

SMART LEARN HUB



An

Object-Oriented Programming through Java Course Project Report

in partial fulfilment of the degree

Bachelor of Technology
in
Computer Science & Engineering

By

Mallikarun Marthi

2103A51173

Sai Charan Reddy Nallala

2103A51100

Adithya Bodige

2103A51551

Under the guidance of

Mr. Mahender Nagurla

Submitted to





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the **Object Oriented Programming through Java - Course Project** Report entitled “**SMART LEARN HUB**” is a record of bonafide work carried out by the students Mallikarjun Marthi, Sai Charan Nallala ,Adithya Bodige bearing Roll No(s) 2103A51173,2103A51100,2103A51551 during the academic year 2022-23 in partial fulfillment of the award of the degree of *Bachelor of Technology* in **Computer Science & Engineering** by the SR University, Warangal.

Lab In-charge

Head of the Department

CONTENTS

Abstract.....	4
Objective.....	5
Definition of elements used in the Project.....	6
Design Screens.....	7
Implementation Code.....	45
Result Screens.....	48
Conclusions.....	49
References	50

ABSTRACT :

The envisioned project, "Educational Interaction Platform with Flashcards and Q&A," is a cutting-edge Java application poised to revolutionize the way students and teachers engage in the learning process. By seamlessly integrating a set of carefully crafted classes and features, this platform aims to create an immersive and collaborative educational experience.

At its core, the project revolves around several pivotal classes. The User file, with properties including user username, password, and user type (student or teacher), ensures secure access and differentiated roles. The Subject class captures subject details such as subject name, and a reference to the respective teacher, enhancing subject-specific organization.

The StudyMaterial class empowers educators to upload comprehensive learning resources. With properties like material ID, subject reference, content, and upload date, this class serves as a repository for educational content. Equally important, the Question and Answer classes stimulate interactive learning. Questions, identified by question ID and post date, enable students to seek clarification. Correspondingly, answers, complete with answer ID, teacher reference, and content, foster effective knowledge sharing.

The application's pivotal functionalities are orchestrated by the Main Application class. It initializes essential data structures and offers functions for uploads, credit assignments, and interactive Q&A. Robust search and filtering mechanisms further enhance usability. The User Interface class, developed using Java Swing or JavaFX, provides an intuitive and aesthetically pleasing interface, ensuring a seamless user experience.

The application's pivotal functionalities are orchestrated by the Main Application class. It initializes essential data structures and offers functions for uploads, credit assignments, and interactive Q&A. Robust search and filtering mechanisms further enhance usability. The User Interface class, developed using Java Swing or JavaFX, provides an intuitive and aesthetically pleasing interface, ensuring a seamless user experience.

OBJECTIVE :

The primary objective of the "Educational Interaction Platform with Flashcards and Q&A" project is to modernize the education experience for both students and teachers. This innovative Java application seeks to foster a more immersive and interactive learning environment by seamlessly integrating technology. It does so by offering a range of carefully crafted classes and features that ensure secure access and differentiated roles for users, enhancing subject-specific organization, and empowering educators to upload comprehensive learning resources.

Additionally, the platform aims to stimulate interactive learning through its Question and Answer classes, where students can seek clarification and teachers can provide effective knowledge sharing. Robust search and filtering mechanisms further enhance usability, while an intuitive User Interface class ensures a seamless and aesthetically pleasing user experience. In essence, the project's objective is to amalgamate technology and education to create a holistic, dynamic, and engaging educational environment.

DEFINITIONS OF ELEMENTS USED IN PROJECT :

javax.swing:

Provides tools for creating graphical user interfaces (GUIs) in Java applications, including components like buttons and windows.

java.awt:

Abstract Window Toolkit for GUI component foundation in Java, often used with Swing for creating GUIs.

java.awt.event.ActionEvent and java.awt.event.ActionListener:

Handle GUI action events (e.g., button clicks) in Java applications.

java.awt.event.ComponentAdapter and java.awt.event.ComponentEvent:

Manage component-related events like resizing and repositioning in GUIs.

java.awt.image.BufferedImage:

Java class for image manipulation and processing.

java.io:

Java package for input/output operations, including file handling.

javax.imageio.ImageIO:

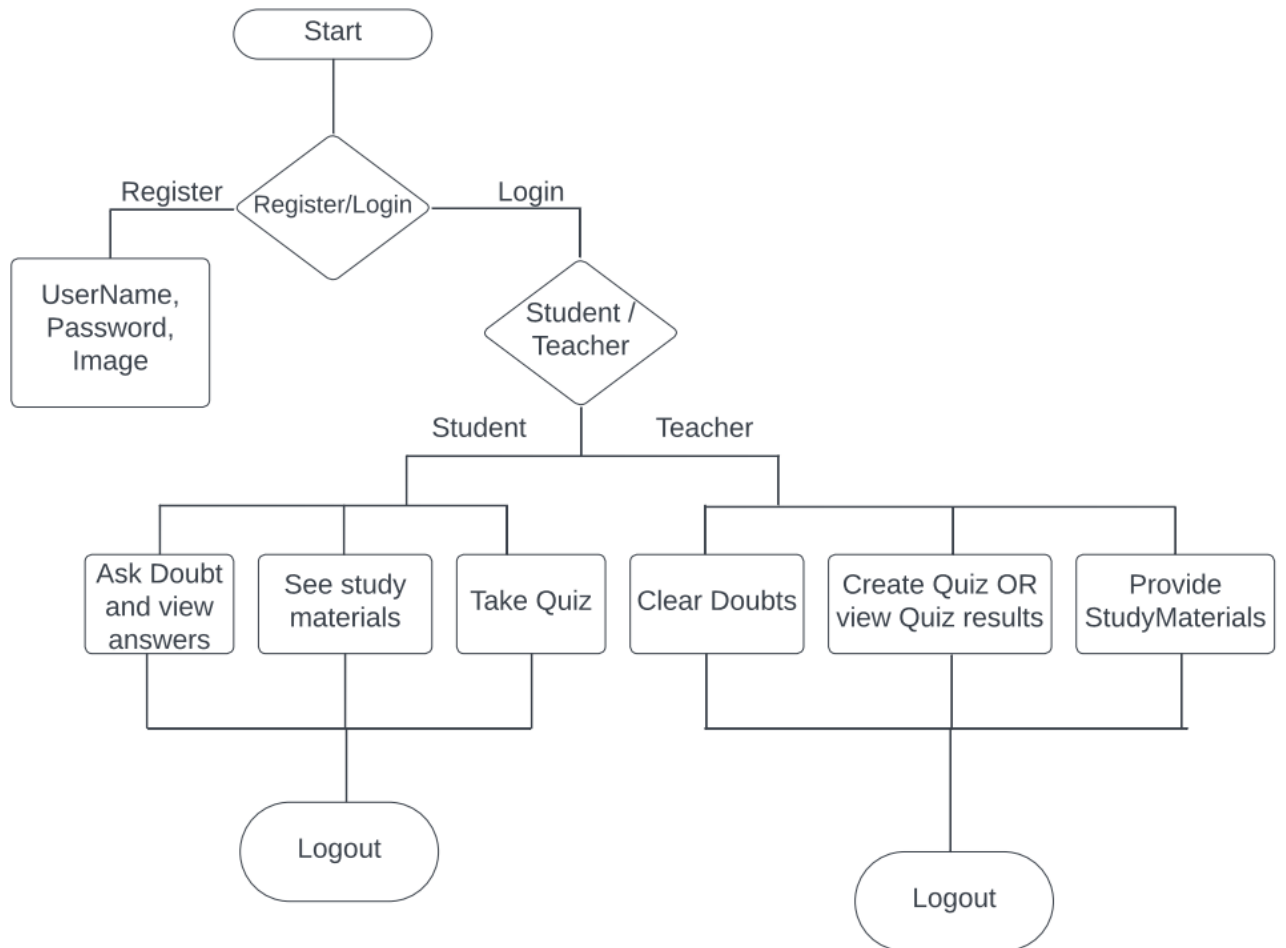
Java library for reading and writing images in various formats.

java.util.ArrayList and java.util.List:

Java classes for managing lists and collections of data.

java.util.HashMap and java.util.Map: Java classes for working with key-value pairs, offering efficient data storage and retrieval.

