VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590 018

A Mini - Project Report

On

"SMART QUIZ ARENA"

Submitted in partial fulfilment of the requirements for the **MINI PROJECT (BCD586)** course of the 5th semester

Bachelor of Engineering

In

Computer Science & Engineering (DATA SCIENCE)

Submitted by

Mr. Arun Kumar K S
(4AI23CD003)

Mr. C K Yeshas
(4AI23CD010)

Mr. Shashikumar T A
(4AI23CD049)

Mr. Sudeep S
(4AI23CD055)

Under the guidance of

Ms. HARSHITHA H D **B.E., M.Tech.**

Assistant Professor

Department of CS&E (DATA SCIENCE)
Adichunchanagiri Institute of Technology
CHIKKAMAGALURU - 577102

2025-26

ADHICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

Jyothinagar, Chikkamagaluru-577102

DEPARTMENT OF CS&E (DATA SCIENCE)

CERTIFICATE

This is to certify that the Mini project work entitled **"SMART QUIZ ARENA"** is a bonafied work carried out by **Mr. Arun Kumar K S (4AI23CD003), Mr. C K Yeshas (4AI23CD010) Mr. Shashikumar T A (4AI23CD049), Mr. Sudeep S (4AI23CD055),** in partial fulfillment for the **Mini Project (BCS586)** course of 5th semester Bachelor of Engineering in **Computer Science and Engineering (Data Science)** of the Visvesvaraya Technological University, Belagavi during the academic year **2025-2026**. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The Mini project report has been approved as it satisfies the academic requirements in respect of Project Work prescribed for the said Degree.

Signature of the Guide
Ms. Harshitha H D B.E., M.Tech
Assistant Professor

Signature of Coordinator
Mrs. Shilpa K V. B.E., M.Tech
Assistant Professor

Signature of the HOD
Dr. Adarsh M J B.E., M.Tech., Ph.D
Associate Professor and Head

ABSTRACT

Accurate and efficient assessment plays an important role in learning and systematic evaluation. Traditional quiz methods often rely on manual preparation, paper-based tests, or limited digital tools, which makes the process slow and difficult to maintain. Smart Quiz Arena overcomes these limitations by providing a structured, automated, and easy-to-use platform for delivering Multiple Choice Question (MCQ) based quizzes. The system enables quick quiz generation, organized question management, and instant score calculation, reducing human effort and ensuring consistent evaluation for all learners.

Smart Quiz Arena offers students secure login, question banks, timed MCQ quizzes, scoring, and clear result reports. The system provides students with an easy-to-use interface for attempting quizzes, while questions are managed through the backend and AI-generated logic. Developed using Django, SQLite, and standard web technologies, the platform offers smooth performance, secure data handling, and easy maintenance. By automating quiz delivery and evaluation, Smart Quiz Arena reduces manual work and enables a structured and reliable assessment process.

ACKNOWLEDGEMENTS

We express our humble pranamas to his holiness **Divine Soul Parama Poojya Jagadguru Padmabushana Sri Sri Sri Dr.Balagangadharanatha Maha Swamiji** and **Parama Poojya Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Maha Swamiji Pontiff, Sri Adichunchanagiri Maha Samsthana Matt and Sri Sri Gunanatha Swamiji,** Chikkamagaluru branch, Sringeri who have showered their blessings on us.

The completion of any project involves the efforts of many people. We have been lucky enough to have received a lot of help and support from all quarters during the making of this project, so with gratitude, we take this opportunity to acknowledge all those whose guidance and encouragement helped us emerge successful.

We express our gratitude to **Dr. C K Subbaraya**, Director, Adichunchanagiri Institute of Technology.

We express our sincere thanks to our beloved principal**, Dr. C T Jayadeva** for having supported us in our academic endeavors.

We are thankful to the resourceful guidance, timely assistance and graceful gesture of our guide **Ms. Harshitha H D,** Asst. Professor, Department of CS&E (DATA SCIENCE), who has helped us in every aspect of our project work.

We thank our project coordinator **Mrs. Shilpa K V,** Asst. Professor, Department of CS&E (DATA SCIENCE), for her lively correspondence and assistance in carrying on with this project

We are also indebted to **Dr. Adarsh M J,** HOD of CS&E (DATA SCIENCE) Department, for the facilities and support extended towards us.

We would be very pleased to express our heart full thanks to all the teaching and non-teaching staff of CS&E (DATA SCIENCE) Department and our friends who have rendered their help, motivation and support.

Arun Kumar K S

C K Yeshas

Shashikumar T A

Sudeep S

CONTENTS

ABSTRACT	**i**
ACKNOWLEDGEMENTS	**ii**
CONTENTS	**iii**
LIST OF FIGURES	**v**
LIST OF TABLES	**vi**
LIST OF SNAPSHOTS	**vii**
REFERENCES	

CHAPTERS	**PAGE NO**
1. Introduction	**01**
1.1 Background	01
1.2 Problem Statement	01
1.3 Objectives of the system	02
1.4 Significance of the system	02
1.5 Scope of the project	03
1.6 Methodology	03
1.7 Target Audience	04
1.8 Overview of the report	04
2. System Design	**05**
2.1 System Architecture	05
2.2 Module Design	06
2.3 Database Design	06
2.4 User Interface Design	07
2.5 Technology Stack	07
3. Implementation	**08**
3.1 Backend Implementation	08
3.2 Frontend Implementation	09
3.3 AI Module Implementation	10
3.4 Database Implementation	10
4. Testing	**11**
4.1 Testing Objectives	11
4.2 Testing Environment	11

4.3 Type of testing 11
4.4 Test Cases 13
5. Results and Discussions 14
6. Conclusion and Future Enhancement 22

List of Figures

Sl. No	Description	Page No
1.8.1	Smart Quiz Arena	4
2.1.1	System Architecture	5

List of Tables

Sl. No	Description	Page No
4.4.1	Test Cases	13

List of Snapshots

Sl. No	Description	Page No
5.1.1	Login and Registration Screen	14
5.1.2	Home	15
5.1.3	Single Player mode	15
5.1.3.1	Question in Single Player mode	16
5.1.3.2	Result Window	16
5.1.4.1	Multiplayer Mode	17
5.1.4.2	Result	17
5.1.5.1	Coding Challenge mode	18
5.1.5.2	Coding in two different browsers	18
5.1.5.3	Result	19
5.1.6	Leaderboard	19

Chapter 1

INTRODUCTION

1.1 Background

Assessments are an essential part of the learning process, as they help measure understanding, track progress, and identify areas where students need improvement. In many institutions, quizzes are still conducted through traditional means such as printed papers, manual checking, or basic digital forms.

- **Context:** Educational environments commonly rely on scattered question papers, informal storage methods, or simple tools that are not designed for systematic quiz management. As the number of students increases, preparing and evaluating quizzes becomes more difficult.
- **Problem:** Manual handling of MCQ-based quizzes leads to mistakes, delays, and unnecessary workload. Most existing systems do not offer quick quiz creation, structured question banks, or automatic scoring. Students often have no proper record of previous attempts, and existing systems do not manage or update questions efficiently
- **Opportunity:** With the availability of modern web frameworks, secure databases, and automated scoring mechanisms, it is now possible to build a complete system for conducting MCQ quizzes online. Such a platform can simplify quiz creation, deliver questions in a clear format, and provide instant results. Smart Quiz Arena uses these technologies to create a reliable and easy-to-use quiz environment for students , making the assessment process more structured and efficient.

