REPORT ON AGENTIC TO DO LIST

DONE BY: M.VISHALI(II-ECE-B)

SUBMITTED TO: MRS.GAYATHRI

1. INTRODUCTION

In today's fast-paced digital environment, effective task management is essential for productivity and time optimization. Traditional to-do list applications are static and require manual prioritization and planning, which often leads to inefficiency. To overcome these limitations, intelligent systems capable of reasoning, planning, and adapting to user behavior are required.

THIS is an **Agentic AI-based To-Do Management System** designed to intelligently manage tasks by automatically planning subtasks, tracking progress, adapting task priority, and providing productivity analytics. The system simulates **agentic behavior** using rule-based reasoning, making it reliable, explainable, and suitable for offline usage without dependency on external AI APIs.

---

2. OBJECTIVES OF THE PROJECT

The main objectives of this project are:

- To design an intelligent task management system using agent-based logic
- To automatically decompose tasks into manageable subtasks
- To track progress dynamically and update task priority
- To provide a productivity dashboard for performance evaluation
- To implement user authentication and persistent storage
- To develop a clean, user-friendly interface using Streamlit

---

3. PROBLEM STATEMENT

Most existing to-do applications suffer from the following drawbacks:

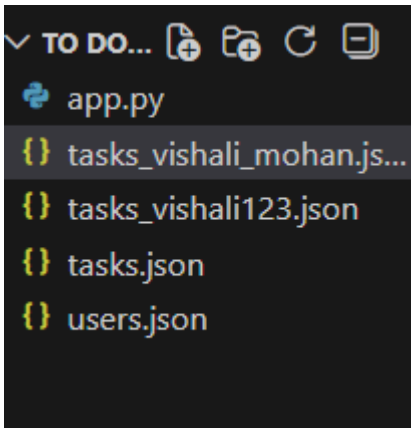
- No intelligent task planning or breakdown
- Manual priority assignment
- No adaptive behavior based on progress
- Lack of productivity analytics
- No personalized task storage for multiple users

This project aims to solve these problems by introducing an **Agentic AI system** that can autonomously reason, plan, and adapt tasks based on user interaction.

---

4. PROPOSED SYSTEM

→PROJECT STRUCTURE:



The proposed system, uses an agent-based approach where tasks are treated as goals, and subtasks are generated as actionable steps. The system continuously evaluates subtask completion to update progress, priority, and completion status.

Key Features:

- Agentic task planning (task decomposition)
- Smart priority auto-update
- Progress tracking using subtasks
- Productivity dashboard with analytics
- User authentication (login & registration)
- Persistent storage using JSON files
- Clean and interactive UI with sidebar controls

---

5. SYSTEM ARCHITECTURE

The architecture of TaskMind AI consists of the following components:

1. **User Interface (Streamlit)**
   Handles task input, visualization, and user interaction.

2. **Agent Logic Module**
   Performs task categorization, breakdown, progress calculation, and priority adaptation.
3. **Authentication Module**
   Manages user registration, login, and session handling.
4. **Persistence Layer (JSON)**
   Stores user credentials and per-user task data locally.
5. **Analytics Module**
   Generates productivity metrics and progress dashboards.
6. TECHNOLOGIES USED

Technology	Purpose
Python	Core programming language
Streamlit	Web application framework
JSON	Persistent storage
HTML/CSS (embedded)	UI styling
Rule-Based AI	Agentic decision-making

7. MODULE DESCRIPTION

7.1 User Authentication Module

- Allows users to register and login securely
- Maintains session-based authentication
- Ensures task data is user-specific

7.2 Task Management Module

- Accepts tasks in natural language
- Automatically assigns categories
- Stores tasks persistently

7.3 Agentic Planning Module

- Decomposes tasks into subtasks
- Enables structured execution of goals
- Simulates autonomous agent planning

7.4 Progress Tracking Module

- Calculates task completion percentage
- Based on subtask completion
- Prevents inaccurate or partial completion

7.5 Smart Priority Module

- Dynamically updates priority based on progress
- High → Medium → Low → Completed
- Reduces cognitive load for users

7.6 Productivity Dashboard Module

- Displays total tasks, completed tasks, and productivity score
- Uses charts and metrics for visualization
- Updates in real-time

---

8. ALGORITHM USED

Task Progress Calculation Algorithm

1. Count the total number of subtasks
2. Count completed subtasks
3. Compute progress percentage
4. If progress = 100%, mark task as completed
5. Update priority based on progress level

This ensures accurate and reliable task completion tracking.

