SOFTWARE REQUIREMENTS SPECIFICATION

Sprint 2

NIMRA ASLAM
MUHAMMAD FAIZAN
SHEHRYAR

Contents

1. Introduction:

1.1. Purpose:

This SRS document outlines the requirements for an **Event Management System (EMS)** designed to streamline the planning, execution, and post-event activities for organizers and attendees. It specifies functional and non-functional requirements, use cases, user stories, and design diagrams.

1.2. Scope:

The EMS will allow event organizers to create, manage, and close events, while attendees can register, receive updates, and provide feedback. The system supports features like notifications, reports, and attendee management. It uses users email for notifications, authentication, and tracking of users and events which is done through database that is integrated. It will be a web-based application accessible to customers, attendees, organizers.

1.3. Definitions, Acronyms, and Abbreviations:

- EMS: Event Management System
- Organizer: User responsible for creating and managing events
- Attendee: User registering for events
- Customer: User using the software and is not registered for any event
- Administrator: User who checks performance of events

1.4. References:

- IEEE 830-1998 Standard for SRS
- Event Management System Sprint 1

1.5. Overview:

This document is structured as follows:

- Section 2 provides an overall description of the EMS, including its perspective, functions, user characteristics, constraints, and assumptions.
- Section 3 details the specific requirements, encompassing functional and non-functional requirements, a use case diagram, user stories with pre- and post-conditions, sequence diagrams, and a class diagram.

2. Overall Description:

2.1. Product Perspective:

The EMS is a web application that interfaces with the university's student database for user authentication and email services for notifications. It aims to provide a centralized platform for event operations, enhancing efficiency and user experience

2.2. Product Functions:

- Event creation, updating, and cancellation
- Attendee registration and management
- Notification and reminder system
- Reporting and feedback collection

2.3. User Characteristics:

- Event Organizer: Manages event creation, updating, cancellation.
- Attendee: Require features to register for event, cancellation of registration, updating registration details, providing feedback of event.
- Customer: Requires features to Login and Signup. Also has features of attendee. A customer will be an attendee if he/she is registered for an event.

2.4. Constraints:

- The system must be implemented using Java, JavaFX and MySQL workbench.
- Compatible with Windows, macOS, and Linux.

2.5. Assumption and Dependencies:

- University's email address is for authentication of users
- Notifications are sent via email addresses
- Users have basic computer literacy.

3. Requirement Specification:

3.1. External Requirements:

- **User Interfaces**:
    - The system shall provide a desktop-based graphical user interface (GUI) built with JavaFX, accessible on standard operating systems (e.g., Windows, macOS, Linux).
- **Hardware Interfaces**:
    - The system shall operate on standard client hardware, such as personal computers or laptops.
- **Software Interfaces**:
    - The system shall integrate with a relational database (e.g., MySQL workbench) via JDBC for persistent storage of events, attendees, and reports.
    - The system shall interface with an email service using university email addresses.

- Developed in Eclipse IDE using Java and JavaFX libraries, ensuring compatibility with Java SE 17 or later.

- **Communication Interfaces**:
  - To send emails using an email address (e.g., Gmail), you configure JavaMail to connect to an SMTP server.
  - Local file system access shall be supported for exporting attendee lists and reports in CSV format.

