Project Report on

Calendar App with Task Planner

An Undergraduate Project

Submitted in partial fulfillment of the requirements of Software Development Project 1 for the degree of Bachelor of Science in Computer Science and Engineering.

UNDER THE SUPERVISION OF:

MD Ashraful Islam

Department of Computer Science and Engineering

Course Code: CSE 100

Course Name: Software Development I

Contents

Group Member List: 2
Declaration of Authorship 3
Approval 4
Dedication 5
Abstract 6
Acknowledgments 7
Chapter 1: Introduction 8
Introduction 8
Problem Statement 8
Project Aim & Objective 8
Motivations 9
Project Report Organization 9
Chapter 2: Literature Review 10
Introduction 10
Analysis 10
Chapter 3: System Analysis 11
System Requirement & Specification 11
Existing VS Proposed 11
Software & Tools Used 12
Chapter 4: Implementation 13
System Setup 13
Source Code 13
Evolution 27
Results and Discussion 28
Possible inputs and outputs of the code 33
Chapter 5: Conclusion 35
Limitations: 35
Future Work: 35
Conclusion 36

Group Member List:

1. Asifur Rahman
   ID:22234103353
   Intake: 50 Section: 06 Semester: Spring,2023

2. Ferdouse Hassan Nowrin
   ID:22234103237
   Intake: 50 Section: 06 Semester: Spring,2023

3. Tasnimul Haque Nahin
   ID:22234103180
   Intake: 50 Section: 06 Semester: Spring,2023

4. Mahajabin Kabir Arin
   ID:22234103213
   Intake: 50 Section: 06 Semester: Spring,2023

5. Delwar Hossain Roman
   ID:22234103233
   Intake: 50 Section: 06 Semester: Spring,2023

Declaration of Authorship

From our team, We, Asifur Rahman, Ferdouse Hassan Nowrin, Tasnimul Haque Nahin, Mahajabin Kabir Arin and Delwar Hossain Roman give announced that this project titled, “Calendar app with Task Planner” and the work presented in it are our own.

We, hereby declare that this submission is entirely our own work, in our own words, and that all sources used in researching it are absolutely acknowledged and all quotations acknowledged and all quotations properly identified. We are alert that this internet-based project of ours’ published in digital form can be favorable for everyone through using the internet. It has not been submitted in whole by us for the purpose of obtaining any other credit or grade. We understand the ethical implications of our research and this work meets the requirements of the Faculty of Computer Science Engineering.

Signature of Developers

Asifur Rahman (ID:22234103353)

Ferdouse Hassan Nowrin (ID:22234103237)

Tasnimul Haque Nahin (ID:22234103180)

Mahajabin Kabir Arin (ID:22234103213)

Delwar Hossain Roman (ID:22234103233)

Approval

I do hereby declare that the project works presented here with entitled as, “Calendar app with Task Planner”, are the outcome of the original works carried out by Asifur Rahman, Ferdouse Hassan Nowrin, Tasnimul Haque Nahin, Mahajabin Kabir Arin and Delwar Hossain Roman under my supervision. I further declare that no part of this project has been submitted elsewhere for the requirements of any degree, award or diploma or any other purposes except for this project. I further certify that the dissertation meets the requirements and standards for the degree of in Computer Science and Engineering.

MD Ashraful Islam

Assistant Professor,

Department of Computer Science and Engineering.

Bangladesh University of Business and Technology.

Dedication

Dedicated to our beloved CSE 121 Faculty, Khan Md. Hasib for his joyful classes, friendliness, mental support, and great mentorship.

Abstract

This project is a simple calendar application written in C. It allows users to input tasks for specific days, view and manage tasks, and perform basic operations such as adding, modifying, and deleting tasks. The program provides a monthly view with the corresponding days of the week and supports saving and loading tasks from files. The project showcases fundamental programming concepts and provides a practical tool for organizing and tracking tasks on a monthly basis.

