Software Engineering (Sessional) Final Report

Course: CSE-434

CUET CP Community Management System (Web Application)

Team Members

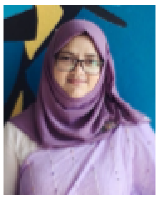
Zerin Shaima Meem
ID: 1804057

Ashfaqur Rahman
ID: 1804055

Md.Taosiful Islam
ID: 1804041

Department of Computer Science and Engineering (CSE)
Chittagong University of Engineering & Technology (CUET)
Chattogram- 4349, Bangladesh

Team Members

Photo	Name	ID	Email	Contact NO.	Total Credit Passed	CGPA acquired
	Zerin Shaima Meem	1804057	u1804057@ student.cuet.ac.bd	01839645766	141.75	3.68
	Ashfaqur Rahman	1804055	u1804055@ student.cuet.ac.bd	01996400496	141.75	3.56
	Md.Taosiful Islam	1804041	u1804041@ student.cuet.ac.bd	01627485404	141.75	3.10

Executive Summary

The proposed Online Competitive Programming Community Management System is a comprehensive platform designed to foster a vibrant and supportive coding environment for Chittagong University of Engineering and Technology (CUET) students. Leveraging modern technologies and best practices, the system aims to enhance students' problem-solving skills while cultivating a competitive yet collaborative atmosphere. It also seeks to provide recognition for top performers within the community.

The system is developed using a robust technology stack, including Visual Studio Code as the integrated development environment (IDE), GitHub for version control, and a frontend built with JavaScript, HTML, CSS, and the Angular framework. On the backend, Node.js and Express.js handle server-side operations, while MongoDB serves as the database to store and manage data efficiently.

From the User View perspective, students can access essential features to engage with the platform effectively. They can submit Registration Requests to gain access, view the Leaderboard to track their performance relative to peers, view their Profile to track progress and personal information, explore upcoming Contests to participate in coding competitions and challenges, and stay informed about important announcements via the Announcements page.

Administrators, on the other hand, have access to an Administrative View that empowers them to manage the platform efficiently. They can review and process User Registration Requests to maintain community integrity, manage Announcements to keep users informed about upcoming events and opportunities and oversee Contests by adding new contests, updating details, or removing outdated ones as needed.

Key features and benefits of the system include a focus on enhancing problem-solving skills through a variety of resources, contests, and practice problems tailored to different skill levels. Community engagement is encouraged. Recognition and motivation are provided through the Leaderboard, acknowledging top performers and motivating users to excel. Additionally, timely information about upcoming contests, events, and announcements enables users to plan their participation effectively, contributing to a dynamic and engaging coding community at CUET.

In conclusion, the Online Competitive Programming Community Management System represents a pivotal initiative to cultivate a thriving coding community at CUET. The platform aims to empower students to enhance their problem-solving skills, engage in healthy competition, and contribute positively to the coding landscape. With its user-friendly interface, robust features, and commitment to continuous improvement, the system is poised to become a cornerstone of CUET's competitive programming ecosystem.

Contents

List of Figures

List of Tables

Introduction

The envisioned project, titled the CUET CP Community Management System (CP-CMS), emerges as a response to the evolving needs of the Chittagong University of Engineering and Technology (CUET) coding community. Recognizing the significance of an organized platform to foster collaboration and engagement among students and faculty, the CP-CMS is conceived to be a comprehensive solution overseen and managed by the community itself.

The primary goal of the CP-CMS is to establish a centralized hub that caters to the diverse requirements of the CUET coding community. By providing real-time, accessible, and customizable features, the system aims to enhance the efficiency and vibrancy of the coding culture within CUET. The project envisions a user-centric approach, allowing individuals to tailor their interactions with the system based on their unique needs.

1.1 Goals and Objectives of the project

The overarching objectives of the CP-CMS project are as follows:

- Develop a user-friendly web interface for seamless navigation and interaction.
- Implement features that enable users to customize their experience based on their roles and preferences.
- Create a platform to visualize performance rank in different online judges.
- Create a dynamic platform for users to access and participate in coding contests.
- Establish an efficient communication channel through announcements and customized content.
- To Provide an intuitive administrative portal for the systematic management of user data, contests, announcements and leader-board.

1.2 Scope of the work

The CP-CMS is tailored to serve the coding community within CUET, encompassing students and mentors. The system defines specific user roles, ensuring role-specific access and capabilities. It places a primary focus on real-time updates concerning coding contests, fostering a dynamic environment for effective planning and participation. Additionally, users will benefit from features tailoring information to their ranks and performance in real-time.

1.2.1 Current situation and context

The current management of the coding community relies on manual process, leading to potential challenges and security concerns. The transition to the CP-CMS, into a web application is pivotal to streamline interactions, enhance efficiency, and address security vulnerabilities inherent in the current manual processes.

1.3 System overview

The CP-CMS consists of three interconnected modules: User Registration Module, Leaderboard Management Module and Adminstrative Operation Module. These modules collectively aim to provide a secure, personalized, and engaging experience for the CUET coding community, fostering a collaborative and efficient coding culture.

1.4 Structure of the document

The entire document is divided into 6 sections, these are given below:

1. **Chapter 1:** This Chapter aims to provide a general overview and introduction to the project. We provide a summary of the project's objective and scope, as well as a list of relevant terminology.
2. **Chapter 2:** In this Chapter, we will discuss our web app's project management strategy. Our presentation will cover project contributions, process model, risk analysis, implementation constraints, hardware and software requirements, timeline, budget, and social, cultural, and environmental impacts.
3. **Chapter 3:** In this Chapter, we will define our project's requirements. Our system's stakeholders will be described in depth, as will use case diagrams with graphical and textual descriptions, activity diagrams, class diagrams, sequence diagrams, and safe security requirements.
4. **Chapter 4:** In this Chapter, we will go over our project's architecture. Technologies, software, and hardware used in architecture. What we used will be explained.
5. **Chapter 5:** In this Chapter, we'll describe how our system is set up. We'll talk about how different parts work together (component level design) and how the user interface looks (GUI design).
6. **Chapter 6:** In this Chapter, we'll explain how we plan to test our project and make it sustainable. We'll discuss tests that match our requirements, how we link tests to our usage scenarios, how we create tests, and check the overall quality of our testing setup. We'll also cover what our project aims to achieve, our future plans, and how we plan to keep it sustainable. At the end, we'll include references and give acknowledgments

1.5 Terms, Acronyms, and Abbreviations Used

Stakeholders.

Project Management Plan

2.1 Project Organization

1.5 In our teamwork to develop the CUET CP Community Management System, our teachers are guiding us. They help us understand the project requirements, offer valuable insights, and ensure that our work aligns with the goals and standards set for the project. Their guidance is crucial in making sure our efforts contribute to the success of the overall system. There are three of us in the team. Each person has a different job: one for the backend part, one for the frontend part, and one for the admin side. Initially, we identify the project requirements and design its appearance. Subsequently, each team member focuses on their individual tasks.

2.1.1 Individual Contribution to the project

Individual Contributions to the project are shown in Table-2.1

Member Name	Requirement Specification	Planning	Designing	Frontend	Backend	Testing	Deploy-ment
Zerin Shaima Meem	✓	✓	✓	✓		✓	
Ashfaqur Rahman	✓	✓			✓	✓	✓
Md.Taosiful Islam	✓	✓	✓	✓			✓

Table 2.1: Individual Contribution to the project

2.2 Process Model Used

In this project, we will adopt the Agile model for software development. The Agile model follows an iterative and collaborative approach, allowing flexibility and adaptability throughout the development process. The key stages in Agile include:

1. **Planning**

2. **Designing**
3. **Coding**
4. **Testing**
5. **Deployment**

Agile allows for concurrent execution of stages, encouraging continuous collaboration and adaptation. This dynamic methodology aims to deliver functional software in shorter cycles, enabling rapid responses to changing requirements and ensuring client satisfaction.

2.2.1 Rationale for choosing lifecycle model

The rationale for choosing a specific lifecycle model in software development is crucial and depends on various factors. For the chosen lifecycle model, the reasons may include:

1. **Project Requirements:** Depends on project needs. Waterfall suits well-defined requirements, while Agile is good for evolving needs.
2. **Project Size and Complexity:** Large projects may benefit from Waterfall, while Agile suits smaller, flexible projects.
3. **Client Involvement:** Agile involves clients throughout, while Waterfall has specific client stages.
4. **Flexibility and Adaptability:** Agile is flexible to changes, while Waterfall is less adaptable.
5. **Resource Constraints:** Waterfall is chosen when there are strict resource constraints; Agile is more flexible.
6. **Previous Project Success:** Past successes with a model influence the choice for consistency.
7. **Risk Tolerance:** Agile adapts to risks, while Waterfall plans to mitigate them.
8. **Regulatory Compliance:** The chosen model must align with industry regulations.

In summary, the choice depends on project-specific factors, organizational preferences, and the balance between planning and adaptability. Each model has strengths and weaknesses, and the selection should align with the project's unique characteristics and goals.

2.3 Risk Analysis

The risk analysis for the CP Management System identifies potential challenges and their mitigations. Incomplete requirements pose a risk mitigated through regular stakeholder communication, while technical complexity is addressed with thorough feasibility studies. User acceptance concerns are managed by involving users in the design process and providing training. Security risks are mitigated through robust measures and regular audits. Integration challenges are handled by analyzing integration points and collaborating with IT teams. Resource constraints are addressed with realistic budgeting and effective resource allocation. Scalability concerns are mitigated by designing the system for scalability and conducting stress testing. Adapting to changes in technology is managed by

staying updated and choosing flexible technologies.
User engagement risks are addressed with features encouraging participation and community events. Regulatory compliance is ensured by regularly reviewing and adhering to relevant regulations. Regular updates to the risk analysis throughout the project ensure ongoing risk management.

In the follwing table shows possible problems in the CP Management System project, putting them into categories. It also rates how likely each problem is and how much it might affect the project. This helps us plan ahead to avoid or handle these issues.[?].

Risks	**Category**	**Probability**	**Impact**
Incomplete Requirements	Project Requirements	Moderate	High
Technical Complexity	Technical	High	Moderate
User Acceptance	User Engagement	Low	High
Security Concerns	Security	Moderate	High
Integration Issues	Technical	Moderate	Moderate
Resource Constraints	Project Management	Low	High
Scalability	Technical	Moderate	Moderate
Changes in Technology	Technical	Low	High
Lack of User Engagement	User Engagement	Low	Moderate
Regulatory Compliance	Compliance	Moderate	High

Table 2.2: **Risk Analysis for CP Management System**

2.4 Constraints to project implementation

Constraints to project implementation refer to limitations or restrictions that may affect the successful execution of a project. These can include:

- **Budget Constraints:** Limited financial resources impacting tool, technology, and personnel acquisition.
- **Time Constraints:** Tight deadlines restricting comprehensive planning, development, and testing time.
- **Resource Limitations:** Insufficient manpower, expertise, or technology hindering project progress and quality.
- **Technical Constraints:** Compatibility issues, outdated technologies, or hardware limitations posing challenges during implementation.
- **Scope Creep:** Expansion of project requirements beyond the initial scope straining resources and timelines.

2.5 Hardware and Software Resource (Tools/Language) Requirements

- Software Tools: VS Code, POSTMAN
- Software Languages
  - **Front-end**: Angular
  - **Back-end**: Node JS
  - **Database**: MongoDb
- Hardware tools: Personal computer
- Hosting: Netlify

2.6 Project Timeline and Schedule

- **Specifying requirements:** 14 days
- **Planning:** 10 days
- **Modeling:** 10 days
- **Code generation:** 14 days
- **Testing:** 4 days
- **Deployment:** 2 days

2.7 Estimated Budget

The Estimated Budget is a plan for the money needed in the CP Management System project, covering costs for software, infrastructure, and people. It helps manage spending and ensures the project stays within its financial limits. Here is the budget estimation for the development of the CUET CP Management System.

