-

A blue and black logo

Description automatically generated

**Event Management System**

A CSCI 313 project

Presented to:

DR. Mohamed ElGazar

Eng: Ahmed Jamal

**Name:** Marwan Ibrahim **ID:** 221001906

**Name:** Yousef Yehia **ID:** 221001899

**Name:** Omar Elhossiny **ID:** 221001028

**Name:** Mohamed Amir **ID:** 221001854

**Name:** Zeyad Shawky  **ID:** 221001447

**Table of Contents**

**Table of Contents**

1. **Introduction** ........................................................................................................... 1  
   1.1 Purpose ........................................................................................................... 1  
   1.2 Scope of Project ............................................................................................... 1  
   1.3 Glossary .......................................................................................................... 2  
   1.4 Overview of Document ...................................................................................... 2
2. **Overall Description** ............................................................................................... 4  
   2.1 Functional Requirements Specification .............................................................. 4  
   2.1.1 User Management ...................................................................................... 4  
   2.1.2 Event Management .................................................................................... 5  
   2.1.3 Registration Management .......................................................................... 5  
   2.1.4 Vendor Services ........................................................................................ 5  
   2.1.5 Feedback Mechanism ............................................................................... 6  
   2.1.6 Access Control ......................................................................................... 6  
   2.2 Non-Functional Requirements .......................................................................... 7  
   2.2.1 Performance ............................................................................................. 7  
   2.2.2 Scalability ................................................................................................ 7  
   2.2.3 Usability ................................................................................................... 7  
   2.2.4 Reliability ................................................................................................. 7  
   2.2.5 Security .................................................................................................... 8  
   2.2.6 Maintainability .......................................................................................... 8  
   2.2.7 Accessibility ............................................................................................. 8  
   2.2.8 Interoperability ......................................................................................... 8
3. **Diagrams** ........................................................................................................... 9  
   3.1 Use Case Diagram ......................................................................................... 9  
   3.2 Use Case Diagram Scenarios ......................................................................... 9  
   3.2.1 Register for Event ................................................................................... 9  
   3.2.2 Cancel Registration ................................................................................ 10  
   3.2.3 View Event Details ................................................................................. 10  
   3.2.4 Sign Up .................................................................................................. 10  
   3.2.5 Login ...................................................................................................... 11  
   3.2.6 Logout .................................................................................................... 11  
   3.2.7 Validate Email ......................................................................................... 11  
   3.2.8 Create Event .......................................................................................... 12  
   3.2.9 View Event ............................................................................................. 12  
   3.2.10 Update Event ....................................................................................... 12  
   3.2.11 Delete Event ........................................................................................ 13  
   3.2.12 Provide Feedback ................................................................................ 13  
   3.2.13 Offer Services ...................................................................................... 13  
   3.2.14 Receive Notification .............................................................................. 14  
   3.2.15 Contact Admin ..................................................................................... 14  
   3.3 Class Diagrams .............................................................................................. 15  
   3.3.1 Registration ............................................................................................ 15  
   3.3.2 User ....................................................................................................... 15  
   3.3.3 Event ...................................................................................................... 16  
   3.3.4 Service .................................................................................................... 16  
   3.3.5 Notification ............................................................................................. 16
4. **GitHub Repository Link …………………………………………………….23**

#### 1.0 Introduction

##### 1.1 Purpose

The Event Management System is a web-based platform designed to simplify the planning, organization, and participation in various events. The system serves as a centralized platform where admins can create and manage events, while users can browse, register for, and attend events. The goal is to enhance user convenience and streamline event management processes, eliminating inefficiencies and manual efforts.

##### 1.2 Scope of Project

Ticketawy is a web application developed to manage Event Management System designed to streamline the organization, management, and participation in events. The system serves as a platform where event organizers can create and manage events, vendors can offer their services, and attendees can register and provide feedback. By automating critical processes such as registrations, notifications, and vendor coordination, the system aims to enhance user experience, increase efficiency, and ensure seamless event execution.

