E LEARNING SYSTEM

MINI PROJECT REPORT

Submitted by

Vaibhav Singh [RA2311003010167],
Smyan Arora [RA2311003010168]
Siddhant Tomar [RA2311003010173],
Sayak [RA2311003010174]

Under the Guidance of

Dr. S Ashwini

21CSC203P – ADVANCED PROGRAMMING PRACTICES

DEPARTMENT OF COMPUTING TECHNOLOGIES

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2024

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the 21CSC203P Advance Programming Practice course project report titled "**E-Learning System"** is the bonafide work done by **Vaibhav Singh [RA2311003010167], Smyan Arora [RA2311003010168], Siddhant Tomar [RA2311003010173], Sayak Satpati [RA2311003010174] of II Year/III Sem B.Tech(CSE)** who carried out the mini project under my supervision.

SIGNATURE
Faculty In-Charge
Faculty Name
Dr. S Ashwini
Department of Computing Technologies,
School of Computing,
SRM Institute of Science and Technology
Kattankulathur

SIGNATURE
Dr. NIRANJANA. G
Head of the Department
Professor and Head
Department of Computing Technologies,
School of Computing,
SRM Institute of Science and Technology
Kattankulathur

ABSTRACT

This project presents the development of an advanced E-Learning System to address the fragmented nature of educational resources that college students often face. By integrating various functionalities such as resource management, quizzes, real-time communication, and peer reviews into a user-friendly Java Swing GUI, and leveraging SQL for robust backend data management, the system enhances the overall educational experience for both students and teachers. This comprehensive platform provides a centralized solution for accessing verified resources, participating in interactive assessments, engaging in continuous learning discussions, and fostering a collaborative educational environment. The architecture ensures scalability, reliability, and high performance, setting a new standard for educational platforms and paving the way for future enhancements to meet the evolving needs of modern education.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honourable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavours. We would like to express my warmth of gratitude to our **Registrar Dr. S.PONNUSAMY,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.GOPAL,** for bringing out novelty in all executions. We would like to express my heartfelt thanks to our Chairperson, School of Computing **Dr. REVATHI VENKATARAMAN,** for imparting confidence to complete my course project.

We extend my gratitude to our Associate Chairperson **Dr. PUSHPALATHA. M**, **Professor** and our **HOD Dr. NIRANJANA. G, Professor and Head, Department of Computing Technologies** for their Support.

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu.G, Professor, Department of Data Science and Business Systems** and **Course Coordinators Dr. MANJULA. R, Assistant Professor and Dr. N.A.S VINOTH, Assistant Professor** for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr. S Ashwini, Assistant Professor, Department of Computing Technologies** for her assistance, timely suggestion and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1.Introduction

In the dynamic and continually evolving landscape of education, the need for a centralized, efficient, and comprehensive platform for resource management and student engagement has never been more pronounced. The E-Learning System project was conceived to address the challenges faced by college students, such as the fragmentation of educational resources across multiple platforms like WhatsApp groups and Google Classrooms. This project provides an integrated solution that consolidates various educational tools and resources into a single, user-friendly platform.

This system is developed with a focus on enhancing the accessibility, efficiency, and collaborative aspects of the learning process. It recognizes the necessity for an all-encompassing platform that not only facilitates seamless access to educational materials but also fosters an environment of continuous interaction and mutual support among students and educators. The E-Learning System enables students to access verified resources, participate in interactive quizzes, engage in real-time discussions, and perform peer reviews, all within a cohesive interface.

The backend of this system is powered by SQL, which securely manages and stores extensive datasets, including user information, quiz results, and chat histories. The integration of SQL ensures data integrity, quick retrieval, and efficient data manipulation, which are critical for handling the large volumes of information typical in an educational environment.

In an era where digital transformation is reshaping educational practices, this E-Learning System stands at the forefront, bridging the gap between traditional learning methods and modern technological advancements. It aims to enhance operational efficiency, improve resource management, and foster a collaborative and engaging learning environment for both students and educators. This project exemplifies the effective use of Java and SQL to develop a robust, scalable, and user-centric educational platform.

2.Literature Survey

Literature Survey: A Centralized E-Learning System for Enhanced Student Resource Access and Collaborative Learning

Introduction

The growing number of online learning platforms presents both opportunities and challenges for college students. While offering flexibility and convenience, a fragmented landscape of resources across various platforms (e.g., WhatsApp groups, Google Classrooms) can hinder a student's ability to find reliable and comprehensive materials. This research addresses this issue by proposing a centralized e-learning system specifically designed to enhance resource access and promote collaborative learning among college students.

Problem Statement

College students often face challenges in efficiently accessing course materials and finding reliable resources to support their studies. Dispersed resources across platforms like WhatsApp groups and Google Classrooms create confusion and hinder students' ability to locate high-quality materials. Furthermore, traditional learning methods often limit opportunities for peer interaction and collaborative learning, potentially hindering the development of critical thinking and communication skills.

Literature Review

This review explores the benefits of centralized e-learning systems and collaborative learning methods in higher education.

2.1 Benefits of Centralized E-Learning Systems

- Improved Resource Management: Centralized platforms allow for organized storage and accessibility of course materials, ensuring students have a single point of reference for all learning resources. Research by [Sun et al., 2018] suggests that centralized LMS (Learning Management Systems) can significantly improve student satisfaction with the availability and organization of learning materials.
- Enhanced Communication: Features like chatboxes and discussion forums can facilitate communication between students and instructors in a structured environment. Studies by [Liu et al., 2017] highlight the effectiveness of online discussion forums in fostering student engagement and knowledge construction.

2.2 Collaborative Learning Strategies

- Peer Review: Peer review, where students provide feedback on each other's work, fosters critical thinking skills and exposes students to diverse perspectives. Studies by [Topping, 2009] demonstrate that peer review can lead to improved learning outcomes and self-assessment skills.
- Collaborative Problem-Solving: Collaborative features like chatboxes or online workspaces can enable students to work together on assignments and problem-solving tasks. According to [Johnson et al., 2008], collaborative learning environments promote teamwork, communication skills, and deeper understanding of course material.

Proposed System Features

- Student Dashboard: Allows students to view uploaded learning materials, take quizzes, interact with peers and instructors through chatboxes, and participate in peer review of assignments.
- Teacher Dashboard: Enables instructors to upload course materials, create quizzes, address student questions through chat, and review submitted assignments.
- Peer Review Feature: Provides a platform for students to review each other's assignments, fostering collaboration and critical thinking.

Expected Benefits

- Streamlined Resource Access: Students can access all course materials in one central location, reducing time spent searching for resources.
- Enhanced Communication and Collaboration: Chatboxes and peer review features promote interaction between students and instructors, fostering collaborative learning experiences.
- Improved Learning Outcomes: The system encourages student engagement and active learning, potentially leading to better academic achievement.

Conclusion

This literature review highlights the advantages of centralized e-learning systems and collaborative learning strategies. The proposed system aims to address the challenges of dispersed resources and limited interaction within existing online learning platforms. By providing a central platform for organized resource access, communication, and peer review, the system has the potential to optimize student learning experiences and foster collaboration among college students.

3.Requirement Analysis

3.1 Functional Requirements

3.1.1 Resource Management

- File Upload and View: Teachers can upload course materials. Students can view and download these resources.
- Assignment Submission and Peer Review: Students can submit assignments, which are viewable by other students for peer reviews.

3.1.2 Quiz Management

- Create and Manage Quizzes: Teachers can create, edit, and delete quizzes.
- Take Quizzes: Students can participate in quizzes, receive instant results, and review past quiz attempts.

3.2 Communication

3.2.1 Chatbox

- Real-time chat functionality for students and teachers to discuss doubts and share information.

3.3 Reporting

3.3.1 Performance Tracking

- Generate reports on student performance in quizzes and assignments.

3.3.2 Engagement Reports

- Track participation in chat discussions and peer reviews.

3.4 Authentication and Authorization

3.4.1 Secure Login

- Enable secure access with unique login credentials for students and teachers.

3.5 Non-Functional Requirements

3.5.1 Performance

- The system should handle up to [X] concurrent users without performance degradation.

3.5.2 Security

- Sensitive data such as personal information, quiz results, and chat logs must be encrypted.
- Implement role-based access control to ensure only authorized users can access specific functionalities.

3.5.3 Usability

- The GUI must be intuitive, user-friendly, and accessible to users with basic computer skills.
- Provide clear instructions and help sections to guide users through different features.

3.5.4 Scalability

- The system should be capable of handling an increasing number of users and data without compromising performance.
- Allow for easy integration of additional features and modules in the future.

By addressing these functional and non-functional requirements, the E-Learning System aims to provide a comprehensive, secure, and user-friendly platform that enhances the educational experience for both students and educators. This analysis ensures that the system is not only functional but also efficient, scalable, and reliable, meeting the needs of a modern educational environment.

4.Architecture and Design

Architecture

The E-Learning System employs a client-server architecture to provide a seamless and integrated educational experience. The client application (Java Swing GUI) interacts with a server-based SQL database to manage user interactions, educational resources, quizzes, and communication. The architecture ensures efficient data handling and robust performance.

Architecture Diagram

The architecture diagram consists of

- Client-Side (Java Swing GUI)
    - Frontend Interface: Provides a user-friendly graphical interface for students and teachers to manage educational activities.
    - Features: Enables functionalities such as viewing uploaded files, taking quizzes, submitting assignments, participating in chat discussions, and conducting peer reviews.

- Server-Side (SQL Database):
    - Backend Data Management: Handles the storage and retrieval of all educational data, including user information, quiz details, assignment submissions, chat histories, and peer reviews.
    - Data Integrity: Ensures secure and efficient management of large datasets, maintaining data integrity and consistency.

- Connection Layer (JDBC):
    - Database Connectivity: Utilizes JDBC (Java Database Connectivity) to establish and manage communication between the Java Swing GUI and the SQL database.
    - Query Execution: Executes SQL queries for fetching, updating, and storing data, ensuring real-time synchronization between the client and the server.

