

# Coding Contest Arena

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

TO

**RGUKT- SRIKAKULAM**

*for the award of the degree of*

**Bachelor of Technology in Computer Science and Engineering by**

**G.Poojith(S191104)**

tnpc@gmail.com

**P.Swapna(S190403)**

**M.Sindu(S190566)**





**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE  
TECHNOLOGIES - SRIKAKULAM CAMPUS  
ANDHRA PRADESH - ETCHERLA**

**MAY 2025**

## THESIS CERTIFICATE

This is to certify that the report entitled “**CODING CONTEST ARENA**” submitted by **G.Poojith**, bearing ID-No: **S191104**, **P.Swapna**, bearing ID-No: **S190403**, **M.Sindu**, bearing ID-bearing ID-No: **S190566** to the Department of Computer Science & Engineering, Rajiv Gandhi University of Knowledge Technologies - Srikakulam, AP for the award of the degree of Bachelor of Technology in Computer Science & Engineering is a bonafide record of work carried out by them under my supervision and guidance. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**ch.Lakshmi Bala mam,**  
Project Internal Guide,  
head of Department  
( CSE), RGUKT,  
SRIKAKULAM.

## DECLARATION

K.Raghavendara, B.Murari ,V.Siva Prasad hereby  
declare that this report entitled “**CODING CONTEST ARENA**” submitted by us  
under the guidance and supervision of **ch.Lakshmi Bala mam** bonafide  
work. I also declare that it has not been submitted previously in part or in full to  
this University or other University or Institution for the award of any degree or  
diploma.

Date:

Place :

## **ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to **Ch.Lakshmi Bala madam** who has always been a constant motivation and guiding factor throughout the project time. It has been a great pleasure for us to get an opportunity to work under his guidance and complete the project work successfully.

We wish to extend our sincere thanks to **Ch.Lakshmi Bala madam** of the Computer Science and Engineering Department, for his constant encouragement throughout the project. We are also grateful to other members of the department without their support our work would have not been carried out so successfully.

At the outset, I would like to thank **Rajiv Gandhi University of Knowledge Technologies, Srikakulam**, for providing all the necessary resources for the successful completion of the course work. At last, but not the least I thank one and all who have rendered help to me directly or indirectly in the completion of my thesis work.

**With Sincere Regards,**

G.Poojith,  
P.Swapna ,  
M.sindu .

# Abstract

The "Coding Contest Arena" is a web-based application for coding competitions, featuring contest creation, user registration, and secure code execution. Contest organisers manage contests, define problems, and schedule timings. Participants register, submit code, and get feedback. The backend handles authentication, contest management, and code evaluation, with HTTPS and password hashing for security. Built with React for the frontend and NodeJs with GraphQL as backend, it ensures responsiveness. Data is securely stored in mongoDB database set up in cloud. Implemented tab switch detection, full screen mode and restricting copy pasting code to ensure fairness and prevent cheating during contests. Implemented mailing feature to send emails for reminding participants about their registered contests.

*Keywords:* CodingContest, Security, Programming Challenges, Test Cases, Authentication, UserRegistration, Code Evaluation, ReactJs, NodeJs, GraphQL API, Notifications.

# Table of Contents

ABSTRACT.....	vi
<b>1.INTRODUCTION.....</b>	<b>i</b>
	<b>x</b>
1.1. INTRODUCTION.....	
1.2. APPLICATIONS.....	ix
1.3. MOTIVATION TOWARDS PROJECT.....	ix
1.3.1 INCREASING PREVALENCE OF ARENA.....	
<b>2.Proposed System Analysis &amp; Design.....</b>	
2.1 PROBLEM STATEMENT.....	xiii
2.2. ANALYSIS.....	xiii
2.2.1 FEASIBILITY STUDY.....	
2.2.2 REQUIREMENT ANALYSIS.....	
2.3 BASIC IDEA .....	
2.4 TECHNOLOGIES USED .....	
2.5 EXPLAIN ABOUT YOUR PROJECT.....	
<b>3.TECHNOLOGY IMPLEMENTATION &amp; TESTING.....</b>	<b>XV</b>
3.1 REACT OVERVIEW.....	
3.2 NODE.JS OVERVIEW.....	XVI
3.3 GIT OVERVIEW.....	XVIII
3.4 GITHUB OVERVIEW.....	
3.5 EXPRESS.JS OVERVIEW.....	
3.6 MONGODB OVERVIEW.....	
3.7 GRAPHQL OVERVIEW.....	
<b>4.OUTPUT SNAPSHOTS.....</b>	<b>xx</b>
4.1 HOME PAGE.....	
4.2 LOGIN PAGE.....	
4.3 CODING PLAY GROUND PAGE.....	

4.4 CONTEST PAGE.....	
4.5 PROBLEM PAGE.....	
4.6 USER DASHBOARD PAGE.....	
4.7 PROFILE EDIT PAGE.....	
4.8 CONTEST CREATION PAGE.....	
4.9 ADDING QUESTION PAGE.....	
4.10 UPSOLVING PAGE.....	
4.11 TAB SWITCHING PAGE.....	
4.12 COPY PASTING CODE PAGE.....	
4.13 CONTEST CREATION PAGE.....	
<b>5. CONCLUSION.....</b>	
5.1 CONCLUSION.....	
<b>6. REFERENCES.....</b>	<b>xxxvii</b>



# Chapter 1

## Introduction

### 1.1 Introduction to Your Project

The "Coding Contest Arena" is a user-friendly web application that simplifies running coding competitions for organizers and participants. It includes features like setting up contests, registering users, automatic scoring, and hosting online contests. To keep the contests fair, it blocks tab switching and copying and pasting. Organizers can manage contests and set the rules, while participants get a secure and responsive platform that supports multiple programming languages. Additionally, it offers updates and leaderboards, making the competition more engaging. With its intuitive interface and robust functionality, "Coding Contest Arena" is the perfect solution for hosting safe, exciting, and well-organized coding contests.

### 1.2 Applications

- **Educational Institutions:** Universities and schools can use the platform to organize coding competitions, hackathons, and programming challenges for students, enhancing their coding skills and encouraging healthy competition.
- **Corporate Training:** Companies can utilize the platform for internal coding contests to identify and nurture talent, provide training, and prepare employees for technical roles or projects.

- **Online Coding Communities:** Programming communities and forums can host regular contests to engage their members, encourage learning, and build a sense of community.
- **Recruitment and Hiring:** Organizations can use coding contests as a tool for assessing the skills of potential hires in a practical, real-world setting.
- **Skill Development Platforms:** Websites and platforms focused on teaching coding can use the arena to provide hands-on practice, track progress, and motivate learners through competition.
- **Event Organizers:** Individuals or groups organizing local or international coding events can leverage the platform for a seamless and professional contest experience.

## 1.3 Motivation Towards our Project

Our project, "Coding Contest Arena," aims to make organizing and participating in coding competitions easy and enjoyable. We want to create a platform where organizers can set up contests effortlessly, and participants can compete in a fair and secure environment. By preventing cheating and supporting multiple programming languages, we ensure a level playing field for everyone.

### 1.3.1 Increasing Prevalence of Arena

- **Skill Improvement:** Coding contests help participants enhance their programming skills through practice and competition.
- **Talent Identification:** Companies and organizations use contests to find and recruit skilled developers.
- **Educational Benefits:** Schools and universities integrate coding contests to challenge students and improve their learning experience.

# Chapter -2

## Proposed System Analysis & Design

### 2.1 problem statement

With the growing popularity of coding contests among students, professionals, and organizations, there is a significant demand for platforms that can effectively host these events. These platforms must ensure a fair and secure environment for all participants while providing a seamless user experience. Current platforms often lack essential features like strong security to prevent cheating, support for multiple programming languages, and easy management tools for organizers. This creates challenges in organizing and participating in contests. Therefore, a dedicated solution is needed to make coding contests accessible, secure, and enjoyable for everyone involved.

### 2.2 Analysis

- **Scalability and Flexibility:** MongoDB provides a scalable and flexible database solution, accommodating varying data structures and high-volume data storage for contest details, user registrations, and submissions.
- **Server-Side Logic and API Development:** Express.js simplifies the creation of robust APIs, facilitating seamless integration between the front-end React components and the back-end services. This enables efficient data retrieval, updates, and real-time interaction during contests.
- **Interactive User Interface:** React enhances user experience with its component-based architecture, ensuring responsive and dynamic interfaces for participants to view contest details, submit solutions, and track progress in real-time.
- **Real-Time Updates and Notifications:** Node.js supports asynchronous event-driven architecture, crucial for delivering real-time updates, notifications, and scoring updates during contests, enhancing participant engagement and competitiveness.
- **Secure and Efficient:** The MERN stack's emphasis on security best practices ensures secure data handling, user authentication, and access control, safeguarding participant submissions and contest integrity.
- **Community and Collaboration:** Features such as leaderboards, participant forums, and integrated communication tools foster a sense of community among contestants, promoting collaboration, learning, and friendly competition.

## **2.2.1 Feasibility Study**

Feasibility means whether some idea will work or not. In other work, feasibility study involves an examination of the operations. A project feasibility study is an exercise that involves documenting each of the potential solutions to a particular business problem or opportunity. Feasibility studies can be undertaken by any type of business, project or team and as a critical part of the project life cycle. A procedure that identifies, describes, and evaluates candidate systems and selects the best system for the job is called as Feasibility study.

Three key considerations are involved in the feasibility analysis:

### **1. Technical Feasibility:-**

The use of React, HTML, CSS, Java Script makes form design easy and convenient. The project can be run on any system with minimum requirements. It reduces data entry errors because of applying validation in most of all the forms, it can be easily handled by any user, and it also helps in faster data updations. Also the project though developed in latest GUI, it is very easy to operate. Hence the project is technically feasible.

### **2. Economic Feasibility:-**

Cost benefit analysis is very important in deciding whether the project is economically feasible or not. It is alone sufficient to save our time and money. It is one time investment and does not require regular maintenance. Through cost benefit analysis it was concluded that the benefits outweigh costs and thus the project is economically feasible.

### **3. Behavioral Feasibility:-**

Behavioral feasibility determines how much effort will go into educating, selling and training the users on a candidate system. As everyone now-a-days are users of social Network it is very easy to handle normally by anyone. The project was also evaluated to be behaviorally feasible as it is very user-friendly and hardly needs any extra efforts to educate user for its utility and functioning.

## **2.2.2 Requirement Analysis**

The "Coding Contest Arena" project, based on the MERN stack, aims to simplify the organization and participation in coding competitions. It will feature user-friendly interfaces for both students and teachers: students can register, submit code, view results, and engage in discussions, while teachers can manage contests, configure parameters, and assess student performance. Key dependencies include MongoDB for data storage, Express.js for robust API development, React for responsive front-end interfaces, and Node.js for server-side logic. Integration of compilers for C, C++, and Java ensures seamless code execution, while security measures and scalability are prioritized to support a reliable and engaging contest environment.

## **2.3 Basic Idea**

The "Coding Contest Arena" is a versatile platform designed for hosting coding competitions supporting a wide array of programming languages. It serves both students and teachers with intuitive interfaces tailored for efficient contest management and participation. Teachers can evaluate student submissions in real-time, upload practical aims, and provide descriptions specific to their classes. Students register with their academic details to access contests where they write, debug, and submit code online. Real-time error checking ensures smooth interaction, supported by integrated compilers for secure code execution across diverse programming languages. This MERN stack-based platform fosters a collaborative and engaging environment, enhancing learning and skill development through interactive contest experiences.

## 2.4 Technologies Used

**MongoDB:** A NoSQL database used for storing contest details, user information, submissions, and scoring data. MongoDB's flexibility allows for efficient data handling and scalability.

