QUIZ MANAGEMENT SYSTEM

CS23333 – Introduction to OOPS and JAVA Project Report

Submitted by

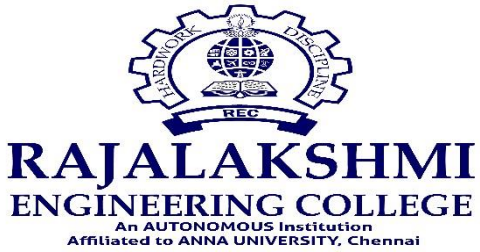
AADHI S - 231001001
GOKUL RAJ G - 231001047

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY
RAJALAKSHMI ENGINEERING COLLEGE

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM

(An Autonomous Institution)

BONAFIDE CERTIFICATE

Certified that this project titled **"QUIZ MANAGEMENT SYSTEM"** is the bona-fide work of **AADHI S (231001001)** and **GOKUL RAJ G (231001047)** who carried out the project work under my supervision.

SIGNATURE
Dr. Valarmathie P
HEAD OF THE DEPARTMENT

Information Technology
Rajalakshmi Engineering College.

SIGNATURE
Mrs.Usha S
COURSE INCHARGE
Assistant professor(S.G)
Information Technology
Rajalakshmi Engineering College

This mini project is submitted for CS23333– Introduction to Oops and Java held on _____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Abstract:

The primary objective of a JDBC-powered Quiz Management System is to leverage Java Database Connectivity (JDBC) to efficiently manage, store, and retrieve quiz-related data from a relational database. By integrating JDBC, the system ensures seamless data interaction between the application and the database, allowing for dynamic quiz creation, real-time updates, and secure data handling. This approach enables instructors to store quiz questions, answers, and user performance data in a database, which can be easily accessed, modified, and analyzed. The system aims to automate quiz grading, track participant progress, and offer instant feedback while maintaining data integrity, security, and scalability for managing large volumes of users and quizzes. Ultimately, the goal is to provide a robust, database-driven solution that improves the management, performance tracking, and delivery of quizzes in a secure and efficient manner.

TABLE OF CONTENTS

CHAPTER 1

1. Introduction:

A Quiz Management System is a software application designed to facilitate the creation, administration, and evaluation of quizzes in an efficient and automated manner. It provides a digital platform for educators, trainers, and administrators to create quizzes, manage question banks, and assess the performance of participants in real-time. The system is typically used in educational institutions, corporate training, and online learning platforms to streamline the process of quiz delivery and result analysis.

In addition to improving efficiency and accuracy in grading, a Quiz Management System also supports scalability, allowing for the management of a large number of quizzes, questions, and participants. It can be integrated with databases (such as through JDBC in Java) to store quiz data and user performance securely, providing real-time access to analytics and reports.

1.1 Purpose:

The purpose of this project is to develop an efficient and automated platform that simplifies the process of creating, managing, and evaluating quizzes in educational or training environments**.**The system aims to:

- Simplify Quiz Creation and Management
- Automate Grading and Evaluation.
- Enhance Learning and Feedback
- Track and Monitor Performance
- Improve User Engagement
- Enable Easy Data Access and Reporting

1.2 Scope of the Project:

Quiz Management System project involves creating a comprehensive platform for quiz creation, management, and evaluation, targeted at educational institutions or training environments. The system will have distinct user roles: Admin**,** Instructor, and Student. Admins will manage system configurations and user roles, instructors will create and assign quizzes, while students will take quizzes, view results, and track performance**.**

The system will use MySQL as the relational database to store and manage critical data such as user profiles, quiz questions, student responses, and performance records. JDBC (Java Database Connectivity) will be employed to connect Java to the MySQL database, ensuring seamless interaction between the application and the database for querying, retrieving, and updating data. The database will be structured with tables for users, quizzes, questions, answers, and results to maintain organized and easily accessible data.

1.3 Software Requirement:

1. Introduction

This Software Requirement Specification (SRS) **document outlines the requirements for the** Quiz **Management System** (QMS). The system will be built using **Java** for the application logic and **MySQL** for data storage. The primary purpose of this system is to manage quizzes, automate grading, and track student performance. The system will provide different roles, including Admin, Instructor, and Student, with distinct functionalities for managing quizzes, taking exams, and tracking results.

2. Overall Description

2.1 Product Perspective

The Quiz Management System will be a standalone application or web-based system designed to manage quizzes, administer exams, and generate detailed performance reports for educational institutions, organizations, and training centers. The system will provide features for creating, editing, and managing quizzes, as well as providing automated grading and performance analytics.

2.2 Product Features

- **Role-based access**: Admins, Instructors, and Students will have different levels of access.

