**CodeClash**

**A PROJECT REPORT**

**for**

**Mini Project-II (ID201B)**

**Session (2024-25)**

**Submitted by**

**Mohammad Daud**

**202410116100121**

**Mayank Srivastava**

**202410116100118**

**Submitted in partial fulfillment of the**

**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

**Dr. Ankit Verma**

**Associate Professor**

**Ms. Shruti Aggarwal**

### Assistant Professor



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**(MAY - 2025)**

**Declaration**

We, the undersigned, hereby declare that the Mini Project titled "**CodeClash**" is an original work completed by us as part of the curriculum requirement for the course under the Master of Computer Applications (MCA) program at **KIET Group of Institutions**.

We affirm that we have undertaken this during the academic year 2024-25 under the guidance of **Ms. Shruti Aggarwal and Dr. Ankit Verma**. All the content and ideas presented in this report are the result of our own efforts, except where explicitly stated otherwise. Proper citations have been provided wherever references to external sources have been made.

We further declare that this project has not been submitted, either in part or in full, to any other university or institution for any degree or diploma.

Mohammad Daud

Mayank Srivastava

**CERTIFICATE**

Certified that **MOHAMMAD DAUD (202410116100121), MAYANK SRIVASTAVA (202410116100118)** have carried out the project work having “**CodeClash**” (**Mini Project-II, ID201B**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself, and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Dr. Ankit Verma Dr. Akash Rajak**

**Associate Professor Dean**

**Department of Computer Applications Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad**

**Abstract**

The CodeClash is a web application designed for students, which is an educational platform where user can do competitive coding along with variety of questions. Along with coding, users can do quiz with choice of questions where difficulty level is low to high and once submitting the quiz user can check his performance and also get feedback. In this project we address some insufficiencies about previously developed quiz platforms and with this we make this interactive and knowledge based. This project addresses inefficiencies in traditional educational model and ensures accuracy and quality education.

Key features of the system include:

* A user-friendly interface built using React and Tailwind CSS.
* Toasts are used to make interface interactive.
* Create a smooth backend using Spring Boot ( Java ).
* MongoDB is used for backend.
* Focused on quality education, feels like a game.
* Secure and scalable architecture for efficient performance.

This project demonstrates the use of modern web technologies to enhance administrative workflows in educational institutions and help the students and competitive programmers to enhance programming skills with problem solving.

**Keywords**: codeclash, coding, React, Tailwind CSS, SpringBoot, Java

**Acknowledgments**

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Shruti Aggarwal and Dr. Ankit Verma,** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to **Dr. Akash Rajak**, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Name - Mohammad Daud**

**Roll No - 202410116100121**

**Name – Mayank Srivastava**

**Roll No - 202410116100118**

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Declaration……………………………………………………………... | | i |
|  | Certificate………………………………………………………………. | | ii |
|  | Abstract…………………………………………………………………. | | iii |
|  | Acknowledgments……………………………………………………… | | iv |
|  | Table of Contents………………………………………………………. | | v |
| 1 | Introduction…………………………………………………………….. | | 7-8 |
|  | 1.1 | Overview…………………………………………………………. | 7 |
|  | 1.2 | Basic Communication Model…………………………………….. | 7 |
|  | 1.3 | Data Communication M…………………………………………... | 8 |
| 2 | Literature Review………………………………………………………. | | 9-11 |
|  | 2.1 | Key features of CodeClash……………………………………….. | 9 |
|  | 2.2 | Role of modern web technologies..………………………………. | 10 |
| 3 | Project Flow………………………………………………………….. | | 12-14 |
|  | 3.1 | Use Case Diagram………………………………………………… | 12 |
|  | 3.2 | Data Flow Diagram……………………………………………….. | 13 |
|  | 3.3 | ER Diagram……………………………………………………….. | 14 |
| 4 | Database Tables…..……………………………………………………. | | 15 |
| 5 | System Structure……………………………………………………….. | | 20-21 |
|  | 5.1 | System Architecture………………………………………………. | 18 |
| 6 | Implementation.…………………………………………….................... | | 22 |
| 7 | Project Outcome………………………………………………………… | | 23 |
| 8 | Testing…………………………………………………………………… | | 24 |
| 9 | Project Output…...………………………………………………………. | | 25-26 |
| 10 | Output Screenshots……………………………………………………… | | 27-42 |
| 11 | Conclusion………………………………………………………………. | | 43 |
| 12 | References………………………………………………………………. | | 44 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| 1 | Project Flow………………………………………………………….. | 12 |
| 2 | Use Case Diagram……………………………………………………. | 12 |
| 3 | Level 0 DFD..………………………………………………………… | 12 |
| 4 | Level 1 DFD……………...…………………………………………... | 13 |
| 5 | Level 2 DFD……………...…………………………………………... | 13 |
| 6 | ER Diagram..…………………………………………………………. | 14 |

**1. Introduction**

In today’s fast-paced world of technology, competitive coding is a vital skill for aspiring developers. While platforms like LeetCode, CodeChef, and HackerRank offer a wide range of problems, they often lack the excitement of a live, interactive environment. CodeClash is a single-player real-time coding challenge platform designed to make problem-solving more engaging and performance-driven. Users can register, create a profile, and test their skills by taking on timed coding challenges categorized by difficulty and topic. The platform automatically evaluates submissions and provides instant feedback based on accuracy and speed. To complement hands-on coding, CodeClash also features a dedicated Quiz Section that helps users strengthen their theoretical foundation in areas such as data structures, algorithms, time complexity, and language-specific concepts. These quizzes are designed to be short, dynamic, and educational—providing immediate results and explanations to enhance learning. Though currently focused on solo learning and improvement, CodeClash is built with future multiplayer support in mind, where users will be able to create challenge rooms, invite friends, and compete in live coding battles with real-time leaderboards and collaborative learning features.

