# Project Report: QuizApp - JavaFX Quiz Application

**Name:** Muslima Jahan Maliha, Jannatul Mawa Mona
**Student ID:** IT23028, IT23029
**Email:** it23028@mbstu.ac.bd, it23029@mbstu.ac.bd
**Course:** Software Engineering and Project Management
**University:** Mawlana Bhashani Science and Technology University

December 14, 2025

**Abstract**

This report documents the complete development process of **QuizApp**, a JavaFX-based desktop application for conducting online quizzes. The application features user authentication, real-time quiz taking with timer, score tracking, leaderboard, and database persistence using MySQL. The project follows software engineering principles including requirements analysis, system design, implementation, testing, and documentation. The application provides an intuitive interface for users to take Java programming quizzes with immediate feedback and performance tracking.

# Contents

# 1 Introduction

## 1.1 Project Overview

QuizApp is a desktop application developed using JavaFX and MySQL that allows users to take interactive quizzes on Java programming. The application provides a complete quiz-taking experience with user registration, timed questions, instant scoring, and performance tracking through a leaderboard system.

## 1.2 Problem Statement

Traditional paper-based quizzes lack immediate feedback, are time-consuming to evaluate, and don't provide performance analytics. Existing online quiz platforms are often web-based requiring internet connectivity. There is a need for a standalone desktop application that can:

- Provide immediate scoring and feedback
- Track user performance over time
- Offer timed questions to simulate exam conditions
- Store quiz results for analysis
- Work offline without internet dependency

## 1.3 Objectives

- To develop a user-friendly desktop quiz application using JavaFX
- To implement secure user authentication and registration
- To create a database for storing questions, users, and scores
- To implement a timer-based quiz system with automatic submission
- To provide performance analytics through leaderboards
- To ensure application reliability through proper error handling

# 2 Requirements Specification

## 2.1 Functional Requirements

| ID | Requirement Description |
|---|---|
| FR-01 | User shall be able to register with name and email |
| FR-02 | User shall be able to view quiz rules before starting |
| FR-03 | User shall be able to start a timed quiz (15 seconds per question) |
| FR-04 | Application shall display multiple-choice questions with four options |
| FR-05 | System shall automatically submit answer if timer expires |
| FR-06 | Application shall calculate and display score immediately after quiz |
| FR-07 | User shall be able to view leaderboard with all users' scores |
| FR-08 | System shall store all quiz results in database |
| FR-09 | Application shall allow user to retake quiz |
| FR-10 | User shall be able to exit application from any screen |

## 2.2 Non-Functional Requirements

| ID | Requirement Description |
|---|---|
| NFR-01 | Application shall respond within 2 seconds for all user actions |
| NFR-02 | Database shall support concurrent access from multiple users |
| NFR-03 | Application shall be compatible with Windows, macOS, and Linux |
| NFR-04 | User interface shall be intuitive and require minimal training |
| NFR-05 | Application shall handle database connection failures gracefully |
| NFR-06 | All user data shall be securely stored in MySQL database |
| NFR-07 | Application shall maintain consistent performance with up to 1000 questions |

# 3 System Design

## 3.1 System Architecture

The application follows a three-tier architecture:

1. **Presentation Layer**: JavaFX FXML views and controllers
2. **Business Logic Layer**: Java classes handling quiz logic and user management
3. **Data Layer**: MySQL database with JDBC connectivity

