**Project Summary**

**Problem Statement-**

Identify a Developer’s profile on GitHub, when following attributes are provided:

1. First Name

2. Last Name

3. Location

Once the profile has been identified, extract following attributes from the profile:

1. Public profile details

2. Name of repositories where they are contributing

3. No. of commits made by that developer on those repositories

**Approach-**

• Python was selected as the coding language.

• The GitHub REST and GitHub GraphQL API’s were used to fetch the user data.

• Multithreading was used to improve the time performance.

• Input and output of the program were in the form of a CSV file.

**Understanding the Rate Limit-**

• GitHub limits unauthenticated users to 60 requests per hour and authenticated users to 5000 requests per hour.

• By default, the GitHub REST API returns 30 results per page.

• GitHub’s GraphQL API allows us to return the specified number of results in the range of 1 to 100 per page.

• GitHub REST API limits to only first 1000 results per API search request.

• When the quota of request limit is crossed for the authenticated/unauthenticated users, GitHub stops sending data for that session hour.

• Quota gets renewed only after the next hour session begins.

**Managing the Rate Limit-**

• To get the full quota of 5000 requests per hour, we generated a user token and sent it with the API request.

• The per-page limit was increased to 100 to get 100 results per page instead of 30 results per page.

• In GraphQL with the help of cursors, we managed to traverse to the next pages to get more than 100 results.

• When search results exceeded 100, pagination was used to get the results from the remaining pages.

• We managed to obtain the results of more than 1000 users in a single program execution without breaking the rate limits.

**Input/Output Summary-**

• Input format-‘CSV file’.

• Output format-‘CSV file’.

• API responses were in the form of a JSON object which was parsed into the dictionary, and the data in the dictionary were dumped into the output file.

• The output file was sorted alphabetically to make result search easier.

**Assumptions-**

• Maximum 200 returns were permitted for a particular First Name, Last name, and Location.

• This limit was set to fetch records of maximum users present in the input file.

**Additional Parameters-**

• We can add additional parameters such as ‘Hirable’ if the search is for hirable users to increase the accuracy.

• To search for active users/contributors, we can specify the repository count.

**Future Implementations-**

• We can create a database to store the results of the requests and caching can be used to reduce the API calls of those search whose results are already present within the database.

• Multithreading can be implemented in the GraphQL requests for the repository to increase the time efficiency of the program further.