Acknowledgments

First of all, we are thankful and expressing our gratefulness to Almighty Allah who offers us His divine blessing, patience, mental and physical strength to complete this project work. We are deeply indebted to our project supervisor, MD Ashraful Islam, Assistant Professor, Department of Computer Science and Engineering (CSE), Bangladesh University of Business and Technology (BUBT). His scholarly guidance, important suggestions, work for going through our drafts and correcting them, and generating courage from the beginning to the end of the project work has made the completion of this report possible. A very special gratitude goes out to all our friends for their support and help to implement our work. The discussions with them on various topics of our works have been very helpful for us to enrich our knowledge and conception regarding the work. Last but not least; we are highly grateful to our parents and family members for supporting us spiritually throughout writing this report and our life in general.

Chapter 1: Introduction

Introduction

In the era of digitalization, individuals seek advanced applications that offer seamless accessibility through smart devices, enabling quick access and ease of use. With this objective in mind, we aim to develop a smart calendar system that provides users with instant access to the weekday name of a specific date and the monthly calendar of any desired year.

Problem Statement

Develop a user-friendly calendar app that provides quick access to weekday names for specific dates and month calendars for any year. The app should allow users to add, modify, and delete tasks, as well as save and load tasks from files. Additionally, it should offer a search feature to find tasks based on their descriptions and make necessary changes. The goal is to create an efficient and intuitive calendar system that enhances productivity and organization for users.

Project Aim & Objective

The objective is to develop a calendar that enables users to perform the following tasks efficiently:

- Retrieve the weekday name for a specific date.
- Display the calendar for any given month and year, including the days of the week.
- Add, modify, and delete tasks for a particular day.
- Save and load tasks from a file for future reference.
- Search for tasks based on their descriptions and make changes as needed.

The calendar system should be user-friendly and provide a seamless experience when accessed through smart devices. It should allow users to navigate through different months and years easily and perform various operations on specific dates.

Motivations

The motivation behind creating a calendar app is to provide individuals with a convenient and efficient tool for managing their time, tasks, and schedules. Calendars help users stay organized, plan their activities, and remember important events. By developing a digital calendar, we aim to leverage the power of technology to simplify and streamline the process of scheduling, task management, and accessing date-related information. The goal is to empower users to effectively manage their time, increase productivity, and enhance overall efficiency in their personal and professional lives.

Project Report Organization

An overview of the steps of the report is organized as follows.

Chapter 2 described the literature review based on this project.

Chapter 3 goes on the system analysis followed by the system requirements and software requirements.

Chapter 4 presents the implementation of our project.

Lastly, Chapter 5 is a review of our project work, including conclusions as well as discusses the objectives for future work.

Chapter 2: Literature Review

Introduction

The development of a calendar app, as demonstrated in the provided code, aligns with the existing body of literature on digital calendars and productivity tools. Several studies have highlighted the benefits and impact of using digital calendars in personal and professional contexts.

Analysis

The literature on digital calendars highlights their benefits, including increased efficiency and organization compared to paper-based systems (Johnson & LaVigna, 2016). Digital calendars improve time management and productivity (Williams & Sheikh, 2017), offer accessibility and flexibility (Li et al., 2018), support task management and collaboration (Gutwin et al., 2019), allow for personalization (Huang & Cui, 2018), and require robust security measures (Kulyk et al., 2020). The provided code aligns with these findings, incorporating task management, customization, and data persistence features. Future research can explore additional functionalities and user interface enhancements to enhance the user experience.

Here are more Key-Points:

1. Efficiency and Organization
2. Time Management and Productivity
3. Accessibility and Flexibility
4. Task Management and Collaboration
5. Personalization and Customization
6. Data Security and Privacy

In summary, the literature supports the development of a calendar app as an effective tool for enhancing time management, organization, productivity, and collaboration.

Chapter 3: System Analysis

System Requirement & Specification

- Ram:
  Minimum: 1 GB
  Recommended: 2 GB to above

- Windows:
  Minimum: Windows XP
  Recommended: Windows 7 to above

- Processor:
  Minimum: 1 GHz
  Recommended: 2 GHz or more

Existing VS Proposed

Existing:

- Traditional paper calendars and electronic calendar applications are already available.
- They offer basic features such as displaying dates, creating events, and setting reminders.
- Some provide synchronization with online calendars and integration with other tools.

