AI Study Planner Agent
System Design Document

Priyanshi Agrawal

September 16, 2025

Contents

1 Introduction

The AI Study Planner Agent is designed to automate the scheduling and execution of daily study tasks for students. It provides reasoning, planning, execution, and knowledge retrieval features (via a simulated RAG system) through both CLI and Streamlit-based interfaces.

2 Objectives

- Automate task scheduling based on importance, duration, and deadlines.
- Allow multi-agent collaboration (Planner + Executor).
- Integrate external tools and knowledge sources (RAG, search tools).
- Provide an interactive UI for monitoring and editing schedules.
- Support batch or parallel execution of tasks.

3 Architecture Overview

Components

1. **User Interface (UI)**
    - CLI (Command Line Interface)
    - Streamlit Web App
2. **Planner Agent**
    - Reasoning module
    - Task prioritization based on importance and deadline
    - Generates a schedule for tasks
3. **Executor Agent**
    - Executes tasks sequentially or in parallel
    - Fetches relevant notes via RAG
    - Integrates optional tools (search, custom tools)
4. **RAG Knowledge Base**
    - Stores study notes
    - Provides similarity search for relevant content
5. **Task Database / Memory**
    - Session-level storage (Streamlit session state)
    - Stores task objects, schedule, and execution status

Diagram (Example)

User Input → Planner Agent → Schedule → Executor Agent → Task Execution / RAG Notes

4 Data Design

Task Object Structure

```
@dataclass
class Task:
    id: str
    name: str
    duration: int
    importance: int
    deadline: Optional[datetime.date]
    metadata: Dict[str, Any] = field(default_factory=dict)
```

Schedule Item Structure

```
{
    "task": Task,
    "start": datetime,
    "end": datetime
}
```

RAG Document Structure

```
{
    "text": "content of note or document chunk",
    "source": "filename or topic",
    "id": "unique identifier"
}
```

5 Component Breakdown

- **Planner Agent**
  - Input: List of tasks
  - Output: Ordered schedule (start and end times)
  - Key logic: Sort by importance and deadline, check available time
- **Executor Agent**
  - Executes scheduled tasks
  - Sequential or parallel execution
  - Logs progress

  - Retrieves relevant notes from RAG

- **RAG Store**
  - Optional: FAISS + sentence-transformers embedding
  - Stores chunks of documents
  - Returns top-k relevant notes for a task

- **Tools**
  - Custom plugin interface
  - Example: web search tool

- **UI**
  - CLI: simple prompts and outputs
  - Streamlit: task form, schedule display, execution logs

6 Technology Choices

Component	Choice	Reason
Programming Language	Python	Easy to implement AI logic, Streamlit, and RAG
UI	CLI & Streamlit	CLI for lightweight interaction, Streamlit for rich UI
Multi-agent system	Classes: PlannerAgent, ExecutorAgent	Clear separation of concerns
RAG Integration	FAISS + SentenceTransformer	Efficient similarity search
Tools integration	Plugin interface	Extensible for external components
Execution	Threading (parallel)	Simulate multiple tasks execution concurrently

7 Execution Flow

1. User inputs tasks via CLI or Streamlit
2. Planner Agent generates a schedule
3. Executor Agent executes tasks
   - Logs progress
   - Retrieves notes from RAG
   - Uses external tools if configured
4. User monitors via UI and can edit tasks or schedule

8 Future Extensions

- Connect to real databases (SQLite/PostgreSQL)
- Real web search / APIs
- Notifications / reminders
- Integration with calendar apps
- AI reasoning to dynamically adjust schedule

9 Social Impact

- Reduces stress and planning effort for students
- Helps in knowledge retention via RAG notes
- Encourages productivity and efficient time management

10 Interaction Logs

This section presents example logs of the AI Study Planner Agent in different versions (CLI, Multi-agent, RAG, and Streamlit). The logs illustrate the planning, execution, and retrieval of notes.

