**CPP106: University Grade Encoding System GUI**

**Instructions:**

Your task is to design and construct a Java GUI application for encoding and managing grades in a university setting. Ensure that each group member has a diverse skill set, including programming, design, and problem-solving abilities. The deadline for submission is Sunday at 11:59 PM. Ensure that your completed project is submitted on time. You can use any Integrated Development Environment (IDE) of your choice for development. Working together with your team, follow the POLYA's problem-solving method:

1. **Understand the Problem:**

Begin by understanding the requirements of the grade encoding system. Discuss with your team the essential features it should include, such as inputting student grades, calculating averages, and displaying results. While these features are essential, you may also want to consider other useful functionalities, such as displaying grade distributions visually (e.g., charts) or allowing for the export of reports, to enhance the user experience.

1. **Plan a Solution:**

* Brainstorm potential solutions and sketch the layout and functionality of your system. Consider how the user will interact with the interface and how data will be processed. Allocate tasks among team members based on their strengths and expertise.
* Create pseudocode and a flowchart to outline the logic of your program. This will help visualize the steps needed to implement the solution.

1. **Execute the Plan:**

* Implement your solution using Java Swing to create the graphical interface. Divide the work among team members, ensuring each component is integrated seamlessly. Test each feature as it's developed to identify and address any issues promptly.
* Embed code comments within your Java files to explain the functionality of key components and any complex algorithms used. Comments should clarify the purpose of each method, describe input and output parameters, and provide insights into the underlying logic. This will make your code more understandable and maintainable for both your team members and future developers.

1. **Assess the Solution:**

* Test your grade encoding system thoroughly to ensure it meets the requirements and functions correctly. Verify that calculations are accurate and that the interface is user-friendly.
* Create documentation for your program, including a user manual explaining how to use the system and any technical documentation necessary for maintenance and future development.

**Documentation Outline: University Grade Encoding System GUI**

1. **Introduction:**

The goal of the University Grade Encoding System was to help university professors to manage the grades their handled students using Java Swing GUI.

• Key Objectives

1. To manage grades of students easier for university professors

2. To calculate grades of students easier and accurately

• Purpose of the System

1. To simplify the calculations of grades of students

2. To manage the input of grades by the university professors

• Primary Functions

1. Log in window for university professors

2. Dashboard with including adding students, removing students and table for viewing all students

1. **System Overview:**

• Key Features

1. By logging in, the user will be directed at the dashboard with text field about inputting the first name, last name, age, student number, prelim, midterm and final grade.
2. By clicking submit, the information will be inserted in the table.
3. The final rating in the table was calculated by (prelim grade\*0.3) + (midterm grade\*0.3) + (final grade\*0.4)/3. If the final rating was equal or greater than 60, the remarks in the table will be marked as “PASSED” or else “FAILED”.
4. By clicking the drop button, the selected specific row will be deleted in the table.
5. **User Manual:**

• Step-by-Step Instructions

1. Launch the system
2. Log in using the account
3. Fill out all the student information such as first name, last name, age, student number, prelim, midterm and final grade.
4. Submit the filled out student information and view it in the table
5. Select a row from the table and click drop button to remove student
6. **Technical Documentation:**

START

Create Log In Window Frame:

Add JPanel and JLabel for Title

Add JPanel:

Add JLabel and JTextfield for Username input

Add JLabel and JPasswordField for Password input

Add JButton for Log In

On Log In JButton:

If Username and Password input are filled and matched:

Proceed to main dashboard

Else:

Display message box (“Invalid Username or Password)

Create Main Dashboard Window Frame:

Add JPanel and JLabel for Title

Add JLabel and JTextfield for Student First Name, Last Name, Age, Section, Year Level, Prelim, Midterm and Final Grade

Add JLabel for Student Number

Add JButton for Submit:

Add JButton for Drop

Add JTable:

Set Column 1 Title to Name

Set Column 2 Title to Student Number

Set Column 3 Title to Prelim

Set Column 4 Title to Midterm

Set Column 5 Title to Final

Set Column 6 Title to Final Rating

Set Column 7 Title to Remarks

On JLabel for Student Number:

Set the Student Number to random

On Submit JButton:

If all fields are filled out:

Append student information into table and txt file

Else:

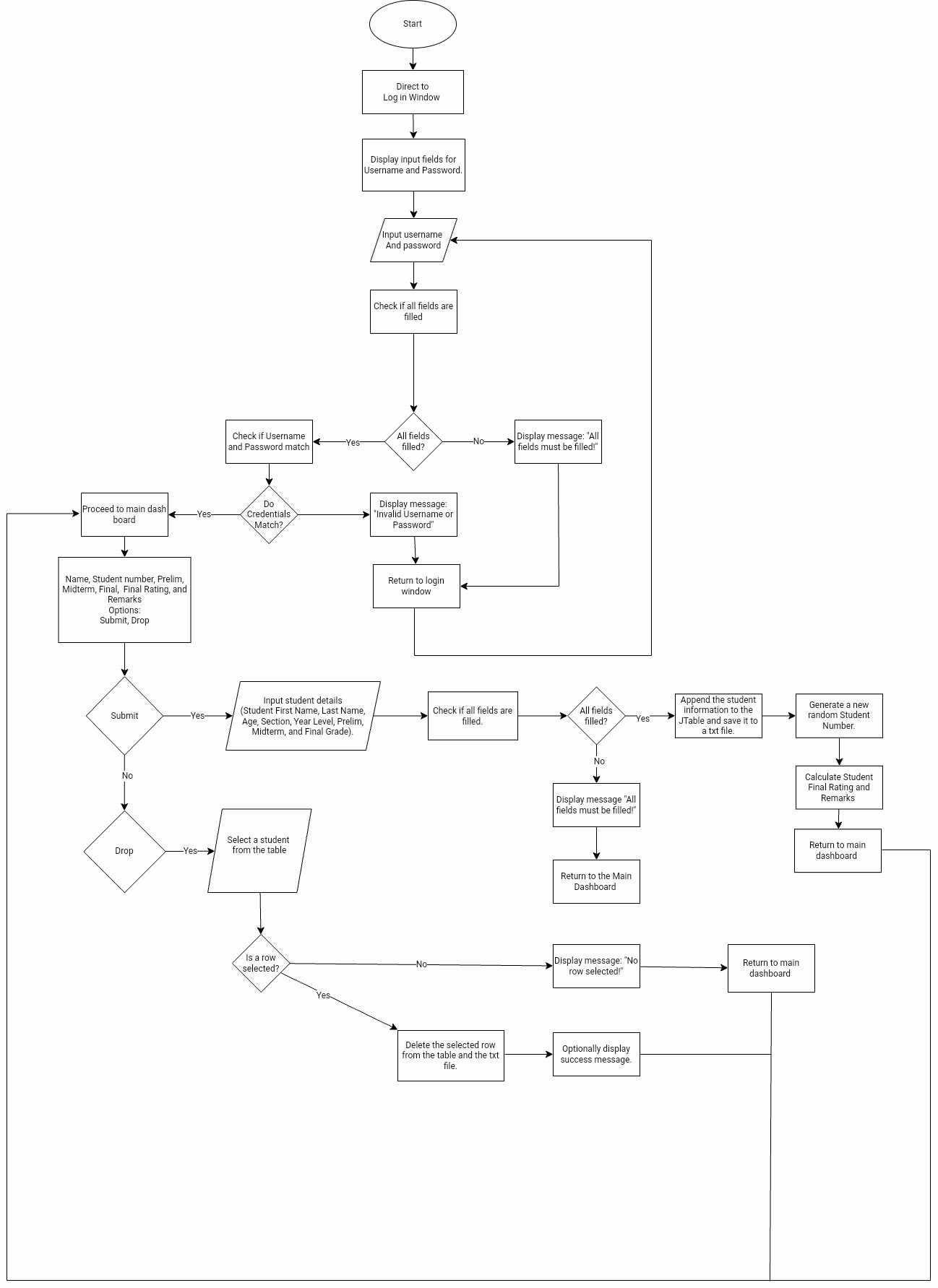
Display message box (“All fields must be filled!”)

On Drop JButton:

If selected row:

Delete the selected row from the table and txt file

END



1. **Testing and Validation:**

1. Unit Testing: Each major function outlined in the pseudocode—such as logging in, adding students, removing students, and viewing all students—was tested individually to ensure correct operation.  
   
