components *18*
(S.2) Functionality *19*
(S.2.1) Registration & Ticketing Module 19
(S.2.2) Payment Processing Module 19
(S.2.3) Bus / RSVP / Table Sign-ups Module 20
(S.2.4) Role / Feature-Based Access Control (RBAC/FBAC) 20
(S.2.5) Privacy, Data Integrity, and Security Controls 21
(S.3) Interfaces *21*
(S.3.1) User Interface API 21
(S.3.2) Payment System API 22
(S.3.3) Role Based Access Control / Feature Based Access Control API 22
(S.3.4) Event Management System API 23
(S.3.5) Sign-Ups API 23
(S.4) Detailed usage scenarios *24*
(S.4.1) Register for an Event 24

(S.4.2) Join Bus, Table, or RSVP Sign-Up . . . 24
(S.4.3) Check-In at Event . . . 25
(S.4.4) Create and Manage Events (Organizer/Admin) . . . 25
(S.4.5) Monitor Registrations and Analytics (Organizer/Executive) . . . 26
(S.4.6) Handle Waitlist Updates . . . 26
(S.5) Prioritization . . . 26
(S.5.1) Priority Table for Event Registration and Ticketing . . . 26
(S.5.2) Priority Table for Payment Processing Module . . . 27
(S.5.3) Priority Table for Bus / RSVP / Table Sign-ups Module . . . 28
(S.5.4) Priority Table for Role / Feature-Based Access Control (RBAC/FBAC) . . . 28
(S.5.5) Priority Table for Privacy, Data Integrity, and Security Controls . . . 29
(S.5.2) Prioritization Analysis . . . 29
(S.6) Verification and Acceptance Criteria . . . 29
Validation of Functional Requirements . . . 30
Fit Criteria for Non-Functional Requirements . . . 30
Selected Scenario for Testing . . . 30
Traceability Matrix . . . 31
(P) Project . . . 32
(P.1) Roles and personnel . . . 32
(P.2) Imposed technical choices . . . 33
(P.3) Schedule and milestones . . . 34
(P.3) Schedule and milestones . . . 34
P.3.1 Sprint Plan . . . 35
(P.4) Tasks and deliverables . . . 36
(P.5) Required technology elements . . . 38
(P.6) Risk and mitigation analysis . . . 39
Risk Analysis . . . 39
Threat Analysis . . . 40
(P.7) Requirements process and report . . . 40
(P.7.1) Interviews with Direct Stakeholders . . . 41
(P.7.2) Justification for Interviews . . . 41
References . . . 42

Academic Integrity Disclaimer

We would like to acknowledge that as a dedicated students of McMaster University, we have thoroughly read and comprehended the Academic Integrity Policy published by the university. We are committed to upholding the principles of academic honesty and integrity in all aspects of our educational journey. We understand the importance of acknowledging the work and ideas of others, and we pledge to ensure that all our academic endeavors are conducted with the utmost originality and compliance with the university's policy.

We affirm that the content presented in this document is entirely our own, and any external sources used have been appropriately cited and referenced.

Mahad Ahmed

As I submit my work, I, **Mahad Ahmed**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Abyan Jaigirdar

As I submit my work, I, **Abyan Jaigirdar**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Prerna Prabhu

As I submit my work, I, **Prerna Prabhu**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Farhan Rahman

As I submit my work, I, **Farhan Rahman**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Ali Zia

As I submit my work, I, **Ali Zia**, take full responsibility for the integrity of my work and promise to avoid any form of plagiarism, cheating, or dishonest behavior. This acknowledgment serves as a testament to my dedication to academic excellence and the fostering of a trustworthy academic community at McMaster University.

Control Information

Version	**Delivery**		**Feedback**	
	Deadline	*Delivered*	*Received*	*Integrated*
V1	2025-10-10	2025-10-10		

Mahad Ahmed

Here is a quick biography of **Mahad Ahmed**. You can contact them at **ahmem73@mcmaster.ca**

Abyan Jaigirdar

Here is a quick biography of **Abyan Jaigirdar**. You can contact them at **jaigia1@mcmaster.ca**

Prerna Prabhu

Here is a quick biography of **Prerna Prabhu**. You can contact them at **prabhp3@mcmaster.ca**

Farhan Rahman

Here is a quick biography of **Farhan Rahman**. You can contact them at **rahmam88@mcmaster.ca**

Ali Zia

Here is a quick biography of **Ali Zia**. You can contact them at **ziam8@mcmaster.ca**

(G) Goals

Goals are "needs of the target organization, which the system will address". While the development team is the principal user of the other books, the Goals book addresses a wider audience: essentially, all stakeholders [1]

It must contain enough information to provide — if read just by itself — a general sketch of the entire project. To this effect, chapter G.3 presents a short overview of the system and G.1 will typically include some key properties of the environment. As it addresses a wide readership, it should be clear and minimize the use of specialized technical terms. Together, G.1, G.2 and G.3 describe the rationale for the project. It is important to state these justifications explicitly. Typically, they are well understood at the start of the project, but management and priorities can change [1]

Control Information

Table 1. MacSync — Versionning Information — Goal Book

Section	Version	Lead	Delivered on	Reviewer	Approved on
G.1	V1	Abyan	2025-10-09	Prerna	2025-10-10
G.2	V1	Ali	2025-10-09	Abyan	2025-10-10
G.3	V1	Farhan	2025-10-02	Ali	2025-10-10
G.4	V1	Mahad	2025-10-09	Farhan	2025-10-10
G.5	V1	Prerna	2025-10-09	Mahad	2025-10-10
G.6	V1	Abyan	2025-10-09	Prerna	2025-10-10
G.7	V1	Ali	2025-10-09	Abyan	2025-10-10

(G.1) Context and Overall Objectives

The Large Event Management System (LEMS) is being developed to support the McMaster Engineering Society (MES) in planning and running large events such as formals, conferences, and national surveys. These events bring together many participants and require coordination of ticket sales, registrations, payments, waivers, bus and table sign-ups, and event check-ins.

Currently, organizers rely on tools like Google Forms, Google Sheets, Instagram, and Discord. This approach spreads information across different platforms, creates inefficiencies, and increases the amount of manual work. It also results in a less consistent experience for participants, who often need to use multiple systems depending on the event.

The goal of LEMS is to provide a single system that centralizes these processes. The expected outcomes are:

- Centralized registration, payments, waivers, scheduling, and check-ins
- Greater efficiency for organizers through reduced manual coordination and better reporting
- A consistent participant experience on both web and mobile interfaces
- Flexibility to support different event types, from formals to national surveys
- A sustainable platform that can continue to be used in future years

This project addresses current challenges while also creating a long-term solution for MES. In the near term, it will support events such as the Fireball Formal, CALE Conference, and CFES National Survey. In the long term, it will provide a reliable and reusable platform that future students can maintain and expand.

(G.2) Current situation

> *Current state of processes to be addressed by the project and the resulting system. It describes the current situation, upon which the system is expected to improve* [1]

The McMaster Engineering Society (MES) organizes many large student events on the McMaster University campus intended to attract hundreds of attendees. The current state of processes involved in running these events, from registration and ticketing to event updates are oftentimes scattered across multiple third-party platforms and tools such as Discord servers, Instagram stories and Google Forms. Due to this scattered approach, several barriers and inefficiencies are created for both students intending to participate in events organized by the MES and organizers dedicating their time to event logistics. From the perspective of attendees, there is frustration and a struggle to stay up to date and access clear information regarding event logistics. The scattered nature of information sources results in confusion, as attendees are forced to navigate multiple social media platforms or forms to stay up to date on events and complete a single registration. The result can be missed deadlines or uncertainty around event arrangements from seating to transportation arrangements. Additionally, the absence of centralized notifications further diminishes the experience, as the possibility of missing critical updates or reminders increases which can impact the event attendance of users. From the organizers perspective, there is a significant load in administrative work. Managing ticketing and signups across multiple platforms is time-consuming and with the added work of tracking accessibility requests and waivers can further increase the likelihood of omissions or errors in tracking. Overall, the current uncentralized process increases administrative overhead work for organizers reducing their efficiency and reduces the quality of experience for attendees, negatively impacting their experiences.

(G.3) Expected Benefits

The successful completion of this project will result in several key business benefits, directly addressing the challenges identified in G.1 and G.2. These benefits represent improvements to current processes and new capabilities made possible by the system.

Centralized Registration & Ticketing

By consolidating ticket purchasing, RSVPs, and waivers into one platform, students will have a consistent and reliable source of event information. This centralization reduces confusion, eliminates the need for scattered tools like spreadsheets, Google Forms, posters, instagram stories/posts, and ensures attendees have access to the latest updates at all times. For organizers, it significantly decreases redundant manual work, allowing them to focus more on event quality and participant engagement.

Payment Integration

Secure and flexible payment options (Stripe, Square, PayPal) will reduce risks associated with cash handling and provide students with fast, reliable payment methods. This creates trust between attendees and organizers while ensuring financial data is accurately tracked in real time.

Role-Based Access Control (RBAC/FBAC)

Granular permissions improve security and operational efficiency by ensuring organizers only access the tools relevant to their responsibilities. This reduces human error, improves accountability within event teams, and allows large-scale events to be managed more smoothly.

Bus & Table Sign-ups

Automated reservations with built-in capacity tracking will simplify logistics and eliminate overbooking. This reduces stress on organizers, prevents conflicts during event planning, and provides students with a transparent and fair reservation system.