1.2 Problem Statement

- **Overview of the Problem**: Traditional quiz methods used in many institutions depend on manual preparation, scoring, and record-keeping. This leads to delays, inconsistencies, and extra workload for teachers. Students also do not receive immediate feedback, and there is no simple way to manage or review quiz performance. Without a structured system, conducting MCQ-based assessments becomes slow, difficult to maintain, and less effective for both teaching and learning.

- **Specific Issues**:

    - Difficulty in maintaining and organizing large collections of MCQ questions.
    - Time-consuming quiz preparation and evaluation processes.
    - Inaccuracies caused by manual scoring and data entry.
    - Lack of instant result generation and performance reports

1.3 Objective of the System

The primary objective of the Smart Quiz Arena system is to provide an automated, reliable, and easy-to-use platform for conducting MCQ-based quizzes. The system aims to streamline quiz creation, delivery, and evaluation while reducing manual effort and improving overall accuracy in assessments.

- **Key Goals**:
    - **Ease of Use:** Provide automatically generated MCQ quizzes and allow students to attempt them through a clean and user-friendly interface.
    - **Accuracy:** Ensure that quiz evaluation is handled automatically, eliminating mistakes that occur in manual scoring.
    - **Instant Results:** Provide immediate score generation and performance summaries for students after completing a quiz.
    - **Security and Privacy:** Protect user data, quiz questions, and results through secure authentication and controlled access.
    - **Scalability:** Support large numbers of users, questions, and quizzes without affecting system performance.
    - **User Access Control:** : Students can log in securely, access single-player quizzes, join multiplayer rooms, and participate in coding challenges based on their verified account

1.4 Significance of the System

- **Efficiency:** By automating quiz creation, delivery, and evaluation, the system reduces the manual workload for teachers and simplifies the entire assessment process.
- **Accuracy:** Digital scoring eliminates common errors found in manual correction, ensuring that student results are reliable, consistent, and easily verifiable.
- **Immediate Feedback:** Students receive their quiz results instantly, allowing them to understand their performance and identify areas that require improvement.
- **Performance Insights:** The system can generate reports that highlight student progress, strengths, and weaknesses, helping teachers plan appropriate learning interventions.
- **Cost-Effective:** By reducing the need for printed question papers and manual record-keeping, the platform lowers long-term operational costs for institutions.

1.5 Scope of the Project

- **In Scope**:
    - Development of a web-based platform for conducting MCQ-based quizzes.
    - Features for generation of AI-generated questions Automatic scoring and result generation.
    - Automated scoring and generation of quiz reports for students .
    - Secure storage of questions, quiz attempts, and scores in the database.
- **Out of Scope**:
    - Advanced analytics such as detailed skill profiling unless added separately.
    - Integration with third-party learning management systems (LMS) unless specified later.
    - Features like subjective question evaluation or manual grading.

1.6 Methodology

- **Approach**: The system will be built as a web-based application using modern web technologies such as HTML, CSS, and JavaScript for the interface. Django will be used as the backend framework, and SQLite will serve as the database for storing quiz data, questions, and user information. The platform will be designed to allow smooth communication between the frontend and backend using REST APIs.
- **Agile Development**: The system will follow an **Agile development** methodology, involving iterative design and feedback cycles to ensure that the system meets the needs of users at each stage of development.
- **Testing**: Testing will involve unit tests for backend logic, integration tests to verify data flow between components, and user acceptance testing to ensure that students find the system reliable and easy to use. The goal is to validate core functionalities such as quiz creation, timing, scoring, and result generation.

1.7 Target Audience

- **Students**: Students will attempt quizzes, view their scores instantly, and track their progress over time through the dashboard.
- **Participants**: Anyone with an account can compete and view results.
- **Teachers**: Update the coding challenge problems in database locally.

1.8 Overview of the Report

This report is organized into several chapters that explain the design, development, and evaluation of the Smart Quiz Arena system. Each chapter covers a different aspect of the project, ensuring a complete understanding of the system's structure and functioning.

- **Chapter 1: Introduction** – Gives a brief introduction of the project.
- **Chapter 2: System Design** – Explains the overall architecture of Smart Quiz Arena, including module design, technology stack, and database structure.
- **Chapter 3: Implementation** – Discusses the development process, backend logic, frontend interfaces, and the integration of system components.
- **Chapter 4: Testing and Validation** – Describes the testing methods, test cases, and validation of system performance.
- **Chapter 5: Results and Discussions** – Presents the outcomes of the system, observations, and the limitations identified during development.
- **Chapter 6: Conclusion and Future Enhancements** – Summarizes the project's contributions and outlines potential improvements for future versions.

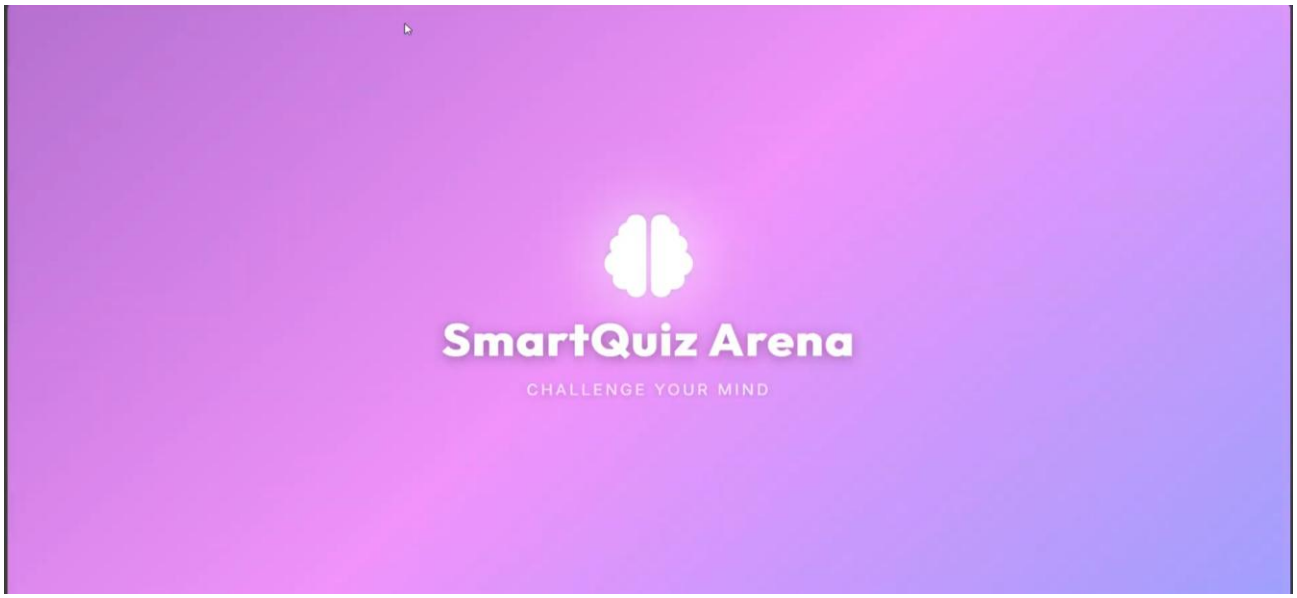

Fig 1.8.1: Smart Quiz Arena

Chapter 2

SYSTEM DESIGN

2.1 System Architecture

This chapter describes the technical design of the Smart Quiz Arena system, explaining its architecture, core components, and how they work together to create, deliver, and evaluate MCQ-based quizzes. The design focuses on providing accurate scoring, smooth user experience, secure data management, and efficient handling of quiz-related operations.