Use Case Diagram

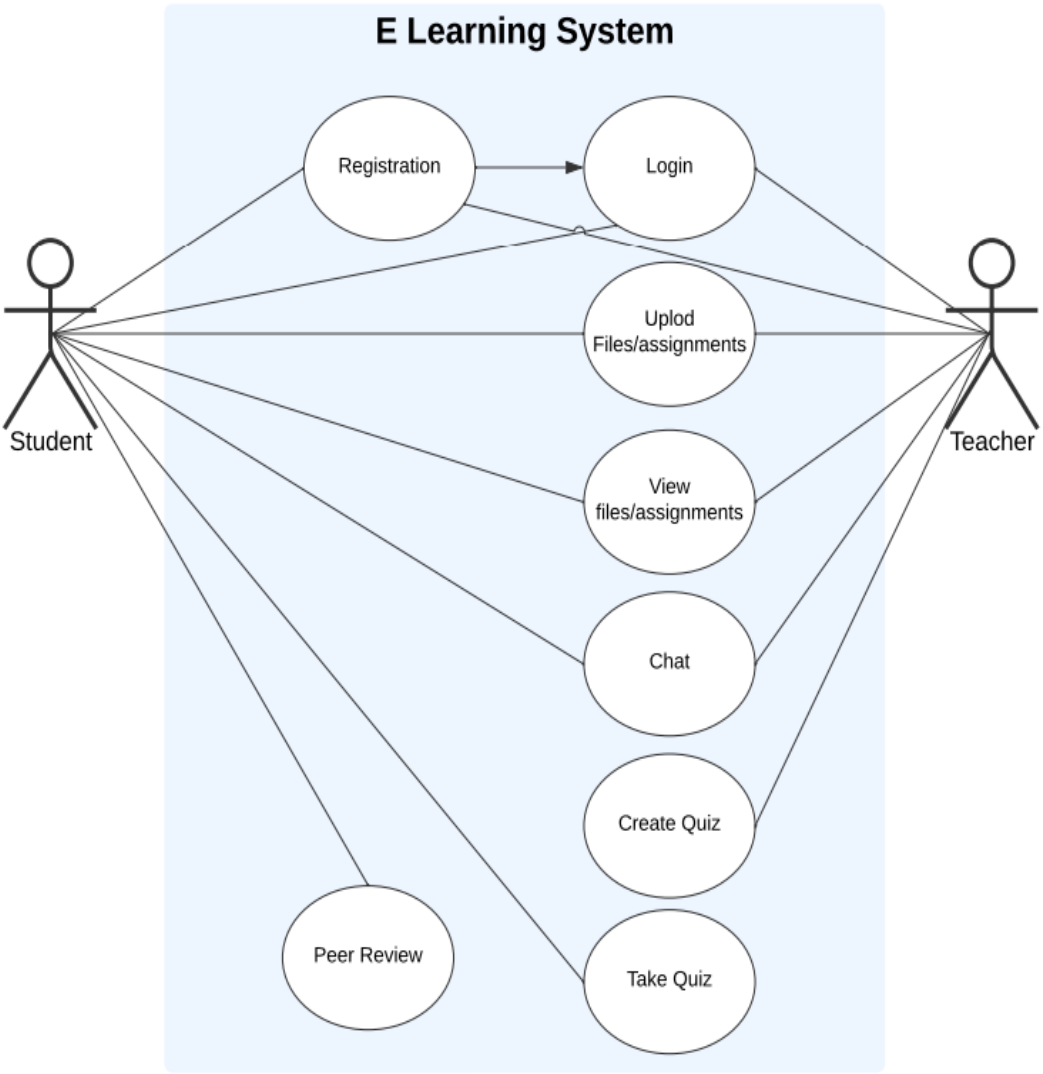


fig.(i) Use Case Diagram for E learning System

5.Implementation

1)Registration Page

```java
package dashboard;

import database.DBConnection;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class RegistrationForm extends JFrame {
    private JTextField usernameField;  3 usages
    private JPasswordField passwordField;  3 usages
    private JComboBox<String> roleComboBox;  3 usages
    private JButton registerButton;  3 usages

    public RegistrationForm() {  1 usage
        setTitle("User Registration");
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new GridBagLayout()); // Use GridBagLayout for proper alignment
        GridBagConstraints gbc = new GridBagConstraints();
        mainPanel.setBackground(Color.LIGHT_GRAY);

        // Create form elements
        JLabel usernameLabel = new JLabel( text: "Username:");
        usernameField = new JTextField( columns: 15);

        JLabel passwordLabel = new JLabel( text: "Password:");
        passwordField = new JPasswordField( columns: 15);

        JLabel roleLabel = new JLabel( text: "Role:");
        String[] roles = {"Student", "Teacher"};
```

```
        roleComboBox = new JComboBox<>(roles);

        registerButton = new JButton( text: "Register");
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);

        // Layout setup
        // Username label
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.EAST; // Align to the right
        mainPanel.add(usernameLabel, gbc);

        // Username field
        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.WEST; // Align to the left
        mainPanel.add(usernameField, gbc);

        // Password label
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.EAST; // Align to the right
        mainPanel.add(passwordLabel, gbc);

        // Password field
        gbc.gridx = 1;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.WEST; // Align to the left
        mainPanel.add(passwordField, gbc);
```

```
        // Role label
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.EAST; // Align to the right
        mainPanel.add(roleLabel, gbc);

        // Role combo box
        gbc.gridx = 1;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.WEST; // Align to the left
        mainPanel.add(roleComboBox, gbc);

        // Login button
        gbc.gridx = 1;
        gbc.gridy = 3;
        gbc.anchor = GridBagConstraints.CENTER; // Center the button
        mainPanel.add(registerButton, gbc);
        getContentPane().setBackground(Color.CYAN);  // Set the entire window's background colo

        // Add the mainPanel to the JFrame
        add(mainPanel);

        // Register button action listener
        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                String role = (String) roleComboBox.getSelectedItem();
                if (username.isEmpty() || password.isEmpty()) {
```

```
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Username or password cannot be e
            } else {
                // Save user to the database
                registerUser(username, password, role);
            }
        }
    });

    // JFrame settings
    setSize( width: 600,  height: 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

// Method to register the user in the database
private void registerUser(String username, String password, String role) {  1 usage
    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        // Get a connection to the database
        connection = DBConnection.getConnection();

        // SQL query to insert the user data into the 'users' table
        String insertQuery = "INSERT INTO users (username, password, role) VALUES (?, ?, ?)";
        preparedStatement = connection.prepareStatement(insertQuery);
        preparedStatement.setString( parameterIndex: 1, username);
```

```
        // Execute the query
        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "User registered successfully!");
        } else {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "Failed to register the user.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog( parentComponent: null,  message: "An error occurred while registering the user
    } finally {
        // Close the resources
        DBConnection.closeConnection(connection);
        try {
            if (preparedStatement != null) preparedStatement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public static void main(String[] args) {
    new RegistrationForm();
}
```

2)Login Page

```java
package dashboard;

import database.DBConnection;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;



public class Login extends JFrame {
    private JTextField usernameField;  3 usages
    private JPasswordField passwordField;  3 usages
    private JComboBox<String> roleComboBox;  3 usages
    private JButton loginButton;  3 usages

    public Login() {  1 usage
        setTitle("User Login");

        Image backgroundImage;
        try {
            BufferedImage originalImage = ImageIO.read(new File( pathname: "C:\\Users\\vaibh\\Desktop\\login.jpg"));
            backgroundImage = originalImage.getScaledInstance( width: 600,  height: 400, Image.SCALE_SMOOTH); // Scale
            System.out.println("Background image loaded successfully.");
        } catch (IOException e) {
```

```java
            e.printStackTrace();
            backgroundImage = null;
            System.out.println("Failed to load background image.");
        }


        BackgroundPanel mainPanel = new BackgroundPanel(backgroundImage);
        mainPanel.setLayout(new GridBagLayout()); // Use GridBagLayout for proper alignment
        GridBagConstraints gbc = new GridBagConstraints();
        mainPanel.setBackground(Color.LIGHT_GRAY);


        // Create form elements
        JLabel usernameLabel = new JLabel( text: "Username:");
        usernameField = new JTextField( columns: 15);

        JLabel passwordLabel = new JLabel( text: "Password:");
        passwordField = new JPasswordField( columns: 15);

        JLabel roleLabel = new JLabel( text: "Role:");
        String[] roles = {"Student", "Teacher"};
        roleComboBox = new JComboBox<>(roles);

        loginButton = new JButton( text: "Login");

        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);

        // Username label
        gbc.gridx = 0;
        gbc.gridy = 0;
```

```
        gbc.anchor = GridBagConstraints.EAST; // Align to the right
        mainPanel.add(usernameLabel, gbc);

        // Username field
        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.WEST; // Align to the left
        mainPanel.add(usernameField, gbc);

        // Password label
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.EAST; // Align to the right
        mainPanel.add(passwordLabel, gbc);

        // Password field
        gbc.gridx = 1;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.WEST; // Align to the left
        mainPanel.add(passwordField, gbc);

        // Role label
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.EAST; // Align to the right
        mainPanel.add(roleLabel, gbc);

        // Role combo box
        gbc.gridx = 1;
```

```
        mainPanel.add(roleComboBox, gbc);

        // Login button
        gbc.gridx = 1;
        gbc.gridy = 3;
        gbc.anchor = GridBagConstraints.CENTER; // Center the button
        mainPanel.add(loginButton, gbc);
        getContentPane().setBackground(Color.CYAN);
        add(mainPanel);

        // Login button action listener
        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                String role = (String) roleComboBox.getSelectedItem();

                if (username.isEmpty() || password.isEmpty()) {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Username or password cannot be emp
                } else {
                    // Verify login credentials
                    if (loginUser(username, password, role)) {
                        JOptionPane.showMessageDialog( parentComponent: null,  message: "Login successful!");

                        // Redirect based on role
                        if (role.equalsIgnoreCase( anotherString: "teacher")) {
                            new TeacherDashboard(username);
                        } else if (role.equalsIgnoreCase( anotherString: "student")) {
                            new StudentDashboard(username);
                        }
```

```
                    dispose(); // Close the login window
                } else {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Invalid credentials. Ple
                }
            }
        }
    });

    // JFrame settings
    setSize( width: 600,  height: 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

// Method to verify the user's credentials from the database
private boolean loginUser(String username, String password, String role) {  1 usage
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;


        DBConnection.closeConnection(connection);
        try {
            if (resultSet != null) resultSet.close();
            if (preparedStatement != null) preparedStatement.close();
        } catch (SQLException e) {
            e.printStackTrace();  Call to 'printStackTrace()' should probably be replaced with more robust logging
        }
    }

    return false; // Return false if no user matches the provided credentials
```

```
            preparedStatement.setString( parameterIndex: 3, role.toLowerCase());

            // Execute the query
            resultSet = preparedStatement.executeQuery();

            // Return true if a matching user is found
            return resultSet.next();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // Close the resources
            DBConnection.closeConnection(connection);
            try {
                if (resultSet != null) resultSet.close();
                if (preparedStatement != null) preparedStatement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        return false; // Return false if no user matches the provided credentials
    }

    public static void main(String[] args) {
        new Login();
    }
    class BackgroundPanel extends JPanel {  2 usages
        private Image backgroundImage;  3 usages

        public BackgroundPanel(Image backgroundImage) {  1 usage
```

```
    }
    class BackgroundPanel extends JPanel {  2 usages
        private Image backgroundImage;  3 usages

        public BackgroundPanel(Image backgroundImage) {  1 usage
            this.backgroundImage = backgroundImage;
        }

        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (backgroundImage != null) {
                g.drawImage(backgroundImage,  x: 0,  y: 0, getWidth(), getHeight(),  observer: this);
            }
        }
    }
}
```

3)Chat box

```
package dashboard;

import database.DBConnection;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Chatbox extends JFrame {
    private JTextArea chatArea;  4 usages
    private JTextField inputField;  4 usages
    private JButton sendButton;  3 usages

    public Chatbox() {  1 usage
        setTitle("Chatbox");

        chatArea = new JTextArea( rows: 20,  columns: 30);
        inputField = new JTextField( columns: 20);
        sendButton = new JButton( text: "Send");

        setLayout(new java.awt.FlowLayout());
        add(new JScrollPane(chatArea));
        add(inputField);
        add(sendButton);

        // Load chat history on startup
        loadChatHistory();

        // Send button action
        sendButton.addActionListener(new ActionListener() {
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            String message = inputField.getText();
            chatArea.append("You: " + message + "\n");
            saveMessageToDatabase( username: "You", message);  // Save message to DB
            inputField.setText("");
        }
    });

    setSize( width: 400,  height: 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

// Method to load chat history from the database
private void loadChatHistory() {  1 usage
    Connection connection = DBConnection.getConnection();
    try {
        String query = "SELECT * FROM chat_history";
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        ResultSet resultSet = preparedStatement.executeQuery();
        while (resultSet.next()) {
            chatArea.append(resultSet.getString( columnLabel: "username") + ": " + resultSet.getString( columnLabel: "
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        DBConnection.closeConnection(connection);
    }
}
```

```
    }

    // Method to save a message to the database
    private void saveMessageToDatabase(String username, String message) {  1 usage
        Connection connection = DBConnection.getConnection();
        try {
            String query = "INSERT INTO chat_history (username, message) VALUES (?, ?)";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString( parameterIndex: 1, username);
            preparedStatement.setString( parameterIndex: 2, message);
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DBConnection.closeConnection(connection);
        }
    }

    public static void main(String[] args) {
        new Chatbox();
    }
}
```