Expense Category	**Estimated Cost**
Personal Computer (3 units)	80,000 x 3
Report/Documentation Writing Papers	2,000
Miscellaneous Expenses	2,000
Total Estimated Budget	**2,44,000**

Table 2.3: Estimated Budget for CP Management System (in BDT)

2.8 Social/Cultural/Environmental impact of the project

Social Interaction:
The system brings people together to work and share ideas. It helps create a strong and supportive community.

Economic Impact:
The CP Management System stimulates economic growth by enhancing coding efficiency, fostering skill development, attracting opportunities, and gaining industry recognition.

Networking Opportunities:
The system provides chances to meet and connect with others in the field. It opens up opportunities to build professional networks.

Knowledge Sharing:
The system allows people to share what they know with others. It enhances learning and helps everyone grow their knowledge.

Requirement Specifications

3.1 Stakeholders for the system

The stakeholders for the ”CUET CP Community” system include:

- **Users:** The general public who will use the system, including students (competitive programmers) in the CUET CP community.
- **Admins:** Individuals responsible for system maintenance, user management, and addressing issues. Trainers and faculty members are included in this section

3.2 Use case diagram with Graphical and Textual Description

The Use Case Diagram for the CUET CP Management System illustrates the interactions between different stakeholders and the system itself with three key use cases [**?**]. This overall use case is further designed into three use cases according to the three modules of the system.

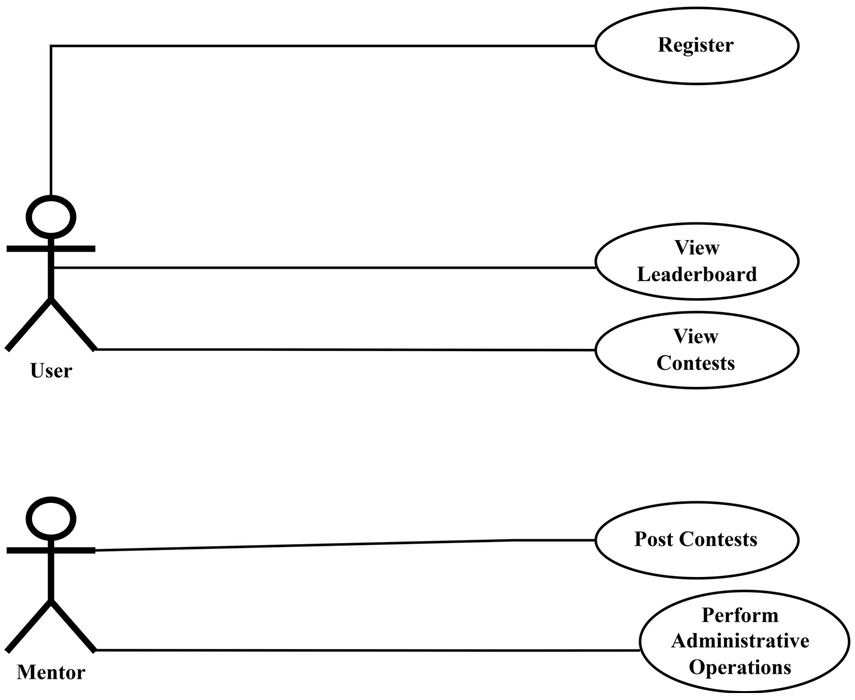


Figure 3.1: Use Case Diagram for the overall system

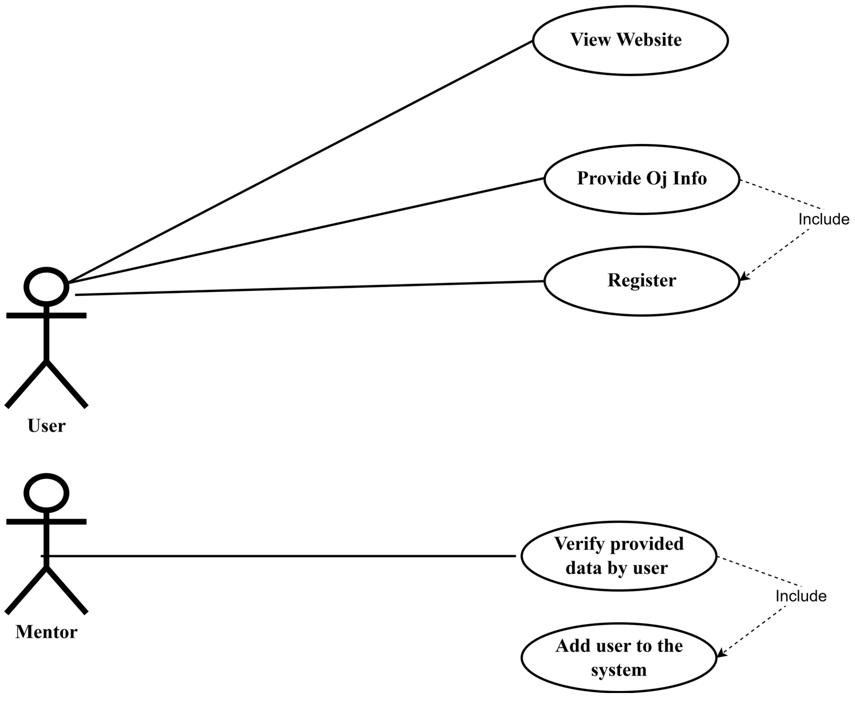


Figure 3.2: Use Case Diagram for User Management Module

Use Case: Overall System Functionality

Iteration: 2

Primary Actor: User

Goal in Context: To facilitate user registration, administrative operations, leaderboard viewing, and contest exploration in the CP Community Management System.

Preconditions:

- User has access to the CP Community Management System.
- Admin is logged into the system.
- System is operational.
- APIs for fetching online contests are accessible.

Trigger: User decides to interact with the CP Community Management System.

Scenario:

1. **User Registration:**
    - User initiates the registration process.
    - System validates user data from different online judges.
    - Admin verifies and approves the user registration.
    - Users are added to the system.
2. **Admin Operations:**
    - Admin performs various administrative activities (e.g., update user information, post announcements, post custom contests, delete user instances).
3. **User Views Leaderboard:**
    - User logs into the system.
    - System fetches the latest user performance data from different online judges' APIs.
    - System processes the data and generates the leaderboard.
    - User views the rankings and performance metrics of users.
4. **User Views Upcoming Contests:**
    - User navigates to the contest page.
    - System fetches and displays the list of upcoming contests from different online judges' APIs.

Exceptions:

1. Issues in user registration—see use case "Handle User Registration Issues."
2. Issues in admin operations—see use case "Handle Admin Operation Issues."
3. Issues in fetching online contests from APIs—see use case "Handle API Fetching Issues."

Priority: High priority, essential for community engagement.

When Available: First increment

Frequency of Use: Frequent

Channel to Actor: Via web browser and Internet connection.

Secondary Actors: Mentor, System, API.

Channels to Secondary Actors:

- Admin: System dashboard with admin tools.
- System: Integration with online judges' APIs for data fetching and processing.
- API: Providing data for online contests.

Open Issues:

1. Mechanisms to handle potential issues during user registration, admin operations, and API data fetching.
2. Admin tools for efficient user management, announcement, and custom contest posting.
3. Considerations for secure storage and handling of user information.
4. Notification mechanism to inform users of announcements or posted custom contests.
5. User interface design for an intuitive and informative experience across various functionalities.

Use Case 1: User Registration

Iteration: 2

Primary Actor: User

Goal in Context: To register as a user in the CP Community Management System.

Preconditions:

- The user is on the CP Community Management System website.
- The user provides his information from different online judges.

Trigger: The user decides to register on the CP Community Management System.

Scenario:

1. User explores the CP Community Management System.
2. User initiates registration.
3. User provides his information from different online judges (e.g., Codeforces, AtCoder, CodeChef).
4. User submits the registration form.
5. The system notifies the admin of a new user registration.
6. Admin checks the validity and consistency of the information.
7. If the data is verified, the admin adds the user to the CP Community Management System.
8. System notifies the user of successful registration.

Exceptions:

1. User provides invalid or inconsistent data—see use case "Handle Invalid User Data."
2. Admin encounters issues verifying user data—see use case "Admin Data Verification Issue."

Priority: High priority, essential for user engagement.

When Available: Initial release.

Frequency of Use: Frequent

Channel to Actor: Via web browser and Internet connection.

Secondary Actors: Mentor.

Channels to Secondary Actors:

- Admin: System dashboard with user registration notification and verification tools.

Open Issues:

1. Mechanisms to handle invalid or inconsistent user data.
2. Admin tools for efficient verification and user addition to the system.
3. Considerations for secure storage and handling of user information.
4. Notification mechanism to inform users of successful registration.
5. User interface design for a seamless registration experience.

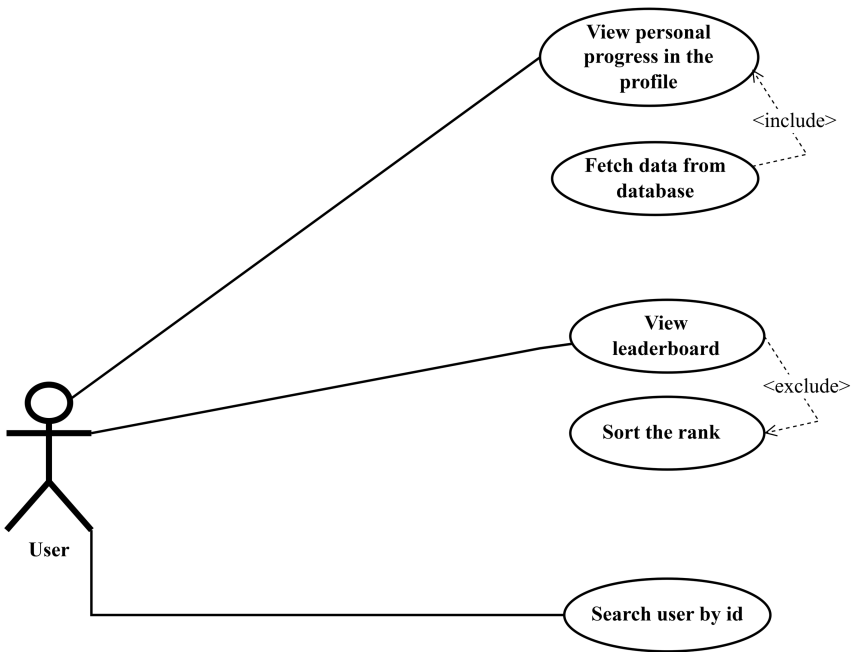


Figure 3.3: Use Case Diagram for Leaderboard management

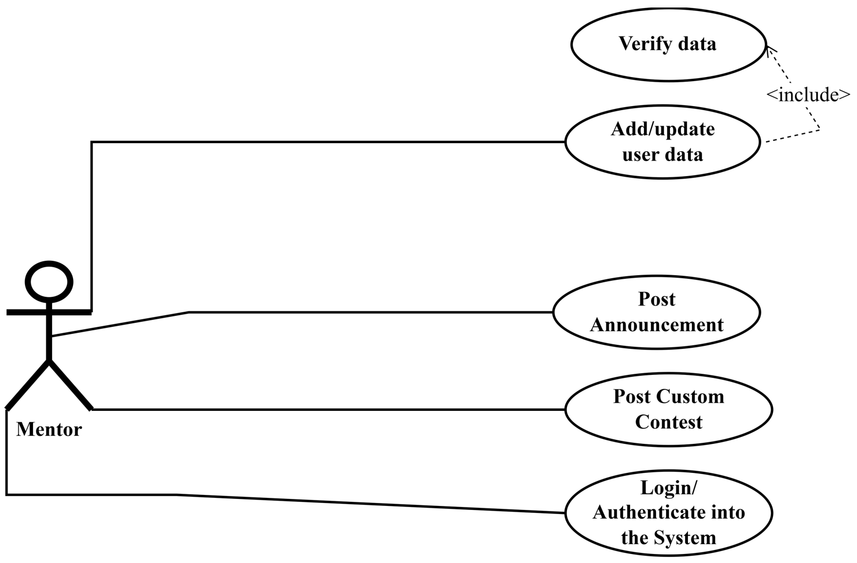


Figure 3.4: Use Case Diagram for administrative management

Use Case 2: Interact with Leader-board

Iteration: 2

Primary Actor: User

Goal in Context: To view the leader-board/rankings based on user performance data fetched from different online judges.