- **High-Level Overview:**

  The system follows a client–server structure where users access the quiz platform through a web interface. The backend handles requests, applies quiz logic, and communicates with the database to store and retrieve questions, quiz attempts, and results.
- **Architecture Diagram:**

  A simple diagram should represent the main components: the frontend interface, the backend server, and the database.
- **Components:**
  - **Frontend:** A web interface where students access quiz modes, attempt MCQ quizzes, participate in multiplayer sessions, and view results
  - **Backend Server:** Handles user authentication, manages quiz sessions, synchronizes multiplayer modes, processes submissions, and generates results
  - **Database:** Stores user accounts, quiz sessions, coding challenge problems, submitted answers, and score records used for evaluating student performance.

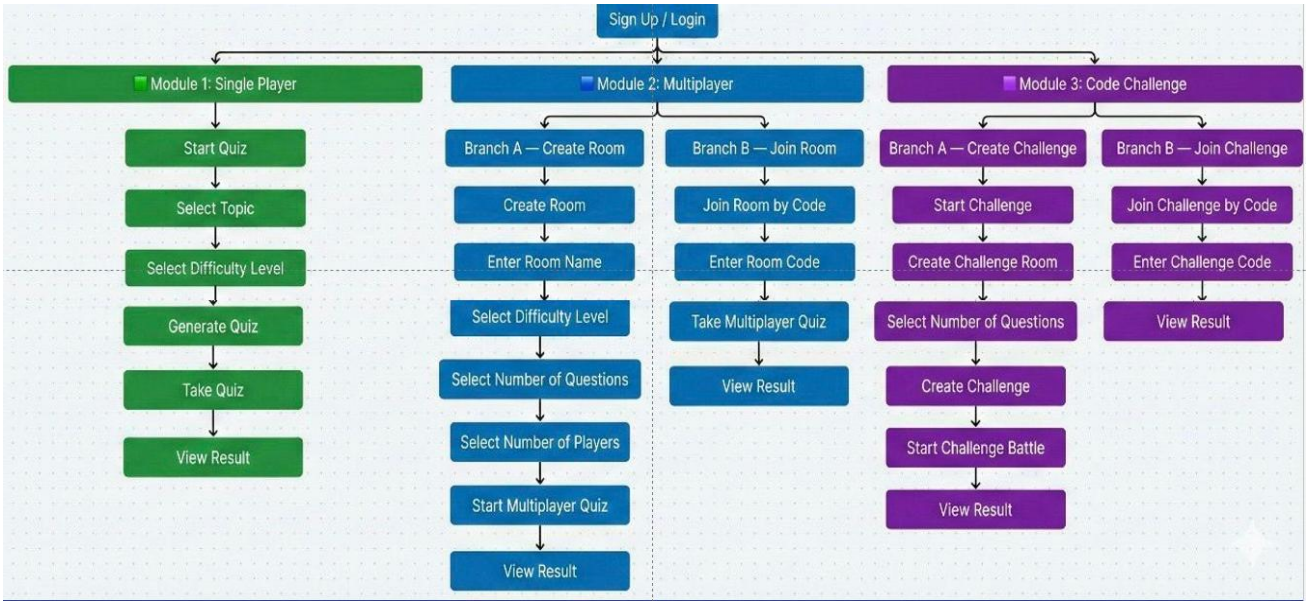

Fig 2.1.1: System Architecture

2.2 Module Design

The Smart Quiz Arena system is organized into several modules that work together to support quiz delivery through a clean and API-driven structure. Each module performs a specific function, ensuring that quizzes load correctly, run smoothly, and return accurate results to the user. The modular approach keeps the system easy to maintain and allows different components to operate in a dependable and coordinated manner.

2.2.1 User Authentication Module

Login via:

- Username
- Password

Stores credentials in SQLite.

2.2.2 Question Retrieval Module

Retrieves MCQs by:

- Selected Topic
- Difficulty level
- Random selection from the database

Provides questions to the quiz interface.

2.2.3 Quiz Execution Module

Handles:

- Display of MCQs
- Timer control
- Collecting selected options
- Submitting answers for evaluation

2.2.4 Result Generation Module

Generates:

- Instant quiz score
- Correct checked with selected answers
- Basic performance summary

2.3 Database Design

The database for Smart Quiz Arena is built using SQLite, which provides a simple, lightweight, and efficient structure suitable for storing quiz questions, user details, and quiz attempts. The database is organized into multiple tables, each responsible for managing a specific type of information within the system.

Table overview:

- **Users**: Stores student login details along with profile information such as total score, games played, win rate, and avatar.
- **Questions**: Contains MCQ questions, options, correct answers, difficulty levels, categories, and related metadata.
- **Quiz Sessions**: Holds information about each quiz session, including session type, status, time limits, and linked questions.
- **Player Scores**: Stores the student's quiz performance, including score, correct answers, total attempts, and activity timestamps.
- **Session Questions**: Maintains the order and mapping of questions assigned to each quiz session.

2.4 User Interface (UI) Design

The user interface of Smart Quiz Arena is designed to be simple and easy to use so that students can focus on attempting quizzes without confusion. The layout is kept clean, with clear navigation and minimal distractions.

Key Screens:

- **Login Screen**
  Allows students to enter their username and password to access the quiz system.
- **Home**
  Displays the available quizzes that a student can attempt, along with basic details such as quiz name and duration.
- **Quiz-Screen**
  Shows the MCQ questions one by one options for each question, and a timer for the quiz duration. Provides buttons to move between questions and to submit the quiz.
- **Result-Screen**
  Displays the student's score immediately after submission. It may also show which questions were answered correctly or incorrectly, depending on the design.
- **Dashboard**
  Allows students to view their past quiz attempts and scores to track their progress over time.

2.5 Technology Stack

- **Frontend**: HTML, CSS, JavaScript, or frontend frameworks for user interface.
- **Backend**: Django for business logic and server-side processing.
- **Database**: SQLite for reliable data storage and retrieval.

Chapter 3

IMPLEMENTATION

This chapter outlines the steps taken to implement the Smart Quiz Arena system, covering the backend logic, frontend interface, database structure, and integration processes. It explains the technologies used, the organization of the codebase, and the key development techniques applied to ensure smooth quiz delivery and accurate result generation.

3.1 Backend Implementation

The backend was developed as a REST API to handle quiz operations, communicate with the frontend, and interact with the SQLite database. Each endpoint manages a specific part of the quiz flow, including authentication, question retrieval, quiz sessions, and result processing.

3.1.1 Authentication

- **POST /login –** Authenticates students using username and password.
- **POST /register –** Allows creation of new student accounts.
- **Session handling –** Tracks active quiz sessions for each logged-in student.

3.1.2 Question Retrieval

GET /questions/<quiz_id> – Fetches all MCQ questions assigned to a quiz.

3.1.3 Quiz Session Management

POST /start-quiz

- Creates a new quiz session for a student.
- Initializes timer and question sequence.

POST /submit-quiz

- Receives selected answers.
- Performs auto-evaluation and stores the score.

