# TIMETABLE GENERATOR



An

Object-Oriented Programming through Java Course Project Report in partial fulfilment of the degree

**Bachelor of Technology**

In

**Computer Science & Engineering**

**By**

U. REETHU VARMA 2103A52037

CH. RISHIKA 2103A52010

**Submitted to**

**DR T SAMPATH KUMAR**

**Assistant Professor**

**Department of Computer Science And Engineering**





# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**CERTIFICATE**

**This is to certify that the Object Oriented Programming through Java – Course Project Report entitled “TIMETABLE GENERATOR” is a record of bonafide work carried out by the student REETHU VARMA & RISHIKA *bearing Roll No(s) 2103A52037 AND 2103A52010 during the academic* year 2023 in partial fulfillment of the award of the degree of *Bachelor of Technology* in Computer Science & Engineering by the SR University, Warangals.**

**Lab In-charge Head of the Department**

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| **CONTENTS** | **PAGE NO.** |
| Abstract | 4 |
| Objective of the project | 5 |
| Definition of the elements used in the project | 6-7 |
| Design | 8-10 |
| Implementation | 11-17 |
| Result screen | 18-20 |
| Conclusion | 21 |

**ABSTRACT**

The Time Table Generator presented in this study addresses the complex task of scheduling courses, classes, and resources within educational institutions. Efficient timetable creation is crucial for optimizing resource utilization, minimizing conflicts, and enhancing overall productivity. Traditional manual methods of timetable generation are time-consuming and error-prone, often leading to suboptimal schedules.

The TTG employs advanced algorithms and artificial intelligence techniques to automate the timetable generation process, ensuring the creation of schedules that meet diverse constraints and preferences. The system takes into account various factors such as teacher availability, room capacities, course prerequisites, and student preferences. Machine learning models are utilized to predict and adapt to dynamic scheduling patterns, optimizing the allocation of resources over time.

# OBJECTIVE OF THE PROJECT

# The objective of the provided code is to create a Timetable Generator application with a graphical user interface (GUI) using Java's Swing library. The application enables users to input the number of subjects, the number of timeslots, and details about each subject and timeslot. It then generates a random timetable based on this input and displays it in a table format within the GUI.

# DEFINITIONS OF THE ELEMENTS THAT ARE USED IN THE PROJECT

**Swings:**

Java Swing is a lightweight Java graphical user interface (GUI) widget toolkit that includes a rich set of widgets.

**JFrame:**

JFrame is a class in Swing that represents a top-level container (window) for a Java application. It provides a standard framework for GUI applications and includes features like closing operations and resizing.

**JTable:**

JTable is a Swing component that provides a grid-based view for displaying and editing data, typically in a tabular form. It is often used to represent data in rows and columns.

**DefaultTableModel:**

DefaultTableModel is a class that provides a default data model implementation for JTable. It allows dynamic manipulation of data in a tabular form.

**JTextField:**

JTextField is a Swing component that allows the user to input a single line of text. It provides an area where the user can type in information.

**JButton:**

JButton is a Swing component that represents a button. It can trigger actions or events when clicked.

**JPanel:**

Panel, a part of the Java Swing package, is a container that can store a group of components.

**JLabel:**

JLabel is a class of java Swing,JLabel is used to display a short string or an image icon.

**ActionListener:**

ActionListener is an interface in Java that provides a mechanism to respond to events triggered by user actions, such as button clicks. It includes the actionPerformed method that is called when the registered event occurs.

**Map<String, Map<String, String>>:**

This is a nested Map structure used to represent the timetable data. The outer Map uses day names (e.g., "MON") as keys, and the corresponding values are inner Map objects. The inner Map uses timeslots as keys and subject names as values.

**Random:**

Random is a class in Java that provides methods for generating pseudorandom numbers. In this code, it is used to randomly assign subjects to timeslots.

**SwingUtilities.invokeLater:**

This method ensures that the Swing components are created and modified on the event dispatch thread, which is the thread responsible for handling GUI events.

**BufferedWriter and FileWriter:**

These classes are used for writing characters to a file. BufferedWriter provides efficient writing by buffering characters, and FileWriter is used to write characters to a file.

**generateTimetable method:**

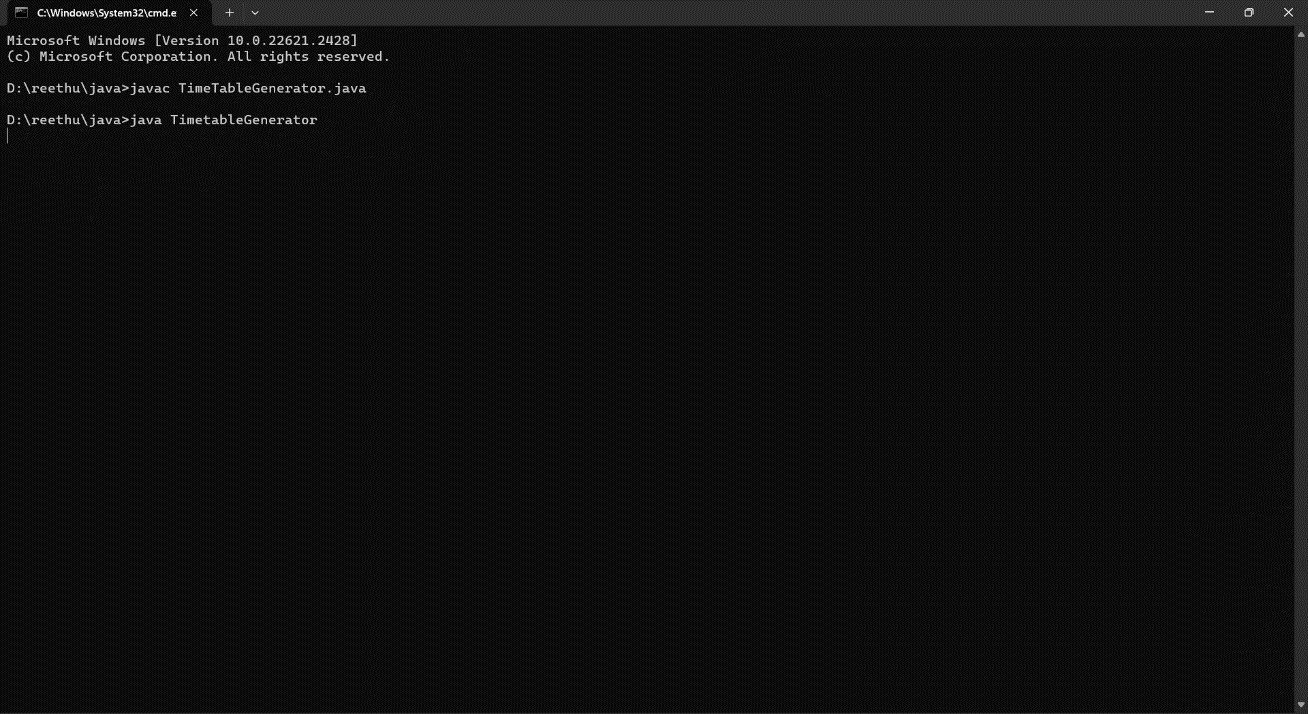
This method generates a random timetable based on the specified number of subjects, timeslots, subject names, and timeslot details. It returns a Map representing the timetable structure.

