Code Arena - A Programming Platform

By

Aryan Langhanoja - 922001330300
Abhay Nathwani - 92310133007

Under the guidance of

Prof. Chandrasinh Parmar

26 September 2025

Marwadi University
Marwadi Chandarana Group
NAAC A+

TABLE OF CONTENTS

LIST OF TABLES

ABSTRACT

This report outlines the development of a **Competitive Programming Platform**, undertaken as part of an academic initiative to enhance coding proficiency and placement readiness among college students. The platform is designed to unify and digitize the process of coding practice and assessment by providing an end-to-end system for managing student submissions and automated evaluations. It addresses key challenges faced by both students and faculty by offering a consistent environment for learning ,thereby reducing inefficiencies.

The system is developed using **Laravel-PHP** for backend services and a **responsive front-end interface using ReactJS**, ensuring a seamless user experience across practice modules. A **MySQL database** is used to manage structured data such as user profiles, coding problems, submissions. The project adheres to modern software development methodologies, including version control, modular architecture..

For students, the platform offers access to a curated bank of **previously asked coding questions**, categorized by difficulty level. It includes a real-time coding interface for practice, with support for automated code evaluation using predefined test cases.

For faculty members, the platform simplifies the recruitment of questions and evaluation of student performance. It features **automated code evaluation**, **checking for large no test cases**, and **bulk upload options** for questions.

Role-based access control is implemented to ensure that students and faculties have appropriate levels of access, thereby maintaining the system's security and operational integrity.

Chapter 1: Introduction

1.1 Project Purpose

The primary objective of this project is to develop a centralized and structured platform that consolidates coding questions previously asked in placement drives. By providing a unified repository of categorized problems and a real-time coding environment with automated evaluation, the platform aims to significantly enhance students' coding proficiency and problem-solving skills. It serves as a practical tool to prepare students for technical interviews and competitive programming challenges. Additionally, the platform streamlines the assessment process for faculty, enabling efficient performance tracking and content management, thereby fostering a more effective and consistent learning experience for both students and educators.

1.2 Product Scope

The Competitive Programming Platform is designed to serve as an all-in-one solution for coding practice, assessment, and preparation for technical placements. It offers a centralized repository of coding problems categorized by difficulty and relevance to previous placement drives. Students can practice in a real-time coding environment that supports automated code evaluation using predefined test cases, helping them gain immediate feedback and improve their programming skills.

For faculty, the platform provides tools for managing question banks, uploading problems in bulk, and evaluating student submissions efficiently. It includes features like role-based access control to differentiate between student and faculty functionalities, ensuring secure and structured access. The responsive user interface ensures accessibility across various devices, while the use of PHP, MySQL, and front-end technologies like ReactJS ensures scalability and maintainability.

The platform aims to bridge the gap between academic learning and industry expectations by promoting consistent practice and performance tracking.

1.3 Intended Audience

The Competitive Programming Platform is designed to cater to a diverse set of users involved in academic and technical skill development. The primary intended audiences include:

- **College Students:**
    - Aspiring to improve their coding skills and prepare for campus placements.
    - Looking for a structured, real-time coding environment with instant feedback.
- **Faculty Members and Educators:**
    - Responsible for conducting coding assessments and monitoring student progress.
    - Seeking efficient tools for uploading, managing, and evaluating coding questions.

1.4 Proposed Solution

To address the functional and non-functional requirements of the Competitive Programming Platform, we propose a scalable, secure, and modular web-based application that supports real-time coding practice, automated evaluations, and progress tracking for students, while offering powerful tools for faculty management.

User Management & Authentication:

- Implement secure user registration and login with email-password combination.
- Passwords will be hashed and salted using secure algorithms like **md5**.
- Password reset flow will include tokenized links.

Coding Interface & Evaluation Engine:

- Integrate a browser-based code editor using **CodeMirror** in the frontend.
- Backend will handle code compilation and automated evaluation using a **Judge0 API**.
- PHP scripts will manage compilation, check test cases, and return real-time feedback to the frontend using **Judge0 Docker Image**.

