StudyBuddy

Overview

Study Buddy is a task automation system that uses Flask as the backend and Next.js as the frontend. The system efficiently manages multiple task executions using a multi-threaded approach. It allows users to initiate tasks, track their progress, and manage them dynamically. Crew AI is used to coordinate and optimize the execution of multiple concurrent tasks.

Features

Next.js Frontend: Provides an interactive UI for users to input and view task results.

Flask API: Manages requests, processes tasks, and tracks job statuses.

Multi-threaded Execution: Runs multiple task units ("Crew") concurrently for efficiency.

Job Status Management: Maintains a record of running jobs and their states.

Crew AI Integration: Automates task scheduling and execution using AI-powered workflows.

Architecture

1. Frontend (Next.js)

Provides a user-friendly interface for submitting and monitoring tasks.

Communicates with the Flask API to trigger actions and fetch results.

Sends:

POST /api/crew to start a new task.

GET /api/crew/edit to retrieve and edit crew-related information.

2. Backend (Flask API)

Handles API requests from the frontend.

Processes task initiation and retrieval requests.

Maintains job execution details in the Job Status Database.

3. Task Execution (Crew AI & Threads)

Kickoff Crew: Triggers a new task execution.

Threads: Each crew runs as a separate thread for parallel execution.

Crew AI: Optimizes task execution by managing dependencies and resource allocation.

4. Job Status Management

Job Status DB: Stores job execution details.

Job Manager Lock: Ensures thread safety in task management.

Installation

1. Clone the repository:

```
git clone https://github.com/your-username/StudyBuddy.git
cd StudyBuddy
```

2. Basic Setup
    - To install poetry visit poetry
3. Setup backend
    - Change your directory to server `cd server`
    - Run `poetry install --no-root`
    - Add api keys for `YOUTUBE OPENAI` and `SERPER` as mentioned in `.env.example`
    - Run `api.py`
    - Additionally test the following endpoints on postman :

```
###
POST http://localhost:3001/api/crew/
Content-Type: application/json

{
 "subjects": ["SUBJECT 1","SUBJECT 2",....,""],
 "topics":["TOPIC 1","TOPIC2",....,""]
}

Expect code 202 and

{
"job_id": "39c6cd78-c4c6-4069-bac0-ffc7d757c3d0"
}


GET https://localhost:3001/api/crew/<job_id>

######

Expect code 200 and

{
    "events": [
        {
            "data": "Task Started",
            "timestamp": ""
```

```
            },
            {
                "data": "",
                "timestamp": ""
            }
        ],
        "job_id": "",
        "result": "",
        "status": "STARTED"
```


}

```
4. Setup frontend
    - Change directory to client `cd client`
    - Run `pnpm install`
    - Add api keys for clerk as mentioned in `.env.example`
    - Run `pnpm dev`



## License

This project is licensed under the [MIT License](LICENSE).
```