Prepared by

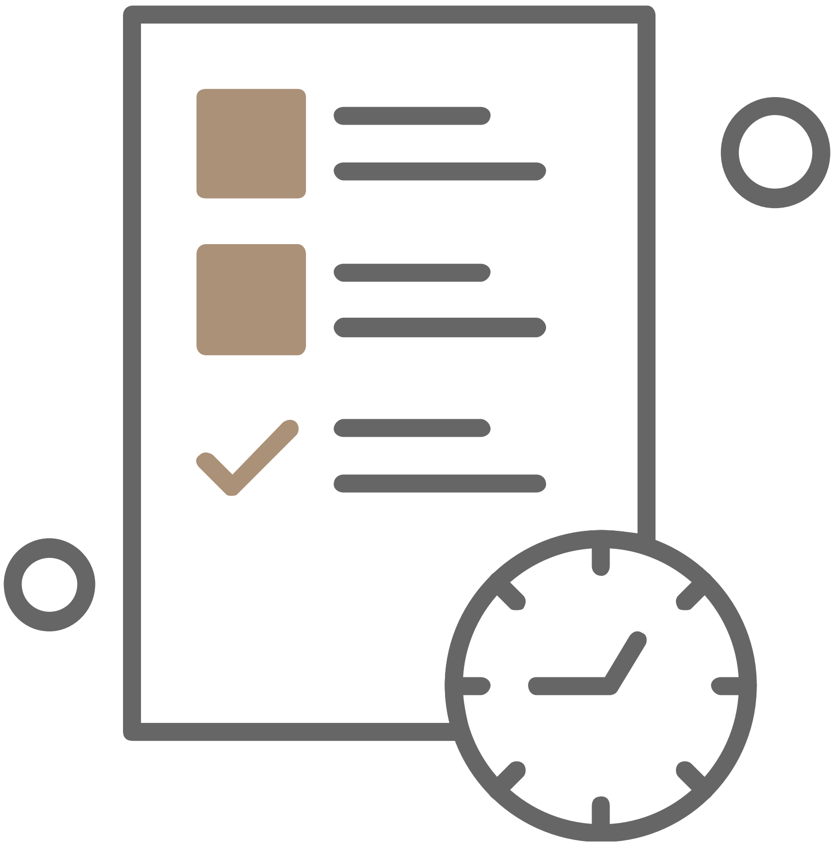
Aashna Bansal 22csu006
Mehak Kaur 22csu117
Shefali Khera 22csu426
Shreya 22csu515

Under the supervision of

Dr. Anuradha Dhull

TIME TABLE SCHEDULING PROJECT REPORT

INDEX



INTRODUCTION

The Timetable Scheduler, a Java-based program, has been meticulously developed to address the intricate task of scheduling and editing class timetables within an educational context. Grounded in fundamental principles of Data Structures and Algorithms (DSA), this software solution seeks to optimize the allocation of classes to various time slots and days, thereby enhancing the overall efficiency of timetable management. The intricacies inherent in this process are managed through a carefully designed user interface, providing a seamless experience for users to dynamically schedule and edit class timetables for a predefined set of subjects. The program accommodates a spectrum of constraints, including the classification of classes and the availability of time slots. Its architecture embodies a systematic approach, employing DSA principles to navigate the complexities associated with educational timetable management, offering a refined and streamlined solution for scheduling challenges.

ABOUT SYSTEM

Talking about the Timetable Scheduler in Java, this system is simple and very much user-friendly. Here the anchor of the system is the user. He can perform CRUD operations. Also, you can use this to generate weekly timetable and even edit it according to your preferences. With this project, you can easily manage your classes.

The design of this is so simple that the user won't find difficulties while working on it. This project is easy to operate and understood by the users. To run this project you must have installed Eclipse IDE or Netbeans IDE or Visual Studio on your PC.

KEY FEATURES

1. Dynamic Scheduling Algorithm:
The core functionality of the scheduler revolves around a dynamic scheduling algorithm. It iterates through a set of subjects, evaluates constraints, and efficiently allocates classes to available time slots.

2. Error Handling Mechanisms:
Robust error handling mechanisms are integrated into the scheduler to manage diverse scenarios. This includes checks for invalid inputs, attempts to schedule labs or tutorials for subjects already assigned, and more.

3. Interactive Editing Capabilities:
Users can interactively edit the timetable by specifying the day and slot they intend to modify. The program intelligently checks the feasibility of the edit, ensuring that no edits occur in between lab or tutorial slots.

4. Constraints Management:
The scheduler enforces constraints such as preventing the scheduling of additional classes if a lab or tutorial is already assigned for the same subject on any given day. This enhances the integrity of the timetable.

