AI-Driven Task Management Tool [Python Project Report]

INTRODUCTION:

In today's fast-paced world, efficient task management is essential for maximizing productivity and minimizing stress. Traditional to-do lists and planners often fall short when it comes to dynamically organizing, prioritizing, and adapting tasks based on changing schedules, urgency, and workload. To address these challenges, this project presents an AI-Driven Task Management Tool developed using Python.

The tool allows users to add tasks with attributes such as descriptions, due dates, and priority levels, and supports marking tasks as completed. It also simulates AI-based prioritization by sorting tasks based on urgency and importance.

☑ Why This Code is Useful for the Future:

1. Scalable Foundation – Provides a solid base to build advanced task management apps with AI features like NLP and smart scheduling.
2. Boosts Productivity – Automates task prioritization, helping users save time and stay organized.
3. AI Integration Ready – Designed for future enhancements with machine learning, natural language input, and predictive task handling
4. Customizable & Open Source – Easily modifiable for personal, academic, or enterprise use thanks to Python’s flexibility.
5. Practical Learning Tool – Ideal for understanding real-world AI applications in personal productivity tools.

PYTHON CODE:

```
import datetime
import json
import os
from colorama import init, Fore, Style

init(autoreset=True)

TASK_FILE = "tasks.json"

class Task:
    def __init__(self, description, due_date=None, priority="medium", completed=False):
        self.description = description
        self.due_date = due_date
        self.priority = priority
        self.completed = completed

    def to_dict(self):
        return {
            "description": self.description,
            "due_date": self.due_date.strftime('%Y-%m-%d') if self.due_date else None,
            "priority": self.priority,
            "completed": self.completed
        }

    @classmethod
    def from_dict(cls, data):
        due_date = datetime.datetime.strptime(data['due_date'], '%Y-%m-%d').date() if data['due_date'] else None
        return cls(data['description'], due_date, data['priority'], data['completed'])
```

```
    def __str__(self):
        status = Fore.GREEN + "Completed" if self.completed else Fore.YELLOW + "Pending"
        due_info = f" (Due: {self.due_date.strftime('%Y-%m-%d')})" if self.due_date else ""
        priority_color = {"high": Fore.RED, "medium": Fore.BLUE, "low": Fore.CYAN}
        return f"- {self.description} [{priority_color.get(self.priority, Fore.WHITE)}Priority: {self.priority}]{due_info} [{status}]"


class TaskManager:
    def __init__(self):
        self.tasks = []
        self.load_tasks()


    def save_tasks(self):
        with open(TASK_FILE, 'w') as f:
            json.dump([task.to_dict() for task in self.tasks], f)


    def load_tasks(self):
        if os.path.exists(TASK_FILE):
            with open(TASK_FILE, 'r') as f:
                self.tasks = [Task.from_dict(t) for t in json.load(f)]


    def add_task(self, description, due_date_str=None, priority="medium"):
        if not description.strip():
            print("Description cannot be empty.")
            return
        due_date = None
        if due_date_str:
            try:
```

```
            due_date = datetime.datetime.strptime(due_date_str, '%Y-%m-%d').date()
        except ValueError:
            print("Invalid date format. Please use YYYY-MM-DD.")
            return
    if priority not in ["high", "medium", "low"]:
        priority = "medium"
    task = Task(description, due_date, priority)
    self.tasks.append(task)
    self.save_tasks()
    print(f"Task '{description}' added.")

def view_tasks(self):
    if not self.tasks:
        print("No tasks to display.")
        return
    print("\n--- Your Tasks ---")
    for i, task in enumerate(self.tasks):
        print(f"{i+1}. {task}")
    print("--------------------")

def mark_completed(self, task_index):
    if 0 <= task_index < len(self.tasks):
        self.tasks[task_index].completed = True
        self.save_tasks()
        print(f"Task '{self.tasks[task_index].description}' marked as completed.")
    else:
        print("Invalid task number.")

def delete_task(self, task_index):
    if 0 <= task_index < len(self.tasks):
```

```
            removed = self.tasks.pop(task_index)
            self.save_tasks()
