AI-Powered Developer Performance Analytics Dashboard

Overview

The AI-Powered Developer Performance Analytics Dashboard is a tool designed to analyze and visualize developer performance based on GitHub repository data. The dashboard provides performance metrics and supports natural language queries via the Groq LLaMA API, allowing users to query developer performance insights interactively.

Project Objective

The objective of this project is to deliver an interactive Streamlit-based dashboard that:

- Analyzes developer performance using metrics derived from GitHub data.
- Visualizes the metrics with interactive charts and graphs.
- Supports natural language queries to retrieve insights using the Groq LLaMA API.

1. Data Collection Module

Purpose

The Data Collection Module gathers raw data from a specified GitHub repository. This includes commits, pull requests, issues, and code reviews.

How It Works

- Input: GitHub repository URL.
- Output: Raw data about commits, pull requests, issues, and code reviews stored in CSV format.

Files:

- data_collection/github_api.py:
    - Fetches repository data using the GitHub API (PyGithub).
    - Functions:
        - get_repo_data(repo_url): Retrieves repository data.
        - fetch_commits(repo): Fetches commit data.
        - fetch_issues(repo): Fetches issue data.
        - fetch_pull_requests(repo): Fetches pull request data.
- data_collection/data_storage.py:
    - Saves and retrieves raw GitHub data in CSV format.
    - Functions:
        - save_to_csv(data, filename): Saves data to CSV.
        - load_from_csv(filename): Loads data from a CSV file.

2. Metrics Calculation Module

Purpose

The Metrics Calculation Module computes various performance metrics from the raw GitHub data, helping to evaluate developer contributions and effectiveness.

Metrics Calculated

- Commit Frequency: Measures how frequently commits are made over a period of time.
- PR Merge Rate: The percentage of pull requests that have been merged.
- Issue Resolution Time: The average time it takes to resolve issues.
- Code Review Participation: Measures the number of code reviews a developer is involved in.

Files:

- metrics/calculator.py:
    - Performs statistical calculations of performance metrics.
    - Functions:
        - calculate_commit_frequency(data): Computes commit frequency.
        - calculate_pr_merge_rate(data): Calculates the PR merge rate.
        - calculate_issue_resolution_time(data): Calculates the average issue resolution time.
- metrics/definitions.py:
    - Contains the definitions of all metrics to be calculated.

3. Dashboard Visualization Module

Purpose

The Dashboard Visualization Module displays calculated performance metrics in a user-friendly, interactive format using charts and graphs.

How It Works

- Input: Calculated performance metrics.
- Output: Interactive visualizations rendered in the Streamlit dashboard.

Files:

- visualization/charts.py:
    - Creates interactive charts using the Plotly library.
    - Functions:

        - plot_commit_frequency(data): Plots a line chart showing commit frequency.
        - plot_pr_merge_rate(data): Plots a bar chart showing the PR merge rate.
        - plot_issue_resolution_time(data): Creates a scatter plot showing issue resolution time.
- visualization/dashboard.py:
    - Manages the Streamlit dashboard layout and structure. Displays overall repository metrics and individual developer statistics.

4. Natural Language Query Module

Purpose

The Natural Language Query Module allows users to ask questions in plain language and retrieve relevant performance metrics and visualizations. This module uses the Groq LLaMA API to interpret natural language queries.

How It Works

- Input: User's natural language query.
- Output: Relevant metrics and corresponding visualizations based on the query.

Files:

- query_interface/nlp_processor.py:
    - Handles natural language processing using the Groq LLaMA API.
    - Functions:
        - process_query(query): Sends the user's query to the Groq LLaMA API and returns the relevant metrics.
- query_interface/response_generator.py:
    - Generates responses based on the metrics returned by the nlp_processor.py.
    - Functions:
        - generate_response(metric_name): Retrieves the requested metric and generates a response.

---

5. Streamlit User Interface

Purpose

The Streamlit User Interface provides an interactive front-end for the dashboard, allowing users to explore overall metrics, individual developer performance, and ask natural language queries.