DESIGN:



IMPLEMENTATION:

UserRegistrationAndLoginClass:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;

public class UserRegistrationForm {
    private JFrame frame;
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JComboBox<String> userTypeComboBox;
    private JPanel registrationPanel;
    private StudyMaterial studyMaterial;

    public UserRegistrationForm() {
        frame = new JFrame("Smart LearnHub");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(512, 512);
        frame.setLayout(new BorderLayout());
        frame.setResizable(false);
        registrationPanel = createRegistrationPanel();

        frame.add(createHeaderPanel(), BorderLayout.NORTH);
        frame.add(registrationPanel, BorderLayout.CENTER);

        studyMaterial = new StudyMaterial(); // Initialize study materials

        frame.setVisible(true);
    }

    private JPanel createHeaderPanel() {
        JPanel headerPanel = new JPanel();
```



```

headerPanel.setLayout(new BorderLayout(headerPanel, BorderLayout.Y_AXIS));
JLabel headerLabel = new JLabel("Smart LearnHub");
headerLabel.setFont(new Font("Arial", Font.BOLD, 24));
headerLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
JLabel quotationLabel = new JLabel("Enriching Minds, Empowering Futures");
quotationLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

headerPanel.add(headerLabel);
headerPanel.add(quotationLabel);

return headerPanel;
}

```

```

private JPanel createRegistrationPanel() {
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(5, 2));
    panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

    JLabel usernameLabel = new JLabel("Username:");
    usernameField = new JTextField();
    usernameField.setHorizontalAlignment(JTextField.CENTER);

    JLabel passwordLabel = new JLabel("Password:");
    passwordField = new JPasswordField();
    passwordField.setHorizontalAlignment(JTextField.CENTER);

    JLabel userTypeLabel = new JLabel("User Type:");
    String[] userTypes = { "Student", "Teacher" };
    userTypeComboBox = new JComboBox<>(userTypes);

    JButton registerButton = new JButton("Register");
    registerButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

```

```

        registerUser();
    }
});

JButton loginButton = new JButton("Login");
loginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        login();
    }
});

panel.add(usernameLabel);
panel.add(usernameField);
panel.add(passwordLabel);
panel.add(passwordField);
panel.add(userTypeLabel);
panel.add(userTypeComboBox);
panel.add(registerButton);
panel.add(loginButton);

return panel;
}

private void registerUser() {
    String username = usernameField.getText();
    String password = new String(passwordField.getPassword());
    String userType = (String) userTypeComboBox.getSelectedItem();

    if (username.isEmpty() || password.isEmpty()) {
        JOptionPane.showMessageDialog(frame, "Both username and password are required.",
"Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
}

```

```

        if (isUserExists(username, userType)) {
            JOptionPane.showMessageDialog(frame, "User with the same username and user type
already exists. Please choose a different username or user type.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        try (FileWriter writer = new FileWriter("user_data.txt", true)) {
            String userEntry = username + "," + password + "," + userType + "\n";
            writer.write(userEntry);
            JOptionPane.showMessageDialog(frame, "Registration successful!", "Success",
JOptionPane.INFORMATION_MESSAGE);
            new UploadImage(username, userType);

        } catch (IOException e) {
            JOptionPane.showMessageDialog(frame, "Registration failed!", "Error",
JOptionPane.ERROR_MESSAGE);
        }

        usernameField.setText("");
        passwordField.setText("");
    }

    private void login() {
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());
        String userType = (String) userTypeComboBox.getSelectedItem();

        if (isUserValid(username, password, userType)) {
            frame.setTitle(userType + " Dashboard");
            if ("Student".equals(userType)) {
                openStudentDashboard(username, userType);
            }
        }
    }

```

```

        } else if ("Teacher".equals(userType)) {
            openTeacherDashboard(username,userType);
        }
    } else {
        JOptionPane.showMessageDialog(frame, "Login failed!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void openStudentDashboard(String username,String usertype) {
    frame.dispose();
    new StudentDashboardForm(username,usertype);
}

private void openTeacherDashboard(String username,String usertype) {
    frame.dispose();
    new TeacherDashboardForm(username,usertype);
}

private boolean isUserExists(String username, String userType) {
    try (BufferedReader reader = new BufferedReader(new FileReader("user_data.txt"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split(",");
            String storedUsername = parts[0];
            String storedUserType = parts[2];
            if (username.equals(storedUsername) && userType.equals(storedUserType)) {
                return true;
            }
        }
    } catch (IOException e) {
    }
    return false;
}

```

```

private boolean isValid(String username, String password, String userType) {
    try (BufferedReader reader = new BufferedReader(new FileReader("user_data.txt"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split(",");
            String storedUsername = parts[0];
            String storedPassword = parts[1];
            String storedUserType = parts[2];
            if (username.equals(storedUsername) && password.equals(storedPassword) &&
                userType.equals(storedUserType)) {
                return true;
            }
        }
    } catch (IOException e) {
    }
    return false;
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new UserRegistrationForm();
        }
    });
}
}

```

StudentDashboardClass:

```

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

```

```

import java.awt.event.ActionListener;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class StudentDashboardForm extends JFrame {
    private JLabel profilePhotoLabel;
    private String username;

    public StudentDashboardForm(String username, String usertype) {
        this.username = username;

        setTitle("Student Dashboard");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(512, 512);
        setResizable(false);
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(5, 2, 10, 10));

        panel.add(new JLabel("Welcome, " + username));
        profilePhotoLabel = new JLabel();
        panel.add(new JLabel("Profile Photo:"));
        panel.add(profilePhotoLabel);

        JLabel branchLabel = new JLabel("Select Branch:");
        String[] branches = {"CSE", "ECE", "MECH", "EEE"};
        JComboBox<String> branchComboBox = new JComboBox<>(branches);
        panel.add(branchLabel);
        panel.add(branchComboBox);

        JLabel subjectLabel = new JLabel("Select Subject:");

```

```

JComboBox<String> subjectComboBox = new JComboBox<>();
subjectComboBox.setEnabled(false);
panel.add(subjectLabel);
panel.add(subjectComboBox);

JButton viewMaterialsButton = new JButton("View Study Materials");
viewMaterialsButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedBranch = (String) branchComboBox.getSelectedItem();
        String selectedSubject = (String) subjectComboBox.getSelectedItem();
        String content = getStudyMaterialContent(selectedBranch, selectedSubject);
        openStudyMaterialViewer(selectedBranch, selectedSubject, content);
    }
});
panel.add(new JPanel());
panel.add(viewMaterialsButton);

JButton askQuestionButton = new JButton("Ask a Question");
askQuestionButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        openQuestionForm();
    }
});
panel.add(new JPanel());
panel.add(askQuestionButton);

JButton viewQuestionsButton = new JButton("View Questions");
viewQuestionsButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        openQuestionViewer(username);
    }
});

```

```
});
panel.add(new JPanel());
panel.add(viewQuestionsButton);
```

```
JButton viewAnswersButton = new JButton("View Answers");
viewAnswersButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        openAnswerViewer(username);
    }
});
panel.add(new JPanel());
panel.add(viewAnswersButton);
```

```
JButton takeQuizButton = new JButton("Take Quiz");
takeQuizButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedBranch = (String) branchComboBox.getSelectedItem();
        String selectedSubject = (String) subjectComboBox.getSelectedItem();
        openQuizForBranchAndSubject(selectedBranch, selectedSubject);
    }
});
panel.add(new JPanel());
panel.add(takeQuizButton);
```

```
JButton logoutButton = new JButton("Logout");
logoutButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        dispose();
        openLoginDialog();
    }
});
```



```

panel.add(new JPanel());
panel.add(loginButton);

add(panel, BorderLayout.CENTER);

branchComboBox.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        subjectComboBox.setEnabled(true);
        subjectComboBox.setModel(new
DefaultComboBoxModel<>(getSubjectsForBranch((String)
branchComboBox.getSelectedItem())));
    }
});

addComponentListener(new ComponentAdapter() {
    @Override
    public void componentResized(ComponentEvent e) {
        String sourceFilePath =
StudentDashboardForm.class.getProtectionDomain().getCodeSource().getLocation().getPath();
        new File(sourceFilePath).getParentFile();

        loadAndSetProfilePhoto(usertype + username + ".jpg");
    }
});

setVisible(true);
}

private void loadAndSetProfilePhoto(String imagePath) {
    int labelWidth = profilePhotoLabel.getWidth();
    int labelHeight = profilePhotoLabel.getHeight();
    if (labelWidth <= 0 || labelHeight <= 0) {
        return;
    }
}

```

