Introduction

Domain analysis is an important step in the development of a programming club website. It involves identifying the key components and requirements of the website, as well as the relationships between these components. The domain of a programming club website is usually in IT educational institutions. The motivation behind developing a programming club website is to provide an online platform for members to promote the club and its activities to a wider audience. A programming club website can help to build a community of like-minded individuals, provide resources and opportunities for members to learn and grow, and improve the club's visibility and impact.

General Information About the Domain

Users generally need an online platform related to the programming club in order to perform activities like search event, event registration, event de-registration, view standings list, etc. While the club members usually have to perform activities, like event scheduling, attendance tracking, update standings list, etc. The programming club website contributes to these scenarios by increasing efficiency, reducing errors, and improving the overall user experience. It also allows members and users to efficiently manage their activities. Additionally, the system can help enhance security and privacy by securely storing user information and processing payments in a safe and reliable manner.

Users

The end-user involved in system interaction is generally interacting with the frontend and database, responding to backend calls. The end-user requires no training to use the system for the first time. When API calls to the backend are made, the backend system will make the appropriate database calls.

Existing Software

Some common features of the programming club website include:

- Event management: This feature allows club members and users to manage the activities related to events, like, event scheduling, event registration, event de-registration, etc.

- Standings/rank list management: This feature allows club members to manage the standings/rank list of the participants for the events.

- Attendance tracking management: This feature allows club members to manage the attendance of the users for the events.
- User profile management: This feature allows users to manage their profiles, like updating their profile details,

- Feedback management: This feature allows club members to manage the feedback provided by the users about various measures like performance, usability, reliability, security, and efficiency.

Identify boundary, entity, control object

Entity Objects

Entity objects in the context of a programming club website refer to the different entities or objects that are present within the website's domain. In the context of a programming club website, some examples of entity objects might include:

- User: An object representing an Individual who registers on the website to access its features and services.
- Member An object representing an Individual who joins the programming club and has access to the club's resources and events.
- Event: An object representing a programming club-related activity organized by the club, such as a hackathon, coding competition, or session.
- Resources: Instructional materials like coding tutorials, coding challenges, etc.
- Feedback: An object representing a form that users fill out to review the functionalities of the website.
- News and announcements - Updates about the club's activities and upcoming events.
- Code Repository: An object representing a collection of code that members can contribute to and access.
- APIs: Tools that allow users to integrate the website's functionality into their own applications.
- Search engines: Tools that allow users to search for specific code or programming resources on the website.

Entity objects on a programming club website are important because they provide a way to organize and represent the various elements of the website's domain and allow developers and users to interact with these elements in a structured and consistent manner. This makes it easier to manage and maintain the website's functionality and provides a better user experience for those who use the website.

Boundary Objects

The concept of "boundary objects" refers to objects that are used in different contexts and are able to bridge the gap between different communities or domains. In the context of a programming club website, some examples of boundary objects might include:

- Login Page: A boundary object that allows members to log in to the website using their username and password.
- Registration Form: A boundary object that allows new members to sign up for the programming club by providing their name, email, username, password, programming languages known, and areas of interest.
- Event Calendar: A boundary object that displays upcoming programming-related events organized by the club, including the event name, date, time, location, and registration information.
- Resources Page: A boundary object that provides access to helpful programming resources such as coding tutorials, coding challenges, etc.
- Code Repository Page: A boundary object that displays code repositories that members can contribute to and access.
- Bug tracking systems: These are tools that help developers identify and track software bugs or issues, and can be used by both developers and users to report and resolve issues.

These objects serve as intermediaries between the different groups of people involved in programming, such as developers, users, and designers. They facilitate communication and collaboration across different domains and help ensure that everyone involved in the programming process is on the same page.

Control Objects

In the context of a programming club website, control objects can refer to various features and tools that are used to control or manipulate the behavior of the website, its content, or its users. Some examples of control objects of a programming club website are as follows:

- User authentication and authorization system: A control object that verifies the identity of users who log in to the website and controls their access to different parts of the website based on their authorization level, such as event scheduling, event registration, attendance tracking, volunteer hours tracking, etc.
- Event Management System: A control object that allows club members to create, modify, and delete events, as well as manage event registration and attendance.
- Database Management: A control object that stores and manages user account information, event details, and other club-related information.

