

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



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 VINOOTH, 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

Sr. No.	Title	Page No.
1.	Introduction	6
2.	Literature Survey	7
3.	Requirement Analysis	9
4.	Architecture and Design	11
5.	Implementation	13
6.	Experiment Result and Analysis	40
7.	Future Scope	49
8.	Conclusion	50
9.	References	51

fig.no.	Description	Page No.
i.	Use Case Diagram of E learning System	13
ii.	Output: Registration Page	7
iii.	Output: Login Page	9
iv.	Chatbox	11
v.	Student Dashboard	13
vi.	View Uploaded Files	40
vii.	Submit Assignment	49
.		
viii.	Quiz	50
ix.	Peer Review	51

fig.no.	Description	Page No.
x.	Chat	6
xi.	Teacher's Dashboard	7
xii.	Upload Files	9
xiii.	View Submitted Assignment	11
xiv.	Create Quiz	13
xv.	Chat	40
xvi.	User's Table	49
xvii.	Chat Table	50
xviii.	Assignments Table	51

fig.no.	Description	Page No.
xix.	Upload Files Table	6
xx.	Quiz Table	7
xxi.	Question Table	9
xxii.	Response Table	11

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

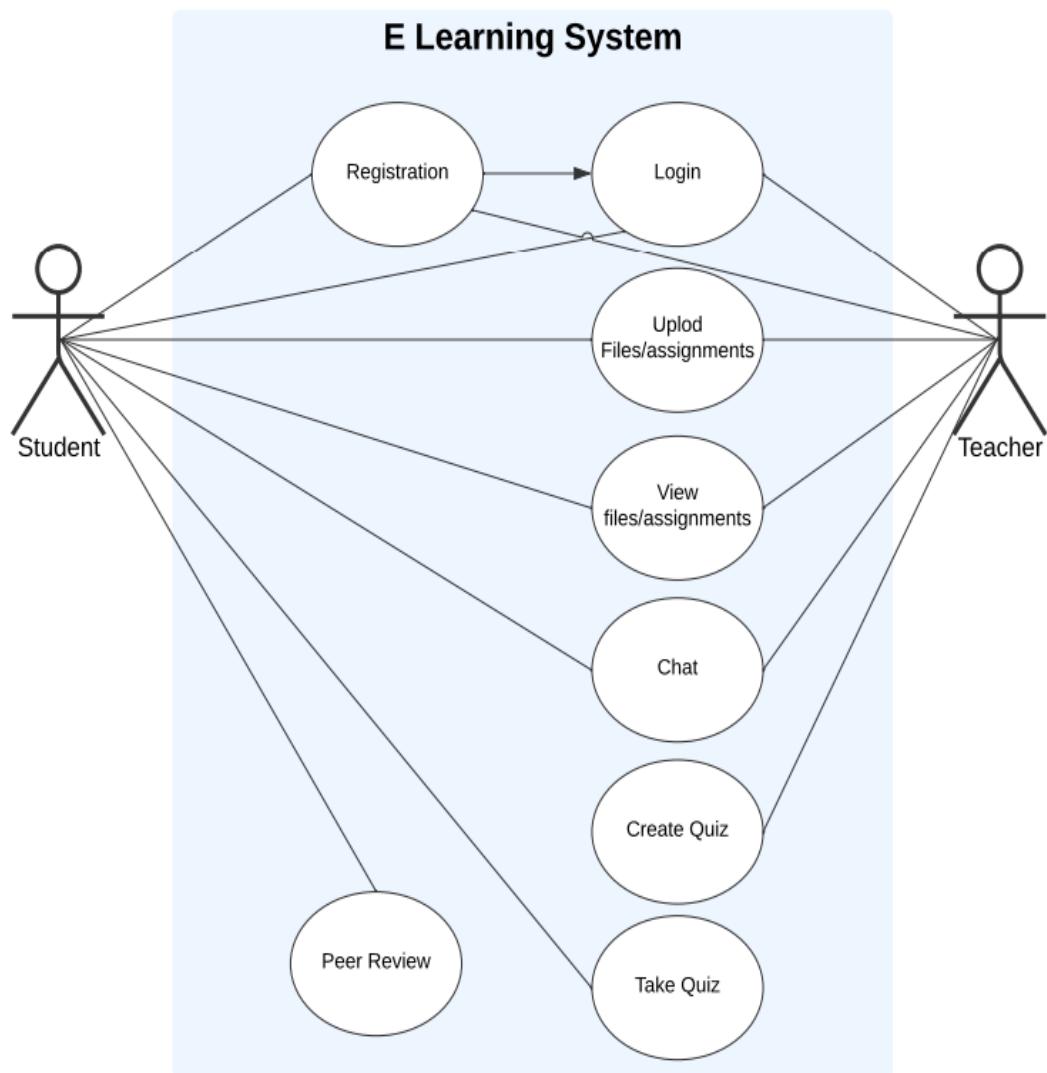


fig.(i) Use Case Diagram for E learning System

5.Implementation

1)Registration Page

```
1 package dashboard;
2
3 import database.DBConnection;
4
5 import javax.swing.*;
6 import java.awt.*;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import java.sql.Connection;
10 import java.sql.PreparedStatement;
11 import java.sql.SQLException;
12
13 ▷ public class RegistrationForm extends JFrame {
14     private JTextField usernameField; 3 usages
15     private JPasswordField passwordField; 3 usages
16     private JComboBox<String> roleComboBox; 3 usages
17     private JButton registerButton; 3 usages
18
19     public RegistrationForm() { 1 usage
20         setTitle("User Registration");
21         JPanel mainPanel = new JPanel();
22         mainPanel.setLayout(new GridBagLayout()); // Use GridBagLayout for proper alignment
23         GridBagConstraints gbc = new GridBagConstraints();
24         mainPanel.setBackground(Color.LIGHT_GRAY);
25
26         // Create form elements
27         JLabel usernameLabel = new JLabel( text: "Username:");
28         usernameField = new JTextField( columns: 15);
29
30         JLabel passwordLabel = new JLabel( text: "Password:");
31         passwordField = new JPasswordField( columns: 15);
32
33         JLabel roleLabel = new JLabel( text: "Role:");
34         String[] roles = {"Student", "Teacher"};
```

```

35         roleComboBox = new JComboBox<>(roles);
36
37         registerButton = new JButton( text: "Register");
38         gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10);
39
40     // Layout setup
41     // Username label
42     gbc.gridx = 0;
43     gbc.gridy = 0;
44     gbc.anchor = GridBagConstraints.EAST; // Align to the right
45     mainPanel.add(usernameLabel, gbc);
46
47     // Username field
48     gbc.gridx = 1;
49     gbc.gridy = 0;
50     gbc.anchor = GridBagConstraints.WEST; // Align to the left
51     mainPanel.add(usernameField, gbc);
52
53     // Password label
54     gbc.gridx = 0;
55     gbc.gridy = 1;
56     gbc.anchor = GridBagConstraints.EAST; // Align to the right
57     mainPanel.add(passwordLabel, gbc);
58
59     // Password field
60     gbc.gridx = 1;
61     gbc.gridy = 1;
62     gbc.anchor = GridBagConstraints.WEST; // Align to the left
63     mainPanel.add(passwordField, gbc);
64

```

```

65     // Role label
66     gbc.gridx = 0;
67     gbc.gridy = 2;
68     gbc.anchor = GridBagConstraints.EAST; // Align to the right
69     mainPanel.add(roleLabel, gbc);
70
71     // Role combo box
72     gbc.gridx = 1;
73     gbc.gridy = 2;
74     gbc.anchor = GridBagConstraints.WEST; // Align to the left
75     mainPanel.add(roleComboBox, gbc);
76
77     // Login button
78     gbc.gridx = 1;
79     gbc.gridy = 3;
80     gbc.anchor = GridBagConstraints.CENTER; // Center the button
81     mainPanel.add(registerButton, gbc);
82     getContentPane().setBackground(Color.CYAN); // Set the entire window's background color
83
84     // Add the mainPanel to the JFrame
85     add(mainPanel);
86
87     // Register button action listener
88     registerButton.addActionListener(new ActionListener() {
89
90         @Override
91         public void actionPerformed(ActionEvent e) {
92             String username = usernameField.getText();
93             String password = new String(passwordField.getPassword());
94             String role = (String) roleComboBox.getSelectedItem();
95             if (username.isEmpty() || password.isEmpty()) {
96
97             }
98         }
99     });

```

```

95         JOptionPane.showMessageDialog( parentComponent: null, message: "Username or password cannot be empty");
96     } else {
97         // Save user to the database
98         registerUser(username, password, role);
99     }
100 }
101 );
102
103 // JFrame settings
104 setSize( width: 600, height: 400);
105 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
106 setVisible(true);
107 }
108
109 // Method to register the user in the database
110 @private void registerUser(String username, String password, String role) { 1usage
111     Connection connection = null;
112     PreparedStatement preparedStatement = null;
113
114     try {
115         // Get a connection to the database
116         connection = DBConnection.getConnection();
117
118         // SQL query to insert the user data into the 'users' table
119         String insertQuery = "INSERT INTO users (username, password, role) VALUES (?, ?, ?)";
120         preparedStatement = connection.prepareStatement(insertQuery);
121         preparedStatement.setString( parameterIndex: 1, username);

```

```

    preparedStatement.setString( parameterIndex: 2, password);
    preparedStatement.setString( parameterIndex: 3, role);
    preparedStatement.executeUpdate(); // Store role in lowercase
}

// 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

```
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 1
            System.out.println("Background image loaded successfully.");
        } catch (IOException e) {
            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);
    gbc.gridx = 0;
    gbc.gridy = 0;
```

```
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
```

```
65     gbc.anchor = GridBagConstraints.EAST; // Align to the right
66     mainPanel.add(usernameLabel, gbc);
67
68     // Username field
69     gbc.gridx = 1;
70     gbc.gridy = 0;
71     gbc.anchor = GridBagConstraints.WEST; // Align to the left
72     mainPanel.add(usernameField, gbc);
73
74     // Password label
75     gbc.gridx = 0;
76     gbc.gridy = 1;
77     gbc.anchor = GridBagConstraints.EAST; // Align to the right
78     mainPanel.add(passwordLabel, gbc);
79
80     // Password field
81     gbc.gridx = 1;
82     gbc.gridy = 1;
83     gbc.anchor = GridBagConstraints.WEST; // Align to the left
84     mainPanel.add(passwordField, gbc);
85
86     // Role label
87     gbc.gridx = 0;
88     gbc.gridy = 2;
89     gbc.anchor = GridBagConstraints.EAST; // Align to the right
90     mainPanel.add(roleLabel, gbc);
91
92     // Role combo box
93     gbc.gridx = 1;
```

```
96     mainPanel.add(roleComboBox, gbc);
97
98     // Login button
99     gbc.gridx = 1;
100    gbc.gridy = 3;
101    gbc.anchor = GridBagConstraints.CENTER; // Center the button
102    mainPanel.add(loginButton, gbc);
103    getContentPane().setBackground(Color.CYAN);
104    add(mainPanel);
105
106    // Login button action listener
107    loginButton.addActionListener(new ActionListener() {
108        @Override
109        public void actionPerformed(ActionEvent e) {
110            String username = usernameField.getText();
111            String password = new String(passwordField.getPassword());
112            String role = (String) roleComboBox.getSelectedItem();
113
114            if (username.isEmpty() || password.isEmpty()) {
115                JOptionPane.showMessageDialog( parentComponent: null, message: "Username or password cannot be empty");
116            } else {
117                // Verify login credentials
118                if (loginUser(username, password, role)) {
119                    JOptionPane.showMessageDialog( parentComponent: null, message: "Login successful!");
120
121                    // Redirect based on role
122                    if (role.equalsIgnoreCase("teacher")) {
123                        new TeacherDashboard(username);
124                    } else if (role.equalsIgnoreCase("student")) {
125                        new StudentDashboard(username);
126                    }
127                }
128            }
129        }
130    });
131}
```

