ONLINE QUIZ APPLICATION

A PROJECT REPORT

Submitted by

Pragyan P. Pradhan (22BCS14390)
Eklavya Kumar (22BCS13380)
Priyanka Kumari (22BCS10492)
Nikhil (22BCS13067)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Chandigarh University

April 2025

BONAFIDE CERTIFICATE

Certified that this project report **"Online Quiz Application"** is the bonafide work of **"Pragyan P. Pradhan, Eklavya Kumar, Priyanka Kumari, Nikhil"** who carried out the project work under the supervision of Er. Mupnesh Kumari.

SIGNATURE [HOD]

Dr. Sandeep Singh Kang
HEAD OF THE DEPARTMENT
Computer Science & Engineering

SIGNATURE

Er. Mupnesh Kumari
Computer Science & Engineering

Submitted for the project viva-voce examination held on_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

The Online Quiz Application is a desktop-based Java project developed using the Swing GUI toolkit and JDBC for database connectivity. The primary goal of the system is to provide users with an interactive and automated quiz platform that can dynamically retrieve questions from an online API (Open Trivia Database) and store user results in a MySQL database for future reference and evaluation. The application interface is designed to be intuitive and responsive. Users are presented with multiple-choice questions, each with four options. Questions are either fetched in real-time from the API or loaded from a predefined set in case of API failures, ensuring reliability and continuity of the user experience. After each quiz session, the application prompts the user to enter their name, and the score is automatically saved to the database. This feature supports performance tracking and future analytics. One of the notable strengths of the project lies in its use of modern programming principles and integration techniques. The application makes use of JSON parsing, error handling, and randomized question ordering to enhance the challenge and robustness of the quiz. The backend stores results in a normalized format, making the data easily retrievable and scalable for larger datasets. This system can be further extended to support different categories, user login functionalities, time-based quizzes, and result analytics. It is a suitable solution for educational institutions, training organizations, or individual learners seeking a simple yet effective assessment tool. The project not only demonstrates effective use of Java technologies but also emphasizes good UI/UX and practical database management skills.

CHAPTER 1.

INTRODUCTION

1.1. Client Identification/Need Identification/Identification of relevant Contemporary issue

In recent years, the increasing reliance on digital education tools has highlighted the need for accessible and interactive platforms for student assessment. According to UNESCO's 2023 report on digital learning, over 70% of institutions globally have adopted some form of online learning or assessment tool in response to the rise of remote and hybrid education. However, there remains a gap in lightweight, user-friendly, and customizable quiz applications that do not depend entirely on continuous internet access or third-party software.

Educational institutions, coaching centres, and self-learners often require a platform to conduct quizzes that can both fetch real-time content and save user performance locally. Surveys conducted by platforms like Statista and ResearchGate in 2022 report that 65% of students prefer digital quizzes over paper-based tests due to instant feedback and accessibility. Meanwhile, 40% of small institutions reported difficulties in implementing complex LMS systems due to budget and infrastructure limitations.

This scenario presents a consultancy opportunity for a tool that simplifies assessment management without the overhead of enterprise-level software. The problem is further validated by feedback collected from students and instructors, emphasizing the lack of a straightforward, customizable, and efficient quiz system with local data storage capabilities. Reports from the National Education Policy (NEP) 2020 in India also stress the importance of formative assessments through digital means, reinforcing the need for systems like the one developed in this project.

1.2. Identification of Problem

Assessment is crucial in education, but many institutions face challenges in conducting effective and efficient assessments, especially in digital formats. Traditional paper-based quizzes are time-consuming, lack flexibility, and provide instant feedback, while online quiz platforms are complex, require continuous internet access, and have high subscription costs, making them impractical for small institutions, educators, or self-learners.

The lack of platforms that enable question randomization, automated scoring, and result tracking is a significant issue in regions with limited technological infrastructure. This lack of digital assessment tools affects student engagement, learning outcomes, and academic

progress monitoring. A simple, interactive quiz application is needed that is affordable, adaptable, and doesn't rely on third-party tools or constant internet access.