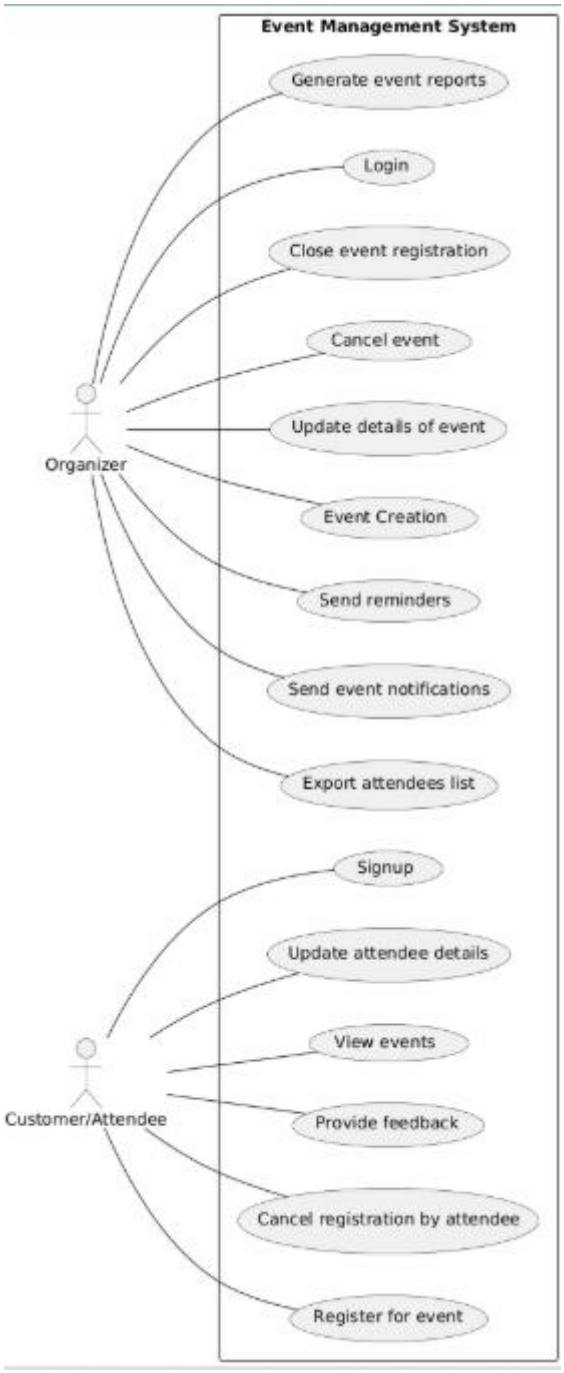
3.2. Functional Requirements:

1. **Event Creation:** The system shall allow organizers to create events with details.
2. **Register for event:** The system shall enable attendees to register for events.
3. **Update details of event:** The system shall allow organizers to update event details.
4. **Cancel event:** The system shall allow organizers to cancel events.
5. **Cancel registration by attendee:** The system shall allow attendees to cancel their registration.
6. **Send event notifications:** The system shall send event notifications to attendees about an event (like you have registered for this event or you have cancelled the registration).
7. **Close event registration:** The system shall close event registration when specified.
8. **Provide feedback:** The system shall collect feedback from attendee's post-event.
9. **Login:** The system shall allow the organizer to login.
10. **Generate event reports:** The system shall generate event reports (e.g., attendance stats).
11. **Export attendees list:** The system shall export attendees list in CSV format.
12. **View events:** The system shall allow users to view event details.
13. **Send reminders:** The system shall send reminders to attendees.
14. **Update attendee details:** The system shall allow organizers to update attendee details.
15. **Signup:** The system shall new user to signup as customers.

3.2.1. Use case Diagram:

Description: A single UML use case diagram illustrating the overall functionality.

- **Actors**: Event Organizer, Customer/Attendee
- **Event Organizer:** Login, close registration, cancel event, update event details, event creation, send reminder, send event notifications, export attendee list, generate events reports.
- **Customer/Attendee:** View events, provide feedback, update attendee details, cancel event registration, register for event, signup



3.2.2. User stories:

Organizer User Stories:

1. **User Story 1: Event Creation**
    - **Role**: Organizer
    - **Goal**: Create an event

- **Reason**: To schedule and manage an event efficiently, allowing attendees to register and participate.
- **Pre-Conditions**: Organizer is logged into the system and, event doesn't already exist, another event should not be at same location.
- **Post-Conditions**: A new event is stored in the database with details (e.g., name, date, location).

