IT-314 Software Engineering
Lab-6

Group Members:

Pranav Patil (202001055)
Meet Sutariya (202001046)
Dev Patel (202001014)
Vyom Patel (202001045)
Vandan Patel (202001041)
Nisarg Jadav (202001010)
Rohit Rao (202001003)
Kushal Soni (202001058)
Kanishk Kant (202001047)
M M Ajeya (202001036)

Q-1 : Create the domain analysis models for your course Project.

Answer : Our project is based on a hackathon website which will be mainly used by Data science students for competitive programming (Based on Machine learning models / Data Science algorithms) .

Below is the gathered information we get while doing the domain analysis.

- **User categories:**

Participants: Individuals or teams who will be participating in the hackathon event. They will need to register, submit their projects, and communicate with other participants.

Sponsors: Companies or organizations that provide financial or other resources to support the event. They may also be interested in promoting their products or services to the participants.

Organizers: The team that plans and manages the hackathon event, including logistics, communication, and coordination.

Viewer: The viewer category can visit the website and give us feedback.

- **User needs:**

Participants: Participants need to be able to register for the event, find teams to join or create their own team, submit their projects, communicate with other participants, and access resources such as guidelines, rules, and toolkits.

Sponsors: Sponsors need to be able to promote their products or services to the participants, get visibility for their brand, and access the submissions.

Track Organizers: Organizers need to be able to manage the registration process, coordinate the event, communicate with the participants, judges, and sponsors, and track the progress of the event.

Viewer: Able to visit the homepage, search the tracks, and visit the question.

- **Activities:**

Registration : Participants will need to register for the hackathon event, providing their personal and contact information.

Team formation : Participants will be able to find other participants to form a team, or create their own team.

Code submission: Participants will be able to submit their codes, including modules.

Evaluation: Based on certain criteria evaluation will be done for the user's algorithm.

Resources: The website will provide resources such as guidelines, rules, toolkits, and links to useful tools and technologies.

- **Information resources:**

Guidelines: Rules and guidelines for the hackathon event.
Rules: Detailed rules for the event, including eligibility criteria, submission guidelines, and judging criteria.

Toolkits: Toolkits to help participants with their projects, including links to relevant software, APIs, and datasets.

Schedule: The schedule for the hackathon event, including important deadlines and milestones.

Judging criteria: Criteria used for checking accuracy of user's algorithms.

- **Technical requirements:**

Submission portals: Portals for participants to submit their codes.

Judging tools: Tools for judges to evaluate the submissions.

- **Social media integration:**

Social sharing: Participants will be able to share their experiences on social media, promoting the event and increasing its visibility.

