NATIONAL UNIVERSITY - MANILA
COLLEGE OF COMPUTING AND INFORMATION TECHNOLOGY

DATA STRUCTURE AND ALGORITHM

PREREQUISITE

(MINI [NUIS] STUDENT PORTAL)

FINAL PROJECT

Submitted by:

Rae S. Paulos

Submitted to:

Prof. Jensen A. Santillan

Table of Contents

NOTES AND OTHER COMMENTS

1. **Project Scope Clarification:** Although the file is named *"finaprojectcircularqueue"*, it is important to note that the project does not implement circular queues as per what was discussed during the consultation phase.
2. **Code Modifications:** Portions of the code automatically generated by NetBeans for JFrame components have been omitted to enhance readability and focus on custom implementations. For access to the complete version of the code, please contact me directly for the GitHub repository link.
3. **Project Focus:** While the program includes various features, the primary focus of the project is the *My Flowchart* section, which spans from pages 25 to 54. The content on pages 3 to 24 is provided to simulate a condensed version of the National University Information System (NUIS) student portal.

MY FLOWCHART CODE FUNCTIONALITY OVERVIEW

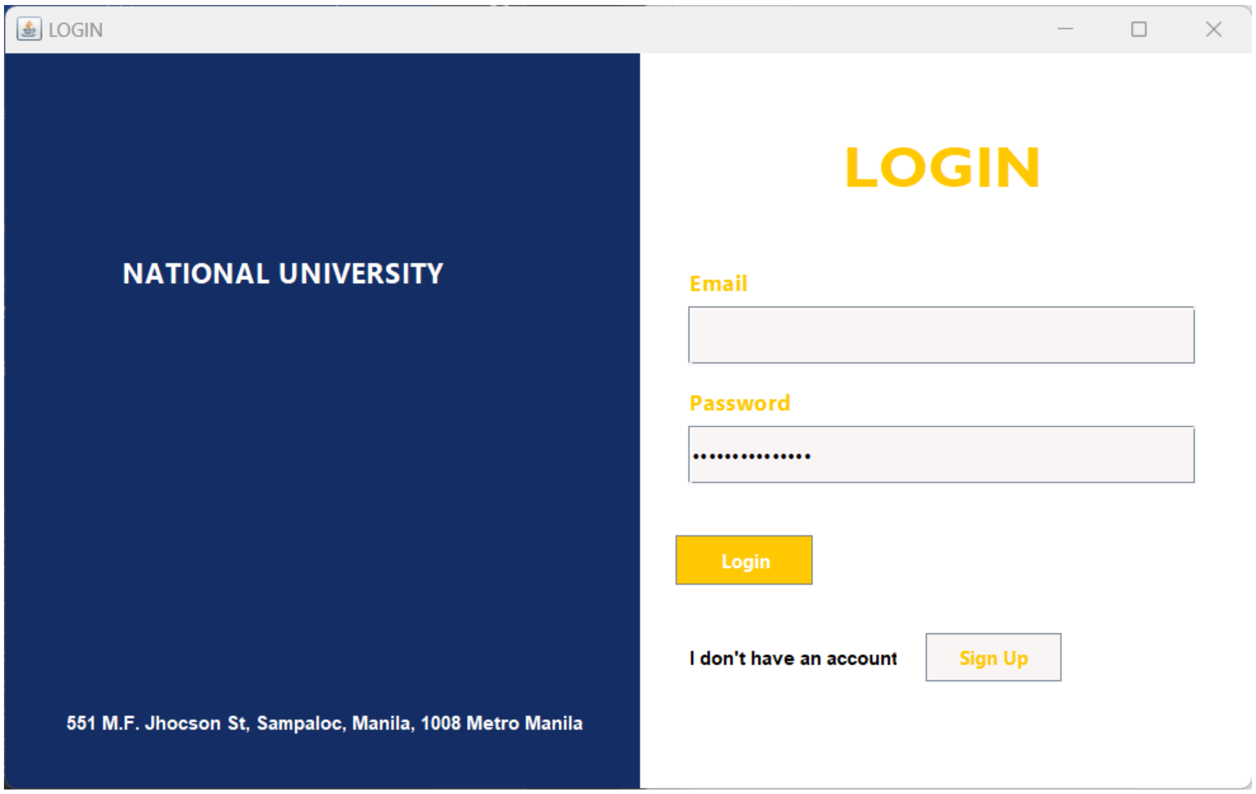
1. **Term Selection:** The system allows users to choose a term, with *Term 1* set as the default. If a user attempts to select a term without having submitted grades for the previous term, a pop-up window will alert them that access to that term is restricted until prerequisites are completed.
2. **Grade Input Mechanism:** Users can input grades for each course through a combo box with the following options: 4.0 to 1.0, *R* (Repeat), *Inc* (Incomplete), and *Drp* (Dropped).
3. **Print Functionality:** The system includes a *Print* button that outputs the student's flowchart, showcasing their academic progress.
4. **Interactive Course Boxes:** Each course code box features an event listener. When clicked, it opens a small pop-up window titled *Course Information*, displaying the course code, grade, and any prerequisite subjects associated with the course.
5. **NUIS-Inspired Visual Feedback:** The system follows the color-coding standards of the NUIS *My Flowchart*:
    - Red: Courses marked with an 'R' grade.
    - Blue: Courses that have not been taken.
    - White: Passed courses.
    - Yellow: Courses marked as incomplete.
    - Orange: Dropped courses.
6. **Prerequisite Management:** If a course in a selected term has an unpassed prerequisite from a previous term, a warning window will appear, notifying the user that grades for that course cannot be entered until the prerequisite is passed. Users must achieve a passing grade (greater or equal to 1.0) in prerequisite courses to proceed.
7. **Graduation Notification:** When all courses have been passed, a final pop-up window will notify the user that they are eligible for graduation.

FINALPROJECTCIRCULARQUEUE;

```
package finalprojectcircularqueue;

public class FinalProjectCircularQueue {


    public static void main(String[] args) {

        login LoginFrame = new login();
        LoginFrame.setVisible(true);
        LoginFrame.pack();
        LoginFrame.setLocationRelativeTo(null);

    }

    public boolean validateCredentials(String email, String password) {
        String validEmail = "fakeemail@gmail.com";
        String validPassword = "validPassword";

        return email.equals(validEmail) && password.equals(validPassword);
    }

}
```

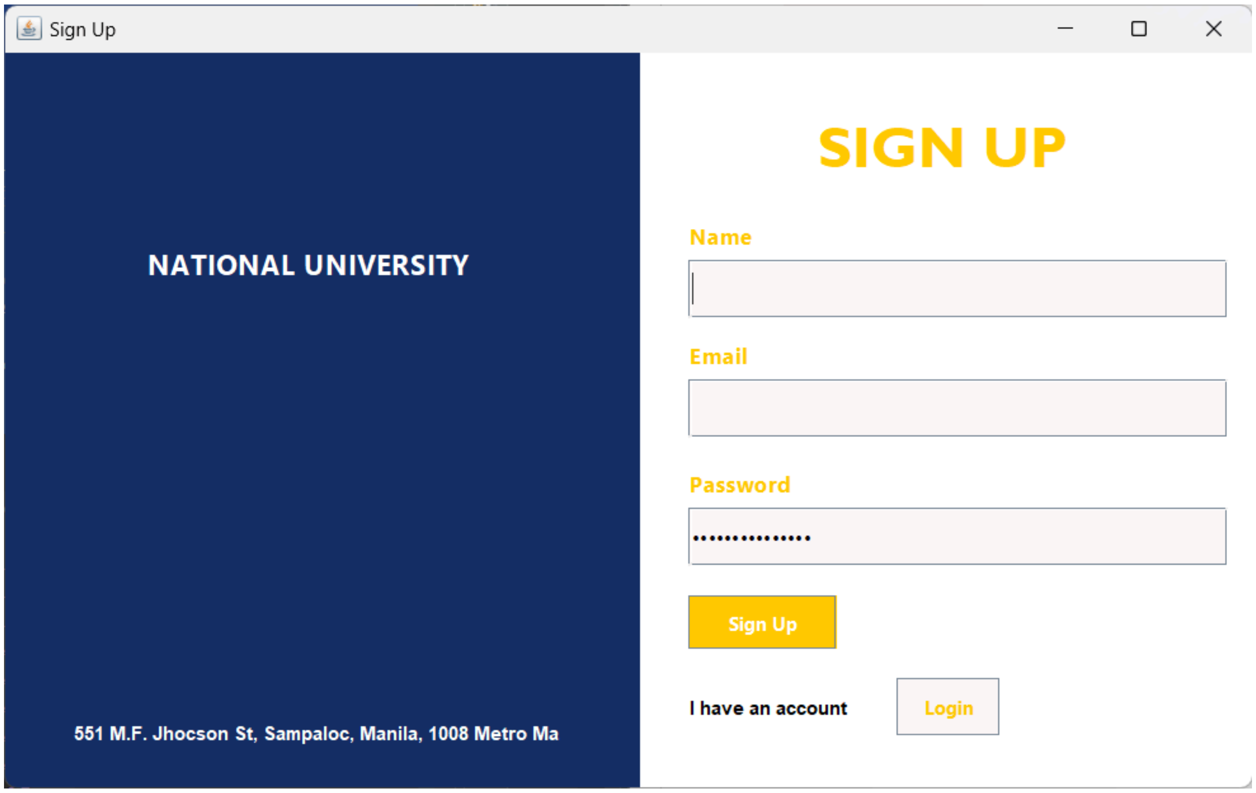
LOGIN PAGE

```
package finalprojectcircularqueue;

import javax.swing.JOptionPane;

public class login extends javax.swing.JFrame {


    public login() {
        initComponents();
    }

    public boolean validateCredentials(String email, String password) {
        String validEmail = "fakeemail@gmail.com";
        String validPassword = "validPassword";

        return email.equals(validEmail) && password.equals(validPassword);
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        //Sign up button in the login page
        SignUp SignUpFrame = new SignUp();
        SignUpFrame.setVisible(true);
        SignUpFrame.pack();
        SignUpFrame.setLocationRelativeTo(null);
        this.dispose();
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        //Email text field
        String enteredEmail = jTextField1.getText();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Login button
        String enteredEmail = jTextField1.getText();
        String enteredPassword = new String(jPasswordField1.getPassword());

        if (validateCredentials(enteredEmail, enteredPassword)) {
            JOptionPane.showMessageDialog(null, "Access authorized",
"Validated", JOptionPane.INFORMATION_MESSAGE);

            landingPage landingPage = new landingPage();
            landingPage.setVisible(true);
            landingPage.pack();
            landingPage.setLocationRelativeTo(null);
            this.dispose();

        } else {
            JOptionPane.showMessageDialog(this, "Inavlid email or
passowrd", "Login Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void jPasswordField1ActionPerformed(java.awt.event.ActionEvent 
evt) {
        // TODO add your handling code here:
        //Password
        String enteredPassword = new String(jPasswordField1.getPassword());
    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new login().setVisible(true);
            }
        });
    }
```