File:

- app.py:
    - This file runs the Streamlit application.
    - Features:
        - Sidebar for selecting repositories and time periods.
        - Main dashboard to view charts and graphs of performance metrics.
        - A text input field for users to enter natural language queries.

6. Data Storage

Purpose

Collected GitHub data is stored locally in CSV format to enable persistent access and easy retrieval for analysis.

File:

- data_collection/data_storage.py:
    - Manages the saving and loading of raw data in CSV format.

Installation

Prerequisites

- Python 3.8 or higher.
- Install the dependencies listed in requirements.txt by running:

bash

pip install -r requirements.txt

Setting Up API Tokens

Before running the application, you must update your .env file with the following:

- GitHub Personal Access Token: To authenticate with the GitHub API for data collection.
- Groq LLaMA API Token: Required to process natural language queries using the Groq LLaMA API.

The .env file should have the following structure:

GITHUB_TOKEN=<your_github_personal_access_token>

GROQ_LLAMA_TOKEN=<your_groq_llama_api_token>

Running the Dashboard

To run the Streamlit dashboard, execute the following command:

bash

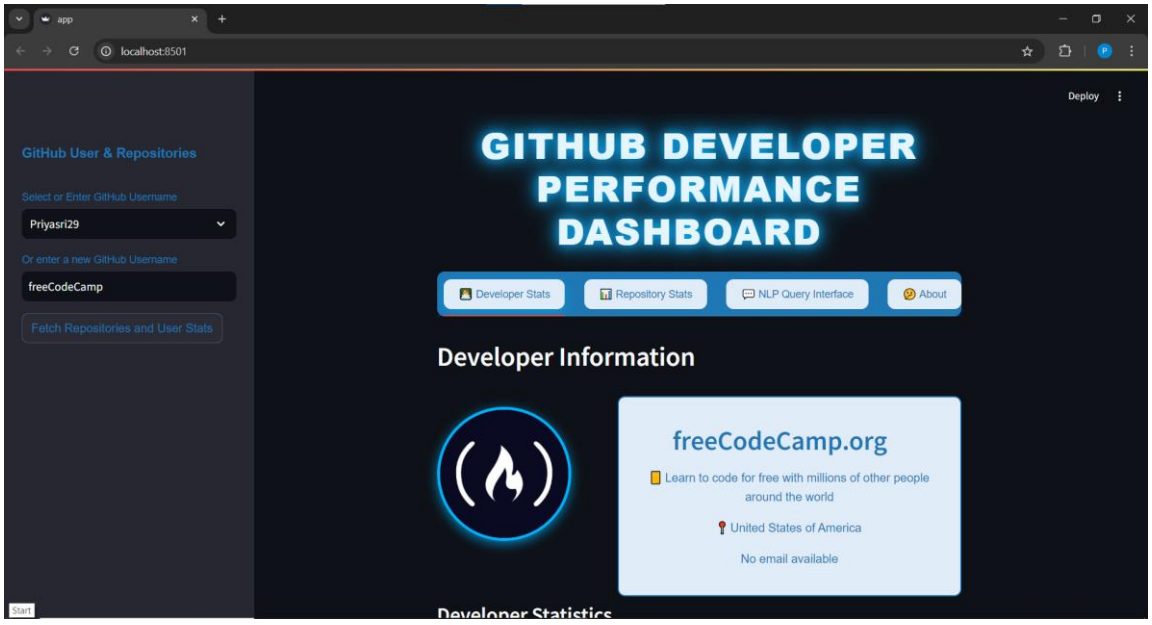
streamlit run app.py

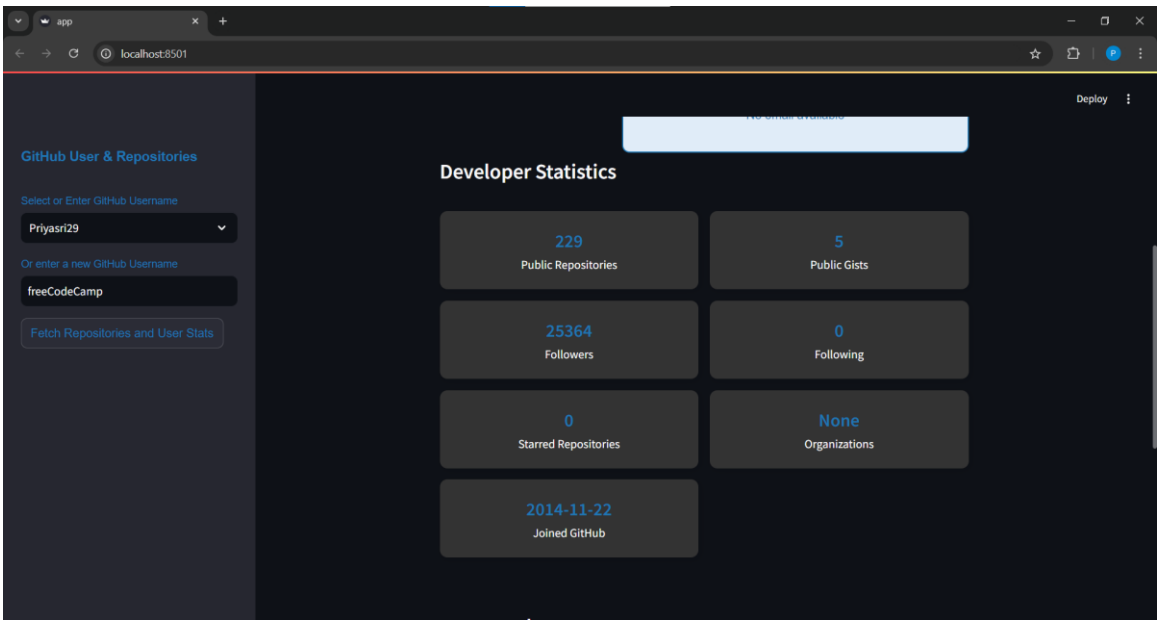
This will launch the dashboard on your local server, allowing you to explore the developer performance metrics and use the natural language query interface.