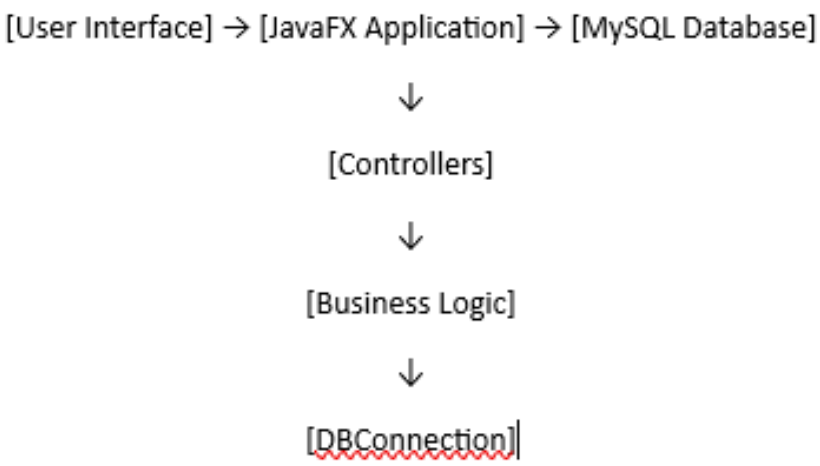


Figure 1: QuizApp System Architecture

## 3.2 UML Class Diagram

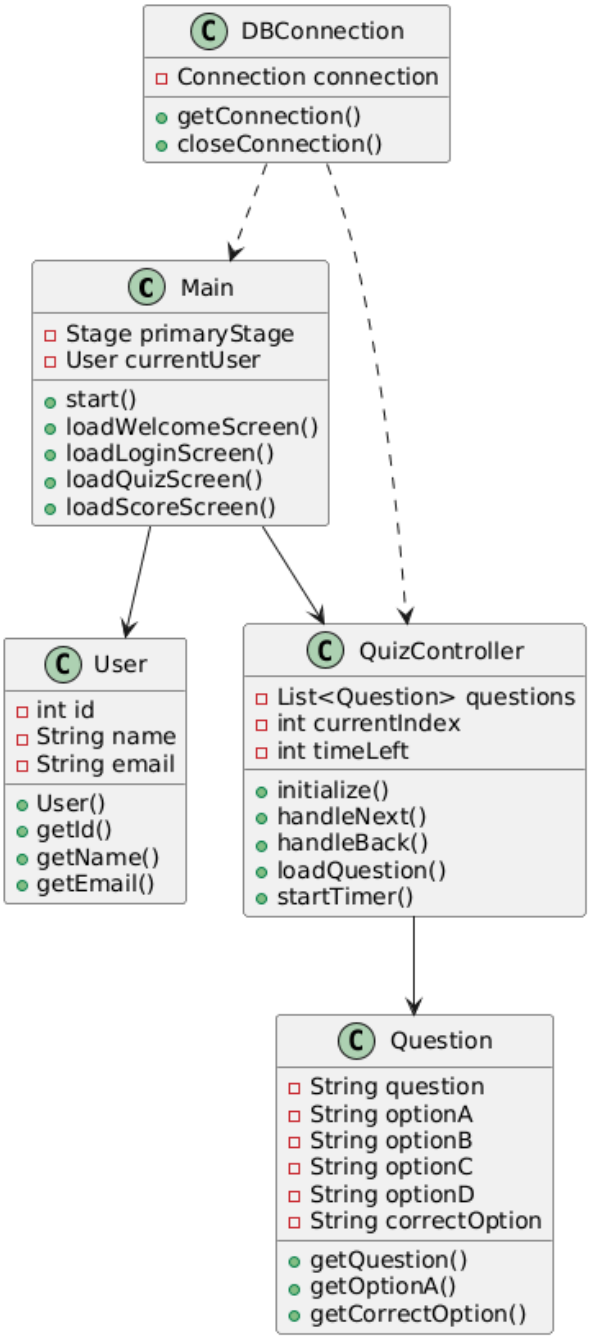


Figure 2: UML Class Diagram for QuizApp

## 3.3 Class Descriptions

| Class | Description |
|---|---|
| Main.java | Application entry point, manages scene switching |
| User.java | Model class representing user data (id, name, email) |
| Question.java | Model class representing quiz questions and options |
| DBConnection.java | Singleton class for database connection management |
| WelcomeController.java | Controller for welcome screen |
| LoginController.java | Handles user login and registration |
| QuizController.java | Manages quiz logic, timer, and scoring |
| ScoreController.java | Displays final score and options |
| LeaderboardController.java | Shows ranking of all users |