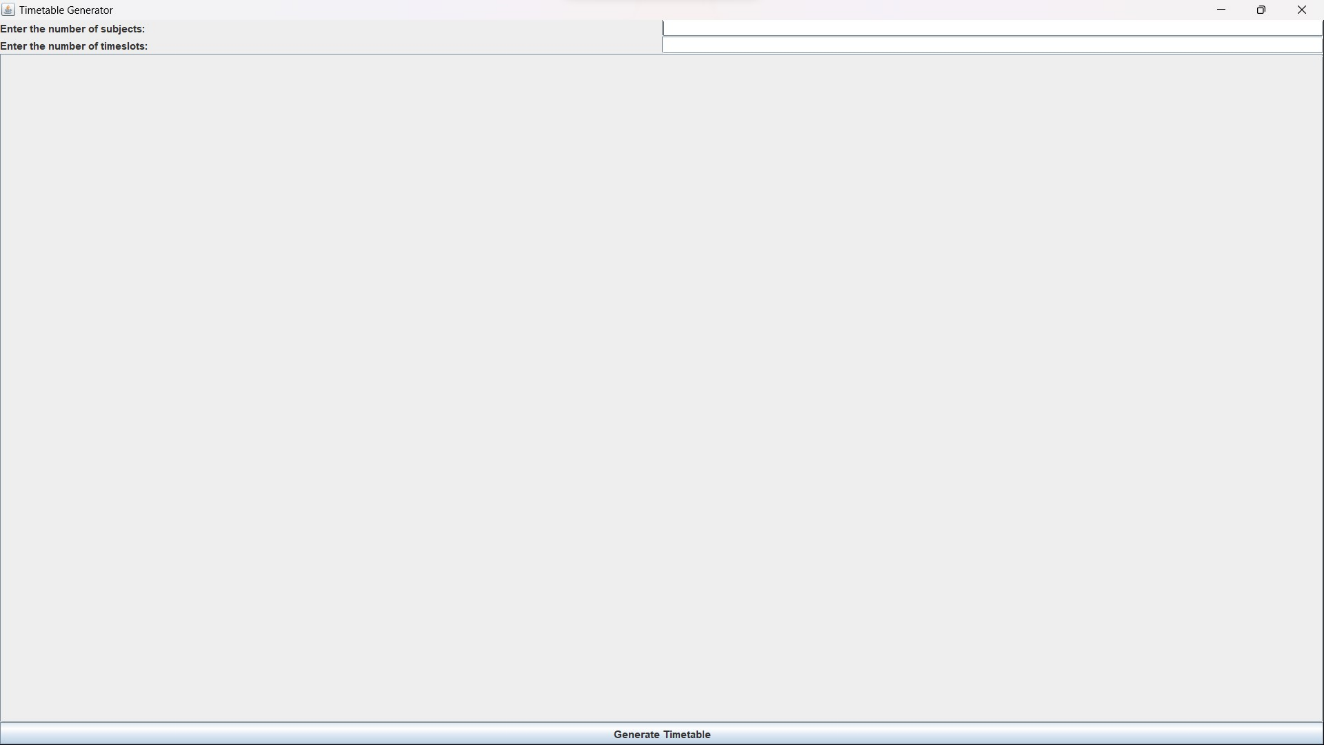
**saveTimetableToFile method:**

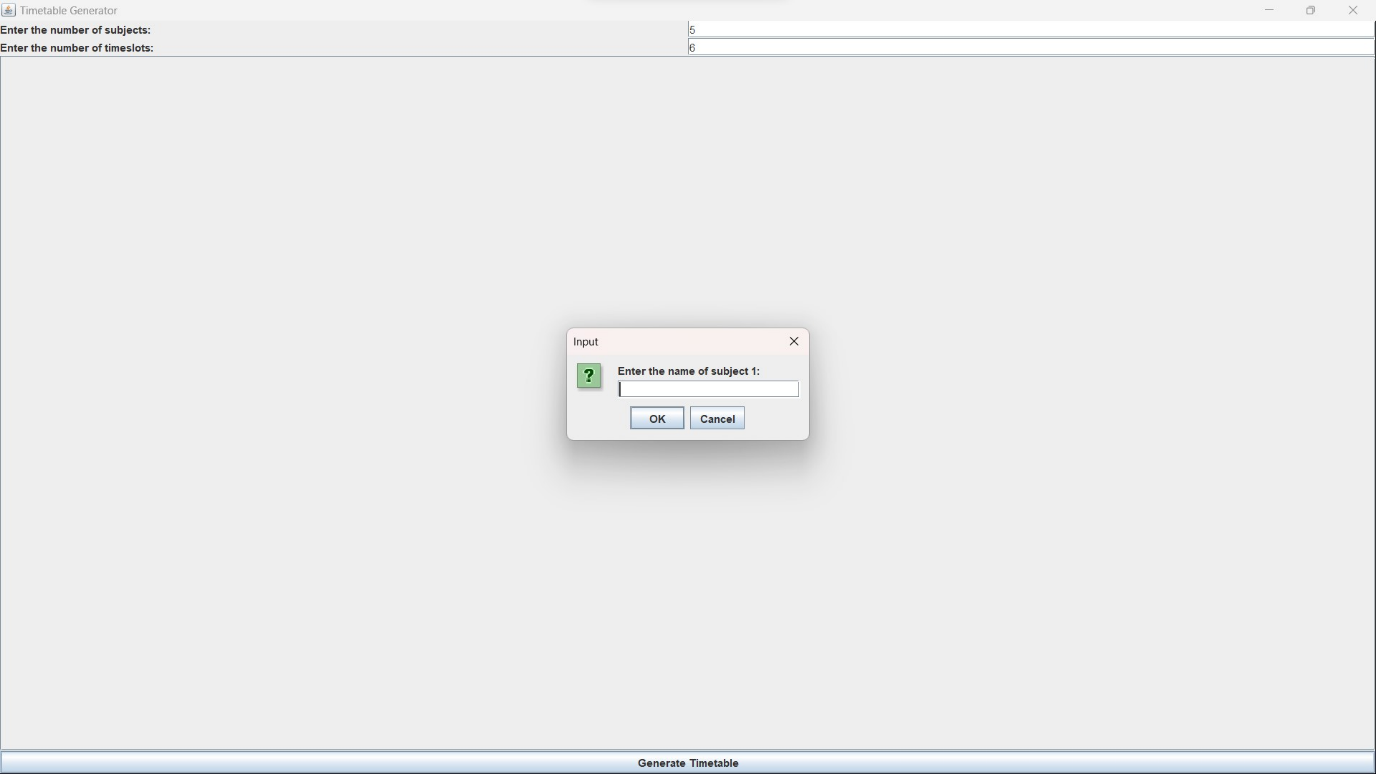
This method takes the generated timetable data and writes it to a text file in a tabular format, including column headings and timetable data.

