

Event Management System

**Authors:**

Muhammad Bin Nasir 220201012

Aiza Shafqat 220201036

Muzammil Ali 220201061

Ali Jawad 220201013

**Affiliation:**

Department of Computer Science

Institute of Space Technology

**Date:** 4th December, 2024

**Document Version:** Final

## 1. Introduction:

The purpose of this section is to provide a foundational understanding of the Event Management System (EMS) and establish the framework for its development. Results It guarantees that all participants will agree on the aims, scope, and outputs of the project.

### 1.1. Purpose

Event management system (EMS) is designed to change the way events are planned, carried out and managed. Its purpose includes:

1. **Centralized Event Management:** Offer a single, easy-to-use platform that combines the planning, coordination, and control of events of different sizes.
2. **Efficiency in Operations:** Identify important issues that event organizers have to deal with, including manual ticketing, complex scheduling, or poor attendee management tools, etc.
3. **Streamlined Communication:** Promote smooth communication among the parties involved (organizers, vendors and attendees), by providing successful synergies and regular updates for the benefit of all involved.

### 1.2. Scope

The EMS is envisaged as an adaptive platform for meeting the evolving needs of event planners. Its scope includes:

1. **Core Features:**

* Event creation with customizable templates.
* Real-time updates for last-minute changes or announcements.
* Automated ticket sales and payment handling.
* Comprehensive post-event reporting to measure success and gather insights.

1. **Target Audience:**

* **Corporate Planners:** Professionals managing seminars, product launches, and employee engagement activities.
* **Concert Organizers:** Event producers handling large-scale musical events with complex logistics.
* **Personal Event Hosts:** Individuals planning weddings, birthdays, and other personal occasions.

1. **System Highlights:**

* Integration with popular payment systems (e.g., PayPal, Stripe).
* Mobile-friendly interface for on-the-go access and management.
* Templates offering customizability to suit the requirements of individual users.

### 1.3. Definitions, Acronyms, and Abbreviations

In order to ensure a clear project documentation for all the stages of the work, the following terms and acronyms are defined.

1. **EMS:** Event Management System - the core platform being developed.
2. **Event Organizer:** The person or group in charge of planning and executing the event.
3. **Attendee:** A participant who registers for or attends the event.
4. **API:** Application Programming Interface- provides integration with third party services like payment systems, CRM systems, and social media services.

### 1.4. References

The system is configured on the basis of these documented industry standards and knowledge, including:.

1. IEEE 830-1998 (SRS Standard): Provides a codified method for the specification of software systems requirements.
2. Industry Case Studies: Offers practical information on the best practices and tools for event management, which guides the EMS design.

## 2. Overall Description

### 2.1. Product Perspective

The EMS has a distinctive value as it is independent and integrating at the same time.

* **Standalone System:**Functions independently as a comprehensive solution for event management, eliminating the need for additional tools.
* **Integration Capabilities**:
* Seamlessly interfaces with third-party tools such as
* **CRM Platforms:** For customer relationship management.
* **Payment Gateways:** Ensuring secure and efficient financial transactions.
* **Marketing Tools:** Streamlining event promotion through social media or email campaigns.
* **Competitive Advantage:** Competitive Advantage:

Combining those key features into one integrated system simplifies the workings of the EMS for users, eliminates the need for multiple pieces of technology, and provides more streamlined workflows.

### 2.2. Product Functions

Key features of the EMS include:

#### Event Creation:

Allow the user to create events with information, including names, date, location, and agenda. Supports reusable templates for consistency and efficiency.

#### Ticket Management:

Provides tools for selling, reserving, and managing event tickets. Features include dynamic pricing, discount codes, and attendee confirmations.

#### Attendee Tracking:

By having real-time track registrations, it provides features such as QR code check-ins to perform convenient registration for entering the building.

#### Notifications:

Automatic email and SMS updates are sent to attendees to alert them about changes, reminders, or promotional offers.

#### Analytics:

Provides comprehensive post-event reports, including attendance data, results and survey data, which enables organizers to assess performance and to pin down potential improvements.

### 2.3. Product Environment

The EMS is designed to be used in the following environment

#### Deployment:

Cloud-based hosting for scalability and accessibility.

Web browsers (Chrome, Firefox, Safari) and mobile platforms (iOS, Android) compatible.

