**DOCUMENTATION:**

**OBJECTIVE**

The objective of the project is to determine the common bugs after analyzing pattern of patches for various issue/bug reports.

**I. Creating the environment**

Operating System: The Project has been developed and simulated on Windows environment. It can run in other environment if the Paths are correctly inputted.

**1.Installation:**

**a. Scitools Understand**

1. Kindly download Understand API from scitools website https://scitools.com/non-commercial-license/
2. Keep a note of the directory where the Understand API is downloaded and extracted in your local system.
3. Kindly traverse inside the extracted scitools folder and go to scitools/bin/<”YOUR SYSTEM”> / and check if “und” executable is present.
4. Capture the path/location of the directory of und executable.

**Note**: While executing the project “License Expired” error is thrown. Kindly open understand.exe and enter your license in the License dialog box.

The und Command Prompt is used for the following tasks:

1. Create Understand Database
2. Add project files to the Understand Database
3. Analyze the Database

The remaining tasks of analyzing, generating lexemes, tokens, type and entities of the project is done using Python Understand API.

**b. Python 3.X**

The entire project had been coded in Python. Install the 3.X version of python for the project.

Python libraries used:

* PyGitHub : pip install PyGithub
  + PyGitHub is used for authentication of user using token.
* GitPython : pip install gitpython
  + GitPython is used for git clone requests and checkout.
* Networkx: pip install networkx
  + network is used for graph generation.
* requests: pip install requests
  + Requests is used for handling GET and POST HTTP requests.
* Subprocess: pip install subprocess
  + Subprocess is used for running shell commands through python.

**2. Configuration**

The project has a property file named GitHubVariable.properties that has the following parameters configurable.

* Number\_of\_projects: Set the umber of projects to be cloned
* Number \_of\_commits: Set the number of commit requests to be considered for that project

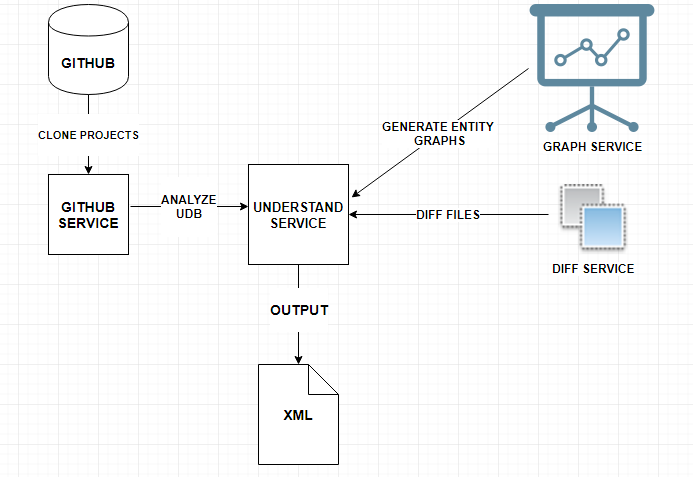
(The above parameters are configurable for reducing the process time )

* Project\_path: Path where the repository is cloned, udbs are generated and xml is created.

**II. ARCHITECTURE**

The project architecture consists of the following components:

1. GitHub repository: The git repository with multiple opensource repositories, projects and patches.
2. GitHubService: GitHubService creates a search for java repositories using GitHub Developer API, obtains a list of relevant repositories. Further for each repository ,a set of issues, commits and other metadata is filtered which has closed issues. The services then fetches a list of pull requests using the HTTP GET requests. The project is then cloned in two directories in the configured path. The service then populates a list of pull requests , corresponding commits and respective. The two cloned project directories are then checked out to the respective version using SHAs. Further consecutive versions of the project are used create Understand databases. Each database is then passed to the Understand Service for further analysis



1. **Understand Service**: Understand Service takes input of two understand databases. We analyze the project in two levels namely File level changes and class level changes. File level changes include changes when files have been added, modified or deleted. When classes are it checks for added or removed we check what classes were added or removed. When classes are modified it checks what are the changes in between the two files. We generate lexemes for the two versions of understand database and find all the differences in a class.
2. **Graph Service**: It generates a graph for entities of each version of understand database.
3. **Diff Service**: It checks for differences for two list of lexemes generated from two versions of understand databases.
4. **Changes XML:** Outputs the different changes encountered for each patch and the changes are additions, deletions and modifications to the following constructs:-
5. Object Creation
6. Variable Definition
7. If-statement
8. For-statement
9. While statement
10. Do statement
11. Switch
12. Method
13. Class
14. Inheritance
15. Exception handling
16. Dependency
17. Variable definition

**III. ALGORITHM**

We pass understand databases for two consecutive versions to the understand service which generates the entity relationship graph for every class. At every step we compare two versions of the graph for a class, before and after the patch. With this information we analyze and output the changes.

We also generate lexemes of each class and compare both versions using the diff service. This gives us more fine grained differences between the classes. Next we perform comparisons at a line level to obtain more information about modifications.

Finally the above three methods are used to obtain all the changes.

**IV.HOW TO RUN**

Run the file main.py

Once the execution is complete, to view the metrics execute the following command

Python XMLComparator.py

Sample output:

{'objectCreation': 10, 'tryblock': 6, 'method': 6, 'dependency': 4, 'variableDefinition': 4}

**V. HOW TO TEST**

Run python testsuite.py

**VI. OUTPUT**