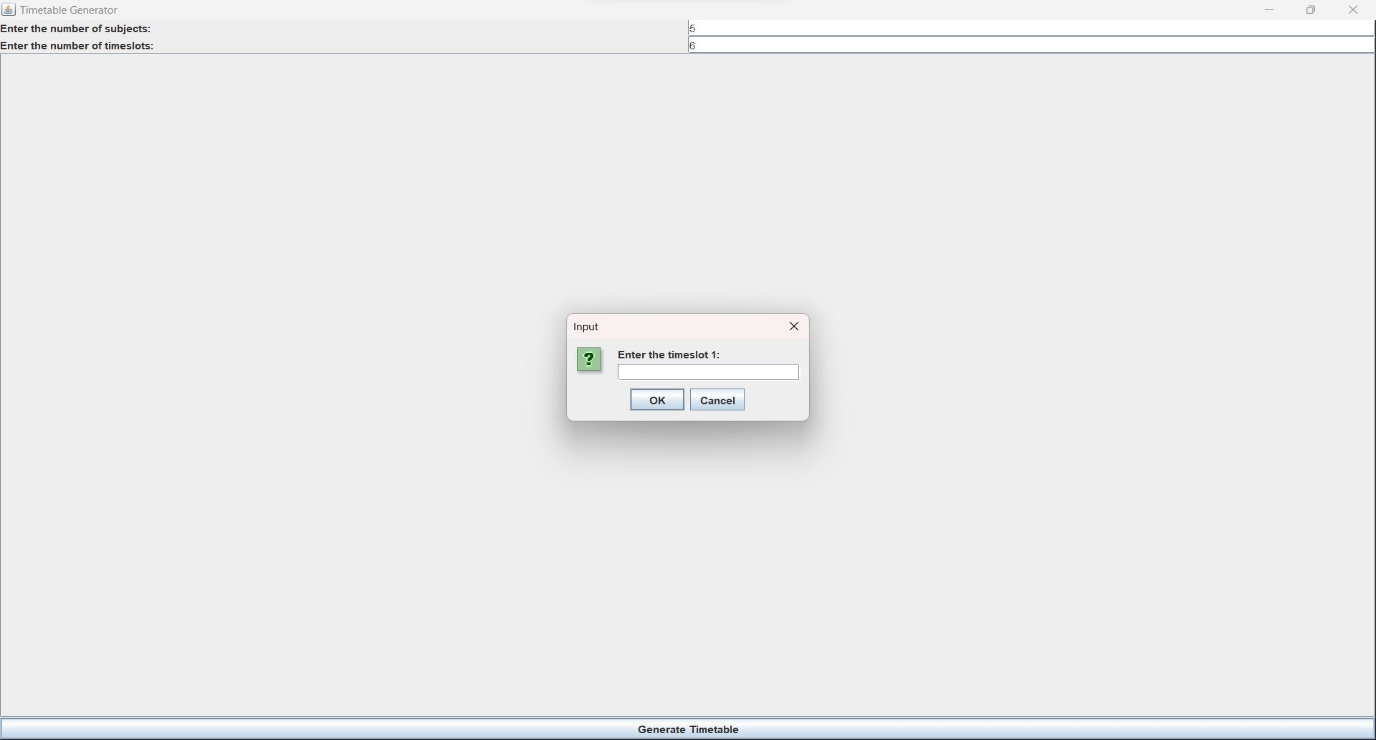
# DESIGN

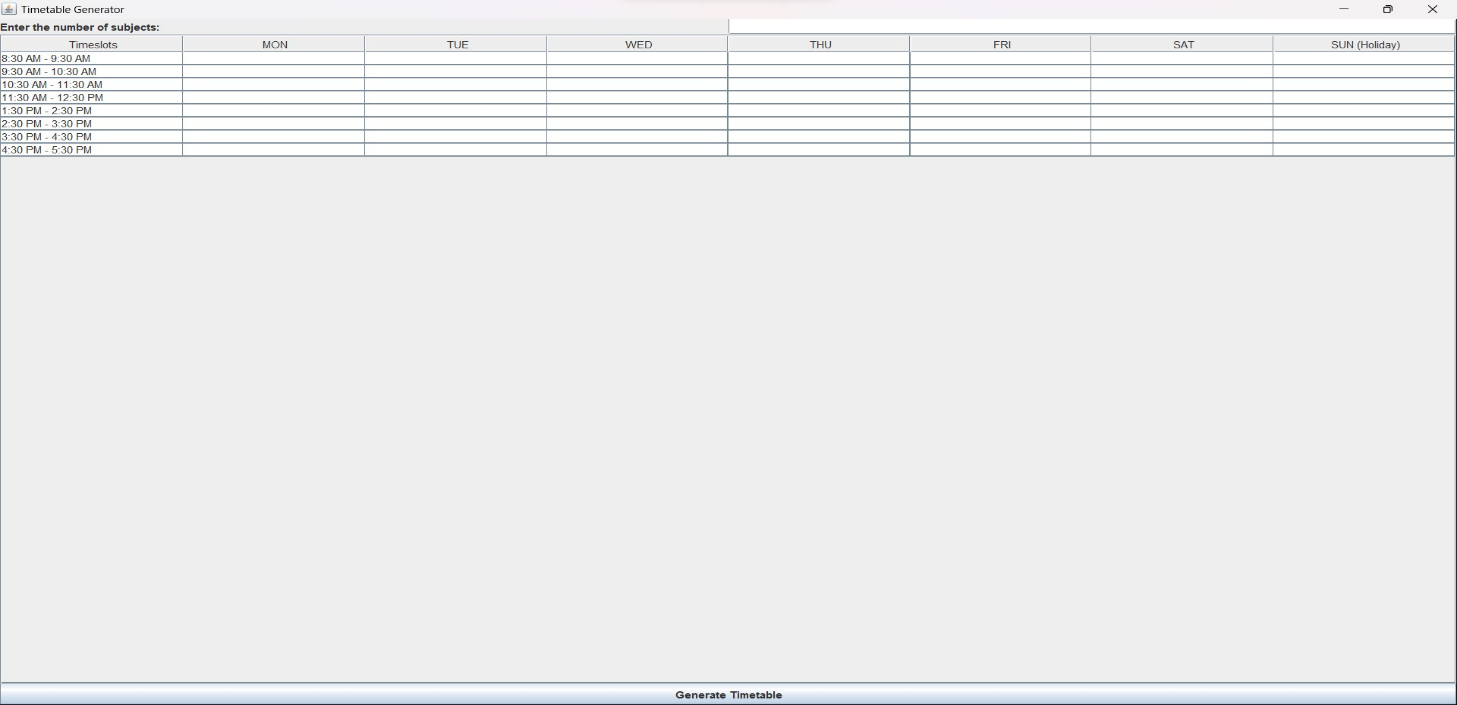
**SCREENS:**











# IMPLEMENTATION

**CODE**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.\*;

import java.util.List;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

class TimetableGenerator {

private JFrame frame;

private JTable timetableTable;

private DefaultTableModel tableModel;

private JTextField subjectField, timeslotField;

private JButton generateButton;

private static final String[] DAYS\_OF\_WEEK = {"MON", "TUE", "WED", "THU", "FRI", "SAT", "SUN (Holiday)"};

public TimetableGenerator() {

frame = new JFrame("Timetable Generator");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(800, 400);

frame.setLayout(new BorderLayout());

JPanel inputPanel = new JPanel();

inputPanel.setLayout(new GridLayout(2, 2));

subjectField = new JTextField(20);

timeslotField = new JTextField(20);

inputPanel.add(new JLabel("Enter the number of subjects: "));

inputPanel.add(subjectField);

inputPanel.add(new JLabel("Enter the number of timeslots: "));

inputPanel.add(timeslotField);

generateButton = new JButton("Generate Timetable");

generateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

generateTimetable();

}

});

tableModel = new DefaultTableModel();

timetableTable = new JTable(tableModel);

JScrollPane tableScrollPane = new JScrollPane(timetableTable);

frame.add(inputPanel, BorderLayout.NORTH);

frame.add(generateButton, BorderLayout.SOUTH);

frame.add(tableScrollPane, BorderLayout.CENTER);

frame.setVisible(true);

}

public static Map<String, Map<String, String>> generateTimetable(int numSubjects, int numTimeslots,String[] subjectNames,String[] timeslots) {

Random random = new Random();

Map<String, Map<String, String>> timetable = new HashMap<>();

// Define subject names

String[] subjectNames = new String[numSubjects];