Key Features:

* User Profiles – Register and manage your progress.
* Coding Challenges – Timed coding tasks with instant feedback.
* Quiz Section – Theory-based quizzes to boost conceptual understanding.
* Real-Time Evaluation – Scoring based on accuracy and speed.

1.1 Overview  
A good system must meet the following criteria:

* Performance: Fast and efficient operation.
* Reliability: Consistent and accurate results.
* Scalability: Ability to grow and support more users.

1.2 Basic Communication Model  
CodeClash operates on a client-server model, where the frontend (Next.js) communicates with the backend (Spring Boot) using REST APIs. Users interact with the web application through a seamless UI, while the backend handles user authentication, room management, challenge execution, and result processing.

1.3 Data Communication  
The data communication model ensures that user actions are processed and stored efficiently. The platform follows a structured request-response cycle, where:

* Frontend (React.js) sends requests (user authentication, challenge submissions) to the backend.
* Backend (Spring Boot) processes these requests, validates data, and interacts with the database.
* MongoDB stores user profiles, challenge data, and rankings.

By integrating Spring Security for authentication and MongoDB for data persistence, CodeClash ensures scalability, security, and efficiency in handling real-time competitions.

**2. Literature Review**

CodeClash is an interactive coding platform designed to make competitive programming more dynamic and engaging. Unlike traditional coding platforms that emphasize solo problem-solving, CodeClash focuses on an immersive environment that fosters performance tracking, conceptual reinforcement, and an intuitive user experience. While currently built as a single-player application, CodeClash is architected to support real-time multiplayer battles in future updates. The platform encourages both practical coding and theoretical learning by integrating live coding challenges and a Quiz Section that enhances conceptual understanding in data structures, algorithms, and programming fundamentals.

2.1 Key Features of CodeClash

* User Registration & Profile Management: Users can register, log in, and manage their profiles, including editing their name and date of birth. Email and username are non-editable to preserve identity consistency.
* Coding Challenge Module: Users can choose a difficulty level and topic, attempt problems under time constraints, and receive instant feedback on correctness and performance.
* Theory-Based Quiz Section: A dedicated quiz section helps users strengthen theoretical foundations. Each quiz provides instant results and explanations to support deeper understanding.
* Timer-Based Competitions: Coding tasks and quizzes are time-bound, encouraging users to develop accuracy and speed under pressure.
* Result & Progress Tracking: After each challenge or quiz, users receive detailed feedback on their performance, which helps in tracking personal growth.
* Multiplayer Ready (Future Update): CodeClash is being developed with future support for real-time multiplayer features such as room creation, invitation codes, and leaderboard-based competitions.

2.2 Role of Modern Web Technologies

1. Frontend – React.js

* The frontend of CodeClash is built using Next.js, a React-based framework that enhances performance and provides server-side rendering (SSR) and static site generation (SSG).

🔹 Why React.js?

* Improves SEO and page load speed with server-side rendering (SSR).
* Supports fast client-side navigation for a smooth user experience.
* Allows easy integration with real-time features (e.g., WebSockets for live updates).

🔹 User Interface (UI) Technologies

* Tailwind CSS for a responsive, modern, and customizable UI.
* React Hooks for state management and dynamic UI updates.

2. Backend – Spring Boot (Java Framework)

The backend is developed using Spring Boot, a powerful Java framework that simplifies backend development, authentication, and API handling.

🔹 Why Spring Boot?

* Provides a robust and scalable architecture for handling multiple user requests.
* Supports Spring Security for secure user authentication and authorization.
* Easily integrates with databases (MongoDB) and external services.

🔹 Core Backend Features in CodeClash

* RESTful APIs for user authentication, room creation, challenge execution, and result processing.
* Role-based authentication using Spring Security (JWT tokens for secure access).
* Efficient data handling with MongoDB.

3. Database – MongoDB (NoSQL Database)

The platform uses MongoDB, a NoSQL database, to store and manage dynamic data related to user profiles, challenges, results, and rankings.

🔹 Why MongoDB?

* Schema flexibility to store various types of data (users, challenges, results).
* High performance and scalability to handle large numbers of participants.
* Fast queries for real-time leaderboard updates and competition results.

4. Real-Time Communication

WebSockets, To ensure real-time updates, CodeClash can integrate WebSockets, which allow live interaction between users.

🔹 Benefits of WebSockets in CodeClash

* Enables real-time status updates (e.g., players joining a room).
* Updates leaderboards dynamically as users submit answers.
* Reduces server load compared to frequent HTTP polling.

5. Authentication & Security – Spring Security & JWT

Security is crucial for any multiplayer platform. CodeClash uses Spring Security and JWT (JSON Web Tokens) for user authentication and session management.

🔹 Security Features in CodeClash

* Password hashing & encryption for secure user data storage.
* JWT-based authentication to manage user sessions.
* Access control to ensure only authorized users can join specific rooms.