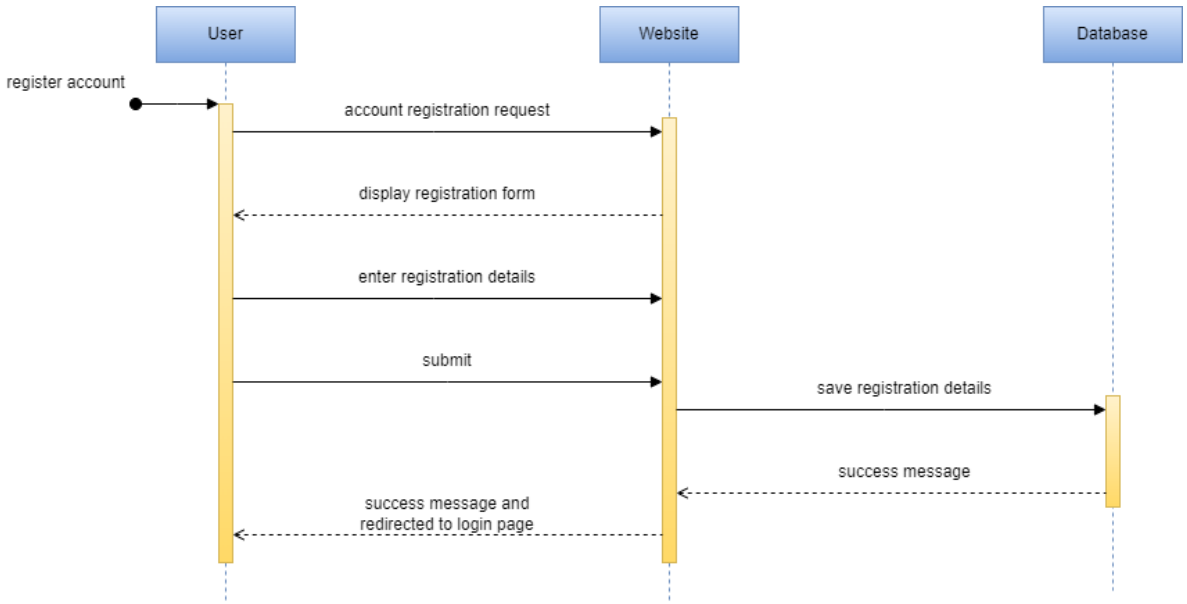
- Content management system: A control object that allows website administrators to control the creation, organization, and publication of content on the website, such as tutorials, and documentation.
- Version control system: A control object that allows developers to control and manage changes to the website's source code, and track different versions of the website over time.
- Testing and debugging system: A control object that helps developers test and debug the website's code, identify and fix errors, and ensure that the website behaves as expected.
- Analytics and monitoring system: A control object that allows club officers to track website usage and performance, including monitoring traffic, user behavior, and engagement levels, and identifying areas for improvement.

Overall, control objects on a programming club website are designed to help website administrators and developers manage and control the website's behavior, content, and users efficiently and effectively.