```

    }

    try {
        BufferedImage originalImage = ImageIO.read(new File(imagePath));
        if (originalImage != null) {
            int newWidth;
            int newHeight;

            if ((double) originalImage.getWidth() / originalImage.getHeight() > (double) labelWidth
/ labelHeight) {
                newWidth = labelWidth;
                newHeight = (int) (labelWidth / (double) originalImage.getWidth() *
originalImage.getHeight());
            } else {
                newHeight = labelHeight;
                newWidth = (int) (labelHeight / (double) originalImage.getHeight() *
originalImage.getWidth());
            }

            BufferedImage resizedImage = new BufferedImage(newWidth, newHeight,
BufferedImage.TYPE_INT_ARGB);
            Graphics2D g = resizedImage.createGraphics();
            g.drawImage(originalImage, 0, 0, newWidth, newHeight, null);
            g.dispose();

            profilePhotoLabel.setIcon(new ImageIcon(resizedImage));
        } else {
            profilePhotoLabel.setText("Profile Photo Not Found");
        }
    } catch (IOException e) {
        e.printStackTrace();
        profilePhotoLabel.setText("Error Loading Profile Photo");
    }
}

```

```

private String[] getSubjectsForBranch(String branch) {
    if ("CSE".equals(branch)) {
        return new String[]{"ComputerProgramming", "DataStructures", "AlgorithmDesign"};
    } else if ("ECE".equals(branch)) {
        return new String[]{"ElectronicCircuits", "Digital Signal Processing", "Wireless
Communication"};
    } else if ("MECH".equals(branch)) {
        return new String[]{"Thermodynamics", "Mechanics of Materials", "Fluid Mechanics"};
    } else if ("EEE".equals(branch)) {
        return new String[]{"Electric Circuits", "Power Systems", "Renewable Energy"};
    } else {
        return new String[0];
    }
}

```

```

private String getStudyMaterialContent(String branch, String subject) {
    String fileName = branch + subject + ".txt";
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
        StringBuilder content = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            content.append(line).append("\n");
        }
        return content.toString();
    } catch (IOException e) {
        e.printStackTrace();
        return "Content not found";
    }
}

```

```

private void openStudyMaterialViewer(String branch, String subject, String content) {
    new StudyMaterialViewer("Branch: " + branch + "\nSubject: " + subject + "\n\n" + content);
}

```

```

private void openQuestionForm() {
    JFrame questionFrame = new JFrame("Ask a Question");
    questionFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    questionFrame.setSize(400, 250);
    questionFrame.setLayout(new BorderLayout());

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(3, 2, 10, 10));

    JLabel questionLabel = new JLabel("Your Question:");
    JTextArea questionArea = new JTextArea();
    questionArea.setLineWrap(true);
    questionArea.setWrapStyleWord(true);
    JScrollPane scrollPane = new JScrollPane(questionArea);

    JButton submitButton = new JButton("Submit Question");
    submitButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String questionText = questionArea.getText();
            if (!questionText.isEmpty()) {
                saveQuestion(username, questionText);
                JOptionPane.showMessageDialog(questionFrame, "Question submitted successfully.", "Success", JOptionPane.INFORMATION_MESSAGE);
                questionFrame.dispose();
            } else {
                JOptionPane.showMessageDialog(questionFrame, "Please enter a question.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    panel.add(questionLabel);

```

```

panel.add(scrollPane);
panel.add(new JPanel());
panel.add(new JPanel());
panel.add(submitButton);

questionFrame.add(panel, BorderLayout.CENTER);
questionFrame.setVisible(true);
}

private void saveQuestion(String studentName, String question) {
    String fileName = studentName + "_questions.txt";
    try (FileWriter writer = new FileWriter(fileName, true)) {
        writer.write(question + "\n\n");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void openQuestionViewer(String username) {
    String fileName = username + "_questions.txt";
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
        StringBuilder questionsText = new StringBuilder("Your Questions:\n");
        String line;
        while ((line = reader.readLine()) != null) {
            questionsText.append(line).append("\n\n");
        }
        new QuestionViewer("Questions by " + username, questionsText.toString());
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this, "No questions available.", "Questions",
JOptionPane.INFORMATION_MESSAGE);
    }
}

private void openAnswerViewer(String studentName) {

```

```

String fileName = username + "_answers.txt";
try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
    StringBuilder answersText = new StringBuilder("Answers for " + username + ":\n");
    String line;
    while ((line = reader.readLine()) != null) {
        answersText.append(line).append("\n");
    }
    new AnswerViewer("Answers for " + username, answersText.toString());
} catch (IOException e) {
    JOptionPane.showMessageDialog(this, "No answers available.", "Answers",
JOptionPane.INFORMATION_MESSAGE);
}
}

private void openQuizForBranchAndSubject(String branch, String subject) {
    String questionsFileName = branch + subject + "questions.txt";
    String answersFileName = branch + subject + "answers.txt";

    List<String> questions = loadQuestions(questionsFileName);
    List<String> answers = loadAnswers(answersFileName);

    if (questions.isEmpty() || questions.size() != answers.size()) {
        JOptionPane.showMessageDialog(this, "No questions available for this quiz.", "Quiz
Error", JOptionPane.INFORMATION_MESSAGE);
    } else {
        QuizPanel quizPanel = new QuizPanel(questions, answers);
        quizPanel.setQuizListener(new QuizPanelListener() {
            @Override
            public void onAnswerSubmitted(int questionIndex, String answer) {
                String correctAnswer = answers.get(questionIndex);
                if (correctAnswer.equalsIgnoreCase(answer)) {
                    JOptionPane.showMessageDialog(quizPanel, "Correct answer!", "Quiz Result",
JOptionPane.INFORMATION_MESSAGE);
                } else {

```

```

        JOptionPane.showMessageDialog(quizPanel, "Incorrect answer. The correct
answer is: " + correctAnswer, "Quiz Result", JOptionPane.ERROR_MESSAGE);
    }
}
});
setGlassPane(quizPanel);
quizPanel.setVisible(true);
}
}

```

```

private List<String> loadQuestions(String fileName) {
    List<String> questions = new ArrayList<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
        StringBuilder questionBuilder = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            if (line.isEmpty() && questionBuilder.length() > 0) {
                // An empty line indicates the end of a question
                questions.add(questionBuilder.toString());
                questionBuilder = new StringBuilder();
            } else {
                questionBuilder.append(line).append("\n");
            }
        }
        // Add the last question if it doesn't end with an empty line
        if (questionBuilder.length() > 0) {
            questions.add(questionBuilder.toString());
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return questions;
}

```

```

private List<String> loadAnswers(String fileName) {
    List<String> answers = new ArrayList<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
        String line;
        while ((line = reader.readLine()) != null) {
            answers.add(line);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return answers;
}

private void openLoginDialog() {
    new UserRegistrationForm();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new StudentDashboardForm("arjun", "Student");
        }
    });
}
}

```

TeacherDashboardClass:

```

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.image.BufferedImage;
import java.io.File;

```



```

import java.io.FileWriter;
import java.io.IOException;
import javax.imageio.ImageIO;

public class TeacherDashboardForm {
    private JFrame frame;
    private String username;
    private JLabel profilePhotoLabel;

    public TeacherDashboardForm(String username, String usertype) {
        this.username = username;
        frame = new JFrame("Teacher Dashboard - Smart Learn Hub");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(512, 512);
        frame.setLayout(new BorderLayout());

        JPanel panel = createDashboardPanel();

        frame.add(panel, BorderLayout.CENTER);

        frame.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                loadAndSetProfilePhoto(usertype + username + ".jpg");
            }
        });

        frame.setVisible(true);
    }

    private JPanel createDashboardPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(4, 1));
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        JLabel welcomeLabel = new JLabel("Welcome, " + username + "!");
        welcomeLabel.setFont(new Font("Arial", Font.BOLD, 24));
        welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);

        profilePhotoLabel = new JLabel();
        profilePhotoLabel.setHorizontalAlignment(SwingConstants.CENTER);

        JButton uploadMaterialsButton = new JButton("Upload Study Materials");
        JButton manageQuestionsButton = new JButton("Manage Q&A");
        JButton createQuizButton = new JButton("Create Quiz");
        JButton logoutButton = new JButton("Logout");

        uploadMaterialsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                openUploadMaterialsForm();
            }
        });
    }

```