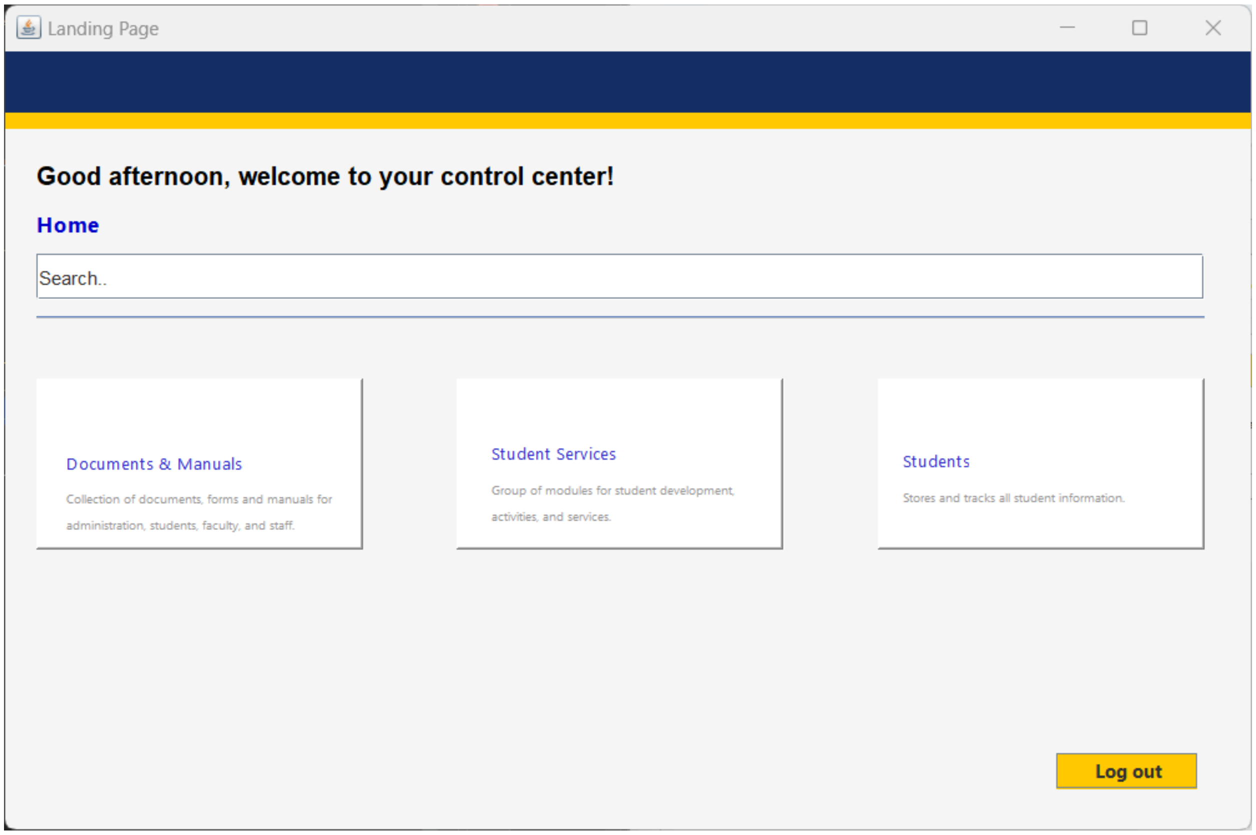
SIGN UP PAGE

```
package finalprojectcircularqueue;

import javax.swing.JOptionPane;

public class SignUp extends javax.swing.JFrame {

    public SignUp() {
        initComponents();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Sign up button
        JOptionPane.showMessageDialog(null, "Account created",
"Authorization", JOptionPane.INFORMATION_MESSAGE);

        landingPage landingPage = new landingPage();
        landingPage.setVisible(true);
        landingPage.pack();
        landingPage.setLocationRelativeTo(null);
        this.dispose();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        login loginFrame = new login();
        loginFrame.setVisible(true);
        loginFrame.pack();
        loginFrame.setLocationRelativeTo(null);
        this.dispose();
    }
}
```



LANDING PAGE

```
package finalprojectcircularqueue;

public class landingPage extends javax.swing.JFrame {

    public landingPage() {
        initComponents();
    }

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Students - Stores and tracks all student information.
        students students = new students();
        students.setVisible(true);
        students.pack();
        students.setLocationRelativeTo(null);
        this.dispose();
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Log out button
        login loginFrame = new login();
        loginFrame.setVisible(true);
        loginFrame.pack();
        loginFrame.setLocationRelativeTo(null);
        this.dispose();
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Student Services
        errorCode ErrorCode = new errorCode();
        ErrorCode.setVisible(true);
        ErrorCode.pack();
        ErrorCode.setLocationRelativeTo(null);
        this.dispose();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Documents and manuals
        errorCode ErrorCode = new errorCode();
        ErrorCode.setVisible(true);
        ErrorCode.pack();
        ErrorCode.setLocationRelativeTo(null);
        this.dispose();
    }

public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new landingPage().setVisible(true);
            }
        });
    }
}
```


Landing Page

Good afternoon, welcome to your control center!

Home

Search..

Documents & Manuals

Collection of documents, forms and manuals for administration, students, faculty, and staff.

Student Services

Group of modules for student development, activities, and services.

Students

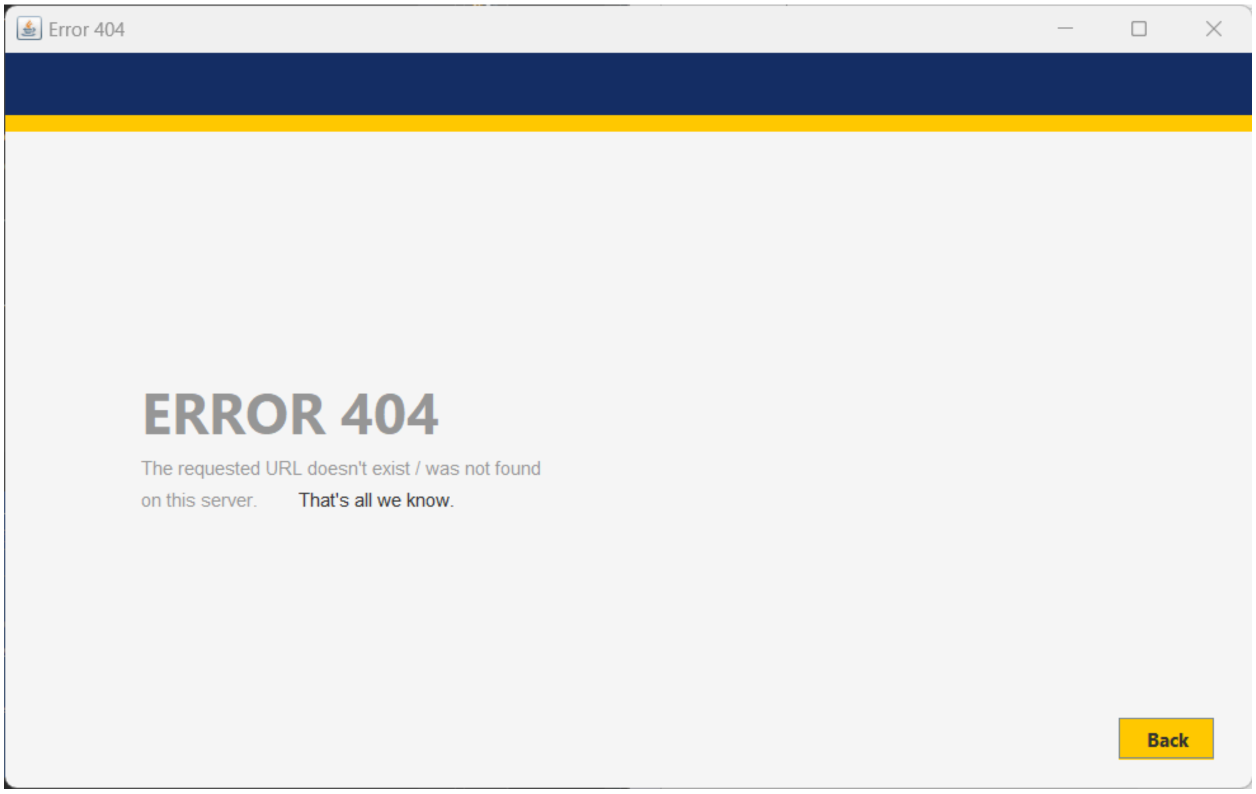
Stores and tracks all student information.

Log out

DOCUMENTS & MANUALS

Error page placeholder.

```
package finalprojectcircularqueue;

public class errorCode extends javax.swing.JFrame {

    public errorCode() {
        initComponents();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Back button
        landingPage LandingPage = new landingPage();
        LandingPage.setVisible(true);
        LandingPage.pack();
        LandingPage.setLocationRelativeTo(null);
        this.dispose();
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new errorCode().setVisible(true);
            }
        });
    }
```



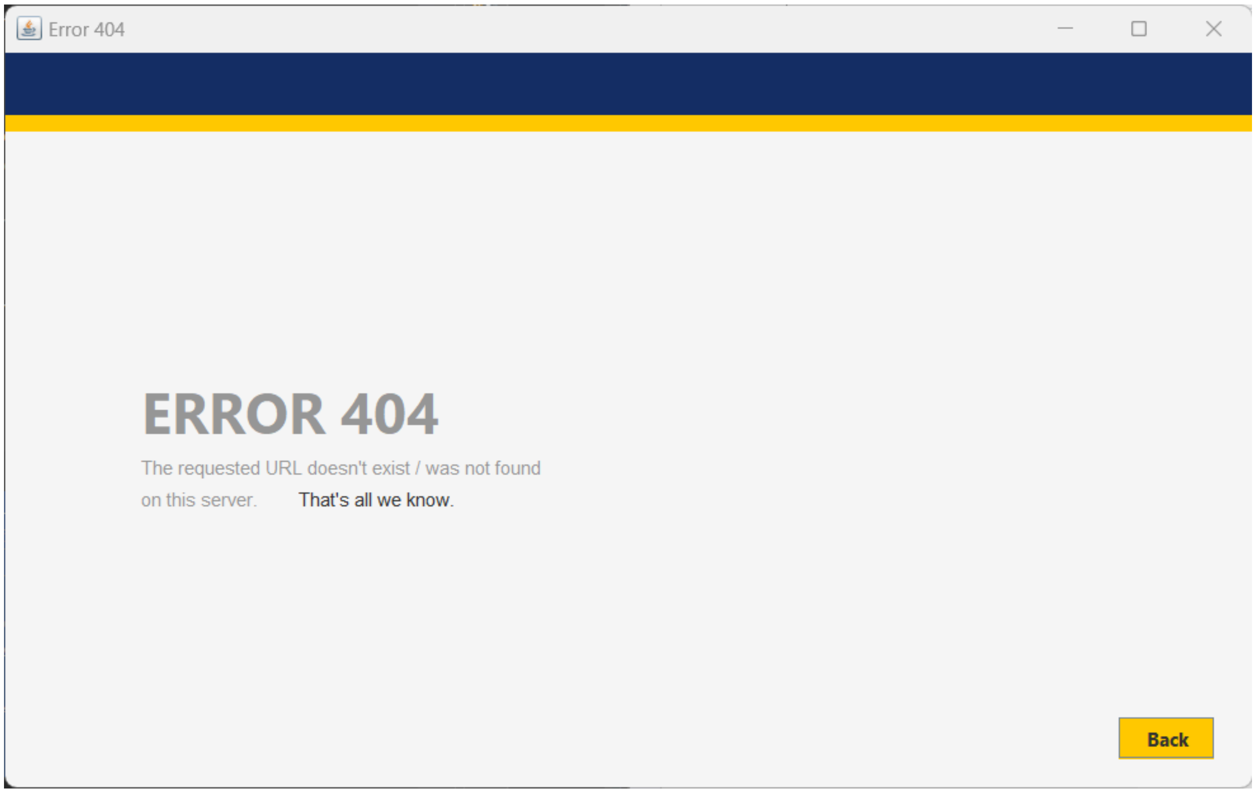
ERROR 404

The requested URL doesn't exist / was not found on this server. That's all we know.