for (int i = 0; i < numSubjects; i++) {

subjectNames[i] = "Subject " + (i + 1);

}

// Define timeslots

String[] timeslots = new String[numTimeslots];

for (int i = 0; i < numTimeslots; i++) {

timeslots[i] = "Timeslot " + (i + 1);

}

for (String dayOfWeek : DAYS\_OF\_WEEK) {

if (!dayOfWeek.endsWith("(Holiday)")) {

Map<String, String> classTimetable = new HashMap<>();

List<String> availableSubjects = new ArrayList<>(Arrays.asList(subjectNames));

for (String timeslot : timeslots) {

if (availableSubjects.isEmpty()) {

break;

}

int subjectIndex = random.nextInt(availableSubjects.size());

String subject = availableSubjects.remove(subjectIndex);

classTimetable.put(timeslot, subject);

}

timetable.put(dayOfWeek, classTimetable);

} else {

timetable.put(dayOfWeek, Collections.singletonMap("Holiday", "Holiday"));

}

}

return timetable;

}

public void generateTimetable() {

tableModel.setColumnCount(0);

int numSubjects = Integer.parseInt(subjectField.getText());

int numTimeslots = Integer.parseInt(timeslotField.getText());

String[] subjectNames = new String[numSubjects];

String[] timeslots = new String[numTimeslots];

for (int i = 0; i < numSubjects; i++) {

subjectNames[i] = JOptionPane.showInputDialog("Enter the name of subject " + (i + 1) + ": ");

}

for (int i = 0; i < numTimeslots; i++) {

timeslots[i] = JOptionPane.showInputDialog("Enter the timeslot " + (i + 1) + ": ");

}

tableModel.addColumn("Timeslots"); // Add the "Timeslots" heading

for (String dayOfWeek : DAYS\_OF\_WEEK) {

tableModel.addColumn(dayOfWeek);

}

Map<String, Map<String, String>> timetable = generateTimetable(numSubjects, numTimeslots, subjectNames, timeslots);

for (String timeslot : timeslots) {

Vector<String> rowData = new Vector<>();

rowData.add(timeslot);

for (String dayOfWeek : DAYS\_OF\_WEEK) {

rowData.add("");

}

tableModel.addRow(rowData);

