Software Requirements Specification

for

Event Management System

Version 2.0

Prepared by

Team no :5

|  |  |  |
| --- | --- | --- |
| Donta Chaitanya | se22uari047 | se22uari047@mahindrauniversity.edu.in |
| Pilli Sachethan | se22uari199 | se22uari199@mahindrauniversity.edu.in |
| Susmitha Manthena | se22uari170 | se22uari170@mahindrauniversity.edu.in |
| Sanjana Singam | se22uari153 | se22uari153@mahindrauniversity.edu.in |
| Kannikanti Chandana Priya | se22uari067 | se22uari067@mahindrauniversity.edu.in |
| Sameera Nallamothu | se22uari150 | se22uari150@mahindrauniversity.edu.in |

|  |  |
| --- | --- |
| Instructor: | *Arun Avinash Chuhan* |
| Course: | Software Engineering |
| Lab Section: | *<* place your lab section here *>* |
| Teaching Assistant: | *Nartkannai . K* |
| Date: | 04-03-2025 |

Contents

Contents ii

Revisions ii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 Overall Description 2

2.1 Product Overview 2

2.2 Product Functionality 3

2.3 Design and Implementation Constraints 3

2.4 Assumptions and Dependencies 3

3 Specific Requirements 4

3.1 External Interface Requirements 4

3.2 Functional Requirements 4

3.3 Use Case Model 5

4 Other Non-functional Requirements 6

4.1 Performance Requirements 6

4.2 Safety and Security Requirements 6

4.3 Software Quality Attributes 6

5 Other Requirements 7

Appendix A – Data Dictionary 8

Appendix B - Group Log 9

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Draft Type and Number | Full Name | Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded. | 00/00/00 |

# 

# Introduction

## Document Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed and structured description of the *Event Management System* developed for managing college or institutional events. This document outlines the system's intended functionality, features, and behavior, serving as a guide for developers, testers, and stakeholders. It ensures that all requirements are clearly understood and agreed upon by all parties involved in the project.

The document describes the roles of three primary user types—Admin, Organizer, and Student—and specifies the functional and non-functional requirements associated with each. It also acts as a reference for any future enhancements or maintenance activities.

## Product Scope

The *Event Management System* is a web-based application designed to streamline the process of organizing, managing, and tracking institutional events. It enables efficient coordination between Admins, Organizers, and Students by automating event approval, registration, expense tracking, and reporting.

The system allows Organizers to create and manage events, while Students can view, register for, and cancel participation in those events. The Admin oversees the entire platform, with the authority to approve or reject events, monitor expenses, view detailed reports, and manage registered student data. Admin credentials are preloaded in the system to ensure security and control.

The product aims to:

* Simplify event creation and approval workflows.
* Maintain a centralized event database.
* Provide transparency through detailed reports and tracking features.
* Enhance user experience through a responsive and intuitive interface.

This system reduces manual overhead, improves decision-making through analytics, and ensures secure role-based access control for all users.

## Intended Audience and Document Overview

#### ****Intended Audience****

This document is intended for all stakeholders involved in the design, development, deployment, testing, and use of the Event Management System. The target audience includes:

* **Developers** – to understand the system functionalities and implement the required features.
* **Testers/QA Engineers** – to design test cases based on the functional and non-functional requirements.
* **Project Managers** – to track progress and ensure deliverables align with the specified requirements.
* **Institutional Stakeholders** – to validate that the system meets the organizational needs and goals.
* **Future Maintenance Teams** – to refer for enhancements, bug fixes, or further development.

#### ****Document Overview****

This Software Requirements Specification (SRS) provides a detailed overview of the Event Management System. It is organized into the following sections:

* **Section 1**: Introduction, including purpose, scope, intended readers, and key definitions.
* **Section 2**: Overall system description including product functionality and constraints.
* **Section 3**: Specific requirements divided into external interface, functional needs, and use cases.
* **Section 4**: Non-functional requirements such as performance, security, and quality attributes.
* **Section 5**: Any other requirements not covered above.
* **Appendices**: Data dictionary and group contribution log.

