**Project Report: Retrieving Package Information from GitHub Repositories**

**1. Introduction:**

The purpose of this project is to retrieve package information from GitHub repositories using the GitHub API and store the data in an Excel file. The Python script utilizes the openpyxl library to read and update the Excel file and the requests library to interact with the GitHub API. The key objective is to extract essential details of the repositories, such as name, description, default branch, license, dependencies, dependents, releases, stars, forks, watchers, and contributors.

**2. Setup and Configuration:**

The project begins by importing the necessary Python libraries: openpyxl for handling Excel files and requests for making HTTP requests to the GitHub API. A personal access token is provided to authenticate with the GitHub API. This access token grants the required permissions to retrieve data from repositories.

**3. Excel File and Required Fields:**

The script loads the Excel file named output.xlsx, which is assumed to be present in the specified file path. The active sheet is chosen for processing data. The required\_fields list is defined, containing the fields we wish to extract from each GitHub repository, namely 'name', 'description', and 'default\_branch'.

**4. Retrieving Package Information:**

The script iterates through each row of the Excel sheet, retrieving the GitHub link for each repository. It checks if the link starts with 'http://' or 'https://', and if not, it adds 'http://' as a prefix. This ensures the link is valid for the GitHub API.

The GitHub API URL is constructed by replacing 'github.com' with 'api.github.com/repos' in the GitHub link. The script then sends an HTTP GET request to the GitHub API, passing the personal access token in the headers for authentication.

**5. Handling API Responses:**

The script handles potential errors in API responses using try-except blocks. It checks for HTTP errors and JSON parsing errors. If the API request is successful, the JSON response is parsed, and the required package information is extracted for further processing.

**6. Extracted Package Information:**

The script extracts the desired package information from the JSON response and stores it in a dictionary named package\_info. Information such as name, description, default\_branch, license name, dependencies count, dependents count, dependent repositories count, total releases, latest release date, first release date, stars count, forks count, watchers count, and contributors count are collected.

**7. Displaying Package Information:**

For each repository, the extracted package information is displayed on the console, providing visibility into the collected data for verification purposes. The information is displayed with appropriate headings, making it easy to understand and review.

**8. Updating Excel Sheet:**

After retrieving and displaying the package information, the script updates the corresponding cells in the Excel sheet. The package information and other details are written to the respective columns, enabling further analysis and tracking.

**9. Error Handling:**

If there is an error in the API response or JSON parsing, the script sets 'NULL' values in the Excel sheet for the corresponding row. This ensures that even if some data is missing or unavailable, the script continues to process the remaining repositories.

**10. Saving the Modified Excel File:**

Once all the repositories have been processed, the modified package information is saved to a new Excel file named output.xlsx. This file contains the additional package details collected from the GitHub API.

* **Description of code snippet 2:**

(add link from s3 bucket)

* This code snippet expands the "Go Packages Details Retriever" tool to fetch additional package information from the GitHub API and update the Excel sheet accordingly. Here's a detailed explanation of the code:
* The code sets the access\_token variable with your personal access token to authenticate requests to the GitHub API.
* It loads the previously generated Excel file containing Go package details into the workbook and accesses the active sheet.
* The required\_fields list is defined, specifying the package information to be fetched from the GitHub API.
* The code iterates through each row (package) in the Excel sheet using enumerate() and retrieves the GitHub link for each package.
* It constructs the GitHub API URL for the package's repository by replacing "github.com" with "api.github.com/repos" in the URL.
* The code then makes a GET request to the GitHub API, passing the access\_token in the headers for authentication.
* If the API call is successful (status code 200), it parses the JSON response into the repo\_info dictionary.
* The desired package information, such as name, description, default branch, and license, is extracted from the repo\_info dictionary.
* Additional package statistics, such as the number of dependencies, dependents, repositories
* The Output is Finally printed into an excel sheet, with the required details copied from the github API repository for all the package links given in the input excel sheet from the previous code.

**11. Flow Chart:**

A screenshot of a computer screen

Description automatically generated

* **Output:**

A screenshot of a computer

Description automatically generated

* **Challenges Faced:**
* API Rate Limits: When querying the GitHub API, rate limits may be encountered. Handling rate limits requires implementing rate-limiting strategies to avoid exceeding the allowed number of requests per hour.
* GitHub Repository Variations: GitHub repositories may have variations in naming and structure, requiring robust handling to ensure accurate data retrieval for all types of repositories.
* Authentication and Security: Managing personal access tokens securely and ensuring they are used correctly for API authentication is crucial for the security of the project.
* Data Inconsistencies: The data retrieved from GitHub repositories may not always be consistent or up-to-date, leading to potential inaccuracies in the collected information.
* **Improvements:**
* Pagination Support: Implementing pagination support for repositories with many stars, forks, or contributors would enhance the efficiency of data retrieval.
* Enhanced Error Handling: Implementing more comprehensive error handling and logging mechanisms to provide detailed information about encountered errors for easier debugging.
* User Input Validation: Implementing input validation to ensure that the Excel sheet contains valid GitHub links before making API requests.
* Data Preprocessing: Adding data preprocessing steps to clean and standardize GitHub links and repository names could improve data consistency.
* Interactive User Interface: Developing an interactive user interface that allows users to provide input and visualize the data in a more user-friendly manner.

**12. Here are a couple of questions we can answer based on the data retrieved:**

* **Top 10 repos in terms of the Number of watchers**

A graph with different colored bars

Description automatically generated

* **Top 10 repos in terms of the Number of Forks**

A graph showing different colored bars

Description automatically generated

* **Top 10 repos in terms of the Number of Stars**

A graph showing the number of data

Description automatically generated with medium confidence

**13. Conclusion:**

The Python script successfully retrieves package information from GitHub repositories using the GitHub API and stores the data in an Excel file. The project demonstrates the use of external APIs, error handling, data extraction, and manipulation of Excel files. By leveraging this script, users can efficiently gather information about various GitHub repositories and analyze the package details for their projects. A copy of all the codes and output info can be found in this drive link <https://drive.google.com/drive/folders/1aaeilltwH8R4QmTYkcTR8-dYD57iFlQo?usp=drive_link>

**References:**

* GitHub API Documentation: <https://docs.github.com/en/rest>
* openpyxl Documentation: <https://openpyxl.readthedocs.io/en/stable/>
* Python 'requests' Library Documentation: <https://docs.python-requests.org/en/latest/>
* Python Official Website: <https://www.python.org/>
* Stack Overflow and GitHub: For community discussions and code examples related to Python, APIs, and Excel manipulation.