Faculty Features:

- Faculty can upload coding questions with unlimited test cases using forms uploads.
- The backend will store structured problems and test cases in MySQL, indexed for quick access and scalability.

Student Features:

- Students will practice with no submission limits, view past attempts
- Students having a centralized Repository for question practice with topics included.

1.5 Technology Stack

Frontend Framework: ReactJS

Backend Framework: Laravel - PHP

Database: MySQL

Version Control: Git

Chapter 2: Project Management

The development process was divided into distinct phases to ensure systematic progress, better task allocation, and quality delivery. The following sections detail each phase of the project lifecycle.

2.1 Phase 1: Research and Analysis

This foundational phase was dedicated to understanding the needs of the target users (students and faculty), studying existing online code submission platforms, and framing a realistic scope for the project.

- Requirements were gathered from faculty through discusssion pain points in manual code submission and evaluation.
- We observed the need for such a kind of platform with our personal experience.

2.2 Phase 2: System Design and Architecture

With clear requirements defined, the next step was to design the structure and flow of the platform, ensuring modularity and future scalability.

- A modular architecture was defined, assigning each core functionality (login, practice environment, evaluations, etc.) to separate modules.
- A tech stack comprising HTML, CSS, JavaScript, PHP, and MySQL was finalized with local development using XAMPP.
- The database schema was designed using ER modeling to include tables like `Students`, `Submissions`, `Questions`.

2.3 Phase 3: Development

Development was done in a staged, module-wise manner with low-fidelity sketches leading to functional and then high-fidelity prototypes.

User Authentication Module

- Developed using HTML, CSS, PHP, and MySQL with form validation and session management.
- Login/signup screens were built and tested for different user types (student, faculty).

Code Management Module

- Enabled faculty to create, upload, and manage coding questions.
- Implemented basic CRUD operations for code entries using PHP and MySQL.

Code Submission & Evaluation Module

- Integrated CodeMirror/Ace editor for student code input.
  Designed a submission system that stores and evaluates user code.
- Feedback mechanisms were added to show success/failure and run-time errors.

2.4 Phase 4: Testing

- **Unit Testing:**
  - Used for individual backend components such as code submission, database insertion (e.g., submission tokens), and Judge0 API interaction.
  - **Tools:** PHPUnit for PHP backend testing.

- **Integration Testing:**
  - Ensures different modules like frontend (CodeMirror), backend (PHP), and Judge0 API work together seamlessly.
  - **Tools:** Postman for API testing, Browser DevTools for AJAX responses.

- **System Testing:**
  - Full platform tests from code submission to result evaluation.
  - Simulates user behavior (student login → code write → submit → receive output).
  - **Tools:** Selenium for browser automation.

- **Validation Testing:**
  - Verifies the correctness of output by comparing actual results with expected output stored in the database.
  - Ensures the platform accurately evaluates code submissions against provided test cases.

Chapter 3: System Requirements Study

This chapter outlines the various system requirements identified during the analysis phase. These include both functional and non-functional requirements, along with user-specific needs and hardware/software prerequisites for successful deployment and operation of the system.

3.1 Functional and Non-Functional Requirement

3.1.1 Functional Requirement

- The system must allow users to register for an account using an email address and password.
- The system must allow faculty to upload coding questions with predefined test cases.
- The system must provide an automatic evaluation feature for submitted codes.
- The system must allow students to practice coding questions.
- The system must allow students to view their previous submissions for all questions.
- The system must allow faculty to add unlimited test cases for each coding question.

3.1.2 Non-Functional Requirement

These requirements define the quality attributes and performance expectations of the system:

Performance

- The system must be capable of handling concurrent users without noticeable performance degradation.
- Code compilation results should be displayed within 5 seconds after submission.

Security

- All user passwords must be encrypted.
- User data (such as coding submissions) must be stored.

Usability

- The system must provide a user-friendly interface that allows new users to register, log in, and start practicing within 3 minutes of access.

Maintainability

- The system must be developed with modular code architecture to allow for easy updates and maintenance.