- **Quiz Creation**: Instructors can create quizzes with multiple question types.
- **Question Bank**: A central repository for storing and categorizing questions.
- **Automated Grading**: The system will automatically grade student responses.
- **Performance Analytics**: Instructors and students will receive detailed performance reports.
- **User Authentication**: Secure login system for all users.
- **Feedback**: Students will receive immediate feedback on their performance.

2.3 User Classes and Characteristics

- **Admin**: Has full access to the system, including managing users (instructors, students) and viewing all quiz results and reports.
- **Instructor**: Can create and manage quizzes, view student results, and analyze performance.
- **Student**: Can take quizzes, view their results, and track their progress over time.

2.4 Assumptions and Dependencies

- The system assumes that the MySQL database is available and properly configured.
- The system will be built with Java and MySQL, and assumes users have the appropriate software installed.
- The system may be deployed as a desktop application or a web-based platform.

3. System Design Constraints

3.1 Technologies

- **Java** (for application logic and user interface)
- **MySQL** (for data storage)
- **JDBC** (for database connectivity)
- **JavaFX** or **Swing** (for the graphical user interface)
- **Apache Tomcat or Jetty** (if the system is web-based)

3.2 Data Storage

- The system will store user information, quizzes, questions, answers, and results in MySQL databases.
- The database schema will include tables for users, quizzes, questions, answers, and results.

4. External Interface Requirements

4.1 User Interfaces

- The system will provide a GUI for desktop users (using JavaFX or Swing) or a web interface for online access.

- The interface will be responsive and user-friendly, ensuring ease of use for admins, instructors, and students.

4.2 Hardware Interfaces

- No specific hardware interfaces are required beyond standard personal computers and network access for the web-based version.

4.3 Software Interfaces

- **MySQL** will be used for data storage.
- **JDBC** will be used to interface between the Java application and the MySQL database.

4.4 Communication Interfaces

- The system may use HTTPS for secure communication between the client and server (for web-based deployment).

5. Functional Requirements:

5.1 Administrator Module (AM):

- **User Management**: Add, edit, delete, and view users (instructors, students, admins).
- **Quiz Management**: Create, edit, delete, and approve/reject quizzes.
- **Reporting**: Generate and export reports on quiz results and user activity.
- **System Settings**: Configure system settings like quiz time limits and grading rules.
- **Security Management**: Manage user roles, reset passwords, and monitor security.
- **Content Approval**: Approve or reject quizzes and questions submitted by instructors.
- **Logs and Audit**: View system activity logs and track administrative actions.
- **Communication**: Send notifications and provide support resources to users.

5.2 Registered Users Module (RUM):

- **User Registration**: New users can register by providing details like username, email, and password.
- **Login/Logout**: Registered users can securely log in and log out of the system.

- **Profile Management**: Users can view and update their personal information and change their passwords.
- **View Assigned Quizzes**: Users (students) can view quizzes assigned to them by instructors.
- **Take Quizzes**: Students can attempt quizzes within specified time limits and submit their answers.
- **Track Results**: Users can view their quiz results and performance history.
- **Notifications**: Users receive notifications for upcoming quizzes, deadlines, and results.
- **Password Recovery**: Users can reset forgotten passwords through a secure recovery process.

5.3 Server Module (SM):

- SM acts as an intermediary between various modules and the database (DB).

- Receives requests from different modules and formats ps for display.

- Validates and executes requests received from other modules.

- Handles communication with the database, ensuring data consistency and integrity, especially regarding student details and phone evaluation.

6. Non-functional Requirements:

1. **Performance**
    - The system should support multiple concurrent users without significant performance degradation.
    - Response time for any user action (such as loading a quiz or submitting answers) should be under 3 seconds.
2. **Scalability**
    - The system should be able to scale horizontally to handle increasing numbers of users, quizzes, and data as the user base grows.
    - It should allow for additional server or database resources to be added without major changes to the system.
3. **Reliability**
    - The system should be 99.9% available, with minimal downtime for maintenance or unforeseen issues.
    - Data should be stored in a redundant, backed-up manner, ensuring protection from data loss.
4. **Security**
    - User data, including passwords and quiz results, must be stored securely using encryption and hashed passwords**.**
    - The system must support role-based access control (RBAC) to restrict unauthorized access to sensitive information.
    - All communications should be transmitted securely using SSL/TLS encryption.
5. **Usability**

- The system must have an intuitive and user-friendly interface, requiring minimal training for new users.
- It should be accessible on a variety of devices, including desktops, tablets, and mobile devices.

6. **Maintainability**
   - The system should have **a** modular architecture to facilitate easy updates, bug fixes, and feature additions.
   - Code should be well-documented to ensure easy understanding and maintenance by developers.
7. **Backup and Recovery**
   - The system must have an automated backup mechanism to prevent data loss.
   - There should be a clear and effective data recovery process in place in case of system failures or crashes.
8. **Compliance**
   - The system should comply with relevant data protection laws (such as GDPR or local regulations) regarding user privacy and data handling.
   - It must adhere to industry standards for security, including regular security audits.