#### Hardware Requirements:

* End-user devices: Smartphone (with at least 4GB RAM), tablet, or personal computer.
* Servers: High-availability cloud servers with redundant storage and backups.
* Software Requirements: Software Requirements:
* Operating systems: Windows, macOS, Android, and iOS.
* Internet browser: Support for HTML5, CSS3, and JavaScript.

### 2.4. Assumptions:

The EMS assumes the following conditions for effective operation:

* Users will be able to use stable internet to get real-time information and alerts.
* Event organisers will have basic computer literacy in order to operate the system and use it successfully.
* Integrations with payment gateways (e.g., Stripe or PayPal) will be working and ready to be used in the target countries.
* The majority of users will access the system via web browsers or mobile applications.

### 2.5. Dependencies

The EMS is supported by a number of external systems and technologies for core activities:.

1. **Payment Gateways:**Reliance on securely and verified payment processing services (e.g., Stripe, PayPal) for both ticket sales and refunds.
2. **Third-Party APIs:**Integration with CRM tools for attendee management and social media platforms for event promotion.
3. **Hosting Providers:**Usage of cloud infrastructure as a service (e.g., AWS, Azure) for data storage, processing and availability (i.e.
4. **Regulatory Compliance:**Adherence to data protection regulations such as GDPR and PCI DSS for secure processing of personal and payment data.

### 2.6. Constraints

Although the EMS provides strong functionalities, it is also subject to some restrictions:.

* **Technical Constraints:**
  + Requires a stable internet connection for up to date/real time reporting as well as notification.Modern web and mobile browsers compatible, with accessibility.
* **Budgetary Constraints:**
  + Built to be affordable for individual event planning businesses, and top-shelf for enterprise sized events.
* **Legal Constraints:**
  + Complies strictly with global data protection norms, such as the General Data Protection Regulation (GDPR), with respect to the privacy and secure storage of sensitive data.

## 3. Specific Requirements

### 3.1. Functional Requirements:

#### User Authentication:

* Users must enter valid credentials to login.
* System will verify the credentials of the user from the database.

#### Event Creation:

* Admins and some authorized users can create, manage, delete and supervise events.
* Event details include event name, event venue (including remotely), event timings, motive and participant limit.

#### Participant Registration:

* Users can register for the available events.
* System manages overbooking by taking into account the participant limit of any said event.

#### Event Notifications:

* Users can opt to receive the notifications as mail, sms or in the form of pop up notifications.
* The system will generate the notifications for upcoming events, cancellation of events, or any updates.

#### Payment Processing:

* Integration done with Paypal and HBL secure payment for both International and Domestic (Pakistan) payments.
* Upon payments, system will generate payment receipts and send them to the provided email address and mobile number of the user,

#### Report Generation:

* Admins can generate detailed reports on event participation, including attendee numbers and demographics.
* The system provides insights into the revenue generated from ticket sales and other financial metrics.
* Feedback collected from participants is compiled into the report to assess event success and areas for improvement.

### 3.2. Non-functional Requirements:

#### Performance:

* The System should be able to handle up to 2000 users concurrently logged in to it.
* It will be upgraded as per requirements.

#### Scalability:

* The architecture must support further additions, like the support for vendors.

#### Usability:

* The system should be understandable to a lay user, and must be easy to use.
* The user interface must allow user to perform basic tasks (eg. registering for an event) within three clicks.

#### Reliability:

* 99% uptime, with failover backup and recovery.
* In case of a crash, the system must save the current state for a user, and resume the activity where it was left off.

#### Compatibility:

* Ensures compatibility across all major browsers, including Firefox, Chrome, and Edge.
* Fully supports HTML5 standards for consistent functionality and user experience.
* Provides seamless performance on desktop and mobile versions of supported browsers.

### 3.3. System Features:

#### Role Based Access:

* Admins have access to advanced controls for managing events, users, and system settings.
* Users are provided with intuitive interfaces for event browsing, registration, and feedback submission.
* Organizers can access tools to create, update, and monitor events, along with participant management options.

#### Feedback and Reviews:

* After the events, participants can provide a feedback for the event, which is only visible to the admins.

#### Search and Filter:

* Advanced search filters such as event category, date, or venue.

#### Calendar integration:

* Event can directly be added to the Google calendar or outlook directly from the system for a reminder.

