SMART TASK MANAGER

MINOR PROJECT REPORT

By

CH.S.K .GOWTHAM (RA2211027010149)
S.SAI CHARANI (RA2211027010186)

Under the guidance of

Dr. ARCHANA K.S

In partial fulfilment for the Course

Of

21CSS201J- DATA STRUCTURE AND ALGORITHM

in **DATA SCIENCE AND BUSINESS SYSTEMS**

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSS201J-DATA STRUCTURE AND ALGORITHM** entitled in "**SMART TASK MANAGER**" is the bonafide work of **CH.S.K.GOWTHAM(RA2211027010149)** and **S.SAI CHARANI (RA2211027010186)** who carried out the work under my supervision.

SIGNATURE

Dr. Archana K.S
Assistant Professor
Department of Data Science and Business System
SRM Institute of Science and Technology
Kattankulathur

ABSTRACT

The task manager project is designed to provide users with a comprehensive and user-friendly platform for organizing and managing their tasks efficiently. The system supports key functionalities, including task addition, updating, deletion, and various display options. Users can input task details such as title, description, deadline, and estimated time during task creation. The update feature allows users to modify existing tasks by providing new information. Deletion enables users to remove tasks from the manager, providing a streamlined approach to task organization. The display functionalities encompass presenting all tasks, sorting tasks by priority (typically based on deadlines), and arranging tasks by the shortest job first (sorted by estimated time).

The utilization of priority queues optimizes the sorting process, enhancing the overall performance of the system. The modular design of the code promotes readability and maintainability, facilitating potential future enhancements. With a menu-driven interface guiding user interactions, including the option to quit, the task manager project offers a versatile solution for effective task management.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. Muthamizhchelvan**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G, Professor, Department of Data Science and Business Systems and Course Coordinator** for their constant encouragement and support.

We are highly thankful to my Course project Faculty **Dr.K.S.Archana, Assistant Professor**,**Department of Data Science and Business System** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. M Lakshmi, Professor, Department of Data Science and Business Systems** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1.INTRODUCTION

The task manager project is conceived as a practical and efficient solution for individuals seeking an organized approach to managing their tasks. In today's dynamic and fast-paced environment, the need for a streamlined task management system is more crucial than ever. This project aims to address this need by providing users with a feature-rich application that enables them to seamlessly add, update, delete, and display tasks. The project's foundation lies in the utilization of C++ programming language, employing concepts such as structures, queues, and priority queues to ensure robust functionality.

The project's user-centric design allows individuals to input detailed information for each task, such as title, description, deadline, and estimated time required. The intuitive update feature permits users to modify task details, while the deletion functionality streamlines the removal of completed or irrelevant tasks. Display options include showcasing all tasks, sorting them by priority (typically based on deadlines), and arranging them by the shortest job first, which is particularly useful for optimizing time management.

A key feature of the project is the use of priority queues for efficient sorting, enhancing the system's performance and responsiveness. The modular structure of the code promotes code readability, maintainability, and provides a foundation for potential future expansions or feature additions.

1.1.MOTIVATION

The Task manager project is motivated by the recognition of the growing complexities individuals face in managing their tasks amid the fast-paced nature of modern life. In a world filled with diverse responsibilities, shifting priorities, and stringent deadlines, there is a clear need for an efficient and user-friendly task management system. The project is driven by a commitment to address the challenges associated with task organization, offering a centralized platform that empowers users to seamlessly add, update, delete, and visualize their tasks.

Efficiency and productivity stand as key drivers for the development of the task manager. The project seeks to simplify the often intricate process of task management, providing users with a streamlined approach to navigate through their responsibilities. By utilizing technology to address these daily challenges, the task manager not only serves as a practical solution for end-users but also as a valuable platform for individuals to enhance their programming skills. Through the implementation of data structures and algorithms, the project contributes to skill development, aligning with the contemporary trend of utilizing software solutions to tackle real-world problems. In essence, the task manager project is motivated by a dual purpose: to enhance the daily lives of users through a reliable task management tool and to foster learning and skill development in the realm of programming.

1.2. OBJECTIVE

The central objective of the task manager project is to offer a comprehensive solution for individuals grappling with the complexities of modern task management. This involves creating an intuitive and efficient platform where users can seamlessly add, update, and delete tasks.

By providing a user-friendly interface, the project aims to simplify the task management process, allowing users to navigate through their responsibilities with ease. Prioritization is a key focus, intending to assist users in making informed decisions about task urgency and importance. The incorporation of sorting mechanisms, such as arranging tasks by priority based on deadlines or by the shortest job first, aims to empower users in optimizing their time and resources.

1.3. PROBLEM STATEMENT

The problem statement for the task manager project revolves around the challenges individuals face in efficiently managing their tasks within the contemporary, fast-paced lifestyle. In this dynamic environment, individuals often struggle with organizing and prioritizing tasks, leading to inefficiencies, missed deadlines, and increased stress levels. The absence of a centralized and user-friendly tool exacerbates the problem, as existing solutions may lack the necessary features or fail to provide an intuitive interface. Additionally, the growing complexities of task details, such as varying deadlines and estimated completion times, further contribute to the need for a more sophisticated task management system.

1.4. CHALLENEGES

Developing the task manager project entails overcoming a series of challenges to ensure its seamless functionality and user satisfaction. First and foremost, the creation of an intuitive and user-friendly interface is a critical challenge. Striking the right balance between simplicity and comprehensive functionality is essential to ensure users can navigate and utilize all features effortlessly. Additionally, efficient management and storage of task data, including titles, descriptions, and deadlines, pose a significant challenge, necessitating the implementation of a robust database or file system.

Real-time updates and synchronization across multiple devices present another formidable challenge. Users expect the ability to access and modify tasks seamlessly from various platforms, demanding the development of robust mechanisms for data synchronization. The implementation of effective sorting and prioritization algorithms is paramount for the project's success, requiring careful consideration of computational efficiency while sorting tasks based on deadlines, prioritizing them, and arranging them by the shortest job first.

2.LITERATURE SURVEY

1. "**Getting Things Done: The Art of Stress-Free Productivity**" by David Allen:This book provides insights into personal productivity and task management, offering principles that can inspire features in a task manager.

2. **"The Pragmatic Programmer: Your Journey to Mastery**" by Dave Thomas and Andy Hunt:Offers practical advice and tips for software development, covering a wide range of topics that can be beneficial for building the task manager.

3. **"Secure Coding in C and C++"** by Robert C. Seacord: This book addresses secure coding practices, providing guidance on implementing security measures in C++ programming, relevant for your task manager project.