```

128             dispose(); // Close the login window
129         } else {
130             JOptionPane.showMessageDialog( parentComponent: null, message: "Invalid credentials. Please try again." );
131         }
132     }
133 }
134 });
135
136 // JFrame settings
137 setSize( width: 600, height: 400 );
138 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
139 setVisible(true);
140 }
141
142 // Method to verify the user's credentials from the database
143 @
144 private boolean loginUser(String username, String password, String role) { 1 usage
145     Connection connection = null;
146     PreparedStatement preparedStatement = null;
147     ResultSet resultSet = null;
148
149     DBConnection.closeConnection(connection);
150     try {
151         if (resultSet != null) resultSet.close();
152         if (preparedStatement != null) preparedStatement.close();
153     } catch (SQLException e) {
154         e.printStackTrace(); [Call to 'printStackTrace()' should probably be replaced with more robust logging]
155     }
156 }
157
158 return false; // Return false if no user matches the provided credentials

```

```

159
160     preparedStatement.setString( parameterIndex: 2, password); // If you're using hashing, compare hashed passw
161     preparedStatement.setString( parameterIndex: 3, role.toLowerCase());
162
163     // Execute the query
164     resultSet = preparedStatement.executeQuery();
165
166     // Return true if a matching user is found
167     return resultSet.next();
168 } catch (SQLException e) {
169     e.printStackTrace();
170 } finally {
171     // Close the resources
172     DBConnection.closeConnection(connection);
173     try {
174         if (resultSet != null) resultSet.close();
175         if (preparedStatement != null) preparedStatement.close();
176     } catch (SQLException e) {
177         e.printStackTrace();
178     }
179 }
180
181 return false; // Return false if no user matches the provided credentials
182 }
183
184 public static void main(String[] args) {
185     new Login();
186 }
187 class BackgroundPanel extends JPanel { 2 usages
188     private Image backgroundImage; 3 usages
189
190     public BackgroundPanel(Image backgroundImage) { 1 usage
191         this.backgroundImage = backgroundImage;
192     }
193 }

```

```

182     }
183     class BackgroundPanel extends JPanel { 2 usages
184         private Image backgroundImage; 3 usages
185
186         public BackgroundPanel(Image backgroundImage) { 1 usage
187             this.backgroundImage = backgroundImage;
188         }
189
190         @Override
191         protected void paintComponent(Graphics g) {
192             super.paintComponent(g);
193             if (backgroundImage != null) {
194                 g.drawImage(backgroundImage, x: 0, y: 0, getWidth(), getHeight(), observer: this);
195             }
196         }
197     }
198 }
199

```

3) Chat box

```

1  package dashboard;
2
3  import database.DBConnection;
4
5  import javax.swing.*;
6  import java.awt.event.ActionEvent;
7  import java.awt.event.ActionListener;
8  import java.sql.Connection;
9  import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12
13 > public class Chatbox extends JFrame {
14     private JTextArea chatArea; 4 usages
15     private JTextField inputField; 4 usages
16     private JButton sendButton; 3 usages
17
18     public Chatbox() { 1 usage
19         setTitle("Chatbox");
20
21         chatArea = new JTextArea( rows: 20, columns: 30 );
22         inputField = new JTextField( columns: 20 );
23         sendButton = new JButton( text: "Send" );
24
25         setLayout(new java.awt.FlowLayout());
26         add(new JScrollPane(chatArea));
27         add(inputField);
28         add(sendButton);
29
30         // Load chat history on startup
31         loadChatHistory();
32
33         // Send button action
34         sendButton.addActionListener(new ActionListener() {

```

```

35     @Override
36     public void actionPerformed(ActionEvent e) {
37         String message = inputField.getText();
38         chatArea.append("You: " + message + "\n");
39         saveMessageToDatabase(username: "You", message); // Save message to DB
40         inputField.setText("");
41     }
42 );
43
44 setSize( width: 400, height: 400);
45 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46 setVisible(true);
47 }
48
49 // Method to load chat history from the database
50 private void loadChatHistory() { 1 usage
51     Connection connection = DBConnection.getConnection();
52     try {
53         String query = "SELECT * FROM chat_history";
54         PreparedStatement preparedStatement = connection.prepareStatement(query);
55         ResultSet resultSet = preparedStatement.executeQuery();
56         while (resultSet.next()) {
57             chatArea.append(resultSet.getString(columnLabel: "username") + ":" + resultSet.getString(columnLabel: "message"));
58         }
59     } catch (SQLException e) {
60         e.printStackTrace();
61     } finally {
62         DBConnection.closeConnection(connection);
63     }
64 }
65
66 // Method to save a message to the database
67 private void saveMessageToDatabase(String username, String message) { 1 usage
68     Connection connection = DBConnection.getConnection();
69     try {
70         String query = "INSERT INTO chat_history (username, message) VALUES (?, ?)";
71         PreparedStatement preparedStatement = connection.prepareStatement(query);
72         preparedStatement.setString(parameterIndex: 1, username);
73         preparedStatement.setString(parameterIndex: 2, message);
74         preparedStatement.executeUpdate();
75     } catch (SQLException e) {
76         e.printStackTrace();
77     } finally {
78         DBConnection.closeConnection(connection);
79     }
80 }
81
82 ▶ public static void main(String[] args) {
83     new Chatbox();
84 }
85 }
86

```

4)Student Dashboard

```
1 package dashboard;
2
3 import database.DBConnection;
4
5 import javax.swing.*;
6 import java.awt.*;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import java.io.File;
10 import java.io.IOException;
11 import java.sql.Connection;
12 import java.sql.PreparedStatement;
13 import java.sql.ResultSet;
14 import java.sql.SQLException;
15 import java.util.HashMap;
16 import java.util.Map;
17 import javax.swing.*;
18 import java.awt.*;
19 import java.awt.event.*;
20
21 public class StudentDashboard extends JFrame {
22     private String username; 9 usages
23     private JPanel cardPanel; // CardLayout to switch between different panels 22 usages
24     private CardLayout cardLayout; 12 usages
25
26     public StudentDashboard(String username) { 2 usages
27         this.username = username;
28         setTitle("Student Dashboard - Welcome " + username);
29
30         cardLayout = new CardLayout();
31         cardPanel = new JPanel(cardLayout); // Set CardLayout on the panel
32
33         // Main panel containing buttons for each option
34         JPanel mainPanel = createMainPage();
35         // Add the mainPanel to cardPanel
36         cardPanel.add(mainPanel, constraints: "Main");
37
38         // Set JFrame settings
39         setLayout(new BorderLayout());
40         add(cardPanel, BorderLayout.CENTER); // Set cardPanel as the center
41
42         setSize( width: 600, height: 400 );
43         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
44         setVisible(true);
45     }
46
47     @
48     private JPanel createMainPage() { 1 usage
49         JPanel mainPanel = new JPanel(new GridLayout( rows: 6, cols: 1, hgap: 5, vgap: 5));
50         mainPanel.setBorder(BorderFactory.createEmptyBorder( top: 10, left: 10, bottom: 10, right: 10)); // Add padding
51
52
53         JButton viewFilesButton = new JButton( text: "View Uploaded Files");
54         JButton chatButton = new JButton( text: "Chat");
55         JButton submitAssignmentButton = new JButton( text: "Submit Assignment");
56         JButton quizButton = new JButton( text: "Quiz");
57         JButton peerReviewButton = new JButton( text: "Peer Review");
58
59
60         // Add action listeners to buttons
61         viewFilesButton.addActionListener(e -> showViewFilesPanel());
62         chatButton.addActionListener(e -> showChatPanel());
63         submitAssignmentButton.addActionListener(e -> showSubmitAssignmentPanel());
64         quizButton.addActionListener(e -> showQuizPanel());
65         peerReviewButton.addActionListener(e -> showPeerReviewPanel());
66     }
```

```

69     // Add buttons to the main panel
70     mainPanel.add(viewFilesButton);
71     mainPanel.add(chatButton);
72     mainPanel.add(submitAssignmentButton);
73     mainPanel.add(quizButton);
74     mainPanel.add(peerReviewButton);
75
76
77     return mainPanel;
78 }
79 void showPeerReviewPanel() { 2 usages
80     // Create peerReviewPanel
81     JPanel peerReviewPanel = new JPanel(new BorderLayout());
82     JPanel reviewListPanel = new JPanel();
83     reviewListPanel.setLayout(new BoxLayout(reviewListPanel, BoxLayout.Y_AXIS));
84
85     JLabel titleLabel = new JLabel( text: "Assignments Available for Review:");
86     titleLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 16));
87     reviewListPanel.add(titleLabel);
88
89     try (Connection connection = DBConnection.getConnection();
90          PreparedStatement statement = connection.prepareStatement(
91              sql: "SELECT a.id, a.file_path, a.student_username " +
92                  "FROM assignments_submitted1 a " +
93                  "LEFT JOIN peer_reviews p ON a.id = p.assignment_id AND p.reviewer_username = ? " +
94                  "WHERE p.assignment_id IS NULL AND a.reviewable = TRUE")) {
95
96         statement.setString( parameterIndex: 1, username); // Set the current user's username
97         ResultSet resultSet = statement.executeQuery();
98
99         if (!resultSet.isBeforeFirst()) { // Check if there are any results
100             JLabel noAssignmentsLabel = new JLabel( text: "No assignments available for review at this time.");
101             reviewListPanel.add(noAssignmentsLabel);
102         } else {
103             while (resultSet.next()) {
104                 int assignmentId = resultSet.getInt( columnLabel: "id");
105                 String studentUsername = resultSet.getString( columnLabel: "student_username");
106                 String filePath = resultSet.getString( columnLabel: "file_path");
107
108                 JPanel assignmentPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
109                 JLabel assignmentLabel = new JLabel( text: "Assignment from: " + studentUsername);
110
111                 // Open File Button
112                 JButton openFileButton = new JButton( text: "Open File");
113                 openFileButton.addActionListener(e -> openAssignmentFile(filePath));
114
115                 JTextField feedbackField = new JTextField( columns: 30);
116                 JButton submitFeedbackButton = new JButton( text: "Submit Feedback");
117
118                 submitFeedbackButton.addActionListener(e -> {
119                     String feedback = feedbackField.getText();
120                     System.out.println("Submitting feedback: " + feedback);
121
122                     if (submitFeedback(assignmentId, feedback)) {
123                         JOptionPane.showMessageDialog( parentComponent: null, message: "Feedback submitted successfully!");
124                         System.out.println("Feedback submission successful");
125                         showPeerReviewPanel(); // Refresh the panel
126                     } else {
127                         JOptionPane.showMessageDialog( parentComponent: null, message: "Failed to submit feedback.");
128                         System.out.println("Feedback submission failed");
129                     }
130                 });
131
132                 assignmentPanel.add(assignmentLabel);

```

```

132         assignmentPanel.add(assignmentLabel);
133         assignmentPanel.add(openFileButton);
134         assignmentPanel.add(feedbackField);
135         assignmentPanel.add(submitFeedbackButton);
136         reviewListPanel.add(assignmentPanel);
137     }
138 }
139 } catch (SQLException e) {
140     JLabel errorLabel = new JLabel( text: "Error loading assignments: " + e.getMessage());
141     reviewListPanel.add(errorLabel);
142     e.printStackTrace();
143 }
144
145 // Add the review list to the main peer review panel
146 peerReviewPanel.add(new JScrollPane(reviewListPanel), BorderLayout.CENTER);
147
148 // "Back" button to return to the main dashboard
149 JButton backButton = new JButton( text: "Back to Dashboard");
150 backButton.addActionListener(e -> cardLayout.show(cardPanel, name: "Main")); // Switch back to the main panel
151
152 // Add the "Back" button at the bottom
153 peerReviewPanel.add(backButton, BorderLayout.SOUTH);
154
155 // Ensure panel is added to the card layout and displayed
156 cardPanel.add(peerReviewPanel, constraints: "PeerReview"); // Add peerReviewPanel to cardPanel
157 cardLayout.show(cardPanel, name: "PeerReview"); // Show peerReviewPanel
158
159 revalidate();
160 repaint();
161 }
162
163 private void openAssignmentFile(String filePath) { 1usage
164     try {
165         File file = new File(filePath);
166         if (file.exists()) {
167             Desktop.getDesktop().open(file); // Open file in default application
168         } else {
169             JOptionPane.showMessageDialog( parentComponent: this, message: "File not found: " + filePath);
170         }
171     } catch (IOException e) {
172         JOptionPane.showMessageDialog( parentComponent: this, message: "Error opening file: " + e.getMessage());
173         e.printStackTrace();
174     }
175 }
176
177
178 // Method to submit feedback to the database
179 // Method to submit feedback to the database
180 private boolean submitFeedback(int assignmentId, String feedback) { 1usage
181     String reviewedStudentUsername = getReviewedStudentUsername(assignmentId);
182
183     // Log assignmentId, username, and reviewedStudentUsername to verify they are correctly retrieved
184     System.out.println("Attempting to submit feedback...");
185     System.out.println("Assignment ID: " + assignmentId);
186     System.out.println("Feedback: " + feedback);
187     System.out.println("Reviewer (username): " + username);
188     System.out.println("Reviewed Student Username: " + reviewedStudentUsername);
189
190     if (reviewedStudentUsername == null || reviewedStudentUsername.isEmpty()) {
191         System.out.println("Error: Reviewed student username is null or empty for assignment ID " + assignmentId);
192         return false;
193     }
194
195     // Ensure username is not null
196     if (username == null || username.isEmpty()) {
197

```

