**PROJECT - 1**

**Enhanced Text Extractor Tool**

**Objective:**

Develop an **Enhanced Text Extractor** tool that uses the **Langchain library** to process extracted text and display the results in an interactive table on a **Next.js/ REACT JS / NODE JS** web application. The application should provide a robust front-end for file uploads, handle text extraction from **pdf** formats, enrich the data using **OpenAI**, and display the processed results in a user-friendly format.

**Deliverables:**

- A functional Next.js web application.

- An interactive UI for file uploads and data display.

- Documentation for setup, usage, and design decisions.

**Requirements:**

**Core Features:**

**1.** **Text Extraction**:

- Extract text from uploaded file.

- Support for pdf file format.

**2. API Development:**

- Develop RESTful API routes for text extraction.

**3. Natural Language Processing:**

- Integrate Langchain for post-extraction text analysis.

- Fetch Key-Pair attributes using OpenAI.

**4. Data Display:**

- Display extracted text and NLP analysis results in an interactive table.

**Advanced Features:**

1. **Testing:**

- Provide unit tests for critical functions.

- Write end-to-end tests for the user flow.

2. **Documentation:**

- Include a README with setup instructions and dependencies.

- Code comments and function/module descriptions.

3. **Deployment**:

- Deploy the application on a cloud platform (Vercel, Netlify, heroku, or similar) or share in zip file

**Evaluation Criteria:**

- Adherence to project requirements.

- Cleanliness and organization of code.

- UI/UX design and usability.

- Efficiency and reliability of text extraction and NLP processes.

- Langchain and OpenAI implementation.

**PROJECT - 2**

**Build a dashboard that provides analytics for users & repositories on GitHub. The**

**dashboard should allow users to search for a GitHub user and display a table of their repositories with details like stars, forks, open issues, and recent commits.**

Features List:

1. User search: An input box to search for users and show a list of cards with the

responses. (Need not be paginated, pick first few results as you deem fit)

2. User Repository Table: Clicking on the user navigates to a different page to

display a list of a user's repositories with columns for the name, description, number

of stars, forks, and open issues.

3. Sorting: Allow the user to sort repositories by name, stars, forks, or open issues.

(could be client side / server side)

4. Search Filter: Implement a search filter to find repositories by name within the

user's list. (could be client side / server side)

5. Commit History Visualization: Clicking on a row on the repository table should

select the repository and for the selected repository, show a graph of the commit

activity over the last year.

6. Contributors List: List contributors for each repository and their contributions

count.

7. Repository Detail View: Create a detailed view for each repository that includes

the README file, a list of recent commits, and open issues.

page which opens up on clicking a “View” button available per each row on the User

Repository Table

8. Rate Limit Handling: Implement a feature to handle GitHub's API rate limiting

gracefully, perhaps by caching data or informing the user.