10.1 CLI Version (ai_agent.py)

```
       AI Study Planner Agent
Enter your tasks (type 'done' when finished):
Task name: Computer network lab exam
Duration (in minutes): 30
Importance (1-5): 5
Deadline (YYYY-MM-DD, or leave blank):
Task name: computer network theory quiz
Duration (in minutes): 30
Importance (1-5): 3
Deadline (YYYY-MM-DD, or leave blank):
Task name: done

    Today's Plan:
09:00 - 09:30: Computer network lab exam
09:30 - 10:00: computer network theory quiz
```

10.2 Multi-Agent Version (multi_agent.py)

```
    Planned Schedule:
09:00 - 10:00: Math Homework
10:00 - 11:30: Lab Report
11:30 - 12:00: Read Notes

    Executor Agent starting execution...

Starting: Math Homework at 09:00
Finished: Math Homework at 10:00

Starting: Lab Report at 10:00
Finished: Lab Report at 11:30

Starting: Read Notes at 11:30
Finished: Read Notes at 12:00
```

10.3 Multi-Agent with RAG (multi_agent_rag.py)

```
       Planned Schedule
09:00 - 10:00: Study Machine Learning
10:00 - 10:45: Review Compiler Design
10:45 - 11:15: Practice Networking

    Running Executor Agent...
    Starting: Study Machine Learning at 09:00
       Study Note: Machine learning is the study of algorithms that
   improve from experience.
    Finished: Study Machine Learning at 10:00
    Starting: Review Compiler Design at 10:00
       Study Note: A compiler translates source code into executable
   machine code.
    Finished: Review Compiler Design at 10:45
    Starting: Practice Networking at 10:45
       Study Note: Computer networks enable devices to communicate and
   share resources.
    Finished: Practice Networking at 11:15
```

10.4 Streamlit Version (multi_agent_ui.py)

```
       AI Study Planner - Streamlit UI

       Added Task:
Task: Prepare for AI Exam
Duration: 60 minutes
Importance: 5
Deadline: 2025-09-20

       Added Task:
Task: Write Lab Report
Duration: 90 minutes
Importance: 4
Deadline: 2025-09-19

       Planned Schedule:
09:00 - 10:00: Prepare for AI Exam
10:00 - 11:30: Write Lab Report

    Executor Agent:
    Starting: Prepare for AI Exam at 09:00
    Finished: Prepare for AI Exam at 10:00
    Starting: Write Lab Report at 10:00
    Finished: Write Lab Report at 11:30
```

Streamlit UI Screenshots

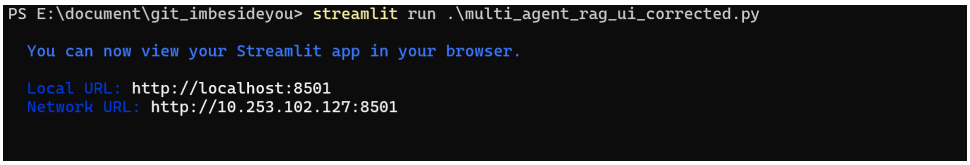

Figure 1: Streamlit user interface command line instruction.

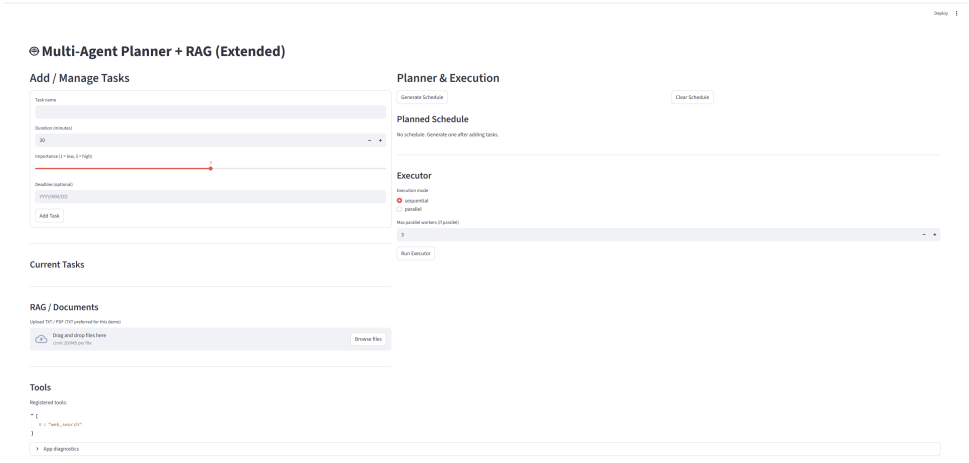

Figure 2: Streamlit user interface .