Notifications & Reminders

Timely push notifications and reminders will improve event memorability, reduce no-shows, and prevent students from missing important updates. Organizers will also benefit from improved communication channels that minimize last-minute confusion and improve turnout rates.

Analytics & Reporting

Real-time dashboards will empower organizers and MES executives to make data-driven decisions. By providing insights into demographics, sales, accessibility needs, and financial tracking, the system enhances resource allocation, inclusivity, and overall evaluation of event success.

Attendee Experience

A mobile-first, intuitive, and accessible platform ensures that all students including those with accessibility or dietary needs can easily participate. This not only maximizes engagement but also reinforces inclusivity and fairness as core values of MES events.

In the long term, these benefits establish a sustainable and scalable foundation for MES event management. The system reduces administrative burden, improves user satisfaction, and ensures the continuity of high-quality student experiences. Additionally, the modular nature of the platform supports future expansions, including personalized recommendations, sponsor visibility, and post-event engagement features, keeping the system adaptable to evolving needs.

(G.4) Functionality overview

Overview of the functions (behavior) of the system. Principal properties only (details are in the System book). It is a short overview of the functions of the future system, a kind of capsule version of book S, skipping details but enabling readers to get a quick grasp of

> *what the system will do.* [1]

The MacSync Application provides a centralized platform for McMaster Engineering Society (MES) to host large-scale events. Listed below is a high-level overview and relevancy of the functional and non-functional requirements integral to the sytem.

Functional Requirements

- **Payments / Ticketing:** The application must integrate a payment gateway to create charges for event tickets. Each transaction should generate an invoice for the attendee and a sales report for system admin. Furthermore, The gateway should be able to handle real-time availability checks to prevent overselling tickets.
- **Role / Feature Based Access Control (RBAC/FBAC):** The system must provide access to various features depending on users privelege level. This will also include an admin UI to assign/revoke roles. This will ensure a clear seperation of use-cases for end users.
- **Bus / RSVP / Table Signups:** The system must provide an RSVP feature for students to automatically sign-up, the system will show live capacity information (i.e. the number of spots left). In the event that many users try to sign up at the same time, the system will ensure tickets are given out on a first-come first-serve basis to ensure fairness for users to register and prevent over selling tickets.

Key Functional Requirements

- **End-to-End Registration:** The full registration system including the payments, RSVP sign-ups, generated analytics and registration forms is the key component of the MacSync platform. It provides the full end-to-end solution for users to sign up for MES events and stands as the flag ship feature of the application.

Non-Functional Requirements

- **Privacy & Security:** The system must comply with regulations such as PIPEDA and/or PCI-DSS. Students will be providing personal information to create their MacSync platform profiles and be providing payment information to purchase tickets. The application must be able to ensure that any senstive information is protected.
- **Data-Integrity & Consistency:** The system must be able to have strong consistency in the event of concurrent transactional operations. This ensures that events arent over sold with limited capacity in the instance where many users are attempting to register at the same time.

RACI Matrix

Functional Requirement	MES Executives (Client)	Dev Team	Attendees / Students	Event Organizers	Check-in Staff / Event Volunteers
Payments / Ticketing	A	R	I	C	I
RBAC / FBAC (role & feature access)	C	R	I	C	I

Functional Requirement	MES Executives (Client)	Dev Team	Attendees / Students	Event Organizers	Check-in Staff / Event Volunteers
Bus / RSVP / Table Sign-ups	I	R	I	A	C
End-to-End Registration	A	R	I	C	I

Legend: R = Responsible · A = Accountable · C = Consulted · I = Informed

(G.5) High-level usage scenarios

Fundamental usage paths through the system. It presents the main scenarios (use cases) that the system should cover. The scenarios chosen for appearing here, in the Goals book, should only be the ***main usage patterns****, without details such as special and erroneous cases; they should be stated in user terms only, independently of the system's structure. Detailed usage scenarios, taking into account system details and special cases, will appear in the System book (S.4).* [1]

Principle Use Cases: 1. User Registration and Profile Setup 2. Browse and Select Event 3. Purchase Ticket 4. Register for Bus/Table 5. Organizer Event Management 6. Check-In and Attendance Tracking 7. Notifications and Reminders 8. Calendar Integration 9. Admin Oversight and Reporting 10. Submit Waiver & Preferences (dietary, accessibility)

Coarse Grained Scenarios ---

UC1: User Registration and Profile Setup 1. A student accesses the LEMS web or mobile platform. 2. The student signs up using their McMaster credentials or a verified email. 3. The system prompts for profile information, including dietary restrictions, accessibility needs, and emergency contacts. 4. The user accepts event policies and privacy terms. 5. The system stores the user profile and displays a personalized dashboard.

UC2: Browse and Select Event 1. The student logs in and navigates to "Upcoming Events." 2. The system displays available MES events such as Fireball Formal, Grad Dinner, and Pub Nights. 3. The student filters or searches by event type, date, or price. 4. The student selects a specific event to view details, including date, venue, ticket types, and transportation options. 5. The student chooses to register or purchase tickets for the event.

UC3: Purchase Tickets 1. The student opens the app or web platform. 2. The student selects the desired event from the list. 3. The system displays ticket types (e.g., general admission, dinner ticket, VIP). 4. The student selects a ticket and provides payment details using a supported provider (Stripe, Square, or PayPal). 5. The system processes payment securely and confirms the transaction. 6. The student receives a digital ticket confirmation via email and within their profile, containing a unique QR code for check-in.

UC4: Register for Bus/Table 1. The student navigates to the transportation or seating section of their registered event. 2. The system displays available bus routes and/or tables, along with real-time availability. 3. The student selects their preferred bus route or table seat. 4. The system validates availability and confirms the assignment. 5. The student receives an in-app and email confirmation with assigned bus and/or table details.

UC5: Organizer Event Management 1. An organizer logs in using their role-based access credentials. 2. The organizer navigates to the "Event Dashboard." 3. The dashboard displays analytics such as ticket sales, payment summaries, and waitlist numbers. 4. The organizer can update event information, adjust capacities, or open/close registration. 5. Organizers review dietary and accessibility data to prepare accommodations. 6. All updates are reflected for attendees in real time.

UC6: Check-In and Attendance Tracking 1. On event day, organizers access the "Check-In" interface via a tablet or mobile device. 2. Attendees present their QR ticket for scanning. 3. The system verifies the ticket, confirms entry, and marks the attendee as checked in. 4. If the device is offline, check-ins are cached and synchronized later. 5. Organizers can view attendance counts and identify no-shows in real time.

UC7: Notifications and Reminders 1. As event dates approach, the system automatically sends reminders to registered users. 2. Notifications include event time, location, and special instructions. 3. Organizers can send manual announcements (e.g., weather or venue changes). 4. Users may customize notification preferences in their profiles.

UC8: Calendar Integration 1. After ticket confirmation, the student adds the event to their preferred calendar (Google, Apple, or Outlook). 2. The system generates a `.ics` file containing event details and reminders. 3. Any organizer-initiated updates automatically sync to user calendars via notifications.

UC9: Admin Oversight and Reporting 1. An administrator logs in with elevated permissions. 2. The admin reviews system-wide metrics, including payment logs, event statistics, and error reports. 3. The admin exports reports for financial or attendance auditing. 4. The admin manages user roles, system settings, and compliance with McMaster's data policies. 5. Post-event, the admin archives event data and closes active sessions.

UC10: Submit Waivers and Preferences 1. After ticket purchase, the student is prompted to complete event-specific forms such as waivers, dietary restrictions, and accessibility preferences. 2. The system displays digital waiver forms that can be signed electronically. 3. The student enters any dietary preferences (e.g., vegetarian, nut-free) and accessibility needs (e.g., mobility assistance, interpreter request). 4. The system validates required fields and confirms submission. 5. The preferences and waiver are stored securely and linked to the student's event registration record. 6. Organizers can later view and export this data to support event preparation and accommodation planning.

(G.6) Limitations and Exclusions

The Large Event Management System (LEMS) is designed to address the key needs of event organization and participation, but there are aspects that the system will not attempt to cover. Clearly stating these exclusions helps define the scope and ensures expectations are realistic.

- The system will not replace external communication platforms such as email or Discord for general discussion outside of event-specific notifications.
- The system will not provide built-in social networking or messaging features beyond the delivery of event-related announcements.
- The system will not manage non-event related finances for the McMaster Engineering Society (MES) or other student groups. Its financial functionality is limited to event-related payments and reporting.
- The system will not include venue booking, catering, or logistics management beyond recording relevant event details entered by organizers.
- The system will not integrate with university academic systems (e.g., course registration or student records).
- The system will not provide offline functionality; access requires an internet connection to use web or mobile applications.
- The system will not guarantee support for events outside the scope of MES and affiliated student groups, though it may be adaptable by others in the future.

By identifying these exclusions, the scope of LEMS remains focused on providing a streamlined event management experience without overextending into unrelated domains.

(G.7) Stakeholders and requirements sources

Groups of people who can affect the project or be affected by it, and other places to consider for information about the project and system. It lists stakeholders and other requirements sources. It should define stakeholders as categories of people, not individuals, even if such individuals are known at the time of writing. The main goal of chapter G.7 is to avoid forgetting any category of people whose input is relevant to the project. It also lists documents and other information that the project, aside from soliciting input from stakeholders, can consult for requirements information. [1]

G.7.1 Stakeholder Table

Stakeholder Type	Stakeholder	Description
Direct	Event Attendees	Students registering for and attending MES events who interact with the signups, ticketing, payment and RSVP systems.
Direct	Event Organizers	MES club members responsible for event logistics from planning, managing and executing events.
Direct	Sponsors/Promoters	Companies or businesses interested in promoting their brand by associating with clubs and sponsoring events.
Indirect	University Administration	Interested in and overseeing compliance with institutional and legal laws and policies on events.
Indirect	Capstone Development Team	Responsible for building and maintaining the platform, integrating features to align with requirements.