CHAPTER 2

2. System Flow Diagram:

System Flow Diagram for Quiz Management System

1. **User Registration and Login**
   - **User Registration**: New users register with basic details (username, password, role).
   - **Login**: Registered users log in with username/email and password.
   - **Authentication**: The system validates the credentials.
     - **Valid Credentials**: Proceed to the appropriate dashboard.
     - **Invalid Credentials**: Display error and prompt for re-entry or password reset.
2. **Dashboard**
   - **Admin Dashboard**: Admin can manage users, quizzes, and view reports.
   - **Instructor Dashboard**: Instructors can create/edit quizzes, assign them to students.
   - **Student Dashboard**: Students can view and attempt assigned quizzes.
3. **Quiz Management**
   - **Admin/Instructor**: Admins and instructors create quizzes, add questions, and set time limits.
   - **Admin**: Approves/rejects quizzes created by instructors.
   - **Database**: Stores quiz data (questions, options, time limits).
4. **Taking Quizzes**
   - **Student**: View available quizzes assigned to them.
   - **Attempt Quiz**: Students answer questions within the time limit.
   - **Submit Quiz**: Once completed, the quiz is submitted.
5. **Results Management**
   - **Database**: Stores student quiz results, grades, and feedback.
   - **View Results**: After submission, students can view their scores and feedback.
6. **Reports and Notifications**
   - **Admin**: Generate detailed reports on quizzes, user activity, and results.
   - **Instructor**: View individual student results.
   - **Notifications**: Notify users about new quizzes, deadlines, and results.
7. **System Interaction with Database**
   - **User Data**: Stored in `Users` table (username, role, login details).
   - **Quiz Data**: Stored in `Quizzes`, `Questions`, and `Answers` tables.
   - **Results Data**: Stored in `Results` table (scores, feedback).
8. **Logging and Error Handling**
   - Logs and errors are tracked for monitoring and debugging purposes.

Flow Summary:

1. User Registration → Login
2. Admin/Instructor**:** Create/manage quizzes → Approve/reject quizzes → Generate reports.
3. Student**:** View/attempt quizzes → Submit quiz → View results.
4. Database**:** Stores all user, quiz, and result data**.**

5. Notifications**:** Inform users about quiz deadlines and results.

2.1. Use Case Diagram

1. Admin
2. Instructor
3. Student
4. Database (System)

Use Cases:

1. **Admin**:
    - **Manage Users**: Add, delete, or update user profiles (students, instructors).
    - **Manage Quizzes**: Create, approve, and delete quizzes.
    - **Generate Reports**: View and export reports on quiz results and user activity.
    - **Manage System Settings**: Configure settings like quiz duration, scoring rules, etc.
    - **View Logs**: Access system logs for auditing and monitoring.
2. **Instructor**:
    - **Create Quizzes**: Create new quizzes by adding questions and setting quiz parameters (time, number of attempts).
    - **Assign Quizzes**: Assign quizzes to students or specific groups.
    - **View Results**: Review student performance on quizzes assigned by them.
    - **Edit Quizzes**: Modify quizzes, update questions, or correct errors.
3. **Student**:
    - **Register/Login**: Register as a new user or log in with existing credentials.
    - **View Assigned Quizzes**: View quizzes assigned to them by instructors.
    - **Attempt Quizzes**: Take the quizzes, submit answers, and follow the time limits.
    - **View Results**: After submitting a quiz, students can view their results and feedback.
    - **Track Progress**: View past quiz results and progress.
4. **Database**:
    - **Store User Data**: Save user credentials and profile information.
    - **Store Quiz Data**: Store quiz questions, answers, and time limits.
    - **Store Results**: Record quiz results, including student scores and feedback.