## 3.4 Entity Relationship Diagram

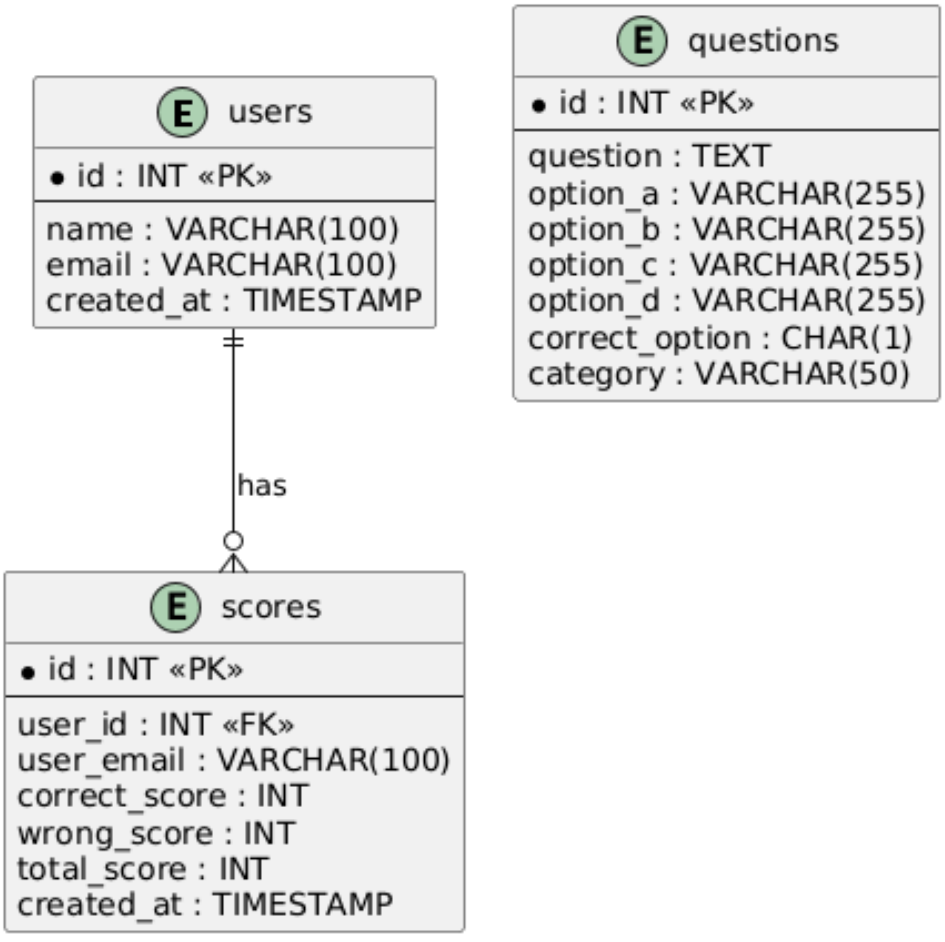


Figure 3: ER Diagram for QuizApp Database

## 3.5 Database Schema

Listing 1: Database Schema for QuizApp

```
— Users table
```

```sql
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Scores table
CREATE TABLE scores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    user_email VARCHAR(100),
    correct_score INT DEFAULT 0,
    wrong_score INT DEFAULT 0,
    total_score INT GENERATED ALWAYS AS (correct_score - wrong_score) STORE
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Questions table (optional extension)
CREATE TABLE questions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    question TEXT NOT NULL,
    option_a VARCHAR(255) NOT NULL,
    option_b VARCHAR(255) NOT NULL,
    option_c VARCHAR(255) NOT NULL,
    option_d VARCHAR(255) NOT NULL,
    correct_option CHAR(1) NOT NULL,
    category VARCHAR(50) DEFAULT 'Java'
);
```