Preconditions:

- The system has fetched user performance data from different online judges.
- System processed fetched data and generated a rank/leaderboard of all users.

Trigger: The user decides to view the leader-board.

Scenario:

1. System requests and fetches user performance data.
2. The system calculates user rankings and generates the leaderboard.
3. The user navigates to the leaderboard.
4. The system displays the processed data on the leaderboard page.

Exceptions:

1. Issues in fetching data from online judges' APIs—see use case "Handle API Fetching Issues."
2. The system encounters errors while processing data—see use case "Handle Data Processing Errors."

Priority: High priority, essential for user engagement.

When Available: Initial release.

Frequency of Use: Frequent.

Channel to Actor: Via a web browser having an internet connection.

Secondary Actors: System

Channels to Secondary Actors:

- System: Integration with online judges' APIs for data fetching and processing.

Open Issues:

1. Mechanisms to handle potential issues when fetching data from online judges' APIs.
2. Robust data processing to ensure accurate and timely leaderboard generation.
3. User interface design for an intuitive and informative leaderboard display.
4. Considerations for real-time or periodic updates of the leaderboard based on the frequency of data updates from online judges.

Use Case 3: Administrative Operations

Iteration: 2

Primary Actors: Mentor

Goal in Context: To manage user registrations, verify user data, update user information, post announcements, and post custom contests in the CP Community Management System.

Preconditions:

- The admin is logged into the system.
- The admin has the necessary authorization to manage transportation information.

Trigger: Admin decides to perform administrative activities.

Scenario:

1. **Adding User**
    - System notifies the admin when a new user registers.
    - Admin reviews the provided data from different online judges.
    - Admin checks the validity and consistency of the information.
    - Admin decides to approve or reject the user registration.
    - If the data is verified and approved, admin adds the user to the CP Community Management System.
2. **Admin updates/deletes user information in the system.**
3. **Admin creates and posts announcements in the system.**
4. **Admin posts custom contests in the system.**

Exceptions:

1. Issues in verifying or updating user data—see use case "Handle User Data Verification Issues."
2. Admin encounters errors while posting announcements or custom contests—see use case "Handle Admin Posting Errors."

Priority: High priority, essential for system management.

When Available: Initial release.

Frequency of Use: Moderate.

Channel to Actors: Via web browser and Internet connection.

Secondary Actors: System

Channels to Secondary Actors:

- System: Database updates for user management, announcements, and custom contests.

Open Issues:

1. Mechanisms to handle potential issues when verifying or updating user data.
2. Admin tools for efficient user management, announcement, and custom contest posting.
3. Considerations for secure storage and handling of user information.
4. Notification mechanism to inform users of announcements or posted custom contests.
5. User interface design for an admin dashboard and announcement/custom contest creation.

1.5

3.3 Sequence diagrams

A UML sequence diagram visually depicts dynamic interactions among users, administrators, and the database over time. It showcases key actions such as user registration, login, filtering challenges, viewing details, commenting, and challenge management. Lifelines represent these entities, illustrating synchronous and asynchronous collaborations for efficient user interactions, information retrieval, and system management [?].

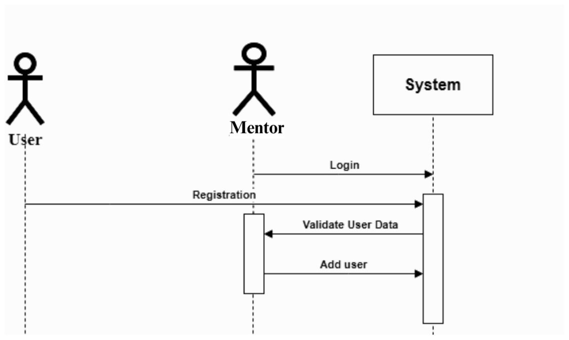


Figure 3.5: Sequence Diagram of User Management

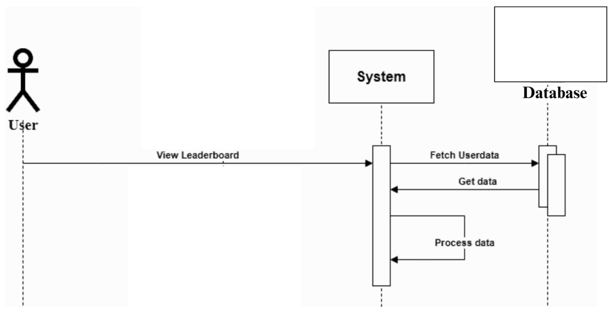


Figure 3.6: Sequence Diagram of Leaderoard

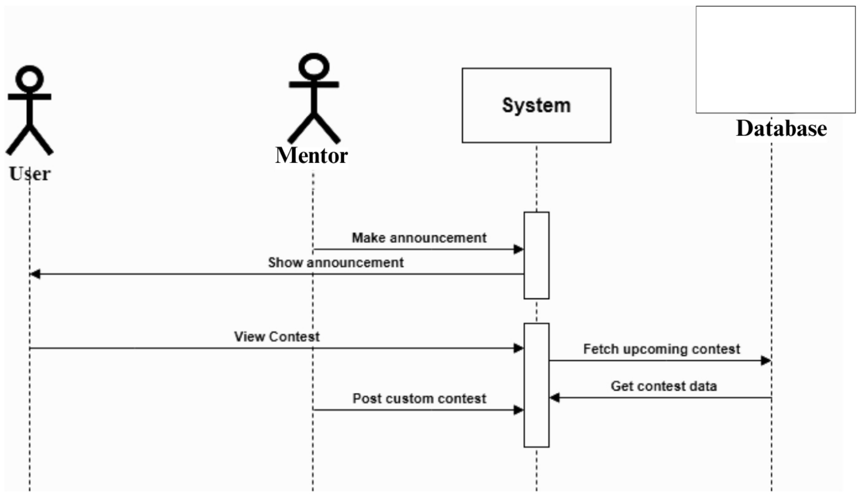


Figure 3.7: Sequence Diagram of Mentor activity

3.4 Activity diagrams

Activity diagrams in software engineering are graphical representations that illustrate the dynamic aspects of a system, showcasing the flow of activities and actions. These diagrams utilize various symbols, such as activities, actions, decision nodes, merge nodes, initial and final nodes, as well as fork and join nodes to depict the sequence and relationships between different processes [?].