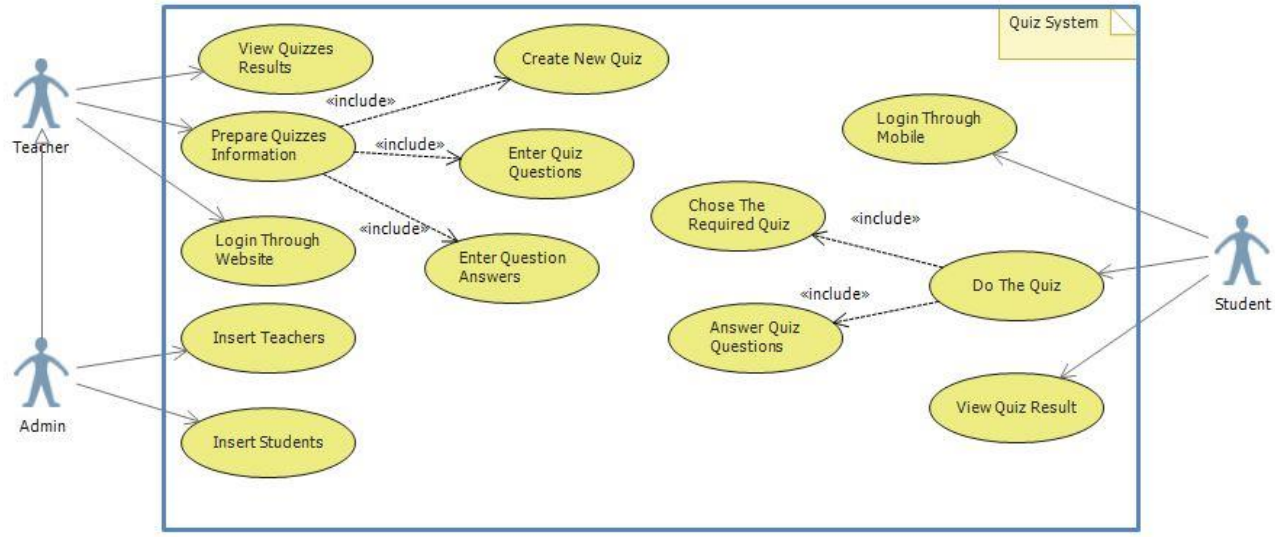


Figure 2.1.1 Use Case Diagram

Explanation of the Diagram:

- **Admin** manages user roles, quizzes, system settings, and generates reports.
- **Teacher** can create, assign, and edit quizzes, as well as view student results.
- **Student** can register, log in, attempt quizzes, and view their results.
- **Database** stores all user information, quiz data, and results.

This diagram illustrates the interactions and use cases in the **Quiz Management System**, showing how each role (Admin, Instructor, Student) interacts with the system and the database.

2.2. Entity - Relationship Diagram

E-R (Entity Relationship) Diagram is used to represent the relationship between entities in table.