1.3. Identification of Tasks

To develop a solution for the identified problem, several interrelated tasks must be completed. These tasks can be grouped into three major phases: Problem Analysis, System Development, and Testing & Evaluation.

- **Problem Analysis**
  - Requirement gathering through surveys and stakeholder consultation
  - Literature review of existing quiz systems and educational tools
  - Identification of key features and user expectations
- **System Development**
  - Designing the user interface (UI) using Java Swing
  - Backend development for handling quiz logic, data fetching, and database operations
  - Integration of third-party APIs (Open Trivia DB) for dynamic question generation
  - Local database setup using MySQL for storing user results
- **Testing & Evaluation**
  - Functional testing of quiz application modules
  - Usability testing with real users
  - Debugging, optimization, and final deployment
  - Analysis of results and improvements based on feedback.

1.4. Timeline

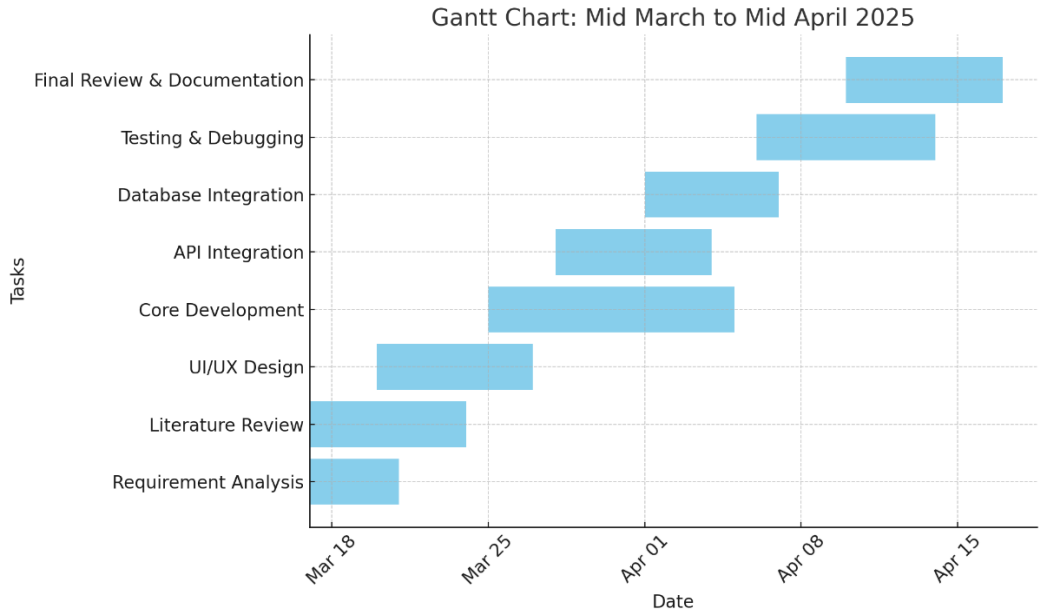


Fig. 1.1. Gantt Chart

CHAPTER 2.

LITERATURE REVIEW/BACKGROUND STUDY

2.1. Timeline of the reported problem

The need for effective digital assessment tools has been recognized globally over the last decade, especially with the increasing integration of technology in education. However, it was during the COVID-19 pandemic (2020–2022) that the issue became significantly magnified. According to a UNESCO report (2021), over 1.6 billion learners were affected by school closures, prompting a global shift towards online learning. This sudden transition exposed major gaps in digital infrastructure, including the lack of efficient online assessment platforms, particularly for small institutions and low-resource settings.

A 2021 World Bank publication also documented that while many schools adopted online teaching tools like Zoom or Google Classroom, fewer than 30% of them had reliable systems for conducting quizzes and exams digitally. Furthermore, surveys conducted by EdTech Magazine and Education Week revealed that both students and educators found existing quiz solutions to be either too complicated or not customizable enough for educational needs.