STUDENT SERVICES

Error page placeholder.

```
package finalprojectcircularqueue;

public class errorCode extends javax.swing.JFrame {
    public errorCode() {
        initComponents();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Back button
        landingPage LandingPage = new landingPage();
        LandingPage.setVisible(true);
        LandingPage.pack();
        LandingPage.setLocationRelativeTo(null);
        this.dispose();
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new errorCode().setVisible(true);
            }
        });
    }
}
```

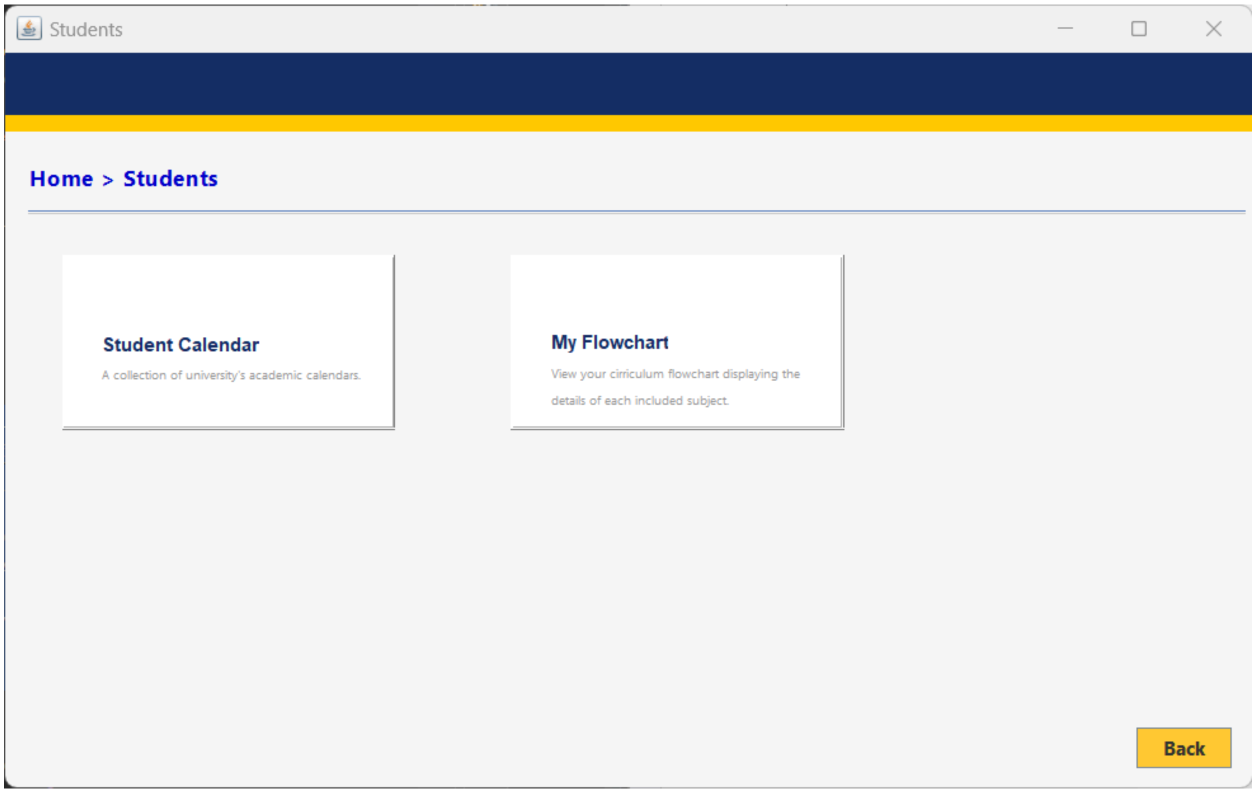

Error 404

ERROR 404

The requested URL doesn't exist / was not found on this server. That's all we know.

STUDENT CALENDAR & FLOWCHART LANDING PAGE

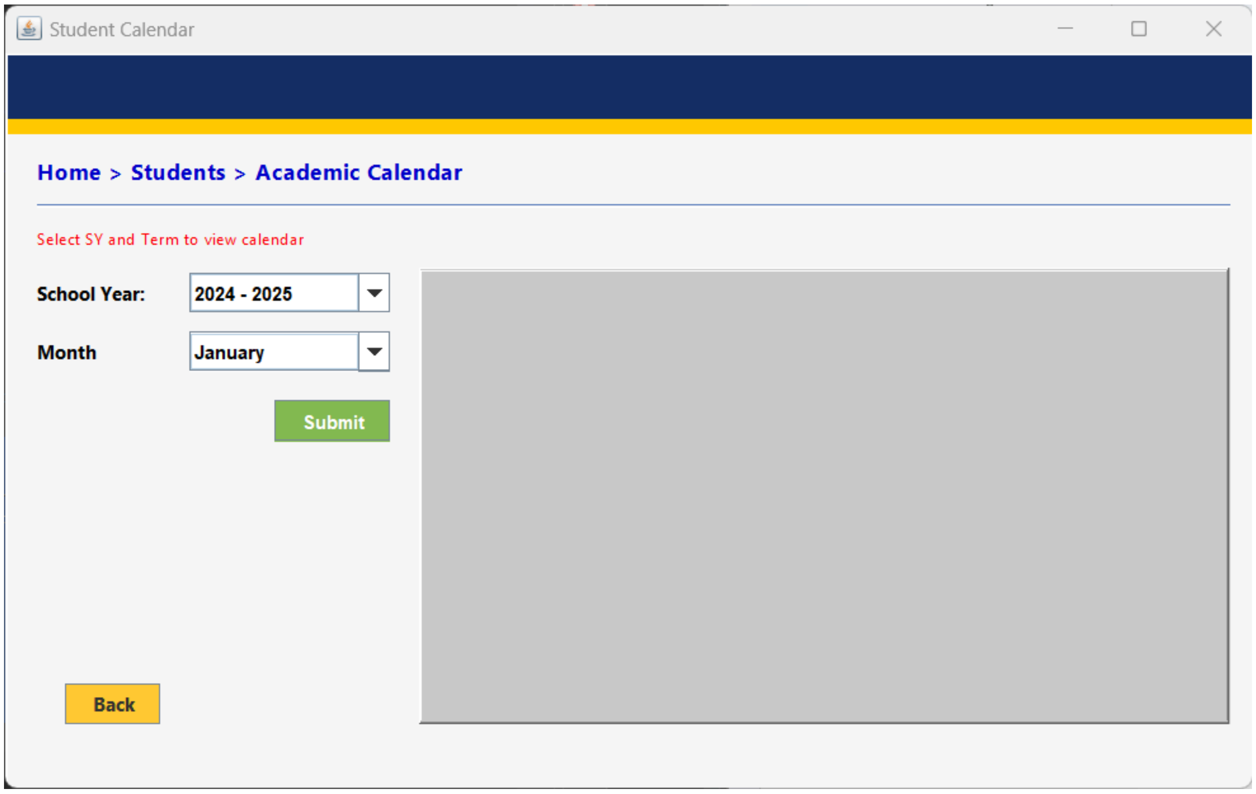
```
package finalprojectcircularqueue;

public class students extends javax.swing.JFrame {

    public students() {
        initComponents();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        myFlowChart MyFlowChart = new myFlowChart();
        MyFlowChart.setVisible(true);
        MyFlowChart.pack();
        MyFlowChart.setLocationRelativeTo(null);
        this.dispose();
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Student Calendar
        studentCalendar StudentCalendar = new studentCalendar();
        StudentCalendar.setVisible(true);
        StudentCalendar.pack();
        StudentCalendar.setLocationRelativeTo(null);
        this.dispose();
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Back button
        landingPage landingPage = new landingPage();
        landingPage.setVisible(true);
        landingPage.pack();
        landingPage.setLocationRelativeTo(null);
        this.dispose();
    }

public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new students().setVisible(true);
            }
        });
    }
}
```


Students
Home > Students
Student Calendar
A collection of university's academic calendars.
My Flowchart
View your cirriculum flowchart displaying the details of each included subject.
Back

STUDENT CALENDAR

```
package finalprojectcircularqueue;

import javax.swing.JOptionPane;

public class studentCalendar extends javax.swing.JFrame {


    private static januaryCalendar JanuaryCalendar = new januaryCalendar();
    private static februaryCalendar FebruaryCalendar = new
februaryCalendar();
    private static marchCalendar MarchCalendar = new marchCalendar();
    private static aprilCalendar AprilCalendar = new aprilCalendar();
    private static mayCalendar MayCalendar = new mayCalendar();
    private static juneCalendar JuneCalendar = new juneCalendar();
    private static julyCalendar JulyCalendar = new julyCalendar();
    private static augustCalendar AugustCalendar = new augustCalendar();
    private static septemberCalendar SeptemberCalendar = new
```

```
septemberCalendar();
    private static octoberCalendar OctoberCalendar = new octoberCalendar();
    private static novemberCalendar NovemberCalendar = new
novemberCalendar();
    private static decemberCalendar DecemberCalendar = new
decemberCalendar();

    public studentCalendar() {
        initComponents();
    }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Back button
        students students = new students();
        students.setVisible(true);
        students.pack();
        students.setLocationRelativeTo(null);
        this.dispose();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Submit button
        String selectedMonth = (String) jComboBox2.getSelectedItem();
        String message = "You selected: " + selectedMonth;
        JOptionPane.showMessageDialog(this, message, "Term",
JOptionPane.INFORMATION_MESSAGE);

        switch (selectedMonth) {

            case "January":
                handleJanuary();
                this.dispose();
                break;

            case "February":
                handleFebruary();
                this.dispose();
                break;

            case "March":
                handleMarch();
                this.dispose();
                break;

            case "April":
                handleApril();
                this.dispose();
                break;

            case "May":
                handleMay();
                this.dispose();
                break;

            case "June":
                handleJune();
                this.dispose();
                break;

            case "July":
                handleJuly();
                this.dispose();
                break;

            case "August":
                handleAugust();
                this.dispose();
                break;

            case "September":
                handleSeptember();
                this.dispose();
                break;

            case "October":
                handleOctober();
                this.dispose();
                break;

            case "November":
                handleNovember();
                this.dispose();
                break;

            case "December":
                handleDecember();
                this.dispose();
                break;

            default:
                break;
        }
    }


    private void handleJanuary() {
    JanuaryCalendar.setVisible(true);
    JanuaryCalendar.pack();
    JanuaryCalendar.setLocationRelativeTo(null);
    }

    private void handleFebruary() {
        FebruaryCalendar.setVisible(true);
        FebruaryCalendar.pack();
        FebruaryCalendar.setLocationRelativeTo(null);
    }

    private void handleMarch() {
        MarchCalendar.setVisible(true);
        MarchCalendar.pack();
        MarchCalendar.setLocationRelativeTo(null);
    }

    private void handleApril() {
        AprilCalendar.setVisible(true);
        AprilCalendar.pack();
        AprilCalendar.setLocationRelativeTo(null);
    }

    private void handleMay() {
        MayCalendar.setVisible(true);
        MayCalendar.pack();
        MayCalendar.setLocationRelativeTo(null);
    }

    private void handleJune() {
        JuneCalendar.setVisible(true);
        JuneCalendar.pack();
        JuneCalendar.setLocationRelativeTo(null);
    }

    private void handleJuly() {
        JulyCalendar.setVisible(true);
        JulyCalendar.pack();
        JulyCalendar.setLocationRelativeTo(null);
    }

    private void handleAugust() {
        AugustCalendar.setVisible(true);
        AugustCalendar.pack();
        AugustCalendar.setLocationRelativeTo(null);
    }

    private void handleSeptember() {
        SeptemberCalendar.setVisible(true);
        SeptemberCalendar.pack();
        SeptemberCalendar.setLocationRelativeTo(null);
    }

    private void handleOctober() {
        OctoberCalendar.setVisible(true);
        OctoberCalendar.pack();
        OctoberCalendar.setLocationRelativeTo(null);
    }

    private void handleNovember() {
        NovemberCalendar.setVisible(true);
        NovemberCalendar.pack();
        NovemberCalendar.setLocationRelativeTo(null);
    }

    private void handleDecember() {
        DecemberCalendar.setVisible(true);
        DecemberCalendar.pack();
        DecemberCalendar.setLocationRelativeTo(null);
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new studentCalendar().setVisible(true);
            }
        });
    }
}
```