GET /session/<session_id>

- Returns current quiz state to handle refresh or interruptions.

3.1.4 Result & Performance Services

- **GET /results/<session_id> –** Returns quiz score and evaluation summary.
- **GET /history/<user_id> –** Retrieves past quiz attempt.
- **Score calculation:** Processes correct vs. selected answers and generates summary.

3.1.5 Security

- Password hashing is handled by Django's built-in authentication.
- Access is restricted to logged-in users for all quiz-related endpoints.
- Database protection is ensured through model-level validation and controlled queries.

3.2 Frontend Implementation

The frontend provides the user interface for students to interact with the system.

3.2.1 Web Interface

Developed with HTML, CSS, and JavaScript:
• Login screen for student authentication
• Quiz selection screen showing available quizzes
• MCQ quiz interface with timer and navigation
• Result display panel showing score and evaluation

3.2.2 Device Compatibility

• The system is currently tested and functional on two laptops within the same network.
• Frontend screens run in a web browser and interact with the backend through normal HTTP requests.
• Currently optimized for desktop browsers; mobile optimization can be added in future versions.
• Multi-device support can be added in future improvements.

3.2.3 User Screens

• **Students:**
– Login and start quizzes
– Answer MCQs with live timer
– View quiz results and attempt history
– Can view quiz instructions or home page
– Must log in to attempt quizzes

3.2.4 UI Features

• Clean and minimal layout to reduce distractions during quizzes.
• Visual timer and progress indicators for better clarity.
• Error handling for network interruptions and answer submission.
• Consistent theme across all pages for smooth user experience.

3.3 Database Implementation

SQLite Tables

- **CustomUser** → stores student login and profile details
- **Question** → stores MCQ questions, options (JSON), difficulty, and correct answers
- **QuizSession** → records quiz sessions, status, timestamps, and settings
- **SessionQuestion** → links sessions with questions and preserves question order
- **PlayerScore** → stores scores, correct answers, and performance statistics

• Schema Features

- Relational structure managed through Django.
- Foreign key references to maintain connections between sessions, questions, and scores
- JSON fields used for storing answer options and other dynamic data
- Time-stamped entries for quiz creation, attempts, and activity tracking

Chapter 4

TESTING

This chapter covers the testing processes and methodologies applied to the Smart Quiz Arena system. Testing ensures that all components function correctly, that quizzes run smoothly from start to finish, and that the system behaves reliably under different usage conditions. It also helps identify errors, validate core features, and confirm that the system meets both functional and performance requirements.

4.1 Testing Objectives

• Verify that all quiz features—login, question retrieval, quiz sessions, and result generation—function correctly.
• Ensure that students can access quizzes, submit answers, and view results without encountering errors.
• Test the accuracy of score calculation, question loading, and session tracking.
• Confirm that the system handles invalid inputs, incorrect login attempts, and broken requests safely.
• Evaluate the stability and performance of the system when multiple quiz attempts are made in succession.

4.2 Testing Environment

- **Hardware**: Laptop/PC with at least 4GB–8GB RAM and a multi-core processor. The system was tested using two laptops connected on the same local network.
- **Software**:
    - **Backend:** Django development server running locally.
    - **Frontend:** HTML, CSS, and JavaScript served through a local browser.
    - **Database:** SQLite used for storing users, questions, quiz sessions, and scores.
    - **Testing Tools:** Django's built-in test framework for backend validation.
- **Browser**: Google Chrome and Microsoft Edge for cross-browser testing.
- **Operating Systems:** Windows 11 used for development and testing.

4.3 Types of Testing

4.3.1 Unit Testing

- **Objective**: To test individual components or functions in isolation to verify their correctness.
- **Tools**: Django's built-in unit testing framework for backend logic and simple JavaScript-based checks for frontend behavior.

Example Test Cases:

- **User Authentication:** Verifies that the login function correctly validates student credentials and rejects invalid inputs.
- **Question Loading:** Ensures that MCQ questions are retrieved correctly from the database.
- **Score Calculation:** Confirms that the scoring function evaluates answers accurately and returns the correct total.

4.3.2 Integration Testing

- **Objective**: To test how different modules interact, such as the communication between the frontend and backend or between backend and database.
- **Example Test Cases**:

  - **Quiz Submission:** Checks that the frontend sends selected answers to the backend and that the backend stores and processes them correctly.
  - **Result Retrieval:** Ensures that the result module correctly retrieves stored quiz performance data and displays it to the user.
  - **Question Flow:** Confirms that quizzes load questions in the correct sequence and that session tracking works as expected.

4.3.3 Functional Testing

- **Objective**: To validate the system against functional requirements and ensure it meets the needs of the user.
- **Test Scenarios**:

  - **Login:** Tests the student login process and ensures that only valid users can access quizzes.
  - **Quiz Attempt:** Verifies that students can start a quiz, answer MCQs, submit responses, and receive accurate results.
  - **Result View:** Checks that students can view their scores and that displayed data matches stored results.

4.4 Test Cases

Below are sample test cases for various components:

Table 4.4.1: Test Cases

Test Case ID	Description	Test Steps	Expected Result	Status
TC-001	Login with valid credentials	Enter valid username and password → click "Login"	User is redirected to home screen and session starts	Pass
TC-002	Login with invalid credentials	Enter wrong username or password → click "Login"	Error message shown; login fails	Pass
TC-003	Load MCQ questions for single-player quiz	Start a quiz → frontend requests questions from backend	Questions load correctly from DB	Pass
TC-004	Start multiplayer quiz session	Two systems join same session → backend updates player list	Both players successfully enter the same session	Pass
TC-005	Sync questions for multiplayer quiz	Session starts → each laptop requests the next question	Both systems display the same question in the same order	Pass
TC-006	Submit MCQ quiz (single-player)	Student submits selected answers	Score is calculated and stored; result page shown	Pass
TC-007	Coding challenge submission	Player writes code → submits to backend	Code stored; status updated (pending, passed, failed, error)	Pass
TC-008	Validate coding test cases	System executes submissions against test cases	Correct pass/fail count returned	Pass
TC-009	Multiplayer coding challenge	Both players submit code → backend records both submissions	Winner is determined based on passed tests or performance	Pass
TC-010	Scoreboard update	After coding/MCQ completion → backend updates PlayerScore	Scores and statistics update correctly	Pass

Chapter 5

RESULTS AND DISCUSSION

This chapter summarizes the results of the Smart Quiz Arena project, discussing its effectiveness, reliability, and alignment with the intended objectives. The chapter also covers the challenges encountered during development, observations gathered during testing, and recommendations for future improvements.