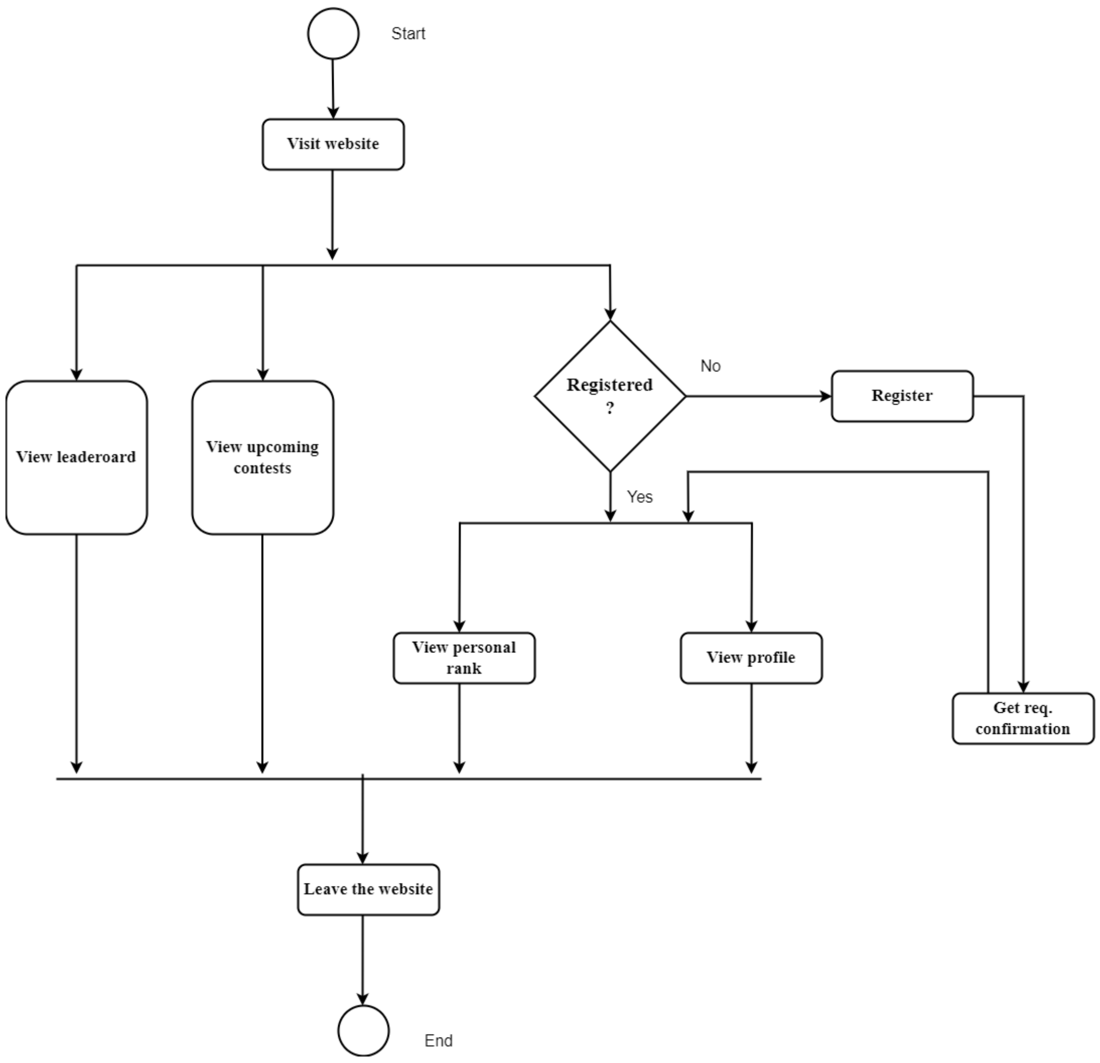


Figure 3.8: Activity Diagram of User Management Module

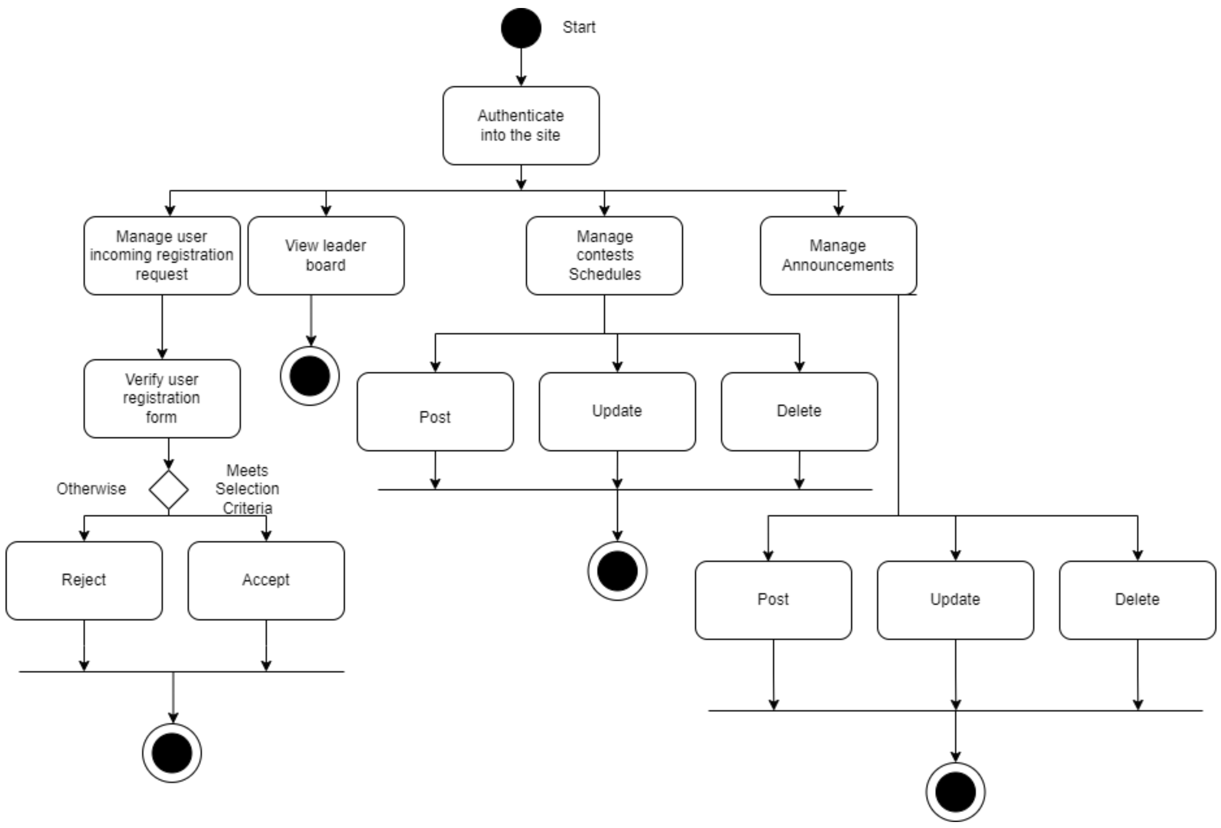
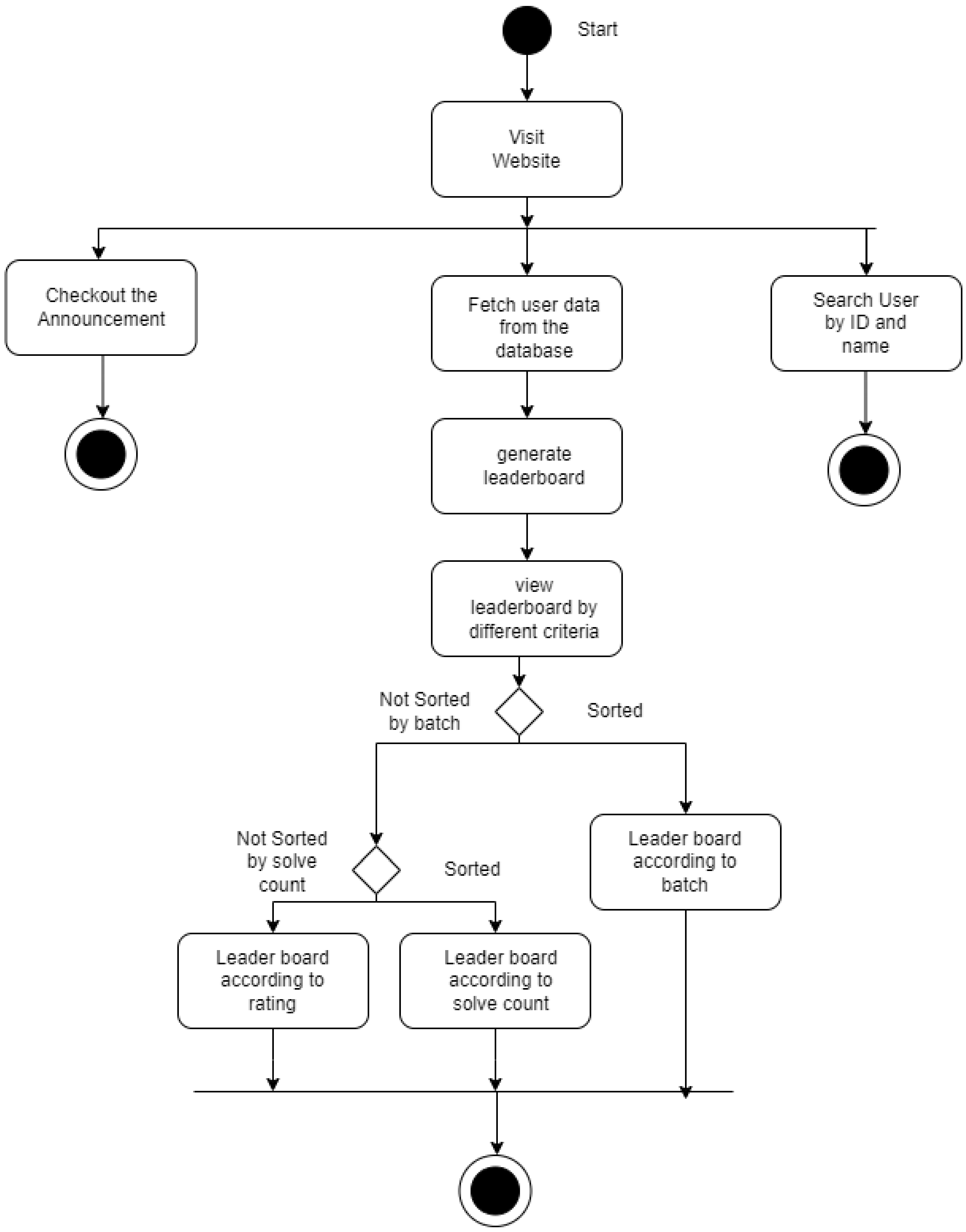

Start
Authenticate into the site
Manage user incoming registration request
View leader board
Manage contests Schedules
Manage Announcements
Verify user registration form
Otherwise
Meets Selection Criteria
Reject
Accept
Post
Update
Delete
Post
Update
Delete

Figure 3.9: Activity Diagram of Mentors Administrative Actions



Start
Visit Website
Checkout the Announcement
Fetch user data from the database
Search User by ID and name
generate leaderboard
view leaderboard by different criteria
Not Sorted by batch
Sorted
Leader board according to batch
Not Sorted by solve count
Sorted
Leader board according to rating
Leader board according to solve count

Figure 3.10: Activity Diagram of Leader-board

The state diagram represents the state transitions and interactions within CP club management system for different user roles. This state diagram provides a visual representation of the flow of user and mentor interactions in the system.[**?**].

3.5 Class diagram

This diagram provides a concise overview of management systems and relationships[?]. The class diagram depicts two main classes: User and Admin. A User has attributes

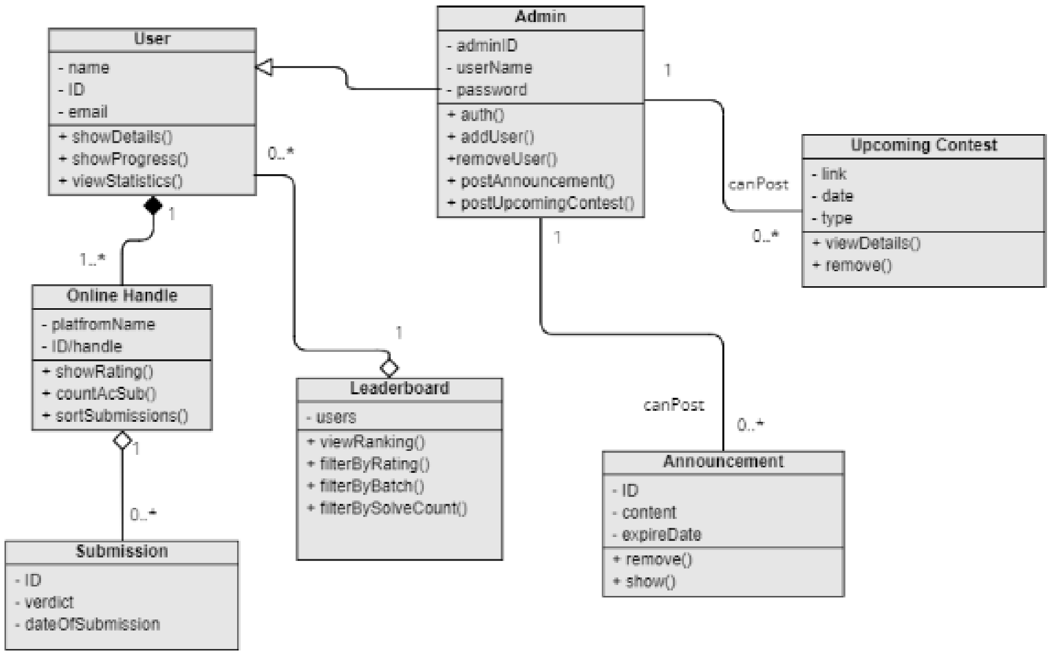


Figure 3.11: Class Diagram

such as name, id, and email, with methods to view their profile and the leaderboard. The Admin class includes attributes like id, name, and password, along with methods for user verification, addition, update, deletion, and posting announcements and contests. The associations show that one admin can post multiple contests and announcements. Additionally, there's an implicit association between User and Contest where users provide handle information to register for contests and view leaderboards.

3.6 Safety and Security Requirements

3.6.1 Access Requirements

Users

- Users must authenticate by logging into their respective user accounts.
- Users have the capability to update their profiles within the system.
- Users are granted access to browse data and rent bicycles at any time after logging in.

Administrators

- Administrators are provided with access to an admin panel for system management purposes.

- Administrators possess the authority to manage user accounts, including account creation, modification, and deactivation.
- Administrators are empowered to monitor system activities to ensure smooth operation and identify potential issues.

Guest Users

- Guest users, who haven't logged in, can access general information about the system.
- Guest users are required to register in order to rent bicycles from the system.

3.6.2 Integrity Requirements

Database Integrity

- Implement robust data validation checks to ensure that only valid data is accepted into the system.
- Apply database constraints to prevent the entry of inconsistent or incomplete data.

Product Integrity

- Address potential integrity issues within the system promptly and effectively.
- Implement version control mechanisms to track changes and updates to the software.
- Conduct regular testing and validation procedures to ensure that the system functions as intended without compromising data integrity.

3.6.3 Privacy Requirements

Data Privacy Measures

- Implement strong encryption protocols for sensitive data during transmission and storage to safeguard against unauthorized access.
- Utilize anonymization techniques for user information whenever feasible to enhance individual privacy protection.
- Conduct regular audits and monitoring of access to sensitive data to detect and prevent any unauthorized access attempts.