3.REQUIREMENTS

3.1. HARDWARE REQUIREMENTS

Processor:

- Minimum: Intel Core i3 or equivalent
- Recommended: Intel Core i5 or equivalent

RAM (Random Access Memory):

- Minimum: 4GB
- Recommended: 8GB or higher

Storage:

- Minimum: 128GB SSD or HDD
- Recommended: 256GB SSD or higher for better performance

Operating System:

- Ubuntu 18.04 LTS or later
- macOS 10.14 Mojave or later
- Ubuntu 18.04 LTS or later

3.2. SOFTWARE REQUIREMENTS

1.Integrated Development Environment (IDE):

- Select a suitable IDE for software development, such as Visual Studio Code, JetBrains IntelliJ IDEA, or Eclipse.

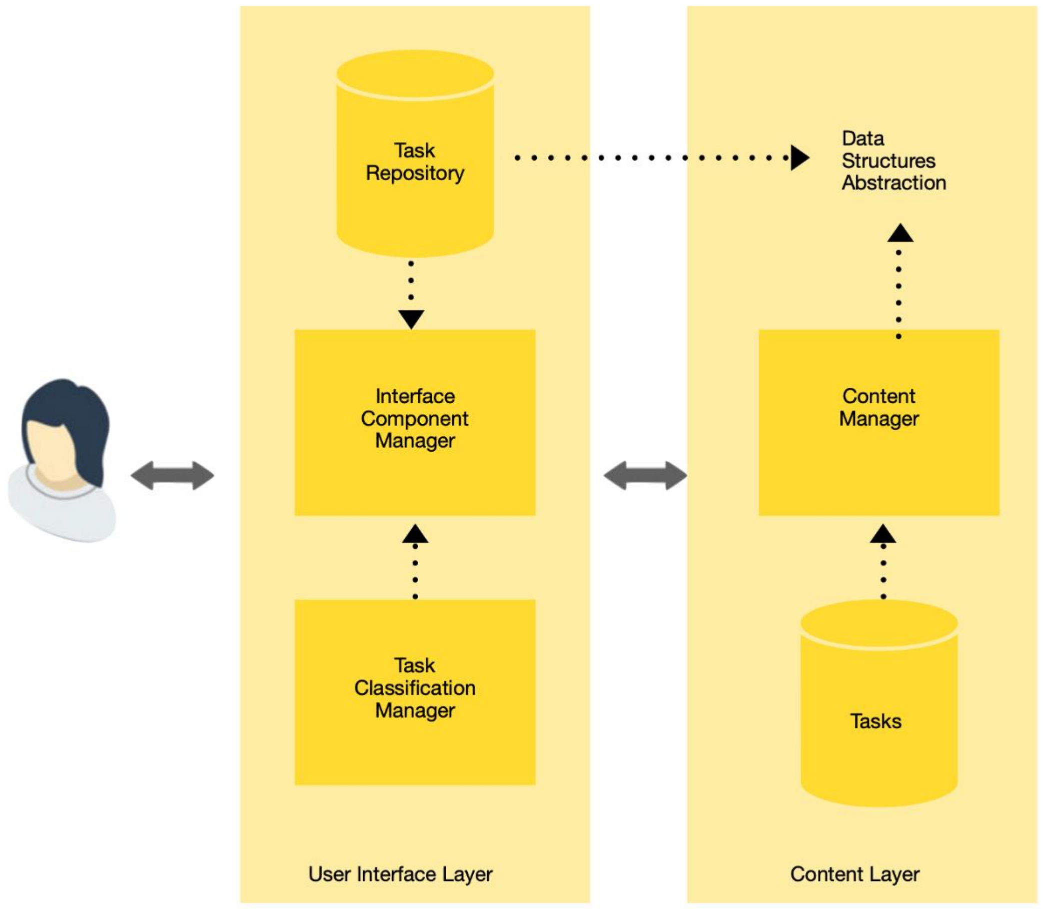
2. Programming Language:

- Specify the programming language used for development. Common choices include Java, C++, Python, or JavaScript (for web-based applications).

3.Version Control:

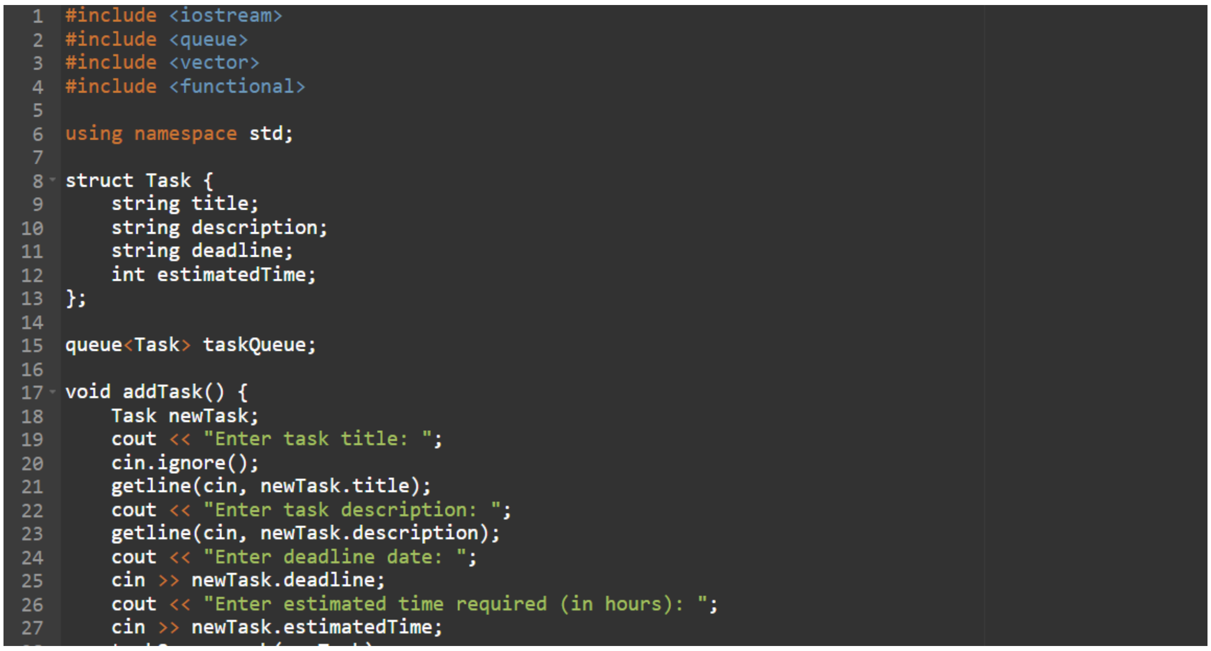
- Utilize version control systems like Git for collaborative development and code management.