- **Event management:**

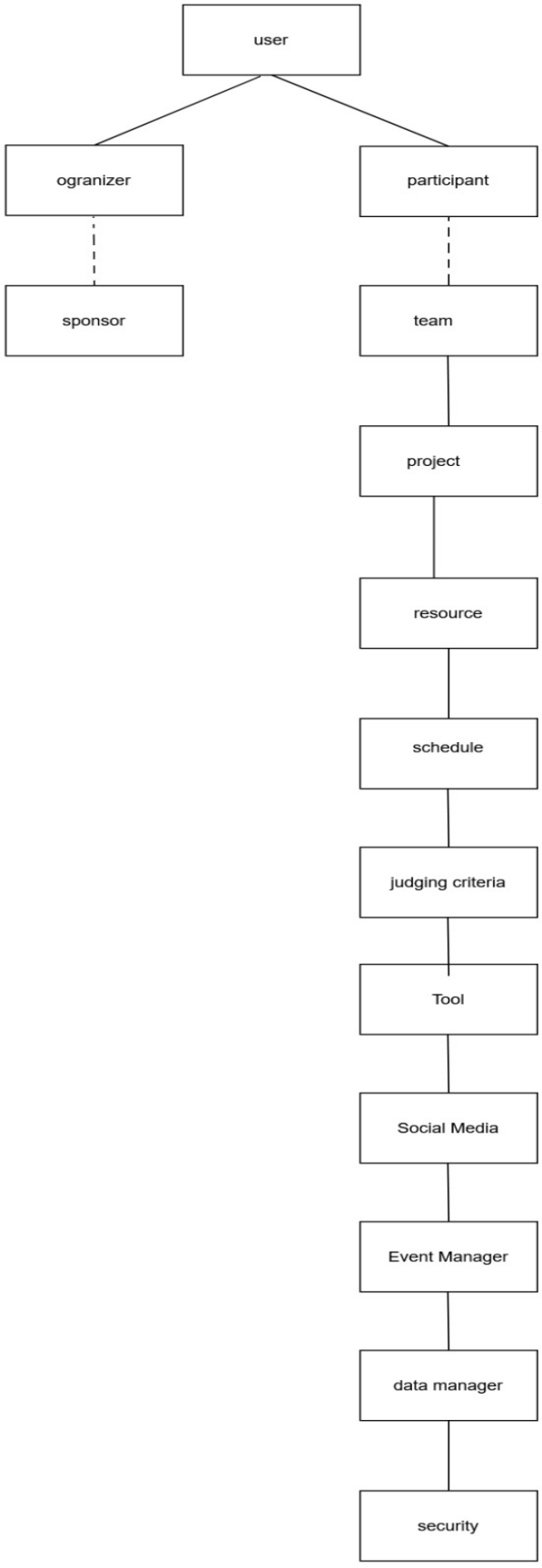
The hackathon website needs to support event management, such as coordinating with sponsors and ensuring compliance with rules and regulations.

- **Data management:**

The hackathon website needs to manage and store various data, such as participant information, project submissions, judging results.

- **Security and privacy:**

The hackathon website needs to ensure the security and privacy of user data, such as implementing secure logins etc.


user
ogranizer
participant
sponsor
team
project
resource
schedule
judging criteria
Tool
Social Media
Event Manager
data manager
security

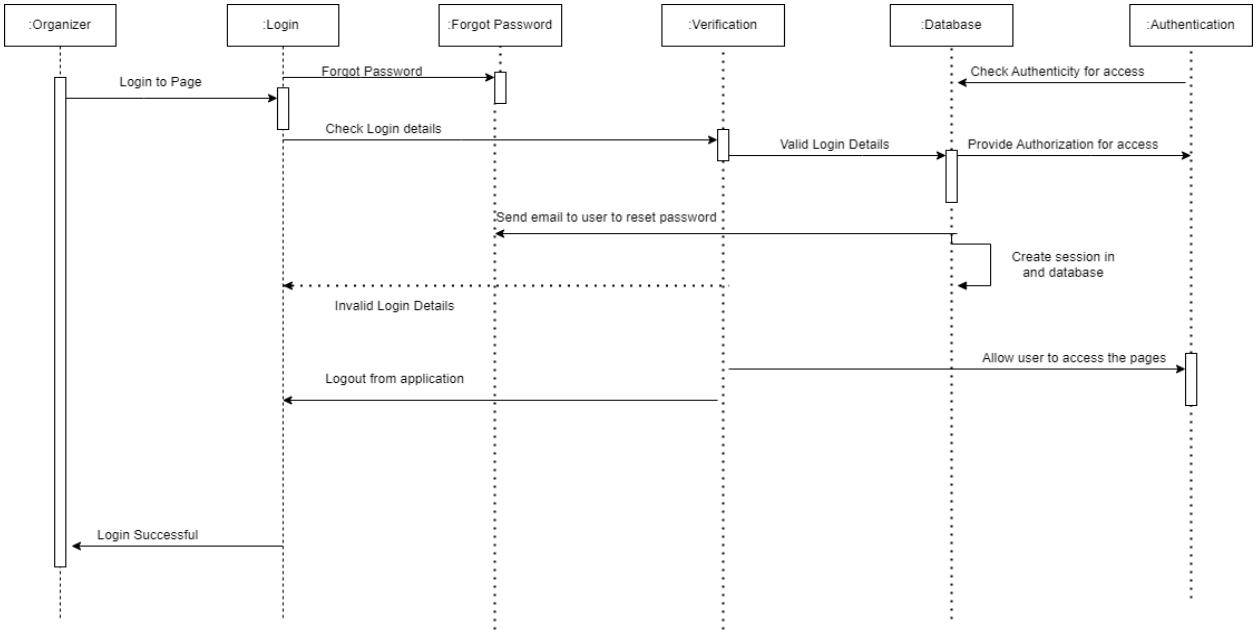
Q2.Identify boundary, entity, control object.

