**DELIVERABLE 3 & 4**

**SOFTWARE DEVELOPMENT & CONSTRUCTION**

**SMART ACADEMIC ASSISTANCE SYSTEM**

**MEMBERS**

1. **TOUSEER AMIR (SP21312)**
2. **SHERAZ BIN TAHIR (SP21299)**
3. **MUHAMMAD ASIM (SP21277)**
4. **SAIFULLAH KHAN (SP21306)**
5. **IMRAN KHAN(SP-21317)**

**IMPLEMENTATION:**

import javax.swing.\*;

import java.awt.\*;

import java.util.ArrayList;

import java.util.List;

public class Main {

    static List<Course> courses = new ArrayList<>();

    static List<User> users = new ArrayList<>();

    static List<Instructor> instructors = new ArrayList<>();

    static List<Admin> admins = new ArrayList<>();

    static List<Feedback> feedbacks = new ArrayList<>();

    static List<Company> companies = new ArrayList<>();

    static User currentUser;

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> new MainFrame());

    }

}

class MainFrame extends JFrame {

    CardLayout cardLayout;

    JPanel mainPanel;

    JPanel studentPanel;

    JPanel instructorPanel;

    JPanel adminPanel;

    public MainFrame() {

        setTitle("University Management System");

        setSize(600, 400);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        // Adding sample data

        Admin admin = new Admin("admin", "password");

        Main.admins.add(admin);

        Instructor instructor = new Instructor("instructor", "password");

        Main.instructors.add(instructor);

        Student student = new Student("student", "password");

        Main.users.add(student);

        Main.courses.add(new Course("Java 101"));

        // Creating main panel with card layout

        cardLayout = new CardLayout();

        mainPanel = new JPanel(cardLayout);

        getContentPane().add(mainPanel);

        // Creating panels for each screen

        JPanel welcomePanel = new JPanel();

        JPanel registerPanel = new JPanel();

        JPanel loginPanel = new JPanel();

        studentPanel = new JPanel();

        instructorPanel = new JPanel();

        adminPanel = new JPanel();

        // Adding panels to main panel

        mainPanel.add(welcomePanel, "Welcome");

        mainPanel.add(registerPanel, "Register");

        mainPanel.add(loginPanel, "Login");

        mainPanel.add(studentPanel, "Student");

        mainPanel.add(instructorPanel, "Instructor");

        mainPanel.add(adminPanel, "Admin");

        // Welcome panel

        welcomePanel.setLayout(new GridLayout(3, 1));

        JButton registerButton = new JButton("Register");

        JButton loginButton = new JButton("Login");

        JButton exitButton = new JButton("Exit");

        welcomePanel.add(registerButton);

        welcomePanel.add(loginButton);

        welcomePanel.add(exitButton);

        // Register panel

        registerPanel.setLayout(new GridLayout(4, 2));

        JTextField registerUsernameField = new JTextField();

        JPasswordField registerPasswordField = new JPasswordField();

        String[] userTypes = {"Student", "Instructor", "Admin"};

        JComboBox<String> userTypeComboBox = new JComboBox<>(userTypes);

        JButton registerSubmitButton = new JButton("Register");

        registerPanel.add(new JLabel("Username:"));

        registerPanel.add(registerUsernameField);

        registerPanel.add(new JLabel("Password:"));

        registerPanel.add(registerPasswordField);

        registerPanel.add(new JLabel("User Type:"));

        registerPanel.add(userTypeComboBox);

        registerPanel.add(new JLabel());

        registerPanel.add(registerSubmitButton);

        // Login panel

        loginPanel.setLayout(new GridLayout(3, 2));

        JTextField loginUsernameField = new JTextField();

        JPasswordField loginPasswordField = new JPasswordField();

        JButton loginSubmitButton = new JButton("Login");

        loginPanel.add(new JLabel("Username:"));

        loginPanel.add(loginUsernameField);

        loginPanel.add(new JLabel("Password:"));

        loginPanel.add(loginPasswordField);

        loginPanel.add(new JLabel());

        loginPanel.add(loginSubmitButton);

        // Student panel

        studentPanel.setLayout(new GridLayout(5, 1));

        JButton enrollButton = new JButton("Enroll in Course");

        JButton accessLecturesButton = new JButton("Access Lectures");

        JButton submitAssignmentButton = new JButton("Submit Assignment");

        JButton attemptQuizButton = new JButton("Attempt Quiz");

