AI-POWERED DEVELOPER ANALYTICS DASHBOARD

1.Methodology

Objective:

This project aims to analyze GitHub repository performance metrics, such as Pull Request (PR) merge rates, issue closure rates, and commit history, to provide insightful answers to user queries regarding the repository.

Tools and Framework:

Streamlit: Used to build an interactive web interface for analyzing GitHub repository URLs.
GitHub API: Used to fetch repository data, including pull requests, issues, and commit information.
Backend Architecture: Processes the fetched data and supports natural language queries, allowing users to ask questions such as "What is the PR merge rate?" after the analysis.

Steps:

1. GitHub URL Input: Users provide a GitHub repository URL in the interface.
2. Data Fetching: Upon clicking the "Analyze" button, the backend fetches the required data from the GitHub API.
3. Data Processing: The repository's PRs, issues, and commits are analyzed to extract metrics such as the PR merge rate and issue closure rate.
4. Question Parsing: After the analysis, users can input questions related to repository performance (e.g., "What is the PR merge rate?").
5. Result Display: Answers to the user's questions are displayed below the input field based on the analyzed data.

2.Findings:

- Data Collection Success: The system successfully fetches data from the GitHub API based on the repository URL provided by the user.
- Performance Metrics Extraction: The system correctly calculates key performance metrics, such as:
- PR Merge Rate: The percentage of pull requests that were successfully merged into the repository.
- Issue Closure Rate: The percentage of issues that were closed over a certain period.
- Commit Frequency: Provides insights into how often contributions are being made to the repository.

Challenges Encountered:

- Page Reload Issue: Initially, the page was reloading after asking questions, which interrupted the user experience. This was resolved by ensuring that the question-answering function was appropriately placed in the Streamlit app execution flow.

- Parsing Natural Language Queries: Ensuring the correct interpretation of user queries required building robust keyword-matching logic.

3.Recommendations:

- User Experience Improvement: To enhance the dashboard experience, real-time data visualization (such as graphs showing PR merge rate trends, commit frequency, and issue resolution timelines) could be added.
- Scalability: For larger repositories with thousands of PRs and issues, additional optimization may be required to handle the volume of data.
- Expand Query Capabilities: By integrating NLP models or further improving keyword matching, the system can support more complex queries and provide more nuanced answers.