Shri Gajanan Shikshan Sanstha's

SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING

SHEGAON – 444203, DIST. BULDANA (MAHARASHTRA STATE), INDIA

"Recognized by A.I.C.T.E., New Delhi" Affiliated to Sant Gadge Baba Amravati University, Amravati

"Approved by the D.T.E., M.S. Mumbai", Institution Accredited by N.A.A.C. (UGC) Bangalore

DEPARTMENT OF INFORMATION TECHNOLOGY

TIMETABLE GENERATION APP

OOP TEC PROJECT

PRESENTED BY:	HRIDAY AMLE (2N-40)
	ADITYA TALE (2N-29)
YEAR:	SECOND (SEMESTER-3)
BRANCH:	INFORMATION TECHNOLOGY
GUIDED BY:	MS. P.V. KALE
DATE:	11 DECEMBER 2024

TABLE OF CONTENTS

Introduction

1.1 Overview of the Topic

The **Timetable Application** is a Java-based desktop application designed to simplify and automate the process of generating academic timetables. It is particularly useful for educational institutions where students and faculty need to manage schedules for theory and lab subjects.

Using a graphical user interface (GUI) built with **Java Swing**, this application allows users to input parameters such as their academic program, semester, and available days to generate a customized timetable. It also features the ability to view the syllabus for the selected program and semester in a PDF format. The project leverages **CSV files** to store subject data and uses **scheduling algorithms** to allocate time slots to theory and lab subjects based on availability.

1.2 Project Objectives

The primary objectives of this project are:

- To build a **user-friendly application** that helps students and faculty generate timetables based on program, semester, and active days.
- To **automate** the scheduling of theory and lab subjects, ensuring no overlap and accounting for necessary breaks.
- To provide a **feature to view program syllabi directly** within the application by opening the corresponding PDF.
- To offer a **seamless user experience** by reducing manual efforts for timetable generation and ensuring it is accurate and error-free.

By focusing on these goals, the project aims to create a system that can be easily adapted for any academic program and can scale to accommodate larger sets of subjects or programs in the future.

1.3 Languages & Tools Used

- **Java**: The primary programming language used to build the application, including the GUI and backend logic for timetable generation.
- **Java Swing**: A lightweight GUI toolkit used to create interactive user interfaces in Java applications. It provides a wide range of components like buttons, combo boxes, and labels.

- **CSV Files**: Used for storing the subject data for various academic programs. The program reads these files at runtime to load subjects into memory for generating timetables.

- **PDF Files**: These are used to display syllabi for different programs and semesters. The application opens these files in a PDF viewer on the user's system.

1.4 Scope of the Project

The **Timetable Management System** is a desktop-based application designed to assist students and educational institutions in managing and organizing their academic timetables. The system allows the user to select a program, semester, active days, and generate a personalized timetable for the selected semester. The timetable will include theory and lab sessions, with break times scheduled appropriately. The system is built using **Java** and **Swing** for the graphical user interface (GUI), and it loads subject data from **CSV files**.

Key Features:

- **Program and Semester Selection**: Students can select their program (e.g., CSE, IT, MECH) and the corresponding semester (e.g., Semester 3, Semester 5), with the system dynamically adjusting available semesters based on the selected program and semester type (Autumn/Spring).

- **Active Day Selection**: Users can select the days they are available for classes (e.g., Monday, Wednesday, Friday). The system then schedules the subjects accordingly, allowing for customized timetables.

- **Timetable Generation**: The system generates a clear and organized timetable, listing theory subjects with time slots and lab sessions after the theory classes, with appropriate breaks in between.

- **Syllabus Access**: Students can view the syllabus for their selected program and semester by opening a PDF file, ensuring easy access to course details and requirements.

Technology: The application is built using **Java Swing** for the graphical user interface (GUI), providing a simple and intuitive user experience. Subject and lab data are loaded from **CSV files**, making it easy to update and manage the content.

App Design & Architecture

2.1 App Components

The **Timetable Generation App** consists of the following main components:

1. **User Interface (UI)**: The application uses **Java Swing** to create a graphical user interface (GUI) that allows users to interact with the system. The interface includes dropdown menus, buttons, and checkboxes for selecting the program, semester, and active days.
2. **CSV File Reader**: The system reads subject and lab information from **CSV files** (theorySubjects.csv and labSubjects.csv). These files store the list of subjects for different programs and semesters. This approach allows easy updates and management of the subject data.
3. **Timetable Generator**: This part of the system is responsible for generating the actual timetable. Based on the user's selections (program, semester, active days), the system schedules theory subjects and lab sessions and formats them into a timetable.
4. **Syllabus Viewer**: This component allows users to view the syllabus for their selected program and semester. The syllabus is stored as a PDF file, and the user can open it using their default PDF viewer.
5. **Data Storage**: The subjects and lab data are stored in **HashMaps**, which allows fast access and easy management of subject data for different programs and semesters.

2.2 Diagrams

Program Flow:

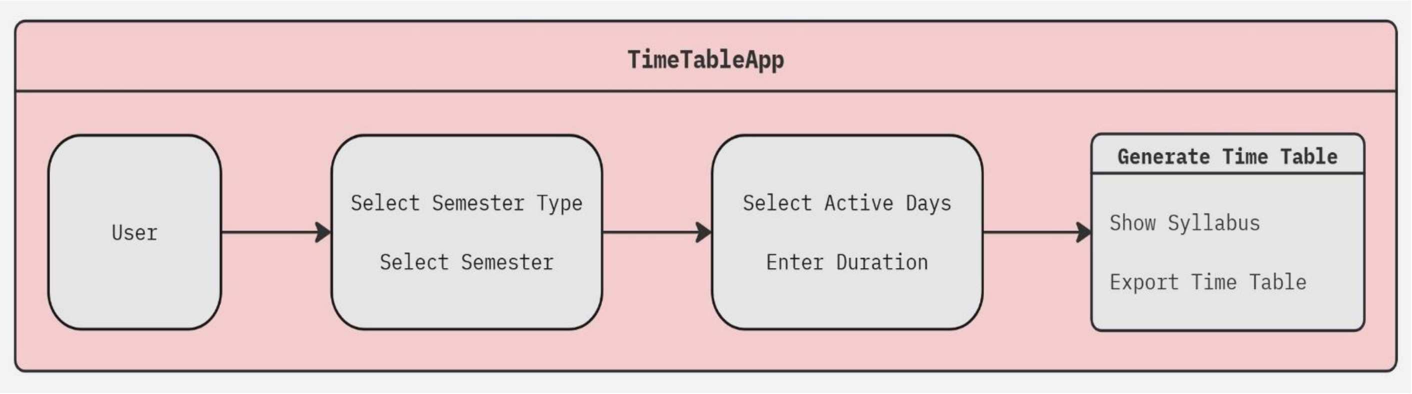


Fig: Program Flow

Case Flowchart:

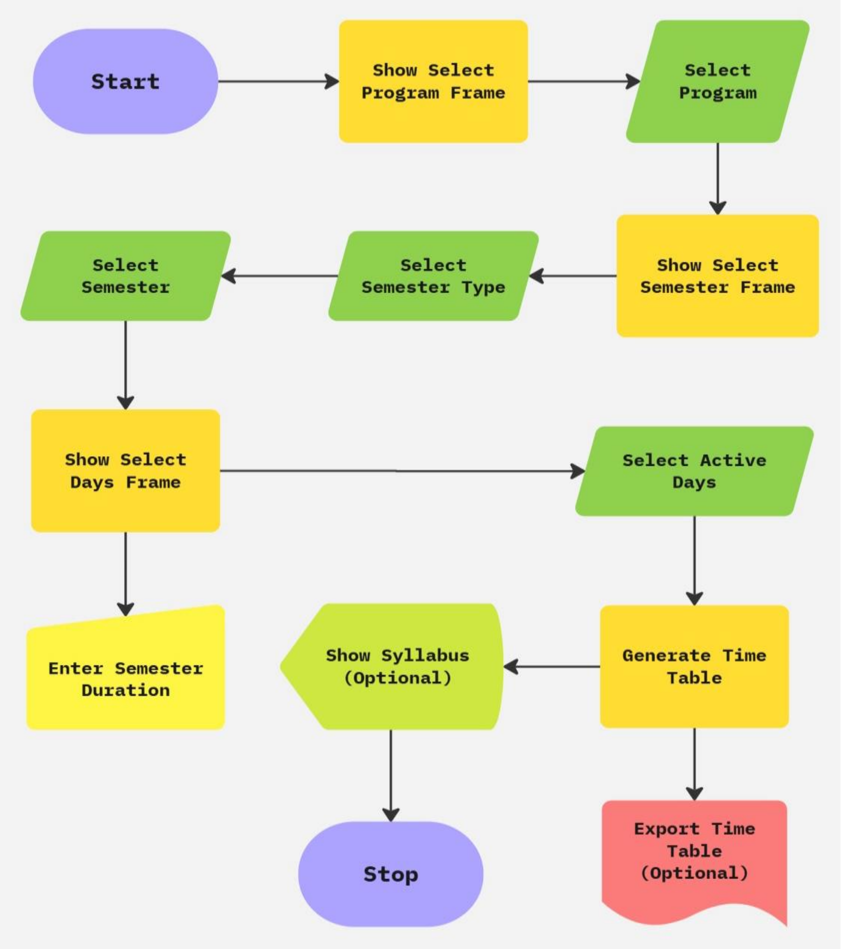


Fig: Case Flowchart

Class Diagram:

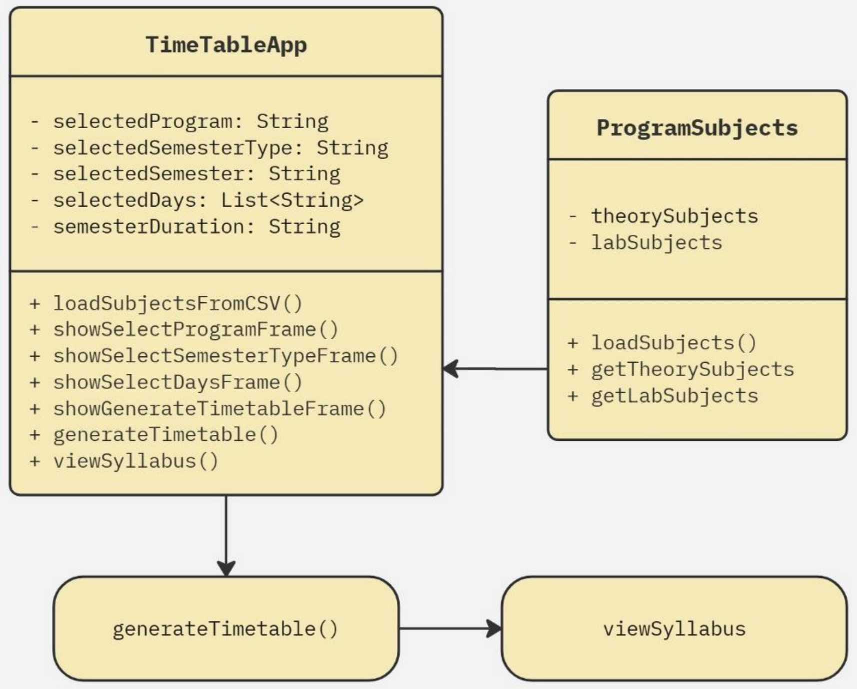
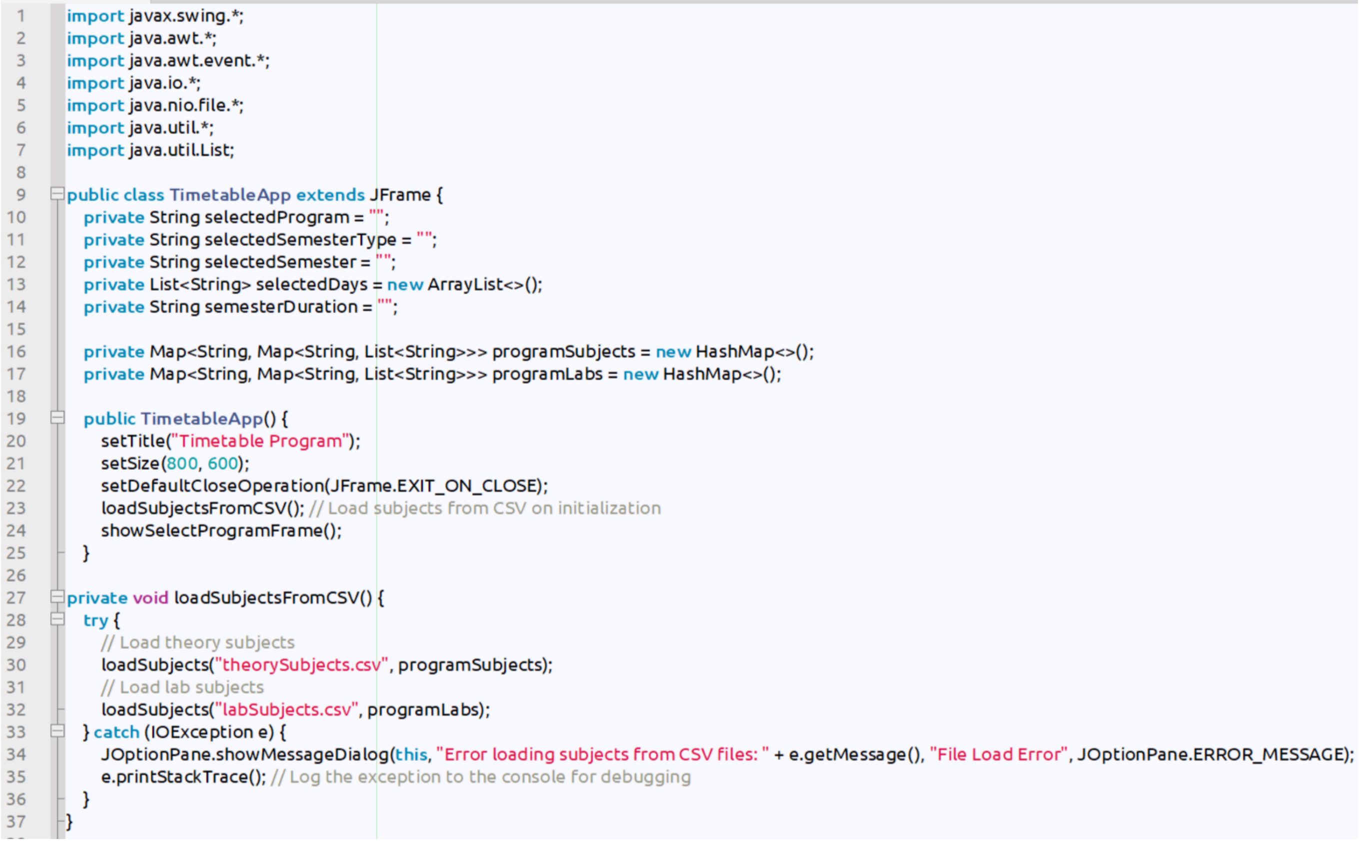

TimeTableApp
- selectedProgram: String
- selectedSemesterType: String
- selectedSemester: String
- selectedDays: List<String>
- semesterDuration: String
+ loadSubjectsFromCSV()
+ showSelectProgramFrame()
+ showSelectSemesterTypeFrame()
+ showSelectDaysFrame()
+ showGenerateTimetableFrame()
+ generateTimetable()
+ viewSyllabus()
ProgramSubjects
- theorySubjects
- labSubjects
+ loadSubjects()
+ getTheorySubjects
+ getLabSubjects
generateTimetable()
viewSyllabus