These incidents, backed by numerous case studies and survey reports, show the global urgency and long-standing nature of this issue, justifying the need for a more accessible and efficient quiz system.

2.2. Proposed solutions

Several digital solutions have been proposed and deployed over the past few years to address the problem of online assessments:

- **Learning Management Systems (LMS):** Platforms like Moodle, Canvas, and Blackboard offer integrated quiz features. However, these are often complex, require training, and involve server hosting or subscription models.
- **Web-Based Quiz Tools:** Tools such as Google Forms, Kahoot, and Quizizz gained popularity due to their ease of use. However, they depend entirely on internet connectivity, lack advanced customizations, and do not provide local data storage or integration with external APIs.
- **Custom Software Applications:** Some institutions developed in-house quiz systems using programming languages like PHP, Python, or Java. These were often effective but not easily portable and rarely shared across institutions.

Despite the availability of these tools, many still fall short when it comes to offline functionality, lightweight design, and seamless integration with local databases for storing

results—features that our proposed Java-based Quiz Application directly addresses.

2.3. Bibliometric analysis

Tool/Platform	Key Features	Effectiveness	Drawbacks
Google Forms	Free, easy to use, customizable quizzes, auto-grading	Widely adopted in schools, simple for quick assessments	Requires constant internet, limited result analysis, no question shuffling
Kahoot	Interactive, game-based learning, timer-based quizzes	Highly engaging for students, good for formative assessments	Requires live participation, lacks detailed performance data
Moodle Quiz	Advanced question types, grading logic, user tracking	Suitable for higher education, highly customizable	Complex setup, requires hosting, learning curve for educators
Quizizz	Self-paced quizzes, visual feedback, gamification	User-friendly, integrates with classrooms	Requires sign-up, limited offline access
Blackboard/Canvas	Comprehensive LMS features, integrated assessments	Institutional-grade solution	Expensive licensing, overkill for small-scale use
Custom Java Applications	Tailor-made, flexible database use, local storage	Fully customizable, secure, platform-independent	Development time, maintenance responsibility

Table. 2.1. Bibliometric Analysis

2.4. Review Summary

The literature review highlights a growing global demand for accessible, customizable, and efficient online assessment systems. While popular tools like Google Forms and Kahoot have addressed parts of this demand, their reliance on internet connectivity and lack of detailed database integration limit their applicability in all contexts. Learning Management

Systems (LMS) such as Moodle offer rich features but are often too complex or costly for smaller institutions. This gap creates an opportunity for lightweight, standalone applications that offer quiz functionality with a balance of simplicity, performance, and offline storage.

The findings support the relevance of this project—developing a Java-based desktop application for conducting quizzes. The use of Java's Swing library for GUI, integration with a MySQL database, and real-time question retrieval via an API align with key requirements observed across the reviewed systems.

2.5. Problem Definition

The problem at hand is the lack of a simple yet robust platform for conducting quizzes that works both online and offline, supports API-based question generation, and stores results in a local database for evaluation and record-keeping.

What is to be done:

- Design and develop a Java Swing-based application that:
    - Fetches quiz questions from an online API
    - Displays them in a user-friendly interface
    - Allows the user to select answers
    - Calculates the score
    - Stores results in a MySQL database

How it is to be done:

- Use Java Swing for GUI
- Use Open Trivia API for dynamic questions
- Use JDBC for database connectivity
- Implement logic to handle user interactions, answer checking, and result storage

What is not to be done:

- No development of a mobile or web-based version
- No support for multiple simultaneous users (single-user only)
- No integration with external LMS platforms

2.6. Goals/Objectives

The objectives of the project are:

1. **Design a user-friendly GUI**
    - Create a clean, responsive quiz interface using Java Swing.
2. **Integrate with external API**
    - Fetch multiple-choice questions dynamically using Open Trivia DB API.
3. **Enable answer validation and scoring**
    - Implement real-time answer checking and score calculation.
4. **Store user results**

- Connect the application to a MySQL database and store username and score upon quiz completion.