```

    }
});

manageQuestionsButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        openManageQuestionsForm();
    }
});

createQuizButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        openCreateQuizForm();
    }
});

logoutButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        frame.dispose();
        new UserRegistrationForm();
    }
});

panel.add(welcomeLabel);
panel.add(profilePhotoLabel);
panel.add(uploadMaterialsButton);
panel.add(manageQuestionsButton);
panel.add(createQuizButton);
panel.add(logoutButton);

return panel;
}

private void loadAndSetProfilePhoto(String imagePath) {
    int labelWidth = profilePhotoLabel.getWidth();
    int labelHeight = profilePhotoLabel.getHeight();
    if (labelWidth <= 0 || labelHeight <= 0) {
        return;
    }

    try {
        BufferedImage originalImage = ImageIO.read(new File(imagePath));
        if (originalImage != null) {
            int newWidth;
            int newHeight;

            if ((double) originalImage.getWidth() / originalImage.getHeight() > (double) labelWidth /
labelHeight) {
                newWidth = labelWidth;

```

```

        newHeight = (int) (labelWidth / (double) originalImage.getWidth() *
originalImage.getHeight());
    } else {
        newHeight = labelHeight;
        newWidth = (int) (labelHeight / (double) originalImage.getHeight() *
originalImage.getWidth());
    }

    BufferedImage resizedImage = new BufferedImage(newWidth, newHeight,
BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = resizedImage.createGraphics();
    g.drawImage(originalImage, 0, 0, newWidth, newHeight, null);
    g.dispose();

    profilePhotoLabel.setIcon(new ImageIcon(resizedImage));
} else {
    profilePhotoLabel.setText("Profile Photo Not Found");
}
} catch (IOException e) {
    e.printStackTrace();
    profilePhotoLabel.setText("Error Loading Profile Photo");
}
}

private void openUploadMaterialsForm() {
    JFrame uploadMaterialsFrame = new JFrame("Upload Study Materials");
    uploadMaterialsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    uploadMaterialsFrame.setSize(400, 250);
    uploadMaterialsFrame.setLayout(new BorderLayout());

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(4, 1, 10, 10));

    JLabel branchLabel = new JLabel("Select Branch:");
    String[] branches = {"CSE", "ECE", "MECH", "EEE"};
    JComboBox<String> branchComboBox = new JComboBox<>(branches);

    JLabel subjectLabel = new JLabel("Select Subject:");
    JComboBox<String> subjectComboBox = new JComboBox<>();

    branchComboBox.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String selectedBranch = (String) branchComboBox.getSelectedItem();
            String[] subjects = getSubjectsForBranch(selectedBranch);
            subjectComboBox.setModel(new DefaultComboBoxModel<>(subjects));
        }
    });

    JTextArea materialsTextArea = new JTextArea();
    materialsTextArea.setWrapStyleWord(true);

```

```

materialsTextArea.setLineWrap(true);
JScrollPane scrollPane = new JScrollPane(materialsTextArea);

JButton uploadButton = new JButton("Upload Materials");
uploadButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedBranch = (String) branchComboBox.getSelectedItem();
        String selectedSubject = (String) subjectComboBox.getSelectedItem();
        String materials = materialsTextArea.getText();

        // Store the study materials in a file named after the branch and subject.
        String fileName = selectedBranch + selectedSubject + ".txt";
        try (FileWriter writer = new FileWriter(fileName)) {
            writer.write(materials);
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        JOptionPane.showMessageDialog(uploadMaterialsFrame, "Materials uploaded
successfully.", "Success", JOptionPane.INFORMATION_MESSAGE);
    }
});

panel.add(branchLabel);
panel.add(branchComboBox);
panel.add(subjectLabel);
panel.add(subjectComboBox);
panel.add(scrollPane);
panel.add(uploadButton);

uploadMaterialsFrame.add(panel, BorderLayout.CENTER);
uploadMaterialsFrame.setVisible(true);
}

private void openManageQuestionsForm() {
    new ManageQuestionsForm(username);
}

private void openCreateQuizForm() {
    JFrame createQuizFrame = new JFrame("Create Quiz");
    createQuizFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    createQuizFrame.setSize(600, 400);
    createQuizFrame.setLayout(new BorderLayout());

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(8, 1, 10, 10));

    JLabel branchLabel = new JLabel("Select Branch:");
    String[] branches = {"CSE", "ECE", "MECH", "EEE"};

```

```

JComboBox<String> branchComboBox = new JComboBox<>(branches);

JLabel subjectLabel = new JLabel("Select Subject:");
JComboBox<String> subjectComboBox = new JComboBox<>();

branchComboBox.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedBranch = (String) branchComboBox.getSelectedItem();
        String[] subjects = getSubjectsForBranch(selectedBranch);
        subjectComboBox.setModel(new DefaultComboBoxModel<>(subjects));
    }
});

JTextArea quizTextArea = new JTextArea();
quizTextArea.setBorder(BorderFactory.createTitledBorder("Quiz Questions"));
quizTextArea.setWrapStyleWord(true);
quizTextArea.setLineWrap(true);
JScrollPane quizScrollPane = new JScrollPane(quizTextArea);

JTextArea answersTextArea = new JTextArea();
answersTextArea.setBorder(BorderFactory.createTitledBorder("Quiz Answers"));
answersTextArea.setWrapStyleWord(true);
answersTextArea.setLineWrap(true);
JScrollPane answersScrollPane = new JScrollPane(answersTextArea);

JButton addQuestionButton = new JButton("Add Question");
addQuestionButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        quizTextArea.append("\n\nQuestion: ");
        answersTextArea.append("\n\nAnswer: ");
    }
});

JButton saveQuizButton = new JButton("Save Quiz");
saveQuizButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedBranch = (String) branchComboBox.getSelectedItem();
        String selectedSubject = (String) subjectComboBox.getSelectedItem();
        String quizText = quizTextArea.getText();
        String answersText = answersTextArea.getText();

        // Store the quiz questions in a file named after the branch and subject.
        String quizFileName = selectedBranch + selectedSubject + "questions.txt";
        String answersFileName = selectedBranch + selectedSubject + "answers.txt";

        try (FileWriter writer = new FileWriter(quizFileName)) {
            writer.write(quizText);
        } catch (IOException ex) {

```

```

        ;
    }

    try (FileWriter writer = new FileWriter(answersFileName)) {
        writer.write(answersText);
    } catch (IOException ex) {
    }

    JOptionPane.showMessageDialog(createQuizFrame, "Quiz questions and answers saved
successfully.", "Success", JOptionPane.INFORMATION_MESSAGE);
}
});

panel.add(branchLabel);
panel.add(branchComboBox);
panel.add(subjectLabel);
panel.add(subjectComboBox);
panel.add(quizScrollPane);
panel.add(answersScrollPane);
panel.add(addQuestionButton);
panel.add(saveQuizButton);

createQuizFrame.add(panel, BorderLayout.CENTER);
createQuizFrame.setVisible(true);
}

private String[] getSubjectsForBranch(String branch) {
    if ("CSE".equals(branch)) {
        return new String[]{"ComputerProgramming", "DataStructures", "AlgorithmDesign"};
    } else if ("ECE".equals(branch)) {
        return new String[]{"Electronic Circuits", "Digital Signal Processing", "Wireless
Communication"};
    } else if ("MECH".equals(branch)) {
        return new String[]{"Thermodynamics", "Mechanics of Materials", "Fluid Mechanics"};
    } else if ("EEE".equals(branch)) {
        return new String[]{"Electric Circuits", "Power Systems", "Renewable Energy"};
    } else {
        return new String[0];
    }
}
}

```

UploadImage Module:

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.*;
import javax.imageio.ImageIO;

```

```

public class UploadImage extends JFrame {
    private String username;
    private String usertype;
    private JPanel closePanel;
    private JButton browseButton;
    private JLabel statusLabel;

    public UploadImage(String username,String usertype) {
        this.username=username;
        this.usertype=usertype;
        setTitle("Image Uploader");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 150);
        setLayout(new BorderLayout());

        JPanel panel = new JPanel();
        browseButton = new JButton("Browse for Image");
        statusLabel = new JLabel("Status: ");

        panel.add(browseButton);
        panel.add(statusLabel);

        add(panel, BorderLayout.CENTER);

        browseButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                int returnVal = fileChooser.showOpenDialog(null);

                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    File selectedFile = fileChooser.getSelectedFile();
                    try {
                        saveImage(selectedFile);
                    } catch (IOException ex) {
                        ex.printStackTrace();
                        statusLabel.setText("Status: Error saving the image.");
                    }
                }
            }
        });

        setVisible(true);
    }

    private void saveImage(File selectedFile) throws IOException {
        final FileInputStream fileInputStream = new FileInputStream(selectedFile);
        final BufferedImage image1 = ImageIO.read(fileInputStream);
        fileInputStream.close(); // ImageIO.read does not close the input stream

        final BufferedImage convertedImage = new BufferedImage(image1.getWidth(),

