



# Presentation on

Software Development Project

## Calendar App with Task Planner

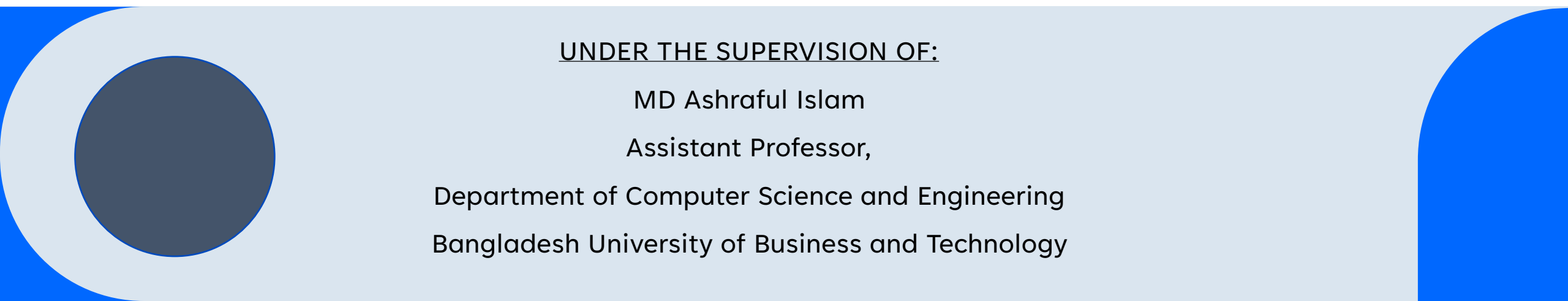
UNDER THE SUPERVISION OF:

MD Ashrafur Islam

Assistant Professor,

Department of Computer Science and Engineering

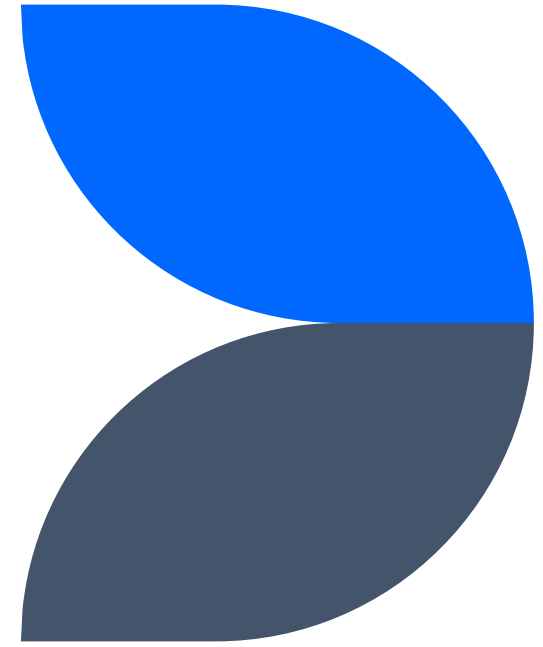
Bangladesh University of Business and Technology



# Group Members

Student Name	Student ID	Intake(Section)
Ferdouse Hassan Nowrin	22234103237	50(06)
Asifur Rahman	22234103353	50(06)
Delwar Hossain Roman	22234103233	50(06)
Tasnimul Haque Nahin	22234103180	50(06)
Mahajabin Kabir Arin	22234103213	50(06)

# Introduction



# About The Project

- The calendar project is a comprehensive solution for managing tasks and events efficiently.
- It provides users with a user-friendly interface to organize their schedules effectively.



# Why a calendar application?

- In our fast-paced lives, staying organized and managing time is crucial.
- A calendar application serves as a valuable tool for individuals, professionals, and organizations alike.
- The importance of managing tasks and events efficiently is unavoidable in these days. A Calendar App come useful in this cases.



