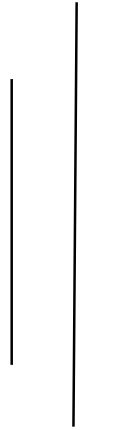


National College of Computer Studies  
Paknajol, Kathmandu



Project Report on Data Structures And Algorithm with Java  
Quiz Application

Submitted by:

Name: Siddhartha Shakya

Program: BIM

Section: B

Roll no: 23

Submitted to:

Mr. Chhetra Bahadur Chhetri

Mr. Rikesh Shrestha

Date of submission: 2025/04/25

## Table of Contents

<b>CHAPTER 1 INTRODUCTION .....</b>	<b>1</b>
<b>1.1 ORGANIZATION DESCRIPTION .....</b>	<b>1</b>
<b>1.2 OBJECTIVES .....</b>	<b>1</b>
<b>1.3 SCOPE AND LIMITATION .....</b>	<b>1</b>
<b>1.3.1 Scope .....</b>	<b>1</b>
<b>1.3.2 Limitations.....</b>	<b>1</b>
<b>1.4 METHODOLOGY .....</b>	<b>2</b>
<b>1.4.1 Project Framework.....</b>	<b>2</b>
<b>1.4.2 Data collection .....</b>	<b>2</b>
<b>1.4.3 Tools used .....</b>	<b>2</b>
<b>CHAPTER 2 ANALYSIS OF ACTIVITIES DONE AND PROBLEM SOLVED .....</b>	<b>3</b>
<b>2.1 REQUIREMENT ANALYSIS.....</b>	<b>3</b>
<b>2.1.1 Requirement Collection Method .....</b>	<b>3</b>
<b>2.1.2 Functional Requirement .....</b>	<b>3</b>
<b>2.1.3 Non-functional Requirement .....</b>	<b>3</b>
<b>2.2 FEASIBILITY STUDY .....</b>	<b>4</b>
<b>2.2.1 Technical Feasibility .....</b>	<b>4</b>
<b>2.2.2 Operational Feasibility .....</b>	<b>4</b>
<b>2.2.3 Economic Feasibility.....</b>	<b>4</b>
<b>2.2.4 Schedule Feasibility .....</b>	<b>5</b>
<b>2.3 MODULE DESCRIPTION.....</b>	<b>5</b>
<b>2.4 TESTING.....</b>	<b>6</b>
<b>2.4.1 Test Cases .....</b>	<b>6</b>
<b>2.4.2 Testing methodology.....</b>	<b>6</b>
<b>CHAPTER 3 DISCUSSION AND CONCLUSION .....</b>	<b>7</b>
<b>3.1 DISCUSSION .....</b>	<b>7</b>
<b>3.2 CONCLUSION .....</b>	<b>7</b>
<b>3.3 LESSON LEARNT.....</b>	<b>7</b>
<b>3.4 FUTURE RECOMMENDATIONS .....</b>	<b>7</b>
<b>REFERENCES .....</b>	<b>9</b>
<b>APPENDIX: QUIZ APPLICATION OUTPUT .....</b>	<b>10</b>

## **Abstract**

This report outlines a quiz application that allows users to log in with a username, answer 10 random questions, and receive a score upon completion. The application features a timer for added challenge and displays the user's score at the end. Score history is maintained using a **Stack** data structure, which stores scores in a last-in, first-out manner. The questions and options are stored in an **ArrayList** for dynamic management. To display the leaderboard, a **Selection Sort** algorithm is used to sort scores in descending order, ensuring the highest scores appear at the top. This application effectively integrates key data structures and algorithms for an interactive and efficient quiz experience.

## Acknowledgement

I'd want to thank everyone who helped make this project a success. First and foremost, I'd like to express my gratitude to my supervisors, **Mr. Chhetra Bahadur Chhetri** and **Mr. Rikesh Shrestha**, for their invaluable guidance and continuous assistance. Their advice and support helped shape the direction and quality of this quiz application. I'd also like to thank my friends whose recommendations and support contributed significantly to the project's success. Their eagerness to share ideas and provide feedback helped to improve the application's functionality and design. In addition, I am grateful towards National College of Computer Studies (NCCS) for giving me the environment and resources I needed to do this project. The academic environment and the availability of research materials have made a huge difference in the caliber of this work.