## Definitions, Acronyms and Abbreviations

* **F1** – Functional Requirement 1 (Section 3.2)
* **P1** – Performance Requirement 1 (Section 4.1)
* **U1** – Use Case 1 (Section 3.3.1)
* **S1**– Security Requirement (Section 4.2)

## Document Conventions

This document follows the IEEE formatting requirements. The formatting and typographical conventions used in this SRS are as follows:

#### Formatting Conventions

* **Font:** Arial, size 11 or 12, is used throughout the document.
* **Spacing:** Text is single-spaced, with standard 1” margins.
* **Titles and Headings:** Section and subsection titles adhere to the provided template for consistency.
* **Italics:** Used for comments or notes that are not part of the formal document content.

#### Naming Conventions

* **Requirements Identifiers:** Functional and non-functional requirements are labeled systematically (e.g., FR-1, FR-2 for functional requirements and NFR-1, NFR-2 for non-functional requirements).
* **Acronyms and Abbreviations:** Defined in the Definitions, Acronyms, and Abbreviations section to ensure clarity.

These conventions ensure that the document remains structured, readable, and professional.

## References and Acknowledgments

Draw.io for Use Case Diagram.

# Overall Description

## Product Overview

The *Event Management System* is a centralized web-based platform developed to streamline the organization and participation in university events. It enables seamless interaction among three user roles: Admin, Organizer, and Student.

* **Admin** is the system’s superuser with authority to approve or reject events proposed by Organizers. The Admin can also view and generate reports, edit or cancel events, monitor event expenses, and track student participation. Admin credentials are preloaded into the database to prevent unauthorized access or sign-up.
* **Organizers** are responsible for creating, modifying, and managing event details. They can also cancel events and track associated expenses. Organizers have access to event reports that summarize metrics such as number of participants and budget usage.
* **Students** are end-users who can view ongoing events, register for events of their interest, and cancel registrations when needed.

The system is designed to minimize manual administrative effort, ensure secure role-based access control, and provide real-time insights through reports and tracking features. It also offers an intuitive interface and seamless workflows to enhance user experience for all roles.

## 2.2 Product Functionality

The *Event Management System* provides distinct functionalities based on user roles—Admin, Organizer, and Student. Each user type interacts with the system through role-specific features that collectively enable seamless event management within an institution.

#### **Admin Functionalities:**

* Approve or reject event requests submitted by organizers.
* Edit or cancel approved events.
* Track and manage event-related expenses.
* View and export detailed event reports, including participant data.
* View and monitor registered students and their event participation details.
* Access is restricted—admin credentials are hard-coded in the database (no signup).

#### **Organizer Functionalities:**

* Create new event proposals with relevant details (date, description, budget, etc.).
* Edit or cancel existing events before they are approved.
* Track event expenses and update financial logs.
* View reports related to their own events including student participation.

#### **Student Functionalities:**

* View all approved and ongoing events.
* Register for events of interest.
* Cancel previously made event registrations.

The system enforces **role-based access control** to ensure data integrity and security and provides a unified interface for all types of users. It facilitates smooth workflows for event management, participation, and administration within a college or organizational setup.

## 2.3 Design and Implementation Constraints

The design and implementation of the *Event Management System* are subject to the following constraints:

**Role-Based Access Control (RBAC)**:  
 The system must enforce strict access control. Admin credentials are hardcoded in the database and cannot be created or modified through the interface. Only one Admin exists, and only authenticated users can access their respective dashboards.

**Technology Stack Constraint**:

* Frontend: HTML, CSS, JavaScript
* Backend: Node.js with Express.js
* Database: MySql

**Single Admin Limitation**:  
Only one Admin is allowed in the system. Admin privileges are not transferable or shareable, which limits administrative scalability.

**Data Persistence**:  
 Event, registration, and user data are stored in Mysql.

**Scalability Considerations**:  
 The current system design is focused on a single university use-case.

These constraints must be considered during development, testing, deployment, and any future upgrades or enhancements.

## 2.4 Assumptions and Dependencies

The development and successful operation of the *Event Management System* are based on the following assumptions and dependencies:

#### **Assumptions**

1. Users (Organizers and Students) will have basic knowledge of how to interact with a web-based application.
2. The admin user credentials are securely stored and only accessible to the designated system administrator.
3. Internet connectivity is consistently available to all users for accessing the application (if deployed).
4. Events will follow a structured approval process — no event will proceed without Admin approval.

