# Assessment B: Implementation Report

NC1605B – Group Project Module
Oguz Kagan Cayir – 2377273
2377273@brunel.ac.uk

## 1 Table of Contents



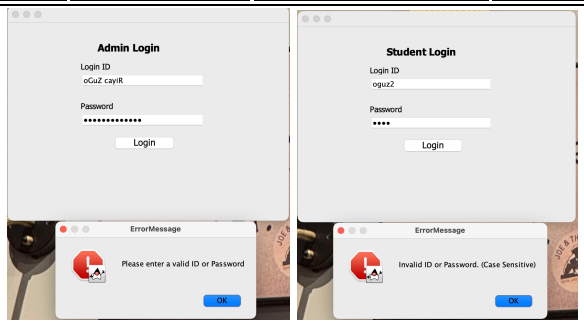
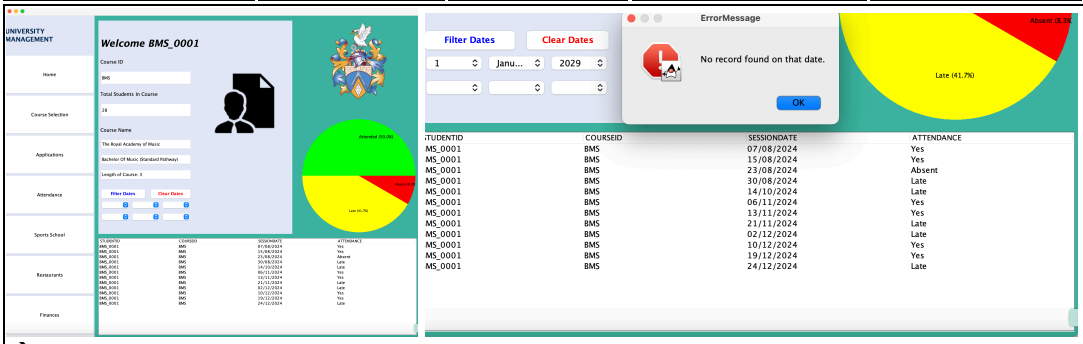
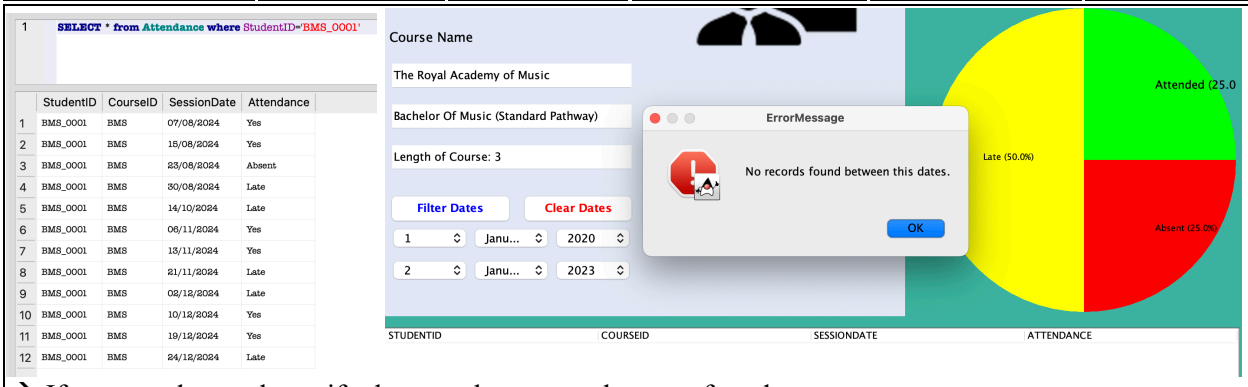
## 2 Introduction

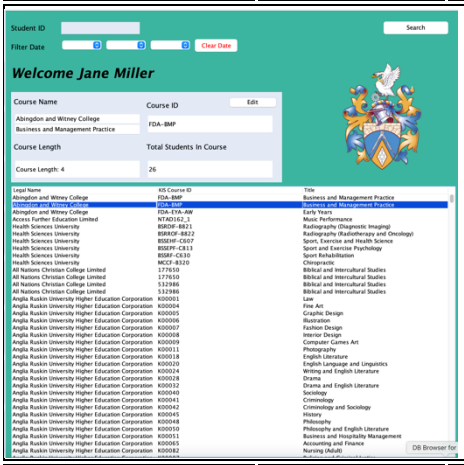
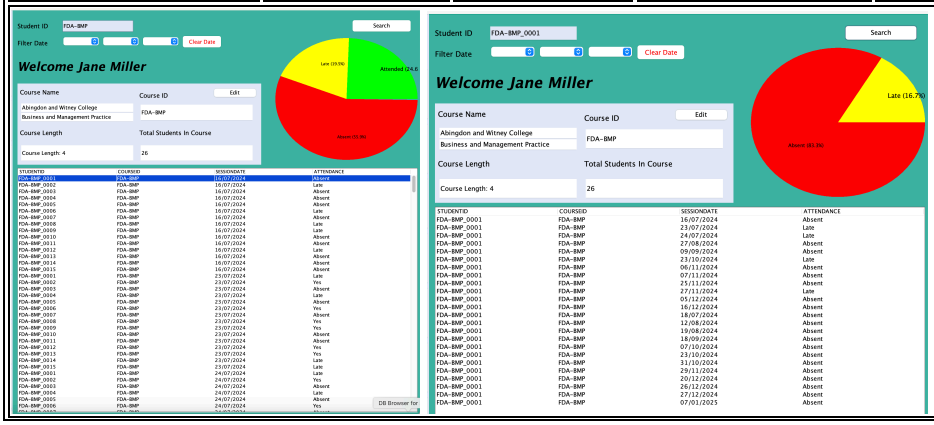
The purpose of this report is to document the implementation phase of the Attendance Monitoring component for the University Management System (UMS). This phase focused on translating the design from Assessment A into functional code, addressing challenges such as database integration, role-based authentication, and dynamic UI elements like pie charts. The report also evaluates testing outcomes and deviations from the original design.

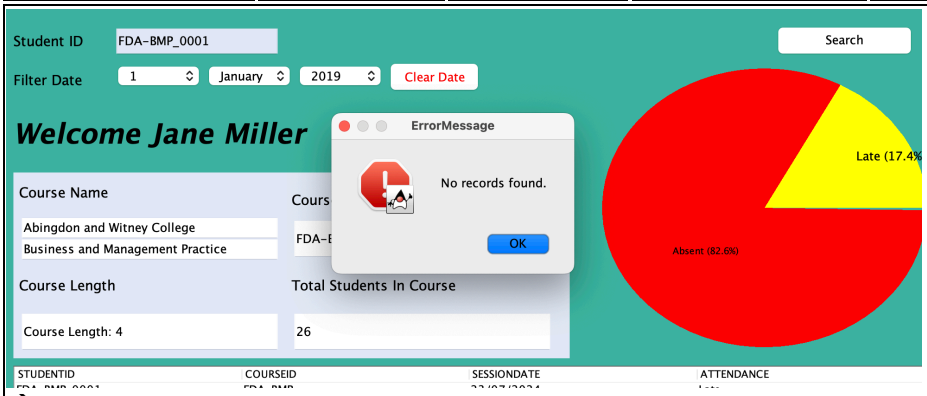
## 3 Requirements

*Table 1: Functional Requirement Specification*

| Functional Requirement | User Role | Input | Output | Implemented? | Notes |
|---|---|---|---|---|---|
| **1. User Login** | Student/ Admin | Username and Password | System validates credentials, grants access to the respective user interface based on role (Student or Admin). | **YES**. ✓ | "Admin Login" uses specifically the word "like" instead of "=". Because Admin credentials are created randomly as name and surname. They are text instead of ID number unlike "Student Login". |

| Functional Requirement | User Role | Input | Output | Implemented? | Notes |
|---|---|---|---|---|---|
|  <br>→ If the username or password is incorrect, display an error message. | | | | | |
| **2. View Attendance** | Student | Student ID, Course ID *(This data to be collected in the backend without a manual input from user)* | Displays the student's attendance data including total attendance percentage and individual session data. | **YES.** ✓ | In order to collect Student ID, created loggedInStudentID variable as public. |
|  <br>→ If no attendance data exists for the course, show a message indicating no records are available. | | | | | |
| **3. Filter Attendance Records** | Student | Filters: Date | Displays the filtered attendance records based on the criteria selected | **YES.** ✓ | Unfortunately given data from university does not contain attendance record for specific module student attended. Only the date. So, filtering by modules (student attended) is impossible. |
|  <br>→ If no records match, notify the user that no results were found. | | | | | |

| Functional Requirement | User Role | Input | Output | Implemented? | Notes |
|---|---|---|---|---|---|
| **4. View All Students' Attendance (Admin)** | Admin | None (for viewing all records) | Displays attendance data for all students across the institution, with the option to filter by student, course, or date. | **YES.** ✓ | Since all studentID's are substrings of courses registered, it also technically works as course search. (Only in this project with given data, it may different in real-life case since studentID's may differ. |
|  | →MouseListener implemented to show the data related to course.<br>→There are some repeating lines coming from the original data given. They have slight differences such as KISMODE or NUMSTAGE. However it is not my area at this point. | | | | |
| **5. Filter Attendance Records (Admin)** | Admin | Filters: Date, Course, StudentID | Displays attendance records filtered based on selected criteria (e.g., by course, student, or date). | **YES.** ✓ | Open for improvement to implement a date range. |
|  | | | | | |

| Functional Requirement | User Role | Input | Output | Implemented? | Notes |
|---|---|---|---|---|---|
|  <br>→ If no records match the filter, inform the admin that no data matches the search parameters. | | | | | |
| **6. Add/Edit Attendance Record** | Admin | Student ID, Course ID, Date, Session, Attendance Status | Updates or adds attendance data for a specific student, course, and date. The attendance status is updated (attended(yes), absent, late). | PARTIALLY. 🟡 | CourseID and Course Names are the only editable area with "Edit" button.<br>Course Length and Total Students are not. |
|  | | | →Even though Course Name, Insititution Name and Course ID were editable as expected; Attendance record were not able to. Attendance records are locked on the table printed below of the screen. However, I decided to setEditable(False) for not making confusion. Later on, this decision came as a challenge and I did not have the time to make more changes with the remaining time for the report submission. | | |
| **7. Security and Access Control** | Student/Admin | Username and Password (for login) | Ensures that only authorized users (students and admins) can access their respective roles and data. | **YES.** ✓ | As intended, system only allows users registered within the database. Admins can create new student registrations.<br>*However, initial idea at Assessment A was also implementing a tracking system(log) to see all activities. It failed due to time-constraints with the project.* |

**Non-Functional Requirements**

- **Performance**: The system retrieves and display attendance data within a tolerable and efficient timeframe.
- **Usability**: The interface is intuitive, allowing easy navigation for both students and admins.
- **Scalability**: The system is designed to efficiently handle attendance records for more than 400,000 students while maintaining optimal performance and reliability.
- **Notifications and Alerts: FAILED.** Initial idea was, similar to Brunel University, to inform students via email, phone number etc.. It could have been also implemented such as a pop-up notification on StudentDisplay class, however it was skipped due to being non-functional requirement and also time-constraints of the report.

# 4 TECHNICAL CODE DESCRIPTION

### 1. Technologies and Tools

Languages: Java (core logic), DB Browser for SQLite (database queries).
Frameworks: Java Window Builder (UI), JDBC (database connectivity).

