



Faculty of Engineering
Electrical and Electronic Department

Final Report
BER2053 JAVA PROGRAMMING
JAN-APRIL 2023

Project Name: Course Management System GUI

Name	ID
Mohammed Ahmed Hussien	1002163078
Mohamed Tarek Essam	1002163249
Abdullah Ahmed Abdullah Elhaddad	1002060687
Amtul Shaafi Hanaa	1002162952

ASSIGNMENT RUBRIC

	Unsatisfactory (0 to 4 marks)	Satisfactory (5 to 6marks)	Good (7 to 8marks)	Excellent (9 to 10 marks)	Marks Scored
Delivery PLO1	Completed less than 70% of the requirements. Not delivered on time or not in correct format	Completed between 70-80% of the requirements. Delivered on time, and in correct	Completed between 80-90% of the requirements. Delivered on time, and in correct format	Completed between 90-100% of the requirements. Delivered on time, and in correct format	
Document ation Algorithm PLO1 & PLO2	No or incomplete documentation included.	Basic documentation has been completed including descriptions of all variables. Purpose is noted for each method. Little to non- discussion and analysis is provided	Clearly documented including descriptions of all variables. Specific purpose is noted for each algorithm and data structure. Acceptable discussion and analysis is provided	Clearly and effectively documented including descriptions of all variables. Specific purpose is noted for each method, algorithm, data structure, input requirements, and output results. discussion and analysis is provided	
Coding Standards PLO2	No name, date, or experiment title included Poor use of white space (indentation, blank lines). Disorganized and messy Poor use of variables (ambiguous naming).	Includes name, date, and experiment title. White space makes program easy to read. Organized work. Good use of variables (unambiguous naming).	Includes name, date, and experiment title. Good use of white space. Organized work. Good use of variables (unambiguous naming)	Includes name, date, and experiment title. Excellent use of white space. Creatively organized work. Excellent use of variables (Unambiguous naming).	
Total					

Table of Contents

1. Abstract:.....	4
2. Introduction:	5
Problem Statement:.....	5
Objective:	5
Features for Student portal:	5
Features for Professor portal:.....	6
Scope of Project:	6
3. Methodology:.....	7
3.1. UML:.....	7
3.2. DATA FLOW DIAGRAM:.....	10
4. Comparison:	11
5. Result and discussion:.....	12
Code Explanation:	12
- The main class: CourseManagementSystem	12
- The student portal: LoginPortal	13
- ActionPerformed:.....	14
- ProfessorPortal:	17
6. Difficulties:	32
7. Future Improvements:	32
8. Conclusion:.....	32

1. Abstract:

A course management system is a software application that helps institutes to handle course interactions, especially beneficial for online learning. Managing a large number of students studying various courses could prove to be a hectic task for institutions. Reaching out to every lecturer for syllabus and assignments would seem extremely inefficient manually. However, a systematic method to implement the management of such tasks can resolve the issue. CMS helps create a space for students to ease submissions, review grades, manage courses and enable effective communication between professors and students. The system can also be integrated with student management software and learning management software to channel the required information by various departments. In this project we aim to produce a graphic user interface for professors and students to manage and handle the various tasks involved in CMS by implementing exceptions using Java, ADTs and classes from the collection API. We utilize concepts of inheritance polymorphism to help to let the child classes acquire properties of the parent class if the attributes are similar. To avoid system crashing, exceptions can catch the errors occurring in programs. Thus, by creating a user-friendly interface and portals for both parties we solve the issues faced by the educational institutions.

2. Introduction:

As university students, we all need a system that connect us with the teacher and the university institution. In UCSI university, we have the CN and the IIS. What if we combine them and have one system? So, we started brainstorming and connecting the ideas together to make the dream come true. And we got:

- Course Management System GUI

This project is done under the BER2053 Java Programming Assignment. We aim to design a GUI for a course management system that integrates with a MySQL database. The system would allow students (user) to view their courses, course schedules, grades, and other academic information. The system would have an option for the professors (other user) to edit the courses, add grades, add assignments, and other course managing activities.

Problem Statement:

In universities, managing courses and related activities can be a complex and time-consuming process. Course schedules, course offerings, student registrations, course assignments, grading, and other activities will turn chaotic without a proper course management system in place, universities may face difficulties in managing and coordinating these activities efficiently.

With the increasing use of technology in education, a digital platform that can integrate with existing university systems and provide students and faculty with access to course information, assignments, and grades is necessary. To address these challenges and streamline their course management processes, we need a reliable CMS by developing a user-friendly interface which also acts as a secure platform to exchange data.

Objective:

- The primary objective of the CMS would be to generate a convenient and efficient way for student and staff to manage the activities
- The system should allow students to view course schedules, check attendance, track their performance and progress
- The system should let lecturers to manage schedules grades assignments and reports
- It should be secure scalable easy to use graphic interface
- The system should include the following features:

Features for Student portal:

- Login (user authentication)
- A dashboard that contains the following:
 - Schedule
 - Courses taken (posts from the professor under it)
 - Grades

- Notifications bar on the side (time sensitive notifications)
- A tab for course add/drop
- A tab for assignments due and submissions
- Button for logout

Features for Professor portal:

- Login (user authentication)
- Dashboard that contains:
 - Schedule
 - Courses teaching
 - Meetings
 - Notifications bar on the side to remind about meetings
- When we open courses teaching, we find:
 - Attendance
 - A tab where the teacher adds assignments, submission links, grades, and announcements (posts)
- Button for logout

Scope of Project:

The system could be designed to integrate with technologies such as learning management systems, student information systems, and academic advising systems. As the system is implemented using Java and a variety of other technologies such as databases, web servers, and user interface frameworks, it proves a feasible option for many institutes to upgrade their management system.

3. Methodology:

3.1. UML:

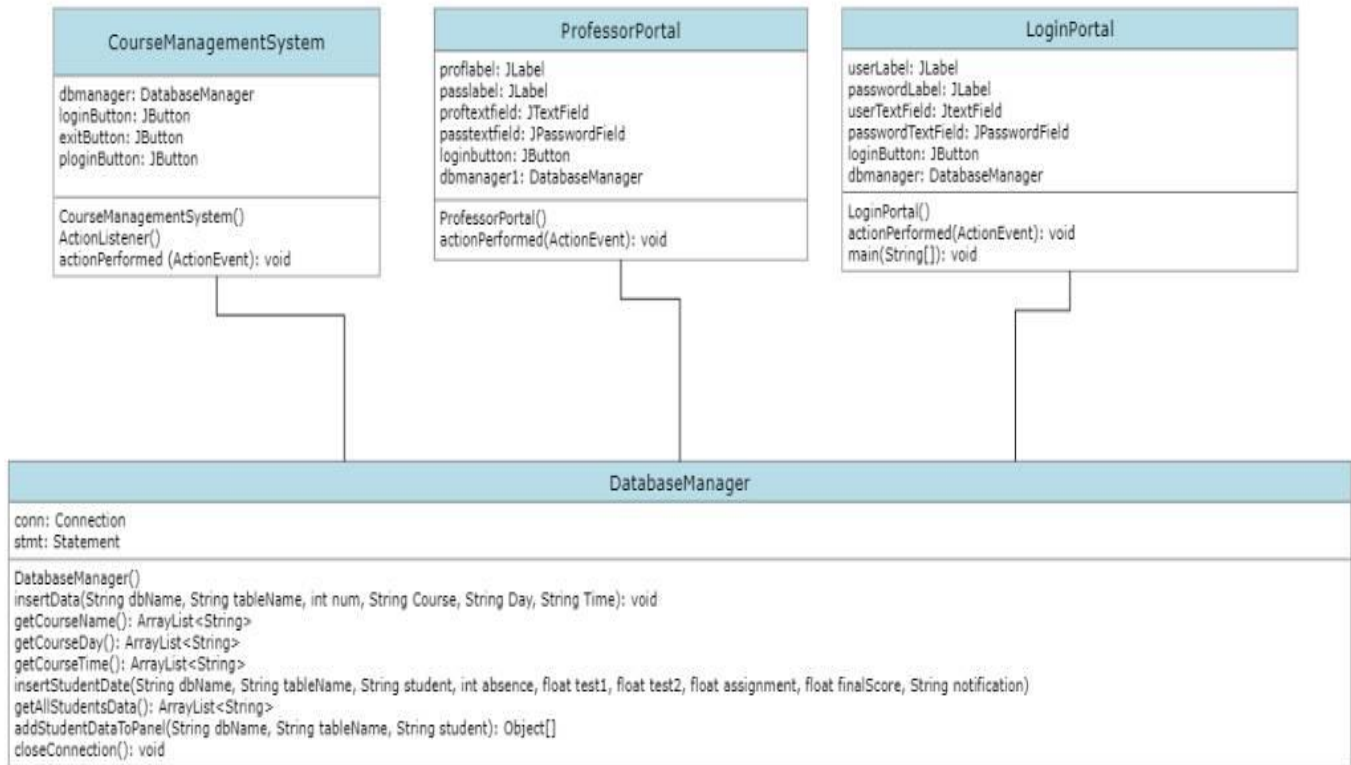


Fig.1.1.UML DIAGRAM

There are 4 classes in the course management system, a brief description of them is as follows:

The **CourseManagementSystem** class is the main class that contains the main method, it extends to the J Frame class which provides a window for the user interface. It is also connected to the database manager and has three buttons in its attributes for login, exit and plugin. It includes three methods which are:

- CourseManagementSystem which contains the main method for the whole program.
- ActionListener to initiate the events when the buttons are clicked
- actionPerformed which is void and has an inner variable ActionEvent.

The second class is the **professor portal** is also to the Database Manager and has the following labels and textfields, buttons:

- profflabel
- passlabel

- proftextfield
- passtextfield
- loginbutton
- dbmanager

The two methods are

- ProfessorPortal()
- actionPerformed () which is void.

The third class is the **login portal** which has three methods login(), actionPerformed() and main() and the following attributes:

- userlabel
- passwordlabel
- usertextfield
- passwordtextfield
- loginbutton
- dbmanager

The parent class which helps connect to the SQL database is DatabaseManager.