#### **Dependencies**

1. **Web Server and Hosting**: The system depends on a Node.js server and a compatible hosting environment for backend operation.
2. **Database**: MongoDB is required as the backend database system for storing user data, events, and registration details.
3. **Authentication Library**: The system uses an authentication mechanism (e.g., token-based login) for role-based access control.
4. **Third-Party Libraries**: Frontend and backend frameworks or dependencies (e.g., Express, Mongoose) must be available and properly configured.

Failure or misconfiguration in any of these dependencies may result in limited or non-functional system performance.

# Specific Requirements

## External Interface Requirements

### User Interfaces

The *Event Management System* features a user-friendly and responsive interface tailored to the needs of three distinct user roles: Admin, Organizer, and Student. Below are the key UI components:

#### Key UI Elements:

* **Authentication Pages** – Login, Signup (for students & organizers).
* **Dashboard** – Personalized overview for admin, organizers, and students showing events, alerts, and stats.
* **Event Management Interface** – Create, edit, cancel, and view event details (organizers); approve/reject events (admin).
* **Registration Interface** – Students can register/unregister; organizers and admin can manage registrations.
* **Profile Pages** – Role-specific dashboards for students (my registrations), organizers (my events), and admin (event oversight).
* **Expense Tracker** – Track, edit, and categorize expenses for each event.
* **Report Generation** – Generate and download reports on event performance, expenses, and student participation.

#### User Interaction:

* **Click-Based Navigation:** Enables users to interact with the platform primarily through mouse clicks on desktop devices.
* **Drop-Down Menus & Forms:** Used for entering event details, selecting roles, dates, venues, and managing event registrations.
* **Interactive Calendar:** Allows organizers and students to view event schedules, deadlines, and avoid date conflicts.
* **Search & Filters:** Helps users quickly find events, filter by type, venue, date, or status, and manage their participation efficiently.

### Hardware Interfaces

The system will interact with various hardware components to ensure smooth functionality.

#### Supported Devices:

* **Client Devices:** Desktops and Laptops (Windows, macOS, Linux) for accessing the web-based platform.
* **Server Requirements:** Cloud-based or on-premises server setup.

Minimum 8 GB RAM, multi-core processor for performance.

Scalable storage to handle event-related data, documents, and media

* **Peripheral Devices:**

**Smart Displays:** Used at venues to show real-time event updates and schedules

### Software Interfaces

The **Event Management System** will integrate with multiple software components to enhance functionality.

#### Key Software Connections:

* **Supported Operating Systems:**

Web Access: Windows, macOS, Linux

* **Database Management System:**

MySQL for reliable, structured data storage

* **Web Technologies:**

**Frontend:** HTML, CSS, JavaScript

**Backend:** Node.js with Express.js and RESTful API

## Functional Requirements

**F1: User Authentication & Management**

* The system shall allow new users (students and organizers) to register using email and password.
* The system shall allow all users to log in securely with their credentials.
* The system shall implement role-based access control with three roles: Student, Organizer, and Admin.
* Admin credentials shall be preloaded into the database (no sign-up for Admin).

**3.2.2**

**F2: Event Creation & Management**

* The system shall allow organizers to create events by specifying details such as title, date, venue, description, and participant capacity.
* The system shall allow organizers to edit or cancel their created events before approval.
* The system shall prevent scheduling conflicts by disallowing multiple events with "pending" or "approved" status at the same venue and date.
* The system shall save the event as "pending" until it is approved by the Admin.
* The system shall automatically remove past-date pending events from organizer and admin approval sections
* The system shall display rejected events separately on the organizer dashboard
* The system shall allow duplicate event names only if the older event is rejected or already past.

**3.2.3**

**F3: Event Registration & Participation**

* The system shall allow students to browse and view details of all approved and ongoing events.
* The system shall allow students to register for events and receive confirmation of successful registration.
* The system shall allow students to cancel their event registration before the specified deadline.

**3.2.4**

**F4: Expense Tracking (for Organizers and Admin)**

* The system shall allow event organizers and the admin to log expenses related to each event.
* The system shall allow both organizers and the admin to view and generate expense reports.

