

## 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:

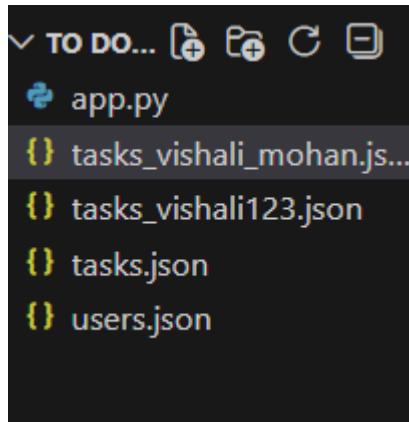
- No intelligent task planning or breakdown
- Manual priority assignment
- No adaptive behavior based on progress
- Lack of productivity analytics
- No personalized task storage for multiple users

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**

<b>Technology</b>	<b>Purpose</b>
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.

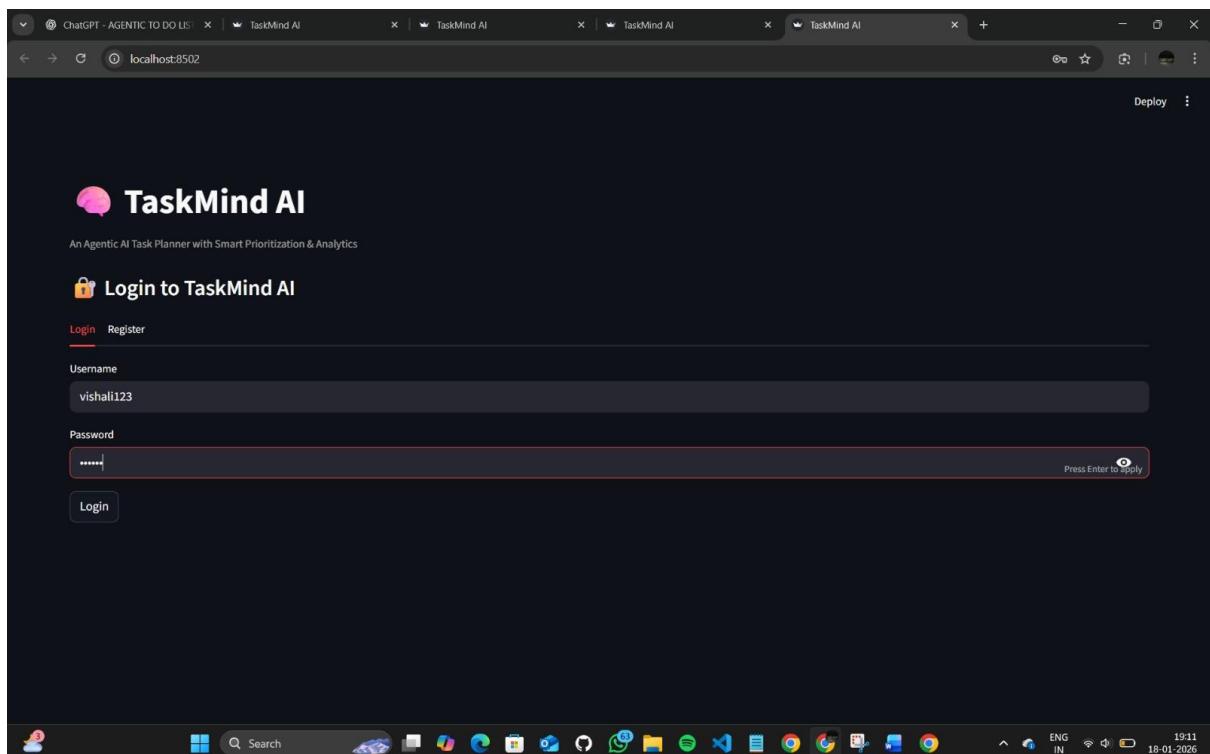
---

## 9. USER INTERFACE DESIGN

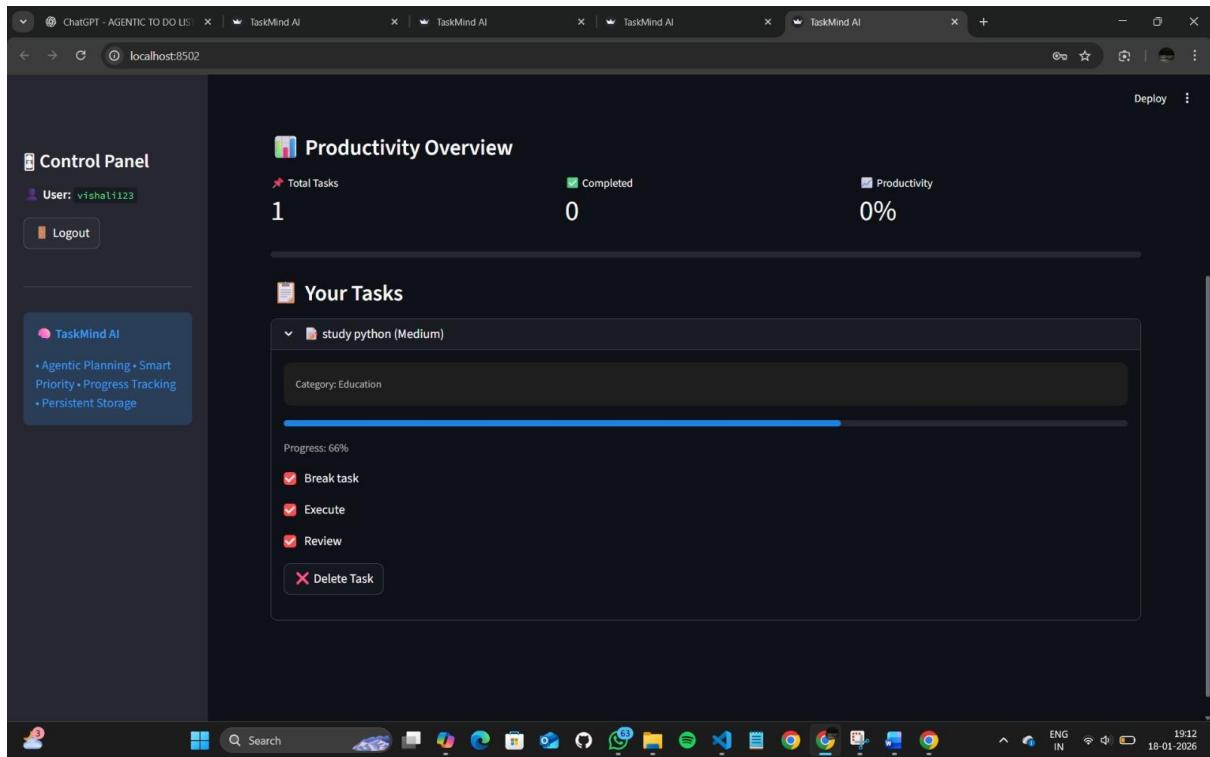
The user interface is designed with usability and clarity in mind:

- 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



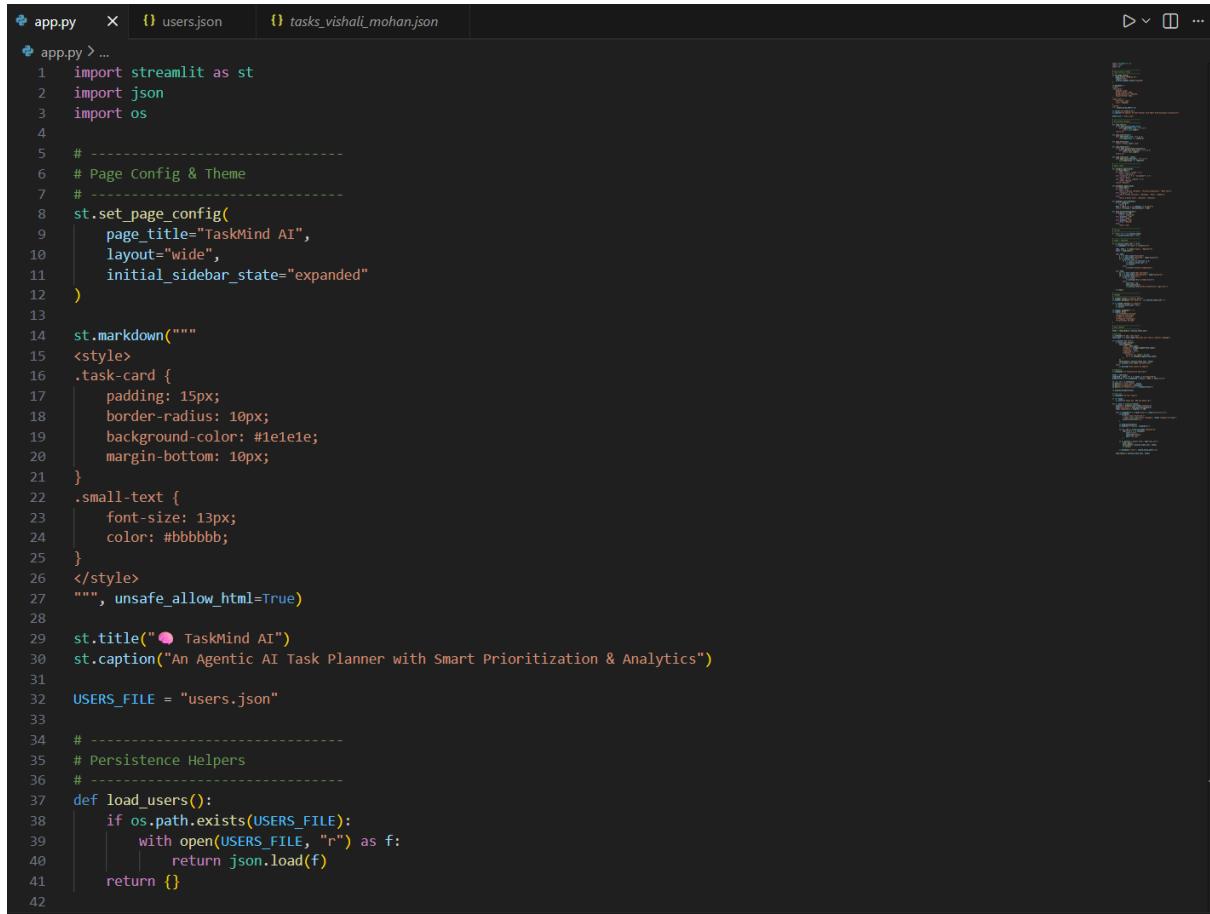
The screenshot shows the same browser window after logging in. The left sidebar has a "Control Panel" section with a user icon, "User: vishali123", and a "Logout" button. Below it is a "TaskMind AI" summary box with features: "Agentic Planning + Smart Priority", "Progress Tracking", and "Persistent Storage". The main content area starts with a "Productivity Overview" section. It shows "Total Tasks: 1", "Completed: 0", and "Productivity: 0%". Below this is a "Your Tasks" section, which lists a single task: "study python (High)". The browser's address bar shows "localhost:8502". The system tray at the bottom right indicates the date as 18-01-2026 and the time as 19:11.



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:



The image shows a code editor interface with three tabs open: `app.py`, `users.json`, and `tasks_vishali_mohan.json`. The `app.py` tab is active and displays the following Python code:

```
 1 import streamlit as st
 2 import json
 3 import os
 4
 5 # -----
 6 # Page Config & Theme
 7 # -----
 8 st.set_page_config(
 9     page_title="TaskMind AI",
10     layout="wide",
11     initial_sidebar_state="expanded"
12 )
13
14 st.markdown("""
15 <style>
16 .task-card {
17     padding: 15px;
18     border-radius: 10px;
19     background-color: #f1f1f1;
20     margin-bottom: 10px;
21 }
22 .small-text {
23     font-size: 13px;
24     color: #bbbbbb;
25 }
26 </style>
27 """", unsafe_allow_html=True)
28
29 st.title("● TaskMind AI")
30 st.caption("An Agentic AI Task Planner with Smart Prioritization & Analytics")
31
32 USERS_FILE = "users.json"
33
34 # -----
35 # Persistence Helpers
36 # -----
37 def load_users():
38     if os.path.exists(USERS_FILE):
39         with open(USERS_FILE, "r") as f:
40             return json.load(f)
41     return {}
```