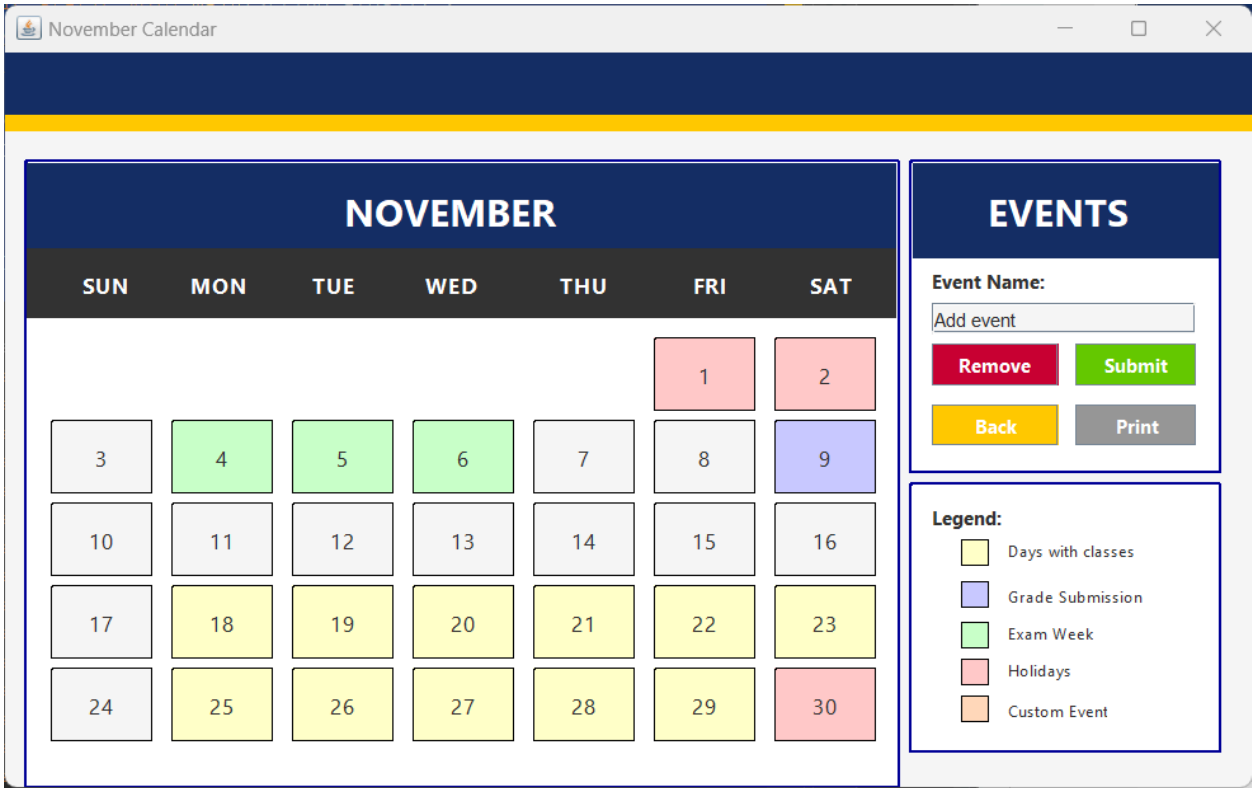

Student Calendar
Home > Students > Academic Calendar
Select SY and Term to view calendar
School Year:
2024 - 2025
Month
January
Submit
Back

CALENDAR EXAMPLE (NOVEMBER)

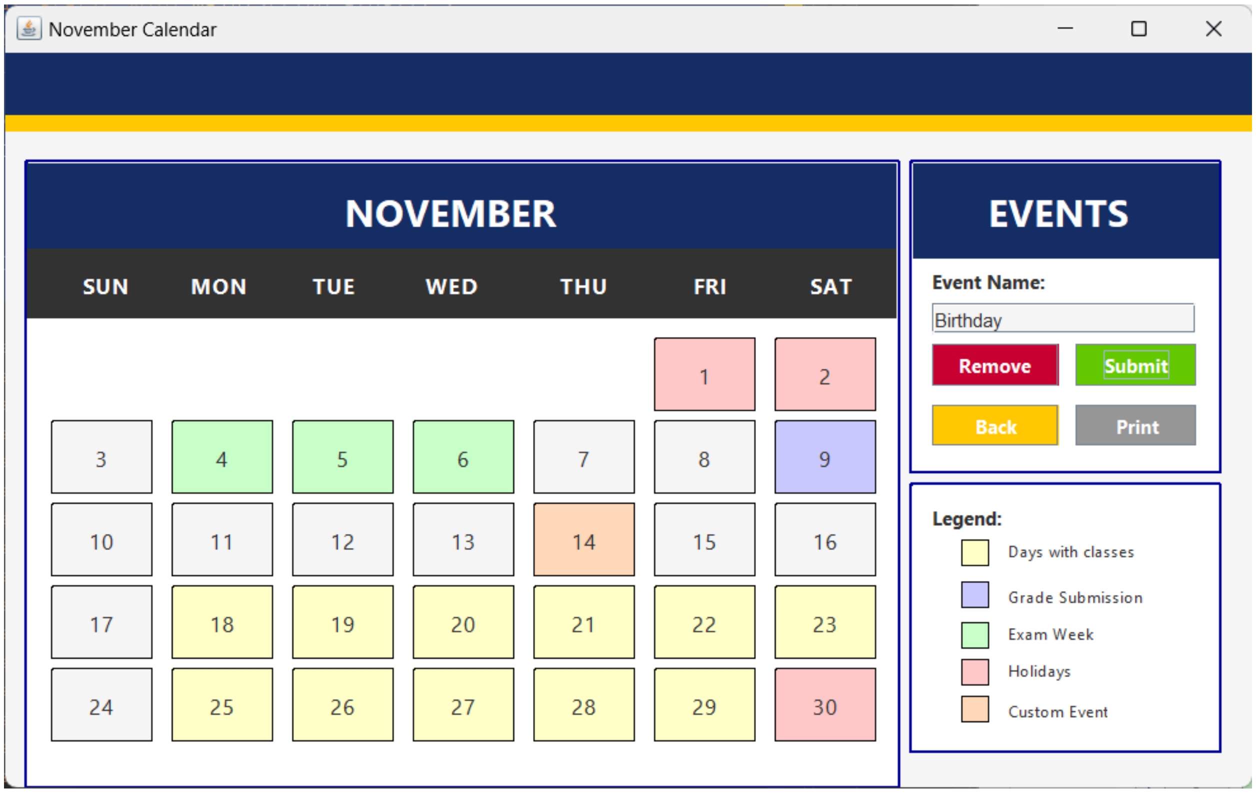
Other months are removed from this file to shorten the number of pages included. Original code accounts for the whole academic year 2024 - 2025.

```
package finalprojectcircularqueue;

import java.util.LinkedList;
import javax.swing.JOptionPane;
import java.awt.Color;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
```

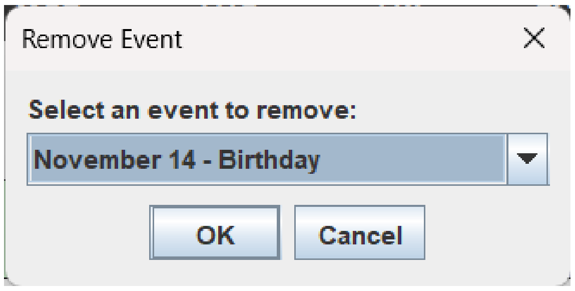
```
public class novemberCalendar extends javax.swing.JFrame {

    private LinkedList<String> eventsList = new LinkedList<>();  //
LinkedList to store formatted events
    private int selectedDay = 0;
    private studentCalendar StudentCalendar;

    public novemberCalendar() {
        initComponents();
        StudentCalendar = new studentCalendar();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Button day 1 (there will be more buttons 1 to 30 but for the sake
of simplicity, this will do for now)
        selectedDay = 1;
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Sumbmit button
        //Takes in the selectedDay value and the string in the text field.
        //Add both in a linked list. Format the string as "<selectedDay> -
<augustEvent>"
        String augustEvent = jTextField1.getText();

        if (augustEvent.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter an event
description.", "Input Error", JOptionPane.ERROR_MESSAGE);
        } else if (selectedDay == 0) {
            JOptionPane.showMessageDialog(this, "Please select a day.",
"Input Error", JOptionPane.ERROR_MESSAGE);
        } else {
            String formattedEvent = ("November " + selectedDay + " - " +
augustEvent);
            eventsList.add(formattedEvent);

            for (String event : eventsList) {
                System.out.println(event);
            }
        }
```



```
JButton[] buttons = {
    jButton1, jButton5, jButton6, jButton7, jButton8,
    jButton9, jButton10, jButton11, jButton12, jButton13,
    jButton14, jButton15, jButton16, jButton17, jButton18,
    jButton19, jButton20, jButton21, jButton22, jButton23,
    jButton24, jButton25, jButton26, jButton27, jButton28,
    jButton29, jButton30, jButton31, jButton32, jButton33
};

// Change color of the day
if (selectedDay >= 1 && selectedDay <= 30) {
    buttons[selectedDay - 1].setBackground(Color.decode("#FFDBBB"));
} else {
    System.out.println("Invalid day selected");
}
```



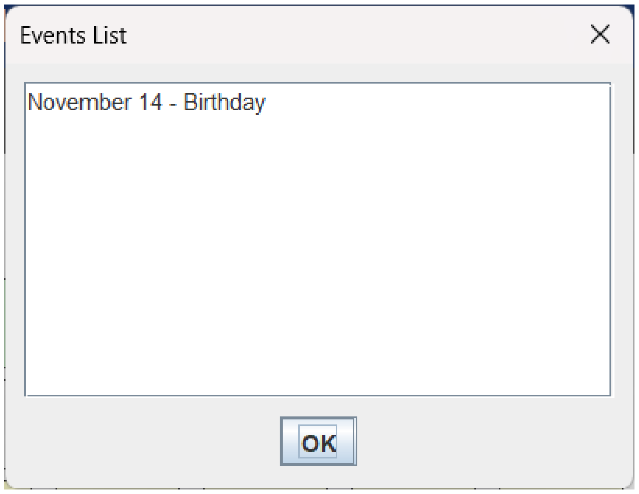
```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Remove button
        if (eventsList.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No events to remove.", "No
Events", JOptionPane.PLAIN_MESSAGE);
        } else {
            // Create a list of events as an array to display in the dialog
            String[] eventsArray = eventsList.toArray(new String[0]);

            // Show dialog with options to select an event to remove
            String eventToRemove = (String) JOptionPane.showInputDialog(
                this,
                "Select an event to remove:",
                "Remove Event",
                JOptionPane.PLAIN_MESSAGE,
                null,
                eventsArray,
                eventsArray[0] // default selection
            );
            // If an event was selected (and not cancelled)
            if (eventToRemove != null) {
                eventsList.remove(eventToRemove);
                JOptionPane.showMessageDialog(this, "Event removed: " +
eventToRemove, "Success", JOptionPane.PLAIN_MESSAGE);
            }
        }
    }
```



Select an event to remove:

November 14 - Birthday

```
   private void jButton35ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        StudentCalendar.setVisible(true);
        StudentCalendar.pack();
        StudentCalendar.setLocationRelativeTo(null);
        this.setVisible(false);
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Print button

        if (eventsList.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No events to display.",
"No Events", JOptionPane.PLAIN_MESSAGE);
        } else {
            // Create a JTextArea to display the events
            JTextArea textArea = new JTextArea(10, 30); // 10 rows, 30
columns
            textArea.setEditable(false); // Make it read-only

            // Append each event to the text area
            for (String event : eventsList) {
                textArea.append(event + "\n");
            }

            // Create a JScrollPane to make the text area scrollable
            JScrollPane scrollPane = new JScrollPane(textArea);

            // Show the events in a dialog
            JOptionPane.showMessageDialog(this, scrollPane, "Events List",
JOptionPane.PLAIN_MESSAGE);
        }
    }
```