# Implementation in C Programming

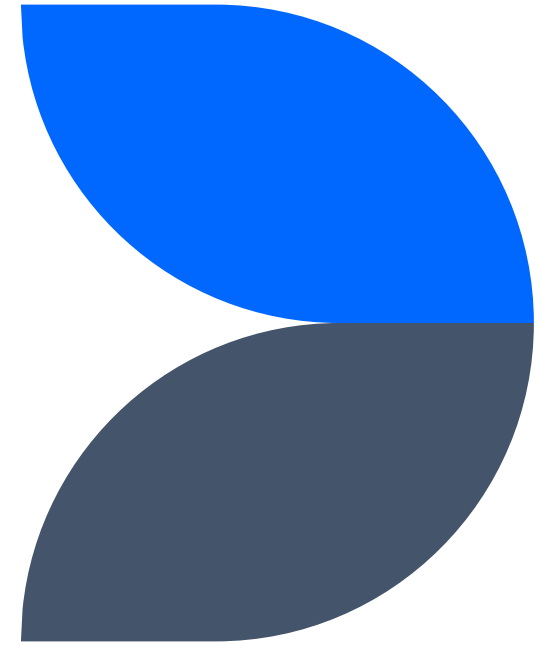
- The calendar project is implemented using the C programming language, renowned for its efficiency and versatility.
- C allows us to create a robust and high-performance application to meet the demands of managing complex calendars.



# Software & Tools Used in the Project

- Code::Blocks
- Windows Terminal
- GitHub

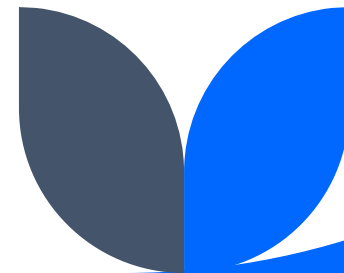
# Objectives





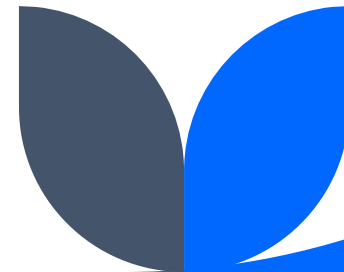
# What are the main objects?

- The main objective of our project is to create a calendar application that simplifies the process of managing tasks and events.
- With our application, users can conveniently add, modify, and delete tasks, ensuring nothing falls through the cracks.

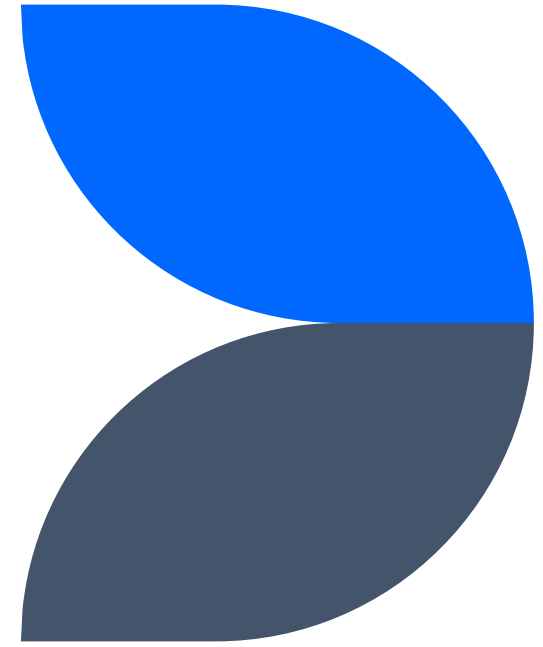


# Benefits of the Calendar Project

- Increased productivity: Users can track deadlines, appointments, and important events in one app.
- Time management: The application helps individuals allocate their time effectively.
- Data persistence: Users can save their tasks to a file, preventing any loss of important information.



# Project Structure



# Before Diving into Project Structure!

The Source code of our project is available in our GitHub public repository.

Anyone can see it and use it for educational purpose.

Click on this text for the repository :

[Software-Development-Project-1](#)

# The “Month” Struct

- The core structure of the project is based on the Month struct, which represents a specific month of the calendar.
- The Month struct encapsulates essential information about a month, including the year, month number, and an array to store tasks for each day.
- This struct serves as the foundation for creating and managing the calendar.



# Components for Managing Tasks

- The project includes several functions designed for managing tasks and events within the application.
- These functions enable users to add, modify, delete, view, and search for tasks.
- The task management components ensure efficient organization and retrieval of task-related information.



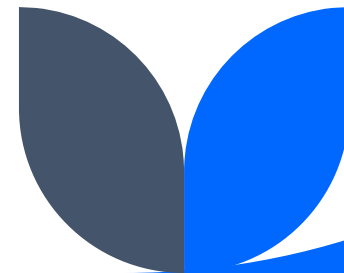
# User Interaction

- The calendar application interacts with users through a console interface.
- Users are prompted for various inputs, such as selecting options, entering dates, and providing task details.
- The user interaction components facilitate seamless communication between the user and the calendar application.