Architecture

1.5 The architectural design of the CUET CP Community Management System (CP-CMS) is carefully crafted to ensure efficiency, scalability, and seamless interaction between its components. The chosen architectural style encompasses a mix of technologies, with a focus on flexibility and modern development practices.

4.1 Architectural model/style used

4.1.1 Data Centered Architecture

1.5 The data-centered architecture of the CP-CMS is meticulously crafted to handle the system's data processing, storage, and communication needs. This architecture ensures robust performance, scalability, and reliability.

4.1.2 MVC Architecture

1.5 The CP-CMS adopts the Model-View-Controller (MVC) architecture, a widely used software design pattern in web and application development. MVC separates the application concerns into three interconnected components, namely Model, View, and Controller. This separation enhances code manageability, scalability, and reusability, allowing for modular development and maintenance.

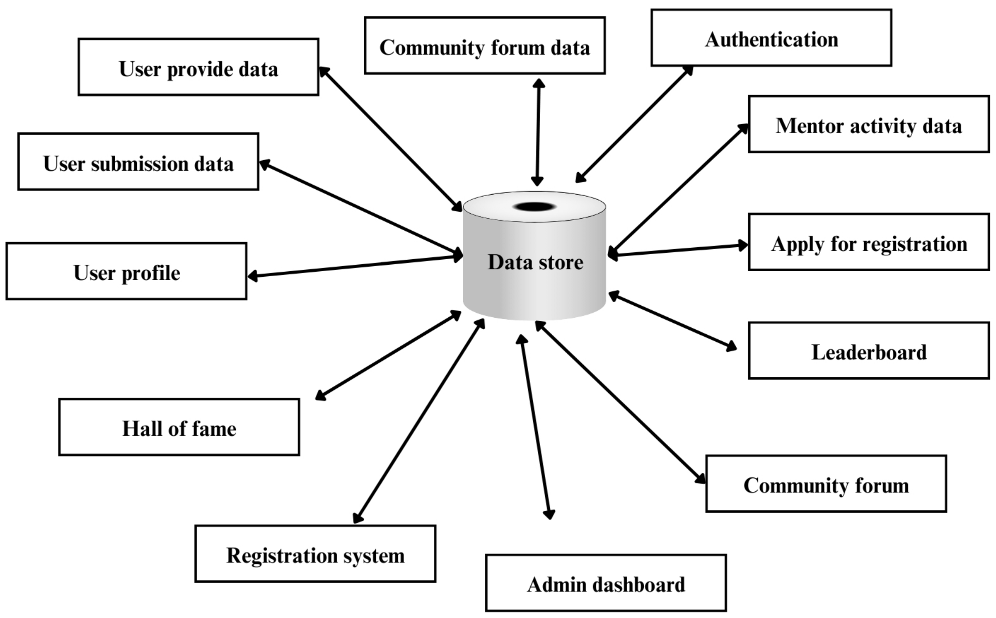


Figure 4.1: Data Centered Architecture of the System

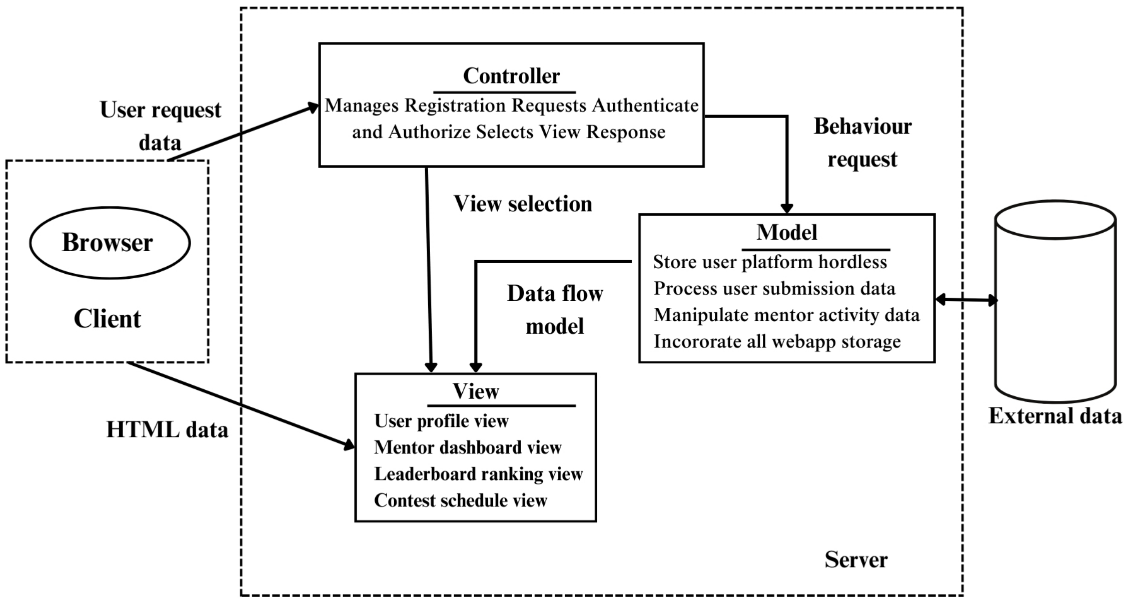


Figure 4.2: MVC Architecture of the System

4.2 Technology, Software, and Hardware Used

- **Angular:**
  - A TypeScript-based front-end web application framework.
  - Facilitates the development of dynamic and single-page applications.
  - Follows a component-based architecture for building modular and maintainable code.
- **Node.js:**
  - An open-source, server-side JavaScript runtime.
  - Allows the execution of JavaScript code on the server.
  - Enables building scalable and high-performance network applications.
- **Express:**
  - A minimal and flexible Node.js web application framework.
  - Simplifies the process of building robust web applications and APIs.
  - Integrates seamlessly with Node.js to create efficient server-side applications.
- **MongoDB database:**
  - MongoDB is a NoSQL database system designed for storing and managing unstructured data.
  - Its flexible document-oriented structure allows for easy scalability and adaptation to changing data requirements.
  - MongoDB integrates seamlessly with Node.js and Express, offering efficient data handling and real-time analytics capabilities for modern applications.

Design

5.1 Component level design following pattern

The Structure Chart for the CUET CP Community Management System has two main parts: ”User” and ”Admin.” It separates the experiences and tasks of regular users and administrators. The ”User” part includes signing up, joining coding contests, and checking leaderboards. On the other side, the ”Admin” part has tools for managing contests, making announcements, and handling user information. This simple structure makes the system easy to use and helps everyone, encouraging teamwork in the CUET coding community. With the help of [?] we have drawn the component design in figure 5.1:

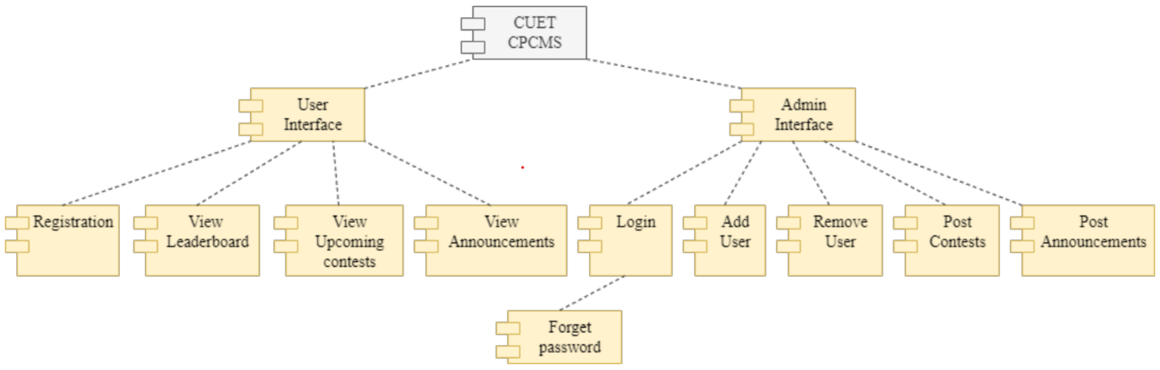


Figure 5.1: Structure Chart of CUET CP Community Management System

5.2 User Management Component

Users can register and sign up to join the CUET CP Community. Once registered, they can easily check their statistics and performance to see how well they are doing in coding from the leaderboard. This feature provides a personalized experience, allowing users to track their progress within the community. With the help of [?] we have drawn the component design in figure 5.2:

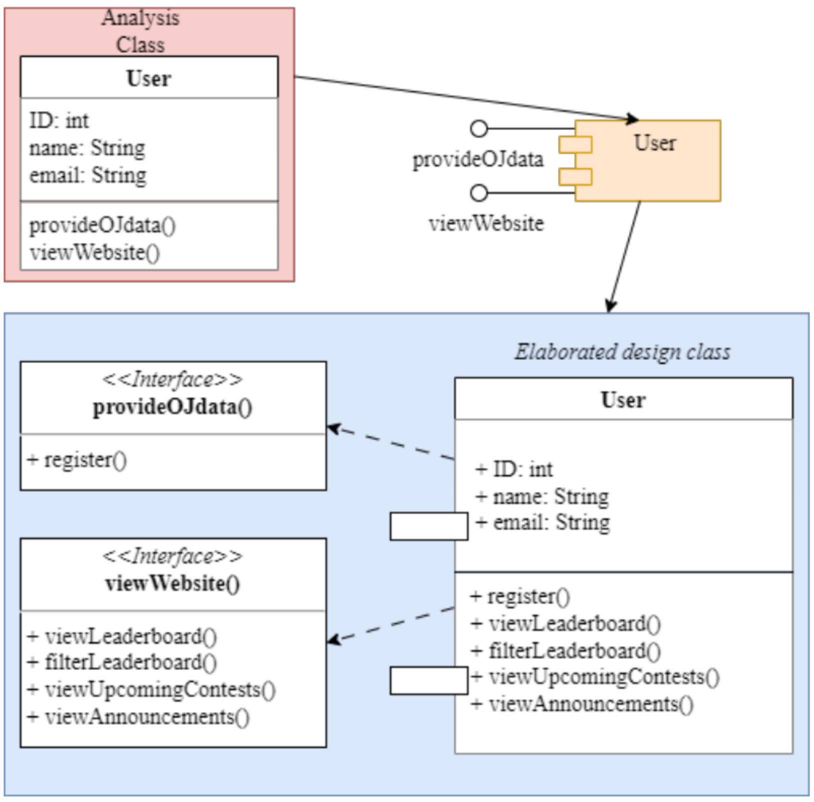


Figure 5.2: Component Design for User Management

5.3 Admin Management Component

The Admin Management part in the CUET CP Community System is like the control centre for important tasks. An admin in the CUET CP Community System can not only create new user accounts but also update user information as needed. Admins can easily share announcements, and check user information. This capability allows administrators to manage and keep user details accurate and up-to-date. With the help of [?] we have drawn the component design in figure 5.3:

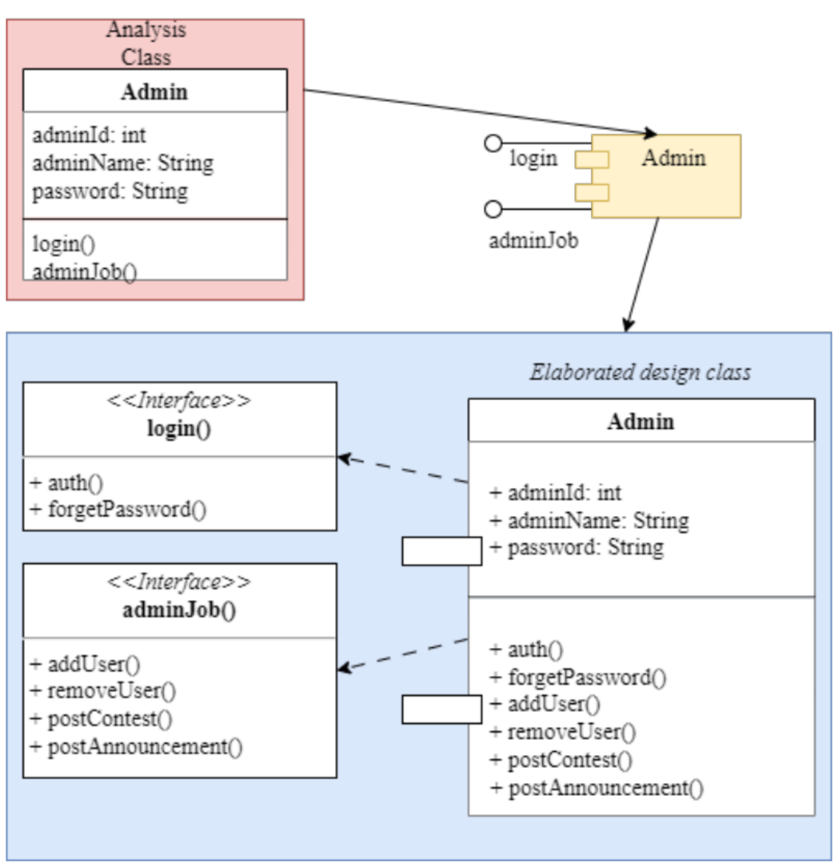
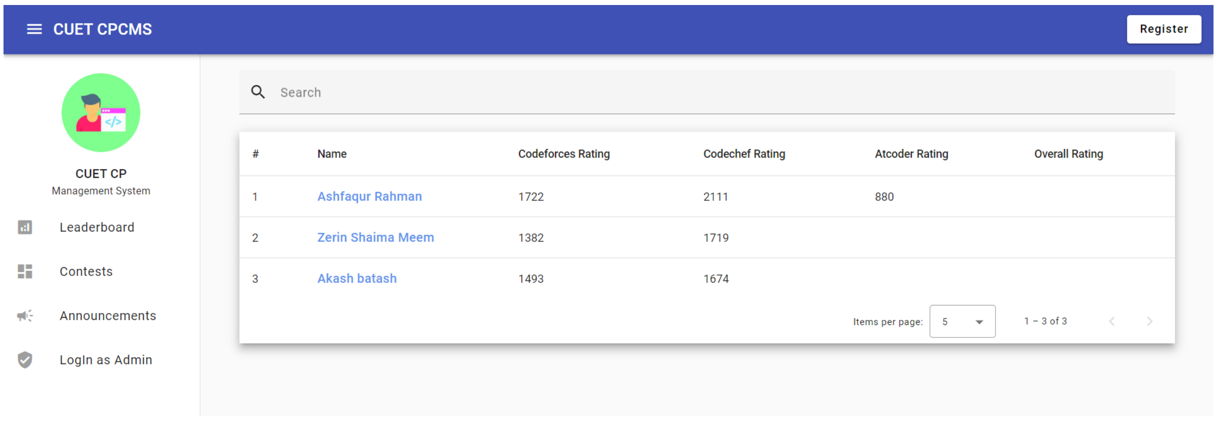


Figure 5.3: Component Design for Admin Management

5.4 GUI (Graphical User Interface) design

Leaderboard (User View)

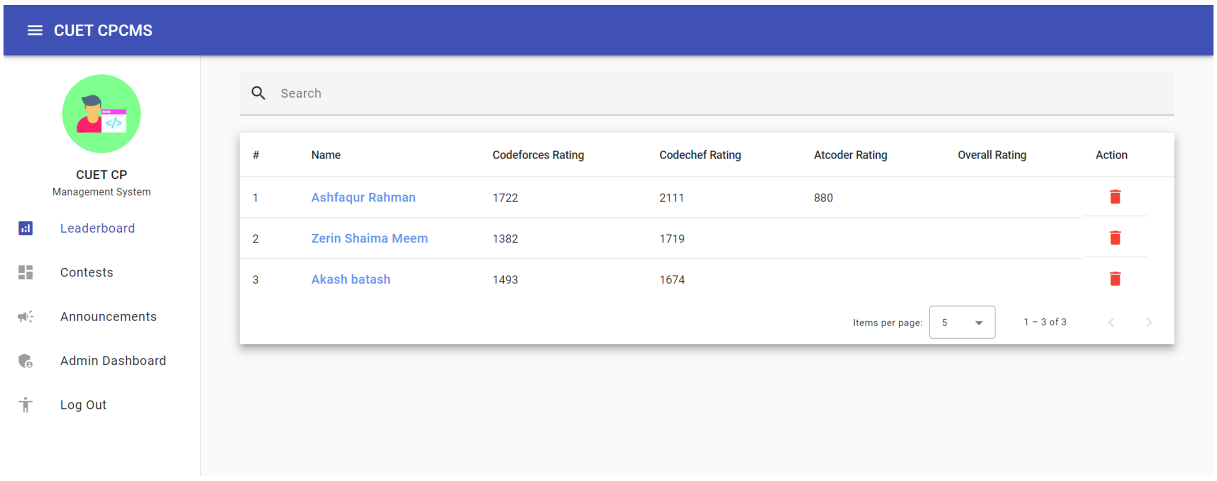


#	Name	Codeforces Rating	Codechef Rating	Atcoder Rating	Overall Rating
1	Ashfaqur Rahman	1722	2111	880	
2	Zerin Shaima Meem	1382	1719		
3	Akash batash	1493	1674		

Items per page: 5 1 – 3 of 3

Figure 5.4: Leaderboard (User View)

Leaderboard (Admin View)

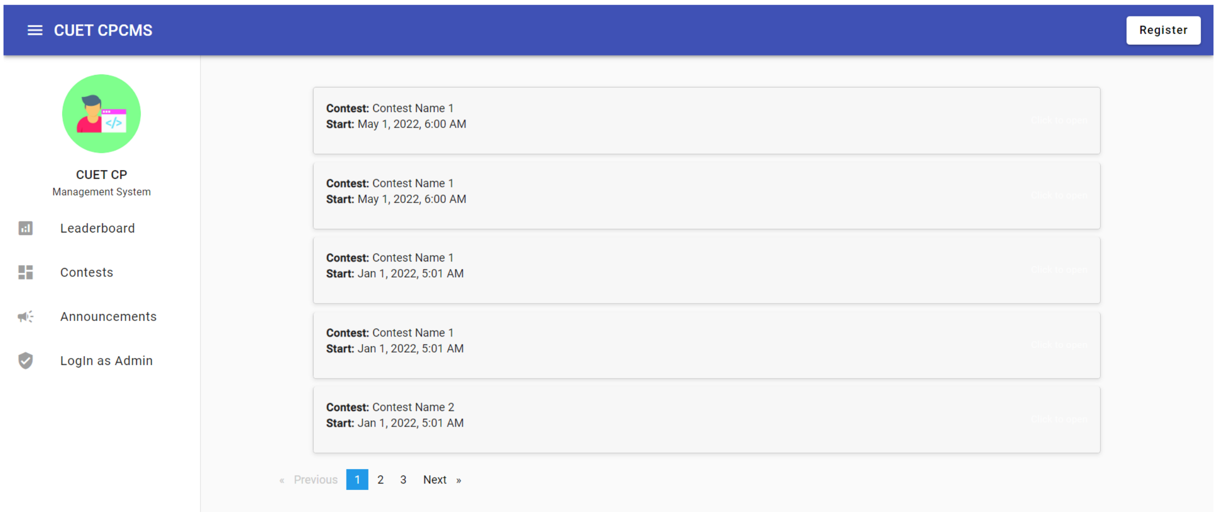


#	Name	Codeforces Rating	Codechef Rating	Atcoder Rating	Overall Rating	Action
1	Ashfaqur Rahman	1722	2111	880		🗑
2	Zerin Shaima Meem	1382	1719			🗑
3	Akash batash	1493	1674			🗑

Items per page: 5 1 – 3 of 3

Figure 5.5: Leaderboard (Admin View)

Contests (User View)


CUET CPCMS

Register

CUET CP
Management System

Leaderboard

Contests

Announcements

LogIn as Admin

Contest: Contest Name 1
Start: May 1, 2022, 6:00 AM

Contest: Contest Name 1
Start: May 1, 2022, 6:00 AM

Contest: Contest Name 1
Start: Jan 1, 2022, 5:01 AM

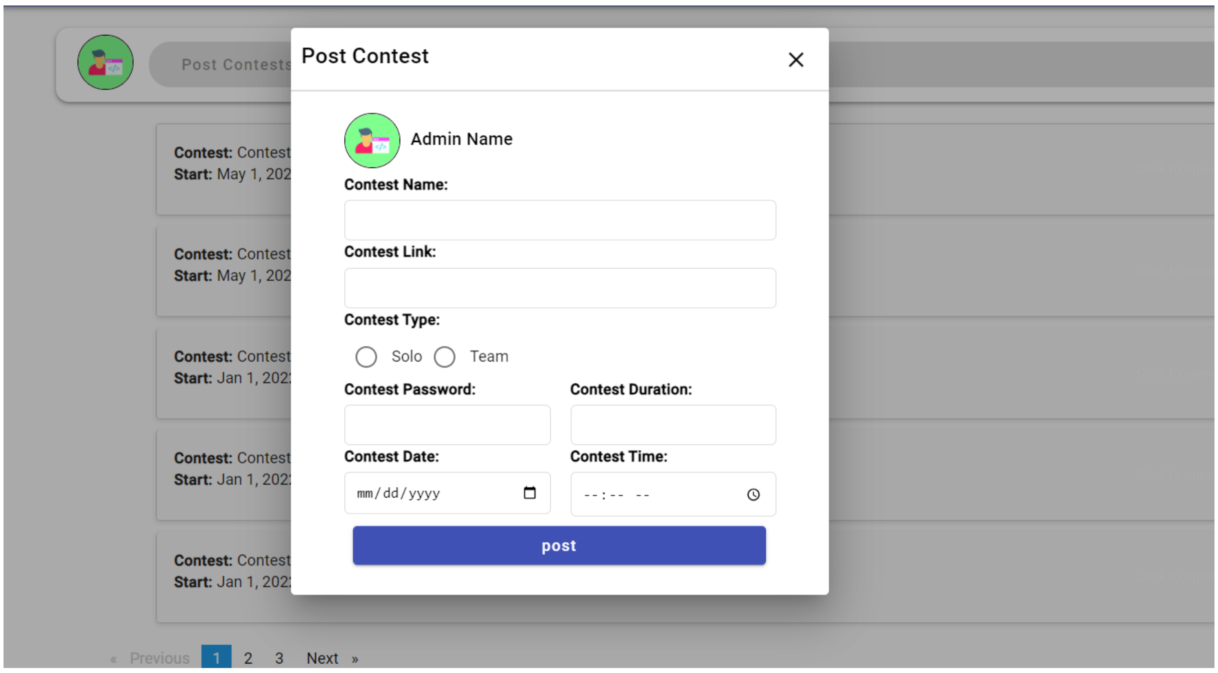
Contest: Contest Name 1
Start: Jan 1, 2022, 5:01 AM

Contest: Contest Name 2
Start: Jan 1, 2022, 5:01 AM

« Previous 1 2 3 Next »

Figure 5.6: Contests (User View)

Post Contest (Admin View)