November 14 - Birthday

```
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new firstTermCalendar().setVisible(true);
            }
        });
    }
}
```

MY FLOWCHART

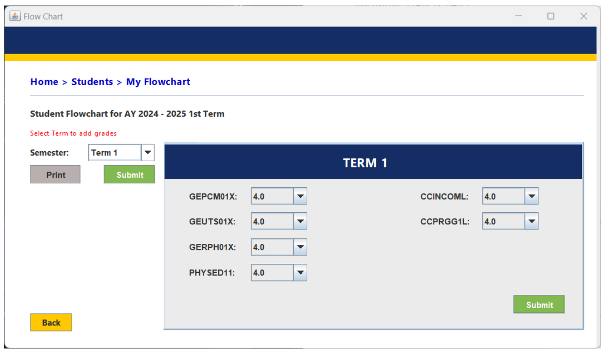
```
package finalprojectcircularqueue;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.util.Arrays;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import javax.swing.BorderFactory;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;

public class myFlowChart extends javax.swing.JFrame {
    private int selectedIndex;

    public myFlowChart() {
        initComponents();
        hideTabHeaders();
    }

    private void hideTabHeaders() {
        jTabbedPane2.setUI(new javax.swing.plaf.basic.BasicTabbedPaneUI() {
            @Override
            protected int calculateTabAreaHeight(int tabPlacement, int
runCount, int maxTabHeight) {
                return 0;  // Set tab area height to 0, effectively hiding
the tabs
            }
        });
    }
    private String[][] selectedGrades = new String[12][6];
    private boolean[][] gradeResults = new boolean[12][6];
    private boolean isInitialized = false;
    private static int a = 0;
    private final LinkedHashMap<String, String> courseGradeMap = new
LinkedHashMap<>();
    private final LinkedHashMap<String, String> preReqMap = new
LinkedHashMap<>();

    private final String[][] courses = {
            {"GEPCM01X", "GEUTS01X", "GERPH01X", "PHYSED11", "CCINCOML",
"CCPRGG1L"},   //Term 1
            {"GECTW01X", "GEMMW01X", "GESTS01X", "PHYSED12", "CTHASOPL",
"CCPRGG2L"},   //Term 2
            {"GEETH01X", "GEENT01X", "MCWTS01X", "PHYSED13", "CCDISTR1",
"CCOBJPGL"},   //Term 3
            {"GEFID01X", "CCMATAN1", "PHYSED14", "MCWTS02X", "CCDISTR2",
"CCDATRCL"},   //Term 4
            {"GEACM01X", "CCMATAN2", "MCNAT01R", "CTINFMGL", "CCPHYS1L",
"CCOMPORG"},   //Term 5
            {"CCQUAMET", "CTADVDBL", "CCALCOMP", "CTBASNTL", "CCPHYS2L"},
//Term 6
            {"CCSFEN1L", "CCAUTOMA", "CCOPSYSL", "CCMACLRL", "GERIZ01X"},
//Term 7
            {"CCADMACL", "CCSFEN2L", "CTINASSL", "GEITE01X"},   //Term 8
            {"CCINTHCI", "CTAPDEVL", "CCDEPLRL", "CCMETHOD"},   //Term 9
            {"CCTHESS1", "CTPRFISS", "CCPGLANG", "CCRNFLRL"},   //Term 10
            {"CCTHESS2", "CCDATSCL"},   //Term 11
            {"CCINTERN"}    //Term 12
    };

    //Flowchart window
    public void finalFlowchart() {

        addPreReqValues();
        addGradeToHashMap();

        JFrame flowchartFrame = new JFrame("My Flowchart");
        flowchartFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        flowchartFrame.setSize(700, 340);
        flowchartFrame.setLocationRelativeTo(null);

        flowchartFrame.getContentPane().setBackground(new Color(247, 247,
247));

        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();

        // Loop through terms and courses
        for (int term = 0; term < courses.length; term++) {
            gbc.gridx = term; // Set the column for the term
            gbc.gridy = 0;    // Reset row index for each term

            // Create a panel for the term label
            JPanel termPanel = new JPanel();
            termPanel.setBackground(new Color(0, 66, 118));
            termPanel.setPreferredSize(new Dimension(120, 40));
            termPanel.setLayout(new GridBagLayout());

termPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, 1));

            // Add a label for each term
            JLabel termLabel = new JLabel("Term " + (term + 1),
JLabel.CENTER);
            termLabel.setFont(new Font("Arial", Font.BOLD, 16));
            termLabel.setForeground(Color.WHITE);

            // Center the term label
            GridBagConstraints labelGbc = new GridBagConstraints();
            labelGbc.gridx = 0;
            labelGbc.gridy = 0;
            labelGbc.weightx = 1;
            labelGbc.weighty = 1;
            labelGbc.anchor = GridBagConstraints.CENTER;

            termPanel.add(termLabel, labelGbc); // Add the label to the
term panel

            mainPanel.add(termPanel, gbc); // Add the term panel to the
main panel

            // Add course codes for the current term
            for (String courseCode : courses[term]) {
                gbc.gridy++; // New row for each course code
                gbc.insets = new Insets(0, 0, 0, 0); // Remove spacing
```

```
between components
                JPanel courseBox = createCourseBox(courseCode); // Create
the course box
                mainPanel.add(courseBox, gbc);
            }
        }
        flowchartFrame.add(new JScrollPane(mainPanel));
        flowchartFrame.setVisible(true);
    }

    //Adds design to the flowchart
    private JPanel createCourseBox(String courseCode) {
        JPanel courseBox = new JPanel();

        // Retrieve the grade from the HashMap
        String grade = courseGradeMap.getOrDefault(courseCode, "Grade not
available");

        if (grade == null || grade.isEmpty()) {
            courseBox.setBackground(new Color(179, 224, 255));
        } else {
            courseBox.setBackground(switchColors(grade));
        }

        // Set the background color based on the grade
        courseBox.setBackground(switchColors(grade));

courseBox.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, 1));
// Grey border
        courseBox.setPreferredSize(new Dimension(120, 40)); // Set
preferred size of the course box

        JLabel label = new JLabel(courseCode, JLabel.CENTER);
        label.setHorizontalAlignment(SwingConstants.CENTER); // Center text
horizontally
        label.setVerticalAlignment(SwingConstants.CENTER);   // Center text
vertically

        // Set layout to center the label
        courseBox.setLayout(new GridBagLayout());
        courseBox.add(label); // Add the label to the course box

        // Add a mouse click listener to the course box
        courseBox.addMouseListener(new java.awt.event.MouseAdapter() {
            @Override
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                // Show a popup window when the course box is clicked
                String preReq = preReqMap.get(courseCode);
                String preReqMessage = (preReq != null) ? preReq : "No
prerequisite";

                // Show a popup window when the course box is clicked
                String message = "Course: " + courseCode + "\nGrade: " +
grade + "\nPrerequisite: " + preReqMessage;

                JOptionPane.showMessageDialog(courseBox,
                    message,
                    "Course Information", JOptionPane.INFORMATION_MESSAGE);
            }
        });
        return courseBox;
    }

    //Switches the color of the courses based on the grades attached to the
keys
    private Color switchColors (String grade) {
        switch (grade) {
            case "4.0":
            case "3.5":
            case "3.0":
            case "2.5":
            case "2.0":
            case "1.5":
            case "1.0":
                return new Color(255, 255, 255); //White for passed subject
            case "R":
                return new Color(255, 102, 102); //Red for failed subject
            case "Inc":
                return new Color(255, 255, 153); //Yellow for incomplete
subjects
            case "Drp":
                return new Color(252, 195, 104); //Orange for droped
subjects
            default:
                return new Color(179, 224, 255); //Blue for subject not yet
taken
```

```
        }
    }

    //Adds the grades to the LinkedHashMap
    private void addGradeToHashMap() {
        // Adding keys and values to the LinkedHashMap
        for (int i = 0; i < courses.length; i++) {
            for (int j = 0; j < courses[i].length; j++) {
                // Add to map if the course code is not empty
                if (!courses[i][j].trim().isEmpty()) {
                    courseGradeMap.put(courses[i][j],
selectedGrades[i][j]);
                }
            }
        }

        // Printing out the LinkedHashMap to verify
        for (String key : courseGradeMap.keySet()) {
            System.out.println("Course: " + key + ", Grade: " +
courseGradeMap.get(key));
        }
    }

    private void addPreReqValues() {
        //Term 2 preRequisites
        preReqMap.put("CCPRGG2L", "CCPRGG1L");
        preReqMap.put("CTHASOPL", "CCINCOML");

        //Term 3 preRequisites
        preReqMap.put("CCOBJPGL", "CCPRGG2L");

        //Term 4 preRequisites
        preReqMap.put("CCDATRCL", "CCPRGG2L");
        preReqMap.put("CCDISTR2", "CCDISTR1");
        preReqMap.put("CCMATAN1", "CCDISTR1");

        //Term 5 preRequisites
        preReqMap.put("CCOMPORG", "CCOBJPGL");
        preReqMap.put("CCPHYS1L", "CCMATAN1");
        preReqMap.put("CTINFMGL", "CCDATRCL");
        preReqMap.put("CCMATAN2", "CCMATAN1");

        //Term 6 preRequisites
        preReqMap.put("CCQUAMET", "CCDISTR1");
        preReqMap.put("CTADVDBL", "CTINFMGL");
        preReqMap.put("CCALCOMP", "CCDATRCL");
        preReqMap.put("CTBASNTL", "CCOMPORG");
        preReqMap.put("CCPHYS2L", "CCPHYS1L");

        //Term 7 preRequisites
        preReqMap.put("CCSFEN1L", "CCDATRCL");
        preReqMap.put("CCAUTOMA", "CCALCOMP");
        preReqMap.put("CCOPSYSL", "CCOBJPGL");

        //Term 8 preRequisites
        preReqMap.put("CCADMACL", "CCMACLRL");
        preReqMap.put("CCSFEN2L", "CCSFEN1L");
        preReqMap.put("CTINASSL", "CTINFMGL");

        //Term 9 preRequisites
        preReqMap.put("CCINTHCI", "CCPRGG2L");
        preReqMap.put("CTAPDEVL", "CCOBJPGL");
        preReqMap.put("CCDEPLRL", "CCADMACL");
        preReqMap.put("CCMETHOD", "CCSFEN1L");

        //Term 10 preRequisites
        preReqMap.put("CCTHESS1", "CCMETHOD");
        preReqMap.put("CCPGLANG", "CCDATRCL");
        preReqMap.put("CCRNFLRL", "CCDEPLRL");

        //Term 11 preRequisites
        preReqMap.put("CCTHESS2", "CCTHESS1");
        preReqMap.put("CCDATSCL", "CCRNFLRL");

        //Term 12 preRequisites
        preReqMap.put("CCINTERN", "CCTHESS1");
    }

    //Inputs dummy grade for each course
    private void initializeArray() {
        for (int i = 0; i < selectedGrades.length; i++) {
            for (int j = 0; j < selectedGrades[i].length; j++) {
                selectedGrades[i][j] = "0";  // Assign "0" to each element
            }
        }
    }
    //
    private void storeSelectedGrades(int termIndex, JComboBox<String>[]
comboBoxes) {
        for (int i = 0; i < comboBoxes.length; i++) {
            selectedGrades[termIndex][i] = (String)
comboBoxes[i].getSelectedItem();
        }
        checkGradeValidity(termIndex);

        System.out.println("Term " + (termIndex + 1) + (" Grades: " +
Arrays.toString(selectedGrades[termIndex]) + "storesSelectedGrades\n"));
    }

    //
    private void checkGradeValidity(int termIndex) {
        for (int i = 0; i < selectedGrades[termIndex].length; i++) {
            String grade = selectedGrades[termIndex][i];
            gradeResults[termIndex][i] = isValidGrade(grade);
        }
    }

    //A method that checks if you can access the next term
    private boolean isEmpty(int termIndex) {
        boolean[] defaultArray = new boolean[6];
        return Arrays.equals(gradeResults[termIndex], defaultArray);
    }

    //
    private boolean isValidGrade(String grade) {
        switch(grade) {
            case "4.0":
            case "3.5":
            case "3.0":
            case "2.5":
            case "2.0":
            case "1.5":
            case "1.0":
                return true;
            case "R":
            case "Inc":
            case "Drp":