4.ARCHITECTURE AND DESIGN

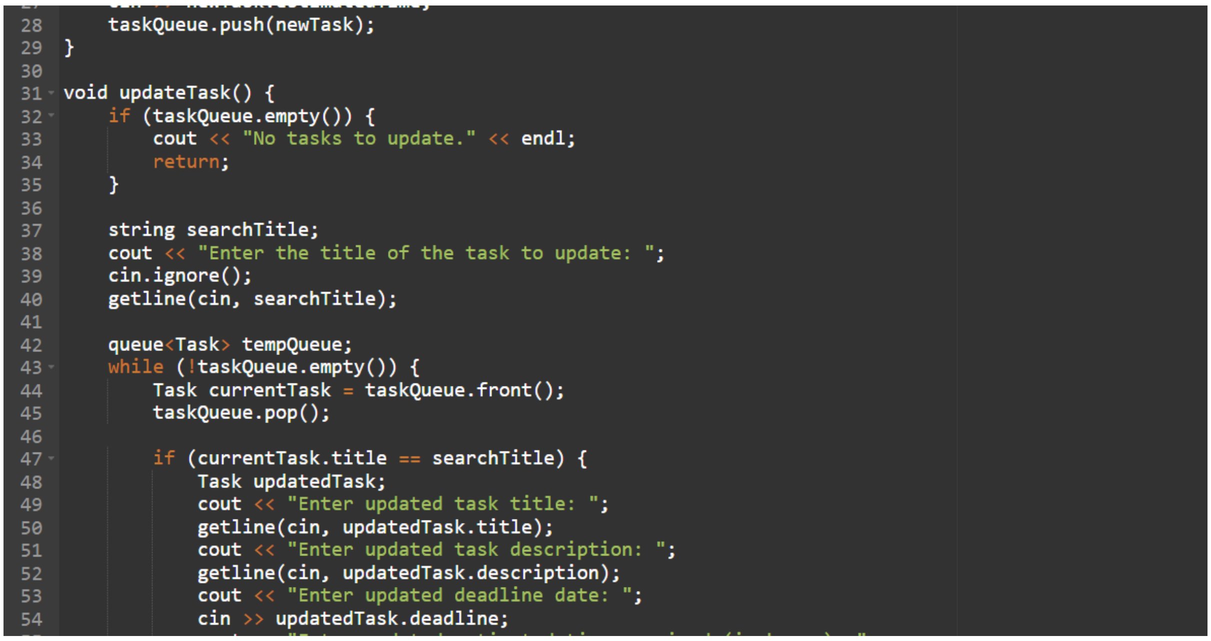

Task Repository
Data Structures Abstraction
Interface Component Manager
Content Manager
Task Classification Manager
Tasks
User Interface Layer
Content Layer

5.IMPLEMENTATION

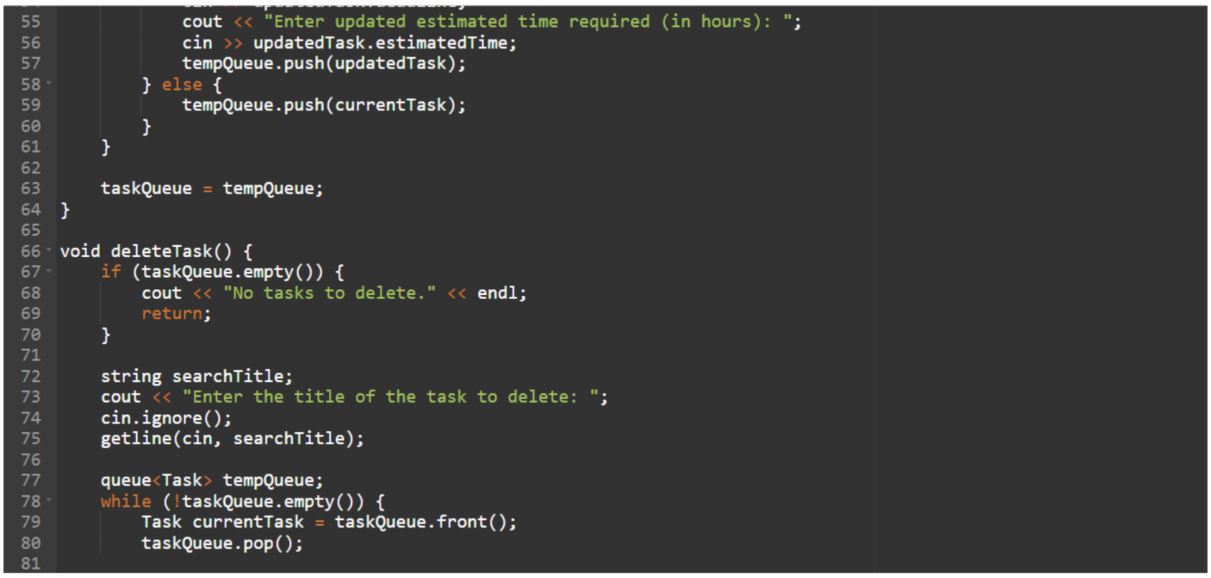
CODE SNIPPETS:



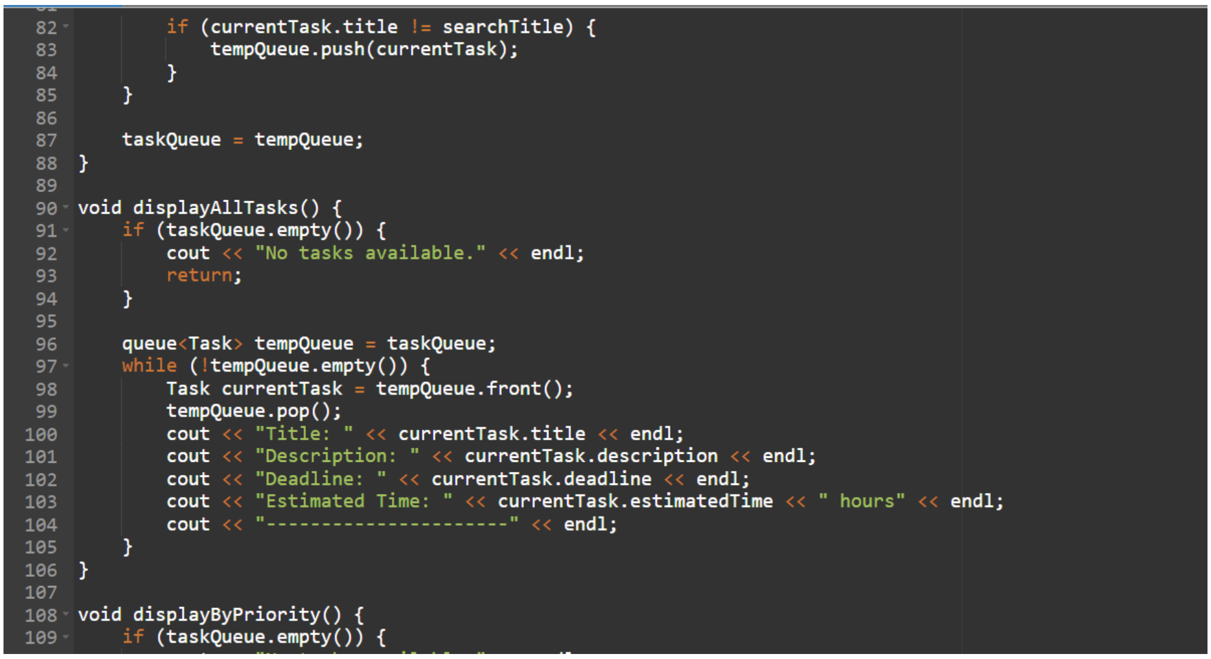
```cpp
#include <iostream>
#include <queue>
#include <vector>
#include <functional>

using namespace std;

struct Task {
    string title;
    string description;
    string deadline;
    int estimatedTime;
};

queue<Task> taskQueue;

void addTask() {
    Task newTask;
    cout << "Enter task title: ";
    cin.ignore();
    getline(cin, newTask.title);
    cout << "Enter task description: ";
    getline(cin, newTask.description);
    cout << "Enter deadline date: ";
    cin >> newTask.deadline;
    cout << "Enter estimated time required (in hours): ";
    cin >> newTask.estimatedTime;
```