**3.2.5**

**F5: Admin Functions**

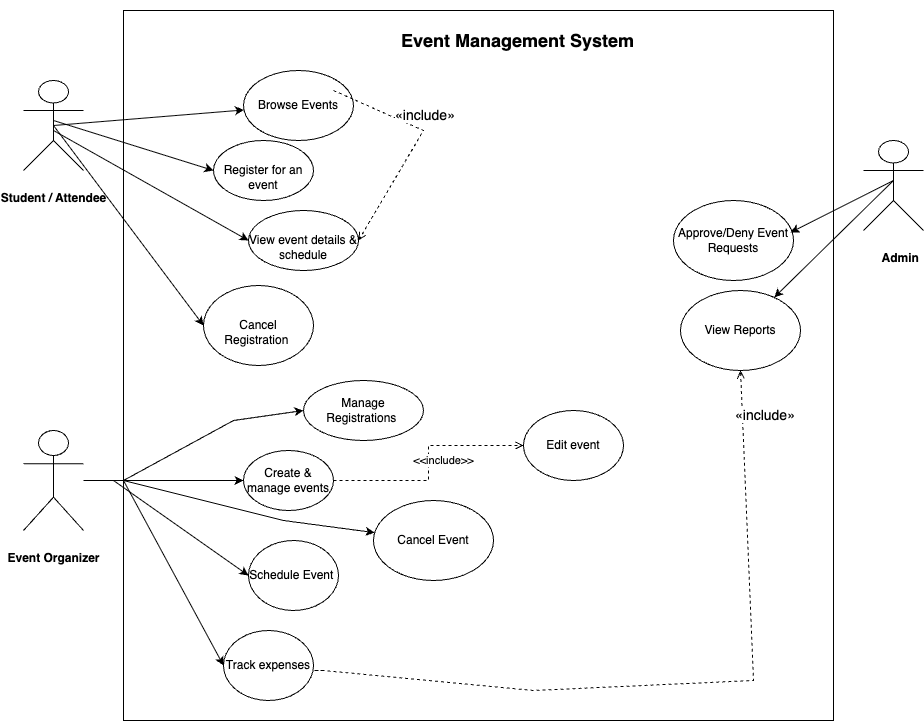
* The system shall allow the admin to approve, deactivate, or remove organizer accounts.
* The system shall allow the admin to approve or deny event creation requests submitted by organizers.
* The system shall provide access to detailed reports on events, expenses, and user activity.

**3.2.6**

**F6: Reports & Analytics**

* The system shall generate attendance reports for each event, viewable by both organizers and the admin.
* The system shall generate expense reports categorized by event, accessible to organizers and the admin.

## Use Case Model

Provided a use case diagram that will encapsulate the entire system and all actors.   
  


### **Use Case #1: Register for Event (U1)**

* **Author** – Kannikanti Chandana Priya

#### **Purpose**

The purpose of this use case is to allow a student to register for an approved event through the Event Management System. Upon successful registration, and the event's available seat count is updated.

### **2 . Requirements Traceability**

* The system shall allow students to view and register for upcoming events.
* The system shall prevent over-registration if event seats are full.

#### **3. Priority**

**High** – This feature is essential for the functionality of the system. Without this, students cannot participate in events, defeating the system’s primary purpose.

## **4. Preconditions**

* The student is logged into the system.
* The event is approved and open for registration.
* The event has available seats.

#### **5. Postconditions**

* The student is successfully registered for the selected event.
* The event's available seat count is decremented.
* If notification fails, the registration still succeeds and a log is generated for admin action.

#### **6. Actors: Student**

#### **Related Use Cases**

* **Includes**:
  + **Browse Events** *(students must view events before selecting one)*
  + **View Registered Events** *(after registration, student can track upcoming events)*
* **Extends**:
  + **Cancel Registration** *(if the student chooses to withdraw from the event later)*

#### **Flow of Events**

**1. Basic Flow**

1. Student logs in and selects **Browse Events**.
2. Student reviews upcoming approved events.
3. Student selects an event and clicks the **Register** button.
4. System checks:
   1. If the event is approved and open for registration.
   2. If seats are available.
   3. If the student has not already registered.
5. If all checks pass:
   1. The system registers the student.
   2. Updates the attendee count.
   3. Displays a success message.