                return false;
            default:
                return false;
        }
    }

    //Back button
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        students students = new students();
        students.setVisible(true);
        students.pack();
        students.setLocationRelativeTo(null);
        this.dispose();
    }

    //Select term submit button. Handles if you can access the term or not
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int selectedIndex = jComboBox1.getSelectedIndex();


        if (selectedIndex == 0) {
            jTabbedPane2.setSelectedIndex(0);
        } else if (selectedIndex == 1) {
            if (isEmpty(0)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 2",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(1);
            }
        } else if (selectedIndex == 2) {
            if (isEmpty(1)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 3",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(2);
            }
        } else if (selectedIndex == 3) {
            if (isEmpty(2)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 4",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(3);
            }
        } else if (selectedIndex == 4) {
            if (isEmpty(3)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 5",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(4);
            }
        } else if (selectedIndex == 5) {
            if (isEmpty(4)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 6",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(5);
            }
        } else if (selectedIndex == 6) {
            if (isEmpty(5)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 7",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(6);
            }
        } else if (selectedIndex == 7) {
            if (isEmpty(6)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 8",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(7);
            }
        } else if (selectedIndex == 8) {
            if (isEmpty(7)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 9",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(8);
            }
        } else if (selectedIndex == 9) {
            if (isEmpty(8)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 10",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(9);
            }
        } else if (selectedIndex == 10) {
            if (isEmpty(9)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 11",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(10);
            }
        } else if (selectedIndex == 11) {
            if (isEmpty(10)) {
                JOptionPane.showMessageDialog(null, "Can't Access Term 12",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                jTabbedPane2.setSelectedIndex(11);
            }
        }
    }

    //Term 1
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 1 submit button

        // Create an array of combo boxes for Term 1
        JComboBox<String>[] comboBoxesTerm1 = new JComboBox[] {
            jComboBox2, jComboBox3, jComboBox4, jComboBox5, jComboBox6,
jComboBox7
        };
        storeSelectedGrades(0, comboBoxesTerm1);
    }

    //Term 2
    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 2 submit button

        System.out.println("Term 2 submit button clicked.");

        // Create an array of combo boxes for Term 2
        JComboBox<String>[] comboBoxesTerm2 = new JComboBox[] {
            jComboBox14, jComboBox15, jComboBox16, jComboBox17,
jComboBox18, jComboBox19
        };
        String jComboBox6Grade = (String) jComboBox6.getSelectedItem();
        if (jComboBox6Grade != null &&
            (jComboBox6Grade.equals("R") || jComboBox6Grade.equals("Inc")
|| jComboBox6Grade.equals("Drp") || jComboBox6Grade.equals("0"))) {

            jComboBox18.setSelectedItem("0");

            courseGradeMap.put("CCINCOML", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCINCOML is unmet. Pass CTHASOPL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox7Grade = (String) jComboBox7.getSelectedItem();
        if (jComboBox7Grade != null &&
            (jComboBox7Grade.equals("R") || jComboBox7Grade.equals("Inc")
|| jComboBox7Grade.equals("Drp") || jComboBox7Grade.equals("0"))) {

            jComboBox19.setSelectedItem("0");

            courseGradeMap.put("CCPRGG2L", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCPRGG2L is unmet. Pass CCPRGG1L first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        storeSelectedGrades(1, comboBoxesTerm2); // Term 2 corresponds to
index 1
    }

    //Term 3
    private void jButton15ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 3 submit button

        // Create an array of combo boxes for Term 3
        JComboBox<String>[] comboBoxesTerm3 = new JComboBox[] {
            jComboBox74, jComboBox75, jComboBox76, jComboBox77,
jComboBox78, jComboBox79
        };

        String jComboBox19Grade = (String) jComboBox19.getSelectedItem();
        if (jComboBox19Grade != null &&
            (jComboBox19Grade.equals("R") || jComboBox19Grade.equals("Inc")
|| jComboBox19Grade.equals("Drp") || jComboBox19Grade.equals("0"))) {

            jComboBox79.setSelectedItem("0");

            courseGradeMap.put("CCOBJPGL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCOBJPGL is unmet. Pass CCPRGG2L first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        storeSelectedGrades(2, comboBoxesTerm3);
    }

    //Term 4
    private void jButton16ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 4 submit button

        // Create an array of combo boxes for Term 4
        JComboBox<String>[] comboBoxesTerm4 = new JComboBox[] {
            jComboBox80, jComboBox81, jComboBox82, jComboBox83,
jComboBox84, jComboBox85
        };

        String jComboBox78Grade = (String) jComboBox78.getSelectedItem();
        if (jComboBox78Grade != null &&
            (jComboBox78Grade.equals("R") || jComboBox78Grade.equals("Inc")
|| jComboBox78Grade.equals("Drp") || jComboBox78Grade.equals("0"))) {

            jComboBox81.setSelectedItem("0");
            courseGradeMap.put("CCMATAN1", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCMATAN1 is unmet. Pass CCDISTR1 first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        if (jComboBox78Grade != null &&
            (jComboBox78Grade.equals("R") || jComboBox78Grade.equals("Inc")
|| jComboBox78Grade.equals("Drp") || jComboBox78Grade.equals("0"))) {

            jComboBox84.setSelectedItem("0");

            courseGradeMap.put("CCDISTR2", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCDISTR2 is unmet. Pass CCDISTR1 first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox19Grade = (String) jComboBox19.getSelectedItem();
        if (jComboBox19Grade != null &&
            (jComboBox19Grade.equals("R") || jComboBox19Grade.equals("Inc")
|| jComboBox19Grade.equals("Drp") || jComboBox19Grade.equals("0"))) {

            jComboBox85.setSelectedItem("0");

            courseGradeMap.put("CCDATRCL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCDATRCL is unmet. Pass CCPRGG2L first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        storeSelectedGrades(3, comboBoxesTerm4);
```

```
    }

    //Term 5
    private void jButton17ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 5 submit button

        // Create an array of combo boxes for Term 5
        JComboBox<String>[] comboBoxesTerm5 = new JComboBox[] {
            jComboBox86, jComboBox87, jComboBox88, jComboBox89,
jComboBox90, jComboBox91
        };

        String jComboBox81Grade = (String) jComboBox81.getSelectedItem();
        if (jComboBox81Grade != null &&
            (jComboBox81Grade.equals("R") || jComboBox81Grade.equals("Inc")
|| jComboBox81Grade.equals("Drp") || jComboBox81Grade.equals("0"))) {

            jComboBox87.setSelectedItem("0");

            courseGradeMap.put("CCMATAN2", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCMATAN2 is unmet. Pass CCMATAN1 first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox85Grade = (String) jComboBox85.getSelectedItem();
        if (jComboBox85Grade != null &&
            (jComboBox85Grade.equals("R") || jComboBox85Grade.equals("Inc")
|| jComboBox85Grade.equals("Drp") || jComboBox85Grade.equals("0"))) {

            jComboBox89.setSelectedItem("0");

            courseGradeMap.put("CTINFMGL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CTINFMGL is unmet. Pass CCDATRCL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        if (jComboBox81Grade != null &&
            (jComboBox81Grade.equals("R") || jComboBox81Grade.equals("Inc")
|| jComboBox81Grade.equals("Drp") || jComboBox81Grade.equals("0"))) {

            jComboBox90.setSelectedItem("0");

            courseGradeMap.put("CCPHYS1L", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCPHYS1L is unmet. Pass CCMATAN1 first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox79Grade = (String) jComboBox79.getSelectedItem();
        if (jComboBox79Grade != null &&
            (jComboBox79Grade.equals("R") || jComboBox79Grade.equals("Inc")
|| jComboBox79Grade.equals("Drp") || jComboBox79Grade.equals("0"))) {

            jComboBox91.setSelectedItem("0");

            courseGradeMap.put("CCOMPORG", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCOMPORG is unmet. Pass CCOBJPGL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        storeSelectedGrades(4, comboBoxesTerm5); // Term 5 corresponds to
index 4

    }

    //Term 6
```

```java
    private void jButton18ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 6 submit button

        // Create an array of combo boxes for Term 6
        JComboBox<String>[] comboBoxesTerm6 = new JComboBox[] {
            jComboBox92, jComboBox93, jComboBox94, jComboBox95, jComboBox96
        };

        String jComboBox78Grade = (String) jComboBox78.getSelectedItem();
        if (jComboBox78Grade != null &&
            (jComboBox78Grade.equals("R") || jComboBox78Grade.equals("Inc")
|| jComboBox78Grade.equals("Drp") || jComboBox78Grade.equals("0"))) {

            jComboBox92.setSelectedItem("0");

            courseGradeMap.put("CCQUAMET", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCQUAMET is unmet. Pass CCDISTR1 first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox89Grade = (String) jComboBox89.getSelectedItem();
        if (jComboBox89Grade != null &&
            (jComboBox89Grade.equals("R") || jComboBox89Grade.equals("Inc")
|| jComboBox89Grade.equals("Drp") || jComboBox89Grade.equals("0"))) {

            jComboBox93.setSelectedItem("0");

            courseGradeMap.put("CTADVDBL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CTADVDBL is unmet. Pass CTINFMGL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        String jComboBox85Grade = (String) jComboBox85.getSelectedItem();
        if (jComboBox85Grade != null &&
            (jComboBox85Grade.equals("R") || jComboBox85Grade.equals("Inc")
|| jComboBox85Grade.equals("Drp") || jComboBox85Grade.equals("0"))) {

            jComboBox94.setSelectedItem("0");

            courseGradeMap.put("CCALCOMP", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCALCOMP is unmet. Pass CCDATRCL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox91Grade = (String) jComboBox91.getSelectedItem();
        if (jComboBox91Grade != null &&
            (jComboBox91Grade.equals("R") || jComboBox91Grade.equals("Inc")
|| jComboBox91Grade.equals("Drp") || jComboBox91Grade.equals("0"))) {

            jComboBox95.setSelectedItem("0");