### *2. Code Structure and Key Classes*

| CLASS | ROLE |
|---|---|
| **DataBaseAdmin** | Creates a database connection via user input from Login class. If the credentials are matched/found with input, changes the Boolean value "x" to "true" and "IsAdminValid()" method returns true. |
| **DataBaseStudent** | Creates a database connection via user input from Login class. If the credentials are matched/found with input, changes the Boolean value "x" to "true" and "IsStudentValid()" method returns true.<br>→Uses repaint and revalidate for better user experience. *(e.g. avoid a student trying to login as an admin)* |
| **Login** | Handles authentication for Students/Admins. Validates credentials against DataBaseAdmin or DataBaseStudent. Also creates class level global variables to be used by other classes (*loggedInStudentID and loggedInAdminID)* |
| **AdminDisplay** | Renders admin dashboard with course tables, search/filter options, and access to RegistrationForm. Upon search, it creates pie charts to visualize output. Mouse-click listeners implemented to display selected course information and allow editing. |
| **StudentDisplay** | Displays student-specific attendance data (via *loggedInStudentID)*, integrates pie chart for visual analytics. Allows user to filter a specific date or a range of dates. |
| **PieChart** | Generates pie charts using awt such as *Graphics2d* and *Rectangle*. Renders attendance states (Attended/Absent/Late). *(Reference: https://stackoverflow.com/questions/13662984/creating-pie-charts-programmatically)* |
| **DatePicker** | Validates and formats date inputs for filtering. Uses simple JComboBox and arrays. getSelectedDate() returns dd/mm/yyyy format for accurate DB date(text format) search. clearSelectedDate() clears the dropboxes as setting null. |
| **RegistrationForm** | Allows admins to add new students via INSERT queries into Unique_Students_With_Passwords and Attendance tables. Admin can choose student's university and department via dropbox (existing universities and departments) |

*NOTE: DataBaseAdmin and DataBaseStudent were one of the first classes created through guidance of Zear Ibrahim. Implementing these connections into Try-Catch statements were later learned but I did not change it to display learning process of mine.*

### *3. Deviations from Assessment A Design*

- ***ADMINS/STUDENTS***: Some layout changes and displaying only one pie chart instead of two. Initial idea was to keep one chart as non-affected by search results, therefore always displays total attendance rate overall. Reason for change is simplicity and avoid confusion.
- ***ADMINS***: Edit function in the table is deducted and implemented into studentInfoPanel above for Admins.
- ***LOGIN***: Forgot your password panel is removed due to not being able to make this panel functional. Initial idea was to contact admin/student to reset their passwords, but it became impossible to implement

as intended on Design phase. However, I believe it is possible to add a new panel into Admins to see "requests" for password changes from students. It may be implemented as a future work.
- ***ADMINS***: Editing records through tables became a challenge near submission date. I set the panel non-editable initially to allow admin to use only StudentInfoPanel. I was unable to make required changes in that time frame to "set only the Attendance column editable".
- ***DATE FIELD***: In original design, Date field was in modern format (*Showing a callendar of the day and month such as in modern booking websites*). However, I was not able to implement in same format without using external libraries, therefore I just gave up on this simply because its being a non-functional, quality of life feature.

### *4. Critical Challenges and Solutions*

- **No login credentials**: Creating login tables in sql database by using ai tools to generate random numbers. Used "Select Distinct" command to pick only unique StudentID's from Attendance table.
- **Date Range**: In sql database,date fields are text formatted. However I came to realatiosn of that situation very late and I had to write a new method in StudentDisplay called reformatDate. What this method do is to change dd/mm/yyyy format into yyyy-mm-dd format in order to achieve search date ranges in SQL. However that was not enough and I also had to add a new line (ln 277, StudentDisplay) to also format results.

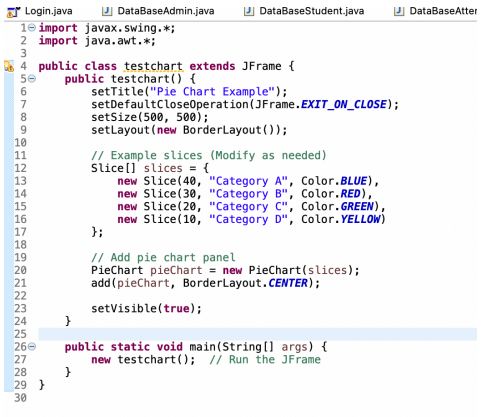
```
262                btnsearch = new JButton("Filter Dates");
263                btnsearch.setFont(new Font("Lucida Grande", Font.BOLD, 13));
264                btnsearch.setForeground(Color.BLUE);
265                btnsearch.setBounds(6, 519, 141, 35);
266                studentInfoPanel.add(btnsearch);
267                btnsearch.addActionListener(new ActionListener() {
268                    public void actionPerformed(ActionEvent e) {
269                        int attendedgetter = 0, lategetter = 0, absentgetter = 0;
270                        //TWO dates are entered:
271                        if (DateField1.getSelectedDate()!=null && DateField2.getSelectedDate()!=null) {
272                            //issue with only 'Having 2 dates". SQl database dates are "text" formats. Comparing is not possible. Solution is to format user inputs in SQL query.
273                                String formattedDateField1 = reformatDate(DateField1.getSelectedDate()) ;
274                                String formattedDateField2 = reformatDate(DateField2.getSelectedDate()) ;
275                                try (Connection conn = DriverManager.getConnection("jdbc:sqlite:/Users/oguzcayir/eclipse-workspace/ProjectAttendanceDataBase.db");
276                                        PreparedStatement stmt = conn.prepareStatement("SELECT * FROM Attendance " +
277                                                "WHERE (substr(SessionDate, 7, 4) || '-' || substr(SessionDate, 4, 2) || '-' || substr(SessionDate, 1, 2)) " +
278                                                "BETWEEN '" + formattedDateField1+"' AND '"+formattedDateField2+"' " +
279                                                "AND StudentID = '" + Login.loggedInStudentID + "'" )  ){
280
281                                    ResultSet rs =  stmt.executeQuery();
```

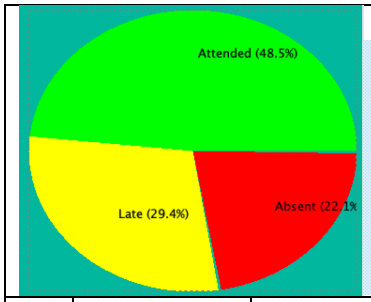
-**Allowing Unauthorised Users**: Before adding the boolean values in DatabaseAdmin and DataBaseStudent, system was allowing users to proceed next page without checking confirmation. Therefore, I added IsAdminValid and IsStudentValid methods to only allow if this values are true.

- **Pie Chart**: To avoid relying on external libraries, I chose to develop a custom pie chart implementation using core Java functionalities. Initially, this task presented a significant learning curve, as I had limited prior experience in manual chart rendering. After extensive research, I referenced a foundational solution from Stack Overflow (*https://stackoverflow.com/questions/13662984/creating-pie-charts-programmatically*) and utilized AI tools to refine the code.

# 5 TESTING

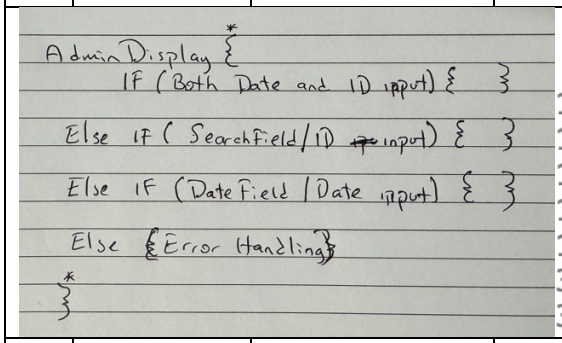
How did I test my program?

| ID | Component | Description | Corresponding Requirements | Test Data / Input | Expected Output | Pass/Fail | Correction? |
|---|---|---|---|---|---|---|---|
| 1 | Pie Chart | Verifying PieChart class works as expected. | 2,3,5 | Created a test class with main to specifically run only PieChart class under some static conditions (fixed values). | Creating 4 slices and displaying %40 blue “Category A”, %30 red “Category B”… | ✓ | - |
|  | | | | | | | |
| 2 | Pie Chart | Overflowing text on charts. | 2,3,5 | Depending on the input numbers, “Absent “and “Attended” overflows and partially disappears (out of panel) | Wrap portion of the text (percentages) below. Or changing the font size and style. | ✗ | None of the options worked. Therefore, rearranged the size of the panel in StudentDisplay and AdminDisplay classes. Partially worked. |




```
53          double angle = Math.toRadians(startAngle + arcAngle / 2.0);
54          int labelX = area.x + area.width / 2 + (int) (area.width / 3 * Math.cos(angle));
55          int labelY = area.y + area.height / 2 - (int) (area.height / 3 * Math.sin(angle));
56
57          g.setColor(Color.BLACK);
58          g.drawString(slice.label + "\n (" + String.format("%.1f", (slice.value / total) * 100) + "%)", labelX, labelY);
59          g.setFont( new Font("Lucida Grande", Font.PLAIN, 10));
60
61          curValue += slice.value;
62          }
63          }
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | Admin Display | Planning for search function. | 5 | Initially implemented only 3 conditions and built fundamentals. Adding pie-chart-maker method on later stages. | Functionality for this 3 conditions and error handling. | ✓ | - |

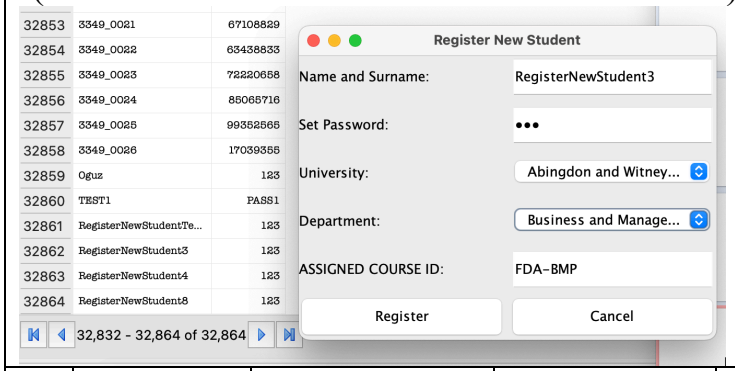


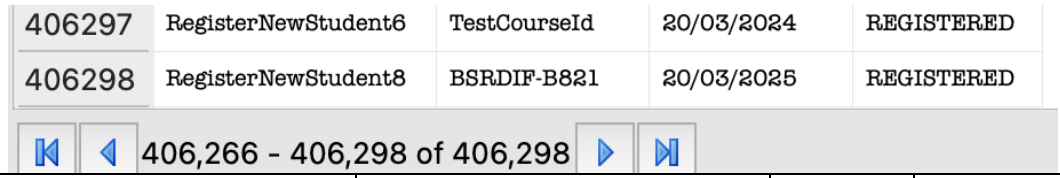

```
117         JButton btnsearch = new JButton("Search");
118         btnsearch.setBounds(1111, 21, 149, 35);
119         contentPane.add(btnsearch);
120         btnsearch.addActionListener(new ActionListener() {
121             public void actionPerformed(ActionEvent e) {..
339         });
340
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | Register New Student | While registering new students in for login, also creating a record for attendance. | 6 | Insert samples both Attendance and student login credential area. | Tested and had successful results for different types | ✓ | - |

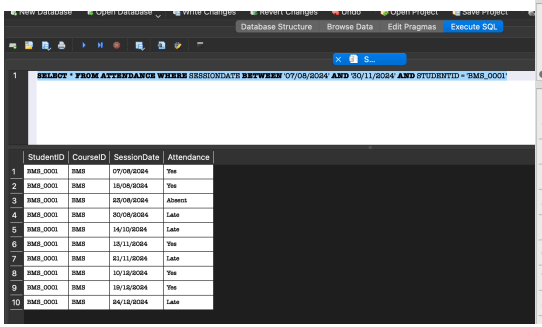
```
86  ection("jdbc:sqlite:/Users/oguzcayir/eclipse-workspace/ProjectAttendanceDataBase.db");
87  tement("INSERT INTO Unique_Students_With_Passwords (StudentID, Password) VALUES ('" + nameField.getText()+  "','" +passwordField.getText() + "')");
88  atement("INSERT INTO ATTENDANCE (StudentID, COURSEID, SESSIONDATE,ATTENDANCE) VALUES ('" + nameField.getText()+  "','" +AssignedCourseID.getText() + "','" +formattedDate +"','" +"REGISTERED')");
89
90
```

-(Local date needed to be formatted before use.)