5. User-Friendly Interface:
The program offers a user-friendly console interface, guiding users through the scheduling and editing processes. This ensures a seamless and intuitive user experience.

6. Code Organization:
The code follows a structured and modular approach, utilizing methods for different functionalities, enhancing readability and maintainability.

7. Scheduling Logic:
The scheduleClasses method efficiently allocates classes to available slots, considering constraints such as one class per subject per day and different slot requirements for lectures, labs, and tutorials.

8. Conflict Handling:
The program checks for existing labs or tutorials for a subject before scheduling a new one to avoid conflicts.

CODE STRUCTURE

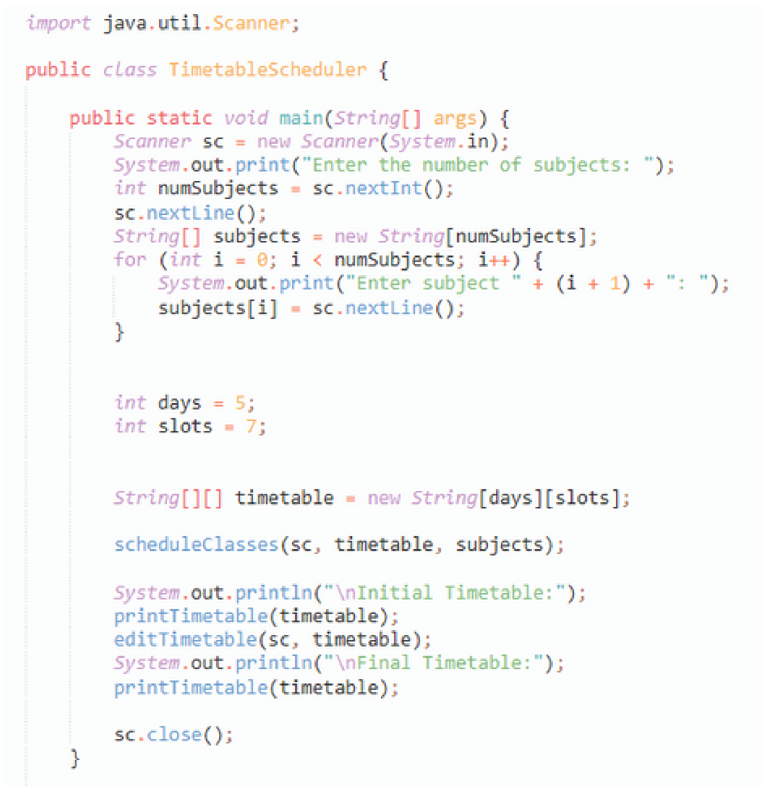
Scheduling Logic:
The central scheduling logic is encapsulated in the "scheduleClasses" function. It systematically processes subjects, adheres to constraints, and allocates classes to suitable time slots.

Error Handling:
The "isScheduled" function is dedicated to error handling, specifically checking for existing labs or tutorials scheduled for the same subject on any day. This serves to preempt scheduling conflicts

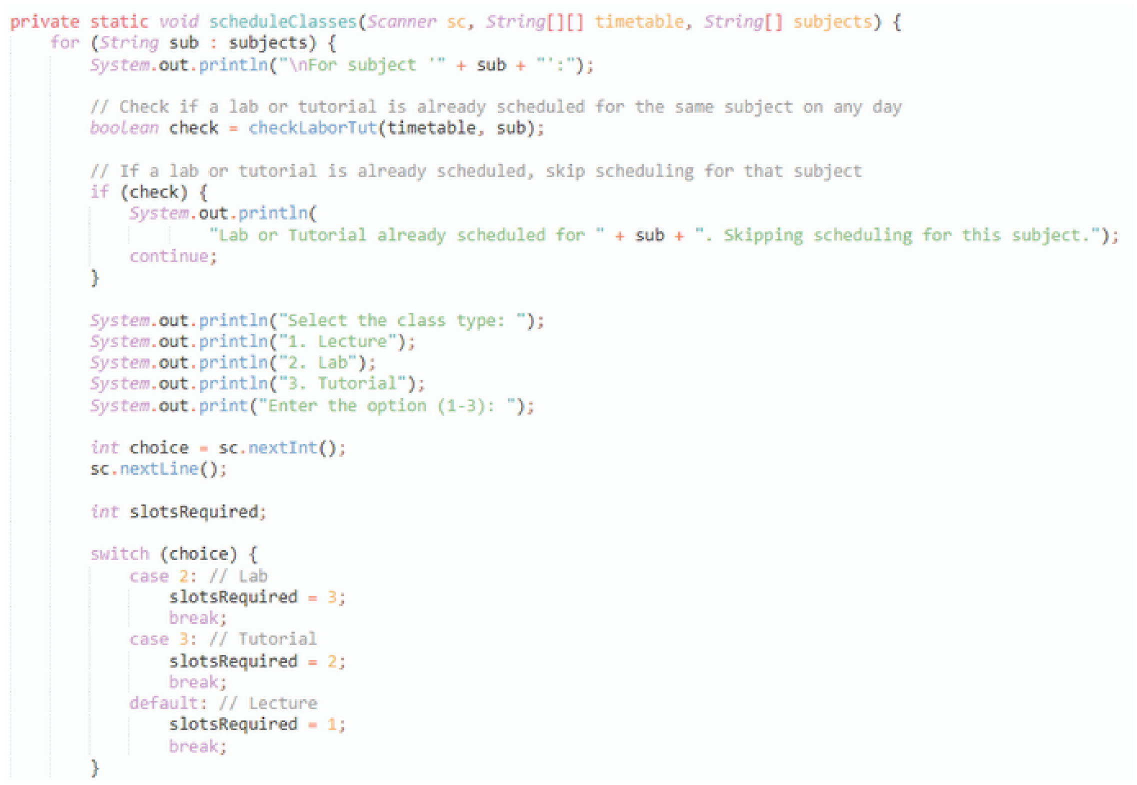
Interactive Editing:
The "editTimetable" function provides an interactive platform for timetable editing. It enforces constraints, preventing disruptions in lab or tutorial slots, and validates subject scheduling on the same day.

