Team: McBain Fire Alarm

Team Members: Arsalaan Ansari (aaa2325) Trevor Jones (tcj2110)

Raphael Owino (ro2295) Kevin Mejia (km3204)

Part 0:

Language: Python

Platform: Mac OS

Part 1:

Our team wants to work on a plugin for an already existing text editor named Sublime Text. The purpose of this plugin is to allow users of the popular version control platform GitHub to view the GitHub Issues and code review comments on any open pull requests in their GitHub repository within the user interface of the Sublime Text editor. The plugin will extend an existing source version control plugin to avoid writing code to commit, push, monitor and sync the repo state. The creation of this plugin will require a permanent data store that can store user credentials, and preferences for how their GitHub Issues and code review comments will be presented to avoid users having to set up their preferences every time on startup. Additionally, this plugin will make use of API's heavily, specifically the Sublime Text API and the GitHub API. The Sublime Text API will be used to build the base plugin to ensure compatibility with the Sublime Text Editor Platform. The GitHub API will be used to retrieve the user's GitHub information, including but not limited to their GitHub issues, pull requests, code review comments, and commits.

Part 2:

<Front end>: As a <software developer>, I want <to query all of my open issues on a GitHub repo> so that <I may view them in the Sublime Text editor>

My conditions of satisfaction are <

The plugin can login to a GitHub repo,

The plugin can query GitHub Issues,

The plugin can query GitHub Issues assigned to me,

The plugin can query GitHub Issues which are open,

The plugin can query the Github repo's pull requests and their status,

The plugin can query the Github repo's pull requests commits and comments.

>

<Front end>: As a <software developer>, I want <to be able to open and close GitHub Issues, Pull review commit coments in my Sublime Text editor> so that <I don't have to open GitHub.com in a browser to close or view the issue there>

My conditions of satisfaction are <

The plugin can query open GitHub Issues and pull review comments,

The plugin can close an open GitHub Issue,

The plugin can submit responses to comments on open pull requests,

The plugin can read GitHub Issues in the Sublime Text Editor,

The plugin can close a GitHub Issue in the Sublime Text Editor

>

<Front end>: As a <software developer>, I want <to be able to add comments, open new GitHub Issues in my Sublime Text editor> so that <I don't have to open GitHub.com in a browser to add new issues, comments there>

My conditions of satisfaction are <

The plugin can send data to a GitHub repo,

The plugin can submit comments to open pull requests,

The plugin can add new GitHub Issues to a GitHub repo

>

<Front end>: As a <software developer>, I want <to be able to save my user credentials> so that < the plugin can automatically login to my GitHub.com account>

My conditions of satisfaction are <

The plugin can query a database (SQL or NoSQL tbd),

The plugin can insert data into a database,

The plugin can prompt the user for credentials,

The plugin can verify user credentials

The database stores the data persistently (in-memory, commitlog, etc.)

The plugin can make an oath authentication request to GitHub

>

<Front end>: As a <software developer>, I want <to be able to change the layout of how GitHub Issues are presented to me in the Sublime Text editor > so that < I can interact with my GitHub Issues according to my preferences.>

My conditions of satisfaction are <

The plugin can query data pertaining to the layout of text in Sublime Text Editor,

The plugin can modify data pertaining to the layout of text in Sublime Text Editor

>

<Front end>: As a <software developer>, I want <to be able to save the layout preferences of how GitHub Issues are presented to me in the Sublime Text editor> so that <I do not have to set up the configuration on each startup>

My conditions of satisfaction are <

The plugin can query a database,

The plugin can insert data into the database,

The database stores the data persistently

>

NB: GitHub Issues may include but not limited to code review comments to pull requests and subsequent commits to that particular pull request.