**Express.js:** A web application framework for Node.js that facilitates the development of robust APIs. Express.js is crucial for handling HTTP requests, routing, middleware integration, and interacting with the MongoDB database.

**React:** A JavaScript library for building user interfaces. React enables us to create interactive and responsive front-end components, providing participants and organizers with a seamless user experience.

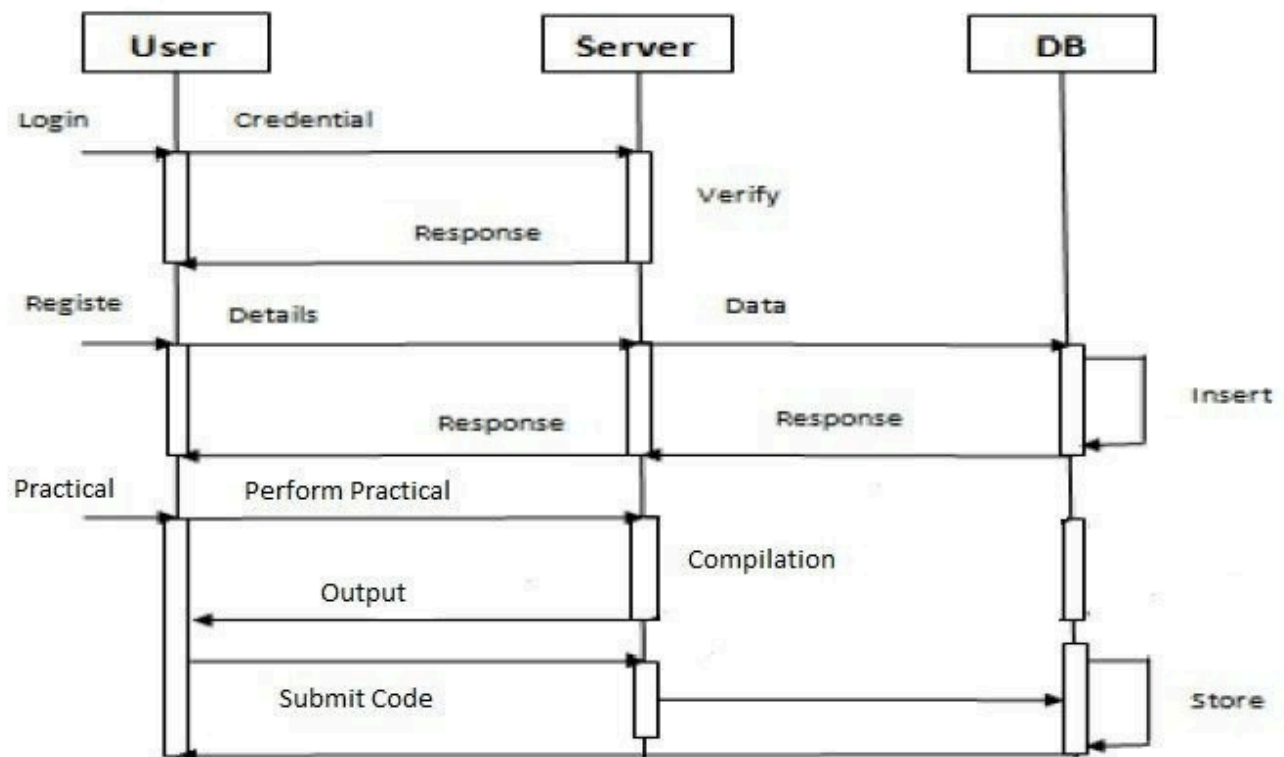
**Node.js:** A server-side JavaScript runtime environment that executes JavaScript code outside of a web browser. Node.js powers the back-end of our application, handling server-side logic, data processing, and integration with external services.

## 2.5 Explain About Your Project

"Coding Contest Arena" is a streamlined web platform for organizing and participating in coding competitions, ensuring fair play, supporting multiple languages, and enhancing community engagement among coding enthusiasts.

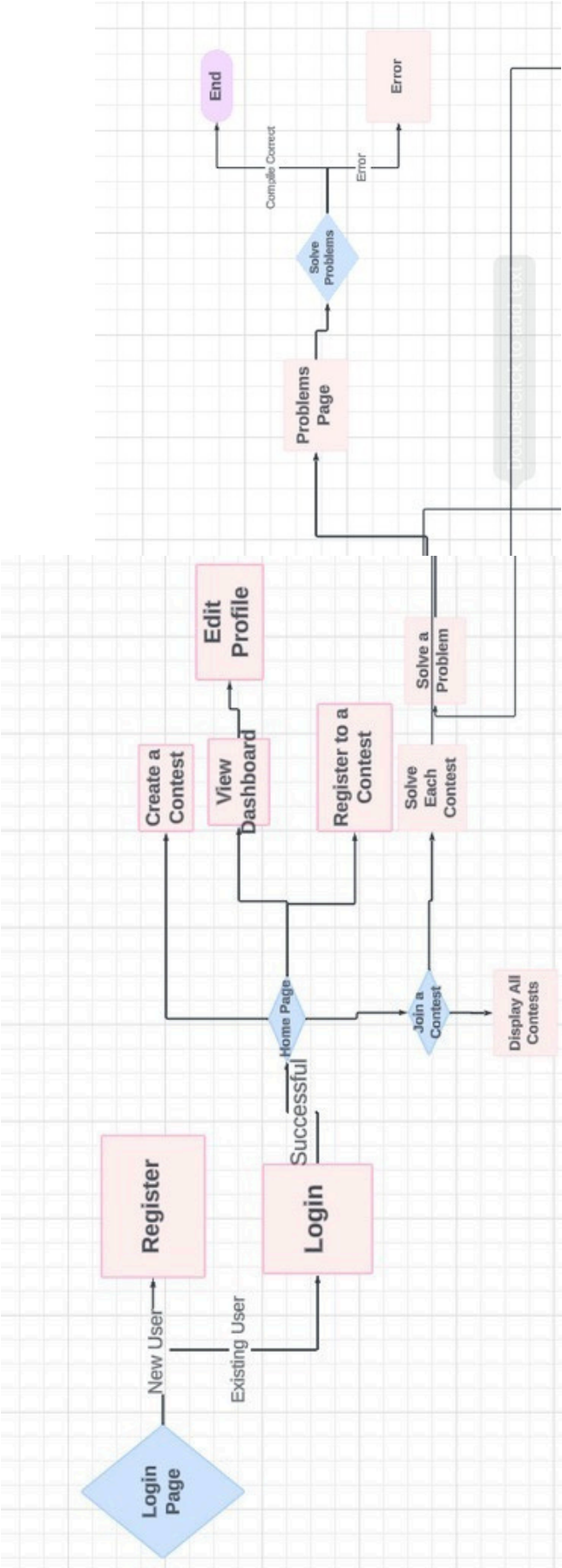
## UML Diagrams

### Sequence Diagram

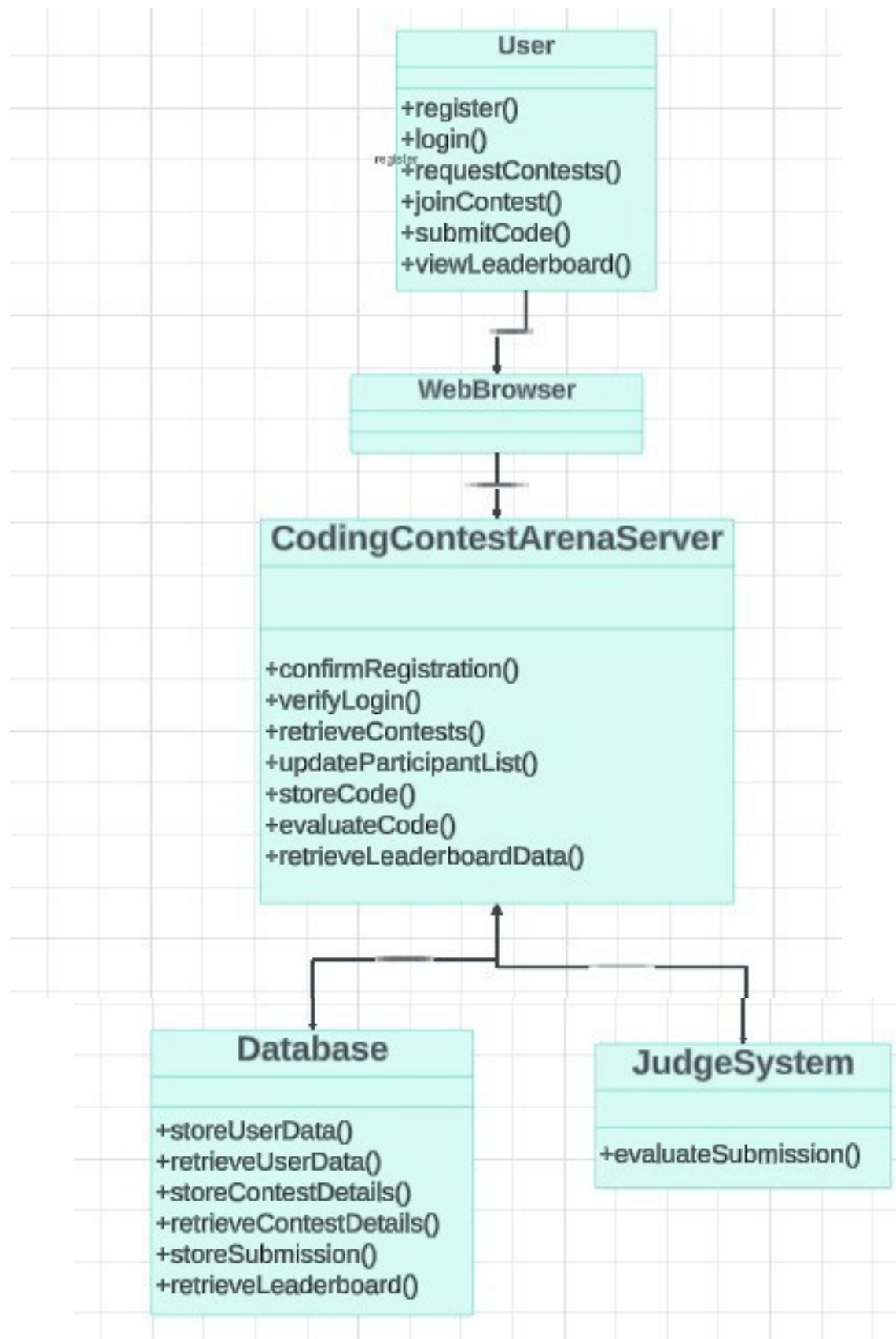


sequence diagram of participant

# Acticity Diagram







# chapter-3

## Technology Implementation & Testing

### 3.1 React Overview

- React is a JavaScript library for building user interfaces, developed by Facebook. It has become immensely popular for its component-based architecture and efficient rendering using a virtual DOM (Document Object Model).
- **Key Features and Concepts:**
- **Component-Based Architecture:**
  - React applications are built using components, which are self-contained, reusable pieces of UI. Components can be nested within each other, allowing for a modular and maintainable codebase.
- **Virtual DOM:**
  - React uses a virtual DOM to efficiently update the UI. When data changes, React compares the virtual DOM with the actual DOM and only updates the parts that have changed, minimizing browser reflows and improving performance.
- **JSX (JavaScript XML):**
  - JSX is a syntax extension for JavaScript that allows developers to write HTML-like code within JavaScript. It makes React component code more readable and intuitive, blending UI markup and logic together.
- **Unidirectional Data Flow:**
  - React follows a unidirectional data flow, where data flows in one direction from parent to child components via props. This ensures predictable state management and facilitates debugging.
- **Component Lifecycle Methods:**
  - React components have lifecycle methods that allow developers to hook into specific points in a component's lifecycle, such as mounting, updating, and unmounting. This enables developers to perform actions like fetching data or cleaning up resources at the appropriate times.
- **State Management:**
  - React components can have state, which represents the local state of the component. When the state changes, React automatically re-renders the component and its children to reflect the updated state, maintaining UI consistency.
- **Reusable Components and Composition:**
  - React promotes component reusability and composability. Developers can create complex UIs by composing smaller, reusable components together, encapsulating functionality and enhancing code maintainability.
- **React Hooks:**
  - Introduced in React 16.8, hooks are functions that allow functional components to use state and other React features without writing a class. Hooks like `useState` and `useEffect` simplify state management and side effects in functional components.