---

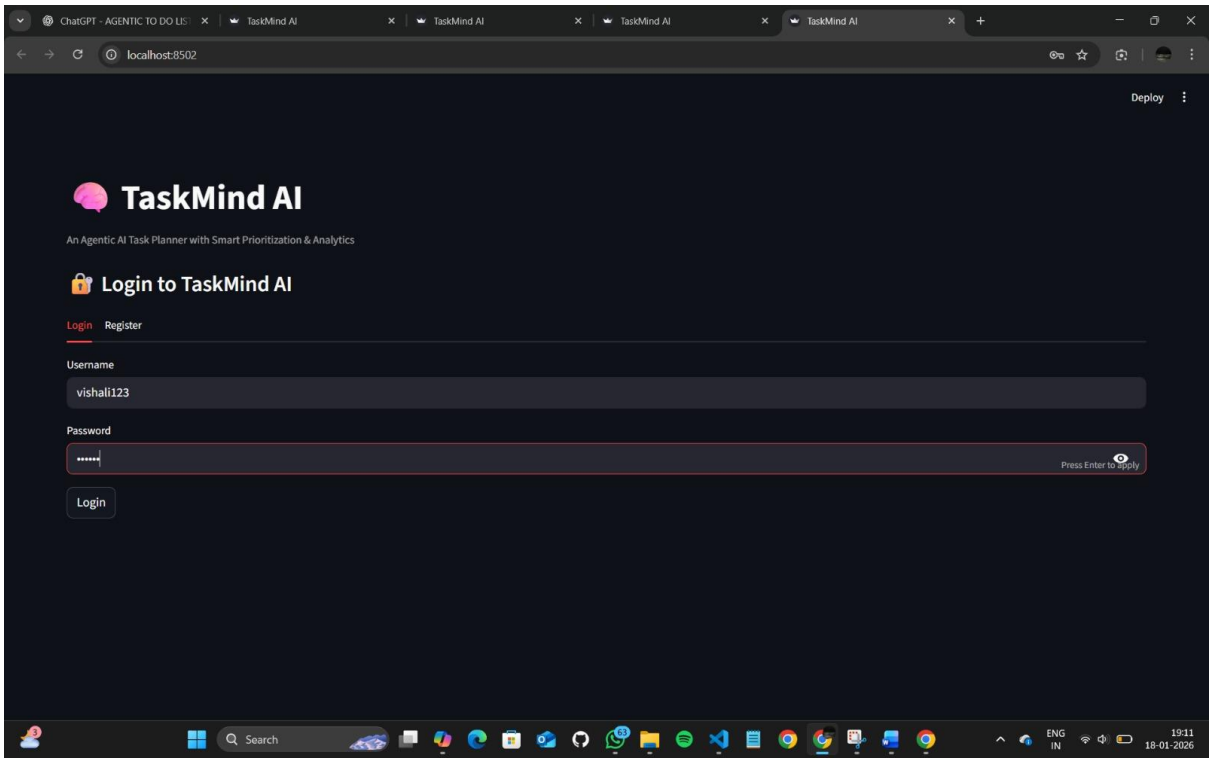
9. USER INTERFACE DESIGN

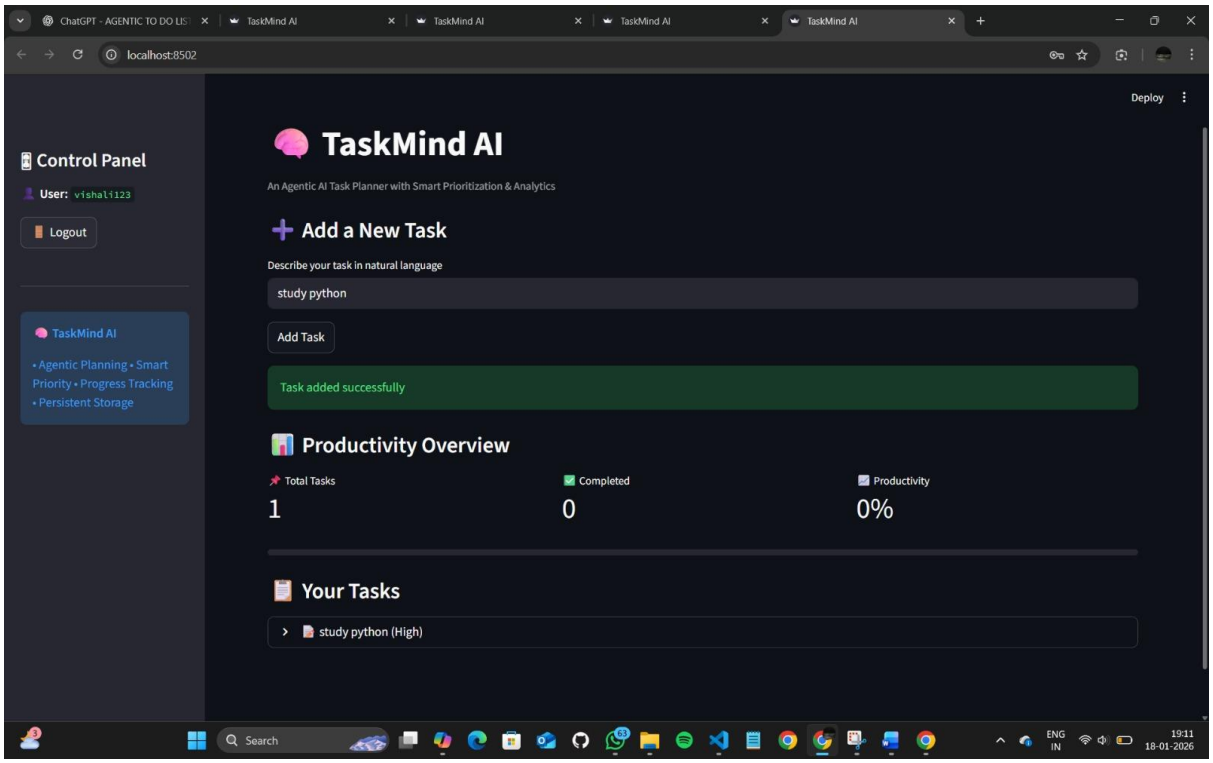
The user interface is designed with usability and clarity in mind:

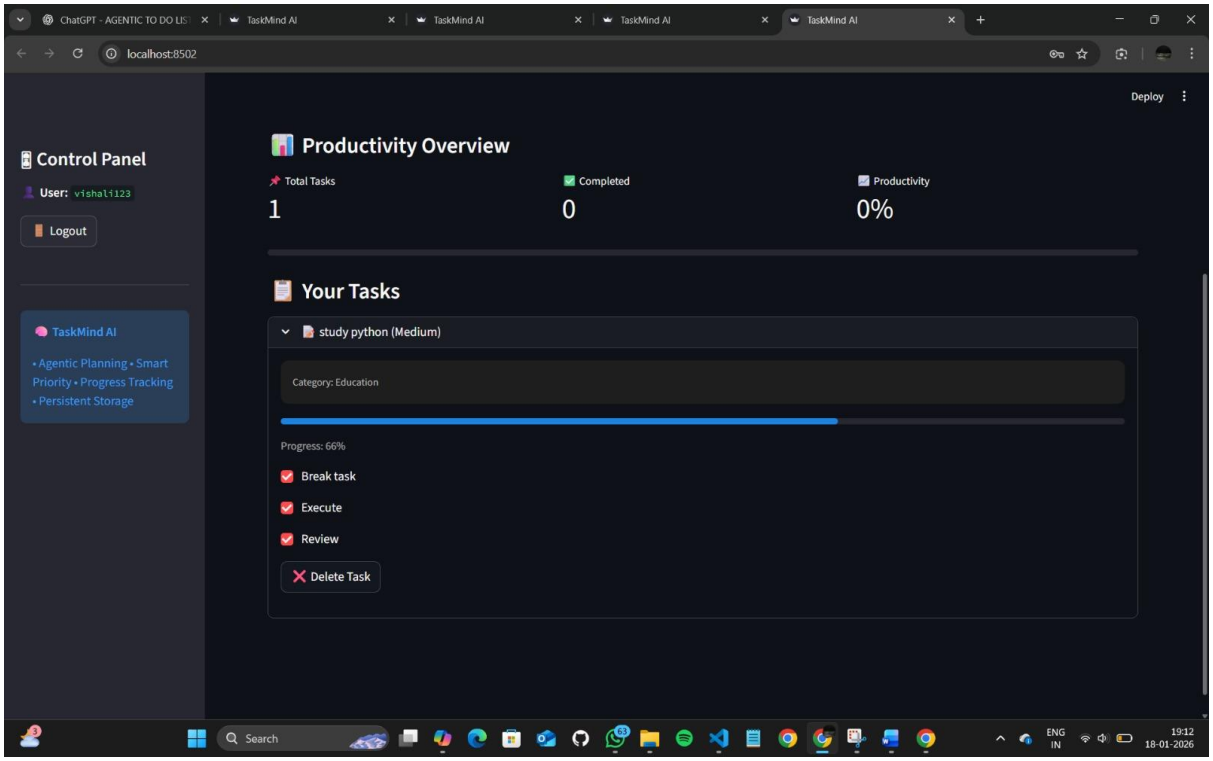
- Sidebar-based navigation
- Expandable task cards to avoid clutter
- Visual progress bars for task tracking
- Dashboard metrics for quick insights
- Dark-themed modern UI