4)Student Dashboard

```java
package dashboard;

import database.DBConnection;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class StudentDashboard extends JFrame {
    private String username;  9 usages
    private JPanel cardPanel; // CardLayout to switch between different panels  22 usages
    private CardLayout cardLayout;  12 usages

    public StudentDashboard(String username) {  2 usages
        this.username = username;
        setTitle("Student Dashboard - Welcome " + username);

        cardLayout = new CardLayout();
        cardPanel = new JPanel(cardLayout); // Set CardLayout on the panel

        // Main panel containing buttons for each option
        JPanel mainPanel = createMainPage();
        // Add the mainPanel to cardPanel
        cardPanel.add(mainPanel,  constraints: "Main");

        // Set JFrame settings
        setLayout(new BorderLayout());
        add(cardPanel, BorderLayout.CENTER); // Set cardPanel as the center

        setSize( width: 600,  height: 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
    private JPanel createMainPage() {  1 usage
        JPanel mainPanel = new JPanel(new GridLayout( rows: 6,  cols: 1,  hgap: 5,  vgap: 5));
        mainPanel.setBorder(BorderFactory.createEmptyBorder( top: 10,  left: 10,  bottom: 10,  right: 10)); // Add padding




        JButton viewFilesButton = new JButton( text: "View Uploaded Files");
        JButton chatButton = new JButton( text: "Chat");
        JButton submitAssignmentButton = new JButton( text: "Submit Assignment");
        JButton quizButton = new JButton( text: "Quiz");
        JButton peerReviewButton = new JButton( text: "Peer Review");



        // Add action listeners to buttons
        viewFilesButton.addActionListener(e -> showViewFilesPanel());
        chatButton.addActionListener(e -> showChatPanel());
        submitAssignmentButton.addActionListener(e -> showSubmitAssignmentPanel());
        quizButton.addActionListener(e -> showQuizPanel());
        peerReviewButton.addActionListener(e -> showPeerReviewPanel());
```

```
        // Add buttons to the main panel
        mainPanel.add(viewFilesButton);
        mainPanel.add(chatButton);
        mainPanel.add(submitAssignmentButton);
        mainPanel.add(quizButton);
        mainPanel.add(peerReviewButton);


        return mainPanel;
    }
    void showPeerReviewPanel() {  2 usages
        // Create peerReviewPanel
        JPanel peerReviewPanel = new JPanel(new BorderLayout());
        JPanel reviewListPanel = new JPanel();
        reviewListPanel.setLayout(new BoxLayout(reviewListPanel, BoxLayout.Y_AXIS));

        JLabel titleLabel = new JLabel( text: "Assignments Available for Review:");
        titleLabel.setFont(new Font( name: "Arial", Font.BOLD,  size: 16));
        reviewListPanel.add(titleLabel);

        try (Connection connection = DBConnection.getConnection();
             PreparedStatement statement = connection.prepareStatement(
                     sql: "SELECT a.id, a.file_path, a.student_username " +
                             "FROM assignments_submitted1 a " +
                             "LEFT JOIN peer_reviews p ON a.id = p.assignment_id AND p.reviewer_username = ? " +
                             "WHERE p.assignment_id IS NULL AND a.reviewable = TRUE")) {

            statement.setString( parameterIndex: 1, username); // Set the current user's username
            ResultSet resultSet = statement.executeQuery();

            if (!resultSet.isBeforeFirst()) { // Check if there are any results
                JLabel noAssignmentsLabel = new JLabel( text: "No assignments available for review at this time.");
                reviewListPanel.add(noAssignmentsLabel);
            } else {
                while (resultSet.next()) {
                    int assignmentId = resultSet.getInt( columnLabel: "id");
                    String studentUsername = resultSet.getString( columnLabel: "student_username");
                    String filePath = resultSet.getString( columnLabel: "file_path");

                    JPanel assignmentPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
                    JLabel assignmentLabel = new JLabel( text: "Assignment from: " + studentUsername);

                    // Open File Button
                    JButton openFileButton = new JButton( text: "Open File");
                    openFileButton.addActionListener(e -> openAssignmentFile(filePath));

                    JTextField feedbackField = new JTextField( columns: 30);
                    JButton submitFeedbackButton = new JButton( text: "Submit Feedback");

                    submitFeedbackButton.addActionListener(e -> {
                        String feedback = feedbackField.getText();
                        System.out.println("Submitting feedback: " + feedback);

                        if (submitFeedback(assignmentId, feedback)) {
                            JOptionPane.showMessageDialog( parentComponent: null,  message: "Feedback submitted successfully!");
                            System.out.println("Feedback submission successful");
                            showPeerReviewPanel(); // Refresh the panel
                        } else {
                            JOptionPane.showMessageDialog( parentComponent: null,  message: "Failed to submit feedback.");
                            System.out.println("Feedback submission failed");
                        }
                    });

                    assignmentPanel.add(assignmentLabel);
```

```
                    assignmentPanel.add(assignmentLabel);
                    assignmentPanel.add(openFileButton);
                    assignmentPanel.add(feedbackField);
                    assignmentPanel.add(submitFeedbackButton);
                    reviewListPanel.add(assignmentPanel);
                }
            }
        } catch (SQLException e) {
            JLabel errorLabel = new JLabel( text: "Error loading assignments: " + e.getMessage());
            reviewListPanel.add(errorLabel);
            e.printStackTrace();
        }

        // Add the review list to the main peer review panel
        peerReviewPanel.add(new JScrollPane(reviewListPanel), BorderLayout.CENTER);

        // "Back" button to return to the main dashboard
        JButton backButton = new JButton( text: "Back to Dashboard");
        backButton.addActionListener(e -> cardLayout.show(cardPanel,  name: "Main")); // Switch back to the main panel

        // Add the "Back" button at the bottom
        peerReviewPanel.add(backButton, BorderLayout.SOUTH);

        // Ensure panel is added to the card layout and displayed
        cardPanel.add(peerReviewPanel,  constraints: "PeerReview"); // Add peerReviewPanel to cardPanel
        cardLayout.show(cardPanel,  name: "PeerReview"); // Show peerReviewPanel

        revalidate();
        repaint();
    }

    private void openAssignmentFile(String filePath) {  1 usage
        try {
            File file = new File(filePath);
            if (file.exists()) {
                Desktop.getDesktop().open(file); // Open file in default application
            } else {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "File not found: " + filePath);
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Error opening file: " + e.getMessage());
            e.printStackTrace();
        }
    }

    // Method to submit feedback to the database
    // Method to submit feedback to the database
    private boolean submitFeedback(int assignmentId, String feedback) {  1 usage
        String reviewedStudentUsername = getReviewedStudentUsername(assignmentId);

        // Log assignmentId, username, and reviewedStudentUsername to verify they are correctly retrieved
        System.out.println("Attempting to submit feedback...");
        System.out.println("Assignment ID: " + assignmentId);
        System.out.println("Feedback: " + feedback);
        System.out.println("Reviewer (username): " + username);
        System.out.println("Reviewed Student Username: " + reviewedStudentUsername);

        if (reviewedStudentUsername == null || reviewedStudentUsername.isEmpty()) {
            System.out.println("Error: Reviewed student username is null or empty for assignment ID " + assignmentId);
            return false;
        }

        // Ensure username is not null
        if (username == null || username.isEmpty()) {
```

```
            System.out.println("Error: Reviewer username is not set. Ensure 'username' is properly initialized.");
            return false;
        }

        try (Connection connection = DBConnection.getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
                     sql: "INSERT INTO peer_reviews (assignment_id, reviewer_username, reviewed_student_username, feedback)

            // Set parameters for the prepared statement
            preparedStatement.setInt( parameterIndex: 1, assignmentId);
            preparedStatement.setString( parameterIndex: 2, username);
            preparedStatement.setString( parameterIndex: 3, reviewedStudentUsername);
            preparedStatement.setString( parameterIndex: 4, feedback);

            int rowsAffected = preparedStatement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Feedback successfully saved to the database.");
                return true;
            } else {
                System.out.println("Failed to save feedback: No rows affected.");
                return false;
            }
        } catch (SQLException e) {
            System.out.println("Error saving feedback to the database:");
            e.printStackTrace();
            return false;
        }
    }

    // Method to get the reviewed student's username for an assignment
    private String getReviewedStudentUsername(int assignmentId) {  1 usage
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement statement = connection.prepareStatement(
                     sql: "SELECT student_username FROM assignments_submitted1 WHERE id = ?")) {

            statement.setInt( parameterIndex: 1, assignmentId);
            ResultSet resultSet = statement.executeQuery();

            if (resultSet.next()) {
                return resultSet.getString( columnLabel: "student_username");
            } else {
                System.out.println("No student found for assignment ID " + assignmentId);
                return null;
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving reviewed student username:");
            e.printStackTrace();
            return null;
        }
    }



    // Panel to view uploaded files by teachers
    private void showViewFilesPanel()  {  1 usage
        JPanel filesPanel = new JPanel(new BorderLayout());
        JPanel fileListPanel = new JPanel();
        fileListPanel.setLayout(new BoxLayout(fileListPanel, BoxLayout.Y_AXIS));

        JLabel titleLabel = new JLabel( text: "Files Uploaded by Teachers:");
        titleLabel.setFont(new Font( name: "Arial", Font.BOLD,  size: 16));
```

```
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement statement = connection.prepareStatement( sql: "SELECT file_path, teacher_username FROM upload
             ResultSet resultSet = statement.executeQuery()) {

            if (!resultSet.isBeforeFirst()) {
                JLabel noFilesLabel = new JLabel( text: "No files have been uploaded by teachers yet.");
                fileListPanel.add(noFilesLabel);
            } else {
                while (resultSet.next()) {
                    String filePath = resultSet.getString( columnLabel: "file_path");
                    String uploadedBy = resultSet.getString( columnLabel: "teacher_username");

                    JPanel fileEntryPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
                    JLabel fileLabel = new JLabel( text: "File: " + filePath + " (Uploaded by: " + uploadedBy + ")");
                    JButton openButton = new JButton( text: "Open");

                    openButton.addActionListener(e -> openFile(filePath));

                    fileEntryPanel.add(fileLabel);
                    fileEntryPanel.add(openButton);
                    fileListPanel.add(fileEntryPanel);
                }
            }
        } catch (SQLException e) {
            JLabel errorLabel = new JLabel( text: "Error loading files: " + e.getMessage());
            fileListPanel.add(errorLabel);
            e.printStackTrace();
        }

        filesPanel.add(new JScrollPane(fileListPanel), BorderLayout.CENTER);
```

```
        JButton backButton = new JButton( text: "Back to Dashboard");
        backButton.addActionListener(e -> cardLayout.show(cardPanel,  name: "Main"));

        filesPanel.add(backButton, BorderLayout.SOUTH);

        cardPanel.add(filesPanel,  constraints: "ViewFiles"); // Add filesPanel to cardPanel with identifier "ViewFiles"
        cardLayout.show(cardPanel,  name: "ViewFiles"); // Display the filesPanel

        cardPanel.revalidate();
        cardPanel.repaint();
    }


    // Method to open the selected file
    private void openFile(String filePath) {  1 usage
        try {
            File file = new File(filePath);
            if (file.exists()) {
                Desktop.getDesktop().open(file);
            } else {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "File not found: " + filePath);
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Error opening file: " + e.getMessage());
            e.printStackTrace();
        }
    }

    // Panel to submit an assignment
    private void showSubmitAssignmentPanel() {  1 usage
        JPanel submitPanel = new JPanel(new BorderLayout());
        JLabel instructionLabel = new JLabel( text: "Select an assignment file to upload:");
```

```
        selectedFileArea.setEditable(false);
        JButton submitButton = new JButton( text: "Submit Assignment");
        JButton backButton = new JButton( text: "Back to Dashboard");

        submitPanel.add(instructionLabel, BorderLayout.NORTH);
        submitPanel.add(selectedFileArea, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel(new FlowLayout());
        buttonPanel.add(submitButton);
        buttonPanel.add(backButton);
        submitPanel.add(buttonPanel, BorderLayout.SOUTH);

        submitButton.addActionListener(e -> {
            JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showOpenDialog( parent: null);

            if (result == JFileChooser.APPROVE_OPTION) {
                File selectedFile = fileChooser.getSelectedFile();
                String filePath = selectedFile.getAbsolutePath();
                selectedFileArea.setText("Selected file: " + filePath);

                if (saveAssignmentToDatabase(filePath)) {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Assignment submitted successfully!");
                } else {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Failed to submit assignment.");
                }
            }
        });

        // Back button to return to dashboard
        backButton.addActionListener(e -> cardLayout.show(cardPanel,  name: "Main"));

        cardPanel.add(submitPanel,  constraints: "SubmitAssignment");
        cardLayout.show(cardPanel,  name: "SubmitAssignment");
    }



    // Method to save submitted assignment to the database
    private boolean saveAssignmentToDatabase(String filePath) {  1 usage
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
                     sql: "INSERT INTO assignments_submitted1 (student_username, file_path) VALUES (?, ?)")) {

            preparedStatement.setString( parameterIndex: 1, username);
            preparedStatement.setString( parameterIndex: 2, filePath);
            int rowsAffected = preparedStatement.executeUpdate();
            return rowsAffected > 0;
        } catch (SQLException e) {
            System.out.println("Error saving assignment:");
            e.printStackTrace();
            return false;
        }
    }

    // Chat panel to display chat messages and send new ones
    private void showChatPanel() {  1 usage
        JPanel chatPanel = new JPanel(new BorderLayout());
        chatPanel.add(new JLabel( text: "Chat"), BorderLayout.NORTH);

        JTextArea chatArea = new JTextArea( rows: 15,  columns: 50);
        chatArea.setEditable(false);
        JTextField chatInputField = new JTextField( columns: 40);
        JButton sendButton = new JButton( text: "Send");
```

```
        JPanel inputPanel = new JPanel();
        inputPanel.add(chatInputField);
        inputPanel.add(sendButton);

        chatPanel.add(new JScrollPane(chatArea), BorderLayout.CENTER);
        chatPanel.add(inputPanel, BorderLayout.SOUTH);

        // Back button at the top to return to the main dashboard
        JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
        topPanel.add(backButton);
        chatPanel.add(topPanel, BorderLayout.NORTH);

        // Load chat history from the database
        loadChatHistory(chatArea);

        // Send button action to save message to the database
        sendButton.addActionListener(e -> {
            String message = chatInputField.getText();
            if (!message.isEmpty()) {
                if (saveChatMessageToDatabase(username, message)) {
                    chatArea.append("You: " + message + "\n");
                    chatInputField.setText("");
                } else {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "Failed to send message.");
                }
            }
        });

        // Back button action to return to the main dashboard
        backButton.addActionListener(e -> cardLayout.show(cardPanel,  name: "Main"));
        if (cardPanel.getComponentCount() == 1) { // Add only if it hasn't been added before
            cardPanel.add(chatPanel,  constraints: "Chat");
        }
        cardLayout.show(cardPanel,  name: "Chat");
    }


    // Method to load chat history from the database
    private void loadChatHistory(JTextArea chatArea) {  1 usage
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
                     sql: "SELECT sender_username, message FROM chats ORDER BY timestamp");
             ResultSet resultSet = preparedStatement.executeQuery()) {

            while (resultSet.next()) {
                String sender = resultSet.getString( columnLabel: "sender_username");
                String message = resultSet.getString( columnLabel: "message");
                chatArea.append(sender + ": " + message + "\n");
            }
        } catch (SQLException e) {
            System.out.println("Error loading chat history:");
            e.printStackTrace();
        }
    }

    // Method to save chat messages to the database
    private boolean saveChatMessageToDatabase(String senderUsername, String message) {  1 usage
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
                     sql: "INSERT INTO chats (sender_username, message) VALUES (?, ?)")) {

            preparedStatement.setString( parameterIndex: 1, senderUsername);
```

```
            int rowsAffected = preparedStatement.executeUpdate();
            return rowsAffected > 0;
        } catch (SQLException e) {
            System.out.println("Error saving chat message:");
            e.printStackTrace();
            return false;
        }
    }

    private void showQuizPanel() {  1 usage
        getContentPane().removeAll();
        JPanel quizSelectionPanel = new JPanel(new BorderLayout());
        JPanel listPanel = new JPanel();
        listPanel.setLayout(new BoxLayout(listPanel, BoxLayout.Y_AXIS));

        JLabel titleLabel = new JLabel( text: "Select a Quiz:");
        titleLabel.setFont(new Font( name: "Arial", Font.BOLD,  size: 16));
        listPanel.add(titleLabel);

        // Fetch all quizzes from the database
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement statement = connection.prepareStatement( sql: "SELECT quiz_id, title FROM quizzes ORDER BY crea
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()) {
                int quizId = resultSet.getInt( columnLabel: "quiz_id");
                String quizTitle = resultSet.getString( columnLabel: "title");

                JButton quizButton = new JButton(quizTitle);
                quizButton.addActionListener(e -> showQuizPanel(quizId, quizTitle));
                listPanel.add(quizButton);

        } catch (SQLException e) {
            listPanel.add(new JLabel( text: "Error loading quizzes: " + e.getMessage()));
            e.printStackTrace();
        }

        quizSelectionPanel.add(new JScrollPane(listPanel), BorderLayout.CENTER);
        add(quizSelectionPanel);
        revalidate();
        repaint();
    }

    // Display questions for the selected quiz
    private void showQuizPanel(int quizId, String quizTitle) {  1 usage
        getContentPane().removeAll();
        JPanel quizPanel = new JPanel(new BorderLayout());
        JPanel questionPanel = new JPanel();
        questionPanel.setLayout(new BoxLayout(questionPanel, BoxLayout.Y_AXIS));

        JLabel titleLabel = new JLabel( text: "Quiz: " + quizTitle);
        titleLabel.setFont(new Font( name: "Arial", Font.BOLD,  size: 16));
        questionPanel.add(titleLabel);

        JButton submitQuizButton = new JButton( text: "Submit Quiz");
        JButton backButton = new JButton( text: "Back to Dashboard"); // Back button

        // Maps to store questions and options
        Map<Integer, String> questionMap = new HashMap<>();
        Map<Integer, String> correctAnswers = new HashMap<>();
        Map<Integer, JRadioButton> selectedOptions = new HashMap<>();

        // Fetch questions for the selected quiz ID
```

```
            PreparedStatement statement = connection.prepareStatement( sql: "SELECT * FROM questions WHERE quiz_id = ?");
            statement.setInt( parameterIndex: 1, quizId);
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()) {
                int questionId = resultSet.getInt( columnLabel: "question_id");
                String questionText = resultSet.getString( columnLabel: "question_text");
                String optionA = resultSet.getString( columnLabel: "option_a");
                String optionB = resultSet.getString( columnLabel: "option_b");
                String optionC = resultSet.getString( columnLabel: "option_c");
                String optionD = resultSet.getString( columnLabel: "option_d");
                String correctAnswer = resultSet.getString( columnLabel: "correct_answer");

                questionMap.put(questionId, questionText);
                correctAnswers.put(questionId, correctAnswer);

                questionPanel.add(new JLabel( text: "Question: " + questionText));

                ButtonGroup optionGroup = new ButtonGroup();
                JRadioButton optA = new JRadioButton(optionA);
                JRadioButton optB = new JRadioButton(optionB);
                JRadioButton optC = new JRadioButton(optionC);
                JRadioButton optD = new JRadioButton(optionD);

                optionGroup.add(optA);
                optionGroup.add(optB);
                optionGroup.add(optC);
                optionGroup.add(optD);

                JPanel optionsPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
                optionsPanel.add(optA);
                optionsPanel.add(optB);
                optionsPanel.add(optC);
                optionsPanel.add(optD);

                questionPanel.add(optionsPanel);

                // Store selected option for each question
                selectedOptions.put(questionId, optA);  // Default selection as example
                optA.addActionListener(e -> selectedOptions.put(questionId, optA));
                optB.addActionListener(e -> selectedOptions.put(questionId, optB));
                optC.addActionListener(e -> selectedOptions.put(questionId, optC));
                optD.addActionListener(e -> selectedOptions.put(questionId, optD));
            }
        } catch (SQLException e) {
            questionPanel.add(new JLabel( text: "Error loading quiz: " + e.getMessage()));
            e.printStackTrace();
        }

        submitQuizButton.addActionListener(e -> {
            int score = 0;
            try (Connection connection = DBConnection.getConnection()) {
                connection.setAutoCommit(false); // Start a transaction

                for (Map.Entry<Integer, JRadioButton> entry : selectedOptions.entrySet()) {
                    int questionId = entry.getKey();
                    JRadioButton selectedOption = entry.getValue();
                    String selectedAnswer = selectedOption.getText();

                    // Check the answer and calculate the score
                    if (selectedAnswer.equals(correctAnswers.get(questionId))) {
                        score++;
                    }

                    // Prepare the SQL statement to save the response
```

```
                    // Prepare the SQL statement to save the response
                    String responseQuery = "INSERT INTO responses (student_username, quiz_id, question_id, poll_id, selected
                            "VALUES (?, ?, ?, NULL, ?, NOW())";
                    try (PreparedStatement responseStmt = connection.prepareStatement(responseQuery)) {
                        responseStmt.setString( parameterIndex: 1, username); // student's username
                        responseStmt.setInt( parameterIndex: 2, quizId); // quiz ID
                        responseStmt.setInt( parameterIndex: 3, questionId); // question ID
                        responseStmt.setString( parameterIndex: 4, selectedAnswer);
                        // selected answer

                        // Execute the insert
                        int rowsInserted = responseStmt.executeUpdate();
                        if (rowsInserted == 0) {
                            throw new SQLException("Failed to insert response for question ID: " + questionId);
                        }
                    }
                }

                // Commit the transaction
                connection.commit();

                // Display the score to the user
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Your Score: " + score + " out of " + questionMa
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Failed to save quiz responses: " + ex.getMessag

                // Rollback if any failure occurred
                try (Connection connection = DBConnection.getConnection()) {
                    connection.rollback();
                } catch (SQLException rollbackEx) {
            }
        });

        // Back button action to return to the main dashboard
        backButton.addActionListener(e -> cardLayout.show(cardPanel,  name: "Main"));

        // Add quizPanel to cardPanel with identifier "Quiz" if not already added
        if (cardPanel.getComponentCount() == 1) { // Add only if it hasn't been added before
            cardPanel.add(quizPanel,  constraints: "Quiz");
        }
        cardLayout.show(cardPanel,  name: "Quiz");

        // Add the back button to the quiz panel
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(backButton);
        quizPanel.add(buttonPanel, BorderLayout.NORTH);

        quizPanel.add(new JScrollPane(questionPanel), BorderLayout.CENTER);
        quizPanel.add(submitQuizButton, BorderLayout.SOUTH);

        revalidate();
        repaint();
    }


    // Main method to run the application
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new StudentDashboard( username: "student1"));
    }
}
```