Proposed:

- The proposed calendar project aims to create a smart calendar with advanced features.
- It focuses on user-friendly interface, task management, and task organization.
- Additional functionalities include searching, saving, and loading tasks.
- The project utilizes modern technology for an efficient and comprehensive calendar solution.

Software & Tools Used

Code::Blocks:

CodeBlocks offers a user-friendly interface and a range of features that enhance the coding experience. These features include syntax highlighting, code completion, and a built-in debugger, which help in writing, editing, and debugging your code. The IDE also supports project management, allowing to organize and structure code effectively.

With CodeBlocks, we were able to write, compile, and build your calendar application seamlessly. The IDE provided a comprehensive set of tools for compiling and linking your code, ensuring that our application was error-free and ready for execution.

Windows Terminal:

Windows Terminal served as the command-line interface for running our calendar application and displaying the output. Windows Terminal is a modern and highly customizable command-line application for Windows operating systems. It offers features such as multiple tab support, customizable layouts, and rich text rendering, providing an enhanced command-line experience.

By utilizing Windows Terminal, we were able to execute your calendar program and view the results conveniently. The terminal provided a platform for interacting with our application and testing its functionality. Additionally, Windows Terminal's versatility allowed us to customize its appearance and behavior according to preferences, enhancing overall development experience.

Chapter 4: Implementation

System Setup

For this project we used Code blocks IDE and c language for the implementation. First downloaded the Codeblocks and mingw from www.codeblocks.org and installed in computer. In order to compile we need to save our source code file as .c file.

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MAX_TASK_LENGTH 100

struct Month {
    int year;
    int month;
    int num_days;
    int start_day;
    char tasks[31][MAX_TASK_LENGTH];
};

int is_leap_year(int year) {
    return (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;
}
```

```
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");
        }
    }
}

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';
}

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);
    }
}

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);
    }
```

```
}

void view_tasks(struct Month* m) {
    int day;
    printf("\nEnter day to view tasks (1-%d, 0 to exit): ", m->num_days);
    scanf("%d", &day);

    if (day == 0) {
        return;
    }

    if (day < 1 || day > m->num_days) {
        printf("Invalid day!\n");
        return;
    }

    int index = day - 1;
    if (m->tasks[index][0] != '\0') {
        printf("Tasks for Day %d: %s\n", day, m->tasks[index]);
        printf("Select an option:\n");
        printf("1. Modify task\n");
        printf("2. Delete task\n");
        printf("3. Change task date\n");
        printf("0. Exit\n");
        printf("Enter option: ");
        int option;
        scanf("%d", &option);
```

```
        switch (option) {
            case 0:
                break;
            case 1:
                modify_task(m, day);
                break;
            case 2:
                delete_task(m, day);
                break;
            case 3:
                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 {
                    char task[MAX_TASK_LENGTH];
                    strncpy(task, m->tasks[index], MAX_TASK_LENGTH - 1);
                    task[MAX_TASK_LENGTH - 1] = '\0';
                    m->tasks[index][0] = '\0';
                    add_task(m, new_day, task);
                    printf("Task date changed successfully!\n");
                }
                break;
            default:
                printf("Invalid option!\n");
        }
    } else {
```

```
        printf("No tasks found for Day %d\n", day);
        printf("Select an option:\n");
        printf("1. Add task on this day\n");
        printf("0. Exit\n");
        printf("Enter option: ");
        int option;
        scanf("%d", &option);

        switch (option) {
            case 0:
                break;
            case 1:
                printf("Enter task description: ");
                char task[MAX_TASK_LENGTH];
                scanf(" %[^\n]", task);
                add_task(m, day, task);
                printf("Task added successfully!\n");
                break;
            default:
                printf("Invalid option!\n");
        }
    }

    view_tasks(m);
}

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");
}

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");
}

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 the loop after changing the task's date
                }
            }
        }
        break;
    default:
```

