DIGITAL NOTICE BOARD

A PROJECT REPORT

Submitted by

AKASH R (2303811710421007)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"DIGITAL NOTICE BOARD"** is the bonafide work of **AKASH R (2303811710421007)** who carriedout the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

SIGNATURE

Mr. M. Saravanan, M.E.,

SUPERVISOR

ASSISTANTP ROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"DIGITAL NOTICE BOARD"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201- JAVA PROGRAMMING.**

.

Signature

R.Akash

AKASH R

Place: Samayapuram

Date: 02.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project guide **Mr.M.SARAVANAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This Java program is a desktop application that provides a graphical user interface (GUI) for managing and viewing a weekly meeting timetable. Built using the Swing library, the application displays daily schedules with meetings categorized by time, description, and location. Each day of the week is associated with a unique schedule, offering a dynamic and tailored user experience.The main window features a clean and organized layout divided into three sections. At the top, a real-time clock displays the current date and time, which updates every second using a timer. The central panel showcases the timetable in a grid format, where meetings are neatly arranged under headers for "Time," "Meeting," and "Location." At the bottom, navigation buttons labeled for each day of the week allow users to view the schedule for their chosen day with a single click.The program initializes by displaying Monday's timetable by default. When a user selects a day, the application updates the timetable dynamically, clearing the previous data and replacing it with the selected day's schedule. The design ensures that users can navigate between days seamlessly without refreshing or restarting the application.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
This Java program is a desktop application that provides a graphical user interface (GUI) for managing and viewing a weekly meeting timetable. Built using the Swing library, the application displays daily schedules with meetings categorized by time, description, and location. Each day of the week is associated with a unique schedule, offering a dynamic and tailored user experience.The main window features a clean and organized layout divided into three sections.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

This Java-based application is to provide a user-friendly graphical user interface (GUI) for managing and viewing a weekly meeting timetable. Using the Swing library, the program is designed to display meeting schedules for each day of the week in an organized and dynamic format. The application caters to users who need to stay informed about their daily meetings, including details such as time, description, and location.The main goal is to offer a simple yet efficient tool for navigating and updating schedules. Users can switch between days using dedicated buttons, and the timetable updates dynamically without the need for restarting the application. Additionally, the program integrates a real-time clock at the top of the interface, displaying the current date and time, which updates every second to keep users informed.

1.2 Overview

This Java-based application is a GUI-driven weekly meeting scheduler built with the Swing library. Its primary function is to provide a clear, interactive interface for users to view and manage meeting schedules for each day of the week. Each day has its unique set of meetings, categorized by time, description, and location, making it easy to plan and track daily activities.The interface is divided into three main sections: a real-time clock at the top displaying the current date and time, a central grid panel for showing the daily timetable, and a set of navigation buttons at the bottom for selecting different days of the week. The timetable updates dynamically when a day is selected, ensuring a smooth user experience.The program starts with Monday's schedule by default and allows users to navigate seamlessly to other days using labeled buttons. The real-time clock updates every second, adding an element of practicality and ensuring users are always aware of the current time.This application is ideal for professionals or teams who require an organized approach to managing weekly schedules. It combines functionality with a user-friendly design, making it a reliable tool for enhancing productivity and maintaining an organized workflow.

1.3.Java programming concepts

Here are some key Java programming concepts**:**

1. Object-Oriented Programming (OOP)

- ✓ **Class and Object:** A class is a blueprint for objects. Objects are instances of classes.
- ✓ **Encapsulation:** Restricting access to certain parts of an object to protect data integrity.
- ✓ **Inheritance:** A mechanism to create a new class using properties and methods of an existing class.
- ✓ **Polymorphism:** The ability to process objects differently based on their data type or class.
- ✓ **Abstraction:** Hiding the implementation details and exposing only the functionality.

2. Basic Syntax

- ✓ **Variables:** Used to store data values.
- ✓ **Data Types: I**ncludes primitives (int, float, boolean) and reference types (objects, arrays).
- ✓ **Control Statements: if, else, switch, loops (for, while, do-while).**

3. Collections Framework

- ✓ **Lists, Sets, Maps:** Interfaces like ArrayList, HashSet, and HashMap to manage collections of objects.

4. Exception Handling

- ✓ Use try, catch, finally, and throw/throws to handle runtime errors.

5. Multithreading

- ✓ Creating threads using Thread class or Runnable interface for concurrent execution.

6. File I/O

- ✓ Using File, BufferedReader, BufferedWriter, and Scanner classes for file handling.

7. Java Swing

- ✓ A GUI toolkit for creating window-based applications (e.g., JFrame, JPanel, JButton).

8. Generics

- ✓ Writing flexible and type-safe code (e.g., ArrayList<Integer>).