```

199     System.out.println("Error: Reviewer username is not set. Ensure 'username' is properly initialized.");
200     return false;
201 }
202
203 try (Connection connection = DBConnection.getConnection();
204      PreparedStatement preparedStatement = connection.prepareStatement(
205          sql: "INSERT INTO peer_reviews (assignment_id, reviewer_username, reviewed_student_username, feedback)
206
207          // Set parameters for the prepared statement
208          preparedStatement.setInt( parameterIndex: 1, assignmentId);
209          preparedStatement.setString( parameterIndex: 2, username);
210          preparedStatement.setString( parameterIndex: 3, reviewedStudentUsername);
211          preparedStatement.setString( parameterIndex: 4, feedback);
212
213          int rowsAffected = preparedStatement.executeUpdate();
214
215          if (rowsAffected > 0) {
216              System.out.println("Feedback successfully saved to the database.");
217              return true;
218          } else {
219              System.out.println("Failed to save feedback: No rows affected.");
220              return false;
221          }
222      } catch (SQLException e) {
223          System.out.println("Error saving feedback to the database:");
224          e.printStackTrace();
225          return false;
226      }
227 }
228
229 // Method to get the reviewed student's username for an assignment
230 @
231 private String getReviewedStudentUsername(int assignmentId) { 1 usage
232     try (Connection connection = DBConnection.getConnection();
233          PreparedStatement statement = connection.prepareStatement(
234              sql: "SELECT student_username FROM assignments_submitted1 WHERE id = ?")) {
235
236         statement.setInt( parameterIndex: 1, assignmentId);
237         ResultSet resultSet = statement.executeQuery();
238
239         if (resultSet.next()) {
240             return resultSet.getString( columnLabel: "student_username");
241         } else {
242             System.out.println("No student found for assignment ID " + assignmentId);
243             return null;
244         }
245     } catch (SQLException e) {
246         System.out.println("Error retrieving reviewed student username:");
247         e.printStackTrace();
248         return null;
249     }
250
251
252
253
254 // Panel to view uploaded files by teachers
255 private void showViewFilesPanel() { 1 usage
256     JPanel filesPanel = new JPanel(new BorderLayout());
257     JPanel fileListPanel = new JPanel();
258     fileListPanel.setLayout(new BoxLayout(fileListPanel, BoxLayout.Y_AXIS));
259
260     JLabel titleLabel = new JLabel( text: "Files Uploaded by Teachers:");
261     titleLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 16));

```

```

264     try (Connection connection = DBConnection.getConnection();
265          PreparedStatement statement = connection.prepareStatement(sql: "SELECT file_path, teacher_username FROM upload"));
266     ResultSet resultSet = statement.executeQuery() {
267
268     if (!resultSet.isBeforeFirst()) {
269         JLabel noFilesLabel = new JLabel(text: "No files have been uploaded by teachers yet.");
270         fileListPanel.add(noFilesLabel);
271     } else {
272         while (resultSet.next()) {
273             String filePath = resultSet.getString(columnLabel: "file_path");
274             String uploadedBy = resultSet.getString(columnLabel: "teacher_username");
275
276             JPanel fileEntryPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
277             JLabel fileLabel = new JLabel(text: "File: " + filePath + " (Uploaded by: " + uploadedBy + ")");
278             JButton openButton = new JButton(text: "Open");
279
280             openButton.addActionListener(e -> openFile(filePath));
281
282             fileEntryPanel.add(fileLabel);
283             fileEntryPanel.add(openButton);
284             fileListPanel.add(fileEntryPanel);
285         }
286     }
287 } catch (SQLException e) {
288     JLabel errorLabel = new JLabel(text: "Error loading files: " + e.getMessage());
289     fileListPanel.add(errorLabel);
290     e.printStackTrace();
291 }
292
293 filesPanel.add(new JScrollPane(fileListPanel), BorderLayout.CENTER);
294

```

```

295     JButton backButton = new JButton(text: "Back to Dashboard");
296     backButton.addActionListener(e -> cardLayout.show(cardPanel, name: "Main"));
297
298     filesPanel.add(backButton, BorderLayout.SOUTH);
299
300     cardPanel.add(filesPanel, constraints: "ViewFiles"); // Add filesPanel to cardPanel with identifier "ViewFiles"
301     cardLayout.show(cardPanel, name: "ViewFiles"); // Display the filesPanel
302
303     cardPanel.revalidate();
304     cardPanel.repaint();
305 }
306
307
308
309 // Method to open the selected file
310 private void openFile(String filePath) { 1 usage
311     try {
312         File file = new File(filePath);
313         if (file.exists()) {
314             Desktop.getDesktop().open(file);
315         } else {
316             JOptionPane.showMessageDialog(parentComponent: this, message: "File not found: " + filePath);
317         }
318     } catch (IOException e) {
319         JOptionPane.showMessageDialog(parentComponent: this, message: "Error opening file: " + e.getMessage());
320         e.printStackTrace();
321     }
322 }
323
324 // Panel to submit an assignment
325 private void showSubmitAssignmentPanel() { 1 usage
326     JPanel submitPanel = new JPanel(new BorderLayout());
327     JLabel instructionLabel = new JLabel(text: "Select an assignment file to upload:");

```

```

329     selectedFileArea.setEditable(false);
330     JButton submitButton = new JButton( text: "Submit Assignment");
331     JButton backButton = new JButton( text: "Back to Dashboard");
332
333     submitPanel.add(instructionLabel, BorderLayout.NORTH);
334     submitPanel.add(selectedFileArea, BorderLayout.CENTER);
335
336     JPanel buttonPanel = new JPanel(new FlowLayout());
337     buttonPanel.add(submitButton);
338     buttonPanel.add(backButton);
339     submitPanel.add(buttonPanel, BorderLayout.SOUTH);
340
341     submitButton.addActionListener(e -> {
342         JFileChooser fileChooser = new JFileChooser();
343         int result = fileChooser.showOpenDialog( parent: null);
344
345         if (result == JFileChooser.APPROVE_OPTION) {
346             File selectedFile = fileChooser.getSelectedFile();
347             String filePath = selectedFile.getAbsolutePath();
348             selectedFileArea.setText("Selected file: " + filePath);
349
350             if (saveAssignmentToDatabase(filePath)) {
351                 JOptionPane.showMessageDialog( parentComponent: null, message: "Assignment submitted successfully!");
352             } else {
353                 JOptionPane.showMessageDialog( parentComponent: null, message: "Failed to submit assignment.");
354             }
355         }
356     });
357
358     // Back button to return to dashboard
359     backButton.addActionListener(e -> cardLayout.show(cardPanel, name: "Main"));
360
361     cardPanel.add(submitPanel, constraints: "SubmitAssignment");
362     cardLayout.show(cardPanel, name: "SubmitAssignment");
363 }
364
365
366
367
368     // Method to save submitted assignment to the database
369     private boolean saveAssignmentToDatabase(String filePath) { 1 usage
370         try (Connection connection = DBConnection.getConnection();
371              PreparedStatement preparedStatement = connection.prepareStatement(
372                  sql: "INSERT INTO assignments_submitted1 (student_username, file_path) VALUES (?, ?)")) {
373
374             preparedStatement.setString( parameterIndex: 1, username);
375             preparedStatement.setString( parameterIndex: 2, filePath);
376             int rowsAffected = preparedStatement.executeUpdate();
377             return rowsAffected > 0;
378         } catch (SQLException e) {
379             System.out.println("Error saving assignment:");
380             e.printStackTrace();
381             return false;
382         }
383     }
384
385     // Chat panel to display chat messages and send new ones
386     private void showChatPanel() { 1 usage
387         JPanel chatPanel = new JPanel(new BorderLayout());
388         chatPanel.add(new JLabel( text: "Chat"), BorderLayout.NORTH);
389
390         JTextArea chatArea = new JTextArea( rows: 15, columns: 50);
391         chatArea.setEditable(false);
392         JTextField chatInputField = new JTextField( columns: 40);
393         JButton sendButton = new JButton( text: "Send");

```

```

395     JPanel inputPanel = new JPanel();
396     inputPanel.add(chatInputField);
397     inputPanel.add(sendButton);
398
399     chatPanel.add(new JScrollPane(chatArea), BorderLayout.CENTER);
400     chatPanel.add(inputPanel, BorderLayout.SOUTH);
401
402     // Back button at the top to return to the main dashboard
403     JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
404     topPanel.add(backButton);
405     chatPanel.add(topPanel, BorderLayout.NORTH);
406
407     // Load chat history from the database
408     loadChatHistory(chatArea);
409
410     // Send button action to save message to the database
411     sendButton.addActionListener(e -> {
412         String message = chatInputField.getText();
413         if (!message.isEmpty()) {
414             if (saveChatMessageToDatabase(username, message)) {
415                 chatArea.append("You: " + message + "\n");
416                 chatInputField.setText("");
417             } else {
418                 JOptionPane.showMessageDialog(parentComponent: this, message: "Failed to send message.");
419             }
420         }
421     });
422
423
424     // Back button action to return to the main dashboard
425     backButton.addActionListener(e -> cardLayout.show(cardPanel, name: "Main"));
426     if (cardPanel.getComponentCount() == 1) // Add only if it hasn't been added before
427         cardPanel.add(chatPanel, constraints: "Chat");
428
429     cardLayout.show(cardPanel, name: "Chat");
430
431 }
432
433
434
435     // Method to load chat history from the database
436     private void loadChatHistory(JTextArea chatArea) { 1usage
437         try (Connection connection = DBConnection.getConnection();
438              PreparedStatement preparedStatement = connection.prepareStatement(
439                  sql: "SELECT sender_username, message FROM chats ORDER BY timestamp");
440              ResultSet resultSet = preparedStatement.executeQuery()) {
441
442             while (resultSet.next()) {
443                 String sender = resultSet.getString(columnLabel: "sender_username");
444                 String message = resultSet.getString(columnLabel: "message");
445                 chatArea.append(sender + ": " + message + "\n");
446             }
447         } catch (SQLException e) {
448             System.out.println("Error loading chat history:");
449             e.printStackTrace();
450         }
451     }
452
453     // Method to save chat messages to the database
454     private boolean saveChatMessageToDatabase(String senderUsername, String message) { 1usage
455         try (Connection connection = DBConnection.getConnection();
456              PreparedStatement preparedStatement = connection.prepareStatement(
457                  sql: "INSERT INTO chats (sender_username, message) VALUES (?, ?)")) {
458
459             preparedStatement.setString(parameterIndex: 1, senderUsername);

```