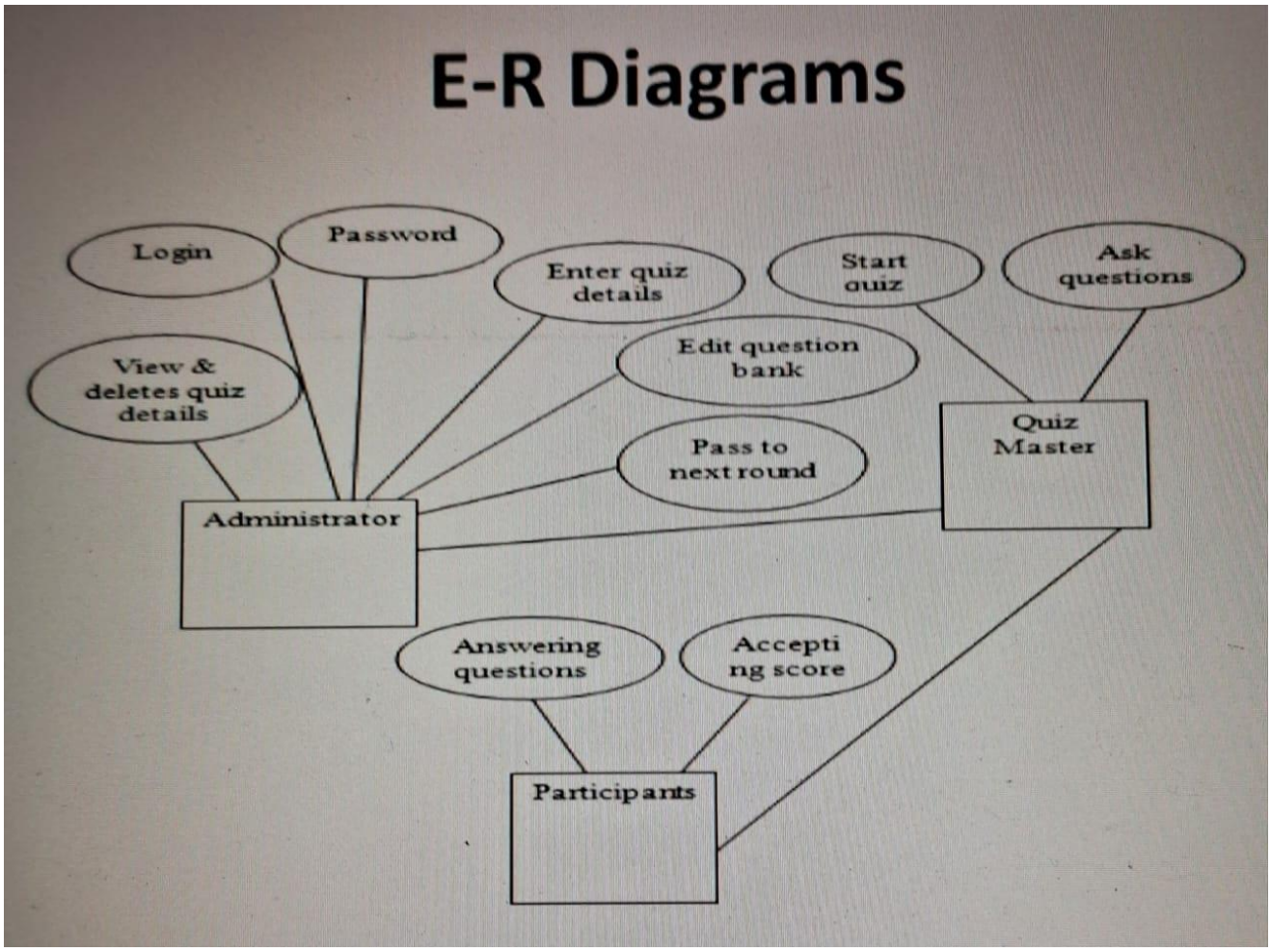


Figure 2.2.1 ENTITY RELATIONSHIP DIAGRAM

2.3 Data Flow Diagram

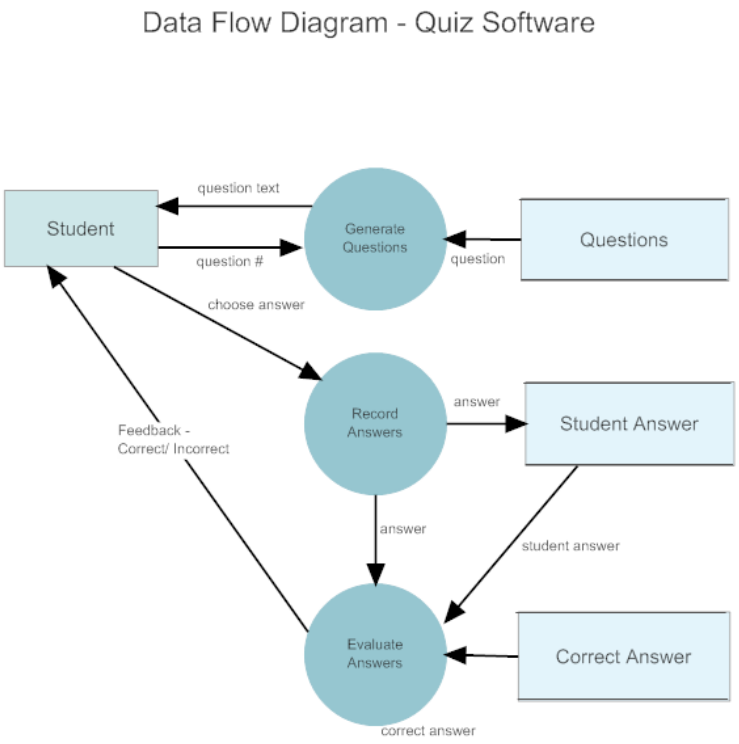


Figure 2.3 Data Flow Diagram

CHAPTER 3

3. Module Description:

1. User Management Module

Description:

This module handles all user-related functionalities, including registration, authentication, and management of user profiles.

- **Features**:
  - **User Registration**: Allows new users to create accounts (with roles like student, instructor, or admin).
  - **Login and Authentication**: Verifies user credentials and grants access based on the role (admin, instructor, or student).
  - **User Profile Management**: Allows users to view and edit their personal information.
  - **Role Management**: Differentiates the access and functionalities based on user roles (Admin, Instructor, Student)
- **Technologies Used**:
  - Java (Swing for UI, JDBC for database connection).
  - MySQL for storing user details and credentials.

2. Admin Module

Description:

The Admin module provides administrative control to manage users, quizzes, and system settings.

- **Features**:
  - **Manage Users**: Admin can add, edit, and delete users (students and instructors).
  - **Manage Quizzes**: Admin can approve/reject quizzes created by instructors and manage quiz details.
  - **Generate Reports**: Admin can generate reports on quiz results and user activities.
  - **System Settings**: Admin can configure system settings such as quiz time limits, scoring, and question types.
- **Technologies Used**:
  - Java (Swing for UI).
  - MySQL (for user and quiz management).

3. Instructor Module

Description:

This module is designed for instructors to create, assign, and manage quizzes for students.

- **Features**:
  - **Create Quizzes**: Instructors can create new quizzes, add questions, set time limits, and define scoring.
  - **Assign Quizzes**: Instructors can assign quizzes to specific students or groups of students.
  - **View Student Results**: Instructors can review quiz results for students and provide feedback.
- **Technologies Used**:
  - Java (Swing for UI).
  - MySQL (for storing quizzes, questions, and answers).

