

**FACULTY OF TECHNOLOGY**

**BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY**

**UNIT:** PROJECT PROGRAMMING

**UNIT CODE:** BIT 03105

**PROJECT TITLE:**  KCAU EVENT MANAGEMENT SYSTEM

**DOCUMENT:** SYSTEM DESIGN DOCUMENT

**NAME:** AUGUSTINE ODUMA

**REG NO:** 21/04921

**SUPERVISOR:** Mr. Clive Onsomu

# Introduction

# Purpose

An inventive and effective system can be developed with the help of the Software Design Document, which is critical for ensuring that the developed system meets the required specifications, is maintainable, and can be properly tested. It is also a resource for system developers, testers, and other stakeholders involved in the development process. In order to guarantee that the developer fully comprehends how the system should be built, it attempts to offer a comprehensive blueprint that includes both narrative and graphical representations. From idea to execution, this document will play a crucial role in directing the software development process.

The SDS for KCA University's Event Management System (EMS) is specifically intended to provide information on the design considerations and minute details particular to this specialized software. This includes detailing how the system will make event planning easier, enable enrolling users in a user-friendly manner, and guarantee precise tracking of attendance. The SDS for the EMS at KCA University turns into a customized manual that offers insights into the architecture, procedures, and design components particular to the difficulties and goals of improving the general event experience for the university community.

# Scope

The purpose of this event management system at KCA University is to improve and streamline event planning and attendance. The system's user-friendly enrollment procedures, accurate attendance tracking, and simplified event creation empower users, including administrators and students. Specifically designed for KCA University, the system's reach includes:

* *Streamlined Event Creation:* The system makes it simple for administrators to plan and oversee events, guaranteeing a simple procedure for arranging a range of university-wide activities.
* *User-friendly Enrollment Procedures:* Enrolling students in events is simple and encourages greater engagement and participation. The system's goal is to offer a user-friendly interface that promotes engagement.
* *Precise Attendance Monitoring:* The system enables precise monitoring of event attendance, providing managers with trustworthy information for well-informed budgetary and decision-making decisions.

Improving the university community's overall event experience is the main objective. In order to meet the specific requirements and goals of KCA University, this scope has been carefully defined. This will guarantee that the Event Management System plays a crucial role in improving the university's event management procedures.

# System Design Constraints

Various limitations that shape the development of the KCA University Event Management System (EMS) have an impact on the system design. These constraints include user-specific requirements as well as the technological landscape. Here are some important constraints to take into account:

* *Challenge in Technology Compatibility:* Ensuring smooth communication and data exchange among the different components of the EMS system.
* *Difficulty scaling up the system:* As the system expands, it may become more difficult to make sure it can accommodate additional users and data without experiencing performance problems.
* *Difficulty in Putting Security Measures in Place:* It can be challenging to put basic security measures in place to safeguard user data because hackers are always coming up with new ways to get access to systems.
* *The Difficulty of Creating User-Friendly Interfaces:* It can be difficult to keep it simple and intuitive, but it is critical for users to navigate and interact with the system without confusion.
* *Difficulties with Data Management:* During development and testing, dealing with initial data setup and migration between different environments may be difficult.

# Design Goals and Objectives

**Goal:** The primary goal is to develop an efficient and reliable Event Management System for KCA University, which will simplify event processes and provide a positive user experience.

The design objectives for KCA University's Event Management System (EMS) have been carefully designed to ensure the creation of a robust, user-friendly, and efficient system that caters to the university community's unique needs. The design process is guided by the following **objectives**:

* To increase user engagement and satisfaction, create an intuitive and visually appealing interface.
* Build flexibility into the system so that future requirements can be easily accommodated without extensive redesign.
* Implement a user-friendly design to make interactions between administrators, staff, and students easier.
* Enhance security by implementing industry best practices in authentication, authorization, and data encryption.
* Create the system with scalability in mind, to accommodate growing user bases and data volumes.

# Overview of Document.

This document is divided into sections, each of which focuses on a different aspect of the Event Management System:

*Section 2* - System Architecture: Describes the overall structure and deployment strategies designed specifically for KCA University.

*Section 3* - Process Design: Use case diagrams, activity diagrams, and other visuals are used to demonstrate how the system works.

*Section 4* - Database Design: Describes data organization techniques such as normalization and entity-relationship diagrams.

*Section 5* - Interface Design: Emphasizes the design of various forms and interfaces to provide a consistent user experience.

*Section 6* - Conclusion: Summarizes everything in the document, ensuring that everyone involved in the development process has a clear understanding.

# System Architecture

# Introduction

System architecture serves as the blueprint and organizational framework for designing and building complex software systems. It includes the high-level structure, components, modules, and connections between them, giving a complete picture of how a system works.