}

for (String timeslot : timeslots) {

int rowIndex = getRowIndexForTimeslot(timeslot);

for (String dayOfWeek : DAYS\_OF\_WEEK) {

if (!dayOfWeek.endsWith("(Holiday)")) {

int columnIndex = getColumnIndexForDayOfWeek(dayOfWeek);

String subject = timetable.get(dayOfWeek).get(timeslot);

tableModel.setValueAt(subject, rowIndex, columnIndex);

}

}

}

// Automatically save the timetable to a file

saveTimetableToFile(subjectNames, timeslots, DAYS\_OF\_WEEK);

}

public int getRowIndexForTimeslot(String timeslot) {

for (int i = 0; i < tableModel.getRowCount(); i++) {

if (tableModel.getValueAt(i, 0).equals(timeslot)) {

return i;

}

}

return -1;

}

public int getColumnIndexForDayOfWeek(String dayOfWeek) {

for (int i = 0; i < tableModel.getColumnCount(); i++) {

if (tableModel.getColumnName(i).equals(dayOfWeek)) {

return i;

}

}

return -1;

}

public void saveTimetableToFile(String[] subjectNames, String[] timeslots, String[] daysOfWeek) {

try {

String filePath = "timetable.txt"; // Specify the file path where you want to save the timetable

BufferedWriter writer = new BufferedWriter(new FileWriter(filePath));

// Write the column headings

writer.write("Timeslots\t");

for (String dayOfWeek : daysOfWeek) {

writer.write(dayOfWeek + "\t");

}

writer.write("\n");

// Write the timetable data

for (int row = 0; row < tableModel.getRowCount(); row++) {

writer.write(timeslots[row] + "\t");

for (int col = 1; col < tableModel.getColumnCount(); col++) {

writer.write(tableModel.getValueAt(row, col) + "\t");

}

writer.write("\n");

}

writer.close();

System.out.println("Timetable saved to " + filePath);

} catch (IOException e) {

e.printStackTrace();

System.err.println("Error saving the timetable.");

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new TimetableGenerator();