Boundary: The boundary for the website would be the outermost layer of the website that separates the website from the external world. In this case, the boundary would be the website itself and its associated components such as servers, databases, etc.

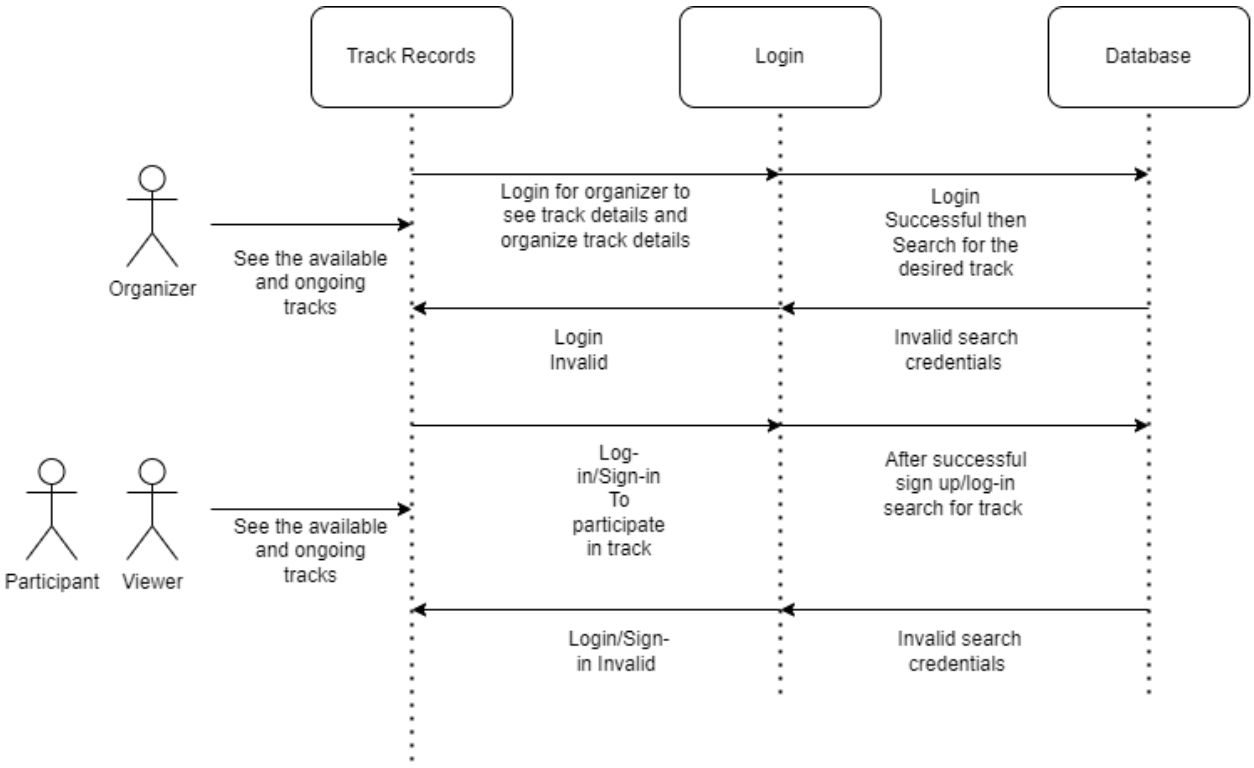
Entity: There are several entities associated with the website, such as the competition organizer, participant, viewer, and admin. Additionally, there are also entities such as the past standings, past competitions, and their results.