In essence, system architecture defines the fundamental design principles and guidelines that govern software application development. It addresses critical issues such as data flow, system modules, communication protocols, and component interaction.

System architecture's main responsibility is to make sure the software system effectively satisfies both functional and non-functional requirements. It provides the foundation for developers to better understand how different system components interact, promoting efficient development procedures and effective collaboration.

Numerous benefits come with a well-designed system architecture, such as component reusability, maintainability, and scalability. It offers a road map for the effective development and implementation of systems by directing decisions about technology selections, deployment tactics, and system behavior.

In conclusion, system architecture serves as a structure for software system design, development, and operation. It serves as a guide to make sure the finished product complies with the aims and objectives stated in the planning stage.

# The Client-Server Model

The client-server architectural model was chosen for the EMS. The client in this model is the React-based front-end, which is in charge of the user interface and interactions. Meanwhile, the Spring Boot back-end serves as the server, managing business logic, handling client requests, and interacting with the MySQL database. This model ensures that concerns are clearly separated, allowing for scalable and maintainable development.

Front-End (React):

* Manages the user interface, ensuring a responsive and engaging experience.
* Handles user input, validation, and state management.
* Navigates within the application using client-side routing.

*Back-End (Spring Boot):*

* Executes core business logic, processing requests and validating data.
* Interacts with the MySQL database for data storage and retrieval.
* Manages authentication, authorization, and server-side routing.

# Design Approach

The Event Management System (EMS) design approach adopted a modular and component-based methodology, emphasizing simplicity and efficiency in development. The entire EMS is conceptualized as a collection of self-contained and reusable modules, each responsible for specific functionalities in this approach.This modular design not only allows for a better understanding of the system's structure, but it also allows for independent module development, testing, and maintenance. Modules could be dedicated to user authentication, event creation, and attendance tracking, for example. This design approach adheres to the modularity and reusability principles, allowing the system to scale gracefully as new features are added or existing ones are improved.

Additionally, the design approach's component-based structure encourages the development of reusable front-end user interface components and back-end business logic encapsulation. This makes system development and maintenance easier while guaranteeing a consistent and cohesive user experience. For instance, user interface (UI) elements that can be easily integrated into different areas of the application are those that are developed as separate components, such as  the forms for user authentication. The codebase can be made more manageable by encapsulating business logic pertaining to event management or user interactions within modules. The design strategy seeks to enable the developer to effectively create and modify the EMS by striking a balance between flexibility and simplicity.

# Architectural Design

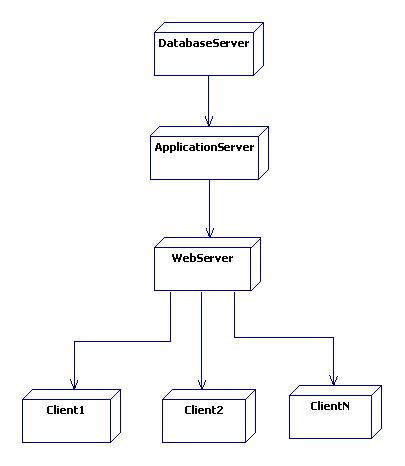
The architectural design section articulates the EMS's high-level structure, defining the major system components and their mutual dependence. It identifies critical modules, the responsibilities assigned to them, and the interactions that define their relationships. This architectural design, which serves as the system's blueprint, acts as a guide for developers, ensuring that the EMS's organization aligns seamlessly with the predefined objectives and design goals.

# Logical Design

The logical design aspect dives deeply into the EMS's intricacies, concentrating on the representation of its logic. This includes an in-depth description of the system's data flow, structures, and algorithms. This section provides a comprehensive understanding of the EMS's internal workings by addressing the logical relationships between different components. It is an important reference for developers, guiding them through the implementation of the system's logic. The logical design ensures that the EMS's data processing, storage, and retrieval are efficient and functional.

# Deployment Diagram

The deployment diagram represents the physical arrangement of EMS components visually. It depicts the system's distribution across different nodes or hardware, including the placement of servers, databases, and client devices. This visual aid improves understanding of the EMS's physical structure, guiding resource allocation and system performance optimization decisions. The deployment diagram depicts key components such as server locations, database placements, and the network of client devices, providing a clear and concise view of how the EMS is physically organized.



*Deployment diagram for a web-based event management system.*

# Process Design

Process design is the systematic planning and detailing of activities, tasks, and interactions within a system or business process. It entails creating a workflow, defining the sequence of actions, allocating resources, defining decision points, and designing how data is handled. The goal is to develop a structured and efficient set of processes that can be implemented within an organization or system to achieve specific goals. Process Design acts as a road map for developers, ensuring that activities are well-organized and optimized for effective execution.