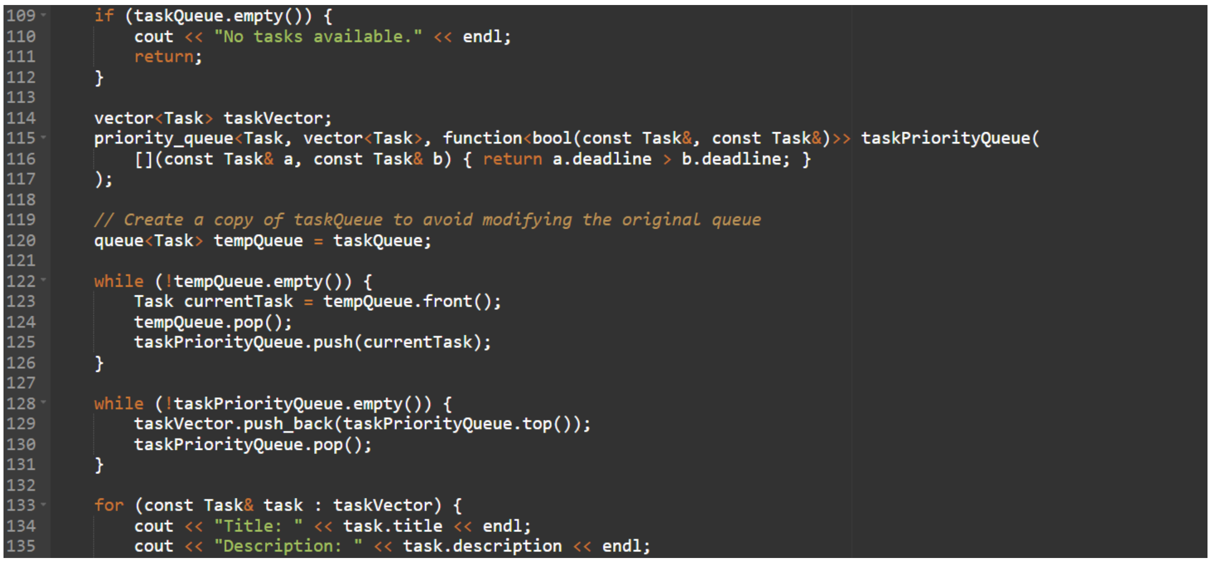
```cpp
    taskQueue.push(newTask);
}

void updateTask() {
    if (taskQueue.empty()) {
        cout << "No tasks to update." << endl;
        return;
    }

    string searchTitle;
    cout << "Enter the title of the task to update: ";
    cin.ignore();
    getline(cin, searchTitle);

    queue<Task> tempQueue;
    while (!taskQueue.empty()) {
        Task currentTask = taskQueue.front();
        taskQueue.pop();

        if (currentTask.title == searchTitle) {
            Task updatedTask;
            cout << "Enter updated task title: ";
            getline(cin, updatedTask.title);
            cout << "Enter updated task description: ";
            getline(cin, updatedTask.description);
            cout << "Enter updated deadline date: ";
            cin >> updatedTask.deadline;
```



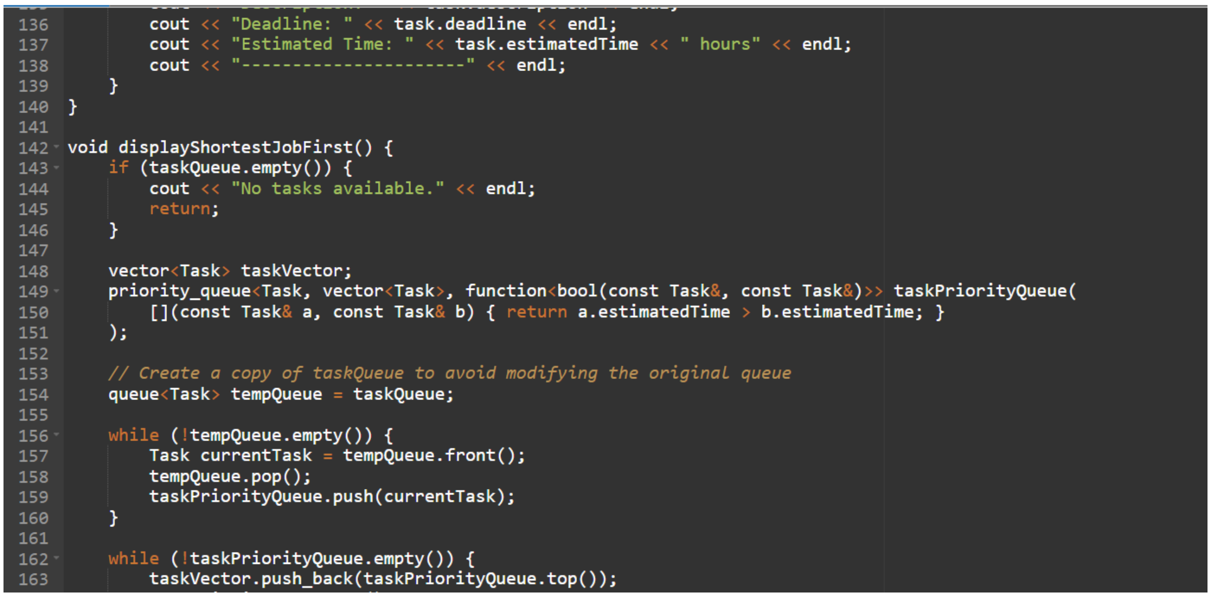
```
            cout << "Enter updated estimated time required (in hours): ";
            cin >> updatedTask.estimatedTime;
            tempQueue.push(updatedTask);
        } else {
            tempQueue.push(currentTask);
        }
    }

    taskQueue = tempQueue;
}

void deleteTask() {
    if (taskQueue.empty()) {
        cout << "No tasks to delete." << endl;
        return;
    }

    string searchTitle;
    cout << "Enter the title of the task to delete: ";
    cin.ignore();
    getline(cin, searchTitle);

    queue<Task> tempQueue;
    while (!taskQueue.empty()) {
        Task currentTask = taskQueue.front();
        taskQueue.pop();