Utility Functions:
Utility functions like "getDay" and "getDayName" enhance code clarity by facilitating seamless conversion between day names and indices.

CODE

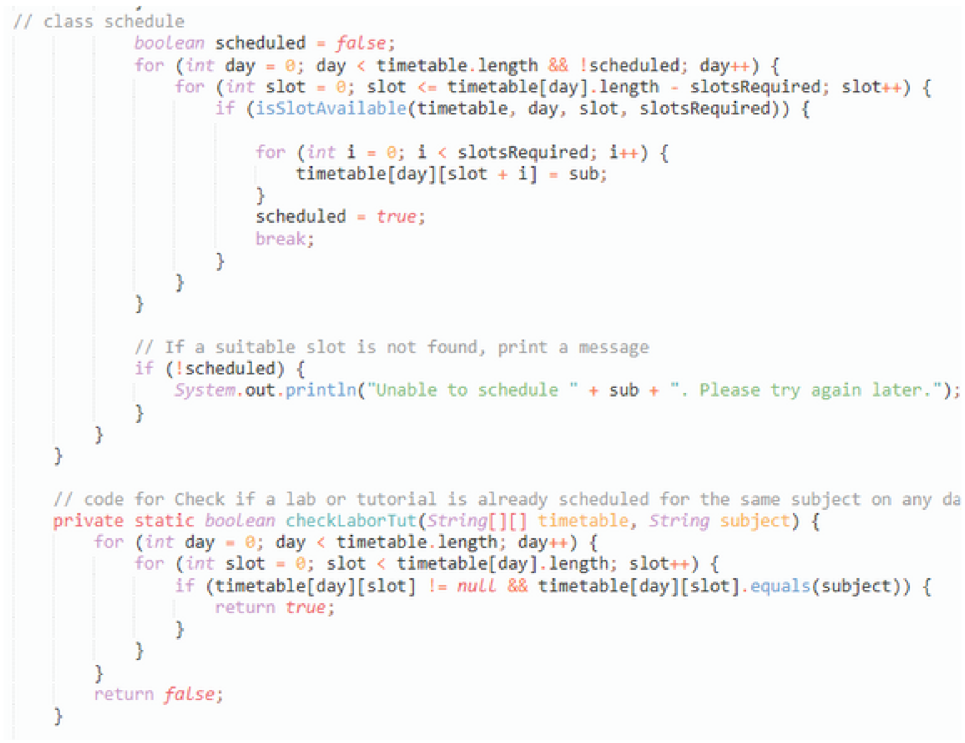


```java
import java.util.Scanner;

public class TimetableScheduler {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of subjects: ");
        int numSubjects = sc.nextInt();
        sc.nextLine();
        String[] subjects = new String[numSubjects];
        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter subject " + (i + 1) + ": ");
            subjects[i] = sc.nextLine();
        }


        int days = 5;
        int slots = 7;


        String[][] timetable = new String[days][slots];

        scheduleClasses(sc, timetable, subjects);

        System.out.println("\nInitial Timetable:");
        printTimetable(timetable);
        editTimetable(sc, timetable);
        System.out.println("\nFinal Timetable:");
        printTimetable(timetable);

        sc.close();
    }
```