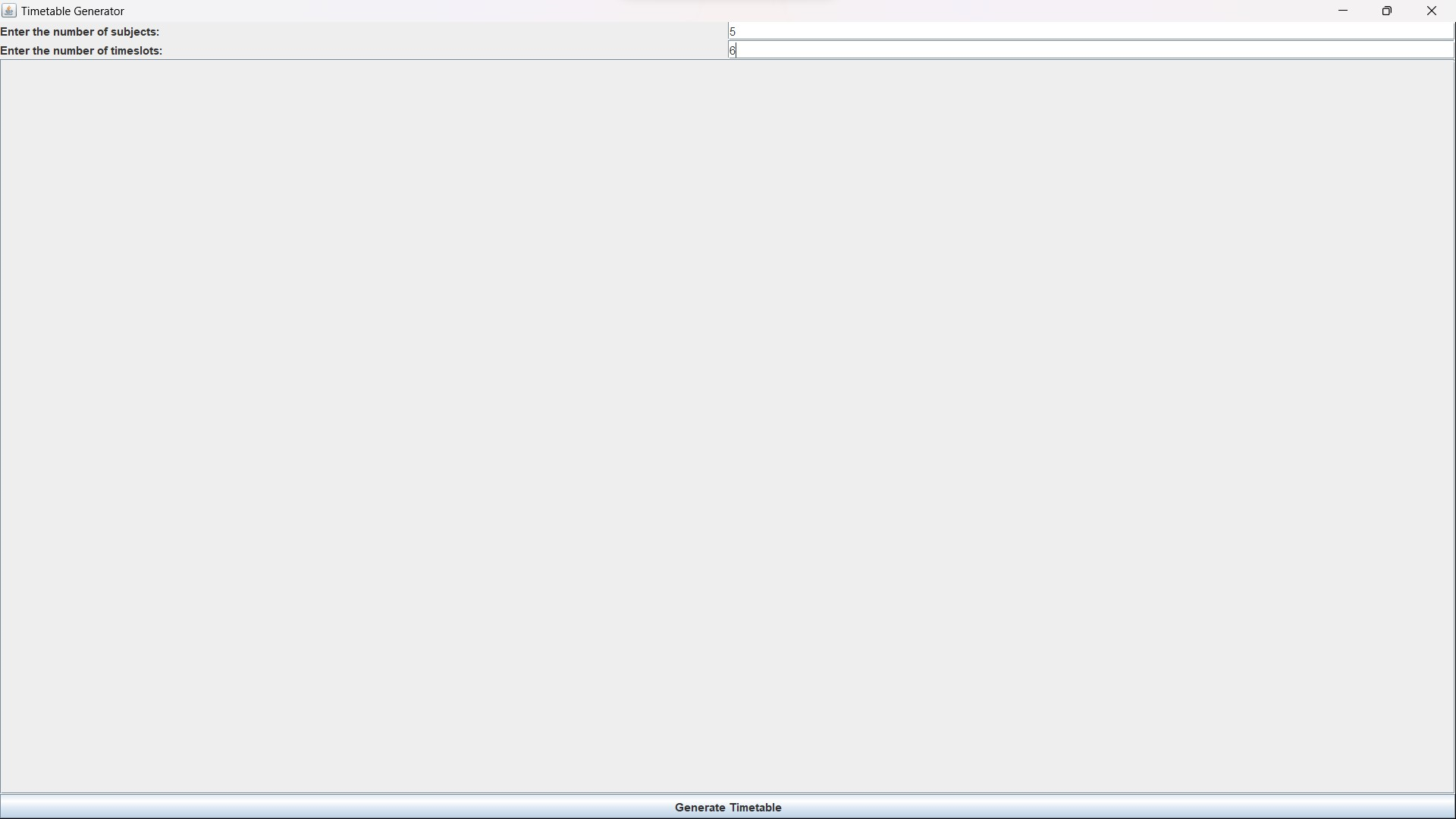
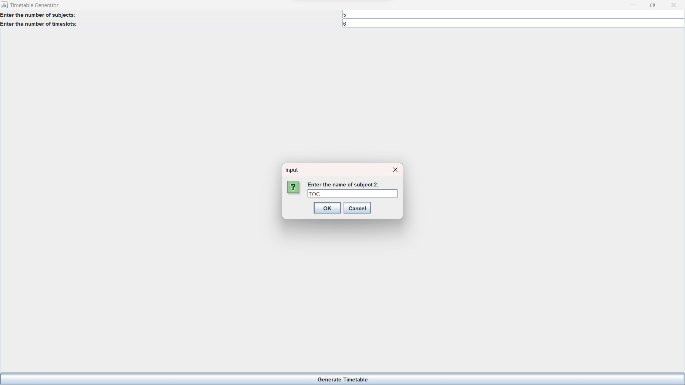
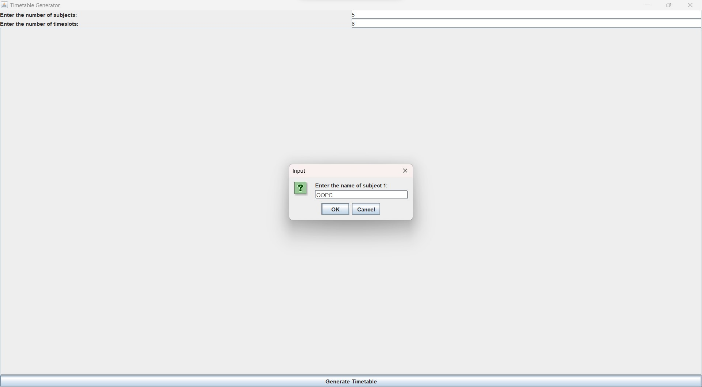
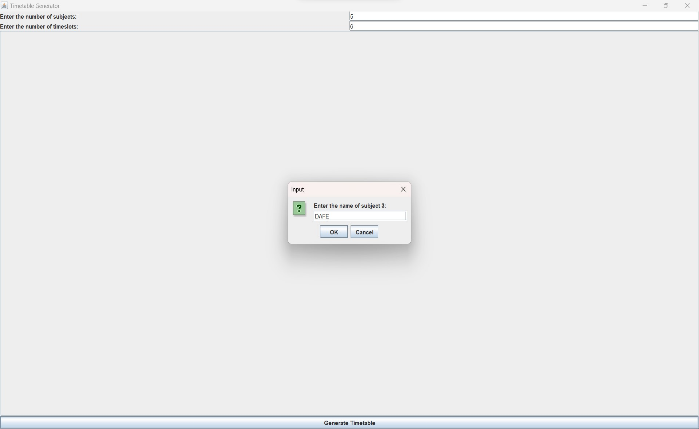
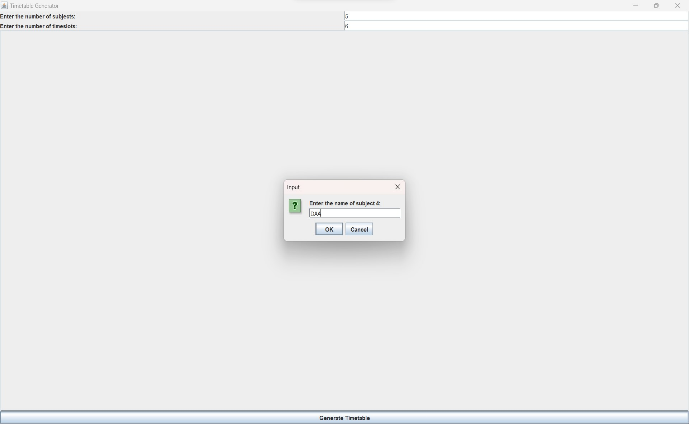
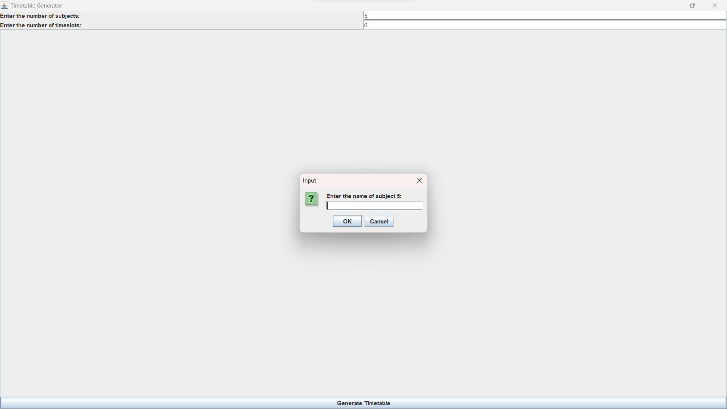
}

