**Instruction**

This document explains in detail, the test plan for the MU GitHub Analyzer.

**Test Scope**

We will be testing all the top priority requirements for MU GitHub Analyzer version 1.0. All functions will be tested by the developers, Liam Innes and Swarna Muralidharan. During development, user acceptance testing may take place with Mike Bush at MITRE.

**Test Plan**

The following is a table of test cases that will determine the working functionality of what MUGHA is supposed to do. Test cases were derived directly from the requirements with a test number correlating to their entry in the Software Requirements Specification document.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Title | Precondition | Input | Activity | Output |
| 3.1.1 | Single interactive user interface |  | Run the program |  | A User Interface pops up |
| 3.2.1.2.1 | Search GitHub Repositories | User is in the search tab | 2. Click Search | MUGHA searches GitHub | Repositories appear in the list |
| 3.2.1.2.2 | Search based on programming language | User is in the search tab | 1. Click the programming language drop-down menu  2. Click Java  3. Click Search | MUGHA searches GitHub | Repositories appear in the list and it says they all use Java |
| 3.2.1.2.3 | Search based on Number (range) of contributors. | User is in the search tab | 1. Enter 2 in the minimum textbox for contributors  2. Enter 3 in the maximum textbox  3. Click Search | MUGHA searches GitHub | Repositories appear in the list and it says they all have 2 or 3 contributors |
| 3.2.1.2.3.2 | Search contributors with blank maximum text box | User is in the search tab | 1. Enter 5 in the minimum textbox for contributors  2. Click Search | MUGHA searches GitHub | Repositories appear in the list and it says they all have 5 or more contributors |
| 3.2.1.2.3.3 | Search contributors with blank minimum text box | User is in the search tab | 1. Enter 3 in the maximum textbox for contributors  2. Click Search | MUGHA searches GitHub | Repositories appear in the list and it says they all have 3 or less contributors |
| 3.2.1.2.5 | Search based on Number (range) of commits. | User is in the search tab | 1. Enter 5 in the minimum textbox for contributors  2. Enter 20 in the maximum textbox  3. Click Search | MUGHA searches GitHub | Repositories appear in the list and it says they all have between 5 and 20 commits |
| 3.2.1.2.5.2 | Search commits with blank maximum text box | User is in the search tab | 1. Enter 20 in the minimum textbox for contributors  2. Click Search | MUGHA searches GitHub | Repositories appear in the list and it says they all have 20 or more commits |
| 3.2.1.2.5.3 | Search commits with blank minimum text box | User is in the search tab | 1. Enter 5 in the maximum textbox for contributors  2. Click Search | MUGHA searches GitHub | Repositories appear in the list and it says they all have 5 or less commits |
| 3.2.1.2.7 | Search based on Date (range) of latest commit. | User is in the search tab | 1. Enter 01/01/2017 in the date box for latest commit  2. Enter 01/01/2016 in the date box for earliest commit  3. Click Search | MUGHA searches GitHub | Repositories appear in the list and they have all been last committed to in 2016 |
| 3.2.1.2.7.2 | Search based on Date with blank latest date | User is in the search tab | 1. Enter 01/01/2016 in the date box for latest commit  2. Click Search | MUGHA searches GitHub | Repositories appear in the list and they have all been last committed to before 2016 |
| 3.2.1.2.7.3 | Search based on Date with blank earliest date | User is in the search tab | 1. Enter 01/01/2016 in the date box for earliest commit  2. Click Search | MUGHA searches GitHub | Repositories appear in the list and they have all been last committed to between 2016 and now |
| 3.2.1.11 | Has a search menu |  | Clicks the Search tab from another tab |  | The search menu shows up |
| 3.2.2.1 | Displays the candidate pool | User has repositories in the Candidate Pool | Click the Candidate Pool tab from another tab |  | The candidate pool list appears |
| 3.2.2.3 | Creating a new candidate pool | User has search results in the search tab | 1. Add search results to candidate pool  2. Save the candidate pool |  | The user has a file for their candidate pool |
| 3.2.2.5 | UI shall display list of repositories based on search criteria and display the metrics found in static analysis. |  | Open a candidate pool | MUGHA accesses GitHub for up-to-date information on the list items | Attributes of the repositories appear on the list |
| 3.2.2.7 | Has a candidate pool menu |  | Click the Candidate Pool tab from another tab |  | There are buttons for 'Remove Selection', 'Static Analysis', and 'Import Local Repository' |
| 3.2.2.7.3 | Has a static analysis button |  | Click the Candidate Pool tab from another tab |  | There is a button for Static Analysis |
| 3.2.2.10 | Update candidate repositories during static analysis. | There are repositories in the candidate pool | Perform static analysis on a repository |  | All information about the repository becomes up to date |
| 3.2.2.13 | Remove candidate repositories. | There are repositories in the candidate pool | 1. Select a repository  2. Click 'Remove Selection' |  | The repository is removed from the pool |
| 3.2.3.1 | Import repository information |  | 1. Click 'Import Local Repository'  2. Select local repository in file manager | MUGDHA performs static analysis on the repository | A new repository appears in the candidate pool with the project attributes filled out |
| 3.2.5.2.1 | Query projects based on project ID | The user should be in MUGHA User Interface | Enter project ID | 1.Enter the project ID and click search.  2. Display all the attributes related to that project in the database. | Project details appear in the list |
| 3.2.5.2.1 | Query projects based on project ID |  | Enter invalid project ID | 1.Enter an invalid Project ID and click search.  2. Error message displayed from database | Error message displayed. |
| 3.2.5.2.2 | Query projects based on programming language | The user should be in MUGHA User Interface | Enter programming language | 1. Enter programming language to be queried.  2. Display projects details based on programming language entered from database | Project details appear in the list |
| 3.2.5.2.3 | Query projects based on number of contributors | The user should be in MUGHA User Interface | Enter number of contributors | 1. Enter number of contributors to be queried.  2. Display projects details based on number of contributors entered from database | Project details appear in the list |
| 3.2.5.2.4 | Query projects based on number of pull requests | The user should be in MUGHA User Interface | Enter number of pull requests | 1. Enter number of pull request to be queried.  2. Display projects details based on number of pull requests entered from database | Project details appear in the list |
| 3.2.5.2.6 | Query projects based on number of commits | The user should be in MUGHA User Interface | Enter number of commits | 1. Enter number of commits to be queried.  2. Display projects details based on number of commits from database | Project details appear in the list |
| 3.2.5.2.7 | Query projects based on number of releases | The user should be in MUGHA User Interface | Enter number of releases | 1. Enter number of releases of project to be queried.  2. Display projects details based on number of releases from database | Project details appear in the list |
| 3.2.5.2.8 | Query projects based on date of latest commit | The user should be in MUGHA User Interface | Choose latest commit date | 1.Choose the latest commit date.  2. Display projects details based on date of latest commit from database | Project details appear in the list |
| 3.3.2.1.1 | Project file of every iteration of main branch received from GitHub |  | All files for a Git project | 1. Connect to Git  2. Clone the project | Clone the project from Git is done |
| 3.3.2.1.2 | A unique identifier is received from GitHub | Project exists in Git |  | 1. Connect to Git  2.Retrieve unique identifier from GitHub. | Unique identifier is stored in database |
| 3.3.2.1.3 | Number of contributors is received from GitHub | Project exists in Git | Download the number of contributors for a chosen project or projects | 1. Connect to Git  2. Download number of contributors for the chosen Git Project | Number of contributors for a project will be retrieved from Git |
| 3.3.2.1.4 | Number of pull request is received from GitHub | Project exists in Git | Download the number of pull requests for a chosen project or projects | 1. Connect to Git  2. Download number of pull request for the chosen Git Project | Number of pull requests for a project will be retrieved from Git |
| 3.3.2.1.5 | Number of commits is received from GitHub | Project exists in Git | Download the number of commits for a chosen project or projects | 1. Connect to Git  2. Download number of commits for the chosen Git Project | Number of commits for a project will be retrieved from Git |
| 3.3.2.1.6 | Date of latest commit is received from GitHub | Project exists in Git | Retrieve the date of latest commit from Git | 1. Connect to Git  2. Retrieve date of latest commit from Git | Date of latest commit is received from Git |
| 3.3.2.2 | Retrieve the information in one instant. |  | Retrieve the information from Git in (TBD) minutes | 1. Connect to git  2. Retrieve information from Git | Retrieve information in (TBD) minutes |
| 3.3.2.3 | Store information in database | Database is up and running | Project attributes retrieved from Git | 1. Connect to DB  2. Establish a DB connection  3. Persist project attributes in DB | Database table has project attributes |
| 3.3.3.1 | Database storage interface will be used to store and retrieve data | Database is up and running | Project attributes retrieved from Git | 1. Connect to DB  2. Establish a DB connection  3. Persist or retrieve project attributes in DB | Project attributes are stored in database or Project attributes  are retrieved from database |
| 3.3.3.2.2 | The project unique identifier will be stored in database | Database is up and running | Project unique identifier from Git | 1. Connect to DB  2. Establish a DB connection  3. Persist project unique identifier in DB | Project unique identifier are stored in database |
| 3.3.3.3 | MUGHA shall be able to retrieve information from database | Database is up and running | Userid, Password for DB | 1. Connect to DB  2. Establish a DB connection  3. Retrieve project attributes from DB | Project attributes from DB are provided to MUGHA |
| 3.10.4.1.1 | Failed access to connect to GitHub error message | The user has not signed in to GitHub on MUGHA yet | 1. Open a candidate pool or attempt to search.  2. Enter a wrong username and password | 1. Connect to DB  2. Establish a DB connection  3. Persist project unique identifier in DB | An error message indicates that the username and password didn't work |

