# Task Management and Productivity Tracker

Course: Introduction to Problem Solving & Programming
Student: Kuval Dayal
Reg. No.: 25BAI11610

## 1. Abstract

This project is a simple menu-based Task Management System using only the basic concepts of Python programming. The aim was to create a clean, working program that encapsulates the essence of problem-solving, modularity, and structured design principles. A user will be able to add tasks, list them, mark them as completed, and delete them within the system. This is a deliberately simplified project; clarity, not unwarranted complexity, is the objective.

## 2. Introduction

Even the management of everyday personal matters becomes inefficient without a systematic way of keeping track of such tasks. Although there are many suitable digital tools for this, making a small solution of one's own is an excellent opportunity to apply step-wise refinement, modular programming, and algorithmic thinking-all corner stones of this course.
This application only uses core Python features, and the code has been written to remain easy to understand, test, and extend.

## 3. Problem Statement

Users struggle a lot with keeping track of basic tasks, either missing deadlines or just generally being disorganized. A simple system to keep track of tasks and their status will go a long way toward smoothing personal workflow and increasing productivity. The problem being solved is: How can we make a lightweight, easy-to-use program for users to manage their tasks more effectively?

## 4. Objectives

The project aspires to:
Create a simple and functional task manager.
Demonstrate foundational problem-solving techniques.
Apply modular program design using functions and classes.
Keep code readable and simple.

## 5. Methodology

### 5.1 Modular Design

The program is divided into two files:
main.py – menu display, user input, program loop.
task_manager.py – core logic for adding, viewing, updating, and deleting tasks

### 5.2 Data Structure

Tasks are stored in the form of a list of dictionaries, each containing a description and completion status.

### 5.3 Algorithms (Summary)

Add Task:
Accept user input
Append a new dictionary to the task list
View Tasks:
Iterate over task list
Display index, description, and status
Mark Completed:
Validate task number
Update completion status
Delete Task:
Validate task number
Remove from list

## 6. System Design Diagram (Flow Overview)

Menu → User Choice → (Add / View / Complete / Delete / Exit)

Each of the options results in a simple function call in the TaskManager class.
The program continues until the user chooses to exit.

## 7. Implementation

This system was implemented in Python, using basic constructs like loops, lists, conditionals, and simple input/output. No external libraries were used, which keeps the implementation in tune with the course objectives.
Screenshots of the program in action are included in a separate directory /screenshots.

## 8. Testing

All functions were tested with:
Empty task list
Numerous tasks
Normal user flow
Expected behavior was observed in all the application scenarios.

## 9. Conclusion

This project shows that even for simple problems, one can adopt a structured approach. The final implementation is readable, functional, and understandable — which is actually the real target for an introductory programming course. It reinforces modular design, algorithmic thinking, and iterative problem-solving without unnecessary complexity.