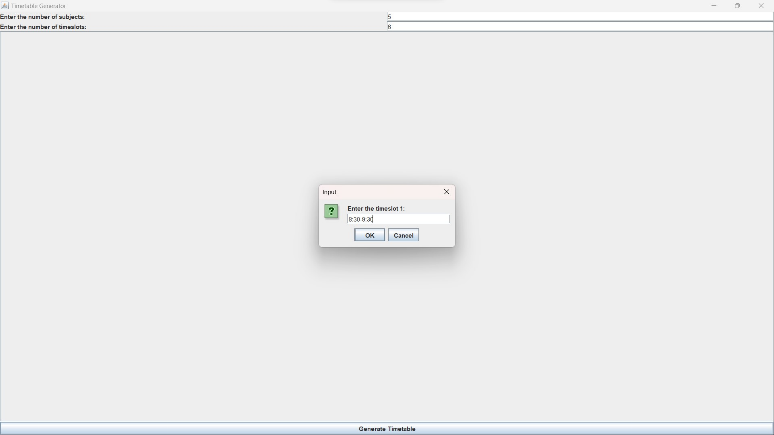
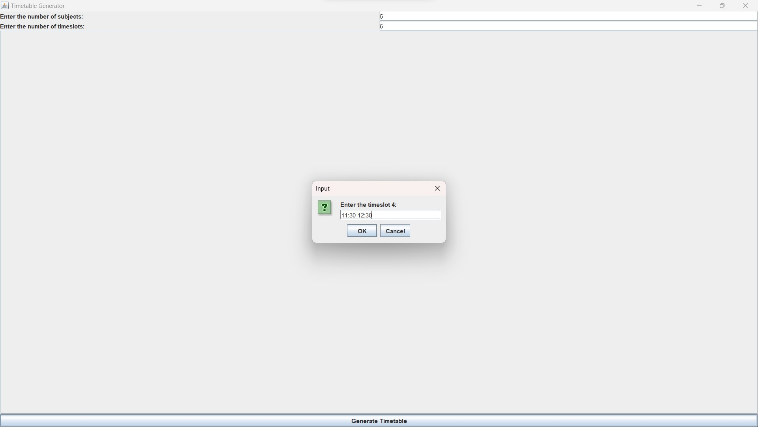
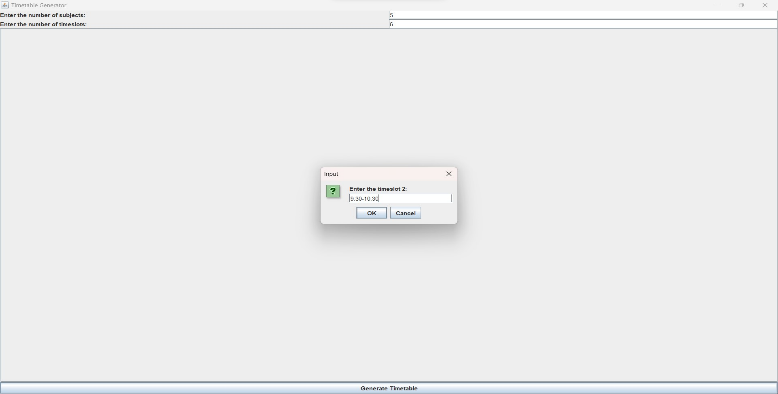
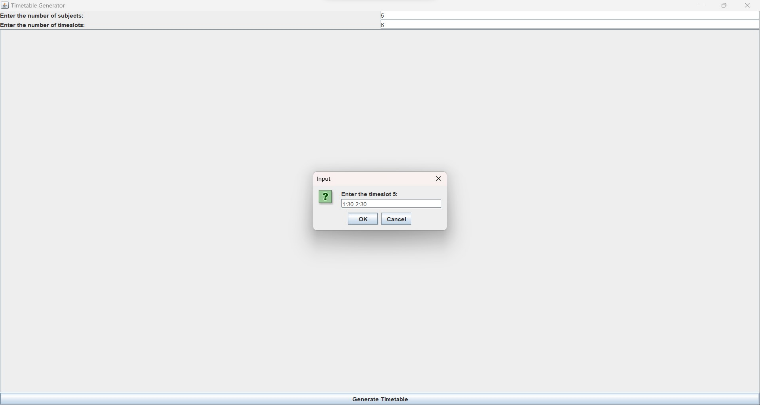
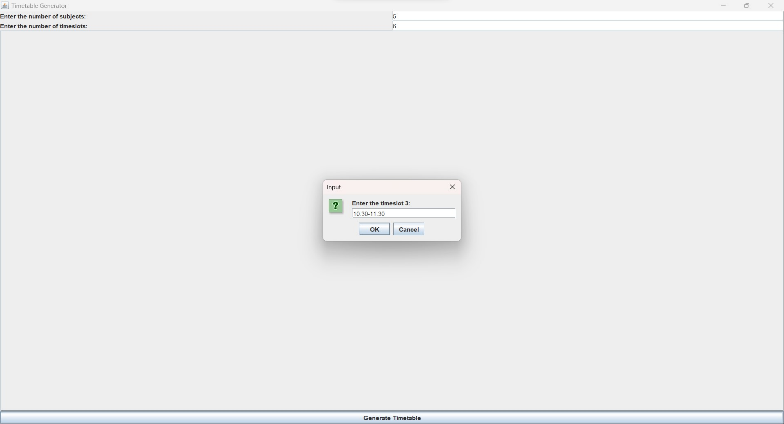
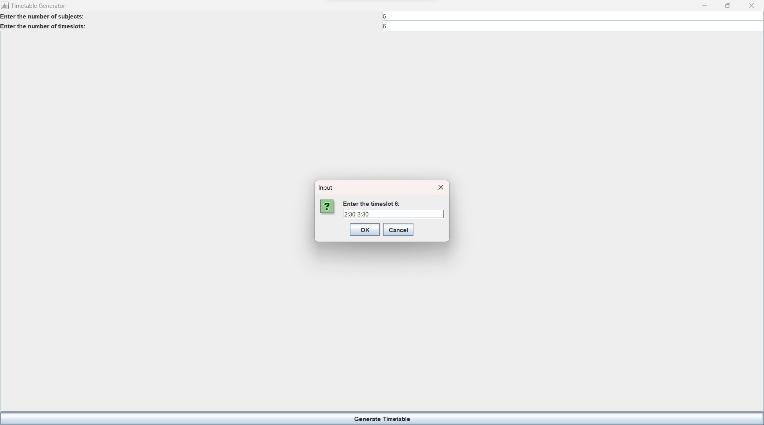
});