G.7.2 Personas Table

Stakeholder Type	Stakeholder	Persona Name	Persona Description
Direct	Event Attendees	James Thompson	A first year business student interested in attending events throughout his academic career to meet new people and gain new experiences.
Direct	Event Organizers	Sarah Davis	An MES club executive striving to attract visitors by planning event logistics and executing them.
Direct	Sponsors/Promoters	Laura Nguyen	A marketing manager of a small business focused on promoting her company's brand and sponsoring an event.
Indirect	University Administration	Ahmed Patel	A University admin that is focused on ensuring the platform and the events within them comply with McMaster's financial and data policies.
Indirect	Capstone Development Team	Alex Kim	Developer responsible for developing the platform and integrating features from payment systems to RSVP signups to create a seamless, engaging user experience.

G.7.3 Additional Requirements Sources

- **User Feedback Surveys**: Surveys from users to provide insights into feature preference and areas for improvement.

- **University Policy Documents**: Guidelines on policies regarding accessibility standards and privacy, legal and financial compliance.
- **Industry Practices:**: Analysis of other event management platforms for design and feature inspiration and implementation.

(E) Environment

The Environment book describes the application domain and external context, physical or virtual (or a mix), in which the system will operate. [1]

Control Information

Table 2. MacSync — Versionning Information — Environment Book

Section	Version	Lead	Delivered	Reviewer	Approved
E.1	V1	Farhan & Mahad	2025-10-02	Ali & Farhan	2025-10-10
E.2	V1	Prerna	2025-10-09	Mahad	2025-10-10
E.3	V1	Abyan	2025-10-09	Prerna	2025-10-10
E.4	V1	Ali	2025-10-09	Abyan	2025-10-10
E.5	V1	Farhan	2025-10-02	Ali	2025-10-10
E.6	V1	Mahad	2025-10-09	Farhan	2025-10-10

(E.1) Glossary

Clear and precise definitions of all the vocabulary specific to the application domain, including technical terms, words from ordinary language used in a special meaning, and acronyms. It introduces the terminology of the project; not just of the environment in the strict sense, but of all its parts. [1]

(E.1.1) Vocabulary

MacSync

The name of the software application being developed. MacSync is a centralized event management platform designed for the McMaster Engineering Society (MES) to handle registrations, ticketing, payments, notifications, and event logistics.

User

Represents any person interacting with the system, either as an attendee or organizer.

Attendee

A student or guest who registers for and participates in an event. Attendees interact primarily with the mobile application to purchase tickets, complete registrations, and check in on event day.

Organizer

A member of the McMaster Engineering Society (MES) or a volunteer responsible for setting up and managing events. Organizers rely on the platform to handle registrations, ticketing, waivers, and check-ins efficiently.

MES (McMaster Engineering Society)

The student society overseeing large events, MES members act as stakeholders ensuring the system aligns with organizational needs and compliance requirements.

Event

Any MES hosted activity managed through the system. Events are associated with registrations, tickets, payments, and check-in data.

Registration

The process by which attendees submit details to join an event, including personal information, waivers, and preferences such as dietary or accessibility requirements.

Waiver

A required acknowledgment form (e.g., for liability or risk management) that attendees must complete prior to attending certain events. Stored digitally within the system.

Check-in

The process of validating an attendee's ticket at the entrance of an event.

Dashboard

The administrative web interface where organizers create events, manage ticket sales, view analytics, and track attendee data.

Bus Sign-Up

A specialized registration feature allowing attendees to reserve seats on transportation provided for an event.

Table Sign-Up

A feature allowing attendees to reserve tables for formals or similar events, ensuring balanced distribution and meeting capacity constraints.

RBAC (Role-Based Access Control) / FBAC (Feature-Based Access Control)

A permissions model that restricts what each user (e.g., attendee, organizer, executive) can access or modify within the system. This prevents errors and improves security by limiting privileges to the user's role.

Payment Integration

The system's connection to secure third-party services

(E.1.2) Domain Model

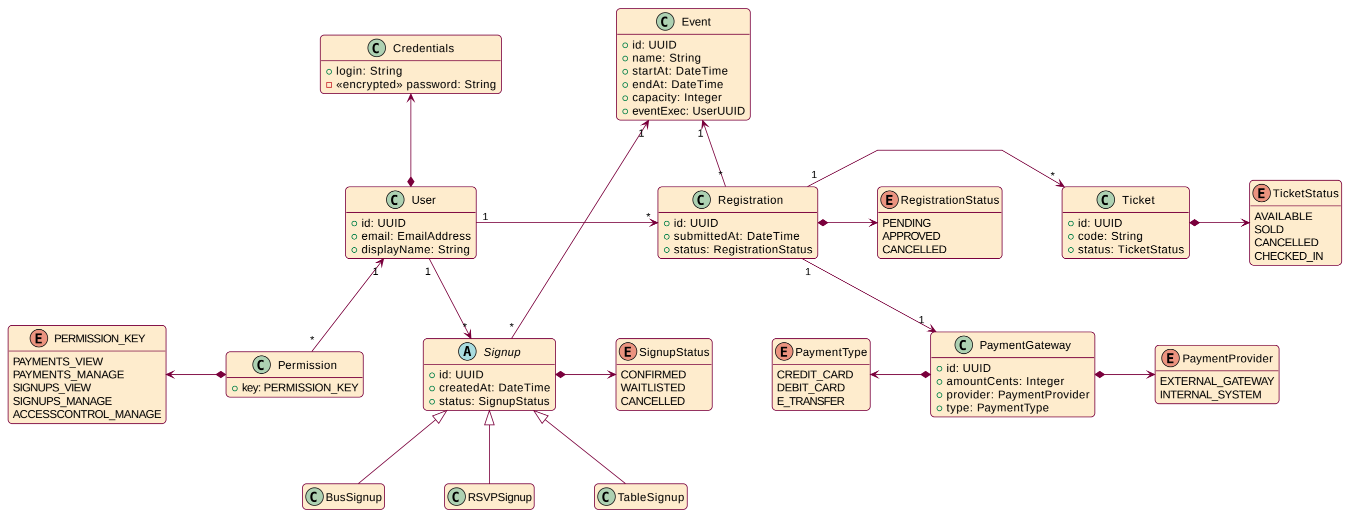

Figure 1. Domain Model for MacSync

(E.2) Components

List of elements of the environment that may affect or be affected by the system and project. It includes other systems to which the system must be interfaced. These components may include existing systems, particularly software systems, with which the system will interact — by using their APIs (program interfaces), or by providing APIs to them, or both. These are interfaces provided to the system from the outside world. They are distinct from both: interfaces provided by the system to the outside world (S.3); and technology elements that the system's development will require (P.5). [1]

The following external components represent the environment in which the Large Event Management System (LEMS) operates. These are systems and services with which LEMS must interface, or which constrain its operation:

- **University Authentication (McMaster SSO / MacID)** :: Provides secure authentication for both attendees and organizers. Ensures only valid students and authorized staff/volunteers can access the system.
- **Third-Party Payment Gateway Integration (Stripe, Square, PayPal)** :: Handle all financial transactions for ticket purchases and refunds. LEMS relies on their APIs for secure payment processing and confirmation.
- **Notification Services (e.g., Apple Push Notification Service, Google FCM for Android)** :: Used to deliver event reminders, confirmations, and updates to attendees' mobile devices.
- **Email Service Integration (SMTP provider or university-managed email infrastructure)** :: Provide a fallback for important notifications (e.g., payment confirmations, waiver receipts) if push notifications fail.
- **QR Code Libraries and Mobile Camera APIs** :: Provide functionality for generating, displaying, and scanning QR codes for event check-in.
- **Calendar Integration (Google Calendar, Apple Calendar, Microsoft Outlook)** :: Allows attendees to add event dates, reminders, and details directly into their personal calendars. LEMS must export event information in standard calendar formats (iCal/ICS) and synchronize updates where supported.

- **Device Ecosystem (iOS, Android, modern web browsers)** :: Represents the hardware/software environments in which the system must run reliably. Includes smartphones for attendees and laptops/desktops for organizers.

(E.3) Constraints

The development of the Large Event Management System (LEMS) is subject to a number of environmental and organizational constraints that must be observed throughout the project. These constraints are considered non-negotiable and define the limits within which the system must operate.

Business and Organizational Constraints:

- The system must comply with McMaster University policies for student organizations, including financial tracking and data management expectations.
- Payment processing must be handled through approved providers (Stripe) to ensure compliance with institutional and financial regulations.
- Data collected from participants must follow privacy and confidentiality rules, with secure handling of personal information and waivers.
- The system must remain maintainable beyond the current project team, supporting future MES groups with minimal retraining.

Technical and Infrastructure Constraints:

- The backend must use PostgreSQL as the database technology, as specified by project leadership.
- Hosting will be provided through DigitalOcean, managed by the MES, which limits available deployment platforms.
- Containerization with Docker will be used for database instances, and this approach must be maintained to simplify deployment and consistency across groups.
- The chosen technology stack (JavaScript, Next.js, React Native, NestJS, Drizzle ORM) must be consistently applied across groups to avoid integration issues.

