

Criterion C: Development

Table of Contents

1. Introduction.....	2
2. External Libraries Used.....	2
3. Database and SQL Queries.....	3
4. Password Encryption Algorithm.....	7
5. Progress Bar Algorithms.....	8
5.1 Assignment Progress Bar.....	8
5.2 Deadline Progress Bar.....	9
6. Object-Oriented Programming Concepts.....	10
6.1 Inheritance.....	10
6.2 Polymorphism.....	10
6.2 Encapsulation.....	11
7. Look and Feel Changes.....	12
8. GUI Elements.....	13
9. Error Handling.....	18
10. Works Cited.....	20

1. Introduction

This application was for a client who wished to have an assignment tracking software that would be easily navigable, secure, visually intuitive, and stylistically catered to her tastes. The application was centered around these objectives, using various classes and other OOP and database features to help bring this software solution to fruition.

2. External Libraries Used

Libraries are collections of resources used in software development to easily gain access to different features, usually through the help of specific methods and classes. Even with the comprehensive set of features already built into Java, it is necessary to expand the feature set with libraries when developing really particular software. Since this applied in my case, I used a number of libraries during the development of my application; they are listed below.

1. `java.util.Properties`
2. `javax.swing`
3. `java.time`
4. `java.time.format.DateTimeFormatter`

Some libraries were also downloaded in the form of “.jar” files and they are listed below.

1. `rs2xml.jar`
2. `jcalender-1.4.jar`
3. `sqlite-jdbc-3.26.2.1.jar`
4. `JTattoo-1.5.13.jar`

3. Database and SQL Queries

Since my application was so reliant on data that had to be stored for indefinite periods of time, implementing a database was paramount. I opted to pair my database with an SQLite database since it would be easy to operate and would not be very resource-intensive. I retained the same names for the tables as was planned in the Design document. I connected the database to my application through the use of the “sqlite-jdbc-3.26.2.1.jar” library and the following objects that came with it.

```
Connection conn;  
ResultSet rs;  
PreparedStatement pst;
```

Figure 1: Java SQL objects

As mentioned in the Design Document, the following SQL queries were used:

1. SELECT
2. INSERT
3. DELETE

SELECT was used in the following instances.

```
String sql = "SELECT * from Account where Email=? and Decrypted_Password=?";
```

Figure 2: SELECT used in the “Login” class

```

public void retrieveOne() {
    String a1 = jTextField1.getText();
    String sql = "select * from Account where Email='" + a1 + "'";
    try{
        pst=conn.prepareStatement(sql);
        rs=pst.executeQuery();
        if (rs.next()){
            jTextField2.setText(rs.getString(3));

        } else {
            JOptionPane.showMessageDialog(null, "incorrect email");
        }
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
}

public void retrieveTwo() {
    String a1 = jTextField1.getText();
    String a2 = jTextField3.getText();
    String sql = "select * from Account where Email='"+a1+"' AND PIN='"+a2+"'";
    try{
        pst=conn.prepareStatement(sql);
        rs=pst.executeQuery();
        jTextField4.setText(rs.getString(5));
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null, "Fields Inputted Incorrectly");
    }
}
}

```

Figure 3: SELECT used in the “forgot” class

```

public void jTable1() {
    try{
        String sql = "select Name, Subject, Deadline, Importance from Pending";
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        jTable1.setModel(DbUtils.resultSetToTableModel(rs));
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,e);
    }
}

public void jTable2() {
    try{
        String sql = "select Name, Subject, Deadline, Completed from Completed";
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        jTable2.setModel(DbUtils.resultSetToTableModel(rs));
        rs.close();
        pst.close();
        conn.close();
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,e);
    }
}
}

```

Figure 4: SELECT used in the “homepage” class

SELECT was used in these cases for the desired values from the database either to be displayed to the user or to perform functions like checking credentials.

The following figures show the use of **INSERT** in the application.

```
try{
    String sql = "Insert into Account (Name,Email,Password,PIN,Decrypted_Password) values(?,?,?, ?,?)";
    pst = conn.prepareStatement(sql);
    pst.setString(1, jTextField1.getText());
    pst.setString(2, jTextField2.getText());
    pst.setString(3, Hint_Password);
    pst.setString(4, jTextField3.getText());
    pst.setString(5, jPasswordField1.getText());
    pst.execute();
    JOptionPane.showMessageDialog(null,"New Account Created");
    pst.close();
    conn.close();
}
catch(Exception e) {
    JOptionPane.showMessageDialog(null,e);
}
```

Figure 5: INSERT used in the “signUp” class

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        String sql = "Insert into Pending (Name,Subject,Deadline,Importance) values(?,?,?, ?)";
        pst = conn.prepareStatement(sql);
        pst.setString(1, jTextField1.getText());
        pst.setString(2, jTextField2.getText());
        pst.setString(3, ((JTextField)jDateChooser1.getDateEditor().getUiComponent()).getText());
        pst.setString(4, (String) jComboBox1.getSelectedItem());
        pst.execute();
        JOptionPane.showMessageDialog(null,"Assignment Added!");
        pst.close();
        conn.close();
    }
    catch(Exception e) {
        JOptionPane.showMessageDialog(null,e);
    }
}
```

Figure 6: INSERT used in the “addAssignment” class