The login process was tested to verify that valid username and password inputs correctly transition to the dashboard, while invalid inputs display error messages. The addition of student records was tested to confirm that all fields (e.g., first name, last name, student number) are properly validated and saved to the students.txt file.   
  
Unit tests were conducted for the grade update function, ensuring that the system computes and updates grades (Prelim, Midterm, Final) in the table and TXT file.   
  
The removal function was checked to ensure the selected row is accurately removed from both the table and the TXT file.   
  
Validation tests were conducted to ensure that missing or incorrect data (e.g., blank fields, non-existent student numbers) trigger appropriate error messages, as specified in the pseudocode.  
  
2. Integration Testing:   
After unit testing, the different components were integrated to ensure they functioned together smoothly. This included: Ensuring that the navigation between windows (Login, Sign In, Dashboard, Add Students, Table to View All Students) was seamless, with no data loss during transitions.   
  
Verifying that updates made to student information are reflected across all relevant sections of the system, such as when a student was added, it shows both in the JTable and the TXT file.   
  
Ensuring consistent behavior of buttons (Submit, Drop) across the main dashboard window, with data being handled appropriately.  
   
3. User Acceptance Testing (UAT):   
The system was tested by end-users to ensure ease of use and proper functionality as outlined in the pseudocode.   
  
Users tested the workflow for adding students, and removing students to ensure the interface was intuitive and worked as expected.   
  
Test cases were designed to simulate common actions described in the pseudocode, such as logging in, submitting student information, and removing selected row.   
  
Users also intentionally entered incomplete or incorrect data (e.g., missing fields during student addition, invalid grades) to ensure the system provided meaningful error messages.   
  
4. Stress Testing:   
To test the system’s ability to handle a larger number of records, simulations were run by adding numerous student entries. The performance was monitored during operations such as updating grades and generating reports, especially focusing on how well the system handled reading from and writing to the students.txt file.

1. **Known Issues and Limitations:**

• Limitations

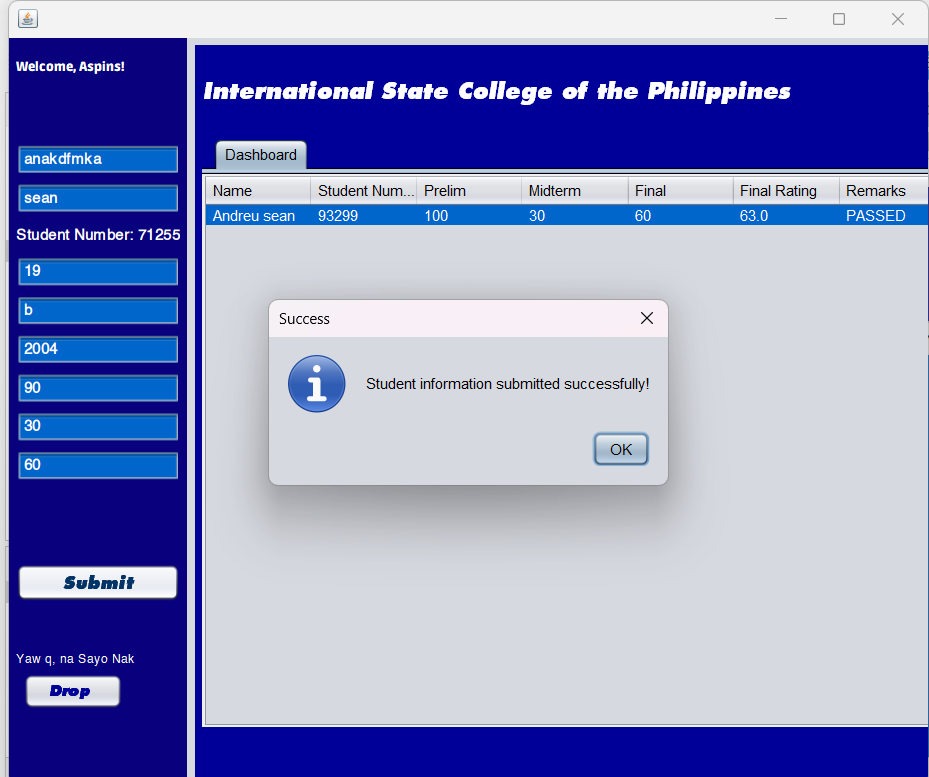
1. Only log in was available and no sign in for users and no file handling was used for the log in account
2. Grades cannot be edited or updated once the user submitted it
3. TXT file was the only database that holds the student information