## Use Case Diagram(DFD Level 0 or Context Diagram)

A **use case diagram** is a visual representation of the interactions between various actors (users or external systems) and a system.

An **actor**represents an external entity that interacts with the system. Actors are usually users or other systems who initiate or participate in one or more use cases.

* + 1. **actors:**

**Guest:** Represents users who are yet to register in the system.

**User:** Represents both students and university personnel.

**Organizer:** When users choose to create events, they become

organizers.

**Administrator:**Administrators are in charge of the EMS and have

the authority to monitor and manage the overall event

management process.They have access to and control over all

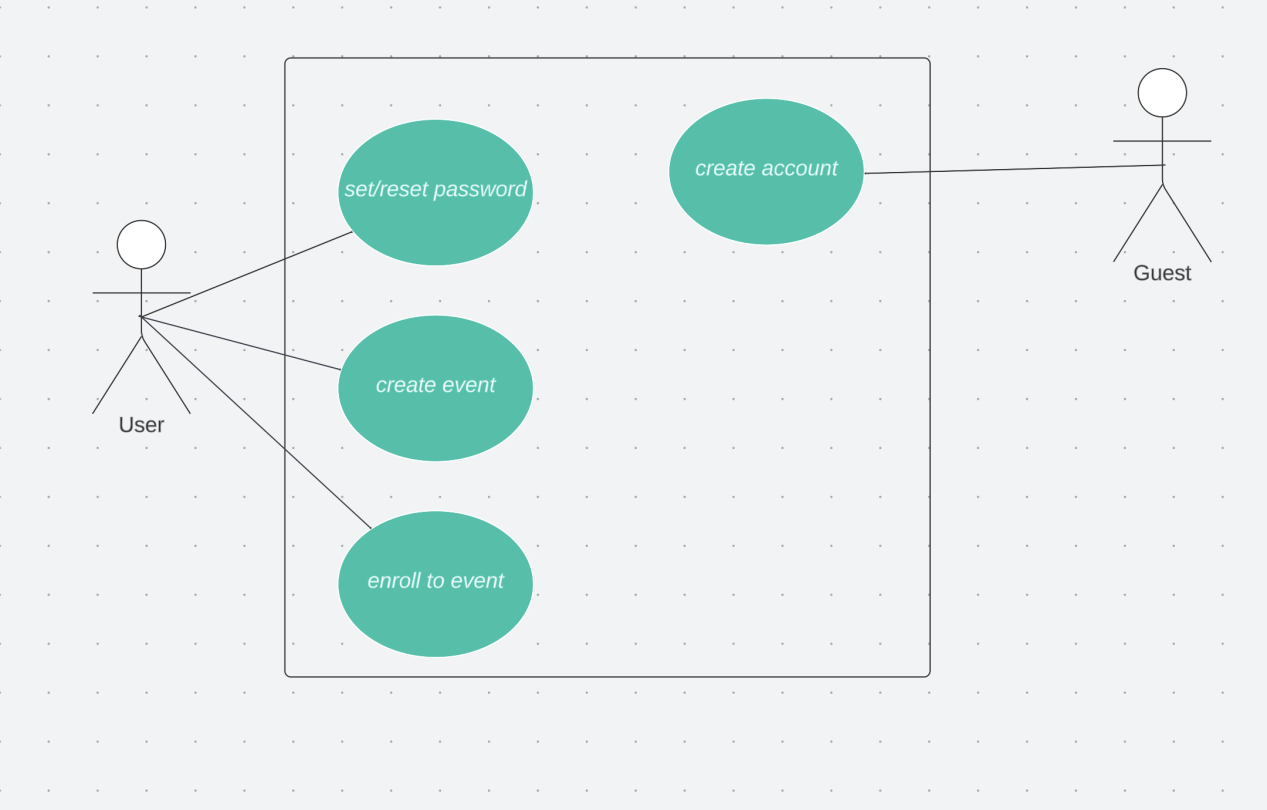
system events.They can also organize special events that involve

everyone in the institution.