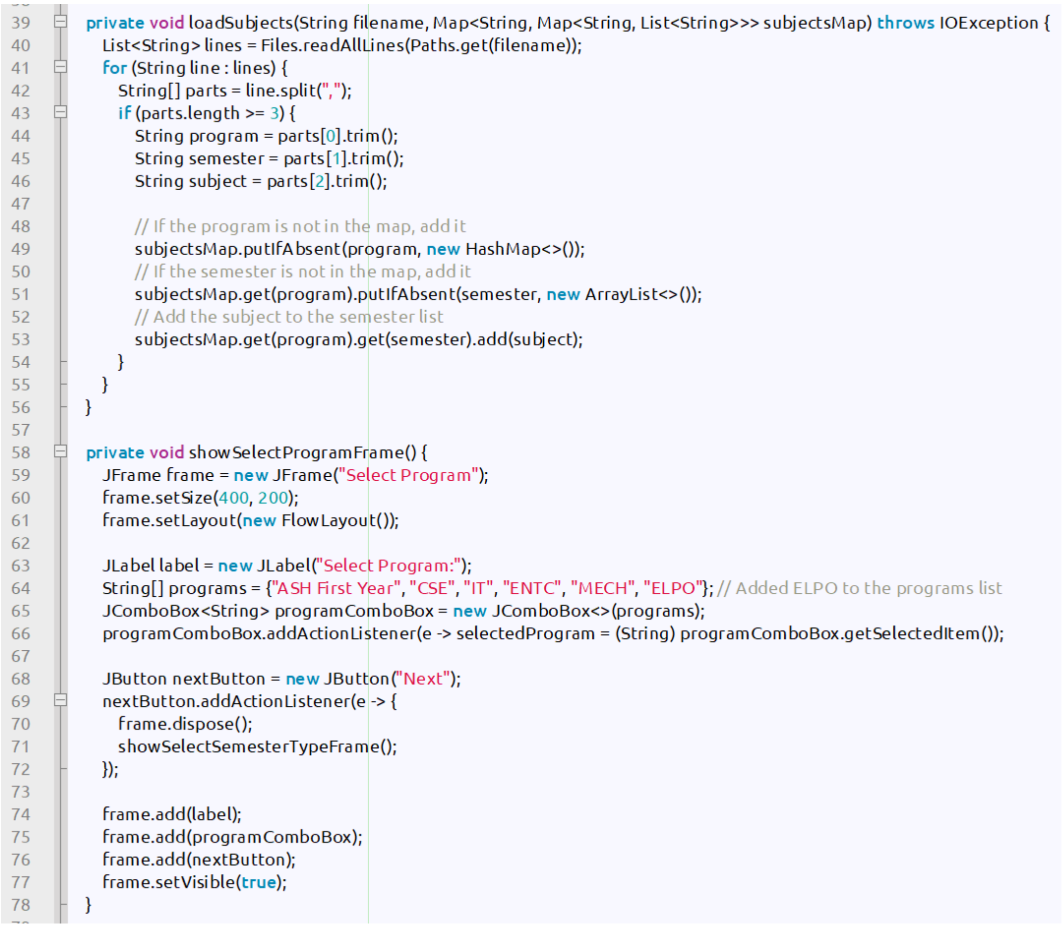
Fig: Class Diagram

Code Implementation

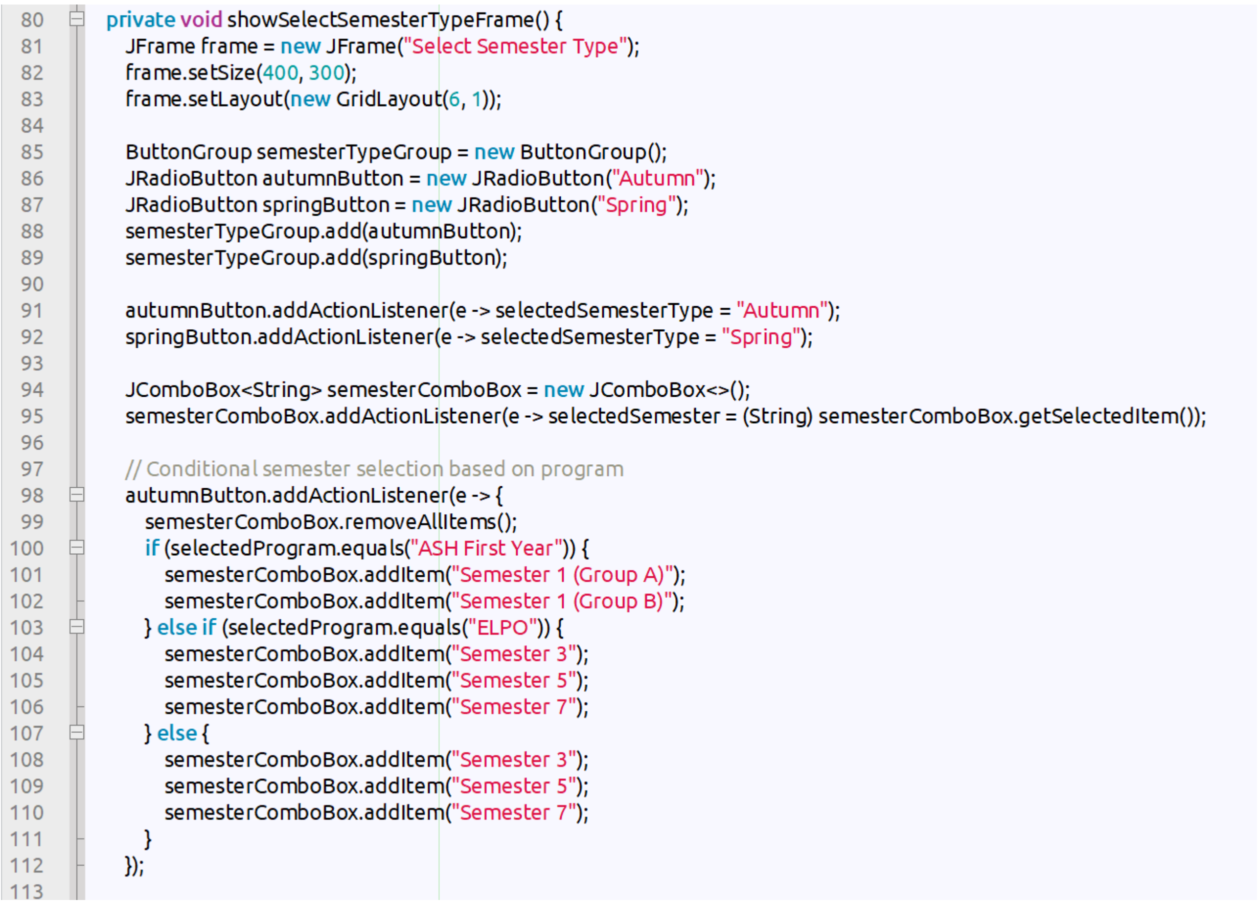
3.1 Program



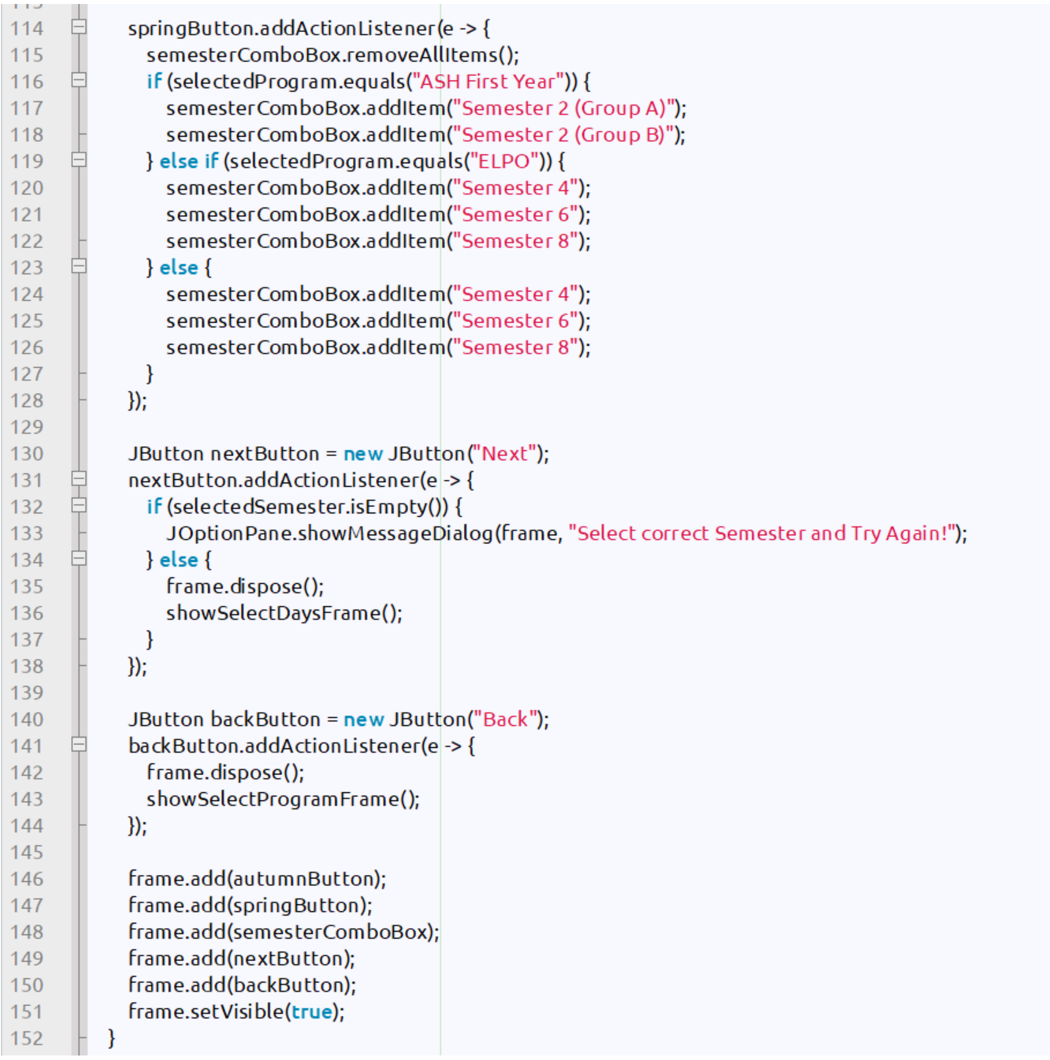
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.nio.file.*;
import java.util.*;
import java.util.List;

public class TimetableApp extends JFrame {
    private String selectedProgram = "";
    private String selectedSemesterType = "";
    private String selectedSemester = "";
    private List<String> selectedDays = new ArrayList<>();
    private String semesterDuration = "";

    private Map<String, Map<String, List<String>>> programSubjects = new HashMap<>();
    private Map<String, Map<String, List<String>>> programLabs = new HashMap<>();

    public TimetableApp() {
        setTitle("Timetable Program");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        loadSubjectsFromCSV(); // Load subjects from CSV on initialization
        showSelectProgramFrame();
    }

private void loadSubjectsFromCSV() {
    try {
        // Load theory subjects
        loadSubjects("theorySubjects.csv", programSubjects);
        // Load lab subjects
        loadSubjects("labSubjects.csv", programLabs);
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this, "Error loading subjects from CSV files: " + e.getMessage(), "File Load Error", JOptionPane.ERROR_MESSAGE);
        e.printStackTrace(); // Log the exception to the console for debugging
    }
}
```



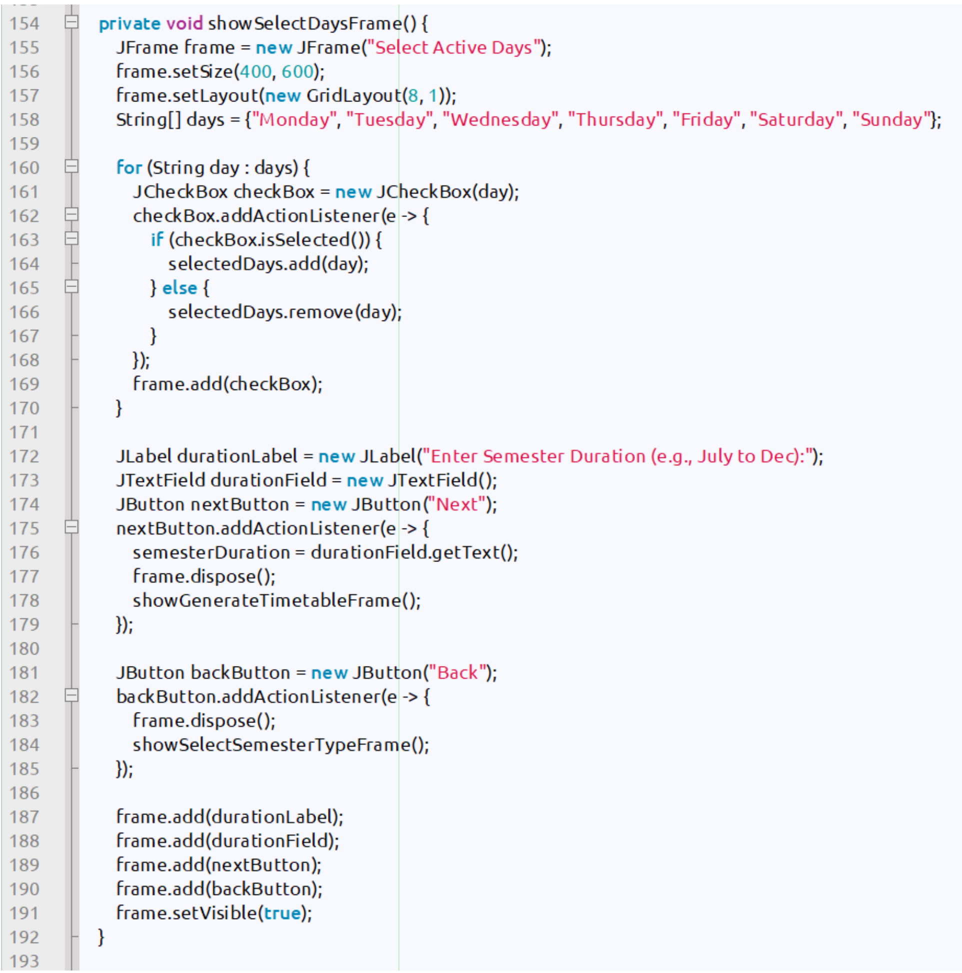
```java
    private void loadSubjects(String filename, Map<String, Map<String, List<String>>> subjectsMap) throws IOException {
        List<String> lines = Files.readAllLines(Paths.get(filename));
        for (String line : lines) {
            String[] parts = line.split(",");
            if (parts.length >= 3) {
                String program = parts[0].trim();
                String semester = parts[1].trim();
                String subject = parts[2].trim();

                // If the program is not in the map, add it
                subjectsMap.putIfAbsent(program, new HashMap<>());
                // If the semester is not in the map, add it
                subjectsMap.get(program).putIfAbsent(semester, new ArrayList<>());
                // Add the subject to the semester list
                subjectsMap.get(program).get(semester).add(subject);
            }
        }
    }

    private void showSelectProgramFrame() {
        JFrame frame = new JFrame("Select Program");
        frame.setSize(400, 200);
        frame.setLayout(new FlowLayout());

        JLabel label = new JLabel("Select Program:");
        String[] programs = {"ASH First Year", "CSE", "IT", "ENTC", "MECH", "ELPO"}; // Added ELPO to the programs list
        JComboBox<String> programComboBox = new JComboBox<>(programs);
        programComboBox.addActionListener(e -> selectedProgram = (String) programComboBox.getSelectedItem());