Operational Constraints:

- The system must be able to support peak usage during major events such as the Fireball Formal or Graffiti Pub, where hundreds of concurrent sign-ups and check-ins may occur.
- Interfaces must be usable on both web and mobile platforms, as attendees will expect access from their own devices.
- The project must be completed within the academic term schedule, which limits available time for development, testing, and deployment.

These constraints ensure that the system respects institutional rules, technical choices, and practical limits of the development environment. They set the boundaries within which design decisions can be made.

(E.4) Assumptions

Properties of the environment that may be assumed, with the goal of facilitating the project and simplifying the system. It defines properties that are not imposed by the environment (like those in E.3) but assumed to hold, as an explicit decision meant to facilitate the system's construction. [1]

Some assumptions made can be found below:

- **Operating system support**: We assume our application only needs to support modern operating systems for both computer (Windows and MacOS) and mobile purposes (Android and iOS), meaning other systems are excluded from our scope. Additionally, we will support the relevant versions of the operating systems that are used by the majority of people and cover a large part of the target market.
- **Compliance Alignment**: We assume that all MES events that are posted on the platform comply with University and legal standards for student run events. The platform will not introduce any new regulatory safeguards.
- **Standardized payment infrastructure support**: We assume that the payment infrastructure provided by the selected payment provider is functional and reliable with APIs exposed for usage within the MES system.

(E.5) Effects

Elements and properties of the environment that the system will affect. It defines effects of the system's operations on properties of the environment. Where the previous two categories (E.3, E.4) defined influences of the environment on the system, effects are influences in the reverse direction. [1]

The introduction of the MacSync platform will have several direct effects on the environment in which it operates:

Improved Student Experience

Attendees gain a single, consistent interface for event discovery, registration, ticketing, and updates. This effect reduces confusion caused by separated tools and increases overall event participation.

Reduced Administrative Burden

Organizers will spend less time on manual tasks such as consolidating spreadsheets, managing waivers, or checking attendees in by hand. The platform streamlines these workflows, freeing volunteer time for event planning and quality improvements.

Financial Clarity

By integrating payment systems and providing dashboards, the platform enables real-time tracking of revenue and ticket sales. This enhances MES executives' ability to monitor finances and reduces risks tied to cash handling.

Inclusivity and Accessibility

Centralized collection of dietary and accessibility requirements ensures that attendees' needs are acknowledged and acted upon, improving inclusivity at MES events.

Community Engagement

Timely reminders, waitlist handling, and centralized communication increase the likelihood of students attending and staying connected with MES activities, strengthening the sense of community.

Volunteer Experience

The platform reduces stress during event execution (e.g., check-ins, capacity management) and creates a more positive environment for student organizers and volunteers, encouraging continued involvement in future years.

(E.6) Invariants

Properties of the environment that the system's operation must preserve, i.e., properties of the environment that operations of the system may assume to hold when they start, and must maintain. [1]

Properties of the environment that the system's operation must preserve will be assumed as follows:

- MacSync will assume that all events listed on the MacSync platform has an MES exectutive owner of record that is either (a) hosting the event or (b) the supervising/approving authority of the event. This ensures that all events listed on MacSync are MES sanctioned events.
- MacSync will assume that all students registering for events will have a valid and unique MacID. This means that all registered MacSync users will be associated with their unique MacID to ensure identity verification and prevent duplicate accounts.
- MacSync will assume that the `event-capacity` entered upon creating events is the single source of truth for event capacity. At all times: `registered-students ▯ event-capacity` must hold. This ensures that tickets are not oversold for events with limited capacity.

(S) System

the System book refines the Goal one by focusing on more detailed requirements.

Control Information

Section	Version	Lead	Delivered	Reviewer	Approved
S.1	V1	Prerna	2025-10-09	Mahad	2025-10-10
S.2	V1	Abyan	2025-10-09	Prerna	2025-10-10
S.3	V1	Ali	2025-10-09	Abyan	2025-10-10
S.4	V1	Farhan	2025-10-09	Ali	2025-10-10
S.5	V1	Mahad	2025-10-09	Farhan	2025-10-10
S.6	V1	Prerna & Abyan	2025-10-09	Mahad & Prerna	2025-10-10

(S.1) Components

Overall structure expressed by the list of major software and, if applicable, hardware parts. [1]

The system relies on the following **software components**:

- **Registration & Ticketing Module** :: Provides functionality for attendees to view upcoming events, purchase tickets, join waitlists, and confirm registration.
- **Payment Processing Module** :: Interfaces with third-party gateways (Stripe, Square, PayPal) to process transactions securely and issue confirmations.
- **Waiver & Preference Module** :: Collects required liability waivers and optional attendee information (dietary restrictions, accessibility needs).
- **Bus/Table/RSVP Module** :: Manages sign-ups for buses, table seating, and session RSVPs, enforcing capacity limits and fairness in assignment.
- **Check-In Module** :: Validates digital tickets via QR/Barcode scanning, including offline validation support for unreliable connectivity.
- **Notification & Reminder Module** :: Sends push notifications, confirmations, and event reminders through integrated services (e.g., Calendar Integration).
- **Admin Dashboard & Analytics Module** :: Provides organizers with an interface to monitor registrations, ticket sales, waitlists, finances, and demographic data.
- **Role-Based / Feature-Based Access Control Module (RBAC/FBAC)** :: Ensures organizers only access features and data relevant to their role.
- **Database Management System** :: Stores user accounts, ticketing data, waiver records, payment confirmations, and system logs.
- **Audit Logging Subsystem** :: Records all administrative actions (refunds, seat assignments, event

edits) for accountability and traceability.

The system relies on the following **hardware components**:

- **Mobile Devices (iOS/Android smartphones)** :: Used by attendees to register, purchase tickets, receive notifications, and check in to events.
- **Laptops/Desktops (Organizers)** :: Used by MES volunteers and executives to access the admin dashboard, manage events, and perform analytics.
- **QR Code Scanners (Smartphone Cameras or Organizer Devices)** :: Used at event entrances to validate attendee tickets efficiently.

Together, these components define the operational structure of the system and highlight the dependencies critical for secure and reliable operation.

(S.2) Functionality

(S.2.1) Registration & Ticketing Module

Functional Requirements

1. **Event browsing and registration:** The system shall allow users to view upcoming MES events and register by purchasing tickets. (F101)
2. **Ticket purchasing and invoicing:** Each transaction shall generate a digital ticket and invoice for the attendee, recorded in the system database. (F102)
3. **Waitlist management:** When an event reaches capacity, the system shall automatically queue users on a waitlist and notify them when a slot becomes available. (F103)
4. **Cancellation and refunds:** Users shall be able to cancel registrations within the organizer-defined window, and the system shall trigger refunds when applicable. (F104)

Non-Functional Requirements

1. **Performance:** The registration and ticketing operations shall complete within 2 seconds under normal load. (NF101)
2. **Data integrity:** The system shall prevent overselling tickets by using atomic database transactions. (NF102)
3. **Usability:** The registration interface shall provide clear feedback messages for successful and failed transactions. (NF103)

(S.2.2) Payment Processing Module

Functional Requirements

1. **Payment gateway integration:** The backend shall integrate with a payment gateway API for all credit/debit card transactions. (F105)
2. **Transaction verification:** The system shall confirm successful payment and record transaction details in PostgreSQL. (F106)
3. **Refund processing:** The system shall synchronize refund actions initiated through the admin

dashboard or payment gateway interface. (F107)

Non-Functional Requirements

1. **Security compliance:** All payment data shall comply with PCI-DSS and be encrypted with TLS 1.3 during transmission. (NF104)
2. **Reliability:** Payment services shall maintain 99 percent uptime during registration periods. (NF105)
3. **Auditability:** All payment events shall be logged with timestamps and user identifiers for reconciliation. (NF106)

(S.2.3) Bus / RSVP / Table Sign-ups Module

Functional Requirements

1. **Capacity management:** The system shall display real-time seat availability and enforce maximum capacity limits. (F108)
2. **First-come-first-serve allocation:** When multiple users attempt to sign up simultaneously, the system shall assign spots based on request order. (F109)
3. **Reservation editing:** Users shall be able to modify or cancel sign-ups prior to organizer-defined deadlines. (F110)

Non-Functional Requirements

1. **Fairness:** The system shall guarantee equitable allocation by locking contested seats during concurrent requests. (NF107)
2. **Performance:** Reservation confirmations shall occur within 1 second of submission. (NF108)
3. **Consistency:** All updates shall propagate across related tables in under 3 seconds. (NF109)

(S.2.4) Role / Feature-Based Access Control (RBAC/FBAC)

Functional Requirements

1. **Role definition:** Administrators shall create and assign predefined roles such as student, organizer, and executive. (F111)
2. **Feature access restriction:** The system shall expose or hide functions according to each role’s privileges. (F112)
3. **Privilege modification:** Admin users shall update role permissions through a secure dashboard interface. (F113)

Non-Functional Requirements

1. **Security:** Access control logic shall follow the principle of least privilege. (NF110)
2. **Maintainability:** Role definitions shall be stored in configuration files editable without code changes. (NF111)
3. **Traceability:** All privilege changes shall be logged for audit review. (NF112)

(S.2.5) Privacy, Data Integrity, and Security Controls

Functional Requirements

1. **Data encryption:** The system shall encrypt personal and financial information both in transit and at rest. (F114)
2. **Authentication and authorization:** Users shall authenticate with institutional credentials, and sessions shall expire after inactivity. (F115)
3. **Regulatory compliance:** The system shall comply with PIPEDA and institutional data protection policies. (F116)

Non-Functional Requirements

1. **Confidentiality:** Only authorized services shall access sensitive data fields. (NF113)
2. **Integrity:** Data modification operations shall use hashing or checksums to detect tampering. (NF114)
3. **Availability:** Security and privacy services shall maintain 99 percent uptime and recover automatically after failure. (NF115)

(S.3) Interfaces

How the system makes the functionality of S.2 available to the rest of the world, particularly user interfaces and program interfaces (APIs). It specifies how that functionality will be made available to the rest of the world, including people (users) and other systems. These are interfaces provided by the system to the outside; the other way around, interfaces from other systems, which the system may use, are specified in E.2. [1]

The MES system offers several external interfaces, which facilitate interaction with both users and external systems. This section details the external APIs provided by each core component of the system. These APIs enable key functionalities for the platform such as user navigation, payment system interaction, access control, event management and sign-ups.