```

461         int rowsAffected = preparedStatement.executeUpdate();
462         return rowsAffected > 0;
463     } catch (SQLException e) {
464         System.out.println("Error saving chat message:");
465         e.printStackTrace();
466         return false;
467     }
468 }
469
470 private void showQuizPanel() { 1 usage
471     getContentPane().removeAll();
472     JPanel quizSelectionPanel = new JPanel(new BorderLayout());
473     JPanel listPanel = new JPanel();
474     listPanel.setLayout(new BoxLayout(listPanel, BoxLayout.Y_AXIS));
475
476     JLabel titleLabel = new JLabel( text: "Select a Quiz:");
477     titleLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 16));
478     listPanel.add(titleLabel);
479
480     // Fetch all quizzes from the database
481     try (Connection connection = DBConnection.getConnection();
482          PreparedStatement statement = connection.prepareStatement(sql: "SELECT quiz_id, title FROM quizzes ORDER BY crea
483          ResultSet resultSet = statement.executeQuery();
484
485          while (resultSet.next()) {
486              int quizId = resultSet.getInt( columnLabel: "quiz_id");
487              String quizTitle = resultSet.getString( columnLabel: "title");
488
489              JButton quizButton = new JButton(quizTitle);
490              quizButton.addActionListener(e -> showQuizPanel(quizId, quizTitle));
491              listPanel.add(quizButton);
492
493          } catch (SQLException e) {
494              listPanel.add(new JLabel( text: "Error loading quizzes: " + e.getMessage()));
495              e.printStackTrace();
496          }
497
498      }
499
500      quizSelectionPanel.add(new JScrollPane(listPanel), BorderLayout.CENTER);
501      add(quizSelectionPanel);
502      revalidate();
503      repaint();
504  }
505
506  // Display questions for the selected quiz
507  private void showQuizPanel(int quizId, String quizTitle) { 1 usage
508      getContentPane().removeAll();
509      JPanel quizPanel = new JPanel(new BorderLayout());
510      JPanel questionPanel = new JPanel();
511      questionPanel.setLayout(new BoxLayout(questionPanel, BoxLayout.Y_AXIS));
512
513      JLabel titleLabel = new JLabel( text: "Quiz: " + quizTitle);
514      titleLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 16));
515      questionPanel.add(titleLabel);
516
517      JButton submitQuizButton = new JButton( text: "Submit Quiz");
518      JButton backButton = new JButton( text: "Back to Dashboard"); // Back button
519
520      // Maps to store questions and options
521      Map<Integer, String> questionMap = new HashMap<>();
522      Map<Integer, String> correctAnswers = new HashMap<>();
523      Map<Integer, JRadioButton> selectedOptions = new HashMap<>();
524
525      // Fetch questions for the selected quiz ID

```

```

527     PreparedStatement statement = connection.prepareStatement(sql: "SELECT * FROM questions WHERE quiz_id = ?"));
528     statement.setInt(parameterIndex: 1, quizId);
529     ResultSet resultSet = statement.executeQuery();
530
531     while (resultSet.next()) {
532         int questionId = resultSet.getInt(columnLabel: "question_id");
533         String questionText = resultSet.getString(columnLabel: "question_text");
534         String optionA = resultSet.getString(columnLabel: "option_a");
535         String optionB = resultSet.getString(columnLabel: "option_b");
536         String optionC = resultSet.getString(columnLabel: "option_c");
537         String optionD = resultSet.getString(columnLabel: "option_d");
538         String correctAnswer = resultSet.getString(columnLabel: "correct_answer");
539
540         questionMap.put(questionId, questionText);
541         correctAnswers.put(questionId, correctAnswer);
542
543         questionPanel.add(new JLabel(text: "Question: " + questionText));
544
545         ButtonGroup optionGroup = new ButtonGroup();
546         JRadioButton optA = new JRadioButton(optionA);
547         JRadioButton optB = new JRadioButton(optionB);
548         JRadioButton optC = new JRadioButton(optionC);
549         JRadioButton optD = new JRadioButton(optionD);
550
551         optionGroup.add(optA);
552         optionGroup.add(optB);
553         optionGroup.add(optC);
554         optionGroup.add(optD);
555
556         JPanel optionsPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
557         optionsPanel.add(optA);
558         optionsPanel.add(optB);
559         optionsPanel.add(optC);
560         optionsPanel.add(optD);
561
562         questionPanel.add(optionsPanel);
563
564         // Store selected option for each question
565         selectedOptions.put(questionId, optA); // Default selection as example
566         optA.addActionListener(e -> selectedOptions.put(questionId, optA));
567         optB.addActionListener(e -> selectedOptions.put(questionId, optB));
568         optC.addActionListener(e -> selectedOptions.put(questionId, optC));
569         optD.addActionListener(e -> selectedOptions.put(questionId, optD));
570     }
571 } catch (SQLException e) {
572     questionPanel.add(new JLabel(text: "Error loading quiz: " + e.getMessage()));
573     e.printStackTrace();
574 }
575
576 submitQuizButton.addActionListener(e -> {
577     int score = 0;
578     try (Connection connection = DBConnection.getConnection()) {
579         connection.setAutoCommit(false); // Start a transaction
580
581         for (Map.Entry<Integer, JRadioButton> entry : selectedOptions.entrySet()) {
582             int questionId = entry.getKey();
583             JRadioButton selectedOption = entry.getValue();
584             String selectedAnswer = selectedOption.getText();
585
586             // Check the answer and calculate the score
587             if (selectedAnswer.equals(correctAnswers.get(questionId))) {
588                 score++;
589             }
590
591             // Prepare the SQL statement to save the response

```

```

571 // Prepare the SQL statement to save the response
572 String responseQuery = "INSERT INTO responses (student_username, quiz_id, question_id, poll_id, selected
573           "VALUES (?, ?, ?, NULL, ?, NOW())";
574 try (PreparedStatement responseStmt = connection.prepareStatement(responseQuery)) {
575     responseStmt.setString( parameterIndex: 1, username); // student's username
576     responseStmt.setInt( parameterIndex: 2, quizId); // quiz ID
577     responseStmt.setInt( parameterIndex: 3, questionId); // question ID
578     responseStmt.setString( parameterIndex: 4, selectedAnswer);
579     // selected answer
580
581     // Execute the insert
582     int rowsInserted = responseStmt.executeUpdate();
583     if (rowsInserted == 0) {
584         throw new SQLException("Failed to insert response for question ID: " + questionId);
585     }
586 }
587
588 // Commit the transaction
589 connection.commit();
590
591 // Display the score to the user
592 JOptionPane.showMessageDialog( parentComponent: this, message: "Your Score: " + score + " out of " + questionMa
593 } catch (SQLException ex) {
594     ex.printStackTrace();
595     JOptionPane.showMessageDialog( parentComponent: this, message: "Failed to save quiz responses: " + ex.getMessage()
596
597     // Rollback if any failure occurred
598     try (Connection connection = DBConnection.getConnection()) {
599         connection.rollback();
600     } catch (SQLException rollbackEx) {
601
602     });
603
604     // Back button action to return to the main dashboard
605     backButton.addActionListener(e -> cardLayout.show(cardPanel, name: "Main"));
606
607     // Add quizPanel to cardPanel with identifier "Quiz" if not already added
608     if (cardPanel.getComponentCount() == 1) { // Add only if it hasn't been added before
609         cardPanel.add(quizPanel, constraints: "Quiz");
610     }
611     cardLayout.show(cardPanel, name: "Quiz");
612
613     // Add the back button to the quiz panel
614     JPanel buttonPanel = new JPanel();
615     buttonPanel.add(backButton);
616     quizPanel.add(buttonPanel, BorderLayout.NORTH);
617
618     quizPanel.add(new JScrollPane(questionPanel), BorderLayout.CENTER);
619     quizPanel.add(submitQuizButton, BorderLayout.SOUTH);
620
621     revalidate();
622     repaint();
623 }
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650 // Main method to run the application
651 public static void main(String[] args) {
652     SwingUtilities.invokeLater(() -> new StudentDashboard( username: "student1"));
653 }
654

```

5)Teacher Dashboard

```
1 package dashboard;
2
3 import database.DBConnection;
4
5 import javax.swing.*;
6 import java.awt.*;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.ActionEvent;
12 import java.awt.event.ActionListener;
13 import java.io.File;
14 import java.io.IOException;
15 import java.sql.*;
16 import java.util.ArrayList;
17 import java.util.List;
18 import javax.swing.*;
19 import java.util.ArrayList;
20 import java.util.List;
21 import java.util.Map;
22 import java.util.HashMap;
23
24 public class TeacherDashboard extends JFrame {
25     private String username; 4 usages
26     private JPanel cardPanel; // CardLayout to switch between different panels 6 usages
27     private CardLayout cardLayout; 4 usages
28
29     public TeacherDashboard(String username) { 2 usages
30         this.username = username;
31         setTitle("Teacher Dashboard - Welcome " + username);
32         cardLayout = new CardLayout();
33         cardPanel = new JPanel(cardLayout); // Set CardLayout on the panel
34 }
```

```
36         JPanel mainPanel = createMainPage();
37
38         // Add the mainPanel to cardPanel
39         cardPanel.add(mainPanel, constraints: "Main");
40
41         // Set JFrame settings
42         setLayout(new BorderLayout());
43         add(cardPanel, BorderLayout.CENTER); // Set cardPanel as the center
44
45         setSize( width: 600, height: 400 );
46         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
47         setVisible(true);
48     }
49
50     @
51     private JPanel createMainPage() { 4 usages
52         JPanel mainPanel = new JPanel(new GridLayout( rows: 4, cols: 1, hgap: 5, vgap: 5));
53         mainPanel.setBorder(BorderFactory.createEmptyBorder( top: 10, left: 10, bottom: 10, right: 10));
54         // Create buttons for each option
55         JButton uploadFileButton = new JButton( text: "Upload File");
56         JButton viewSubmittedAssignmentsButton = new JButton( text: "View Submitted Assignments");
57         JButton chatButton = new JButton( text: "Chat");
58         JButton createQuizButton = new JButton( text: "Create Quiz");
59
60         // Add action listeners for each button
61         uploadFileButton.addActionListener(e -> showUploadFilePanel());
62         viewSubmittedAssignmentsButton.addActionListener(e -> showSubmittedAssignmentsPanel());
63         chatButton.addActionListener(e -> showChatPanel());
64         createQuizButton.addActionListener(e -> showCreateQuizPanel());
65
66         // Add buttons to the main panel
67         mainPanel.add(uploadFileButton);
```

⚠ 33

```

68     mainPanel.add(viewSubmittedAssignmentsButton);
69     mainPanel.add(chatButton);
70     mainPanel.add(createQuizButton);
71
72     return mainPanel;
73 }
74 // Panel to upload a file
75 private void showUploadFilePanel() { 1 usage
76     getContentPane().removeAll();
77
78     // Create the back button
79     JButton backButton = new JButton( text: "Back");
80     backButton.addActionListener(e -> showMainDashboard()); // Action to go back to the main dashboard
81
82     // Create the upload panel
83     JPanel uploadPanel = new JPanel(new BorderLayout());
84     JLabel instructionLabel = new JLabel( text: "Select a file to upload:");
85     JTextArea selectedFileArea = new JTextArea( rows: 2, columns: 30);
86     selectedFileArea.setEditable(false);
87     JButton uploadButton = new JButton( text: "Upload File");
88
89     // Add components to the panel
90     uploadPanel.add(instructionLabel, BorderLayout.NORTH);
91     uploadPanel.add(selectedFileArea, BorderLayout.CENTER);
92     uploadPanel.add(uploadButton, BorderLayout.SOUTH);
93
94     // Add the back button at the top
95     JPanel topPanel = new JPanel(new BorderLayout());
96     topPanel.add(backButton, BorderLayout.WEST); // Align the back button to the left
97     uploadPanel.add(topPanel, BorderLayout.NORTH);
98
99     add(uploadPanel, BorderLayout.CENTER);
100
101    // Action listener for upload button
102    uploadButton.addActionListener(e -> {
103        JFileChooser fileChooser = new JFileChooser();
104        int result = fileChooser.showOpenDialog( parent: null);
105
106        if (result == JFileChooser.APPROVE_OPTION) {
107            File selectedFile = fileChooser.getSelectedFile();
108            String filePath = selectedFile.getAbsoluteFilePath();
109            selectedFileArea.setText("Selected file: " + filePath);
110
111            // Save the file path to the database
112            if (saveFilePathToDatabase(filePath)) {
113                JOptionPane.showMessageDialog( parentComponent: null, message: "File uploaded successfully!");
114            } else {
115                JOptionPane.showMessageDialog( parentComponent: null, message: "Failed to upload file.");
116            }
117        }
118    });
119
120    revalidate();
121    repaint();
122 }
123
124 // Method to show the main dashboard when 'Back' is clicked
125 private void showMainDashboard() { 1 usage
126     // Assuming the previous screen is a main dashboard or the main menu
127     // You can use this method to return to the previous dashboard
128     // If you're using CardLayout or multiple screens, you can switch the current panel here.
129
130     // Example:

```