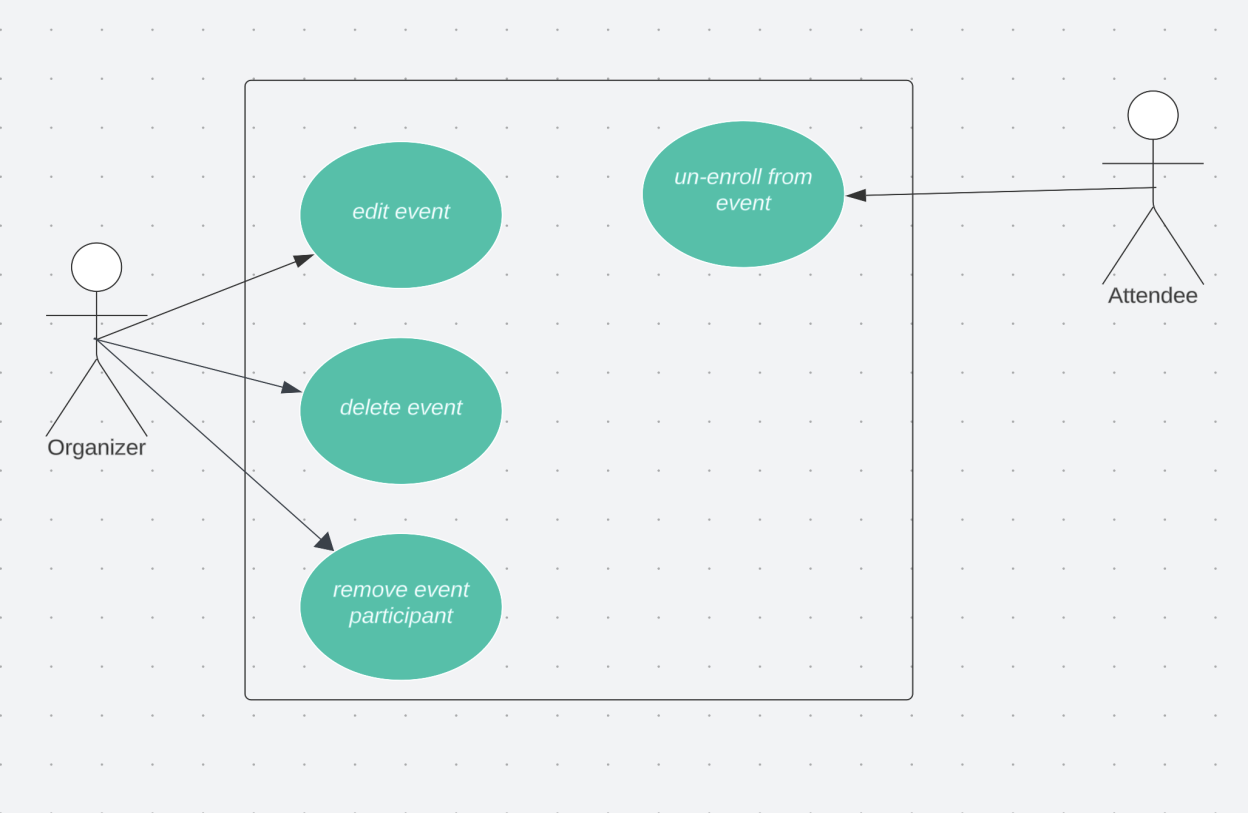
**Attendee:**When users, both students and university staff, register

for events, they become attendees.