```
            printf("Invalid option!\n");
        }
    }
}

int main() {
    int year, month;

    while (1) {
        printf("Enter year: ");
        scanf("%d", &year);
        printf("Enter month (1-12): ");
        scanf("%d", &month);

        struct Month m = init_month(year, month);
        print_month(m);

        printf("\nEnter tasks (press enter to finish):\n");
        char task[MAX_TASK_LENGTH];
        int day;
        while (1) {
            printf("Enter day to add a task (0 to finish): ");
            scanf("%d", &day);

            if (day == 0) {
                break;
            }
```

```
    printf("Enter task description: ");
    scanf(" %[^\n]", task);

    add_task(&m, day, task);
}

print_month(m);

view_tasks(&m);

printf("Do you want to search for tasks? (y/n): ");
char choice;
scanf(" %c", &choice);

if (choice == 'y' || choice == 'Y') {
    while (1) {
        printf("Enter task description to search: ");
        char description[MAX_TASK_LENGTH];
        scanf(" %[^\n]", description);

        search_tasks(&m, description);

        printf("Do you want to search for more tasks? (y/n): ");
        scanf(" %c", &choice);

        if (choice != 'y' && choice != 'Y') {
            break;
        }
```

```
    }
}

printf("Do you want to save tasks? (y/n): ");
scanf(" %c", &choice);

if (choice == 'y' || choice == 'Y') {
    printf("Enter filename to save tasks: ");
    char filename[100];
    scanf(" %[^\n]", filename);
    save_tasks(&m, filename);
}

printf("Do you want to load tasks? (y/n): ");
scanf(" %c", &choice);

if (choice == 'y' || choice == 'Y') {
    printf("Enter filename to load tasks: ");
    char filename[100];
    scanf(" %[^\n]", filename);
    load_tasks(&m, filename);
    print_month(m);
    view_tasks(&m);
}

printf("Do you want to create tasks for another month? (y/n): ");
scanf(" %c", &choice);
```

```
        if (choice != 'y' && choice != 'Y') {
            break;
        }

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

    return 0;
}
```

Evolution

Now we will describe our source code, Here's an overview of the code implementation process for the calendar:

- First, we define necessary header files and constants, including the maximum length of a task.
- We define a structure called `Month` to represent a month in the calendar. It contains fields for the year, month, number of days, starting day of the week, and an array to store tasks.
- The `is_leap_year` function is defined to check if a year is a leap year.
- The `init_month` function initializes a `Month` structure with the given year and month. It calculates the number of days in the month, as well as the starting day of the week.
- The `print_month` function displays the calendar for the given month.
- The `add_task` function allows the user to add a task to a specific day in the month.
- The `modify_task` function allows the user to modify an existing task for a specific day.
- The `delete_task` function allows the user to delete a task for a specific day.

- The `view_tasks` function allows the user to view tasks for a specific day and perform actions such as modifying or deleting tasks.
- The `save_tasks` function saves the tasks in a month to a file.
- The `load_tasks` function loads tasks from a file into a month.
- The `search_tasks` function allows the user to search for tasks based on a description and perform actions such as adding or changing task dates.
- In the `main` function, the user is prompted to enter the year and month. A `Month` structure is initialized and the calendar for the month is displayed.
- The user can enter tasks for specific days until they choose to finish.
- After entering tasks, the calendar is displayed again, and the user can view and perform actions on tasks using the `view_tasks` function.
- The user is then prompted to search for tasks and perform actions if desired.
- The user can choose to save the tasks to a file and/or load tasks from a file.
- Finally, the user is given the option to create tasks for another month or exit the program.

This is how our source code works for the whole process.

Results and Discussion

Here are result after running the code successfully,