## 4. External User Interface:

### 4.1. User Interface:

#### Web Interface:

* Built on HTML5, can run on all supported browsers.
* Responsive design for both desktop and mobile browsers.
* Easy navigation with easy-to-use dashboard, search bar and tabs for various different tasks.

#### Mobile App Interface:

* A native app for both Android and iOS.
* Similar functionality to the web interface.

### 4.2. Hardware Interface:

#### Server:

* Minimum: Intel Xeon MP3.0, 16GB RAM, 500GB storage
* Recommended: Intel Xeon Platinum 8360H or AMD EPYC 7F72, 64GB RAM, 4TB SSD storage.

#### Client:

* Device with at least 2GB RAM and a browser with HTML5 support.

### 4.3. Software Interface:

#### Operating System:

* Server: OS compatible with Windows Server, Linus (Ubuntu, CentOS)
* Client: Windows 10 or 11, macOS, Linux with HTML5 support, Android, iOS.

#### Database:

* MySQL or PostgreSQL for Database Management System at server side.
* No requirement at client side.

#### APIs:

* Rest APIs for payment gateway integration and notification service.

#### Third Party tools:

* Integration with platforms like Zoom for virtual events.

### 4.4. Communication Interface:

#### Network Protocols:

* System uses a secure HTTPS protocol to ensure the security of communication between server and client.
* All sensitive information, such as payment information, personal information is encrypted before transmission.
* SSL and TLS certificates must be implemented to validate the server’s identity and provide secure connection.
* Redirection of all HTTP requests to HTTPS to reduce the weak points.

#### Email / SMS Gateway:

* The system must integrate with a reliable Simple Mail Transfer Protocol (SMTP) server to send automated email notifications for various events.
* The Email server should support different encryption and authentication technologies such as STARTTLS.
* The system should also have a mechanism to queue the mails in case the server is down.

#### SMS Notifications:

* The system will integrate with a trusted SMS gateway, such as Nexmo.
* It should support both domestic and international SMS delivery.
* SMS service will primarily be used for the important notifications, such as payment details, event updates, or last minute cancellations.

#### Real time Communication:

* Websocket enables real time updates about registration status such as confirmation and waitlist alerts.
* It also helps have a real time chat during events for participant interaction and support.
* Moreover, it updates the event details on user interfaces dynamically, without any refresh of the page required.

## 5. Non-Functional Requirements

#### 5.1. Introduction

The qualities and characteristics of the EMS are described as non-functional requirements that enable it to meet the needs of its users and fulfill its technical standards. These requirements focus on reliability, performance, usability, scalability, security, and maintainability for the system.

#### 5.2. Performance Requirements

Performance is a function of responsiveness, speed, and efficiency in various workloads in the system.

* The system should respond to user interactions such as page loading and form submissions within 2 seconds
* Ticket booking and payment processing must complete within 5 seconds.
* Event search and filtering results must be returned within 2 seconds regardless of the size of the database.
* Real-time monitoring should monitor system performance and log delays over 3 seconds for diagnostic purposes.

#### 5.3. Scalability Requirements

Scalability is the ability of the system to handle growth in users, events, and data.

* The system should handle 1000 concurrent users in normal conditions and be able to scale to support 5000 concurrent users during peak demand.
* The database must store and retrieve up to 10 million records, which include attendee data, event details, and transaction logs.
* The architecture must support horizontal scaling, where servers or cloud instances are added as the user demand grows.
* Leverage cloud platforms such as AWS, Azure, to allocate resources in real-time according to the traffic.

#### 5.4. Availability Requirements

Availability requirements ensure that the system is available and reliable for users.

* The system must have 99.9% uptime, therefore a maximum downtime of only 8.76 hours in a year
* Scheduled maintenance downtime must occur at least between 12 AM to 4 AM and there has to be at least 48 hours advance notice
* Employ redundant servers and failover mechanisms to ensure uninterrupted service during hardware or software failures.
* A disaster recovery plan should restore operations within 30 minutes after a major system failure, using off-site backups.

## 6. Preliminary Schedule

The development of the Event Management System (EMS) will follow an agile methodology with multiple sprint. Following are the six development phases.

#### 6.1. Requirement Analysis

* Objectives: Determine the user requirements and system goals.
* Stakeholders: Project managers, business analysts, and key users.