**3. Project Flow**

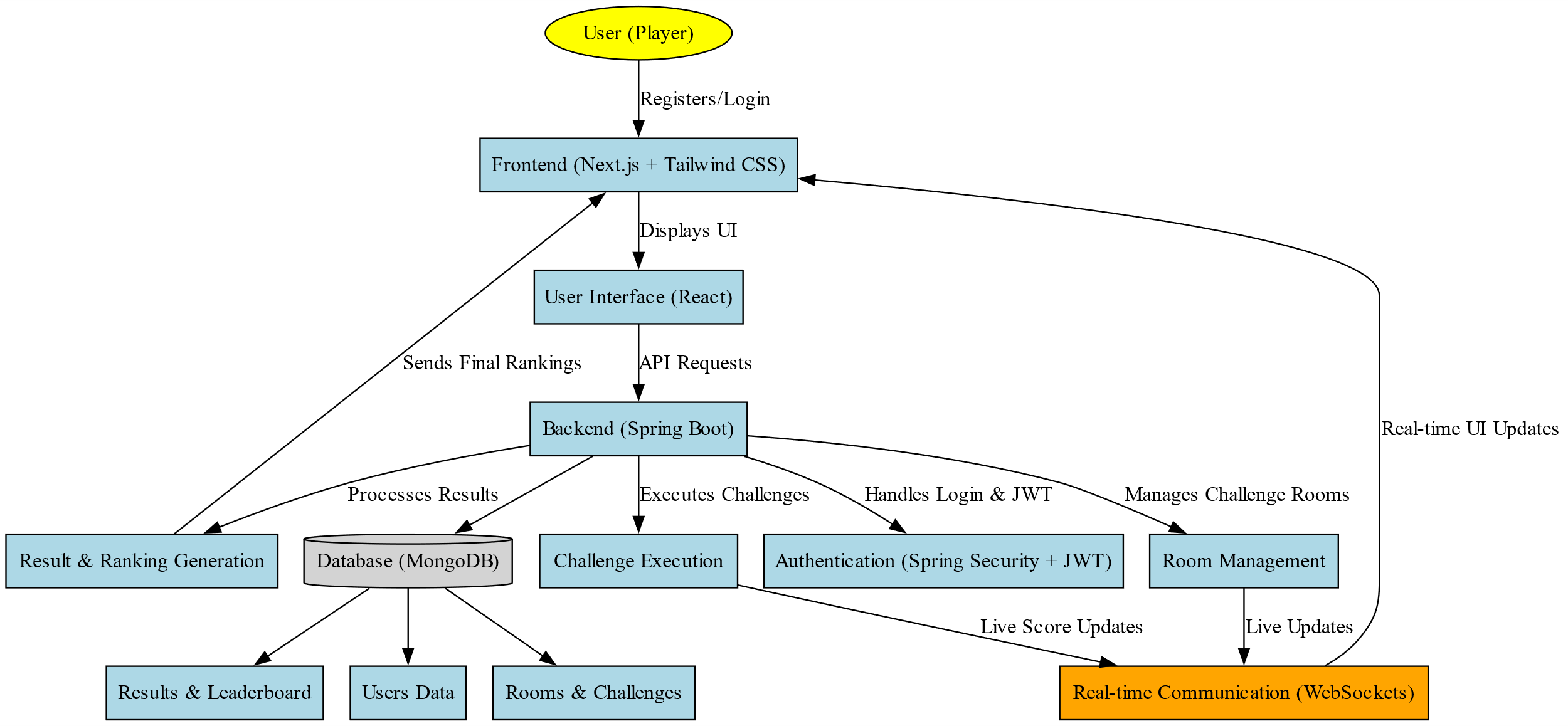
****

Figure – 3.1

3.1 Use Case Diagram



Figure – 3.2

3.2 Level 0 DFD

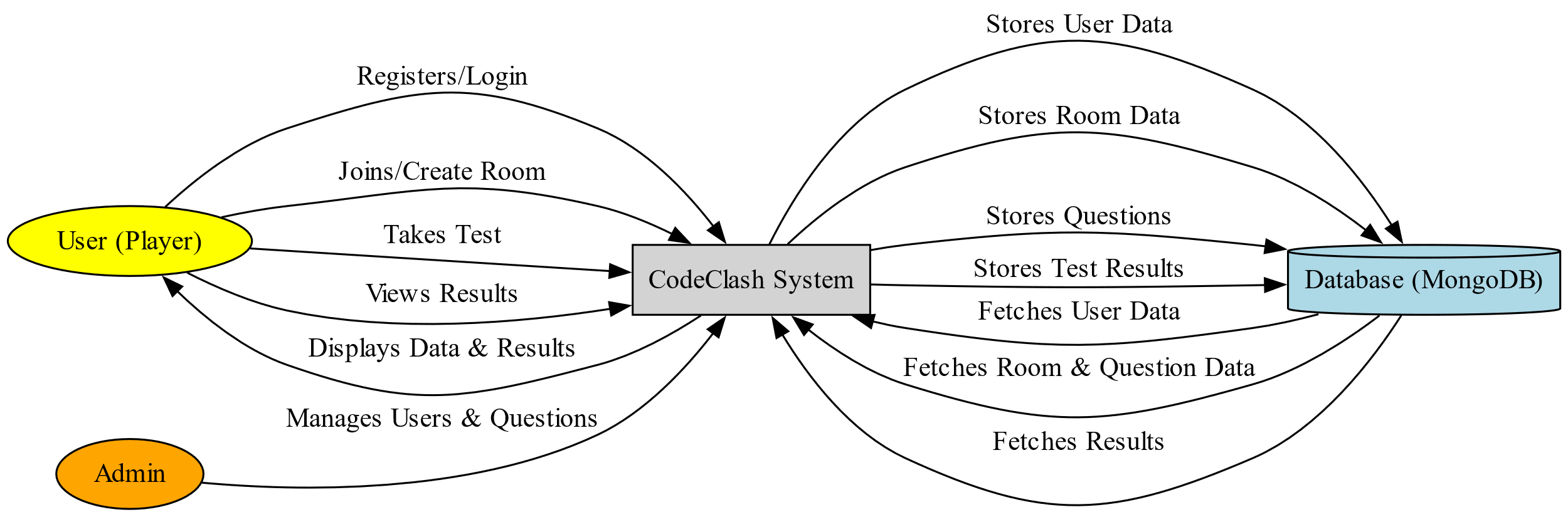


Figure – 3.3

3.3 Level 1 DFD

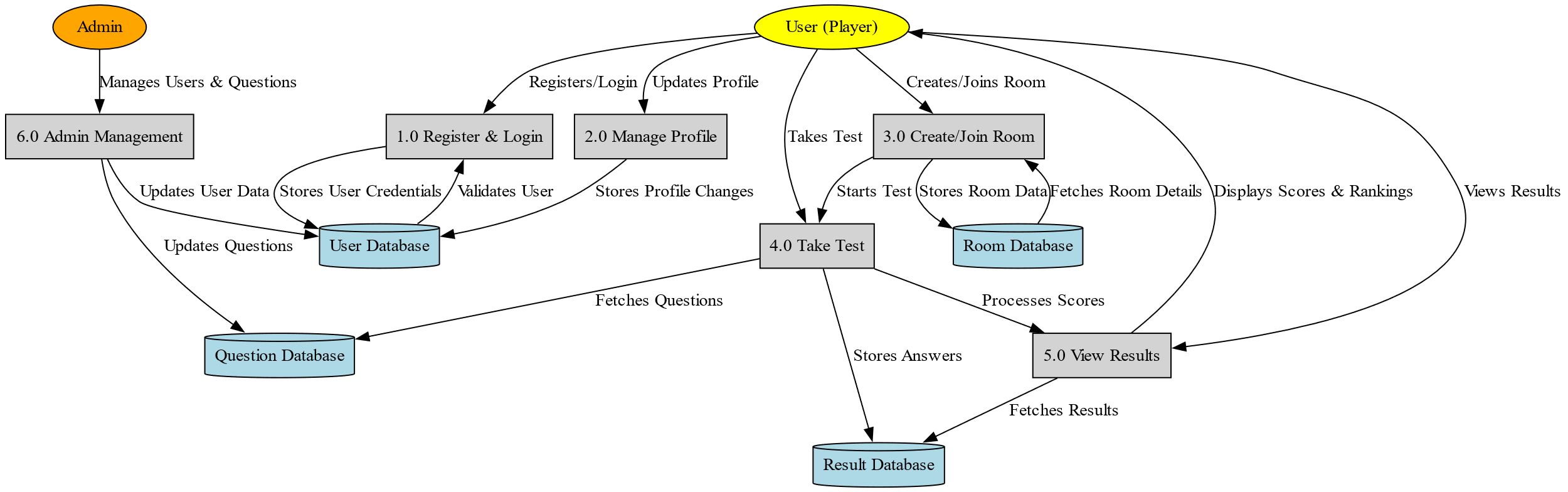


Figure – 3.4

3.4 Level 2 DFD

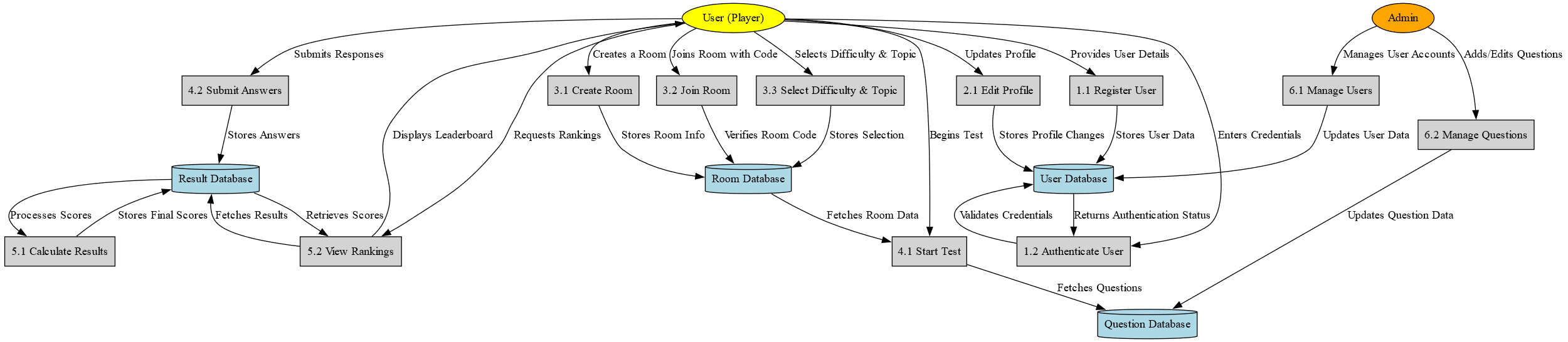


Figure – 3.5

3.2 ER Diagram

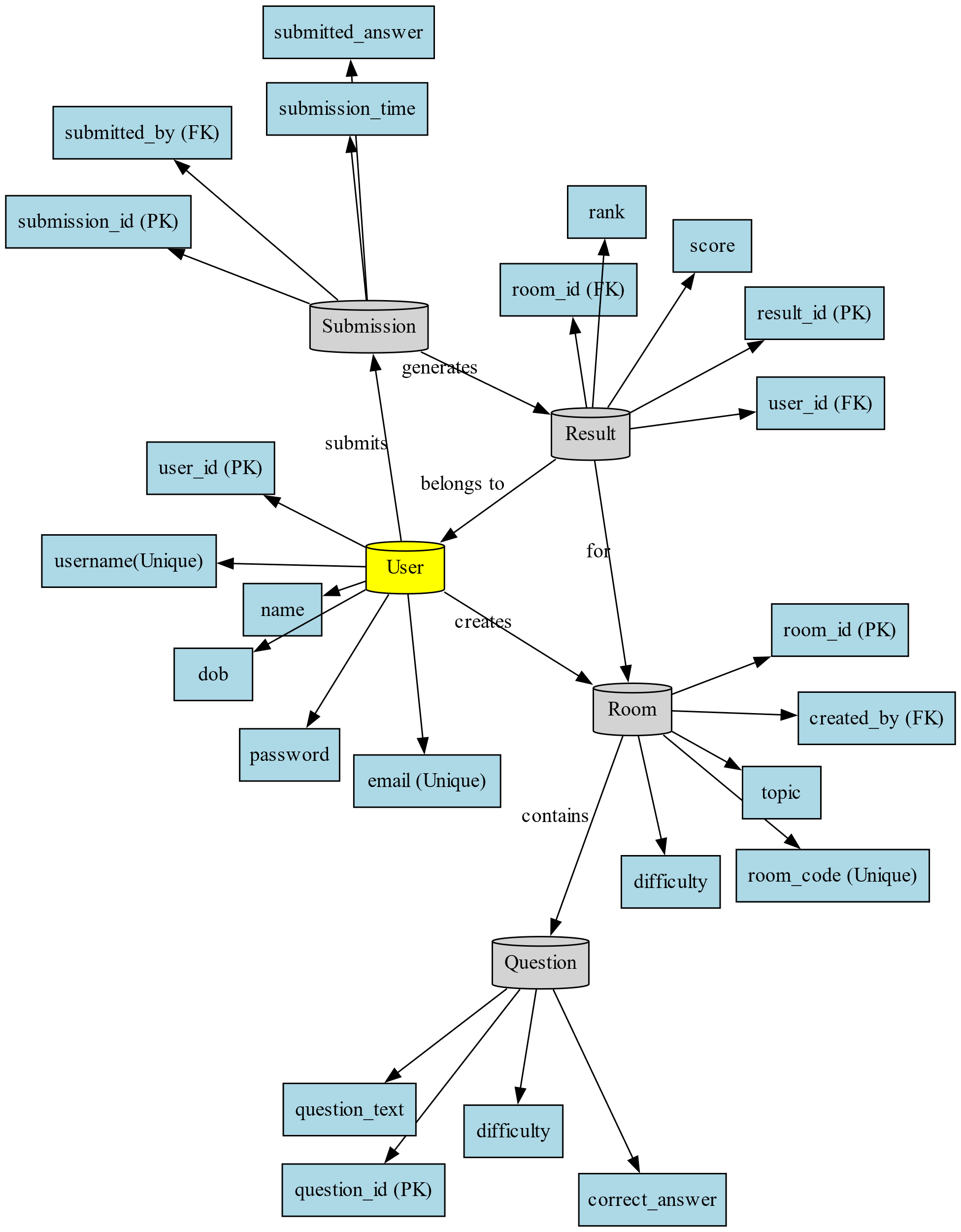


Figure – 3.6

**4. Database Tables for CodeClash**

4.1 Users Table

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| userId | String (PK) | Unique identifier for the user |
| name | String | Full name of the user |
| email | String | User's email (must be unique) |
| password | String | Hashed password |
| role | String | User role: STUDENT / ADMIN |
| dob | Date | Date of Birth |
| createdAt | Timestamp | Date and time the user was created |

4.2 Questions Collection

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| questionId | String (PK) | Unique ID for the question |
| quizId | String (FK) | ID of the quiz this question belongs to |
| question | Text | The question statement |
| optionA | String | Option A |
| optionB | String | Option B |
| optionC | String | Option C |
| optionD | String | Option D |
| correctAnswer | String | Correct option (A/B/C/D) |

4.3 QuizSubmissions Collection

| **Field Name** | **Type** | **Description** |