## Chapter 1

### Introduction

#### 1.1 Organization Description

This project is being built for MeroSiksha, an organization in Kathmandu, Nepal that specializes in offering educational materials and technologies to improve professional and student learning experiences. The company specializes in developing interactive platforms that increase learning's effectiveness, accessibility, and engagement for a broad range of users. Its services cover a wide range of topics, such as professional certifications, programming languages, and academic disciplines.

The quiz app that is being built will be a seamless integration with the company's current offerings. Its multiple-choice questions are intended to assist users in honing their skills in programming. Because the questions and answers are randomly generated, the technology keeps the information interesting and provides a dynamic experience for every user.

#### 1.2 Objectives

- To develop a Java-based quiz application to provide an interactive and engaging learning experience.
- To provide a user-friendly platform for students to test their knowledge.
- To use data structures and algorithm for efficient storage and management of data.
- Support better understanding of concepts and facilitate exam preparation

#### 1.3 Scope and Limitation

##### 1.3.1 Scope

This project's scope includes creating an intuitive Java quiz application with a number of multiple-choice questions about Java programming. It will track user scores, provide replay functionality for better performance, and randomly select questions for a distinctive quiz experience. Important Java subjects including data types, exceptions, collections, and object-oriented programming will all be covered in the test. By letting users add their names, the program will provide a personalized experience. It will be made to function well on desktop platforms, emphasizing front-end development over more complex capabilities like database integration.

##### 1.3.2 Limitations

- **No Database Integration:** Neither user information nor test results are saved in a database by the quiz application.
- **Limited Customization:** Question categories and difficulty levels cannot be customized in the application. Restricted Question Set: The variety of quizzes is limited because the amount of questions is fixed.
- **Platform Dependency:** The program is not compatible with mobile platforms and is only intended to operate in desktop settings.
- **No Feedback or Explanation:** The system doesn't explain why a response is right or wrong.

## 1.4 Methodology

### 1.4.1 Project Framework

The project is organized into many stages to enable smooth execution and logical progression as follows:

- **Requirement Analysis:** First of all, I identified the quiz application's major features, user demands, and goals.
- **System Design:** In the next step, I laid out the framework of the application, including components such as the user interface.
- **Implementation:** Now, I translated the design into code by breaking it down into smaller parts like the question and answers bank, login, rules, main quiz section, score calculator, and randomization mechanism.
- **Testing:** Then, I performed functional and usability tests to guarantee that the program is bug-free and user-friendly.
- **Deployment:** Finally, I compiled it into an executable format and made it ready for use.

### 1.4.2 Data collection

In order to construct the quiz application, data collection includes:

- **Survey:** To collect a diverse set of Java-related questions suited for a multiple-choice format, I performed research on educational websites, textbooks, and online question banks.
- **User Feedback:** Early users of the application provided feedback that helped to improve the question bank, interface design, and overall operation.

### 1.4.3 Tools used

The following resources were utilized in the development of the quiz application:

- **Programming Language:** Java is the programming language used for application development and core logic.
- **Integrated Development Environment (IDE):** Java code was written and compiled using Apache Netbeans.
- **User Interface:** The graphical user interface (GUI) is designed using javax.swing library.
- **Data Structure and Algorithm Used:** Stack for storing the quiz score of various users and selection sort algorithm for sorting the users based on their scores.

## Chapter 2

### Analysis of Activities done and problem solved

#### 2.1 Requirement Analysis

The project's objectives, user requirements, and technical specifications were all carefully outlined throughout the requirement analysis phase. To make sure that the project development is in line with the expectations of the stakeholders and users, this phase is essential. Needs were gathered using a variety of techniques and divided into functional and non-functional needs.

##### 2.1.1 Requirement Collection Method

The following techniques were employed in order to gather the requirements for the quiz application project:

- **Interviews:** Insights regarding the features required for the application were obtained through conversations with instructors and prospective users.
- **Surveys:** To learn about the expectations, preferences, and technical expertise of potential users with regard to quiz applications, surveys were sent to them.
- **Research:** A benchmark for functional and non-functional needs was established by looking at academic materials, Java programming courses, and existing quiz applications.
- **Use Cases:** To make sure the project will satisfy user expectations, use case analysis was carried out to model the interactions between users and the system.

