Internship Details:

Stipend: €200-400 per month, based on experience and performance

Duration: 3 months

Location: Remote

Project Deadline and Submission Details

- **Deadline**: November 18th 2024
- **Submission**: Please send your completed project, including the GitHub repository link, to **kapil@breakoutinvesting.in**

Welcome to the assessment! This document provides a complete overview of the requirements and expected results, along with answers to potential questions. This project will help us evaluate your skills in machine learning, API integration, and prompt engineering, as well as your approach to creating user-friendly applications.

Project Overview

You will create an AI agent that reads through a dataset (CSV or Google Sheets) and performs a web search to retrieve specific information for each entity in a chosen column. The AI will leverage an LLM to parse web results based on the user's query and format the extracted data in a structured output. The project includes building a simple dashboard where users can upload a file, define search queries, and view/download the results.

Expected Deliverables

1. **Github Repository**: A well-organized repo containing the project code, organized into relevant directories, with a `readme.md` file that explains the setup, usage, and key features of the project.
2. **Readme File**: This should include a project summary, setup instructions, usage guide, and details on any third-party APIs or tools you used.
3. **Loom Video**: A 2-minute video walkthrough explaining the project's purpose, demonstrating key functionalities, and highlighting any notable code implementations or decisions.
4. **Working Dashboard**: A simple, intuitive UI that allows users to:
    - Upload CSV files or connect a Google Sheet.
    - Choose the primary column with the list of entities (e.g., companies).
    - Input a custom prompt for information retrieval.
    - View and download the extracted results.

Key Features & Requirements

Here's a breakdown of each feature you'll be building, along with details on the expected outcome for each:

1. Dashboard for File Upload and Google Sheets Connection

- **Goal**: Allow users to upload a CSV file or connect to a Google Sheet directly for data input.
- **Expected Outcome**:
  - Users can upload a CSV file with a simple "Browse" button or enter Google Sheet credentials to link to a sheet.
  - Display the available columns from the CSV or Google Sheet, allowing the user to select the main column (e.g., company names).
  - Show a preview of the uploaded data.
- **Technical Details**:
  - Integrate with the Google Sheets API for real-time Google Sheet access, allowing users to authenticate and pull data from their sheets.

2. Dynamic Query Input with Prompt Template

- **Goal**: Allow users to specify the type of information they want to retrieve for each entity in the main column.
- **Expected Outcome**:
  - A text input box where users can define a custom prompt, such as "Get me the email address of {company}".
  - Ensure users understand they can use placeholders (e.g., `{company}`) that will be dynamically replaced by each entity in the selected column.

3. Automated Web Search for Information Retrieval

- **Goal**: Perform web searches for each entity using the custom prompt and gather relevant web results.
- **Expected Outcome**:
  - For each entity in the selected column, your agent should conduct a web search (e.g., "Get me the email address of {company}").
  - Gather and store search results (e.g., URLs, snippets) for each entity.
  - Consider using `SerpAPI`, `ScraperAPI`, or other web scraping/search APIs, ensuring proper API usage limits.
- **Technical Details**:
  - Implement logic to handle rate limiting and avoid potential blocking.

- Store each entity's results in a structured format, ready for further processing by the LLM.

4. Passing Results to an LLM for Parsing and Information Extraction

- **Goal**: Use an LLM to extract specific information based on the user-defined prompt and web results.
- **Expected Outcome**:
  - Send each entity's search results to the LLM, along with a backend prompt like "Extract the email address of {company} from the following web results.". This prompt could be asked from the user as well.
  - Ensure the LLM processes the search results and extracts the requested information (e.g., email, address, etc.) for each entity.
- **Technical Details**:
  - Implement LLM integration (e.g., Groq or OpenAI's GPT API) for processing the data.
  - Handle any errors gracefully, such as retrying failed queries or notifying the user if data retrieval is unsuccessful.

5. Displaying and Storing the Extracted Information

- **Goal**: Show extracted data in a user-friendly format and provide an option to download the data.
- **Expected Outcome**:
  - Display the extracted data in a table format within the dashboard, organized by entity and extracted information.
  - Offer an option to download the results as a CSV or update a connected Google Sheet with the extracted information.
- **Technical Details**:
  - Provide a "Download CSV" button and an option to update the Google Sheet.