• Recommendations

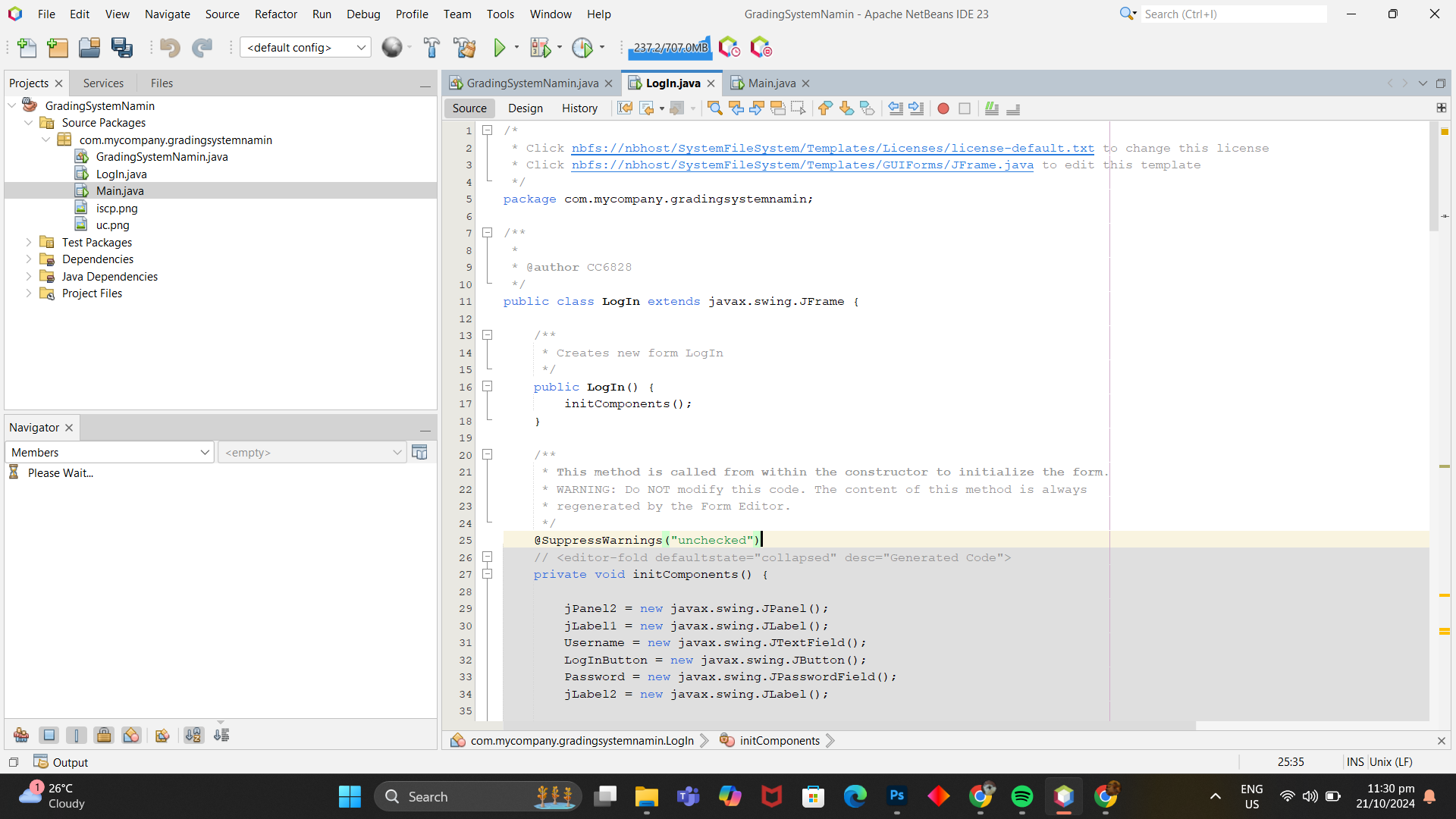
1. Add file handling for the log in account
2. Add sign in button and include file handling for it
3. Add edit student grades to avoid errors in inputting grades
4. Use other database tools to organize the data inserted in the system
5. **Appendices:**