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | Student Display | Date Range | 3 | For double date input use date-formatting. | Works as expected. | ✓ | Date area is considered text in sql. Had to format both search and inputs. |



```
SELECT * FROM Attendance
WHERE (substr(SessionDate, 7, 4) || '-' || substr(SessionDate, 4, 2) || '-' || substr(SessionDate, 1, 2))
BETWEEN '2020-01-01' AND '2024-11-30'
AND StudentID = 'BMS_0001';
```

| | StudentID | CourseID | SessionDate | Attendance |
|---|---|---|---|---|
| 1 | BMS_0001 | BMS | 07/08/2024 | Yes |
| 2 | BMS_0001 | BMS | 15/08/2024 | Yes |
| 3 | BMS_0001 | BMS | 23/08/2024 | Absent |
| 4 | BMS_0001 | BMS | 30/08/2024 | Late |
| 5 | BMS_0001 | BMS | 14/10/2024 | Late |
| 6 | BMS_0001 | BMS | 06/11/2024 | Yes |
| 7 | BMS_0001 | BMS | 13/11/2024 | Yes |
| 8 | BMS_0001 | BMS | 21/11/2024 | Late |

Only solution is to format both results ( *WHERE (substr(SESSIONDATE, 7, 4) || '-' || substr(SESSIONDATE, 4, 2) || '-' || substr(SESSIONDATE, 1, 2))* and input (reformattedDate method)

Solution:

SELECT * FROM ATTENDANCE

***WHERE date(substr(SESSIONDATE, 7, 4) || '-' || substr(SESSIONDATE, 4, 2) || '-' || substr(SESSIONDATE, 1, 2))***

BETWEEN '2024-08-07' AND '2024-11-30'

AND STUDENTID = 'BMS_0001';

```
public static String reformatDate(String dateDDMMYYYY) {
    DateTimeFormatter inputFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    DateTimeFormatter outputFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    LocalDate date = LocalDate.parse(dateDDMMYYYY, inputFormatter);
    return date.format(outputFormatter);
}
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | Admin Display | Pie chart and Logo overlaps upon search. | 4,5 | Use SetVisible (false) for logo for each time pie chart is created. | Works as expected. | ✓ | - |

.

# 6 Conclusion

The development of the Attendance Monitoring component has been a challenging journey, however having a great impact on my understanding of object-oriented programming (OOP) and the realities of real-world software development. While translating design concepts into functional code, I encountered unforeseen challenges such as balancing between design limitations and creativity. These hurdles taught me that software development is rarely straightforward, especially when working within team-defined boundaries or accommodating idealistic designs.

Despite falling short on some features, this project has brought me a huge confidence to tackle larger, more complex systems. I also learned critical lessons the hard way: jumping straight into coding without planning leads to avoidable mistakes and time-consuming revisions. Yet, I am deeply proud of what I achieved—and even what I could have achieved with more experience and time. The process revealed gaps in my skillset but also highlighted my ability to adapt, research, and persevere.

A unique aspect of this project is the transparency in my code. I intentionally left behind comments, notes, and even reworked code snippets to document my iterative thought process and errors. While such  might be streamlined in a professional setting, they serve as a record of my growth—proof that mistakes are not failures but stepping stones to mastery.

Looking ahead, I aim to build on this foundation by embracing structured planning, refining collaborative workflows, and exploring advanced OOP paradigms. This experience has not only deepened my technical proficiency but also reinforced the value of resilience.

# 7 References


1. **Oracle (Java JFrame Documentation)**
   Oracle (2025) *JFrame (Java Platform SE 8)*. Available at: https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html .
2. **Oracle (JDBC Tutorial)**
   Oracle (2025) *Processing SQL Statements with JDBC*. Available at: https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html .
3. **Oracle (JScrollPane Documentation)**
   Oracle (2025) *JScrollPane (Java Platform SE 8)*. Available at: https://docs.oracle.com/javase/8/docs/api/javax/swing/JScrollPane.html .
4. **SQLite Documentation**
   SQLite (2025) *SQLite Home Page*. Available at: https://www.sqlite.org .
5. **Stack Overflow (Label Wrapping Solution)**
   Stack Overflow (2025) *Make a JLabel wrap its text by setting a max width*. Available at: https://stackoverflow.com/questions/2420742/make-a-jlabel-wrap-its-text-by-setting-a-max-width .

# DataBaseAdmin

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.JOptionPane;

public class DataBaseAdmin {

     //Implemented later on. Solution to problem: I need to
check the validity, because otherwise even if the admin login
is wrong, it still allows to login.
     static boolean x= false;


     public static void ReadFromDB(String admin_name, String
admin_password) throws SQLException {

          String driver = "jdbc:sqlite";
          String db = "src/ProjectAttendanceDataBase.db";
          String url = driver + ":" + db;
          Connection c = DriverManager.getConnection(url);

          System.out.println("Connection to db\n");

          String sql = "SELECT * FROM [ADMINS] WHERE [Admin
Name] LIKE " + "'" + admin_name + "'" + "AND [Admin Password]
IS " + "'" + admin_password + "'";


          PreparedStatement stmnt = c.prepareStatement(sql);
          ResultSet results = stmnt.executeQuery();

          x=false;

          if (results.next()) {
               String name = results.getString("Admin Name");
               System.out.println("Welcome "+name);
               x=true;
          }

          //Error message condition
          else  {
               JOptionPane.showMessageDialog(null, "Please
enter a valid ID or Password", "ErrorMessage", 0);
```

```
          }

          //Good practice and habit. Like closing
BufferedReader/Writers
          stmnt.close();
          c.close();

     }

     public static boolean IsAdminValid(){
          return x;

     }


}
```

# DataBaseStudent

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.JOptionPane;

public class DataBaseStudent {

     //Implemented later on. Solution to problem: I need to
check the validity, because otherwise even if the admin login
is wrong, it still allows to login.
     static boolean x= false;


     public static void ReadFromDB(String studentID_in, String
studentPassword_in) throws SQLException {

          String driver = "jdbc:sqlite";
          String db = "src/ProjectAttendanceDataBase.db";
          String url = driver + ":" + db;
          Connection c = DriverManager.getConnection(url);

          System.out.println("Connection to db\n");

          String sql = "SELECT * FROM
[UNIQUE_STUDENTS_WITH_PASSWORDS] WHERE [StudentID] = " + "'" +
studentID_in + "'" + "AND [Password] IS " + "'" +
studentPassword_in + "'";


          PreparedStatement stmnt = c.prepareStatement(sql);
```

```
          ResultSet results = stmnt.executeQuery();

          x=false;

          if (results.next()) {
               String name = results.getString("StudentID");
               System.out.println("Welcome "+name);
               x=true;
          }

          //Error message condition
          else {
               JOptionPane.showMessageDialog(null, "Invalid ID
or Password. (Case Sensitive)", "ErrorMessage", 0);
          }

          //Good practice and habit. Like closing
BufferedReader/Writers
          stmnt.close();
          c.close();
     }

     public static boolean isStudentValid(){
          return x;

     }

}
```

# Login

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import java.awt.Font;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Login extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField ID;
    private JPasswordField Password;
    private JButton btnAdminLogin;
```

```
    private JButton btnStudentLogin;

    public static String loggedInStudentID = ""; //it will
help to identify student in StudentDisplay class
    public static String loggedInAdminID = "";

    //Main for login. / Start the app
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Login frame = new Login();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }


    public Login() {

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 500, 380);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        // Buttons for Admin or Student first
        btnAdminLogin = new JButton("Admin Login");
        btnAdminLogin.setBounds(170, 120, 160, 40);
        btnAdminLogin.setFont(new Font("Tahoma", Font.PLAIN,
16));
        contentPane.add(btnAdminLogin);

        btnStudentLogin = new JButton("Student Login");
        btnStudentLogin.setBounds(170, 200, 160, 40);
        btnStudentLogin.setFont(new Font("Tahoma", Font.PLAIN,
16));
        contentPane.add(btnStudentLogin);

        // Action listeners for the login buttons. So I
condensed different classes into one
        btnAdminLogin.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showLoginForm("Admin");
            }
        });
```

```
        btnStudentLogin.addActionListener(new ActionListener()
{
            public void actionPerformed(ActionEvent e) {
                showLoginForm("Student");
            }
        });
    }

    // Function to show the login form based on Admin or
Student choice
    private void showLoginForm(String loginType) {
        // Clear the main page. Otherwise you can't out
everything on top of each other.
        contentPane.removeAll();
        contentPane.repaint();

        // Setup new login details for Admin or Student
        JLabel lblLoginType = new JLabel(loginType + "
Login"); //to clarify for user
        lblLoginType.setFont(new Font("Tahoma", Font.BOLD,
18));
        lblLoginType.setBounds(160, 30, 150, 30);
        contentPane.add(lblLoginType);

        //Labels for ID and Password
        JLabel lblID = new JLabel("Login ID");
        lblID.setFont(new Font("Tahoma", Font.PLAIN, 14));
        lblID.setBounds(130, 70, 80, 16);
        contentPane.add(lblID);

        JLabel lblPassword = new JLabel("Password");
        lblPassword.setFont(new Font("Tahoma", Font.PLAIN,
14));
        lblPassword.setBounds(130, 140, 80, 16);
        contentPane.add(lblPassword);

        // Create ID and Password fields
        ID = new JTextField();
        ID.setBounds(130, 90, 220, 26);
        contentPane.add(ID);
        ID.setColumns(10);

        Password = new JPasswordField();
        Password.setBounds(130, 160, 220, 26);
        contentPane.add(Password);
        Password.setColumns(10);

        //Login Button
        JButton btnLogin = new JButton("Login");
        btnLogin.setFont(new Font("Tahoma", Font.PLAIN, 16));
        btnLogin.setBounds(185, 200, 117, 29);
        contentPane.add(btnLogin);
```

```
        // Action for the Login Button
        btnLogin.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (loginType.equals("Admin")) {
                    // Validate Admin credentials
                    try {


    DataBaseAdmin.ReadFromDB(ID.getText(),(Password.getText()
));

                        if (DataBaseAdmin.IsAdminValid()) {
                         loggedInAdminID= ID.getText();
////store the ID to use in AdminDisplay
                            AdminDisplay frame = new
AdminDisplay();
                            frame.setVisible(true);
                            setVisible(false); //nice, now i
can hide the login panel.
                        } else {

JOptionPane.showMessageDialog(null, "Invalid Admin
Credentials", "Login Error", JOptionPane.ERROR_MESSAGE);
                        }
                    } catch (Exception e1) {
                        e1.printStackTrace();
                    }
                } else if (loginType.equals("Student")) {
                    // Validate Student credentials
                    try {

    DataBaseStudent.ReadFromDB(ID.getText(),(Password.getText
()));

                        if (DataBaseStudent.isStudentValid())
{
                            loggedInStudentID = ID.getText();
//store the ID to use in StudentDisplay hehehe
                            StudentDisplay frame = new
StudentDisplay();
                            frame.setVisible(true);
                            setVisible(false); //nice, now i
can hide the login panel.
                        } else {

JOptionPane.showMessageDialog(null, "Invalid Student
Credentials", "Login Error", JOptionPane.ERROR_MESSAGE);
                        }
                    } catch (Exception e2) {
                        e2.printStackTrace();
                    }
```

```
                }
            }
        });

        // Refresh the UI
        contentPane.revalidate();
        contentPane.repaint();
    }
}
```

