

# PROJECT TRACKING SYSTEM

**Mini Project**



**Name :Tom Vinod**

**Roll NO. :70**

**Course Name:** C programming

**Date :** 12-07-2024

# INTRODUCTION

## Project overview:

## Problem statement:

This project implements a simple project management system in C. It allows users to add tasks and assign them to team members, register new team members, view all tasks along with their assigned team members and status, and mark tasks as complete. The system is menu-driven, easy to use, and helps manage project tasks and team assignments effectively.

## Objective:

The objective of this project is to develop a simple, menu-driven project management system in C for adding, assigning, viewing, and tracking tasks and team members.

# System Requirements

Minimum Requirements for C Programming Code to Run:

### Hardware Requirement:

* + A computer with at least 4GB RAM
  + 500MB of free disk space

### Software Requirement:

* + **Operating System:** Windows/Linux/MacOS
  + **Compiler:** GCC or any C compiler
  + **IDE**: Code: Blocks, Dev-C++, or any C IDE

# Design and Development

### Program Logic:

The program is designed to manage project tasks and team members using a simple menu-driven interface in C. It employs two primary data structures: Task, which stores details of each task (name, assigned team member, and completion status), and Project, which stores the project name, an array of tasks, and the count of tasks. The global variables include an instance of the Project structure named project, a 2D array teamMembers to store team member names, and teamMemberCount to track the number of team members. The program includes several functions: addTask for adding a new task to the project, addTeamMember for adding a new team member, viewTasks for displaying all tasks along with their details, and markTaskAsComplete for marking a specific task as complete. The menu function provides a menu-driven interface allowing the user to interact with the system by choosing options to add tasks, add team members, view tasks, mark tasks as complete, or exit the program. The main function initializes the project by prompting the user for the project name, sets the initial task count to zero, and calls the menu function to start the interface..

#### Data Structures

* **Task**: Stores task details, including name, assigned team member, and completion status.
* **Project**: Stores project name, an array of tasks, and task count.

#### Global Variables

* **project**: Instance of Project structure.
* **teamMembers**: 2D array to store team member names.
* **teamMemberCount**: Counter for the number of team members.

#### Functions Used

* **addTask**: Adds a new task to the project.
* **addTeamMember**: Adds a new team member to the project.
* **viewTasks**: Displays all tasks with their details.
* **markTaskAsComplete**: Marks a specified task as complete.
* **menu**: Provides a menu-driven interface for user interaction.
* **main**: Initializes the project and starts the menu interface.

### Pseudocode:

Start

Define structures:

Task:

name: string[50]

assignedTo: string[50]

completed: integer

Project:

name: string[50]

tasks: array of Task[10]

taskCount: integer

Declare global variables:

project: Project

teamMembers: array of string[10][50]

teamMemberCount: integer = 0

Function addTask():

If project.taskCount >= 10:

Print "Task limit reached."

Return

Print "Enter task name: "

Read project.tasks[project.taskCount].name

Print "Enter team member to assign task: "

Read project.tasks[project.taskCount].assignedTo

Set project.tasks[project.taskCount].completed to 0

Increment project.taskCount

Print "Task added successfully."

Function addTeamMember():

If teamMemberCount >= 10:

Print "Team member limit reached."

Return

Print "Enter team member name: "

Read teamMembers[teamMemberCount]

Increment teamMemberCount

Print "Team member added successfully."

Function viewDetails():

For each task in project.tasks up to project.taskCount:

Print "Task %d: %s", index + 1, task.name

Print "Assigned to: %s", task.assignedTo

Print "Status: %s", (task.completed ? "Completed" : "Incomplete")

If teamMemberCount != 0:

Print "Additional Team Members:"

For each member in teamMembers up to teamMemberCount:

Print member

Function viewTasks():

For each task in project.tasks up to project.taskCount:

Print "Task %d: %s", index + 1, task.name

Function markTaskAsComplete():

Call viewTasks()

Print "Enter task number to mark as complete: "

Read taskNumber

If taskNumber < 1 or taskNumber > project.taskCount:

Print "Invalid task number."

Return

Set project.tasks[taskNumber - 1].completed to 1

Print "Task marked as complete."

Main function:

Print "Enter project name: "

Read project.name

Set project.taskCount to 0

Initialize choice to 0

Repeat until choice equals 5:

Print "Menu:"

Print "1. Add Task"

Print "2. Add Team Member"

Print "3. View Details"

Print "4. Mark Task as Complete"

Print "5. Exit"

Print "Enter your choice: "

Read choice

Switch choice:

Case 1:

Call addTask()

Break

Case 2:

Call addTeamMember()

Break

Case 3:

Call viewDetails()

Break

Case 4:

Call markTaskAsComplete()

Break

Case 5:

Print "EXITING..."

Break

Default:

Print "Invalid choice, please try again."