It has a SQL connector and the methods are as follows:

insertdata

getcoursename

getcourseday

getcoursetime

insertStudentDate

getAllstudentdata

addstudentdatatopanel

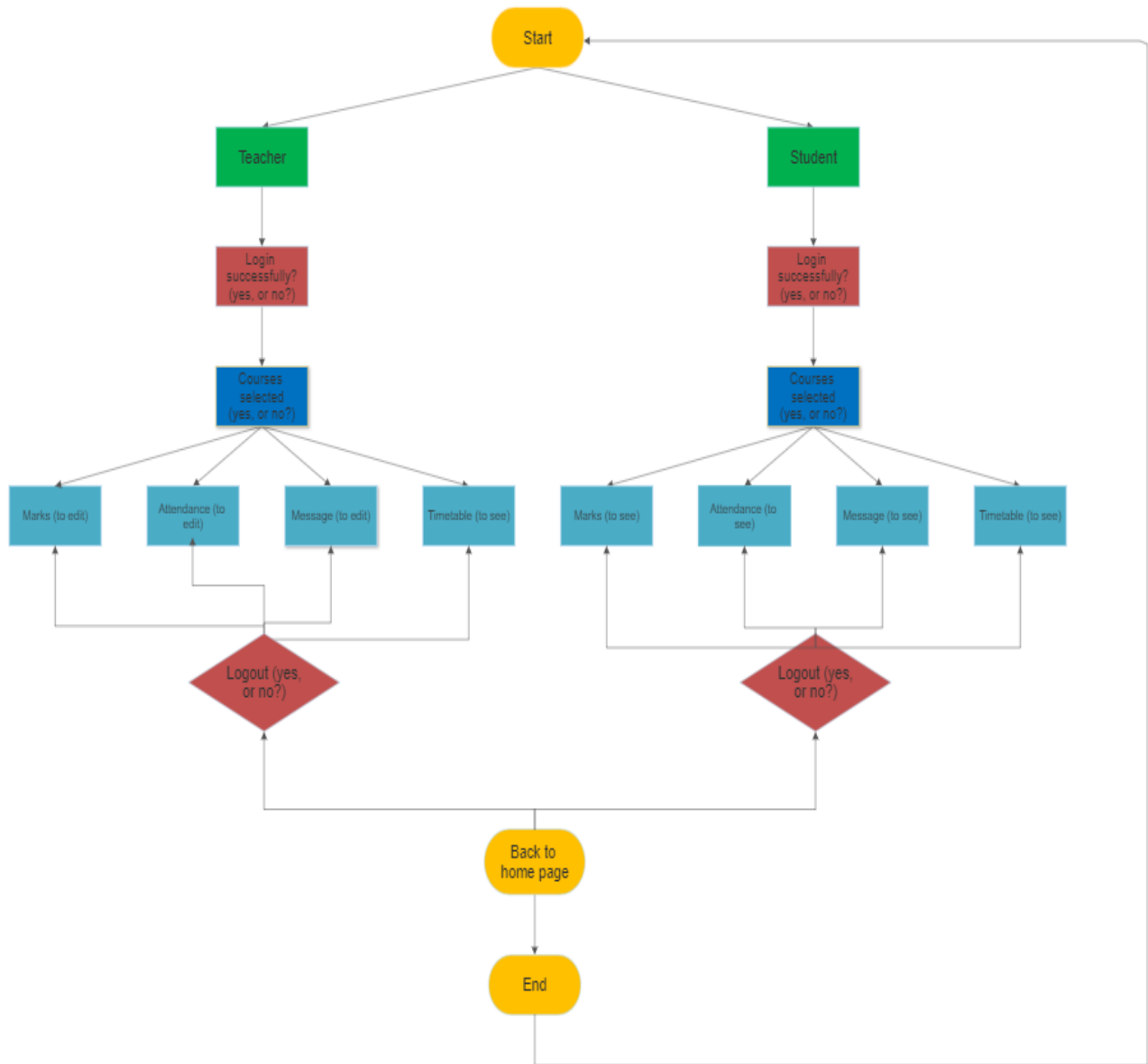
closeconnection

Th GUI uses the swing library from JAVA to use components like JFrame, JPanel, Jtextfield, Jlist, JLabel and Jbuttons to make it interactive with the user.

- JFrame: JFrame is a top-level container class that represents the main window of a Java GUI application. It provides a framework for adding and arranging other GUI components such as buttons, text fields, labels, etc. on the application window. The JFrame class provides several methods to set the window's size, title, icon, layout, and visibility.

- **JPanel:** JPanel is a container class that provides a lightweight and flexible way to group and organize other components. It can be used to hold and arrange other components like buttons, text fields, and labels within a JFrame or another JPanel. JPanel is also used as a container to hold and manage other containers within a GUI application.
- **TextField:** TextField is a component used for entering and displaying one line of text. It is used to capture user input from the keyboard or to display a value. The TextField class provides methods to set and get the text, font, and size of the text field.
- **JList:** JList is a component used to display a list of items from which the user can make a selection. It provides a way to display a set of items that can be selected by the user. The JList class provides methods to add and remove items, set the selection mode, and get the selected items.
- **JLabel:** JLabel is a non-editable text component used to display a short string or an image. It provides a way to display a message or an image to the user. The JLabel class provides methods
- **JButton:** JButton is a component used to add a clickable button to a Java GUI application. It provides a way to trigger an action in the application, such as submitting a form or navigating to another screen. The JButton class provides methods to set and get the text, font, and size of the button.

3.2. DATA FLOW DIAGRAM:



The system contains two actors specifically: Professor and student.

The login portal validates their id and password and directs them to the respective pages

If not, it shows a dialog box for incorrect password.

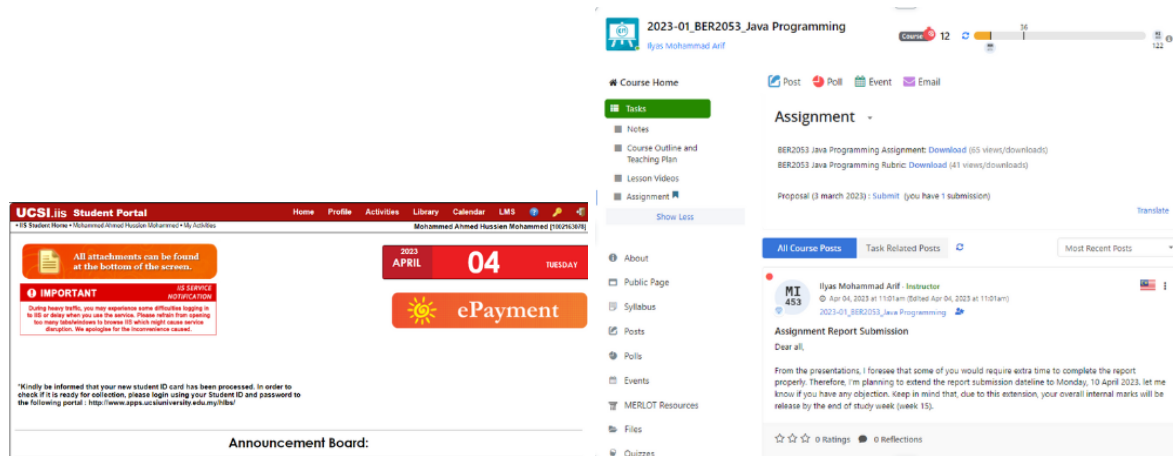
If the login is successful, they can interact with their pages where the professor can key in marks, edit attendance, send notifications as well as the timetable

The student is able to access the content once it has been updated in the database internally

The last function is logout which helps the user go back to the home page as in the login portal.

4. Comparison:

While both the portals perform efficiently, the course networking has a GUI similar to any social media page, having functionalities like post, tasks, course material access. It helps connect in the university more as a community. So, it has more actors in the system compared to the IIS, which would only allow professor and student interaction while CN allows students to communicate as well. It lacks access to content such as attendance, marks and other information.



IIS has functionalities that makes it more private it has access to marks, attendance, schedule and is based on each student's courses individually.

Our Course Management System allows the functionality like notifications for individual assignments and tests. This is the main function derived from CN and integrated with IIS in our customized system.

5. Result and discussion:

Code Explanation:

- The main class: CourseManagementSystem

```

public CourseManagementSystem() {
    dbmanager = new DatabaseManager();

    // inserting course name, day, and time to the table in mysql
    dbmanager.insertData("Courses_db", "tabel", 1, "Operation Management", "Monday", "8 - 9:30");
    dbmanager.insertData("Courses_db", "tabel", 2, "Statistics", "Tuesday", "11 - 12:30");
    dbmanager.insertData("Courses_db", "tabel", 3, "Java Programming", "Wednesday", "2 - 3:30");
    dbmanager.insertData("Courses_db", "tabel", 4, "Software Engineering", "Thursday", "5 - 6:30");
    dbmanager.insertData("Courses_db", "tabel", 5, "Object Oriented Programming", "Friday", "11 - 12:30");
    dbmanager.insertData("Courses_db", "tabel", 6, "Bahasa Melayu Komunikasi 3", "Monday", "9:30 - 12:30");
    dbmanager.insertData("Courses_db", "tabel", 7, "Circuit Theory", "Tuesday", "11 - 12:30");
    dbmanager.insertData("Courses_db", "tabel", 8, "Electromagnetic Theory", "Wednesday", "8 - 9:30");
    dbmanager.insertData("Courses_db", "tabel", 9, "Analogue Electronics", "Thursday", "5 - 6:30");
    dbmanager.insertData("Courses_db", "tabel", 10, "Digital Electronics", "Friday", "4 - 5:30");
    dbmanager.insertData("Courses_db", "tabel", 11, "Material Engineering", "Monday", "2 - 3:30");
    dbmanager.insertData("Courses_db", "tabel", 12, "Control Systems", "Tuesday", "8 - 9:30");

    setTitle("Welcome to UCSI Student Portal");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 150);
    setLocationRelativeTo(null);

    // student portal button
    loginButton = new JButton("Student Portal");
    loginButton.addActionListener(this);

    // professor portal login button
    ploginButton = new JButton("Professsor Portal");
    ploginButton.addActionListener(this);

    // exit button
    exitButton = new JButton("Exit");
    exitButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });

    JPanel panel = new JPanel(new GridLayout(3, 1));
    panel.add(loginButton);
    panel.add(ploginButton);
    panel.add(exitButton);

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

// using one function to detect which button was clicked and open the portal of the clicked button
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == loginButton) {
        LoginPortal loginportal = new LoginPortal();
    } else if (e.getSource() == ploginButton) {
        ProfessorPortal professorportal = new ProfessorPortal();
    }
    dispose();
}

```

In this constructor, we add the lectures using the insertData function. We are also setting the frame properties like title and size etc. We also add buttons such as login buttons and plogin button, and lastly the exit button. We add action listeners to all the buttons. Then we create a new panel and add all the components.