---

10. RESULTS AND OUTPUT



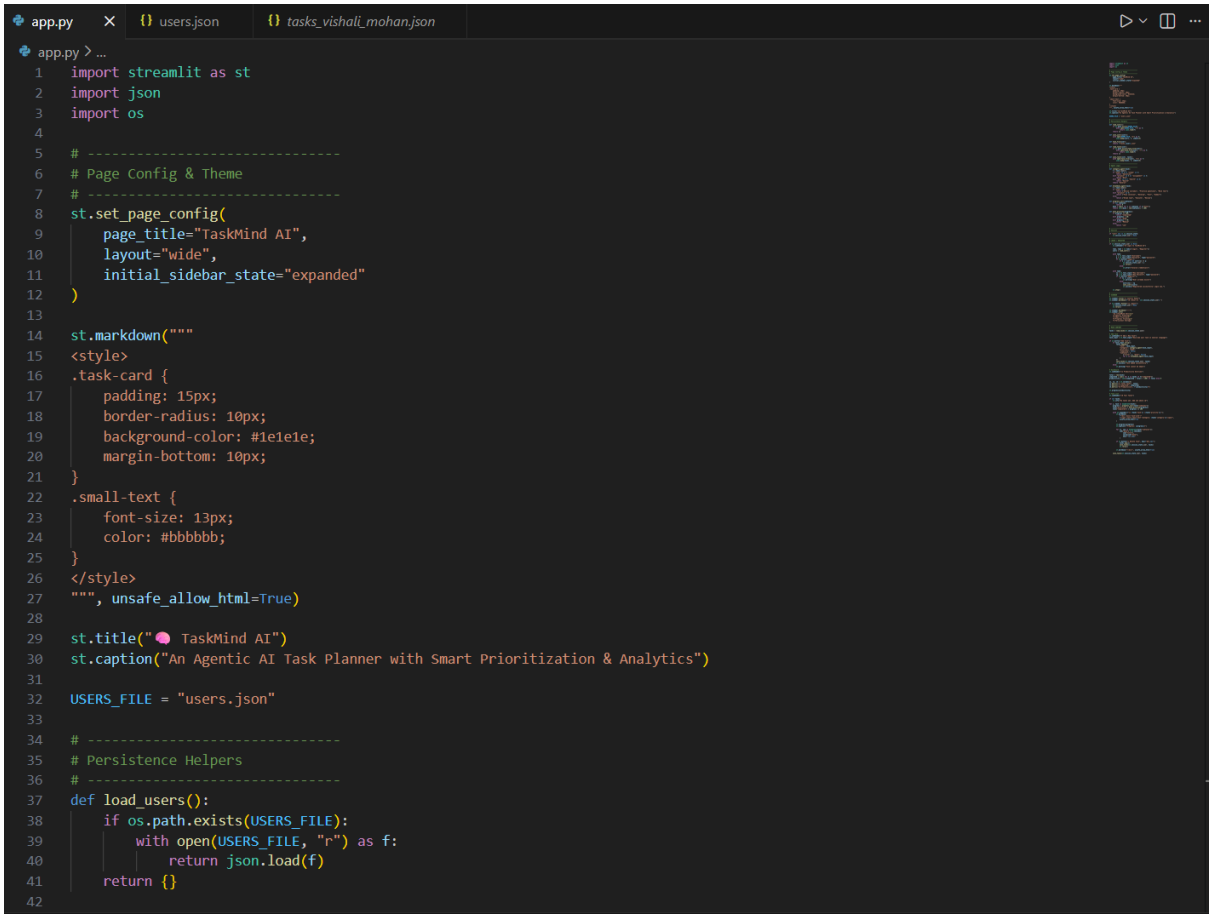


