C MICROPROJECT

NAME: GAYATHRI ANI

ROLL NO: 35

COURSE NAME: C PROGRAMMING

DATE:22/07/2024

INTRODUCTION

OVERVIEW

The primary aim of the Task Management System is to help individuals and teams plan, track, and manage their tasks efficiently. It allows users to create tasks, assign them to team members, set deadlines, and monitor progress.

The system will cater to both personal task management and team project coordination, providing essential tools for efficient task handling and collaboration

PROBLEM STATEMENT

The Task Management System project involves developing a console-based application in C to help users manage their tasks effectively. This system will enable users to create, edit, delete, and view tasks, set priorities, and assign deadlines. By leveraging file handling for persistent storage, the application ensures that tasks are saved and accessible across sessions. The goal is to provide a straightforward, efficient tool for improving task organization and productivity.

OBJECTIVE

The objective of the Task Management System project is to create a user-friendly console application that allows users to manage their tasks efficiently. Specifically, the project aims to:

1. Task Creation: Enable users to create tasks with essential details like title, description, due date, and priority.
2. Task Management: Provide functionalities to edit, delete, and view tasks.
3. Prioritization and Deadlines: Allow users to prioritize tasks and set deadlines for effective time management.
4. Persistent Storage: Implement file handling to save tasks persistently, ensuring data is retained between sessions.
5. User Interface: Develop an intuitive, text-based interface that simplifies task management.
6. Performance: Ensure the application performs efficiently, handling a reasonable number of tasks without degradation.
7. Portability: Design the system to be platform-independent, running on any standard C compiler.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

The hardware requirements for a C Program Project about a Task Management System are minimal due to its console-based nature. Here are the basic hardware requirements:

1. Processor: Any modern processor (e.g., Intel Core i3 or equivalent) should be sufficient.

2. RAM: At least 2GB of RAM to ensure smooth operation, although modern systems typically have much more.

3. Storage: A few megabytes of free storage space for the C compiler, source code, and task data files. A minimum of 100MB is recommended.

4. Display: A monitor with a resolution of at least 800x600 pixels.

5. Keyboard: Standard keyboard for input.

6. Operating System: The project can run on any operating system that supports a C compiler, such as Windows, Linux, or macOS.

SOFTWARE REQUIREMENT

The software requirements for a C Program Project about a Task Management System include the following:

1. Operating System:

- Windows, Linux, or macOS.

2. C Compiler:

- GCC (GNU Compiler Collection) for Linux or macOS.

- MinGW (Minimalist GNU for Windows) for Windows.

- Alternatively, any standard C compiler compatible with the chosen operating system.

3.Text Editor:

- Any text editor such as Notepad++, Sublime Text, Atom, or Visual Studio Code if not using an IDE.

4.Version Control System (optional but recommended for larger projects or collaboration):

- Git for version control and GitHub or GitLab for repository hosting.

5. Build Automation Tool :

- Make (on Linux and macOS) or CMake for managing the build process.

6.Debugger:

- GDB (GNU Debugger) for debugging the C program.

DESIGN AND DEVELOPMENT

DESCRIPTION OF PROGRAM LOGIC

program logic for a Task Management System in a C program:

1. Data Storage: Tasks are stored in a structured format using arrays. Each task includes details like title, description, due date, and priority.

2. File Handling: Tasks are saved to a file when added or modified, ensuring they persist between program runs. The program loads tasks from this file when started.

3. User Interaction: The program presents a menu where users can choose to add, edit, delete, or view tasks. Each option prompts the user for specific details or actions related to managing tasks.

4. Task Operations:

- Add Task: Users enter details for a new task, which is then added to the task list.

- Edit Task: Users select a task by its ID and can modify its details such as title, description, due date, or priority.

- Delete Task: Users specify a task ID to remove it from the task list.

- **View Tasks**: Displays all tasks currently stored, showing their details.

5. Main Function: The main function initializes the program, loads tasks from a file, displays the menu for user interaction, and handles user inputs accordingly.

6. Simple and Efficient: This approach ensures that users can manage their tasks efficiently within a straightforward

console-based application, suitable for personal or small team use cases.

TESTING AND RESULTING

TEST CASES

- Test case 1**:** Add a new task with valid inputs (title, description, due date, priority)
- Test Case 2:Attempt to add more tasks than the system limit
- Test Case 3:Add a task with minimal inputs (e.g., only title and priority).

Test Case 4: Edit an existing task with valid new inputs.