        JButton studentLogoutButton = new JButton("Logout");

        studentPanel.add(enrollButton);

        studentPanel.add(accessLecturesButton);

        studentPanel.add(submitAssignmentButton);

        studentPanel.add(attemptQuizButton);

        studentPanel.add(studentLogoutButton);

        // Instructor panel

        instructorPanel.setLayout(new GridLayout(5, 1));

        JButton uploadLectureButton = new JButton("Upload Lecture");

        JButton uploadQuizButton = new JButton("Upload Quiz");

        JButton uploadAssignmentButton = new JButton("Upload Assignment");

        JButton viewAttendanceButton = new JButton("View Attendance");

        JButton instructorLogoutButton = new JButton("Logout");

        instructorPanel.add(uploadLectureButton);

        instructorPanel.add(uploadQuizButton);

        instructorPanel.add(uploadAssignmentButton);

        instructorPanel.add(viewAttendanceButton);

        instructorPanel.add(instructorLogoutButton);

        // Admin panel

        adminPanel.setLayout(new GridLayout(5, 1));

        JButton manageCoursesButton = new JButton("Manage Courses");

        JButton manageCompaniesButton = new JButton("Manage Companies");

        JButton viewFeedbackButton = new JButton("View Feedback");

        JButton adminProfileButton = new JButton("Profile Management");

        JButton adminLogoutButton = new JButton("Logout");

        adminPanel.add(manageCoursesButton);

        adminPanel.add(manageCompaniesButton);

        adminPanel.add(viewFeedbackButton);

        adminPanel.add(adminProfileButton);

        adminPanel.add(adminLogoutButton);

        // Button actions

        registerButton.addActionListener(e -> cardLayout.show(mainPanel, "Register"));

        loginButton.addActionListener(e -> cardLayout.show(mainPanel, "Login"));

        exitButton.addActionListener(e -> System.exit(0));

        registerSubmitButton.addActionListener(e -> {

            String username = registerUsernameField.getText();

            String password = new String(registerPasswordField.getPassword());

            String userType = (String) userTypeComboBox.getSelectedItem();

            switch (userType) {

                case "Student":

                    Main.users.add(new Student(username, password));

                    JOptionPane.showMessageDialog(this, "Student registered successfully.");

                    break;

                case "Instructor":

                    Main.instructors.add(new Instructor(username, password));

                    JOptionPane.showMessageDialog(this, "Instructor registered successfully.");

                    break;

                case "Admin":

                    Main.admins.add(new Admin(username, password));

                    JOptionPane.showMessageDialog(this, "Admin registered successfully.");

                    break;

                default:

                    JOptionPane.showMessageDialog(this, "Invalid user type.");

            }

            cardLayout.show(mainPanel, "Welcome");

        });

        loginSubmitButton.addActionListener(e -> {

            String username = loginUsernameField.getText();

            String password = new String(loginPasswordField.getPassword());

            User user = getUserByUsername(username);

            if (user != null && user.getPassword().equals(password)) {

                JOptionPane.showMessageDialog(this, "Login successful.");

                Main.currentUser = user;

                if (user instanceof Student) {

                    cardLayout.show(mainPanel, "Student");

                } else if (user instanceof Instructor) {

                    cardLayout.show(mainPanel, "Instructor");

                } else if (user instanceof Admin) {

                    cardLayout.show(mainPanel, "Admin");

                }

            } else {

                JOptionPane.showMessageDialog(this, "Invalid username or password.");

            }

        });

        // Student functionalities

        enrollButton.addActionListener(e -> {

            String[] courseTitles = Main.courses.stream().map(Course::getTitle).toArray(String[]::new);

            String selectedCourse = (String) JOptionPane.showInputDialog(this, "Select Course", "Enroll", JOptionPane.QUESTION\_MESSAGE, null, courseTitles, courseTitles[0]);

            if (selectedCourse != null) {

                Course course = Main.courses.stream().filter(c -> c.getTitle().equals(selectedCourse)).findFirst().orElse(null);

                if (course != null) {

                    course.addStudent(Main.currentUser);

                    JOptionPane.showMessageDialog(this, "Enrolled in " + selectedCourse);

                }

            }

        });

        accessLecturesButton.addActionListener(e -> extracted());

        submitAssignmentButton.addActionListener(e -> {

            String[] courseTitles = Main.courses.stream().map(Course::getTitle).toArray(String[]::new);