5.1 Result

5.1.1 Login Page

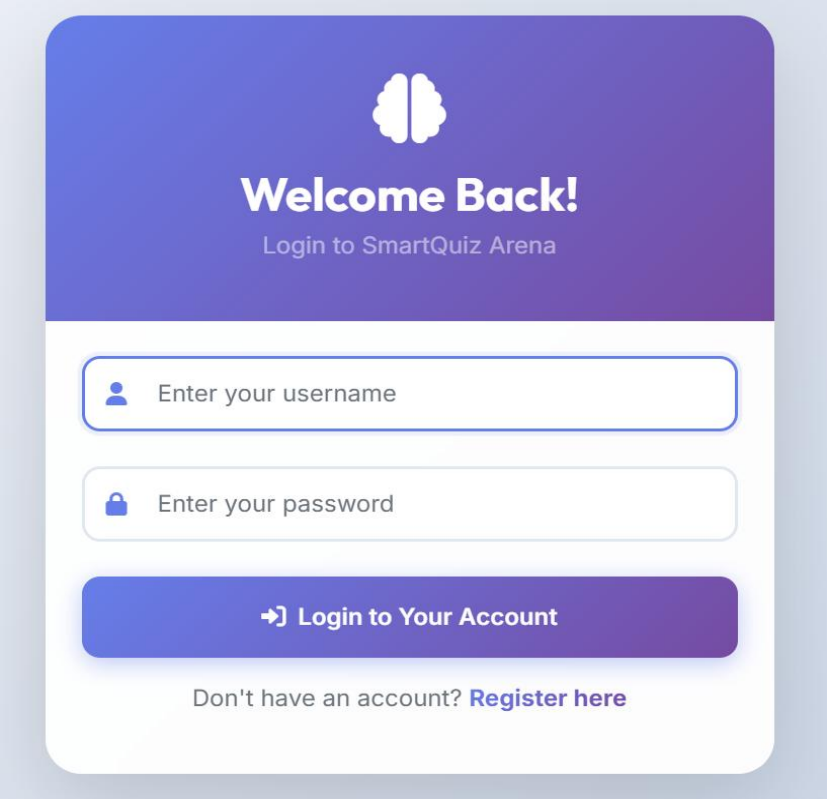

Fig 5.1.1: Login page

- This module validates the user's credentials by sending the entered username and password to the Django backend.
- The backend checks the data against the CustomUser table stored in SQLite.
- If authentication succeeds, a session is created and stored, allowing secure access to the system.
- Invalid login attempts are rejected using backend validation rules.

5.1.2 Home Page

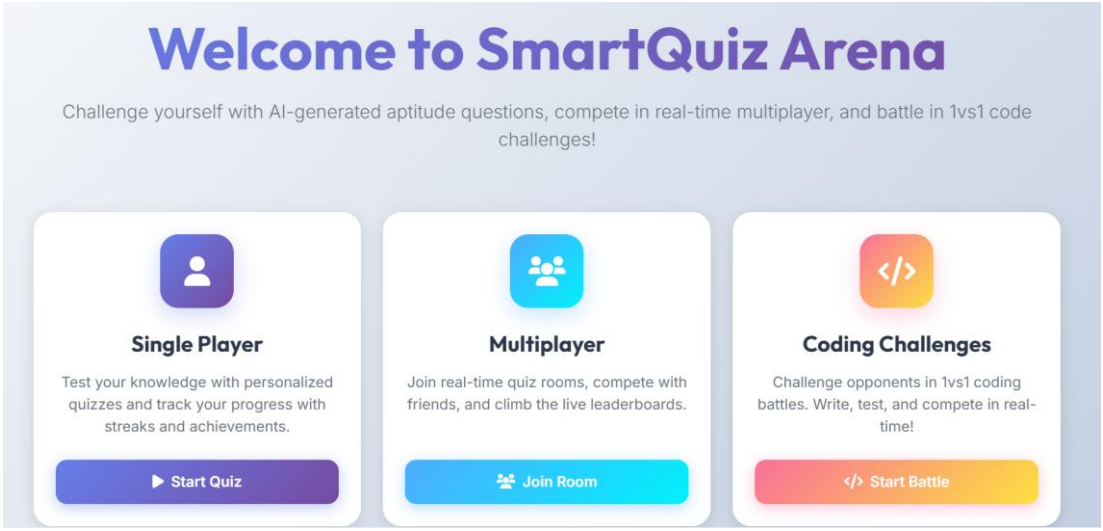

Fig 5.1.2: Home Page

- The home page displays all available quiz modes after confirming the user's active session.
- Data about the user's profile and quiz history is fetched from the CustomUser and PlayerScore tables.
- This interface acts as the main navigation layer and sends requests to the backend when a quiz mode is selected.