- **Advantages of React:**

- **Declarative Syntax:** With JSX, React makes it easier to describe how the UI should look, abstracting away DOM manipulation complexities.
- **Component Reusability:** Components can be reused across different parts of the application, promoting code reusability and reducing redundancy.
- **Performance:** React's virtual DOM and efficient diffing algorithm minimize DOM updates, resulting in faster rendering and improved application performance.
- **Community and Ecosystem:** React has a large and active community, with extensive documentation, libraries (like React Router for routing and Redux for state management), and tools (like React DevTools) to support development and debugging.
- React's strengths in component reusability, performance optimization, and developer experience make it a powerful choice for building modern, interactive web applications, including complex platforms like the "Coding Contest Arena" within the MERN stack.

## 3.2 Node.js Overview

- Node.js is a server-side JavaScript runtime environment built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code outside of a web browser, enabling server-side scripting and back-end development.
- Key Features and Concepts:
  - Asynchronous and Event-Driven:
    - Node.js operates on a non-blocking, event-driven architecture. It uses an event loop to handle multiple concurrent operations efficiently, making it suitable for applications that require high concurrency and real-time interaction.
  - Single-Threaded, Non-Blocking I/O:
    - Unlike traditional server-side platforms, Node.js uses a single-threaded event loop model for handling requests. This model allows Node.js to handle many connections simultaneously without blocking, optimizing resource utilization and improving scalability.
  - NPM (Node Package Manager):
    - NPM is the default package manager for Node.js, providing a repository of reusable packages and modules. It simplifies dependency management and facilitates the integration of third-party libraries and frameworks into Node.js applications.
  - Server-Side Development:
    - Node.js is widely used for developing server-side applications and APIs. It provides built-in modules (such as HTTP, HTTPS, and File System) and community-supported libraries (like Express.js) that streamline the development of web servers and APIs.
  - Cross-Platform Compatibility:
    - Node.js is designed to run on multiple platforms (Windows, macOS, Linux), offering flexibility and consistency in application deployment across different environments.

## 3.3 Git Overview

Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It allows multiple developers to work on a project simultaneously without overwriting each other's changes.

### **Key Features and Concepts:**

#### **Distributed Version Control:**

Every developer has a local copy of the entire project history, including all commits, branches, and tags. This makes Git highly resilient to data loss and allows for offline work.

#### **Commit and Log History:**

Git tracks changes to files through commits. Each commit is a snapshot of the project at a particular point in time, allowing developers to navigate and revert to previous states if necessary.

#### **Branching and Merging:**

Git supports multiple branches, enabling developers to work on different features or fixes concurrently. Branches can be merged back into the main branch once the work is complete, facilitating parallel development.

#### **Staging Area:**

The staging area (or index) allows developers to prepare commits by adding specific changes. This enables granular control over what changes are included in each commit.

#### **Conflict Resolution:**

Git helps manage merge conflicts by highlighting conflicting changes and providing tools for resolving them. This ensures that collaborative development progresses smoothly.

## 3.4 GitHub Overview

GitHub is a web-based platform built on top of Git, providing additional features for collaboration, project management, and social coding. It hosts Git repositories and offers tools for version control and collaborative development.

### **Key Features and Concepts:**

#### **Repositories:**

A repository (or repo) is a central location where the project's codebase, history, and metadata are stored. GitHub repositories can be public (open to everyone) or private (restricted access).

#### **Forking and Pull Requests:**

Forking allows users to create a personal copy of someone else's repository. Developers can make changes in their fork and propose those changes to the original repository via pull requests, facilitating collaboration.

#### **Issues and Project Management:**

GitHub provides issue tracking for reporting bugs, requesting features, and discussing project-related topics. Integrated project boards and milestones help manage development tasks and progress.

#### **Continuous Integration (CI) and Continuous Deployment (CD):**

GitHub integrates with CI/CD tools (like GitHub Actions, Jenkins, Travis CI) to automate testing, building, and deploying code, ensuring code quality and speeding up the development lifecycle.

#### **Collaborative Features:**

GitHub supports code reviews, comments, and discussions directly within pull requests, fostering a collaborative development environment. It also offers features like wikis and documentation hosting for project resources.

## Advantages of Git and GitHub:

**Version Control:** Git ensures a detailed history of changes, enabling easy tracking, reverting, and branching.

**Collaboration:** GitHub enhances collaboration with tools for code reviews, issue tracking, and pull requests.

**Community:** GitHub's vast user base and public repositories foster community contributions and open-source development.

**Integration:** GitHub's integration with CI/CD pipelines, third-party services, and development tools streamlines the development process.

**Security:** GitHub offers features like protected branches, code scanning, and secret management to maintain code integrity and security.

## 3.5 Express.js Overview

Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. It simplifies the development of server-side applications by providing a range of tools and utilities to manage HTTP requests, routing, middleware, and more.

### Key Features and Concepts:

#### Middleware:

Middleware functions in Express.js are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle.

Middleware can perform various tasks such as logging, authentication, parsing request bodies, and handling errors.

#### Routing:

Express.js provides a powerful routing mechanism that allows developers to define URL endpoints (routes) and specify how the application should respond to client requests for those endpoints.

Routes can handle different HTTP methods (GET, POST, PUT, DELETE, etc.) and can include route parameters and query strings.

#### Template Engines:

Express.js supports various template engines (such as Pug, EJS, and Handlebars) to generate dynamic HTML pages.

Template engines enable the separation of application logic and presentation, making it easier to manage and maintain the codebase.

### **Static File Serving:**

Express.js can serve static files (such as images, CSS files, and JavaScript files) using the built-in middleware `express.static`.

This is useful for delivering front-end assets directly from the server.

### **Error Handling:**

Express.js includes robust error handling capabilities. Developers can define custom error-handling middleware to manage errors consistently across the application.

This helps in logging errors, returning appropriate error messages to the client, and maintaining application stability.

### **Scalability and Extensibility:**

Express.js is designed to be unopinionated and minimal, allowing developers to add only the components and middleware they need.

Its modular nature makes it highly extensible and suitable for building both small and large-scale applications.

## **Advantages of Express.js:**

**Simplicity and Ease of Use:** Express.js offers a straightforward API and a minimal setup, making it easy for developers to get started and quickly build server-side applications.

**Performance:** Built on top of Node.js, Express.js benefits from the non-blocking, event-driven architecture of Node.js, enabling it to handle a large number of simultaneous connections efficiently.

**Flexibility:** Express.js does not impose any strict conventions or structures, giving developers the flexibility to structure their applications as they see fit. This makes it adaptable to a wide range of use cases and development styles.

**Middleware Ecosystem:** Express.js has a vast ecosystem of middleware, both built-in and third-party, that can be easily integrated to add functionality such as authentication, input validation, logging, and more.

**Community and Support:** As one of the most popular Node.js frameworks, Express.js has a large and active community. This ensures a wealth of resources, tutorials, and support options available for developers.

## Use Cases for Express.js:

RESTful APIs: Building APIs to serve data to client applications (web, mobile, IoT).

Single Page Applications (SPAs): Serving the back-end logic for SPAs built with front-end frameworks like React, Angular, or Vue.js.

Web Applications: Developing traditional multi-page web applications with server-side rendering.

Microservices: Creating lightweight, independent services in a microservices architecture.

## 3.6 MongoDB Overview

MongoDB is a NoSQL database that uses a document-oriented data model. It is designed for scalability, flexibility, and performance, making it suitable for a wide range of applications from small startups to large enterprises.

### Key Features and Concepts:

#### Document-Oriented Storage:

MongoDB stores data in JSON-like documents, which are more expressive and flexible than traditional row-column databases. Each document can have a different structure, allowing for dynamic schemas.

#### Collections:

Documents are grouped into collections. A collection is analogous to a table in relational databases, but unlike tables, collections do not enforce a fixed schema on the documents they store.

#### Dynamic Schemas:

MongoDB allows you to change the structure of documents over time without modifying the entire collection. This flexibility makes it easy to iterate on application features and adapt to changing requirements.

#### Scalability and High Availability:

- MongoDB supports horizontal scaling through sharding, distributing data across multiple servers to handle large volumes of data and high throughput.
- It also provides built-in replication with replica sets, ensuring high availability and data redundancy by automatically maintaining multiple copies of data across different servers.



## **Indexing:**

MongoDB supports various types of indexes, including single field, compound, geospatial, and text indexes. Indexes improve query performance by allowing the database to quickly locate relevant documents.

## **Aggregation Framework:**

The aggregation framework allows for advanced data processing and transformation operations, such as filtering, grouping, sorting, and computing aggregate values. This is similar to SQL's GROUP BY and JOIN operations but optimized for MongoDB's document model.

## **Ad-Hoc Queries:**

MongoDB supports rich, ad-hoc queries using a flexible query language. Queries can include field-value pairs, range queries, and regular expression searches.

## **Replication:**

Replica sets in MongoDB provide redundancy and high availability by replicating data across multiple servers. Replica sets can automatically failover to a secondary node if the primary node becomes unavailable.

## **Sharding:**

Sharding enables MongoDB to distribute data across multiple machines, allowing for horizontal scaling. It splits data into ranges (shards) and distributes them across shards to balance the load.

# **Advantages of MongoDB:**

**Flexibility:** MongoDB's schema-less nature allows for the storage of heterogeneous and evolving data structures without requiring schema migrations.

**Scalability:** Designed to handle large volumes of data, MongoDB scales horizontally through sharding and provides high availability with replica sets.

**Performance:** MongoDB offers high performance for both read and write operations, supported by its efficient indexing and memory management features.

## 3.7 GraphQL Overview

GraphQL is a query language for APIs and a runtime for executing those queries by using a type system you define for your data. Developed by Facebook in 2012 and released as an open-source project in 2015, GraphQL provides a more efficient, powerful, and flexible alternative to the traditional REST API.

### Key Features and Concepts:

#### **Declarative Data Fetching:**

With GraphQL, clients can specify exactly what data they need in a single query. This avoids over-fetching or under-fetching of data, which is common in REST APIs.

#### **Strongly Typed Schema:**

A GraphQL API is defined by a schema that specifies the types of data and the relationships between them. This schema acts as a contract between the client and the server.

#### **Single Endpoint:**

Unlike REST, which typically uses multiple endpoints for different resources, GraphQL uses a single endpoint to handle all requests. Clients can specify their requirements in the query.

#### **Nested Queries:**

GraphQL allows for nested queries, enabling clients to retrieve related data in a single request. This is particularly useful for fetching hierarchical or relational data.

#### **Real-Time Updates with Subscriptions:**

GraphQL supports real-time updates via subscriptions, allowing clients to subscribe to specific events or data changes and receive updates as they occur.

#### **Introspection:**

GraphQL APIs are self-documenting. The schema can be queried to understand the available types, queries, mutations, and subscriptions, making it easier for developers to explore and understand the API.