            String selectedCourse = (String) JOptionPane.showInputDialog(this, "Select Course", "Submit Assignment", JOptionPane.QUESTION\_MESSAGE, null, courseTitles, courseTitles[0]);

            if (selectedCourse != null) {

                String assignmentDetails = JOptionPane.showInputDialog(this, "Assignment Details:");

                if (assignmentDetails != null) {

                    Course course = Main.courses.stream().filter(c -> c.getTitle().equals(selectedCourse)).findFirst().orElse(null);

                    if (course != null) {

                        Assignment assignment = new Assignment(course, assignmentDetails);

                        course.addAssignment(assignment);

                        JOptionPane.showMessageDialog(this, "Assignment submitted.");

                    }}}

        });

        attemptQuizButton.addActionListener(e -> {

            String[] courseTitles = Main.courses.stream().map(Course::getTitle).toArray(String[]::new);

            String selectedCourse = (String) JOptionPane.showInputDialog(this, "Select Course", "Attempt Quiz", JOptionPane.QUESTION\_MESSAGE, null, courseTitles, courseTitles[0]);

            if (selectedCourse != null) {

                String quizDetails = JOptionPane.showInputDialog(this, "Quiz Details:");

                if (quizDetails != null) {

                    Course course = Main.courses.stream().filter(c -> c.getTitle().equals(selectedCourse)).findFirst().orElse(null);

                    if (course != null) {

                        Quiz quiz = new Quiz(course, quizDetails);

                        course.addQuiz(quiz);

                        JOptionPane.showMessageDialog(this, "Quiz attempted.");

                    }}}

        });

        studentLogoutButton.addActionListener(e -> {

            Main.currentUser = null;

            cardLayout.show(mainPanel, "Welcome");

        });

        // Instructor functionalities

        uploadLectureButton.addActionListener(e -> {

            String[] courseTitles = Main.courses.stream().map(Course::getTitle).toArray(String[]::new);

            String selectedCourse = (String) JOptionPane.showInputDialog(this, "Select Course", "Upload Lecture", JOptionPane.QUESTION\_MESSAGE, null, courseTitles, courseTitles[0]);

            if (selectedCourse != null) {

                String lectureDetails = JOptionPane.showInputDialog(this, "Lecture Details:");

                if (lectureDetails != null) {

                    Course course = Main.courses.stream().filter(c -> c.getTitle().equals(selectedCourse)).findFirst().orElse(null);

                    if (course != null) {

                        Lecture lecture = new Lecture(lectureDetails);

                        course.addLecture(lecture);

                        JOptionPane.showMessageDialog(this, "Lecture uploaded.");

                    }}}

        });

        uploadQuizButton.addActionListener(e -> {

            String[] courseTitles = Main.courses.stream().map(Course::getTitle).toArray(String[]::new);

            String selectedCourse = (String) JOptionPane.showInputDialog(this, "Select Course", "Upload Quiz", JOptionPane.QUESTION\_MESSAGE, null, courseTitles, courseTitles[0]);

            if (selectedCourse != null) {

                String quizDetails = JOptionPane.showInputDialog(this, "Quiz Details:");

                if (quizDetails != null) {

                    Course course = Main.courses.stream().filter(c -> c.getTitle().equals(selectedCourse)).findFirst().orElse(null);

                    if (course != null) {

                        Quiz quiz = new Quiz(course, quizDetails);

                        course.addQuiz(quiz);

                        JOptionPane.showMessageDialog(this, "Quiz uploaded.");

                    }}}

        });

        uploadAssignmentButton.addActionListener(e -> {

            String[] courseTitles = Main.courses.stream().map(Course::getTitle).toArray(String[]::new);

            String selectedCourse = (String) JOptionPane.showInputDialog(this, "Select Course", "Upload Assignment", JOptionPane.QUESTION\_MESSAGE, null, courseTitles, courseTitles[0]);

            if (selectedCourse != null) {

                String assignmentDetails = JOptionPane.showInputDialog(this, "Assignment Details:");

                if (assignmentDetails != null) {

                    Course course = Main.courses.stream().filter(c -> c.getTitle().equals(selectedCourse)).findFirst().orElse(null);

                    if (course != null) {

                        Assignment assignment = new Assignment(course, assignmentDetails);

                        course.addAssignment(assignment);

                        JOptionPane.showMessageDialog(this, "Assignment uploaded.");