```



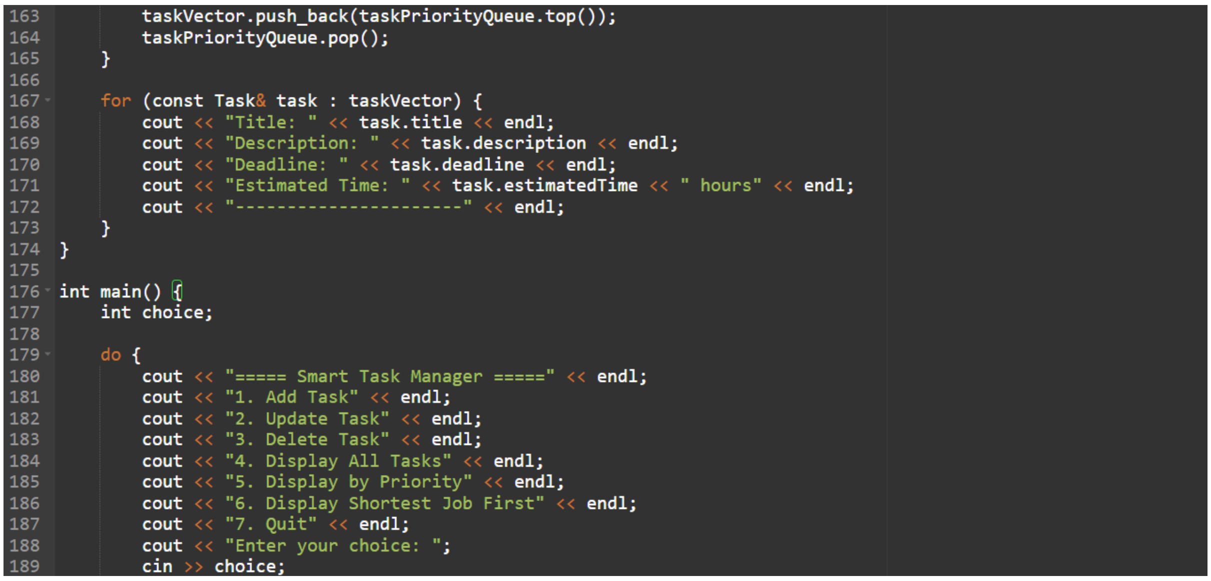
```
        if (currentTask.title != searchTitle) {
            tempQueue.push(currentTask);
        }
    }

    taskQueue = tempQueue;
}

void displayAllTasks() {
    if (taskQueue.empty()) {
        cout << "No tasks available." << endl;
        return;
    }

    queue<Task> tempQueue = taskQueue;
    while (!tempQueue.empty()) {
        Task currentTask = tempQueue.front();
        tempQueue.pop();
        cout << "Title: " << currentTask.title << endl;
        cout << "Description: " << currentTask.description << endl;
        cout << "Deadline: " << currentTask.deadline << endl;
        cout << "Estimated Time: " << currentTask.estimatedTime << " hours" << endl;
        cout << "----------------------" << endl;
    }
}

void displayByPriority() {
    if (taskQueue.empty()) {
```



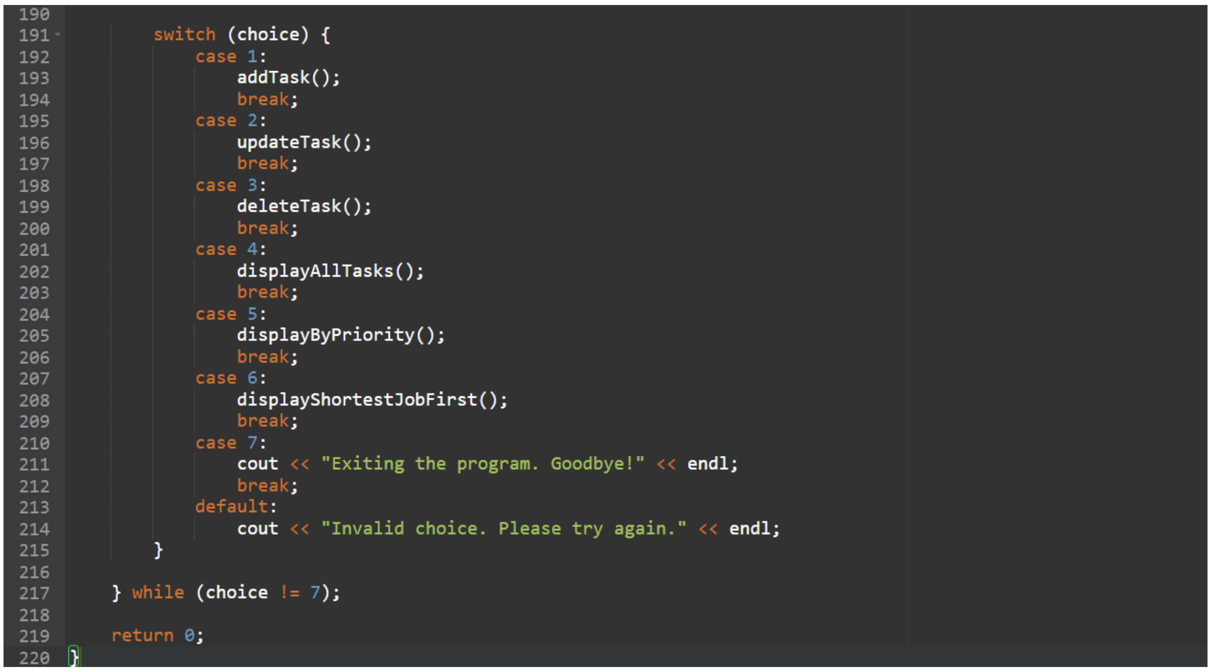
```cpp
    if (taskQueue.empty()) {
        cout << "No tasks available." << endl;
        return;
    }

    vector<Task> taskVector;
    priority_queue<Task, vector<Task>, function<bool(const Task&, const Task&)>> taskPriorityQueue(
        [](const Task& a, const Task& b) { return a.deadline > b.deadline; }
    );

    // Create a copy of taskQueue to avoid modifying the original queue
    queue<Task> tempQueue = taskQueue;

    while (!tempQueue.empty()) {
        Task currentTask = tempQueue.front();
        tempQueue.pop();
        taskPriorityQueue.push(currentTask);
    }

    while (!taskPriorityQueue.empty()) {
        taskVector.push_back(taskPriorityQueue.top());
        taskPriorityQueue.pop();
    }

    for (const Task& task : taskVector) {
        cout << "Title: " << task.title << endl;
        cout << "Description: " << task.description << endl;
```



```cpp
        cout << "Deadline: " << task.deadline << endl;
        cout << "Estimated Time: " << task.estimatedTime << " hours" << endl;
        cout << "----------------------" << endl;
    }
}

void displayShortestJobFirst() {
    if (taskQueue.empty()) {
        cout << "No tasks available." << endl;
        return;
    }

    vector<Task> taskVector;
    priority_queue<Task, vector<Task>, function<bool(const Task&, const Task&)>> taskPriorityQueue(
        [](const Task& a, const Task& b) { return a.estimatedTime > b.estimatedTime; }
    );

    // Create a copy of taskQueue to avoid modifying the original queue
    queue<Task> tempQueue = taskQueue;

    while (!tempQueue.empty()) {
        Task currentTask = tempQueue.front();
        tempQueue.pop();
        taskPriorityQueue.push(currentTask);
    }

    while (!taskPriorityQueue.empty()) {
        taskVector.push_back(taskPriorityQueue.top());
```