(S.3.1) User Interface API

Table 3. API Endpoints for User Interaction

Endpoint	Method	Description
/dashboard/user	GET	Retrieves information related to upcoming events, the events a user signed up to, and current sign-up status on those events.
/dashboard/admin	GET	Retrieves admin information related to event metrics, role details, and consolidates information on events they have set up from signups to payment information.

The User Interface API enables engagement through core functions such as giving admins a dashboard to view their event details and also giving users a dashboard to keep track of events going on, signed up events, and user profile details.

(S.3.2) Payment System API

Table 4. API Endpoints for Payment System Services

Endpoint	Method	Description
/payments	POST	Initiates a new payment transaction for ticket purchases, providing users the ability to put in their payment details.
/payments/{transactionID}	GET	Retrieves all user payment details.
/payments/{transactionID}/confirm	POST	Allows users to confirm their payment transaction.
/payments/{transactionID}/refund	POST	Allows users to submit a refund request and process them.
/payments/{transactionID}/status	GET	Retrieves current status of the payment transaction made by the user (pending, complete, or failed).

The Payment System API leverages external payment gateways to manage all financial interactions within the platform related to tickets, transactions and payment status. The endpoints allow smooth and secure handling of data to ensure a transparent user experience.

(S.3.3) Role Based Access Control / Feature Based Access Control API

Table 5. API Endpoints for Access Controls

Endpoint	Method	Description
/auth/login	POST	Authenticates user credentials, managing user authentication to ensure secure access to accounts.
/auth/logout	POST	Ends user session and clears sensitive session data.
/roles/permissions/{userId}	GET	Retrieves access rights granted to a specific user.
/roles/permissions/{userId}	PUT	Updates a user’s role and feature-based access.

The Role Based Access Control / Feature Based Access Control API is critical for determining the type of access rights for each user type, governing authorization across the platform so that users can only access features they are permitted to as well as allowing admins to assign access.

(S.3.4) Event Management System API

Table 6. API Endpoints for Event Management System

Endpoint	Method	Description
/events	POST	Allows admins to create an event with details and content.
/events	GET	Retrieves a list of all events created by an admin.
/events/{eventId}	GET	Allows admin to retrieve all content details for a specific event.
/events/{eventId}	PUT	Allows admin to update existing event details and content.
/events/{eventId}	DELETE	Allows admin to delete existing event details and content.

The Event Management System API gives administrators the ability to manage, create and update events so that they are able to have their events available for users to see and also ensure content remains relevant and up to date.

(S.3.5) Sign-Ups API

Table 7. API Endpoints for Sign-Ups System

Endpoint	Method	Description
/signups/list	GET	Allows users to see a list of all signups available on events.
/signups/{signupId}/rsvp	POST	Allows users to RSVP for a specific event.
/signups/{signupId}/bus	POST	Allows users to reserve seats on a bus for a specific event.
/signups/{signupId}/table	POST	Allows users to reserve seats on a table for a specific event.
/signups/{signupId}/cancel	DELETE	Cancels user’s signup to a specific event.
/signups/{signupId}/status	GET	Retrieves status of a specific event a user signed up to (waitlist or confirmed).

The Sign-Ups API allows event attendees to see all event signups, sign up for specific events by confirming details about rsvp, buses or tables, see their status on a specific event and also cancel their signup. The API manages capacity limits, waitlists and provides real-time information helpful to both users and admins.

(S.4) Detailed usage scenarios

Examples of interaction between the environment (or human users) and the system, expressed as user stories. Such scenarios are not by themselves a substitute for precise descriptions of functionality (S.3), but provide an important complement by specifying cases that these behavior descriptions must support; they also serve as a basis for developing test cases. The scenarios most relevant for stakeholders are given in chapter G.5 in the Goals book, at a general level, as use cases; in contrast, S.4 can refer to system components and functionality (from other chapters of the System book) as well as special and erroneous cases, and introduce more specific scenarios. [1]

(S.4.1) Register for an Event

- **Primary Actor**: Student User
- **Supporting Actors**: MES Event Organizers, Payment Gateway (Stripe)
- **Precondition**:
  - User has a valid MacID and is logged into MacSync.
  - At least one active event is published by MES organizers.
- **Trigger**: User selects an event and presses “Register”.
- **Main Success Scenario**:
  1. System displays event details (tickets, capacity, bus/table options, waivers).
  2. User selects ticket type and fills required details (dietary, accessibility).
  3. System validates availability.
  4. User proceeds to payment if required.
  5. System sends payment request to Stripe and receives confirmation.
  6. System confirms registration and issues a digital ticket (QR code).
  7. User receives confirmation via email and in-app notification.
- **Secondary Scenarios**:
  - If capacity is full, user is placed on a waitlist and notified of position.
  - If payment fails, system displays an error and allows retry.
- **Success Postcondition**: User is successfully registered, payment recorded, ticket issued, and capacity updated.

(S.4.2) Join Bus, Table, or RSVP Sign-Up

- **Primary Actor**: Student User
- **Supporting Actors**: Event Organizers, System (Seat Allocator)
- **Precondition**:
  - User is registered for the event.
  - Bus/table or RSVP sign-up is enabled for that event.
- **Trigger**: User navigates to the “Logistics” tab and selects an available option.
- **Main Success Scenario**:
  1. System displays available buses, tables, or RSVP options along with remaining capacity.
  2. User selects their preferred option.

3. System locks the slot and verifies availability atomically to prevent overbooking.
4. System records the assignment or RSVP selection and updates capacity in real time.
5. User receives confirmation and can view their assigned group, table, or RSVP status.

- **Secondary Scenarios**:
  - If a slot or RSVP limit is reached during selection, the system notifies the user and displays remaining options.
  - If the user changes their selection, the system allows switching until the event's cutoff time.
  - If RSVP confirmation is required, the system sends a reminder notification before the confirmation deadline.
- **Success Postcondition**: User's bus, table, or RSVP selection is stored and visible to organizers for event logistics and attendance planning.

(S.4.3) Check-In at Event

- **Primary Actor**: Event Volunteer / Organizer
- **Supporting Actors**: Student User, System (Scanner Module)
- **Precondition**:
  - Event is active.
  - User has a valid digital ticket (QR code).
- **Trigger**: Volunteer scans QR code using the MacSync Check-In interface.
- **Main Success Scenario**:
  1. System validates ticket and marks attendee as "Checked-In".
  2. System displays attendee details (name, bus/table, dietary info).
  3. Volunteer confirms entry and system records timestamp.
  4. Attendance dashboard updates in real time.
- **Secondary Scenarios**:
  - If ticket is invalid or already used, system displays an error with reason.
- **Success Postcondition**: User is marked as checked in and attendance data updates in admin dashboard.

(S.4.4) Create and Manage Events (Organizer/Admin)

- **Primary Actor**: MES Organizer / Admin
- **Supporting Actors**: MES Executives (for approval), System (Admin Portal)
- **Precondition**:
  - Organizer has admin permissions.
- **Trigger**: Organizer clicks "Create Event" in the Admin Dashboard.
- **Main Success Scenario**:
  1. Organizer enters event details (title, date, capacity, ticket types, pricing, dietary/accessibility requirements).
  2. System validates required fields and saves as draft.
  3. Organizer adds optional bus/table sign-ups and uploads waiver template.
  4. Organizer can later edit event details or schedule notifications.
- **Success Postcondition**: Event is successfully created, reviewed, and published with specific event details and requirements.

(S.4.5) Monitor Registrations and Analytics (Organizer/Executive)

- **Primary Actor**: MES Organizer or Executive
- **Supporting Actors**: System (Analytics Module)
- **Precondition**:
  - User has access to the Analytics Dashboard.
- **Trigger**: Organizer or Executive navigates to the dashboard to monitor events.
- **Main Success Scenario**:
  1. System displays key performance metrics, total registrations, waitlist size, check-ins, and engagement trends.
  2. Organizer filters and compares data across events or time periods.
  3. System visualizes participation patterns (e.g. peak registration times, most popular events, demographic breakdowns).
  4. Organizers and executives use these insights to inform future event planning, capacity management, and communication strategies.
- **Secondary Scenarios**:
  - If non-privileged user attempts access, system denies with an error message.
- **Success Postcondition**: Organizers and executives gain actionable insights to improve future event design, attendee engagement, and overall MES event operations.

(S.4.6) Handle Waitlist Updates

- **Primary Actor**: System
- **Supporting Actors**: Student Users, Organizers
- **Precondition**:
  - Event has a waitlist.
  - A registered attendee cancels or refund is processed.
- **Trigger**: System detects available slot.
- **Main Success Scenario**:
  1. System identifies next user in waitlist queue.
  2. System notifies user of waitlist update and provides payment window.
  3. Upon payment confirmation, user is registered and removed from waitlist.
  4. If no response within window, next user is automatically notified.
- **Success Postcondition**: Event capacity remains consistent, and unused spots are filled efficiently.