#### **Mutations:**

- While queries are used to fetch data, mutations are used to modify data. This distinction provides a clear separation between read and write operations.

#### **Resolvers:**

- Resolvers are functions that handle the fetching and manipulation of data for a particular field in the schema. They allow developers to define how data should be retrieved or modified.

# Advantages of GraphQL:

**Efficiency:** Clients receive exactly the data they request, reducing the amount of data transferred over the network and improving performance.

**Flexibility:** Clients can specify their data needs, allowing for more dynamic and adaptable applications.

**Consistency:** A single endpoint simplifies API management and reduces the complexity of client-server interactions.

**Developer Experience:** GraphQL's introspection and strong typing improve tooling and documentation, making it easier for developers to work with the API.

**Ecosystem:** GraphQL has a growing ecosystem of tools, libraries, and services that support various aspects of development, including client libraries (Apollo Client, Relay), server libraries (Apollo Server, Express-GraphQL), and development tools (GraphiQL, GraphQL Playground).

# Chapter 4

## 4.1 Home Page

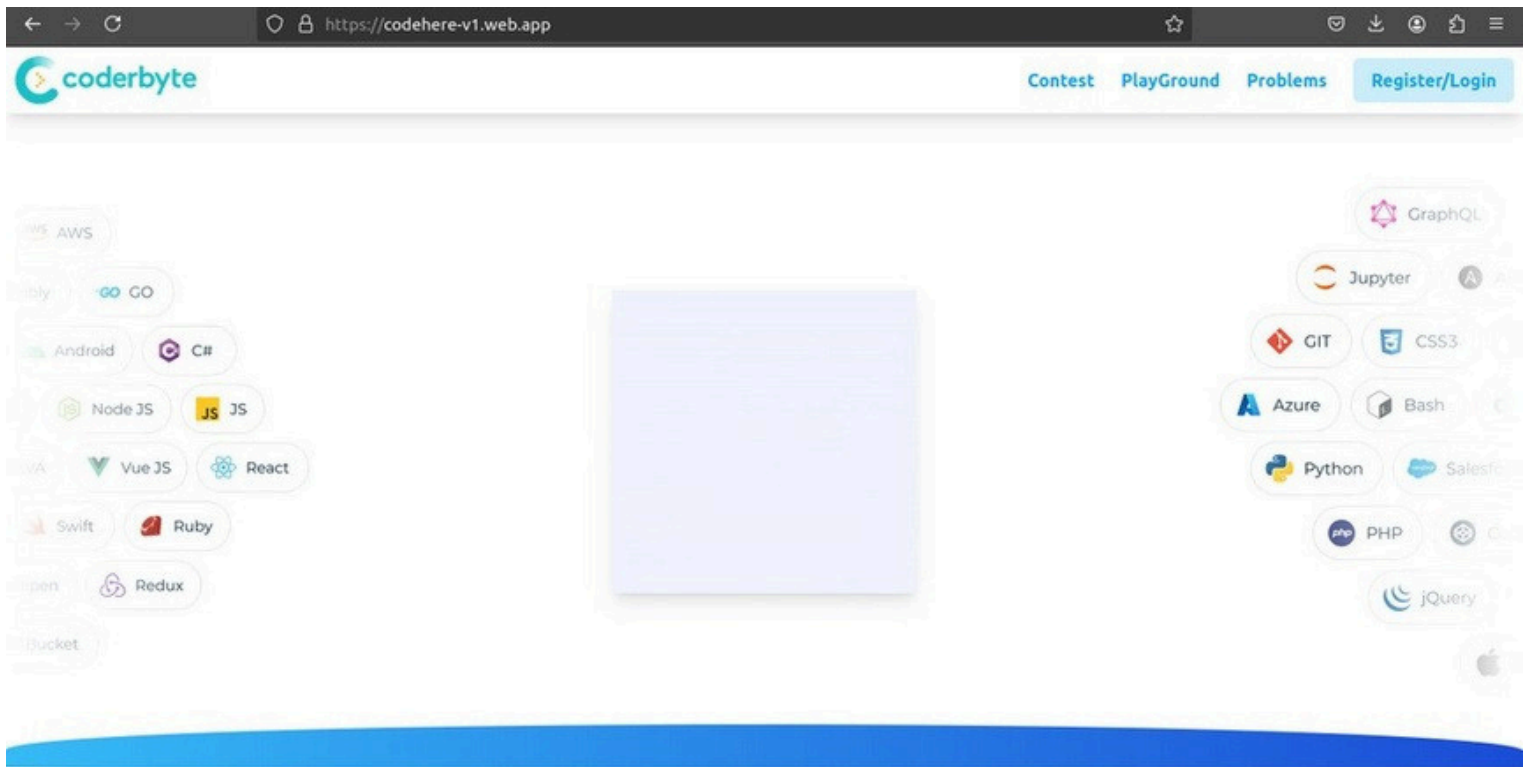


Figure 4.1: Homepage

The home page of the Coding Contest Arena features a clean and intuitive layout. At the top, there is a navigation bar with links to key sections:

- **Problems:** Browse and solve coding challenges.

- **Contest:** View and join ongoing contests.

- **Playground:** Experiment with code in an interactive environment.

- **Login:** Access your account or register.

This user-friendly interface ensures easy access to all essential features for both participants and organizers.

## 4.2 Login Page

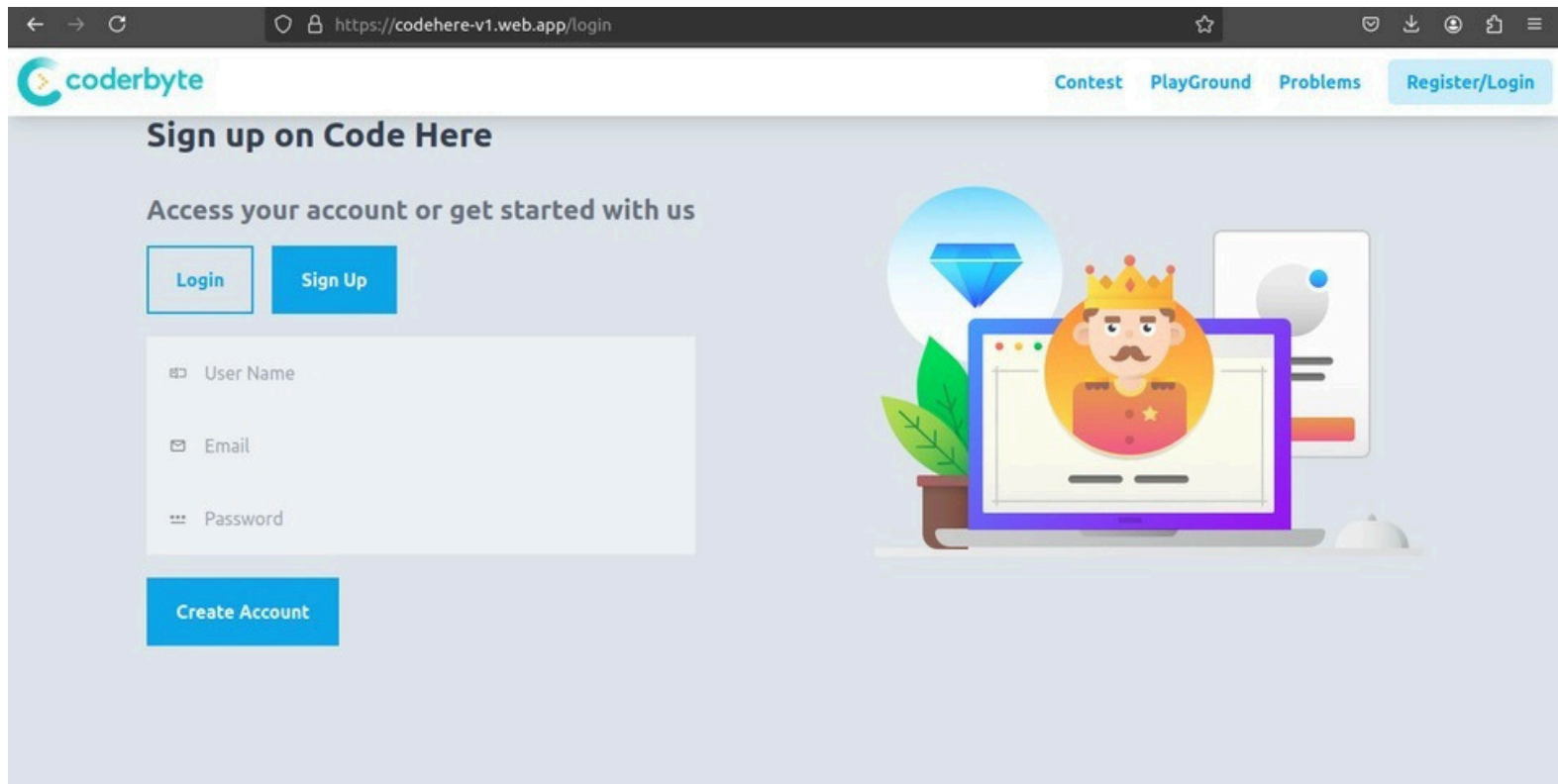


Figure 4.2: Login/SignUp

The Login and Signup page of the Coding Contest Arena is designed to be simple and user-friendly.

### **Login:**

- Username/Email: Users enter their registered username or email.
- Password: Users enter their password.
- Submit Button: Clicking this logs the user into their account.

### **Signup:**

- Username: Users choose a unique username.
- Email: Users provide a valid email address.
- Password: Users create a secure password.
- Confirm Password: Users re-enter the password to confirm.
- Submit Button: Clicking this registers the user and creates a new account.