Reliability

- The system must achieve 99.9% uptime with backup and disaster recovery measures in place.

3.2 User Requirements

Different user roles require tailored functionalities to perform their tasks effectively. User-specific needs include:

- **Faculties :-** Can Create , Update the Questions , View the submissions specific to status , student , submission.
- **Students :-** Can View the question list , can practice the question and view their past submission.

3.3 Hardware and Software Requirement

3.3.1 Server-side Hardware Requirements

Table 3.1. Server-side Hardware Requirements

Component	**Specification**
Processor	Intel i5 or higher , multi-core
RAM	Minimum 8 GB
Hard Disk	256 GB SSD or more
Network	Reliable internet connection
Operating System	Windows 10/11, Linux (Ubuntu/CentOS)

Minimum hardware specifications required for hosting and running the server-side components of the application.

3.3.2 Software Requirements

Table 3.2. Software Requirements

Software	**Description**
Backend Framework	Laravel - PHP
Frontend Framework	ReactJS
Database	MySQL
IDEs	Visual Studio Code
Version Control	Git
OS Compatibility	Windows, Linux
Web Server	Apache

List of essential software tools, frameworks, and platforms required for developing and deploying the server-side application.

3.3.3 Client-side Requirements

Table 3.3. Client-side Requirements

Component	**Specification**
Web Browser	Any Browser Supporting Javascript
OS	Windows, macOS, or Linux
Internet Connection	Required for accessing web application

Minimum hardware and software specifications required on the client side to access and use the web application effectively.

Chapter 4: System Analysis

System analysis plays a vital role in designing a user-centric, functional, and technically feasible platform. This phase involved gathering insights from students and faculty, understanding the requirements of online coding assessments, and translating them into system modules with clearly defined roles and interactions. The objective was to create a platform that streamlines code practice, submission, evaluation, and result reporting.

4.1 Feature Specification

The platform is divided into key functional modules, each catering to a specific stage in the online coding assessment lifecycle. Below is a breakdown of the primary features:

4.1.1 User Authentication and Role Management

Users must authenticate before accessing any module. The system supports secure session handling and multiple user roles:

- **Roles**:
    - **Admin**: Full control over users and content
    - **Faculty**: Can upload/manage problems, view results, and schedule exams
    - **Student**: Can practice problems, take assessments, and view results

Key Features:

- Sign-up and login functionality
- Password hashing and secure session management
- Role-based redirection and access control

4.1.2 Code Management and Question Upload

This module allows faculty and admins to create, manage, and organize coding problems.

Key Features:

- Problem creation with fields such as title, description, constraints, and sample input/output
- Upload test cases and assign tags (e.g., Easy, Medium, Hard)
- Ability to edit, delete, and categorize questions
- Preview functionality to check formatting and correctness.

4.1.3 Code Submission and Evaluation Engine

This is the core module responsible for evaluating student code in real-time using predefined test cases.

Key Features:

- In-browser code editor using **CodeMirror**
- Support for multiple programming languages
- Real-time syntax highlighting and auto-indent features
- Backend code execution via a secure sandboxed environment
  Automated evaluation with verdicts like "Accepted," "Wrong Answer," or "Runtime Error"
- Display of compilation or runtime errors if present.

4.1.4 Practice Module

Students can use this module for self-paced learning by solving curated problems.

Key Features:

- View the question with topic included
- Practice questions are not timed or graded
- Code can be submitted multiple times

4.1.5 Exam/Assessment Module (Planned Future Integration)

This module will allow faculty to create and manage time-bound coding exams.

Key Features:

- Scheduled start and end time
- Auto-submit upon timeout
- Plagiarism check integration (future scope)
- Role-specific dashboard showing upcoming and past assessments.

4.1.6 Role-Based Access Control (RBAC)

RBAC ensures that each user can only access functionalities that align with their role.