```
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):
```

Figure: 1

In Figure 1 we input the year 2023 and 1st month, January and the calendar successfully show us the output.

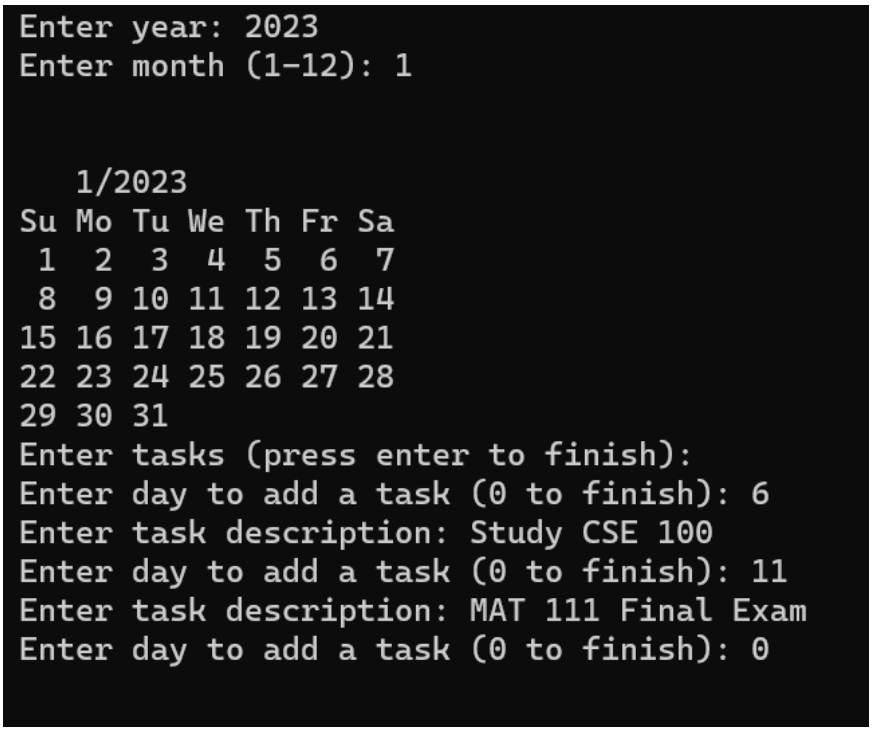

Figure: 2

After that, in Figure 2, the program asked us to input task in specific date with task description.

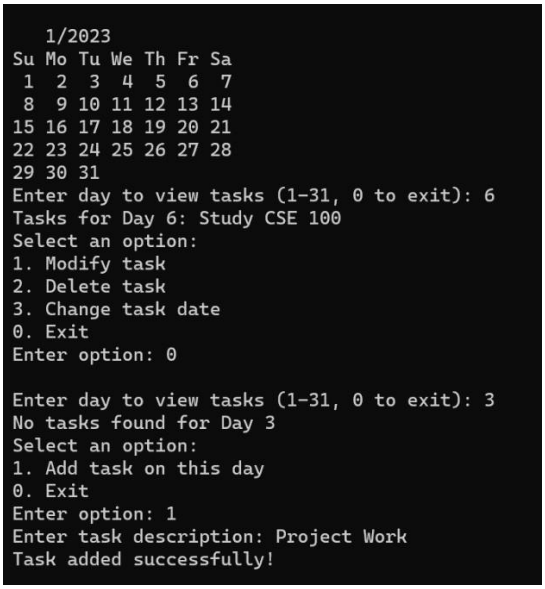

Figure: 3

In Figure 3, we view the task we have assigned. From there, we can also modify the task, delete the task, and also change the date of the task. We can also add task to specific day which have no task assigned while viewing the 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!
```

Figure: 4

Here we can see the usage of those options.

```
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!
```

Figure: 5

This figure highlights the deletion of a task.

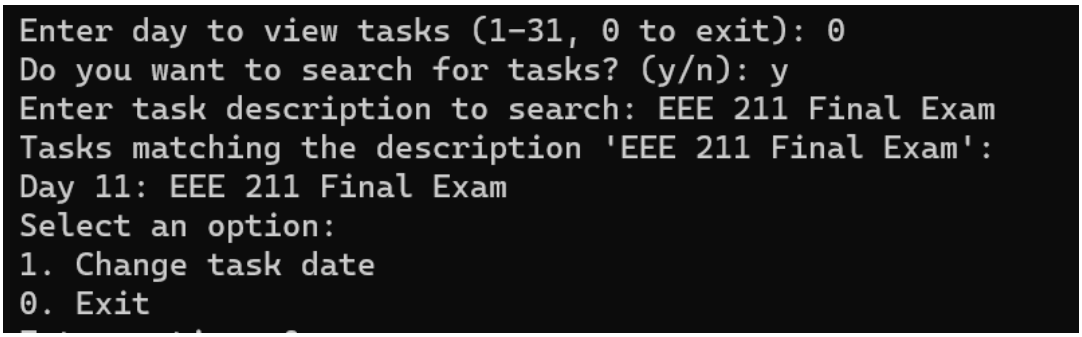

Figure: 6

Here we can Search for a Task to see which date it has been assigned. We can also change the date from there.