#### 6.2. System Design

* Objectives: Establish the technical foundation for the system.
* Activities: Create architecture diagrams to define system components and their interactions.

#### 6.3. Development

* Goals: Build the system’s functionality.
* Activities:

1. Building frontend interfaces (for example: event listings, booking forms).
2. Backend logic such as payment processing and event scheduling

#### 6.4. Testing & QA

* Goals: Ensure that the system is bug-free, secure, and operates as expected.
* Activities:

1. Perform unit testing, integration testing, and UAT.
2. Fix bugs found during testing.

#### 6.5. Deployment

* Goals: Transition the EMS from development to live production.
* Activities:

1. Configure the production server.
2. Deploy the system and measure performance metrics after deployment.

#### 6.6. Post-Deployment Support

* Goals: Provide ongoing support and maintenance.
* Activities:

1. Monitor system performance.
2. Minor updates and resolve user-reported issues.

## 7. Budget

### 7.1. Purpose

This report aims to present an approximate budget for the development of an Event Management System (EMS). The system will help event coordinators manage events, registrations, ticket sales, and communications.

#### 7.1.1 Scope

The scope is divided into the following budget areas:

1. Development and testing
2. Hosting and infrastructure
3. Licenses, marketing, and contingency funds
4. Web-based and mobile-friendly application

### 7.2. Budget Breakdown

#### 7.2.1. Cost of Development

The actual most important cost is associated with the development phase in the salaries of developers, designers, and project managers during that time. This includes:

1. **Developers:**

* Two skilled developers will work on the system's frontend and backend for 4 months.

1. **UI/UX Designer:**

* A designer will focus on creating an intuitive and visually appealing interface for both web and mobile platforms and work for 3 months.

#### 7.2.2. Infrastructure and Hosting

Infrastructure and hosting costs ensure that the event management system is reliable and accessible 24/7 for both current and future demands of users. These will include:

1. **Cloud Hosting**

* This system will be hosted in a scalable cloud platform (like AWS or Azure) which will handle traffic efficiently. Hosting is planned at $3,000 per year.

1. **Backup and Storage**

* The data for the events, users, and transactions will be secured by regular data backups.

#### 7.2.3. Software and Licensing

The Software and Licensing Costs are costs of tools and resources necessary to build and run the Event Management System efficiently. These costs will provide access to professional-grade development environments, frameworks, and third-party services:

1. **Development Tools**

* Licenses for Integrated Development Environments (IDEs) like Visual Studio or JetBrains.

##### 

1. **API Integrations**

* Fees for integrations with third-party services that will be used in a project like payment gateways and email notification systems .

#### 7.2.4. Marketing and Promotions

The Marketing and Promotions budget will support the successful launch and uptake of the Event Management System by reaching its target audience through the following:

1. **Digital Marketing**

* SEO, email campaigns, and social media outreach including Facebook, Instagram, and LinkedIn.

1. **Advertisements**

* Paid advert placements on Google Ads and platforms on social media.The campaigns target specific groups, such as event hosts and attendees, to be aware of the service and to find users as fast as possible.

#### 7.2.5 Contingency Fund

A contingency fund is a set aside for unexpected expenses which might arise during the development or deployment of the Event Management System. These might include:

1. **Unexpected technical issues**

* Debugging of unknown bugs or delays in the development of the system.