5)Teacher Dashboard

```java
package dashboard;

import database.DBConnection;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import javax.swing.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.HashMap;

public class TeacherDashboard extends JFrame {
    private String username;  4 usages
    private JPanel cardPanel; // CardLayout to switch between different panels  6 usages
    private CardLayout cardLayout;  4 usages

    public TeacherDashboard(String username) {  2 usages
        this.username = username;
        setTitle("Teacher Dashboard - Welcome " + username);
        cardLayout = new CardLayout();
        cardPanel = new JPanel(cardLayout); // Set CardLayout on the panel
```

```java
        JPanel mainPanel = createMainPage();

        // Add the mainPanel to cardPanel
        cardPanel.add(mainPanel,  constraints: "Main");

        // Set JFrame settings
        setLayout(new BorderLayout());
        add(cardPanel, BorderLayout.CENTER); // Set cardPanel as the center

        setSize( width: 600,  height: 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    private JPanel createMainPage() {  4 usages
        JPanel mainPanel = new JPanel(new GridLayout( rows: 4,  cols: 1,  hgap: 5,  vgap: 5));
        mainPanel.setBorder(BorderFactory.createEmptyBorder( top: 10,  left: 10,  bottom: 10,  right: 10));
        // Create buttons for each option
        JButton uploadFileButton = new JButton( text: "Upload File");
        JButton viewSubmittedAssignmentsButton = new JButton( text: "View Submitted Assignments");
        JButton chatButton = new JButton( text: "Chat");
        JButton createQuizButton = new JButton( text: "Create Quiz");

        // Add action listeners for each button
        uploadFileButton.addActionListener(e -> showUploadFilePanel());
        viewSubmittedAssignmentsButton.addActionListener(e -> showSubmittedAssignmentsPanel());
        chatButton.addActionListener(e -> showChatPanel());
        createQuizButton.addActionListener(e -> showCreateQuizPanel());


        // Add buttons to the main panel
        mainPanel.add(uploadFileButton);
```

```
        mainPanel.add(viewSubmittedAssignmentsButton);
        mainPanel.add(chatButton);
        mainPanel.add(createQuizButton);

        return mainPanel;
    }
    // Panel to upload a file
    private void showUploadFilePanel() {  1 usage
        getContentPane().removeAll();

        // Create the back button
        JButton backButton = new JButton( text: "Back");
        backButton.addActionListener(e -> showMainDashboard());  // Action to go back to the main dashboard

        // Create the upload panel
        JPanel uploadPanel = new JPanel(new BorderLayout());
        JLabel instructionLabel = new JLabel( text: "Select a file to upload:");
        JTextArea selectedFileArea = new JTextArea( rows: 2,  columns: 30);
        selectedFileArea.setEditable(false);
        JButton uploadButton = new JButton( text: "Upload File");

        // Add components to the panel
        uploadPanel.add(instructionLabel, BorderLayout.NORTH);
        uploadPanel.add(selectedFileArea, BorderLayout.CENTER);
        uploadPanel.add(uploadButton, BorderLayout.SOUTH);

        // Add the back button at the top
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(backButton, BorderLayout.WEST);  // Align the back button to the left
        uploadPanel.add(topPanel, BorderLayout.NORTH);
```

```
        add(uploadPanel, BorderLayout.CENTER);

        // Action listener for upload button
        uploadButton.addActionListener(e -> {
            JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showOpenDialog( parent: null);

            if (result == JFileChooser.APPROVE_OPTION) {
                File selectedFile = fileChooser.getSelectedFile();
                String filePath = selectedFile.getAbsolutePath();
                selectedFileArea.setText("Selected file: " + filePath);

                // Save the file path to the database
                if (saveFilePathToDatabase(filePath)) {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "File uploaded successfully!");
                } else {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Failed to upload file.");
                }
            }
        });

        revalidate();
        repaint();
    }

    // Method to show the main dashboard when 'Back' is clicked
    private void showMainDashboard() {  1 usage
        // Assuming the previous screen is a main dashboard or the main menu
        // You can use this method to return to the previous dashboard
        // If you're using CardLayout or multiple screens, you can switch the current panel here.

        // Example:
```

```
        // Add the main panel (or call another method that shows the main screen)
        JPanel mainPanel = createMainPage();  // Your method to create the main dashboard
        this.add(mainPanel);

        revalidate();
        repaint();
    }


    // Method to save uploaded file path to the database in uploaded_files table
    private boolean saveFilePathToDatabase(String filePath) {  1 usage
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
                     sql: "INSERT INTO uploaded_files (teacher_username, file_path) VALUES (?, ?)")) {

            preparedStatement.setString( parameterIndex: 1, username);
            preparedStatement.setString( parameterIndex: 2, filePath);
            int rowsAffected = preparedStatement.executeUpdate();
            return rowsAffected > 0;
        } catch (SQLException e) {
            System.out.println("Error saving uploaded file path:");
            e.printStackTrace();
            return false;
        }
    }

    // Panel to view submitted assignments
    // Panel to view submitted assignments

    // Panel to view submitted assignments
    private void showSubmittedAssignmentsPanel() {  1 usage
        getContentPane().removeAll();
         // Create the back button
         JButton backButton = new JButton( text: "Back");
         backButton.addActionListener(e -> showMainPanel());  // Action to go back to the main page

         // Create the assignment panel
         JPanel assignmentPanel = new JPanel(new BorderLayout());

         // Create a panel to hold the back button
         JPanel topPanel = new JPanel(new BorderLayout());
         topPanel.add(backButton, BorderLayout.WEST);  // Align the back button to the left
         assignmentPanel.add(topPanel, BorderLayout.NORTH);

         assignmentPanel.add(new JLabel( text: "Submitted Assignments"), BorderLayout.CENTER);

         // List model to store assignments
         DefaultListModel<String> assignmentListModel = new DefaultListModel<>();
         JList<String> assignmentList = new JList<>(assignmentListModel);
         assignmentList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
         assignmentPanel.add(new JScrollPane(assignmentList), BorderLayout.CENTER);

         JButton openFileButton = new JButton( text: "Open Selected File");
         assignmentPanel.add(openFileButton, BorderLayout.SOUTH);

         // Map to store file paths with student info as the key
         Map<String, String> filePathMap = new HashMap<>();

         // Fetch submitted assignments from the database
         try (Connection connection = DBConnection.getConnection();
              PreparedStatement preparedStatement = connection.prepareStatement(
                      sql: "SELECT student_username, file_path FROM assignments_submitted1");
              ResultSet resultSet = preparedStatement.executeQuery()) {
```

```
            ResultSet resultSet = preparedStatement.executeQuery()) {

            while (resultSet.next()) {
                String studentUsername = resultSet.getString( columnLabel: "student_username");
                String filePath = resultSet.getString( columnLabel: "file_path");
                String displayText = "Student: " + studentUsername + " - File: " + filePath;

                assignmentListModel.addElement(displayText);
                filePathMap.put(displayText, filePath);  // Map display text to file path
            }
        } catch (SQLException e) {
            System.out.println("Error fetching submitted assignments:");
            e.printStackTrace();
        }

        // Open selected file when button is clicked
        openFileButton.addActionListener(e -> {
            String selectedEntry = assignmentList.getSelectedValue();
            if (selectedEntry != null) {
                String filePath = filePathMap.get(selectedEntry);
                try {
                    Desktop.getDesktop().open(new File(filePath));
                } catch (IOException ex) {
                    ex.printStackTrace();
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Error opening file.");
                }
            } else {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Please select a file to open.");
            }
        });

        add(assignmentPanel, BorderLayout.CENTER);
        repaint();
    }

    // Method to show the main panel when 'Back' is clicked
    private void showMainPanel() {  1 usage
        getContentPane().removeAll();  // Remove all components

        // Create the main page panel and add it to the content pane
        JPanel mainPanel = createMainPage();  // Assuming 'createMainPage' creates the main dashboard
        add(mainPanel, BorderLayout.CENTER);

        revalidate();  // Refresh the layout
        repaint();
    }



    // Panel to create a new quiz
    // Make quizId a class-level variable
    private int quizId = -1;  5 usages

    private void showCreateQuizPanel() {  1 usage
        getContentPane().removeAll();

         // Action to go back to the teacher dashboard
         // Create the back button
        JButton backButton = new JButton( text: "Back");
        backButton.addActionListener(e -> showMainPage());
        // Create the quiz panel
        JPanel quizPanel = new JPanel(new BorderLayout());

```

```
        // Create a panel to hold the back button at the top
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(backButton, BorderLayout.WEST);  // Align the back button to the left
        quizPanel.add(topPanel, BorderLayout.NORTH);

        quizPanel.add(new JLabel( text: "Create New Quiz"), BorderLayout.CENTER);

        // Title input
        JPanel titlePanel = new JPanel(new FlowLayout());
        JLabel titleLabel = new JLabel( text: "Quiz Title:");
        JTextField titleField = new JTextField( columns: 20);
        titlePanel.add(titleLabel);
        titlePanel.add(titleField);
        quizPanel.add(titlePanel, BorderLayout.NORTH);

        // Question input section
        JPanel questionPanel = new JPanel(new GridLayout( rows: 6,  cols: 2));
        questionPanel.add(new JLabel( text: "Question:"));
        JTextField questionField = new JTextField();
        questionPanel.add(questionField);

        questionPanel.add(new JLabel( text: "Option A:"));
        JTextField optionAField = new JTextField();
        questionPanel.add(optionAField);

        questionPanel.add(new JLabel( text: "Option B:"));
        JTextField optionBField = new JTextField();
        questionPanel.add(optionBField);

        questionPanel.add(new JLabel( text: "Option C:"));
    private void showCreateQuizPanel() {  1 usage
        questionPanel.add(optionCField);

        questionPanel.add(new JLabel( text: "Option D:"));
        JTextField optionDField = new JTextField();
        questionPanel.add(optionDField);

        questionPanel.add(new JLabel( text: "Correct Answer (A, B, C, or D):"));
        JTextField correctAnswerField = new JTextField();
        questionPanel.add(correctAnswerField);

        quizPanel.add(questionPanel, BorderLayout.CENTER);

        // Button to add question to quiz
        JButton addQuestionButton = new JButton( text: "Add Question");
        JButton saveQuizButton = new JButton( text: "Save Quiz");
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(addQuestionButton);
        buttonPanel.add(saveQuizButton);
        quizPanel.add(buttonPanel, BorderLayout.SOUTH);

        add(quizPanel, BorderLayout.CENTER);

        // List to display added questions
        DefaultListModel<String> questionListModel = new DefaultListModel<>();
        JList<String> questionList = new JList<>(questionListModel);
        quizPanel.add(new JScrollPane(questionList), BorderLayout.EAST);

        revalidate();
        repaint();

        // List to store the questions before saving
        List<Question> questions = new ArrayList<>();
```