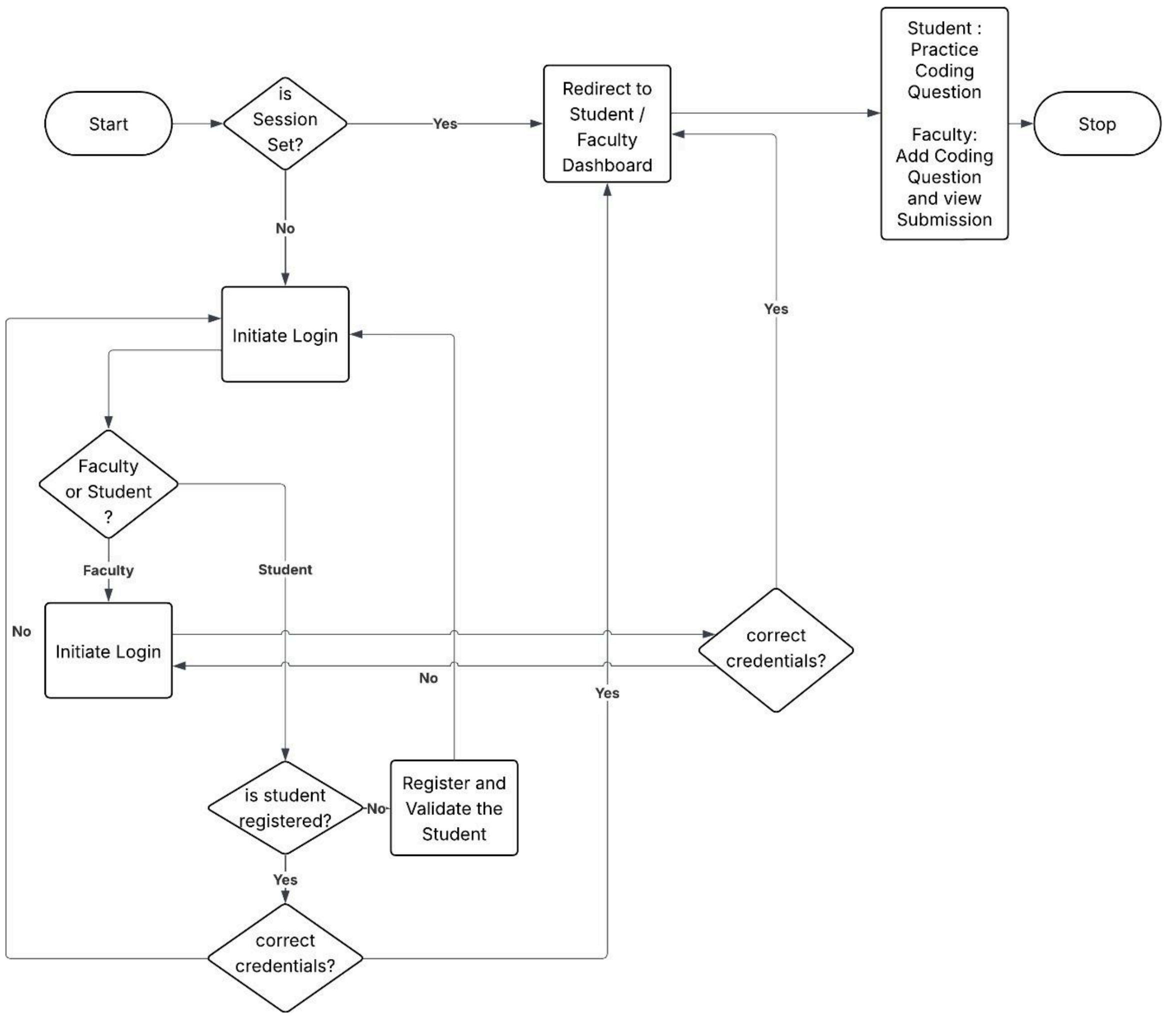
Roles and Permissions:

Role	Features Accessible
Student	Practice module, code submission, view results
Faculty	Add/edit questions, view all student submissions and scores
Admin	User management, system config, full access