The screenshot shows a code editor window with a dark theme. The main pane displays Python code for managing user tasks. The code includes functions for loading and saving users, creating task files, and defining categories for tasks like Education, Work, Health, and General. It also includes a breakdown function for specific task types. The status bar at the bottom shows file statistics: Ln 231, Col 1, Spaces:4, UTF-8, CRLF, and a Python icon.

```
app.py > ...
37 def load_users():
38     if os.path.exists(USERS_FILE):
39         with open(USERS_FILE, "r") as f:
40             return json.load(f)
41     return {}
42
43 def save_users(users):
44     with open(USERS_FILE, "w") as f:
45         json.dump(users, f, indent=4)
46
47 def task_file(user):
48     return f"tasks_{user}.json"
49
50 def load_tasks(user):
51     if os.path.exists(task_file(user)):
52         with open(task_file(user), "r") as f:
53             return json.load(f)
54     return []
55
56 def save_tasks(user, tasks):
57     with open(task_file(user), "w") as f:
58         json.dump(tasks, f, indent=4)
59
60 # -----
61 # Agent Logic
62 # -----
63 def category_agent(task):
64     t = task.lower()
65     if "exam" in t or "study" in t:
66         return "Education"
67     elif "project" in t or "assignment" in t:
68         return "Work"
69     elif "gym" in t or "health" in t:
70         return "Health"
71     return "General"
72
73 def breakdown_agent(task):
74     t = task.lower()
75     if "exam" in t:
76         return ["Revise syllabus", "Practice questions", "Mock test"]
77     elif "project" in t:
78         return ["Plan solution", "Develop", "Test", "Submit"]
```

The screenshot shows a code editor with a dark theme. The left pane displays the Python file `app.py`, which contains functions for task breakdown, progress calculation, auto-priority assignment, session management, and user authentication. The right pane shows a sidebar with a tree view of the project structure, including files like `__init__.py`, `users.json`, and `tasks_vishali_mohan.json`.

```
app.py
def breakdown_agent(task):
    if "type" in task:
        return ["Revise syllabus", "Practice questions", "Mock test"]
    elif "project" in task:
        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

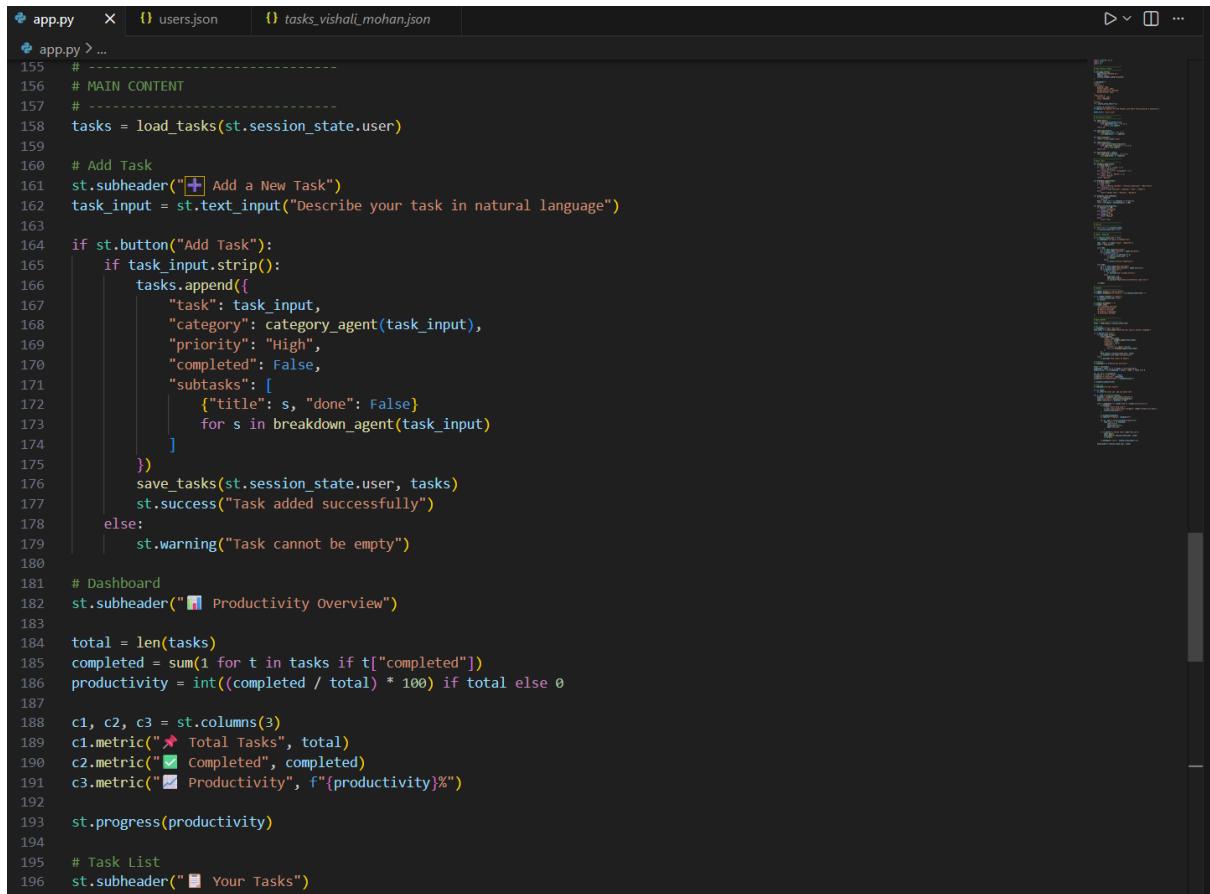
# -----
# LOGIN / REGISTER
# -----
if st.session_state.user is None:
    st.subheader("🔒 Login to TaskMind AI")

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"):
```