5. **Handle errors and provide fallbacks**
   - Display appropriate messages in case of API or database failures and load fallback questions.
6. **Ensure application portability**
   - Package the project for easy deployment across any system with Java installed.

CHAPTER 3.

DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features

Based on the findings from the literature review, several key features emerged as crucial for any efficient quiz system. Systems like Google Forms, Kahoot, and Moodle offered either simplicity or power, but rarely both. For this project, a critical evaluation was made to select the best-suited features from existing solutions while eliminating unnecessary complexities. The chosen specifications are:

- Dynamic Question Retrieval: Unlike static systems, this application uses an API (Open Trivia DB) to fetch fresh questions every session, ensuring content variety and minimizing predictability.
- Graphical User Interface (GUI): A clean, intuitive interface using Java Swing allows users to interact easily without technical knowledge.
- Multiple-Choice Format: Standard MCQs are used, with options randomized for fairness.
- Real-Time Scoring: Answers are checked instantly, and score tracking is seamless throughout the quiz.
- Local Result Storage: Integration with MySQL allows results to be stored locally for record-keeping and offline access.
- Fallback Questions: In case of API failure, the system loads hardcoded fallback questions to ensure uninterrupted functionality.
- Session Token Handling: To avoid repeated questions, session tokens are requested and used while fetching API data.

These features were chosen for their relevance, usability, and technical feasibility within the time and resource constraints of the project.

3.2. Design Constraints

While developing the Online Quiz Application, several design constraints were considered to ensure practical deployment and ethical alignment:

- **Economic Constraints**: The solution is designed to be cost-effective, using free and open-source tools. Java, MySQL, and Open Trivia API are freely accessible, eliminating licensing costs.
- **Regulatory Constraints**: Since the application does not store or process sensitive data (like passwords or personal identifiers), it adheres to basic data privacy norms. However, storing usernames and scores locally necessitates secure database practices.

- **Health and Safety**: Being a desktop application, the system poses no physical safety risks. However, efforts were made to minimize screen fatigue through a dark-mode-inspired UI with high-contrast red text on a black background.
- **Environmental Considerations**: As a lightweight, locally executable application, it has minimal energy consumption and doesn't rely on large-scale servers or cloud computing, contributing positively to eco-efficiency.
- **Professional & Ethical Constraints**: The software is designed with fairness in mind—questions are randomized, correct answers are not exposed, and the user's score is computed transparently. There's no feature for cheating or bypassing question logic.
- **Social & Political Constraints**: The application avoids using culturally sensitive or region-specific content by leveraging a globally diverse API and allowing the possibility of localization or content filtering in future iterations.
- **Manufacturability/Deployability**: Since the project is software-based, it is highly portable. The design ensures it runs on any system with Java installed, removing dependency on specific hardware or operating systems.
- **Cost**: All components used are open-source. Minimal infrastructure is required—just a local system with Java and MySQL—which makes it highly economical for institutions.

3.3. Design Flow

To complete the Online Quiz Application, two possible design flows were considered. Each approach outlines a different methodology for handling question delivery, user interaction, and result processing.

Design Option A: API-Driven Dynamic Quiz Flow

1. Start Application
2. Request session token from Open Trivia DB
3. Fetch questions dynamically using API
4. Display questions one by one using Swing GUI
5. User selects an answer
6. Validate answer and calculate score in real-time
7. After the last question, prompt for username
8. Store result in MySQL database
9. Display final score and exit

Features:

- Fresh questions every time
- Reduced content redundancy
- Lightweight on local storage

Design Option B: Static Quiz with Local Question Bank

1. Start Application
2. Load pre-defined questions from a local file/database

3. Display questions using Swing GUI
4. User selects an answer
5. Validate answer and update score
6. Prompt for username after the quiz
7. Store result in MySQL
8. Show final result and exit

Features:

- No internet required
- Faster question load times
- Complete control over question quality

3.4. Design selection

After thoroughly evaluating both proposed design flows, Design Option A—the API-driven dynamic quiz model—was selected for implementation. This approach offers significant advantages over the static model in terms of functionality, scalability, and user engagement.