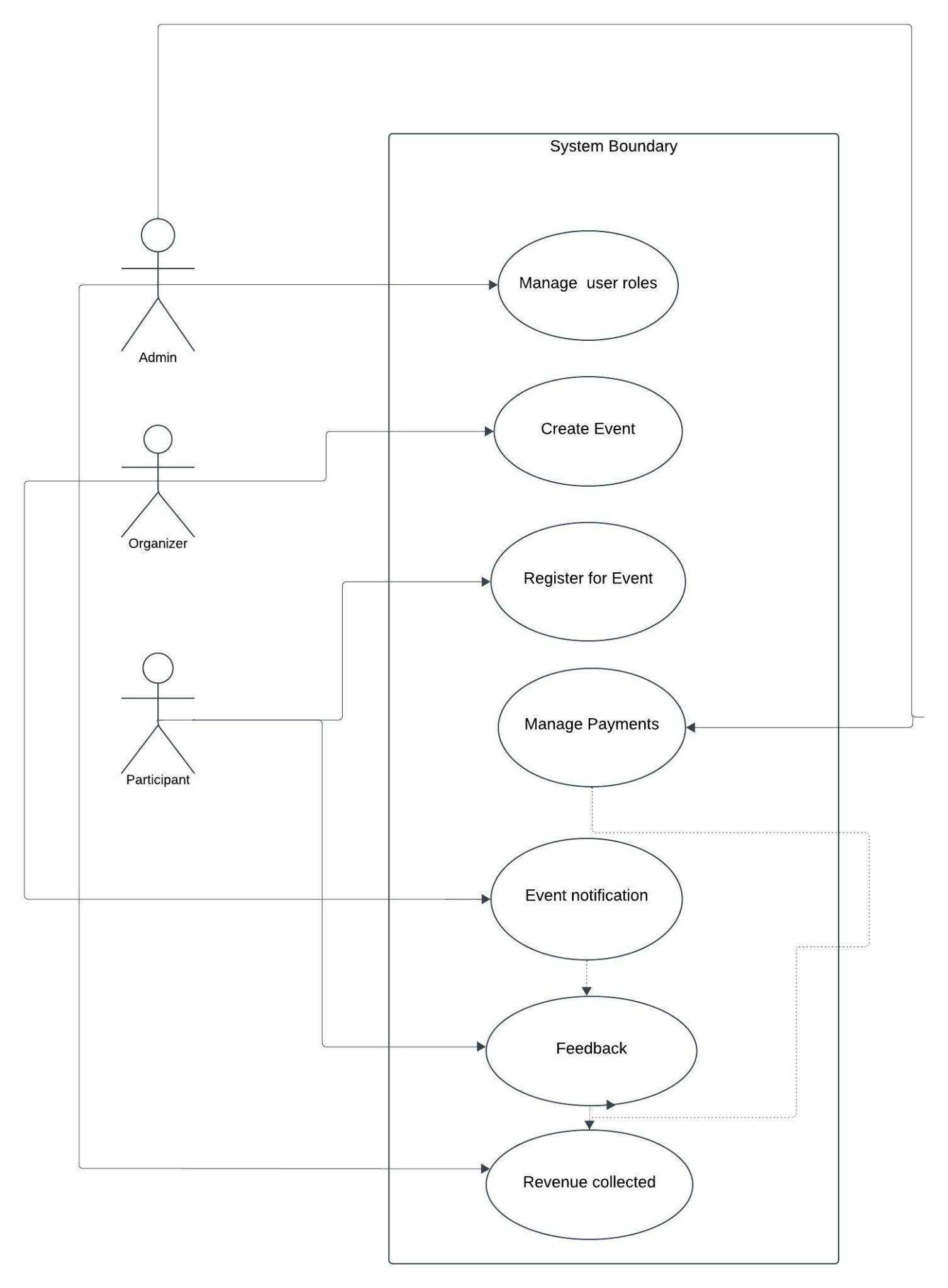
1. **Extra resources**

* Hiring temporary staff or consultants to do specific tasks.

1. **Hardware or software upgrade**

* Procurement of extra tools or infrastructure required to meet the demands of the project.

# **8. Use Case Diagrams**



### 

## **8.1. Actors and Their Roles**

#### **8.1.1. Admin**

**Responsibilities:**

1. Manage user roles (add, remove, or update event organizers, vendors, or attendees).
2. Monitor feedback provided by users to ensure the quality of events.
3. Oversee financial aspects such as payments and revenue collected.

#### **8.1.2. Event Organizer**

**Responsibilities**:

1. Create and manage events (e.g., scheduling, setting event details like venue, time, etc.).
2. Monitor registrations and handle event notifications for updates or changes.

#### **8.1.3. Attendee**

**Responsibilities:**

1. Register for events created by organizers.
2. Provide feedback after attending an event.

## **8.2. Use Case Descriptions**

#### **8.2.1. Manage User Roles (Actor: Admin)**

The Admin assigns and manages roles for Event Organizers, Vendors, and Attendees. For example:

1. Adds new users and assigns appropriate roles.
2. Removes inactive or unauthorized users.

#### **8.2.2. Create Event (Actor: Event Organizer)**

1. Event Organizers can create new events by entering details such as the event title, description, date, and venue.
2. They can also modify or cancel existing events.