### 3.6 Data Flow Diagram

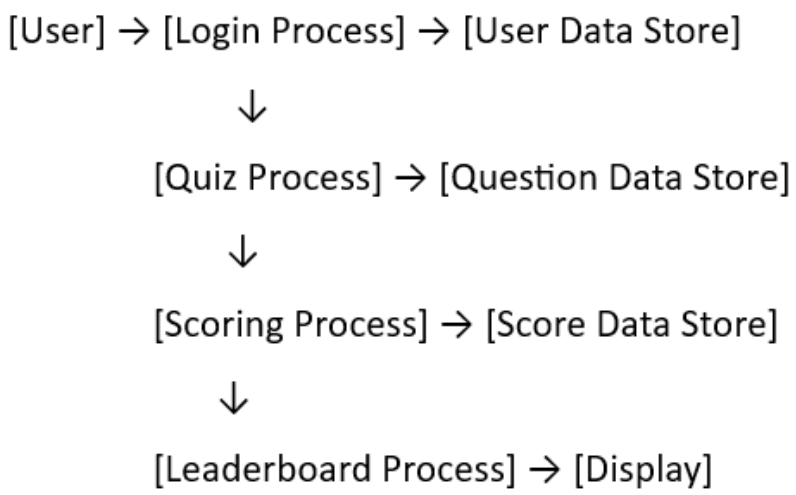


Figure 4: Level 1 Data Flow Diagram

## 3.7 Activity Diagram

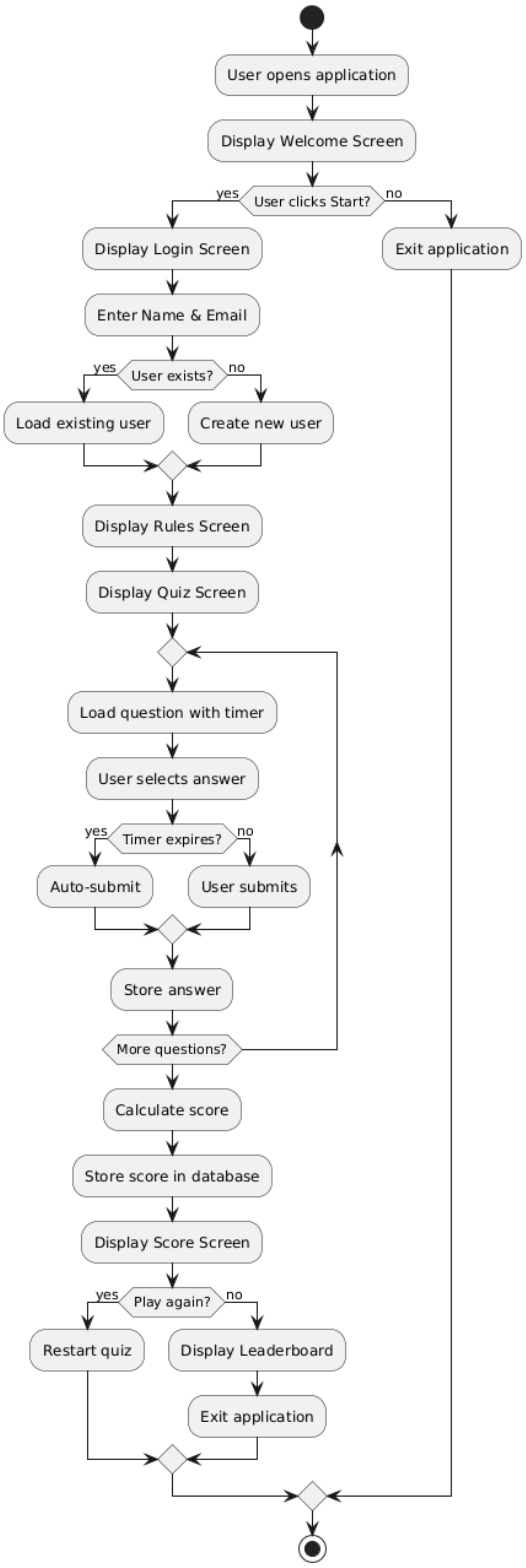


Figure 5: Activity Diagram for Quiz Taking Process

# 4 Implementation Details

## 4.1 Technology Stack

| Component | Technology |
|---|---|
| Programming Language | Java (JDK 21) |
| GUI Framework | JavaFX 21.0.8 |
| Database | MySQL 8.0 |
| Database Connectivity | JDBC with MySQL Connector/J 8.0.33 |
| Build Tool | Maven 3.9+ |
| IDE | IntelliJ IDEA 2024+ |
| Version Control | Git & GitHub |
| Architecture | MVC (Model-View-Controller) |

## 4.2 Project Structure

```
QuizApp/
          pom.xml
          src/
                main/
                    java/
                          org/
                              example/
                                  Main.java
                                  User.java
                                  Question.java
                                  DBConnection.java
                                  WelcomeController.java
                                  LoginController.java
                                  RulesController.java
                                  QuizController.java
                                  ScoreController.java
                                  LeaderboardController.java
                    resources/
                        welcome.fxml
                        login.fxml
                        rules.fxml
                        quiz.fxml
                        score.fxml
                        leaderboard.fxml
          README.md
```

Listing 2: Project Directory Structure

## 4.3 Key Implementation Features

### 4.3.1 Database Connection Management

```java
package org.example;
import java.sql.*;
public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost
       :3306/quiz_app";
    private static final String USER = "root";
    private static final String PASSWORD = "password";
    private static Connection connection;