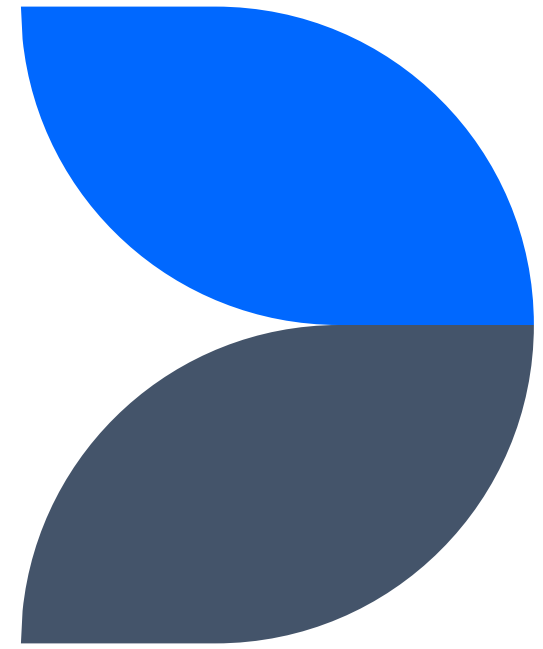
# File Handling

- File handling capabilities are implemented to provide data storing for tasks.
- Users can save their tasks to a file and load tasks from a file when restarting the application.
- File handling components facilitate the storage of task data, ensuring seamless continuity of the calendar application.





# Features and Functionality

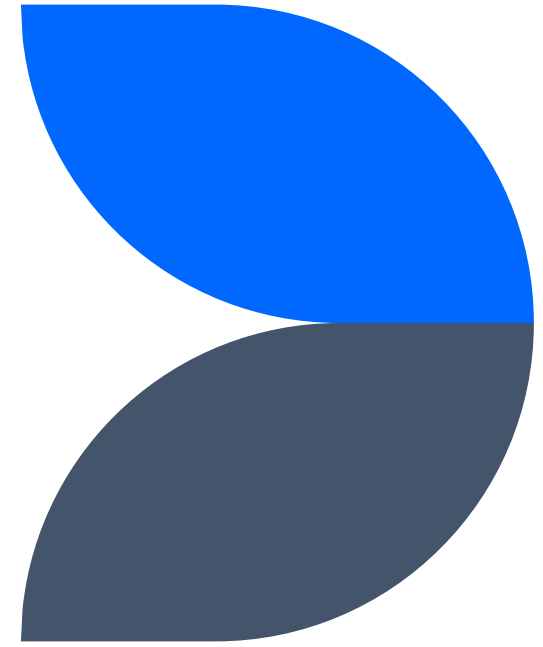


# Key features and functionality of the calendar project

- Generate a monthly calendar for any specified year and month, providing a visual representation of the days.
- Allow users to add, modify, and delete tasks or events on specific days.
- Provide the ability to view tasks for a particular day and search for tasks based on a description.
- Enable users to save their tasks to a file and load tasks from a file for data persistence.

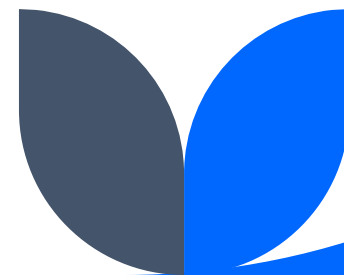


# User Interaction



# Prompts for Input

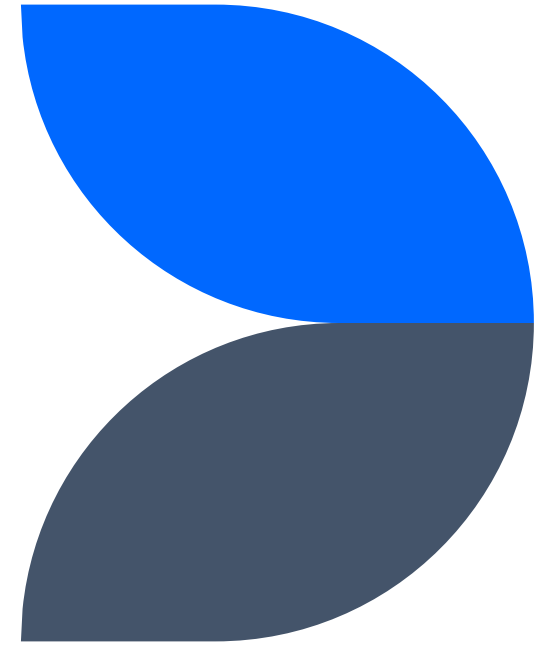
- Users are prompted for various inputs throughout the application.
- They are asked to enter specific information such as selecting options, entering dates, and task details.
- These prompts ensure that the application captures the necessary information from the user to perform the desired actions effectively.