```

public void updateRecord() {
    String sql = "insert into Completed (Name,Subject,Deadline,Completed) values (?, ?, ?, ?)";
    try{
        pst = conn.prepareStatement(sql);
        pst.setString(1, jTextField1.getText());
        pst.setString(2, jTextField2.getText());
        pst.setString(3, jTextField3.getText());
        pst.setString(4, ((JTextField)jDateChooser1.getDateEditor().getUiComponent()).getText());
        pst.execute();
        JOptionPane.showMessageDialog(null, "Assignment Status Updated");
    }
    catch(Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
}

```

Figure 7: INSERT used in the “deleteAssignment” class

In these cases, INSERT is used to insert values from the specified text fields into their corresponding values into the table, as specified by the queries in the Strings named as “sql”.

DELETE was used once in the “deleteAssignment” class, as shown in the following code segment.

```

public void Delete() {
    String sql = "delete from Pending where Name=?";
    try{
        pst = conn.prepareStatement(sql);
        pst.setString(1, jTextField1.getText());
        pst.execute();
        rs.close();
        pst.close();
        conn.close();
    }
    catch(Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
}

```

Figure 8: DELETE used in the “deleteAssignment” class

In this case, delete was used to remove the completed task from the “Pending” table in the database.

4. Password Encryption Algorithm

The following algorithm was used for encrypting the password in the “signUp” class so that the user would be able to obtain a password hint if needed in the Forgot Password window:

```
String Hint_Password = "";
try {
    String Pass = jPasswordField1.getText();
    String Hint = Pass.substring(0,2);
    String Asterisk = "";
    for (int i = 0; i < Pass.length() - 2; i++) {
        Asterisk = Asterisk + "*";
    }
    Hint_Password = Hint + Asterisk;
}
catch(Exception e) {
    JOptionPane.showMessageDialog(null,e);
}
```

Figure 9: Password Encryption Algorithm

Here, the password created by the user is spliced in such a way that only the first two characters remain. After that, the rest of the characters of the password are replaced by asterisks through a loop, and the two strings are joined to form the encrypted password. This version of the password is stored alongside the original password in the database.

5. Progress Bar Algorithms

5.1 Assignment Progress Bar Algorithm

To obtain a numerical value of the workload which could then be visually translated into a progress bar for the assignment density, the following algorithm was used:

```
public void progress() {
    int i = 0;
    int n = 0;
    int Sum = 0;
    try{
        String sql="Select * from Pending";
        pst = conn.prepareStatement(sql);
        rs=pst.executeQuery();
        while (rs.next()) {
            i = Integer.parseInt(rs.getString("Importance"));
            n = i*10;
            Sum = Sum + n;
            jProgressBar1.setValue(Sum);
            jProgressBar1.setStringPainted(true);
        }

        rs.close();
        pst.close();
        conn.close();
    }
    catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Status Already Updated!");
    } finally {
        try{
            rs.close();
            pst.close();
            conn.close();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }
}
```

Figure 10: Assignment_Progress algorithm

Here, the importance of each assignment is extracted from the database, on a scale of one to three, and added together. This value will then determine the percentage on the progress bar. If there are no assignments to report, the bar will be blank and if the user confirms by refreshing the progress bar, the “Status Already Reported!” message will appear.

5.2 Deadline Progress Bar

To obtain a numerical value for the workload in terms of deadlines that could be translated visually, the following algorithm was used:

```
public void dateProgress() {
    int N = 0;
    String date;
    String dateSub;
    int dateInt;
    String now = Instant.now().atZone(ZoneOffset.UTC).format(DateTimeFormatter.ISO_LOCAL_DATE);
    String nowSub = now.substring(0,4) + now.substring(5,7) + now.substring(8);
    int nowInt = Integer.parseInt(nowSub);
    try{
        String sql="Select * from Pending";
        pst = conn.prepareStatement(sql);
        rs=pst.executeQuery();
        while (rs.next()) {
            date = rs.getString(3);
            dateSub = date.substring(0,4) + date.substring(5,7) + date.substring(8);
            dateInt = Integer.parseInt(dateSub);
            if ((dateInt - nowInt) >= 14) {
                N = N+10;
            } else if ((dateInt - nowInt) >= 7 && (dateInt - nowInt) < 14) {
                N = N+20;
            } else if ((dateInt - nowInt) >= 2 && (dateInt - nowInt) < 7) {
                N = N+80;
            } else {
                N = N+100;
            }
            if (N>100){
                N = 100;
            }
            jProgressBar1.setValue(N);
            jProgressBar1.setStringPainted(true);
        }
    }
}
```

Figure 11: Deadline_Progress Algorithm

Here, the date of the current date is extracted and stored in “now” in the format of yyyy/MM/dd, with nowSub removing the slashes and nowInt turning it into an integer. Inside the WHILE loop, the same is done for every pending deadline in the database. The difference between these values and the current date then gives specific percentage values, which are then added to the percentage used for the progress bar.

6. Object-Oriented Programming Concepts

Some of the major OOP concepts used in this application are listed below.

6.1 Inheritance

Inheritance allows for a class to obtain the attributes and behaviors of another class. Inheritance has been used in every class from the application other than “Connector.java”, with the following type of code used.