## 4.3 Coding PlayGround Page

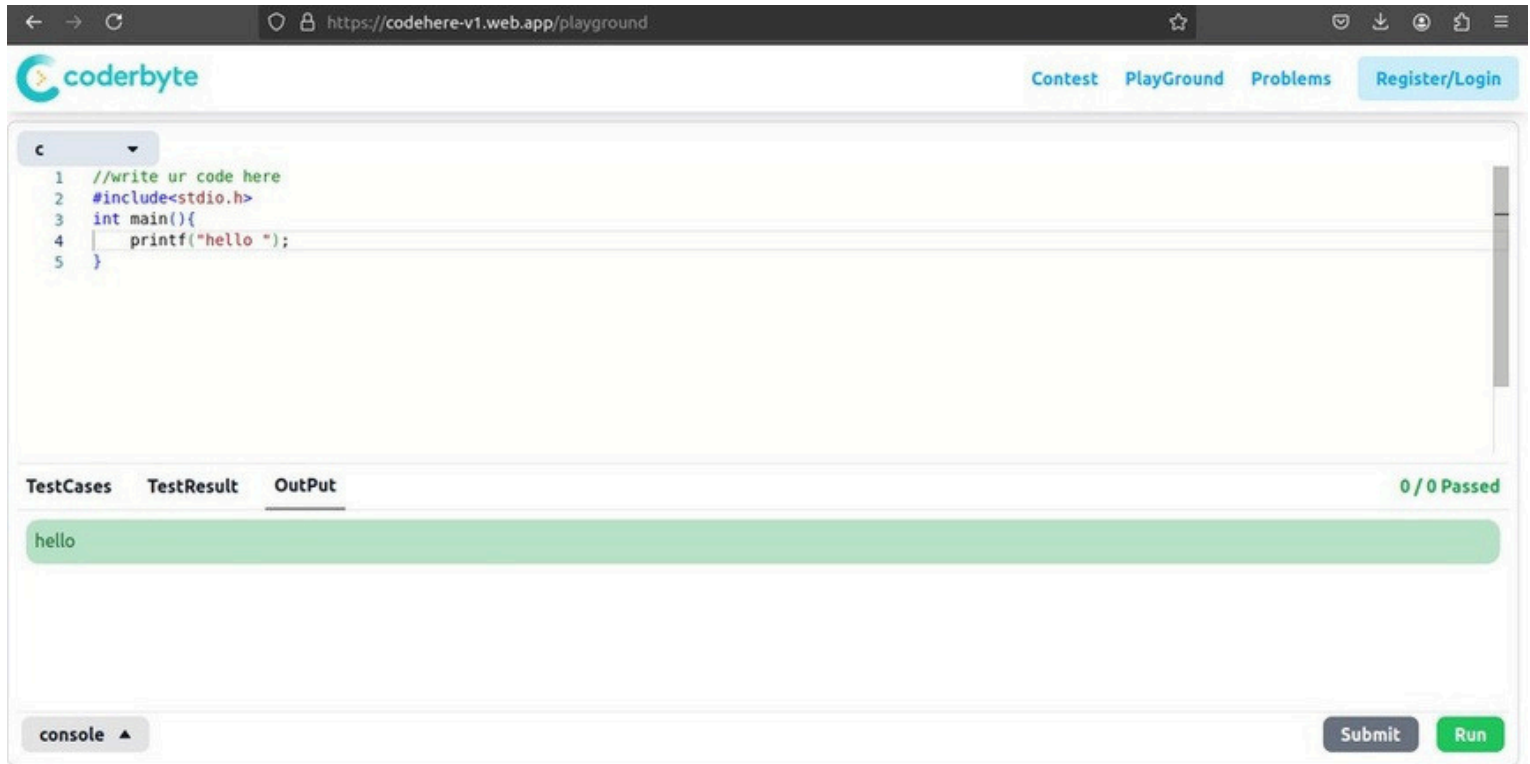


Figure 4.3: Coding PlayGround

### **Certainly! Here's a description for the coding playground page:**

#### Coding Playground Page Description

The Coding Playground page is an interactive and versatile coding environment designed to facilitate writing and testing code in various programming languages. It features:

1. **Multi-Language Support:** Users can choose from a wide range of programming languages such as Python, Java, C++, JavaScript, and more to write their code.
2. **Integrated Code Editor:** A robust code editor with syntax highlighting, auto-completion, and error detection to enhance the coding experience and efficiency.
3. **Real-Time Code Execution:** The ability to run code instantly and see the results in real-time, making it easier to debug and test programs.
4. **Output Console:** A dedicated console to display the output of the code, including standard output, error messages, and any other relevant information.

# 4.4 Contest Page

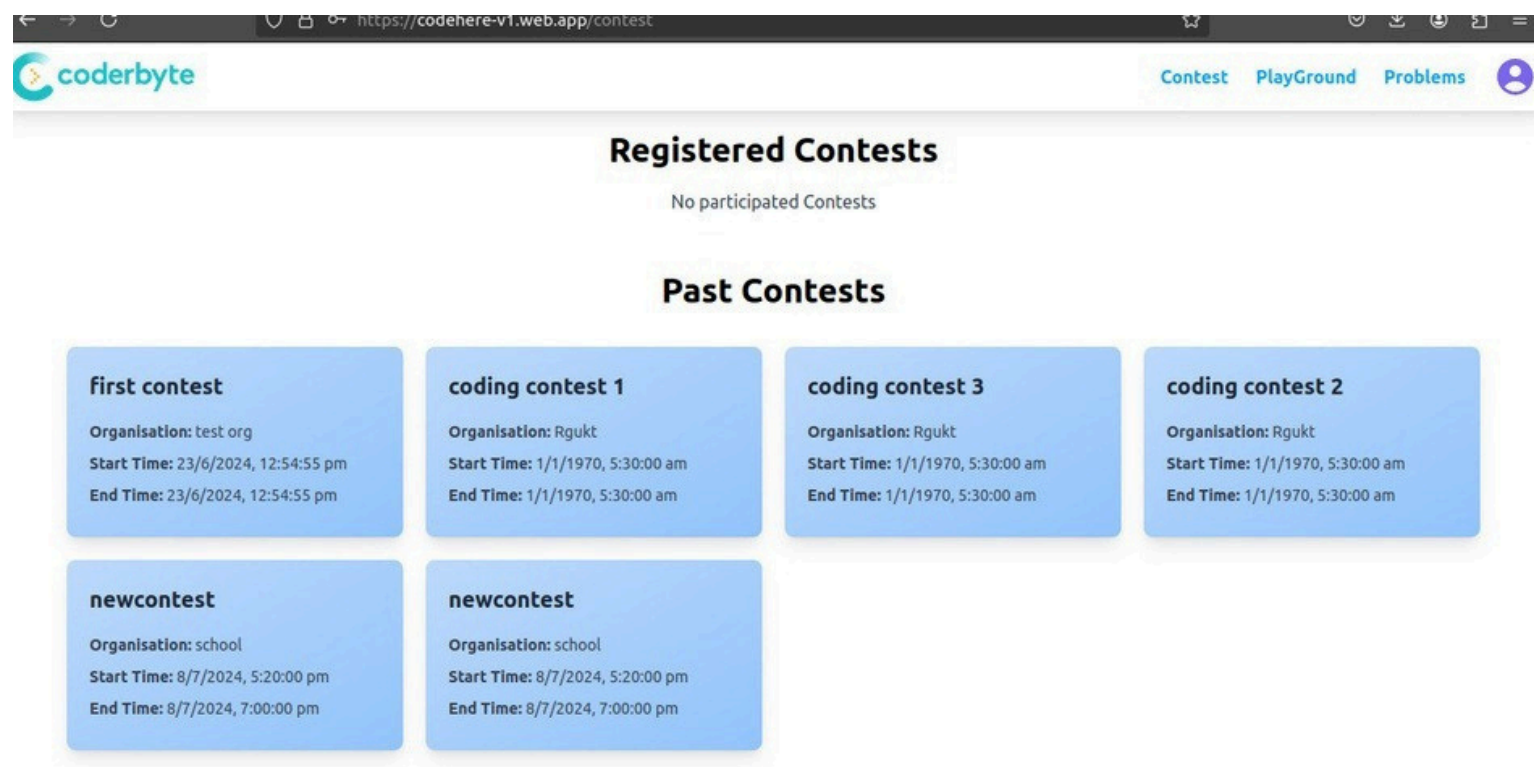


Figure 4.4: Contest Page

**Contest Page Description :** The Contest Page is your gateway to engaging in exciting coding competitions and reviewing previous challenges. It offers a comprehensive and user-friendly interface designed to enhance your competitive programming experience. Key features include:

- 1.Active Contests: A section displaying ongoing and upcoming contests, complete with details such as start and end times, contest rules, and prize information.
- 2.Registration: Easy registration for contests with options to join individually or as part of a team.
- 3.Problem Statements: Access to detailed problem statements for each contest, outlining the challenges and requirements participants need to solve.
- 4.Real-Time Leaderboards: Live leaderboards to track your performance and ranking against other participants during active contests.

# 4.5 Problems Page

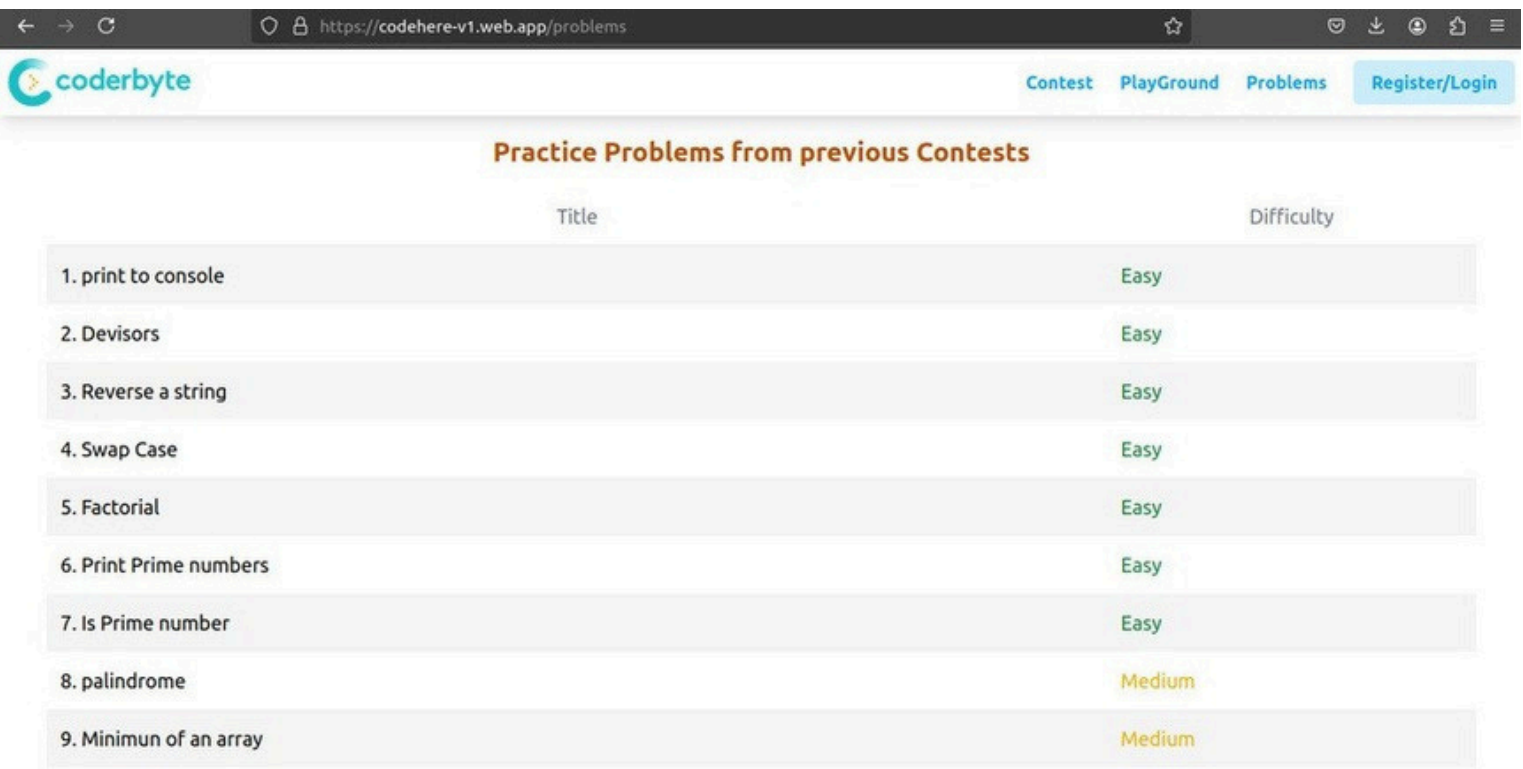


Figure 4.5: Problem Page

**Problem Page Description :** The Problem Page is a comprehensive repository of coding challenges designed to help you practice and improve your programming skills. It features a vast collection of problems categorized by difficulty, topic, and programming language. Key elements include:

- 1.Wide Range of Problems: Access to numerous problems ranging from beginner to advanced levels, covering various topics such as algorithms, data structures, mathematics, and more.
- 2.Categorization and Filters: Advanced filtering options to sort problems by difficulty, popularity, recently added, and specific topics or languages.
- 3.Detailed Problem Statements: Each problem comes with a detailed statement, including a clear description of the task, input and output specifications, and example cases.
- 4.Sample Test Cases: Problems include sample test cases to help you understand the expected input-output format and validate your solutions.