- Test Case 5: Edit a task that does not exist (invalid task ID).
- Test Case 6: Delete an existing task.
- Test Case 7: Delete a task that does not exist (invalid task ID).
- Test Case 8: View all tasks when there are tasks stored.
- Test Case 9: View tasks when no tasks are stored.
- Test Case 10: Input invalid choice in the menu.
- Test Case 11: Load tasks from a file with existing data.
- Test Case 12: Save tasks to a file before exiting the program.
- Test Case 13:Add tasks up to the system limit (100 tasks).

PROGRAM CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct {
    int id;
    char title[100];
    char description[255];
    char dueDate[11];
    int priority;
} Task;
Task tasks[100];
int taskCount = 0;
void loadTasks();
void saveTasks();
void addTask();
void editTask();
void deleteTask();
void viewTasks();
void displayMenu();
int main() {
    loadTasks();
    displayMenu();
    saveTasks();
    return 0;
}
void loadTasks() {
```

```c
}
void loadTasks() {
    FILE *file = fopen("tasks.txt", "r");
    if (file == NULL) {
        return;
    }
    while (fscanf(file, "%d %[^\n] %[^\n] %s %d\n", &tasks[taskCount].id,
        tasks[taskCount].title, tasks[taskCount].description, tasks[taskCount]
        .dueDate, &tasks[taskCount].priority) != EOF) {
        taskCount++;
    }
    fclose(file);
}
void saveTasks() {
    FILE *file = fopen("tasks.txt", "w");
    if (file == NULL) {
        printf("Error saving tasks.\n");
        return;
    }
    for (int i = 0; i < taskCount; i++) {
        fprintf(file, "%d %s %s %s %d\n", tasks[i].id, tasks[i].title,
            tasks[i].description, tasks[i].dueDate, tasks[i].priority);
    }
    fclose(file);
}
void addTask() {
```

```c
void addTask() {
    if (taskCount >= 100) {
        printf("Task limit reached.\n");
        return;
    }
    printf("Enter title: ");
    scanf(" %[^\n]", tasks[taskCount].title);

    printf("Enter description: ");
    scanf(" %[^\n]", tasks[taskCount].description);

    printf("Enter due date (YYYY-MM-DD): ");
    scanf("%s", tasks[taskCount].dueDate);

    printf("Enter priority (1 for highest, larger numbers for lower priorities
        ): ");
    scanf("%d", &tasks[taskCount].priority);

    tasks[taskCount].id = taskCount + 1;
    taskCount++;

    printf("Task added successfully.\n");
}
void editTask() {
    int taskId, found = 0;
    printf("Enter task ID to edit: ");
```

```c
    scanf("%d", &taskId);

    for (int i = 0; i < taskCount; i++) {
        if (tasks[i].id == taskId) {
            printf("Enter new title: ");
            scanf(" %[^\n]", tasks[i].title);

            printf("Enter new description: ");
            scanf(" %[^\n]", tasks[i].description);

            printf("Enter new due date (YYYY-MM-DD): ");
            scanf("%s", tasks[i].dueDate);

            printf("Enter new priority: ");
            scanf("%d", &tasks[i].priority);

            printf("Task updated successfully.\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Task not found.\n");
    }
}
```

```c
}
void deleteTask() {
    int taskId, found = 0;
    printf("Enter task ID to delete: ");
    scanf("%d", &taskId);

    for (int i = 0; i < taskCount; i++) {
        if (tasks[i].id == taskId) {
            for (int j = i; j < taskCount - 1; j++) {
                tasks[j] = tasks[j + 1];
            }
            taskCount--;

            printf("Task deleted successfully.\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Task not found.\n");
    }
}
void viewTasks() {
    if (taskCount == 0) {
        printf("No tasks available.\n");
```

```
122         printf("No tasks available.\n");
123         return;
124     }
125 
126     printf("Tasks:\n");
127     for (int i = 0; i < taskCount; i++) {
128         printf("ID: %d\n", tasks[i].id);
129         printf("Title: %s\n", tasks[i].title);
130         printf("Description: %s\n", tasks[i].description);
131         printf("Due Date: %s\n", tasks[i].dueDate);
132         printf("Priority: %d\n", tasks[i].priority);
133         printf("----------------------\n");
134     }
135 }
136 void displayMenu() {
137     int choice;
138 
139     do {
140         printf("\nTask Management System\n");
141         printf("======================\n");
142         printf("1. Add Task\n");
143         printf("2. Edit Task\n");
144         printf("3. Delete Task\n");
145         printf("4. View Tasks\n");
146         printf("5. Exit\n");
147         printf("======================\n");
```