                    }

                }

            }

        });

        viewAttendanceButton.addActionListener(e -> extracted2());

        instructorLogoutButton.addActionListener(e -> {

            Main.currentUser = null;

            cardLayout.show(mainPanel, "Welcome");

        });

        // Admin functionalities

        manageCoursesButton.addActionListener(e -> {

            String[] options = {"Add Course", "View Courses"};

            String selectedOption = (String) JOptionPane.showInputDialog(this, "Manage Courses", "Course Management", JOptionPane.QUESTION\_MESSAGE, null, options, options[0]);

            if (selectedOption != null) {

                switch (selectedOption) {

                    case "Add Course":

                        String courseTitle = JOptionPane.showInputDialog(this, "Course Title:");

                        if (courseTitle != null) {

                            Main.courses.add(new Course(courseTitle));

                            JOptionPane.showMessageDialog(this, "Course added.");

                        }

                        break;

                    case "View Courses":

                        StringBuilder coursesList = new StringBuilder("Courses:\n");

                        for (Course course : Main.courses) {

                            coursesList.append("- ").append(course.getTitle()).append("\n");

                        }

                        JOptionPane.showMessageDialog(this, coursesList.toString());

                        break;

                }

            }

        });

        manageCompaniesButton.addActionListener(e -> {

            String[] options = {"Add Company", "View Companies"};

            String selectedOption = (String) JOptionPane.showInputDialog(this, "Manage Companies", "Company Management", JOptionPane.QUESTION\_MESSAGE, null, options, options[0]);

            if (selectedOption != null) {

                switch (selectedOption) {

                    case "Add Company":

                        String companyName = JOptionPane.showInputDialog(this, "Company Name:");

                        if (companyName != null) {

                            Main.companies.add(new Company(companyName));

                            JOptionPane.showMessageDialog(this, "Company added.");

                        }

                        break;

                    case "View Companies":

                        StringBuilder companiesList = new StringBuilder("Companies:\n");

                        for (Company company : Main.companies) {

                            companiesList.append("- ").append(company.getName()).append("\n");

                        }

                        JOptionPane.showMessageDialog(this, companiesList.toString());

                        break;

                }

            }

        });

        viewFeedbackButton.addActionListener(e -> {

            StringBuilder feedbackList = new StringBuilder("Feedbacks:\n");

            for (Feedback feedback : Main.feedbacks) {

                feedbackList.append("- ").append(feedback.getUser().getUsername()).append(": ").append(feedback.getContent()).append("\n");

            }

            JOptionPane.showMessageDialog(this, feedbackList.toString());

        });

        adminProfileButton.addActionListener(e -> {

            // Profile management functionalities here

        });

        adminLogoutButton.addActionListener(e -> {

            Main.currentUser = null;

            cardLayout.show(mainPanel, "Welcome");

        });

        setVisible(true);

    }

    private void extracted2() {

        Instructor instructor = (Instructor) Main.currentUser;

        StringBuilder attendance = new StringBuilder("Attendance:\n");

        for (Course course : Main.courses) {

            if (course.getEnrolledStudents().contains(instructor)) {

                attendance.append(course.getTitle()).append(":\n");

                for (User student : course.getEnrolledStudents()) {

                    attendance.append("- ").append(student.getUsername()).append("\n");

                }

            }

        }

        JOptionPane.showMessageDialog(this, attendance.toString());

    }

    private void extracted() {

        Student student = (Student) Main.currentUser;

        StringBuilder lectures = new StringBuilder("Lectures:\n");

        for (Course course : Main.courses) {

            if (course.getEnrolledStudents().contains(student)) {

                lectures.append(course.getTitle()).append(":\n");

                for (Lecture lecture : course.getLectures()) {

                    lectures.append("- ").append(lecture.getDetails()).append("\n");

                }

            }

        }

        JOptionPane.showMessageDialog(this, lectures.toString());

    }

    private User getUserByUsername(String username) {

        for (User user : Main.users) {

            if (user.getUsername().equals(username)) {

                return user;

            }

        }

        for (Instructor instructor : Main.instructors) {

            if (instructor.getUsername().equals(username)) {

                return instructor;

            }

        }

        for (Admin admin : Main.admins) {

            if (admin.getUsername().equals(username)) {

                return admin;

            }}

        return null;