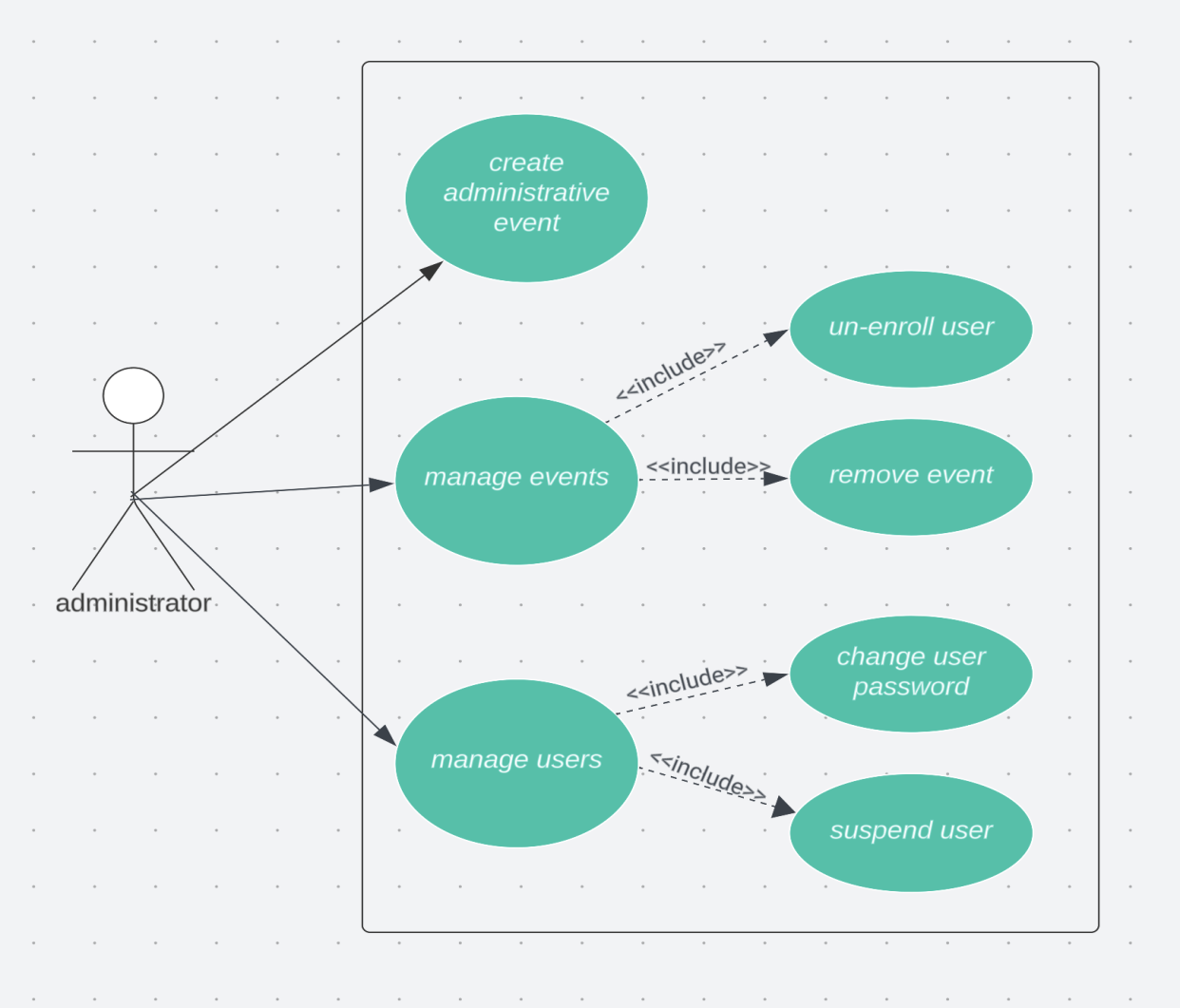
***use case diagram showing the roles for guest and user actors.***