##### 1.3 Glossary

| **Term** | **Definition** |
| --- | --- |
| **Event** | An organized occasion or gathering managed within the system. |
| **Organizer** | A person responsible for planning, coordinating, and overseeing an event. |
| **Participant** | An individual registered to attend an event. |
| **Database** | A repository storing all information tracked by the system. |
| **Vendor** | A supplier providing goods or services for the event. |
| **Booking** | The process of reserving a spot or service for an event. |
| **Dashboard** | The system interface used by organizers to manage events and data. |
| **Notification** | Automated updates sent to users regarding event-related activities. |
| **Role** | A designated function or responsibility assigned to staff or users. |
| **Software Requirements Specification (SRS)** | A document that outlines the system’s functions and constraints, such as this one. |
| **Stakeholder** | Anyone with a vested interest in the project who is not a developer. |
| **User** | Any individual interacting with the system, including organizers, participants, vendors, or staff. |

##### 1. Overview of Document

This document is organized into the following sections:

* **Overall Description (Section 2):** Provides a general overview of the system's functionality, informal requirements, and establishes context for technical specifications.
* **Requirements Specification (Section 3):** Focuses on the technical details of the system's functionality, primarily for developers.

These sections collectively provide a comprehensive description of the software, tailored to different audiences and employing distinct terminologies accordingly.

### 2.0 Overall Description

### 2.1 Functional Requirements

#### 2.1.1 User Management

* **User Registration and Authentication**
  + Users should be able to sign up with a unique email and password.
  + Users should be able to securely log in and log out of the system.
  + Users should have the ability to update their account details securely.
* **Admin Management**
  + Admins should have the ability to manage user permissions, including assigning and revoking roles.

#### 2.1.2 Event Management

* **Event Creation and Updates**
  + Admins should be able to create, edit, update, delete, and cancel events.
* **Event Association**
  + Events should be associated with specific vendors and services, allowing for comprehensive event planning.
* **Event Viewing**
  + Attendees should be able to view event details, including the event's date, location, and capacity.

#### 2.1.3 Registration Management

* **Event Registration**
  + Attendees should be able to register for events.
  + The system should allow attendees to cancel their registrations if necessary.
* **Capacity Tracking**
  + The system should monitor the number of registered attendees and enforce event capacity limits to prevent overbooking.

#### 2.1.4 Vendor Services

* **Vendor Offerings**
  + Vendors should be able to offer services related to events, enabling a seamless integration of vendor-provided offerings into the event management workflow.

#### 2.1.5 Feedback Mechanism

* **Event Feedback**
  + Attendees should have the ability to provide feedback for events to improve future experiences.

#### 2.1.6 Access Control

* **Role-Based Permissions**
  + Only admins should have permission to manage user roles and permissions, ensuring a secure and controlled access hierarchy.

### 2.2 Non-Functional Requirements

#### 2.2.1 Performance

* The system should handle up to 100 simultaneous users without performance degradation.
* Actions like event registration or login should be executed within 2 seconds.

#### 2.2.2 Scalability

* The system should be able to scale to support additional features, users, or events in the future.

#### 2.2.3 Usability

* The user interface should be simple and intuitive for all types of users (attendees, vendors, and admins).
* Provide clear error messages and guides for resolving issues.

#### 2.2.4 Reliability

* Ensure 99.9% uptime for the application.
* Backup data daily to prevent data loss.

#### 2.2.5 Security

* The website is very secure. The user data is stored in Firebase. The admin can’t even know a user’s password.
* Watermark will be added onto the videos as well.

#### 2.2.6 Maintainability

* Code should follow standard conventions and be well-documented for future developers.
* The system should allow easy updates and bug fixes without affecting current functionality.