##### 2.1.2 Functional Requirement

The core functionalities of this project are as follows:

- **User Registration:** In order to monitor their progress, users should be able to register by entering their name.
- **Dynamic Question Display:** To give every user a different experience, the system should show questions at random.
- **Multiple-Choice Questions (MCQs):** There will be one right response for every question, with four possible answers.
- **Score tracking:** Users ought to be able to view their total score.
- **Timer:** To make the quiz more challenging, a timer should be set for each questions.
- **Score History:** To show the quiz score history of the users in sorted way.

##### 2.1.3 Non-functional Requirement

The requirements for smooth functioning of the system are as follows:

- **Reliability:** The system must function well, producing correct questions and answers without crashing.
- **Usability:** The system should have a simple user interface that allows for easy navigation.
- **Speed:** The system must respond quickly, especially when loading questions and displaying results. The application should load in a few seconds, minimizing wait time.
- **Maintainability:** New features should be easy to add, while current ones should be simple to maintain or modify.

## 2.2 Feasibility Study

The feasibility study assesses the viability of building and implementing the quiz application by considering a variety of factors such as technical, operational, economic, and scheduling feasibility. This guarantees that the project is manageable and will benefit both users and stakeholders.

### 2.2.1 Technical Feasibility

The main considerations include:

- **Current Resources:** The system will be built on Java, a programming language that can run on any basic computer with low hardware requirements. The system will require:
  - A computer or server with a basic processor (1.5 GHz or faster).
  - At least 2GB of RAM.
  - The Java Development Kit (JDK) plus any basic Integrated Development Environment (IDE), such as Eclipse or Apache Netbeans.
- **Existing Technology:** The technology needed to build and run the system is readily available, and it may use existing libraries to handle the features making the development process easier.
- **Technical Skills:** The project necessitates basic to intermediate understanding of Java programming, including object-oriented principles and user interface design.

### 2.2.2 Operational Feasibility

Operational feasibility determines if the proposed system can meet the expected user needs and is simple to run:

- **Meeting the Requirements:** The system's fundamental functionality, such as user login, quiz questions, randomization, and scoring, will be tailored to the operational demands of educational institutions and other users who require a quiz platform. The functionality complies with the requirements identified during the analysis process.
- **Ease of Use:** The system is intended to be simple, straightforward, and user-friendly. An easy-to-use interface will be created to ensure smooth interaction with a short learning curve.
- **Maintenance:** The system will be built with simple code, making it simple to maintain and update as needed.

### 2.2.3 Economic Feasibility

Economic feasibility assesses whether the project's benefits justify the expenditures associated with its development and operation.

- **System Design:** The main costs will be for software development (developer hours), testing, and user interface design. These costs will be kept low because to the usage of open-source technologies and available development resources.
- **Hardware and software:** The essential hardware (computers or servers) and software are inexpensive or free, lowering overall costs.
- **Operating Costs:** Because the project does not rely on server hosting or databases, the running costs are mostly focused on the resources and tools needed to develop and maintain the quiz application, such as personal computing devices.



### 2.2.4 Schedule Feasibility

Schedule feasibility evaluates the time required to complete the project and if the development will meet deadlines.

- **Deadline for development:** The project is anticipated to be completed in four stages over a period of 1 month.
  - **Phase 1:** Requirement analysis and system design.
  - **Phase 2:** Development of basic features, including user the quiz module and user login.
  - **Phase 3:** Integration and Testing.
  - **Phase 4:** Deployment.
- **Feasibility:** The timeline is possible given the resources available. Development and testing can proceed as planned, with plenty of time to make any necessary changes or improvements based on feedback.

### 2.3 Module Description

The quiz application is basic and user-friendly, making it accessible and entertaining for users. An outline of its components and features are as follows:

- **Accessibility:**
  - The system is available via a simple graphical user interface (GUI) that does not require any technical knowledge to navigate.
  - The application runs both locally and does not require server hosting or database integration.
  - Users can launch the application on any device that meets the minimal hardware requirements.
- **Customer Module:**
  - **Role:** Allows users to play quiz games and track their success.
  - **Features:** Enjoy a personalized quiz experience by starting with a randomly selected set of questions.
  - Displays questions with multiple-choice alternatives so that users can select their answers.
  - At the end of the quiz, the score is displayed, providing rapid feedback.
- Includes a user-friendly structure that will seamlessly guide consumers through the quiz.

- **Functionality of the System:**

- Use case: A user enters the application, selects the quiz, answers the questions, and receives a score summary at the end. They also get the option to replay the game.
- All functionalities are incorporated within the customer module, with an emphasis on simplicity and efficiency.