```

-->    private void showMainDashboard() { 1 usage
132        // Add the main panel (or call another method that shows the main screen)
133        JPanel mainPanel = createMainPage(); // Your method to create the main dashboard
134        this.add(mainPanel);
135
136        revalidate();
137        repaint();
138    }
139
140
141     // Method to save uploaded file path to the database in uploaded_files table
142     private boolean saveFilePathToDatabase(String filePath) { 1 usage
143         try (Connection connection = DBConnection.getConnection();
144              PreparedStatement preparedStatement = connection.prepareStatement(
145                  sql: "INSERT INTO uploaded_files (teacher_username, file_path) VALUES (?, ?)")) {
146
147             preparedStatement.setString( parameterIndex: 1, username);
148             preparedStatement.setString( parameterIndex: 2, filePath);
149             int rowsAffected = preparedStatement.executeUpdate();
150             return rowsAffected > 0;
151         } catch (SQLException e) {
152             System.out.println("Error saving uploaded file path:");
153             e.printStackTrace();
154             return false;
155         }
156     }
157
158     // Panel to view submitted assignments
159     // Panel to view submitted assignments
160
161     // Panel to view submitted assignments
162     private void showSubmittedAssignmentsPanel() { 1 usage
163         getContentPane().removeAll();
164
165         // Create the back button
166         JButton backButton = new JButton( text: "Back");
167         backButton.addActionListener(e -> showMainPanel()); // Action to go back to the main page
168
169         // Create the assignment panel
170         JPanel assignmentPanel = new JPanel(new BorderLayout());
171
172         // Create a panel to hold the back button
173         JPanel topPanel = new JPanel(new BorderLayout());
174         topPanel.add(backButton, BorderLayout.WEST); // Align the back button to the left
175         assignmentPanel.add(topPanel, BorderLayout.NORTH);
176
177         assignmentPanel.add(new JLabel( text: "Submitted Assignments"), BorderLayout.CENTER);
178
179         // List model to store assignments
180         DefaultListModel<String> assignmentListModel = new DefaultListModel<>();
181         JList<String> assignmentList = new JList<>(assignmentListModel);
182         assignmentList.setSelectionMode(ListSelectionMode.SINGLE_SELECTION);
183         assignmentPanel.add(new JScrollPane(assignmentList), BorderLayout.CENTER);
184
185         JButton openFileButton = new JButton( text: "Open Selected File");
186         assignmentPanel.add(openFileButton, BorderLayout.SOUTH);
187
188         // Map to store file paths with student info as the key
189         Map<String, String> filePathMap = new HashMap<>();
190
191         // Fetch submitted assignments from the database
192         try (Connection connection = DBConnection.getConnection();
193              PreparedStatement preparedStatement = connection.prepareStatement(
194                  sql: "SELECT student_username, file_path FROM assignments_submitted1"));
195         ResultSet resultSet = preparedStatement.executeQuery()) {
196

```

```

195         ResultSet resultSet = preparedStatement.executeQuery() {
196
197             while (resultSet.next()) {
198                 String studentUsername = resultSet.getString( columnLabel: "student_username");
199                 String filePath = resultSet.getString( columnLabel: "file_path");
200                 String displayText = "Student: " + studentUsername + " - File: " + filePath;
201
202                 assignmentListModel.addElement(displayText);
203                 filePathMap.put(displayText, filePath); // Map display text to file path
204             }
205         } catch (SQLException e) {
206             System.out.println("Error fetching submitted assignments:");
207             e.printStackTrace();
208         }
209
210         // Open selected file when button is clicked
211         openFileButton.addActionListener(e -> {
212             String selectedEntry = assignmentList.getSelectedValue();
213             if (selectedEntry != null) {
214                 String filePath = filePathMap.get(selectedEntry);
215                 try {
216                     Desktop.getDesktop().open(new File(filePath));
217                 } catch (IOException ex) {
218                     ex.printStackTrace();
219                     JOptionPane.showMessageDialog( parentComponent: null, message: "Error opening file.");
220                 }
221             } else {
222                 JOptionPane.showMessageDialog( parentComponent: null, message: "Please select a file to open.");
223             }
224         });
225
226         add(assignmentPanel, BorderLayout.CENTER);
227
228         repaint();
229     }
230
231     // Method to show the main panel when 'Back' is clicked
232     private void showMainPanel() { 1 usage
233         getContentPane().removeAll(); // Remove all components
234
235         // Create the main page panel and add it to the content pane
236         JPanel mainPanel = createMainPage(); // Assuming 'createMainPage' creates the main dashboard
237         add(mainPanel, BorderLayout.CENTER);
238
239         revalidate(); // Refresh the layout
240         repaint();
241     }
242
243
244
245
246     // Panel to create a new quiz
247     // Make quizId a class-level variable
248     private int quizId = -1; 5 usages
249
250     private void showCreateQuizPanel() { 1 usage
251         getContentPane().removeAll();
252
253         // Action to go back to the teacher dashboard
254         // Create the back button
255         JButton backButton = new JButton( text: "Back");
256         backButton.addActionListener(e -> showMainPage());
257         // Create the quiz panel
258         JPanel quizPanel = new JPanel(new BorderLayout());
259

```

```

260     // Create a panel to hold the back button at the top
261     JPanel topPanel = new JPanel(new BorderLayout());
262     topPanel.add(backButton, BorderLayout.WEST); // Align the back button to the left
263     quizPanel.add(topPanel, BorderLayout.NORTH);
264
265     quizPanel.add(new JLabel( text: "Create New Quiz"), BorderLayout.CENTER);
266
267     // Title input
268     JPanel titlePanel = new JPanel(new FlowLayout());
269     JLabel titleLabel = new JLabel( text: "Quiz Title:");
270     JTextField titleField = new JTextField( columns: 20);
271     titlePanel.add(titleLabel);
272     titlePanel.add(titleField);
273     quizPanel.add(titlePanel, BorderLayout.NORTH);
274
275     // Question input section
276     JPanel questionPanel = new JPanel(new GridLayout( rows: 6, cols: 2));
277     questionPanel.add(new JLabel( text: "Question:"));
278     JTextField questionField = new JTextField();
279     questionPanel.add(questionField);
280
281     questionPanel.add(new JLabel( text: "Option A:"));
282     JTextField optionAField = new JTextField();
283     questionPanel.add(optionAField);
284
285     questionPanel.add(new JLabel( text: "Option B:"));
286     JTextField optionBField = new JTextField();
287     questionPanel.add(optionBField);
288
289     questionPanel.add(new JLabel( text: "Option C:"));
290
291     private void showCreateQuizPanel() {
292         questionPanel.add(optionCField);
293
294         questionPanel.add(new JLabel( text: "Option D:"));
295         JTextField optionDField = new JTextField();
296         questionPanel.add(optionDField);
297
298         questionPanel.add(new JLabel( text: "Correct Answer (A, B, C, or D):"));
299         JTextField correctAnswerField = new JTextField();
300         questionPanel.add(correctAnswerField);
301
302         quizPanel.add(questionPanel, BorderLayout.CENTER);
303
304         // Button to add question to quiz
305         JButton addQuestionButton = new JButton( text: "Add Question");
306         JButton saveQuizButton = new JButton( text: "Save Quiz");
307         JPanel buttonPanel = new JPanel();
308         buttonPanel.add(addQuestionButton);
309         buttonPanel.add(saveQuizButton);
310         quizPanel.add(buttonPanel, BorderLayout.SOUTH);
311
312         add(quizPanel, BorderLayout.CENTER);
313
314         // List to display added questions
315         DefaultListModel<String> questionListModel = new DefaultListModel<>();
316         JList<String> questionList = new JList<>(questionListModel);
317         quizPanel.add(new JScrollPane(questionList), BorderLayout.EAST);
318
319         revalidate();
320         repaint();
321
322         // List to store the questions before saving
323         List<Question> questions = new ArrayList<>();

```

```

321     // List to store the questions before saving
322     List<Question> questions = new ArrayList<>();
323
324     // Add action to add question to quiz
325     addQuestionButton.addActionListener(e -> {
326         String questionText = questionField.getText();
327         String optionA = optionAField.getText();
328         String optionB = optionBField.getText();
329         String optionC = optionCField.getText();
330         String optionD = optionDField.getText();
331         String correctAnswer = correctAnswerField.getText();
332
333         // If quizId is not set, save the quiz first
334         if (quizId == -1) {
335             String quizTitle = titleField.getText();
336             quizId = saveQuizToDatabase(quizTitle); // Save quiz and get quizId
337             if (quizId == -1) {
338                 JOptionPane.showMessageDialog(parentComponent: this, message: "Failed to save quiz.");
339                 return; // Exit if saving quiz failed
340             }
341         }
342
343         // Add question to the list
344         questions.add(new Question(questionText, optionA, optionB, optionC, optionD, correctAnswer));
345         questionListModel.addElement("Q: " + questionText);
346
347         // Clear fields after adding
348         questionField.setText("");
349         optionAField.setText("");
350         optionBField.setText("");
351         optionCField.setText("");
352         optionDField.setText("");
353
354         // Show message dialog if failed to save quiz and questions
355         JOptionPane.showMessageDialog(parentComponent: this, message: "Failed to save quiz and questions.");
356     });
357 }
358
359
360 // Method to show the teacher dashboard when 'Back' is clicked
361 private void showMainPage() { 1 usage
362     // Assuming the previous screen is a main dashboard or the main menu
363     // You can use this method to return to the previous dashboard
364     // If you're using CardLayout or multiple screens, you can switch the current panel here.
365
366     // Example:
367     this.getContentPane().removeAll();
368     // Add the main panel (or call another method that shows the main screen)
369     JPanel mainPage = createMainPage(); // Your method to create the main dashboard
370     this.add(mainPage);
371
372     revalidate();
373     repaint();
374 }
375
376
377
378
379
380
381
382
383 // Save quiz to the database and return the generated quiz_id
384 private int saveQuizToDatabase(String quizTitle) { 1 usage
385     try (Connection connection = DBConnection.getConnection()) {
386         // Save quiz title
387         String quizQuery = "INSERT INTO quizzes (teacher_username, title, created_date) VALUES (?, ?, NOW())";
388         PreparedStatement quizStmt = connection.prepareStatement(quizQuery, Statement.RETURN_GENERATED_KEYS);
389         quizStmt.setString( parameterIndex: 1, username); // Assuming 'username' is the teacher's username
390         quizStmt.setString( parameterIndex: 2, quizTitle);
391         quizStmt.executeUpdate();

```

```

373
374     // Retrieve generated quiz ID
375     ResultSet rs = quizStmt.getGeneratedKeys();
376     if (rs.next()) {
377         return rs.getInt( columnIndex: 1); // Return the generated quiz ID
378     } else {
379         System.out.println("Failed to retrieve generated quiz ID.");
380         return -1; // Return -1 if failed
381     }
382 } catch (SQLException e) {
383     System.out.println("Error saving quiz to the database:");
384     e.printStackTrace();
385     return -1; // Return -1 if failed
386 }
387
388 // Save questions to the database with the specified quiz_id
389 @
390 private boolean saveQuestionsToDatabase(int quizId, List<Question> questions) { 1usage
391     try (Connection connection = DBConnection.getConnection()) {
392         // Save each question to the questions table
393         for (Question q : questions) {
394             String questionQuery = "INSERT INTO questions (quiz_id, question_text, option_a, option_b, option_c, option_d,
395             PreparedStatement questionStmt = connection.prepareStatement(questionQuery);
396             questionStmt.setInt( parameterIndex: 1, quizId); // Associate the question with the quiz_id
397             questionStmt.setString( parameterIndex: 2, q.question);
398             questionStmt.setString( parameterIndex: 3, q.optionA);
399             questionStmt.setString( parameterIndex: 4, q.optionB);
400             questionStmt.setString( parameterIndex: 5, q.optionC);
401             questionStmt.setString( parameterIndex: 6, q.optionD);
402             questionStmt.setString( parameterIndex: 7, q.correctAnswer);
403
404             // Check if question insertion succeeds
405             int questionRowsAffected = questionStmt.executeUpdate();
406
407             // Check if question insertion succeeds
408             int questionRowsAffected = questionStmt.executeUpdate();
409             if (questionRowsAffected <= 0) {
410                 System.out.println("Failed to save question: " + q.question);
411                 return false;
412             }
413         }
414         return true; // Return true if all questions were saved successfully
415     } catch (SQLException e) {
416         System.out.println("Error saving questions to the database:");
417         e.printStackTrace();
418         return false; // Return false if there was an error
419     }
420 }
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
private void showChatPanel() { 1usage
    // 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
}

```