```
public class Welcome extends javax.swing.JFrame {
```

Figure 12: Inheritance Use

Since the code largely centers on the Java Swing library and uses a GUI, it was necessary to use a frame in which all the code components would be visually represented. As such, the “javax.swing.JFrame” class was used to achieve this easily.

6.2 Polymorphism

Polymorphism is when the class uses methods provided by the class it inherits from to perform various methods in unique ways. The following is an example of Polymorphism used in the application.

```
public Login() {  
    super("Login");  
    initComponents();  
    conn = Connector.connectDB();  
}
```

Figure 13: Polymorphism Use

In these cases, the code is using the “super()” method to set a unique name for each class, which would be displayed on the top of the window.

6.3 Encapsulation

Encapsulation is an OOP concept in which certain segments of the code are bundled together to either be made accessible or inaccessible to the user. In this application, the encapsulation works on the basis of “private” and “public” methods, with user access not being regarded in private methods, vice versa applying for public ones. The following is an example of encapsulation used in this software.

```
public void jTable2() {
    try{
        String sql = "select Name, Subject, Deadline, Completed from Completed";
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        jTable2.setModel(DbUtils.resultSetToTableModel(rs));
        rs.close();
        pst.close();
        conn.close();
    } catch(Exception e) {
        JOptionPane.showMessageDialog(null,e);
    }
}

/** This method is called from within the constructor to initialize the form ...5 lines */
@SuppressWarnings("unchecked")
Generated Code

private void jTable2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    addAssignment ob = new addAssignment();
    ob.setVisible(true);
    setVisible(false);
}
```

Figure 14: Encapsulation example from “Homepage” class

Here, since the method “jTable2()” attempts to display the contents of the “Completed” table to the user, it is declared as a public method. Conversely, “jToggleButton...evt)” is a private method since it is only visible to the developer and the user does not need access to it.

7. Look and Feel Changes

After the client mentioned that she preferred a darker theme over the prototype's default gray color scheme, the look and feel of the application were changed accordingly. For this, the JTattoo set of themes was chosen, for which the following .jar file was added: "JTattoo-1.5.13.jar". JTattoo has a wide variety of themes, and the "Graphite" theme seemed like the best fit for the client, and it was implemented as follows.

```
import com.jtattoo.plaf.graphite.GraphiteLookAndFeel;
```

Figure 15: JTattoo Graphite import