| --- | --- | --- |
| submissionId | String (PK) | Unique ID for the submission |
| quizId | String (FK) | ID of the quiz attempted |
| userId | String (FK) | ID of the user who submitted the quiz |
| totalScore | Integer | Total score achieved by the user |
| submittedAt | Timestamp | Time of submission |

4.4 Questions Table

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| QuestionID | String (PK) | Unique Question ID |
| Topic | String | Category or subject |
| Difficulty | String | Question difficulty level |
| Question | Text | The actual question |
| OptionA | String | Option A |
| OptionB | String | Option B |
| OptionC | String | Option C |
| OptionD | String | Option D |
| CorrectAns | String | Correct answer option (A/B/C/D) |

4.5 Submissions Table

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| SubmissionID | String (PK) | Unique submission ID |
| RoomID | String (FK) | Room where the test was taken |
| UserID | String (FK) | User who submitted answers |
| QuestionID | String (FK) | Question answered |
| SelectedAns | String | User’s selected answer (A/B/C/D) |
| IsCorrect | Boolean | Whether the answer was correct |

4.6 Results Table

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| ResultID | String (PK) | Unique Result ID |
| RoomID | String (FK) | Room where the test was taken |
| UserID | String (FK) | User who participated |
| Score | Integer | Score obtained |

**5. System Structure**

5.1 Overview

The system structure of CodeClash follows a client-server architecture where users interact with the platform via a web interface, and the backend processes requests, manages data, and ensures real-time interactions. The platform is designed using Spring Boot for the backend, Next.js for the frontend, MongoDB as the database, and WebSockets for real-time communication.

5.2 System Architecture

The CodeClash platform consists of the following layers:

5.2.1 Frontend Layer (Client-Side)

* Provides an intuitive user interface for registration, login, profile management, quiz creation, and result visualization.
* Implements WebSockets for real-time communication.

5.2.2 Backend Layer (Server-Side)

* Built using Spring Boot and Spring Security for authentication.
* Handles user management, room creation, question selection, and leaderboard generation.
* Uses WebSockets to handle multiplayer quiz sessions in real time.

5.2.3 Database Layer

* Uses MongoDB for storing user data, quiz questions, submissions, and results.
* The database schema ensures data integrity and quick access.

5.2.4 Communication Layer

* Uses REST APIs for standard HTTP requests.
* Implements WebSockets for real-time synchronization between players during the quiz.