# Available Options for Tasks

- Adding a Task
- Modifying a Task
- Deleting a Task
- Viewing Tasks
- Searching for Tasks

# Task Management

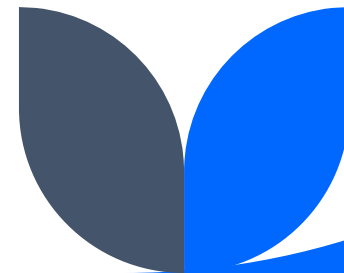


# Task Management is Complex!

In the calendar project, task management is a crucial aspect that allows users to add, modify, and delete tasks or events. The project employs a structured approach to handle tasks effectively.

# Task Storage in a 2D Array

Tasks are stored in a 2D array within the Month struct. Each element of the 2D array represents a specific day of the month. This data structure enables efficient organization and retrieval of tasks based on their corresponding dates.

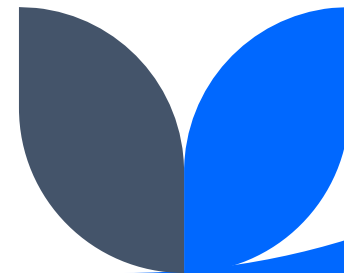




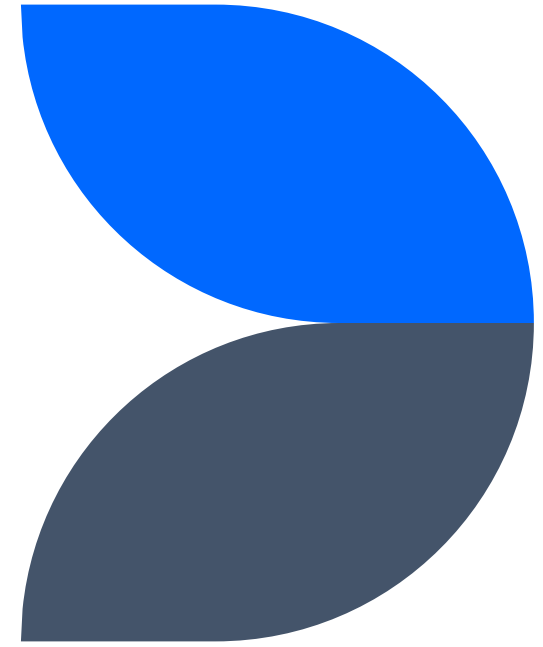
# String Manipulation Functions

To enable task management functionalities, the project employs string manipulation functions. The string manipulation operations in the project include:

- Adding Tasks
- Modifying Tasks
- Deleting Tasks
- Searching and Sorting



# File Handling



# Why File Handling is so Important?

File handling plays a crucial role in the calendar project, providing data persistence for tasks and ensuring that users can save and retrieve their data for future reference

Without file handling, users would lose their task data every time they close or restart the application.

# Saving Mechanism

- Users can save their tasks to a file for future reference.
- When the user chooses to save their tasks, the application creates a file and writes the task data to it.
- This mechanism allows users to retain their tasks even after closing the application.



# Loading Mechanism

- Users can load tasks from a file when restarting the application.
- The loading mechanism reads the task data from the file.
- This allows users to resume their task management seamlessly without losing any previously saved tasks.



# Flexibility and Portability

- By saving tasks to a file, users gain the flexibility to access their task data across different instances of the application or on different systems.
- They can easily transfer the file to another device or share it with others, ensuring portability and convenience.