Control Object: The control object for the website would be the management of competitions and their associated data. This would include tasks such as creating new competitions, managing participant entries, calculating standings, and displaying results. Additionally, the control object would also be responsible for managing user access, such as allowing admin users to modify data and restricting access to certain parts of the website for viewers.

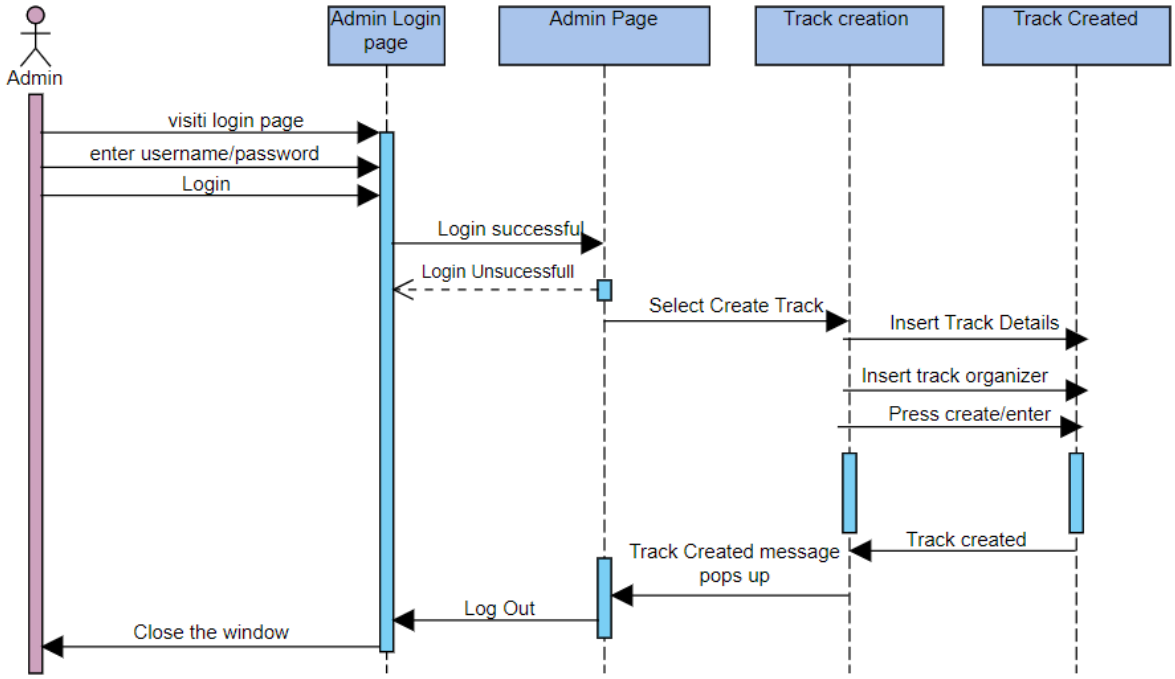
SEQUENCE DIAGRAM FOR LOGIN



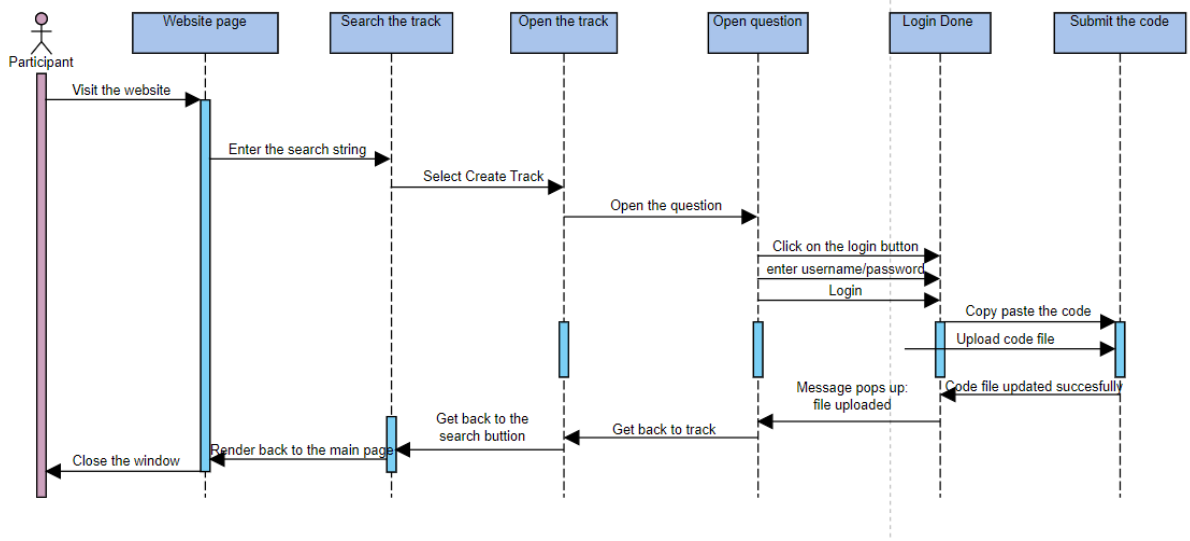
Search for a track:



Edit a track(Via admin):



Submit the code by participant:



Design goals

- Reliability
- Ease of use
- Rapid development
- User - friendliness
- Understandability
- Ease of learning
- Ease of remembering

Q3.Create a high level system design

Architecture:

The website will be built on a **client-server architecture** where the client-side will be implemented using **React** and the server-side will be implemented using a server-side programming language like **Django**. The website will use a database like **MySQL or MongoDB** to store the user data.

Subsystems:

User Management System: This subsystem handles user registration, authentication, and authorization. It also manages user profiles and preferences.

Track Management System: This subsystem manages the creation, management, and promotion of Tracks(challenges). It allows organizers to create and manage Track pages, manage submissions, and judge submissions. It also allows participants to view and join tracks, view challenges, and submit their solutions.

Analytics and Reporting System: This subsystem provides data analytics and reporting tools for Track organizers. It tracks and reports on submission quality, and other important metrics.

Infrastructure and Security System: This subsystem ensures the website's reliability, scalability, and security. It manages hosting, backup, and disaster recovery.