Post Contests

Post Contest

Admin Name

Contest Name:

Contest Link:

Contest Type:

Solo Team

Contest Password:

Contest Duration:

Contest Date:

mm/dd/yyyy

Contest Time:

--:-- --

post

Contest: Contest
Start: May 1, 202

Contest: Contest
Start: May 1, 202

Contest: Contest
Start: Jan 1, 202

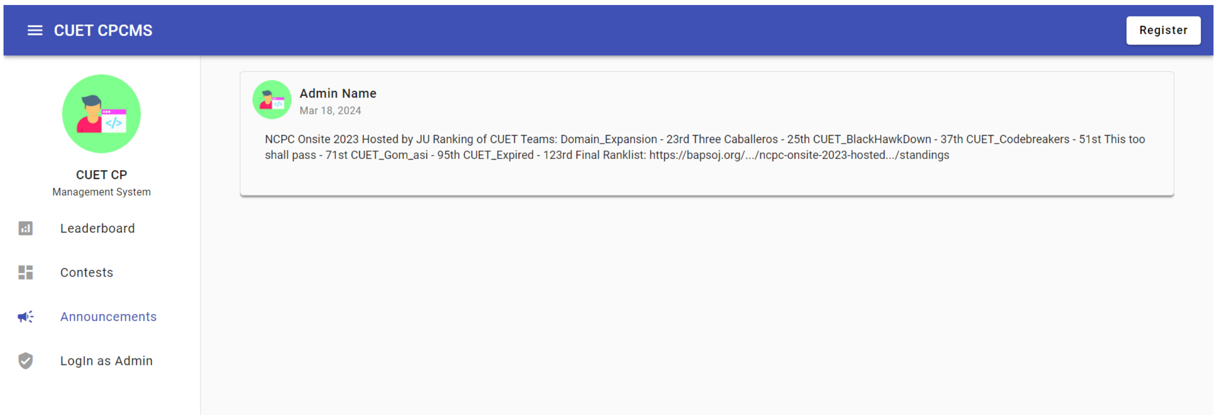
Contest: Contest
Start: Jan 1, 202

Contest: Contest
Start: Jan 1, 202

« Previous 1 2 3 Next »

Figure 5.7: Post Contest (Admin View)

Announcements (User View)

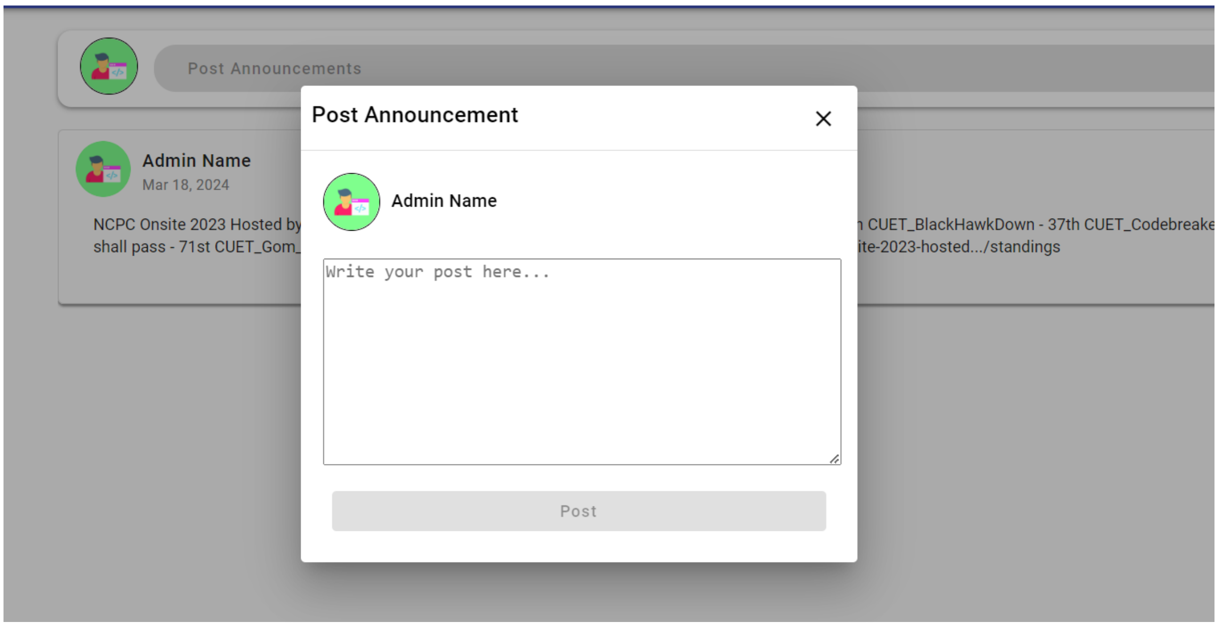


Admin Name
Mar 18, 2024

NCPC Onsite 2023 Hosted by JU Ranking of CUET Teams: Domain_Expansion - 23rd Three Caballeros - 25th CUET_BlackHawkDown - 37th CUET_Codebreakers - 51st This too shall pass - 71st CUET_Gom_asi - 95th CUET_Expired - 123rd Final Ranklist: https://bapsoj.org/.../ncpc-onsite-2023-hosted.../standings

Figure 5.8: Announcements (User View)

Post Announcement (Admin View)



Admin Name
Mar 18, 2024

NCPC Onsite 2023 Hosted by
shall pass - 71st CUET_Gom_
n CUET_BlackHawkDown - 37th CUET_Codebreake
ite-2023-hosted.../standings

Figure 5.9: Post Announcement (Admin View)

User Registration Request

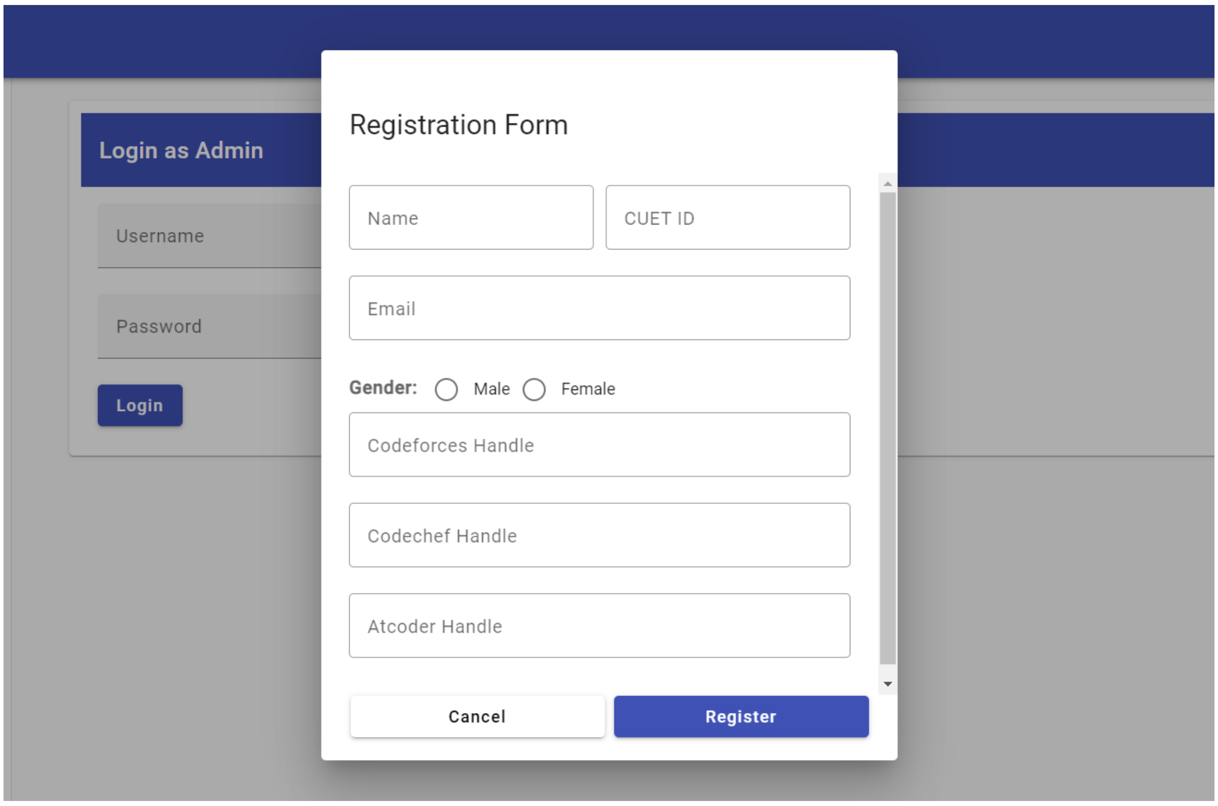


Figure 5.10: User Registration Request

Admin Login

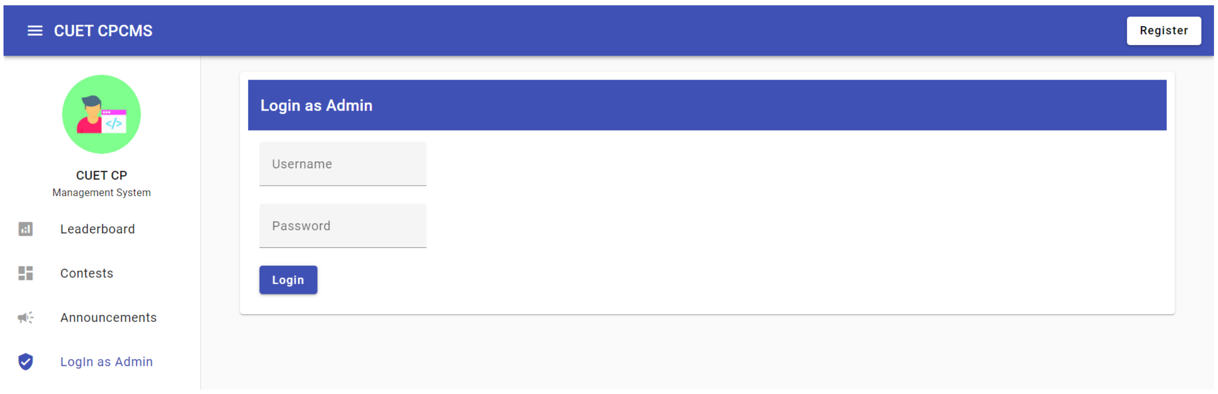
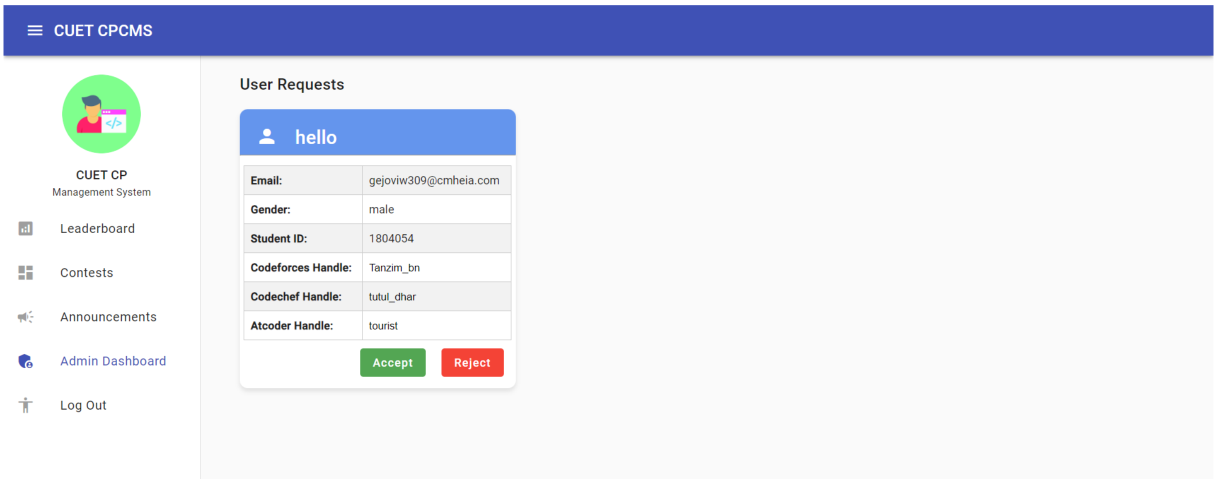