# StudentDisplay

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class StudentDisplay extends JFrame {

    // UI Components
    private JPanel contentPane;
    private static JTable table;
    private JTextArea resultArea;
    private JMenuBar menuBar;
    private PieChart PieChart;
    // Buttons
    private JButton btnsearch;
    private Button HomeButton, CourseSelectionButton,
ApplicationsButton, AttendanceButton,SportsSchoolButton,
RestaurantsButton, FinancesButton;
    // Course Information Fields
    private JLabel CourseNameLabel;
    private JTextField CourseIDDisplayBox,
TotalStudentsInCourseBox, CourseNameField1,CourseNameTitle2,
KISLevel3;
    // Title Label
    private Label UMSTitle;
    //New colours
    Color mycolor = new Color(60,181,160); //TysonTeal Ref.;
Brunel Colour Palette
    Color mycolor2 = new Color(227,234,246); //Mack Mist Ref.
Brunel Colour Palette
    private DatePicker DateField1;
    private DatePicker DateField2;
```

```
    public StudentDisplay() throws SQLException {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 1280, 1024);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);
        contentPane.setBackground(mycolor);

        resultArea = new JTextArea(10, 30);
        resultArea.setEditable(false);
        getContentPane().add(new JScrollPane(resultArea));

        table = new JTable();
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setBounds(289, 688, 985, 300);
        contentPane.add(scrollPane);
        scrollPane.setViewportView(table);

        menuBar = new JMenuBar();
        menuBar.setLayout(new BoxLayout(menuBar,
BoxLayout.PAGE_AXIS));
        menuBar.setBounds(0, 0, 277, 996);
        menuBar.setBackground(mycolor2);

        contentPane.add(menuBar);

        String x = "UNIVERSITY MANAGEMENT";
        UMSTitle = new Label(("<html>"+ x +"</html>"));
        UMSTitle.setAlignment(Label.CENTER);
        UMSTitle.setForeground(new Color(0, 42, 83));
        UMSTitle.setFont(new Font("Dialog", Font.BOLD, 20));
        menuBar.add(UMSTitle);


        HomeButton = new Button("Home");
        HomeButton.setActionCommand("Home");
        menuBar.add(HomeButton);

        CourseSelectionButton = new Button("Course
Selection");
        CourseSelectionButton.setActionCommand("Course
Selection");
        menuBar.add(CourseSelectionButton);

        ApplicationsButton = new Button("Applications");
        ApplicationsButton.setActionCommand("Applications");
        menuBar.add(ApplicationsButton);

        AttendanceButton = new Button("Attendance");
```

```
            AttendanceButton.setActionCommand("Attendance");
            menuBar.add(AttendanceButton);

            SportsSchoolButton = new Button("Sports School");

        SportsSchoolButton.setActionCommand("SportsSchoolButton")
;
            menuBar.add(SportsSchoolButton);

            RestaurantsButton = new Button("Restaurants");
            RestaurantsButton.setActionCommand("Restaurants");
            menuBar.add(RestaurantsButton);

            FinancesButton = new Button("Finances");
            FinancesButton.setActionCommand("Finances");
            menuBar.add(FinancesButton);

            //Implementing a different panel to display student
infos.

            JPanel studentInfoPanel = new JPanel();
            studentInfoPanel.setBounds(289, 15, 600, 660);
            contentPane.add(studentInfoPanel);
            studentInfoPanel.setBackground(mycolor2);
            studentInfoPanel.setLayout(null);

            JLabel WelcomeUserLabel = new JLabel("");
            WelcomeUserLabel.setFont(new Font("Lucida Grande",
Font.BOLD | Font.ITALIC, 30));
            WelcomeUserLabel.setBounds(6, 41, 408, 59);
            studentInfoPanel.add(WelcomeUserLabel);
            WelcomeUserLabel.setText("Welcome
"+Login.loggedInStudentID);

            JLabel logo = new JLabel("");
            logo.setBounds(352, 154, 201, 204);
            studentInfoPanel.add(logo);
            logo.setIcon(new
ImageIcon(getClass().getResource("/defaultprofile.png")));

            JLabel logo2 = new JLabel("");
            logo2.setBounds(1004, 15, 190, 250);
            contentPane.add(logo2);
            logo2.setIcon(new
ImageIcon(getClass().getResource("/bruneloldlogo.png")));

            CourseNameLabel = new JLabel("Course ID");
            CourseNameLabel.setFont(new Font("Lucida Grande",
Font.PLAIN, 15));
            CourseNameLabel.setBounds(6, 112, 141, 30);
            studentInfoPanel.add(CourseNameLabel);
```

```
            CourseIDDisplayBox = new JTextField();
            CourseIDDisplayBox.setEditable(false);
            CourseIDDisplayBox.setBounds(6, 154, 282, 46);
            studentInfoPanel.add(CourseIDDisplayBox);
            CourseIDDisplayBox.setColumns(10);
            CourseIDDisplayBox.setText(""); //get the data in
here.

            //Initially Setting searchbox results. It is
different from AdminDisplay because it is independent from
mouselistener and a table
            Connection conn1 =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
            PreparedStatement stmt1 =
conn1.prepareStatement("SELECT COURSEID FROM ATTENDANCE WHERE
STUDENTID = '" + Login.loggedInStudentID +"'");
            ResultSet rs1 = stmt1.executeQuery();;
            CourseIDDisplayBox.setText(rs1.getString(1));

            JLabel TotalStudentsInCourseLabel = new
JLabel("Total Students In Course");
            TotalStudentsInCourseLabel.setFont(new Font("Lucida
Grande", Font.PLAIN, 15));
            TotalStudentsInCourseLabel.setBounds(6, 212, 201,
30);
            studentInfoPanel.add(TotalStudentsInCourseLabel);

            TotalStudentsInCourseBox = new JTextField();
            TotalStudentsInCourseBox.setText("");
            TotalStudentsInCourseBox.setEditable(false);
            TotalStudentsInCourseBox.setColumns(10);
            TotalStudentsInCourseBox.setBounds(6, 254, 282, 46);
            studentInfoPanel.add(TotalStudentsInCourseBox);
            //SETTING TEXT BUT it is different from Admin so, no
mouselistener needed.
            int numberofrecords = 0;
            try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                        PreparedStatement stmt =
conn.prepareStatement("SELECT DISTINCT STUDENTID FROM
ATTENDANCE WHERE COURSEID =" + "'" +
CourseIDDisplayBox.getText()  + "'" ) ){

                ResultSet rs = stmt.executeQuery();

                while (rs.next()) {
                    numberofrecords +=1;
                }

            } catch (SQLException e1) {
```

```
                e1.printStackTrace();
            };
            int totalstudentsincourse = numberofrecords;
            TotalStudentsInCourseBox.setText(String.valueOf(
totalstudentsincourse)); // Display in text field
            //SETTING TEXT BUT it is different from Admin so, no
mouselistener needed.//END

            JLabel lblCourseName = new JLabel("Course Name");
            lblCourseName.setFont(new Font("Lucida Grande",
Font.PLAIN, 15));
            lblCourseName.setBounds(6, 323, 141, 30);
            studentInfoPanel.add(lblCourseName);

            CourseNameField1 = new JTextField();
            CourseNameField1.setText("");
            CourseNameField1.setEditable(false);
            CourseNameField1.setColumns(10);
            CourseNameField1.setBounds(6, 365, 282, 35);
            studentInfoPanel.add(CourseNameField1);

            //setting text from database search
            String OutputLegalName="";
            try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                        PreparedStatement stmt =
conn.prepareStatement("SELECT LEGAL_NAME FROM
CombinedCourseAndInstitution WHERE KISCOURSEID = " +"'"+
CourseIDDisplayBox.getText()+"'" )  ){

                ResultSet rs = stmt.executeQuery();

                OutputLegalName = rs.getString(1);

            }
            CourseNameField1.setText(OutputLegalName);

            CourseNameTitle2 = new JTextField();
            CourseNameTitle2.setText("");
            CourseNameTitle2.setEditable(false);
            CourseNameTitle2.setColumns(10);
            CourseNameTitle2.setBounds(6, 412, 282, 35);
            studentInfoPanel.add(CourseNameTitle2);

            //setting text from database search
            String OutputTitle = "";
            try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                        PreparedStatement stmt =
conn.prepareStatement("SELECT TITLE FROM
```

```
CombinedCourseAndInstitution WHERE KISCOURSEID = " +"'"+
CourseIDDisplayBox.getText()+"'" )  ){

                ResultSet rs = stmt.executeQuery();

                OutputTitle = rs.getString(1);

            }
            CourseNameTitle2.setText(OutputTitle);


            KISLevel3 = new JTextField();
            KISLevel3.setText("");
            KISLevel3.setEditable(false);
            KISLevel3.setColumns(10);
            KISLevel3.setBounds(6, 459, 282, 35);
            studentInfoPanel.add(KISLevel3);

            //setting text from database search
            String OutputLength = "";
            try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                    PreparedStatement stmt =
conn.prepareStatement("SELECT KISLEVEL FROM
CombinedCourseAndInstitution WHERE KISCOURSEID = " +"'"+
CourseIDDisplayBox.getText()+"'" )  ){

                ResultSet rs = stmt.executeQuery();

                OutputLength = rs.getString(1);

            }
            KISLevel3.setText("Length of Course:
"+OutputLength);

                    DateField1 = new DatePicker();
                    DateField1.setBounds(6, 554, 282, 35);
                    studentInfoPanel.add(DateField1);
                    DateField1.setBackground(mycolor2);
                    DateField1.setLayout(new GridLayout(1, 0,
0, 0));

                    DateField2 = new DatePicker();
                    DateField2.setBounds(6, 591, 282, 35);
                    studentInfoPanel.add(DateField2);
                    DateField2.setBackground(mycolor2);
                    DateField2.setLayout(new GridLayout(1, 0,
0, 0));

                    JButton clearDateButton = new
JButton("Clear Dates");
```

```
                    clearDateButton.setFont(new Font("Lucida
Grande", Font.BOLD, 13));
                    clearDateButton.setForeground(Color.RED);
                    clearDateButton.setBounds(159, 519, 129,
35);
                    studentInfoPanel.add(clearDateButton);
                    clearDateButton.addActionListener(e -> {
                        DateField1.clearSelectedDate(); //
Clear the selected dates -> method comes from DatePicker
class.
                        DateField2.clearSelectedDate();
                        try {
                              AttenadanceTable();
                         } catch (SQLException e1) {
                              // TODO Auto-generated catch
block
                              e1.printStackTrace();
                         }
                    });


                    btnsearch = new JButton("Filter Dates");
                    btnsearch.setFont(new Font("Lucida
Grande", Font.BOLD, 13));
                    btnsearch.setForeground(Color.BLUE);
                    btnsearch.setBounds(6, 519, 141, 35);
                    studentInfoPanel.add(btnsearch);
                    btnsearch.addActionListener(new
ActionListener() {
                         public void
actionPerformed(ActionEvent e) {
                              int attendedgetter = 0,
lategetter = 0, absentgetter = 0;
                              //TWO dates are entered:
                              if
(DateField1.getSelectedDate()!=null &&
DateField2.getSelectedDate()!=null) {
                                   //issue with only 'Having 2
dates". SQl database dates are "text" formats. Comparing is
not possible. Solution is to format user inputs in SQL query.
                                   String
formattedDateField1 =
reformatDate(DateField1.getSelectedDate()) ;
                                   String
formattedDateField2 =
reformatDate(DateField2.getSelectedDate()) ;
                                   try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");