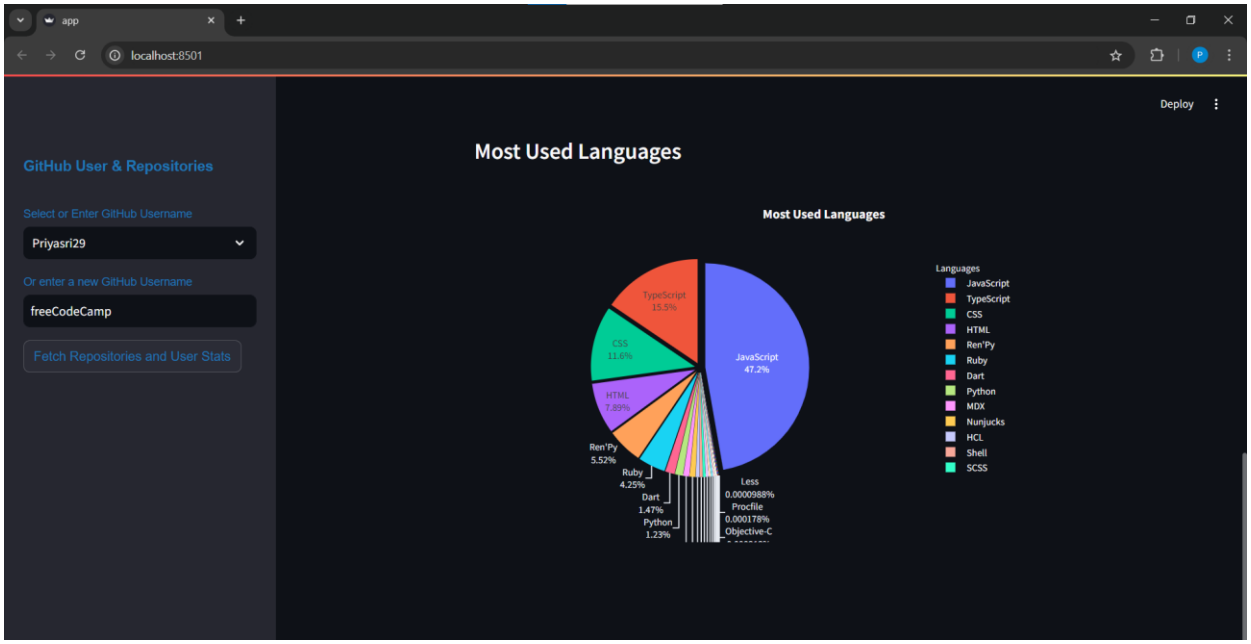
Conclusion

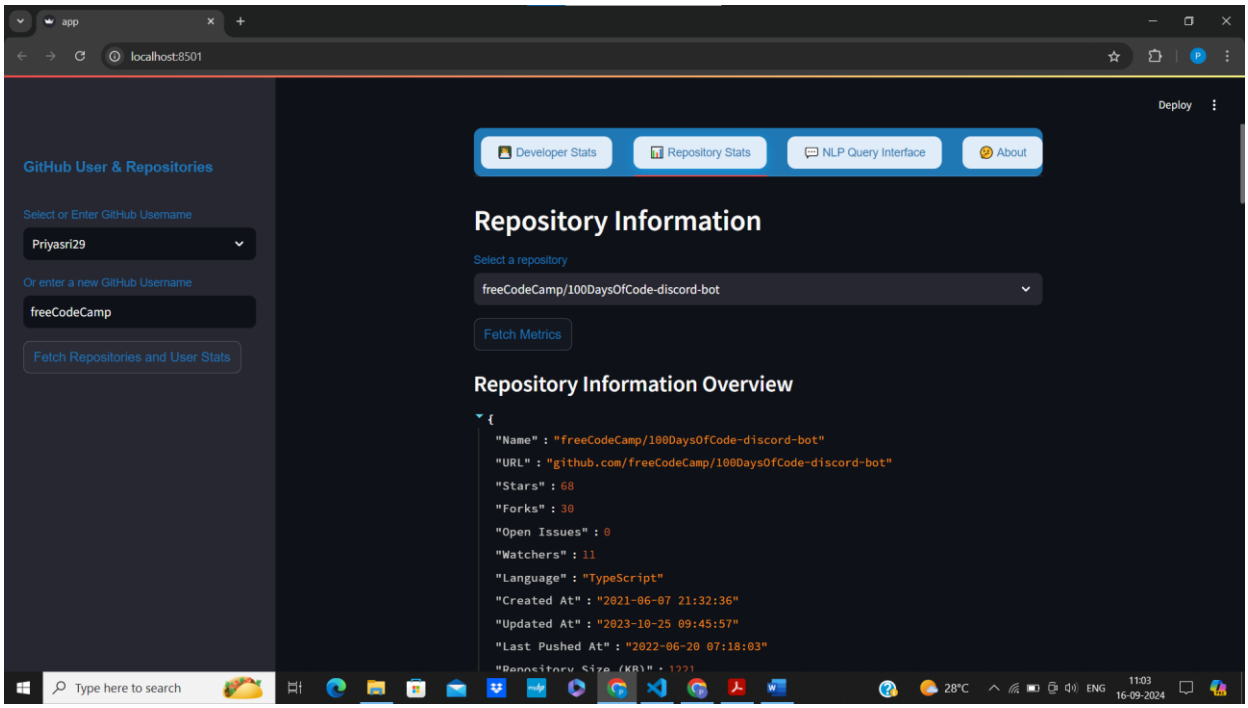
The AI-Powered Developer Performance Analytics Dashboard provides a streamlined way to assess developer performance based on GitHub activity. With interactive visualizations and natural language support via the Groq LLaMA API, this dashboard is a powerful tool for project managers, team leads, and individual developers to monitor and improve productivity.