******

***use case diagram showing the roles for organizer and attendee actors.***

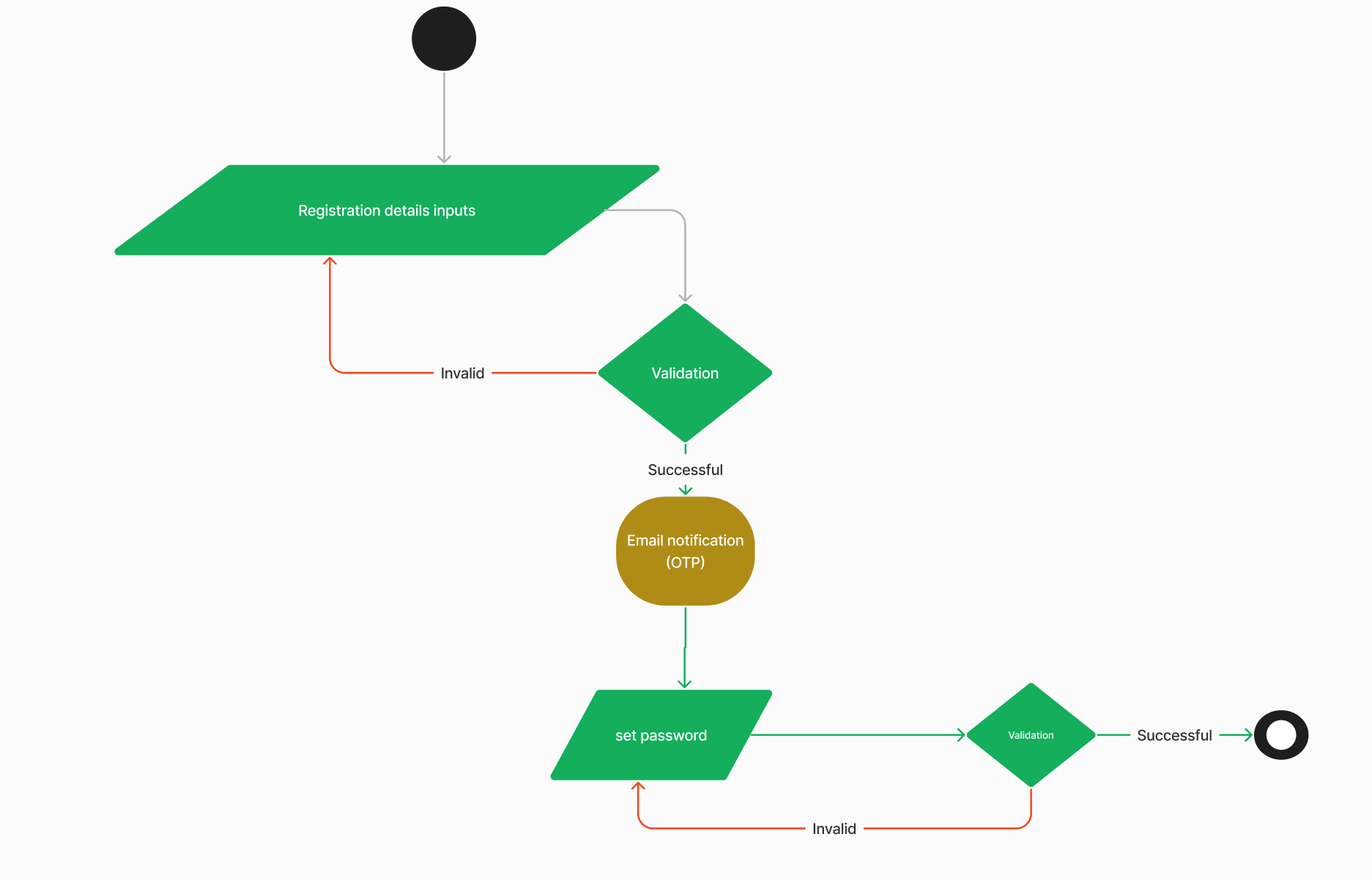
******

***use case diagram showing the roles for administrator.