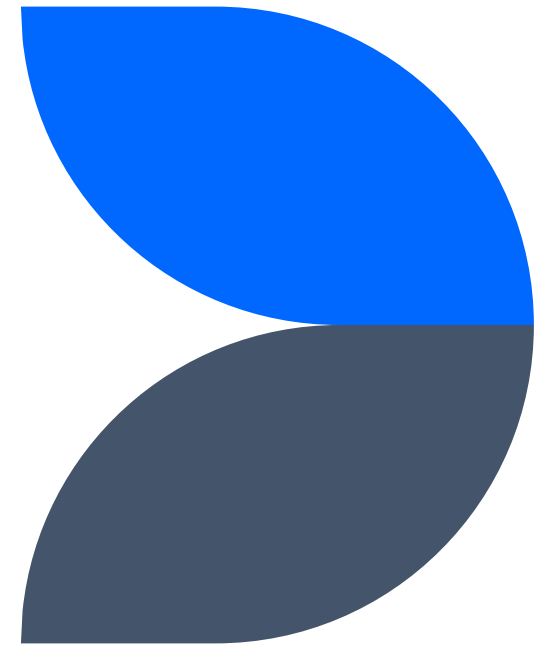


# Error Handling

- The file handling mechanisms in the calendar project incorporate error handling to ensure data integrity.
- Errors such as file not found, read/write failures, or corrupted data are handled gracefully, providing a robust user experience.



# Demonstration of The Project





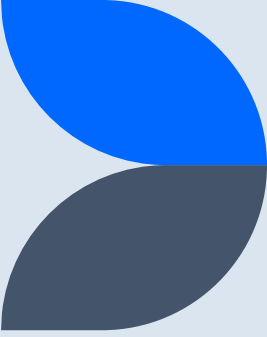
# Creating Calendars

```
Enter year: 2023
Enter month (1-12): 1
```

```
    1/2023
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

```
Enter tasks (press enter to finish):
Enter day to add a task (0 to finish):
```

# Code Snippets of Creating Calendar



```
struct Month init_month(int year, int month) {
    struct Month m;
    m.year = year;
    m.month = month;

    if (month == 2) {
        m.num_days = is_leap_year(year) ? 29 : 28;
    } else if (month == 4 || month == 6 || month == 9 || month == 11) {
        m.num_days = 30;
    } else {
        m.num_days = 31;
    }

    int y = year - (14 - month) / 12;
    int m1 = month + 12 * ((14 - month) / 12) - 2;
    m.start_day = (1 + y + y / 4 - y / 100 + y / 400 + (31 * m1) / 12) % 7;

    for (int i = 0; i < m.num_days; i++) {
        m.tasks[i][0] = '\\0';
    }

    return m;
}
```

```
void print_month(struct Month m) {
    printf("\\n\\n   %d/%d\\n", m.month, m.year);
    printf("Su Mo Tu We Th Fr Sa\\n");

    for (int i = 0; i < m.start_day; i++) {
        printf("   ");
    }
    for (int i = 1; i <= m.num_days; i++) {
        printf("%2d ", i);
        if ((i + m.start_day) % 7 == 0) {
            printf("\\n");
        }
    }
}
```

# Adding Tasks

```
Enter year: 2023
Enter month (1-12): 1
```

```
    1/2023
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
Enter tasks (press enter to finish):
Enter day to add a task (0 to finish): 6
Enter task description: Study CSE 100
Enter day to add a task (0 to finish): 11
Enter task description: MAT 111 Final Exam
Enter day to add a task (0 to finish): 0
```

```
    1/2023
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
Enter day to view tasks (1-31, 0 to exit): 6
Tasks for Day 6: Study CSE 100
Select an option:
1. Modify task
2. Delete task
3. Change task date
0. Exit
Enter option: 0

Enter day to view tasks (1-31, 0 to exit): 3
No tasks found for Day 3
Select an option:
1. Add task on this day
0. Exit
```

# Code Snippets of Adding Tasks

```
void add_task(struct Month* m, int day, const char* task) {
    if (day < 1 || day > m->num_days) {
        printf("Invalid day! Task not added.\n");
        return;
    }

    strncpy(m->tasks[day - 1], task, MAX_TASK_LENGTH - 1);
    m->tasks[day - 1][MAX_TASK_LENGTH - 1] = '\0';
}
```

# Modifying Tasks

```
Enter day to view tasks (1-31, 0 to exit): 11
Tasks for Day 11: MAT 111 Final Exam
Select an option:
1. Modify task
2. Delete task
3. Change task date
0. Exit
Enter option: 1
Current task for Day 11: MAT 111 Final Exam
Enter new task description: EEE 211 Final Exam
Task modified successfully!