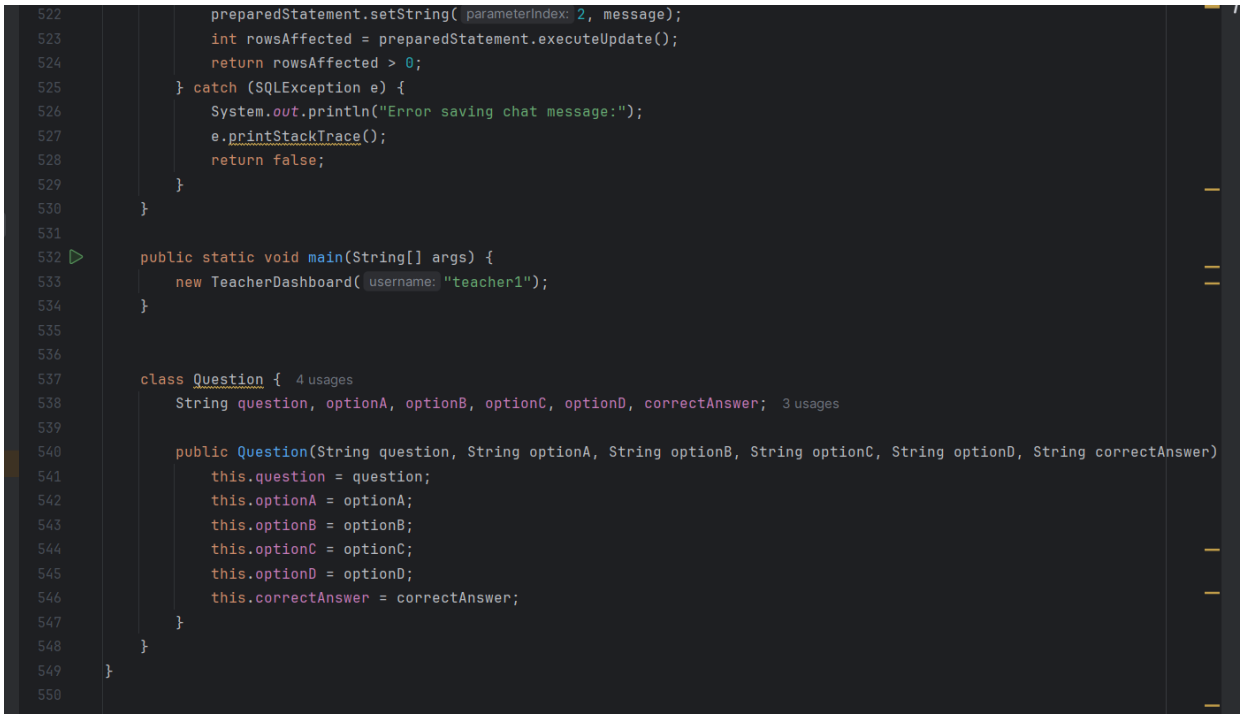
```
        // List to store the questions before saving
        List<Question> questions = new ArrayList<>();

        // Add action to add question to quiz
        addQuestionButton.addActionListener(e -> {
            String questionText = questionField.getText();
            String optionA = optionAField.getText();
            String optionB = optionBField.getText();
            String optionC = optionCField.getText();
            String optionD = optionDField.getText();
            String correctAnswer = correctAnswerField.getText();

            // If quizId is not set, save the quiz first
            if (quizId == -1) {
                String quizTitle = titleField.getText();
                quizId = saveQuizToDatabase(quizTitle); // Save quiz and get quizId
                if (quizId == -1) {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "Failed to save quiz.");
                    return; // Exit if saving quiz failed
                }
            }

            // Add question to the list
            questions.add(new Question(questionText, optionA, optionB, optionC, optionD, correctAnswer));
            questionListModel.addElement("Q: " + questionText);

            // Clear fields after adding
            questionField.setText("");
            optionAField.setText("");
            optionBField.setText("");
            optionCField.setText("");
            optionDField.setText("");
```

```
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Failed to save quiz and questions.");
            }
        });
    }

    // Method to show the teacher dashboard when 'Back' is clicked
    private void showMainPage() {  1 usage
        // Assuming the previous screen is a main dashboard or the main menu
        // You can use this method to return to the previous dashboard
        // If you're using CardLayout or multiple screens, you can switch the current panel here.

        // Example:
        this.getContentPane().removeAll();
        // Add the main panel (or call another method that shows the main screen)
        JPanel mainPage = createMainPage();  // Your method to create the main dashboard
        this.add(mainPage);

        revalidate();
        repaint();
    }


    // Save quiz to the database and return the generated quiz_id
    private int saveQuizToDatabase(String quizTitle) {  1 usage
        try (Connection connection = DBConnection.getConnection()) {
            // Save quiz title
            String quizQuery = "INSERT INTO quizzes (teacher_username, title, created_date) VALUES (?, ?, NOW())";
            PreparedStatement quizStmt = connection.prepareStatement(quizQuery, Statement.RETURN_GENERATED_KEYS);
            quizStmt.setString( parameterIndex: 1, username);  // Assuming 'username' is the teacher's username
            quizStmt.setString( parameterIndex: 2, quizTitle);
            quizStmt.executeUpdate();
```

```
            // Retrieve generated quiz ID
            ResultSet rs = quizStmt.getGeneratedKeys();
            if (rs.next()) {
                return rs.getInt( columnIndex: 1); // Return the generated quiz ID
            } else {
                System.out.println("Failed to retrieve generated quiz ID.");
                return -1; // Return -1 if failed
            }
        } catch (SQLException e) {
            System.out.println("Error saving quiz to the database:");
            e.printStackTrace();
            return -1; // Return -1 if failed
        }
    }

    // Save questions to the database with the specified quiz_id
    private boolean saveQuestionsToDatabase(int quizId, List<Question> questions) {  1 usage
        try (Connection connection = DBConnection.getConnection()) {
            // Save each question to the questions table
            for (Question q : questions) {
                String questionQuery = "INSERT INTO questions (quiz_id, question_text, option_a, option_b, option_c, option_d,
                PreparedStatement questionStmt = connection.prepareStatement(questionQuery);
                questionStmt.setInt( parameterIndex: 1, quizId); // Associate the question with the quiz_id
                questionStmt.setString( parameterIndex: 2, q.question);
                questionStmt.setString( parameterIndex: 3, q.optionA);
                questionStmt.setString( parameterIndex: 4, q.optionB);
                questionStmt.setString( parameterIndex: 5, q.optionC);
                questionStmt.setString( parameterIndex: 6, q.optionD);
                questionStmt.setString( parameterIndex: 7, q.correctAnswer);

                // Check if question insertion succeeds
                int questionRowsAffected = questionStmt.executeUpdate();
                if (questionRowsAffected <= 0) {
                    System.out.println("Failed to save question: " + q.question);
                    return false;
                }
            }
            return true; // Return true if all questions were saved successfully
        } catch (SQLException e) {
            System.out.println("Error saving questions to the database:");
            e.printStackTrace();
            return false; // Return false if there was an error
        }
    }



    private void showChatPanel() {  1 usage
        // Create the chat panel with BorderLayout
        JPanel chatPanel = new JPanel(new BorderLayout());

        // Create the Back button and set its action to return to the main panel
        JButton backButton = new JButton( text: "Back");
        backButton.addActionListener(e -> cardLayout.show(cardPanel,  name: "Main")); // Show the Main panel when clicked

        // Create a top panel to hold the Back button and the chat label
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(backButton, BorderLayout.WEST);
        topPanel.add(new JLabel( text: "Chat"), BorderLayout.CENTER);
        chatPanel.add(topPanel, BorderLayout.NORTH);  // Add topPanel to the top of the chatPanel

```

```
        chatPanel.add(inputPanel, BorderLayout.SOUTH);

        // Load chat history from the database
        loadChatHistory(chatArea);

        // Send button action to save message to the database
        sendButton.addActionListener(e -> {
            String message = chatInputField.getText();
            if (!message.isEmpty()) {
                if (saveChatMessageToDatabase(username, message)) {
                    chatArea.append("You: " + message + "\n");
                    chatInputField.setText("");
                } else {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "Failed to send message.");
                }
            }
        });

        // Add the chatPanel to the cardLayout
        cardPanel.add(chatPanel,  constraints: "Chat");
        cardLayout.show(cardPanel,  name: "Chat"); // Show the chat panel now

        // Revalidate and repaint the layout to update the UI
        revalidate();
        repaint();
    }



    // Method to load chat history from the database
    private void loadChatHistory(JTextArea chatArea) {  1 usage
        try (Connection connection = DBConnection.getConnection();
```

```
    private void loadChatHistory(JTextArea chatArea) {  1 usage
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
                     sql: "SELECT sender_username, message FROM chats ORDER BY timestamp");
             ResultSet resultSet = preparedStatement.executeQuery()) {

            while (resultSet.next()) {
                String sender = resultSet.getString( columnLabel: "sender_username");
                String message = resultSet.getString( columnLabel: "message");
                chatArea.append(sender + ": " + message + "\n");
            }
        } catch (SQLException e) {
            System.out.println("Error loading chat history:");
            e.printStackTrace();
        }
    }

    // Method to save chat messages to the database
    private boolean saveChatMessageToDatabase(String senderUsername, String message) {  1 usage
        try (Connection connection = DBConnection.getConnection();
             PreparedStatement preparedStatement = connection.prepareStatement(
                     sql: "INSERT INTO chats (sender_username, message) VALUES (?, ?)")) {

            preparedStatement.setString( parameterIndex: 1, senderUsername);
            preparedStatement.setString( parameterIndex: 2, message);
            int rowsAffected = preparedStatement.executeUpdate();
            return rowsAffected > 0;
        } catch (SQLException e) {
            System.out.println("Error saving chat message:");
            e.printStackTrace();
            return false;
        }
```



```
            preparedStatement.setString( parameterIndex: 2, message);
            int rowsAffected = preparedStatement.executeUpdate();
            return rowsAffected > 0;
        } catch (SQLException e) {
            System.out.println("Error saving chat message:");
            e.printStackTrace();
            return false;
        }
    }

    public static void main(String[] args) {
        new TeacherDashboard( username: "teacher1");
    }


    class Question {  4 usages
        String question, optionA, optionB, optionC, optionD, correctAnswer;  3 usages

        public Question(String question, String optionA, String optionB, String optionC, String optionD, String correctAnswer)
            this.question = question;
            this.optionA = optionA;
            this.optionB = optionB;
            this.optionC = optionC;
            this.optionD = optionD;
            this.correctAnswer = correctAnswer;
        }
    }
}

```

6.Result and Analysis

Registration Form:

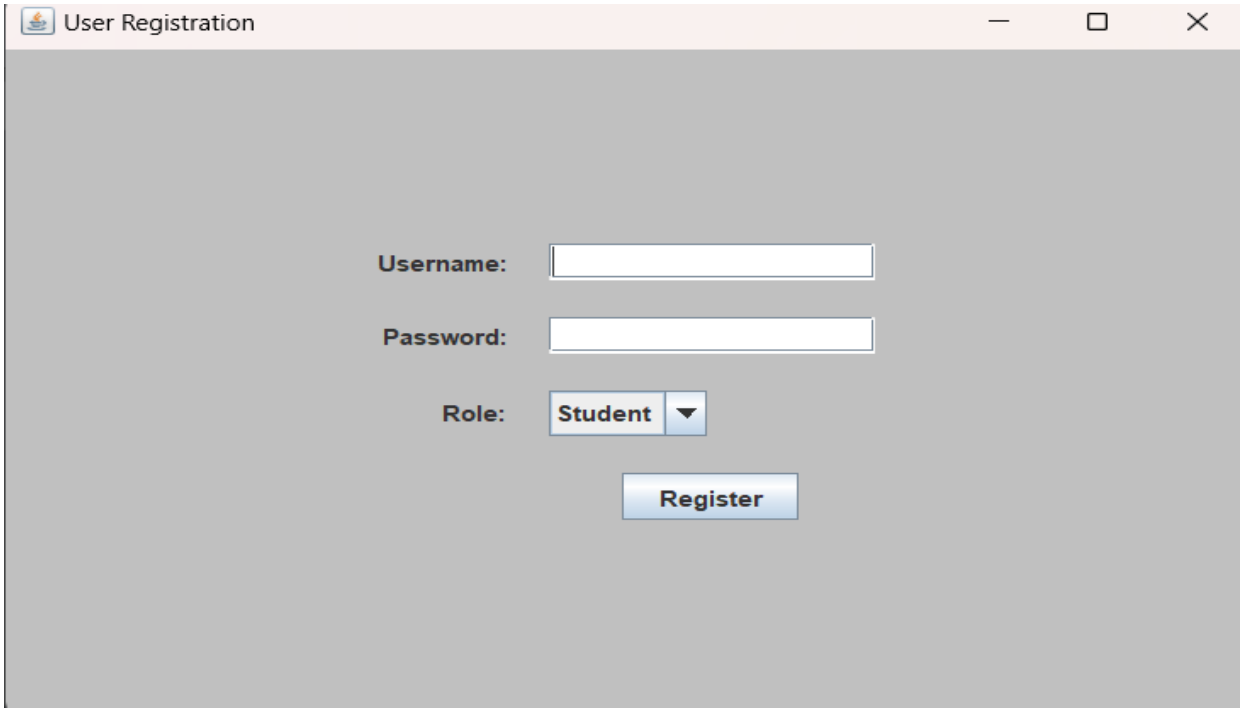


fig.(ii) Output: Registration Form

Login Form:

fig.(iii) Output: Login Form

Chat Box:

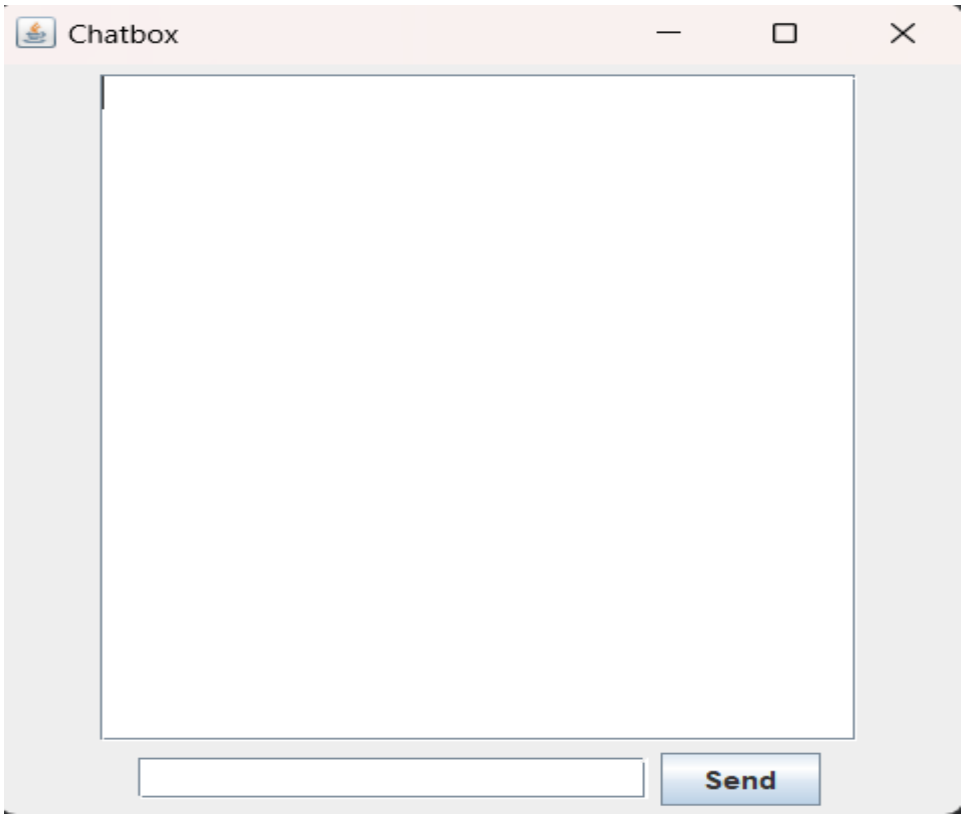


fig.(iv)Chatbox

Student Dashboard:

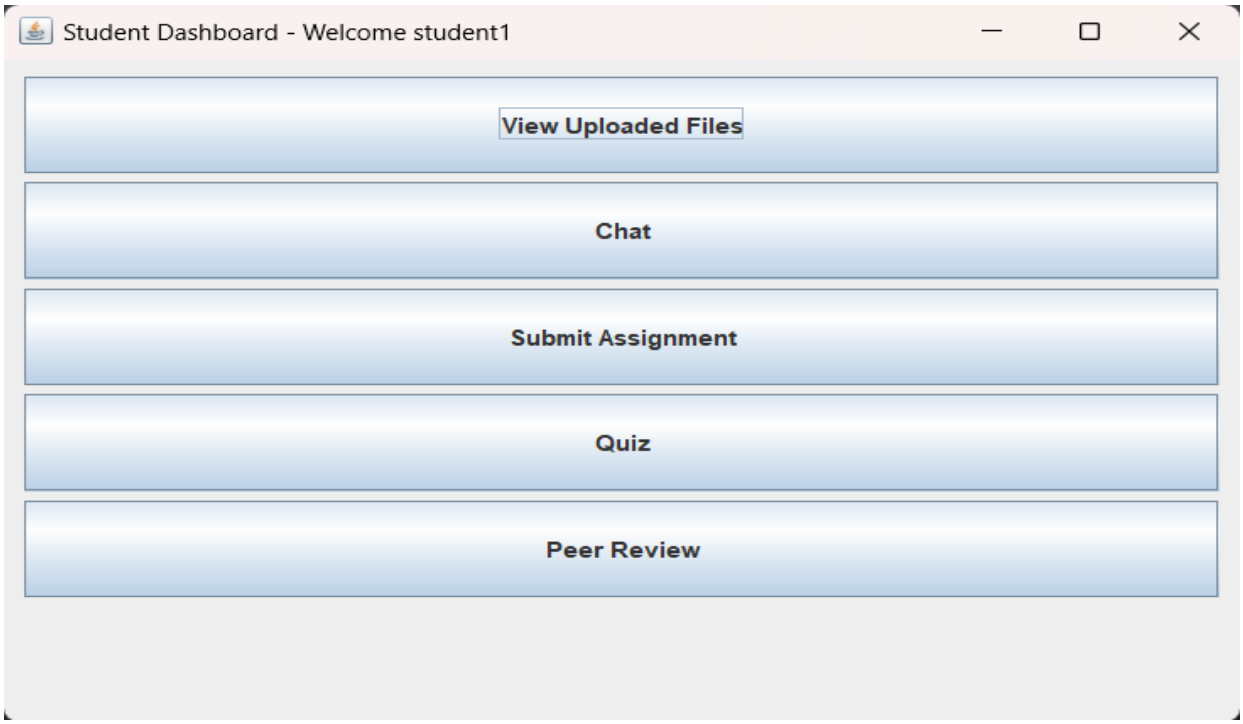


fig.(v)Student Dashboard

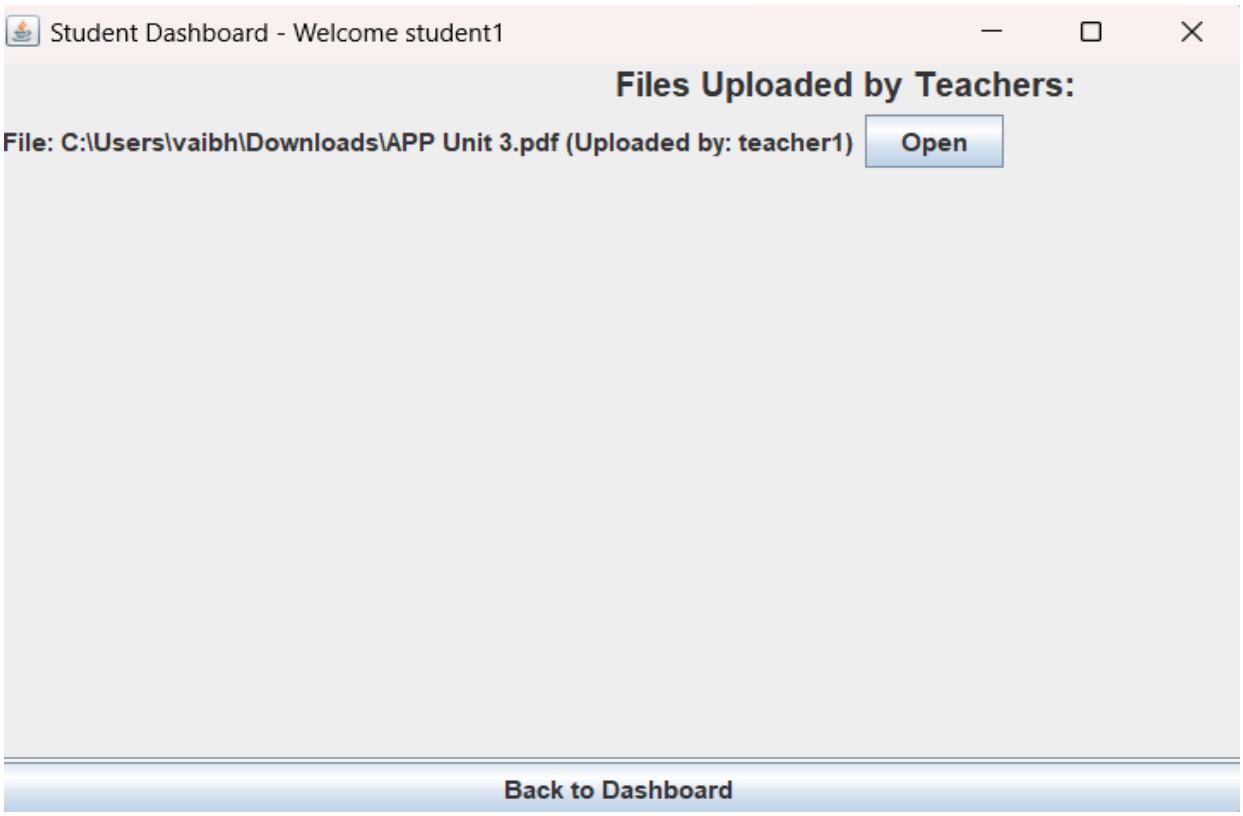


fig.(vi)View Uploaded Files

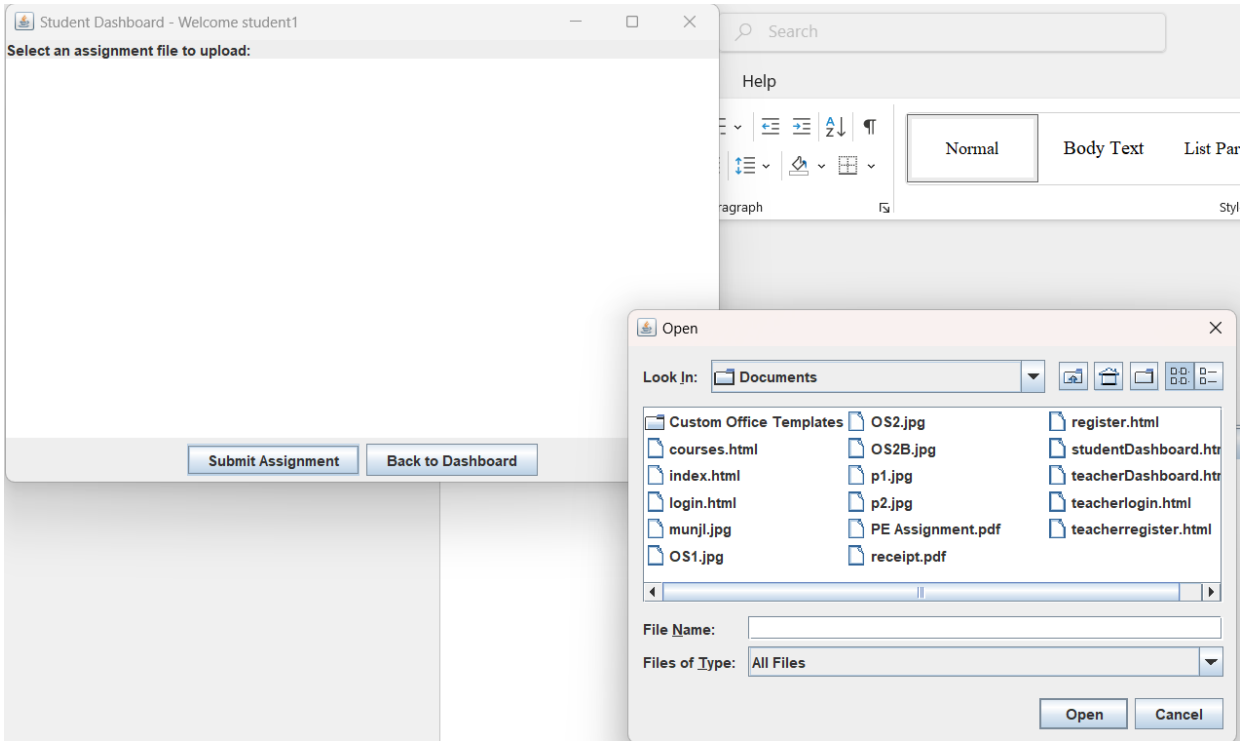


fig.(vii)Submit Assignment

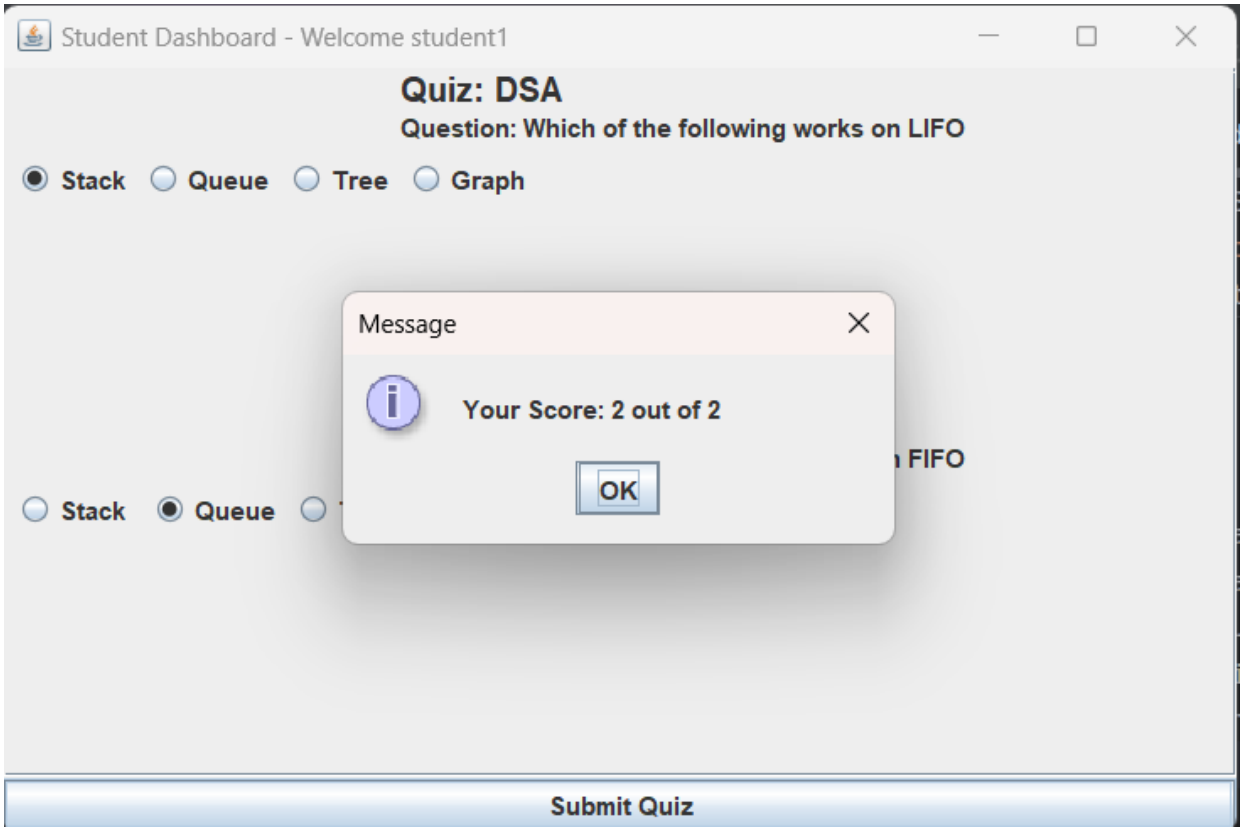


fig.(viii)Quiz

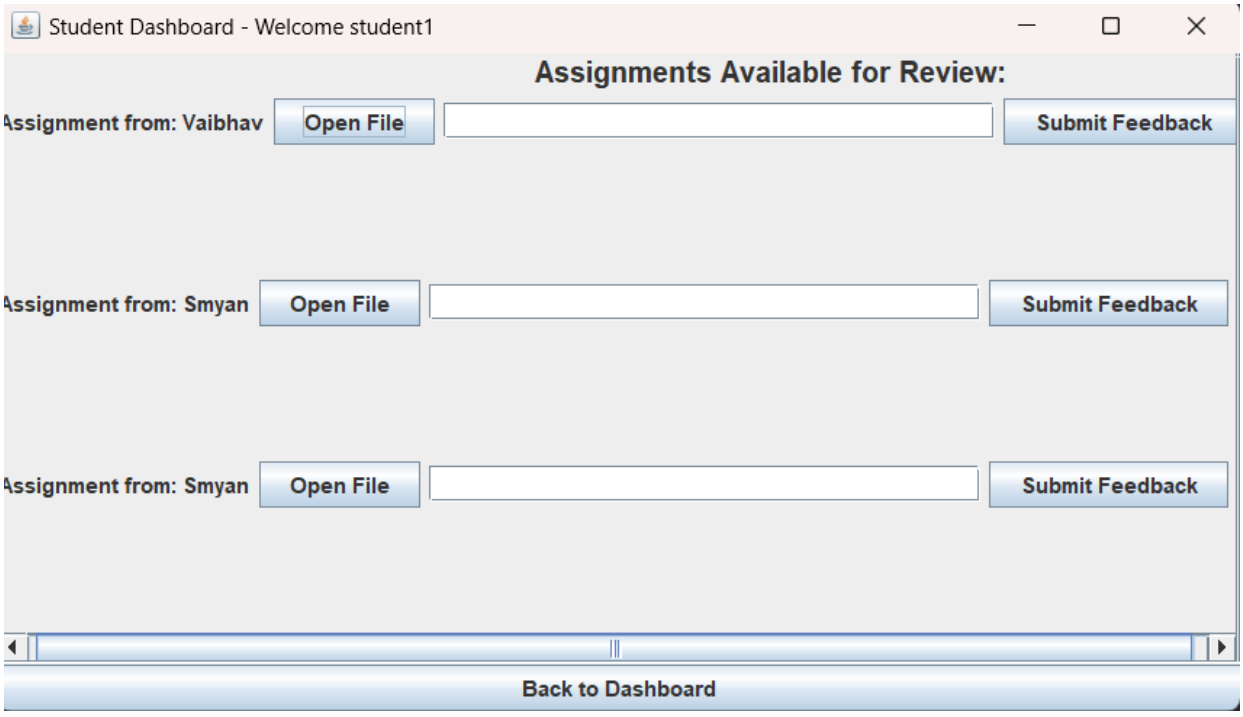


fig.(ix)Peer Review

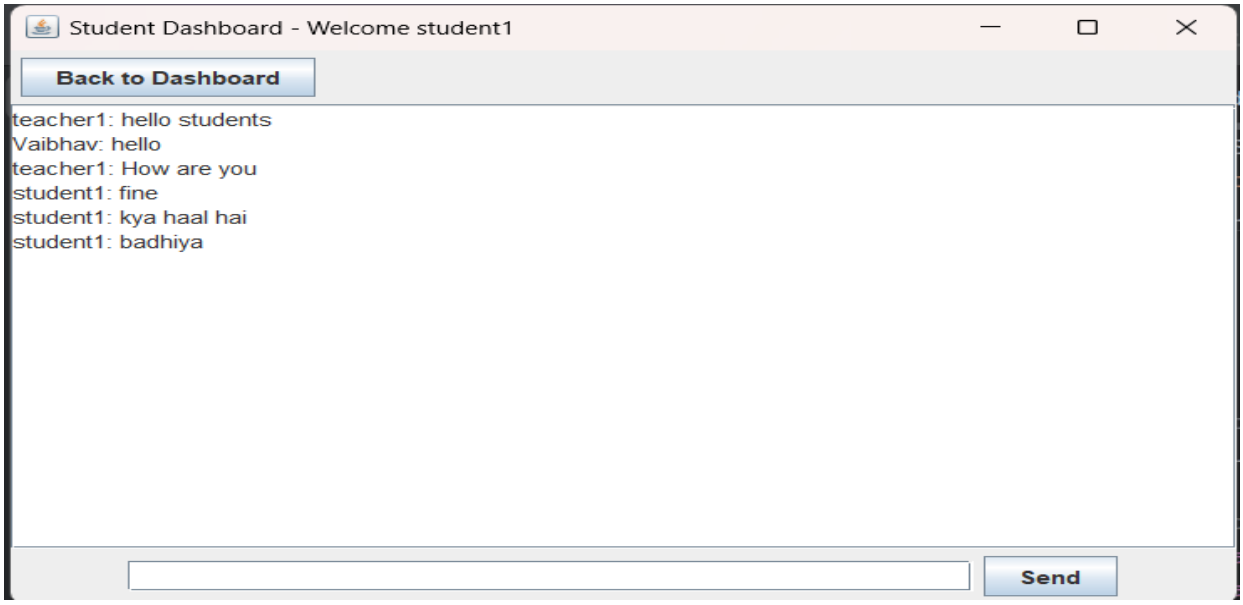


fig.(x)Chat

Teacher Dashboard:

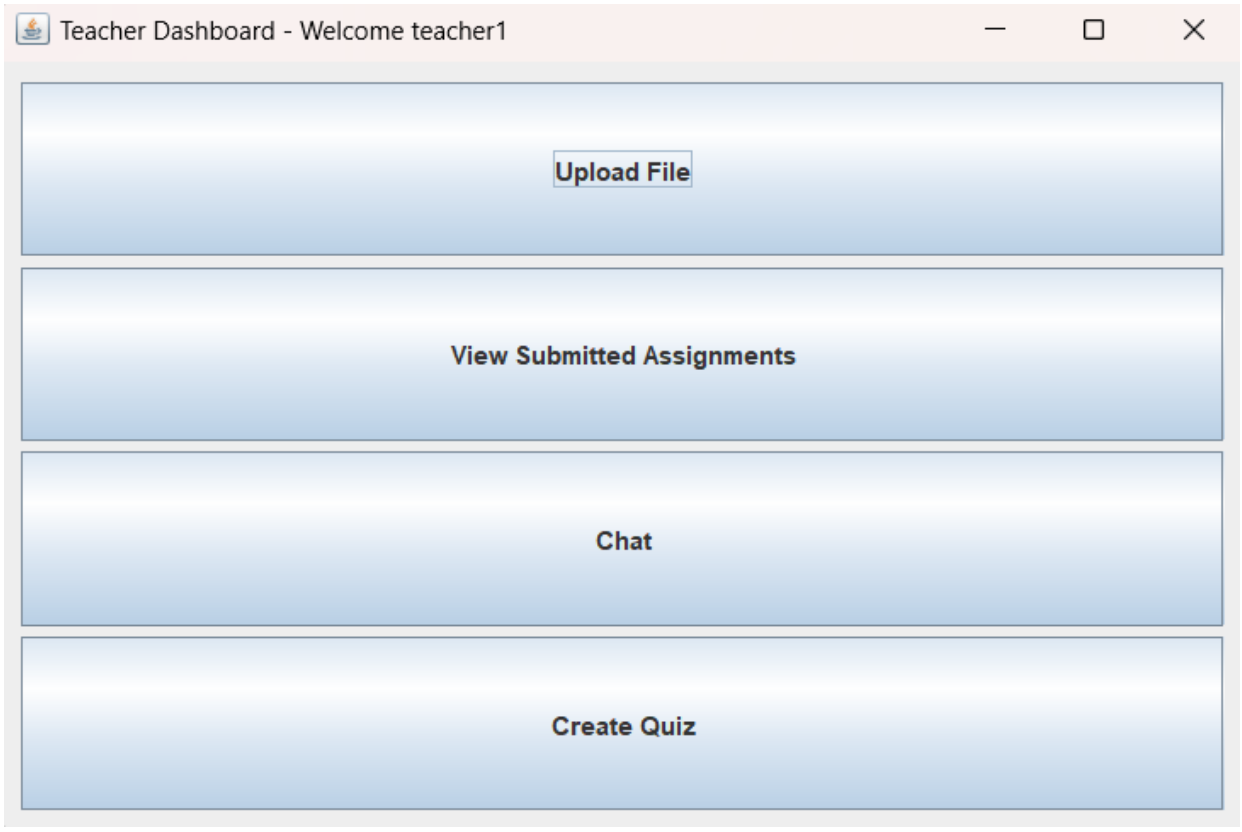


fig.(xi)Teacher's Dashboard

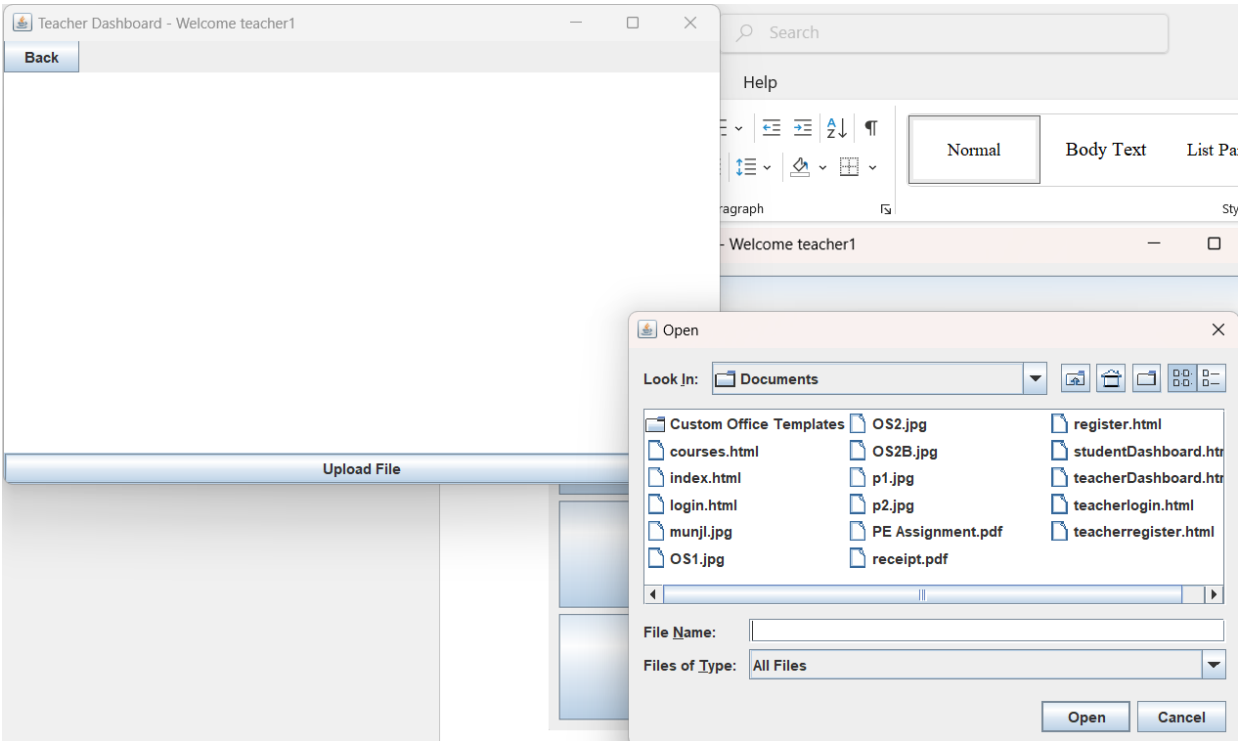


fig.(xii)Upload Files

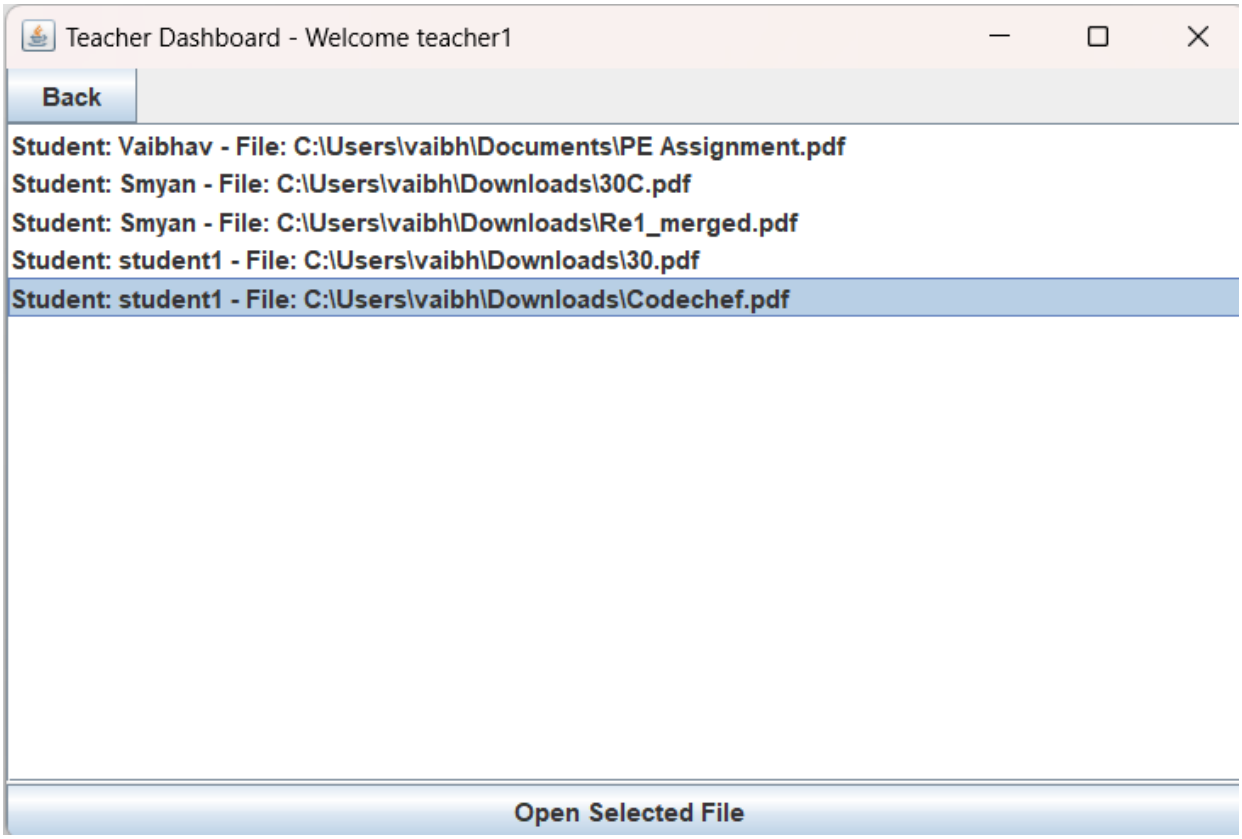


fig.(xiii)View Submitted Assignment

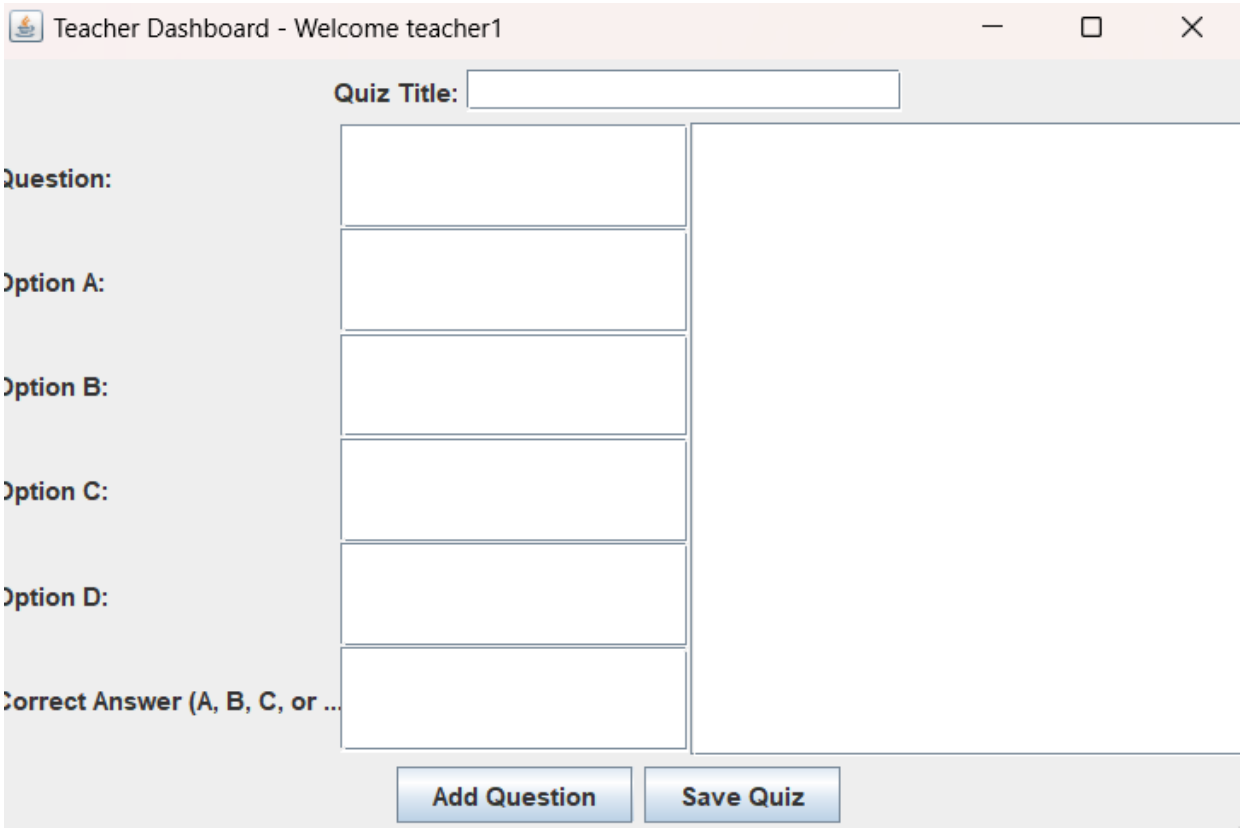


fig.(xiv)Create Quiz

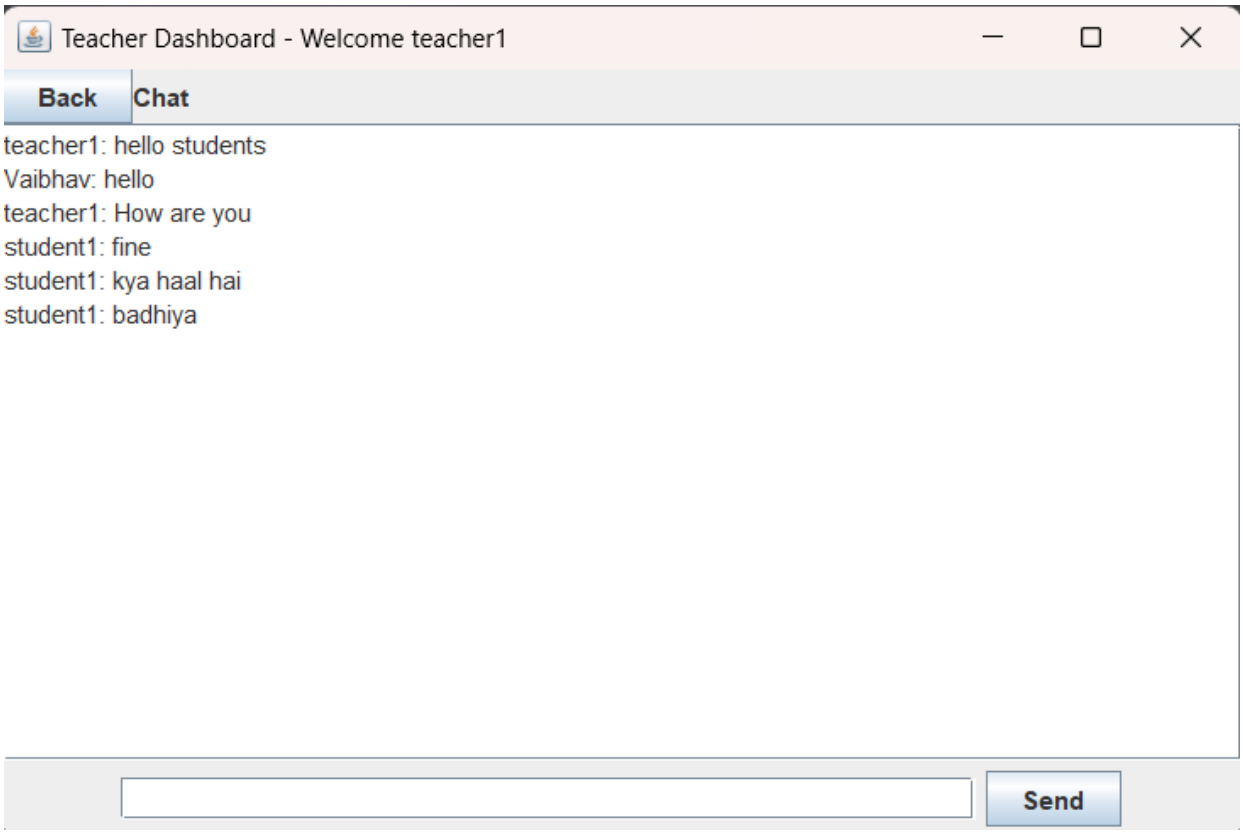


fig.(xv)Chat

Database:

Users:

id	username	password	role
1	Vaibhav	Vaibhav@123	student
2	Dr. S Ashwini	@123	teacher
3	Neha	Neha	teacher
4	Smyan	Smyan@123	student
NULL	NULL	NULL	NULL

fig.(xvi)User's Table

Chats:

id	sender_username	message	timestamp
1	teacher1	hello students	2024-10-29 00:04:34
2	Vaibhav	hello	2024-10-29 18:45:29
3	teacher1	How are you	2024-11-09 21:43:33
4	student1	fine	2024-11-10 02:21:11
5	student1	kya haal hai	2024-11-10 16:02:00
6	student1	badhiya	2024-11-10 17:05:44
NULL	NULL	NULL	NULL

fig.(xvii)Chat Table

Assignments Submitted:

id	student_username	file_path
3	Vaibhav	C:\Users\vaibh\Documents\PE Assignment.pdf
6	Smyan	C:\Users\vaibh\Downloads\30C.pdf
7	Smyan	C:\Users\vaibh\Downloads\Re1_merged.pdf
8	student1	C:\Users\vaibh\Downloads\30.pdf
9	student1	C:\Users\vaibh\Downloads\Codechef.pdf
NULL	NULL	NULL

fig.(xviii)Assignments Table

Uploaded Files By Teachers:

id	teacher_username	file_path	uploaded_at
1	teacher1	C:\Users\vaibh\Downloads\APP Unit 3.pdf	2024-10-29 00:28:07
NULL	NULL	NULL	NULL

fig.(xix)Uploaded Files Table

Quizzes:

quiz_id	teacher_username	title	created_date
15	teacher1	DSA	2024-11-10 03:08:14
16	teacher1	Intro	2024-11-10 03:12:06
17	teacher1	DSA	2024-11-10 03:20:50
NULL	NULL	NULL	NULL

fig.(xxi)Quiz Table

Questions:

question_id	quiz_id	question_text	option_a	option_b	option_c	option_d	correct_answer
15	NULL	In which data structure are elements stored in ...	Tree	Stack	Queue	Stack	Tree
16	NULL	year	2020	2024	2025	2026	B
17	17	Which of the following works on LIFO	Stack	Queue	Tree	Graph	Stack
18	17	Which of the following works on FIFO	Stack	Queue	Tree	Map	Queue
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

fig.(xxii)Question Table

Responses:

response_id	student_username	quiz_id	question_id	poll_id	selected_option	response_date
3	student1	17	18	NULL	Queue	2024-11-10 03:22:30
4	student1	17	17	NULL	Stack	2024-11-10 03:32:40
5	student1	17	18	NULL	Queue	2024-11-10 03:32:40
6	student1	17	17	NULL	Stack	2024-11-10 13:04:33
7	student1	17	18	NULL	Queue	2024-11-10 13:04:33
8	student1	17	17	NULL	Stack	2024-11-10 13:24:43
9	student1	17	18	NULL	Queue	2024-11-10 13:24:43
10	student1	17	17	NULL	Stack	2024-11-10 13:24:47
11	student1	17	18	NULL	Stack	2024-11-10 13:24:47
12	student1	17	17	NULL	Stack	2024-11-10 16:13:12
13	student1	17	18	NULL	Queue	2024-11-10 16:13:12
14	student1	17	17	NULL	Stack	2024-11-10 18:28:42
15	student1	17	18	NULL	Queue	2024-11-10 18:28:42
16	student1	17	17	NULL	Stack	2024-11-11 00:02:17

fig.(xxiii)Response Table

7.Future Scope

Integration with External Educational Tools and Platforms:

- Integrate the E-Learning System with popular educational tools and platforms such as Learning Management Systems (LMS), online libraries, and e-book providers to enhance resource availability and learning opportunities.
- Enable seamless import and export of data between the E-Learning System and external platforms for a more comprehensive learning experience.

Gamification of Learning:

- Integrate gamification elements such as badges, leaderboards, and rewards to motivate students and make the learning experience more engaging and enjoyable.
- Design interactive and game-based quizzes and assignments to foster a competitive and fun learning environment.

Mobile Application Development:

- Develop a mobile application version of the E-Learning System to provide students and teachers with on-the-go access to educational resources, quizzes, and communication tools. This will enhance accessibility and convenience, especially for users who prefer mobile devices.

Enhanced Security Measures:

- Implement advanced security protocols to protect sensitive data and ensure user privacy. This includes multi-factor authentication, encryption, and regular security audits.
- Develop automated backup and recovery systems to prevent data loss and ensure system reliability.

8.Conclusion

In conclusion, the E-Learning System project stands as a significant innovation in addressing the fragmented nature of educational resources. By providing an integrated platform for resource management, quizzes, communication, and peer review, it enhances the learning experience for students and educators alike. The robust architecture, combining Java Swing and SQL, ensures seamless performance and data integrity. This project not only meets current educational needs but also paves the way for future enhancements, making it a scalable and reliable solution in the rapidly evolving landscape of education.

9.References

[1] Adaptive Learning with AI-Based Systems: Challenges and Solutions - Examines AI's role in adaptive e-learning and addresses key challenges in implementation.

[2] Mobile Learning in E-Learning Systems: A Systematic Review - Analyzes mobile learning's integration with e-learning systems.

[3] E-Learning for Lifelong Learning: A Systematic Review - Focuses on e-learning's potential for lifelong education and skill enhancement.

[4] E-Learning Readiness in Higher Education - Looks at readiness factors that contribute to the successful adoption of e-learning in universities.

[5] Blended Learning Models for Effective Teaching - Explores the blended approach, combining online and traditional instruction, for improved educational outcomes.

[6] Personalized E-Learning Models and Student Performance - Investigates how personalized learning paths impact student progress and motivation.

[7] Challenges in Implementing E-Learning in Developing Countries - Analyzes the specific obstacles faced by institutions in low-resource settings.

[8] A Framework for E-Learning Quality Assessment - Proposes criteria for evaluating the effectiveness and quality of e-learning programs.

[9] Social Learning and Peer Review in E-Learning Systems - Studies how peer review and feedback contribute to learning in e-learning models.

[10] AI-Driven E-Learning Systems for Adaptive Learning - Reviews AI applications in e-learning to support personalized, adaptive content delivery.