Sequence Diagrams

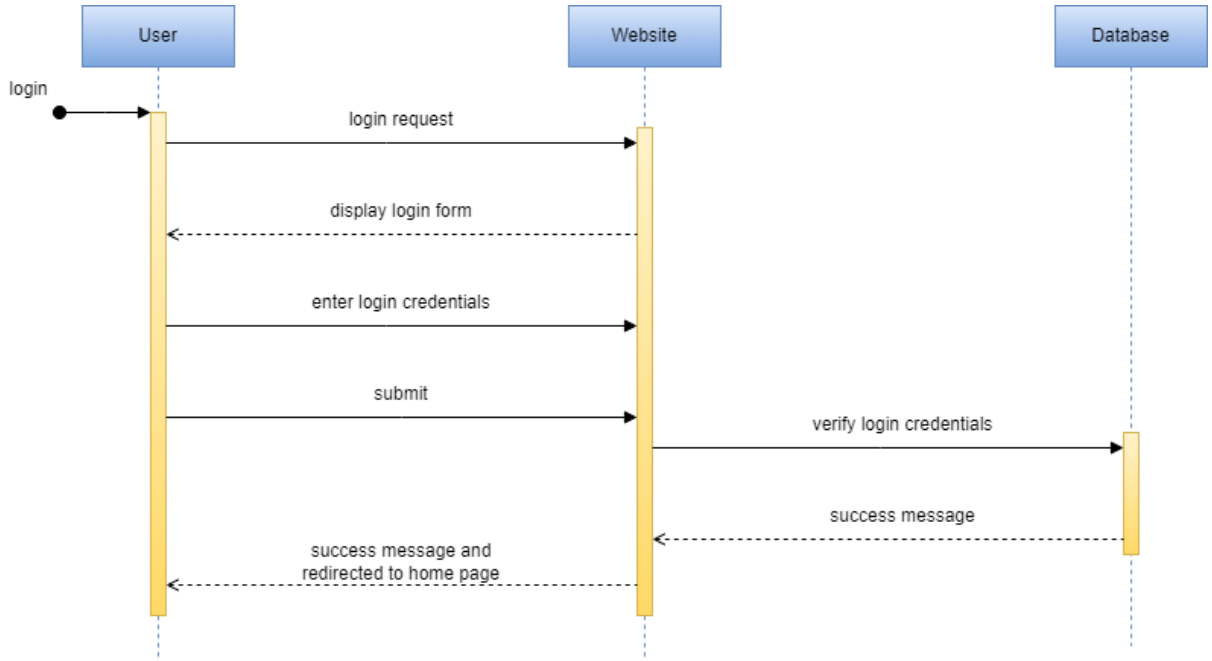
Registration

In this sequence diagram, the user initiates the registration process by requesting to register a new account on the website. The website then displays a registration form, which the user fills out with their registration details, such as their username, email, and password. Once the user submits the registration form, the website saves the user's registration details to the database. If the registration is successful, the database returns a success message to the website, which then displays a success message to the user and redirects them to the login page.



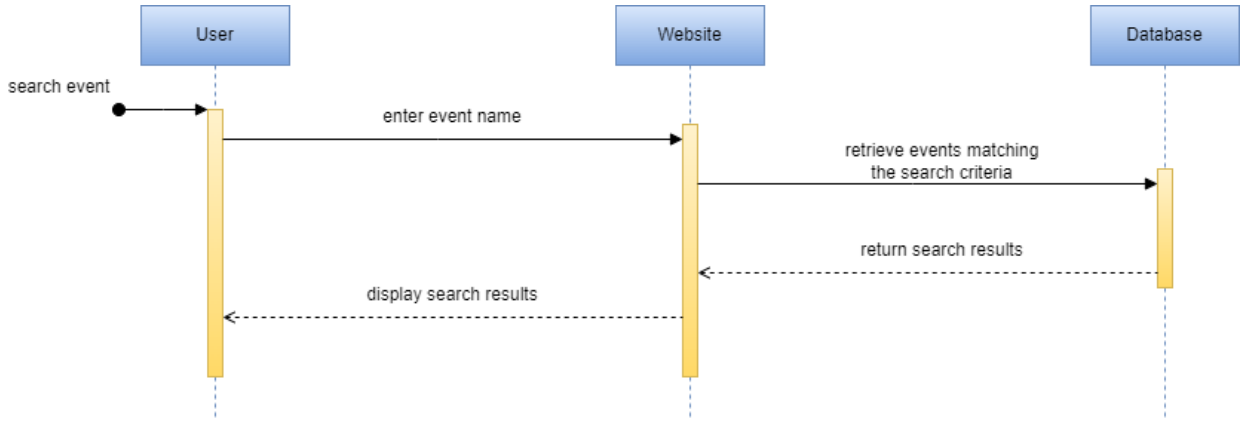
Login

In this sequence diagram, the user initiates the login process by requesting to log in to the website. The website then displays a login form, which the user fills out with their login credentials, such as their username and password. Once the user submits the login form, the website verifies the user's login credentials by querying the database. If the login credentials are valid, the database returns an authentication token to the website. The website then sets the authentication token in the user's session and redirects them to the dashboard.