        JButton nextButton = new JButton("Next");
        nextButton.addActionListener(e -> {
            frame.dispose();
            showSelectSemesterTypeFrame();
        });

        frame.add(label);
        frame.add(programComboBox);
        frame.add(nextButton);
        frame.setVisible(true);
    }
```



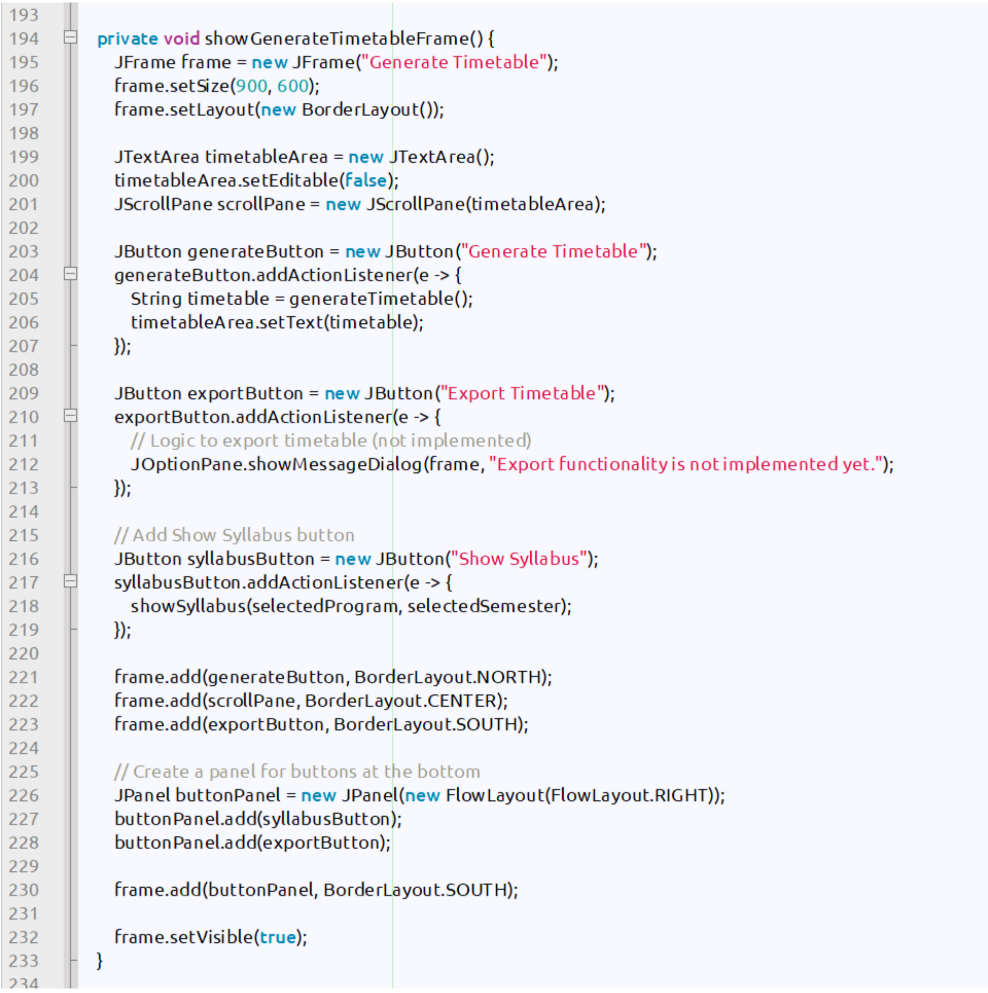
```
private void showSelectSemesterTypeFrame() {
    JFrame frame = new JFrame("Select Semester Type");
    frame.setSize(400, 300);
    frame.setLayout(new GridLayout(6, 1));

    ButtonGroup semesterTypeGroup = new ButtonGroup();
    JRadioButton autumnButton = new JRadioButton("Autumn");
    JRadioButton springButton = new JRadioButton("Spring");
    semesterTypeGroup.add(autumnButton);
    semesterTypeGroup.add(springButton);

    autumnButton.addActionListener(e -> selectedSemesterType = "Autumn");
    springButton.addActionListener(e -> selectedSemesterType = "Spring");

    JComboBox<String> semesterComboBox = new JComboBox<>();
    semesterComboBox.addActionListener(e -> selectedSemester = (String) semesterComboBox.getSelectedItem());

    // Conditional semester selection based on program
    autumnButton.addActionListener(e -> {
        semesterComboBox.removeAllItems();
        if (selectedProgram.equals("ASH First Year")) {
            semesterComboBox.addItem("Semester 1 (Group A)");
            semesterComboBox.addItem("Semester 1 (Group B)");
        } else if (selectedProgram.equals("ELPO")) {
            semesterComboBox.addItem("Semester 3");
            semesterComboBox.addItem("Semester 5");
            semesterComboBox.addItem("Semester 7");
        } else {
            semesterComboBox.addItem("Semester 3");
            semesterComboBox.addItem("Semester 5");
            semesterComboBox.addItem("Semester 7");
        }
    });

```



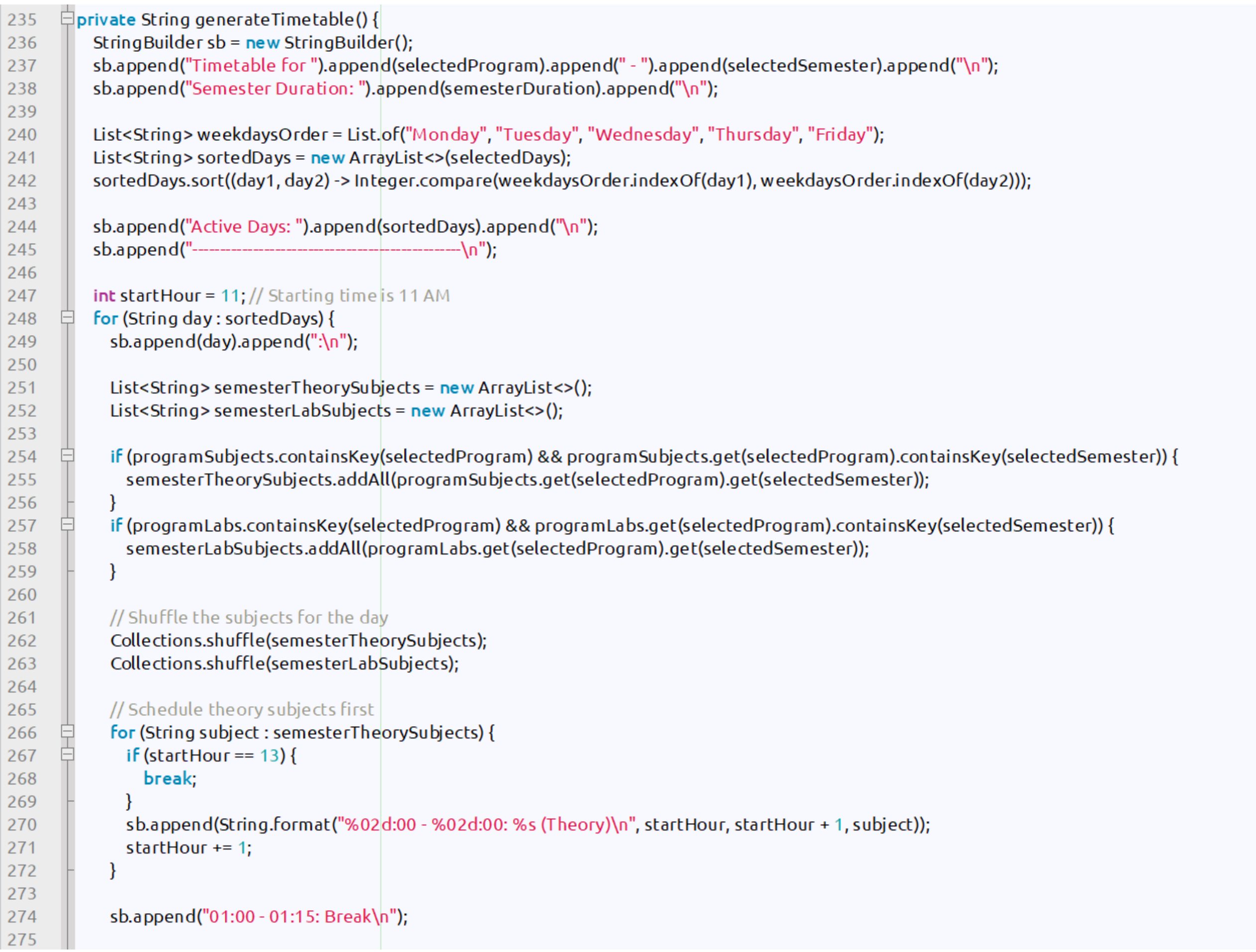
```
    springButton.addActionListener(e -> {
        semesterComboBox.removeAllItems();
        if (selectedProgram.equals("ASH First Year")) {
            semesterComboBox.addItem("Semester 2 (Group A)");
            semesterComboBox.addItem("Semester 2 (Group B)");
        } else if (selectedProgram.equals("ELPO")) {
            semesterComboBox.addItem("Semester 4");
            semesterComboBox.addItem("Semester 6");
            semesterComboBox.addItem("Semester 8");
        } else {
            semesterComboBox.addItem("Semester 4");
            semesterComboBox.addItem("Semester 6");
            semesterComboBox.addItem("Semester 8");
        }
    });

    JButton nextButton = new JButton("Next");
    nextButton.addActionListener(e -> {
        if (selectedSemester.isEmpty()) {
            JOptionPane.showMessageDialog(frame, "Select correct Semester and Try Again!");
        } else {
            frame.dispose();
            showSelectDaysFrame();
        }
    });