```
for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  
    Properties props = new Properties();  
    props.put("logoString", "");  
    GraphiteLookAndFeel.setCurrentTheme(props);  
    UIManager.setLookAndFeel("com.jtattoo.plaf.graphite.GraphiteLookAndFeel");  
}
```

Figure 16: Graphite implementation

With the Graphite theme imported, Figure n shows how setting an empty String for the "props" object removes the JTattoo watermark that would otherwise be visually unappealing in the application.

8. GUI Elements

The following windows were created for use in the application.

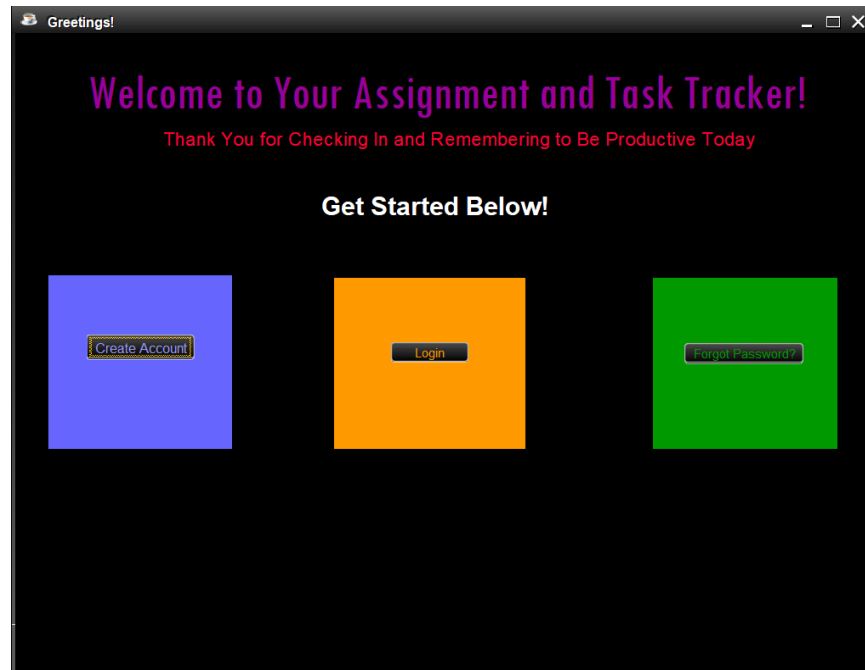


Figure 17: Welcome Window

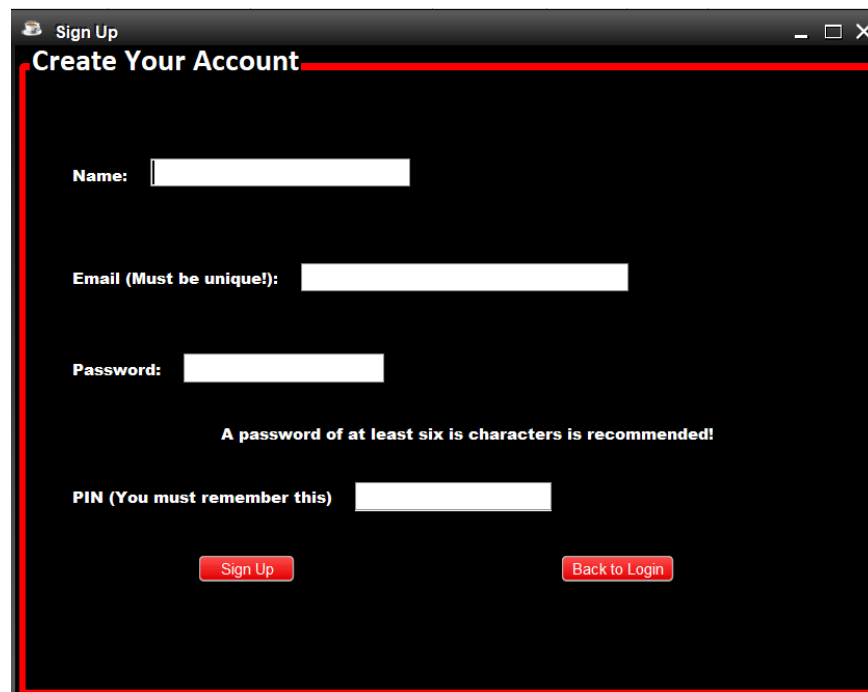
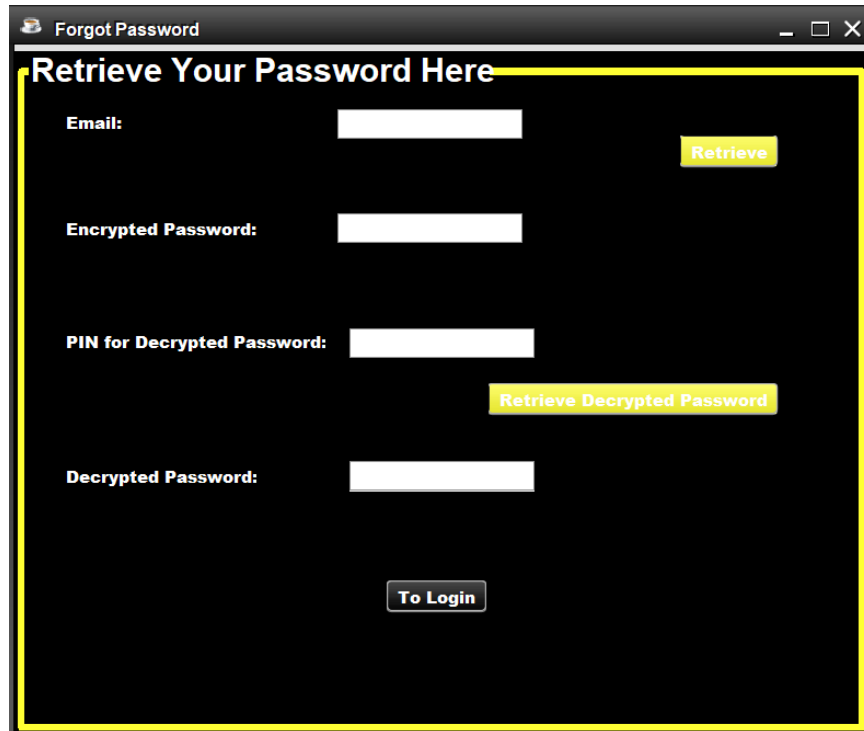
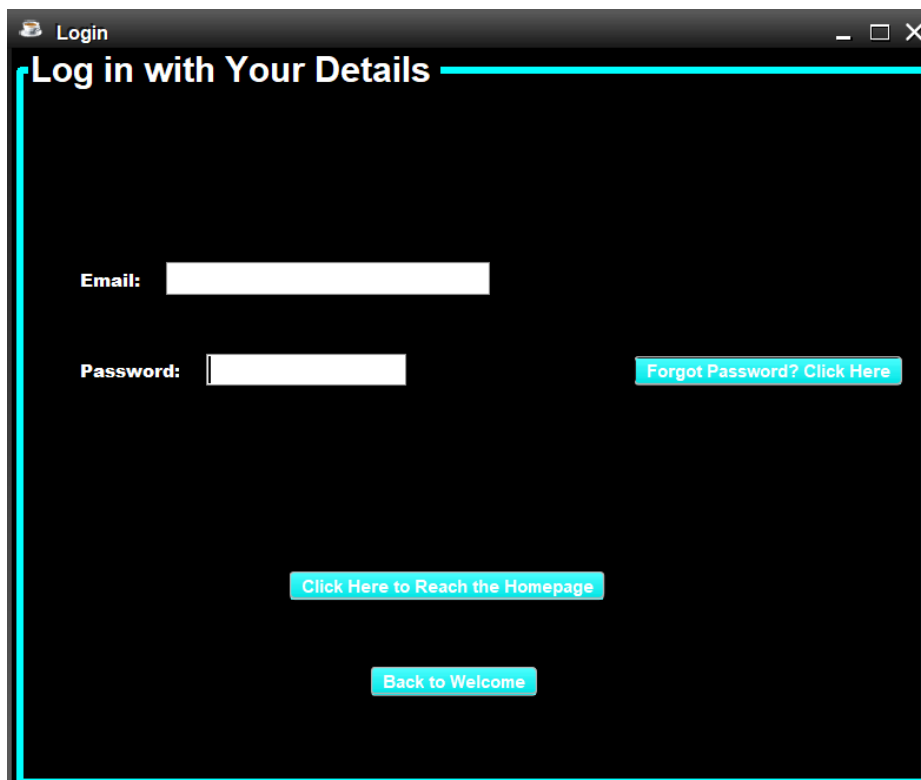
A screenshot of a window titled "Sign Up". The window has a black background. At the top, it says "Create Your Account" in white. Below this, there are four input fields: "Name:", "Email (Must be unique!):", "Password:", and "PIN (You must remember this)". Below the "Password:" field, it says "A password of at least six is characters is recommended!". At the bottom, there are two buttons: "Sign Up" and "Back to Login".