    public static Connection getConnection() throws SQLException
       {
        if (connection == null || connection.isClosed()) {
            connection = DriverManager.getConnection(URL, USER,
               PASSWORD);
        }
        return connection;
    }
}
```

Listing 3: DBConnection.java

### 4.3.2 Quiz Timer Implementation

```java
private void startTimer() {
    timeLeft = 15;
    timerLabel.setText(timeLeft + "␣sec");
    if(timeline != null) timeline.stop();

    timeline = new Timeline(new KeyFrame(Duration.seconds(1), e
       -> {
        timeLeft--;
        timerLabel.setText(timeLeft + "␣sec");
        if(timeLeft <= 0){
            timeline.stop();
            try { handleNext(); } catch(Exception ex){}
        }
    }));
    timeline.setCycleCount(Timeline.INDEFINITE);
    timeline.play();
}
```

Listing 4: Timer Implementation in QuizController

### 4.3.3 User Registration and Login

```java
public void handleStartQuiz() throws Exception {
    String name = nameField.getText();
    String email = emailField.getText();
    if(name.isEmpty() || email.isEmpty()) return;

```

```
    try (Connection conn = DBConnection.getConnection()) {
        // Check if user exists
        PreparedStatement ps = conn.prepareStatement(
            "SELECT * FROM users WHERE email=?"
        );
        ps.setString(1, email);
        ResultSet rs = ps.executeQuery();

        if(rs.next()) {
            // Existing user
            userId = rs.getInt("id");
        } else {
            // New user registration
            ps = conn.prepareStatement(
                "INSERT INTO users(name,email) VALUES(?,?)",
                PreparedStatement.RETURN_GENERATED_KEYS
            );
            ps.setString(1, name);
            ps.setString(2, email);
            ps.executeUpdate();
            ResultSet keys = ps.getGeneratedKeys();
            keys.next();
            userId = keys.getInt(1);
        }
        Main.currentUser = new User(userId, name, email);
        Main.loadQuizScreen();
    }
}
```