            print(f"Task '{removed.description}' deleted.")
        else:
            print("Invalid task number.")

    def edit_task(self, task_index, new_desc=None, new_due_date=None,
new_priority=None):
        if 0 <= task_index < len(self.tasks):
            task = self.tasks[task_index]
            if new_desc:
                task.description = new_desc
            if new_due_date:
                try:
                    task.due_date = datetime.datetime.strptime(new_due_date, '%Y-%m-
%d').date()
                except ValueError:
                    print("Invalid date format. Skipping date change.")
            if new_priority in ["high", "medium", "low"]:
                task.priority = new_priority
            self.save_tasks()
            print(f"Task '{task.description}' updated.")
        else:
            print("Invalid task number.")

    def prioritize_tasks_ai(self):
        print("\n--- Prioritizing tasks (AI-driven simulation) ---")
        self.tasks.sort(key=lambda t: (
            {"high": 0, "medium": 1, "low": 2}.get(t.priority, 1),
            t.due_date if t.due_date else datetime.date.max
        ))
```

```
        self.view_tasks()
        print("----------------------------------------")

def main():
    manager = TaskManager()

    while True:
        print("\n--- Task Manager Menu ---")
        print("1. Add Task")
        print("2. View Tasks")
        print("3. Mark Task as Completed")
        print("4. Prioritize Tasks (AI-driven)")
        print("5. Edit Task")
        print("6. Delete Task")
        print("7. Exit")

        choice = input("Enter your choice: ").strip()

        if choice == '1':
            description = input("Enter task description: ")
            due_date_str = input("Enter due date (YYYY-MM-DD, optional): ")
            priority = input("Enter priority (high, medium, low): ").lower()
            manager.add_task(description, due_date_str, priority)
        elif choice == '2':
            manager.view_tasks()
        elif choice == '3':
            manager.view_tasks()
            try:
                task_num = int(input("Enter task number to mark as completed: "))
                manager.mark_completed(task_num - 1)
```

```
            except ValueError:
                print("Please enter a valid number.")
        elif choice == '4':
            manager.prioritize_tasks_ai()
        elif choice == '5':
            manager.view_tasks()
            try:
                index = int(input("Enter task number to edit: ")) - 1
                new_desc = input("New description (leave blank to skip): ")
                new_date = input("New due date (YYYY-MM-DD, leave blank to skip): ")
                new_priority = input("New priority (high/medium/low, leave blank to skip):
").lower()
                manager.edit_task(index, new_desc or None, new_date or None,
new_priority or None)
            except ValueError:
                print("Invalid input.")
        elif choice == '6':
            manager.view_tasks()
            try:
                task_num = int(input("Enter task number to delete: "))
                manager.delete_task(task_num - 1)
            except ValueError:
                print("Invalid input.")
        elif choice == '7':
            print("Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

CODE OUTPUT:

```
--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 1
Enter task description: Finish project report
Enter due date (YYYY-MM-DD, optional): 2025-06-25
Enter priority (high, medium, low): high
Task 'Finish project report' added.

--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 1
Enter task description: Buy groceries
Enter due date (YYYY-MM-DD, optional): 2025-06-20
Enter priority (high, medium, low): low
Task 'Buy groceries' added.

--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 2

--- Your Tasks ---
1. - Finish project report [Priority: high] (Due: 2025-06-25) [Pending]
2. - Buy groceries [Priority: low] (Due: 2025-06-20) [Pending]
------------------


 --- Task Manager Menu ---
 1. Add Task
 2. View Tasks
 3. Mark Task as Completed
 4. Prioritize Tasks (AI-driven)
 5. Edit Task
 6. Delete Task
 7. Exit
 Enter your choice: 3

 --- Your Tasks ---
 1. - Finish project report [Priority: high] (Due: 2025-06-25) [Pending]
 2. - Buy groceries [Priority: low] (Due: 2025-06-20) [Pending]
 ------------------
 Enter task number to mark as completed: 2
 Task 'Buy groceries' marked as completed.