9. Annotations

- ✓ Metadata for classes, methods, or fields (e.g., @Override, @FunctionalInterface).

10. Lambda Expressions

- ✓ Concise representation of anonymous functions for functional programming.

11. JDBC

- ✓ Java Database Connectivity for connecting and interacting with databases.

12. Packages

- ✓ Organizing classes and interfaces (java.util, java.io).

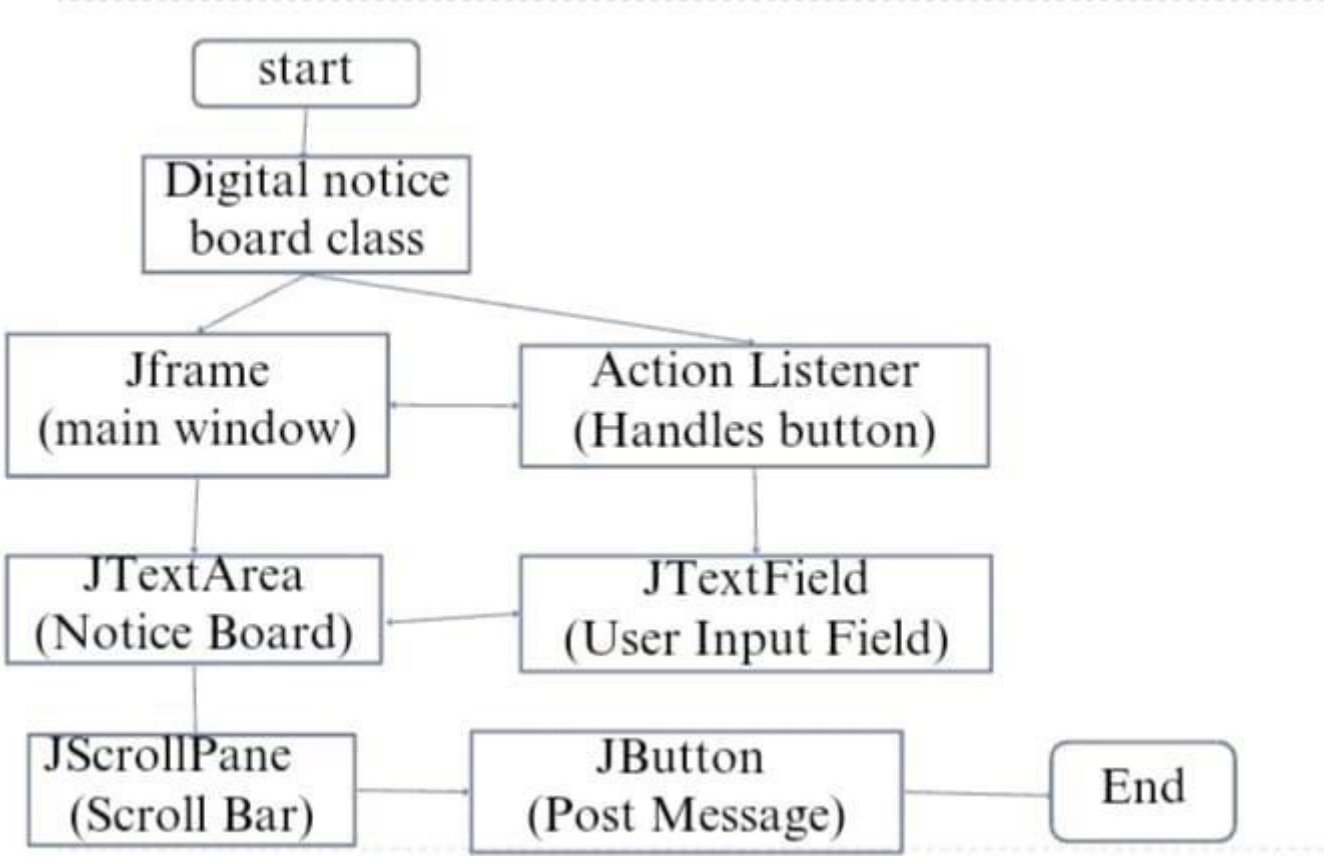
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work involves the development of a graphical application using Java Swing to display a one-week meeting timetable. The application will offer an interactive and user-friendly interface, allowing users to view scheduled meetings for each day of the week.The program integrates real-time updates with a live clock feature, providing the current date and time at the top of the interface. Users can navigate between days using buttons, and the timetable will adapt accordingly, showcasing the meetings planned for that specific day.The implementation emphasizes modularity and reusability. The meeting schedules are predefined and organized as a 3D array for easy data management. The application dynamically generates and updates the GUI elements, ensuring a responsive user experience.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Data Module

Purpose: Maintains the meeting schedule data for all days of the week.

Description: Uses a 3D array (WEEK_MEETINGS) to store meeting details, including time, description, and location, for each day. This ensures centralized and easily manageable data.