```

```

image1.getHeight(), BufferedImage.TYPE_INT_RGB);
    convertedImage.createGraphics().drawImage(image1, 0, 0, Color.WHITE, null);

    final FileOutputStream fileOutputStream = new FileOutputStream(usertype+username+".jpg");
    final boolean canWrite = ImageIO.write(convertedImage, "jpg", fileOutputStream);
    fileOutputStream.close(); // ImageIO.write does not close the output stream

    if (!canWrite) {
        throw new IllegalStateException("Failed to write image.");
    }

    statusLabel.setText("Status: Image saved ");
    JButton closeButton = new JButton("Close");
    closePanel = new JPanel();
    closePanel.add(closeButton);
    add(closePanel, BorderLayout.SOUTH);

    closeButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Close the frame
            dispose();
            new UserRegistrationForm();
        }
    });
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        new UserRegistrationForm();

    });
}
}

```

Subject Selection Module:

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class SubjectSelectionForm {
    private JFrame frame;
    private JComboBox<String> subjectComboBox;
    private StudyMaterial studyMaterial;
    private String selectedBranch;

    public SubjectSelectionForm(StudyMaterial studyMaterial, String selectedBranch) {
        this.studyMaterial = studyMaterial;
        this.selectedBranch = selectedBranch;

        frame = new JFrame("Select Subject");
    }
}

```



```

        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(512, 512);
        frame.setLayout(new BorderLayout());

        JPanel panel = createSubjectSelectionPanel();

        frame.add(panel, BorderLayout.CENTER);

        frame.setVisible(true);
    }

    private JPanel createSubjectSelectionPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(2, 2));
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        JLabel subjectLabel = new JLabel("Select Subject:");
        subjectComboBox = new JComboBox<>(studyMaterial.getSubjects(selectedBranch));

        JButton selectButton = new JButton("Select");
        selectButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                frame.dispose();
                new StudyMaterialForm(studyMaterial); // Opening the Study Material frame for the
selected branch and subject.
            }
        });

        panel.add(subjectLabel);
        panel.add(subjectComboBox);
        panel.add(selectButton);

        return panel;
    }
}

```

StudyMaterialViewer Module:

```

import javax.swing.*.*;
import java.awt.*.*;

public class StudyMaterialViewer {
    private JFrame frame;

    public StudyMaterialViewer(String content) {
        frame = new JFrame("Study Materials");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(512, 512);
        JTextArea textArea = new JTextArea(content);
        textArea.setWrapStyleWord(true);
        textArea.setLineWrap(true);
    }
}

```

```

        textArea.setEditable(false);

        JScrollPane scrollPane = new JScrollPane(textArea);

        frame.add(scrollPane, BorderLayout.CENTER);

        frame.setVisible(true);
    }
}

```

StudyMaterialForm Module:

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class StudyMaterialForm {
    private JFrame frame;
    private JComboBox<String> branchComboBox;
    private JComboBox<String> subjectComboBox;
    private JTextField chapterField;
    private JTextArea contentArea;
    private StudyMaterial studyMaterial;

    public StudyMaterialForm(StudyMaterial studyMaterial) {
        this.studyMaterial = studyMaterial;

        frame = new JFrame("Study Material");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setSize(512, 512);
        frame.setLayout(new BorderLayout());

        JPanel panel = createStudyMaterialPanel();

        frame.add(panel, BorderLayout.CENTER);

        frame.setVisible(true);
    }

    private JPanel createStudyMaterialPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(7, 2));
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        JLabel branchLabel = new JLabel("Select Branch:");
        branchComboBox = new JComboBox<>(studyMaterial.getBranches());
        branchComboBox.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                updateSubjectComboBox();
            }
        });
    }
}

```

```

    });

    JLabel subjectLabel = new JLabel("Select Subject:");
    subjectComboBox = new
JComboBox<>(studyMaterial.getSubjects(branchComboBox.getSelectedItem().toString()));

    JLabel chapterLabel = new JLabel("Chapter:");
    chapterField = new JTextField();

    JLabel contentLabel = new JLabel("Content:");
    contentArea = new JTextArea();
    contentArea.setLineWrap(true);

    JButton saveButton = new JButton("Save");
    saveButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            saveStudyMaterial();
        }
    });

    panel.add(branchLabel);
    panel.add(branchComboBox);
    panel.add(subjectLabel);
    panel.add(subjectComboBox);
    panel.add(chapterLabel);
    panel.add(chapterField);
    panel.add(contentLabel);
    panel.add(new JScrollPane(contentArea));
    panel.add(saveButton);

    return panel;
}

private void updateSubjectComboBox() {
    String selectedBranch = branchComboBox.getSelectedItem().toString();
    subjectComboBox.removeAllItems();
    String[] subjects = studyMaterial.getSubjects(selectedBranch);
    for (String subject : subjects) {
        subjectComboBox.addItem(subject);
    }
}

private void saveStudyMaterial() {
    String branch = branchComboBox.getSelectedItem().toString();
    String subject = subjectComboBox.getSelectedItem().toString();
    String chapter = chapterField.getText();
    String content = contentArea.getText();

    if (!chapter.isEmpty() && !content.isEmpty()) {
        studyMaterial.addStudyMaterial(branch, subject, chapter, content);
    }
}

```

```

        clearFields();
        JOptionPane.showMessageDialog(frame, "Study material saved successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(frame, "Chapter and Content cannot be empty.", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

private void clearFields() {
    chapterField.setText("");
    contentArea.setText("");
}
}

```

StudyMaterial Module:

```

import java.util.HashMap;
import java.util.Map;

```

```

public class StudyMaterial {
    private Map<String, Map<String, String>> subjects;

    public StudyMaterial() {
        subjects = new HashMap<>();
    }

    public String[] getBranches() {
        return subjects.keySet().toArray(new String[0]);
    }

    public String[] getSubjects(String branch) {
        Map<String, String> branchSubjects = subjects.get(branch);
        if (branchSubjects != null) {
            return branchSubjects.keySet().toArray(new String[0]);
        }
        return new String[0];
    }

    public void addStudyMaterial(String branch, String subject, String chapter, String content) {
        Map<String, String> branchSubjects = subjects.get(branch);
        if (branchSubjects == null) {
            branchSubjects = new HashMap<>();
            subjects.put(branch, branchSubjects);
        }

        branchSubjects.put(subject, chapter + ": " + content);
    }

    public String getStudyMaterial(String branch, String subject) {
        Map<String, String> branchSubjects = subjects.get(branch);
        if (branchSubjects != null) {

```

```

        return branchSubjects.get(subject);
    }
    return null;
}
}

```

QuizPanel Module:

```

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class QuizPanel extends JPanel {
    private List<String> questions;
    private List<String> answers;
    private List<String> studentAnswers; // Maintain student's answers
    private int currentQuestionIndex = 0;
    private JLabel questionLabel;
    private JTextField answerField;
    private JButton submitButton;
    private QuizPanelListener quizListener;
    private String username;
    private String quizName;
    private int score = 0; // Initialize score

    public QuizPanel(String username, String quizName, List<String> questions, List<String>
answers) {
        this.username = username;
        this.quizName = quizName;
        this.questions = questions;
        this.answers = answers;
        this.studentAnswers = new ArrayList<>(); // Initialize student's answers
        initialize();
    }

    public void setQuizListener(QuizPanelListener listener) {
        this.quizListener = listener;
    }

    private void initialize() {
        setLayout(new BorderLayout());

        questionLabel = new JLabel();
        add(questionLabel, BorderLayout.NORTH);

        answerField = new JTextField(2);