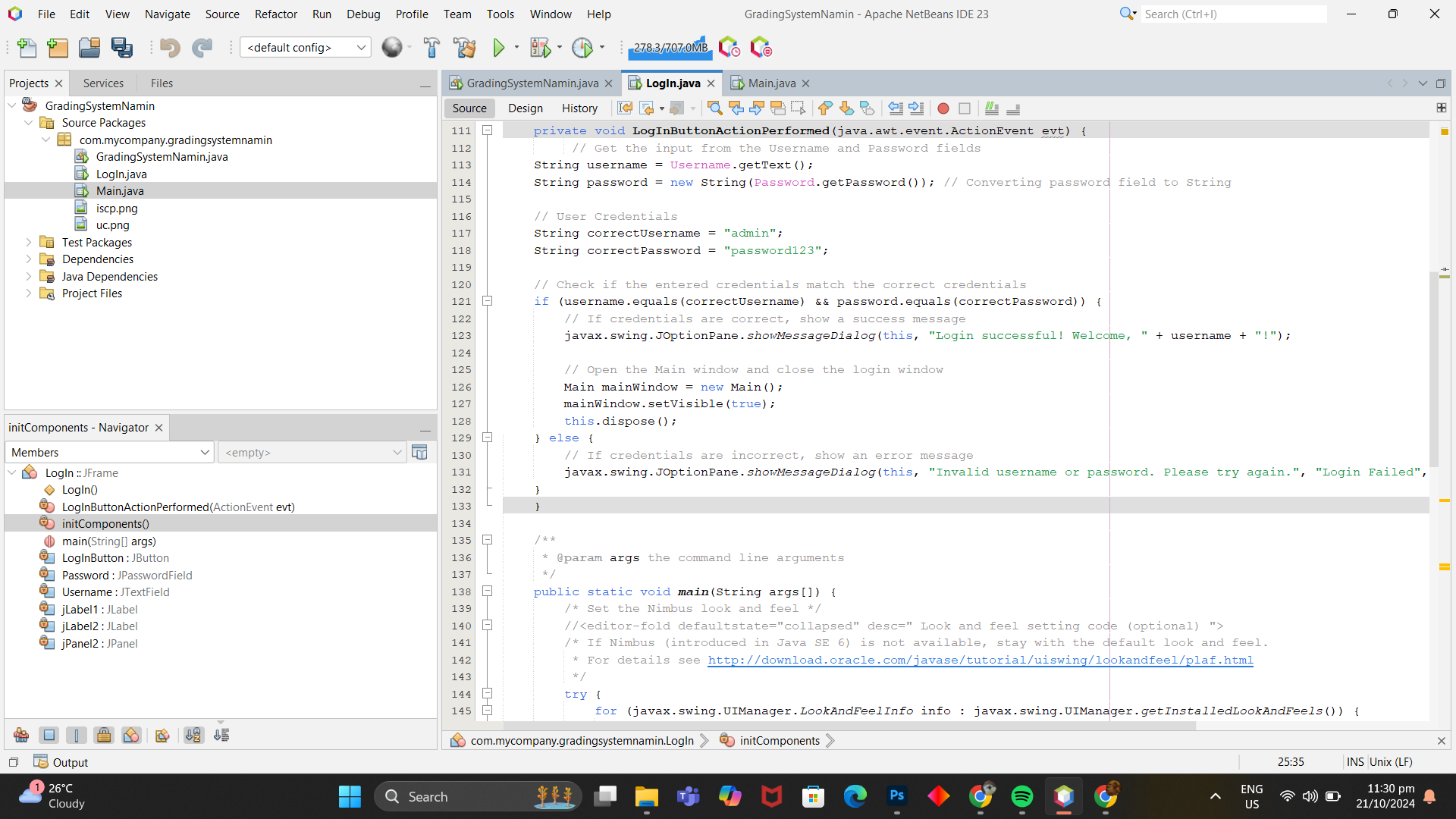
****

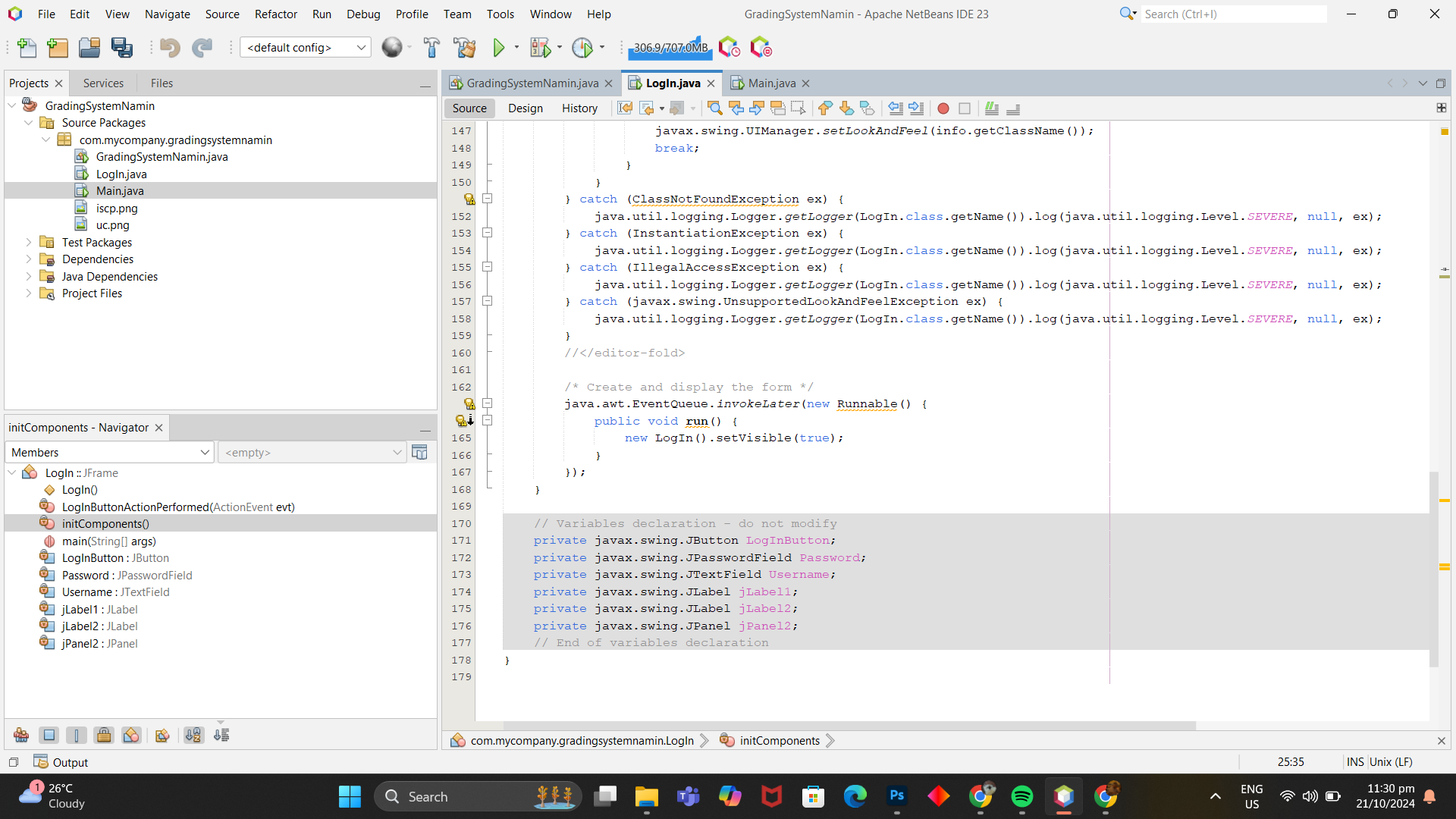
****

****

****

****

****

****

1. **Members:**

* Bejeran, Andrei Sean
* Calina, Bryan
* Castillo, Troy Angelo
* Celeste, Jhon Rey
* Malabanan, Ram Jhake