    JButton backButton = new JButton("Back");
    backButton.addActionListener(e -> {
        frame.dispose();
        showSelectProgramFrame();
    });

    frame.add(autumnButton);
    frame.add(springButton);
    frame.add(semesterComboBox);
    frame.add(nextButton);
    frame.add(backButton);
    frame.setVisible(true);
}
```



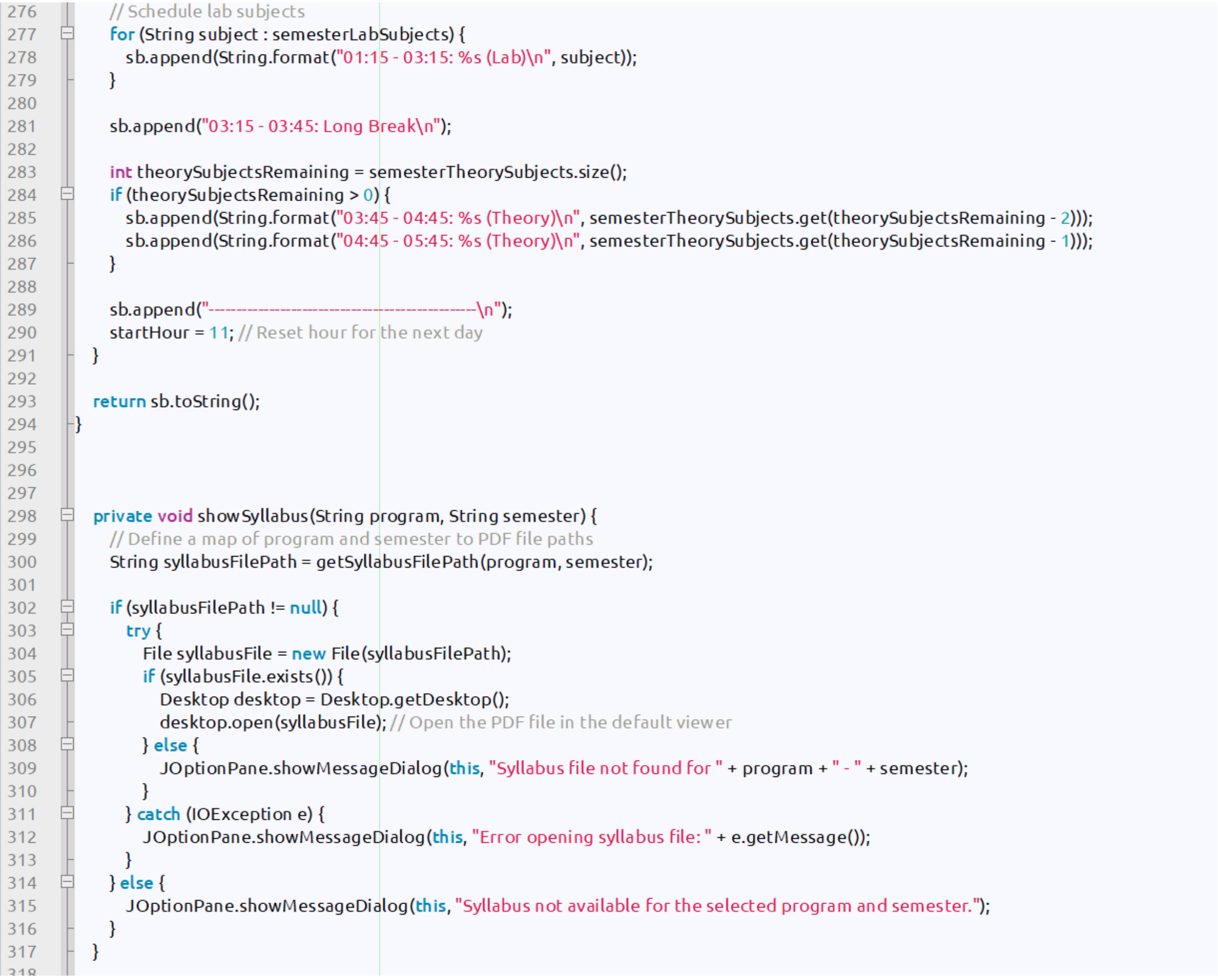
```java
    private void showSelectDaysFrame() {
        JFrame frame = new JFrame("Select Active Days");
        frame.setSize(400, 600);
        frame.setLayout(new GridLayout(8, 1));
        String[] days = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};

        for (String day : days) {
            JCheckBox checkBox = new JCheckBox(day);
            checkBox.addActionListener(e -> {
                if (checkBox.isSelected()) {
                    selectedDays.add(day);
                } else {
                    selectedDays.remove(day);
                }
            });
            frame.add(checkBox);
        }

        JLabel durationLabel = new JLabel("Enter Semester Duration (e.g., July to Dec):");
        JTextField durationField = new JTextField();
        JButton nextButton = new JButton("Next");
        nextButton.addActionListener(e -> {
            semesterDuration = durationField.getText();
            frame.dispose();
            showGenerateTimetableFrame();
        });

        JButton backButton = new JButton("Back");
        backButton.addActionListener(e -> {
            frame.dispose();
            showSelectSemesterTypeFrame();
        });

        frame.add(durationLabel);
        frame.add(durationField);
        frame.add(nextButton);
        frame.add(backButton);
        frame.setVisible(true);
    }

```



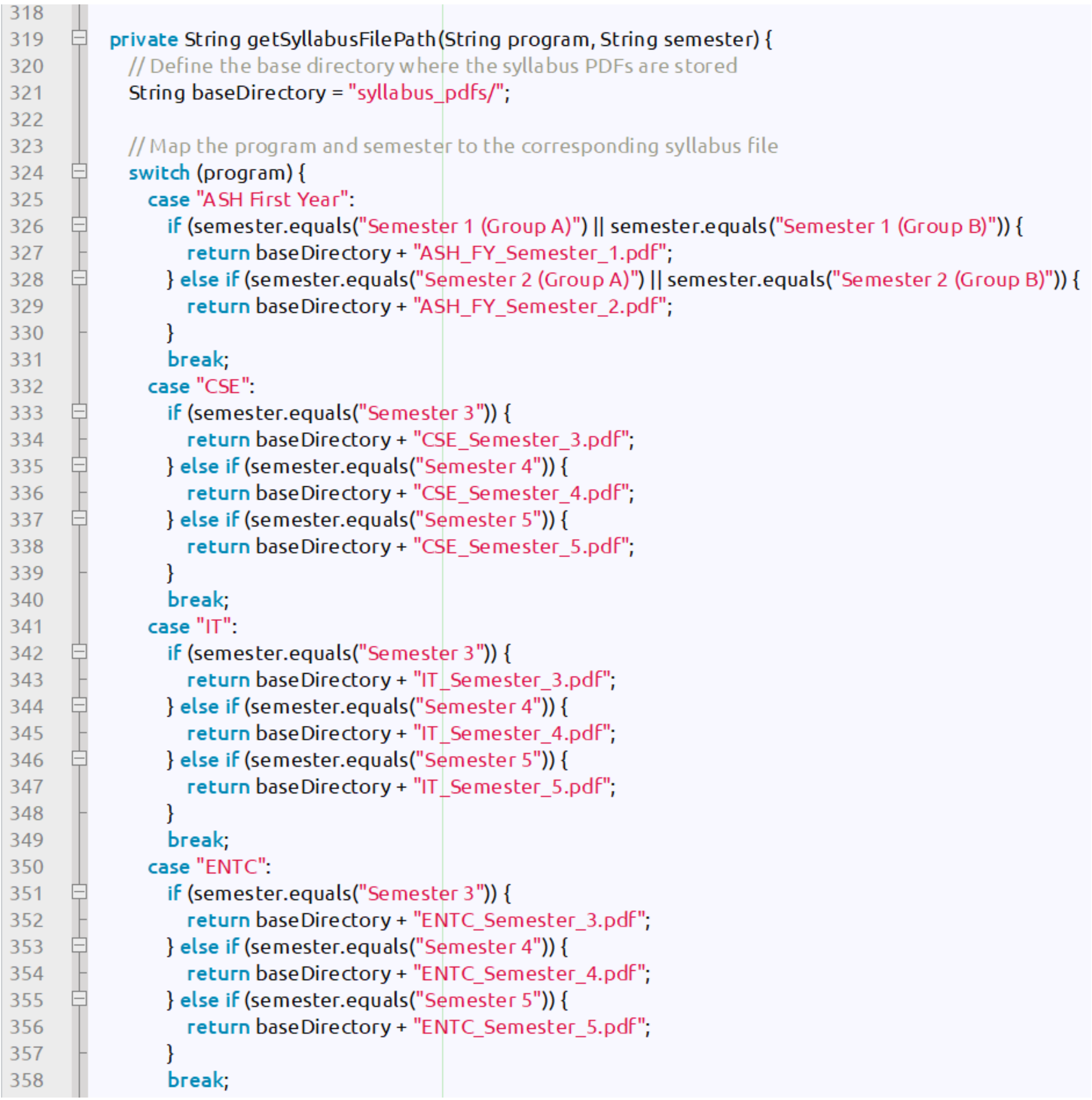
```java

    private void showGenerateTimetableFrame() {
        JFrame frame = new JFrame("Generate Timetable");
        frame.setSize(900, 600);
        frame.setLayout(new BorderLayout());

        JTextArea timetableArea = new JTextArea();
        timetableArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(timetableArea);

        JButton generateButton = new JButton("Generate Timetable");
        generateButton.addActionListener(e -> {
            String timetable = generateTimetable();
            timetableArea.setText(timetable);
        });

        JButton exportButton = new JButton("Export Timetable");
        exportButton.addActionListener(e -> {
            // Logic to export timetable (not implemented)
            JOptionPane.showMessageDialog(frame, "Export functionality is not implemented yet.");
        });

        // Add Show Syllabus button
        JButton syllabusButton = new JButton("Show Syllabus");
        syllabusButton.addActionListener(e -> {
            showSyllabus(selectedProgram, selectedSemester);
        });

        frame.add(generateButton, BorderLayout.NORTH);
        frame.add(scrollPane, BorderLayout.CENTER);
        frame.add(exportButton, BorderLayout.SOUTH);

        // Create a panel for buttons at the bottom
        JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        buttonPanel.add(syllabusButton);
        buttonPanel.add(exportButton);

        frame.add(buttonPanel, BorderLayout.SOUTH);

        frame.setVisible(true);
    }
```



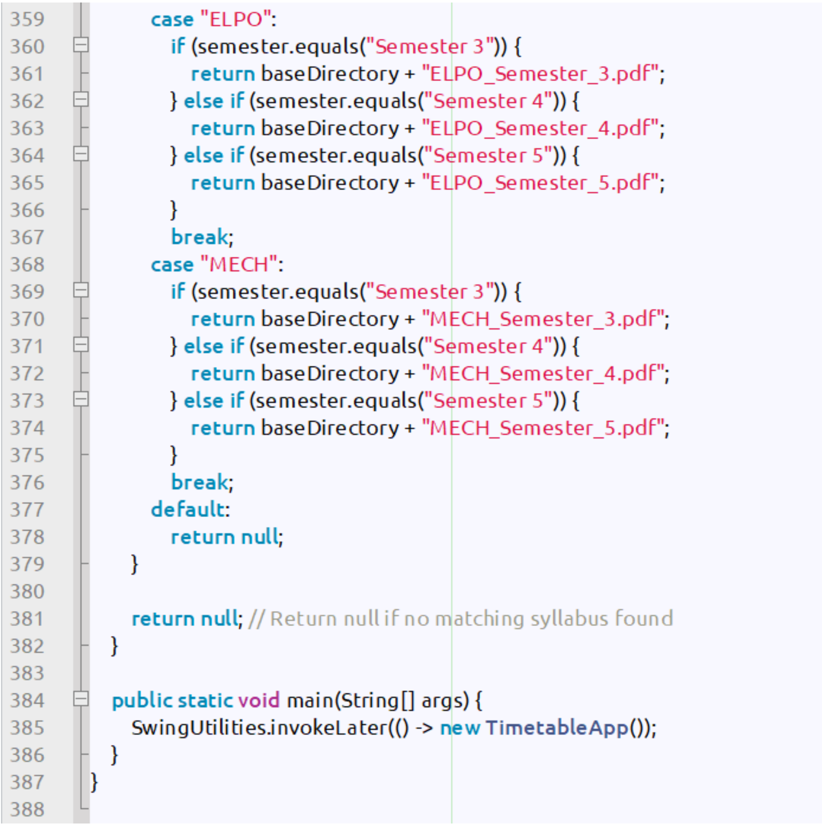
```java
private String generateTimetable() {
    StringBuilder sb = new StringBuilder();
    sb.append("Timetable for ").append(selectedProgram).append(" - ").append(selectedSemester).append("\n");
    sb.append("Semester Duration: ").append(semesterDuration).append("\n");