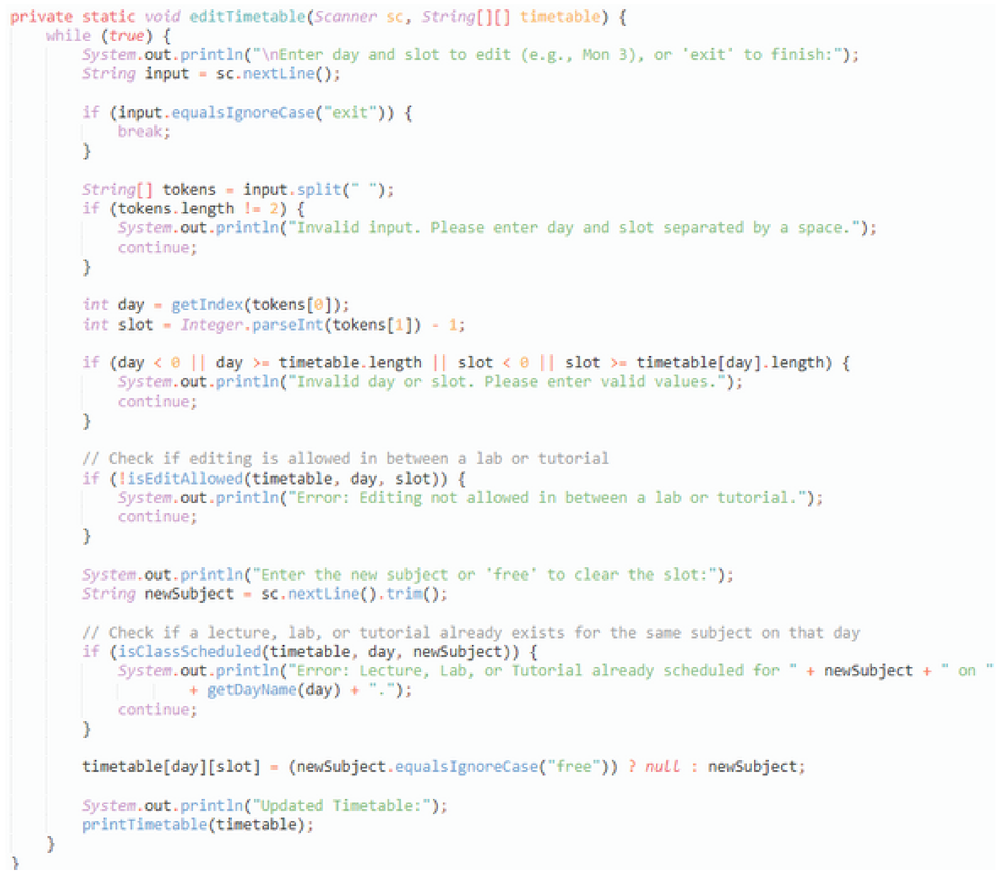


```java
private static void scheduleClasses(Scanner sc, String[][] timetable, String[] subjects) {
    for (String sub : subjects) {
        System.out.println("\nFor subject '" + sub + "':");

        // Check if a lab or tutorial is already scheduled for the same subject on any day
        boolean check = checkLaborTut(timetable, sub);

        // If a lab or tutorial is already scheduled, skip scheduling for that subject
        if (check) {
            System.out.println(
                    "Lab or Tutorial already scheduled for " + sub + ". Skipping scheduling for this subject.");
            continue;
        }

        System.out.println("Select the class type: ");
        System.out.println("1. Lecture");
        System.out.println("2. Lab");
        System.out.println("3. Tutorial");
        System.out.print("Enter the option (1-3): ");

        int choice = sc.nextInt();
        sc.nextLine();

        int slotsRequired;

        switch (choice) {
            case 2: // Lab
                slotsRequired = 3;
                break;
            case 3: // Tutorial
                slotsRequired = 2;
                break;
            default: // Lecture
                slotsRequired = 1;
                break;
        }
```