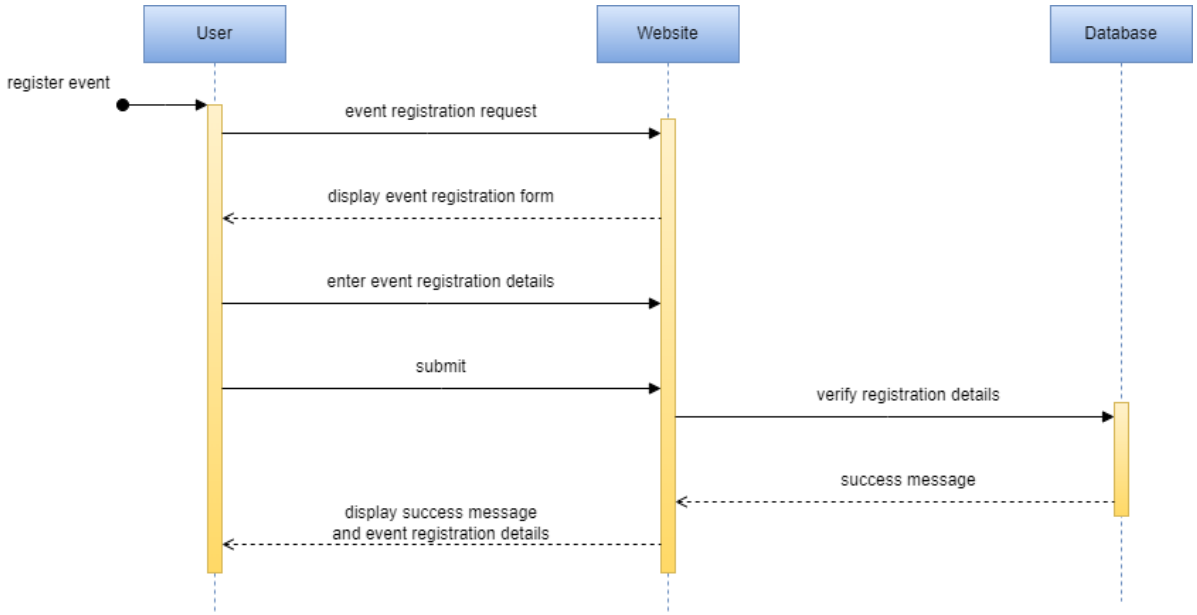
Event Search

In this sequence diagram, the user initiates the event search process by entering the name of the event to be searched. The website searches for events matching the search criteria by querying the database. The database returns a set of search results to the website, which then displays the search results to the user.



