QUIZ APPLICATION

PROJECT REPORT

Project Based Learning in Java

23CSH-304

Submitted by
Riya Kamboj 23BCS10717
Vaishnavi 23BCS13737
Chhavi Sinha 23BCS10304
Vishesh 23BCS12175

Section 609
Group B

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University
Nov, 2025

TABLE OF CONTENTS

1. Introduction

1.1. Introduction to Project

In today's digital learning environment, interactive tools play a crucial role in education. One such tool is a Quiz Application, which serves as an effective method to assess and reinforce students' understanding of a subject. This project is titled "Java Quiz Application using Swing", developed using the Java programming language and Swing framework for GUI (Graphical User Interface) development.

The purpose of this project is to design a lightweight, user-friendly application that presents multiple-choice questions related to core Java programming, evaluates responses, manages time for each question, and displays the final score upon completion. In addition, the application highlights the contributors and the supervising faculty to acknowledge team efforts and guidance.

1.2. Identification of Problem

In many academic environments, quizzes are often conducted using paper-based methods or basic forms that lack interactivity and automation. Common problems include:

- Time-consuming evaluation process
- No countdown or time pressure simulation
- No automatic score calculation
- Lack of modern GUI and user engagement
- No proper acknowledgment of contributors within the tool

The aim is to solve these problems by developing a desktop-based quiz system that is efficient, intuitive, and fully automated using Java and Swing. This application not only improves the quiz experience but also helps in learning and practicing Java programming in a gamified way.

2. Background Study

2.1. Existing solutions

There are numerous quiz platforms available today such as Kahoot, Google Forms, and Moodle, which offer online quiz creation and assessment features. However, these platforms often:

- Rely on internet access
- Require user registration or logins
- Are hosted on web servers or external systems

- Are not always open-source or customizable

For local or offline quizzes (especially in classroom settings or individual learning), there is a need for a lightweight desktop-based application that can work without internet access, be customized for specific subjects like Java, and offer a simple GUI for interaction.

2.2. Problem Definition

The current systems do not meet the following requirements efficiently:

- Offline access
- Personalized content (focused on Java programming)
- Minimalistic, fast-loading, no-installation-based execution
- Local, secure quiz evaluations
- Recognition of development team and faculty within the application

Thus, we define the problem as:

"To design and develop a desktop-based Java Quiz Application using Swing that can administer basic Java questions with time-bound constraints, score calculation, and contributor acknowledgment without requiring internet or external dependencies."

2.3. Goals/Objectives

- Develop a standalone GUI-based Java application for conducting quizzes.
- Provide multiple-choice Java-related questions with options and a timer.
- Evaluate answers in real-time and maintain a cumulative score.
- Display final results after quiz completion in a user-friendly dialog.
- Showcase development team and guide using a formatted JTable in the GUI.
- Maintain code readability and modularity for future enhancements.
- Ensure the application runs on any Java-supported desktop system.

3. Design/Flow process

3.1. Evaluation & Selection of Specifications/Features

Before development began, the key specifications and features were evaluated based on the project goals and practical requirements. The selected specifications include:

- Java Swing GUI: Chosen for its ease of use in building graphical interfaces without external dependencies.
- Multiple-choice questions (MCQs): Simple and effective for testing foundational knowledge.
- Timer for each question: Adds challenge and simulates exam-like pressure.
- Score tracking: Gives immediate feedback on user performance.
- Team and Guide info window: Recognizes contributors within the application.

- Standalone operation: No need for internet or installation, just a runnable `.jar` file.