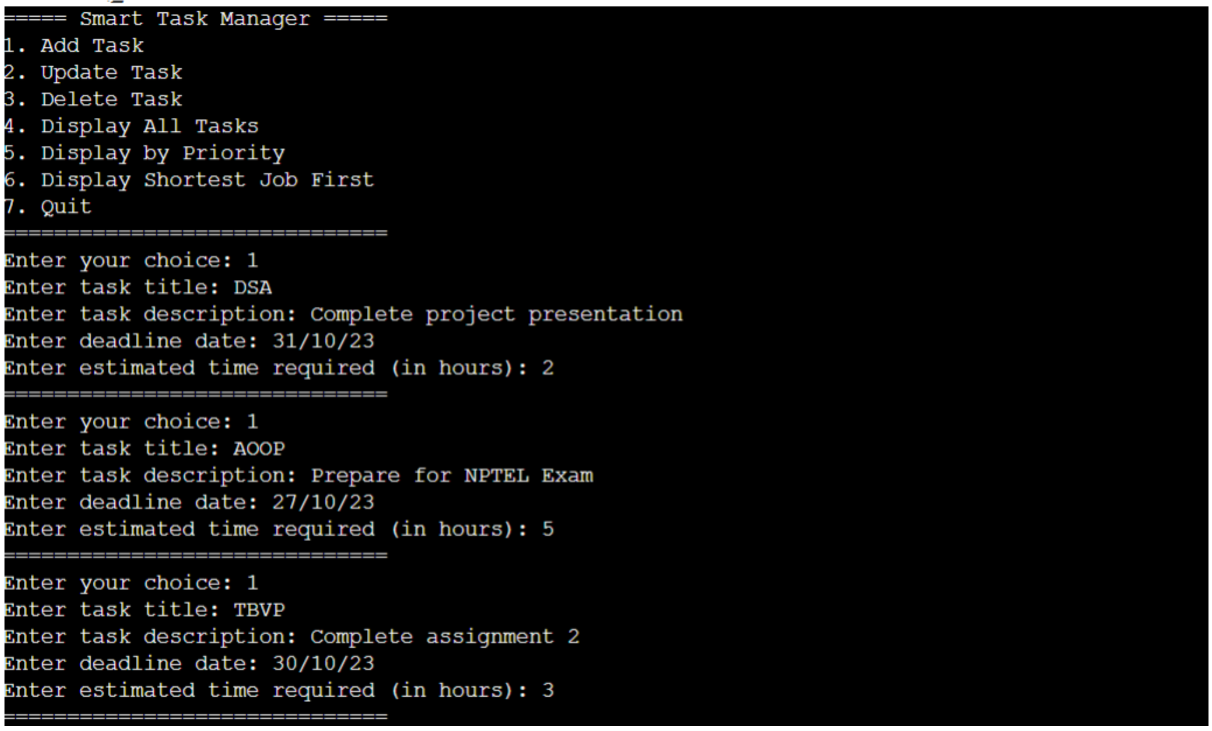


```cpp
        taskVector.push_back(taskPriorityQueue.top());
        taskPriorityQueue.pop();
    }

    for (const Task& task : taskVector) {
        cout << "Title: " << task.title << endl;
        cout << "Description: " << task.description << endl;
        cout << "Deadline: " << task.deadline << endl;
        cout << "Estimated Time: " << task.estimatedTime << " hours" << endl;
        cout << "----------------------" << endl;
    }
}

int main() {
    int choice;

    do {
        cout << "===== Smart Task Manager =====" << endl;
        cout << "1. Add Task" << endl;
        cout << "2. Update Task" << endl;
        cout << "3. Delete Task" << endl;
        cout << "4. Display All Tasks" << endl;
        cout << "5. Display by Priority" << endl;
        cout << "6. Display Shortest Job First" << endl;
        cout << "7. Quit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
```