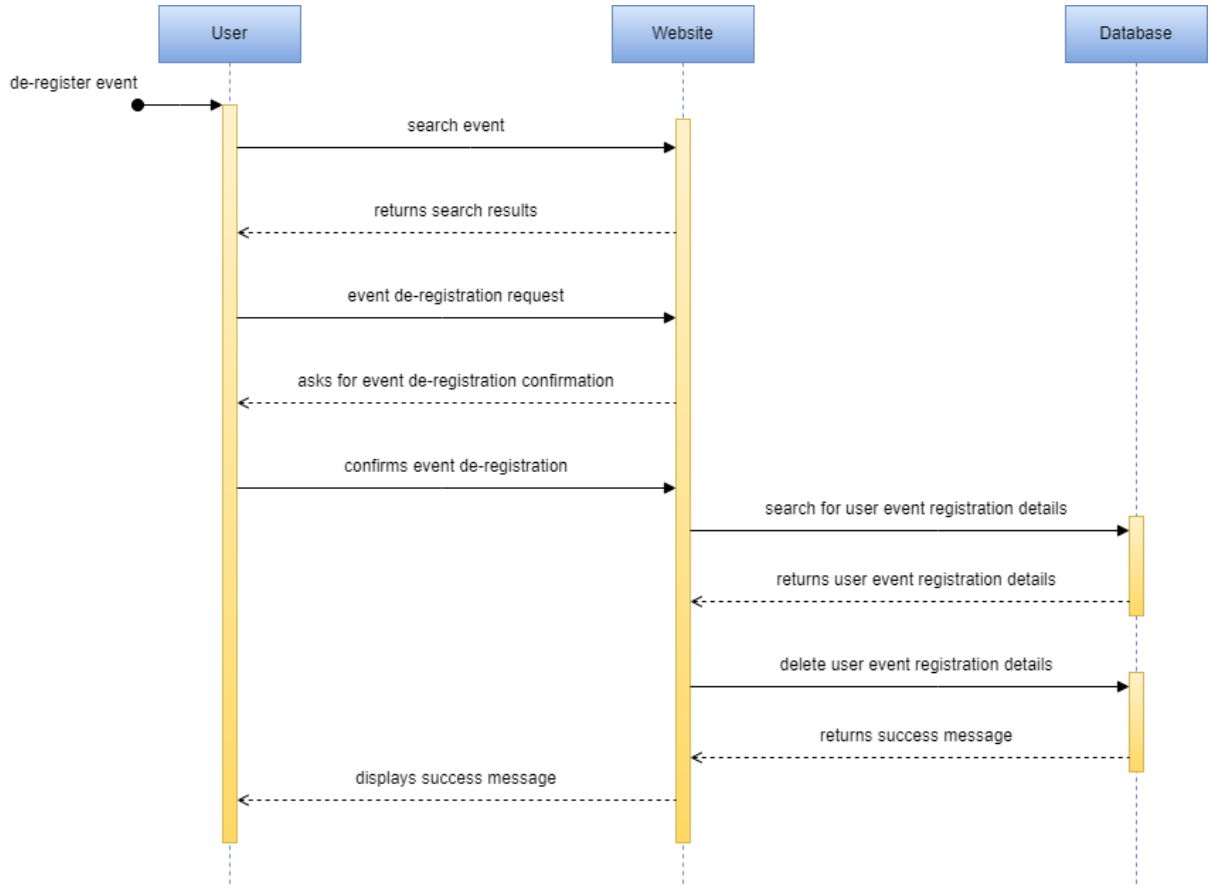
Event Registration

In this sequence diagram, the user initiates the event registration process by requesting to register for an event on the website. The website then displays an event registration form, which the user fills out with their event registration details, such as their name, email, and event selection. Once the user submits the event registration form, the website processes the payment for the event registration using a payment gateway. The payment gateway returns a payment status to the website, indicating whether the payment was successful or not. If the payment is successful, the website saves the user's event registration details to the database. The database returns a success message to the website, which then displays a success message and the user's event registration details to the user.



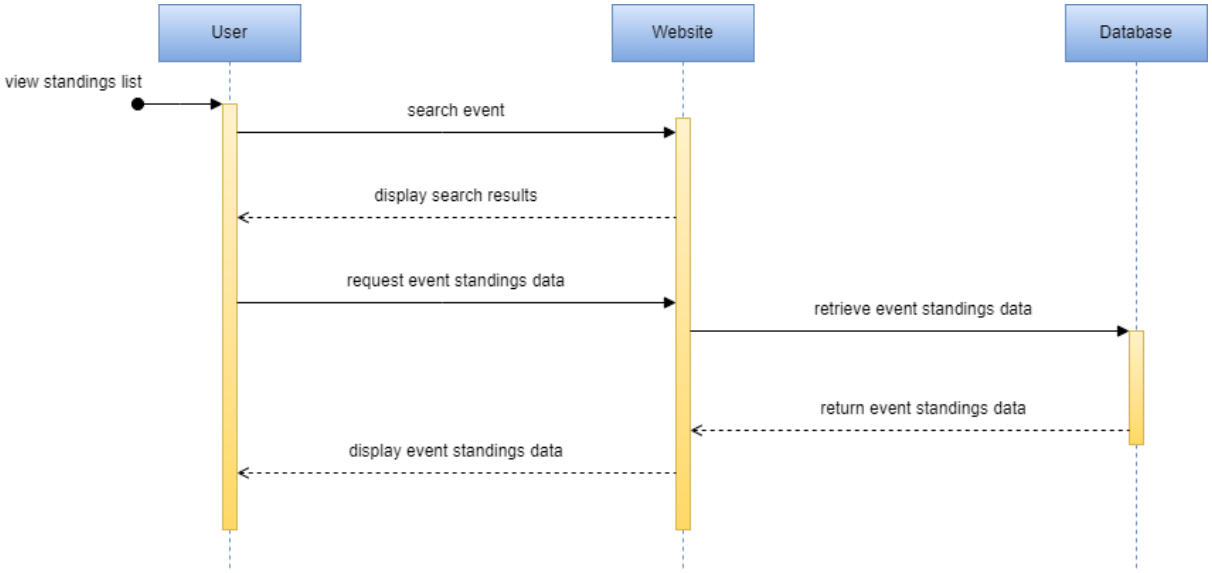
Event De-registration

In this sequence diagram, the user initiates the event de-registration process by requesting to de-register from an event on the website. The website then displays a de-registration confirmation form, which the user fills out to confirm their de-registration. Once the user submits the de-registration confirmation form, the website searches for the user's event registration details in the database. If the registration details are found, the website deletes the user's event registration details from the database. The database returns a success message to the website, which then displays a success message to the user.



