**Developer Tool Project**

**Implementation Steps:**

**Step 1:**

First, developer should have access to our project repository, that he can be able to push or download the code or files.

Detailed Steps:

1. The Code Can be either from the GIT Repository or through the SharePoint Link.
2. Developer needs to clone/download the project code into their local
3. Developer should have read and write access to perform cloning/downloading

**Step 2: Downloading/Cloning Developer tool project:**

User/developer should be able to download/clone developer tool project.

Detailed Steps:

1. Project code contains below shown directories and programs in it.

Example:

Graphical user interface, text, application

Description automatically generated

**Step 3: Writing a Feature:**

User should be able to write feature function for the input

Detailed Steps:

1. Develop a feature function for the input
2. Use comments while writing function definition for everybody understanding
3. Do not use lower functions for whole input string in writing features. We can use in that lower function in conditions.
4. After writing feature function, put that code in a python file which is Module
5. Module name should be same as name of the feature
6. Use input string and schema name parameters while developing feature, as we are using schema name parameter while calling features in driver function

**Example Snippet:**

In the below code snippet, we have created a module with name same as feature function name

We have kept this code into a python to make it a module, so that we can import this anywhere in our project and run independently.

This module is to handle the oracle input, where the data present in between when and then should be replaced with specific data

We are specifying parameter Schema name in below and fetched it from command line using parser.

Module Name: when\_then\_inbetween.py

Text

Description automatically generated

**Step 4: Placing the modules:**

Developed modules should be placed in the “Modules” directory present in project shown below.

Example path: Project

/Code

/Modules

/module1.py--🡪feature1

/module2.py--🡪feature2

Graphical user interface, text, application

Description automatically generated

**Step 5: Placing the input:**

Input file which consists of oracle code (any objects) for which user intended to apply the features, should be placed in Source directory

Example path: Project

/Source

/Input file

Graphical user interface, application

Description automatically generated

**Step 6: Running Driver Program:**

We have written a driver program to read the input from Source directory, handle comments and apply all the features present in modules directory to the given input.

This driver program will be called by main function while running,

Example command: python main.py -o schema\_name

The output will be stored in output file created in Target directory

Graphical user interface, application

Description automatically generated

**Example Code Snippets:**

This driver program will read the list of modules which is present in “Modules” directory.

**Comment Handling:**

* + We are handling comments which are present in input using “commentmaskwithvariable” function internally.
  + Here comments will be replaced with keywords and stored in dictionary key value pairs.
  + Later, once we get the output after applying features, we will be calling “commentmaskreleaase” function which will replace all handled comments back using dictionary.

text, dictdata = commentmaskwithvariable(text)

applying features

output = commentmaskrelease (output, dictdata)

Reading the input file data and applying features which is fetched in above step.

We are importing these modules by importlib package

**Step 7: Checking the Output:**

Output is stores in output file which will be created (if it is not present) in Target directory under the project.

Example path: Project

/Target

/Output file

Graphical user interface, text, application

Description automatically generated

**Code Snippet:**

Here, once we get output after applying features to input, will place that in Target directory in output file.

Even if there is no output file in Target, this will create file and place that output in it

**Step 8:**

Once developer thinks that developed feature is working properly, then he needs to push that code to our repository which can be accessed by all the members of our team.

Later, upon approval or after testing from team lead, this new feature will be moved to our main framework project.

**Structure of this Project implementation:**

Main function 🡪 Driver Function 🡪 Reading Input 🡪Comment Handling 🡪Applying Features 🡪 Comments release 🡪 Store output in Target

**Regular expressions code useful in writing features:**

phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d') Regex to get the digits

haRegex = re.compile(r'(Ha){3}') Regex to repeat the number of search in the statements.

data = re.sub(r"\b(?<!')(\w+)(?!')\b", lambda match: match.group().lower(), data) Converts everything into lower case except the data in single quotes.

begin\_when\_data = re.search(r'\bbegin(.\*?)when\b', removed\_single\_line\_str,

re.DOTALL).group(1) Search the data between begin and when

open\_execute = re.findall(r'open\s+\S+\s+for\s+execute\s+select', excute\_str) finging the data from open to select.

begin\_end\_data\_str = re.findall(r'\bbegin\b.\*?\bend\b', data\_str, re.DOTALL) taking data from begin to end including the begin and end.

dir\_all = re.findall(r'\/\\*[\s\S]\*?\\*/', single\_line\_text) taking the data from /\* to \*/

[abc] – will find all occurrences of a or b or c.

[a-z] will find all occurrences of a to z.

[a-z0–9A-Z] will find all occurrences of ato z , A to Z, o to 9.

pattern = '\d+' has all the patterns for digits,

re.sub() used for replacing stuff.

\s searches for spaces

\S searched for non spaces