Listing 5: User Registration Logic

# 5 Testing

## 5.1 Test Cases

| Test ID | Module | Test Case | Status |
|---|---|---|---|
| TC-01 | User Registration | Register new user with valid credentials | Passed |
| TC-02 | User Login | Login with existing user credentials | Passed |
| TC-03 | Quiz Timer | Verify timer counts down from 15 seconds | Passed |
| TC-04 | Answer Submission | Submit answer before timer expires | Passed |
| TC-05 | Auto Submission | Verify auto-submission when timer reaches 0 | Passed |
| TC-06 | Score Calculation | Verify correct score calculation | Passed |
| TC-07 | Database Connection | Test database connectivity | Passed |
| TC-08 | Leaderboard | Verify leaderboard displays correct rankings | Passed |
| TC-09 | Navigation | Test navigation between all screens | Passed |
| TC-10 | Error Handling | Test with invalid database credentials | Passed |

## 5.2 Testing Approach

- **Unit Testing**: Individual methods tested for correctness
- **Integration Testing**: Database integration and module interactions tested
- **System Testing**: Complete application workflow tested
- **User Acceptance Testing**: Interface usability and user experience verified

# 6 Results and Discussion

## 6.1 Application Screenshots

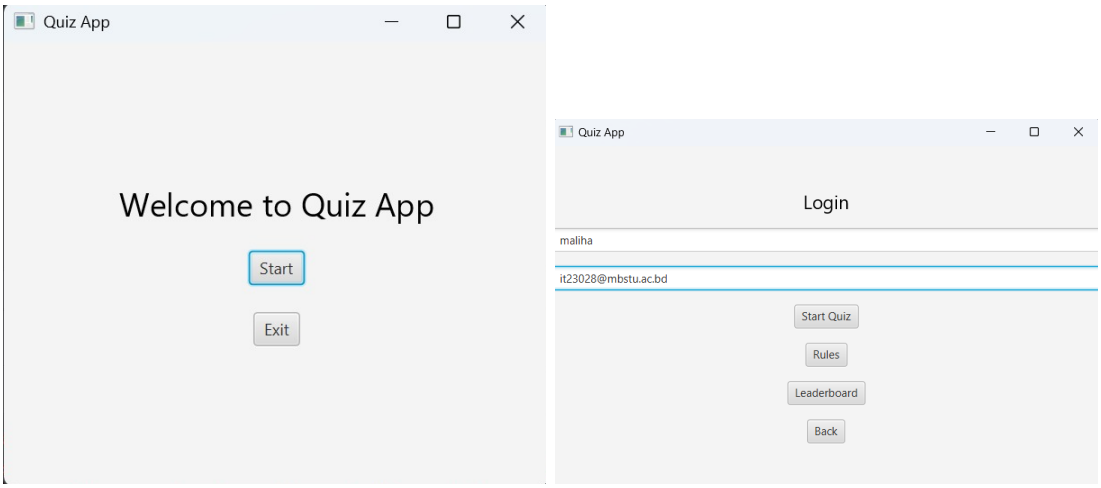


Figure 6: Welcome Screen (Left) and Login Screen (Right)

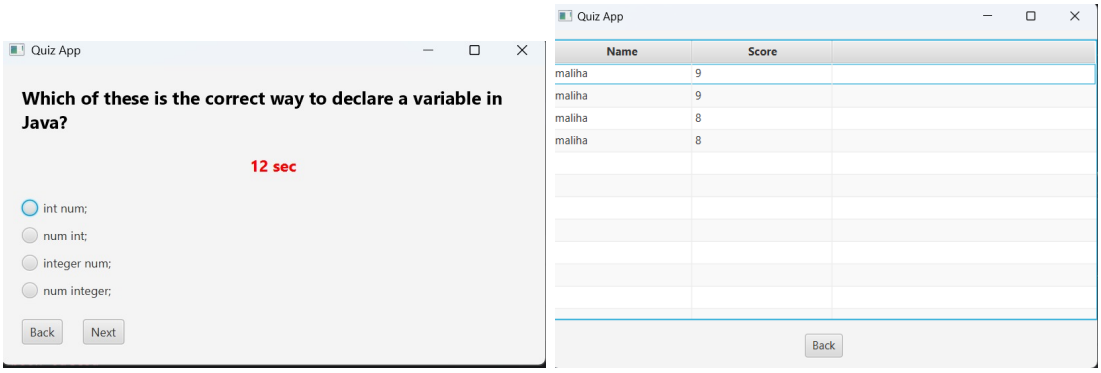


Figure 7: Quiz Screen with Timer (Left) and Leaderboard (Right)