    List<String> weekdaysOrder = List.of("Monday", "Tuesday", "Wednesday", "Thursday", "Friday");
    List<String> sortedDays = new ArrayList<>(selectedDays);
    sortedDays.sort((day1, day2) -> Integer.compare(weekdaysOrder.indexOf(day1), weekdaysOrder.indexOf(day2)));

    sb.append("Active Days: ").append(sortedDays).append("\n");
    sb.append("----------------------------------------------\n");

    int startHour = 11; // Starting time is 11 AM
    for (String day : sortedDays) {
        sb.append(day).append(":\n");

        List<String> semesterTheorySubjects = new ArrayList<>();
        List<String> semesterLabSubjects = new ArrayList<>();

        if (programSubjects.containsKey(selectedProgram) && programSubjects.get(selectedProgram).containsKey(selectedSemester)) {
            semesterTheorySubjects.addAll(programSubjects.get(selectedProgram).get(selectedSemester));
        }
        if (programLabs.containsKey(selectedProgram) && programLabs.get(selectedProgram).containsKey(selectedSemester)) {
            semesterLabSubjects.addAll(programLabs.get(selectedProgram).get(selectedSemester));
        }

        // Shuffle the subjects for the day
        Collections.shuffle(semesterTheorySubjects);
        Collections.shuffle(semesterLabSubjects);

        // Schedule theory subjects first
        for (String subject : semesterTheorySubjects) {
            if (startHour == 13) {
                break;
            }
            sb.append(String.format("%02d:00 - %02d:00: %s (Theory)\n", startHour, startHour + 1, subject));
            startHour += 1;
        }

        sb.append("01:00 - 01:15: Break\n");

```



```java
        // Schedule lab subjects
        for (String subject : semesterLabSubjects) {
            sb.append(String.format("01:15 - 03:15: %s (Lab)\n", subject));
        }

        sb.append("03:15 - 03:45: Long Break\n");

        int theorySubjectsRemaining = semesterTheorySubjects.size();
        if (theorySubjectsRemaining > 0) {
            sb.append(String.format("03:45 - 04:45: %s (Theory)\n", semesterTheorySubjects.get(theorySubjectsRemaining - 2)));
            sb.append(String.format("04:45 - 05:45: %s (Theory)\n", semesterTheorySubjects.get(theorySubjectsRemaining - 1)));
        }

        sb.append("----------------------------------------------\n");
        startHour = 11; // Reset hour for the next day
    }

    return sb.toString();
}



    private void showSyllabus(String program, String semester) {
        // Define a map of program and semester to PDF file paths
        String syllabusFilePath = getSyllabusFilePath(program, semester);

        if (syllabusFilePath != null) {
            try {
                File syllabusFile = new File(syllabusFilePath);
                if (syllabusFile.exists()) {
                    Desktop desktop = Desktop.getDesktop();
                    desktop.open(syllabusFile); // Open the PDF file in the default viewer
                } else {
                    JOptionPane.showMessageDialog(this, "Syllabus file not found for " + program + " - " + semester);
                }
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Error opening syllabus file: " + e.getMessage());
            }
        } else {
            JOptionPane.showMessageDialog(this, "Syllabus not available for the selected program and semester.");
        }
    }
```



```java

    private String getSyllabusFilePath(String program, String semester) {
        // Define the base directory where the syllabus PDFs are stored
        String baseDirectory = "syllabus_pdfs/";

        // Map the program and semester to the corresponding syllabus file
        switch (program) {
            case "ASH First Year":
                if (semester.equals("Semester 1 (Group A)") || semester.equals("Semester 1 (Group B)")) {
                    return baseDirectory + "ASH_FY_Semester_1.pdf";
                } else if (semester.equals("Semester 2 (Group A)") || semester.equals("Semester 2 (Group B)")) {
                    return baseDirectory + "ASH_FY_Semester_2.pdf";
                }
                break;
            case "CSE":
                if (semester.equals("Semester 3")) {
                    return baseDirectory + "CSE_Semester_3.pdf";
                } else if (semester.equals("Semester 4")) {
                    return baseDirectory + "CSE_Semester_4.pdf";
                } else if (semester.equals("Semester 5")) {
                    return baseDirectory + "CSE_Semester_5.pdf";
                }
                break;
            case "IT":
                if (semester.equals("Semester 3")) {
                    return baseDirectory + "IT_Semester_3.pdf";
                } else if (semester.equals("Semester 4")) {
                    return baseDirectory + "IT_Semester_4.pdf";
                } else if (semester.equals("Semester 5")) {
                    return baseDirectory + "IT_Semester_5.pdf";
                }
                break;
            case "ENTC":
                if (semester.equals("Semester 3")) {
                    return baseDirectory + "ENTC_Semester_3.pdf";
                } else if (semester.equals("Semester 4")) {
                    return baseDirectory + "ENTC_Semester_4.pdf";
                } else if (semester.equals("Semester 5")) {
                    return baseDirectory + "ENTC_Semester_5.pdf";
                }
                break;
```



```java
            case "ELPO":
                if (semester.equals("Semester 3")) {
                    return baseDirectory + "ELPO_Semester_3.pdf";
                } else if (semester.equals("Semester 4")) {
                    return baseDirectory + "ELPO_Semester_4.pdf";
                } else if (semester.equals("Semester 5")) {
                    return baseDirectory + "ELPO_Semester_5.pdf";
                }
                break;
            case "MECH":
                if (semester.equals("Semester 3")) {
                    return baseDirectory + "MECH_Semester_3.pdf";
                } else if (semester.equals("Semester 4")) {
                    return baseDirectory + "MECH_Semester_4.pdf";
                } else if (semester.equals("Semester 5")) {
                    return baseDirectory + "MECH_Semester_5.pdf";
                }
                break;
            default:
                return null;
        }

        return null; // Return null if no matching syllabus found
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new TimetableApp());
    }
}

```

3.2 Code Explanation

1. `TimetableApp` Class

The `TimetableApp` class is the main entry point for the application. It controls the flow of the program and is responsible for initializing the GUI, reading the subject data, generating the timetable, and handling user input.

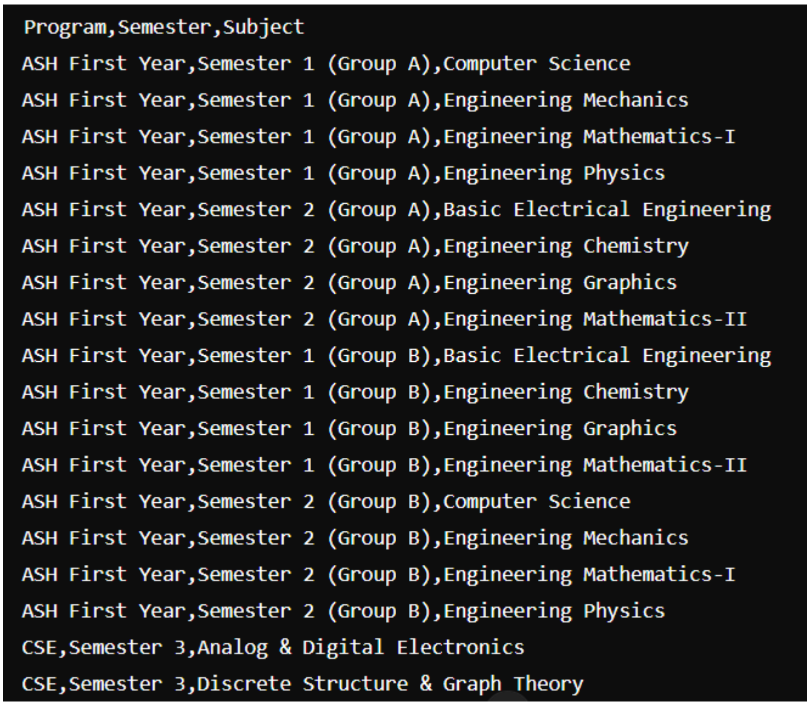
This class is responsible for managing user interactions. It initializes the graphical user interface (GUI), takes user inputs for selecting their program, semester, and active days, and then processes this data to generate a personalized timetable.

- **Key Functionality**:
  - Initializes and sets up the GUI components (like buttons, combo boxes, etc.).
  - Calls the `generateTimetable()` method to create the timetable based on the selected options.
  - Provides the functionality to view the syllabus using the `viewSyllabus()` method.

2. `ProgramSubjects` Class

The `ProgramSubjects` class is responsible for holding the subject data (both theory and lab subjects) for different programs and semesters. It loads the subject data from CSV files and organizes it into lists or maps.

It stores the subjects available for each program and semester and to provide easy access to them when generating the timetable.