2. **User Story 2: Update Details of Event**
   - **Role**: Organizer
   - **Goal**: Update event details
   - **Reason**: To reflect changes (e.g., new time or venue) and keep attendees informed.
   - **Pre-Conditions**: Organizer is logged in, and the event exists in the system.
   - **Post-Conditions**: The event's details are updated in the database, and attendees are optionally notified of changes.
3. **User Story 3: Cancel Event**
   - **Role**: Organizer
   - **Goal**: Cancel an event
   - **Reason**: To terminate an event that can no longer occur, ensuring attendees are informed and resources are freed.
   - **Pre-Conditions**: Organizer is logged in, and the event exists in the system.
   - **Post-Conditions**: The event is marked as cancelled in the database, and attendees are notified.
4. **User Story 4: Close Event Registration**
   - **Role**: Organizer
   - **Goal**: Close event registration
   - **Reason**: To stop new registrations when capacity is reached or the event registration deadline date and time approaches, ensuring manageable attendance.
   - **Pre-Conditions**: Organizer is logged in, and the event exists with open registration.
   - **Post-Conditions**: Registration for the event is closed, and no new attendees can register.
5. **User Story 5: Login**
   - **Role**: Organizer
   - **Goal**: Log into the system
   - **Reason**: To access organizer-specific features securely and manage events.
   - **Pre-Conditions**: Organizer has a registered account with valid credentials.
   - **Post-Conditions**: Organizer is authenticated and granted access to the system's organizer interface.
6. **User Story 6: Generate Event Reports**
   - **Role**: Organizer
   - **Goal**: Generate event reports
   - **Reason**: To analyse event success (e.g., attendance stats) and improve future planning.
   - **Pre-Conditions**: Organizer is logged in, and the event exists with recorded data (e.g., attendees).

- **Post-Conditions**: A report is generated and displayed, summarizing event metrics.

7. **User Story 7: Export Attendees List**
   - **Role**: Organizer
   - **Goal**: Export attendees list.
   - **Reason**: To obtain a portable list of attendees (e.g., for check-in or external use), enhancing event management.
   - **Pre-Conditions**: Organizer is logged in, and the event exists and has registered attendees.
   - **Post-Conditions**: The attendees list is exported as a file (e.g., CSV) to the organizer's device.
8. **User Story 8: Send Event Notifications**
   - **Role**: Organizer
   - **Goal**: Send event notifications
   - **Reason**: To inform attendees about important updates (e.g., schedule changes), ensuring smooth communication.
   - **Pre-Conditions**: Organizer is logged in, the event exists, and attendees are registered.
   - **Post-Conditions**: Notifications are sent to all registered attendees via email.
9. **User Story 9: Send Reminders**
   - **Role**: Organizer
   - **Goal**: Send reminders
   - **Reason**: To prompt attendees about upcoming events, increasing participation and preparedness.
   - **Pre-Conditions**: Organizer is logged in, the event exists, and a reminder schedule is set.
   - **Post-Conditions**: Reminders are sent to registered attendees via email.