**Test Inputs/Outputs**

We will be holding on to the test results and defect log which are found during our testing.

|  |  |
| --- | --- |
| Inputs | Outputs |
| SRS 02 | Test Results |
|  | Defect log |

**Test Strategy**

This describes “how” the test items and features will be tested.

|  |  |  |
| --- | --- | --- |
| **Test Type** | **Description** | **Comment** |
| Unit Test | This type is done while working on single units. | This is performed by the programmers. |
| Integration Test | This type tests all changed functionality end to end. |  |
| User Acceptance Test | Typically performed by Business User, this tests all changed functionality of a system from the end-user perspective. |  |

**Test Environment (tools, system, databases)**

Testing will be performed with Eclipse 4.6 and running the program on Windows systems though the program. For unit testing, we will be executing JUnit test cases in Eclipse. For testing the database we will be executing test cases manually and observing the execution of the test cases.

**Test Roles and Responsibilities**

Liam Innes and Swarna Muralidharan will collaborate on the same test roles and responsibilities with a preference toward testing features they implemented individually.

**Test Milestones**

This is a list of the critical test milestone dates. Be sure to include these dates in the overall project plan.

|  |  |
| --- | --- |
| **Task** | **Milestone Date** |
| Setup Test Environment | February 10 2017 |
| Complete Unit Testing | March 20 2017 |
| Complete Integration Testing | April 20 2017 |
| Complete User Acceptance Testing | May 05 2017 |