```
Program,Semester,Subject
ASH First Year,Semester 1 (Group A),Computer Science
ASH First Year,Semester 1 (Group A),Engineering Mechanics
ASH First Year,Semester 1 (Group A),Engineering Mathematics-I
ASH First Year,Semester 1 (Group A),Engineering Physics
ASH First Year,Semester 2 (Group A),Basic Electrical Engineering
ASH First Year,Semester 2 (Group A),Engineering Chemistry
ASH First Year,Semester 2 (Group A),Engineering Graphics
ASH First Year,Semester 2 (Group A),Engineering Mathematics-II
ASH First Year,Semester 1 (Group B),Basic Electrical Engineering
ASH First Year,Semester 1 (Group B),Engineering Chemistry
ASH First Year,Semester 1 (Group B),Engineering Graphics
ASH First Year,Semester 1 (Group B),Engineering Mathematics-II
ASH First Year,Semester 2 (Group B),Computer Science
ASH First Year,Semester 2 (Group B),Engineering Mechanics
ASH First Year,Semester 2 (Group B),Engineering Mathematics-I
ASH First Year,Semester 2 (Group B),Engineering Physics
CSE,Semester 3,Analog & Digital Electronics
CSE,Semester 3,Discrete Structure & Graph Theory
```

Fig: CSV Example

- **Key Functionality**:
  - Reads the subject data from a CSV file using file handling (e.g., Java NIO).
  - Stores the subject data in `HashMap` or `ArrayList` structures for easy retrieval.
  - Provides methods to fetch subjects based on the program and semester selected by the user.

3. `TimetableGenerator` Class

The `TimetableGenerator` class is the core of the timetable generation logic. It takes the selected program, semester, and active days from the user and creates a timetable by assigning subjects to the appropriate time slots.

It generates the timetable by scheduling subjects and ensuring there are no conflicts. It manages the placement of theory sessions, lab sessions, and breaks in a logical sequence.

- **Key Functionality:**
  - Takes the user input (active days, program, and semester) and uses that data to create a schedule.
  - Ensures no subject is scheduled at the same time by checking time slots.
  - Organizes the timetable in a day-wise manner, placing theory subjects first, followed by lab sessions, and ensuring there are breaks in between.

4. `SyllabusViewer` Class

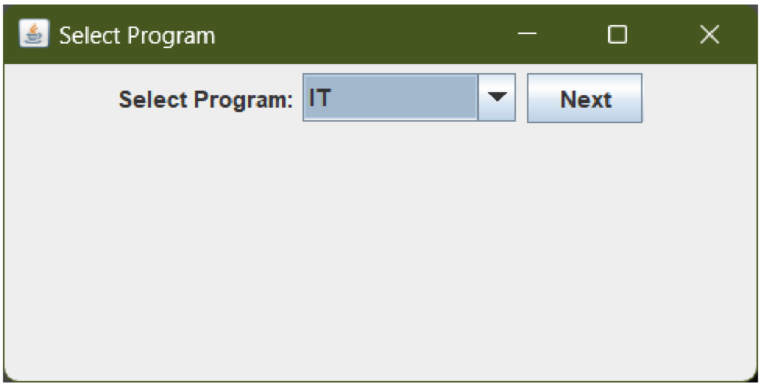
The `SyllabusViewer` class handles the functionality for opening and displaying the syllabus as a PDF. This class uses the Java Desktop API to open PDF files in the default PDF viewer on the user's system.

It allows the user to view their program's syllabus directly from the application.

- **Key Functionality**:
  - Uses the `Desktop` API to open the relevant syllabus PDF file based on the selected program and semester.
  - Handles any errors that might occur if the PDF file cannot be opened (e.g., file not found, wrong format).

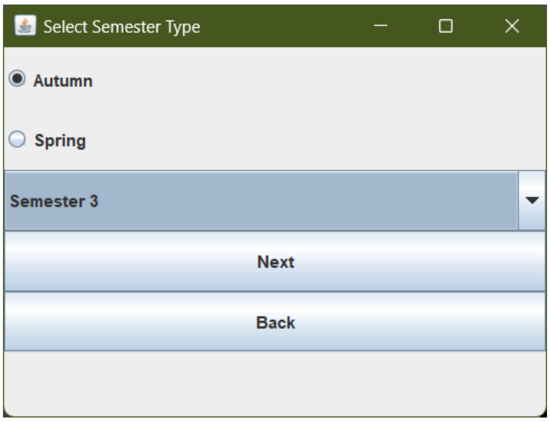
Output & Analysis

4.1 Output



Select Program: IT

Fig: Select Program

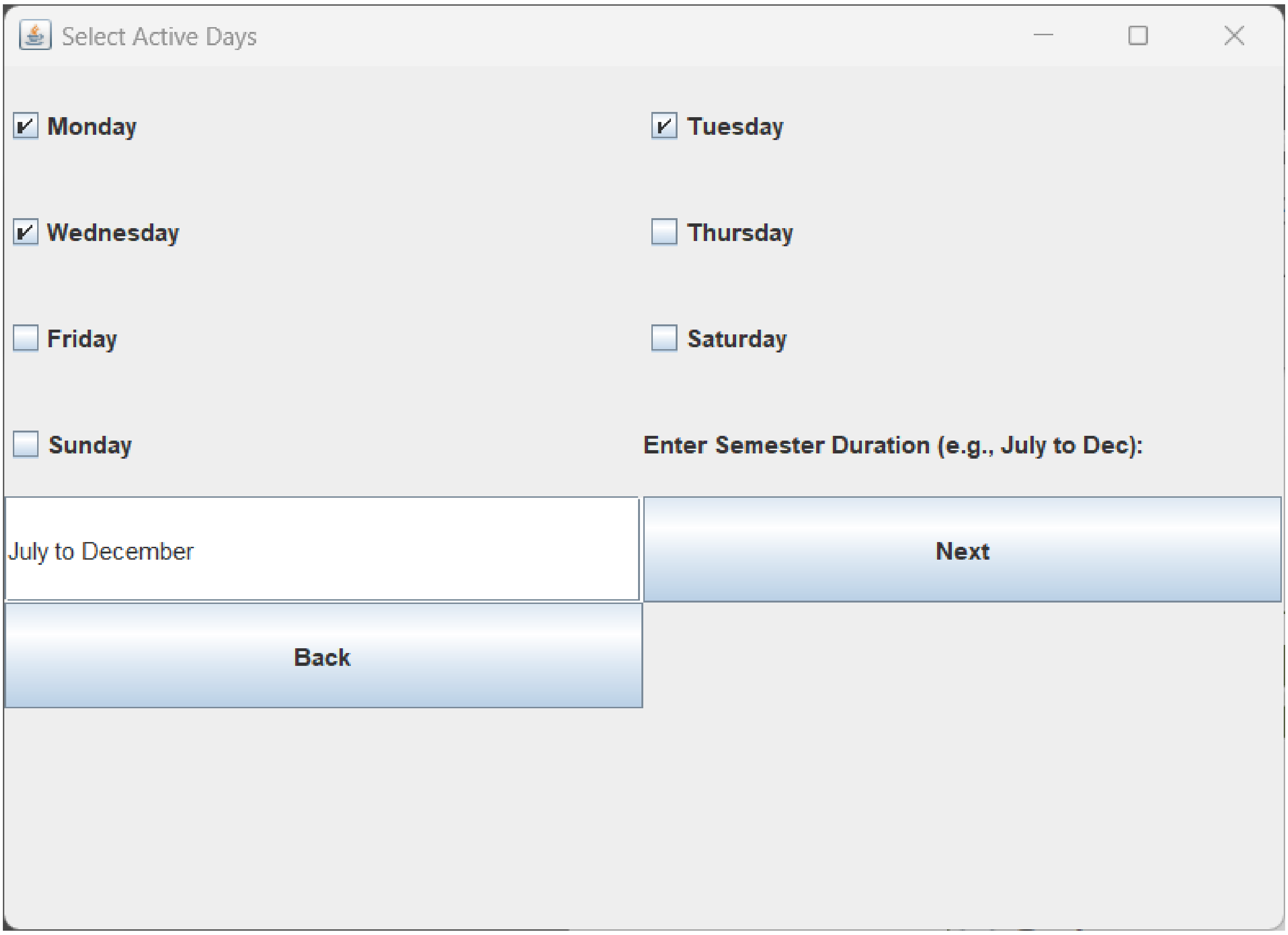


● Autumn

○ Spring

Semester 3

Fig: Select Semester Type



☑ Monday

☑ Tuesday

☑ Wednesday

□ Thursday

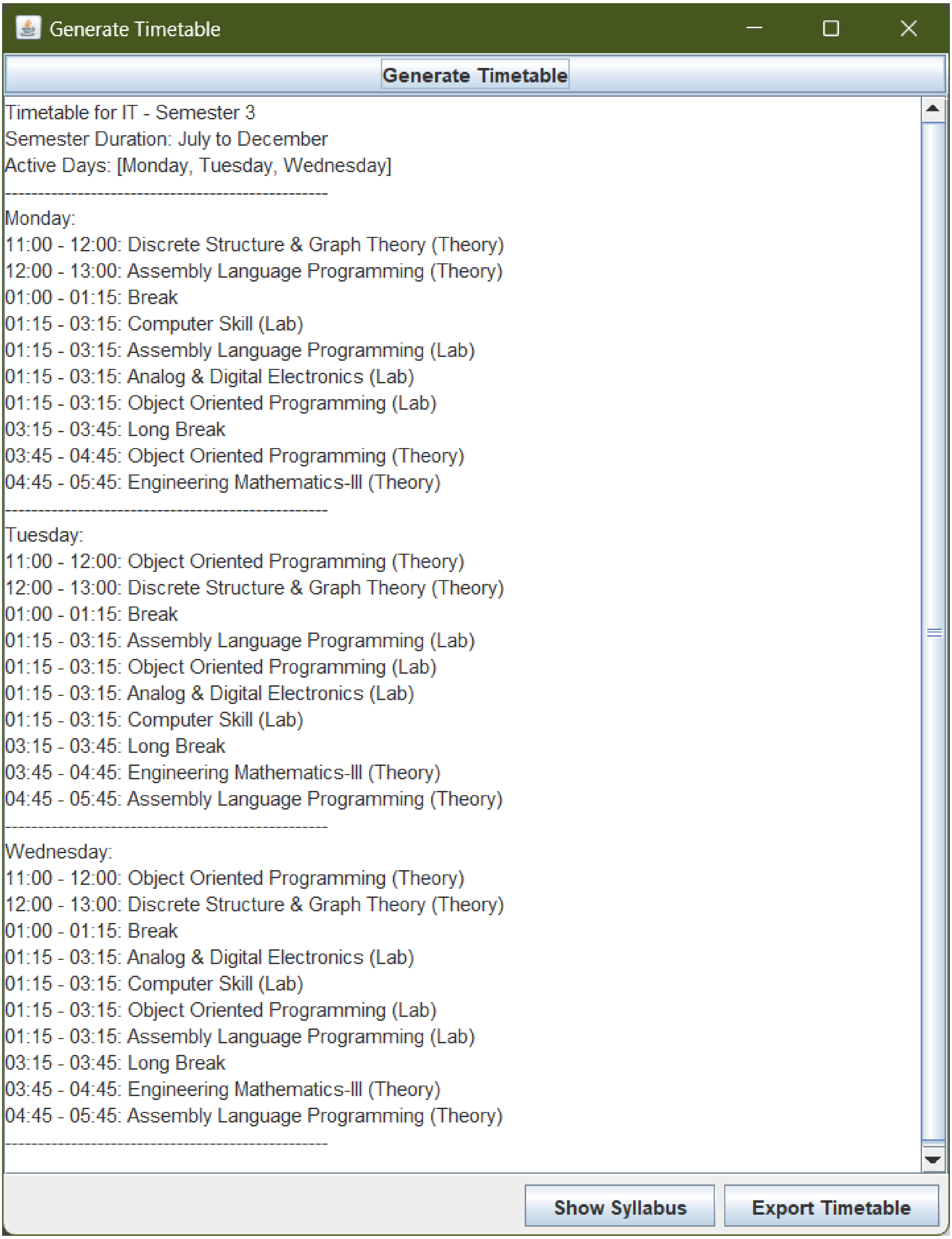
□ Friday

□ Saturday

□ Sunday

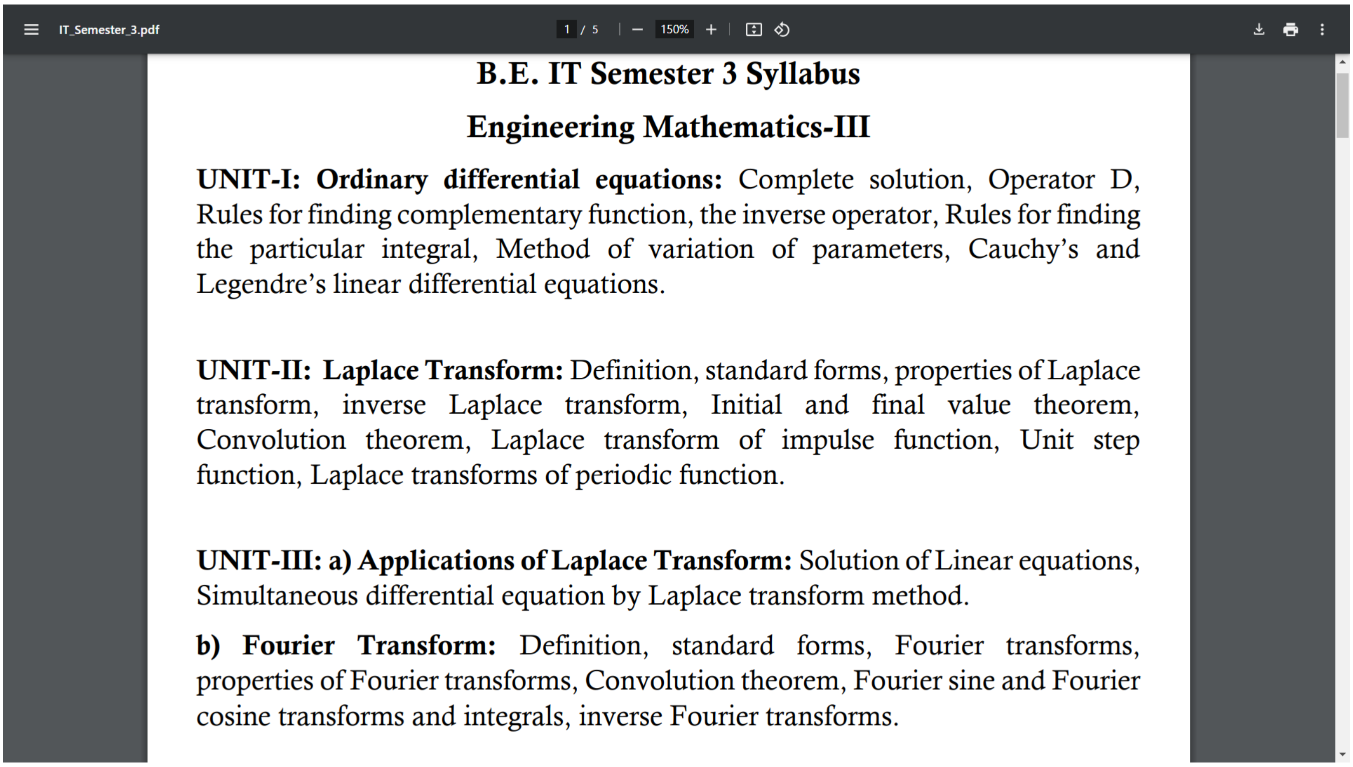
Enter Semester Duration (e.g., July to Dec): July to December

Fig: Select Active Days