5.2.4 System Flow

* User Authentication
* A user registers using email and password.
* After login, Spring Security generates a JWT token for authentication.
* The user is redirected to the home page.
* Room Creation & Quiz Setup
* Users create a quiz room by selecting a topic and difficulty level.
* A unique invitation code is generated for inviting friends.
* Peers enter the invitation code to join the same quiz session.
* Quiz Execution
* Once all users join, the quiz starts in real time.
* Questions are displayed one by one, and users submit answers.
* WebSockets handle synchronization, ensuring the same question appears for all users at the same time.

5.2.5 Result Processing

* After time runs out, the system evaluates the submissions.
* The leaderboard is generated based on correct answers and time taken.
* Users can view their ranks and scores.

**6. Implementation Phases**

Phase 1: Frontend Development

* User Registration & Authentication
* Developed the login and signup pages using Next.js.
* Used React Hooks and Tailwind CSS for UI components.
* Integrated JWT authentication for secure login/logout.
* Dashboard & Profile Management
* Implemented a dashboard displaying user details.
* Allowed users to edit name and DOB, but restricted email/username edits.

Phase 2: Backend Development

* User Authentication with Spring Security
* Implemented JWT-based authentication.
* Used Spring Security’s SecurityContextHolder for session management.
* Room & Quiz Management
* Created REST endpoints for creating, joining, and starting a quiz room.
* Used WebSockets to sync real-time quiz sessions.

Phase 3: Quiz Handling

* Implemented event-driven messages for quiz questions and answers.
* Calculated scores based on speed and accuracy.
* Stored rankings in the MongoDB database.

Phase 4: Testing & Deployment

* Unit & Integration Testing
* Tested API endpoints using Postman and JUnit.
* Debugged WebSocket communication using Spring WebSocket logs.

**Explanation:**

**Phase 1: Frontend Development**

User Registration & Authentication:

The user authentication module was built using Next.js and enhanced with React Hooks to manage state efficiently. Tailwind CSS was used to design responsive and clean UI components for both login and signup pages. JWT (JSON Web Tokens) were integrated to ensure secure authentication, enabling token-based session handling that eliminates the need for server-side sessions, improving scalability.

Dashboard & Profile Management:

A user-specific dashboard was created where each logged-in user can view and manage their profile. The interface was built using component-driven development in React. Users are allowed to edit only non-critical information like their full name and date of birth. Email and username fields are restricted to maintain account integrity and prevent impersonation.

**Phase 2: Backend Development**

User Authentication with Spring Security

The backend leverages Spring Boot along with Spring Security for managing authentication and authorization. JWT was used to authenticate requests. Each token is validated and processed via a filter chain. Spring Security’s SecurityContextHolder maintains session context across secure endpoints, ensuring each request is associated with the right authenticated user.

Quiz Management:

Initially, RESTful APIs were designed to manage quiz room lifecycle—creating, joining, and launching rooms. In earlier versions, WebSocket integration allowed real-time quiz synchronization, handling live interactions like joining events and quiz initiation. These features have since been deprecated in favor of manual test-based quiz attempts.

**Phase 3: Quiz Handling**

Question Presentation and Answer Evaluation:

Quiz questions are presented from a categorized problem set tailored to academic levels such as BTech and MCA. Users solve problems using Java and execute manual test cases. This approach supports test-driven development and encourages students to understand input/output thoroughly.

Score Calculation & Ranking:

Initially designed with speed-based scoring (now removed), the system now emphasizes accuracy. Each correct answer yields a specific score. Rankings are based on number of correct answers. All data including scores and problem attempts are stored in MongoDB, allowing persistent progress tracking.

**Phase 4: Testing & Deployment**

Unit & Integration Testing:

Comprehensive testing was conducted for each backend API using Postman and JUnit. This ensured the reliability of endpoints like registration, login, and quiz problem fetching. Earlier, WebSocket events were debugged using Spring’s built-in WebSocket tracing logs to ensure real-time messages worked correctly.

**7. Project Outcome**

1. **Categorized Learning Paths**

Users can select problems based on their academic level (e.g., BTech, MCA) and difficulty.

* Questions are organized into logical sections, enabling a focused and progressive learning experience.
* Helps students prepare according to their curriculum or knowledge level.

1. **Extensive Problem Set**

The platform includes a curated problem set that users can explore and solve.

* Covers a wide range of topics suitable for different computer science courses.
* Encourages self-paced learning by allowing users to choose what to solve and when.

1. Java-Based Code Execution Environment

Users can write and test Java solutions for selected problems within the platform.

* A custom Java execution engine runs code locally on the server using ProcessBuilder.
* This feature allows users to test their logic and receive output instantly.

1. Manual Test Case Execution

Users can run test cases manually and compare their output.

* Promotes critical thinking by allowing users to debug and verify their code against edge cases.
* Adds flexibility for students to create and test their own cases, just like in competitive programming platforms.

1. Secure User Accounts and Profiles

Users can register, log in, and track their progress through their profile.

* Basic authentication ensures user-specific access and tracking.
* Profile data may include name, level of study, and list of attempted problems.

# **8. Testing Phase for CodeClash**

* The testing phase ensures that CodeClash operates smoothly, providing a bug-free, secure, and efficient multiplayer quiz platform. The platform undergoes multiple testing methodologies, including unit testing, integration testing, functional testing, performance testing, and security testing.

## **Testing Methodologies:**

### 8.1 Unit Testing

* Test individual components (controllers, services, repositories) to ensure they function correctly.

### 8.2 Integration Testing

* Verify that different modules work together correctly.
* Tools Used: Postman (API Testing), Spring Boot Integration Testing

### 8.3 Functional Testing

* Ensure that CodeClash meets the functional requirements.

Approach:

* Manual testing of the entire user journey.
* Automated UI testing using Cypress for Next.js frontend.

### 8.4 Performance Testing

* Ensure CodeClash handles multiple users efficiently.