#### 2.2.7 Accessibility

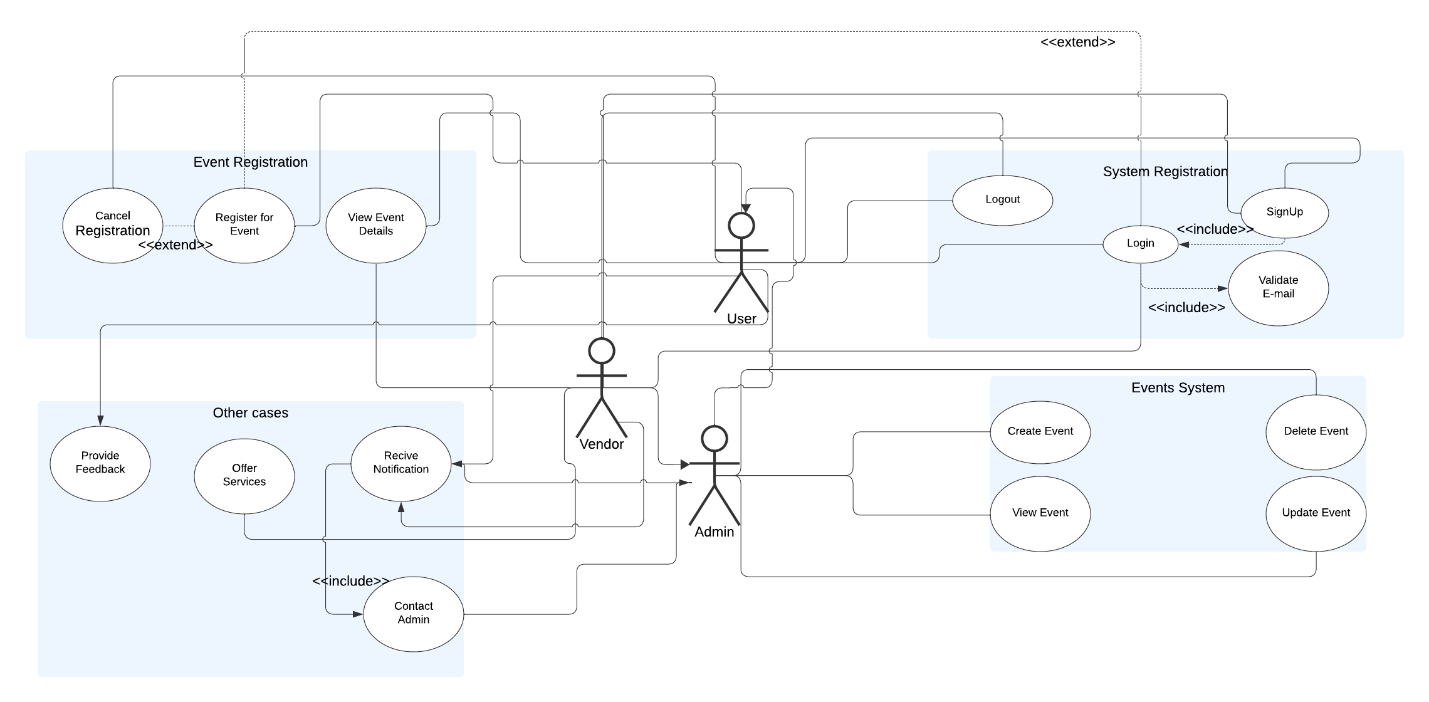
* Ensure the platform is accessible to users with disabilities (e.g., compatibility with screen readers).

#### 2.2.8 Interoperability

* The system should be designed to integrate with third-party services (e.g., payment gateways, email notifications).

### 3 Diagrams

#### 3.1 Use Case Diagram



#### 3.2 Use Case Diagram Scenarios

## 3.2.1 Register for Event

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User |
| **Main Success Scenario** | 1. User selects an event.  2. User clicks the "Register" button.  3. System confirms the registration and sends a notification. |
| **Exceptions** | - Event registration is closed.  - Event capacity is full.  - User is not logged in. |
| **Actions** | - Validate the user session.  - Update the registration database.- Send confirmation notification. |
| **Pre-Condition** | User must be logged in. |
| **Post-Condition** | User is registered for the event, and a confirmation notification is sent. |