     PreparedStatement stmt = conn.prepareStatement("SELECT *
FROM Attendance " +
```

```
                                                                "WHERE
(substr(SessionDate, 7, 4) || '-' || substr(SessionDate, 4, 2)
|| '-' || substr(SessionDate, 1, 2)) " +

"BETWEEN '" + formattedDateField1+"' AND
'"+formattedDateField2+"' " +
                                                                "AND
StudentID = '" + Login.loggedInStudentID + "'" )  ){

                                                ResultSet rs =
stmt.executeQuery();

                                                DefaultTableModel
model = new DefaultTableModel(){
                                                    @Override
                                                    public
boolean isCellEditable(int row, int column) {
                                                        return
false; // Make all cells non-editable
                                                    }
                                                };


     model.addColumn("STUDENTID");

     model.addColumn("COURSEID");

     model.addColumn("SESSIONDATE");

     model.addColumn("ATTENDANCE");

                                                if (!rs.next()) {

     JOptionPane.showMessageDialog(null, "No records found
between this dates.", "ErrorMessage", 0);
                                                }

     resultArea.setText(""); // Clear previous results
                                                while (rs.next())
{

                                                    String
studentid = rs.getString("STUDENTID");
                                                    String
courseid = rs.getString("COURSEID");
                                                    String
sessiondate = rs.getString("SESSIONDATE");
                                                    String
attendance = rs.getString("ATTENDANCE");

                                                    Object []
row = {studentid, courseid,sessiondate,attendance};
```

```
     model.addRow(row);

                                                    //that
should help to create pie chart later on.
                                                    if
(attendance.equalsIgnoreCase("Yes")) {

attendedgetter += 1;
                                                    } else if
(attendance.equalsIgnoreCase("Late")) {

lategetter += 1;
                                                    } else if
(attendance.equalsIgnoreCase("Absent")) {

absentgetter += 1;
                                                    }
                                                }

     table.setModel(model);

                                            } catch (SQLException
e1) {
                                                // TODO Auto-
generated catch block

     e1.printStackTrace();
                                            }


                                        }



                                    //ONE date is entered
                                    else if
(DateField1.getSelectedDate()!=null &&
DateField2.getSelectedDate()==null) {

                                        try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                                                PreparedStatement
stmt = conn.prepareStatement("SELECT * FROM ATTENDANCE WHERE
SESSIONDATE = '" + DateField1.getSelectedDate() + "'"+ " AND
STUDENTID = '" + Login.loggedInStudentID + "'" )  ){

                                            ResultSet rs =
stmt.executeQuery();
```

```
                                        DefaultTableModel
model = new DefaultTableModel(){
                                            @Override
                                            public boolean
isCellEditable(int row, int column) {
                                                return false;
// Make all cells non-editable
                                            }
                                        };


     model.addColumn("STUDENTID");

     model.addColumn("COURSEID");

     model.addColumn("SESSIONDATE");

     model.addColumn("ATTENDANCE");

                                        if (!rs.next()) {

     JOptionPane.showMessageDialog(null, "No record found on
this date.", "ErrorMessage", 0);
                                        }

     resultArea.setText(""); // Clear previous results
                                        while (rs.next()) {

                                            String studentid
= rs.getString("STUDENTID");
                                            String courseid =
rs.getString("COURSEID");
                                            String
sessiondate = rs.getString("SESSIONDATE");
                                            String attendance
= rs.getString("ATTENDANCE");

                                            Object [] row =
{studentid, courseid,sessiondate,attendance};


     model.addRow(row);

                                            //that should
help to create pie chart later on.
                                            if
(attendance.equalsIgnoreCase("Yes")) {

attendedgetter += 1;
                                            } else if
(attendance.equalsIgnoreCase("Late")) {
```

```
                                                    lategetter +=
1;
                                               } else if
(attendance.equalsIgnoreCase("Absent")) {
                                                    absentgetter
+= 1;
                                               }

                                          }
                                          table.setModel(model);

                                     } catch (SQLException e1) {
                                          // TODO Auto-generated
catch block
                                          e1.printStackTrace();
                                     }
                                }

                                else { //If user only fills
second field but not the first one

     JOptionPane.showMessageDialog(null, "To search specific
date; please use only the first field ", "ErrorMessage", 0);
                                }
                                //PieChartMaker data
                                if (attendedgetter > 0 ||
lategetter > 0 || absentgetter > 0) {
                                   PieChartMaker(attendedgetter,
lategetter, absentgetter);
                               } //Make a piechart after the
search function and table-creation has been initialized
                           }

                     });



          //LAUNCH ALL ATTAENDANCE ON ENTRY
          AttenadanceTable();

     }

     //METHODS

     //Update: Not valid anymore, i will use DateField class
instead of textfield for date input from user.
     //w3 school SimpleDataFormat reference
     private boolean isValidDate(String date) {
          SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yyyy");
          sdf.setLenient(false);
```

```
          //Explanation: as default Lenient is true. And dates
are auto-corrected. Therefore we have to set it false. So if
someone inputs 30th of February It gives an error. If Lenient
would be true, It would fix it as 1 March.
          //That is a problem because we want to search EXACT
dates in our datebase.
          try {
               sdf.parse(date);
               return true;
          } catch (ParseException e) {
               return false;
          }
     }
     public void PieChartMaker (double attended, double late,
double absent) {
          //removing if there is any other component, so it
resets the pie chart.
          Component[] components =
contentPane.getComponents();
          for (Component component : components) {
               if (component instanceof JPanel &&
component.getName() != null &&
component.getName().equals("chartContainer")) { //3 conditions
to remove.
                    contentPane.remove(component);
                    break; // Remove only the first found
chart
               }
          }

          // Create slices
          Slice[] slices = {
                    new Slice(attended, Color.GREEN,
"Attended"), //first number is how many is attended. //it is
actually does not make sense t assign "double" type, because a
person can not be half.
                    new Slice(late, Color.YELLOW, "Late"),
                    new Slice(absent, Color.RED, "Absent")
          };

          // Create the PieChart
          PieChart pieChart = new PieChart(slices);

          // Create a container panel with BorderLayout to
handle resizing
          JPanel chartContainer = new JPanel();
          chartContainer.setLayout(new BorderLayout());
          chartContainer.setBounds(920, 320, 350, 350);  //
Position on AdminDisplay //If i want to reposition the chart,
it is this parameters.
          chartContainer.setBackground(mycolor);
          chartContainer.add(pieChart, BorderLayout.CENTER);
```

```
          chartContainer.setName("chartContainer"); // Set a
name to identify it later //Condition to remove piechart with
every search button action.

          // Add to contentPane
          contentPane.add(chartContainer);
          pieChart.setBackground(mycolor);
          contentPane.revalidate(); //Necessary to reload
(kind of) the pie chart panel
          contentPane.repaint();

     }
     public void AttenadanceTable () throws SQLException{
          int attendedgetter = 0, lategetter = 0, absentgetter
= 0;

          Connection conn1 =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
          PreparedStatement stmt1 =
conn1.prepareStatement("SELECT * FROM ATTENDANCE WHERE
STUDENTID = '" + Login.loggedInStudentID +"'");
          ResultSet rs1 = stmt1.executeQuery();;

          DefaultTableModel model = new DefaultTableModel(){
              @Override
              public boolean isCellEditable(int row, int
column) {
                  return false; // Make all cells non-editable
              }
          };

          model.addColumn("STUDENTID");
          model.addColumn("COURSEID");
          model.addColumn("SESSIONDATE");
          model.addColumn("ATTENDANCE");

          resultArea.setText(""); // Clear previous results
          while (rs1.next()) {

               String studentid = rs1.getString("STUDENTID");
               String courseid = rs1.getString("COURSEID");
               String sessiondate =
rs1.getString("SESSIONDATE");
               String attendance =
rs1.getString("ATTENDANCE");

               Object [] row = {studentid,
courseid,sessiondate,attendance};

               model.addRow(row);
```

```
                //that should help to create pie chart later
on.
                if (attendance.equalsIgnoreCase("Yes")) {
                    attendedgetter += 1;
                } else if (attendance.equalsIgnoreCase("Late"))
{
                    lategetter += 1;
                } else if
(attendance.equalsIgnoreCase("Absent")) {
                    absentgetter += 1;
                }
            }
            table.setModel(model);

            if (attendedgetter > 0 || lategetter > 0 ||
absentgetter > 0) {
                PieChartMaker(attendedgetter, lategetter,
absentgetter);
            } //Make a piechart after the search function and
table-creation has been initialized