3.2. Analysis of Features and finalization subject to constraints

The project considered both technical and user-centric constraints before finalizing features:

Feature	Finalized	Reason/Constraint
Java-based GUI	☑	Universally supported and easy to deploy
Countdown timer per question	☑	Encourages time management; implemented via `javax.swing.Timer
Result calculation	☑	Immediate feedback and score visibility
Custom styling	☑	Enhanced font, colors, and layout via `JLabel`, `JButton` etc.
Team info table	☑	Included as part of result screen to give credit

3.3. Design Flow

Below is the simplified design flow of the project: [Start]

↓

[Display GUI Window]

↓

[Show Question + Options + Timer]

↓

[User Selects Answer → Clicks Next]

↓

[Check Answer → Update Score]

↓

[Move to Next Question → Repeat Timer]

↓

[End of Quiz]

↓

[Show Score Dialog Box]

↓

[Display Developer Info + Guide Acknowledgment]

↓

[Exit Application]

4. Result Analysis and Validation

4.1. Implementation of solution

This chapter focuses on how the solution was implemented and how it was validated through testing and feedback. Implementation Highlights

1. Graphical User Interface (GUI)

Built using Java Swing, including components like `JFrame`, `JLabel`, `JButton`, `JRadioButton`, and `JTable`.

Questions and options are rendered dynamically with a clean and user-friendly layout. Fonts and colors were customized to enhance the visual appeal and accessibility.

2. Timer Functionality

A countdown timer of 15 seconds was implemented for each question using `javax.swing.Timer`.

It updates the timer label every second, adding urgency and time-awareness to the quiz.

If the timer hits 0, the next question is auto-triggered.

3. Quiz Logic

The quiz cycles through a set of basic Java questions stored in an array.

Options are stored in a 2D array, and correct answers are validated using an `int[] answers` array.

The selected option is compared to the correct answer; score is updated accordingly.

4. Score Tracking and Result Display

After each correct answer, the score is incremented and shown in real-time. Once the quiz ends, a dialog box displays the final score using `JOptionPane`.

5. Acknowledgment Window

A final popup window presents a table listing the team members and a line showing the project guide's name in green bold color.

Implemented using `JTable`, `JScrollPane`, and a customized `JLabel`.

5. Conclusion and future Work

5.1. Conclusion

This project aimed to design and implement a Java-based GUI Quiz Application that offers users a simple, interactive, and time-bound quiz environment. Through this project:

We successfully built a functional and engaging quiz app using Java Swing.

The application effectively tests users on basic Java concepts, reinforcing programming fundamentals.

Real-time timer and score tracking enhance user interaction and challenge.

The UI was refined for readability and ease of use, offering a smooth experience.

The post-quiz screen honors the development team and the guiding faculty, fostering a sense of teamwork and mentorship acknowledgment.

Overall, the project showcases a practical implementation of core Java programming, eventdriven design, and basic OOP concepts, all wrapped in a clean and intuitive interface.

5.2. Future work

While the current version meets its intended goals, there are several enhancements that can be explored:

1. Question Bank Expansion

   Incorporate more questions from various topics and difficulty levels. Include topics like OOP, Collections, Exception Handling, and File I/O.

2. User Login & Progress Tracking

   Add user authentication and maintain user scores/history for performance tracking.

3. Result Storage and Export

   Store quiz results in files or databases.

   Add feature to export results in PDF/Excel format.

4. Multimedia Integration

   Support for images, audio, or video questions to make the quiz more engaging.

5. Randomized Question Order

Implement shuffling to make each quiz attempt unique.

6. Mobile and Web Versions

   Extend the application to Android or Web platforms using JavaFX, React, or Flutter.

7. Custom Quiz Creation

   Allow users to create their own quizzes with custom questions and answers.

6.References

[1] Oracle, "*The Java™ Tutorials*," Oracle, 2024. [Online]. Available: https://docs.oracle.com/javase/tutorial/

[2] Oracle, "*Java Platform, Standard Edition Documentation*," Oracle, 2024. [Online]. Available: https://docs.oracle.com/en/java/javase/

[3] GeeksforGeeks, "*Java Swing Tutorial*," 2024. [Online]. Available: https://www.geeksforgeeks.org/java-swing/

[4] TutorialsPoint, "*Java Swing - Quick Guide*," 2024. [Online]. Available: https://www.tutorialspoint.com/swing/index.htm

[5] Baeldung, "*A Guide to Java Swing*," 2024. [Online]. Available: https://www.baeldung.com/java-swing