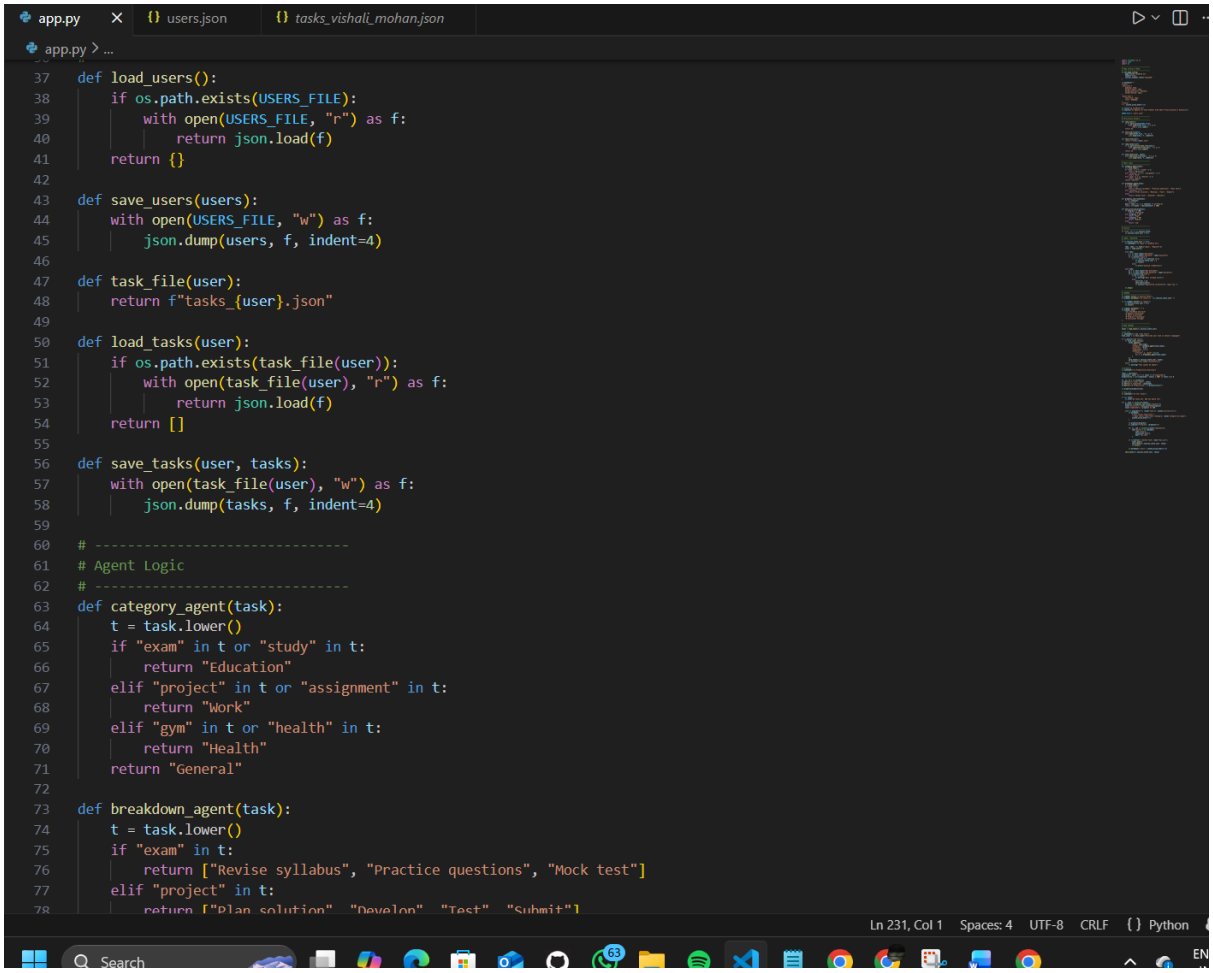


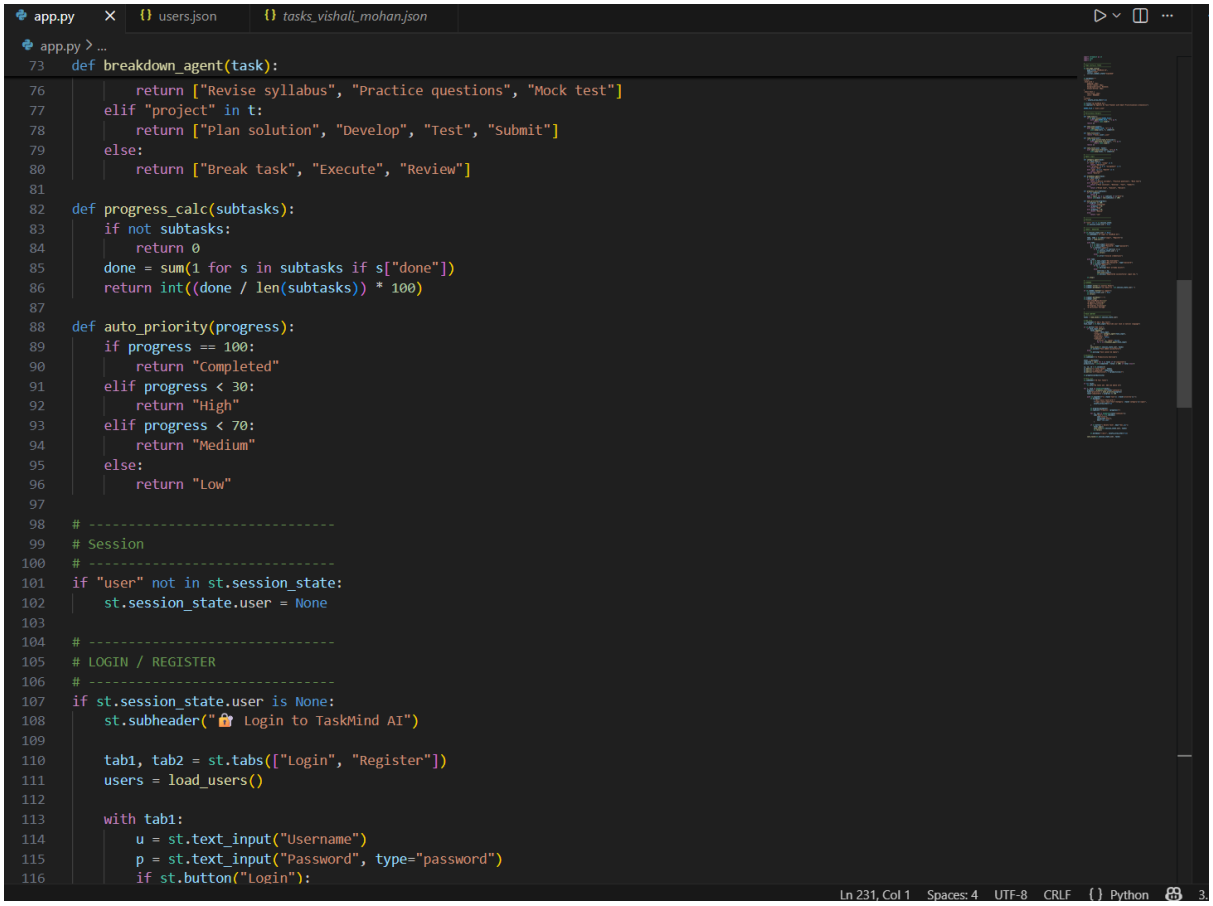
The system successfully:

- Breaks tasks into meaningful subtasks
- Tracks progress accurately up to 100%
- Automatically updates task priority
- Persists data across sessions
- Supports multiple users independently
- Provides real-time productivity analytics

→CODE:

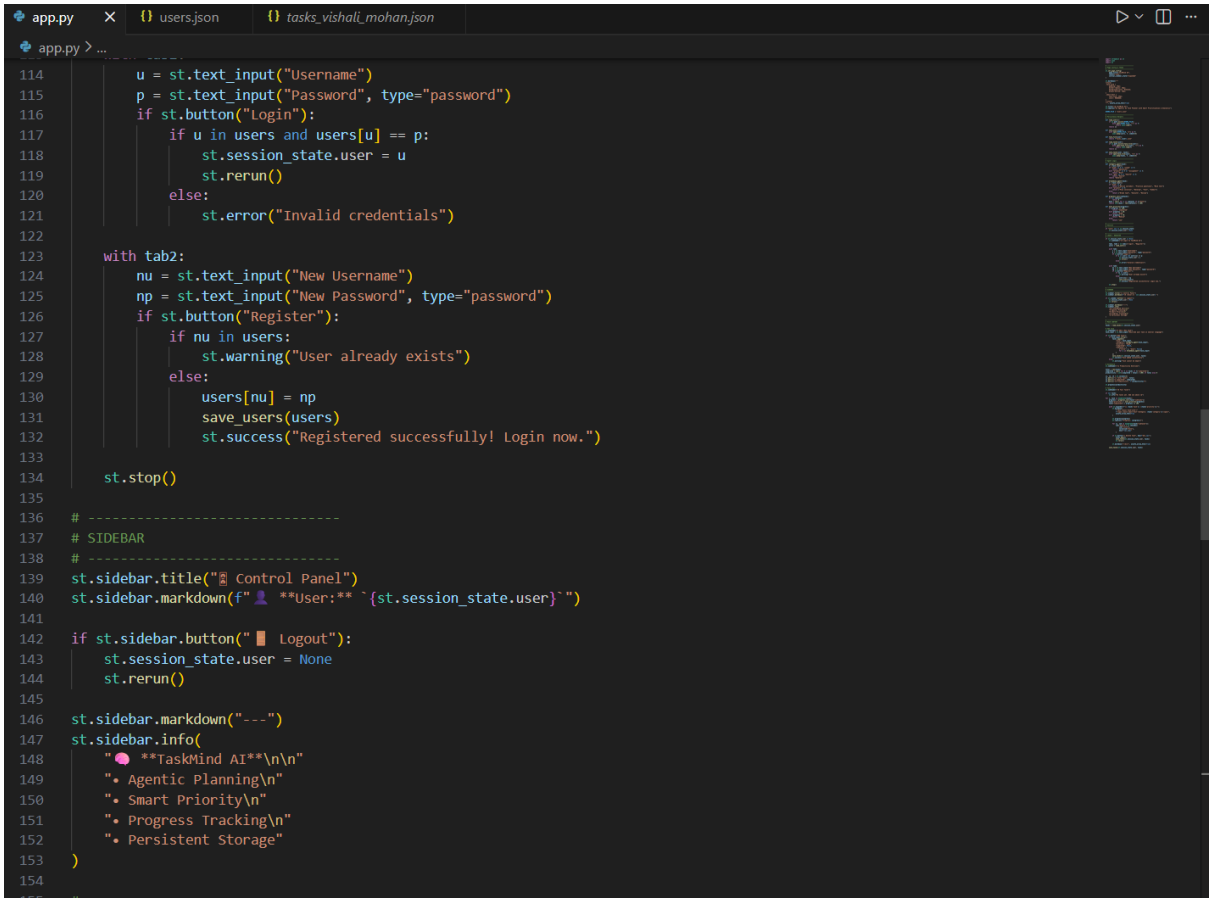

app.py
users.json
tasks_vishali_mohan.json
app.py > ...
import streamlit as st
import json
import os
# ------------------------------
# Page Config & Theme
# ------------------------------
st.set_page_config(
page_title="TaskMind AI",
layout="wide",
initial_sidebar_state="expanded"
)
st.markdown("""
<style>
.task-card {
padding: 15px;
border-radius: 10px;
background-color: #1e1e1e;
margin-bottom: 10px;
}
.small-text {
font-size: 13px;
color: #bbbbbb;
}
</style>
""", unsafe_allow_html=True)
st.title("🧠 TaskMind AI")
st.caption("An Agentic AI Task Planner with Smart Prioritization & Analytics")
USERS_FILE = "users.json"
# ------------------------------
# Persistence Helpers
# ------------------------------
def load_users():
if os.path.exists(USERS_FILE):
with open(USERS_FILE, "r") as f:
return json.load(f)
return {}


app.py
users.json
tasks_vishali_mohan.json
app.py > ...
def load_users():
if os.path.exists(USERS_FILE):
with open(USERS_FILE, "r") as f:
return json.load(f)
return {}
def save_users(users):
with open(USERS_FILE, "w") as f:
json.dump(users, f, indent=4)
def task_file(user):
return f"tasks_{user}.json"
def load_tasks(user):
if os.path.exists(task_file(user)):
with open(task_file(user), "r") as f:
return json.load(f)
return []
def save_tasks(user, tasks):
with open(task_file(user), "w") as f:
json.dump(tasks, f, indent=4)
# ------------------------------
# Agent Logic
# ------------------------------
def category_agent(task):
t = task.lower()
if "exam" in t or "study" in t:
return "Education"
elif "project" in t or "assignment" in t:
return "Work"
elif "gym" in t or "health" in t:
return "Health"
return "General"
def breakdown_agent(task):
t = task.lower()
if "exam" in t:
return ["Revise syllabus", "Practice questions", "Mock test"]
elif "project" in t:
return ["Plan solution", "Develop", "Test", "Submit"]
Ln 231, Col 1
Spaces: 4
UTF-8
CRLF
Python
Search