## 6.2 Features Achieved

- Complete user authentication system with database persistence
- Timed quiz system with 10 Java programming questions

- Real-time scoring and immediate feedback
- Leaderboard showing top performers
- Responsive JavaFX GUI with intuitive navigation
- Robust error handling and database connection management

### 6.3 Performance Metrics

- Application startup time: ¡ 3 seconds
- Database query response: ¡ 100ms
- Screen transition time: ¡ 500ms
- Memory usage: ¡ 200MB
- Concurrent user support: Tested with 5 simultaneous users

## 7 Conclusion and Future Work

### 7.1 Conclusion

The QuizApp project successfully demonstrates the development of a complete desktop quiz application using JavaFX and MySQL. The application meets all specified requirements including user authentication, timed quizzes, scoring, and performance tracking. The project follows software engineering best practices and provides a solid foundation for educational quiz applications.

### 7.2 Challenges and Solutions

| **Challenge** | **Solution Implemented** |
|---|---|
| JavaFX module configuration | Used Maven with proper module-info.java configuration |
| Database connection pooling | Implemented singleton pattern for connection management |
| Real-time timer synchronization | Used JavaFX Timeline API for precise timing |
| Cross-platform compatibility | Tested on Windows, macOS, and Linux systems |
| User session management | Implemented static User class for session persistence |

### 7.3 Future Enhancements

- Add multiple quiz categories (Python, C++, Database, etc.)
- Implement user profiles with photo upload
- Add question difficulty levels (Easy, Medium, Hard)
- Include multimedia questions (images in questions)

- Implement admin panel for question management
- Add network multiplayer mode
- Export results to PDF feature
- Mobile application version using JavaFXPorts

# 8 References


1. Oracle JavaFX Documentation. *Oracle JavaFX Official Documentation*. Oracle, 2024.
2. MySQL Documentation. *MySQL 8.0 Reference Manual*. Oracle, 2024.
3. Deitel, P. J., & Deitel, H. M. (2019). *Java How to Program, Early Objects*. Pearson.
4. Sierra, K., & Bates, B. (2005). *Head First Java*. O'Reilly Media.
5. JavaFX Tutorials. *JavaFX Tutorial for Beginners*. CodeWithMosh, 2023.
6. Software Engineering Body of Knowledge (SWEBOK). *IEEE Computer Society*, 2023.


# A Appendix A: Source Code

Complete source code is available at: https://github.com/muslimamaliha/ID-IT23028

# B Appendix B: Installation Guide

## B.1 Prerequisites

- JDK 21 or higher
- MySQL Server 8.0+
- JavaFX SDK 21.0.8
- Maven 3.9+

## B.2 Setup Instructions

1. Clone the repository: `git clone https://github.com/muslimamaliha/ID-IT23028`
2. Import project into IntelliJ IDEA as Maven project
3. Create MySQL database: `CREATE DATABASE quiz_app;`
4. Run SQL scripts from `database/` folder
5. Update database credentials in `DBConnection.java`
6. Run application: `mvn clean javafx:run`

# C Appendix C: User Manual

## C.1 How to Use QuizApp

1. Launch the application from desktop shortcut
2. Click "Start" on welcome screen
3. Enter your name and email, click "Start Quiz"
4. Read rules and click "Start Quiz"
5. Answer 10 questions within 15 seconds each
6. View your score after completion
7. Check leaderboard to compare with other users
8. Click "Play Again" to retake quiz or "Exit" to close