### 8.5 Security Testing

* Identify vulnerabilities in authentication, database access, and data security.

## 8.6 Bug Fixing & Debugging Process

* During testing, several issues were identified and resolved.

## 8.7 Deployment Readiness

* After successful testing, CodeClash was tested across different devices & browsers, optimized for scalability, and is now ready for deployment.

**9. Project Output**

**Functional Output:**

**User Registration and Authentication:**

Users are able to register and log in securely using JWT-based authentication. The frontend handles the form inputs, while the backend validates the credentials. On successful login, users are granted a token that enables access to protected routes such as the dashboard and problem-solving sections.

**Dashboard and Profile:**

Once logged in, users are directed to a dashboard that displays their profile details. Users can edit their name and date of birth, but are restricted from changing their email and username to maintain account integrity. This section serves as the entry point to access the problem sets and quiz features.

**Problem Set and Quiz Interface:**

Users can browse a categorized list of coding problems. Problems are segmented by academic levels (e.g., BTech, MCA) and topics (e.g., arrays, strings, recursion). Upon selecting a problem, users are provided with a problem description, input/output format, and sample test cases.

**Code Execution and Manual Testing**

The platform allows users to write Java code directly in the editor and run it against predefined sample test cases. This manual testing approach helps students understand the correctness of their logic without depending on real-time auto-evaluation. The results of the test cases are displayed immediately, allowing the user to refine and resubmit as needed.

**Score and Progress Tracking**

Upon successful submission, users receive feedback on their solution’s correctness. Scores are assigned based on correct outputs. All problem attempts, results, and scores are saved in MongoDB Atlas, allowing users to monitor their progress over time.

**Technical Output:**

**Frontend Output:**

Developed using Next.js and Tailwind CSS, the frontend is fully responsive and user-friendly. It ensures cross-device compatibility and provides intuitive navigation from registration to solving problems. The use of React Hooks and modular components ensures maintainability and scalability.