**2. Alternative Flows**

* **AF1**: If the student has already registered for the selected event:  
   → System prevents duplicate registration and shows a relevant message.
* **AF2**: If the event is full:  
   → System restricts registration and displays “No seats available.”

**3. Exception Flows**

* **EF1**: If the server is down during the registration process:  
   → The system halts registration and shows an error message.
* **EF2**: If the Notification System fails to deliver a confirmation:  
   → Registration is still recorded, and a log is created for the system admin.

### **Use Case #2: Browse Events (U2)**

* **Author** – Pilli Sachethan

#### **Purpose**

This use case allows students to browse approved upcoming events, view event details, and apply filters such as date, category, or location to refine their search.

#### **Requirements Traceability**

* **F3.2.3** – The system shall allow students to browse available events with filters.
* The system shall provide event detail pages when an event is selected.
* The system shall support navigation between event listings and event detail views.

#### **Priority**

**High** – Browsing is a critical gateway for students to discover and participate in events.

#### **Preconditions**

* The student is successfully logged into the Event Management System.

#### **Postconditions**

* The student can view full details of one or more events.
* The student may optionally proceed to register for an event.

#### **Actors**

* **Primary**: Student

#### **Related Use Cases**

* **Includes**: None
* **Extends**:
  + *(None explicitly extends this use case, but it is an entry point into U1)*

#### **Flow of Events**

**1. Basic Flow**

1. Student logs into the system.
2. Student navigates to **Browse Events**.
3. System displays a list of approved upcoming events with basic information:
   1. Event name, date, venue, category, and organizer.
4. Student applies optional filters (e.g., date, category, location).
5. System updates the displayed list based on filter criteria.
6. Student clicks on a specific event to view detailed information:
   1. Description, available seats, organizer contact, and registration deadline.
7. Student may choose to register or return to the event list.

**2. Alternative Flows**

* **AF1**: No events match the applied filters  
   → System shows “No events found.”
* **AF2**: Student clears filters  
   → System reloads full list of events.

**3. Exception Flows**

* **EF1**: If the system is temporarily unavailable  
   → System displays an error message and prompts the user to retry later.

### **Use Case #3: Create Event (U3)**

* **Author** – Sameera Nallamothu

#### **Purpose**

To enable an event organizer to create a new event with necessary details (title, date, time, venue, description, and capacity). The event is added to the system and, if required, sent for admin approval before becoming publicly visible.

#### **Requirements Traceability**

* F3.1.1 – The system shall allow organizers to create events with full details.
* F3.1.2 – The system shall validate all required fields and prevent scheduling conflicts.
* F3.1.3 – The system shall support event updates, deletions, and approval workflows.

#### **Priority**

**High**

#### **Preconditions**

* Organizer is authenticated and authorized.
* Mandatory fields are filled (event title, date, time, venue, capacity, description).
* Venue is available at the selected time.

#### **Postconditions**

* Event is created and stored in the system.
  + If no approval required → it becomes visible to students.
  + If approval required → event is marked as *pending*, and admin is notified.

#### **Actors**

* **Primary**: Event Organizer
* **Secondary**: Admin

#### **Related Use Cases**

* **Includes**:
  + **U4: Approve/Deny Event** – Invoked when admin approval is needed after creation.
* **Extends**:
  + **U5: Edit Event** – Organizer may update the event after creation.
  + **U6: Cancel Event** – Organizer may cancel the event later if needed.

#### **Flow of Events**

**1. Basic Flow**

1. Organizer logs in and navigates to the **Create Event** page.
2. Organizer fills in the required fields:
   1. Event title, description, date, time, venue, and capacity.
3. System validates inputs and checks for conflicts.
4. If valid, event is saved and added to the database.
5. If approval is required, system marks event as *pending*.
   1. **Includes**: U4 – Approve/Deny Event (admin receives notification).

**2. Alternative Flows**

* **AF1**: Missing/invalid inputs  
   → System prompts the organizer to correct before submitting.
* **AF2**: Venue already booked  
   → System notifies and suggests alternatives.

