Team: McBain Fire Alarm

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

Raphael Owino (ro2295) Kevin Mejia (km3204)

1)

Date and time of Demo: November 8th 12:15 PM

We were able to successfully show our backend that allows our Sublime Text plugin to retrieve Github Pull Requests. Additionally, our backend was able to successfully authenticate a Github username and token to access private Github repositories. On the front end we showed the panel interface that allowed the user to scroll through Github Pull Requests. We had an error between the front end and the backend so the Pull Request data wasn't populated correctly in the front end panel.

2)

User stories and updates: **Every condition of satisfaction that has been bolded has been implemented in the First Iteration.**

<Front end>: As a <software developer>, I want <to query all of my open pull requests 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 comments 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

>

For the first iteration we were able to implement all conditions of satisfaction relating to querying pull request data. The conditions of satisfaction related to issues will be implemented the next iteration. Additionally, we have not any conditions of satisfaction requiring persistent storage is going to be implemented in a future iteration.

3) We used Travis CI as our continuous integration system in a .travis,yml file which ran a suite of tests on our package named “git_comments”. Our travis system targeted Mac OSX as its operating system because that was the platform our developers programmed on. For the front end travis downloads the Sublime Text Plugin UnitTesting platform to allow travis to run a headless version of Sublime Text that can run the tests written for it. For the backend we used nosetests for unit tests, and pylint for static analysis, coveralls for coverage, and pip env for virtualization of the python environment.

4)

Link: https://github.com/Raphaeljunior/resolve-comments/

Tagged Commit:

Title: All Hacks Removed

Sha: 17e2212e1c2630d29b775f6e24c13d3152110fe6

All commits before and including this tagged commit represent the code in the first iteration