# 4.6 Dash Board

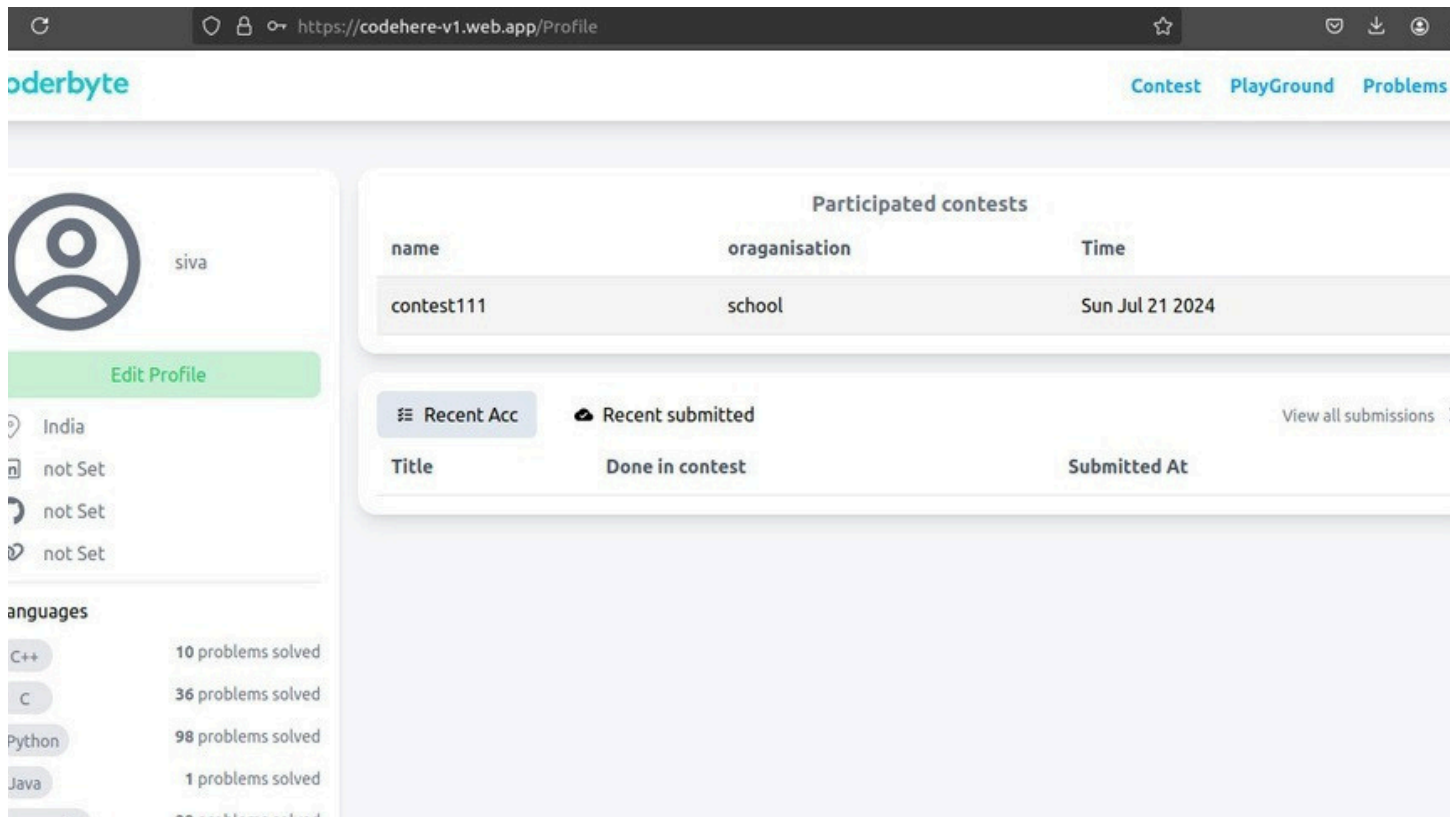


Figure 4.6: Dash Board

**Certainly! Here's a description for the dashboard page:** Dashboard Page Description The Dashboard Page is your personalized hub, offering a comprehensive overview of your activity and achievements on the platform. It provides quick access to your contest history, recent problem-solving activity, and personal information. Key features include:

- 1.Past Contest Participation History: A detailed log of all the contests you have participated in, including dates, rankings, scores, and performance summaries for each contest.
- 2.Recent Solved Problems: A list of the most recent problems you have solved, showcasing your latest coding activity and progress. Each entry includes the problem title, submission date, and status.
- 3.User Information: A summary of your profile information, including your username, profile picture, bio, and any badges or achievements you have earned.
- 4.Performance Analytics: Visual analytics and charts to help you track your performance over time, highlighting trends, strengths, and areas for improvement.

## 4.7 Profile Editing

The screenshot shows a web application interface for profile editing. At the top, there's a navigation bar with the logo 'codebyte' and links for 'Contest', 'PlayGround', and 'Problems'. Below this, a user profile for 'siva' is displayed with a placeholder profile picture. A modal titled 'Basic Info' is open, showing a form with the following fields:

Field	Value	Action
firstName		Edit
lastName		Edit
userName	siva	Edit
githubLink		Edit
linkedinLink		Edit
portfolioLink		Edit

A 'Save' button is located at the bottom right of the modal.

Figure 4.7: Profile Editing

### Profile Editing Page Description :

The Profile Editing Page allows users to update and personalize their account information.

Key features include:

1. Personal Information: Edit your username, bio, and contact details.
2. Profile Picture: Upload or change your profile picture.
3. Password Management: Update your password and manage security settings.
4. Social Links: Add or modify links to your social media profiles.
5. Preferences: Customize your notification preferences and display settings.

## 4.8 Contest Creation

codebyte

Contest PlayGround Problems

### Contest Details

Contest name Enter contest name

Contest URL Enter contest name

Start date dd/mm/yyyy, --:-- --

End date dd/mm/yyyy, --:-- --

Organisation name Enter org name

next

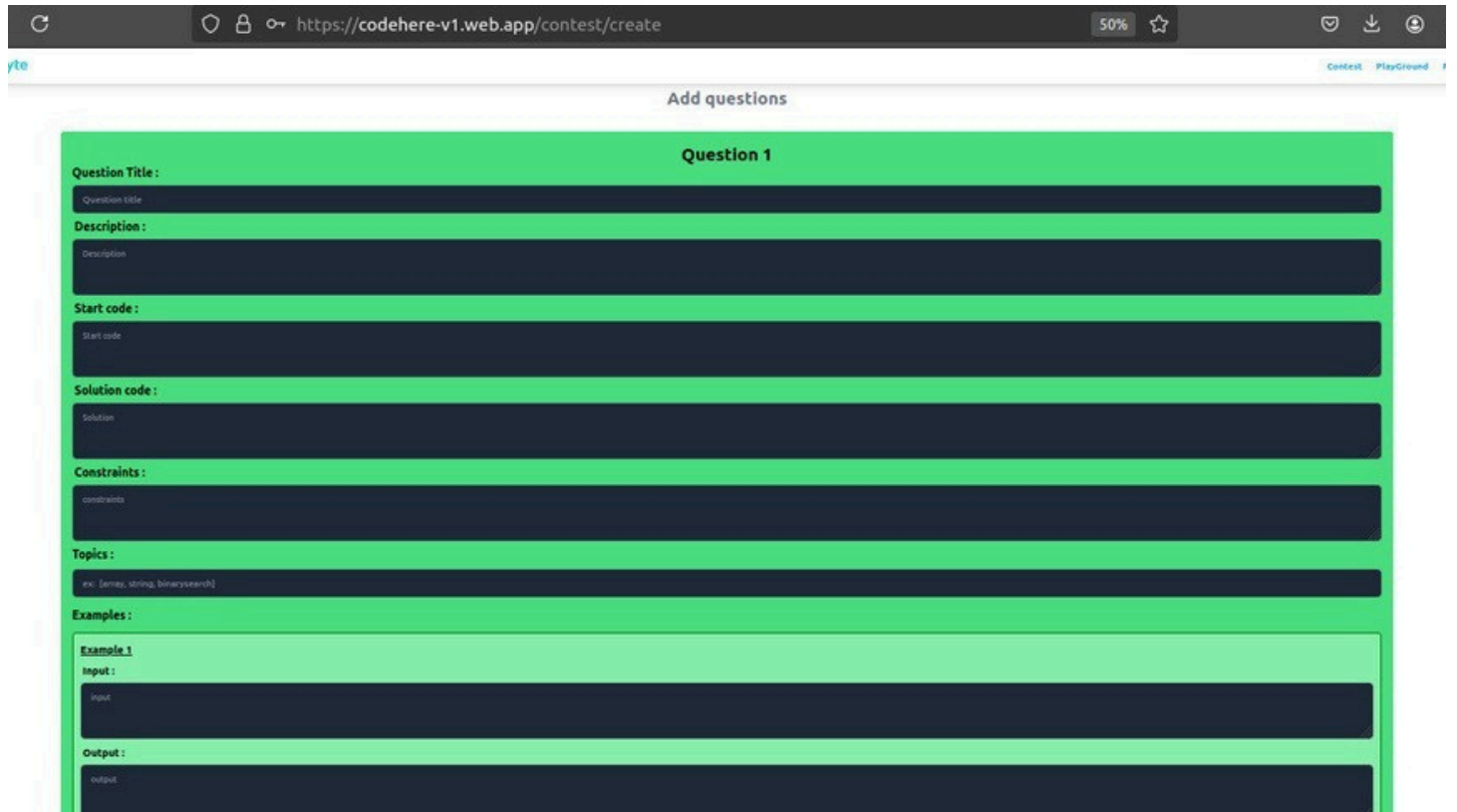
Figure 4.8: Contest Creation

### Contest Creation Page Description :

The Contest Creation Page is a comprehensive tool that allows users to design and launch their own coding contests. Key features include:

1. Contest Details: Input the contest name, description, start and end dates, and rules.
2. Problem Management: Add, edit, and organize problems to be included in the contest, complete with detailed statements, test cases, and solutions.
3. Participant Settings: Set eligibility criteria, participant limits, and team configurations.
4. Scoring System: Define the scoring rules, including points per problem, time penalties, and tie-breaking criteria.

## 4.9 Adding Question



The screenshot shows a web browser window with the URL `https://codehere-v1.web.app/contest/create`. The page title is "Add questions". The form is titled "Question 1" and contains several input fields:

- Question Title :** A text input field with the placeholder "Question title".
- Description :** A text input field with the placeholder "Description".
- Start code :** A text input field with the placeholder "Start code".
- Solution code :** A text input field with the placeholder "Solution".
- Constraints :** A text input field with the placeholder "constraints".
- Topics :** A text input field with the placeholder "ex: [array, string, binarysearch]".
- Examples :** A section containing:
  - Example 1** (highlighted in light blue):
    - input :** A text input field with the placeholder "input".
    - Output :** A text input field with the placeholder "output".