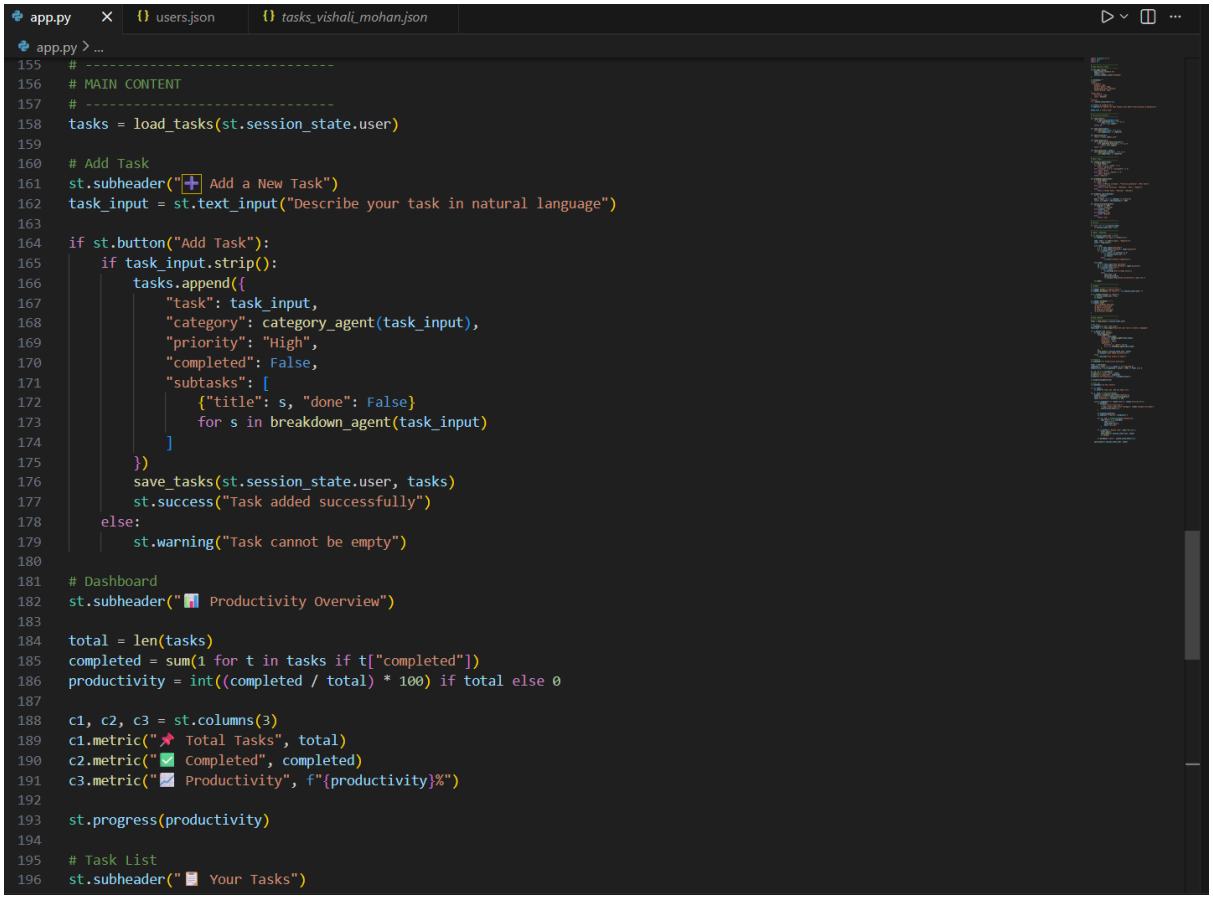


```
app.py  users.json  tasks_vishali_mohan.json
app.py > ...
def breakdown_agent(task):
        return ["Revise syllabus", "Practice questions", "Mock test"]
    elif "project" in t:
        return ["Plan solution", "Develop", "Test", "Submit"]
    else:
        return ["Break task", "Execute", "Review"]

def progress_calc(subtasks):
    if not subtasks:
        return 0
    done = sum(1 for s in subtasks if s["done"])
    return int((done / len(subtasks)) * 100)

def auto_priority(progress):
    if progress == 100:
        return "Completed"
    elif progress < 30:
        return "High"
    elif progress < 70:
        return "Medium"
    else:
        return "Low"

# ------------------------------
# Session
# ------------------------------
if "user" not in st.session_state:
    st.session_state.user = None

# ------------------------------
# LOGIN / REGISTER
# ------------------------------
if st.session_state.user is None:
    st.subheader("🔐 Login to TaskMind AI")

    tab1, tab2 = st.tabs(["Login", "Register"])
    users = load_users()

    with tab1:
        u = st.text_input("Username")
        p = st.text_input("Password", type="password")
        if st.button("Login"):
```

Ln 231, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.



```
app.py  users.json  tasks_vishali_mohan.json
app.py > ...
        u = st.text_input("Username")
        p = st.text_input("Password", type="password")
        if st.button("Login"):
            if u in users and users[u] == p:
                st.session_state.user = u
                st.rerun()
            else:
                st.error("Invalid credentials")

    with tab2:
        nu = st.text_input("New Username")
        np = st.text_input("New Password", type="password")
        if st.button("Register"):
            if nu in users:
                st.warning("User already exists")