- The student portal: LoginPortal

```
public class LoginPortal extends JFrame implements ActionListener {

    private JLabel userLabel;
    private JLabel passwordLabel;
    private JTextField userTextField;
    private JPasswordField passwordTextField;
    private JButton loginButton;
    private DatabaseManager dbmanager;

    public LoginPortal() {
        setTitle("Login Portal");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 150);
        setLocationRelativeTo(null);

        userLabel = new JLabel("Username:");
        passwordLabel = new JLabel("Password:");

        userTextField = new JTextField(10);
        passwordTextField = new JPasswordField(10);

        loginButton = new JButton("Login");
        loginButton.addActionListener(this);

        JPanel panel = new JPanel(new GridLayout(3, 2));
        panel.add(userLabel);
        panel.add(userTextField);
        panel.add(passwordLabel);
        panel.add(passwordTextField);
        panel.add(new JLabel(""));
        panel.add(loginButton);

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

This class is made for the login portal. Here, we define the variables such as the username label and password labels. Also, text fields and buttons. Then in the constructor, we set the frame properties. Then we create a panel, and we add the components and we set the frame visible.

- ActionPerformed:

This is the biggest class of the code where we mainly create the student portal.

```
public void actionPerformed(ActionEvent e) {
    String username = userTextField.getText();
    String password = new String(passwordTextField.getPassword());

    // checks whether the right username and password were entered
    if (username.equals("Abdullah El-Haddad") && password.equals("Haddad-2002")) {

        dbmanager = new DatabaseManager();

        ArrayList<String> courselist = new ArrayList<String>();
        courselist = dbmanager.getCourseName();

        ArrayList<String> coursesTakenlist = new ArrayList<String>(courselist.subList(6, 12)); // storing 6 courses in t
        ArrayList<String> currentCourseslist = new ArrayList<String>(courselist.subList(0, 6)); // and 6 courses in the

        // converting to string to append the text areas
        String coursesTaken = String.join("\n", coursesTakenlist);
        String currentCourses = String.join("\n", currentCourseslist);

        ArrayList<String> coursedaylist = new ArrayList<String>();
        coursedaylist = dbmanager.getCourseDay();

        ArrayList<String> coursetimelist = new ArrayList<String>();
        coursetimelist = dbmanager.getCourseTime();

        JFrame mainFrame = new JFrame("Course Management System(Student)");
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setSize(1100, 650);
        mainFrame.setLocationRelativeTo(null);
        mainFrame.setResizable(false);

        JLabel coursesTakenLabel = new JLabel("Courses Taken:");
        JTextArea coursesTakenTextArea = new JTextArea(5, 20);

        coursesTakenTextArea.setText(coursesTaken);
        coursesTakenTextArea.setEditable(false);
    }
}
```

Here we check whether username and password are correct. If this is true, then we define an array list of all the courses taken and the current courses that we have. We also define new frames, labels and text areas and editing their properties.

```
// creates an empty string
String schedule = "";
// looping through the day and time from the arraylist that got the data from mysql
for(int i = 0; i < 6; i++) {
    // appending the empty string with the day + time, resulting in a schedule format string
    schedule += currentCourseslist.get(i) + ":\n" + coursedaylist.get(i) + "\n" + coursetimelist.get(i) + "\n\n";
}
scheduleTextArea.setText(schedule);
scheduleTextArea.setEditable(false);
JScrollPane scheduleScrollPane = new JScrollPane(scheduleTextArea); // adding a scroll panel because there is too much data

JLabel notificationLabel = new JLabel("Notifications:");
JTextArea notificationTextArea = new JTextArea(5, 20);
notificationTextArea.setText("");
notificationTextArea.setEditable(false);
// to start a new line if the text area width is small
notificationTextArea.setLineWrap(true);
notificationTextArea.setWrapStyleWord(true);
```

Then we create an empty string that will include the schedule. We create a for loop that appends the empty string with the day and time which will result in a schedule formatted string. We add a scroll pane as the content is a lot for the panel to handle. We also create a notification panel for the professor to communicate with the students through it.

```
// Add components for current courses taken
// by creating a table containing all the data that the professor set for this student(data will be added later)
JLabel currentCoursesLabel = new JLabel("Current Courses:");
JPanel currentCoursesPanel = new JPanel(new GridLayout(currentCourseslist.size() + 1, 7));
currentCoursesPanel.add(new JLabel(""));
currentCoursesPanel.add(new JLabel("Absence"));
currentCoursesPanel.add(new JLabel("Test 1"));
currentCoursesPanel.add(new JLabel("Test 2"));
currentCoursesPanel.add(new JLabel("Assignment"));
currentCoursesPanel.add(new JLabel("Final"));
currentCoursesPanel.add(new JLabel("Total"));
for (String course : currentCourseslist) {
    JLabel courseLabel = new JLabel(course);
    JTextField absenceTextField = new JTextField("0", 5);
    absenceTextField.setEditable(false);
    JTextField test1TextField = new JTextField("0", 5);
    test1TextField.setEditable(false);
    JTextField test2TextField = new JTextField("0", 5);
    test2TextField.setEditable(false);
    JTextField assignmentTextField = new JTextField("0", 5);
    assignmentTextField.setEditable(false);
    JTextField finalTextField = new JTextField("0", 5);
    finalTextField.setEditable(false);
    JTextField totalTextField = new JTextField("0", 5);
    totalTextField.setEditable(false);
    currentCoursesPanel.add(courseLabel);
    currentCoursesPanel.add(absenceTextField);
    currentCoursesPanel.add(test1TextField);
    currentCoursesPanel.add(test2TextField);
    currentCoursesPanel.add(assignmentTextField);
    currentCoursesPanel.add(finalTextField);
    currentCoursesPanel.add(totalTextField);
}
}
```

Here we position and add a new label for each component, which are the absence and test 1 and test 2 and assignments and final. We also add a for loop to add all the new text fields for all the parts. Since this is the student portal, all the text fields are set to not editable. All the data in this panel is added by the professors.

```
// Call the addStudentDataToPanel method to fill up the values in the currentCoursesPanel
String student = "Abdullah El-Haddad"; // Replace with the name of the student you want to retrieve data for
Object[] studentData = dbmanager.addStudentDataToPanel("Courses_db", "course1", student);

int absence = (int) studentData[0];
float test1 = (float) studentData[1];
float test2 = (float) studentData[2];
float assignment = (float) studentData[3];
float finalScore = (float) studentData[4];
String notification = (String) studentData[5];

float total = test1 + test2 + assignment + finalScore;
// the professor is teaching this student 3 courses which are number 3,4,5 in the current courses
// so we need to append the data for these specific courses
// the table first row is all labels, the rest of the rows contain:(course label, absence,test1,test2,assignment,final,total)
// so we need to ignore components 0-6, then course 1: 7-13, course 2: 14-20
// hence for course 3 we have to get components: 22-27(we don't need 21-course name)
JTextField absenceTextField1 = (JTextField) currentCoursesPanel.getComponent(22);
absenceTextField1.setText(Integer.toString(absence));
JTextField test1TextField1 = (JTextField) currentCoursesPanel.getComponent(23);
test1TextField1.setText(Float.toString(test1));
JTextField test2TextField1 = (JTextField) currentCoursesPanel.getComponent(24);
test2TextField1.setText(Float.toString(test2));
JTextField assignmentTextField1 = (JTextField) currentCoursesPanel.getComponent(25);
assignmentTextField1.setText(Float.toString(assignment));
JTextField finalTextField1 = (JTextField) currentCoursesPanel.getComponent(26);
finalTextField1.setText(Float.toString(finalScore));
JTextField totalTextField1 = (JTextField) currentCoursesPanel.getComponent(27);
totalTextField1.setText(Float.toString(total));

notificationTextArea.setText(notification);
```

Here we call the function addStudentDataToPanel and we create the variables that will be displayed in the text fields. Then we get the components from the SQL using getComponent function. We recreate the same thing more than once to get the different components for each subject.