            DateField1.clearSelectedDate(); //Upon login,
datefields set January 2025 for some reason.Solution; <-
            DateField2.clearSelectedDate();
        }
        //SQl database date formats are texts. In order to filter
the dates, i have to covert user inputs. (Only for BETWEEN
command.) Because having 2 dates and searching a "range" does
work otherwise.
        //This could have been sorted as changing the SQl
database completely and formatting as "date".
        public static String reformatDate(String dateDDMMYYYY) {
            DateTimeFormatter inputFormatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
            DateTimeFormatter outputFormatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd");
            LocalDate date = LocalDate.parse(dateDDMMYYYY,
inputFormatter);
            return date.format(outputFormatter);
        }
}
```

---

# AdminDisplay

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
```

```
import javax.swing.table.DefaultTableModel;

public class AdminDisplay extends JFrame {

     private static final long serialVersionUID = 1L;
    private static JPanel contentPane;
    private static JTable table;
    private JTextField searchfield;
    private static DatePicker DateField;
    private JLabel
EnterDateLabel,CourseLengthLabel,CourseIDLabel;
    private JButton btnsearch;
    private JTextArea resultArea;
    private JMenuBar menuBar;
    private Button HomeButton, CourseSelectionButton,
ApplicationsButton, AttendanceButton, SportsSchoolButton,
RestaurantsButton, FinancesButton, RegisterStudentButton;
    private Label UMSTitle;
    private PieChart PieChart;
    private JLabel CourseNameLabel;
    private JTextField CourseID,
TotalStudentsInCourseBox,CourseNameBox,CourseNameBox2,Lengthof
Course;
    private boolean isDateSet = false; // Adding flag to track
if the date is explicitly set
    private static JLabel logo;
     Color mycolor = new Color(60,181,160); //Tyson Teal.
Ref.; Brunel Colour Palette
     Color mycolor2 = new Color(227,234,246); //Mack Mist.
Ref.; Brunel Colour Palette


     public AdminDisplay() {
          //1) Main UI
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          setBounds(100, 100, 1280, 1024);
          contentPane = new JPanel();
          contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

          setContentPane(contentPane);
          contentPane.setLayout(null);
          contentPane.setBackground(mycolor);

          resultArea = new JTextArea(10, 30);
          resultArea.setEditable(false);
          getContentPane().add(new JScrollPane(resultArea));

          //2) Menu Bar
          menuBar = new JMenuBar();
          menuBar.setLayout(new BoxLayout(menuBar,
BoxLayout.PAGE_AXIS));
          menuBar.setBounds(0, 0, 277, 996);
```

```
            menuBar.setBackground(mycolor2);
            contentPane.add(menuBar);

            String x = "UNIVERSITY MANAGEMENT";
            UMSTitle = new Label(("<html>"+ x +"</html>"));
            UMSTitle.setAlignment(Label.CENTER);
            UMSTitle.setForeground(new Color(0, 42, 83));
            UMSTitle.setFont(new Font("Dialog", Font.BOLD, 20));
            menuBar.add(UMSTitle);

            HomeButton = new Button("Home");
            HomeButton.setBackground(Color.PINK);
            HomeButton.setActionCommand("Home");
            HomeButton.addActionListener(new ActionListener(){
                 public void actionPerformed(ActionEvent e) {
                      initialInfoDisplay();
                      //tried making piechart disappear, not
exactly worked, it is a problem.
                 }

            });
            menuBar.add(HomeButton);


            CourseSelectionButton = new Button("Course
Selection");
            CourseSelectionButton.setActionCommand("Course
Selection");
            menuBar.add(CourseSelectionButton);

            ApplicationsButton = new Button("Applications");
            ApplicationsButton.setActionCommand("Applications");
            menuBar.add(ApplicationsButton);

            AttendanceButton = new Button("Attendance");
            AttendanceButton.setActionCommand("Attendance");
            menuBar.add(AttendanceButton);

            SportsSchoolButton = new Button("Sports School");

        SportsSchoolButton.setActionCommand("SportsSchoolButton")
;
            menuBar.add(SportsSchoolButton);

            RestaurantsButton = new Button("Restaurants");
            RestaurantsButton.setActionCommand("Restaurants");
            menuBar.add(RestaurantsButton);

            FinancesButton = new Button("Finances");
            FinancesButton.setActionCommand("Finances");
            menuBar.add(FinancesButton);
```

```
          RegisterStudentButton = new
Button("<html>"+"Register New Student"+"</html>");
          RegisterStudentButton.setBackground(Color.PINK);

     RegisterStudentButton.setActionCommand("RegisterNewStuden
t");
          menuBar.add(RegisterStudentButton);
          RegisterStudentButton.addActionListener(new
ActionListener (){
               public void actionPerformed(ActionEvent e)  {
                    RegistrationForm frame = new
RegistrationForm();
                    frame.setVisible(true);

               }
          });

          //3) Search Button And Search Fields
          searchfield = new JTextField();
          searchfield.setBounds(397, 21, 180, 35);
          searchfield.setBackground(mycolor2);
          contentPane.add(searchfield);
          searchfield.setColumns(10);

          JButton btnsearch = new JButton("Search");
          btnsearch.setBounds(1111, 21, 149, 35);
          contentPane.add(btnsearch);
          btnsearch.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {

                    //3.1) Date AND ID search TOGETHER
                    if (DateField.getSelectedDate()!=null &&
!searchfield.getText().isEmpty()) {
                         int attendedgetter = 0;
                         int lategetter =0;
                         int absentgetter = 0;

// if (isValidDate(DateTextField.getSelectedDate())) {

                              try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                                        PreparedStatement stmt
= conn.prepareStatement("SELECT * FROM ATTENDANCE WHERE
SESSIONDATE = " + "'" + DateField.getSelectedDate() + "'" + "
AND " + "[STUDENTID] LIKE " + "'%" + searchfield.getText() +
"%'") ){

                                   ResultSet rs =
stmt.executeQuery();
```

```
                                    DefaultTableModel model =
new DefaultTableModel()
                                    {
                                        @Override
                                        public boolean
isCellEditable(int row, int column) {
                                            return false; //
Make all cells non-editable
                                        }
                                    };


     model.addColumn("STUDENTID");

     model.addColumn("COURSEID");

     model.addColumn("SESSIONDATE");

     model.addColumn("ATTENDANCE");

                                    if (!rs.next()) {

     JOptionPane.showMessageDialog(null, "No records found.",
"ErrorMessage", 0);
                                    }

                                    resultArea.setText(""); //
Clear previous results

                                    while (rs.next()) {

                                         String studentid =
rs.getString("STUDENTID");

                                         String courseid =
rs.getString("COURSEID");

                                         String sessiondate =
rs.getString("SESSIONDATE");

                                         String attendance =
rs.getString("ATTENDANCE");


                                         Object [] row =
{studentid, courseid,sessiondate,attendance};

                                         model.addRow(row);

                                         //that should help to
create pie chart later on.

                                         if
(attendance.equalsIgnoreCase("Yes")) {

                                             attendedgetter +=
1;

                                         } else if
(attendance.equalsIgnoreCase("Late")) {
```

```
                                        lategetter += 1;
                                    } else if
(attendance.equalsIgnoreCase("Absent")) {
                                        absentgetter += 1;
                                    }

                                }
                                table.setModel(model);

                            }
                            catch (SQLException e1){

     JOptionPane.showMessageDialog(null, "Data not found.");
                            }

//}
//else {
/* Update: isValidDate no longer needed. I changed it into
DatePicker
if (!isValidDate(DateTextField.getSelectedDate())) {
      JOptionPane.showMessageDialog(null, "Please enter a
valid date in dd/MM/yyyy format.");
      return;
}
*/
//}

                        //if all 0, it doesn't create pie
chart but text labels are printed from PieChart class. SO it
is the solution;
                        if (attendedgetter > 0 || lategetter
> 0 || absentgetter > 0) {
                            PieChartMaker(attendedgetter,
lategetter, absentgetter);
                            logo.setVisible(false);
                        }
                    }


                    //3.2) This is for ID search ONLY
                    else if (DateField.getSelectedDate()==null
&& !searchfield.getText().isEmpty()) {
                        int attendedgetter = 0;
                        int lategetter =0;
                        int absentgetter = 0;

                        try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
```

```
                                    PreparedStatement stmnt =
conn.prepareStatement("SELECT * FROM ATTENDANCE WHERE
[STUDENTID] LIKE  " + "'%" + searchfield.getText() + "%'") ){

                                ResultSet results =
stmnt.executeQuery();

                                DefaultTableModel model = new
DefaultTableModel(){
                                    @Override
                                    public boolean
isCellEditable(int row, int column) {
                                        return false; // Make
all cells non-editable
                                    }
                                };

                                model.addColumn("STUDENTID");
                                model.addColumn("COURSEID");
                                model.addColumn("SESSIONDATE");
                                model.addColumn("ATTENDANCE");

                                if (!results.next()) {

     JOptionPane.showMessageDialog(null, "No records found.",
"ErrorMessage", 0);
                                }

                                while (results.next()) {
                                    String studentid =
results.getString("STUDENTID");
                                    String courseid =
results.getString("COURSEID");
                                    String sessiondate =
results.getString("SESSIONDATE");
                                    String attendance =
results.getString("ATTENDANCE");

                                    Object [] row = {studentid,
courseid,sessiondate,attendance};

                                    model.addRow(row);

                                    //that should help to
create pie chart later on.
                                    if
(attendance.equalsIgnoreCase("Yes")) {
                                        attendedgetter += 1;
                                    } else if
(attendance.equalsIgnoreCase("Late")) {
                                        lategetter += 1;
```

```
                                    } else if
(attendance.equalsIgnoreCase("Absent")) {
                                        absentgetter += 1;
                                    }
                                }

                                table.setModel(model);
                            }
                            catch (SQLException e1){

     JOptionPane.showMessageDialog(null, "Error. Please input
a valid ID");
                            }

                            if (attendedgetter > 0 || lategetter
> 0 || absentgetter > 0) {
                                PieChartMaker(attendedgetter,
lategetter, absentgetter);
                                logo.setVisible(false);
                            }
                        }

                        //3.3) This is for Date Search ONLY
                        else if (DateField.getSelectedDate()!=null
&& searchfield.getText().isEmpty()) {
                            int attendedgetter = 0;
                            int lategetter =0;
                            int absentgetter = 0;

//if (isValidDate(DateTextField.getSelectedDate())) {

                                //I forgot "jdbc:sqlite:" part
first. Note it to my Debugging file. Also i didn't know we
could use parameter () and write SQL connection inside.
                                //reference:
https://docs.oracle.com/javase/tutorial/jdbc/basics/processing
sqlstatements.html
                                try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                                                PreparedStatement stmt
= conn.prepareStatement("SELECT * FROM ATTENDANCE WHERE
SESSIONDATE =" + "'" + DateField.getSelectedDate() + "'" ) ){

                                    ResultSet rs =
stmt.executeQuery();

                                    DefaultTableModel model =
new DefaultTableModel(){
                                        @Override
                                        public boolean
isCellEditable(int row, int column) {
```

```
                                        return false; //
Make all cells non-editable
                                    }
                                };


     model.addColumn("STUDENTID");

     model.addColumn("COURSEID");

     model.addColumn("SESSIONDATE");

     model.addColumn("ATTENDANCE");

                                if (!rs.next()) {

     JOptionPane.showMessageDialog(null, "No records found.",
"ErrorMessage", 0);
                                }

                                resultArea.setText(""); //
Clear previous results
                                while (rs.next()) {

                                    String studentid =
rs.getString("STUDENTID");
                                    String courseid =
rs.getString("COURSEID");
                                    String sessiondate =
rs.getString("SESSIONDATE");
                                    String attendance =
rs.getString("ATTENDANCE");

                                    Object [] row =
{studentid, courseid,sessiondate,attendance};

                                    model.addRow(row);


                                    //that should help to
create pie chart later on.
                                    if
(attendance.equalsIgnoreCase("Yes")) {
                                        attendedgetter +=
1;
                                    } else if
(attendance.equalsIgnoreCase("Late")) {
                                        lategetter += 1;
                                    } else if
(attendance.equalsIgnoreCase("Absent")) {
                                        absentgetter += 1;
                                    }
```

```
                                    }
                                    table.setModel(model);

                                }
                                catch (SQLException e1){


     JOptionPane.showMessageDialog(null, "Invalid date
format.");
                                }
//}
//else {
//   JOptionPane.showMessageDialog(null, "Invalid date format.
Use YYYY-MM-DD.");
//}

                            if (attendedgetter > 0 || lategetter
> 0 || absentgetter > 0) {
                               PieChartMaker(attendedgetter,
lategetter, absentgetter);
                               logo.setVisible(false);
                           }

                    }

                    else {
                         JOptionPane.showMessageDialog(null,
"Invalid date format.");
                         return;
                    }


                }
            });

            //NOTE: Some of my sanity is lost throughout the
process of implementing action listener to the search button.
Kind Regards.

            table = new JTable();
            JScrollPane scrollPane = new JScrollPane();
            scrollPane.setBounds(289, 388, 985, 602);
            contentPane.add(scrollPane);
            scrollPane.setViewportView(table);