```

```

add(answerField, BorderLayout.CENTER);

submitButton = new JButton("Submit Answer");
submitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        submitAnswer();
    }
});
add(submitButton, BorderLayout.SOUTH);

displayQuestion();
}

private void displayQuestion() {
    if (currentQuestionIndex < questions.size()) {
        questionLabel.setText(questions.get(currentQuestionIndex));
    } else {
        questionLabel.setText("Quiz Completed");
        answerField.setEnabled(false);
        submitButton.setEnabled(false);

        int numberOfParticipants = 1; // Initialize with the current student

        String resultFileName = quizName + ".txt";
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(resultFileName))) {
            writer.write("Number of Participants: " + numberOfParticipants + "\n");
            writer.write(username + ": " + score);
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Display the score in a dialog box
        JOptionPane.showMessageDialog(this, "Quiz Completed\nScore: " + score, "Quiz Result",
JOptionPane.INFORMATION_MESSAGE);
        new StudentDashboardForm(username, "Student");
    }
}

private void submitAnswer() {
    if (quizListener != null) {
        String answer = answerField.getText();
        studentAnswers.add(answer); // Store the student's answer
        if (answer.equals(answers.get(currentQuestionIndex))) {
            // Check if the submitted answer is correct
            score++; // Increase the score by 1 point for a correct answer
        }
        answerField.setText("");
        currentQuestionIndex++;

        displayQuestion();
    }
}

```

```

    }
}

public String getStudentAnswer(int questionIndex) {
    if (questionIndex < studentAnswers.size()) {
        return studentAnswers.get(questionIndex);
    }
    return "";
}
}

```

QuestionViewer Module:

```

import javax.swing.*;
import java.awt.*;

public class QuestionViewer extends JFrame {
    public QuestionViewer(String title, String content) {
        setTitle(title);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(512, 512);

        JTextArea textArea = new JTextArea(content);
        textArea.setEditable(false);

        JScrollPane scrollPane = new JScrollPane(textArea);

        getContentPane().add(scrollPane, BorderLayout.CENTER);

        setVisible(true);
    }
}

```

ManageQuestionForm Module:

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class ManageQuestionsForm extends JFrame {
    private JTextArea questionTextArea;
    private JTextArea answerTextArea;
    private JButton previousButton;
    private JButton nextButton;
    private JButton saveAnswerButton;
    private JComboBox<String> studentComboBox;
    private JComboBox<String> branchComboBox;

```

```

private JComboBox<String> subjectComboBox;
private List<String> students;
private List<String> questions;
private int currentStudentIndex;
private int currentQuestionIndex;
private Set<String> answeredQuestions;

public ManageQuestionsForm(String teacherUsername) {
    students = loadStudentsFromUserData("user_data.txt");
    currentStudentIndex = 0;
    currentQuestionIndex = 0;
    answeredQuestions = new HashSet<>();

    setTitle("Manage Questions");
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setSize(600, 400);
    setLayout(new BorderLayout());

    JPanel panel = createQuestionPanel();

    add(panel, BorderLayout.CENTER);

    setVisible(true);
}

private JPanel createQuestionPanel() {
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());

    JPanel studentSelectionPanel = new JPanel();
    studentSelectionPanel.setLayout(new FlowLayout());

    studentComboBox = new JComboBox<>(students.toArray(new String[0]));
    studentComboBox.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            currentStudentIndex = studentComboBox.getSelectedIndex();
            currentQuestionIndex = 0;
            loadQuestionsForSelectedStudent();
        }
    });

    studentSelectionPanel.add(new JLabel("Select Student:"));
    studentSelectionPanel.add(studentComboBox);

    JPanel branchAndSubjectSelectionPanel = new JPanel();
    branchAndSubjectSelectionPanel.setLayout(new FlowLayout());

    JLabel branchLabel = new JLabel("Select Branch:");
    String[] branches = {"CSE", "ECE", "MECH", "EEE"};
    branchComboBox = new JComboBox<>(branches);

```



```

JLabel subjectLabel = new JLabel("Select Subject:");
subjectComboBox = new JComboBox<>();

branchComboBox.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedBranch = (String) branchComboBox.getSelectedItem();
        String[] subjects = getSubjectsForBranch(selectedBranch);
        subjectComboBox.setModel(new DefaultComboBoxModel<>(subjects));
        loadQuestionsForSelectedStudent();
    }
});

subjectComboBox.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        loadQuestionsForSelectedStudent();
    }
});

branchAndSubjectSelectionPanel.add(branchLabel);
branchAndSubjectSelectionPanel.add(branchComboBox);
branchAndSubjectSelectionPanel.add(subjectLabel);
branchAndSubjectSelectionPanel.add(subjectComboBox);

JPanel questionPanel = new JPanel();
questionPanel.setLayout(new GridLayout(1, 2));

questionTextArea = new JTextArea();
questionTextArea.setWrapStyleWord(true);
questionTextArea.setLineWrap(true);
questionTextArea.setEditable(false);
JScrollPane questionScrollPane = new JScrollPane(questionTextArea);

answerTextArea = new JTextArea();
answerTextArea.setWrapStyleWord(true);
answerTextArea.setLineWrap(true);
JScrollPane answerScrollPane = new JScrollPane(answerTextArea);

questionPanel.add(questionScrollPane);
questionPanel.add(answerScrollPane);

JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new FlowLayout());

previousButton = new JButton("Previous");
nextButton = new JButton("Next");
saveAnswerButton = new JButton("Save Answer");

previousButton.addActionListener(new ActionListener() {

```

```

        @Override
        public void actionPerformed(ActionEvent e) {
            showPreviousQuestion();
        }
    });

    nextButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            showNextQuestion();
        }
    });

    saveAnswerButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            saveAnswer();
        }
    });

    buttonPanel.add(previousButton);
    buttonPanel.add(nextButton);
    buttonPanel.add(saveAnswerButton);

    panel.add(studentSelectionPanel, BorderLayout.NORTH);
    panel.add(branchAndSubjectSelectionPanel, BorderLayout.CENTER);
    panel.add(questionPanel, BorderLayout.CENTER);
    panel.add(buttonPanel, BorderLayout.SOUTH);

    return panel;
}

protected String[] getSubjectsForBranch(String selectedBranch) {
    if ("CSE".equals(selectedBranch)) {
        return new String[]{"ComputerProgramming", "DataStructures", "AlgorithmDesign"};
    } else if ("ECE".equals(selectedBranch)) {
        return new String[]{"Electronic Circuits", "Digital Signal Processing", "Wireless
Communication"};
    } else if ("MECH".equals(selectedBranch)) {
        return new String[]{"Thermodynamics", "Mechanics of Materials", "Fluid Mechanics"};
    } else if ("EEE".equals(selectedBranch)) {
        return new String[]{"Electric Circuits", "Power Systems", "Renewable Energy"};
    } else {
        return new String[0];
    }
}

private void showCurrentQuestion() {
    if (!questions.isEmpty()) {
        if (currentQuestionIndex < 0) {
            currentQuestionIndex = 0;

```

```

    } else if (currentQuestionIndex >= questions.size()) {
        currentQuestionIndex = questions.size() - 1;
    }

    // Check if the question has been answered
    String currentQuestion = questions.get(currentQuestionIndex);
    if (!answeredQuestions.contains(currentQuestion)) {
        questionTextArea.setText(currentQuestion);
        answerTextArea.setText("");
    } else {
        // Move to the next question
        showNextQuestion();
    }
} else {
    questionTextArea.setText("No questions available for this student.");
    answerTextArea.setText("");
}
}

private void showNextQuestion() {
    currentQuestionIndex++;
    showCurrentQuestion();
}

private void showPreviousQuestion() {
    currentQuestionIndex--;
    showCurrentQuestion();
}

private void saveAnswer() {
    if (currentStudentIndex < students.size() && currentQuestionIndex < questions.size()) {
        String answer = answerTextArea.getText();
        if (!answer.isEmpty()) {
            String studentName = students.get(currentStudentIndex);
            String branch = (String) branchComboBox.getSelectedItem();
            String subject = (String) subjectComboBox.getSelectedItem();
            String question = questions.get(currentQuestionIndex);

            // Check if the question has already been answered
            if (!answeredQuestions.contains(question)) {
                String answerFileName = studentName + "_answers.txt";
                try (BufferedWriter writer = new BufferedWriter(new FileWriter(answerFileName,
true))) {
                    writer.write("Question: " + question + "\n");
                    writer.write("Answer: " + answer + "\n\n");
                    answeredQuestions.add(question); // Add the question to the set of answered questions
                    JOptionPane.showMessageDialog(this, "Answer saved successfully.", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } catch (IOException e) {
                    e.printStackTrace();
                    JOptionPane.showMessageDialog(this, "Failed to save answer. Please try again.",

```

