# □ Assistant Scheduler – C++ Project Documentation

This C++ program is a **personal assistant scheduler system** that helps users add tasks for each day of a month, view saved schedules, and search through tasks. It's a simple command-line based project designed for beginners to understand class structures, file I/O, and basic logic.

---

## □ File Overview

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
#include <fstream>
#include <algorithm>
```

These libraries help with:

- `iostream` : input/output (e.g., `cin` , `cout` )
- `vector` : dynamic arrays (used for storing tasks)
- `string` : to work with text
- `iomanip` : formatting output (e.g., aligning text)
- `fstream` : reading/writing to files
- `algorithm` : converting text to lowercase

---

## □ Class: `TimeBlock`

Represents a **single task** with a time and description.

```
class TimeBlock {
public:
    string time;
    string description;

    TimeBlock(string t, string d) {
        time = t;
        description = d;
    }
};
```

□ **Example:**

```
TimeBlock block("10:00 AM", "Team Meeting");
```

---

## □ Class: `DaySchedule`

Stores all tasks for one day.

### Properties:

- `int day`: The day number (1 to 31)
- `vector<TimeBlock> tasks`: List of tasks for the day

### Methods:

**`addTask(time, desc)`**

Adds a task to that day.

```
schedule.addTask("9:00 AM", "Workout");
```

**`display(month)`**

Prints tasks to the console.

```
schedule.display("April");
```

**`saveToFile(out, month)`**

Saves the day's schedule to a file stream.

```
ofstream out("schedule_log.txt");
schedule.saveToFile(out, "April");
```

**`toLower(str)`**

Converts a string to lowercase for consistent comparison.

```
DaySchedule::toLower("February");  // "february"
```

---

## Function: `runScheduleForMonth()`

Handles:

- Getting the user's month
- Adding tasks
- Viewing and saving the schedule
- Summary of total tasks and active days

### Steps:

1. Ask for the month.
2. Determine number of days based on month.
3. Let user input tasks.
4. Ask if they want to view or save the schedule.
5. Save everything to `schedule_log.txt`.
6. Show summary at the end.

**Sample Input Flow:**

```
Enter month: April
Enter day: 12
Enter time: 3:00 PM
Enter task: Doctor appointment
Add another task? (yes/no): no
```

---

## `main()` : Program Flow

This is the entry point of the program. It gives the user a menu:

**Menu Options:**

```
[1] Add new schedule
[2] View saved schedule
[3] Search your schedule
[4] Exit
```

### `Search`

Search through saved tasks using any keyword or number:

```
Enter keyword or day to search: meeting
```

---

## File: `schedule_log.txt`

All tasks are saved here. If you add tasks for April, the file might include:

```
| SCHEDULE FOR April 12
| Time: 03:00 PM  | Task: Doctor appointment
```