```

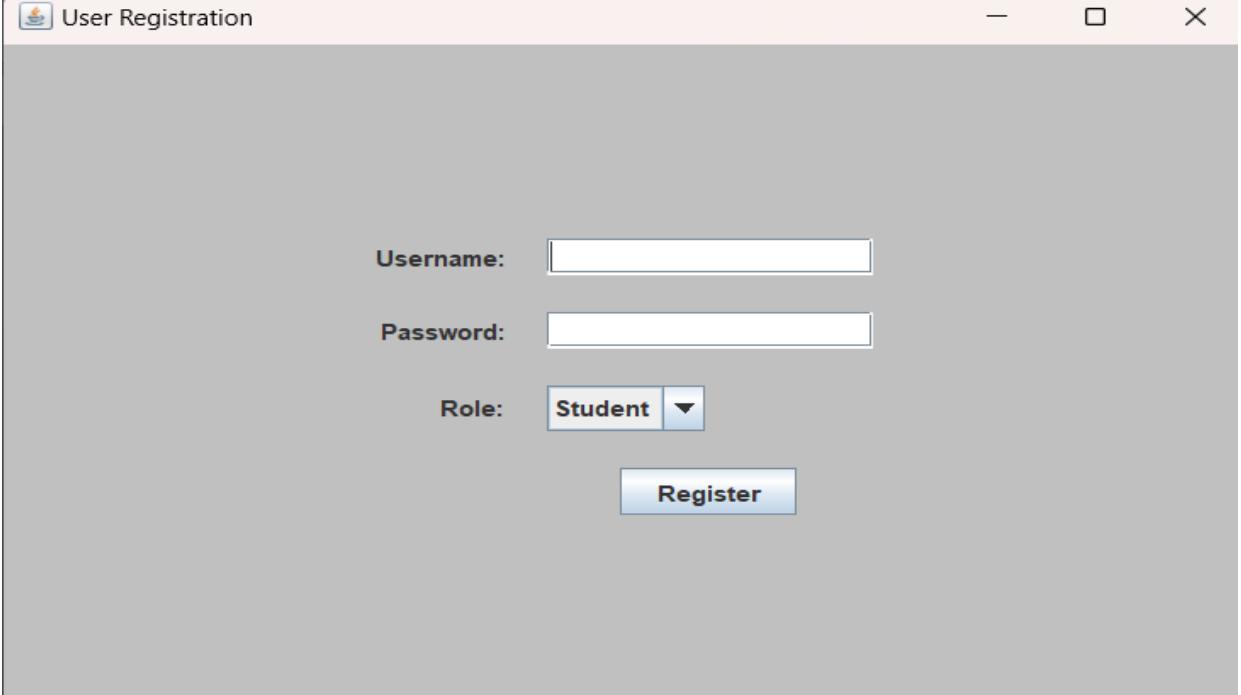
468     chatPanel.add(inputPanel, BorderLayout.SOUTH);
469
470     // Load chat history from the database
471     loadChatHistory(chatArea);
472
473     // Send button action to save message to the database
474     sendButton.addActionListener(e -> {
475         String message = chatInputField.getText();
476         if (!message.isEmpty()) {
477             if (saveChatMessageToDatabase(username, message)) {
478                 chatArea.append("You: " + message + "\n");
479                 chatInputField.setText("");
480             } else {
481                 JOptionPane.showMessageDialog( parentComponent: this, message: "Failed to send message.");
482             }
483         }
484     });
485
486     // Add the chatPanel to the cardLayout
487     cardPanel.add(chatPanel, constraints: "Chat");
488     cardLayout.show(cardPanel, name: "Chat"); // Show the chat panel now
489
490     // Revalidate and repaint the layout to update the UI
491     revalidate();
492     repaint();
493 }
494
495
496
497 // Method to load chat history from the database
498 private void loadChatHistory(JTextArea chatArea) { 1 usage
499     try (Connection connection = DBConnection.getConnection();
500
501         private void loadChatHistory(JTextArea chatArea) { 1 usage
502             try (Connection connection = DBConnection.getConnection();
503                 PreparedStatement preparedStatement = connection.prepareStatement(
504                     sql: "SELECT sender_username, message FROM chats ORDER BY timestamp");
505                 ResultSet resultSet = preparedStatement.executeQuery() {
506
507                     while (resultSet.next()) {
508                         String sender = resultSet.getString( columnLabel: "sender_username");
509                         String message = resultSet.getString( columnLabel: "message");
510                         chatArea.append(sender + ": " + message + "\n");
511                     }
512                 } catch (SQLException e) {
513                     System.out.println("Error loading chat history:");
514                     e.printStackTrace();
515                 }
516
517 // Method to save chat messages to the database
518 private boolean saveChatMessageToDatabase(String senderUsername, String message) { 1 usage
519     try (Connection connection = DBConnection.getConnection();
520         PreparedStatement preparedStatement = connection.prepareStatement(
521             sql: "INSERT INTO chats (sender_username, message) VALUES (?, ?)");
522
523             preparedStatement.setString( parameterIndex: 1, senderUsername);
524             preparedStatement.setString( parameterIndex: 2, message);
525             int rowsAffected = preparedStatement.executeUpdate();
526             return rowsAffected > 0;
527         } catch (SQLException e) {
528             System.out.println("Error saving chat message:");
529             e.printStackTrace();
530             return false;
531         }
532     }

```