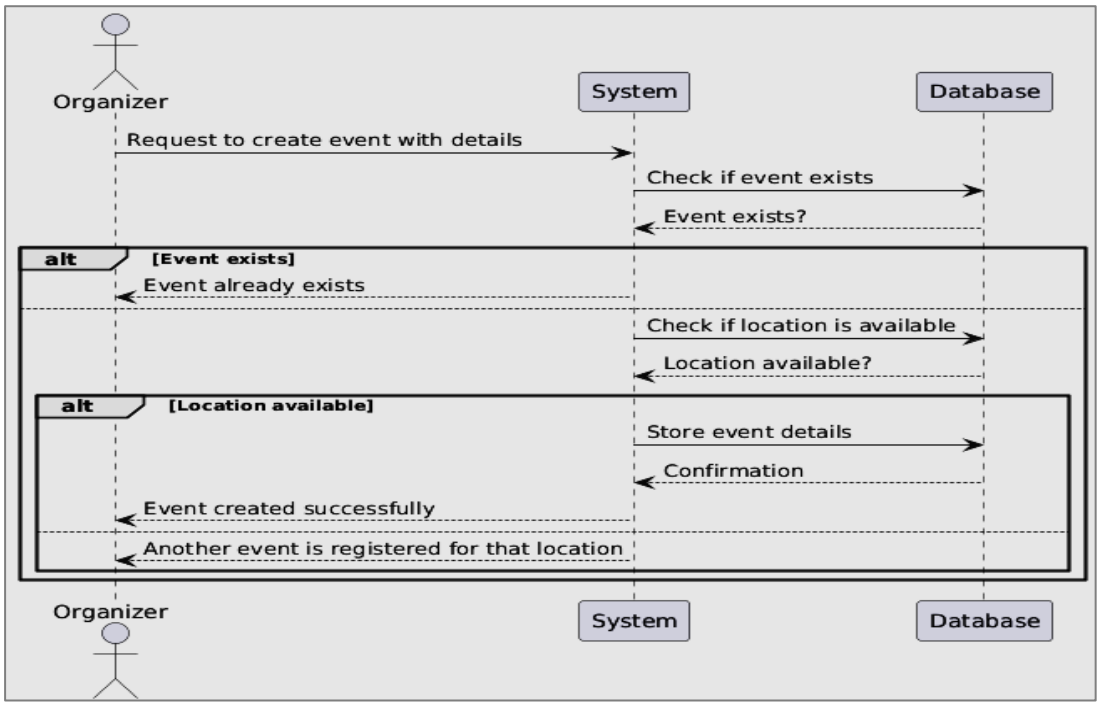
Attendee User Stories

10. **User Story 10: Register for Event**
    - **Role**: Attendee
    - **Goal**: Register for an event
    - **Reason**: To participate in an event of interest and secure a spot.
    - **Pre-Conditions**: Attendee is logged in, and the event exists with open registration.
    - **Post-Conditions**: Attendee is added to the event's attendee list in the database.
11. **User Story 11: Cancel Registration by Attendee**
    - **Role**: Attendee
    - **Goal**: Cancel registration for an event
    - **Reason**: To withdraw from an event they can no longer attend, freeing their spot for others.
    - **Pre-Conditions**: Attendee is logged in and registered for the event.
    - **Post-Conditions**: Attendee is removed from the event's attendee list.
12. **User Story 12: Provide Feedback**
    - **Role**: Attendee

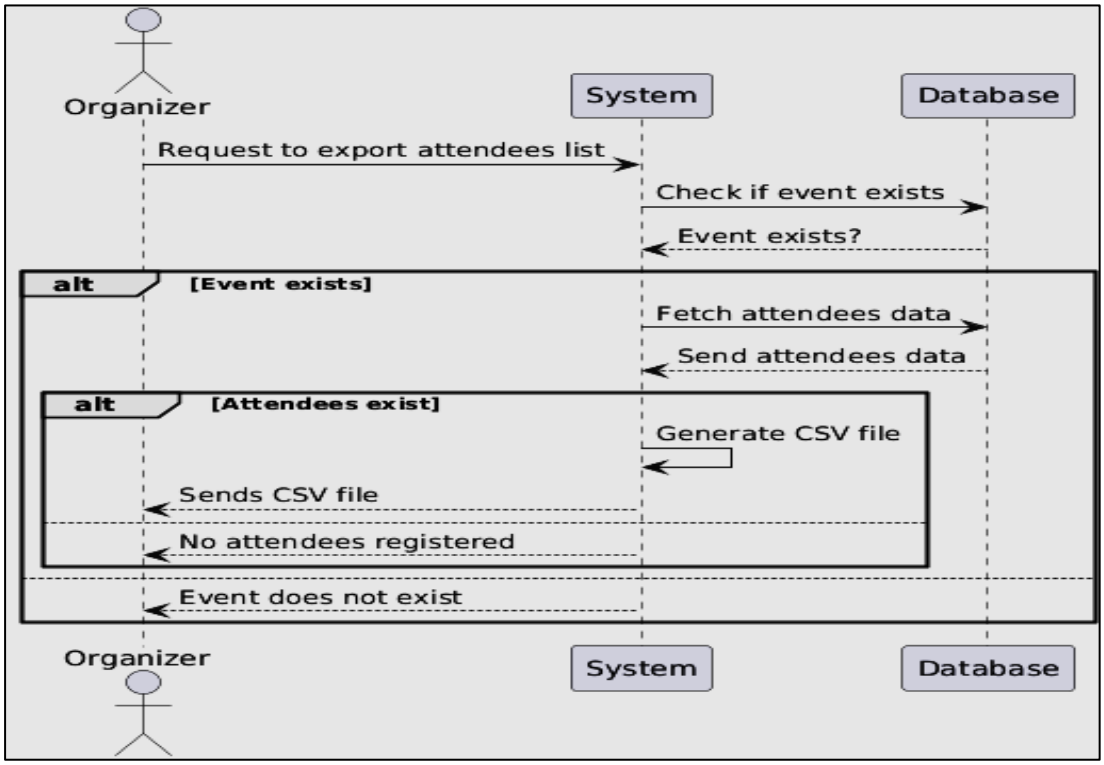
- **Goal**: Provide feedback
- **Reason**: To share their experience, helping organizers improve future events.
- **Pre-Conditions**: Attendee is logged in, and the event has concluded.
- **Post-Conditions**: Feedback is recorded in the database and associated with the event.