Enter day to view tasks (1-31, 0 to exit): 6
Tasks for Day 6: Study CSE 100
Select an option:
1. Modify task
2. Delete task
3. Change task date
0. Exit
Enter option: 3
Enter new day for the task: 12
Task date changed successfully!
```

# Code Snippets of Modifying Tasks

```
void modify_task(struct Month* m, int day) {
    if (day < 1 || day > m->num_days) {
        printf("Invalid day!\n");
        return;
    }

    int index = day - 1;
    if (m->tasks[index][0] != '\0') {
        printf("Current task for Day %d: %s\n", day, m->tasks[index]);
        printf("Enter new task description: ");
        char task[MAX_TASK_LENGTH];
        scanf(" %[^\\n]", task);
        strncpy(m->tasks[index], task, MAX_TASK_LENGTH - 1);
        m->tasks[index][MAX_TASK_LENGTH - 1] = '\0';
        printf("Task modified successfully!\n");
    } else {
        printf("No tasks found for Day %d\n", day);
    }
}
```

# Deleting a Task

```
Enter day to view tasks (1-31, 0 to exit): 12
Tasks for Day 12: Study CSE 100
Select an option:
1. Modify task
2. Delete task
3. Change task date
0. Exit
Enter option: 2
Deleting task for Day 12: Study CSE 100
Task deleted successfully!
```

# Code Snippets of Deleting a Task

```
void delete_task(struct Month* m, int day) {  
    if (day < 1 || day > m->num_days) {  
        printf("Invalid day!\n");  
        return;  
    }  
  
    int index = day - 1;  
    if (m->tasks[index][0] != '\0') {  
        printf("Deleting task for Day %d: %s\n", day, m->tasks[index]);  
        m->tasks[index][0] = '\0';  
        printf("Task deleted successfully!\n");  
    } else {  
        printf("No tasks found for Day %d\n", day);  
    }  
}
```



# Searching for Tasks

```
Do you want to search for tasks? (y/n): y
Enter task description to search: EEE 211 Final Exam
Tasks matching the description 'EEE 211 Final Exam':
Day 11: EEE 211 Final Exam
Select an option:
1. Change task date
0. Exit
```

```
Do you want to search for more tasks? (y/n): y
Enter task description to search: CSE 121
Tasks matching the description 'CSE 121':
No tasks found.
Select an option:
1. Add task with this description
0. Exit
Enter option: 1
Enter day to add the task: 19
Task added successfully!
```

# Code Snippets of Searching for Tasks

```
void search_tasks(struct Month* m, const char* task_description) {
    printf("Tasks matching the description '%s':\n", task_description);
    bool found = false;

    for (int i = 0; i < m->num_days; i++) {
        if (strcmp(m->tasks[i], task_description) == 0) {
            printf("Day %d: %s\n", i + 1, m->tasks[i]);
            found = true;
        }
    }

    if (!found) {
        printf("No tasks found.\n");
        printf("Select an option:\n");
        printf("1. Add task with this description\n");
        printf("0. Exit\n");
        printf("Enter option: ");
        int option;
        scanf("%d", &option);

        switch (option) {
            case 0:
                break;
            case 1:
                printf("Enter day to add the task: ");
                int day;
                scanf("%d", &day);
                if (day < 1 || day > m->num_days) {
                    printf("Invalid day!\n");
                } else {
                    add_task(m, day, task_description);
                    printf("Task added successfully!\n");
                }
                break;
            default:
                printf("Invalid option!\n");
        }
    }
}
```

```
    } else {
        printf("Select an option:\n");
        printf("1. Change task date\n");
        printf("0. Exit\n");
        printf("Enter option: ");
        int option;
        scanf("%d", &option);

        switch (option) {
            case 0:
                break;
            case 1:
                printf("Enter new day for the task: ");
                int new_day;
                scanf("%d", &new_day);
                if (new_day < 1 || new_day > m->num_days) {
                    printf("Invalid day!\n");
                } else {
                    for (int i = 0; i < m->num_days; i++) {
                        if (strcmp(m->tasks[i], task_description) == 0) {
                            m->tasks[i][0] = '\0';
                            add_task(m, new_day, task_description);
                            printf("Task date changed successfully!\n");
                            break;
                        }
                    }
                    break;
                }
            default:
                printf("Invalid option!\n");
        }
    }
}
```

# Saving a Task using File

```
Do you want to search for more tasks? (y/n): n
Do you want to save tasks? (y/n): y
Enter filename to save tasks: Output1
Tasks saved successfully!
Do you want to load tasks? (y/n): n
Do you want to create tasks for another month? (y/n): n
```

# Code Snippets of Saving Task