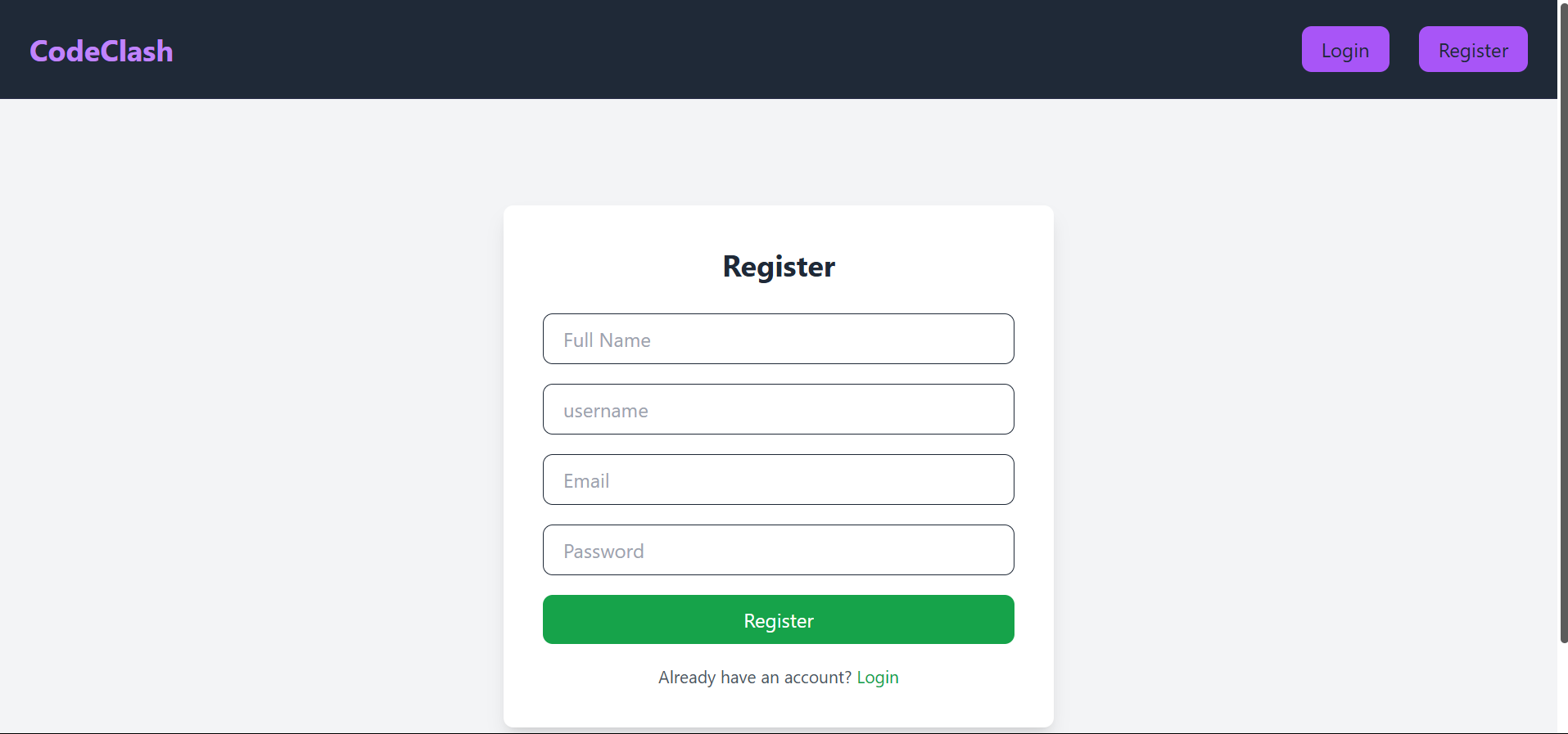
**Backend Output:**

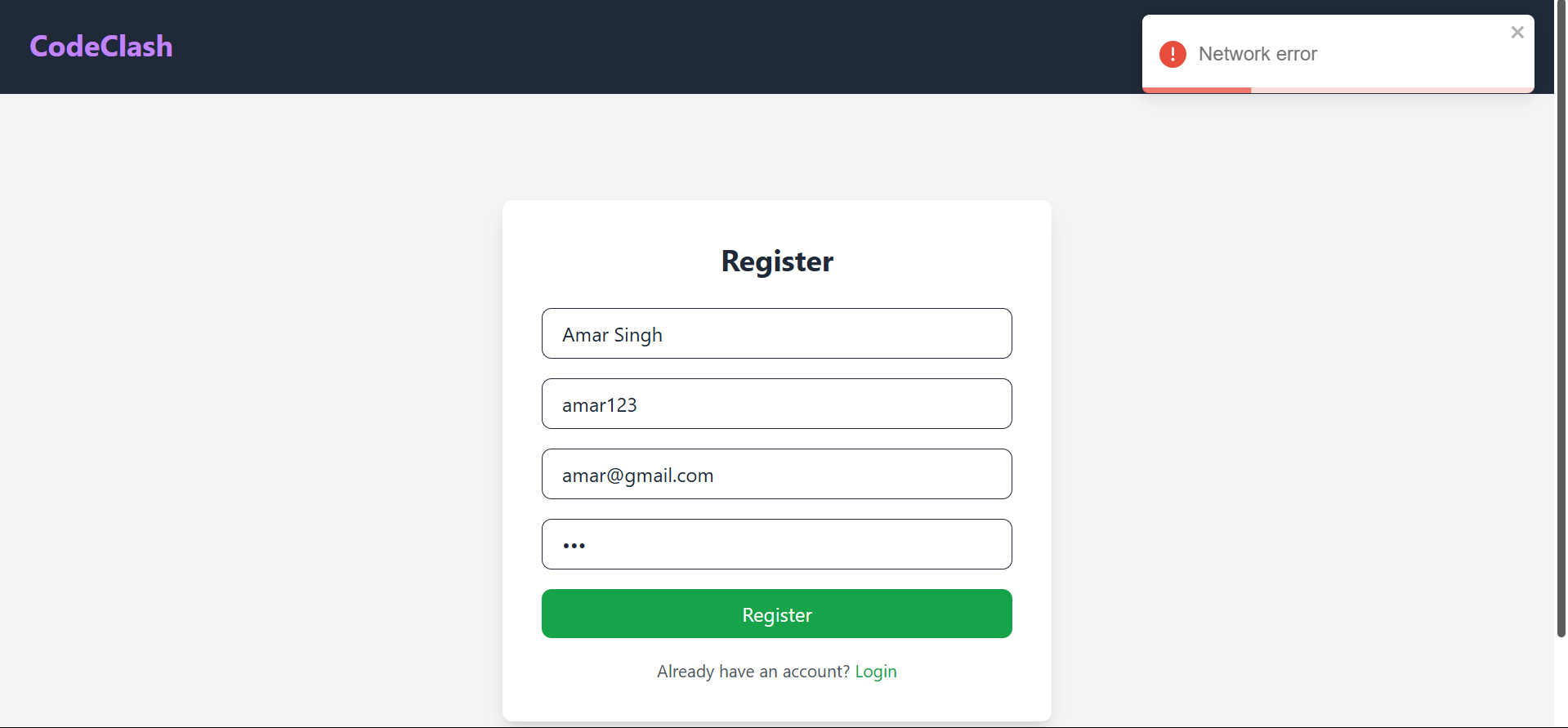
The backend, developed with Spring Boot and MongoDB, handles all data-related and authentication functionalities. JWT authentication is used to protect routes and ensure secure session management. REST APIs handle user actions like registration, login, profile update, and quiz problem management.

**Educational and Practical Impact:**

CodeClash not only strengthens students’ coding abilities but also teaches them how to manually debug and test their code. It serves as a virtual lab for practicing logic-building and preparing for technical interviews. Instructors can use it as a tool to assign problems, evaluate student performance, and track progress over the semester.

**10. Output**

****

****

**A screenshot of a login form

AI-generated content may be incorrect.**

**A screenshot of a login page

AI-generated content may be incorrect.**

**A screenshot of a login page

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a test

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a test

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a login page

AI-generated content may be incorrect.**

**11. Conclusion**

CodeClash successfully provides a real-time multiplayer quiz competition platform where users can challenge their peers in a structured and engaging environment. The project integrates modern web technologies, including Next.js for the frontend, Spring Boot for the backend, and MongoDB for data storage.

Throughout the development lifecycle, various aspects such as user authentication, real-time quiz battles, leaderboard ranking, and invitation-based room creation were implemented with a focus on security and performance. The use of Spring Security ensures data protection, while WebSockets enable real-time interactions among users.

Comprehensive testing, including unit, integration, functional, performance, and security testing, was conducted to ensure CodeClash operates smoothly under different conditions. The platform was optimized for scalability and responsiveness, making it ready for deployment.

In conclusion, CodeClash successfully bridges the gap between learning and competition by offering an interactive and engaging quiz platform. Future enhancements may include AI-based question recommendations, adaptive difficulty levels, and integration with external learning platforms to further enrich the user experience.

**12. References**

Spring Boot Documentation – <https://spring.io/projects/spring-boot>

MongoDB Documentation – <https://www.mongodb.com/docs/manual/>

Spring Security Documentation – Available at: <https://docs.spring.io/spring-security/reference/>

Postman API Testing – <https://learning.postman.com/docs/getting-started/introduction/>

ReactJs Documentation - <https://react.dev/learn>

Toastify Documentation - <https://www.npmjs.com/package/react-toastify>