```

        // Close the database connection
        dbmanager.closeConnection();

        // Add components to main frame
        JPanel mainPanel = new JPanel(new GridLayout(4, 2));
        mainPanel.add(coursesTakenLabel);
        mainPanel.add(coursesTakenTextArea);
        mainPanel.add(scheduleLabel);
        mainPanel.add(scheduleScrollPane);
        mainPanel.add(currentCoursesLabel);
        mainPanel.add(currentCoursesPanel);
        mainPanel.add(notificationLabel);
        mainPanel.add(notificationTextArea);

        mainFrame.add(mainPanel);
        mainFrame.setVisible(true);
        dispose(); // Close the login portal window
    }

    else { // displaying a pop-up window containing a message if the username or password are incorrect
        JOptionPane.showMessageDialog(this, "Invalid Username or Password");
    }
}

public static void main(String[] args) {
    new CourseManagementSystem(); // To run the program
}
}

```

Finally, we close the database connection and we add all the components to the main frame. We also have the else statement to check if the username and password are not correct, a message dialogue will appear. We then call the constructor of course management system in the main function to run the GUI.

- ProfessorPortal:

```

1 public class ProfessorPortal extends JFrame implements ActionListener{
2     private JLabel proflabel;
3     private JLabel passlabel;
4     private JTextField protextfield;
5     private JPasswordField passtextfield;
6     private JButton loginbutton;
7     private DatabaseManager dbmanager1;
8
9     public ProfessorPortal() {
10         setTitle("Login Portal");
11         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
12         setSize(300, 150);
13         setLocationRelativeTo(null);
14
15         proflabel = new JLabel("Username:");
16         passlabel = new JLabel("Password:");
17
18         protextfield = new JTextField(10);
19         passtextfield = new JPasswordField(10);
20
21         loginbutton = new JButton("Login");
22         loginbutton.addActionListener(this);
23
24         JPanel panel = new JPanel(new GridLayout(3, 2));
25         panel.add(proflabel);
26         panel.add(protextfield);
27         panel.add(passlabel);
28         panel.add(passtextfield);
29         panel.add(new JLabel(""));
30         panel.add(loginbutton);
31
32         add(panel, BorderLayout.CENTER);
33         setVisible(true);
34     }
35
36     public void actionPerformed(ActionEvent e) {
37         String username = protextfield.getText();
38         String password = new String(passtextfield.getPassword());
39
40         // checks whether the username and password are correct
41         if (username.equals("Dr. Mohammed Arif") && password.equals("UCSII234")) {
42             dbmanager1 = new DatabaseManager();
43
44             ArrayList<String> courselist1 = new ArrayList<String>();
45             courselist1 = dbmanager1.getCourseName(); // retrieving the courses from mysql
46
47             ArrayList<String> courses = new ArrayList<String>();
48
49             // adding just three courses to the professor to teach
50             for (int i = 2; i < 5; i++) {
51                 courses.add(courselist1.get(i));
52             }
53             String coursesTeaching = String.join("\n", courses); // storing the courses he teaches in a string
54         }
55     }
56 }

```

We start by initializing the necessary variables. Then we must set up the login portal of the professor portal. We set up the GUI components, then the textfields and password field and login button. We add an actionPerformed to the login button. And inside it we get the text of the textfields and use an if-else statement to check whether the entered username/password are correct or not. If they are correct, a new window opens. We start the new window by using the database manager methods to retrieve course name from mysql and using a for-loop to get three course names from it, then storing them in an arraylist and finally converting it to string.

```

// retrieving the course schedule from mysql, and storing it in two arraylists(day & time)
ArrayList<String> coursedaylist = new ArrayList<String>();
coursedaylist = dbmanager.getCourseDay();

ArrayList<String> coursetimelist = new ArrayList<String>();
coursetimelist = dbmanager.getCourseTime();

// declaring an empty arraylist to add the schedule to it
ArrayList<String> currentCoursetimelist = new ArrayList<String>();

// using a for loop to add the day and time of each course,
for (int i = 2; i < 5; i++) {
    currentCoursetimelist.add(coursedaylist.get(i));
    currentCoursetimelist.add(coursetimelist.get(i));
}

// converting the schedule arraylist to a string, because it will be easier to add to text areas
String currentCoursetime = String.join("\n", currentCoursetimelist);

JFrame mainframe = new JFrame("Course Management System(Professor)");
mainframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
mainframe.setSize(550, 350);
mainframe.setLocationRelativeTo(null);
mainframe.setResizable(false);

JPanel mainpanel = new JPanel(new FlowLayout());

JLabel courseschedule = new JLabel("Schedule:");
JTextArea schedulertextarea = new JTextArea(10, 20);
schedulertextarea.append(currentCoursetime); // appending the schedule text area with the string that we created earlier

schedulertextarea.setEditable(false);

JLabel coursesLabel = new JLabel("Courses:");
JTextArea coursestextarea = new JTextArea(10, 20);
coursestextarea.append(coursesTeaching); // appending the courses text area with the string that we created earlier

```

Then, we also use the database manager class methods to get the course time and day, use a for-loop to store them in a single arraylist and again converting it to string. Then we set up the GUI components, and add the schedule text area and the courses text area. And add the strings we created earlier to the text areas.

```

// A button for the first course
JButton course1 = new JButton(courseList1.get(2));
course1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent a) {
        JFrame mainframe = new JFrame("Course Options");
        mainframe.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        mainframe.setSize(800, 400);
        mainframe.setLocationRelativeTo(null);
        mainframe.setResizable(false);
        mainframe.setVisible(true);
        JPanel jPanel1 = new JPanel(new GridLayout());
        DatabaseManager dbmanager2 = new DatabaseManager();

        // creating a table to add the students to it
        DefaultTableModel model = new DefaultTableModel();
        JTable table = new JTable(model);
        // adding columns for the user (professor) to edit and give marks
        model.addColumn("Student Name");
        model.addColumn("Absences");
        model.addColumn("Remaining Absences");
        model.addColumn("Test 1");
        model.addColumn("Test 2");
        model.addColumn("Assignment");
        model.addColumn("Final");
        model.addColumn("Notifications");

        // to determine how the data in a cell is displayed
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment(JLabel.CENTER);

        // using a text file with 10 student names to fill the table
        try {
            File file = new File("Students.txt");
            Scanner scanner = new Scanner(file);
            while (scanner.hasNext()) {
                String line = scanner.nextLine();
                String[] names = line.split(","); // because the names are separated by a comma
                for (String name : names) { // loops through all the names
                    int absences = 0;
                    int remainingAbsences = 7;
                    float test1 = 0;
                    float test2 = 0;
                    float assignment = 0;
                    float finalExam = 0;
                    String notification = "";
                    // adding different columns for each student
                    model.addRow(new Object[] {name, absences, remainingAbsences, test1, test2, assignment, finalExam, notification});
                    // adding the data to database
                    dbmanager2.insertStudentData("Courses.db", "course1", name, absences, test1, test2, assignment, finalExam, notification);
                }
            }
        } catch (FileNotFoundException e2) { // catching file exception
            System.out.println("File not found: " + e2.getMessage());
        }

        // setting table size
        table.getColumnModel().getColumn(0).setPreferredWidth(150);
        table.getColumnModel().getColumn(1).setPreferredWidth(50);
        table.getColumnModel().getColumn(2).setPreferredWidth(120);
        table.getColumnModel().getColumn(3).setPreferredWidth(50);
        table.getColumnModel().getColumn(4).setPreferredWidth(50);
        table.getColumnModel().getColumn(5).setPreferredWidth(80);
        table.getColumnModel().getColumn(6).setPreferredWidth(50);
        table.getColumnModel().getColumn(7).setPreferredWidth(200);
        table.getColumnModel().getColumn(2).setCellRenderer(centerRenderer);
        table.getColumnModel().getColumn(1).setCellEditor(new DefaultCellEditor(new JTextField()));
    }
});

```

After that we create a button for each course, an example is the first course. We add an action listener to the button, so when the button is clicked a new window opens. We start by initializing the GUI components then creating a table with multiple columns. After that, we use a try-catch block to open a text file containing student names. Then, we populate the table using the text file.

```

table.getColumnModel().getColumn(1).getCellEditor().addCellEditorListener(new CellEditorListener() {
    @Override
    public void editingStopped(ChangeEvent e) {
        int row = table.getSelectedRow();
        int column = table.getSelectedColumn();
        String value = table.getValueAt(row, column).toString();
        try {
            if (column == 3) {
                // Handle changes to the attendance column
                int absence = Integer.parseInt(value);
                int remainingAbsences = 1 - absence;
                table.getValueAt(remainingAbsences, row, 3);
            } else {
                // Handle changes to the other columns
                float score = Float.parseFloat(value);
                if (score < 0 || score > 100) {
                    throw new NumberFormatException();
                }
            }

            // Update the MySQL table with the new values once the user finishes changing the value of the first column (attendance)
            System.out.println("Updating student: " + table.getValueAt(row, 0) + " with values: " + table.getValueAt(row, 1) + ", " + table.getValueAt(row, 2) + ", " + table.getValueAt(row, 3) + ", " + table.getValueAt(row, 4) + ", " + table.getValueAt(row, 5) + ", " + table.getValueAt(row, 6) + ", " + table.getValueAt(row, 7) + ", " + table.getValueAt(row, 8) + ", " + table.getValueAt(row, 9));
            DriverManager.getConnection("jdbc:mysql://localhost:3306/courses_db", "root", "root");
            PreparedStatement stmt = DriverManager.getConnection("jdbc:mysql://localhost:3306/courses_db", "root", "root").prepareStatement("UPDATE student SET username=?, password=?, table.getValueAt(row, 1).toString(), score=?, grade=?, table.getValueAt(row, 3).toString(), final=?, grade=?, table.getValueAt(row, 5).toString(), final=?, grade=?, table.getValueAt(row, 7).toString(), final=?, grade=?, table.getValueAt(row, 9).toString()");
            stmt.setString(1, table.getValueAt(row, 1).toString());
            stmt.setString(2, table.getValueAt(row, 2).toString());
            stmt.setFloat(3, score);
            stmt.setString(4, table.getValueAt(row, 3).toString());
            stmt.setString(5, table.getValueAt(row, 4).toString());
            stmt.setString(6, table.getValueAt(row, 5).toString());
            stmt.setString(7, table.getValueAt(row, 6).toString());
            stmt.setString(8, table.getValueAt(row, 7).toString());
            stmt.setString(9, table.getValueAt(row, 8).toString());
            stmt.setString(10, table.getValueAt(row, 9).toString());
            stmt.executeUpdate();
            System.out.println("Invalid input: " + value);
        } catch (Exception e) {
            System.out.println("Invalid input: " + value);
        }
    }
});

// do nothing
}

mainframe.add(new JScrollPane(table));
}
}

```

Then, we add an editing stopped and inside it a database manager class method to store the values of the table in mysql whenever the user stops editing the table. Lastly, we add the table to the mainframe.