## 2.4 Testing

Unit testing was used to check individual component performance and assure system reliability. The following are the test cases created for each key functionality:

### 2.4.1 Test Cases

Component	Test Case	Expected Outcome
Quiz Initialization	Start the quiz and load the questions in random sequence.	The questions should load successfully, and their arrangement should be random.
Answer Submission	Select an answer for one question and move on to the next.	The selected answer is recorded, and the following question appears.
Score Calculation	Finish the quiz and see score summary.	The score should be calculated as per the total number of correct answers given.
Result Display	At the conclusion of the quiz, display feedback or a summary of the results.	The results should be clear and accurate.
Replay	After displaying the total score, there should be an option to replay.	The game restarts with randomized questions.

### 2.4.2 Testing methodology

- **Unit Testing:**
  - Each individual function (e.g., loading questions, scoring, and showing results) was tested separately to assure accuracy.
- **Integration Testing:**
  - Verified that all components (e.g., quiz initialization, question display, score computation) function together effortlessly.
- **Manual Testing:**
  - Manual testing of the application involved simulating user interactions to ensure an intuitive experience and error-free operation.

## Chapter 3

### Discussion and Conclusion

#### 3.1 Discussion

The quiz application was developed through an in-depth process that included planning, designing, and implementing a functional system that provides an interactive and interesting quiz experience. Key changes, such as randomizing question order and delivering quick feedback to users, increased the system's appeal. However, the lack of a backend database hampered the application's scalability and data retention. While the program achieves its primary goals of delivering quizzes and calculating scores, future upgrades could incorporate user administration capabilities, data storage for tracking progress, and analytics to improve the user experience even more.

#### 3.2 Conclusion

The quiz app successfully accomplishes its purpose of offering an accessible, user-friendly environment for playing quiz games. By emphasizing simplicity and functionality, the system provides a smooth experience for users to interact with multiple-choice quizzes and evaluate their results. Despite the lack of a database and administrative features, the application provides a solid foundation for future growth and usefulness.

#### 3.3 Lesson Learnt

Developing this system offered several valuable lessons:

- **Requirement Analysis:** Understanding user demands is critical to creating a system that meets expectations.
- **Design Thinking:** Structuring the system with clarity and user emphasis improves usability.
- **Testing Importance:** Iterative testing ensures that the system is strong and reliable.
- **Time Management:** Balancing development phases and meeting deadlines is critical for project success.
- **Adaptability:** Working without advanced technologies such as databases required innovative problem-solving to meet functional requirements.
- **Programming Skills:** Increased proficiency in algorithm design and user interface development.

#### 3.4 Future Recommendations

To improve the functioning and usefulness of the quiz app, the following suggestions can be considered:

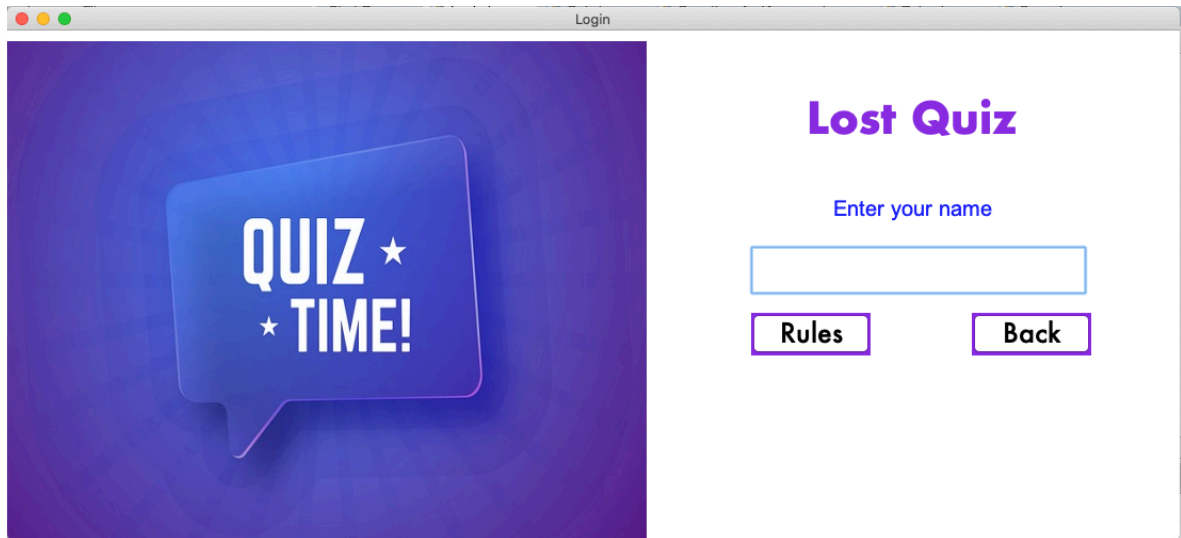
- **User Authentication and Profiles:** Implementing a login mechanism that allows users to create accounts and preserve their quiz progress. Also, individual dashboards can be provided for tracking performance history.
- **Category and Difficulty Selection:** Allow users to select quiz categories (such as sports, science, and history). Similarly, users can be provided quizzes with variable difficulty levels to accommodate different skill levels.