CODE

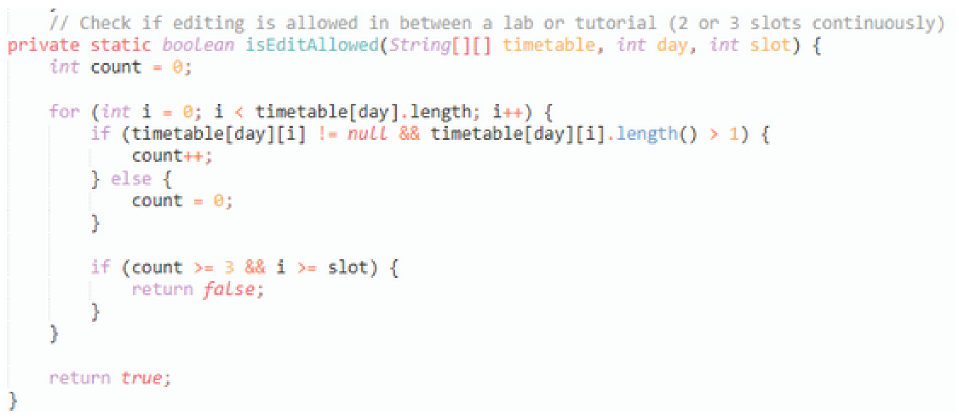


```java
// class schedule
            boolean scheduled = false;
            for (int day = 0; day < timetable.length && !scheduled; day++) {
                for (int slot = 0; slot <= timetable[day].length - slotsRequired; slot++) {
                    if (isSlotAvailable(timetable, day, slot, slotsRequired)) {

                        for (int i = 0; i < slotsRequired; i++) {
                            timetable[day][slot + i] = sub;
                        }
                        scheduled = true;
                        break;
                    }
                }
            }

            // If a suitable slot is not found, print a message
            if (!scheduled) {
                System.out.println("Unable to schedule " + sub + ". Please try again later.");
            }
        }
    }

    // code for Check if a lab or tutorial is already scheduled for the same subject on any da
    private static boolean checkLaborTut(String[][] timetable, String subject) {
        for (int day = 0; day < timetable.length; day++) {
            for (int slot = 0; slot < timetable[day].length; slot++) {
                if (timetable[day][slot] != null && timetable[day][slot].equals(subject)) {
                    return true;
                }
            }
        }
        return false;
    }
```

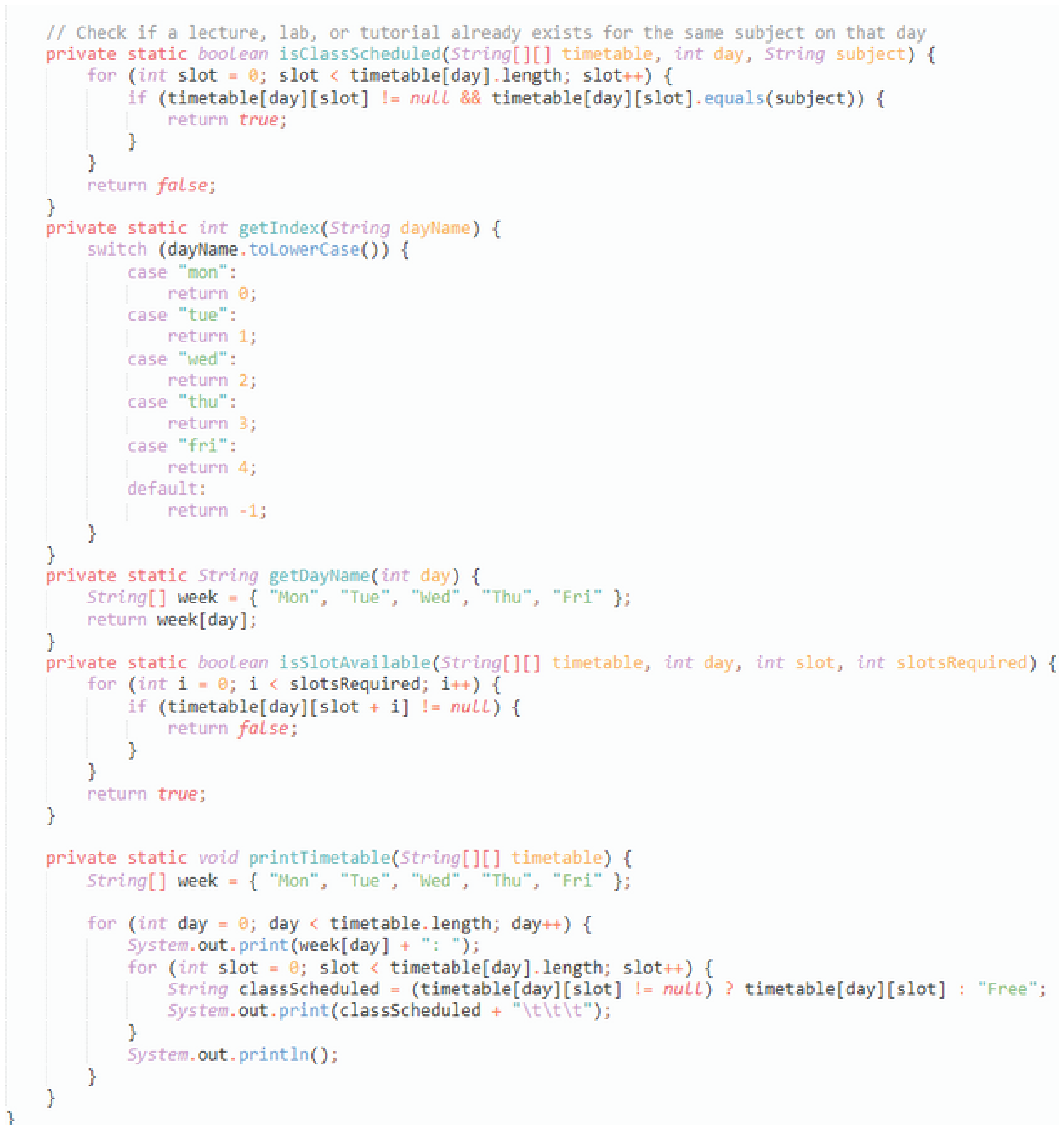


```java
private static void editTimetable(Scanner sc, String[][] timetable) {
    while (true) {
        System.out.println("\nEnter day and slot to edit (e.g., Mon 3), or 'exit' to finish:");
        String input = sc.nextLine();

        if (input.equalsIgnoreCase("exit")) {
            break;
        }

        String[] tokens = input.split(" ");
        if (tokens.length != 2) {
            System.out.println("Invalid input. Please enter day and slot separated by a space.");
            continue;
        }

        int day = getIndex(tokens[0]);
        int slot = Integer.parseInt(tokens[1]) - 1;

        if (day < 0 || day >= timetable.length || slot < 0 || slot >= timetable[day].length) {
            System.out.println("Invalid day or slot. Please enter valid values.");
            continue;
        }

        // Check if editing is allowed in between a lab or tutorial
        if (!isEditAllowed(timetable, day, slot)) {
            System.out.println("Error: Editing not allowed in between a lab or tutorial.");
            continue;
        }

        System.out.println("Enter the new subject or 'free' to clear the slot:");
        String newSubject = sc.nextLine().trim();

        // Check if a lecture, lab, or tutorial already exists for the same subject on that day
        if (isClassScheduled(timetable, day, newSubject)) {
            System.out.println("Error: Lecture, Lab, or Tutorial already scheduled for " + newSubject + " on "
                    + getDayName(day) + ".");
            continue;
        }

        timetable[day][slot] = (newSubject.equalsIgnoreCase("free")) ? null : newSubject;

        System.out.println("Updated Timetable:");
        printTimetable(timetable);
    }
}
```