***

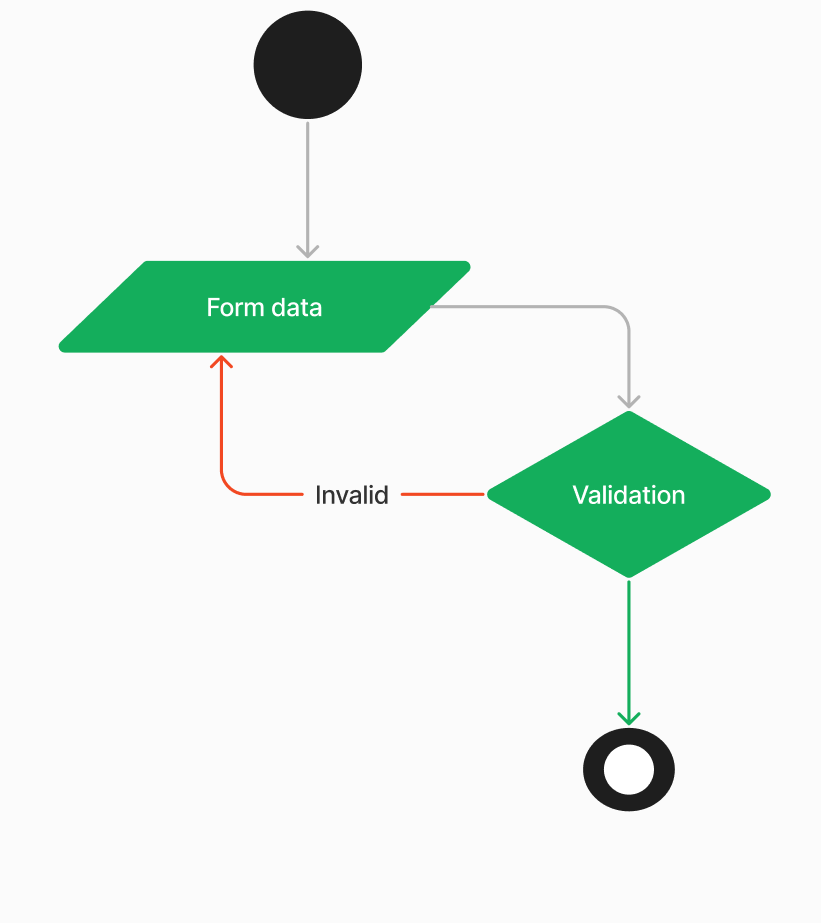
******

## Activity Diagram (Flow chart)

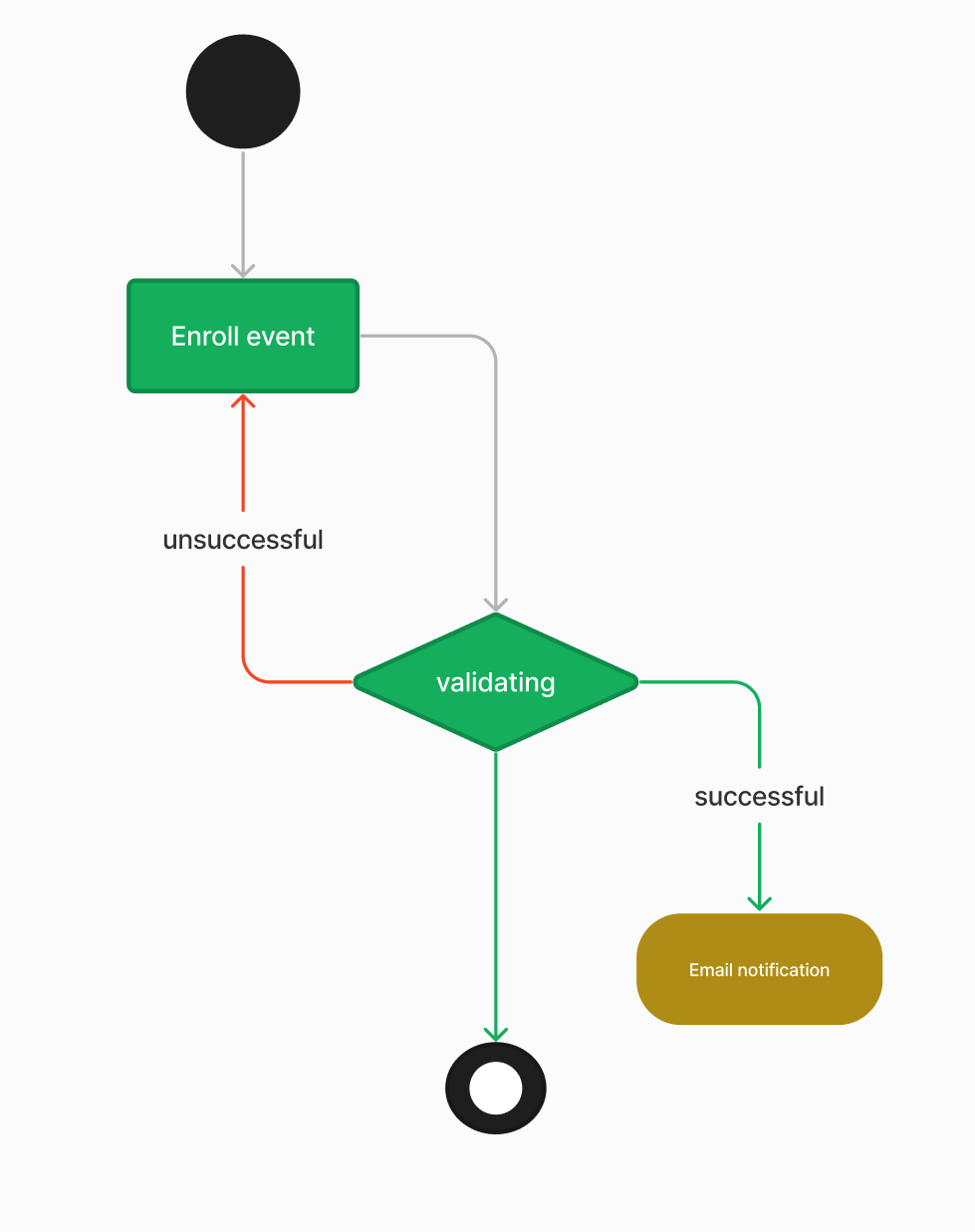
***Activity diagram for registration of new users***