            JLabel WelcomeUserLabel = new JLabel("Welcome
"+Login.loggedInAdminID);
            WelcomeUserLabel.setFont(new Font("Lucida Grande",
Font.BOLD | Font.ITALIC, 30));
            WelcomeUserLabel.setBounds(289, 109, 408, 59);
            contentPane.add(WelcomeUserLabel);
```

```
          JLabel StudentIDlabel = new JLabel("Student ID");
          StudentIDlabel.setFont(new Font("Lucida Grande",
Font.PLAIN, 15));
          StudentIDlabel.setBounds(289, 23, 100, 30);
          contentPane.add(StudentIDlabel);



          EnterDateLabel = new JLabel("Filter Date");
          EnterDateLabel.setFont(new Font("Lucida Grande",
Font.PLAIN, 15));
          EnterDateLabel.setBounds(289, 65, 100, 30);
          contentPane.add(EnterDateLabel);
          /* //This is no longer functional, because i
replaced it with DatePicker class.
          DateTextField = new JTextField();
          DateTextField.setBounds(1069, 65, 179, 26);
          contentPane.add(DateTextField);
          DateTextField.setColumns(10);
          */
          DateField = new DatePicker();
          DateField.setBounds(397, 60, 295, 35);
        DateField.setLayout(new GridLayout(1, 0, 0, 0));
        DateField.setBackground(mycolor);
        contentPane.add(DateField);
        DateField.addFocusListener(new FocusAdapter() {
            @Override
            public void focusLost(FocusEvent e) {
                isDateSet =
!DateField.getSelectedDate().isEmpty(); // Set the flag if the
date is not empty

            }
        });

          //4) Student Information Panel. This area display
info from search function mostly.
          JPanel studentInfoPanel = new JPanel();
          studentInfoPanel.setBounds(289, 180, 600, 200);
          contentPane.add(studentInfoPanel);
          studentInfoPanel.setBackground(mycolor2);
          studentInfoPanel.setLayout(null);

          CourseNameLabel = new JLabel("Course Name");
          CourseNameLabel.setFont(new Font("Lucida Grande",
Font.PLAIN, 15));
          CourseNameLabel.setBounds(6, 6, 141, 30);
          studentInfoPanel.add(CourseNameLabel);

          CourseID = new JTextField();
          CourseID.setEditable(false);
          CourseID.setBounds(300, 48, 282, 46);
```

```
          studentInfoPanel.add(CourseID);
          CourseID.setColumns(10);
          CourseID.setText(""); //get the data in here.
          //Take the data from mouse click

          table.addMouseListener(new MouseAdapter() {

               public void mouseClicked(MouseEvent e) {

                    int selectedRow = table.getSelectedRow();
// Get selected row index
                    int columnIndex = 1; // Change this to the
column index where Course Name is stored

                    if (selectedRow != -1) { // Ensure a row
is selected
                         String selectedCourse =
table.getValueAt(selectedRow, columnIndex).toString(); // Get
course name
                         CourseID.setText(selectedCourse); //
Display in text field
                    }


               }
          });


          JLabel TotalStudentsInCourseLabel = new
JLabel("Total Students In Course");
          TotalStudentsInCourseLabel.setFont(new Font("Lucida
Grande", Font.PLAIN, 15));
          TotalStudentsInCourseLabel.setBounds(300, 106, 249,
30);
          studentInfoPanel.add(TotalStudentsInCourseLabel);

          TotalStudentsInCourseBox = new JTextField();
          TotalStudentsInCourseBox.setText("");
          TotalStudentsInCourseBox.setEditable(false);
          TotalStudentsInCourseBox.setColumns(10);
          TotalStudentsInCourseBox.setBounds(300, 148, 282,
46);
          studentInfoPanel.add(TotalStudentsInCourseBox);
          table.addMouseListener(new MouseAdapter() {
               public void mouseClicked(MouseEvent e) {

                    int selectedRow = table.getSelectedRow();
// Get selected row index

                    if (selectedRow != -1) { // Ensure a row
is selected
```

```
                        // I want to execute this : SELECT
DISTINCT STUDENTID FROM ATTENDANCE WHERE COURSEID = 'BMS'
                        int numberofrecords = 0;
                        try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                                        PreparedStatement stmt =
conn.prepareStatement("SELECT DISTINCT STUDENTID FROM
ATTENDANCE WHERE COURSEID =" + "'" + CourseID.getText()  + "'"
) ){

                                ResultSet rs =
stmt.executeQuery();

                                while (rs.next()) {
                                     numberofrecords +=1;
                                }

                        } catch (SQLException e1) {
                                e1.printStackTrace();
                        };
                        int totalstudentsincourse =
numberofrecords;

     TotalStudentsInCourseBox.setText(String.valueOf(
totalstudentsincourse)); // Display in
TotalStudentsInCourseBox field
                    }

                }
            }); //that is a wild counting system. God have
mercy. //That worked first time without error. :))

            JButton editaccessbutton = new JButton("Edit");
            editaccessbutton.setBounds(477, 9, 117, 29);
            studentInfoPanel.add(editaccessbutton);

            editaccessbutton.addActionListener(new
ActionListener() {
                 private String IDupdate, nameupdate,
nameupdate2;

                public void actionPerformed(ActionEvent e) {
                    if
(editaccessbutton.getText().equalsIgnoreCase("Edit")) {
                        IDupdate = CourseID.getText();
                        nameupdate =
CourseNameBox.getText();
                        nameupdate2 =
CourseNameBox2.getText();
                    // Switch to "Edit" mode
                    CourseID.setEditable(true);
```

```
                    CourseNameBox.setEditable(true);
                    CourseNameBox2.setEditable(true);
                    editaccessbutton.setText("Save"); //
Change button text to "Save"
                }


                else
if(editaccessbutton.getText().equalsIgnoreCase("Save")) {


                    String IDupdateD =
CourseID.getText();
                    String nameupdateD =
CourseNameBox.getText();
                    String nameupdate2D =
CourseNameBox2.getText();

                    if (IDupdateD.isEmpty() ||
nameupdateD.isEmpty() || nameupdate2D.isEmpty()) {
                     JOptionPane.showMessageDialog(null,
"Course name cannot be empty.");
                     return;
                    }

                    //updating real DataBase:
                    try(Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");){
                        try (PreparedStatement stmt =
conn.prepareStatement("UPDATE ATTENDANCE SET COURSEID =" +
"'"+ IDupdateD + "'" + " WHERE COURSEID = "+ "'" + IDupdate +
"'" ) ){

                            //Displaying message:
                            int rowsUpdated =
stmt.executeUpdate();
                            if (rowsUpdated > 0) {

    JOptionPane.showMessageDialog(null, "Course name updated
successfully.");
                            } else {

    JOptionPane.showMessageDialog(null, "No records were
updated. Please check the course name.");
                            }

                        }
                        try (PreparedStatement stmt =
conn.prepareStatement( "UPDATE CombinedCourseAndInstitution
SET LEGAL_NAME ='"+ nameupdateD+"', TITLE ='"+nameupdate2D+"'
```

```
WHERE LEGAL_NAME ='"+nameupdate+"' AND TITLE
='"+nameupdate2+"'" ) ){

                                        //Displaying message:
                                        int rowsUpdated =
stmt.executeUpdate();
                                        if (rowsUpdated > 0) {

     JOptionPane.showMessageDialog(null, "Course name updated
successfully.");
                                        } else {

     JOptionPane.showMessageDialog(null, "No records were
updated. Please check the course name.");
                                        }

                                   }
                              }
                              catch (SQLException e1) {
                                   e1.printStackTrace();

     JOptionPane.showMessageDialog(null, "Error updating
course name: " + e1.getMessage());
                              };

                              //Closing and setting things back to
false and Edit.
                              CourseID.setEditable(false);
                         CourseNameBox.setEditable(false);
                         CourseNameBox2.setEditable(false);
                         editaccessbutton.setText("Edit");
                    }
               }
          });

          LengthofCourse = new JTextField();
          LengthofCourse.setText("");
          LengthofCourse.setEditable(false);
          LengthofCourse.setColumns(10);
          LengthofCourse.setBounds(6, 148, 282, 46);
          studentInfoPanel.add(LengthofCourse);
          table.addMouseListener(new MouseAdapter() {
               public void mouseClicked(MouseEvent e) {
                    try(Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                                   PreparedStatement stmt =
conn.prepareStatement("SELECT KISLEVEL FROM
CombinedCourseAndInstitution WHERE KISCOURSEID =
'"+CourseID.getText() +"'" ) ){

                              ResultSet x = stmt.executeQuery();
```

```
                        LengthofCourse.setText("Course
Length: "+x.getString("KISLEVEL"));



                    } catch (SQLException e1) {
                        e1.printStackTrace();
                    }
                }

            });

            CourseLengthLabel = new JLabel("Course Length");
            CourseLengthLabel.setFont(new Font("Lucida Grande",
Font.PLAIN, 15));
            CourseLengthLabel.setBounds(6, 106, 249, 30);
            studentInfoPanel.add(CourseLengthLabel);

            CourseIDLabel = new JLabel("Course ID");
            CourseIDLabel.setFont(new Font("Lucida Grande",
Font.PLAIN, 15));
            CourseIDLabel.setBounds(300, 14, 165, 30);
            studentInfoPanel.add(CourseIDLabel);

            CourseNameBox = new JTextField();
            CourseNameBox.setText("");
            CourseNameBox.setEditable(false);
            CourseNameBox.setColumns(10);
            CourseNameBox.setBounds(6, 48, 282, 23);
            studentInfoPanel.add(CourseNameBox);

            CourseNameBox2 = new JTextField();
            CourseNameBox2.setText("");
            CourseNameBox2.setEditable(false);
            CourseNameBox2.setColumns(10);
            CourseNameBox2.setBounds(6, 71, 282, 23);

            studentInfoPanel.add(CourseNameBox2);
            table.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    try(Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
                        PreparedStatement stmt =
conn.prepareStatement("SELECT LEGAL_NAME FROM
CombinedCourseAndInstitution WHERE KISCOURSEID =
'"+CourseID.getText() +"'" );
                        PreparedStatement stmt2 =
conn.prepareStatement("SELECT TITLE FROM
CombinedCourseAndInstitution WHERE KISCOURSEID =
'"+CourseID.getText() +"'" )       ){
```

```
                        ResultSet x = stmt.executeQuery();
                        ResultSet y = stmt2.executeQuery();


     CourseNameBox.setText(x.getString("LEGAL_NAME"));

     CourseNameBox2.setText(y.getString("TITLE"));

                    } catch (SQLException e1) {
                        e1.printStackTrace();
                    }
                }

            });



            JButton clearDateButton = new JButton("Clear Date");
            clearDateButton.setForeground(Color.RED);
            clearDateButton.setBounds(693, 60, 100, 35); //
Adjust position as needed
            clearDateButton.addActionListener(e -> {
                DateField.clearSelectedDate(); // Clear the
selected date
            });
            contentPane.add(clearDateButton);

            logo = new JLabel("");
            logo.setIcon(new
ImageIcon(getClass().getResource("/bruneloldlogo.png")));
            logo.setBounds(1021, 109, 190, 250);

     initialInfoDisplay(); //I don't want it to look empty on
welcome screen.



     }//end of the main ADMINDISPLAY

     //METHODS

     //Update: It became useless recently since i changed
textfields to datepicker format, and it is always dd/mm/yyyy.
Keeping it only for records and showing the process of
development
     //w3 school SimpleDataFormat reference
     private boolean isValidDate(String date) {
            SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yyyy");
            sdf.setLenient(false);
            //Explanation: as default Lenient is true. And dates
are auto-corrected. Therefore we have to set it false. So if
```

```
someone inputs 30th of February It gives an error. If Lenient
would be true, It would fix it as 1 March.
          //That is a problem because we want to search EXACT
dates in our datebase.
          try {
               sdf.parse(date);
               return true;
          } catch (ParseException e) {
               return false;
          }
     }

     public void PieChartMaker (double attended, double late,
double absent) {
          //removing if there is any other component, so it
resets the pie chart.
          Component[] components =
contentPane.getComponents();
         for (Component component : components) {
            if (component instanceof JPanel &&
component.getName() != null &&
component.getName().equals("chartContainer")) { //3 conditions
to remove.
                contentPane.remove(component);
                break; // Remove only the first found chart
            }
         }

                    // Create slices
                    Slice[] slices = {
                        new Slice(attended, Color.GREEN,
"Attended"), //first number is how many is attended. //it is
actually does not make sense t assign "double" type, because a
person can not be half.
                        new Slice(late, Color.YELLOW, "Late"),
                        new Slice(absent, Color.RED, "Absent")
                    };

                    // Create the PieChart
                    PieChart pieChart = new PieChart(slices);

                    // Create a container panel with
BorderLayout to handle resizing
                    JPanel chartContainer = new JPanel();
                    chartContainer.setLayout(new
BorderLayout());
                    chartContainer.setBounds(924, 68, 350,
300);  // Position on AdminDisplay //If i want to reposition
the chart, it is this parameters.
                    chartContainer.setBackground(mycolor);
                    chartContainer.add(pieChart,
BorderLayout.CENTER);
```

```
                    chartContainer.setName("chartContainer");
// Set a name to identify it later //Condition to remove
piechart with every search button action.

                    // Add to contentPane
                    contentPane.add(chartContainer);
                    pieChart.setBackground(mycolor);
                    contentPane.revalidate(); //Necessary to
reload (kind of) the pie chart panel
                  contentPane.repaint();
                  logo.setVisible(false);

    }

    public static void initialInfoDisplay() {
         String query = "SELECT LEGAL_NAME,KISCOURSEID, TITLE
FROM CombinedCourseAndInstitution";

        try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAttendance
DataBase.db");
             PreparedStatement stmt =
conn.prepareStatement(query);
             ResultSet rs = stmt.executeQuery()) {

            DefaultTableModel model = new DefaultTableModel()
{
                @Override
                public boolean isCellEditable(int row, int
column) {
                    return false; // Make the table non-
editable
                }
            };

            model.addColumn("Legal Name");
            model.addColumn("KIS Course ID");
            model.addColumn("Title");


            while (rs.next()) {
                String legalName =
rs.getString("LEGAL_NAME");
                String kisCourseId =
rs.getString("KISCOURSEID");
                String title = rs.getString("TITLE");

                model.addRow(new
Object[]{legalName,kisCourseId, title });
            }

            table.setModel(model);
```

```
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error
loading initial data: " + e.getMessage());
        }

        DateField.clearSelectedDate(); // Clear the selected
date because it always starts with a date for some reason


         contentPane.add(logo);
         logo.setVisible(true);


    }
}//end of class
```