            else:
                users[nu] = np
                save_users(users)
                st.success("Registered successfully! Login now.")

    st.stop()

# ------------------------------
# SIDEBAR
# ------------------------------
st.sidebar.title("🎛️ Control Panel")
st.sidebar.markdown(f"👤 **User:** `{st.session_state.user}`")

if st.sidebar.button("🚪 Logout"):
    st.session_state.user = None
    st.rerun()

st.sidebar.markdown("---")
st.sidebar.info(
    "🧠 **TaskMind AI**\n\n"
    "• Agentic Planning\n"
    "• Smart Priority\n"
    "• Progress Tracking\n"
    "• Persistent Storage"
)
```



---

11. ADVANTAGES OF THE SYSTEM

- Intelligent and adaptive task management
- Offline functionality (no external API dependency)
- Explainable and deterministic behavior
- User-friendly interface
- Scalable architecture
- Suitable for academic and real-world use

---

12. LIMITATIONS

- Uses rule-based logic instead of deep learning
- No cloud-based synchronization
- No real-time notifications
- Limited natural language understanding compared to large language models

---

13. FUTURE ENHANCEMENTS

- Integration with ChatGPT or Gemini AI
- Cloud database support (SQLite / Firebase)
- Mobile application version
- Task reminders and notifications
- Voice-based task input
- Calendar integration

---

14. CONCLUSION

TaskMind AI demonstrates how **agentic AI principles** can be effectively applied to real-world productivity applications. By combining intelligent planning, adaptive priority management, and user-centric design, the system provides a robust alternative to traditional to-do applications. This project successfully bridges the gap between theoretical AI concepts and practical software development.

---

15. REFERENCES

1. Russell, S. & Norvig, P. – *Artificial Intelligence: A Modern Approach*
2. Streamlit Documentation
3. Python Official Documentation