- **Question Feedback and Explanations:** Show explanations for right answers to improve the learning experience.
- **Database Integration:** Scalability can be achieved by using a database to hold user profiles, quiz data, and performance indicators.
- **Admin Panel:** Create an administrative interface to conveniently manage questions, categories, and users.

## References

- GeeksforGeeks. (n.d.). Java Programming Tutorials. Retrieved from <https://www.geeksforgeeks.org>
- "CodeWithHarry" for beginner-friendly Java development tutorials.
- "CodeForInterview" for Java Projects.
- Oracle. (n.d.). The Java Tutorials. Retrieved from <https://docs.oracle.com/javase/tutorial/>
- TutorialsPoint. (n.d.). Java Basics and Advanced Concepts. Retrieved from <https://www.tutorialspoint.com/java/index.htm>
- Stack Overflow. (n.d.). Retrieved from <https://stackoverflow.com>

## Appendix: Quiz Application Output



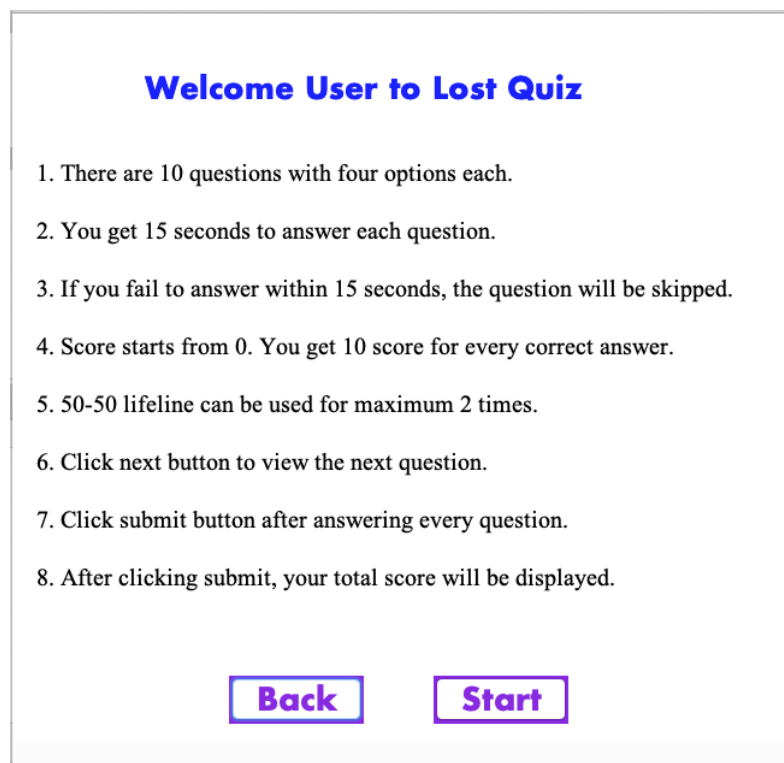
Login

# Lost Quiz

Enter your name

Rules Back

User Login Section

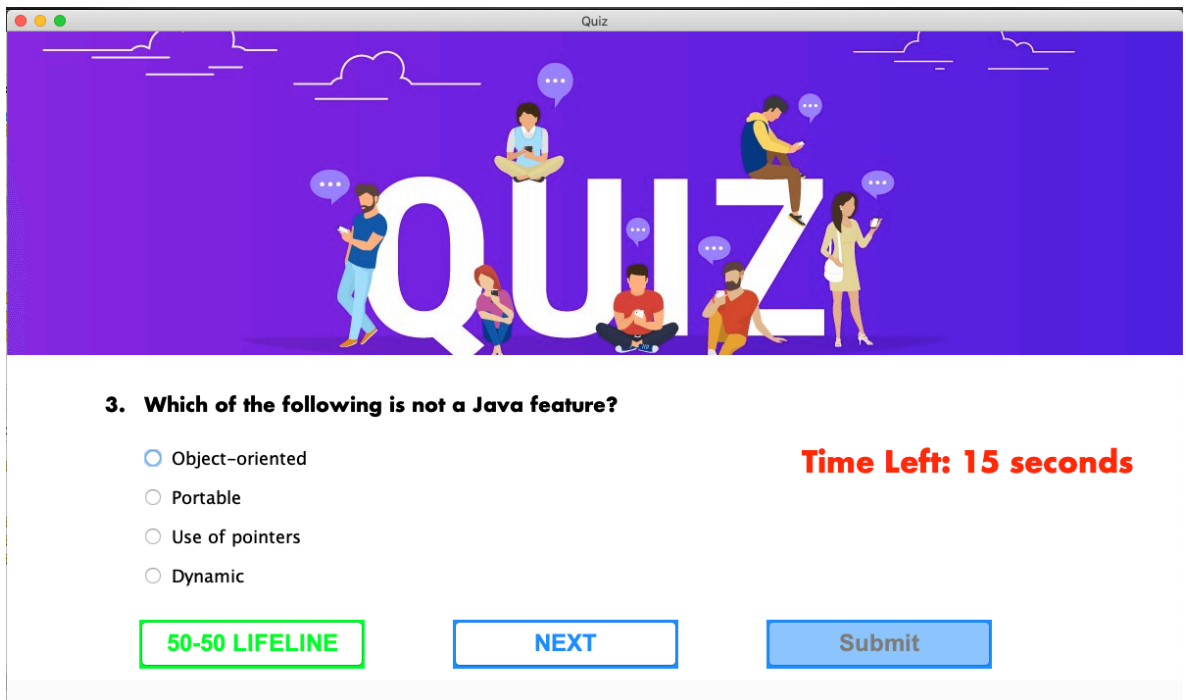


## Welcome User to Lost Quiz

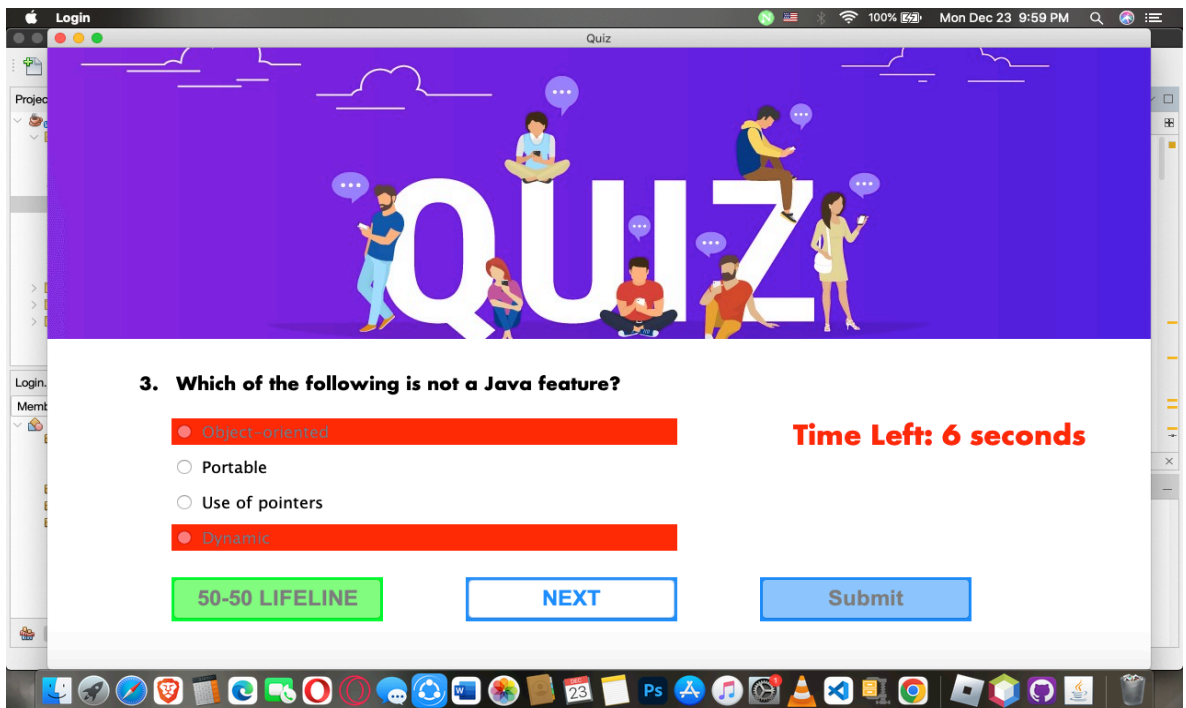
1. There are 10 questions with four options each.
2. You get 15 seconds to answer each question.
3. If you fail to answer within 15 seconds, the question will be skipped.
4. Score starts from 0. You get 10 score for every correct answer.
5. 50-50 lifeline can be used for maximum 2 times.
6. Click next button to view the next question.
7. Click submit button after answering every question.
8. After clicking submit, your total score will be displayed.