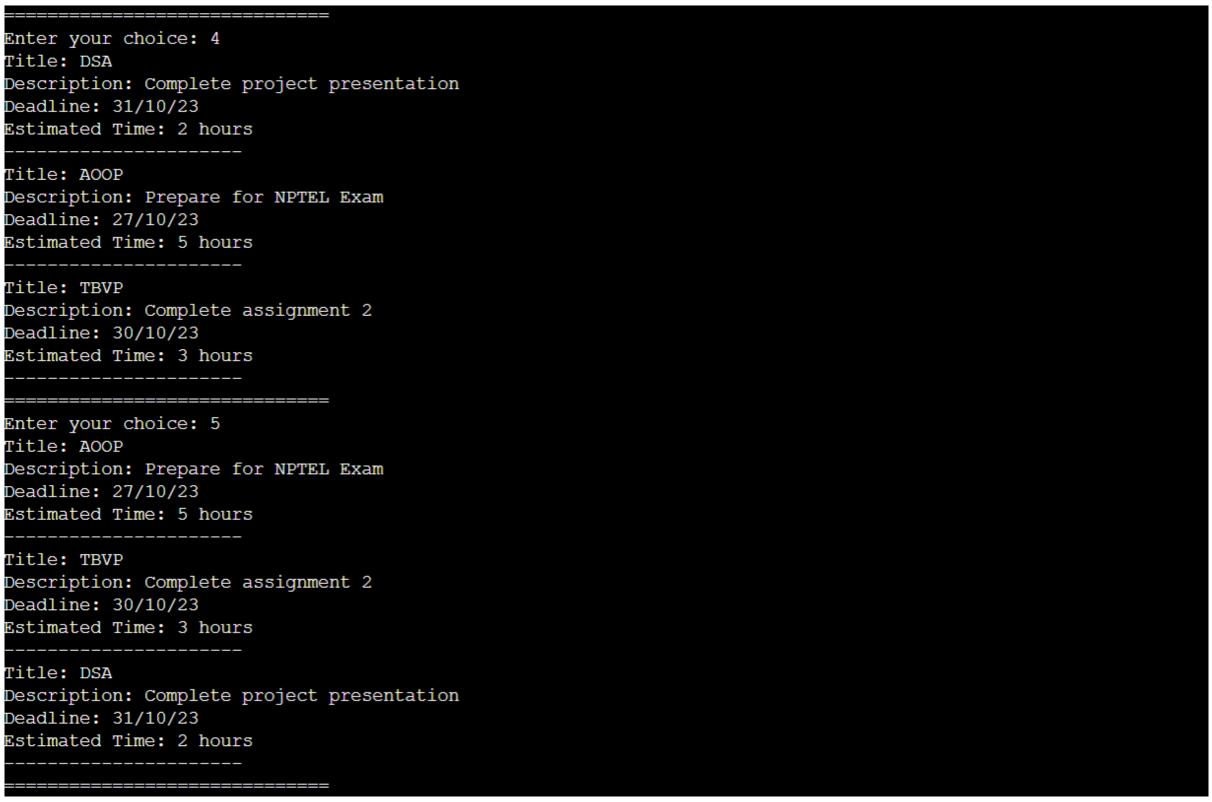


```cpp

        switch (choice) {
            case 1:
                addTask();
                break;
            case 2:
                updateTask();
                break;
            case 3:
                deleteTask();
                break;
            case 4:
                displayAllTasks();
                break;
            case 5:
                displayByPriority();
                break;
            case 6:
                displayShortestJobFirst();
                break;
            case 7:
                cout << "Exiting the program. Goodbye!" << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }

    } while (choice != 7);

    return 0;
}
```