    }}

abstract class User {

    private String username;

    private String password;

    public User(String username, String password) {

        this.username = username;

        this.password = password;

    }

    public String getUsername() {

        return username;

    }

    public String getPassword() {

        return password;

    }}

class Student extends User {

    public Student(String username, String password) {

        super(username, password);

    }}

class Instructor extends User {

    public Instructor(String username, String password) {

        super(username, password);

    }}

class Admin extends User {

    public Admin(String username, String password) {

        super(username, password);

    }}

class Course {

    private String title;

    private List<User> enrolledStudents = new ArrayList<>();

    private List<Lecture> lectures = new ArrayList<>();

    private List<Assignment> assignments = new ArrayList<>();

    private List<Quiz> quizzes = new ArrayList<>();

    public Course(String title) {

        this.title = title;

    }

  public String getTitle() {

        return title;

    }

    public void addStudent(User student) {

        enrolledStudents.add(student);

    }

    public List<User> getEnrolledStudents() {

        return enrolledStudents;

    }

    public void addLecture(Lecture lecture) {

        lectures.add(lecture);

    }

    public List<Lecture> getLectures() {

        return lectures;

    }

    public void addAssignment(Assignment assignment) {

        assignments.add(assignment);

    }

    public void addQuiz(Quiz quiz) {

        quizzes.add(quiz);

    }}

class Assignment {

    private Course course;

    private String details;

    public Assignment(Course course, String details) {

        this.course = course;

        this.details = details;

    }}

class Quiz {

    private Course course;

    private String details;

    public Quiz(Course course, String details) {

        this.course = course;

        this.details = details;

    }}

class Lecture {

    private String details;

    public Lecture(String details) {

        this.details = details;

    }

    public String getDetails() {

        return details;

    }}

class Feedback {

    private User user;

    private String content;

    public Feedback(User user, String content) {

        this.user = user;

        this.content = content;

    }

    public User getUser() {

        return user;

    }

    public String getContent() {

        return content;

    }}

class Company {

    private String name;

    public Company(String name) {

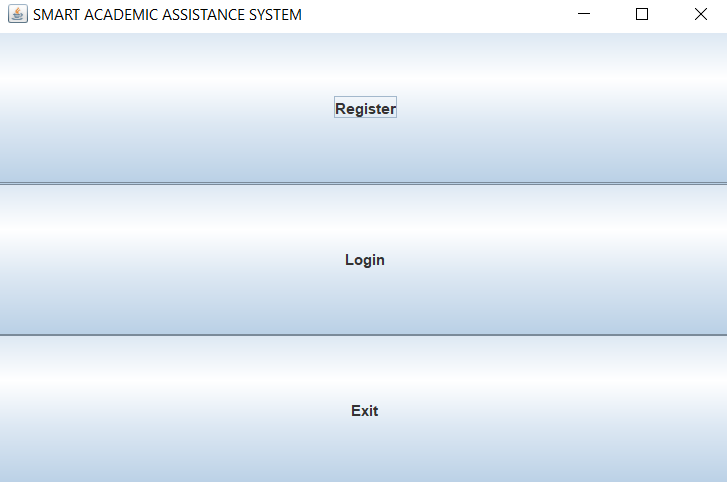
        this.name = name;

    }

    public String getName() {

        return name;

    }}



**UNIT TEST CASES:**

| **Test Case ID** | **Test Description** | **Inputs** | **Partition Tested** | **Expected Output** | **Pass/Fail** |
| --- | --- | --- | --- | --- | --- |
| TC1 | Add a valid user | Username: "touseer\_amir", Role: "User" | Valid input | User is added successfully | Pass |
| TC2 | Add a user with existing username | Username: "touseer\_amir", Role: "Admin" | Duplicate username | Exception: "User already exists" | Pass |
| TC3 | Add a user with empty username | Username: "", Role: "User" | Invalid username | Exception: "Username cannot be empty" | Pass |
| TC4 | Add a user when capacity is full | Username: "new\_user", Role: "User" | Maximum capacity | Exception: "Cannot create user. Maximum capacity reached" | Pass |
| TC5 | Add a user with special characters | Username: "imran@khan", Role: "User" | Invalid characters in username | Exception: "Username contains invalid characters" | Pass |
| TC6 | Add a user with invalid role | Username: "validuser", Role: "invalidrole" | Invalid role | Exception: "Role is not recognized" | Pass |

**GITHUB REPOSITORY & LINK:**