Return 0

End

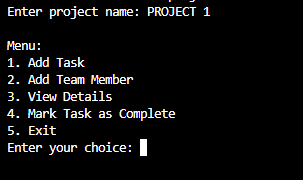
# Testing and Results

### Test cases:

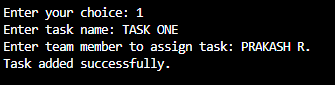
1. **Adding Tasks:**
   * Add task with valid details.
   * Attempt to add task beyond limit.
2. **Adding Team Members:**
   * Add team member with valid name.
   * Attempt to add team member beyond limit.
3. **Viewing Task Details:**
   * View details with existing tasks and team members.
   * View details with no tasks or team members.
4. **Marking Tasks as Complete:**
   * Mark task as complete with valid task number.
   * Attempt to mark task with invalid task number.
5. **Menu Navigation and Exiting:**
   * Navigate menu options and exit program.
   * Test invalid menu option handling.

These concise test cases cover essential functionalities and boundary conditions of the project management system, ensuring robust behavior across different scenarios.

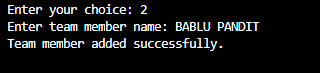
Output:



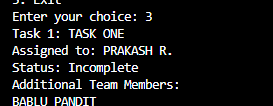
**Test case 1:**



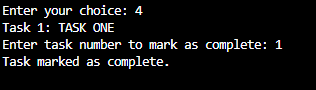
**Test case 2:**

****

**Test case 3:**

****

**Test case 4:**

****

**Test case 5:**

****

**Test case 6:**

****

# Discussion of results

The code efficiently manages project tasks and team members, allowing dynamic additions and updates with clear user interactions, ensuring operational limits are respected.

Conclusion

### Summary of the Project:

### The project implements a task management system using C, featuring task addition, assignment, completion marking, and team member management. It uses structures for tasks and projects, with limits on task and team member counts, providing a menu-driven interface for user interaction and project details display.

### Future Enhancements:

### Potential enhancements include file I/O for data persistence, task priority management, and user authentication for secure project access.

# References

[Stack Overflow - Where Developers Learn, Share, & Build Careers](https://stackoverflow.com/)

[GeeksforGeeks | A computer science portal for geeks](https://www.geeksforgeeks.org/)

[Microsoft Copilot in Bing](https://www.bing.com/chat?form=NTPCHB)

# Appendices

### Source code:

#include <stdio.h>

#include <string.h>

// Define Task and Project structures

typedef struct {

    char name[50];

    char assignedTo[50];

    int completed;

} Task;

typedef struct {

    char name[50];

    Task tasks[10];

    int taskCount;

} Project;

Project project;

char teamMembers[10][50];

int teamMemberCount = 0;

// Function to add a task to the project

void addTask() {

    if (project.taskCount >= 10) {

        printf("Task limit reached.\n");

        return;

    }

    // Get task details from the user

    printf("Enter task name: ");

    scanf(" %[^\n]s", project.tasks[project.taskCount].name);

    printf("Enter team member to assign task: ");

    scanf(" %[^\n]s", project.tasks[project.taskCount].assignedTo);

    project.tasks[project.taskCount].completed = 0;

    project.taskCount++;

    printf("Task added successfully.\n");

}

// Function to add a team member to the project

void addTeamMember() {

    if (teamMemberCount >= 10) {

        printf("Team member limit reached.\n");

        return;

    }

    // Get team member name from the user

    printf("Enter team member name: ");

    scanf(" %[^\n]s", teamMembers[teamMemberCount]);

    teamMemberCount++;

    printf("Team member added successfully.\n");

}

// Function to view details of the project

void viewDetails() {

    for (int i = 0; i < project.taskCount; i++) {

        printf("Task %d: %s\n", i + 1, project.tasks[i].name);

        printf("Assigned to: %s\n", project.tasks[i].assignedTo);

        printf("Status: %s\n", project.tasks[i].completed ? "Completed" : "Incomplete");

    }

    if (teamMemberCount != 0) {

        printf("Additional Team Members:\n");

        for (int i = 0; i < teamMemberCount; i++) {

            printf("%s\n", teamMembers[i]);

        }

    }

}

// Function to view only the task names

void viewTasks() {

    for (int i = 0; i < project.taskCount; i++) {

        printf("Task %d: %s\n", i + 1, project.tasks[i].name);

    }

}

// Function to mark a task as complete

void markTaskAsComplete() {

    viewTasks();

    int taskNumber;

    printf("Enter task number to mark as complete: ");

    scanf("%d", &taskNumber);

    if (taskNumber < 1 || taskNumber > project.taskCount) {

        printf("Invalid task number.\n");

        return;

    }

    project.tasks[taskNumber - 1].completed = 1;

    printf("Task marked as complete.\n");

}

int main() {

    // Initialize the project

    printf("Enter project name: ");

    scanf(" %[^\n]s", project.name);

    project.taskCount = 0;

    int choice;

    do {

        // Print the menu options

        printf("\nMenu:\n");

        printf("1. Add Task\n");

        printf("2. Add Team Member\n");

        printf("3. View Details\n");

        printf("4. Mark Task as Complete\n");

        printf("5. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        // Handle the user's menu choice

        switch (choice) {

            case 1:

                addTask();

                break;

            case 2:

                addTeamMember();

                break;

            case 3:

                viewDetails();

                break;

            case 4:

                markTaskAsComplete();

                break;

            case 5:

                printf("EXITING...\n");

                break;

            default:

                printf("invalid choice, please try again.\n");

        }

    } while (choice != 5);

    return 0;

}