---

# RegistrationForm

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;

public class RegistrationForm extends JFrame {
    private JTextField nameField, surnameField;
    private JPasswordField passwordField;
    private static JComboBox<String> universityBox; //set
static to use it in identifyCourseID method
     private static JComboBox<String> departmentBox; //set
static to use it in identifyCourseID method
    private JButton registerButton, cancelButton;

    public RegistrationForm() {
        setTitle("Register New Student");
        setSize(400, 300);

setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
//https://docs.oracle.com/javase/6/docs/api/java/awt/Wind
ow.html#setLocationRelativeTo(java.awt.Component)
```

```
        setLayout(new GridLayout(6, 2, 5, 5));

        // Labels and Inputs
        add(new JLabel("Name and Surname:"));
        nameField = new JTextField();
        add(nameField);

        add(new JLabel("Set Password:"));
        passwordField = new JPasswordField();
        add(passwordField);

        add(new JLabel("University:"));
        universityBox = new JComboBox<>();
        //method implementation. Choose the uni from
existing universities
        loadUniqueUniversities(universityBox);
        add(universityBox);

        add(new JLabel("Department:"));
        departmentBox = new JComboBox<>();
        //method implementation. Choose the department
from existing universities
        loadUniqueDepartments(departmentBox);
        add(departmentBox);



        add(new JLabel("ASSIGNED COURSE ID:"));
        JTextField AssignedCourseID = new JTextField();
        AssignedCourseID.setEditable(false); //not
editable
        add(AssignedCourseID);
      //Setting text by using search method from selected
Department and uni
        universityBox.addActionListener(new
ActionListener() {

            public void actionPerformed(ActionEvent e) {

    AssignedCourseID.setText(identifyCourseID());
            }
        });

        departmentBox.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
    AssignedCourseID.setText(identifyCourseID());
            }
        });

        //Quick date implementation for Register button
        LocalDate currentDate = LocalDate.now(); // Get
the current date
        DateTimeFormatter myformat =
DateTimeFormatter.ofPattern("dd/MM/yyyy"); // Formatting
to fit among other dates in Attendance database
        String formattedDate =
currentDate.format(myformat); // Format the date

        // Buttons
        registerButton = new JButton("Register");
        add(registerButton);

        registerButton.addActionListener(new
ActionListener() {
         public void actionPerformed(ActionEvent e) {
             if (nameField.getText().isEmpty() ||
passwordField.getText().isEmpty() ||
universityBox.equals("Select University") ||
departmentBox.equals("Select Department")) {
                 JOptionPane.showMessageDialog(null,
"Please fill all fields correctly.");
                 return;
             }

                 try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAtten
danceDataBase.db");
                      PreparedStatement stmt =
conn.prepareStatement("INSERT INTO
Unique_Students_With_Passwords (StudentID, Password)
VALUES ('" + nameField.getText()+  "','"
+passwordField.getText() + "')");
                      PreparedStatement stmt2 =
conn.prepareStatement("INSERT INTO ATTENDANCE (StudentID,
COURSEID, SESSIONDATE,ATTENDANCE) VALUES ('" +
nameField.getText()+  "','" +AssignedCourseID.getText() +
"','" +formattedDate +"','" +"REGISTERED')");
                      ){


                 stmt.executeUpdate();
```

```
                    stmt2.executeUpdate();


                    } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                    }

                JOptionPane.showMessageDialog(null,
"Student Registered Successfully!"); //Reference:
ShowMessageDialog learned from Zear Ibrahim.
                dispose(); // Close window after successful
registration
            }
        });

        cancelButton = new JButton("Cancel");
        add(cancelButton);


        // Cancel button action
        cancelButton.addActionListener(e -> dispose());
//got help from AI

    }


    //METHODS

    public static void
loadUniqueUniversities(JComboBox<String> mydropbox1){
     try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAtten
danceDataBase.db");
                PreparedStatement stmt =
conn.prepareStatement("SELECT DISTINCT LEGAL_NAME FROM
CombinedCourseAndInstitution");) {

            ResultSet rs = stmt.executeQuery();
            ArrayList<String> universityList = new
ArrayList<>();
            universityList.add("Select University"); //to
help register and add a condition to not leave it empty

            while (rs.next()) {

     universityList.add(rs.getString("LEGAL_NAME"));
            }
```

```
            // Convert list to array and add to
JComboBox/dropbox
            for (int i = 0; i < universityList.size(); i++)
{
                mydropbox1.addItem(universityList.get(i));
            }


        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void
loadUniqueDepartments(JComboBox<String> mydropbox2){
        try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAtten
danceDataBase.db");
                PreparedStatement stmt =
conn.prepareStatement("SELECT DISTINCT TITLE FROM
CombinedCourseAndInstitution");) {

            ResultSet rs = stmt.executeQuery();
            ArrayList<String> departmentList = new
ArrayList<>();
            departmentList.add("Select Department"); //to
help register and add a condition to not leave it empty

            while (rs.next()) {
                departmentList.add(rs.getString("TITLE"));
            }

            // Convert list to array and add to
JComboBox/dropbox
            for (int i = 0; i < departmentList.size(); i++)
{
                mydropbox2.addItem(departmentList.get(i));
            }


        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static String identifyCourseID() {
```

```
    String x = "Course not found in this university";
//set as fixed in case try does not re-write this. So
error handling of sort...
    try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:src/ProjectAtten
danceDataBase.db");
             PreparedStatement stmt =
conn.prepareStatement("SELECT KISCOURSEID FROM
CombinedCourseAndInstitution WHERE TITLE = '"+
departmentBox.getSelectedItem().toString() +"' AND
LEGAL_NAME =  '"+
universityBox.getSelectedItem().toString() +"'");) {

         ResultSet rs = stmt.executeQuery();

         if (rs.next()) {
          x = rs.getString("KISCOURSEID");
         }


    } catch (SQLException e) {
        e.printStackTrace();
    }
    return x;

   }


}
```

---

# DatePicker

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

public class DatePicker extends JPanel {
    private JComboBox<Integer> dayBox;
    private JComboBox<String> monthBox;
    private JComboBox<Integer> yearBox;

    public DatePicker() {
        setLayout(new FlowLayout());

        // Day ComboBox (1 to 31)
        dayBox = new JComboBox<>();
        dayBox.addItem(null); // Add null as the first item
(no selection)
        for (int i = 1; i <= 31; i++) {
            dayBox.addItem(i);
```

```
        }

        // Month ComboBox
        // **Add null as the first item (no selection)
        String[] months = {
            "January", "February", "March", "April", "May",
"June",
            "July", "August", "September", "October",
"November", "December"
        };
        monthBox = new JComboBox<>(months);


        // Year ComboBox (from currentYear - 10 to currentYear
+ 1)
        yearBox = new JComboBox<>();
        int currentYear =
Calendar.getInstance().get(Calendar.YEAR);
        yearBox.addItem(null); // Add null as the first item
(no selection)
        for (int i = currentYear - 10; i <= currentYear + 1;
i++) {
            yearBox.addItem(i);
        }
        yearBox.setSelectedItem(currentYear); // Set default
year to current year

        // Add to panel
        add(dayBox);
        add(monthBox);
        add(yearBox);
    }


    public String getSelectedDate() {
     Integer day = (Integer) dayBox.getSelectedItem();
     int monthIndex = monthBox.getSelectedIndex();
     Integer year = (Integer) yearBox.getSelectedItem();

     // Check if any field is null (no selection)
        if (day == null || monthIndex == -1 || year == null) {
//i had to cast "Integer" above to day==null... making sense.
            return null; // Return null if no date is selected
        }
        int month = monthBox.getSelectedIndex() + 1; //
Convert 0-based index to 1-based

        return String.format("%02d/%02d/%04d", day, month,
year); // Ensures format DD/MM/YYYY
    }

    // Method to clear the selected date
```

```
    public void clearSelectedDate() {
        dayBox.setSelectedItem(null);
        monthBox.setSelectedItem(null);
        yearBox.setSelectedItem(null);
    }
}
```

---

# PieChart

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

class Slice {
    double value;
    Color color;
    String label;

    public Slice(double value, Color color, String label) {
        this.value = value;
        this.color = color;
        this.label = label;
    }
}

public class PieChart extends JPanel {
    private Slice[] slices;

    public PieChart(Slice[] slices) {
        this.slices = slices;
        setPreferredSize(new Dimension(400, 400)); // Set
default size
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        drawPie((Graphics2D) g, getBounds(), slices);
    }

    private void drawPie(Graphics2D g, Rectangle area, Slice[]
slices) {
        double total = 0;
        //i will add if value>0 because otherwise all text
overlap.
        for (Slice slice : slices) {
          if (slice.value>0) {
            total += slice.value;
          }
        }

        double curValue = 0.0;
        int startAngle;
```

```
      //i will add if value>0 because otherwise all
text(Attended/LAte/Yes) overlap.
        for (Slice slice : slices) {
          if (slice.value>0) {

            startAngle = (int) (curValue * 360 / total);
            int arcAngle = (int) (slice.value * 360 / total);
            g.setColor(slice.color);
            g.fillArc(area.x, area.y, area.width, area.height,
startAngle, arcAngle);

            // Draw text labels


            double angle = Math.toRadians(startAngle +
arcAngle / 2.0);
            int labelX = area.x + area.width / 2 + (int)
(area.width / 3 * Math.cos(angle));
            int labelY = area.y + area.height / 2 - (int)
(area.height / 3 * Math.sin(angle));

            g.setColor(Color.BLACK);
            g.drawString(slice.label + " (" +
String.format("%.1f", (slice.value / total) * 100) + "%)",
labelX, labelY);
            g.setFont( new Font("Lucida Grande", Font.PLAIN,
10));

            curValue += slice.value;
          }
            }

    }
}
```