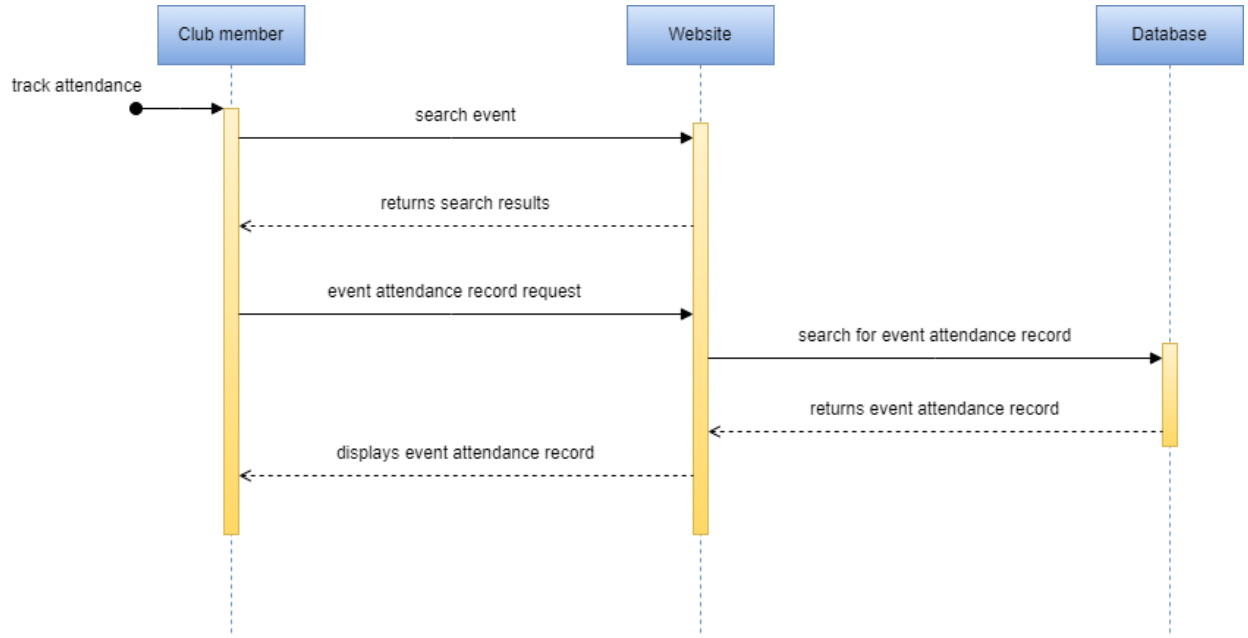
View Standings List

In this diagram, the user starts by searching for the event on the website. The user then sends a request to the website for the event standings data. The website retrieves the data from the database and displays the list to the user.



Attendance Tracking

In this sequence diagram, the club member logs in to the website and selects the event they want to track attendance for. The website then queries the database for the requested event's attendance record and the database returns the requested event's attendance record to the website, which in turn displays it to the club member.



Classes - Attributes and Operations

1) Role

- **Attributes:**
  - role_id: a unique identifier for the user role
  - role_title: the title of the role
  - role_description: the description of the role

- **Operations:**
  - createRole(): allows an authorized user to create a new role
  - editRole(): allows an authorized user to edit an existing role
  - deleteRole(): allows an authorized user to delete an existing role
  - assignRole(): allows an authorized user to assign a role to another user

2) User

- **Attributes:**
    - user_id: a unique identifier for the user
    - user_name: the user's name
    - user_role: the role of the user (e.g. student, club member, faculty, admin)
    - user_email: the email address of the user
    - user_password: the password of the user

- **Operations:**
    - createUser(): allows an authorized user to create a new user account
    - editUser(): allows an authorized user to edit an existing user account
    - deleteUser(): allows an authorized user to delete an existing user account
    - viewProfile(): allows the user to view their own profile information
    - editProfile(): allows the user to edit their profile information
    - viewEvents(): allows the user to view upcoming events
    - registerEvent(): allows the user to register for an upcoming event
    - de-registerEvent(): allows the user to de-register from an upcoming event
    - viewAttendance(): allows the user to view their attendance records
    - viewStandingsList(): allows the user to view the standings list of an event

3) Event

- **Attributes:**
    - event_id: a unique identifier for the event
    - event_name: the name of the event
    - event_description: a description of the event
    - event_date: the date of the event
    - event_time: the time of the event
    - event_location: the location of the event

- **Operations:**
    - createEvent(): allows an authorized user to create a new event
    - searchEvent(): allows the user to search for an event
    - editEvent(): allows an authorized user to edit an existing event
    - deleteEvent(): allows an authorized user to delete an existing event
    - registerEvent(): allows the user to register for an upcoming event
    - de-registerEvent(): allows the user to de-register from an upcoming event
    - viewAttendance(): allows an authorized user to view attendance records for an event
    - viewStandingsList(): allows the user to view the standings list for a particular event

4) Attendance

- **Attributes:**
    - event_id: the event for which attendance is being recorded
    - user_id: the user whose attendance is being recorded
    - event_date: the date of the event

- **Operations:**
    - markAttendance(): marks the attendance of a user at an event and sets the timestamp
    - getAllAttendances(): retrieves all attendance records for a particular event
    - getAttendanceCount(): retrieves the number of users who attended a particular event
    - getAttendancePercentage(): calculates and retrieves the percentage of users who attended a particular event