## 3.2.2 Cancel Registration

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User |
| **Main Success Scenario** | 1. User navigates to "My Events."  2. User selects the event to cancel.  3. System confirms the cancellation and updates the event capacity. |
| **Exceptions** | - Cancellation deadline has passed.  - User is not logged in. |
| **Actions** | - Validate the user session.  - Update the registration database.  - Send cancellation notification. |
| **Pre-Condition** | User must be registered for the event. |
| **Post-Condition** | User is unregistered from the event, and a cancellation notification is sent. |

## 3.2.3 View Event Details

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User |
| **Main Success Scenario** | 1. User selects an event.  2. System displays the event details (e.g., date, time, location, description). |
| **Exceptions** | Event details are unavailable. |
| **Actions** | - Query the database for event details.  - Display the event details to the user. |
| **Pre-Condition** | None. |
| **Post-Condition** | Event details are displayed to the user. |

## 3.2.4 Sign Up

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User |
| **Main Success Scenario** | 1. User fills out the sign-up form.  2. System sends a validation email.  3. User clicks the validation link to complete the process. |
| **Exceptions** | - Email already exists.- Validation email is not delivered. |
| **Actions** | - Collect user information.  - Send validation email.  - Update user status upon validation. |
| **Pre-Condition** | User has not signed up previously. |
| **Post-Condition** | User account is created and validated. |

## 3.2.5 Login

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User |
| **Main Success Scenario** | 1. User enters login credentials.  2. System verifies the credentials and grants access. |
| **Exceptions** | - Incorrect credentials.  - Account is not validated. |
| **Actions** | - Validate user credentials.  - Establish a user session. |
| **Pre-Condition** | User must have a registered and validated account. |
| **Post-Condition** | User is logged in. |

## 3.2.6 Logout

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User |
| **Main Success Scenario** | 1. User clicks the "Logout" button.  2. System ends the user session. |
| **Exceptions** | None. |
| **Actions** | - End the user session.  - Redirect the user to the login page. |
| **Pre-Condition** | User must be logged in. |
| **Post-Condition** | User session is terminated. |

## 3.2.7 Validate Email

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | System |
| **Main Success Scenario** | 1. System sends a validation email.  2. User clicks the validation link.  3. System updates the account status to validated. |
| **Exceptions** | - Invalid validation link.  - Validation link expired. |
| **Actions** | - Generate and send a validation email.  - Validate the user upon link activation. |
| **Pre-Condition** | User must initiate the sign-up process. |
| **Post-Condition** | User account is validated. |

## 3.2.8 Create Event

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | Admin |
| **Main Success Scenario** | 1. Admin enters event details (e.g., name, date, location).  2. System saves the event.  3. System confirms the event creation. |
| **Exceptions** | - Missing required information  .- Duplicate event. |
| **Actions** | - Collect event details.  - Validate and store event information. |
| **Pre-Condition** | Admin must be logged in. |
| **Post-Condition** | Event is created and available for users to view. |

## 3.2.9 View Event

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | Admin |
| **Main Success Scenario** | 1. Admin selects an event.  2. System displays the event details. |
| **Exceptions** | Event details are unavailable. |
| **Actions** | - Query the database for event details.  - Display the event details to the admin. |
| **Pre-Condition** | Admin must be logged in. |
| **Post-Condition** | Event details are displayed to the admin. |

## 3.2.10 Update Event

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | Admin |
| **Main Success Scenario** | 1. Admin selects an event to update.  2. Admin modifies the event details.  3. System saves the updated details. |
| **Exceptions** | Invalid or incomplete update information. |
| **Actions** | - Validate the updated event information.  - Save the updated details. |
| **Pre-Condition** | Event must exist. |
| **Post-Condition** | Event details are updated in the system. |

## 3.2.11 Delete Event

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | Admin |
| **Main Success Scenario** | 1. Admin selects an event to delete.  2. System removes the event and notifies affected users. |
| **Exceptions** | - Event does not exist.  - Deletion impacts active registrations. |
| **Actions** | - Validate the event for deletion.  - Notify registered users about the deletion. |
| **Pre-Condition** | Admin must be logged in. |
| **Post-Condition** | Event is removed from the system. |