3.2 User Interface Module

Purpose: Manages the graphical layout and interaction.

Description: Utilizes JFrame, JPanel, and other Swing components to create a responsive UI. The BorderLayout organizes elements such as the live clock, timetable display, and navigation buttons. A GridLayout arranges timetable details (time, meeting, and location) for clear visualization.

3.3 Navigation Module

Purpose: Enables users to switch between days and view specific schedules.

Description: Provides day-specific buttons (Monday to Sunday). Each button triggers an ActionListener to load the respective day's meetings dynamically.

3.4 Live Clock Module

Purpose: Displays the current date and time at the top of the window.

Description: A Timer periodically updates the date-time label

using SimpleDateFormat.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

By leveraging the Swing framework, the application offers a user-friendly interface that allows users to navigate through daily schedules using buttons for each day of the week. Key functionalities, such as a dynamically updating timetable and a real-time clock, enhance the application's usability and responsivenessThe program demonstrates excellent modularity by organizing data in a multidimensional array (WEEK_MEETINGS) and separating UI components like the timetable panel, navigation buttons, and date-time label. Each module interacts seamlessly, ensuring a clear structure and maintainable codebase.Notably, the application caters to real-world scenarios by accommodating unique schedules for each day. The use of Timer ensures accurate time display, while the dynamic update of timetable entries keeps the interface current and relevant..In conclusion, this program serves as an effective tool for visualizing weekly schedules and can be further expanded for other use cases, such as personal planners or event management systems, with minimal modifications. Its clean design and functional implementation highlight the power and versatility of Java Swing.

4.2 FUTURE SCOPE

The future scope of this Java-based timetable application is promising, with several potential enhancements to improve its functionality and adaptability. Integrating a database can allow users to dynamically add, modify, or delete meetings, making it suitable for more personalized or large-scale scheduling needs. Adding user authentication would enable multiple users to access and manage their schedules securely.Expanding the application to include reminders or notifications via desktop alerts or email integration could significantly enhance its utility. Mobile app integration or synchronization with popular calendar platforms like Google Calendar or Microsoft Outlook could provide seamless access across devices. Additionally, incorporating advanced features such as color-coded events, search functionality, and recurring meeting options could further elevate the user experience.With these advancements, the application could evolve into a comprehensive scheduling tool, serving not only individuals but also teams and organizations looking for efficient time management solutions.

REFERENCES

Here are some useful references for your Java Swing timetable application:

1. GitHub - Java Calendar Project: This repository contains a desktop calendar application built with Java Swing, showcasing features like navigation between months, adding appointments, and visual indicators for scheduling. It can serve as an excellent reference for similar GUI-based scheduling applications. Visit the repository here.

2. Java Scheduling Example by Java Code Geeks: This article explains task scheduling in Java using ScheduledExecutorService and TimerTask, which can be adapted for periodic updates or event-driven triggers in your timetable application. Learn more here.

3. Oracle Swing Tutorials: For understanding the fundamentals of Java Swing, such as layouts, event handling, and component customization, Oracle's official Swing tutorials are a valuable resource. Check them out here.

APPENDIX-A

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Main extends JFrame {

    // Meetings for each day of the week (7 days)
    private static final String[][][] WEEK_MEETINGS = {
            // Monday - Unique schedule
            {{"9:00 AM", "Team Standup Meeting", "Conference Room 1"},
                    {"11:00 AM", "Client Call", "Zoom"},
                    {"1:00 PM", "Project Review", "Conference Room 2"},
                    {"3:00 PM", "Strategy Meeting", "Conference Room 1"},
                    {"5:00 PM", "End of Day Wrap-Up", "Zoom"}},

            // Tuesday - Unique schedule
            {{"9:00 AM", "Team Standup Meeting", "Conference Room 1"},
                    {"11:00 AM", "Client Call", "Zoom"},
                    {"1:00 PM", "Product Demo", "Conference Room 2"},
                    {"3:00 PM", "Strategy Planning", "Conference Room 1"},
                    {"5:00 PM", "End of Day Wrap-Up", "Zoom"}},

            // Wednesday - Unique schedule
            {{"9:00 AM", "Team Standup Meeting", "Conference Room 1"},
                    {"11:00 AM", "Client Call", "Zoom"},
                    {"1:00 PM", "Sprint Review", "Conference Room 2"},
                    {"3:00 PM", "Team Brainstorming", "Conference Room 1"},
                    {"5:00 PM", "End of Day Wrap-Up", "Zoom"}},

            // Thursday - Unique schedule
            {{"9:00 AM", "Team Standup Meeting", "Conference Room 1"},
                    {"11:00 AM", "Client Call", "Zoom"},
                    {"1:00 PM", "Project Kickoff", "Conference Room 2"},
                    {"3:00 PM", "Quarterly Strategy Meeting", "Conference Room 1"},
                    {"5:00 PM", "End of Day Wrap-Up", "Zoom"}},