```

JButton backButton = new JButton("Back");
backButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        mainframe.dispose(); // close the current screen's JFrame
        mainframe.setVisible(false); // hide it to release resources
        setVisible(true); // show the previous screen's JFrame
    }
});

mainpanel.add(courseschedule);
mainpanel.add(scheduledtextarea);
mainpanel.add(coursesLabel);
mainpanel.add(coursestextarea);
mainpanel.add(course1);
mainpanel.add(course2);
mainpanel.add(course3);
mainpanel.add(backButton);

mainframe.add(mainpanel, BorderLayout.CENTER);
mainframe.setVisible(true);
dispose();
} else { // displaying a pop-up window containing a message if the username or password are incorrect
    JOptionPane.showMessageDialog(this, "Invalid Username or Password");
}
}
}

```

Now, we finish the professor portal by adding a back button to go back to the previous window. And finish by adding the components to the mainpanel, and adding the mainpanel to the mainframe.

- DatabaseManager class:

```

// class to connect to mysql using JDBC
public class DatabaseManager {

    private Connection conn; // declaring connection
    private Statement stmt; // declaring statement

    public DatabaseManager() {
        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/courses_db", "root", "root");
            stmt = conn.createStatement(); // creates a statement object to throw to mysql
        } catch (Exception e) { // catch any exceptions
            e.printStackTrace();
        }
    }
}

```

We start by initializing the connection and statements. We use the username and password “root”. But if someone else has different connection username/password he can change the code to match his connection username/password.

```
// method to insert data to the table we already built in mysql workbench
public void insertData(String dbName, String tableName, int num, String Course, String Day, String Time) {
    try {
        stmt.executeUpdate("USE " + dbName);
        stmt.executeUpdate("INSERT INTO " + tableName + " (num, Course, Day, Time) VALUES (" + num + ", '" + Course + "', '" + Day + "', '" + Time + "')");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

The first method we have in this class is used to insert course data to the table in mysql.

```
// method to insert data from the user in the gui to the table we already created in mysql
public void insertStudentData(String dbName, String tableName, String student, int absence, float test1, float test2, float assignment, float finalScore, String notifications) {
    try {
        stmt.executeUpdate("USE " + dbName);
        // Check if the student data already exists in the table
        ResultSet rs = stmt.executeQuery("SELECT * FROM " + tableName + " WHERE student = '" + student + "'");
        boolean dataExists = rs.next();
        rs.close();

        if (dataExists) {
            // If the student data already exists, update the corresponding row in the table
            stmt.executeUpdate("UPDATE " + tableName + " SET absence = " + absence + ", test1 = " + test1 + ", test2 = " + test2 + ", assignment = " + assignment + ", finalScore = " + finalScore + ", notifications = '" + notifications + "'");
        } else {
            // If the student data does not exist, insert it
            stmt.executeUpdate("INSERT INTO " + tableName + " (student, absence, test1, test2, assignment, finalScore, notifications) VALUES ('" + student + "', " + absence + ", " + test1 + ", " + test2 + ", " + assignment + ", " + finalScore + ", '" + notifications + "')");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

The second method is used to insert student data to the table in mysql.

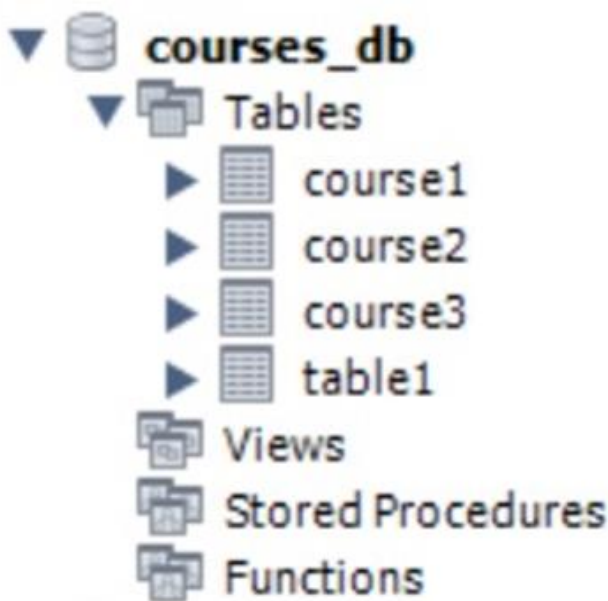
```
// method to add the student data from the professor portal to the panel in the student portal
public Object[] addStudentDataToPanel(String dbName, String tableName, String student) {
    Object[] data = new Object[6];
    try {
        stmt.executeUpdate("USE " + dbName);
        ResultSet rs = stmt.executeQuery("SELECT * FROM " + tableName + " WHERE student = '" + student + "'");
        while (rs.next()) {
            data[0] = rs.getInt("absence");
            data[1] = rs.getFloat("test1");
            data[2] = rs.getFloat("test2");
            data[3] = rs.getFloat("assignment");
            data[4] = rs.getFloat("final");
            data[5] = rs.getString("notifications");

            int absence = rs.getInt("absence");
            float test1 = rs.getFloat("test1");
            float test2 = rs.getFloat("test2");
            float assignment = rs.getFloat("assignment");
            float finalScore = rs.getFloat("final");
            String notifications = rs.getString("notifications");
        }
        rs.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return data;
}
```

The third method is used to add student data from the mysql table to the student portal, but it only adds data where the student is the one we passed to the method.

```
// closing the mysql connection
public void closeConnection() {
    try {
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

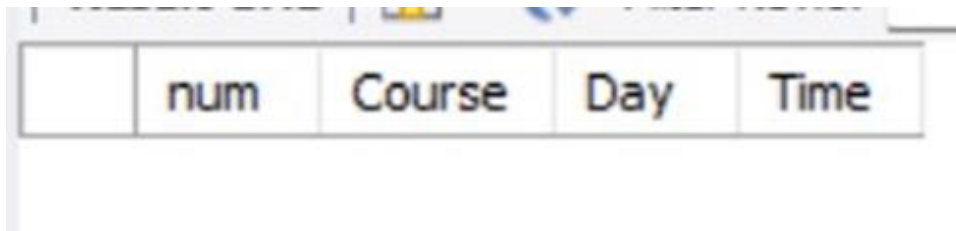
Lastly, we close the connection just to be safe.



The schema name and tables names inside it (mysql workbench)

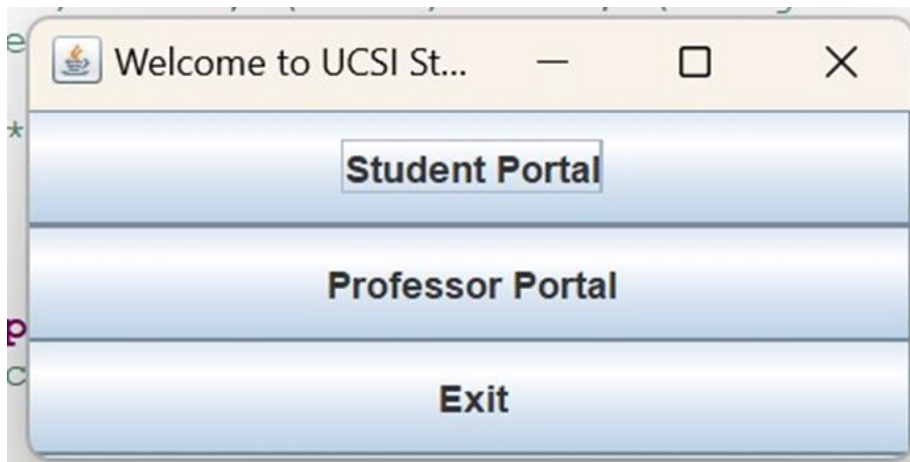
	student	absence	test1	test2	assignment	final	notifications
▶	NULL	NULL	NULL	NULL	NULL	NULL	NULL

course1 - course2 - course3 tables formatting.

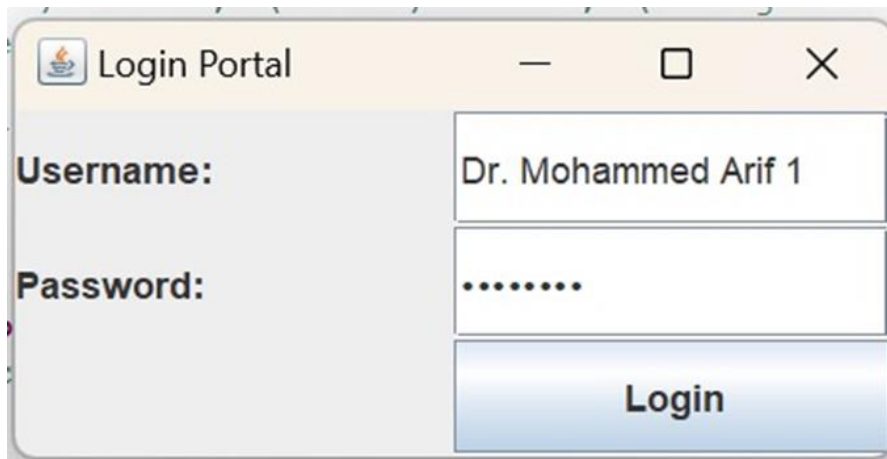


num	Course	Day	Time
-----	--------	-----	------

table1 table formatting



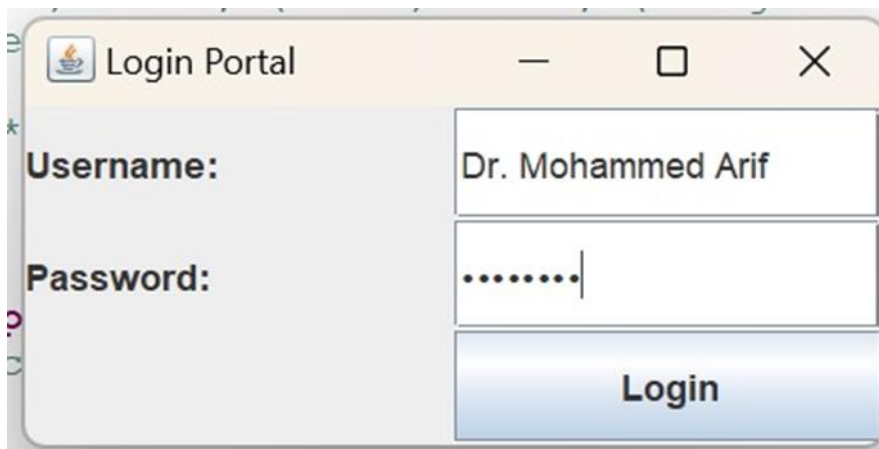
When we run our program, this is the first window that pops up. The exit button closes the program.