```

"Error", JOptionPane.ERROR_MESSAGE);
    }
    } else {
        JOptionPane.showMessageDialog(this, "This question has already been answered.",
"Info", JOptionPane.INFORMATION_MESSAGE);
    }
    }
    }
}
}

```

```

private List<String> loadStudentsFromUserData(String userDataFilePath) {
    List<String> studentList = new ArrayList<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(userDataFilePath))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split(",");
            if (parts.length >= 3 && "student".equalsIgnoreCase(parts[2])) {
                studentList.add(parts[0]);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return studentList;
}

```

```

private void loadQuestionsForSelectedStudent() {
    String studentName = students.get(currentStudentIndex);
    String questionsFilePath = studentName + "_questions.txt";
    questions = loadStudentQuestions(questionsFilePath);
    currentQuestionIndex = 0;
    showCurrentQuestion();
}

```

```

private List<String> loadStudentQuestions(String questionsFilePath) {
    List<String> studentQuestions = new ArrayList<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(questionsFilePath))) {
        StringBuilder questionBuilder = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            questionBuilder.append(line).append("\n");
            if (line.trim().endsWith("?") || line.trim().endsWith(".")) {
                studentQuestions.add(questionBuilder.toString());
                questionBuilder.setLength(0);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return studentQuestions;
}
}

```

AnswerViewer Module:

```
import javax.swing.*;
import java.awt.*;

public class AnswerViewer extends JFrame {
    public AnswerViewer(String title, String content) {
        setTitle(title);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(400, 400);

        JTextArea textArea = new JTextArea(content);
        textArea.setWrapStyleWord(true);
        textArea.setLineWrap(true);
        textArea.setCaretPosition(0);

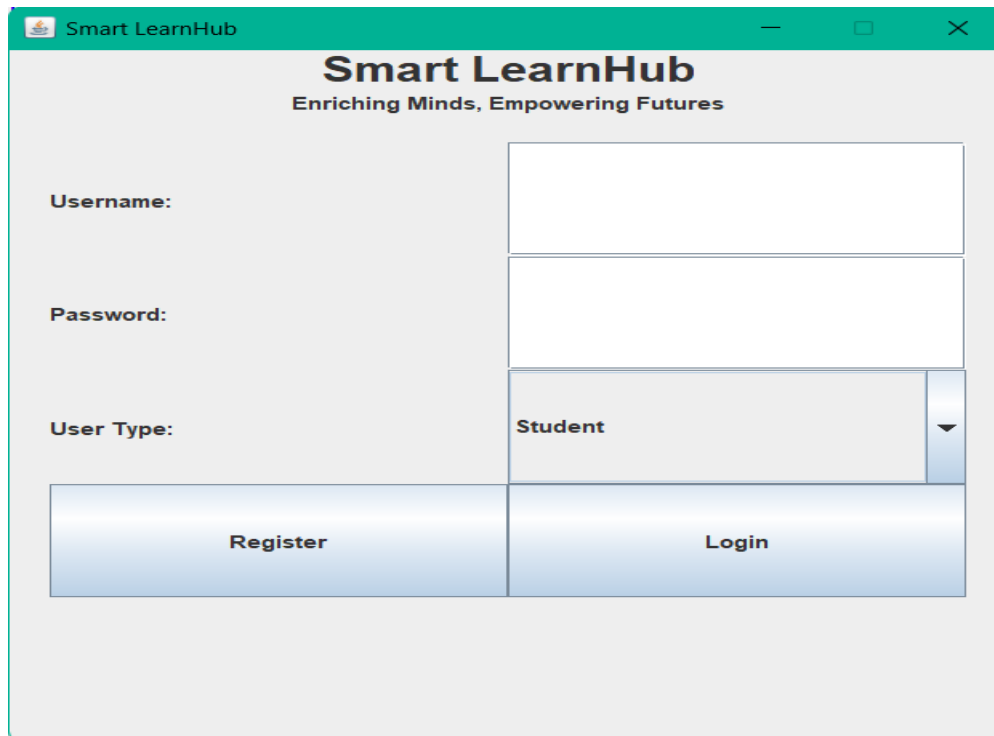
        JScrollPane scrollPane = new JScrollPane(textArea);

        add(scrollPane, BorderLayout.CENTER);