The screenshot shows a Jupyter Notebook interface with several tabs open. The active tab is 'app.py'. The code in 'app.py' is a Streamlit application. It includes sections for user login and registration, a sidebar with a control panel, and a sidebar info section. The Streamlit UI is visible on the right side of the screen.

```
114     u = st.text_input("Username")
115     p = st.text_input("Password", type="password")
116     if st.button("Login"):
117         if u in users and users[u] == p:
118             st.session_state.user = u
119             st.rerun()
120         else:
121             st.error("Invalid credentials")
122
123     with tab2:
124         nu = st.text_input("New Username")
125         np = st.text_input("New Password", type="password")
126     if st.button("Register"):
127         if nu in users:
128             st.warning("User already exists")
129         else:
130             users[nu] = np
131             save_users(users)
132             st.success("Registered successfully! Login now.")
133
134     st.stop()
135
136 # -----
137 # SIDEBAR
138 # -----
139 st.sidebar.title("Control Panel")
140 st.sidebar.markdown(f"**User:** {st.session_state.user}")
141
142 if st.sidebar.button("Logout"):
143     st.session_state.user = None
144     st.rerun()
145
146 st.sidebar.markdown("---")
147 st.sidebar.info(
148     "TaskMind AI\n"
149     "- Agentic Planning\n"
150     "- Smart Priority\n"
151     "- Progress Tracking\n"
152     "- Persistent Storage"
153 )
154
```



The screenshot shows a code editor with a dark theme. The main pane displays a Python script named `app.py`. The code implements a task management application using Streamlit. It includes functions for adding tasks, calculating productivity metrics, and displaying task lists. The code uses various Streamlit components like `st.subheader`, `st.text_input`, and `st.button`. On the right side of the editor, there is a sidebar showing a preview of the Streamlit app's interface, which includes sections for "Productivity Overview" and "Your Tasks".

```
155 # -----
156 # MAIN CONTENT
157 # -----
158 tasks = load_tasks(st.session_state.user)
159
160 # Add Task
161 st.subheader("➕ Add a New Task")
162 task_input = st.text_input("Describe your task in natural language")
163
164 if st.button("Add Task"):
165     if task_input.strip():
166         tasks.append({
167             "task": task_input,
168             "category": category_agent(task_input),
169             "priority": "High",
170             "completed": False,
171             "subtasks": [
172                 {"title": s, "done": False}
173                 for s in breakdown_agent(task_input)
174             ]
175         })
176     save_tasks(st.session_state.user, tasks)
177     st.success("Task added successfully")
178 else:
179     st.warning("Task cannot be empty")
180
181 # Dashboard
182 st.subheader("📊 Productivity Overview")
183
184 total = len(tasks)
185 completed = sum(1 for t in tasks if t["completed"])
186 productivity = int((completed / total) * 100) if total else 0
187
188 c1, c2, c3 = st.columns(3)
189 c1.metric("⭐ Total Tasks", total)
190 c2.metric("✅ Completed", completed)
191 c3.metric("📈 Productivity", f"{productivity}%")
192
193 st.progress(productivity)
194
195 # Task List
196 st.subheader("📋 Your Tasks")
```

---

## 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