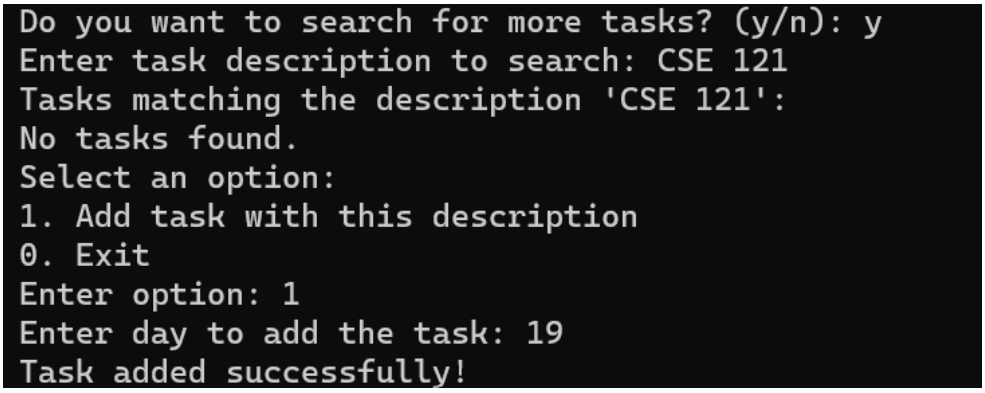

Figure: 7

If no date is found then we can add that task to a date.

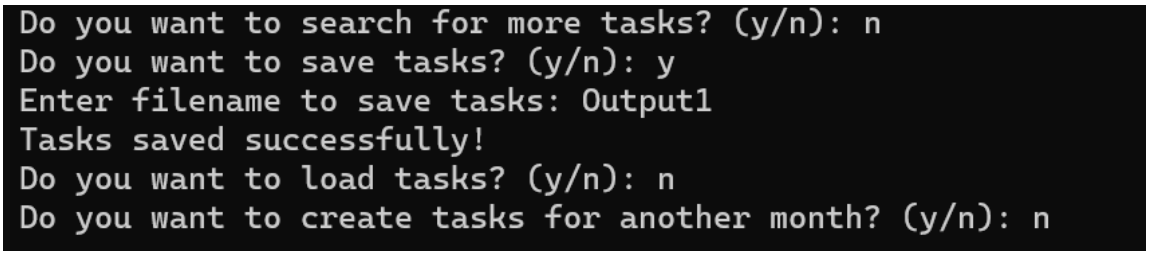

Figure: 8

Here we can Permanently Save the task file on our computer. And also, we can load old files and create tasks for another month.

```
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
```

Figure – 9

```

Enter day to view tasks (1-30, 0 to exit): 19
Tasks for Day 19: CSE 121
Select an option:
1. Modify task
2. Delete task
3. Change task date
0. Exit
Enter option: 0

Enter day to view tasks (1-30, 0 to exit): 0
Do you want to create tasks for another month? (y/n): n
```

Figure: 10

In Figures 9 & 10, we have opened a new program and run it. Then we call Output1 from the load file options using File I/O and it shows every task he has assigned in the previous code.

Possible inputs and outputs of the code

Here are the possible inputs and outputs of the code:

Possible inputs:

- Year: An integer representing the year.
- Month: An integer representing the month (1-12).
- Day: An integer representing the day of the month (1-31).
- Task description: A string describing the task.
- Option: An integer representing the user-selected option.