4. Student Module

Description:

The Student module enables students to view, attempt, and submit quizzes, as well as view results.

- **Features**:
  - **View Assigned Quizzes**: Students can view quizzes assigned to them by instructors.
  - **Attempt Quizzes**: Students can attempt quizzes by answering questions within the allotted time.
  - **Submit Quizzes**: After answering, students submit their quizzes for evaluation.
  - **View Results**: After quiz submission, students can view their scores and feedback.
  - **Track Progress**: Students can monitor their quiz performance and progress over time.
- **Technologies Used**:
  - Java (Swing for UI).
  - MySQL (for storing quiz attempts and results).

5. Quiz Management Module

Description:

This module deals with the creation, management, and evaluation of quizzes within the system.

- **Features**:
  - **Create Questions**: Allows instructors to create multiple-choice, true/false, or descriptive questions.
  - **Assign Time Limits**: Instructors can set a time limit for each quiz.
  - **Question Management**: Allows instructors to edit, update, and delete quiz questions.
  - **Score Calculation**: The system automatically calculates scores based on correct answers after quiz submission.
- **Technologies Used**:
  - Java (Swing for UI).
  - MySQL (for storing questions, answers, and quiz configurations).

6. Database Management Module

Description:

The database management module ensures that all system data (users, quizzes, questions, answers, results) is stored securely and efficiently in the database.

- **Features**:
  - **Store User Data**: Stores user credentials, profile details, and roles.
  - **Store Quiz Data**: Stores quizzes, questions, options, and correct answers.
  - **Store Results Data**: Stores quiz attempt results, scores, and feedback.
  - **Data Integrity**: Ensures data integrity and consistency through proper relational design and foreign keys.
- **Technologies Used**:
  - MySQL for data storage.
  - JDBC (Java Database Connectivity) for interacting with the MySQL database.

7. Reporting Module

Description:

This module generates reports for admins and instructors, providing insights into quiz performance and user activity.

- **Technologies Used**:
  - Java (Swing for UI).
  - Java libraries for report generation (JasperReports or similar).
  - MySQL (for fetching data for reports).

8. Security Module

Description:

The security module ensures that the system is protected from unauthorized access, data breaches, and other security vulnerabilities.

- **Features**:
  - Authentication and Authorization: Secure login system with role-based access control.
  - Password Encryption: User passwords are encrypted for secure storage.
  - Session Management: Manages user sessions to ensure secure access throughout the system.
- **Technologies Used**:
  - Java (for encryption and authentication logic).
  - MySQL (for secure storage of encrypted passwords).
  - Java Security APIs (for cryptographic functions).