CODE

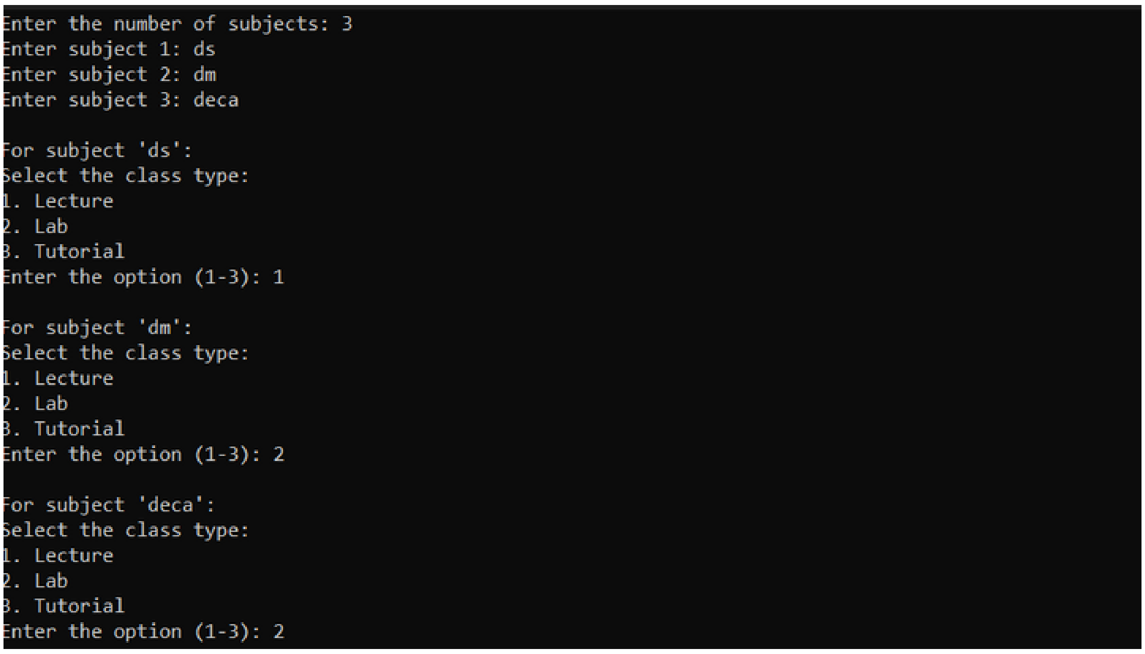


```java
    // Check if editing is allowed in between a lab or tutorial (2 or 3 slots continuously)
private static boolean isEditAllowed(String[][] timetable, int day, int slot) {
    int count = 0;

    for (int i = 0; i < timetable[day].length; i++) {
        if (timetable[day][i] != null && timetable[day][i].length() > 1) {
            count++;
        } else {
            count = 0;
        }

        if (count >= 3 && i >= slot) {
            return false;
        }
    }

    return true;
}
```