Figure 18: Sign Up Window



A screenshot of a web application window titled "Forgot Password". The window has a black background and a yellow border. The title "Retrieve Your Password Here" is displayed in white. Below the title, there are four input fields and two buttons. The first input field is labeled "Email:" and has a yellow "Retrieve" button to its right. The second input field is labeled "Encrypted Password:". The third input field is labeled "PIN for Decrypted Password:" and has a yellow "Retrieve Decrypted Password" button to its right. The fourth input field is labeled "Decrypted Password:". At the bottom center, there is a black button with white text that says "To Login".

Figure 19: Forgot Password Window



A screenshot of a web application window titled "Login". The window has a black background and a cyan border. The title "Log in with Your Details" is displayed in white. Below the title, there are two input fields and three buttons. The first input field is labeled "Email:". The second input field is labeled "Password:" and has a cyan button with white text that says "Forgot Password? Click Here" to its right. At the bottom center, there are two cyan buttons with white text: "Click Here to Reach the Homepage" and "Back to Welcome".

Figure 20: Login Window

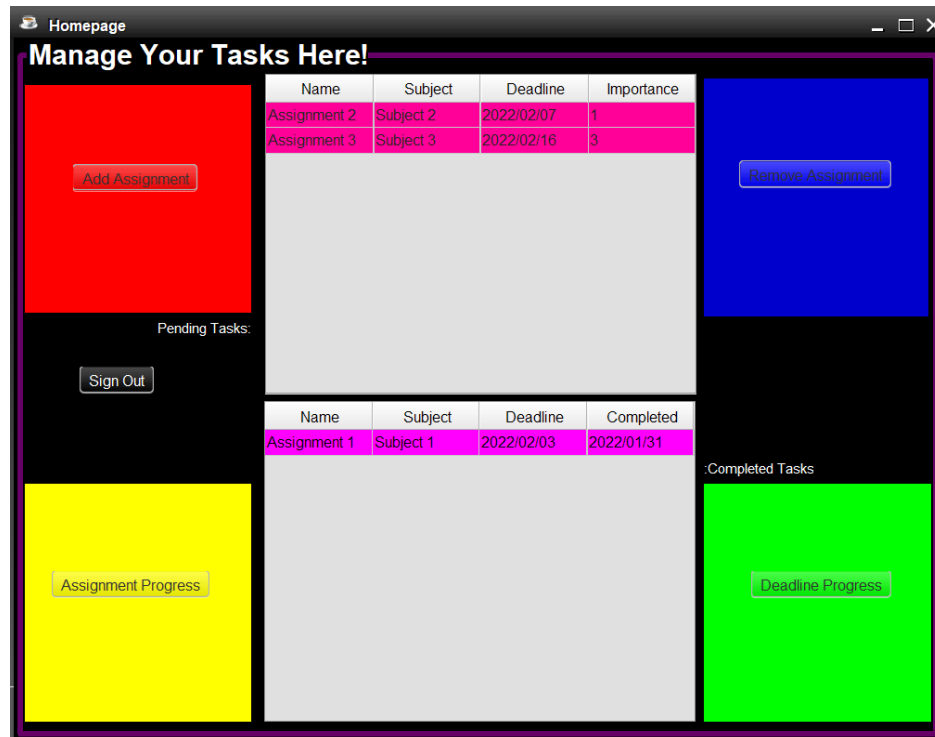


Figure 21: Homepage Window

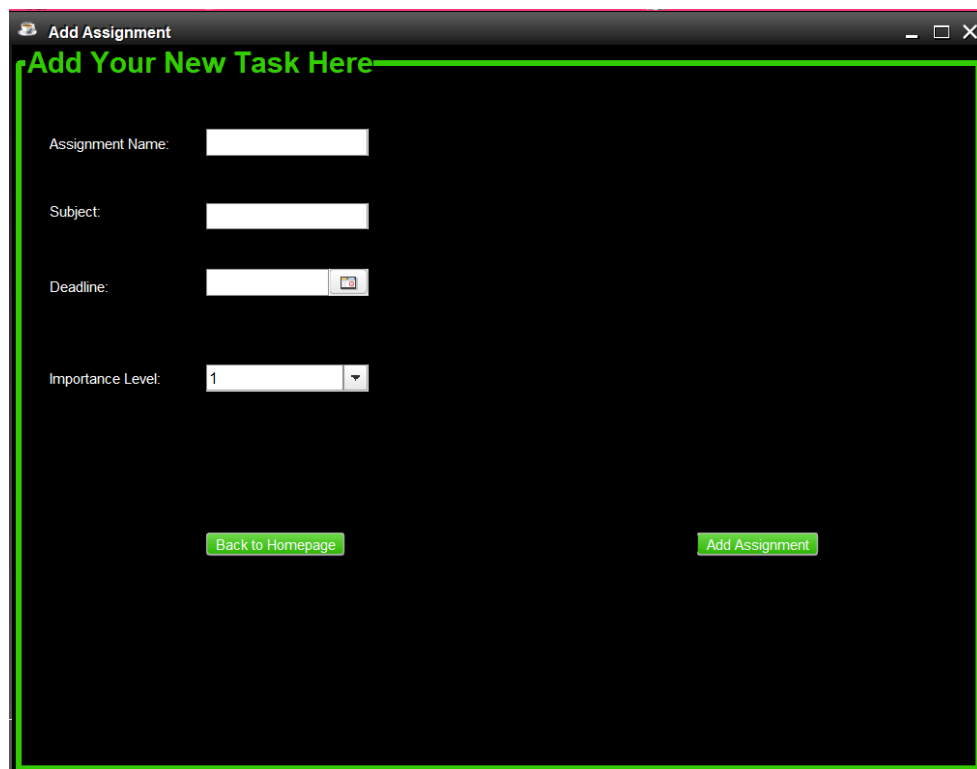
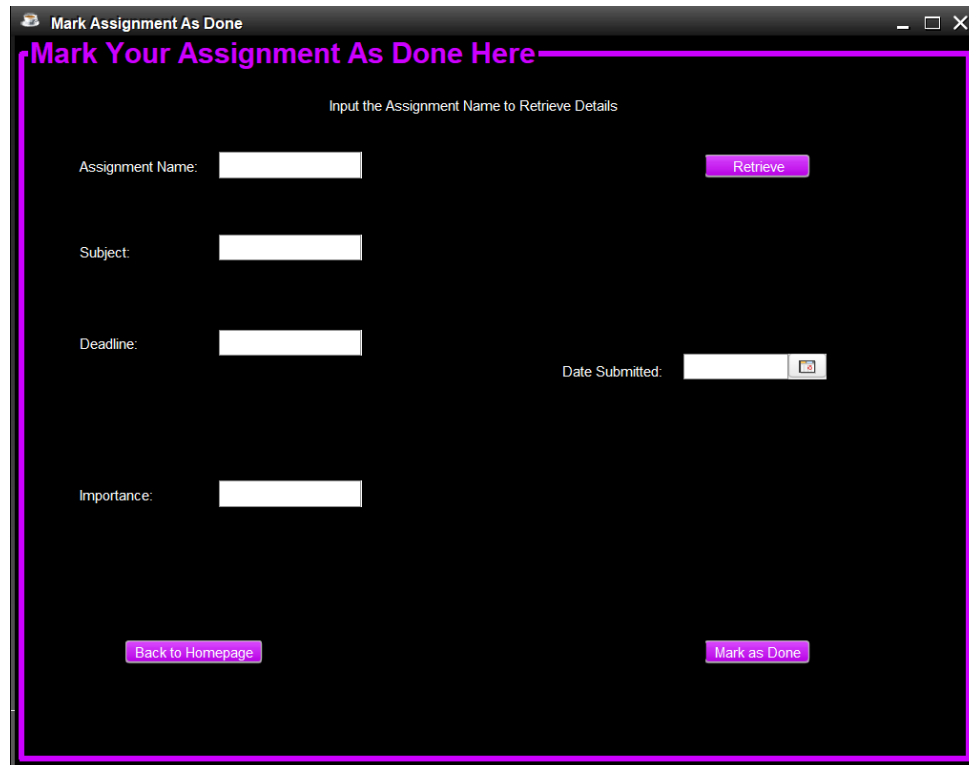


Figure 22: Add Assignment Window



The window titled "Mark Assignment As Done" features a dark blue header with the title in white. Below the header, a large red banner reads "Mark Your Assignment As Done Here". Underneath, a subtitle says "Input the Assignment Name to Retrieve Details". The form contains five input fields: "Assignment Name:", "Subject:", "Deadline:", "Importance:", and "Date Submitted:". The "Date Submitted:" field includes a calendar icon. A red "Retrieve" button is positioned to the right of the "Assignment Name:" field. At the bottom, there are two red buttons: "Back to Homepage" on the left and "Mark as Done" on the right.

Mark Assignment As Done

Mark Your Assignment As Done Here

Input the Assignment Name to Retrieve Details

Assignment Name: Retrieve

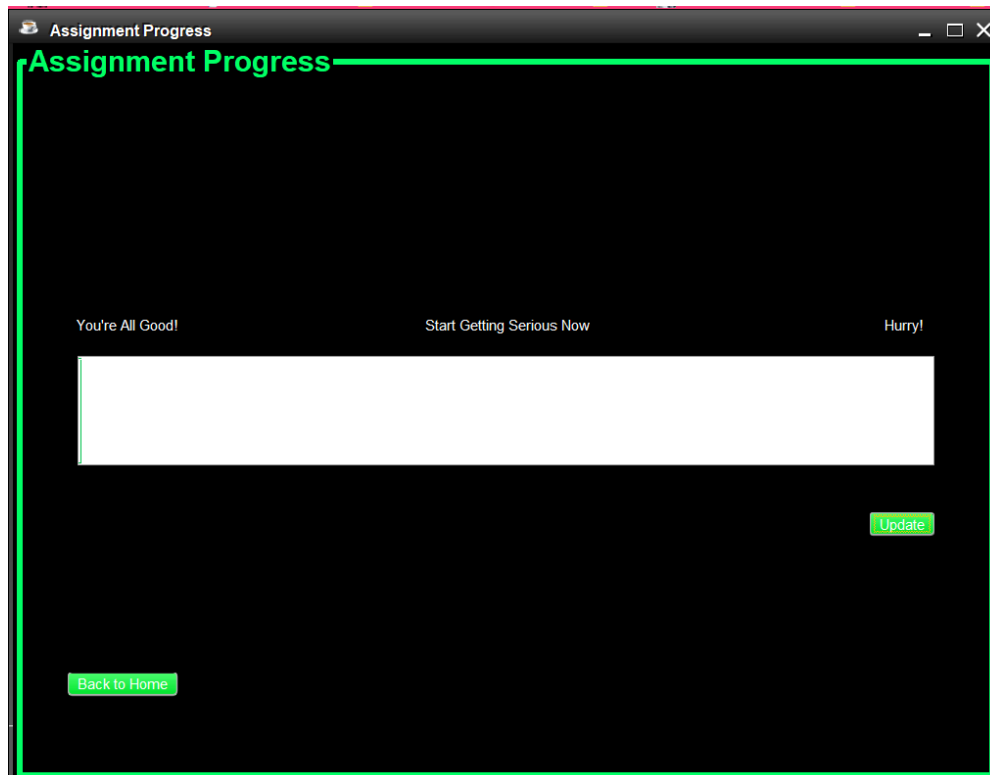
Subject:

Deadline: Date Submitted:

Importance:

Back to Homepage Mark as Done

Figure 23: Mark Assignment As Done Window



The window titled "Assignment Progress" has a dark blue header with the title in white. Below the header, a large green banner reads "Assignment Progress". Underneath, three status indicators are displayed: "You're All Good!", "Start Getting Serious Now", and "Hurry!". A large white rectangular area is positioned below these indicators. A green "Update" button is located at the bottom right of this area. At the bottom left of the window, there is a green "Back to Home" button.

Assignment Progress

You're All Good! Start Getting Serious Now Hurry!

Update

Back to Home

Figure 24: Assignment Progress Window

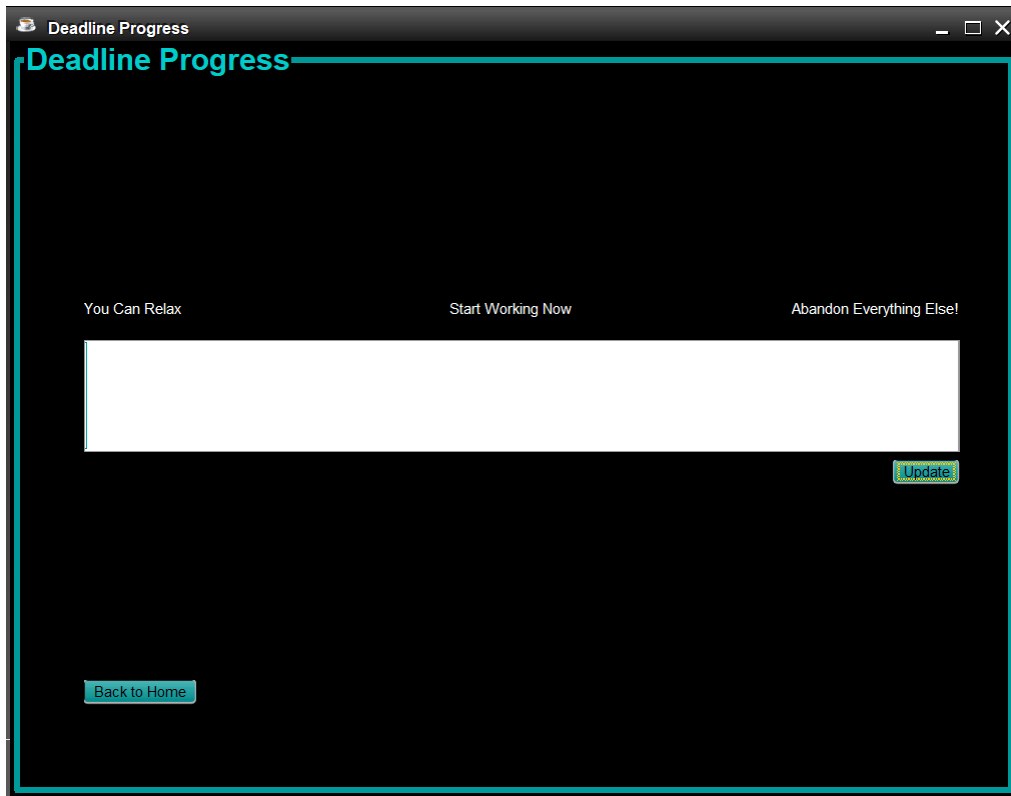


Figure 25: Deadline Progress Window

9. Error Handling

Error handling is used in the application in the form of try-catch blocks. Try-catch blocks allow for a certain task to be fulfilled and when it fails to do so, an alternate action is executed instead. Try-catch blocks were useful in the development of the software to easily figure out the issue caused in the program, such as in the following figure.