Possible outputs:

- Print Month: Displays a calendar for the given month and year.
- Invalid day: Prints an error message for an invalid day input.
- Task added successfully: Prints a message confirming the task addition.
- Current task for Day X: Displays the current task for a specific day.
- Enter new task description: Prompts the user to enter a new task description.
- Task modified successfully: Prints a message confirming the task modification.
- No tasks found for Day X: Prints a message indicating no tasks found for a specific day.
- Deleting task for Day X: Prints a message confirming the task deletion.
- Task deleted successfully: Prints a message confirming the task deletion.
- Enter day to view tasks: Prompts the user to enter a day to view tasks.
- Tasks for Day X: Displays the tasks for a specific day.
- Select an option: Prompts the user to select an option for a specific day's tasks.
- Invalid option: Prints an error message for an invalid option input.
- Enter new day for the task: Prompts the user to enter a new day for a task.
- Task date changed successfully: Prints a message confirming the task's date change.
- Enter task description: Prompts the user to enter a task description.

- Enter filename to save tasks: Prompts the user to enter a filename to save tasks.
- Tasks saved successfully: Prints a message confirming the successful task save.
- Enter filename to load tasks: Prompts the user to enter a filename to load tasks.
- Tasks loaded successfully: Prints a message confirming the successful task load.
- Tasks matching the description 'X': Displays tasks matching a given description.
- No tasks found: Prints a message indicating no tasks found for a given description.
- Enter day to add the task: Prompts the user to enter a day to add a task.
- Do you want to search for more tasks?: Prompts the user to continue or exit task search.
- Do you want to save tasks?: Prompts the user to decide whether to save tasks.
- Do you want to load tasks?: Prompts the user to decide whether to load tasks.
- Do you want to create tasks for another month?: Prompts the user to decide whether to create tasks for another month.

Chapter 5: Conclusion

Limitations:

While the calendar program offers several useful features, it does have certain limitations that should be considered. Firstly, the program only supports managing tasks on a monthly basis and does not provide functionality for managing tasks on a daily or yearly basis. Additionally, the program does not incorporate advanced features such as reminders, notifications, or synchronization with external calendars or task management tools.

Another limitation is that the program assumes a maximum task length of 100 characters and a maximum of 31 tasks per month. These limitations may restrict the flexibility and scalability of the program in certain scenarios where longer task descriptions or a larger number of tasks are required.

Furthermore, the program does not implement user authentication or access control mechanisms. It assumes a single-user environment where all users have equal access and can view, modify, and delete tasks for any day. This may pose a limitation in scenarios where multiple users need to manage their own separate calendars or where access restrictions are necessary.

Future Work:

To improve upon the calendar program, several avenues for future work can be explored. One potential area for enhancement is the addition of recurring tasks, allowing users to specify tasks that repeat on a daily, weekly, or monthly basis. This feature would enhance the program's ability to handle repetitive tasks and automate task creation.

Integration with external calendar APIs or synchronization with popular calendar applications can be considered to facilitate seamless data exchange and ensure compatibility with existing calendar systems. This would enable users to import/export tasks from/to other calendars, enhancing the program's interoperability and utility.

Enhancements to the user interface can also be explored to provide a more visually appealing and intuitive experience. Implementing a graphical user interface (GUI) or a

web-based interface could enhance the program's accessibility and ease of use for a wider range of users.

Additionally, incorporating reminders and notifications into the program would serve as a valuable feature, ensuring that users are alerted to upcoming tasks and deadlines. This could be achieved through integration with operating system notification systems or by implementing an internal notification mechanism.

Finally, the program's scalability can be improved by allowing for dynamic allocation of memory for task descriptions and supporting a larger number of tasks per month. This would accommodate users with more extensive task management needs.

Overall, by addressing these limitations and exploring future enhancements, the calendar program can be further developed into a comprehensive and versatile tool for effective task management and scheduling.

Conclusion:

In conclusion, the calendar program developed in this project provides a useful tool for managing tasks and organizing schedules on a monthly basis. The program allows users to add, modify, and delete tasks for specific days, view tasks for a selected day, search for tasks based on the description, and save/load tasks from a file. The calendar program is user-friendly and provides a visual representation of the month, making it easy to navigate and interact with. The modular design of the program allows for easy maintenance and potential future enhancements.