Figure 5.11: Admin Login

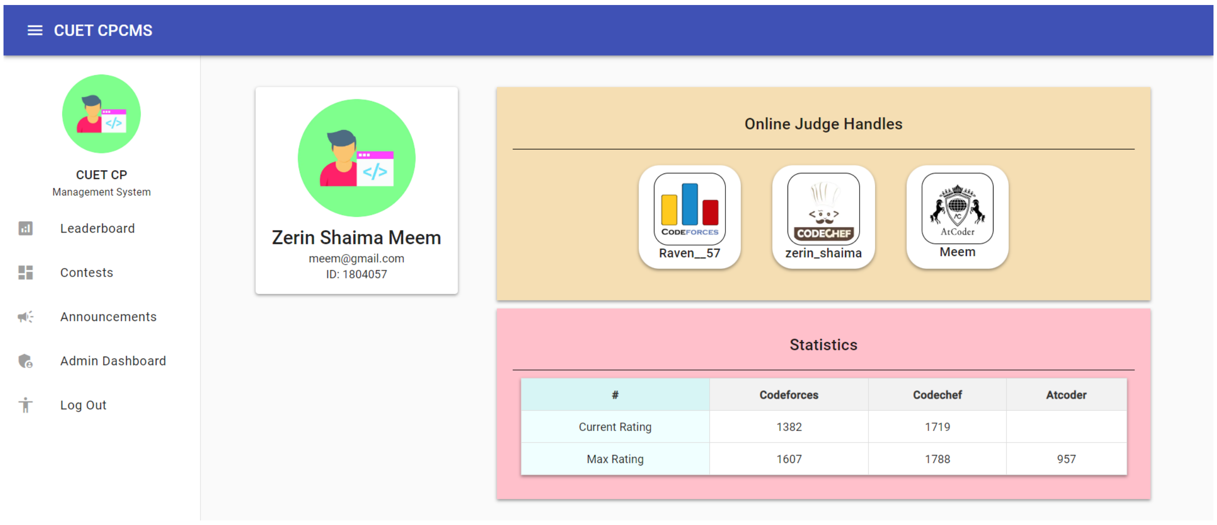
Admin Dashboard



Email:	gejoviw309@cmheia.com
Gender:	male
Student ID:	1804054
Codeforces Handle:	Tanzim_bn
Codechef Handle:	tutul_dhar
Atcoder Handle:	tourist

Figure 5.12: Admin Dashboard

User Profile



#	Codeforces	Codechef	Atcoder
Current Rating	1382	1719	
Max Rating	1607	1788	957

Figure 5.13: User Profile

5.5 API Documentation

POST/api/v1/login: Logs into the admin account.

Parameters:

- **name** (string, required): Name of the admin.
- **password** (string, required): The password of the admin.

Returns:

- If successful, returns a JSON object with the following properties:
  - **session** (object): The session information containing JWT token and others.
  - **user** (object): User information in detail.
- If unsuccessful, returns a JSON object with the following property:
  - **error** (string): The error message.

POST /api/v1/logout: Logs out of the admin account.

Returns:

- No content.

POST /api/v1/signup: Registers a user into the system.

Parameters:

- **name** (string, required): The name of the user.
- **email** (string, required): The email address of the user.

- **student**$_id(integer, required) : The student ID of the user.$ **Returns:**

- No content.

GET /api/v1/ranklist: Ranks the user according to their recent performance.

Returns:

- If successful, returns a JSON object with the following properties:
  - **Userlist** (array): The user object list with their ranking details. Each user object contains the following properties:
    - **Name** (string): The name of the user.
    - **OnlineJudgeIds** (array): The user IDs of various platforms.

        * **Rank** (integer): The position in the list.
        * **SolveCount** (integer): The number of solved problems.
        * **Penalty** (integer): The total penalty of the user.

- If unsuccessful, returns a JSON object with the following property:
    - **Error** (string): The error message.

GET /api/v1/user: Fetches the user account information.

Returns:

- If successful, returns a JSON object with the following properties:
    - **name** (string): The name of the user.
    - **studentId** (integer): The student ID of the user.
    - **onlineJudgeHandles** (array): The user IDs on different platforms.
    - **ratingList** (array): The user rating on different platforms.
    - **solveList** (array): The user solve count on different platforms.
    - **recentSubmissions** (array): List of links of last submissions of the user.
- If unsuccessful, returns a JSON object with the following property:
    - **Error** (string): The error message.

PUT /api/v1/user/userId: Updates user information.

Parameters:

- **userId** (string, path, required): The unique identifier of the user.

Returns:

- If successful, returns a JSON object with the updated user information.
- If unsuccessful, returns a JSON object with the following property:
    - **Error** (string): The error message.

DELETE /api/v1/user/userId: Deletes a user.

Parameters:

- **userId** (string, path, required): The unique identifier of the user.

Returns:

- If successful, returns a JSON object with a success message.

- If unsuccessful, returns a JSON object with the following property:
  - **Error** (string): The error message.

GET /api/v1/contest: Fetches the list of upcoming contests.

Returns:

- If successful, returns a JSON object with the following properties:
  - **contestList** (array): The list of upcoming contests. Each contest object contains details such as:
    * **contestId** (string): Unique identifier for the contest.
    * **name** (string): The name of the contest.
    * **startTime** (string): The start time of the contest.
    * **endTime** (string): The end time of the contest.
    * Additional contest details.
- If unsuccessful, returns a JSON object with the following property:
  - **Error** (string): The error message.

GET /api/v1/announcement: Fetches the list of active announcements.

Returns:

- If successful, returns a JSON object with the following properties:
  - **announcementList** (array): The list of upcoming contests. Each contest object contains details such as:
    * **announcementId** (string): Unique identifier for the announcement.
    * **title** (string): The title of the announcement.
    * **content** (string): The content or details of the announcement.
    * **timestamp** (string): The timestamp when the announcement was made. Additional announcement details.
  - Additional contest details.
- If unsuccessful, returns a JSON object with the following property:
  - **Error** (string): The error message.

Testing and sustainability plan

6.1 Requirements/specifications-based system level test cases

The CUET CP Community Management System web application project aims to provide a stable environment for users to view leaderboard, upcoming contests and announcements. Testing stages include unit, integration, system, and acceptance testing to ensure flawless functionality, user authentication, usability, performance, and security. A clear table of system-level test cases confirms system reliability and adherence to criteria.

Test Case ID	Description	Expected Outcome	Must/ Optional	Remarks
TC001	User Login with Valid Credentials	Successful login with valid credentials	Must	
TC002	User Login with Invalid Credentials	Display appropriate error message for invalid login	Must	
TC003	User Logout	Logout the user and redirect to the login page	Must	
TC004	User Signup with Valid Information	New user account is created successfully	Must	
TC005	User Signup with Invalid Information	Display appropriate error messages for validation	Must	
TC006	Fetch User Account Information	Retrieve user details correctly	Must	
TC007	Update User Information	User information is updated successfully	Must	
TC008	Delete User Account	User account is deleted successfully	Must	
TC009	Fetch List of Upcoming Contests	Retrieve upcoming contests correctly	Must	
TC010	Update Contest Information	Contest information is updated successfully	Must	
TC011	Delete Contest	Contest is deleted successfully	Must	
TC012	Fetch List of Active Announcements	Retrieve active announcements correctly	Must	
TC013	Update Announcement Information	Announcement information is updated successfully	Must	
TC014	Delete Announcement	Announcement is deleted successfully	Must	
TC015	Fetch User Ranklist	Retrieve user ranks correctly	Must	
TC016	Fetch User Recent Submissions	Retrieve user's recent submissions correctly	Must	

Table 6.1: Requirements/specifications-based system level test cases

6.2 Test Generation Techniques

In developing our Competitive Programming Management System (CPMS), we employ two key methodologies for test case generation:

White Box Testing Technique: Statement Coverage

Statement coverage, a form of white box testing, ensures that each statement in the source code is executed at least once during testing. By analyzing the control flow graph, we identify the necessary test cases to achieve full statement coverage. This technique allows us to assess the logical complexity of the code and ensure comprehensive testing of all code paths.

Black Box Testing Technique: Boundary Value Analysis

Boundary value analysis, a black box testing technique, focuses on testing input boundaries to uncover potential errors. By identifying boundary values for input variables, we generate test cases that explore both valid and invalid input conditions. This method helps validate the robustness and reliability of the CPMS by testing its response to boundary conditions.

6.3 Evaluation of Test Suite Quality

The testing phase of our CPMS involves a rigorous evaluation of the test suite to ensure its effectiveness and reliability. By combining both white box and black box testing techniques, we aim to achieve comprehensive test coverage and uncover potential defects in the system.

White box testing provides insight into the internal logic of the CPMS, allowing us to assess code complexity and identify critical paths for testing. By achieving statement coverage, we ensure that every line of code is executed and tested thoroughly.

Complementing white box testing, black box testing focuses on the external behavior of the CPMS, ensuring that it meets functional requirements and behaves as expected from a user's perspective. Boundary value analysis helps uncover edge cases and ensures that the CPMS can handle input variations effectively.

By leveraging both testing methodologies, we can evaluate the CPMS from multiple perspectives, enhancing the overall quality and reliability of the test suite.

6.4 Sustainability Plan

The sustainability plan for our CPMS prioritizes long-term viability, adaptability, and scalability to meet the evolving needs of competitive programming communities.

6.4.1 Scalability

To ensure scalability, we design the CPMS with a modular architecture that allows for easy expansion and integration of new features. By leveraging cloud-based infrastructure and scalable technologies, such as microservices and containerization, we can accommodate growing user bases and increasing demand without compromising performance or reliability.

6.4.2 Flexibility / Customization

Flexibility and customization are integral to the CPMS, allowing users to tailor the platform to their specific needs and preferences. Through configurable settings, customizable dashboards, and plugin support, users can adapt the CPMS to suit their unique workflows and requirements. Additionally, an open-source approach fosters collaboration and community-driven innovation, empowering users to contribute to the platform's development and customization.

By prioritizing scalability, flexibility, and customization, our CPMS is poised to support the evolving needs of competitive programming communities worldwide, ensuring its sustainability and long-term success.

Acknowledgement

We would like to earnestly acknowledge the sincere efforts and valuable time given by our course teachers:

1. Mir Md. Saki Kowsar
   Assistant Professor
   Dept. of CSE, CUET

2. Moumita Sen Sarma
   Lecturer
   Dept. of CSE, CUET

Their valuable guidance and feedback have helped us in completing this project.

References

1. Pressman, Roger and Maxim, Bruce. *Software Engineering: A Practitioner's Approach, 9th Edition.*
2. GeeksforGeeks. *https://www.geeksforgeeks.org/*
3. Tutorialspoint. *https://www.tutorialspoint.com/index.htm*
4. Diagrams.net. *https://app.diagrams.net/*
5. Swagger. *https://swagger.io/*
6. Soft Robotics. *https://www.softrobotics.com.bd/*