#### **8.2.3. Register for Event (Actor: Attendee)**

1. Attendees can browse available events and register for them by providing required details.
2. Notifications are sent to confirm successful registration.

#### **8.2.4. Manage Payments (Actors: Admin, Attendee)**

1. Payments are tracked for event registrations.
2. The Admin monitors all payment flows.
3. Vendors are notified of payments received for services provided.

#### **8.2.5. Event Notification (Actor: Event Organizer)**

Notifications about events, updates, or cancellations are sent to attendees by the Event Organizer.

#### **8.2.6. Feedback (Actors: Attendee, Admin)**

1. Attendees can provide feedback about an event.
2. The Admin reviews the feedback to monitor the quality of events.

#### **8.2.7. Revenue Collected (Actor: Admin)**

The Admin generates reports on total revenue collected from registrations and payments made

## **8.3. Description of the Diagram**

Actors: The roles are administratively defined as Admin, then as Organizer and finally as Participant.

System Boundary: Outlines the demarcation of the system which, in simple terms, separates internal processes from other entities.

#### **8.3.1. Relationships:**

Tying of actors directly to use cases.

#### **8.3.2. Include and Associativity Relationships:**

“Manage Payments” **includes** “Revenue Collected.”

“Include” relationship is used when we see that a use case is dependent on other use case then we give them the relationship of “include” to specify more specification of it.

Other than that direct relationships , also known as “**associations**” are used in all of the use cases as well as in the actors and relationship between among them

Interdependency points the financial circulation of the system.

**Sequential Workflow**: Prompts and feedback are both elicited after a registration phase and after a completion of event offering.

## **8.4. Importance of Use Case Diagram in SRS Specification:**

But a Use Case Diagram is very important in creating a Software Requirements Specification (SRS) since it clearly defines how the system will interface with the various users. Here's how it contributes to the development process, with references to the context of the provided diagram:

#### **8.4.1. Clarity of Functionality Displayed about the Functional Requirements**

The main working operations of the system can be identified on the diagram, including the user roles, creation of events, registration for an event, and payments.

For example, the “Manage Payments” use case demonstrates the monetary transactions involved, and “Revenue Collected” guarantees a proper financial reporting.

This simple picture making provides the stakeholders or developers with a very clear picture of exactly what is expected of the system in the future.

#### **8.4.2. Identification of system scope, to understand and develop its importance there is need of following the steps below:**

The positioned system’s boundary in the diagram clearly identifies internal processes and external actors, including Admin, Organizer, and Participant.

By setting such boundary, developers and stakeholders are able to concentrate on the salient part within the system and avoid a situation where they are drawn in other unrelated areas by the project’s scope.

#### **8.4.3. Role-Based Understanding**

Different actors' roles and responsibilities are explicitly defined:

Admin: Includes such activities as user roles management.

Organizer: Is mainly concerned with the idea creation and the management of funds.

Participant: Uses it only for registration in the system, notifications also gives feedbacks.

The most important is that this differentiation helps other stakeholders adjust their expectations to the capabilities of the system and vice versa, and thus avoid confusion.

#### **8.4.4. Visualization of Dependencies**

The inclusion (for instance the “Manage Payments” include “Revenue Collected”) is an indication of how some activities depend on others.

This helps provide developers a view of the flow of the actions so that the interrelated features are well designed.

#### **8.4.5. Improves Stakeholders’ Interacts :**

Elaborate and regularized, the system’s features and the flow of work become easily understandable to individuals who are not necessarily technical experts, such as the clients.

For instance, the “Register for Event” use case provides an easy understanding of the main steps the Participant takes from the registration of an event to notification.

#### **8.4.6. Facilitates Easy System Design :**

This contributes to easy system design, since most computer hardware components are rather similar to one another, and variations between them are typically only in terms of speed or size.

The use of the diagram is to find certain structures that will help the development team in the process of system modularization.

For example, a developer can code for event genesis, payments, and notifications as each module with knowledge of how other sections function.

#### **8.4.7. Contributes to Find out Missing Requirement:**

Looking at the interactions one can easily say that some use cases or functionalities are missing. For instance if notifications were not incorporated it is clear that the missing segment of the application is notifications.