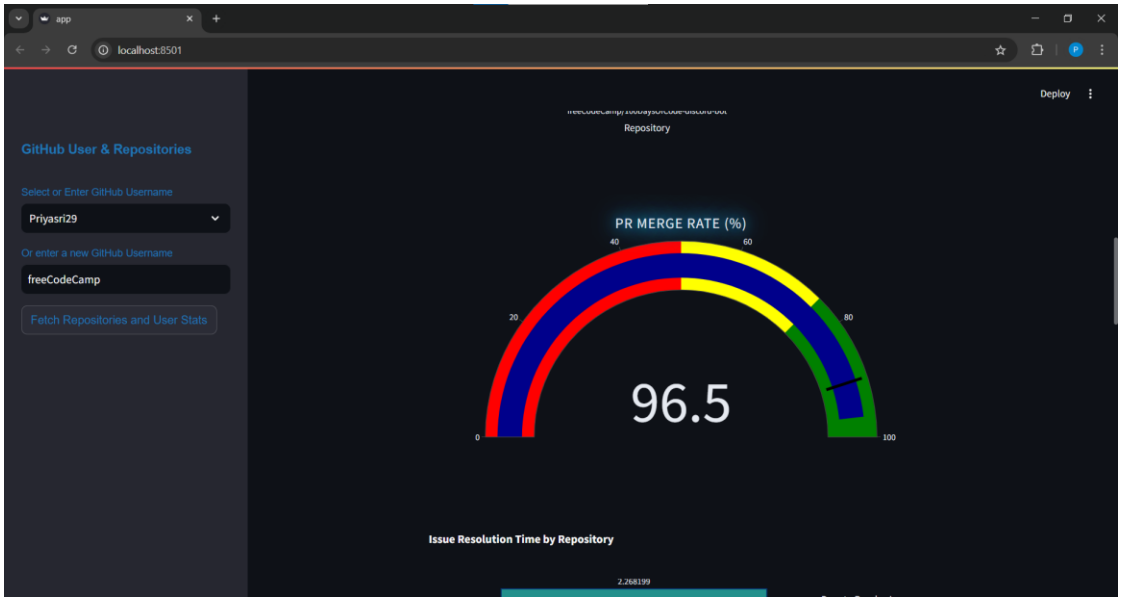
Sample Output Screens:



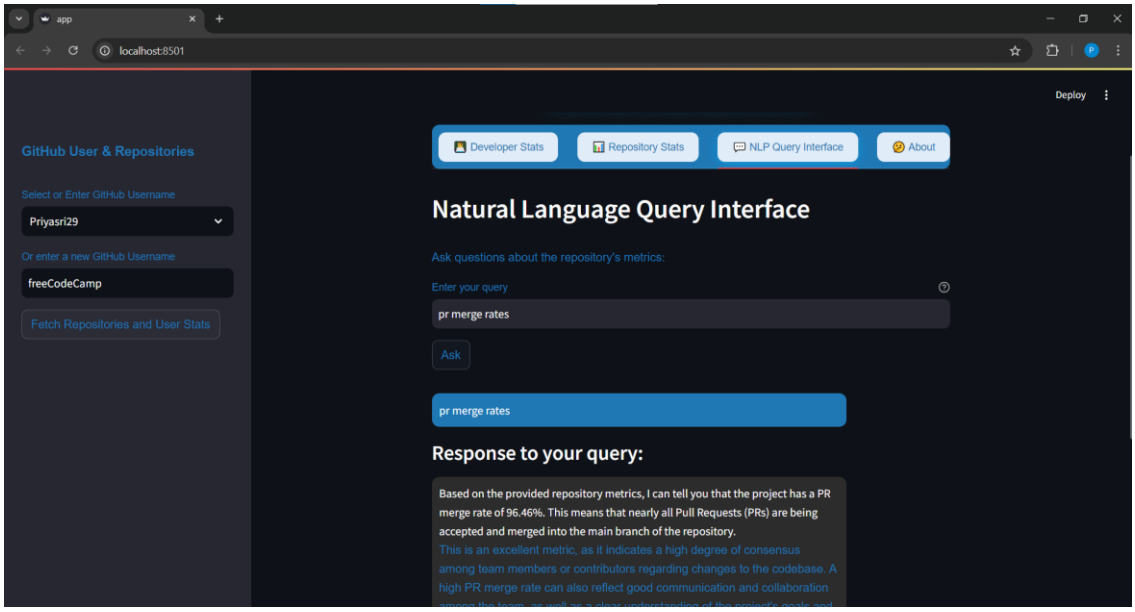



app
localhost:8501
Deploy
GitHub User & Repositories
Select or Enter GitHub Username
Priyasri29
Or enter a new GitHub Username
freeCodeCamp
Fetch Repositories and User Stats
Most Used Languages
Most Used Languages
TypeScript 15.5%
CSS 11.6%
HTML 7.89%
Ren'Py 5.52%
Ruby 4.25%
Dart 1.47%
Python 1.23%
JavaScript 47.2%
Less 0.0000988%
Procfile 0.000178%
Objective-C
Languages
JavaScript
TypeScript
CSS
HTML
Ren'Py
Ruby
Dart
Python
MDX
Nunjucks
HCL
Shell
SCSS


app
localhost:8501
Deploy
GitHub User & Repositories
Select or Enter GitHub Username
Priyasri29
Or enter a new GitHub Username
freeCodeCamp
Fetch Repositories and User Stats
Developer Stats
Repository Stats
NLP Query Interface
About
Repository Information
Select a repository
freeCodeCamp/100DaysOfCode-discord-bot
Fetch Metrics
Repository Information Overview
{
"Name" : "freeCodeCamp/100DaysOfCode-discord-bot"
"URL" : "github.com/freeCodeCamp/100DaysOfCode-discord-bot"
"Stars" : 68
"Forks" : 30
"Open Issues" : 0
"Watchers" : 11
"Language" : "TypeScript"
"Created At" : "2021-06-07 21:32:36"
"Updated At" : "2023-10-25 09:45:57"
"Last Pushed At" : "2022-06-20 07:18:03"
Type here to search
28°C
ENG
11:03
16-09-2024


app
localhost:8501
Deploy
GitHub User & Repositories
Select or Enter GitHub Username
Priyasri29
Or enter a new GitHub Username
freeCodeCamp
Fetch Repositories and User Stats
Repository
PR MERGE RATE (%)
0
20
40
60
80
100
96.5
Issue Resolution Time by Repository
2.268199


app
localhost:8501
Deploy
GitHub User & Repositories
Select or Enter GitHub Username
Priyasri29
Or enter a new GitHub Username
freeCodeCamp
Fetch Repositories and User Stats
Issue Resolution Time by Repository
2.268199
Days to Resolve Issues
Days to Resolve Issues
0
0.5
1
1.5
2
1.8
2
2.2
2.4
2.6
freeCodeCamp/100DaysOfCode-discord-bot
Repository
PR Lead Time by Repository
0.699187
PR Lead Time (days)


app
localhost:8501
Deploy
GitHub User & Repositories
Select or Enter GitHub Username
Priyasri29
Or enter a new GitHub Username
freeCodeCamp
Fetch Repositories and User Stats
Developer Stats
Repository Stats
NLP Query Interface
About
Natural Language Query Interface
Ask questions about the repository's metrics:
Enter your query
pr merge rates
Ask
pr merge rates
Response to your query:
Based on the provided repository metrics, I can tell you that the project has a PR merge rate of 96.46%. This means that nearly all Pull Requests (PRs) are being accepted and merged into the main branch of the repository.
This is an excellent metric, as it indicates a high degree of consensus among team members or contributors regarding changes to the codebase. A high PR merge rate can also reflect good communication and collaboration