**3. Exception Flows**

* **EF1**: System error on saving  
   → System displays an error message and requests retry.
* **EF2**: No admin available  
   → Event remains in *pending* state until reviewed.

### **Use Case #4: Cancel Event (U4)**

* **Author:** Donta Chaitanya Goud

#### **Purpose:**

This use case allows an event organizer or admin to cancel an event and notify all registered students about the cancellation.

#### **Requirements Traceability:**

* **F3.2.2** – The system shall allow event organizers or admins to cancel events.

#### **Priority:**

**Medium**

#### **Preconditions:**

* The actor (organizer or admin) is logged into the system.
* The event exists in the system and has not already been canceled.

#### **Postconditions:**

* The event is marked as *Canceled* and removed from the list of active events.

#### **Actors:**

* **Primary:** Event Organizer
* **Secondary:** Admin

#### **Related Use Cases:**

* **Includes**:

None

* **Extended by**:
  + **U3: Create Event** – Organizers may cancel after creating an event.
  + **U6: Approve/Deny Event** – Admins may cancel during review/approval.

#### **Flow of Events:**

**1. Basic Flow:**

1. Organizer/admin logs into the system.
2. Accessing the list of created events through the dashboard or event list page
3. Selects an existing event and clicks on **Cancel Event**.
4. System prompts for confirmation.
5. Upon confirmation:
   1. The event status is updated to *Canceled*.
   2. The event is removed from the event listing.

**2. Alternative Flows:**

* **AF1**: If no students are registered:  
   → The event is canceled silently without sending notifications.

### **Use Case #5: Approve/Deny Event (U5)**

**Author**: Susmitha Mantenna

**Purpose**:  
 To enable the **admin** to review event creation requests and either approve or deny them. Upon approval, the event becomes visible to students for registration. Upon denial, the organizer is notified. This use case also allows the admin to **track event expenses**, **view registered students**, and **generate reports**.

**Requirements Traceability**:

* F3.3.1 – The system shall allow an admin to approve or deny events.
* F3.3.2 – The system shall allow tracking expenses related to approved events.
* F3.3.3 – The system shall allow admins to view registered participants for an event.
* F3.3.4 – The system shall allow event report generation for approved/denied events.

**Priority**: High

**Preconditions**:

* The **admin** must be logged into the system.
* Events must be pending approval.

**Postconditions**:

* The event is either approved (made visible for registration) or denied (removed or flagged).
* The event status is updated in the system.
* The admin can view related data (expenses, reports, participants) for approved events.

**Actors**:

* **Admin**

**Extends**:

* **Edit Event (U6)** – Admin may make modifications before approving or denying.
* **Cancel Event (U4)** – Admin can choose to cancel even after prior approval if needed.

**Includes**:

* **Track Expense** – For approved events, the admin can monitor budgeting and spending.
* **View Registered Students** – Admin can view the list of participants once registration begins.
* **View Report** – Admin can generate summaries and insights about the event (attendance, costs, etc.).
* **Browse Events** *(students must view events before selecting one)*
* ***Create Event –*** *If the event has to be approved or denied it has to be created at the first place*

**Flow of Events**:

1. The admin logs into the system and navigates to the **pending events dashboard**.
2. The system displays a list of events that require approval.
3. The admin reviews the event details including time, venue, organizer information, and potential conflicts.
4. The admin may **edit event details** (extends Edit Event) if corrections are needed.
5. The admin chooses to either:
   1. **Approve the event**: The system changes the status to “Approved”, makes the event public, and notifies the organizer.
   2. **Deny the event**: The system removes or flags the event as denied and notifies the organizer with a reason.
6. After approval, the admin may:
   1. Track event expenses (includes)
   2. View registered students (includes)
   3. View or generate event reports (includes)
7. If necessary, the admin may **cancel** an approved event due to external constraints (extends Cancel Event).

**Alternative Flows**:

* If the event is incomplete or contains invalid data, the system prevents approval until corrected.
* If the system faces server or data errors during approval, the admin is shown an error message and asked to retry.

### **Use Case #6: Track Expense (U6)**

**Author**: Sanjana Singam

#### **Purpose:**

The objective of this use case is to allow an event organizer to track and manage expenses related to an event. The organizer can input different types of expenses, such as venue booking fees, catering costs, and logistical expenses. The system stores and categorizes these expenses, enabling the organizer to monitor total costs and generate reports. This feature helps ensure proper budgeting and financial planning for events.