```
Timetable for IT - Semester 3
Semester Duration: July to December
Active Days: [Monday, Tuesday, Wednesday]
--------------------------------------------------
Monday:
11:00 - 12:00: Discrete Structure & Graph Theory (Theory)
12:00 - 13:00: Assembly Language Programming (Theory)
01:00 - 01:15: Break
01:15 - 03:15: Computer Skill (Lab)
01:15 - 03:15: Assembly Language Programming (Lab)
01:15 - 03:15: Analog & Digital Electronics (Lab)
01:15 - 03:15: Object Oriented Programming (Lab)
03:15 - 03:45: Long Break
03:45 - 04:45: Object Oriented Programming (Theory)
04:45 - 05:45: Engineering Mathematics-III (Theory)
--------------------------------------------------
Tuesday:
11:00 - 12:00: Object Oriented Programming (Theory)
12:00 - 13:00: Discrete Structure & Graph Theory (Theory)
01:00 - 01:15: Break
01:15 - 03:15: Assembly Language Programming (Lab)
01:15 - 03:15: Object Oriented Programming (Lab)
01:15 - 03:15: Analog & Digital Electronics (Lab)
01:15 - 03:15: Computer Skill (Lab)
03:15 - 03:45: Long Break
03:45 - 04:45: Engineering Mathematics-III (Theory)
04:45 - 05:45: Assembly Language Programming (Theory)
--------------------------------------------------
Wednesday:
11:00 - 12:00: Object Oriented Programming (Theory)
12:00 - 13:00: Discrete Structure & Graph Theory (Theory)
01:00 - 01:15: Break
01:15 - 03:15: Analog & Digital Electronics (Lab)
01:15 - 03:15: Computer Skill (Lab)
01:15 - 03:15: Object Oriented Programming (Lab)
01:15 - 03:15: Assembly Language Programming (Lab)
03:15 - 03:45: Long Break
03:45 - 04:45: Engineering Mathematics-III (Theory)
04:45 - 05:45: Assembly Language Programming (Theory)
--------------------------------------------------
```

Fig: Generate Time Table



B.E. IT Semester 3 Syllabus

Engineering Mathematics-III

UNIT-I: Ordinary differential equations: Complete solution, Operator D, Rules for finding complementary function, the inverse operator, Rules for finding the particular integral, Method of variation of parameters, Cauchy's and Legendre's linear differential equations.

UNIT-II: Laplace Transform: Definition, standard forms, properties of Laplace transform, inverse Laplace transform, Initial and final value theorem, Convolution theorem, Laplace transform of impulse function, Unit step function, Laplace transforms of periodic function.

UNIT-III: a) Applications of Laplace Transform: Solution of Linear equations, Simultaneous differential equation by Laplace transform method.

b) Fourier Transform: Definition, standard forms, Fourier transforms, properties of Fourier transforms, Convolution theorem, Fourier sine and Fourier cosine transforms and integrals, inverse Fourier transforms.

Fig: Syllabus PDF

4.2 Analysis of the Output

1. First image output shows the `Select Program` Frame. We can select the required Program for Time table Generation through the drop-down list.

2. Second image shows the `Select Semester Type` Frame. We can select the required Semester in the drop-down list depending upon the type of semester i.e. `Autumn` or `Spring`.

3. Third image shows the `Select Active Days` Frame. We can check the boxes of required days or active days throughout the semester for time table generation. We can also specify the duration of the semester in this frame.

4. Fourth image shows the `Generate Timetable` Frame. We can generate a timetable by clicking the `Generate Timetable` button. The generated time table is shown in the white space below the button.

5. There are two buttons in the bottom side of the `Generate Timetable` Frame. They are `Show Syllabus` and `Export Timetable`. The fifth image shows the syllabus pdf of the selected program and semester which is shown after clicking the `Show Syllabus` button.

Conclusion

5.1 Key Achievements

The **Timetable Generation System** was developed with several important achievements that contribute to its functionality and usability:

1. **User-friendly Timetable System**: The project successfully developed a **user-friendly timetable generation system** that enables students to create their personalized timetables based on their program, semester, and selected days of the week. The system allows students to view their schedule in a clear, organized manner, helping them manage their academic tasks efficiently.

2. **CSV Integration for Dynamic Data Handling**: One of the key features of the project is the **integration of CSV files** for managing subject data dynamically. By reading data from CSV files, the system ensures that subject lists can be easily updated without modifying the code, making the system more flexible and maintainable in the long run. This feature allows easy modification of subjects across different programs and semesters without needing code changes.

3. **PDF Viewer for Syllabus Access**: The system also incorporates the **PDF viewer functionality** that allows students to view the syllabus directly from the interface. This feature streamlines access to important program and semester information, helping students refer to their course syllabus conveniently while planning their schedule.

4. **Implementation of Java Programming Concepts**: The project involved the implementation of several key Java programming concepts such as **file handling**, **graphical user interface (GUI) design**, and **object-oriented programming (OOP)** principles. These concepts were applied effectively to create a functional, interactive application that addresses the real-world need of timetable management for students.

5.2 Conclusion

The **Timetable Generation System** successfully automates the timetable creation process, providing students with a flexible and user-friendly solution. By allowing students to generate timetables based on their program, semester, and available days, the system saves time and effort. Its intuitive graphical user interface (GUI) makes it accessible to users with minimal technical knowledge.

The dynamic integration of subject data through CSV files ensures easy updates, while the built-in syllabus viewer streamlines access to academic information. Overall, the system is scalable and adaptable to different programs or future academic years. Its flexible design ensures it can meet the diverse needs of students, and the user-friendly interface makes it a practical tool for managing academic schedules.

5.3 Future Improvements

Potential future enhancements include:

- **Exporting Timetable**: Implementing export options (e.g., PDF, Excel) would allow students to easily share or print their timetables.
- **Conflict Detection**: Adding conflict detection would ensure that no scheduling overlaps occur, improving the system's reliability.
- **Manual Editing**: Allowing students to manually edit their timetables would provide greater flexibility and customization.
- **Mobile Version**: A mobile app would enable students to access their timetables on the go, further enhancing convenience.

These improvements would make the system more powerful, flexible, and user-friendly.

5.4 References

- "Java: The Complete Reference" by Herbert Schildt
- "Head First Java" by Kathy Sierra & Bert Bates
- "Timetable Scheduling System Using Java" by N. S. Yadav, M. S. Pradhan
- "A Study on the Timetable Scheduling Algorithm" by K. Srinivasa Rao, A. Usha Rani
- Oracle Java Documentation (Swing & AWT)