(S.5) Prioritization

Classification of the behaviors, interfaces and scenarios (S.2, S.3 and S.4) by their degree of criticality. It is useful in particular if during the course of the project various pressures force the team to drop certain functions. [1]

(S.5.1) Priority Table for Event Registration and Ticketing

Requirement ID	Requirement Name	Priority Level	Reasoning
[F101]	Event browsing and registration	High (Must have)	Core functionality for users, must be able to view events and register to participate.
[F102]	Ticket purchasing and invoicing	High (Must have)	Essential for validating registrations and enabling payment records.
[F103]	Waitlist management	Medium (Should have)	Improves user experience but not critical for initial MVP launch.
[F104]	Cancellation and refunds	Medium (Should have)	Important for flexibility and user trust, but can be introduced after MVP.
[NF101]	Performance	High (Must have)	Directly impacts user satisfaction and perceived reliability.
[NF102]	Data integrity	High (Must have)	Prevents overbooking and financial inconsistencies.
[NF103]	Usability	Medium (Should have)	Improves UX clarity, desirable but not mission-critical.

(S.5.2) Priority Table for Payment Processing Module

Requirement ID	Requirement Name	Priority Level	Reasoning
[F105]	Payment gateway integration	High (Must have)	Necessary to handle secure credit/debit payments.
[F106]	Transaction verification	High (Must have)	Ensures payment confirmation and successful transactions.
[F107]	Refund processing	Medium (Should have)	Important for user trust, but can come after MVP.
[NF104]	Security compliance	High (Must have)	PCI-DSS compliance is mandatory for financial data handling.
[NF105]	Reliability	High (Must have)	Payment service must remain stable during registration spikes.

Requirement ID	Requirement Name	Priority Level	Reasoning
[NF106]	Auditability	Medium (Should have)	Important for financial transparency, but can be implemented after MVP.

(S.5.3) Priority Table for Bus / RSVP / Table Sign-ups Module

Requirement ID	Requirement Name	Priority Level	Reasoning
[F108]	Capacity management	High (Must have)	Core logic preventing overbooking.
[F109]	First-come-first-serve allocation	High (Must have)	Necessary for fair sign-up behavior and user satisfaction.
[F110]	Reservation editing	Medium (Should have)	Improves flexibility and user experience, but can come after MVP.
[NF107]	Fairness	High (Must have)	Ensures fair seat assignment under many sign ups at the same time.
[NF108]	Performance	High (Must have)	Real-time sign-ups must complete quickly for usability.
[NF109]	Consistency	High (Must have)	Prevents data mismatches across connected modules (tickets, buses, tables).

(S.5.4) Priority Table for Role / Feature-Based Access Control (RBAC/FBAC)

Requirement ID	Requirement Name	Priority Level	Reasoning
[F111]	Role definition	High (Must have)	Foundational for user seperation and roles.
[F112]	Feature access restriction	High (Must have)	Required to enforce permissions and data isolation.
[F113]	Privilege modification	Medium (Should have)	Improves administrative flexibility, can come after MVP.

Requirement ID	Requirement Name	Priority Level	Reasoning
[NF110]	Security	High (Must have)	Enforces least-privilege access model, critical for compliance.
[NF111]	Maintainability	Medium (Should have)	Simplifies updates, but not immediately essential for release.
[NF112]	Traceability	Medium (Should have)	Adds audit transparency which can help with transparency.

(S.5.5) Priority Table for Privacy, Data Integrity, and Security Controls

Requirement ID	Requirement Name	Priority Level	Reasoning
[F114]	Data encryption	High (Must have)	Core to protecting user and financial data.
[F115]	Authentication and authorization	High (Must have)	Essential to secure user sessions and McMaster access.
[F116]	Regulatory compliance	High (Must have)	Mandatory McMaster and legal policies (PIPEDA).
[NF113]	Confidentiality	High (Must have)	Prevents unauthorized access to sensitive data.
[NF114]	Integrity	High (Must have)	Ensures that financial and registration records remain consistent.
[NF115]	Availability	High (Must have)	Guarantees system uptime during registration.

(S.5.2) Prioritization Analysis

The prioritization reflects the systems core components such as signing up for events, processing payments and role base access system. Furthermore, it emphasizes security and reliability to prevent sensitive user data. The Medium priority items focus on nice-to-have features that will optimize the system but are not essential for the MVP itself.

(S.6) Verification and Acceptance Criteria

Validation of Functional Requirements

To validate the functional requirements of the MacSync system, the team will perform module, integration, system, and user acceptance testing (UAT).

- **Module testing** will verify that each component listed in S.1 performs its defined behavior independently.
- **Integration testing** will ensure seamless communication between frontend and backend modules.
- **System testing** will confirm end-to-end behavior, focusing on registration, payment, and analytics workflows.
- **User acceptance testing** will involve MES stakeholders performing real registration scenarios to confirm usability and correctness.

Fit Criteria for Non-Functional Requirements

- **Performance:** Core actions such as registration or payment confirmation shall respond within **2 seconds** under normal load.
- **Scalability:** The backend shall handle **500 concurrent users** without more than **10 percent** performance degradation.
- **Reliability:** Overall system uptime shall be **99 percent** during major event periods.
- **Security:** All user and payment data shall be encrypted with TLS 1.3 and stored following PCI-DSS and PIPEDA compliance.
- **Usability:** The registration process shall require **no more than three steps** and yield a **90 percent task-success rate** in UAT.
- **Notification timeliness:** Confirmation emails and in-app messages shall be delivered within **60 seconds** of completed registration.

Selected Scenario for Testing

The scenario **Register for an Event** (S.4) is selected as the primary validation case since it exercises the full registration and payment workflow.

Proposed Tests for "Register for an Event"

1. **Registration Workflow Test**
   - Objective: Verify that users can complete event registration from selection to confirmation.
   - Procedure: Simulate a logged-in user selecting an event, filling the form, and completing payment.
   - Expected Outcome: The system confirms registration, issues a QR code ticket, and sends notifications.
2. **Payment Processing Test**
   - Objective: Validate correct Stripe integration and recording of payment confirmations.
   - Procedure: Execute sandbox transactions and verify backend receipt of success callbacks.
   - Expected Outcome: Transaction details are saved and ticket marked “Paid”.
3. **Waitlist Handling Test**
   - Objective: Ensure users are queued when capacity is reached.
   - Procedure: Attempt registration after capacity is full.

    - Expected Outcome: User is added to waitlist and notified of position.
4. **Error and Retry Test**
    - Objective: Confirm proper recovery from payment or connection failures.
    - Procedure: Interrupt payment midway and retry.
    - Expected Outcome: Clear retry message; no duplicate transaction recorded.
5. **Notification Delivery Test**
    - Objective: Confirm that users receive confirmation messages promptly.
    - Procedure: Check email logs and in-app notification queue post-registration.
    - Expected Outcome: Notifications delivered within 60 seconds.

Traceability Matrix

Test Case ID	Requirement ID	Requirement Description	Status
TC-01	**F101**	System shall allow users to register for events and generate tickets.	Not Started
TC-02	**F105**	System shall integrate with payment gateway for secure payment processing.	Not Started
TC-03	**F103**	System shall manage user waitlists when event capacity is reached.	Not Started
TC-04	**NF102**	System shall ensure data consistency to prevent overselling tickets.	Not Started
TC-05	**NF104**	System shall encrypt payment and personal data following PCI-DSS.	Not Started

The verification and acceptance process for the Large Event Management System (LEMS) defines how both functional and non-functional requirements will be validated and accepted. Verification confirms that the system has been built correctly according to specifications, while validation confirms that the right system has been built to meet MES organizers’ and attendees’ needs.

(P) Project

Control Information

Section	Version	Lead	Delivered	Reviewer	Approved
P.1	V1	Ali	2025-10-09	Abyan	2025-10-10
P.2	V1	Farhan	2025-10-09	Ali	2025-10-10
P.3	V1	Mahad	2025-10-09	Farhan	2025-10-10
P.4	V1	Prerna	2025-10-09	Mahad	2025-10-10
P.5	V1	Abyan	2025-10-09	Prerna	2025-10-10
P.6	V1	Ali & Farhan	2025-10-09	Abyan & Ali	2025-10-10
P.7	V1	Mahad	2025-10-09	Farhan	2025-10-10

(P.1) Roles and personnel

Main responsibilities in the project; required project staff and their needed qualifications. It defines the roles (as a human responsibility) involved in the project. [1]

- **Security Expert**: The responsibility of the security expert entails working to define security requirements and policies that hinge on the CIA triad (Confidentiality, Integrity, Availability) as well as immobilization of threats when they arise which would require them to have prior experience in dealing with the four main types of threats: disclosure, deception, disruption and usurpation. The qualifications of the security expert will typically include a bachelor's degree in a field such as Computer Science, Information Security or Cybersecurity as well as relevant personal or work experience in a relevant field in security.
- **Product Owner**: The product owner is responsible for representing the interest of the customer and market which entails responsibility for the overall product. The encompassing responsibilities range from organization of product backlog of features, ensuring alignment with company goals and vision to understanding stakeholder needs and requirements. The qualifications of the product owner will typically involve a Bachelor's degree in either a technical field related to Engineering, Computer Science or a business field such as Management or Business Administration as well as strong understanding of meeting client and stakeholder needs.
- **Developers**: The responsibility of developers revolves around the development of various technical components and core functionalities of the project on the front end and backend. The qualifications of developers typically will include a Bachelor's degree in Computer Science, Engineering, or equivalent fields. It will also generally require experience in application development, payment integration as well as modern frameworks for both mobile and web application development.
- **Quality Testers (Quality Engineers/Quality Assurance)**: The quality testers are responsible for ensuring system reliability of the product through testing and validation. Their focus is on technical aspects of the project to ensure alignment with defined quality standards, requirements and

specifications. The qualifications of quality testers typically includes a Bachelor's degree in Computer Science, Engineering, or equivalent fields and generally requires experience with modern testing frameworks and tools. Specifically, having an understanding of defining, designing and implementing test cases and test plans based on requirements.