```
522     preparedStatement.setString(parameterIndex: 2, message);
523     int rowsAffected = preparedStatement.executeUpdate();
524     return rowsAffected > 0;
525   } catch (SQLException e) {
526     System.out.println("Error saving chat message:");
527     e.printStackTrace();
528     return false;
529   }
530 }
531
532 ▶ public static void main(String[] args) {
533   new TeacherDashboard(username: "teacher1");
534 }
535
536
537 class Question { 4 usages
538   String question, optionA, optionB, optionC, optionD, correctAnswer; 3 usages
539
540   public Question(String question, String optionA, String optionB, String optionC, String optionD, String correctAnswer)
541     this.question = question;
542     this.optionA = optionA;
543     this.optionB = optionB;
544     this.optionC = optionC;
545     this.optionD = optionD;
546     this.correctAnswer = correctAnswer;
547   }
548 }
549 }
550 }
```

6.Result and Analysis

Registration Form:



User Registration

Username:

Password:

Role: Student ▾

Register

This screenshot shows a Windows-style application window titled "User Registration". It contains three input fields: "Username" (empty), "Password" (empty), and a dropdown menu for "Role" set to "Student". A "Register" button is located at the bottom right.

fig.(ii) Output: Registration Form

Login Form:



User Login

Username:

Password:

Role: Student ▾

Login

This screenshot shows a Windows-style application window titled "User Login". It features a background illustration of an open book, a laptop, and various blue icons related to education and technology. It contains three input fields: "Username" (empty), "Password" (empty), and a dropdown menu for "Role" set to "Student". A "Login" button is located at the bottom right.

fig.(iii) Output: Login Form

Chat Box:

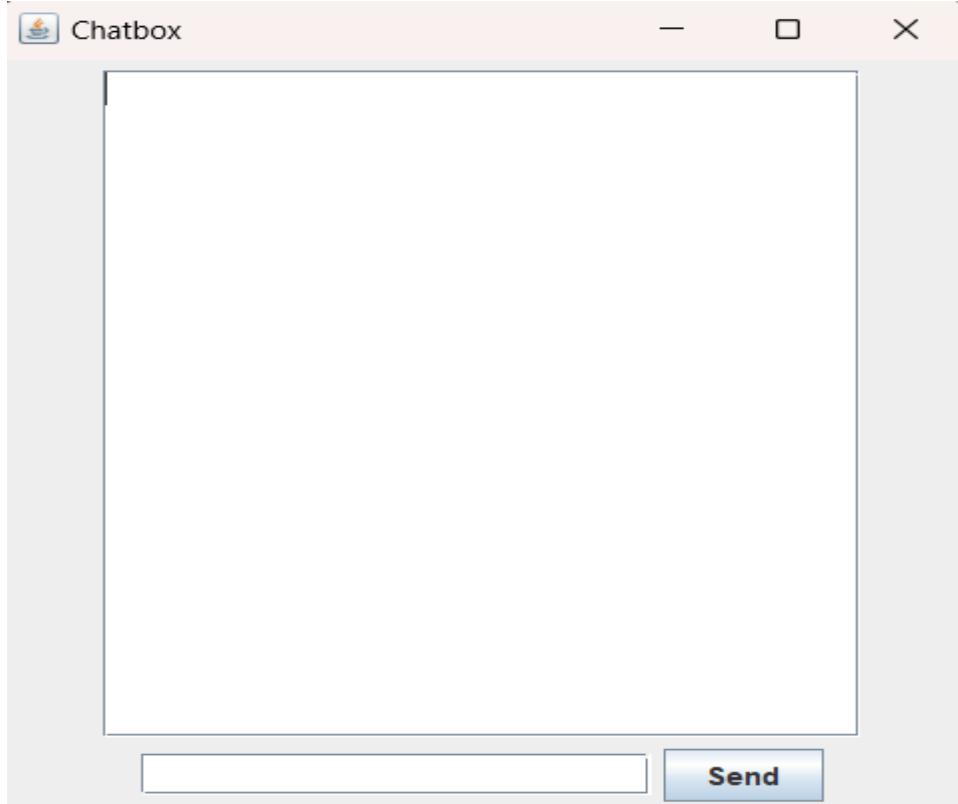


fig.(iv)Chatbox