5.1.3 Single Player mode

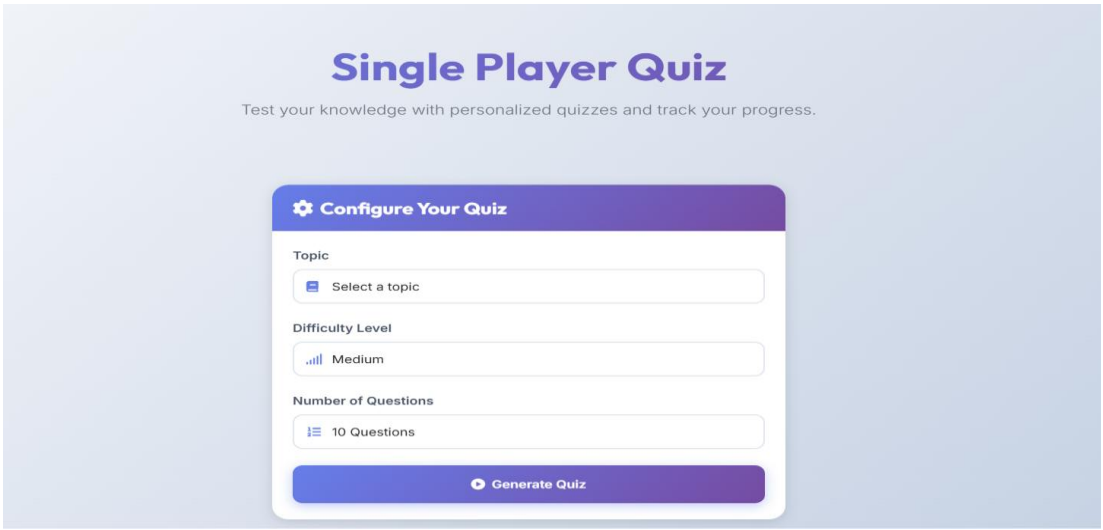

Fig 5.1.3: Single Player Mode

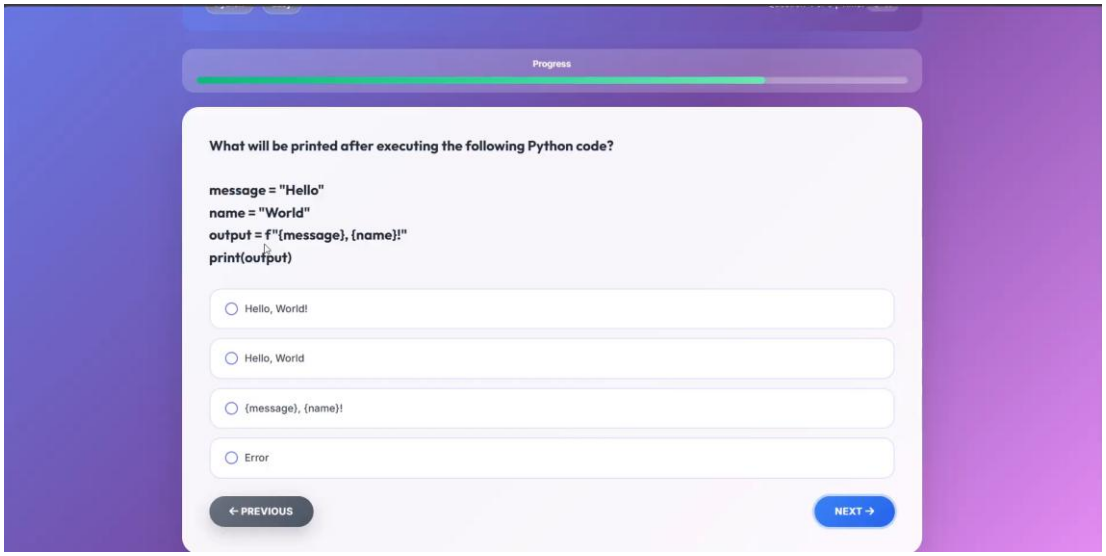

Fig 5.1.3.1: Question in Single Player mode

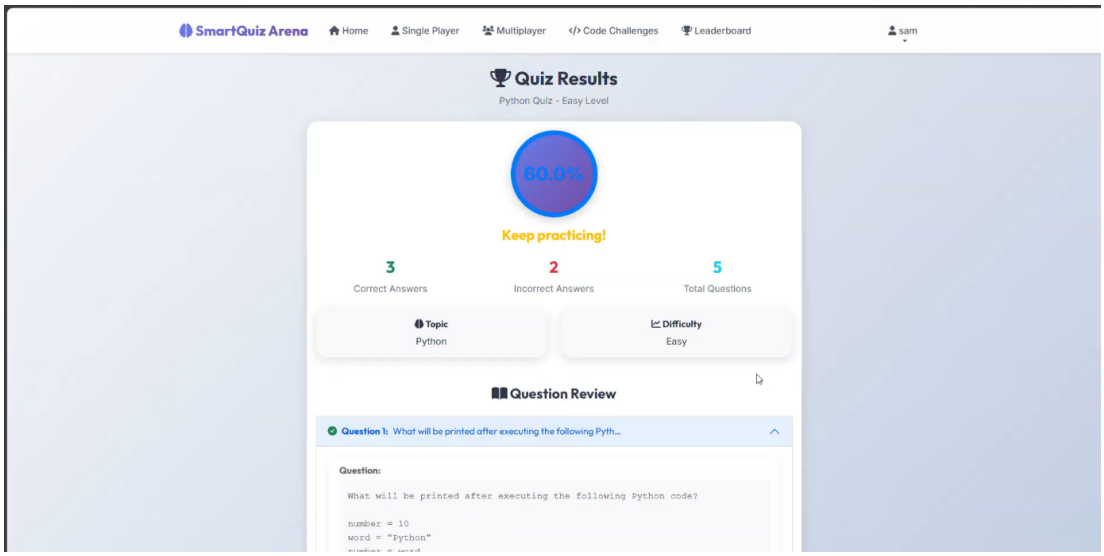

Fig 5.1.3.2: Result Window

- In single-player mode, the system begins by retrieving AI-generated or stored MCQ questions from the Question table and creating a new QuizSession entry to track the user's progress.
- Each question is then displayed through the frontend interface, while the backend maintains the current question index inside the session to ensure proper order.
- The user's selected answers are temporarily stored until submission, after which the backend evaluates them against the correct_answer fields in the database. The final score, accuracy, and performance summary are saved in the PlayerScore table and presented to the user in the result window for review.

5.1.4 Multiplayer mode

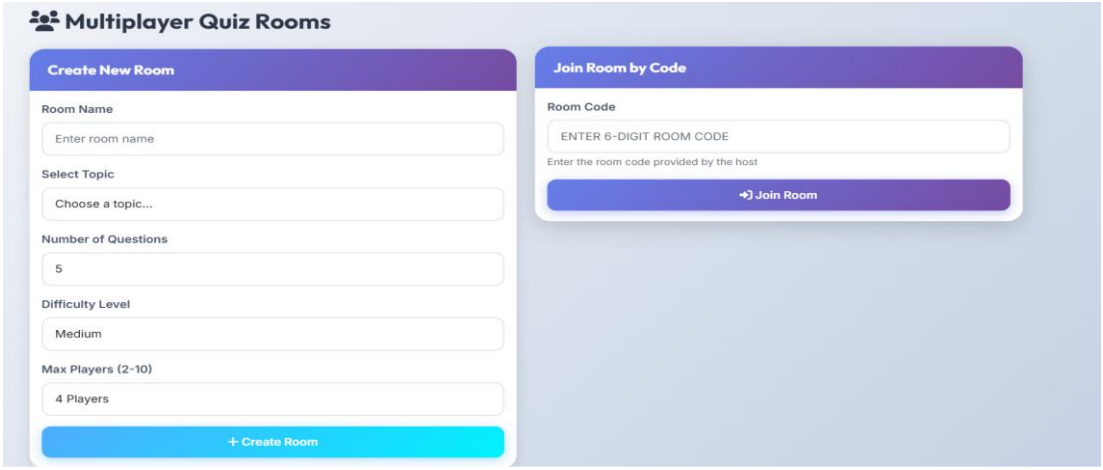

Fig 5.1.4.1:Multiplayer Mode

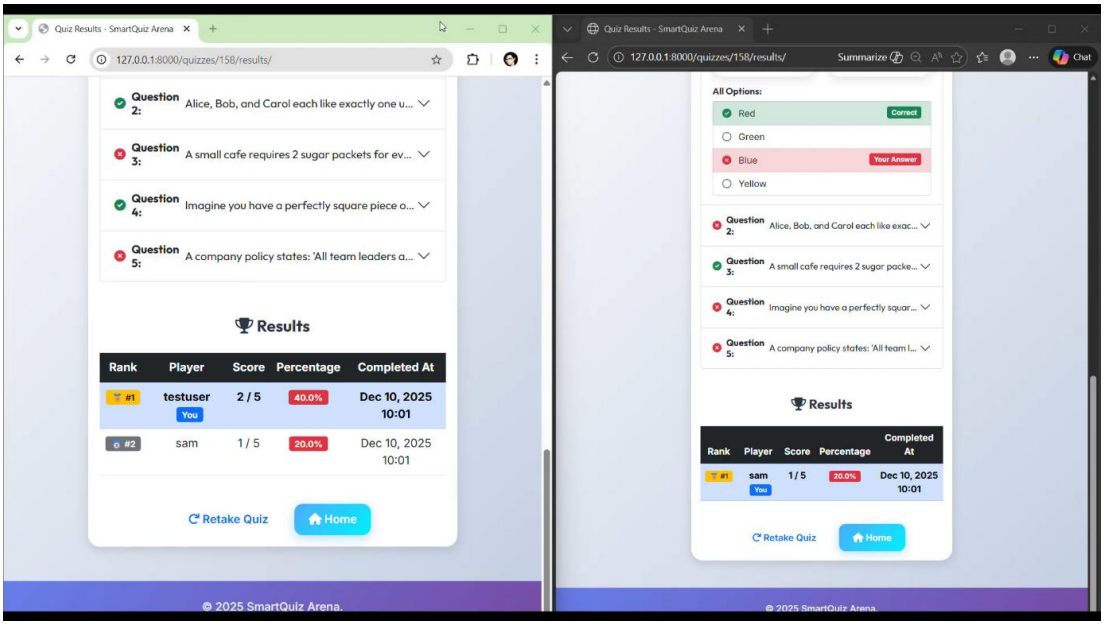

Fig 5.1.4.2:Result(2 browsers)

- In multiplayer mode, both systems connect to the same QuizSession in the database, allowing them to participate in a synchronized quiz.
- The backend ensures that both players receive the same questions and timing updates simultaneously, maintaining fairness throughout the session.
- Each player's responses are evaluated separately, and their scores are stored in the PlayerScore table. Once both players finish, the system compares the results and displays the winner along with individual performance details.