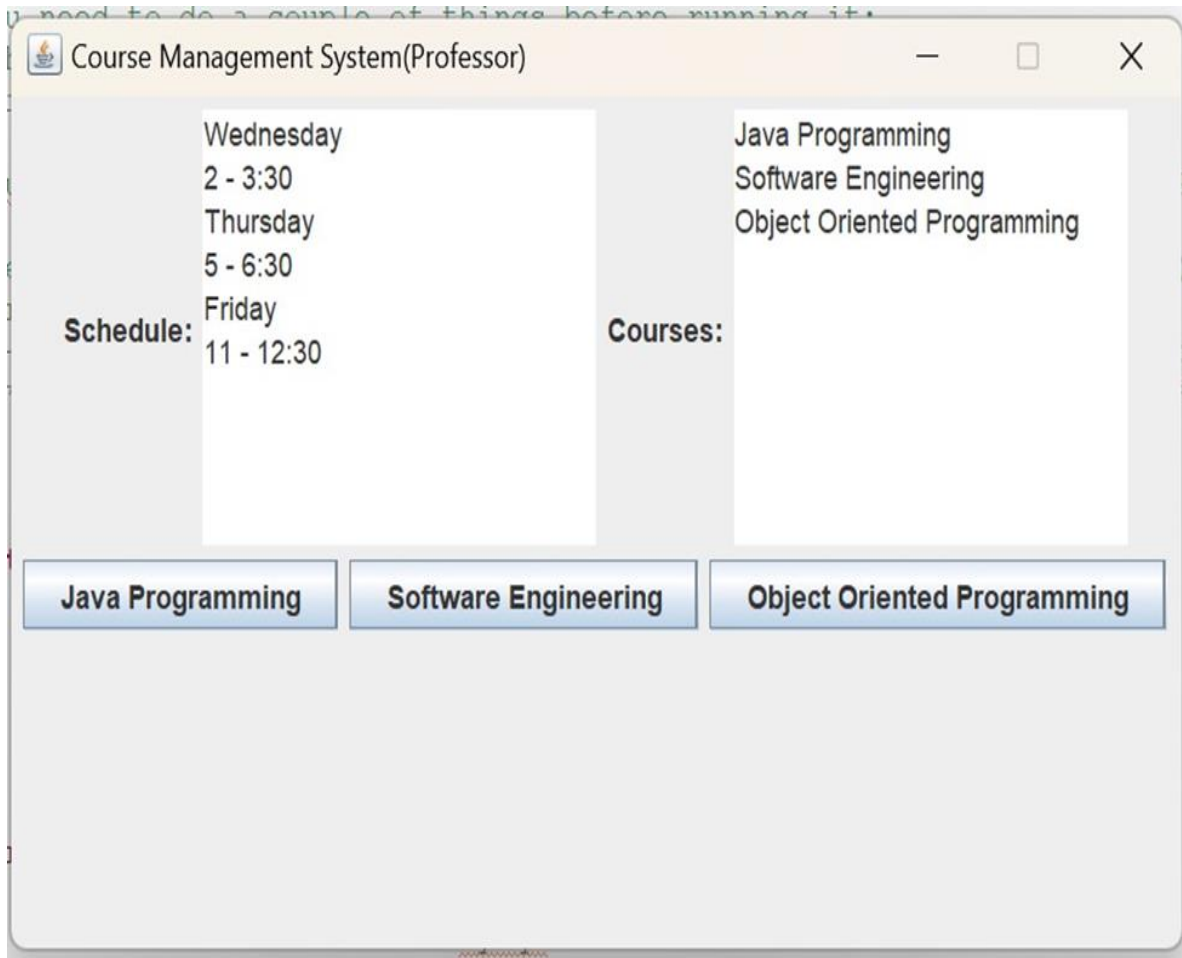
When we press on the professor portal, a login portal appears. If we enter the wrong username/password. the next photo displays what will happen.



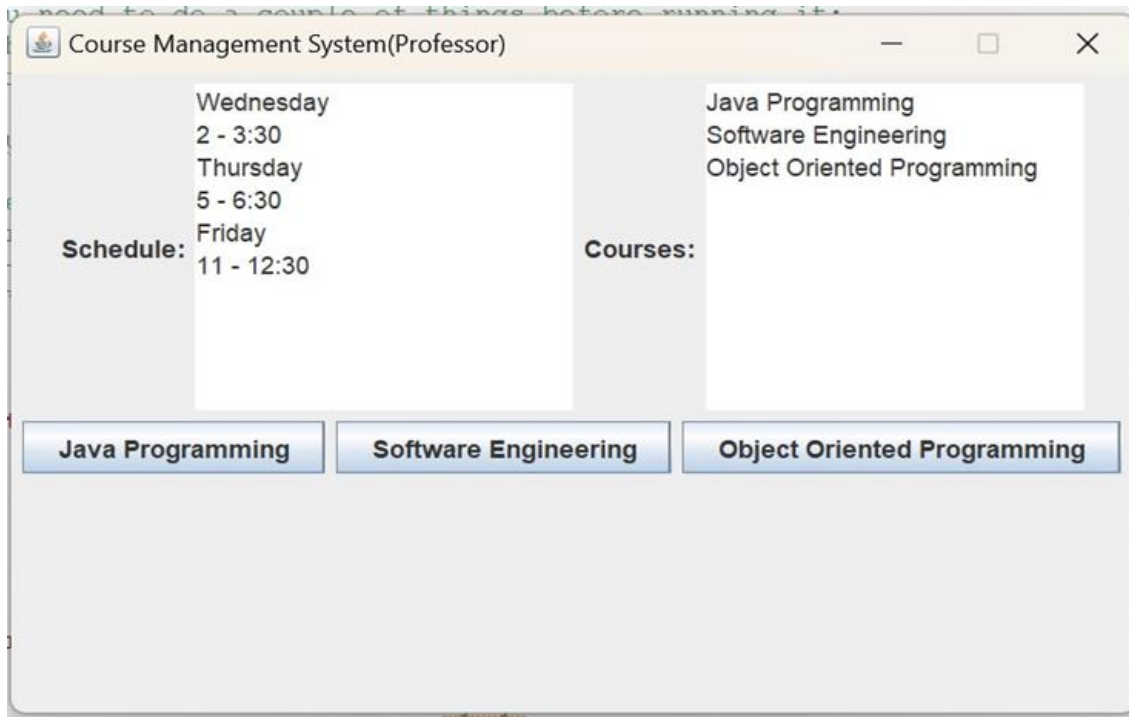
When we enter the wrong username/password in the professor portal.



when we enter the wrong username/password in the professor portal.



now when we enter the correct username/password in the professor portal, the next photo should display the professor portal course management system.



the professor portal course management system, showing the courses the professor is teaching and his schedule. and a button for each course.

The screenshot shows a window titled "Course Options" containing a table with student performance data. The table has 8 columns: Student Name, Absences, Remaining Absences, Test 1, Test 2, Assignment, Final, and Notifications. There are 10 rows of student data.

Student Name	Absences	Remaining Absences	Test 1	Test 2	Assignment	Final	Notifications
Abdullah El-Haddad	0	7	0.0	0.0	0.0	0.0	
Ali Hussain	0	7	0.0	0.0	0.0	0.0	
Mohammed Abdul-Aziz	0	7	0.0	0.0	0.0	0.0	
Ahmed Bader	0	7	0.0	0.0	0.0	0.0	
Yassir Al-Meshel	0	7	0.0	0.0	0.0	0.0	
Abdulrahman kamiyabi	0	7	0.0	0.0	0.0	0.0	
Aymen Al-Zamel	0	7	0.0	0.0	0.0	0.0	
Saed Mohammed	0	7	0.0	0.0	0.0	0.0	
Ahmed Al-Qaseer	0	7	0.0	0.0	0.0	0.0	
Medhet Shalaby	0	7	0.0	0.0	0.0	0.0	

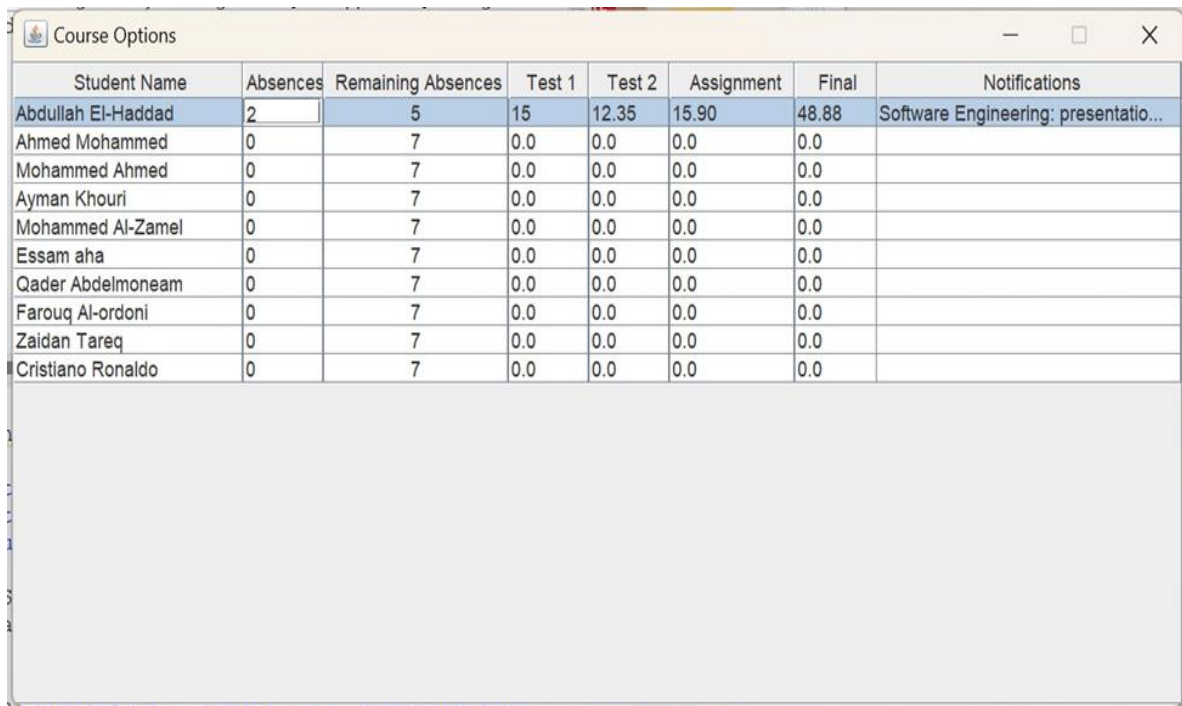
if we press the first button (java programming) a list of students taking this course will appear. with multiple columns that the professor can edit and should be displayed later in the student portal.