Student Dashboard:

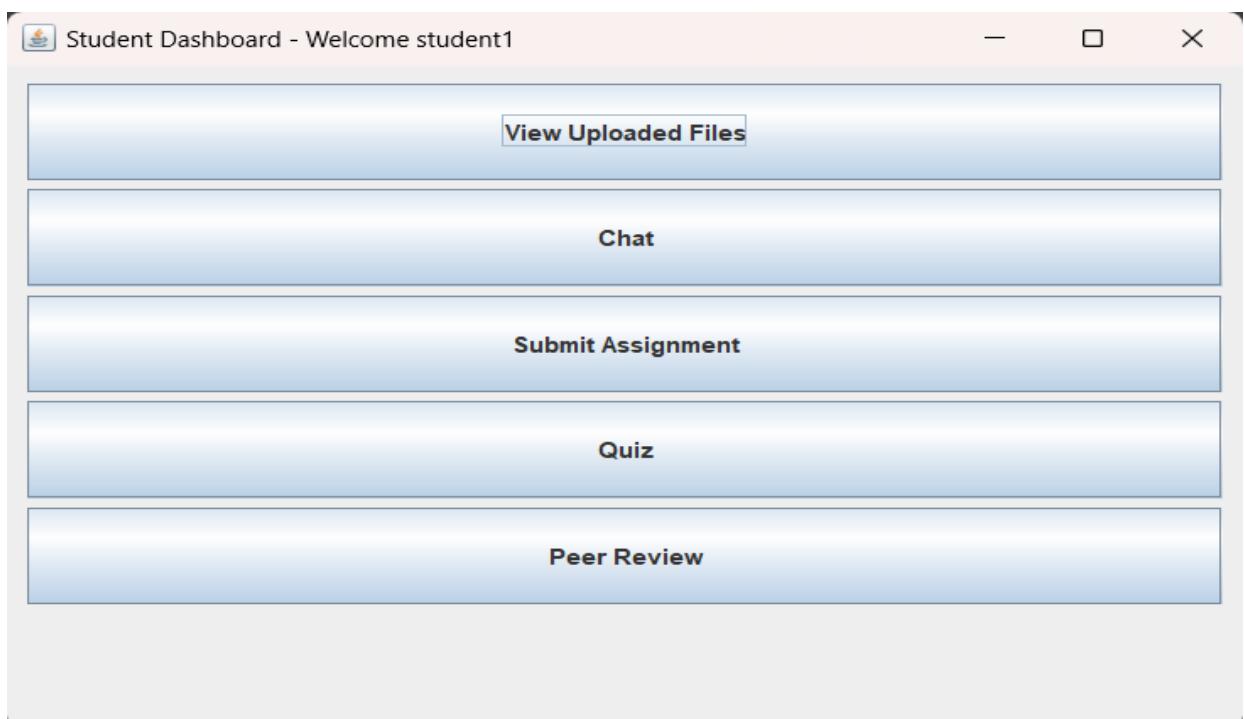


fig.(v)Student Dashboard

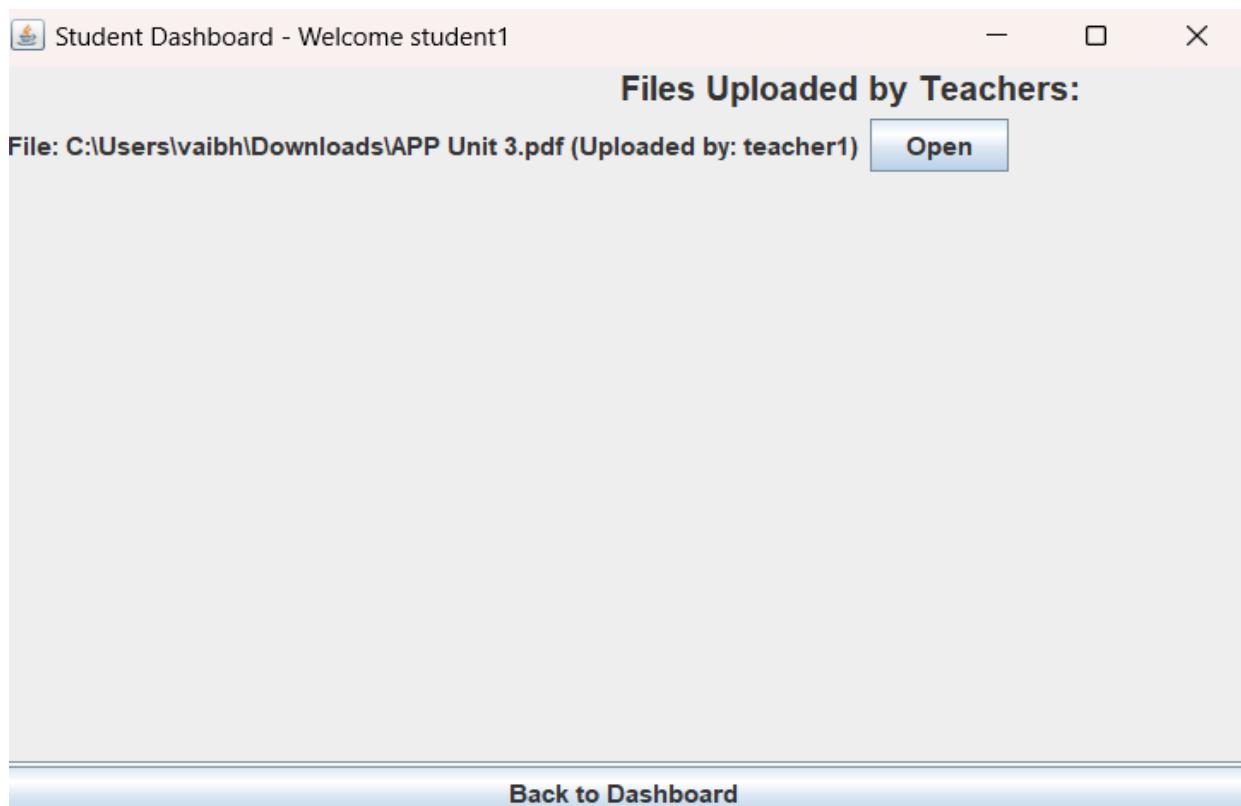


fig.(vi)View Uploaded Files

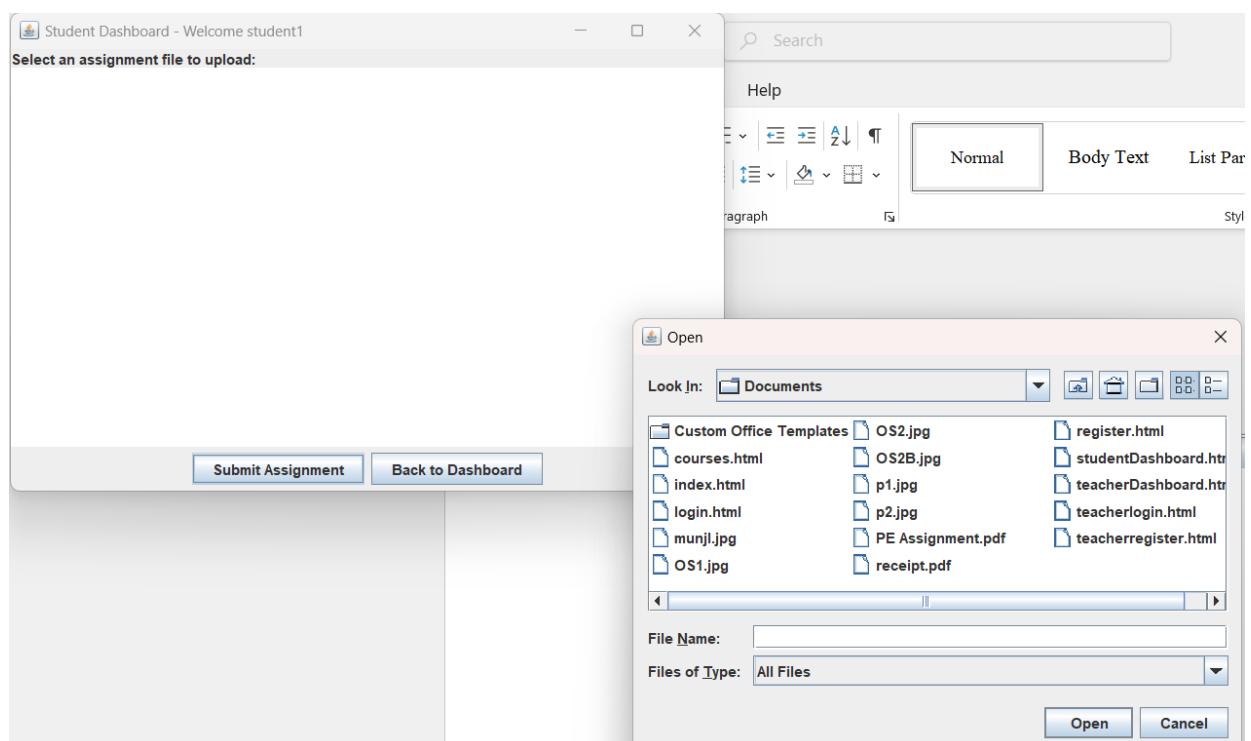
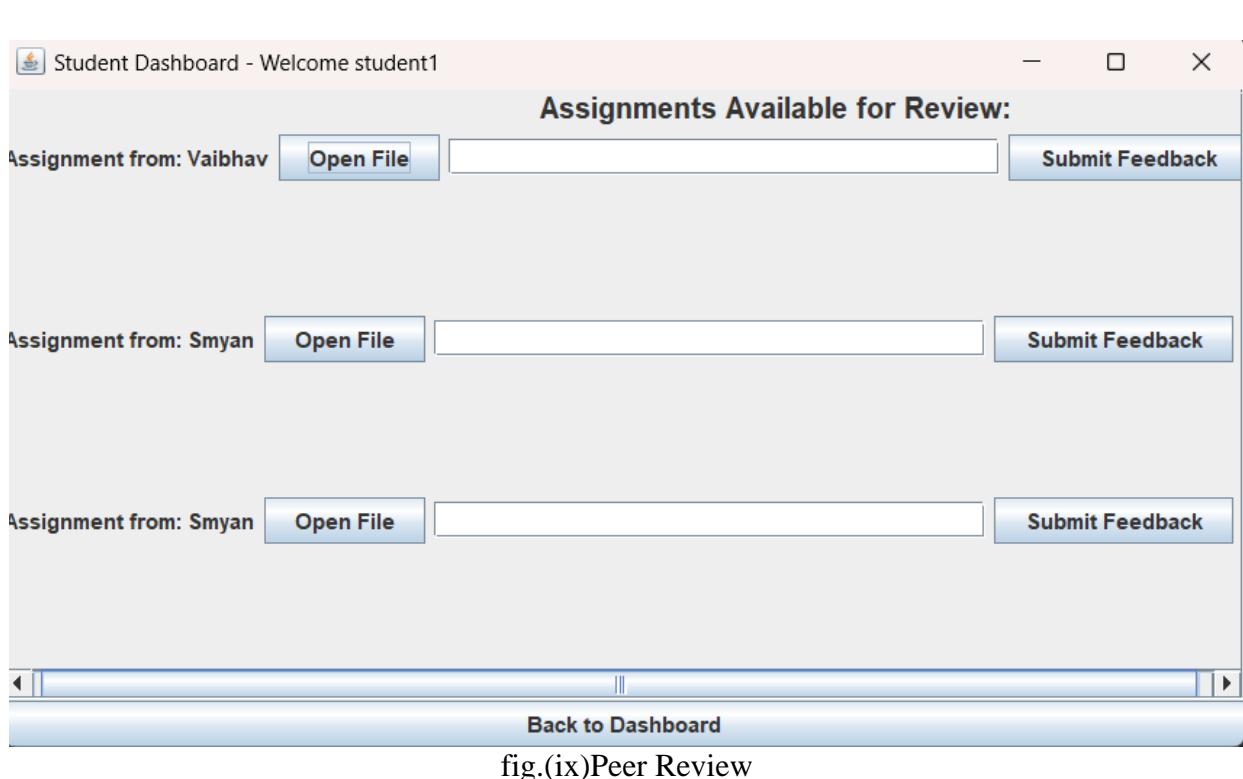
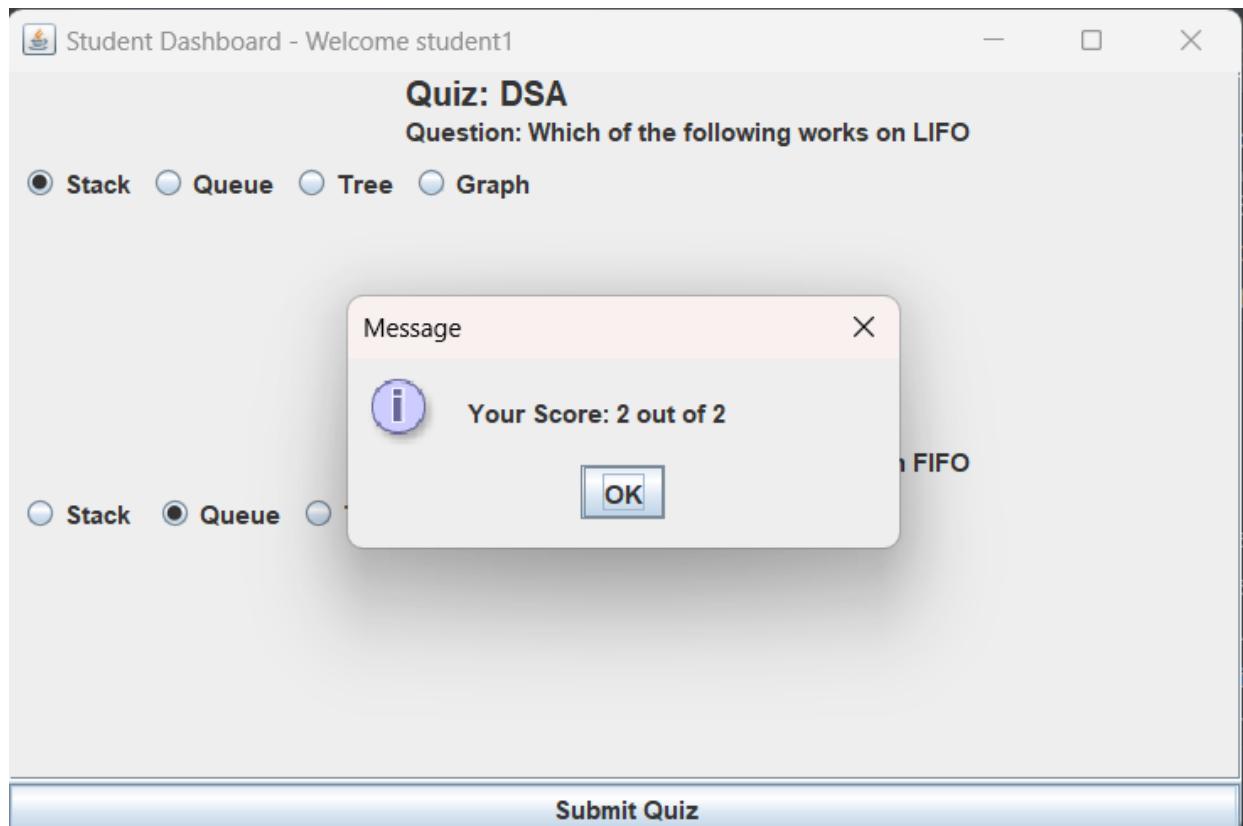


fig.(vii)Submit Assignment



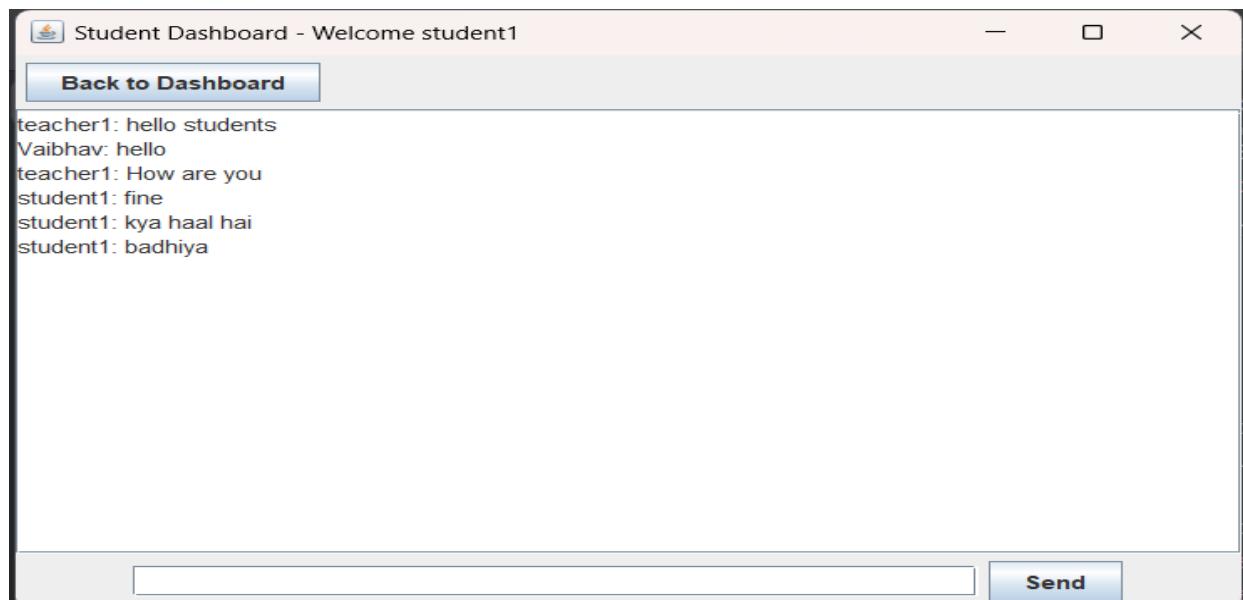


fig.(x)Chat

Teacher Dashboard:

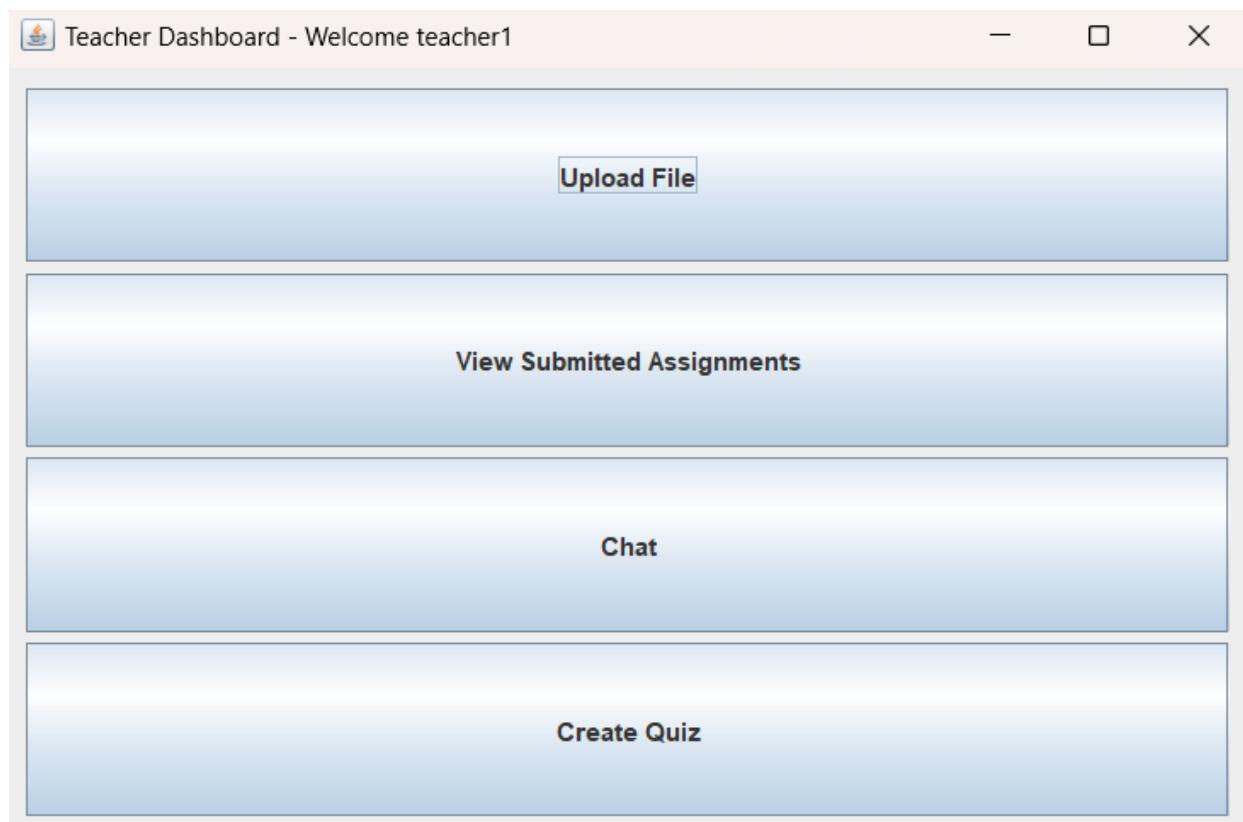


fig.(xi)Teacher's Dashboard

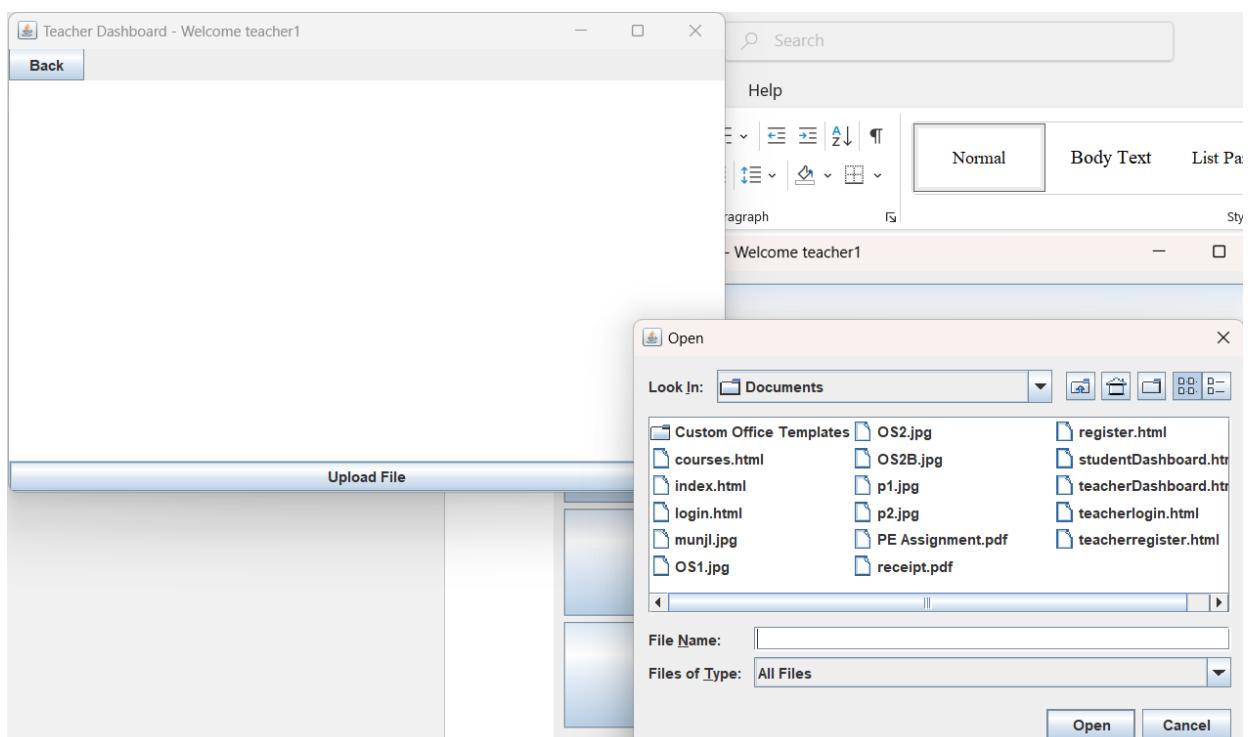


fig.(xii)Upload Files

A screenshot of a web-based teacher dashboard titled 'Teacher Dashboard - Welcome teacher1'. The dashboard displays a list of student submissions: Student: Vaibhav - File: C:\Users\vaibh\Documents\PE Assignment.pdf; Student: Smyan - File: C:\Users\vaibh\Downloads\30C.pdf; Student: Smyan - File: C:\Users\vaibh\Downloads\Re1_merged.pdf; Student: student1 - File: C:\Users\vaibh\Downloads\30.pdf; and Student: student1 - File: C:\Users\vaibh\Downloads\Codechef.pdf. At the bottom of the page is a button labeled 'Open Selected File'.

fig.(xiii)View Submitted Assignment

Teacher Dashboard - Welcome teacher1

Quiz Title:

Question:

Option A:

Option B:

Option C:

Option D:

Correct Answer (A, B, C, or ...):

Add Question **Save Quiz**

fig.(xiv)Create Quiz

Teacher Dashboard - Welcome teacher1

Back **Chat**

teacher1: hello students
Vaibhav: hello
teacher1: How are you
student1: fine
student1: kya haal hai
student1: badhiya

Send

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

fig.(xxi)Quiz Table

Questions:

	question_id	quiz_id	question_text	option_a	option_b	option_c	option_d	correct_answer
▶	15	HULL	In which data structure are elements stored in ...	Tree	Stack	Queue	Stack	Tree
	16	HULL	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

fig.(xxii)Question Table

Responses:

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