            courseGradeMap.put("CTBASNTL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CTBASNTL is unmet. Pass CCOMPORG first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox90Grade = (String) jComboBox90.getSelectedItem();
        if (jComboBox90Grade != null &&
            (jComboBox90Grade.equals("R") || jComboBox90Grade.equals("Inc")
|| jComboBox90Grade.equals("Drp") || jComboBox90Grade.equals("0"))) {

            jComboBox96.setSelectedItem("0");

            courseGradeMap.put("CCPHYS2L", "0");
            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCPHYS2L is unmet. Pass CCPHYS1L first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        selectedGrades[5][5] = (String) "  ";

        storeSelectedGrades(5, comboBoxesTerm6);
    }

    //Term 7
    private void jButton19ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 7 submit button

        // Create an array of combo boxes for Term 7
        JComboBox<String>[] comboBoxesTerm7 = new JComboBox[] {
            jComboBox98, jComboBox99, jComboBox100, jComboBox101,
jComboBox102
        };

        String jComboBox85Grade = (String) jComboBox85.getSelectedItem();
        if (jComboBox85Grade != null &&
            (jComboBox85Grade.equals("R") || jComboBox85Grade.equals("Inc")
|| jComboBox85Grade.equals("Drp") || jComboBox85Grade.equals("0"))) {

            jComboBox98.setSelectedItem("0");

            courseGradeMap.put("CCSFEN1L", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCSFEN1L is unmet. Pass CCDATRCL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox94Grade = (String) jComboBox94.getSelectedItem();
        if (jComboBox94Grade != null &&
            (jComboBox94Grade.equals("R") || jComboBox94Grade.equals("Inc")
|| jComboBox94Grade.equals("Drp") || jComboBox94Grade.equals("0"))) {

            jComboBox99.setSelectedItem("0");

            courseGradeMap.put("CCAUTOMA", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCAUTOMA is unmet. Pass CCALCOMP first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox79Grade = (String) jComboBox79.getSelectedItem();
        if (jComboBox79Grade != null &&
            (jComboBox79Grade.equals("R") || jComboBox79Grade.equals("Inc")
|| jComboBox79Grade.equals("Drp") || jComboBox79Grade.equals("0"))) {

            jComboBox100.setSelectedItem("0");

            courseGradeMap.put("CCOPSYSL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCOPSYSL is unmet. Pass CCOBJPGL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        storeSelectedGrades(6, comboBoxesTerm7);
    }

    //Term 8
    private void jButton20ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 8 submit button

        // Create an array of combo boxes for Term 8
        JComboBox<String>[] comboBoxesTerm8 = new JComboBox[] {
            jComboBox104, jComboBox105, jComboBox106, jComboBox107
        };

        String jComboBox101Grade = (String) jComboBox101.getSelectedItem();
        if (jComboBox101Grade != null &&
            (jComboBox101Grade.equals("R") ||
jComboBox101Grade.equals("Inc") || jComboBox101Grade.equals("Drp") ||
jComboBox101Grade.equals("0"))) {

            jComboBox104.setSelectedItem("0");

            courseGradeMap.put("CCADMACL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCADMACL is unmet. Pass CCMACLRL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox98Grade = (String) jComboBox98.getSelectedItem();
        if (jComboBox98Grade != null &&
            (jComboBox98Grade.equals("R") || jComboBox98Grade.equals("Inc")
|| jComboBox98Grade.equals("Drp") || jComboBox98Grade.equals("0"))) {

            jComboBox105.setSelectedItem("0");

            courseGradeMap.put("CCSFEN2L", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCSFEN2L is unmet. Pass CCSFEN1L first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox89Grade = (String) jComboBox89.getSelectedItem();
        if (jComboBox89Grade != null &&
            (jComboBox89Grade.equals("R") || jComboBox89Grade.equals("Inc")
|| jComboBox89Grade.equals("Drp") || jComboBox89Grade.equals("0"))) {

            jComboBox106.setSelectedItem("0");
            courseGradeMap.put("CTINASSL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CTINASSL is unmet. Pass CTINFMGL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        storeSelectedGrades(7, comboBoxesTerm8);
    }

    //Term 9
    private void jButton21ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 9 submit button

        // Create an array of combo boxes for Term 9
        JComboBox<String>[] comboBoxesTerm9 = new JComboBox[] {
            jComboBox110, jComboBox111, jComboBox112, jComboBox113
        };

        String jComboBox19Grade = (String) jComboBox19.getSelectedItem();
        if (jComboBox19Grade != null &&
            (jComboBox19Grade.equals("R") || jComboBox19Grade.equals("Inc")
|| jComboBox19Grade.equals("Drp") || jComboBox19Grade.equals("0"))) {

            jComboBox110.setSelectedItem("0");

            courseGradeMap.put("CCINTHCI", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCINTHCI is unmet. Pass CCPRGG2L first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox79Grade = (String) jComboBox79.getSelectedItem();
        if (jComboBox79Grade != null &&
            (jComboBox79Grade.equals("R") || jComboBox79Grade.equals("Inc")
|| jComboBox79Grade.equals("Drp") || jComboBox79Grade.equals("0"))) {

            jComboBox111.setSelectedItem("0");

            courseGradeMap.put("CTAPDEVL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CTAPDEVL is unmet. Pass CCOBJPGL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox104Grade = (String) jComboBox104.getSelectedItem();
        if (jComboBox104Grade != null &&
            (jComboBox104Grade.equals("R") ||
jComboBox104Grade.equals("Inc") || jComboBox104Grade.equals("Drp") ||
jComboBox104Grade.equals("0"))) {

            jComboBox112.setSelectedItem("0");

            courseGradeMap.put("CCDEPLRL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCDEPLRL is unmet. Pass CCADMACL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox98Grade = (String) jComboBox98.getSelectedItem();
        if (jComboBox98Grade != null &&
            (jComboBox98Grade.equals("R") || jComboBox98Grade.equals("Inc")
|| jComboBox98Grade.equals("Drp") || jComboBox98Grade.equals("0"))) {

            jComboBox113.setSelectedItem("0");

            courseGradeMap.put("CCMETHOD", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCMETHOD is unmet. Pass CCSFEN1L first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        storeSelectedGrades(8, comboBoxesTerm9);
    }

    //Term 10
    private void jButton22ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 10 submit button

        // Create an array of combo boxes for Term 10
        JComboBox<String>[] comboBoxesTerm10 = new JComboBox[] {
            jComboBox116, jComboBox117, jComboBox118, jComboBox119
        };

        String jComboBox113Grade = (String) jComboBox113.getSelectedItem();
        if (jComboBox113Grade != null &&
            (jComboBox113Grade.equals("R") ||
jComboBox113Grade.equals("Inc") || jComboBox113Grade.equals("Drp") ||
jComboBox113Grade.equals("0"))) {

            jComboBox116.setSelectedItem("0");

            courseGradeMap.put("CCTHESS1", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCTHESS1 is unmet. Pass CCMETHOD first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox85Grade = (String) jComboBox85.getSelectedItem();
        if (jComboBox85Grade != null &&
            (jComboBox85Grade.equals("R") || jComboBox85Grade.equals("Inc")
|| jComboBox85Grade.equals("Drp") || jComboBox85Grade.equals("0"))) {

            jComboBox118.setSelectedItem("0");
            courseGradeMap.put("CCPGLANG", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCPGLANG is unmet. Pass CCDATRCL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox112Grade = (String) jComboBox112.getSelectedItem();
        if (jComboBox112Grade != null &&
            (jComboBox112Grade.equals("R") ||
jComboBox112Grade.equals("Inc") || jComboBox112Grade.equals("Drp") ||
jComboBox112Grade.equals("0"))) {

            jComboBox119.setSelectedItem("0");

            courseGradeMap.put("CCRNFLRL", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCRNFLRL is unmet. Pass CCDEPLRL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        storeSelectedGrades(9, comboBoxesTerm10);
    }

    //Term 11
    private void jButton23ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 11 submit button

        // Create an array of combo boxes for Term 11
        JComboBox<String>[] comboBoxesTerm11 = new JComboBox[] {
            jComboBox122, jComboBox123 // Adjust based on your actual combo
boxes for Term 11
        };

        String jComboBox116Grade = (String) jComboBox116.getSelectedItem();
        if (jComboBox116Grade != null &&
            (jComboBox116Grade.equals("R") ||
jComboBox116Grade.equals("Inc") || jComboBox116Grade.equals("Drp") ||
jComboBox116Grade.equals("0"))) {

            jComboBox122.setSelectedItem("0");

            courseGradeMap.put("CCTHESS2", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCTHESS2 is unmet. Pass CCTHESS1 first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }

        String jComboBox119Grade = (String) jComboBox119.getSelectedItem();
        if (jComboBox119Grade != null &&
            (jComboBox119Grade.equals("R") ||
jComboBox119Grade.equals("Inc") || jComboBox119Grade.equals("Drp") ||
jComboBox119Grade.equals("0"))) {

            jComboBox123.setSelectedItem("0");

            courseGradeMap.put("CCTHESS2", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCDATSCL is unmet. Pass CCRNFLRL first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        storeSelectedGrades(10, comboBoxesTerm11);
    }

    //Term 12
    private void jButton24ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Term 12 submit button

        // Create an array of combo boxes for Term 12
        JComboBox<String>[] comboBoxesTerm12 = new JComboBox[] {
            jComboBox128 // Adjust based on your actual combo box for Term
12
        };

        String jComboBox116Grade = (String) jComboBox116.getSelectedItem();
        if (jComboBox116Grade != null &&
            (jComboBox116Grade.equals("R") ||
jComboBox116Grade.equals("Inc") || jComboBox116Grade.equals("Drp") ||
jComboBox116Grade.equals("0"))) {

            jComboBox128.setSelectedItem("0");

            courseGradeMap.put("CCINTERN", "0");

            JOptionPane.showMessageDialog(
                null, // Center on the screen
                "Prerequisite for CCINTERN is unmet. Pass CCTHESS1 first.",
                "Prerequisite Not Met",
                JOptionPane.ERROR_MESSAGE
            );
        }
        storeSelectedGrades(11, comboBoxesTerm12);
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //Print button
        initializeArray();
        addGradeToHashMap();
        finalFlowchart();

        boolean allGradesPassed = true;

        for (String grade : courseGradeMap.values()) {
            if (grade.equals("R") || grade.equals("Inc") ||
grade.equals("Drp") || grade.equals("0")) {
                allGradesPassed = false;
                break;
            }
        }

        if (allGradesPassed) {
            JOptionPane.showMessageDialog(null, "Congratulations! You have
passed all of your courses. Time for graduation!.", "Graduation Status",
JOptionPane.INFORMATION_MESSAGE);
        }
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new myFlowChart().setVisible(true);
            }
        });
    }
}
```