13. **User Story 13: View Events**
    - **Role**: Attendee
    - **Goal**: View events
    - **Reason**: To browse available events and decide which to attend.
    - **Pre-Conditions**: Attendee is logged in or has access to the public event list.
    - **Post-Conditions**: A list of events with details (e.g., date, location) is displayed to the attendee.
14. **User Story 14: Update Attendee Details**
    - **Role**: Attendee
    - **Goal**: Update attendee details
    - **Reason**: To ensure their personal information (e.g., email, name) is accurate for event communication.
    - **Pre-Conditions**: Attendee is logged in and has an account.
    - **Post-Conditions**: Attendee's updated details are saved in the database.
15. **User Story 15: Signup**
    - **Role**: Attendee
    - **Goal**: Sign up for the system
    - **Reason**: To create an account and access attendee features like registration and feedback.
    - **Pre-Conditions**: Attendee does not yet have an account.
    - **Post-Conditions**: A new attendee account is created in the database with provided credentials.

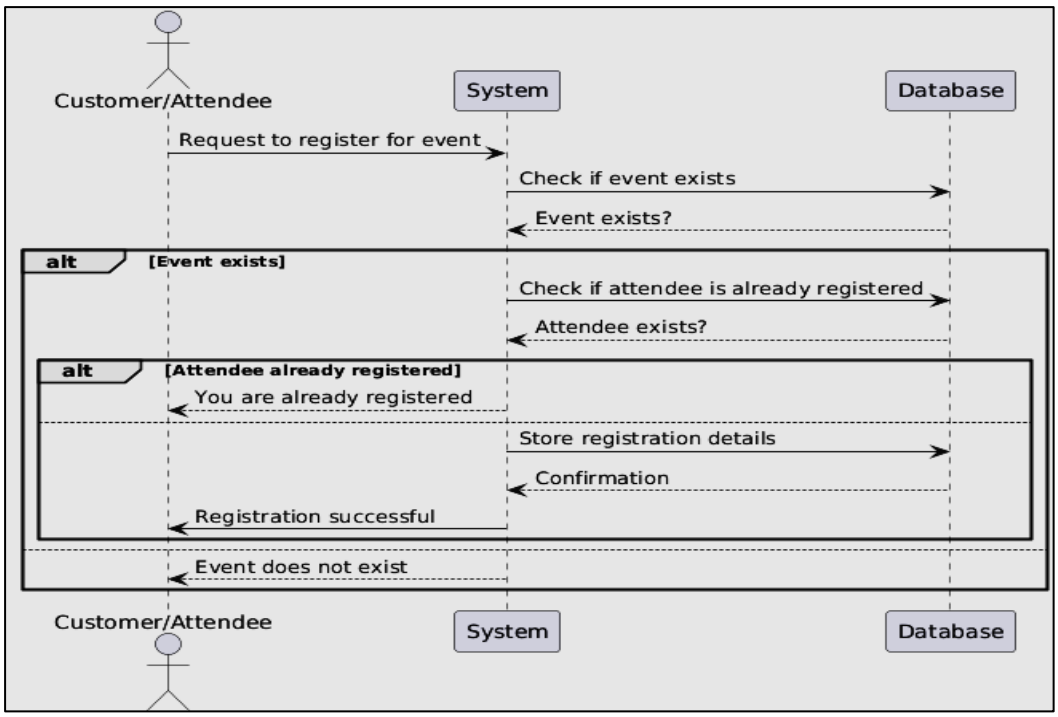
3.2.3. Sequence Diagrams:



Sequence Diagram 1: Event Creation



Sequence Diagram 2: Export attendees list(csv)



Sequence Diagram 3: Register for event

3.3. Non-functional Requirements:

3.2.1 Product Requirements:

- **Usability**: The interface shall be intuitive, with a learning curve of less than 10 minutes.
- **Performance**: The system shall handle up to 1,000 attendees per event without delay.
- **Reliability**: The system shall have 99% uptime during event registration periods.