#### **Requirements Traceability:**

* **F3.3.1** – The system shall allow organizers to log event-related expenses by specifying expense categories, amounts, and descriptions.
* **F3.3.2** – The system shall store these details in the database and allow organizers to edit or delete expense records if needed.
* **F3.3.3** – The system shall generate expense reports summarizing total expenditures and categorizing costs for financial tracking.

#### **Priority: Medium**

#### **Preconditions:**

* The event organizer must be logged into the system and have permissions to manage expenses for an event.
* The event must already exist in the system, and expense details (category, amount, and description) must be provided before logging an expense.
* The event must have been approved by the admin through **Approve/Deny Event (U5)** (i.e., the event status must be "Approved") before the organizer can log expenses.

#### **Postconditions:**

* Once an expense is logged, it is stored in the system and associated with the corresponding event.
* The total event budget is updated to reflect the new expense.
* The organizer can view a breakdown of all costs.
* If the organizer chooses to generate a financial report, the system compiles expense data and presents an overview of total spending.

#### **Actors:**

* Event Organizer
* Admin (in some cases, for approval)

#### **Extends:**

* This use case does not extend any other use cases.

#### **Includes:**

* **Approve/Deny Event (U5)** – The event must be approved before any expense tracking can begin. This use case includes **Track Expense** after the approval of the event.

#### **Flow of Events:**

1. The **event organizer logs into the system** and navigates to the **Expense Tracking** section.
2. The organizer selects the **approved event** for which they want to record an expense.
3. The organizer enters details such as:
   1. **Expense Category** (e.g., venue, catering, transportation)
   2. **Amount** (total cost of the item/service)
   3. **Description** (details about the expense)
4. The system **validates** the input and stores the expense in the database.
5. The system **updates** the event’s total expenses and provides an updated financial summary to the organizer.
6. If needed, the organizer can **generate a financial report** summarizing the total and categorized expenses for the event.

#### **Alternative Flows:**

* **Invalid Data**: If the organizer enters an incorrect or incomplete expense entry (e.g., missing amount or category), the system prompts them to correct the information before proceeding.
* **Modify/Delete Expense**: If the organizer wishes to modify or delete an expense, the system allows them to update or remove expense records. The event’s total budget is updated accordingly.

#### **Exceptional Flows:**

* **System Failure**: If the system encounters a failure while saving the expense (e.g., database error), an error message is displayed, and the expense is not recorded.
* **Report Generation Unavailable**: If financial reporting is temporarily unavailable due to system maintenance, the organizer is notified that reports cannot be generated at that time.

# Other Non-functional Requirements

## Performance Requirements

The *Event Management System*, when deployed, is expected to operate efficiently under standard university use. The following performance requirements outline anticipated benchmarks and guide future optimization:

* **System Responsiveness (if deployed)**:
  + The system should ensure that pages and core functionalities load within **2 seconds** under normal internet conditions.
  + Key actions such as event creation, registration, and approval should be processed with minimal delay.
* **Concurrent User Handling (post-deployment)**:
  + The system should be capable of supporting at least **100 concurrent users** without noticeable performance lags, ensuring a smooth experience during peak usage (e.g., event registrations).
* **Scalability (if expanded)**:
  + The design should support scaling to a larger user base or multi-institutional deployment with minimal refactoring.
* **Data Access Performance (when integrated with live database)**:
  + Common operations like viewing events, retrieving reports, and accessing student registration details should complete in **under 1 second** for average dataset sizes.
* **Server Uptime (upon hosting)**:
  + The system should aim for **99% uptime**, barring maintenance windows, to ensure continuous accessibility.

**Note:** These requirements serve as performance goals to be validated and tuned during system deployment and real-world testing.

## Safety and Security Requirements

The *Event Management System* must ensure the confidentiality, integrity, and availability of data, particularly user credentials, event records, and participant information. The following safety and security requirements are expected to be implemented during development and enforced upon deployment:

**Authentication and Authorization**:

* Role-based access control must be enforced.
* Admin credentials should be hardcoded and encrypted in the database to prevent unauthorized access.
* Organizers and Students must authenticate using secure login credentials. Unauthorized users must be denied access to all system features.