CHAPTER 4

4. Design :

4.1 Implementation of Design:

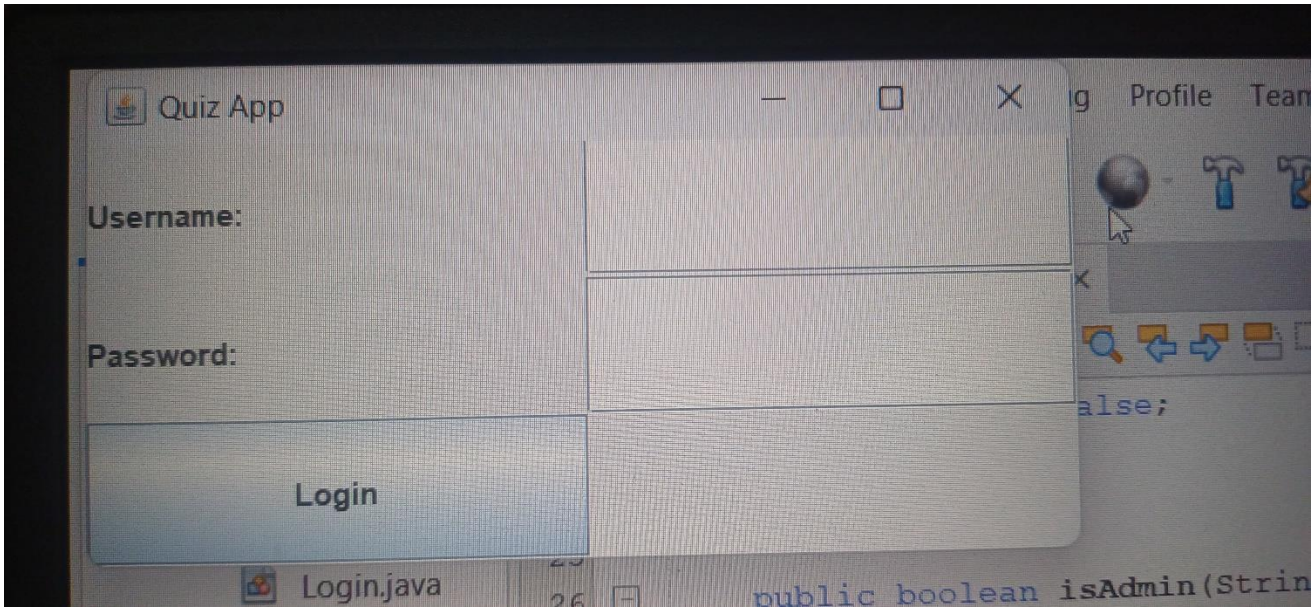


Figure 4.1.1 User Page

4.2 Database Design:

Column Name	Data Type
user_id	INT
username	VARCHAR(50)
password	VARCHAR(255)
role	ENUM('admin', 'user')
email	VARCHAR(100)
created_at	TIMESTAMP

Fig 4.2.1 User Table

4.3 Implementation Code:

```
package quizapp;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.List;

public class Main {
private static JFrame frame;
private static Login login;

public static void main(String[] args) {
login = new Login();
frame = new JFrame("Quiz App");

showLoginScreen();
}

private static void showLoginScreen() {
frame.getContentPane().removeAll();
frame.setLayout(new GridLayout(3, 2));

JLabel userLabel = new JLabel("Username:");
JTextField userText = new JTextField();
JLabel passwordLabel = new JLabel("Password:");
JPasswordField passwordText = new JPasswordField();
```

```
JButton loginButton = new JButton("Login");

frame.add(userLabel);
frame.add(userText);
frame.add(passwordLabel);
frame.add(passwordText);
frame.add(loginButton);

loginButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
String username = userText.getText();
String password = new String(passwordText.getPassword());

if (login.authenticate(username, password)) {
if (login.isAdmin(username)) {
showAdminScreen();
} else {
showUserScreen(username);
}
} else {
JOptionPane.showMessageDialog(frame, "Invalid login credentials!", "Error", JOptionPane.ERROR_MESSAGE);
}
}
});

frame.setSize(400, 200);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
frame.setVisible(true);
}

private static void showAdminScreen() {
frame.getContentPane().removeAll();
frame.setLayout(new FlowLayout());

JButton addQuizButton = new JButton("Add New Quiz");
JButton addQuestionsButton = new JButton("Add Questions");
JButton viewQuizzesButton = new JButton("View Quizzes");

frame.add(addQuizButton);
frame.add(addQuestionsButton);
frame.add(viewQuizzesButton);

addQuizButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
addNewQuiz();
}
});

addQuestionsButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
addQuestionsToQuiz();
}
});

viewQuizzesButton.addActionListener(new ActionListener() {
```

```
public void actionPerformed(ActionEvent e) {
viewQuizzes();
}
});

frame.setSize(400, 200);
frame.setVisible(true);
}

private static void addNewQuiz() {
String quizName = JOptionPane.showInputDialog(frame, "Enter the quiz name:");
try {
Connection conn = DatabaseConnection.getConnection();
String sql = "INSERT INTO quizzes (quiz_name) VALUES (?)";
PreparedStatement stmt = conn.prepareStatement(sql);
stmt.setString(1, quizName);

int rowsAffected = stmt.executeUpdate();
if (rowsAffected > 0) {
JOptionPane.showMessageDialog(frame, "New quiz added successfully!");
} else {
JOptionPane.showMessageDialog(frame, "Failed to add the quiz.");
}
} catch (SQLException e) {
e.printStackTrace();
JOptionPane.showMessageDialog(frame, "Error adding quiz to the database.");
}
}
```

```
private static void addQuestionsToQuiz() {
String quizIdStr = JOptionPane.showInputDialog(frame, "Enter the quiz ID to add questions:");
int quizId = Integer.parseInt(quizIdStr);

boolean addingQuestions = true;
while (addingQuestions) {
String questionText = JOptionPane.showInputDialog(frame, "Enter question text:");

String optionA = JOptionPane.showInputDialog(frame, "Enter option A:");
String optionB = JOptionPane.showInputDialog(frame, "Enter option B:");
String optionC = JOptionPane.showInputDialog(frame, "Enter option C:");
String optionD = JOptionPane.showInputDialog(frame, "Enter option D:");

String correctAnswer = JOptionPane.showInputDialog(frame, "Enter correct answer (A, B, C, or D):");

try {
Connection conn = DatabaseConnection.getConnection();
String sql = "INSERT INTO questions (quiz_id, question_text, option_a, option_b, option_c, option_d, correct_answer) VALUES (?, ?, ?, ?, ?, ?, ?)";
PreparedStatement stmt = conn.prepareStatement(sql);
stmt.setInt(1, quizId);
stmt.setString(2, questionText);
stmt.setString(3, optionA);
stmt.setString(4, optionB);
stmt.setString(5, optionC);
stmt.setString(6, optionD);
```

```
stmt.setString(7, correctAnswer);


int rowsAffected = stmt.executeUpdate();
if (rowsAffected > 0) {
JOptionPane.showMessageDialog(frame, "Question added successfully!");
} else {
JOptionPane.showMessageDialog(frame, "Failed to add question.");
}
} catch (SQLException e) {
e.printStackTrace();
JOptionPane.showMessageDialog(frame, "Error adding question to the database.");