5) Resources

- **Attributes:**
    - resource_id: a unique identifier for the resource
    - resource_name: the name of the resource
    - resource_description: a description of the resource
    - resource_url: the URL where the resource can be accessed
    - resource_type: the type of the resource (e.g. article, video, book)
    - resource_tags: a list of tags associated with the resource (e.g. topic)

- **Operations:**
    - uploadResource(): allows an authorized user to upload a new resource
    - editResource(): allows an authorized user to edit an existing resource
    - deleteResource(): allows an authorized user to delete an existing resource
    - viewResource(): allows the user to view a specific resource

6) Standings

- **Attributes:**
    - event_id: a unique identifier for the standings list
    - event_name: the name of the standings list (e.g. competition name)
    - event_description: a description of the standings list
    - participants: a list of participants in the competition
    - scores: a list of scores for each participant
    - rank: the rank of each participant based on their score

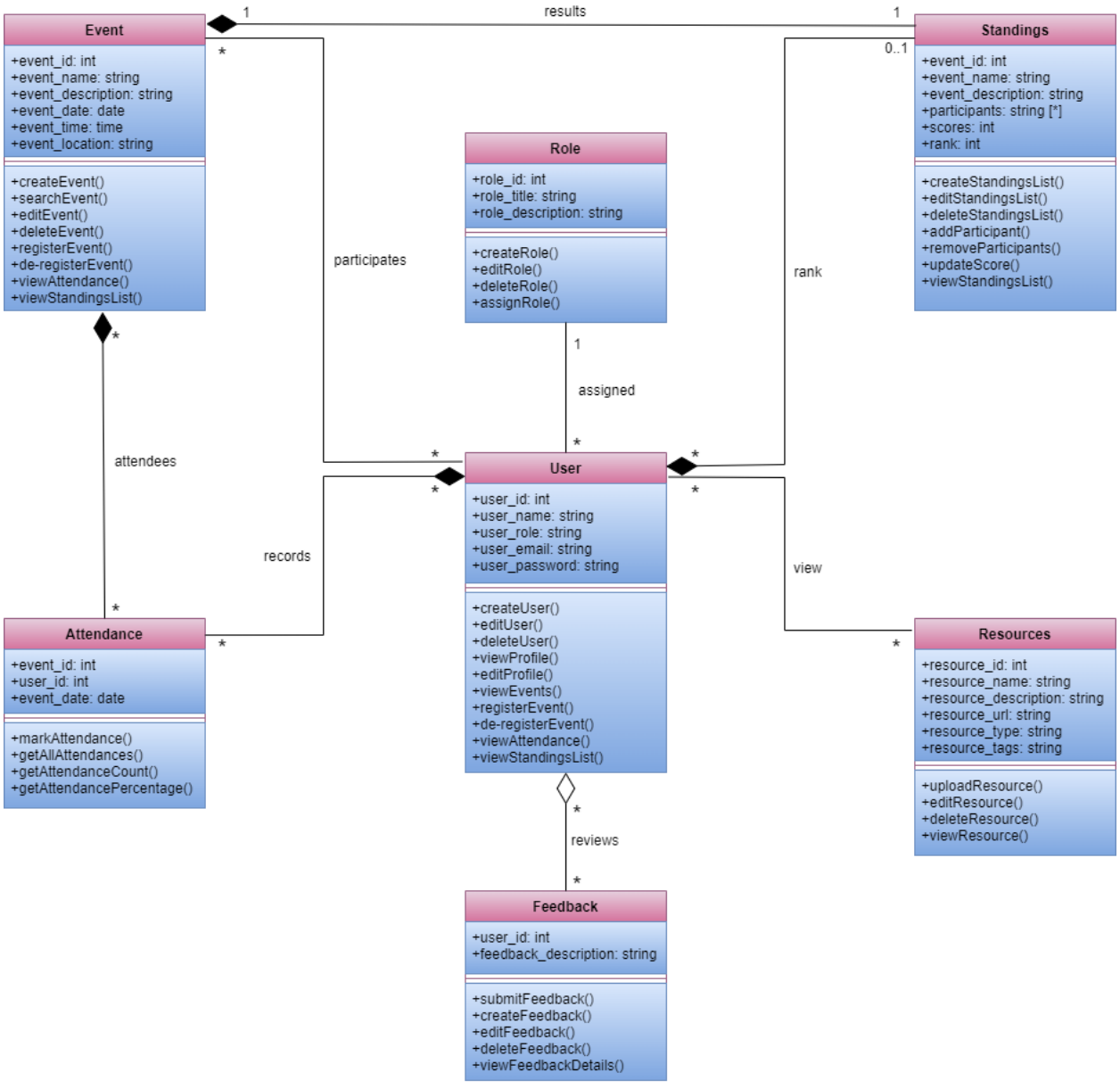
- **Operations:**
    - createStandingsList(): allows an authorized user to create a new standings list
    - editStandingsList(): allows an authorized user to edit an existing standings list
    - deleteStandingsList(): allows an authorized user to delete an existing standings list
    - addParticipant(): allows an authorized user to add a participant to a standings list
    - removeParticipant(): allows an authorized user to remove a participant from a standings list
    - updateScore(): allows an authorized user to update the score of a participant in a standings list
    - viewStandingsList(): allows the user to view the standings list