        setLocationRelativeTo(null);
        setVisible(true);
    }
}
```

Results Screens :

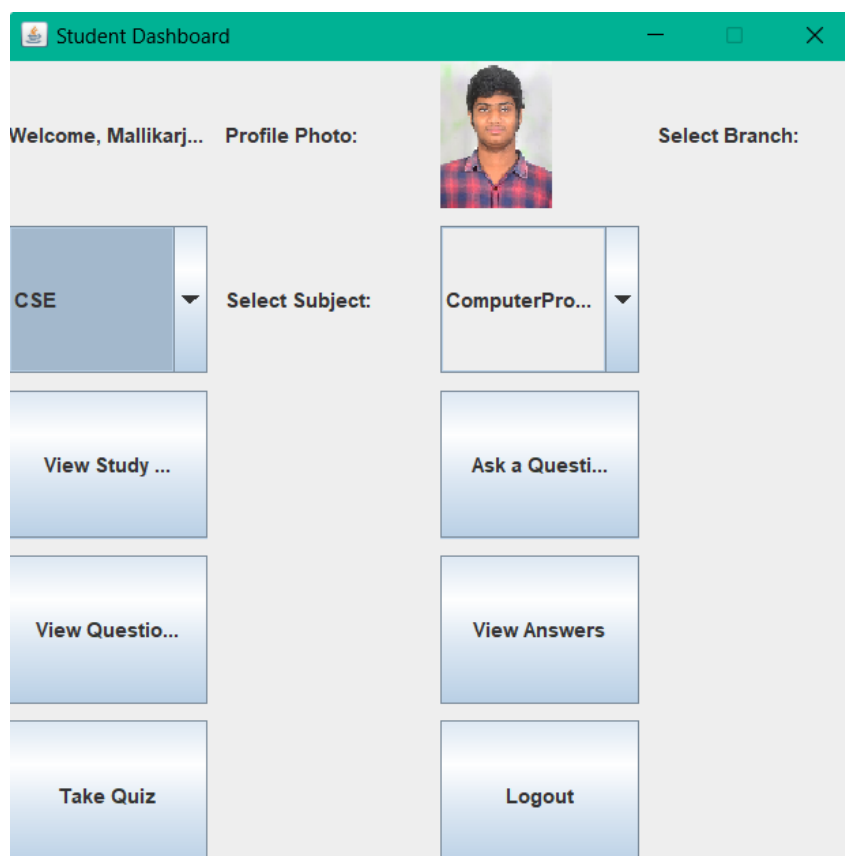
RegistrationForm:



The image shows a web application window titled "Smart LearnHub" with the tagline "Enriching Minds, Empowering Futures". The registration form contains the following elements:

- Username:** A text input field.
- Password:** A text input field.
- User Type:** A dropdown menu with "Student" selected.
- Buttons:** Two buttons labeled "Register" and "Login" at the bottom.

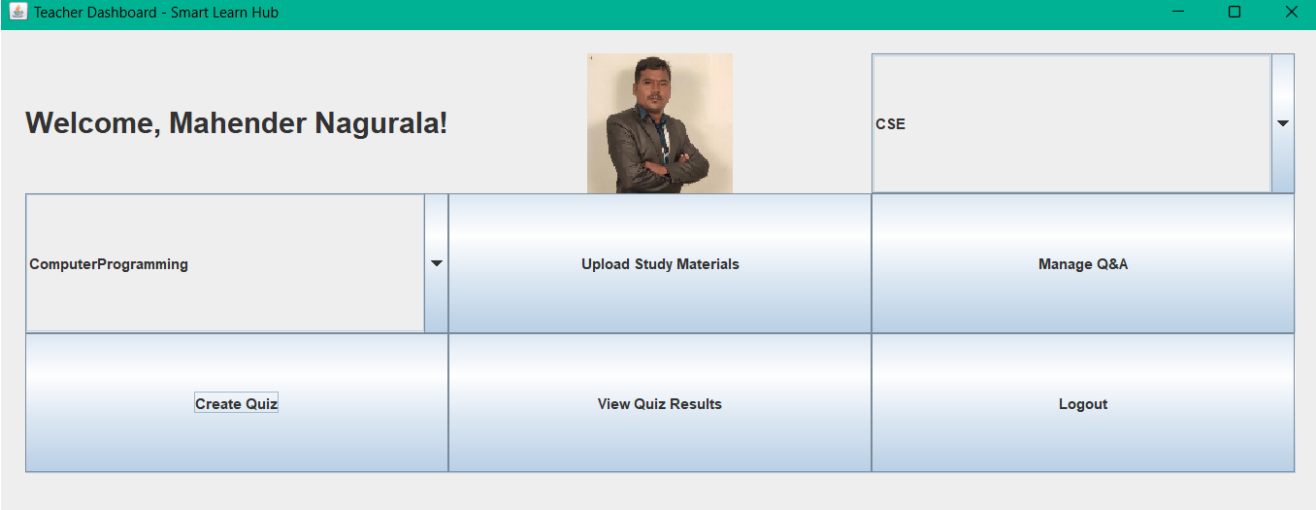
StudentDashboard module:



The image shows a web application window titled "Student Dashboard". The dashboard displays the following information and options:

- Welcome, Mallikarj...**: A greeting message.
- Profile Photo:** A placeholder image of a student.
- Select Branch:** A dropdown menu with "CSE" selected.
- Select Subject:** A dropdown menu with "ComputerPro..." selected.
- Buttons:** A grid of buttons including "View Study ...", "View Questio...", "Take Quiz", "Ask a Questi...", "View Answers", and "Logout".

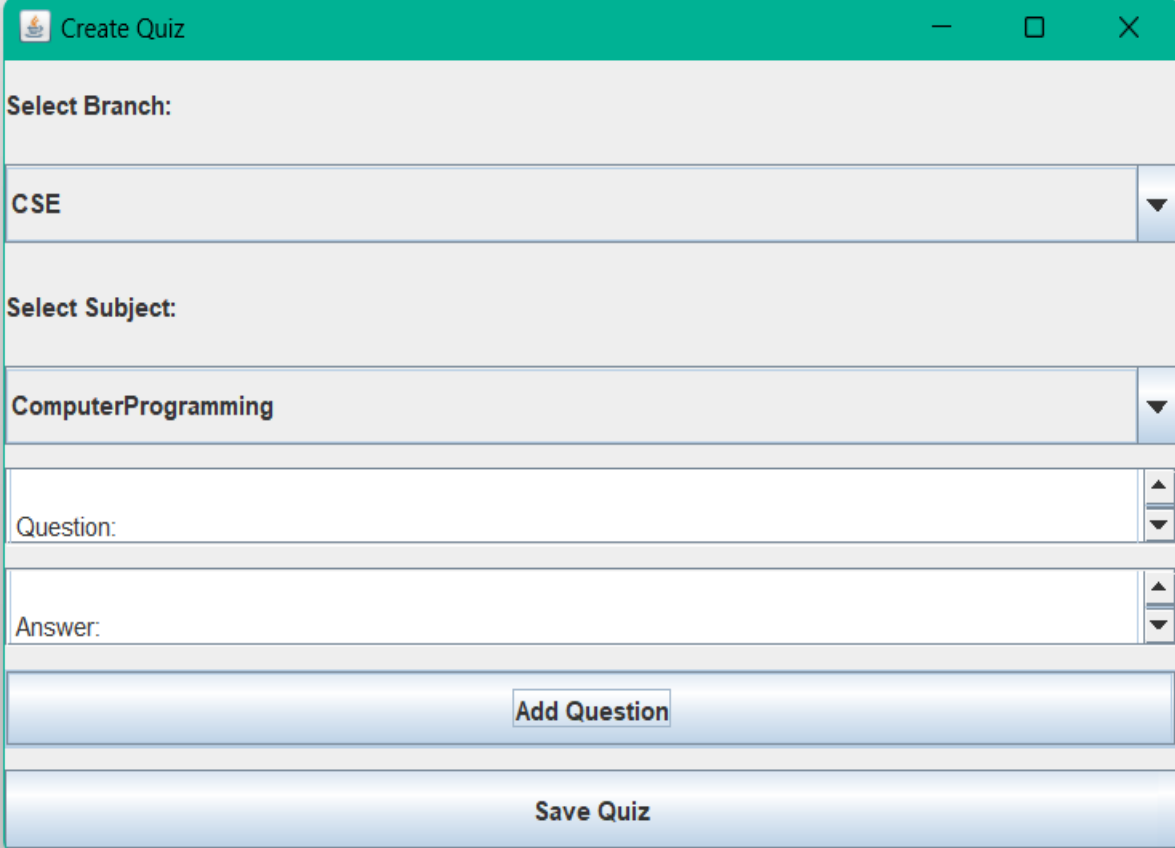
TeacherDashboard module:



The Teacher Dashboard interface features a teal header bar with the title "Teacher Dashboard - Smart Learn Hub" and standard window controls. The main content area is divided into several sections. At the top left, a welcome message "Welcome, Mahender Nagurala!" is displayed. To its right is a profile picture of a man and a dropdown menu currently showing "CSE". Below the welcome message is a grid of six buttons: "ComputerProgramming" (with a dropdown arrow), "Upload Study Materials", "Manage Q&A", "Create Quiz", "View Quiz Results", and "Logout".

Welcome, Mahender Nagurala!		CSE
ComputerProgramming	Upload Study Materials	Manage Q&A
Create Quiz	View Quiz Results	Logout


Create Quiz Module:



The Create Quiz module has a teal header bar with the title "Create Quiz" and window controls. The form contains two dropdown menus for "Select Branch:" (set to "CSE") and "Select Subject:" (set to "ComputerProgramming"). Below these are two text input fields labeled "Question:" and "Answer:", each with up and down arrow icons on the right. At the bottom, there are two large blue buttons: "Add Question" and "Save Quiz".

Select Branch:	CSE
Select Subject:	ComputerProgramming
Question:	
Answer:	
Add Question	
Save Quiz	

Doubts Clarification module:

 Manage Questions

Select Student: MallikarjunMarthi

Sir,is it possible to have multiple inheritance in java?


No,but you can use interfaces inorder to achieve that.

Previous

Next

Save Answer

StudyMaterials module:

 Study Materials

Branch: CSE
Subject: DataStructures

Data Structures in Computer Science

Data structures are fundamental concepts in computer science and are crucial for efficient data organization, storage, and retrieval. They serve as the building blocks for designing algorithms and optimizing program performance. Understanding data structures is essential for any programmer or software engineer.

Common Data Structures

Arrays: Arrays are a collection of elements, each identified by an index. They offer quick access to elements but have a fixed size.

Linked Lists: Linked lists consist of nodes, each pointing to the next node, forming a chain. They are dynamic in size but have slower random access times.

Stacks: Stacks follow the Last-In-First-Out (LIFO) principle, allowing efficient access to the last element added.

Queues: Queues follow the First-In-First-Out (FIFO) principle, making them suitable for tasks like scheduling and managing tasks in a linear order.

Trees: Trees are hierarchical structures with a root node and branches. Binary trees, AVL trees, and red-black trees are examples.

Graphs: Graphs are versatile data structures with nodes and edges, suitable for modeling complex relationships in various applications.

CONCLUSION:

The "Educational Interaction Platform with Flashcards and Q&A" is a groundbreaking Java application that promises to redefine the educational landscape. By combining innovative features and well-designed classes, this platform is poised to revolutionize the way students and teachers engage in the learning process.

The platform's core classes, including User, Subject, StudyMaterial, Question, Answer, Credit, and UserAuthentication, work together to create a secure and interactive learning environment. Users can register, log in, and access content based on their roles as students or teachers. Subjects and study materials are organized efficiently, empowering educators to share comprehensive resources. Interactive Q&A and credit systems encourage meaningful interactions, fostering collaboration and recognition among users.

The Main Application class orchestrates pivotal functionalities, including data initialization, content uploads, and filtering mechanisms, ensuring a user-friendly experience. The User Interface class, developed with Java Swing or JavaFX, offers an intuitive and visually appealing interface, further enhancing user engagement.

In conclusion, this educational platform seamlessly blends technology and education to promote collaboration, enrich learning resources, and encourage engaging interactions. It creates a dynamic and holistic environment where students and teachers can interact, learn, and grow together. By addressing the evolving needs of modern education, this project represents a significant leap forward in the realm of educational technology.

References:

1. Oracle Java Tutorials:
<https://docs.oracle.com/javase/tutorial/>
2. GeeksforGeeks:
<https://www.geeksforgeeks.org/java/>
3. JavaTpoint:
<https://www.javatpoint.com/>
4. Stack Overflow:
<https://stackoverflow.com/>