Team: McBain Fire Alarm

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

Raphael Owino (ro2295) Kevin Mejia (km3204)

1. **Date and time of presentation:** 12/4 6:00 pm

We were able to show all of the sublime plugin features that implemented on the backend in the first iteration on the actual Sublime Plugin interface. This means prompting the user for username and password, displaying all the issues in a searchable panel, and popping up a detailed side panel of issues on the right side of the window. Additionally, we presented the plugin's ability to detect if the file being edited is in a git repository. The main issues that arose deal with coverage of our application being 34% for the entire repo (65% for our code). The primary reason for that there are manually installed libraries in the repository (because Sublime text doesn't allow for pip environments), but we also need to focus more on writing tests that cover failure of our methods.

2. Bolded conditions were satisfied in this iteration. Green conditions were satisfied in the second iteration.

Use Cases Shown:

1.<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 <

1. The plugin can login to a GitHub repo,
2. T**he plugin can query GitHub Issues,**
3. **The plugin can query GitHub Issues assigned to me,**
4. **The plugin can query GitHub Issues which are open,**
5. The plugin can query the Github repo pull requests and their status,

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

>

USE CASES:

- Developer has an internet connection. They open up Sublime Text 3, instead of github.com on the web browser. If the user is previously logged in, they should expect to stay logged in. Some visual confirmation should be visually available to confirm the github username and 'logged on' status. If the user hovers their mouse over the codebase that has comments on a pull request tied to them, they see some visual indicator that they have something to check out. Upon clicking that indicator, they decide on next steps: these include
    - add a summary of the proposed changes,
    - review the changes made by commits,
    - add labels
    - add assignees
    - @mention individual contributors or teams

2.<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 <

1. **The plugin can query open GitHub Issues** and **pull review comments**,
2. The plugin can close an open GitHub Issue,

3. **The plugin can submit responses to comments on open pull requests**,
4. **The plugin can read GitHub Issues in the Sublime Text Editor,**
5. The plugin can close a GitHub Issue in the Sublime Text Editor

>

USE CASES

- Developer has an internet connection. They open up Sublime Text 3, instead of github.com on the web browser. If the user is previously logged in, they should expect to stay logged in. Some visual confirmation should be visually available to confirm the github username and 'logged on' status. When the user accesses some git-backed codebase, they have the option to
  - Click sections of a commented region to view pull review comments
  - Read a recent github issue
  - Close, or dismiss the issue without changing the code or closing the Sublime Text 3 window

3. <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 <

1. **The plugin can send data to a GitHub repo,**

2. **The plugin can submit comments to open pull requests,**
3. **The plugin can add new GitHub Issues to a GitHub repo**

>

USE CASES:

- Developer has an internet connection. They open up Sublime Text 3, instead of github.com on the web browser. If the user is previously logged in, they should expect to stay logged in. Some visual confirmation should be visually available to confirm the github username and 'logged on' status. The expectation is the codebase being accessed is shared among two or more github accounts. When the user accesses some git-backed codebase, they have the option to
    - Highlight a code region
    - Right click to open a tab of options
    - From those options, click "submit comments"
        - From those comments, there is a submit pull request button as a final confirmation

4.<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 <

1. The plugin can query a database (SQL or NoSQL tbd),
2. The plugin can insert data into a database,

3. The plugin can prompt the user for credentials,
4. **The plugin can verify user credentials**
5. The database stores the data persistently (in-memory, commitlog, etc.)
6. **The plugin can make an oath authentication request to GitHub**

>

USE CASES:

- User opens Sublime window:
  - User has used the plugin before, but logged out during the last usage before closing Sublime Text 3. A familiar tab should open, and in that the user provides either (user +password) or (an access token).
  - User is a recurring user of the plugin. A tab opens saying that the current Github username is displayed.
    - If the user is the user shown, all that us needed is the password
    - If the user is different than the one shown. They click a 'log out' button and then provides either (user +password) or (an access token) to access our plugin as the correct user
  - User has never used the plugin before, and this has not logged on to github via Sublime Text 3. A tab opens, and in that the user provides either (user +password) or (an access token).

5. <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

>

USE CASE

- User logs in according to the use cases described above.
    - If no preferences were previously provided, data is laid out using a default setting{font, font size, window location and size, font color etc}
    - Custom exist , so it's a straightforward way to present the github comments in the manner they want.

6. <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.

USE CASE:

- User logs in according to the use cases described above.
  - If no preferences were previously provided, data is laid out using a default setting{font, font size, window location and size, font color etc}
  - Custom settings exist , they are retrievable from an existing database., so it’s a straightforward way to present the github comments in the manner they want.

3. Details on our CI:

https://travis-ci.com/Raphaeljunior/resolve-comments/builds/92942006

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 Unix as its operating system because that was the platform our developers programmed on. For the front end travis downloads the Sublime Text Plugin Unit Testing 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.

Coverage of our code is 65%. Some exceptions need to be defined and their assertion tested to improve our coverage.

We also use github project boards to manage user stories and split work across the different iterations.

4. Link to Repo and Tagged Revisions

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

Tagged Pull Requests:

https://github.com/Raphaeljunior/resolve-comments/pulls?q=is%3Apr+is%3Aclosed+label%3A%22Second+Iteration+Tag%22