(P.2) Imposed technical choices

This section identifies the technical and organizational constraints imposed on the MacSync project. These choices are defined a priori and are binding for all capstone teams collaborating on the system. They have been selected either for consistency across teams, integration with McMaster University infrastructure, or alignment with MES deployment requirements.

- **Common Technology Stack**:
    - All capstone teams developing the MacSync system will adhere to a unified technology stack to ensure cross-team compatibility and simplified integration. This decision was made collaboratively across project groups to enable shared components to operate within a consistent environment.
- **Frontend Framework**:
    - The frontend is implemented using Vite for fast builds and hot module reloading, alongside TanStack Router and TanStack Create for type-safe, modular state management. TypeScript is used to enhance maintainability and code safety.
    - These technologies replace an earlier consideration of Next.js, which was found to be restrictive for a single-page application (SPA) architecture and less suitable for dynamic, role-based dashboards that rely on client-side routing and real-time state management. The migration to Vite and TanStack improves flexibility, developer efficiency, and integration with the external NestJS backend.
- **Backend Framework**:
    - The backend is implemented using NestJS, chosen for its strong TypeScript support, modular structure, and built-in dependency injection.
    - Each capstone team develops independent backend modules that integrate into a larger unified repository. This structure allows for separation of subsystem responsibilities (e.g., event management, registration, analytics) while maintaining consistent API conventions across the entire MacSync platform.
    - RESTful APIs are used to facilitate straightforward communication between the frontend and backend.
- **Database and ORM**:
    - PostgreSQL is selected as the system's relational database due to its reliability, scalability, and strong community support. It provides a mature and well-documented foundation for event registration, ticketing, and analytics data. The data model in MacSync including users, events, tickets, waitlists, and logistics, naturally follows a relational structure with clear entity relationships and constraints, making PostgreSQL a suitable and logical choice.
    - The database will be self-managed through Docker containers during development to allow local testing and consistent environments across all project teams. In production, MES will deploy the database within its DigitalOcean infrastructure.
    - DrizzleORM is used for type-safe database operations and schema management within the

TypeScript ecosystem. This modern ORM is preferred over older or heavier frameworks such as TypeORM or Prisma, offering improved performance and closer integration with the chosen NestJS backend.
    - The decision to use a single ORM across all capstone groups was made to ensure unified migration handling and compatibility between independently developed modules. While multiple ORM options were technically possible, standardizing on DrizzleORM simplifies collaboration and guarantees consistent database migrations across the shared PostgreSQL schema.
    - These decisions were made after discussion with the project supervisor and other groups, prioritizing modern tooling, maintainability, and ease of integration within a multi-team development environment.
- **Payment Integration**:
    - Stripe is designated as the official payment gateway for ticket sales and event transactions.
    - This ensures that the system develops PCI-compliant handling of payment data and supports future MES integration with other financial systems.
- **Notifications**:
    - The system will include support for both in-app notifications and automated email messages.
    - These services are used for event reminders, registration confirmations, and administrative updates to organizers and attendees. The exact implementation of the notification provider may vary, but message delivery will be unified across teams to ensure consistency.
- **Hosting and Deployment**:
    - Final deployment and hosting responsibilities will be managed by the McMaster Engineering Society (MES) using DigitalOcean infrastructure.
    - While this configuration is outside the current capstone development scope, all project modules are being designed with containerization and portability in mind to facilitate later deployment by MES staff.
- **Version Control and Continuous Integration**:
    - All project codebases are maintained in private GitHub repositories.
    - GitHub is used for version control, issue tracking, and coordination between teams.
    - Continuous integration workflows are configured to ensure automated builds and documentation generation. This provides consistent quality assurance across repositories and prevents environment inconsistencies between teams.

These technical choices collectively ensure interoperability, maintainability, and compliance with university policies while aligning the MacSync project with MES's operational and deployment requirements. They also ensure that the system meets its technical requirements while maintaining consistency and compatibility across all collaborating capstone teams.

(P.3) Schedule and milestones

List of tasks to be carried out and their scheduling. It defines the project's key dates. [1]

(P.3) Schedule and milestones

List of tasks to be carried out and their scheduling. It defines the project's key dates. [1]

P.3.1 Sprint Plan

Sprint 1: Core Foundations & Architecture (October 13 – October 31)

- **Epics:**
  - Establish development environment (CI/CD, database schemas).
  - Implement initial UI mockups and API skeleton for proof of concept demo.
  - Plan modules for Payments, RBAC/FBAC, and Sign-Up Systems.
  - Add basic user authentication.
- **Deliverables:**
  - Repository steup for build/deploy pipeline.
  - Database schema for Users, Permissions, and Payment Transactions.
  - Prototype login and role access demo.

Sprint 2: RBAC + Sign-Up Modules / PoC Demo (Nov 1 – Nov 30)

- **Epics:**
  - Implement Mac SSO authentication.
  - Add role-based access controls.
  - Develop Bus/Table/RSVP Sign-Up system with capacity validation.
  - Link Sign-Ups to user IDs and organizer dashboards.
- **Deliverables:**
  - Functional RBAC/FBAC system.
  - Working Sign-Up frontend + backend.
  - Mac SSO.

Sprint 3: Payment System Integration (Dec 1 – Dec 31)

- **Epics:**
  - Implement Payment Gateway.
  - Integrate payments with existing registration and sign-up records..
- **Deliverables:**
  - Working Payment Gateway service (MVP).
  - Payment transaction logs and admin verification dashboard.

Sprint 4: Integration & Deployment (Jan 1 – Jan 15)

- **Epics:**
  - Integrate Payments, RBAC, and Sign-Ups into a single deployable MVP (with other groups of MES)
  - Add notification and calendar sync features.
  - Conduct final V&V and security testing.
  - Deploy to McMaster users
- **Deliverables:**
  - Integrated MVP platform deployed

At the end of sprint 2 we will have developed the core features for the sign-ups and login. With these 2 features we will be able to show our stakeholders a proof of concept demo to get early feedback for further development. By sprint 4 we will have developed the payment gateway and integrating all the components together to produce a MVP to be tested and deployed. This will be an early access application to get feedback from users what they think of the app and how we can further the event registration experience.

(P.4) Tasks and deliverables

> ***This is the core of the Project book**. It details the individual tasks listed under P.3 and their expected outcomes. It define the project's main activities and the results they must produce, associated with the milestone dates defined in P.3.* [1]

The following table outlines the detailed tasks and deliverables for the Large Event Management System (LEMS) – Project B. Each task corresponds to a major system feature, deliverable, or refinement cycle defined in the MES Project Scope and Capstone Calendar. Feature development will occur progressively between October and December 2025, with all core modules integrated and operational by January 2026 (Winter Term, Week 1) for MES internal testing. Final reports, presentations, and documentation will continue through April 2026.

Task ID	Description	Deliverable / Expected Outcome	Milestone / Due Date
T1	Problem Statement and Project Scoping	Initial Problem Statement and Scoping Document outlining project goals, stakeholders, and initial requirements for Payment, RBAC/FBAC, and Sign-Up modules.	**Week 4 – Sep 2025**
T2	Hazard Analysis and SRS	Completed Hazard Analysis and Software Requirements Specification (SRS) defining functional, safety, and security requirements for all system components.	**Week 6 – Oct 2025**
T3	Verification & Validation (V&V) Plan and Proof of Concept Setup	V&V Plan and preliminary Proof of Concept environment showing early registration and UI flow integration.	**Week 8 – Oct 2025**
T4	System Architecture and Design	Design Document Rev 1 with finalized data models, interface diagrams, and module-level designs for Payments, RBAC/FBAC, and Sign-Up systems.	**Week 10 – Nov 2025**
T5	Payment System Implementation	Development of secure Payment Processing using Stripe, Square, and PayPal APIs. Includes transaction confirmation, duplicate protection, and error-handling features.	**End of Oct 2025**