Back Start

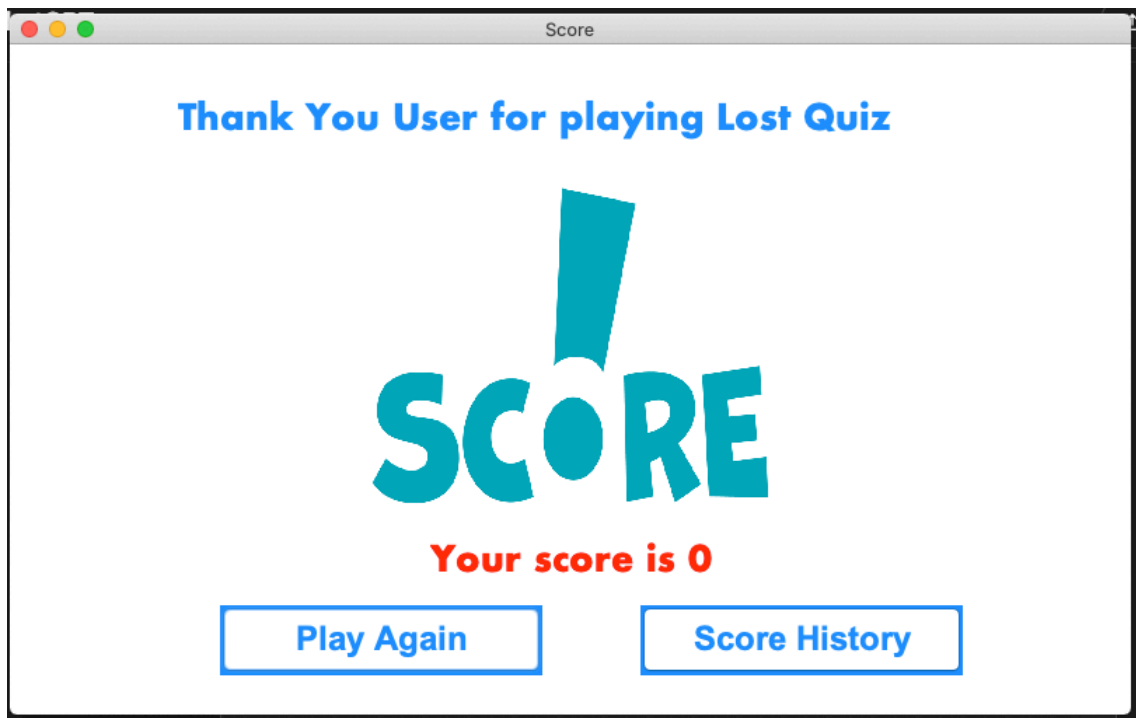
Rules of Game Section



Quiz Section



Use 50-50 Lifeline



**Show Score**

A screenshot of a web application window titled "Leaderboard". The window has a white background and a grey title bar with three colored buttons (red, yellow, green) on the left. Below the title bar is a search bar with the text "Top Score" and a red dropdown arrow on the right. Below the search bar is a table with three columns: "Username", "Score", and "Played On". The table contains three rows of data.

Username	Score	Played On
lost	40	25-04-2025 09:17 AM
sinmbf	30	25-04-2025 09:16 AM
Lost1	10	25-04-2025 09:23 AM

**Display Score History**