7) Feedback

- **Attributes:**
    - user_id: the unique identifier of the user
    - feedaback_description: the description of the feedback provided by the user

- **Operations:**
    - submitFeedback(): allows the user to submit the feedback form
    - createFeedback(): allows an authorized user to create a new feedback form
    - editFeedback(): allows an authorized user to edit existing feedback form
    - deleteFeedback(): allows an authorized user to delete existing feedback form
    - viewFeedbackDetails(): allows an authorized user to view the details of a specific feedback

Class Diagram



Design Goals

- **Easy Navigation:** The website should be designed in such a way that users can easily navigate and find what they are looking for, such as club events, membership information, meeting schedules, and resources.

- **Mobile Responsiveness:** The website should be optimized for different screen sizes and devices, including desktops, laptops, tablets, and smartphones.

- **Attractive and Engaging Design:** The website design should be visually appealing and engaging, with a consistent layout, typography, and color scheme. The use of graphics, images, and multimedia can help to make the website more attractive and interactive.
- **Performance:** The system should be quick and responsive, with minimal lag times for a better user experience. The system should be able to handle high volumes of concurrent users and transactions.

- **Scalability:** The system should be designed to accommodate growth and be easily scalable to meet increasing demands. The system should be good enough to handle a vast database of users and property details.

- **Availability:** The system should have a high availability rate and be designed to minimize downtime. The system should have robust disaster recovery mechanisms in place.

- **Security:** The system should have strong security measures to protect sensitive information, such as user passwords and payment information. The system should comply with relevant data protection regulations.

- **Usability:** The system should be user-friendly and intuitive, with a clear and simple navigation structure. The system should be accessible and usable for users with disabilities.

- **Reliability:** The system should have a high level of reliability, with minimal errors and bugs.

- **Maintainability:** The system should be designed to be easily maintainable, with clear documentation and an organized code structure. The system should have the capability to be updated and improved over time.

High-level System Design

We will be using a 3-layer architecture model that consists of a presentation tier, a business logic (application) tier, and a data tier. The presentation tier is a graphical user interface (GUI), the application tier handles logic and the data tier stores information.

- **Presentation Layer**
    - The presentation layer typically consists of the user interface that users interact with while accessing the website. This includes the design, layout, and styling of the website's pages, as well as any interactive elements such as buttons, forms, menus, images, videos, audios, etc.
    - Some common technologies and tools used to create the presentation layer of a website include HTML, CSS, and JavaScript for the front-end development, and libraries or frameworks such as Bootstrap, React, etc. It is important to ensure that the presentation layer is accessible and responsive, meaning that it can be used on a variety of devices and screen sizes and that it adheres to web accessibility guidelines to ensure that all users can interact with the website.
    - This layer is distributed to a computing device using a web browser or a web-based application and is constructed with ReactJS. Application program interface (API) calls are the primary communication between the presentation tier and the other levels. The user interface should be easy to navigate and aesthetically pleasing. It should include a homepage, events page, and resources page.
- **Business Logic Layer**
    - The business logic layer, also called the application layer, is responsible for managing the business logic and data operations related to the club's activities. It typically includes functionality such as membership management, event registration, and data analytics.
    - The business logic layer acts as an intermediary between the presentation layer (which includes the website's user interface) and the data layer (which includes the database and other data sources). It ensures that data is validated, processed, and stored correctly and that the website functions smoothly and efficiently. We will be implementing the business logic layer using Node.js.

- **Data Access Layer**

    - The data access layer is responsible for managing the communication between the website's business layer and the database. It handles tasks such as retrieving, updating, and deleting data, and ensures that the data is stored and managed efficiently and securely.

    - We will use MongoDb to implement this layer, which should store and manage user account information, event details, and other club-related information.