Software Requirement Specification
For Event Management System
(Language: C programming)

Prepared by:

Masrafi ID: 242-35-369

Shahriar Islam Khan ID: 242-35-370

Alif Mahmud ID: 242-35-358

Daffodil International University

Software Engineering Dept.

Submission date: 15/06/2025

1. Introduction

1.1 Purpose

This document outlines the Software Requirements Specification for the **Event Management System**, developed using the C programming language. The system is designed to assist educational institutions or community organizations in managing events by automating scheduling, registration, attendance, and feedback processes.

1.2 Scope

The system provides a platform for organizers to create and manage events while enabling participants to register and give feedback. It reduces manual errors, streamlines coordination, and stores all data securely using file handling. It will provide features such as:

- Event creation and editing
- Participant registration and attendance
- Scheduling conflict management
- Report and feedback generation

1.3 Definitions, Acronyms, and Abbreviations

- **EMS** – Event Management System
- **SRS** – Software Requirements Specification
- **UI** – User Interface

1.4 References

- ANSI C Programming Language Standards

- File Handling in C – Standard Library Functions
- IEEE 830 SRS Template

2. Overall Description

2.1 Product Perspective

The EMS is a standalone console-based system developed in C. It uses standard input/output and file handling to store and retrieve data.

2.2 Product Functions

- Add, update, and delete events
- Participant registration with confirmation
- View events and details
- Track and record attendance
- Collect and store feedback
- Generate reports for event summaries

2.3 User Classes and Characteristics

- **Organizer:** Can create, manage, and monitor events
- **Participant:** Can register, attend, and provide feedback

2.4 Operating Environment

- OS: Windows/Linux

- Compiler: GCC / Turbo C / Code::Blocks
- Language: C (standard libraries only)

2.5 Design and Implementation Constraints

- Text-based interface (no GUI)
- File-based storage (no database)
- C standard I/O and file handling

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Event Creation and Management

- **FR1**: Organizers can add new events (title, description, date, time, venue).
- **FR2**: Organizers can update or delete existing events.
- **FR3**: The system should check for scheduling or venue conflicts.

3.1.2 Participant Registration

- **FR4**: Participants can view available events.
- **FR5**: Participants can register for events.

- **FR6**: The system should generate a registration confirmation.

3.1.3 Attendance Tracking

- **FR7**: Organizers can mark and view participant attendance.

3.1.4 Feedback Collection

- **FR8**: Participants can submit feedback after attending an event.

3.1.5 Reporting

- **FR9**: The system can generate summary reports for each event (e.g., number of registrations, attendance, feedback stats).

3.1.6 Data Storage and Retrieval

- **FR10**: All event, registration, and feedback data should be stored using file handling.

- **FR11**: The system should retrieve and display stored records on demand.

3.2 Non-Functional Requirements

3.2.1 Usability

- **NFR1**: The system should provide a simple text-based user interface for ease of use.

- **NFR2**: Menus and prompts should be clear and intuitive.

3.2.2 Reliability

- **NFR3**: The system should ensure data is correctly stored and not lost between sessions.

- **NFR4**: The system should handle invalid inputs gracefully (e.g., incorrect date format or duplicate registration).

3.2.3 Performance

- **NFR5**: The system should process operations like registration or event search within a few seconds.

3.2.4 Security

- **NFR6**: Access to event editing features should be restricted to organizers (simple role distinction).

- **NFR7**: All data files should be protected from unauthorized changes.

3.2.5 Portability

- **NFR8**: The system should run on any standard C compiler across Windows/Linux platforms.

3.2.6 Maintainability

- **NFR9**: Code should be modular (e.g., using functions for different operations like registration, reporting, etc.) to allow future enhancements.