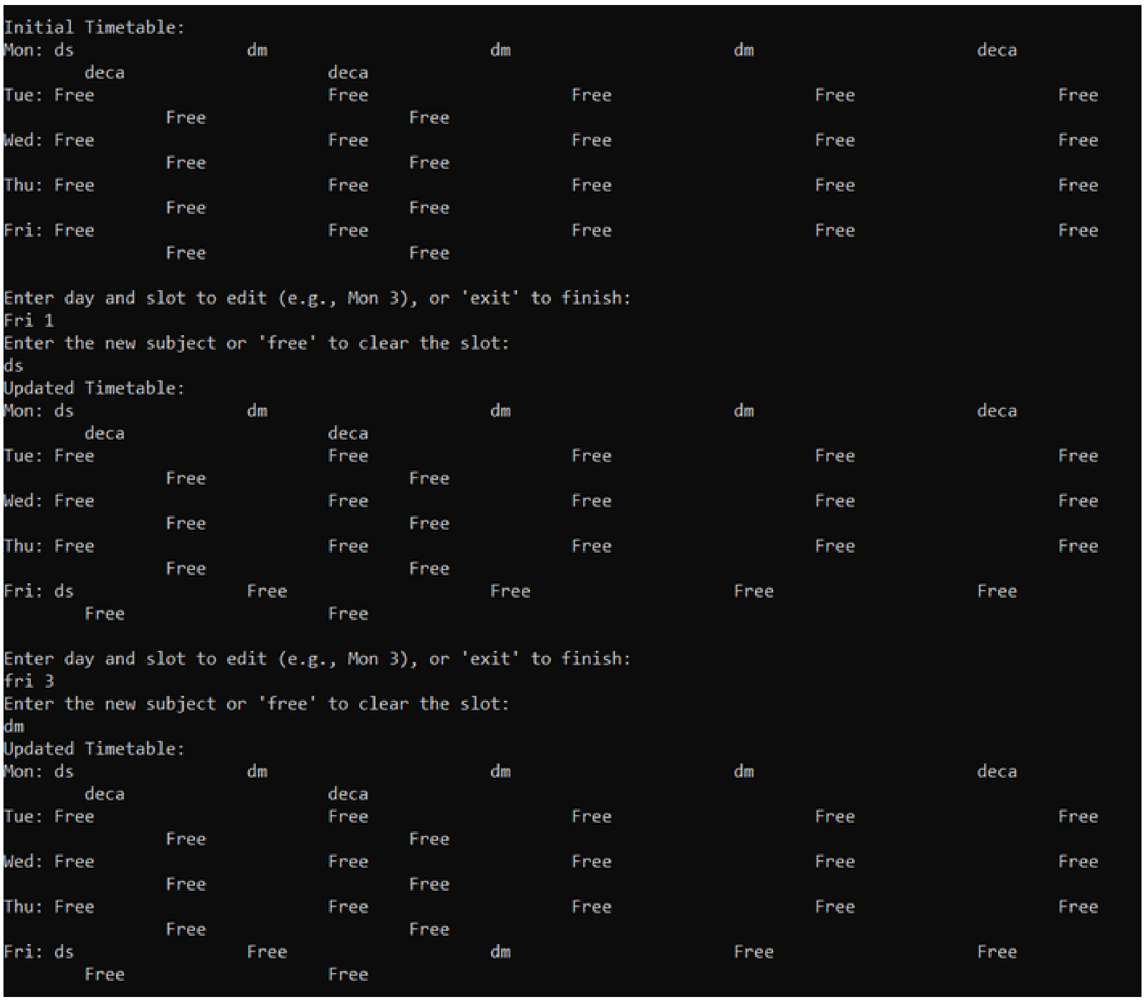


```java
    // Check if a lecture, lab, or tutorial already exists for the same subject on that day
    private static boolean isClassScheduled(String[][] timetable, int day, String subject) {
        for (int slot = 0; slot < timetable[day].length; slot++) {
            if (timetable[day][slot] != null && timetable[day][slot].equals(subject)) {
                return true;
            }
        }
        return false;
    }
    private static int getIndex(String dayName) {
        switch (dayName.toLowerCase()) {
            case "mon":
                return 0;
            case "tue":
                return 1;
            case "wed":
                return 2;
            case "thu":
                return 3;
            case "fri":
                return 4;
            default:
                return -1;
        }
    }
    private static String getDayName(int day) {
        String[] week = { "Mon", "Tue", "Wed", "Thu", "Fri" };
        return week[day];
    }
    private static boolean isSlotAvailable(String[][] timetable, int day, int slot, int slotsRequired) {
        for (int i = 0; i < slotsRequired; i++) {
            if (timetable[day][slot + i] != null) {
                return false;
            }
        }
        return true;
    }

    private static void printTimetable(String[][] timetable) {
        String[] week = { "Mon", "Tue", "Wed", "Thu", "Fri" };

        for (int day = 0; day < timetable.length; day++) {
            System.out.print(week[day] + ": ");
            for (int slot = 0; slot < timetable[day].length; slot++) {
                String classScheduled = (timetable[day][slot] != null) ? timetable[day][slot] : "Free";
                System.out.print(classScheduled + "\t\t\t");
            }
            System.out.println();
        }
    }
}
```