```
try{
    pst = conn.prepareStatement(sql);
    pst.setString(1,jTextField1.getText());
    pst.setString(2,jPasswordField1.getText());
    rs = pst.executeQuery();
    if (rs.next()){
        rs.close();
        pst.close();
        conn.close();
        setVisible(false);
        Homepage ob = new Homepage();
        ob.setVisible(true);
    } else {
        JOptionPane.showMessageDialog(null, "Incorrect Email and/or Password");
    }
} catch(Exception e){
    JOptionPane.showMessageDialog(null, e);
}
```

Figure 26: Try-catch in application development

Here, if issues occur that are unrelated to the client-side—such as inputting the wrong credentials in this case—a dialog box informing the user of the error occurring is shown.

Try-catch blocks are also used for the client-side, such as the following implementation.

```

try{
    String sql="Select * from Pending";
    pst = conn.prepareStatement(sql);
    rs=pst.executeQuery();
    while (rs.next()) {
        i = Integer.parseInt(rs.getString("Importance"));
        n = i*10;
        Sum = Sum + n;
        jProgressBar1.setValue(Sum);
        jProgressBar1.setStringPainted(true);
    }

    rs.close();
    pst.close();
    conn.close();
}
catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Status Already Updated!");
}

```

Figure 27: Try-catch client-side

Here, if the client is trying to update the progress bar after it has already been updated once, an error message informing them that the status has already been updated will appear.

10. Works Cited

1. “Java Tutorial: Learn Java Basics For Free.” *Codecademy*,
www.codecademy.com/learn/learn-java. Accessed 14 Feb. 2022.
2. “Java.Sql (Java Platform SE 7).” *Oracle*, 24 June 2020,
docs.oracle.com/javase/7/docs/api/java/sql/package-summary.html.
3. “JTattoo Appears in JMenu Dropdown.” *Stack Overflow*, 28 Apr. 2012,
stackoverflow.com/questions/10366972/jtattoo-appears-in-jmenu-dropdown/51261116.
4. “What Is the Easiest Way to Get the Current Date in YYYYMMDD Format?” *Stack Overflow*, 30 June 2015,
[stackoverflow.com/questions/31138533/what-is-the-easiest-way-to-get-the-current-date-i
n-yyyyymmdd-format](https://stackoverflow.com/questions/31138533/what-is-the-easiest-way-to-get-the-current-date-in-yyyyymmdd-format).