REPORT

SOFTWARE ARTIFACTS SUMMARISATION TOOL

Name :P.Naga Satya Sai
Roll no :CS23M109

1.What is the problem statement?

Problem statement:

The problem statement is to create a tool for the Software Artifacts Summarisation.The tool should be helpful for the tester to identify the closed issues related to testing provided the url of the github repository and it should summarise those issues and should append the summary to the readme file of cloned github repository

2.How have you solved the problem?

I solved the problem by writing the python code with the below functions:

1. Fetching Closed Issues:

- The **fetch_closed_issues** function constructs a URL for the GitHub API to fetch closed issues of a repository.
- It sends a GET request to this URL and returns the JSON response containing closed issues.

2. Printing Unique Closed Issues:

- The **print_unique_issues** function prints unique closed issues sorted by their creation date.
- It uses a set to keep track of unique issue titles and avoids printing duplicates.

3. Generating Testing-Related Issues:

- The **generate_testing_related_issues** function combines the titles and bodies of closed issues into a single string.
- It uses GenAI's **GenerativeModel** to generate content, aiming to identify testing-related issues.

4. Summarising Issues:

- The **summarise_issues** function summarises the generated testing-related issues.
- It uses Transformers' summarization pipeline with the BART model.

5. Cloning Repository:

- The **clone_repo** function clones the GitHub repository to a specified destination path.
- If the destination directory already exists, it removes it first.

6. Appending Summary to README:

- The **append_summary_to_readme** function appends the generated summary to the README file of the cloned repository.

7. Main Function:

- The **main** function integrates all the above functions to execute the entire process:
- Get user input for the GitHub repository URL.
- Parse the URL to extract owner and repository name.
- Fetch closed issues, print them, and generate testing-related issues.
- Summarise the testing-related issues.
- Clone the repository and append the summary to its README.

3.Technical Information - Techniques, libraries used.

Technical Information:
Techniques:

- API Requests
- Natural Language Generation (GenAI)
- Text Summarization (Transformers)
- Version Control (Git)

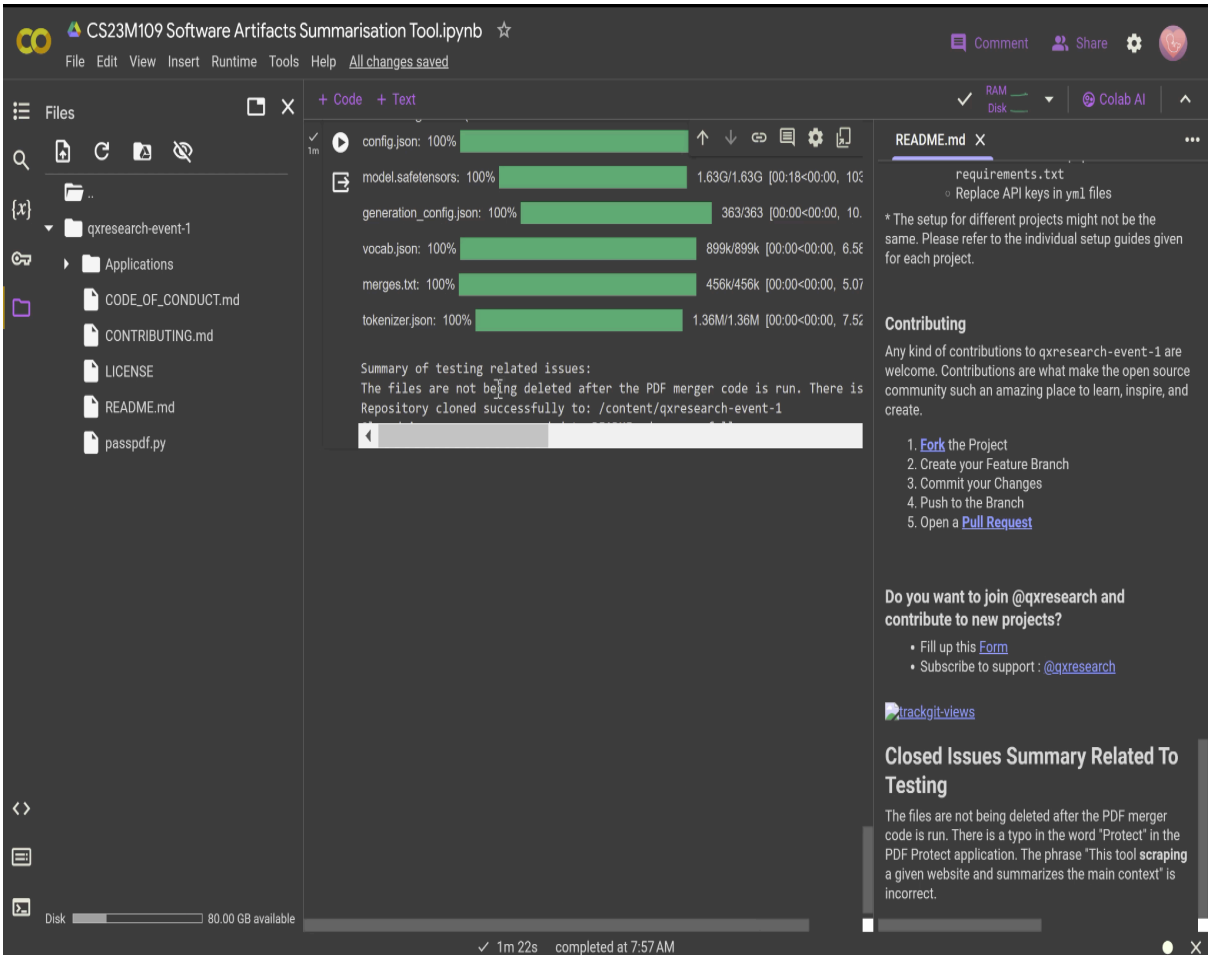
Libraries Used:

- **os:** For operating system dependent functionality.
- **google.generativeai:** GenAI for natural language generation.
- **requests:** For making HTTP requests to the GitHub API.
- **urllib.parse:** For parsing URLs.
- **transformers:** For text summarization using BART model.
- **git:** For cloning GitHub repositories.
- **shutil:** For high-level file operations like removing the directory if it exists.

4.Results (Output)

Results (Output):

- The python script prints unique closed issues sorted by their creation date.
- It identifies and prints testing-related issues.
- A summary of the testing-related issues is generated and printed.
- The GitHub repository is cloned to a local directory.
- The generated summary is appended to the README file of the cloned repository.



5.What new things have you learned.

The new things i have learned are:

- Integrating GenAI by using Gemini LLM model to filter the testing related content.
- Using Transformers for text summarization.
- Using Git Python library for repository cloning.
- Error handling and debugging with Python.
- Parsing and manipulating URLs with **urllib.parse**.