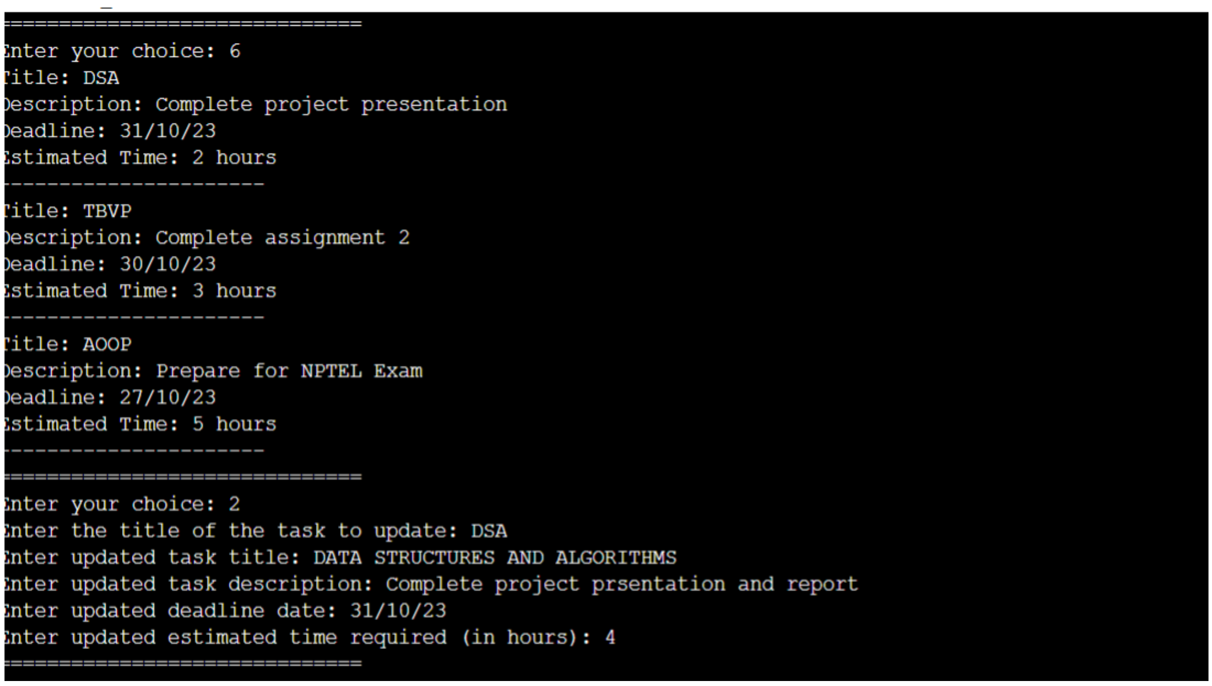
6. RESULTS AND DISCUSSIONS



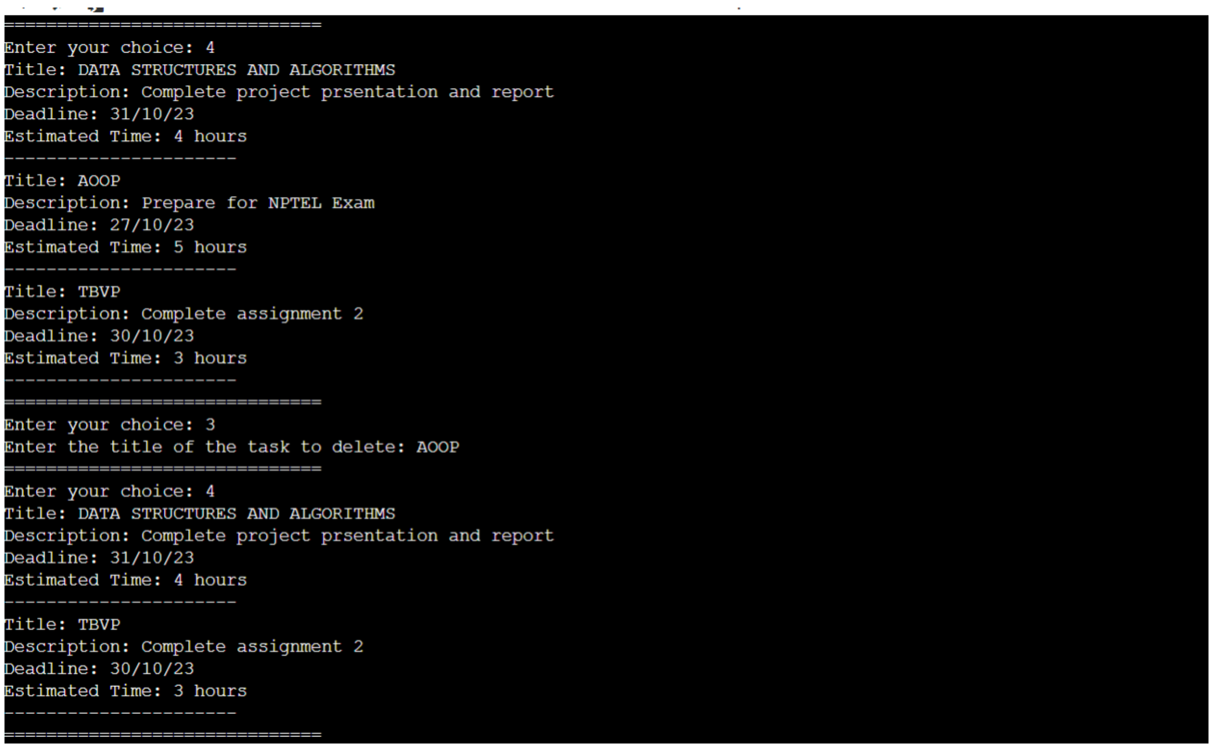
```
===== Smart Task Manager =====
1. Add Task
2. Update Task
3. Delete Task
4. Display All Tasks
5. Display by Priority
6. Display Shortest Job First
7. Quit
==============================
Enter your choice: 1
Enter task title: DSA
Enter task description: Complete project presentation
Enter deadline date: 31/10/23
Enter estimated time required (in hours): 2
==============================
Enter your choice: 1
Enter task title: AOOP
Enter task description: Prepare for NPTEL Exam
Enter deadline date: 27/10/23
Enter estimated time required (in hours): 5
==============================
Enter your choice: 1
Enter task title: TBVP
Enter task description: Complete assignment 2
Enter deadline date: 30/10/23
Enter estimated time required (in hours): 3
==============================
```



```
==============================
Enter your choice: 4
Title: DSA
Description: Complete project presentation
Deadline: 31/10/23
Estimated Time: 2 hours
----------------------
Title: AOOP
Description: Prepare for NPTEL Exam
Deadline: 27/10/23
Estimated Time: 5 hours
----------------------
Title: TBVP
Description: Complete assignment 2
Deadline: 30/10/23
Estimated Time: 3 hours
----------------------
==============================
Enter your choice: 5
Title: AOOP
Description: Prepare for NPTEL Exam
Deadline: 27/10/23
Estimated Time: 5 hours
----------------------
Title: TBVP
Description: Complete assignment 2
Deadline: 30/10/23
Estimated Time: 3 hours
----------------------
Title: DSA
Description: Complete project presentation
Deadline: 31/10/23
Estimated Time: 2 hours
----------------------
==============================
```



```
==============================
Enter your choice: 6
Title: DSA
Description: Complete project presentation
Deadline: 31/10/23
Estimated Time: 2 hours
----------------------
Title: TBVP
Description: Complete assignment 2
Deadline: 30/10/23
Estimated Time: 3 hours
----------------------
Title: AOOP
Description: Prepare for NPTEL Exam
Deadline: 27/10/23
Estimated Time: 5 hours
----------------------
==============================
Enter your choice: 2
Enter the title of the task to update: DSA
Enter updated task title: DATA STRUCTURES AND ALGORITHMS
Enter updated task description: Complete project prsentation and report
Enter updated deadline date: 31/10/23
Enter updated estimated time required (in hours): 4
==============================
```



```
==============================
Enter your choice: 4
Title: DATA STRUCTURES AND ALGORITHMS
Description: Complete project prsentation and report
Deadline: 31/10/23
Estimated Time: 4 hours
----------------------
Title: AOOP
Description: Prepare for NPTEL Exam
Deadline: 27/10/23
Estimated Time: 5 hours
----------------------
Title: TBVP
Description: Complete assignment 2
Deadline: 30/10/23
Estimated Time: 3 hours
----------------------
==============================
Enter your choice: 3
Enter the title of the task to delete: AOOP
==============================
Enter your choice: 4
Title: DATA STRUCTURES AND ALGORITHMS
Description: Complete project prsentation and report
Deadline: 31/10/23
Estimated Time: 4 hours
----------------------
Title: TBVP
Description: Complete assignment 2
Deadline: 30/10/23
Estimated Time: 3 hours
----------------------
==============================
```

7.CONCLUSION

In the culminating phases of the task manager project, we take pride in the successful development of a robust and user-centric system designed to address the core requisites of task management. The project has effectively achieved its primary goals, implementing essential features such as task creation, updates, and deletions, while also incorporating advanced functionalities like sophisticated sorting algorithms and real-time updates.

The user interface, meticulously crafted with a focus on usability principles, ensures a seamless and aesthetically pleasing experience for users. The project's unwavering commitment to security is evident through the integration of robust measures, including user authentication and data encryption, aligning with industry best practices.

7.1FUTURE ENHANCEMENT

- **Collaboration Features:**

  Introduce collaboration features that allow users to share tasks or projects with team members.
  Implement real-time collaboration on tasks for increased productivity.

- **Gamification:**

  Implement gamification elements to motivate users to complete tasks.
  Introduce rewards, achievements, or leaderboards for task completion milestones.

- **Mobile App Enhancements:**

  Optimize the mobile application for various devices and screen sizes.
  Introduce mobile-specific features for on-the-go task management.

- **Offline Mode:**

  Develop an offline mode that allows users to access and modify tasks without an internet connection.
  Synchronize offline changes when the user reconnects.

- **Machine Learning and Smart Suggestions:**

  Implement machine learning algorithms to provide intelligent task recommendations.
  Analyze user behavior to suggest priorities, deadlines, or estimated times for tasks.

8.REFRENCE

- Github repository for a simple task manager using Data Structures: https://github.com/topics/task-manager?l=c

- Github repository for simple taskmanager: https://github.com/ramyani-ghosh/Task-manager-reminder.

- Stack Overflow Documentation and code for task manager using C++:

  https://stackoverflow.com/questions/4725819/process-manager