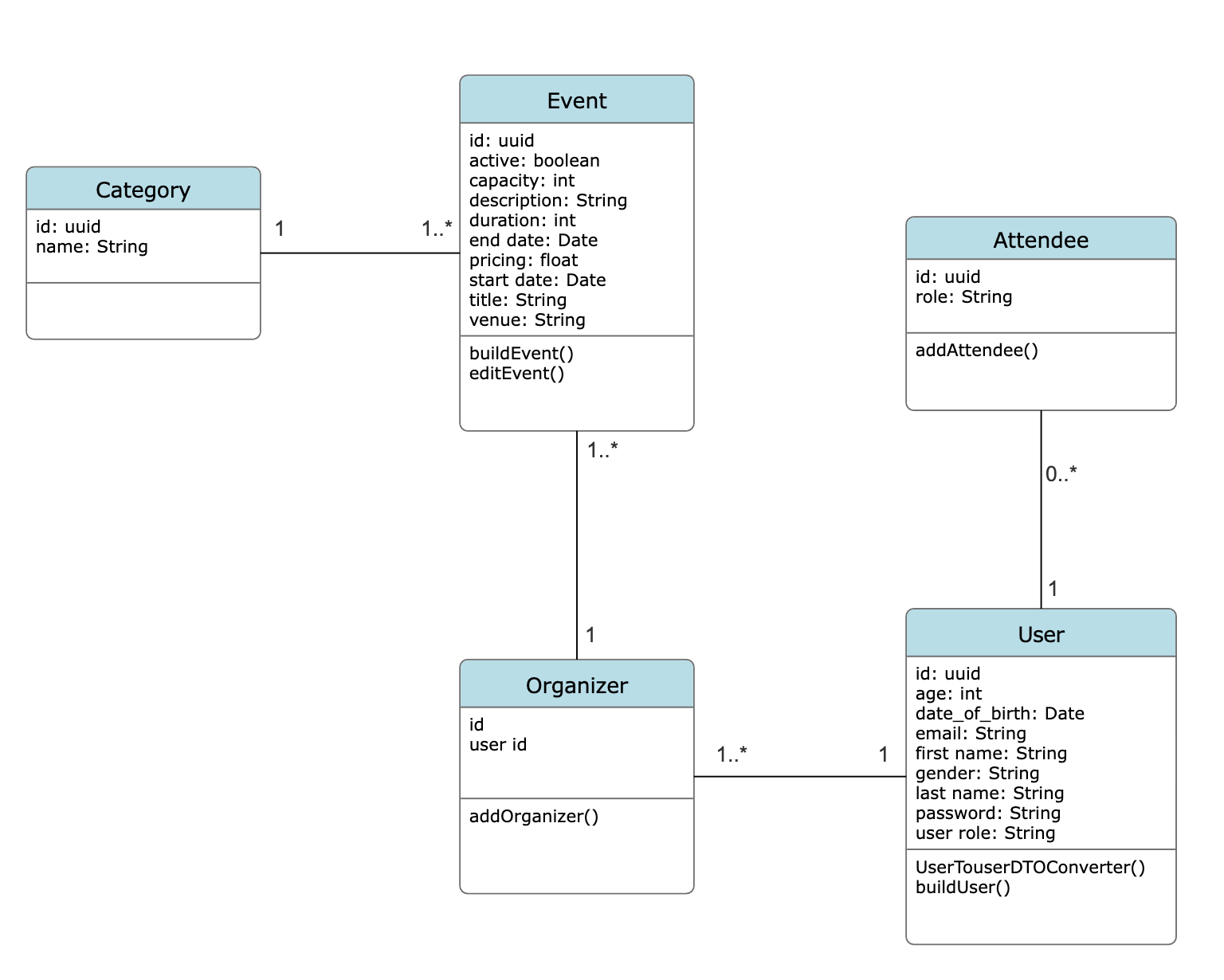
***Activity diagram for event creation***



***Activity diagram for enrolling to an event***



## Class Diagram



## Sequence Diagram

# Database Design

## Normalization (UNF, 1NF, 2NF, 3NF)

## Entity Relationship Diagram

The ERD graphically depicts the relationships between database entities.



## Table Schema

* + 1. User Table Schema

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| user\_id | INT | Primary Key, Auto-increment | Unique identifier for each user. |
| username | VARCHAR(50) | Unique, Not Null | User's username for authentication. |
| email | VARCHAR(100) | Unique, Not Null | User's email address. |
| password\_hash | VARCHAR(255) | Not Null | Hashed password for user security. |
| first\_name | VARCHAR(50) | Not Null | User's first name. |
| last\_name | VARCHAR(50) | Not Null | User's last name. |
| date\_of\_birth | DATE |  | User's date of birth. |
| gender | VARCHAR(10) |  | User's gender. |
| created\_at | TIMESTAMP | Default CURRENT\_TIMESTAMP | Timestamp when the user account was created. |

* + 1. Organizer Table Schema

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| organizer\_id | INT | Primary Key, Auto-increment | Unique identifier for each organizer. |
| user\_id | INT | Foreign Key to User | Reference to the associated user. |

* + 1. Attendee Table Schema

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| attendee\_id | INT | Primary Key, Auto-increment | Unique identifier for each attendee. |
| role | VARCHAR(20) | Not Null | Role of the attendee (organizer, participant, etc.). |
| user\_id | INT | Foreign Key to User | Reference to the associated user. |

* + 1. Category Table Schema

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| category\_id | INT | Primary Key, Auto-increment | Unique identifier for each category. |
| name | VARCHAR(50) | Not Null | Name of the category. |

* + 1. Event Table Schema

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| event\_id | INT | Primary Key, Auto-increment | Unique identifier for each event. |
| title | VARCHAR(255) | Not Null | Title or name of the event. |
| description | TEXT |  | Detailed description of the event. |
| start\_date | DATETIME | Not Null | Date and time when the event starts. |
| end\_date | DATETIME | Not Null | Date and time when the event ends. |
| venue | VARCHAR(255) |  | Location or venue of the event. |
| organizer\_id | INT | Foreign Key to Organizer | Reference to the organizer of the event. |
| category\_id | INT | Foreign Key to Category | Reference to the category of the event. |
| capacity | INT |  | Maximum capacity for the event. |
| age\_restriction | INT |  | Age restriction for the event. |
| active | BOOLEAN | Default True | Status indicating if the event is active. |
| created\_at | TIMESTAMP | Default CURRENT\_TIMESTAMP | Timestamp when the event record was created. |

## Data Dictionary

# Interface Design

## Internal machine interfaces.

## External system interfaces.

## Human interface

## User interface design

# Conclusion