## 3.2.12 Provide Feedback

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User |
| **Main Success Scenario** | 1. User navigates to the feedback section.  2. User submits their feedback.  3. System acknowledges the feedback submission. |
| **Exceptions** | Feedback submission fails due to technical issues. |
| **Actions** | - Collect feedback from the user.  - Store the feedback in the system. |
| **Pre-Condition** | User must have access to the system. |
| **Post-Condition** | Feedback is recorded. |

## 3.2.13 Offer Services

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | Vendor |
| **Main Success Scenario** | 1. Vendor navigates to the service offering section.  2. Vendor submits details of their service.  3. System confirms the submission. |
| **Exceptions** | Service offer submission fails due to missing information. |
| **Actions** | - Collect service details from the vendor.  - Validate and store the service offer. |
| **Pre-Condition** | Vendor must be logged in. |
| **Post-Condition** | Service offer is recorded in the system. |

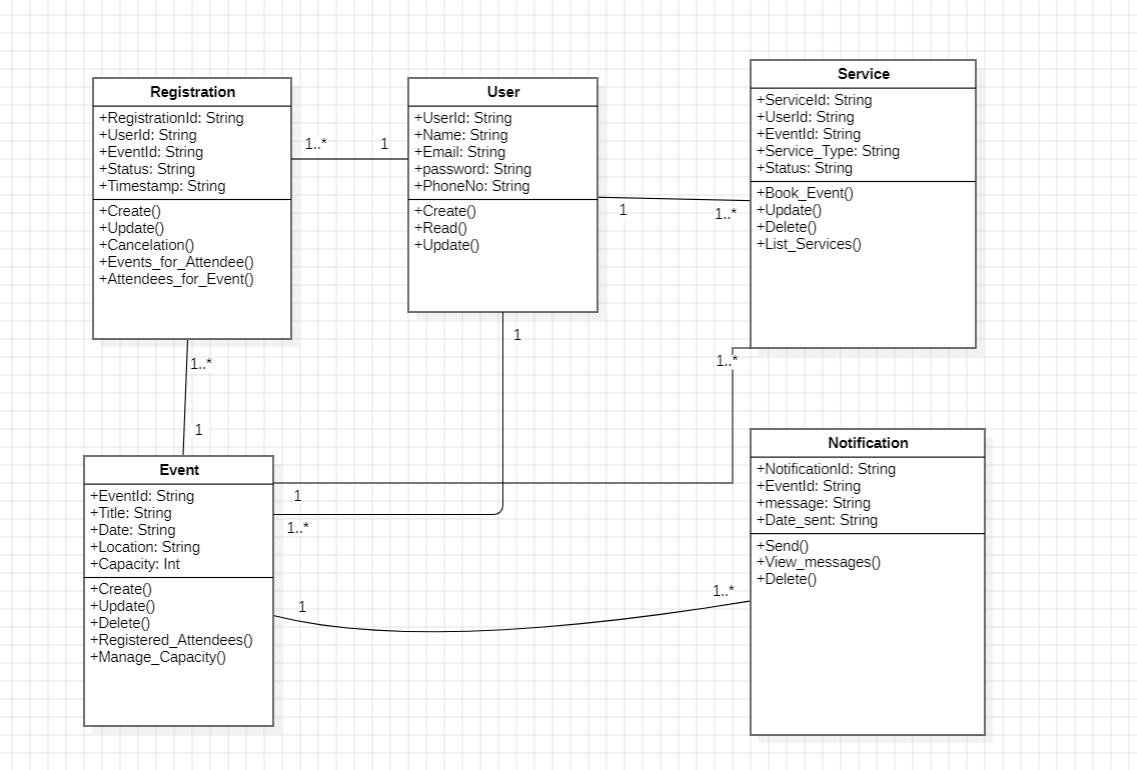
## 3.2.14 Receive Notification

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User, Vendor |
| **Main Success Scenario** | 1. System generates notifications based on triggers (e.g., event updates).  2. System sends notifications to users and vendors. |
| **Exceptions** | Notification delivery fails due to technical issues. |
| **Actions** | - Generate notifications based on system triggers.  - Deliver notifications via appropriate channels. |
| **Pre-Condition** | User or vendor must be registered. |
| **Post-Condition** | Notification is delivered successfully. |