CODE OUTPUT



```
Enter the number of subjects: 3
Enter subject 1: ds
Enter subject 2: dm
Enter subject 3: deca

For subject 'ds':
Select the class type:
1. Lecture
2. Lab
3. Tutorial
Enter the option (1-3): 1

For subject 'dm':
Select the class type:
1. Lecture
2. Lab
3. Tutorial
Enter the option (1-3): 2

For subject 'deca':
Select the class type:
1. Lecture
2. Lab
3. Tutorial
Enter the option (1-3): 2
```



```
Initial Timetable:
Mon: ds                 dm                      dm                      dm                       deca
        deca                    deca
Tue: Free                       Free                    Free                    Free                     Free
                Free                    Free
Wed: Free                       Free                    Free                    Free                     Free
                Free                    Free
Thu: Free                       Free                    Free                    Free                     Free
                Free                    Free
Fri: Free                       Free                    Free                    Free                     Free
                Free                    Free

Enter day and slot to edit (e.g., Mon 3), or 'exit' to finish:
Fri 1
Enter the new subject or 'free' to clear the slot:
ds
Updated Timetable:
Mon: ds                 dm                      dm                      dm                       deca
        deca                    deca
Tue: Free                       Free                    Free                    Free                     Free
                Free                    Free
Wed: Free                       Free                    Free                    Free                     Free
                Free                    Free
Thu: Free                       Free                    Free                    Free                     Free
                Free                    Free
Fri: ds                 Free                    Free                    Free                     Free
        Free                    Free

Enter day and slot to edit (e.g., Mon 3), or 'exit' to finish:
fri 3
Enter the new subject or 'free' to clear the slot:
dm
Updated Timetable:
Mon: ds                 dm                      dm                      dm                       deca
        deca                    deca
Tue: Free                       Free                    Free                    Free                     Free
                Free                    Free
Wed: Free                       Free                    Free                    Free                     Free
                Free                    Free
Thu: Free                       Free                    Free                    Free                     Free
                Free                    Free
Fri: ds                 Free                    dm                      Free                     Free
        Free                    Free
```

CONCLUSION

The Timetable Scheduler project serves as a pragmatic manifestation of Data Structures and Algorithms (DSA) principles within a real-world context. The program adeptly incorporates sophisticated scheduling algorithms, robust error handling mechanisms, and interfaces designed for user-friendly interactions. As a testament to its adherence to DSA tenets, the project underscores its capacity to address complex scheduling challenges efficiently.

This report furnishes a comprehensive overview delineating the salient features, structural intricacies, and operational functionalities of the Timetable Scheduler project. Emphasizing its fidelity to DSA principles, the report discerns areas conducive to future refinement. Prospective enhancements encompass the incorporation of additional scheduling constraints, the implementation of optimization techniques, and the integration of a Graphical User Interface (GUI) to augment overall usability. This discourse thus provides a nuanced examination of the project's current standing and delineates a trajectory for its prospective evolution.