Course Options							
Student Name	Absences	Remaining Absences	Test 1	Test 2	Assignment	Final	Notifications
Abdullah El-Haddad	1	6	15	15	20	50	Java Programming: All marks are p...
Ali Hussain	0	7	0.0	0.0	0.0	0.0	
Mohammed Abdul-Aziz	0	7	0.0	0.0	0.0	0.0	
Ahmed Bader	0	7	0.0	0.0	0.0	0.0	
Yassir Al-Meshel	0	7	0.0	0.0	0.0	0.0	
Abdulrahman kamiyabi	0	7	0.0	0.0	0.0	0.0	
Aymen Al-Zamel	0	7	0.0	0.0	0.0	0.0	
Saed Mohammed	0	7	0.0	0.0	0.0	0.0	
Ahmed Al-Qaseer	0	7	0.0	0.0	0.0	0.0	
Medhet Shalaby	0	7	0.0	0.0	0.0	0.0	

editing the columns for the student Abdullah El-Haddad, notice the remaining absences gets updated when we edit the absences column

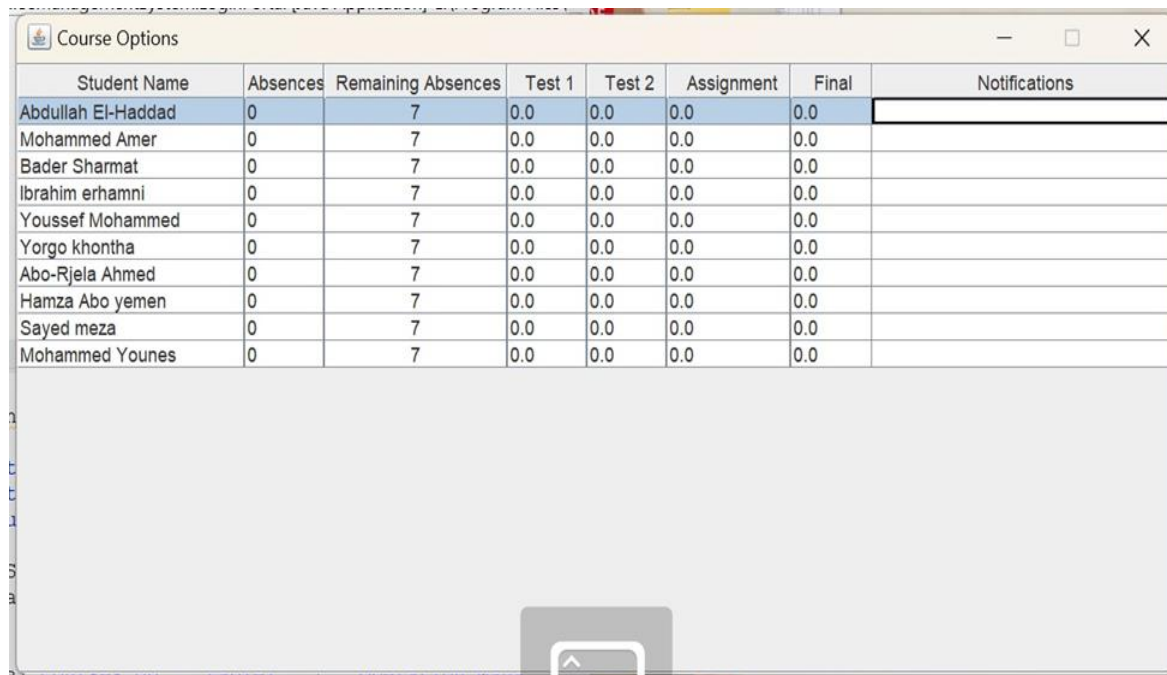
Course Options							
Student Name	Absences	Remaining Absences	Test 1	Test 2	Assignment	Final	Notifications
Abdullah El-Haddad	0	7	0.0	0.0	0.0	0.0	
Ahmed Mohammed	0	7	0.0	0.0	0.0	0.0	
Mohammed Ahmed	0	7	0.0	0.0	0.0	0.0	
Ayman Khouri	0	7	0.0	0.0	0.0	0.0	
Mohammed Al-Zamel	0	7	0.0	0.0	0.0	0.0	
Essam aha	0	7	0.0	0.0	0.0	0.0	
Qader Abdelmoneam	0	7	0.0	0.0	0.0	0.0	
Farouq Al-ordoni	0	7	0.0	0.0	0.0	0.0	
Zaidan Tareq	0	7	0.0	0.0	0.0	0.0	
Cristiano Ronaldo	0	7	0.0	0.0	0.0	0.0	

now for the software engineering button.



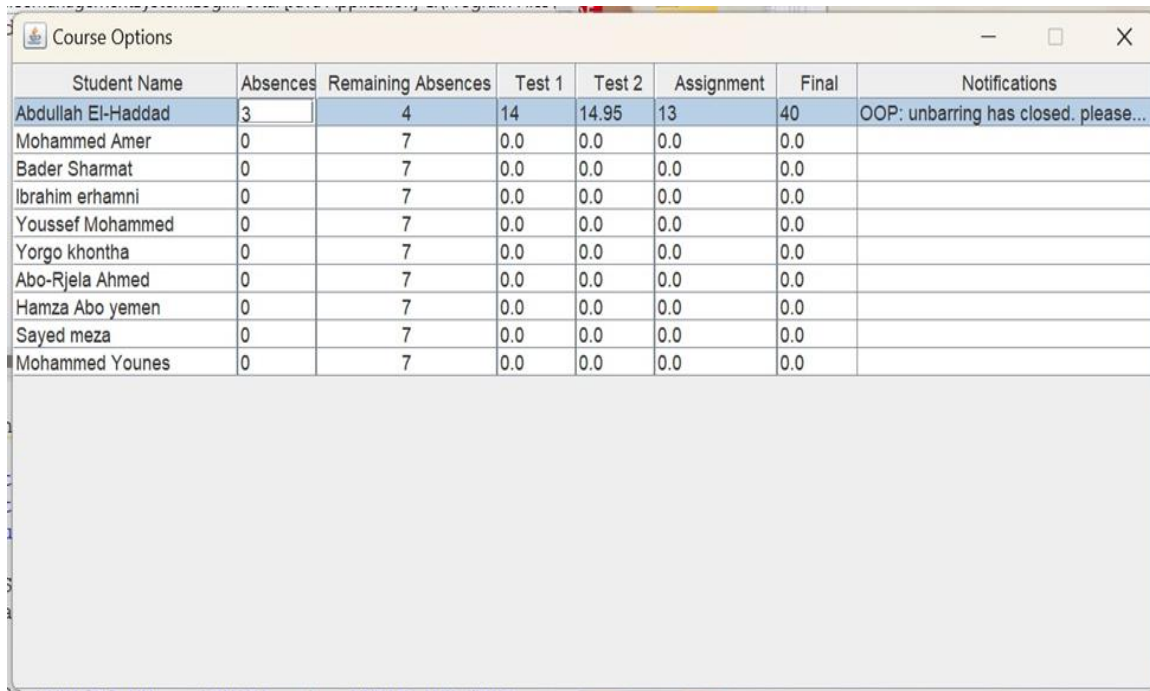
Student Name	Absences	Remaining Absences	Test 1	Test 2	Assignment	Final	Notifications
Abdullah El-Haddad	2	5	15	12.35	15.90	48.88	Software Engineering: presentatio...
Ahmed Mohammed	0	7	0.0	0.0	0.0	0.0	
Mohammed Ahmed	0	7	0.0	0.0	0.0	0.0	
Ayman Khouri	0	7	0.0	0.0	0.0	0.0	
Mohammed Al-Zamel	0	7	0.0	0.0	0.0	0.0	
Essam aha	0	7	0.0	0.0	0.0	0.0	
Qader Abdelmoneam	0	7	0.0	0.0	0.0	0.0	
Farouq Al-ordoni	0	7	0.0	0.0	0.0	0.0	
Zaidan Tareq	0	7	0.0	0.0	0.0	0.0	
Cristiano Ronaldo	0	7	0.0	0.0	0.0	0.0	

the same thing happens



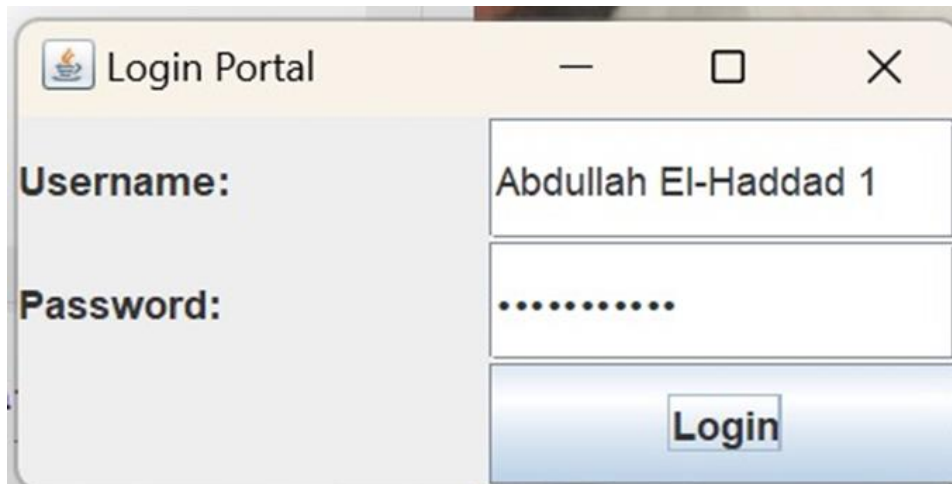
Student Name	Absences	Remaining Absences	Test 1	Test 2	Assignment	Final	Notifications
Abdullah El-Haddad	0	7	0.0	0.0	0.0	0.0	
Mohammed Amer	0	7	0.0	0.0	0.0	0.0	
Bader Sharmat	0	7	0.0	0.0	0.0	0.0	
Ibrahim erhamni	0	7	0.0	0.0	0.0	0.0	
Youssef Mohammed	0	7	0.0	0.0	0.0	0.0	
Yorgo khontha	0	7	0.0	0.0	0.0	0.0	
Abo-Rjela Ahmed	0	7	0.0	0.0	0.0	0.0	
Hamza Abo yemen	0	7	0.0	0.0	0.0	0.0	
Sayed meza	0	7	0.0	0.0	0.0	0.0	
Mohammed Younes	0	7	0.0	0.0	0.0	0.0	