```
void save_tasks(struct Month* m, const char* filename) {
    FILE* file = fopen(filename, "w");
    if (file == NULL) {
        printf("Failed to open file for writing.\n");
        return;
    }

    fprintf(file, "%d,%d\n", m->year, m->month);
    for (int i = 0; i < m->num_days; i++) {
        if (m->tasks[i][0] != '\0') {
            fprintf(file, "%d,%s\n", i + 1, m->tasks[i]);
        }
    }

    fclose(file);
    printf("Tasks saved successfully!\n");
}
```

# Loading a Task

```
Do you want to load tasks? (y/n): y
Enter filename to load tasks: Output1
Tasks loaded successfully!
```

```
1/2023
Su Mo Tu We Th Fr Sa
      1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

```
Enter day to view tasks (1-30, 0 to exit): 3
Tasks for Day 3: Project Work
Select an option:
1. Modify task
2. Delete task
3. Change task date
0. Exit
```

# Code Snippets of Loading Task

```
void load_tasks(struct Month* m, const char* filename) {
    FILE* file = fopen(filename, "r");
    if (file == NULL) {
        printf("Failed to open file for reading.\n");
        return;
    }

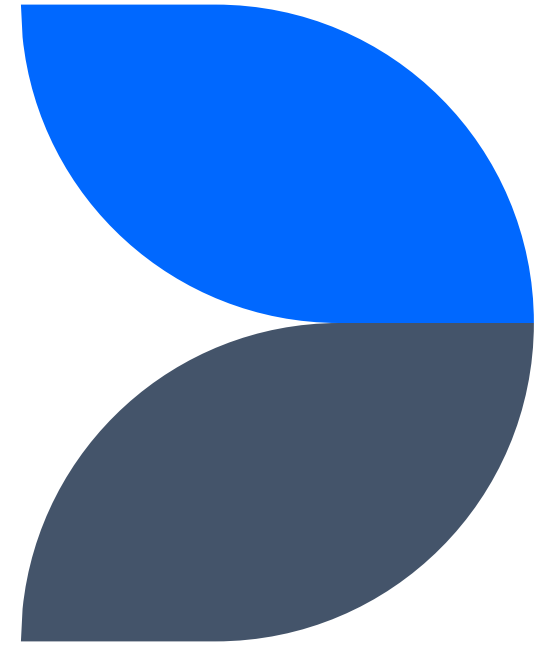
    char line[100];
    if (fgets(line, sizeof(line), file) != NULL) {
        sscanf(line, "%d,%d", &(m->year), &(m->month));
    }

    for (int i = 0; i < m->num_days; i++) {
        m->tasks[i][0] = '\0';
    }

    while (fgets(line, sizeof(line), file) != NULL) {
        int day;
        char task[MAX_TASK_LENGTH];
        sscanf(line, "%d,%[^\\n]", &day, task);
        if (day >= 1 && day <= m->num_days) {
            strncpy(m->tasks[day - 1], task, MAX_TASK_LENGTH - 1);
            m->tasks[day - 1][MAX_TASK_LENGTH - 1] = '\0';
        }
    }

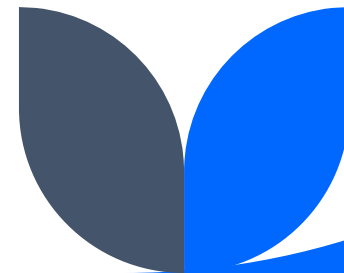
    fclose(file);
    printf("Tasks loaded successfully!\n");
}
```

# Conclusion



# To Summarize

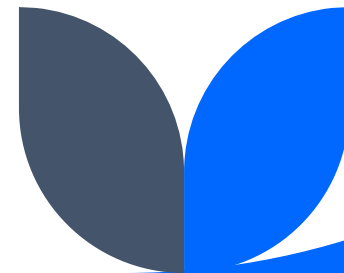
- The calendar application, developed using C programming language, has been successfully implemented.
- The project offers features such as generating monthly calendars, adding, modifying, and deleting tasks or events, and searching for tasks based on descriptions.
- The user-friendly console interface enhances usability and easy interact with the application.





# Any Future Improvements?

- Include incorporating additional features such as reminders, notifications, or recurring events to enhance functionality.
- Enhancing the user interface with graphical elements or developing a graphical user interface (GUI) could further improve usability and appeal.





# **Thank You For Your Patience**

Have A Good Day!