 --- Task Manager Menu ---
 1. Add Task
 2. View Tasks
 3. Mark Task as Completed
 4. Prioritize Tasks (AI-driven)
 5. Edit Task
 6. Delete Task
 7. Exit
 Enter your choice: 4

 --- Prioritizing tasks (AI-driven simulation) ---

 --- Your Tasks ---
 1. - Finish project report [Priority: high] (Due: 2025-06-25) [Pending]
 2. - Buy groceries [Priority: low] (Due: 2025-06-20) [Completed]
 ------------------
 ------------------------------------------
```

```
--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 5

--- Your Tasks ---
1. - Finish project report [Priority: high] (Due: 2025-06-25) [Pending]
2. - Buy groceries [Priority: low] (Due: 2025-06-20) [Completed]
------------------
Enter task number to edit: 2
New description (leave blank to skip): Complete project presentation
New due date (YYYY-MM-DD, leave blank to skip): 2025-06-24
New priority (high/medium/low, leave blank to skip): medium
Task 'Complete project presentation' updated.

--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 2

--- Your Tasks ---
1. - Finish project report [Priority: high] (Due: 2025-06-25) [Pending]
2. - Complete project presentation [Priority: medium] (Due: 2025-06-24) [Completed]
------------------
```

```
--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 6

--- Your Tasks ---
1. - Finish project report [Priority: high] (Due: 2025-06-25) [Pending]
2. - Complete project presentation [Priority: medium] (Due: 2025-06-24) [Completed]
------------------
Enter task number to delete: 2
Task 'Complete project presentation' deleted.

--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 2

--- Your Tasks ---
1. - Finish project report [Priority: high] (Due: 2025-06-25) [Pending]
------------------

--- Task Manager Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Prioritize Tasks (AI-driven)
5. Edit Task
6. Delete Task
7. Exit
Enter your choice: 7
Goodbye!
```

Websites/Apps Used:

Chatgpt ,Google ,Jupyter notebook

Modules and Libraries

Name	Type	Purpose
datetime	Built-in	Used for handling and formatting task due dates.
json	Built-in	For reading from and writing to the JSON file to store tasks persistently.
os	Built-in	Checks if the task file exists before loading data.
colorama	External	Adds color-coded output in the terminal for better task status visibility.
colorama.init()	Function	Initializes colorama (with auto-reset for colors).

Task Class Functions

Function / Method	Purpose
__init__()	Initializes a task with description, due date, priority, and completion status.
to_dict()	Converts task attributes into dictionary format (for JSON saving).
from_dict()	Class method to reconstruct a task object from a dictionary (JSON loading).

Main Program Functions

Function / Component	Purpose
main()	Handles the menu system and user interaction loop for task management.
input()	Captures user input for navigating and interacting with the program.

Applications of AI-Driven Task Management System

	Application Area	Description
1.	Personal Productivity Tools	Helps individuals prioritize daily tasks intelligently based on urgency and importance.
2.	Team Project Management	Can be extended to assign and sort tasks across team members using AI logic.
3.	Educational Task Planners	Assists students in scheduling assignments, exams, and study plans efficiently.
4.	Workplace Automation	Automates reminders and prioritizes tasks based on business-critical deadlines.
5.	Smart Assistants Integration	Can be integrated with AI virtual assistants (like Alexa or Google Assistant) to manage voice-activated task lists.

CONCLUSION

The development of an AI-driven task management tool using Python showcases how simple automation and intelligent sorting can significantly improve productivity and organization. By integrating features such as task creation, editing, deletion, and AI-simulated prioritization, the tool demonstrates practical applications of software that supports human decision-making.

Though the current implementation uses rule-based logic for prioritization, it lays the foundation for future integration of advanced AI models like natural language processing (NLP) and machine learning to predict urgency, classify tasks, and suggest schedules. This project highlights the potential of Python in building real-world productivity tools that are both efficient and user-friendly.