```
145         printf("4. View Tasks\n");
146         printf("5. Exit\n");
147         printf("======================\n");
148         printf("Enter your choice: ");
149         scanf("%d", &choice);
150 
151         switch (choice) {
152             case 1:
153                 addTask();
154                 break;
155             case 2:
156                 editTask();
157                 break;
158             case 3:
159                 deleteTask();
160                 break;
161             case 4:
162                 viewTasks();
163                 break;
164             case 5:
165                 printf("Exiting...\n");
166                 break;
167             default:
168                 printf("Invalid choice. Please try again.\n");
169         }
170     } while (choice != 5);
```

OUTPUT

```
Task Management System
======================
1. Add Task
2. Edit Task
3. Delete Task
4. View Tasks
5. Exit
======================
Enter your choice: 1
Enter title: STUDENTS NAME
Enter description: LIST OUT STUDENTS NAME IN AN ORDER
Enter due date (YYYY-MM-DD): 2024-08-01
Enter priority (1 for highest, larger numbers for lower priorities): 1
Task added successfully.

Task Management System
======================
1. Add Task
2. Edit Task
3. Delete Task
4. View Tasks
5. Exit
======================
Enter your choice: 2
```

```
Enter your choice: 2
Enter task ID to edit: 1
Enter new title: TEACHERS NAME
Enter new description: LIST OUT TEACHERS NAME IN AN ORDER
Enter new due date (YYYY-MM-DD): 2024-09-01
Enter new priority: 1 GAYATHRI 2 GANGA 3 LAKSHMI 4 NEHA 5 NAYANA 6 LAIMY
Task updated successfully.

Task Management System
======================
1. Add Task
2. Edit Task
3. Delete Task
4. View Tasks
5. Exit
```

CONCLUTIONS

The Task Management System developed in C represents a practical solution for organizing and managing tasks effectively. This project aimed to provide users with essential functionalities to handle task creation, modification, deletion, and viewing through a straightforward console-based interface. Key achievements include robust file handling for data persistence and a user-friendly menu-driven system for seamless task management

FUTURE ENHANCEMENTS

Future Enhancements for Task Management System

1. User Authentication and Permissions:

Enhancement: Implement user authentication to support multiple users accessing personalized task lists.

Benefits: Enhances security and privacy, allowing users to manage tasks individually or collaboratively with assigned permissions.

2. Task Prioritization and Sorting:

Enhancement: Introduce prioritization features to categorize tasks based on urgency or importance.

Benefits: Helps users focus on critical tasks, improves task organization, and facilitates efficient task completion.

3. Data Visualization and Analytics:

Enhancement: Integrate graphical representations and analytics to visualize task progress, deadlines, and workload.

Benefits: Provides users with insightful metrics and trends, aiding in decision-making and optimizing productivity.

4. Reminders and Notifications:

Enhancement: Implement reminder and notification features to alert users about upcoming task deadlines or overdue tasks.

Benefits:Improves task management efficiency by ensuring timely actions and reducing the risk of missed deadlines.

5. Collaborative Task Management:

Enhancement: Enable collaborative functionalities such as task assignment, shared task lists, and real-time updates.

Benefits:Facilitates teamwork and project coordination, enhances communication among team members, and supports collective goal achievement.

6. Integration with External Tools

Enhancement: Integrate with external productivity tools or calendar applications to synchronize tasks and schedules.

Benefits Enhances interoperability, streamlines workflow management, and provides users with seamless access to task-related information across platforms.

7. Mobile and Web Compatibility:

Enhancement: Develop mobile-friendly or web-based versions of the Task Management System for accessibility from different devices.

Benefits: Increases flexibility, allowing users to manage tasks on the go, and expands user reach beyond desktop environments.

8. Advanced Search and Filtering Options:

Enhancement: Enhance search capabilities with advanced filtering options based on task attributes (e.g., date range, priority).

Benefits Facilitates quick retrieval of specific tasks, improves task organization, and supports efficient data management.

9. **User Feedback Mechanism:**

- **Enhancement:** Implement a feedback mechanism to gather user input and suggestions for continuous improvement.

- **Benefits:** Engages users in product development, identifies areas for enhancement, and ensures alignment with user needs and expectationS

These future enhancements aim to expand the functionality, usability, and utility of the Task Management System,

catering to diverse user requirements and improving overall task management efficiency and effectiveness.

REFERENCES

By referring google , testbooks