Task ID	Description	Deliverable / Expected Outcome	Milestone / Due Date
T6	Role-Based and Feature-Based Access Control (RBAC/FBAC)	Implementation of user authentication (via McMaster SSO) and access control to restrict organizer/admin privileges by role and function.	**Mid-Nov 2025**
T7	Registration, Waiver, and Preference Module	Completion of user registration workflow, waiver submission, and attendee preference collection (dietary, accessibility).	**Late Nov 2025**
T8	Bus/Table/RSVP Sign-Up Module	Implementation of dynamic seat/bus/table selection system with real-time capacity validation and confirmation notifications.	**Early Dec 2025**
T9	Notification and Reminder System	Integration of push/email notifications via Firebase Cloud Messaging (FCM) and fallback email system. Includes automated event reminders and updates.	**Mid-Dec 2025**
T10	Calendar Integration	Implementation of calendar export and synchronization (Google, Apple, Outlook) allowing attendees to add event dates directly to personal calendars.	**Late Dec 2025**
T11	Core System Integration (MVP Release)	Integration of all modules (Payments, RBAC, Registration, Sign-Ups, Notifications) into a unified platform for MES testing.	**Week 16 – Jan 2026 (Winter Term, Week 1)**
T12	User Testing and Feedback Collection	Internal MES pilot testing using live event data. Collect and log user feedback for iterative refinement.	**Weeks 17–18 – Feb 2026**
T13	Refinement Cycle 1	Refinements based on feedback, including UI/UX adjustments, bug fixes, and performance optimization.	**Week 20 – Feb 2026**
T14	Verification & Validation Testing	V&V Report confirming compliance with functional, safety, and security requirements defined in the SRS.	**Week 22 – Mar 2026**
T15	Refinement Cycle 2 and Final Optimization	Second round of refinements based on testing results and organizer feedback. Enhancements to notification reliability, analytics, and accessibility.	**Week 23 – Mar 2026**
T16	Final Demonstration and Handover	Final Capstone Demo and Handover Package, including full documentation, deployment guide, and repository for future MES development.	**Week 24 – Mar 2026**

Task ID	Description	Deliverable / Expected Outcome	Milestone / Due Date
T17	Poster and Final Deliverables Submission	Poster PDF, Final Documentation Package, and Demo Video prepared and submitted as part of Capstone final evaluation.	**Week 25 – Apr 2026**
T18	Capstone Expo and Final Presentation	Capstone Expo Presentation and project showcase. Completion of course evaluation and public demonstration of final system.	**Week 26 – Apr 2026**

Summary: This schedule ensures that all core Project B deliverables — Payment Integration, RBAC/FBAC, and Bus/Table/RSVP Systems are completed by the end of the Fall 2025 term for functional validation, with enhancements and integration tasks completed in Winter 2026 prior to production testing at major MES events.

(P.5) Required technology elements

External Software Dependencies:

- **Stripe API (Payment Gateway):** Required for secure handling of online ticket purchases and event payments. Stripe provides PCI-compliant transaction processing, webhooks for event confirmations, and a sandbox environment for testing. The availability of API credentials and stable versions is critical to ensure uninterrupted payment services.
- **DrizzleORM and PostgreSQL:** PostgreSQL serves as the relational database for storing user, event, and financial data, while DrizzleORM provides type-safe schema management and migration capabilities within the TypeScript ecosystem. PostgreSQL will be containerized using Docker for development, with production deployment managed by MES through DigitalOcean. The reliability of these components is essential to data integrity and inter-team compatibility.
- **Email and Notification Services (e.g., Nodemailer, Resend, or SendGrid):** Used for automated messaging such as registration confirmations, reminders, and administrative alerts. Availability of SMTP or API configurations is necessary for dependable communication.
- **Frontend Libraries (Vite + TanStack Router + TanStack Create + TypeScript):** Used for building the single-page application interface. These depend on the Node.js and npm ecosystems for package management. Any compatibility or deprecation issues in these frameworks could delay interface development.
- **UI and Component Libraries:** The frontend will rely on open-source React component systems such as ShadCN/UI, Lucide-React icons, and form libraries (e.g., React Bootstrap) to accelerate user interface creation while maintaining accessibility and consistency.
- **Version Control and CI/CD Tools (GitHub + GitHub Actions):** GitHub is used for version control, collaboration, and issue tracking. GitHub Actions supports continuous integration for automated builds, testing, and documentation generation.

External Infrastructure and Hardware:

- **DigitalOcean Infrastructure (Managed by MES):** Used for hosting backend and database services in production. The infrastructure must support containerized deployments, SSL/TLS certificates, and routine backups to ensure availability during high-traffic events.
- **Docker Engine:** Required for local development, testing, and containerization of services to maintain uniform environments across all capstone teams.
- **QR Code Scanning Devices:** Mobile phones or tablets equipped with cameras will be necessary for attendee check-in and waiver validation at events. Reliable hardware access is critical for smooth on-site operations.

Critical Availability Summary:

Component	Purpose	Type	Criticality
Stripe API	Payment processing and ticketing	External service	**High**
PostgreSQL + DrizzleORM	Data persistence and ORM	Software	**High**
DigitalOcean	Hosting and production deployment	Infrastructure	**High**
Email / Notification Services	Automated messaging	External service	**Medium**
TanStack / Vite / TypeScript	Frontend development	Software	**Medium**
Docker	Containerization and local parity	Software	**High**
GitHub + CI/CD	Version control and automation	Platform	**Medium**
QR Code Scanning Devices	Event check-in hardware	Hardware	**Medium**

(P.6) Risk and mitigation analysis

Potential obstacles to meeting the schedule of P.4, and measures for adapting the plan if they do arise. It is essential to be on the lookout for events that could derail the project, and devise mitigation strategies. It can include a SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) for the project. [1]

Risk Analysis

- **Difficulty in integrating payment gateways:** An important part of the system is being able to integrate payment gateways for safe and secure purchasing of event tickets for users. However, there are multiple risks that can arise such as transaction delays or API changes that could disrupt the payment processes. The result of these issues can be failed or duplicate transactions which can frustrate users as well as payments not being processed in time resulting in difficulties for organizers planning paid events. A couple of mitigation strategies for this risk can be incorporating an option for multiple different payment gateways for multiple different providers in case one fails and scheduling regular maintenance or updates to verify API compatibility. Additionally, integrating transaction logs with fail-safe mechanisms for retrying on failures and emailing confirmation receipts can help prevent loss of payment and improve reliability which can prevent any frustration or unease for users while at the same time keeping a dependable trace of data for organizers.

- **Possible security and privacy vulnerabilities related to role based accesses:** With the role based and feature based access control the application provides, it also introduces potential security and privacy vulnerabilities. For instance, incorrectly configured access validation could expose sensitive data such as user payment information or admin dashboard details to unauthorized users, which could violate university policies on data handling. There can also be a risk of violating privacy laws if the data is not handled properly which can result in lawsuits, fines and damage to reputation. Additionally, if the perceived notion from users is that their data is not being handled properly, it could lead to distrust and a loss in user retention and user growth. Mitigation strategies for this risk on the user side would include implementing end-to-end encryption of all user data, collecting only minimum necessary user data and implementing principles of lease privilege for all users. On the administration side, mitigation strategies would include doing regular security audits to identify areas of improvement or potential security vulnerabilities and logging all administrative actions for traceability.

Threat Analysis

- **Integration and Module Dependency Misalignment**
  - **Description:** Since each capstone team is responsible for developing separate modules that will later integrate into a shared MacSync repository, inconsistencies in API contracts, data models, or technology versions could delay integration and cause major refactoring work late in development.
  - **Mitigation Mechanisms:** Establish clear interface definitions and shared schema documentation early in the project. Conduct regular integration checkpoints between teams to identify incompatibilities before final merging. Maintain alignment through shared test/mock data, and frequent communication between teams to ensure extra time is considered for this integration phase.
- **Schedule Delays and Coordination Challenges**
  - **Description:** Tight milestone deadlines and interdependent deliverables increase the risk of schedule slippage, especially when documentation, backend, and frontend progress must all align for project completion.
  - **Mitigation Mechanisms:** Break larger tasks into smaller, trackable subtasks using GitHub issues and GitHub projects. Hold weekly progress meetings to monitor task completion, identify blockers early, and reassign workloads if needed. Maintain extra buffer time before major milestones to accommodate integration testing and unforeseen delays.

(P.7) Requirements process and report

Initially, description of what the requirements process will be; later, report on its steps. It

> *starts out as a plan for conducting the requirements elicitation process, but is meant to be updated as part of that process so that it includes the key lessons of elicitation.* [1]

(P.7.1) Interviews with Direct Stakeholders

- **MES Event Organizers([Sarah Davis])**
  - **Type of Interview: Hybrid**. To understand how the MES currently plans, promotes, and executes events, uncover inefficiencies in the process, and align the MacSync platform's design and functionality with MES's overall vision for improved event management and engagement.
  - **Most Important Question:** What outcomes define success that we should be aiming for to align with your vision for the MacSync platform? From the initial announcement of an event to the start time, where do we lose the most time? What platforms don't gain a lot of traction, and why are users not aware of certain events?
- **Students/Attendees ([James Thompson])**
  - **Type of Interview: Closed**. The purpose of this interview is to gather information on why students aren't showing up to events. This interview will be very similar to a survey.
  - **Most Important Question:** Are you aware of MES events? What platforms do you view the announcements of events? What step of the registration process do you feel hesitant to continue?
- **University Administration ([Ahmed Patel])**
  - **Type of Interview: Closed**. This format ensures that specific compliance, policy, and governance-related details are clearly captured.
  - **Most Important Question:** To understand the university's financial, data security, and compliance requirements that the MacSync platform must adhere to.
- **Sponsors ([Laura Nguyen])**
  - **Type of Interview: Open**. This format encourages an open dialogue to explore how sponsors interact with student events. It allows us to understand their motivations with sponsoring MES events.
  - **Most Important Question:** What features would make sponsoring or promoting events through MacSync most valuable to your business and worth investment?

(P.7.2) Justification for Interviews

The interview types and questions listed above are tailored to each persona to gather the relevant facts of the short comings of the current MES event hosting eco system. The information gathered will help in translating these pain points into functional and non-functional requirements.

References

- [1] Bertrand Meyer. *Handbook of Requirements and Business Analysis*. Springer. 2022.
- [2] Ian Sommerville and Peter Sawyer. *Requirements Engineering: A good Practice Guide*. Wiley. 1997.