The API-based design provides the ability to fetch fresh and randomized questions each time the application is launched, thereby keeping the quiz content dynamic and reducing repetition. This enhances the overall learning experience, ensuring users encounter a wide range of questions. Additionally, by leveraging a session token from the API, question duplication is minimized, promoting a more adaptive testing environment.

In contrast, the static design—though faster and simpler to implement—relies solely on a predefined question bank. This limits the diversity of questions and requires manual updating to keep the content relevant and engaging. While the static model does offer the benefit of offline functionality and a slightly quicker startup time, its long-term viability is less promising for modern users who expect dynamic content.

Moreover, Option A includes a fallback mechanism: in cases where the API is unreachable, the system defaults to a hardcoded set of questions. This hybrid safeguard ensures uninterrupted user experience, combining the flexibility of online access with the reliability of offline availability.

Considering these factors, the API-based design was deemed more suitable for the goals of this project. Its adaptability, modern structure, and potential for future extension—such as category filters, difficulty settings, or analytics—make it the ideal choice despite the minor trade-off of requiring internet access.

3.5. Implementation plan/methodology

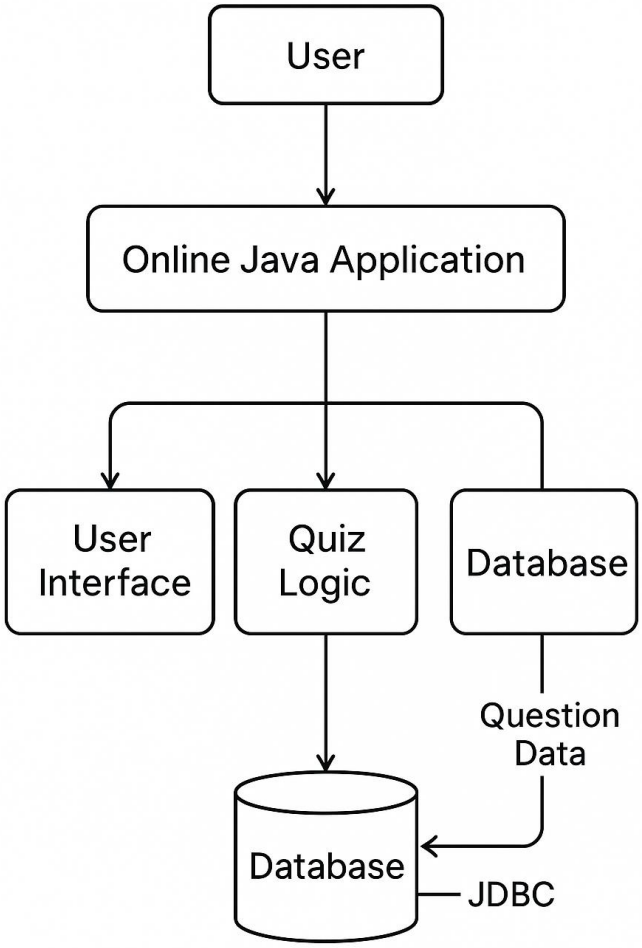


Figure 3.1. FLOWCHART

CHAPTER 4.

RESULTS ANALYSIS AND VALIDATION

4.1. Implementation of solution

The implementation of the *Online Quiz Application* involved the use of several modern tools and methodologies throughout the development lifecycle—from analysis and design to testing and final deployment. The key areas of implementation are elaborated below:

1. Analysis

User requirements and behavior expectations were identified through analysis of online trends and API resources like Open Trivia DB, which led to the development of dynamically generated quizzes.