**Data Protection**:

* Passwords and sensitive user data should be stored using hashing algorithms (e.g., bcrypt).
* User input must be sanitized to prevent SQL/NoSQL injection and cross-site scripting (XSS) attacks.

**Secure Communication (if deployed online)**:

* The system should use HTTPS for all communications between the client and server to protect data in transit.

**Data Backup and Recovery (future deployment)**:

* Regular database backups should be scheduled to prevent data loss.
* A recovery plan should be in place to restore system operations after failures.

**User Privacy**:

* Student details should only be accessible by Admin and relevant Organizers.
* No personal information should be publicly exposed.

## Software Quality Attributes

**Reliability**

* The system is expected to perform consistently without crashes or data loss during normal usage.
* Error handling mechanisms will notify users gracefully without system breakdown.

**Usability**

* The system provides an intuitive, clean, and role-specific interface.
* Navigation is straightforward for all users (Admin, Organizer, Student), with clearly labeled actions like “Create Event”, “Approve”, “Register”, etc.

**Maintainability**

* The codebase is modular and structured, making it easy to fix bugs, introduce new features, or enhance existing functionalities.
* Documentation will be provided for each module to help future developers.

**Scalability**

Although initially developed for a single university, the system is built with an architecture that can be extended to support multiple institutions and higher user volumes if needed.

**Security**

* Role-based access control, password hashing, and secure authentication protocols are enforced to protect data and prevent unauthorized access.

**Performance**

* The system is designed to respond quickly to user actions (e.g., under 2 seconds page load time), and efficient queries ensure fast data retrieval.

**Availability (when deployed)**

* The system is expected to be available 24/7 with minimal downtime, especially during peak usage periods such as event registration.

These attributes aim to ensure a smooth experience for end-users and an efficient foundation for future growth and adaptation.

# Other Requirements

This section includes additional requirements that are essential for the smooth functioning, deployment, and future adaptability of the system but do not fall strictly under functional or non-functional categories.

#### **5.1 Compatibility Requirements**

* The system should be compatible with all major modern browsers (Google Chrome, Mozilla Firefox, Microsoft Edge).
* It should support both desktop and mobile responsive views for enhanced accessibility.

#### **5.2 Backup and Recovery**

* Regular database backups must be scheduled to prevent data loss.
* The system should provide a mechanism for manual backup and recovery (planned for deployment stage).

#### **5.3 Legal and Compliance**

* The application must not store or share student data with third parties without consent.
* Compliance with institutional data protection policies is mandatory.

#### **5.4 Future Enhancement Scope**

* Add functionality for bulk student registration via CSV.
* Enable notification features via email/SMS on event updates.
* Expand support for multiple admin roles with limited privileges.

Appendix A – Data Dictionary

The table below defines key variables, constants, inputs, and outputs used in the system.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| event\_id | Integer (PK) | Unique identifier for each event |
| event\_name | String | Name of the event |
| event\_date | DateTime | Scheduled date and time of the event |
| venue | String | Location of the event |
| organizer\_id | Integer (FK) | Reference to the organizer’s user ID |
| user\_id | Integer (FK) | Reference to the attendee’s user ID |
| Total budget | Float | Estimated budget for the event |
| expenses | Float | Actual expenses incurred |

Appendix B - Group Log

**Appendix B: Group Log**

This log tracks meeting sessions, discussions, and team effort.

|  |  |  |  |
| --- | --- | --- | --- |
| **Meeting #** | **Date** | **Duration (hours)** | **Topics Discussed** |
| 1 | Jan 25, 2025 | 1.5 | Project initiation, team role assignment |
| 2 | Jan 29, 2025 | 1.5 | Finalizing SOW, discussing core system features |
| 3 | Feb 2, 2025 | 1.5 | Database schema, UI/UX design discussion |
| 4 | March 8,2025 | 2 | SRS |
| 5 | April 5, 2025 | 1.5 | SDD |
| 6 | May 6, 2025 | 1 | Software test plan |
| 7 | May 6, 2025 | 2 | Final documentation and demonstration video |

Total Meeting Time: 11 **hours**