Part 3:

<Front end>: As a <software developer>, I want <to query all of my open issues on a GitHub repo> so that <I may view them in the Sublime Text editor>

My conditions of satisfaction are <

The plugin can login to a GitHub repo,

The plugin can query GitHub Issues,

The plugin can query GitHub Issues assigned to me,

The plugin can query GitHub Issues which are open,

The plugin can query the Github repo's pull requests and their status,

The plugin can query the Github repo's pull requests commits and comments.

>

Unit and Integration testing: test that each function which fulfills a condition of satisfaction passes

- Test that we can communicate with the GitHub servers
- Test that we can successfully login to an existing GitHub repository
- Test that we can successfully query GitHub issues for a given repository
- Test that we can successfully query which GitHub Issues are open, if any
- Test that we can successfully query the GitHub repository's pull requests
- Test that we can successfully query the status of pull requests from a GitHub repository
- Test that we can successfully query the commits and comments of a GitHub repository

Test-to-pass:

- Given a valid input, login to an existing GitHub repository successfully
- Given a GitHub repository has existing Issues, query those Issues
- Given a GitHub repository has existing pull requests, query those pull requests and their status
- Given a GitHub repository has commits and comments, query those commits and comments

Test-to-fail: the program should handle the following without crashing or bugs

- Given an invalid input, attempt to login to an existing GitHub repository
- Given any or no input, attempt to login to a nonexistent GitHub repository
- Given a GitHub repository does not have existing Issues, attempt to query those Issues
- Given a GitHub repository does not have existing pull requests, attempt to query those pull requests and their status
- Given a GitHub repository does not have existing commits or comments, attempt to query those commits or comments

<Front end>: As a <software developer>, I want <to be able to open and close GitHub Issues, Pull review commit coments in my Sublime Text editor> so that <I don't have to open GitHub.com in a browser to close or view the issue there>

My conditions of satisfaction are <

The plugin can query open GitHub Issues and pull review comments,

The plugin can close an open GitHub Issue,

The plugin can submit responses to comments on open pull requests,

The plugin can read GitHub Issues in the Sublime Text Editor,

The plugin can close a GitHub Issue in the Sublime Text Editor

>

Unit and Integration testing:

- Test that we can query GitHub Issues for a given repository
- Test that we can query open GitHub Issues
- Test that we can pull comments on GitHub Issues
- Test that we can pull comments on open GitHub Issues
- Test that we can close an open GitHub Issue
- Test that we can query open pull requests
- Test that we can query comments on open pull requests
- Test that we can submit responses to comments on open pull requests
- Test that we can parse data from GitHub Issues in the Sublime Text Editor
- Test that we can format and output data from GitHub Issues in the Sublime Text Editor
- Test that we can close a GitHub Issue from the Sublime Text Editor

Test-to-pass:

- Query the number of GitHub Issues (should be 0 or greater)
- Query the number of open GitHub Issues (should be greater than or equal to number of Issues)
- Query the number of comments on a GitHub Issue (should be 0 or greater)
- Query the number of open pull requests (should be 0 or greater)
- Test whether we successfully close an open GitHub Issue

Test-to-fail:

- Attempt to close a closed or nonexistent GitHub Issue
- Attempt to submit a comment in an unsupported type (if possible), or a comment that otherwise does not conform to GitHub requirements (ie too long)

<Front end>: As a <software developer>, I want <to be able to add comments, open new GitHub Issues in my Sublime Text editor> so that <I don't have to open GitHub.com in a browser to add new issues, comments there>

My conditions of satisfaction are <

The plugin can send data to a GitHub repo,

The plugin can submit comments to open pull requests,

The plugin can add new GitHub Issues to a GitHub repo

>

Unit and Integration testing:

- Test whether we can communicate with GitHub server
- Test whether we can login to an existing GitHub repository
- Test whether we can query comments in an existing GitHub repository
- Test whether we can send data to an existing GitHub repository
- Test whether we can submit a comment to an existing GitHub repository
- Test whether we can query pull requests in an existing GitHub repo
- Test whether we can query open pull requests in an existing GitHub repo
- Test whether we can query comments of pull requests in an existing GitHub repo
- Test whether we can query comments of open pull requests in an existing GitHub repo
- Test whether we can submit comments to open pull requests in an existing GitHub repo
- Test whether we can query GitHub Issues in an existing GitHub repository
- Test whether we can add a new GitHub Issue in an existing GitHub repository

Test-to-pass:

- Send properly formatted data to an existing GitHub repository
- Submit properly formatted comment to open pull request in existing GitHub repo
- Add new GitHub Issue in an existing GitHub repository

Test-to-fail:

- Attempt to send improperly formatted data to an existing GitHub repo
- Attempt to send data to a nonexistent GitHub repo
- Attempt to submit improperly formatted comment to open pull request in existing GitHub repo
- Attempt to add new GitHub Issue in a nonexistent GitHub repository

<Front end>: As a <software developer>, I want <to be able to save my user credentials> so that < the plugin can automatically login to my GitHub.com account>

My conditions of satisfaction are <

The plugin can query a database (SQL or NoSQL tbd),

The plugin can insert data into a database,

The plugin can prompt the user for credentials,

The plugin can verify user credentials

The database stores the data persistently (in-memory, commitlog, etc.)

The plugin can make an oath authentication request to GitHub

>

Unit and Integration testing:

- Test whether database exists
- Test whether we can query information from an existing database
- Test whether we can insert data into database
- Test whether we can create a prompt for user credentials
- Parse user input from aforementioned prompt and test membership in database
- Test whether the data is stored persistently (ie from session to session, no power)
- Test whether we can submit an oath authentication request to GitHub

Test-to-pass:

- If database exists, successfully query information (ie conduct a membership test on a known item in the database)
- Insert properly formatted data into database, then test for membership
- Input test strings into user input and verify that we are parsing the strings correctly
- Test membership in database of valid user input

- Test whether known data still exists in database from session to session
- Test if properly formatted oath authentication request to GitHub passes

Test-to-fail:

- Attempt to query information from nonexistent database
- Attempt to insert improperly formatted data into database; test for membership
- Test membership in database of invalid user input
- Attempt to submit improperly formatted oath authentication request to GitHub

<Front end>: As a <software developer>, I want <to be able to change the layout of how GitHub Issues are presented to me in the Sublime Text editor > so that < I can interact with my GitHub Issues according to my preferences.>

My conditions of satisfaction are <

The plugin can query data pertaining to the layout of text in Sublime Text Editor,

The plugin can modify data pertaining to the layout of text in Sublime Text Editor

>

Unit and Integration testing:

- Test whether we can query data pertaining to layout of text in Sublime Text Editor
- Test whether we can parse data pertaining to layout of text in Sublime Text Editor
- Test whether we can modify data pertaining to layout of text in Sublime Text Editor (this may require removing an item, then inserting a new item)

Test-to-pass:

- Modify data pertaining to layout of text in Sublime Text Editor using properly formatted input

Test-to-fail:

- Attempt to modify data pertaining to layout of text in Sublime Text Editor using improperly formatted input

<Front end>: As a <software developer>, I want <to be able to save the layout preferences of how GitHub Issues are presented to me in the Sublime Text editor> so that <I do not have to set up the configuration on each startup>

My conditions of satisfaction are <

The plugin can query a database,

The plugin can insert data into the database,

The database stores the data persistently

>

Unit and Integration testing:

- Test whether we can query an existing database
- Test whether we can insert data into an existing database
- Test whether data is stored in the database persistently (Mocks are called)

Test-to-pass:

- Test membership of layout preference items in existing database
- Insert properly formatted data into existing database
- Test membership of known elements of database from session to session (ie tests persistent storage)

Test-to-fail:

- Attempt to test membership of items in nonexistent database
- Attempt to insert improperly formatted data into an existing database
- Attempt to insert data into a nonexistent database