4. External Interface Requirements

4.1 User Interfaces

- Command-line interface
- Menu-driven navigation for both organizer and participant

4.2 Hardware Interfaces

- Standard keyboard input and screen output

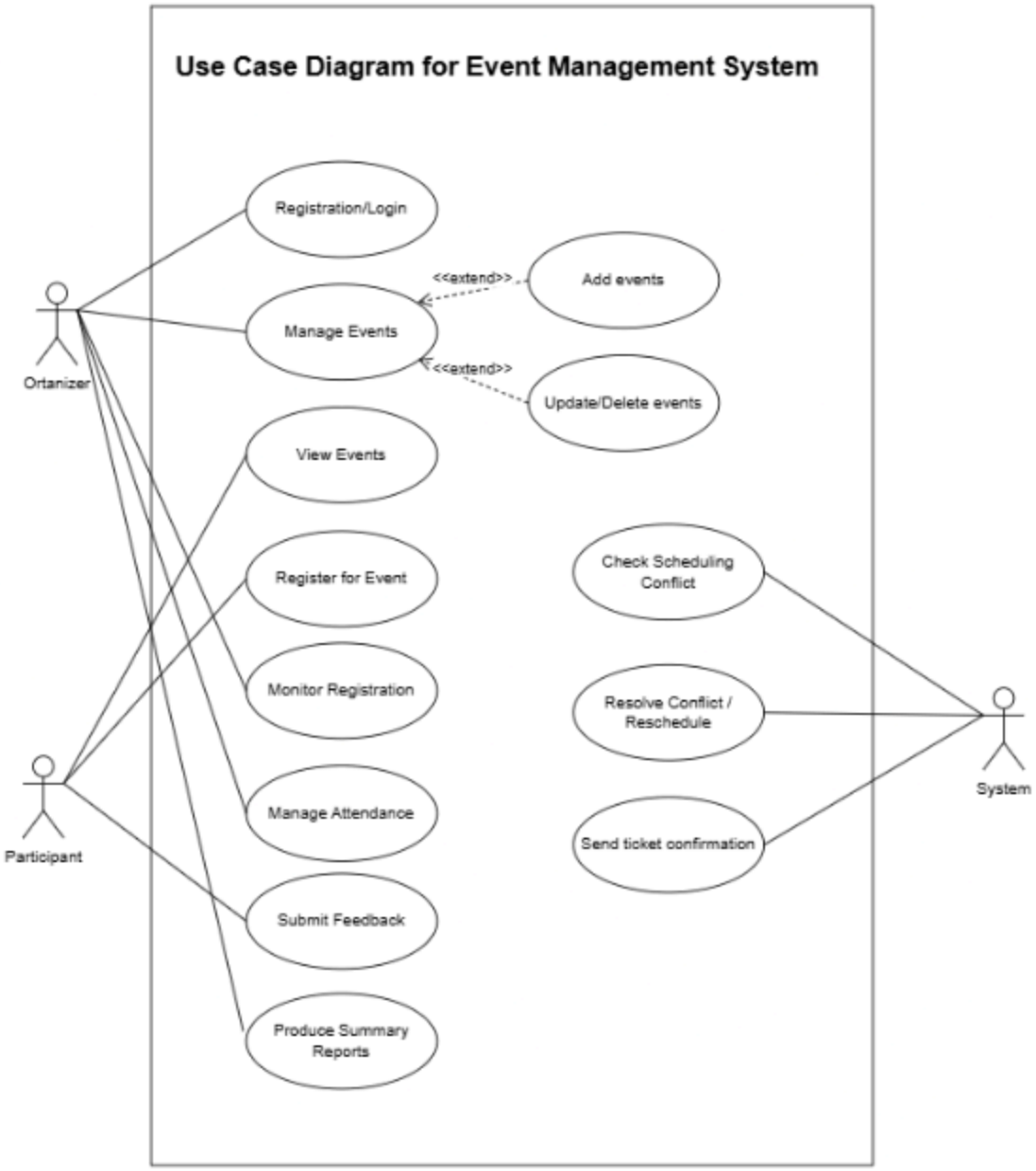
4.3 Software Interfaces

- Standard C I/O libraries: `stdio.h`, `stdlib.h`, `string.h`, `time.h`

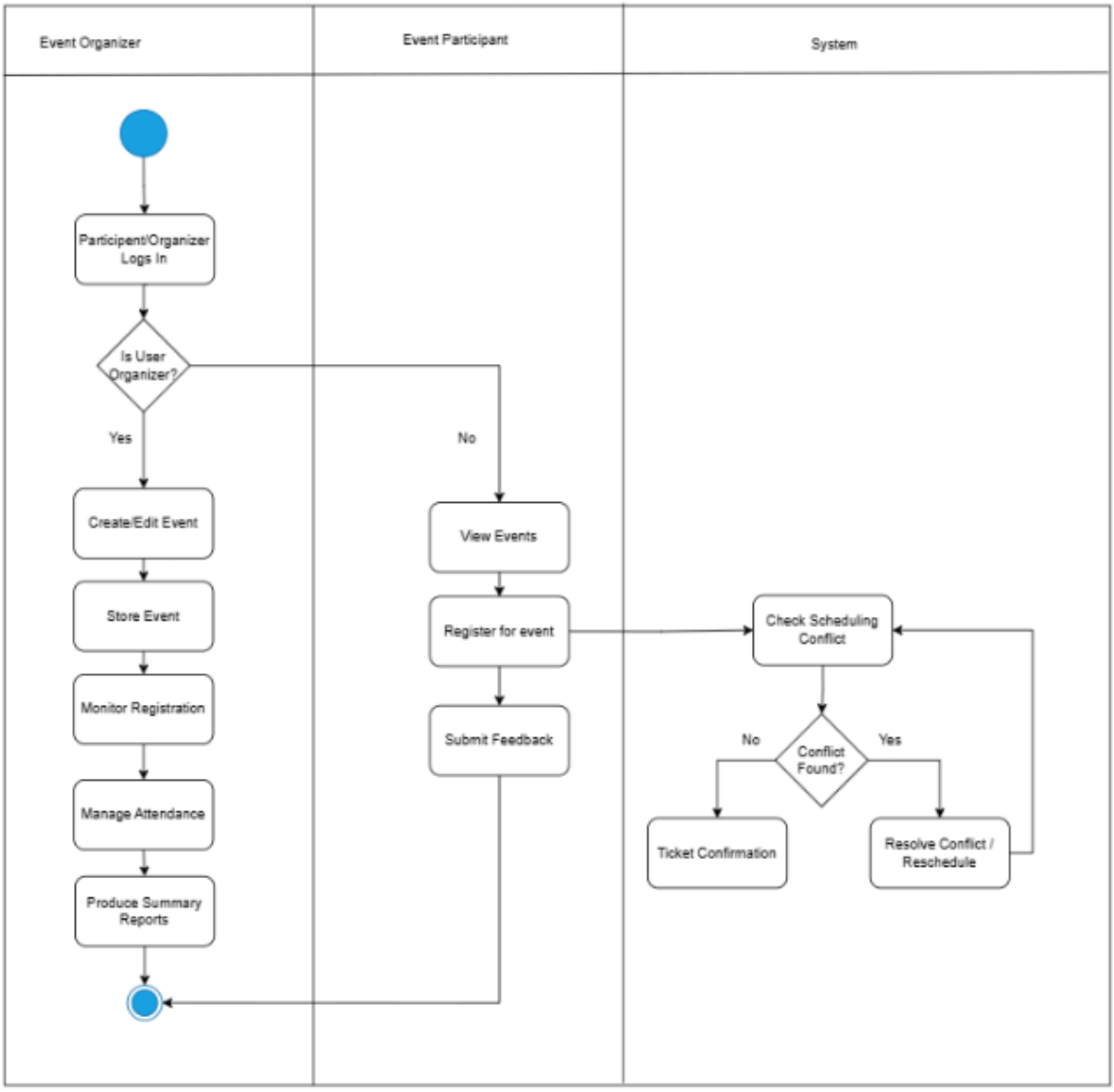
4.4 Communication Interfaces

- Not applicable (standalone, offline system)

5. Use Case Diagram



6. Activity Diagram



7. Appendix

- **Technology Used**: C programming, File Handling, CLI
- **Tools**: GCC Compiler, Code::Blocks / Turbo C
- **Limitations**: No database or network access; Console-only UI

- **Future Scope**: GUI support, database integration, email confirmations