5.1.5 Coding Challenge

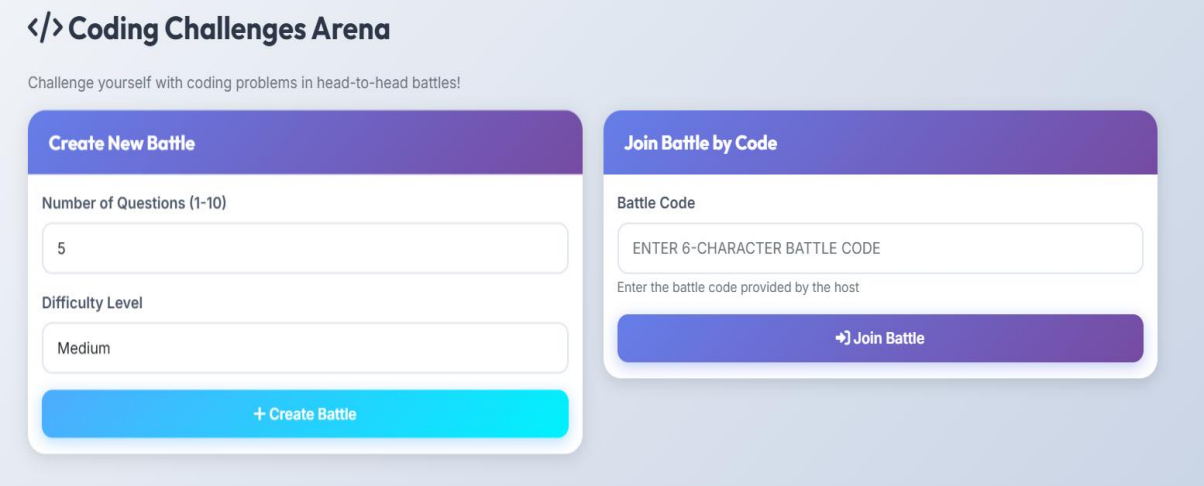

Fig 5.1.5.1:Coding Challenge Mode

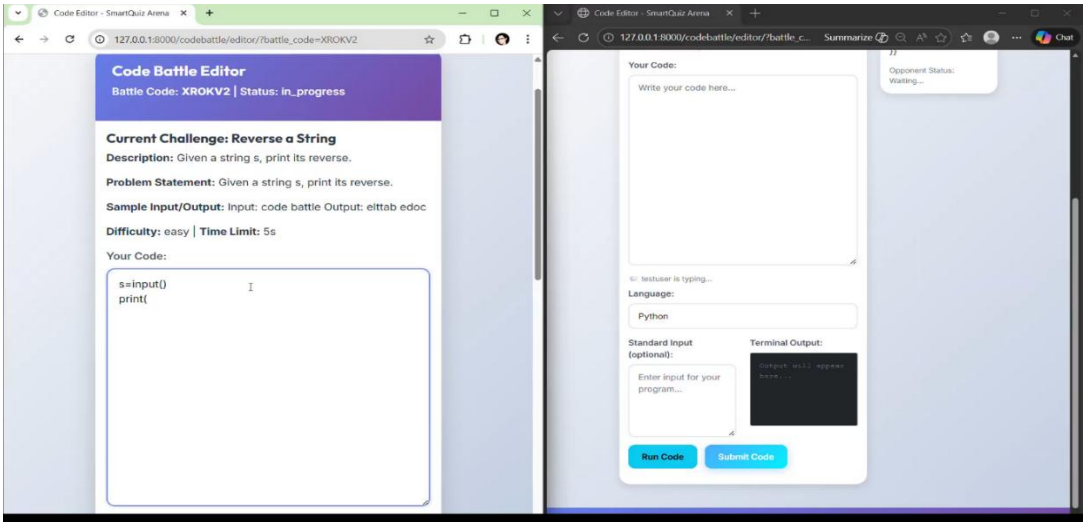

Fig 5.1.5.2: Coding in two different web browsers

- In the coding challenge mode, the problem statement is loaded from the CodingProblem table and shown to both participants; each code submission is saved to the CodeSubmission table and then forwarded to the Judge0 API for secure execution and testing.
- Judge0 runs the submitted code against predefined test cases and returns pass/fail counts, execution time, output, and error messages, which the backend records back into CodeSubmission.
- The system uses these results to compute scores and update PlayerScore entries; final statuses (passed/failed/error) and runtimes are displayed to users. Using Judge0 ensures sandboxed execution and consistent, language-agnostic evaluation while the backend handles submission tracking, result storage, and winner determination.