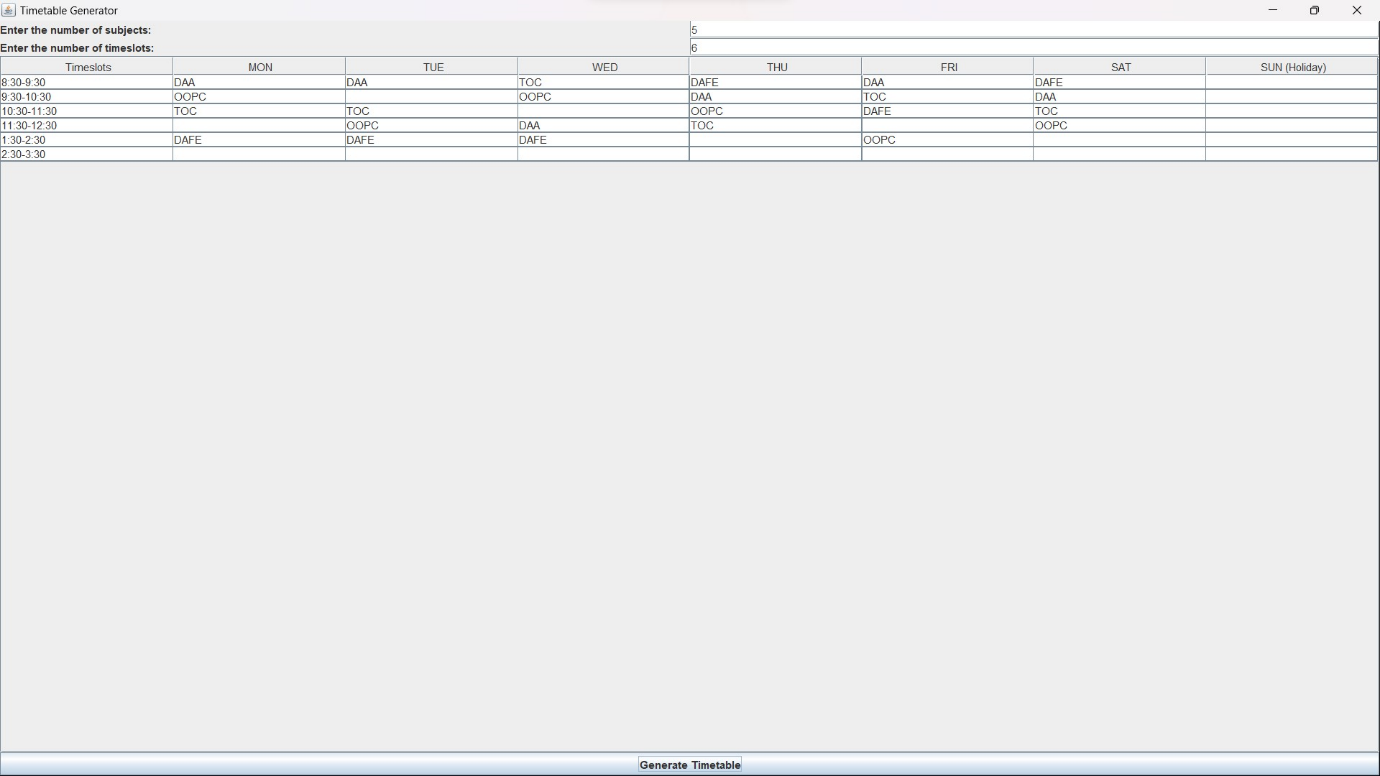
}

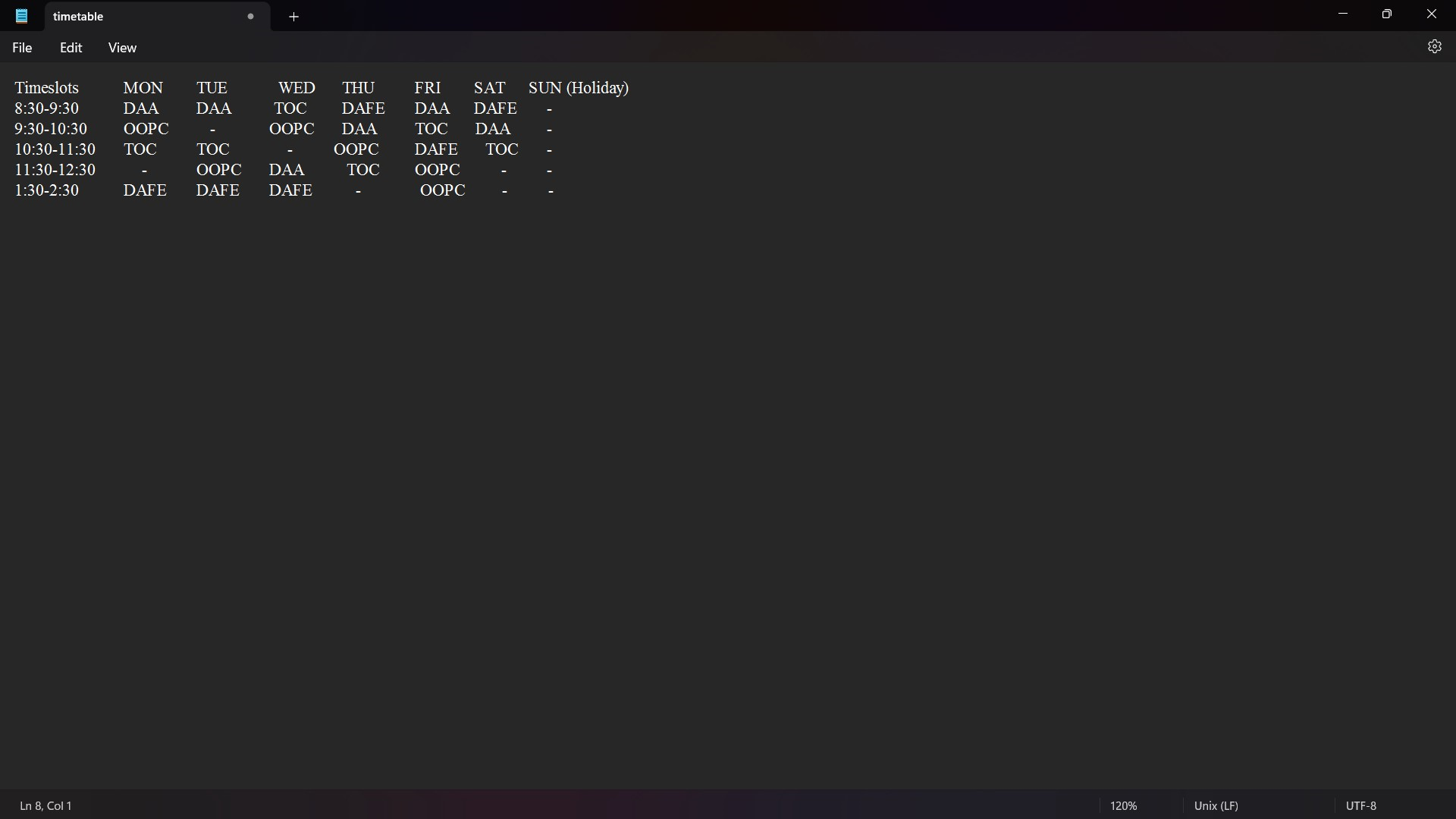
}

**RESULT SCREEN**









**CONCLUSION**

In summary, the Timetable Generator project is a Java-based application utilizing Swing for GUI development. It allows users to input the number of subjects, timeslots, and details, generating a random timetable that is displayed in a user-friendly table. The modular design enhances code organization, and the automatic file saving feature adds practicality for users to retain generated schedules. This project successfully combines functionality and usability,

providing a versatile tool for creating and managing timetables in various settings.