the Object-oriented programming buttons



Student Name	Absences	Remaining Absences	Test 1	Test 2	Assignment	Final	Notifications
Abdullah El-Haddad	3	4	14	14.95	13	40	OOP: unbarring has closed. please...
Mohammed Amer	0	7	0.0	0.0	0.0	0.0	
Bader Sharmat	0	7	0.0	0.0	0.0	0.0	
Ibrahim erhamni	0	7	0.0	0.0	0.0	0.0	
Youssef Mohammed	0	7	0.0	0.0	0.0	0.0	
Yorgo khontha	0	7	0.0	0.0	0.0	0.0	
Abo-Rjela Ahmed	0	7	0.0	0.0	0.0	0.0	
Hamza Abo yemen	0	7	0.0	0.0	0.0	0.0	
Sayed meza	0	7	0.0	0.0	0.0	0.0	
Mohammed Younes	0	7	0.0	0.0	0.0	0.0	

the same thing happens again



Username: Abdullah El-Haddad 1

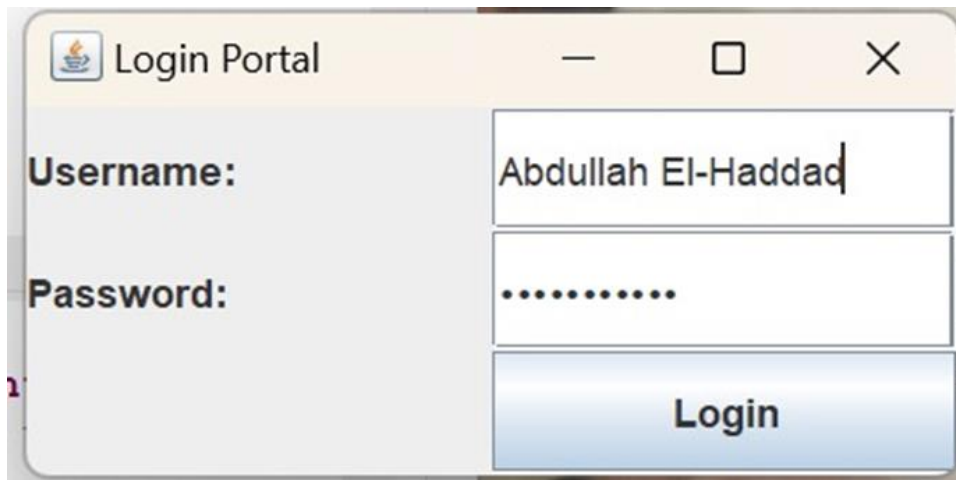
Password:

Login

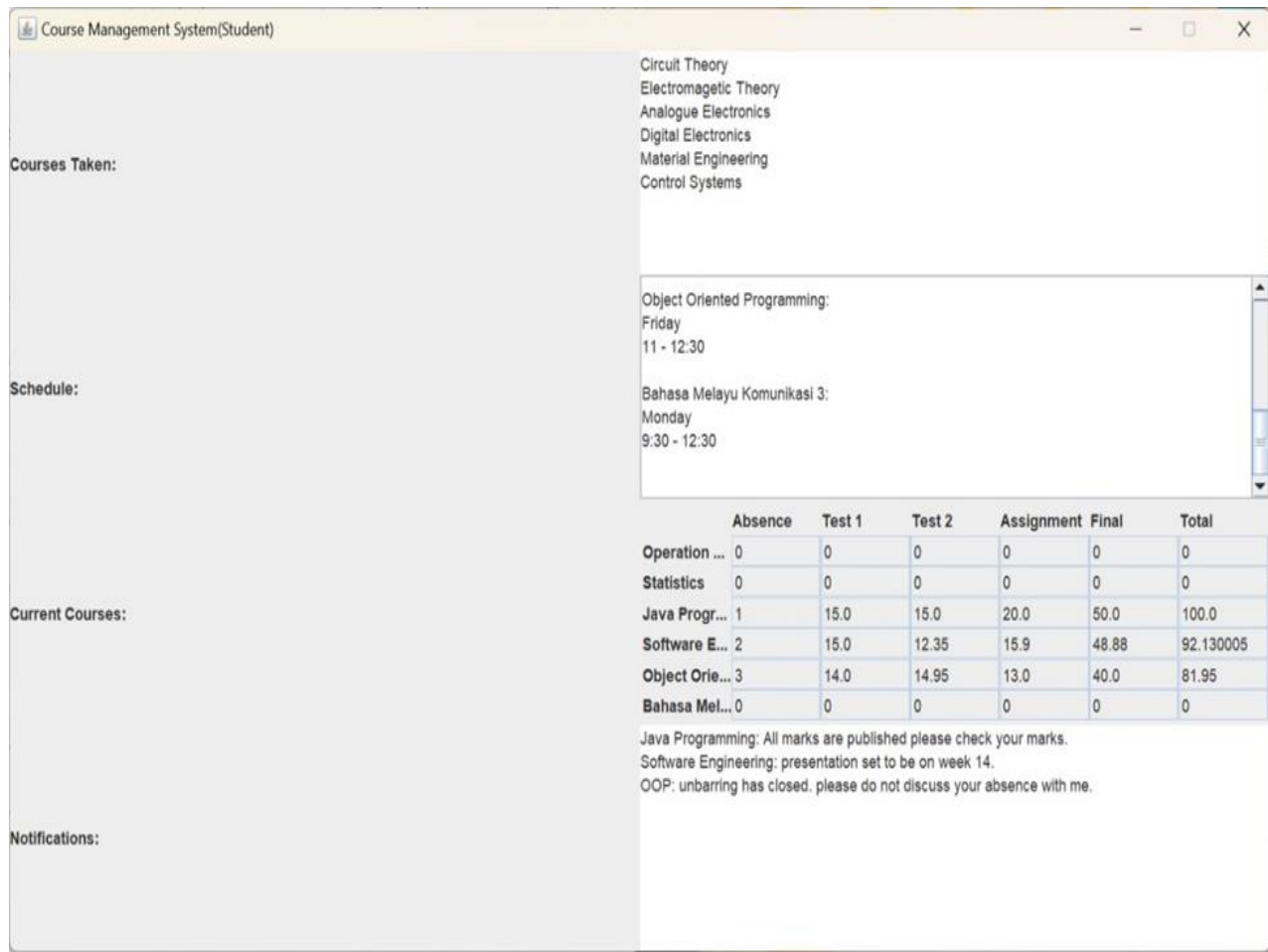
now we need to access the student portal, first we will input a wrong username/password to test the program



it shows a message that the username/password are invalid, so the program works perfectly.



now we will try inputting the correct username/password



Courses Taken:

- Circuit Theory
- Electromagnetic Theory
- Analogue Electronics
- Digital Electronics
- Material Engineering
- Control Systems

Schedule:

Object Oriented Programming:
Friday
11 - 12:30

Bahasa Melayu Komunikasi 3:
Monday
9:30 - 12:30

Current Courses:

	Absence	Test 1	Test 2	Assignment	Final	Total
Operation ...	0	0	0	0	0	0
Statistics	0	0	0	0	0	0
Java Progr...	1	15.0	15.0	20.0	50.0	100.0
Software E...	2	15.0	12.35	15.9	48.88	92.130005
Object Ori...	3	14.0	14.95	13.0	40.0	81.95
Bahasa Mel...	0	0	0	0	0	0

Notifications:

Java Programming: All marks are published please check your marks.
 Software Engineering: presentation set to be on week 14.
 OOP: unbarring has closed. please do not discuss your absence with me.

The student portal course management system appears. working perfectly. with list of courses taken before this sem. the schedule of the courses taken this semester with a scroll panel. the absences and marks of current courses in the form of a table (only three courses are updated because the professor in the professor portal only teaches 3 courses) and the notifications from the professor portal as well.

In this system there are only 2 people in action: one professor and one student. The professor teaches the student three courses. But this system can be widened and expanded to have multiple professors and multiple students. If that happens: the other students in the professor portal can be updated (now when we update them nothing happens because they don't have a student portal) and other professors can teach this student other courses and then all the courses will be updated in the student portal.

6. Difficulties:

While working on this project we have faced some issues like:

- 1- Using the JDBC to connect to the MySQL database. As we had little time to master the chapter, some difficulties were faced while connecting.
- 2- Updating the MySQL table when the professor stops editing, we used editing to solve the problem but it took us sometime to figure it out
- 3- Accessing a specific row and column in the MySQL table. It was hard to access the specific data we wanted to insert in a specific place we want

7. Future Improvements:

Further improvements could include:

- 1- More functions like course selection, fee payment, and academic history can be added
- 2- The GUI can further be enhanced or customized by educational institutions.
- 3- The system could eventually be integrated with a chat box or comment bar so the students can reach out to their lecturers.
- 4- Adding themes and colors to the application
- 5- The system should tell the professors if they entered more marks than permitted for each component.

8. Conclusion:

In conclusion the above code implements a course management system with two portals integrated with a database manager

- Professor portal
- Student portal
- Database manager

It helps serve functionalities like recording the attendance, keying in internal marks for tests and assignments, notifications as well as displaying final score. The database is connected using MySQL connectivity with Java. For implementing security measures, the login portal uses authentication by asking the user to enter the password to verify identity. The graphic user interface is created using the swing library in Java. We have successfully made a course management system that proves beneficial and has higher efficiency compared to a manual system.