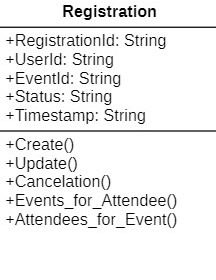
## 3.2.15 Contact Admin

| **Attribute** | **Details** |
| --- | --- |
| **Actors** | User, Vendor |
| **Main Success Scenario** | 1. User or vendor navigates to the contact admin section.  2. User or vendor submits their query or issue.  3. System sends the query to the admin. |
| **Exceptions** | Contact submission fails due to technical issues. |
| **Actions** | - Collect the query or issue details.  - Notify the admin of the submission. |
| **Pre-Condition** | User or vendor must have access to the system. |
| **Post-Condition** | Query or issue is submitted to the admin for resolution. |

**3.3 class diagram**

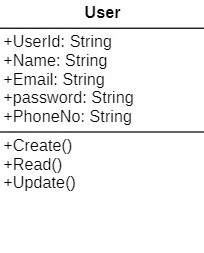


**3.3.1 Registration**

****

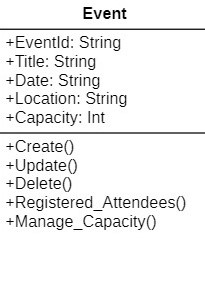
* **Purpose: Manages the process of registering users for events.**
* **Responsibilities:**
  + **Stores information about the registration, including the user and event details, the registration status, and timestamp.**
  + **Provides methods to:**
    - **Create new registrations.**
    - **Update existing registration details.**
    - **Cancel registrations.**
    - **Retrieve events for specific attendees.**
    - **List attendees for a specific event.**

**3.3.2 User**

****

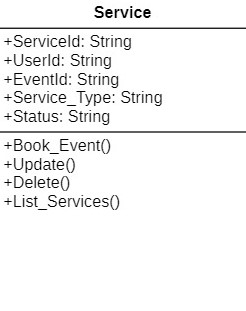
* **Purpose: Manages information about the system's users.**
* **Responsibilities:**
  + **Stores user details such as name, email, and phone number.**
  + **Provides methods to:**
    - **Create new users.**
    - **Read user information.**
    - **Update user details.**

**3.3.3 Event**

****

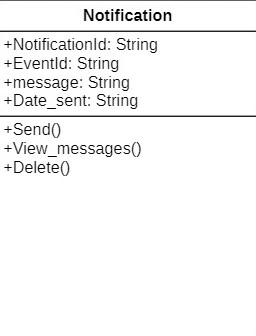
* **Purpose: Represents events that users can register for.**
* **Responsibilities:**
  + **Stores event details such as title, date, location, and capacity.**
  + **Provides methods to:**
    - **Create new events.**
    - **Update existing events.**
    - **Delete events from the system.**
    - **List registered attendees for the event.**
    - **Manage the event's capacity to ensure it is not overbooked.**

**3.3.4 Service**

****

* **Purpose: Handles additional services associated with events or users.**
* **Responsibilities:**
  + **Links specific services to users and events.**
  + **Maintains the type and status of the service.**
  + **Provides methods to:**
    - **Book services for an event or user.**
    - **Update service details.**
    - **Delete services.**
    - **List all available services.**

**3.3.5 Notification**

****

* **Purpose: Sends and manages notifications related to events and services.**
* **Responsibilities:**
  + **Stores notification details, including the associated event, message content, and the date sent.**
  + **Provides methods to:**
    - **Send notifications to users.**
    - **View messages sent to users.**
    - **Delete old or unnecessary notifications.**

**GitHub Repository Link:**

<https://github.com/omar-28-2/Event-Management-System>