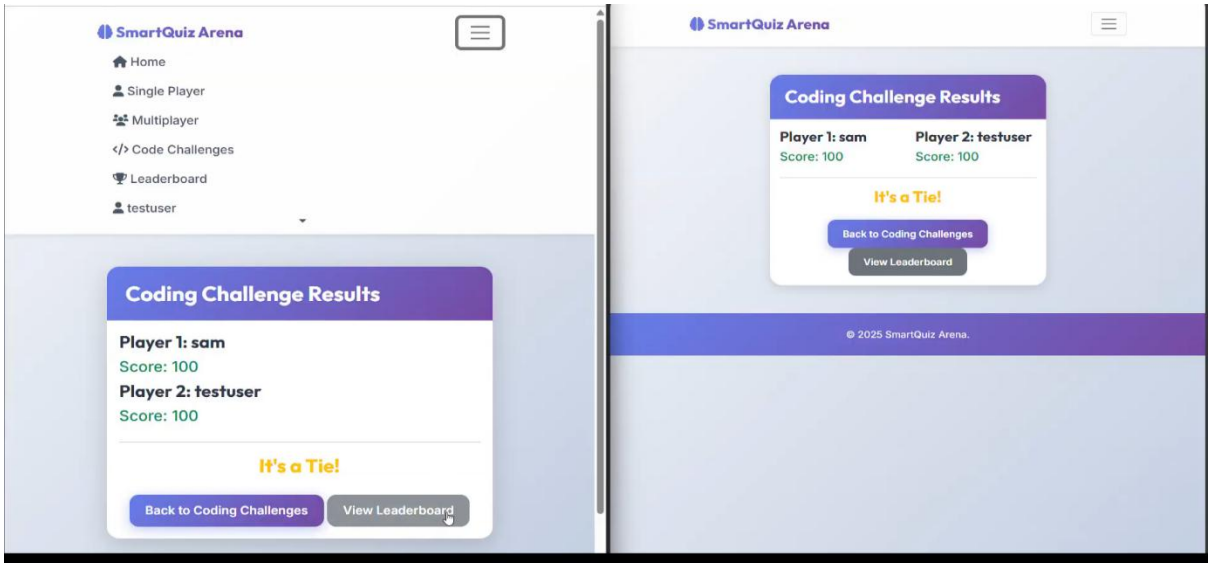

Fig 5.1.5.3:Result

5.1.6 Leaderboard

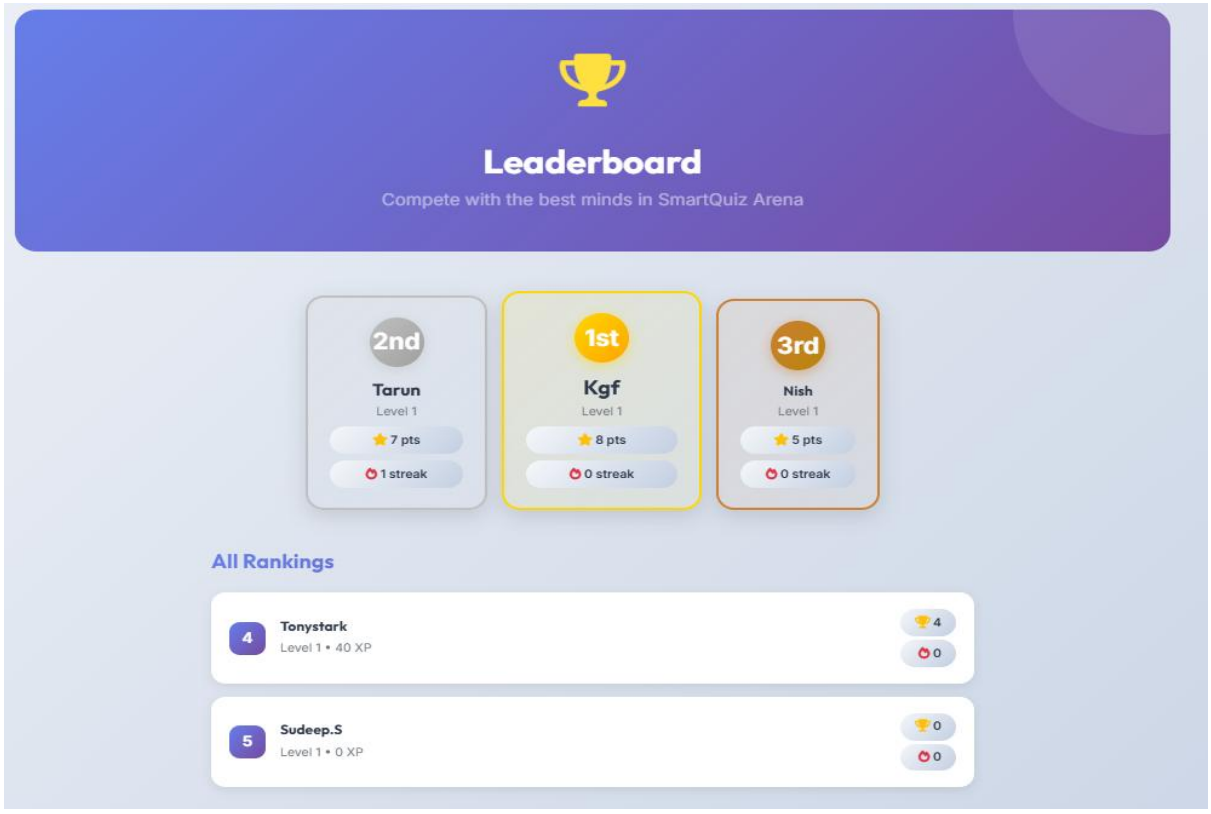

Fig 5.1.6: Leaderboard

- The leaderboard retrieves and ranks players based on their stored scores and performance statistics, displaying the top performers in the system.

5.2 Discussion

5.2.1 Effectiveness of the System:

• The Smart Quiz Arena system successfully provided a working platform for conducting MCQ quizzes, multiplayer quiz sessions, and coding challenges, fulfilling the primary objectives of the project.

• Reliable question loading and consistent scoring ensured accuracy during both single-player and multiplayer quiz attempts.

• The ability for two systems to participate in a synchronized quiz session demonstrated effective communication and session management.

• Coding challenges were executed correctly, with submitted code evaluated against predefined test cases, allowing fair comparisons between players.

• The system showed stable performance, secure handling of quiz data, and maintained a user-friendly interface throughout testing.

5.2.2 Challenges Encountered

• Synchronizing two systems during multiplayer sessions required careful handling of session states and question flow.

• Managing coding submissions and processing outputs for different test cases posed complexity during backend implementation.

• Ensuring that both devices received the same question at the same time required precise control of session and index tracking.

• Handling refreshes or accidental disconnections during an active quiz session created challenges in restoring the correct quiz state.

• Performance varied depending on local network stability, especially when both systems communicated with the backend simultaneously.

5.2.3 Limitations of the Current System

• The system currently operates only between two laptops on the same network, and large-scale multiplayer support has not yet been implemented.

• The system uses WebSockets for real-time quiz synchronization, although further optimization can improve performance for larger multiplayer groups.

• Coding challenge evaluation is based only on predefined test cases and does not include runtime sandboxing or advanced judging features.

• The user interface is optimized for desktop usage and does not yet include a fully responsive mobile layout.

• The system depends on local hosting; cloud deployment or remote accessibility has not been added in the current version.

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

The Smart Quiz Arena system successfully achieved its primary objective of providing a platform for conducting MCQ quizzes, multiplayer quiz sessions, and coding challenges between for students. By enabling students to participate in timed quizzes, submit answers easily, and receive instant results, the system offers a reliable and efficient solution for digital quiz management. The inclusion of multiplayer and coding features further demonstrates its versatility and ability to support competitive learning environments.

Overall, Smart Quiz Arena highlights the advantages of digitizing traditional quiz methods by improving accessibility, accuracy, and user experience. The system reduces manual effort, ensures consistent scoring, and supports smooth interaction between devices. With future enhancements such as large-scale multiplayer support, real-time synchronization, and cloud deployment, Smart Quiz Arena has strong potential to evolve into a comprehensive quiz and coding competition platform for educational and training institutions.

6.2 Future Enhancements

To further increase the effectiveness and usability of the Smart Quiz Arena, the following enhancements are recommended:

• **Expansion of AI-Generated Question Bank:**

- The system uses AI-generated MCQ questions for both single-player and multiplayer quizzes.
- In the future, the AI module can be enhanced to generate more diverse question types, adaptive difficulty levels, and subject-specific question sets.

• **Scalability for Large Multiplayer Sessions:**

- Although real-time synchronization works smoothly between two systems, the platform can be extended to handle more players simultaneously.
- This requires optimization of session management, congestion handling, and scalable server deployment.

• **Advanced Coding Judge Integration:**

- The coding challenge feature can be expanded with sandboxed execution, language-specific optimizations, and automated performance-based scoring.

- This will make the coding challenges more realistic and competitive for larger groups.

•Cloud Hosting and Remote Accessibility:

Deploying the backend and database on cloud infrastructure would allow users to join quizzes from anywhere, removing the current limitation of local network usage.

• Mobile Application Development:

- A dedicated mobile app for Android and iOS would make the quiz platform more accessible.
- Mobile support also enables students to participate in quizzes directly from their phones during classrooms or remote learning.

REFERENCES

[1] Django Software Foundation. (2024). *Django Web Framework*. Retrieved from https://www.djangoproject.com/

[2] Python Software Foundation. (2024). *Python Programming Language*. Retrieved from https://www.python.org/

[3] SQLite Consortium. (2024). *SQLite Database Engine*. Retrieved from https://www.sqlite.org/

[4] JavaScript Guide. (2024). *JavaScript Tutorial – W3Schools*. Retrieved from https://www.w3schools.com/js/