Optional Advanced Features

For extra credit, consider implementing the following:

- **Advanced Query Templates**: Allow users to extract multiple fields in a single prompt, such as "Get the email and address for {company}."
- **Google Sheets Output Integration**: Offer users the option to write back the extracted data directly to the Google Sheet.
- **Error Handling**: Robust error-handling mechanisms for failed API calls or unsuccessful LLM queries, with user notifications.

Technical Stack Suggestions

Below is a list of recommended technologies, but feel free to use alternatives if you're comfortable with them:

- **Dashboard/UI**: Streamlit, Flask
- **Data Handling**: `pandas` for CSV files; Google Sheets API for Sheets
- **Search API**: SerpAPI, ScraperAPI, or another search service
- **LLM API**: Groq or OpenAI's GPT API
- **Backend**: Python
- **Agents:** Langchain

Submission Instructions

1. **Github Repository**: Push your project to a public GitHub repository. Make sure to organize your code and include a comprehensive `README.md`.
2. **Readme File**: Your `readme.md` should contain:
    - **Project Description**: Brief overview of the purpose and capabilities of your project.
    - **Setup Instructions**: How to install dependencies and run the application.
    - **Usage Guide**: Instructions for using the dashboard, including connecting Google Sheets and setting up search queries.
    - **API Keys and Environment Variables**: Explain where users should enter their API keys and any other required environment variables.
    - **Optional Features**: Highlight any extra features you've added.
3. **Loom Video**: Record a 2-minute video walkthrough of your project, explaining:
    - The overall purpose of the project.
    - Key features and how the dashboard works.
    - Any notable code implementations or challenges you encountered.
    - Share this video link in your repository's `README.md`.

Frequently Asked Questions

1. Can I use any other search or scraping API?

Yes, you may use any search or scraping API of your choice as long as it retrieves web data effectively and can handle automated requests. Some popular options include `SerpAPI`, `ScraperAPI`, or even search APIs provided by major search engines if they allow sufficient requests for testing.

2. What if I can't find a free LLM API?

If you're unable to find a free or accessible LLM API for testing, you can use the **Groq API**, which offers a free tier specifically for projects and experimentation. We encourage you to use this option if you're unable to access other LLMs. Include instructions in your `README.md` for how users can set up a Groq API key if you choose this option.

3. What if the LLM fails to extract the needed information?

Implement a fallback mechanism, such as retrying the query or notifying the user that the extraction was incomplete. Consider displaying a message in the dashboard to indicate that data may be missing for some entities and make sure errors are handled gracefully.

4. Are there restrictions on UI design?

Keep the UI simple, clear, and user-friendly. While we suggest using **Streamlit** for a quick setup, **Flask** is also acceptable if you prefer more control over UI customization. The focus is on functionality, so prioritize features and usability over aesthetic design.

5. How should I handle API keys and other sensitive information?

Store sensitive information like API keys in environment variables for security purposes. In your `README.md`, provide instructions on where users should input their API keys and any other required credentials. Avoid hardcoding these details directly in your code.

6. What if I need more time or run into technical difficulties?

Please reach out to us if you encounter specific technical issues or if you require more time to complete the project. We're here to support you, so feel free to communicate any blockers.

7. Are there limits on what data I should scrape or extract?

Focus on retrieving publicly available information related to each entity (e.g., email, address, or products). Respect website terms of service, and do not attempt to scrape sensitive or private data. Always use API services responsibly, adhering to their usage guidelines and rate limits.

8. How do I handle large datasets efficiently?

If processing large datasets, consider limiting the number of queries or implementing a batch processing approach. This will help avoid timeouts and API rate limits while making your solution scalable.

9. Can I add extra features or enhancements?

Absolutely! We encourage you to be creative with optional features (e.g., handling multiple prompts in a single query, more advanced error handling, or exporting results to Google Sheets). Just make sure to clearly highlight any additional functionality in your `README.md`.

Stay Connected:

Follow **Kapil Mittal** on LinkedIn for updates and opportunities: LinkedIn Profile.

Join the Telegram channel for updates in trading and investment: Telegram @breakoutinvesting.

Check out other trading insights and strategies on TradingView: TradingView Profile.

Visit the Breakout AI website for more about what we do: breakoutai.tech.

Assessment for AI Agent Project