Participation management system: This subsystem manages all the participation and their activities during a particular track such as submitting a solution, preparing leaderboard, etc.

This architecture consists of a presentation tier, an application tier, and a data storage tier. Here is an example of how this architecture could be organized using package diagrams:

Presentation Tier:
The presentation tier is responsible for handling user interface and user input. This tier includes the following subsystems:
User Interface: This subsystem includes the web pages and forms that allow users to interact with the website.
Client-side Logic: This subsystem includes the JavaScript and other client-side code that handles user input validation and other user interactions.

Application Tier:
The application tier is responsible for handling the business logic and processing user input. This tier includes the following subsystems:
Controller: This subsystem is responsible for managing the flow of data between the user interface and the data storage tier.
Business Logic: This subsystem is responsible for implementing the business logic of the website, such as user registration, post creation, and ranking evaluation.

Security: This subsystem is responsible for implementing security features, such as user authentication and authorization.

Data Storage Tier:
The data storage tier is responsible for storing and managing data. This tier includes the following subsystems:
Data Access Layer: This subsystem is responsible for accessing and retrieving data from the database.
Database: This subsystem is responsible for storing all of the website's data, such as user information, forum data, topic data, post data, tag data, and ranking data.
Overall, this three-tier architecture allows for a clear separation of concerns between the presentation, application, and data storage tiers. The use of subsystems within each tier further helps to organize and modularize the codebase, making it easier to maintain and update in the future.