            // Friday - Unique schedule
            {{"9:00 AM", "Team Standup Meeting", "Conference Room 1"},
                    {"11:00 AM", "Client Call", "Zoom"},
                    {"1:00 PM", "Project Review", "Conference Room 2"},
                    {"3:00 PM", "Team Social", "Lounge"},
                    {"5:00 PM", "End of Week Review", "Zoom"}},
```

```
// Saturday - Same as before
        {{"10:00 AM", "Weekend Planning", "Conference Room 1"},
            {"2:00 PM", "Team Brainstorming", "Conference Room 2"}},

        // Sunday - Same as before
        {{"11:00 AM", "Weekly Review", "Zoom"},
            {"4:00 PM", "Team Social", "Lounge"}}
    };

    private JLabel dateTimeLabel;
    private JPanel timetablePanel;

    public Main() {
        // Setup JFrame properties
        setTitle("One-Week Timetable");
        setSize(800, 600); // Size of the window
        setLocationRelativeTo(null); // Center the window
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());  // Use BorderLayout for organizing the components

        // Create JLabel for displaying the date and time
        dateTimeLabel = new JLabel();
        dateTimeLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Set font and size
        dateTimeLabel.setForeground(Color.BLACK); // Set text color
        add(dateTimeLabel, BorderLayout.NORTH);  // Add to the top of the window

        // Create a JPanel to hold the timetable
        timetablePanel = new JPanel();
        timetablePanel.setLayout(new GridLayout(8, 3)); // Grid layout for the 7 days + headers
        add(timetablePanel, BorderLayout.CENTER);

        // Add column headers for Time, Meeting, and Location
        timetablePanel.add(new JLabel("Time", JLabel.CENTER));
        timetablePanel.add(new JLabel("Meeting", JLabel.CENTER));
        timetablePanel.add(new JLabel("Location", JLabel.CENTER));
```

```
// Add column headers for Time, Meeting, and Location
        timetablePanel.add(new JLabel("Time", JLabel.CENTER));
        timetablePanel.add(new JLabel("Meeting", JLabel.CENTER));
        timetablePanel.add(new JLabel("Location", JLabel.CENTER));

        // Add navigation buttons for each day of the week
        JPanel buttonPanel = new JPanel(new FlowLayout());

        String[] daysOfWeek = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"};

        for (int i = 0; i < daysOfWeek.length; i++) {
            JButton dayButton = new JButton(daysOfWeek[i]);
            final int index = i;  // Capture the index for this button

            dayButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    // When a day button is clicked, update the timetable for that day
                    updateTimetable(index);
                }
            });
  buttonPanel.add(dayButton);
        }

        add(buttonPanel, BorderLayout.SOUTH);

        // Timer to update the date and time label
        Timer dateTimeTimer = new Timer(1000, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Get current date and time
                SimpleDateFormat dateFormat = new SimpleDateFormat("EEEE, dd MMM yyyy HH:mm:ss");
                String currentDateTime = dateFormat.format(new Date());
                dateTimeLabel.setText("Current Date and Time: " + currentDateTime);
            }
        });dateTimeTimer.start();  // Start the date and time update timer

        // Initially show Monday's timetable
        updateTimetable(0); // 0 is for Monday
    }

    // Method to update the timetable based on the selected day index
    private void updateTimetable(int dayIndex) {
        // Clear the previous timetable entries
        timetablePanel.removeAll();
```

```
// Add column headers
        timetablePanel.add(new JLabel("Time", JLabel.CENTER));
        timetablePanel.add(new JLabel("Meeting", JLabel.CENTER));
        timetablePanel.add(new JLabel("Location", JLabel.CENTER));

        // Get the meetings for the selected day
        String day = getDayName(dayIndex);

        // Add the meeting details for the selected day
        for (String[] meeting : WEEK_MEETINGS[dayIndex]) {
            timetablePanel.add(new JLabel(meeting[0], JLabel.CENTER)); // Time
            timetablePanel.add(new JLabel(meeting[1], JLabel.CENTER)); // Meeting
            timetablePanel.add(new JLabel(meeting[2], JLabel.CENTER));  // Location
        }

        // Revalidate and repaint to update the display
        timetablePanel.revalidate();
        timetablePanel.repaint();
    }

    // Helper method to get the day name from index
    private String getDayName(int index) {
        String[] daysOfWeek = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"};
        return daysOfWeek[index];
    }

    public static void main(String[] args) {
        // Run the meeting timetable application
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Main().setVisible(true);
             }
         });
    }
}
```

APPENDIX B(SCREENSHOTS)