}


int option = JOptionPane.showConfirmDialog(frame, "Do you want to add another question?",
"Add More?", JOptionPane.YES_NO_OPTION);
if (option == JOptionPane.NO_OPTION) {
addingQuestions = false;
}
}
}


private static void viewQuizzes() {
try {
Connection conn = DatabaseConnection.getConnection();
String sql = "SELECT * FROM quizzes";
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(sql);
```

```
StringBuilder quizzes = new StringBuilder("Available Quizzes:\n");
while (rs.next()) {
quizzes.append(rs.getInt("quiz_id")).append(".
").append(rs.getString("quiz_name")).append("\n");
}

JOptionPane.showMessageDialog(frame, quizzes.toString(), "Quizzes",
JOptionPane.INFORMATION_MESSAGE);
} catch (SQLException e) {
e.printStackTrace();
JOptionPane.showMessageDialog(frame, "Error retrieving quizzes from the database.");
}
}

private static void showUserScreen(String username) {
frame.getContentPane().removeAll();
frame.setLayout(new FlowLayout());

JButton startQuizButton = new JButton("Start Quiz");

frame.add(startQuizButton);

startQuizButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
startQuiz(username);
}
});
```

```
frame.setSize(400, 200);
frame.setVisible(true);
}

private static void startQuiz(String username) {

JOptionPane.showMessageDialog(frame, "Welcome, " + username + "! Let's start the quiz.");

String quizIdStr = JOptionPane.showInputDialog(frame, "Enter the quiz ID to start:");
int quizId = Integer.parseInt(quizIdStr);
Quiz quiz = new Quiz(quizId);
List<Question> questions = quiz.getQuestions();
int score = 0;

for (Question question : questions) {
String answer = JOptionPane.showInputDialog(frame, question.getQuestionText() + "\nA. " + question.getOptionA() + "\nB. " + question.getOptionB() + "\nC. " + question.getOptionC() + "\nD. " + question.getOptionD());
if (answer.equalsIgnoreCase(question.getCorrectAnswer())) {
score++;
}
}

JOptionPane.showMessageDialog(frame, "Your score: " + score + "/" + questions.size());
System.out.println("User " + username + " finished the quiz with a score of: " + score);
}
}
```

```
package quizapp;

import java.sql.*;
import java.util.*;

public class Quiz {
private int quizId;
private String quizName;
private List<Question> questions;

public Quiz(int quizId) {
this.quizId = quizId;
this.questions = new ArrayList<>();
loadQuiz();
}

public void loadQuiz() {
try {
Connection conn = DatabaseConnection.getConnection();
String sql = "SELECT * FROM questions WHERE quiz_id = ?";
PreparedStatement stmt = conn.prepareStatement(sql);
stmt.setInt(1, quizId);
ResultSet rs = stmt.executeQuery();

while (rs.next()) {
Question question = new Question(
rs.getInt("question_id"),
rs.getString("question_text"),
```

```
rs.getString("option_a"),
rs.getString("option_b"),
rs.getString("option_c"),
rs.getString("option_d"),
rs.getString("correct_answer")
);
questions.add(question);
}
} catch (SQLException e) {
e.printStackTrace();
}
}

public List<Question> getQuestions() {
return questions;
}
}
```

CHAPTER 5

5. Conclusion:

The Quiz Management System developed using Java and MySQL offers a comprehensive solution for managing and taking quizzes in a simple yet effective manner. The project demonstrates the integration of a user-friendly graphical interface with a robust database backend, allowing both admins and users to efficiently manage and participate in quizzes. The use of MySQL as the database ensures efficient data storage and retrieval for users, quizzes, and questions, while the Java Swing interface provides an intuitive experience for both admins and users. Additionally, the system employs fundamental programming concepts such as object-oriented design, database connectivity, and exception handling, making it a well-rounded project. Overall, this system not only meets its core objectives but also lays the groundwork for further enhancements. Features like adding time limits for quizzes, categorizing quizzes, improving user security, and adding reporting functionalities can be considered for future development. This project is an excellent demonstration of how Java and MySQL can be used together to create an interactive and practical application, with potential use cases in educational platforms, training programs, and online learning environments.

6. References:

1.Java Database Connectivity with MySQL - javatpoint

https://www.javatpoint.com/example-to-connect-to-the-mysql-database

2.https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/