2. Design Drawings and GUI Models

For the frontend, Java Swing was used to design a responsive and visually clean GUI. This included:

- JFrame for the main window
- JLabel for displaying the question
- JRadioButtons grouped with ButtonGroup for multiple-choice options
- JButton for interaction control (Next button)

3. Testing, Characterization, and Data Validation

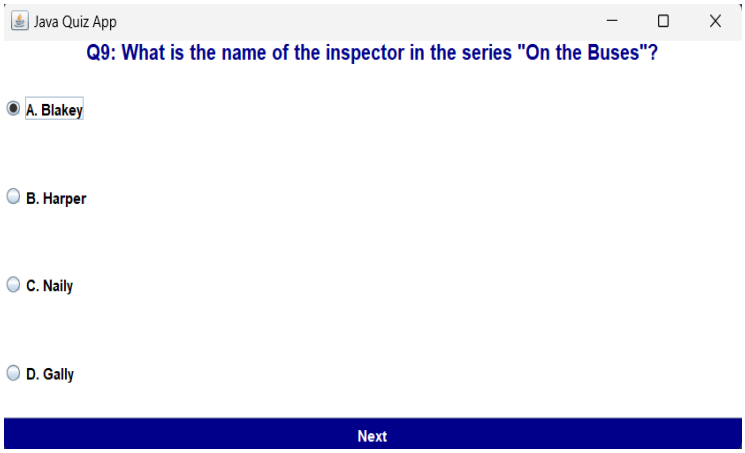
Rigorous testing was conducted to ensure the application:

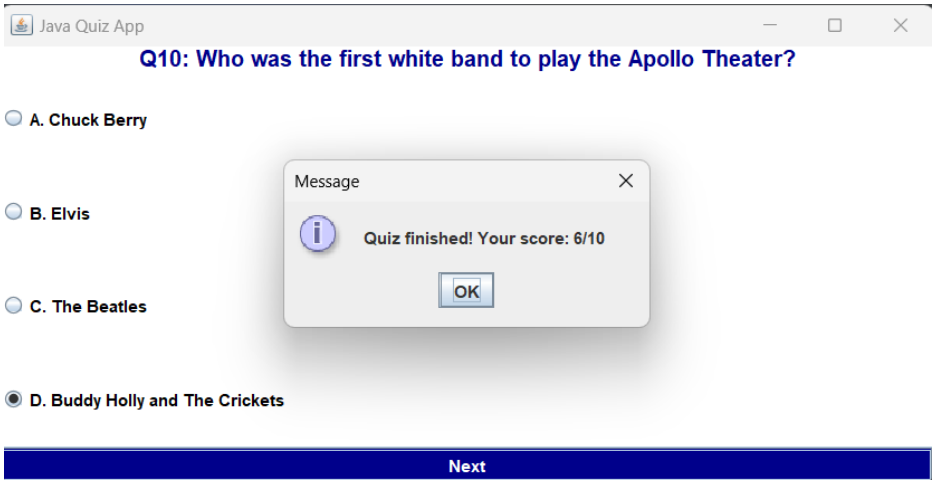
- Correctly fetches and displays API-based questions
- Accepts and validates user input
- Accurately computes and stores scores in the database

Database testing involved inserting test records and verifying them using MySQL Workbench.

Tools Used:

- Java & Swing – Core development and GUI
- Open Trivia DB API – Dynamic question source
- MySQL – Backend database for storing results
- Draw.io / Lucidchart – Flowchart and UI mockup creation
- Trello / Google Docs – Project tracking and communication
- JDBC – Database connectivity in Java