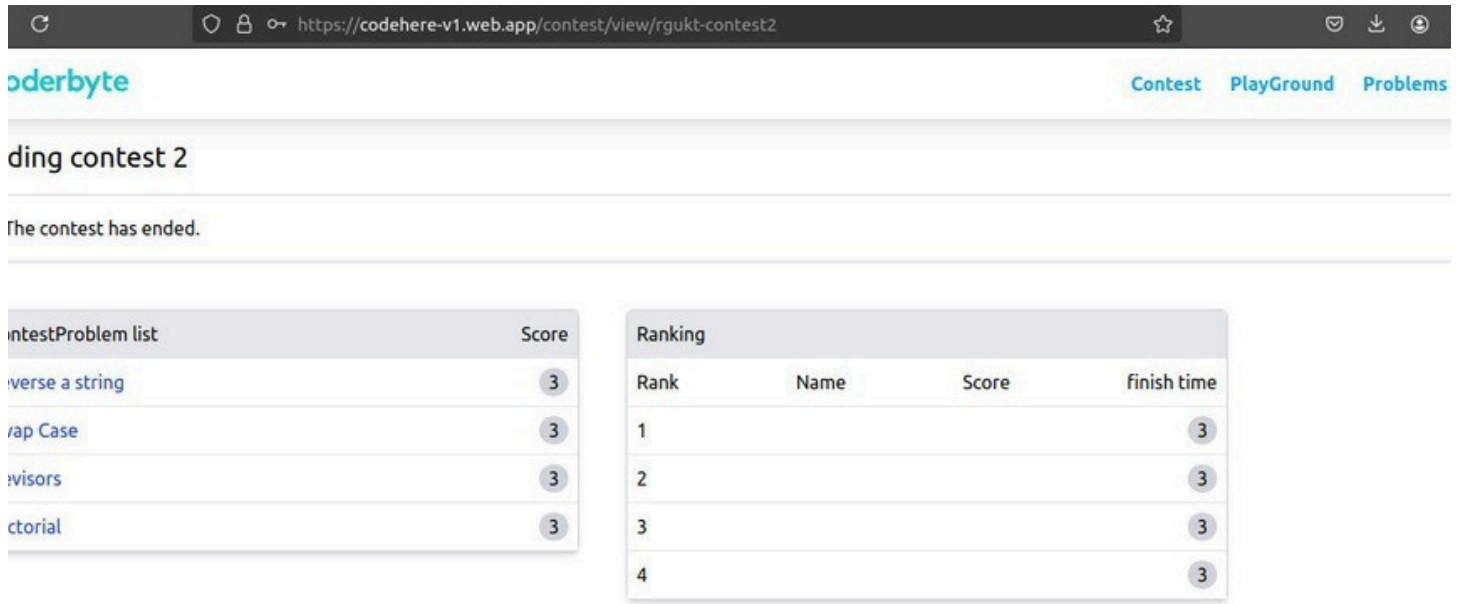
Figure 3.2: Adding Question

### Adding Questions in Contest Page Description :

The Adding Questions in Contest Page allows contest organizers to seamlessly incorporate coding problems into their contests. Key features include:

1. **Problem Input:** Enter the problem title, description, and any relevant instructions for participants.
2. **Input/Output Specifications:** Define the expected input and output formats, along with sample input and output examples.
3. **Test Cases:** Upload or create test cases to validate participants' solutions, including both sample and hidden test cases.
4. **Difficulty Level:** Assign a difficulty level to each problem to guide participants.

## 4.10 Upsolving Questions



The screenshot shows a web browser window with the URL `https://codehere-v1.web.app/contest/view/rgukt-contest2`. The page title is "ding contest 2". A message states "The contest has ended." Below this, there are two tables:

Contest Problem list	Score
Reverse a string	3
Swap Case	3
Divisors	3
Tutorial	3

Ranking			
Rank	Name	Score	finish time
1			3
2			3
3			3
4			3

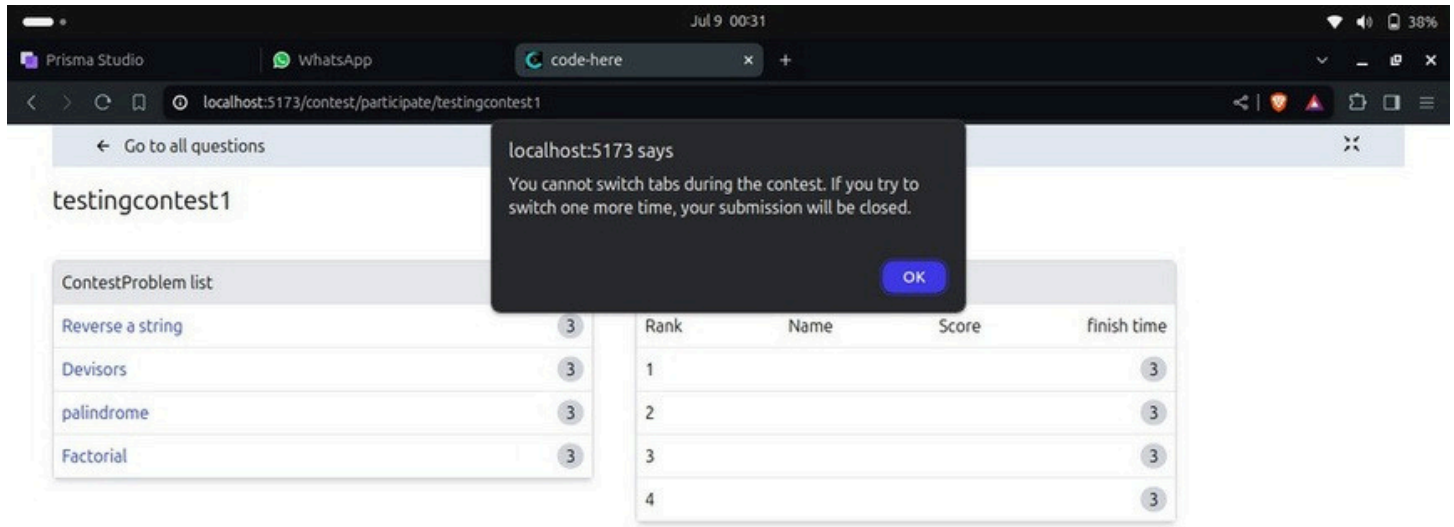
Figure 4.10 Upsolving Questions

### **Upsolving Questions After Contest Description :**

The Upsolving Feature allows participants to continue working on and solving contest problems after the contest has ended. Key features include:

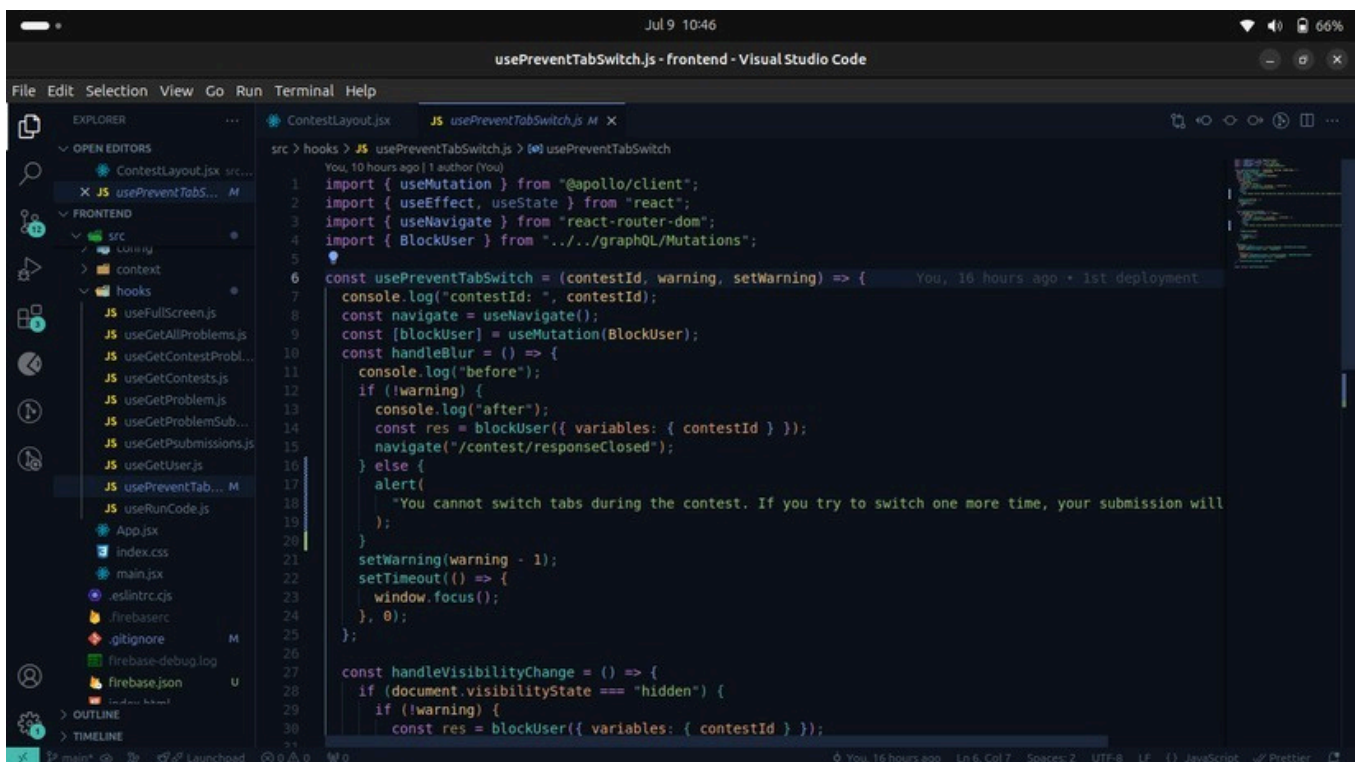
1. Access to Problems: Gain full access to all contest problems and their details even after the contest is over.
2. Solution Submissions: Submit solutions to problems you couldn't solve during the contest and receive feedback on correctness and performance.
3. Editorials and Solutions: Access detailed editorials and official solutions to understand the best approaches to solving each problem.
4. Test Cases Review: Review test cases used during the contest to better understand where your solutions might have gone wrong.

## 4.11 Tab Switch Alert



## 4.11 Tab Switch Alert

The Contest page of the Coding Contest Arena is designed to offer a secure and immersive experience for participants by preventing tab switching.