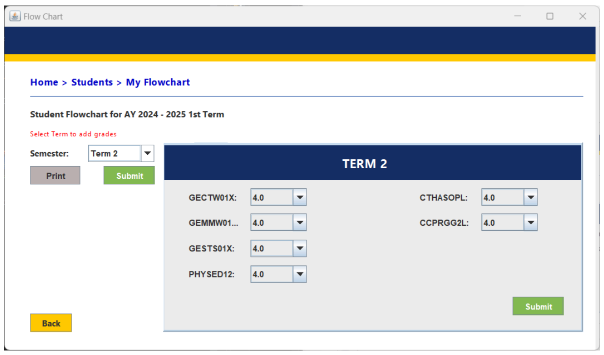


Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 1

GEPCM01X: 4.0
GEUTS01X: 4.0
GERPH01X: 4.0
PHYSED11: 4.0
CCINCOML: 4.0
CCPRGG1L: 4.0

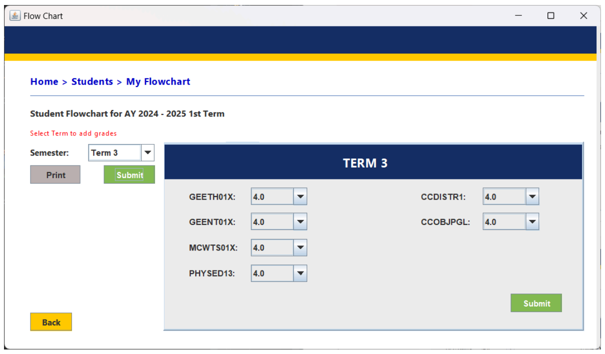


Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 2

GECTW01X: 4.0
GEMMW01... 4.0
GESTS01X: 4.0
PHYSED12: 4.0
CTHASOPL: 4.0
CCPRGG2L: 4.0

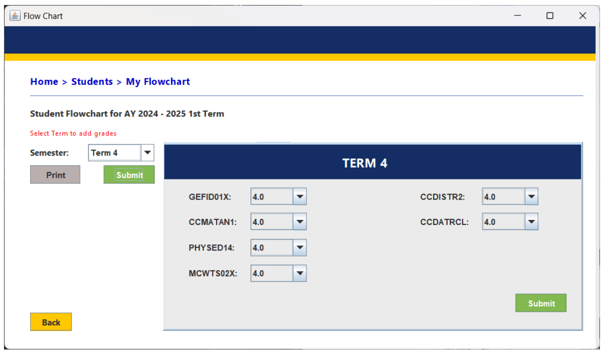


Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 3

GEETH01X: 4.0
GEENT01X: 4.0
MCWTS01X: 4.0
PHYSED13: 4.0
CCDISTR1: 4.0
CCOBJPGL: 4.0

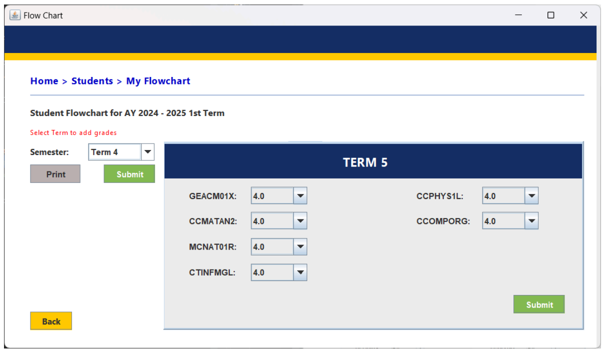


Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

GEFID01X: 4.0
CCMATAN1: 4.0
PHYSED14: 4.0
MCWTS02X: 4.0
CCDISTR2: 4.0
CCDATRCL: 4.0



Home > Students > My Flowchart

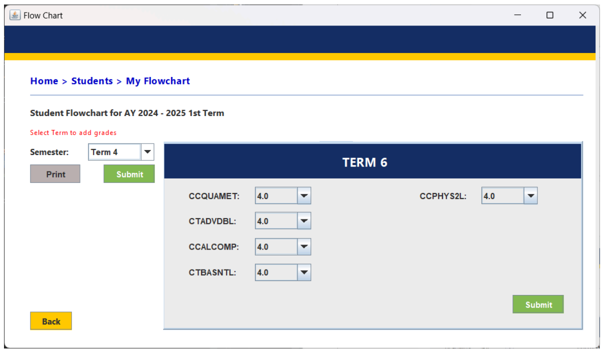
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

TERM 5

GEACM01X: 4.0
CCMATAN2: 4.0
MCNAT01R: 4.0
CTINFMGL: 4.0
CCPHYS1L: 4.0
CCOMPORG: 4.0



Home > Students > My Flowchart

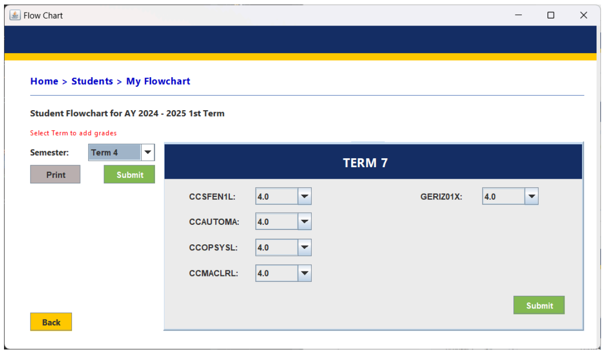
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

TERM 6

CCQUAMET: 4.0
CTADVDBL: 4.0
CCALCOMP: 4.0
CTBASNTL: 4.0
CCPHYS2L: 4.0



Home > Students > My Flowchart

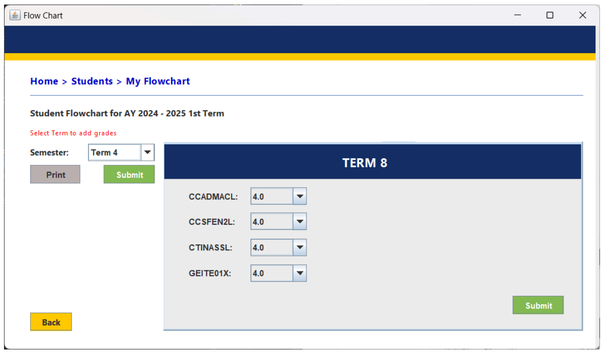
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

TERM 7

CCSFEN1L: 4.0
CCAUTOMA: 4.0
CCOPSYSL: 4.0
CCMACLRL: 4.0
GERIZ01X: 4.0



Home > Students > My Flowchart

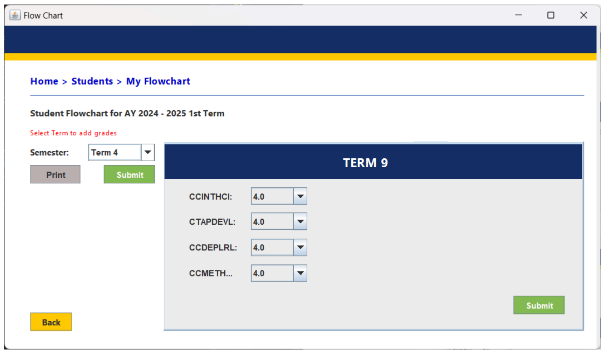
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

TERM 8

CCADMACL: 4.0
CCSFEN2L: 4.0
CTINASSL: 4.0
GEITE01X: 4.0



Home > Students > My Flowchart

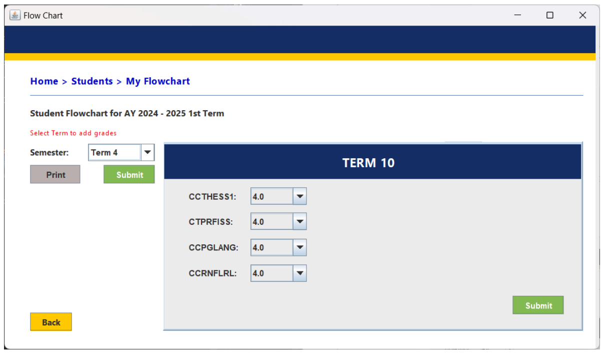
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

TERM 9

CCINTHCI: 4.0
CTAPDEVL: 4.0
CCDEPLRL: 4.0
CCMETH... 4.0



Home > Students > My Flowchart

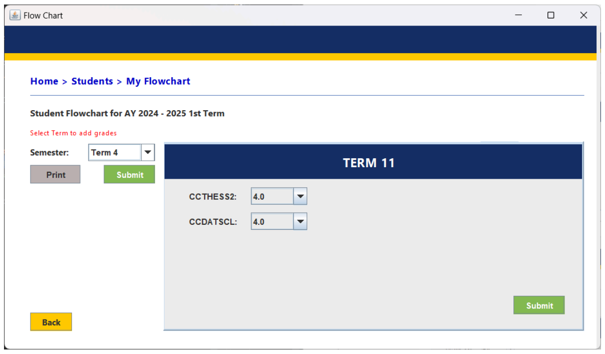
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

TERM 10

CCTHESS1: 4.0
CTPRFISS: 4.0
CCPGLANG: 4.0
CCRNFLRL: 4.0



Home > Students > My Flowchart

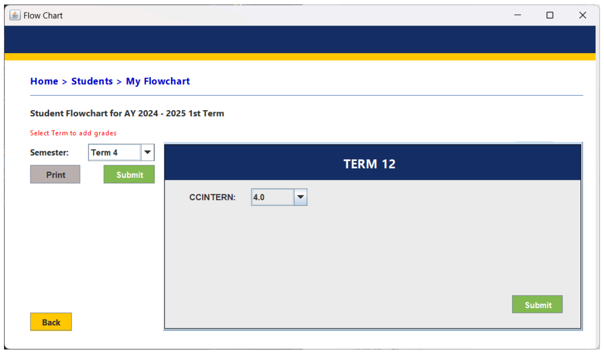
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

Semester: Term 4

TERM 11

CCTHESS2: 4.0
CCDATSCL: 4.0



Home > Students > My Flowchart

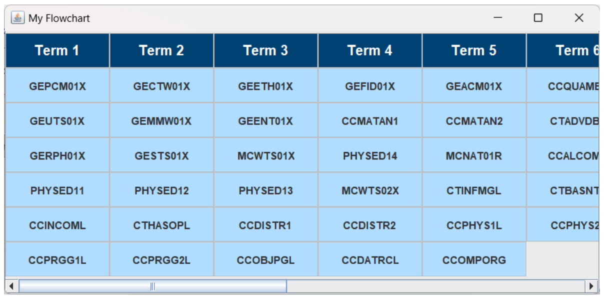
Student Flowchart for AY 2024 - 2025 1st Term

Select Term to add grades

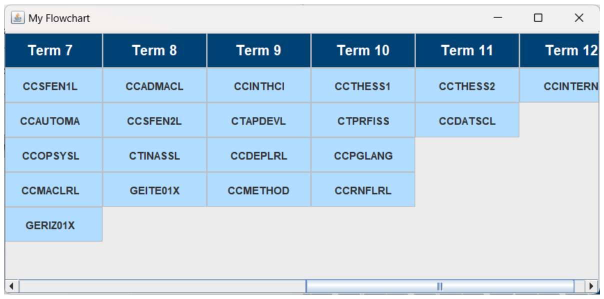
Semester: Term 4

TERM 12

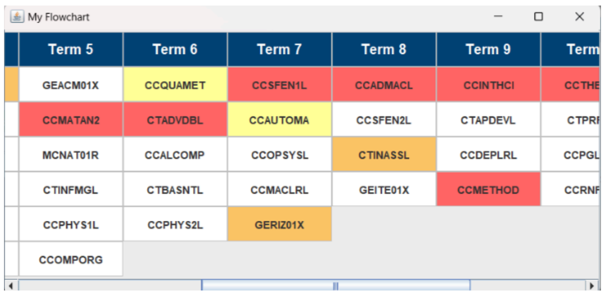
CCINTERN: 4.0



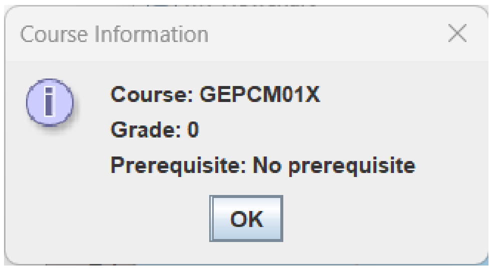
Term 1	Term 2	Term 3	Term 4	Term 5	Term 6
GEPCM01X	GECTW01X	GEETH01X	GEFID01X	GEACM01X	CCQUAME
GEUTS01X	GEMMW01X	GEENT01X	CCMATAN1	CCMATAN2	CTADVDB
GERPH01X	GESTS01X	MCWTS01X	PHYSED14	MCNAT01R	CCALCOM
PHYSED11	PHYSED12	PHYSED13	MCWTS02X	CTINFMGL	CTBASNT
CCINCOML	CTHASOPL	CCDISTR1	CCDISTR2	CCPHYS1L	CCPHYS2
CCPRGG1L	CCPRGG2L	CCOBJPGL	CCDATRCL	CCOMPORG	



Term 7	Term 8	Term 9	Term 10	Term 11	Term 12
CCSFEN1L	CCADMACL	CCINTHCI	CCTHESS1	CCTHESS2	CCINTERN
CCAUTOMA	CCSFEN2L	CTAPDEVL	CTPRFISS	CCDATSCL	
CCOPSYSL	CTINASSL	CCDEPLRL	CCPGLANG		
CCMACLRL	GEITE01X	CCMETHOD	CCRNFLRL		
GERIZ01X					



Term 5	Term 6	Term 7	Term 8	Term 9	Term
GEACM01X	CCQUAMET	CCSFEN1L	CCADMACL	CCINTHCI	CCTHE
CCMATAN2	CTADVDBL	CCAUTOMA	CCSFEN2L	CTAPDEVL	CTPRF
MCNAT01R	CCALCOMP	CCOPSYSL	CTINASSL	CCDEPLRL	CCPGL
CTINFMGL	CTBASNTL	CCMACLRL	GEITE01X	CCMETHOD	CCRNF
CCPHYS1L	CCPHYS2L	GERIZ01X			
CCOMPORG					



Course: GEPCM01X

Grade: 0

Prerequisite: No prerequisite