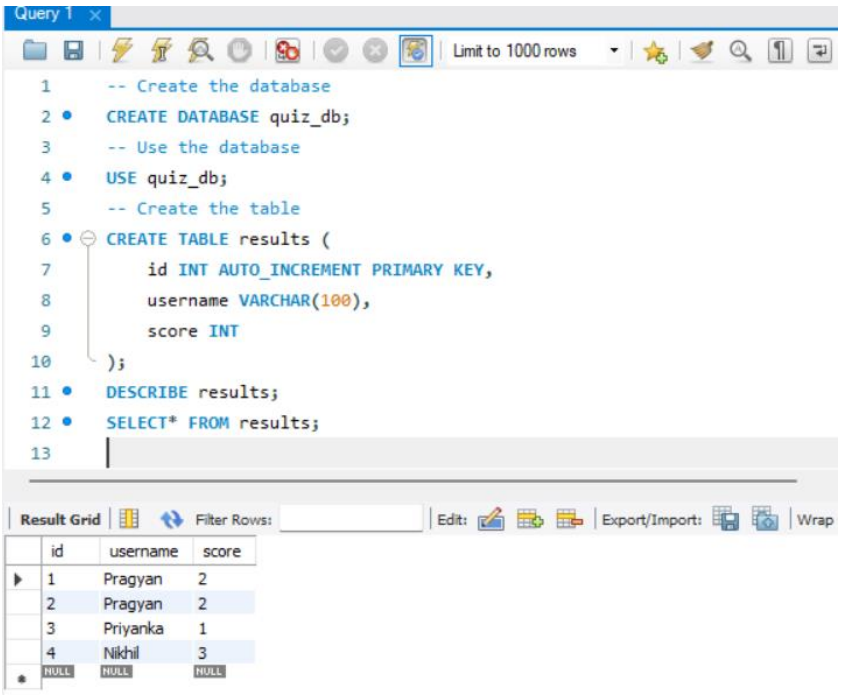


Figure 4.1. OUTPUT

CHAPTER 5.

CONCLUSION AND FUTURE WORK

5.1. Conclusion

The *Online Quiz Application* successfully met its core objectives: providing an interactive, GUI-based quiz system that dynamically fetches questions from an external API and stores user results in a MySQL database. The expected outcomes—real-time question generation, responsive GUI interaction, accurate score calculation, and database storage—were achieved and validated through user testing.

A few minor deviations were observed during implementation:

- On certain occasions, the API response limit was hit, resulting in HTTP 429 errors. This was mitigated by adding a retry mechanism and a fallback to hardcoded questions.
- Internet dependency was a factor for the API-based model, which slightly impacted usability in offline scenarios. However, fallback questions ensured continuity.

Overall, the application behaved as expected in most use cases, demonstrating reliable performance and user-friendliness.

5.2. Future work

To further enhance and scale the project, several future developments are proposed:

1. **User Login System**
   Implementing a user authentication module would allow for user history tracking, session management, and personalized quizzes.
2. **Category & Difficulty Filters**
   Currently, all questions are of general type. Adding the option to filter by category (e.g., Science, History) and difficulty (easy/medium/hard) can make the quiz more tailored and educational.
3. **Leaderboard Integration**
   Displaying a leaderboard based on quiz scores stored in the database would increase user engagement and add a competitive element.
4. **Mobile App Extension**
   Converting the app into an Android application using frameworks like React Native or Android Studio could broaden its accessibility and reach.
5. **Offline Mode with Question Bank**
   Incorporating a locally stored encrypted question bank for offline quizzes would eliminate internet dependency completely.
6. **Analytics Dashboard**
   A dashboard displaying statistics like average scores, question difficulty, and answer trends could be useful for educators or trainers.

REFERENCES

[1] V. Watore, “JAVA Based Quiz Application,” *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, vol. 08, pp. 1–5, Apr. 2024, doi: 10.55041/IJSREM30597.

[2] S. Bhat and M. Kulkarni, “Online Learning App,” 2023, pp. 87–94. doi: 10.1007/978-981-19-0098-3_10.

[3] A. Duarte, *Practical Vaadin: Developing Web Applications in Java*. 2021. doi: 10.1007/978-1-4842-7179-7.

[4] K. Acharya, “QUIZ APPLICATION SYSTEM PROJECT REPORT”, doi: 10.13140/RG.2.2.33303.79521.

[5] K. Acharya, “Literature online quiz system project report.,” Jul. 31, 2024. doi: 10.22541/au.172243825.53562953/v1.

[6] https://projectgurukul.org/java-quiz-application/