AI-Powered Developer Performance Analytics Dashboard

Submission Guidelines

Originality Requirement:

- **Integrity matters!** Ensure all submitted work is original. We use plagiarism checkers, so avoid copying others' work or discussing results until after the selection process.

Submission Format:

- Submit your work via the provided Dropbox link.
- Include all files in a single zip archive for ease of handling and review.

Documentation Requirements:

- Provide a one-page report summarizing your methodology, findings, and recommendations.
- Ensure clarity and coherence in your documentation.

Code and Execution:

- **Code Submission:**
    - Provide well-structured code with clear instructions on how to run and evaluate the system.
    - Include instructions for reproducing results and interpreting outcomes using the provided data.
- **Demo Presentation:**
    - **Demo Preparation:**
        - Create a concise (less than 5 minutes) video demonstrating your system in action.
        - Utilize tools like Yoodli.ai for presentation refinement.
        - Leverage LLMs (Large Language Models) to script and refine your demo for clarity and impact.
    - **Presentation Quality:**
        - Ensure your presentation highlights your coding skills and effectively communicates your solution’s value and innovation.
        - Demonstrate how your solution addresses real-world challenges and stands out from generic approaches.

Objective

Develop a Streamlit-based dashboard that provides insights into developer performance using data from an open-source GitHub repository. The system should focus on collecting and analyzing GitHub data, calculating performance metrics, and implementing a natural language interface for querying these metrics.

Core Requirements

1. Data Collection Module

- **Input**: GitHub repository URL
- **Output**: Raw data on commits, pull requests, issues, and code reviews
- **Tool**: Use GitHub API (PyGithub) for data collection

2. Metrics Calculation Module

- **Input**: Raw GitHub data
- **Output**: Calculated performance metrics (e.g., commit frequency, PR merge rate, issue resolution time)
- **Library**: Implement basic statistical calculations using Pandas

3. Dashboard Visualization Module

- **Input**: Calculated metrics
- **Output**: Interactive charts and graphs
- **Tool**: Use Plotly for creating visualizations

4. Natural Language Query Module

- **Input**: User's natural language question
- **Output**: Relevant metrics and visualizations based on the query
- **Approach**: Use a simple keyword-based approach or integrate with a basic LLM API

Streamlit UI

- **Framework**: Streamlit-based interface
- **Features**:
    - Sections for overall metrics
    - Individual developer statistics
    - Natural language query interface

Data Storage

- **Functionality**: Implement a simple CSV-based storage system for collected data

---

Optional Enhancements (Extra Credit)

1. Multi-Repository Support

- **Feature**: Allow users to switch between multiple GitHub repositories

2. Advanced Query Processing

- **Feature**: Implement more sophisticated natural language understanding for complex queries

---

Sample Project Structure

```
dev_performance_dashboard/
├── data_collection/
│   ├── github_api.py
│   └── data_storage.py
├── metrics/
│   ├── calculator.py
│   └── definitions.py
├── visualization/
│   ├── charts.py
│   └── dashboard.py
├── query_interface/
│   ├── nlp_processor.py
│   └── response_generator.py
├── app.py
├── requirements.txt
└── README.md
```

Implementation Tips

- Start by implementing basic data collection from a single GitHub repository.
- Focus on calculating a core set of metrics before expanding to more complex ones.
- Use Streamlit's built-in components to quickly create an interactive dashboard.
- For the **natural language query module**, begin with a simple keyword-matching system before attempting more advanced NLP techniques.
- Prioritize **data visualization** and **user experience** in the dashboard design.
- Implement caching mechanisms in Streamlit to improve dashboard performance.
- For **data storage**, use Pandas to read and write CSV files efficiently.
- Be mindful of GitHub API rate limits and implement proper error handling.
- Emphasize the creation of a user-friendly dashboard that provides valuable insights into developer performance.

Project Considerations

Interns should focus on delivering a dashboard that offers clear and actionable insights for project managers and team leads. They should think about how this tool could be used in real-world scenarios to track and improve developer productivity.