3.2.2 Organizational Requirements:

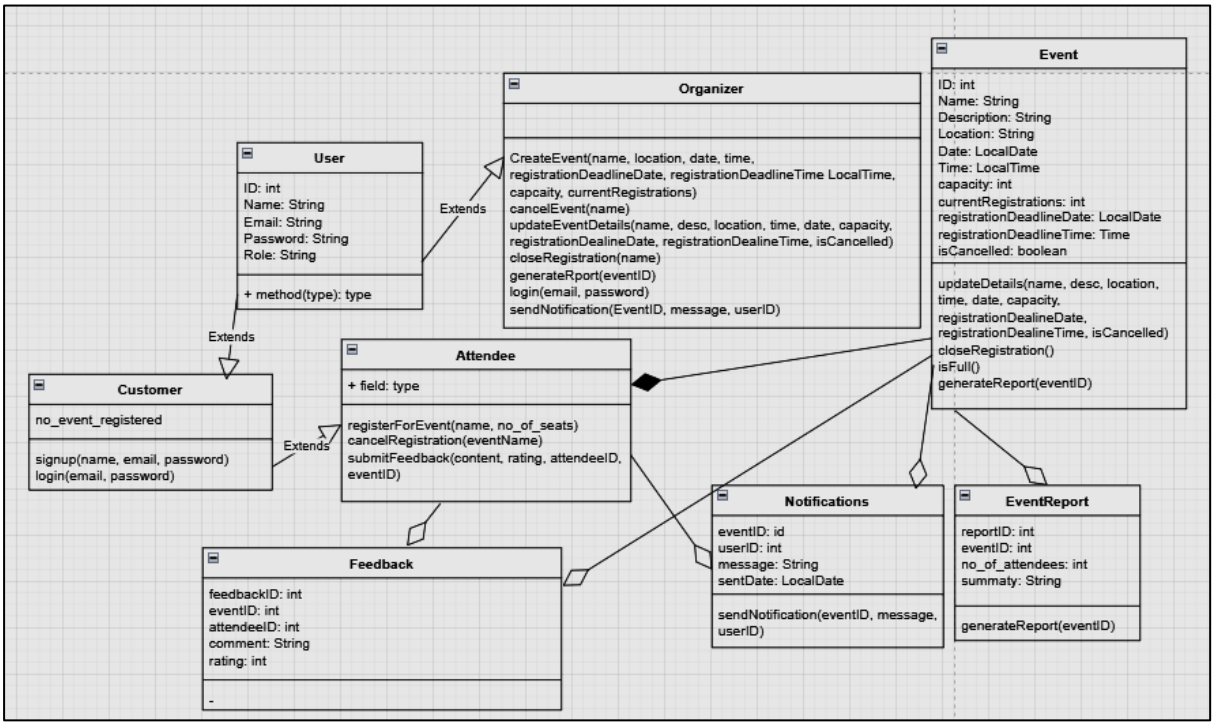
- **Standards**: The system shall adhere to IEEE coding standards.
- **Delivery**: The system shall be developed using Java, JavaFX, and MySQL workbench database.

3.2.3 External Requirements:

- **Scalability**: The system shall support future integration with third-party email marketing tools or attendee management platforms to enhance notification and registration capabilities.
- **Compliance:** The system shall adhere to data protection regulations for storing and processing user information, ensuring attendee and organizer data privacy and security.

4. Appendix:

4.1. Class Diagram:



Class Diagram

4.1.1. Relationships:

Event *—* Attendees (one event can have multiple attendees and one attendee can be registered in multiple events)

Attendee 1—* Feedback (one attendee can give many feedbacks)

Event 1—* Notifications (one event can have many notifications)

Event 1—1 EventReport (one event can have one eventReport)

Event 1—* Feedback (one event can have many feedbacks)

Arigatou gozaimasu