## 4.12 Copy/Pasting Restriction

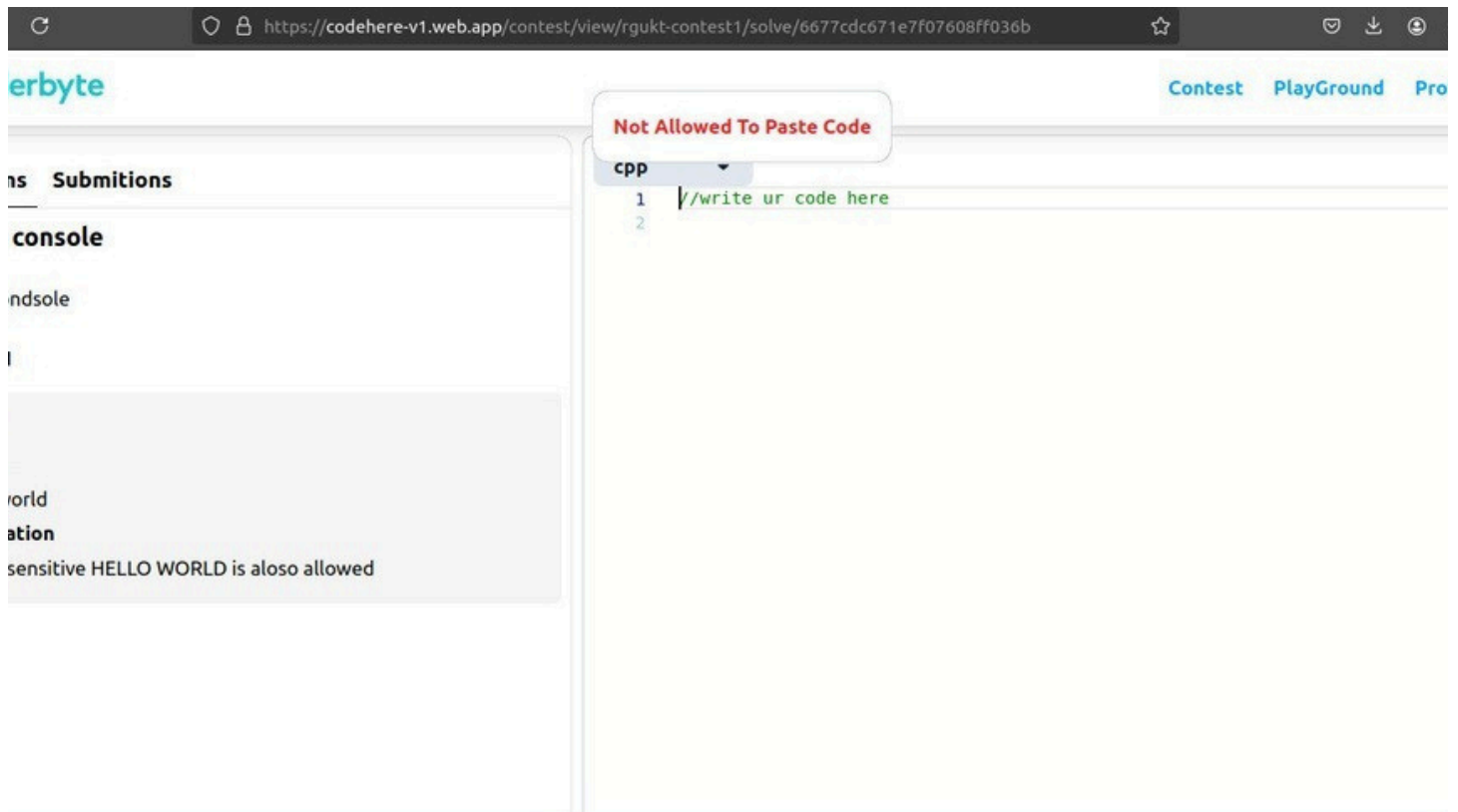
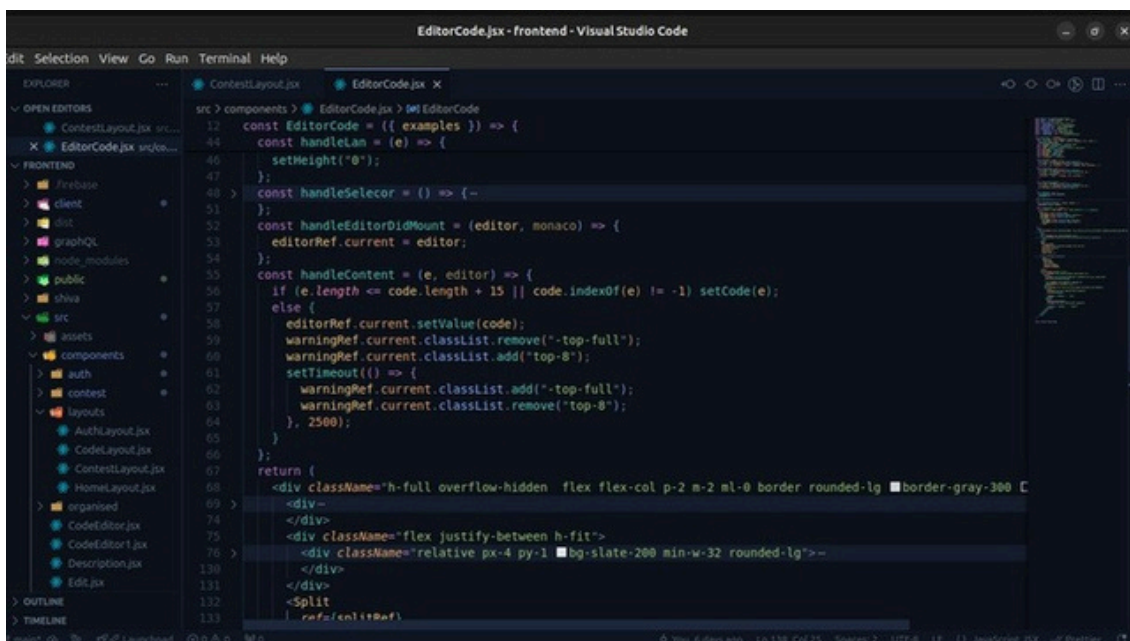


Figure 4.12 Copy/Paste Restriction

here is the code for the implementation of copy pasting code restriction



## 4.13 Email Sending

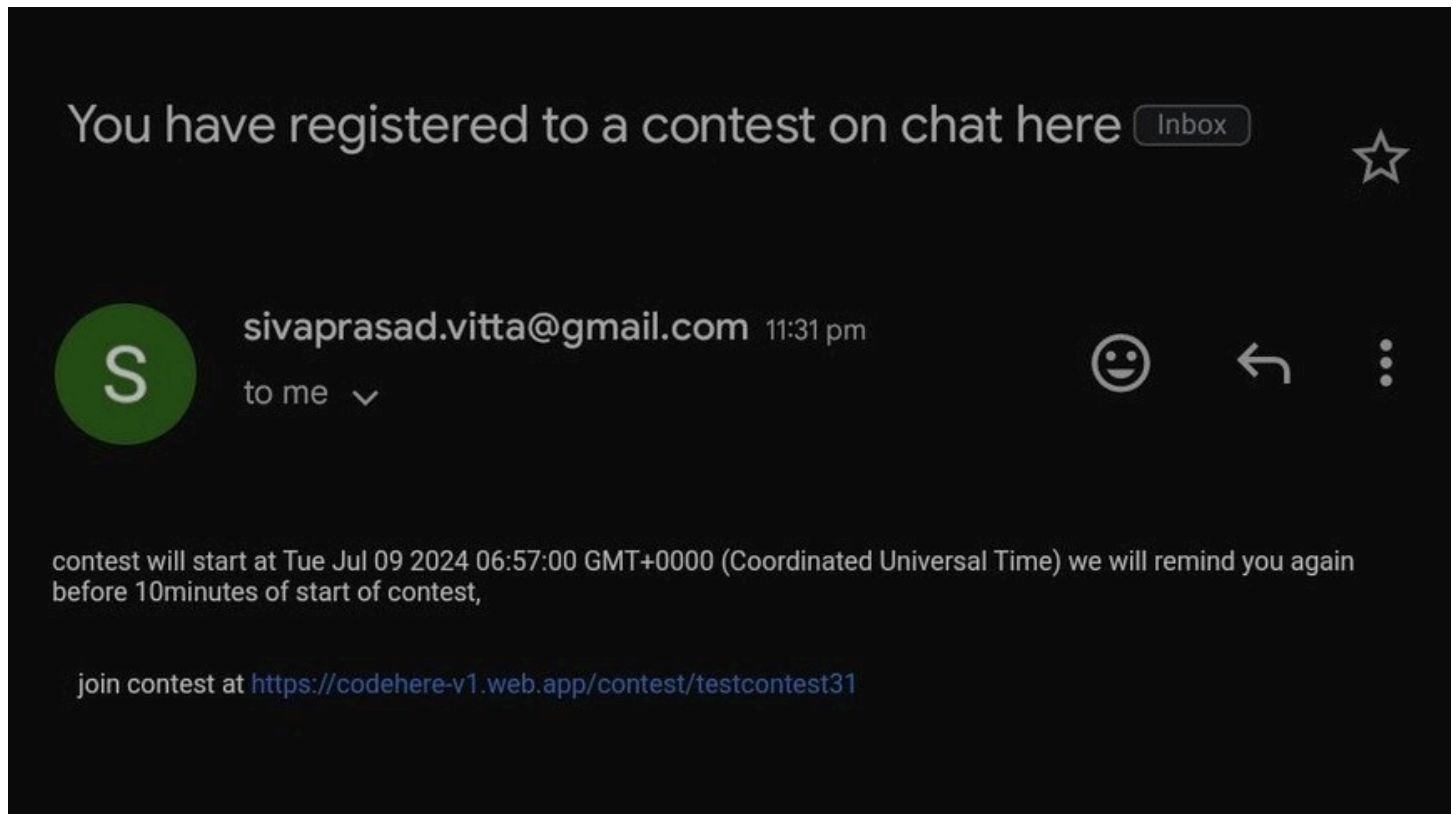
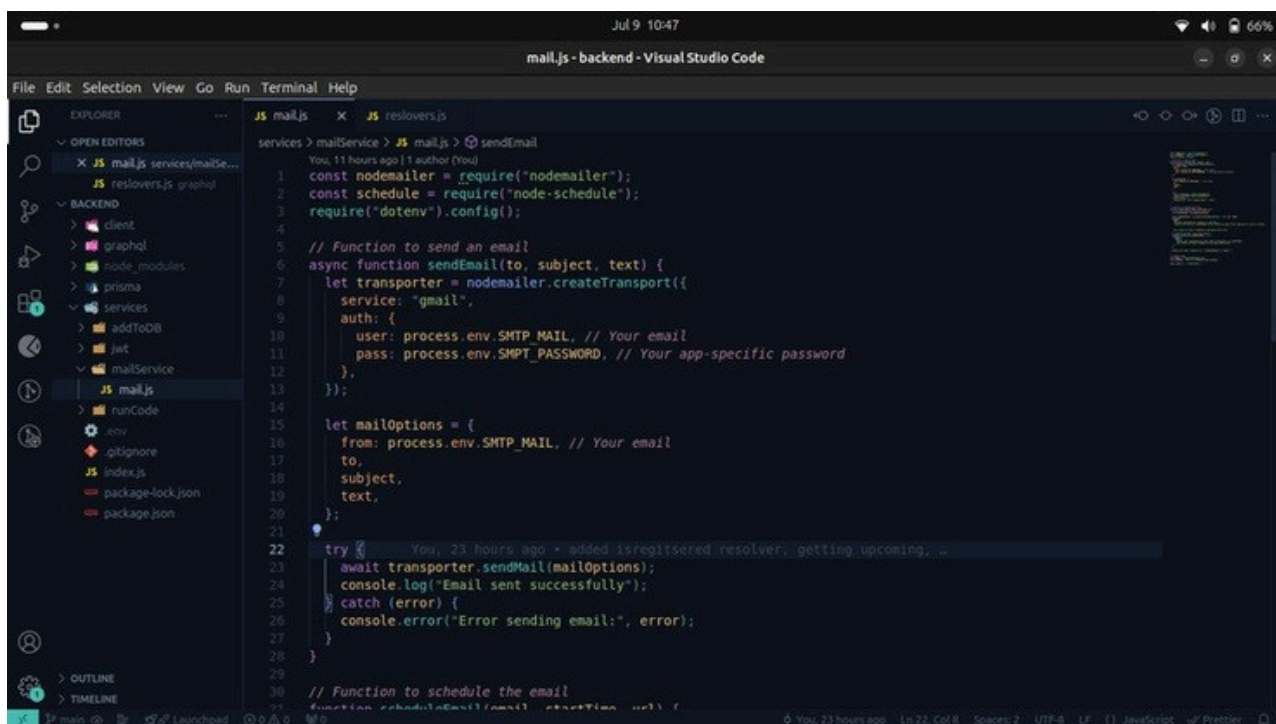


Figure 4.13 Email Sending

here is the backend code for email sending





# CHAPTER 5

## Conclusion

The platform's modern technology stack, including React.js for a responsive frontend and Node.js for a scalable backend, ensures a seamless user experience. The integration of secure data handling practices, such as HTTPS and password hashing, guarantees robust security and privacy for all users.

In conclusion, the "Coding Contest Arena" not only meets the current demands of coding competitions but also sets a new standard for future developments in this space. It empowers participants to hone their skills and provides organizers with the tools needed to create and manage successful contests, thereby contributing to the growth and evolution of the competitive coding community.