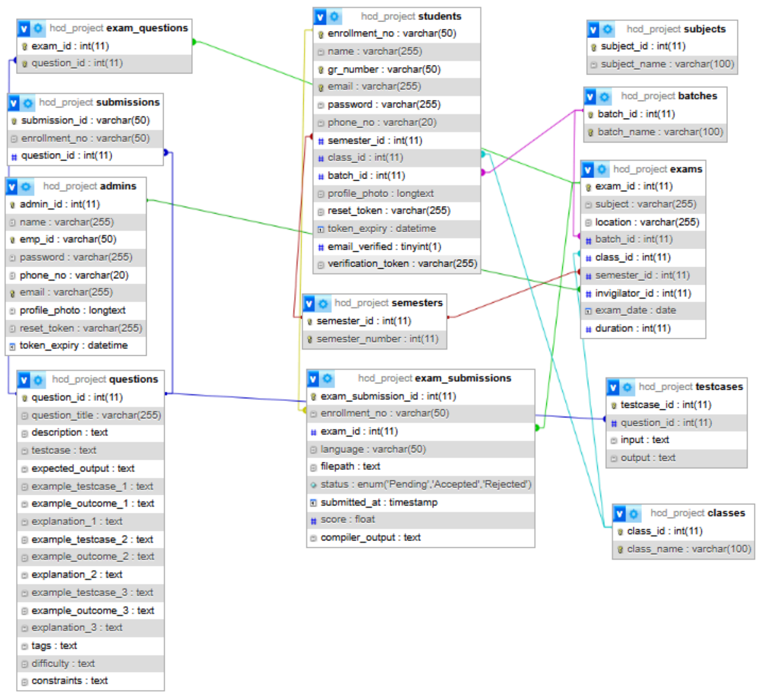
All actions are validated at both frontend and backend layers to ensure data integrity and prevent privilege escalation.

4.2 UML Diagrams

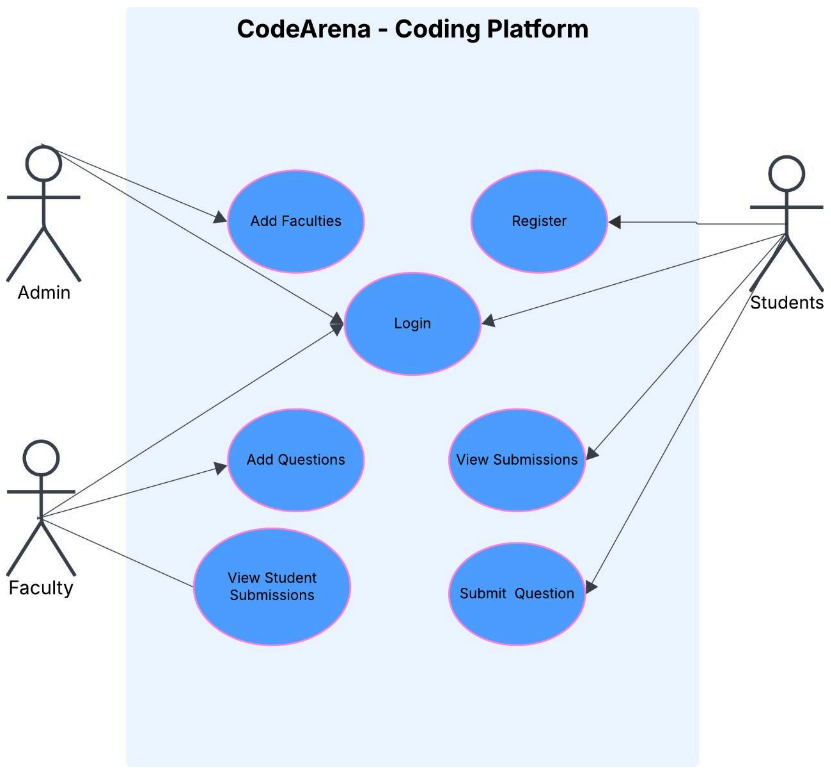
Flow Chart



Database Schema



Use Case Digram


CodeArena - Coding Platform
Admin
Add Faculties
Register
Students
Login
Add Questions
View Submissions
Faculty
View Student Submissions
Submit Question