

# SW Engineering CSC648/848

## WhereWeEatinMeow

### Restaurant Choice Solution

#### Section 04, Team 03

**Raul Cardenas** – *Team Lead / Infrastructure Lead*

**Andy Nguyen** – *Github Master / Back-end*

**David Ye Luo** – *Database Admin / Front-end*

**Dean De Leon** – *Scrum Master / Front-end*

**Fred Gunther** – *Back-End Master / Front-end*

**Jonathan Ko** – *Back-End Master / Front-end*

#### “Milestone 4”

May 3<sup>rd</sup>, 2023

## Content and Structure for Milestone 4 Document:

<b>1) QA Testing:</b>	<b>3</b>
I. Unit Test	3
II. Integration Test	5
<b>2) Coding Practices:</b>	<b>9</b>
I. A Coding Style	9

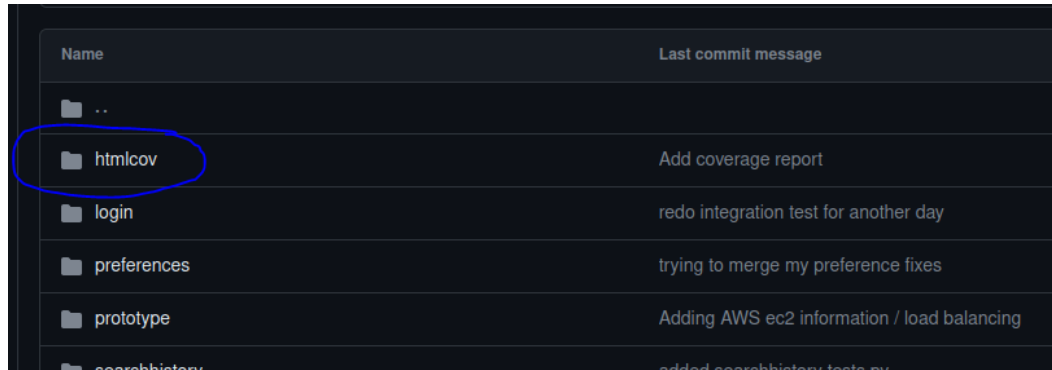
# 1) QA Testing:

## I. Unit Test

Select 5 P1 features to be tested.

- Create unit test cases in the language same as your application (such as Python, Javascript, Java, etc.) for the units (modules/ classes/and methods) associated with the 5 P1 features.
- Run unit test cases in your chosen unit testing framework (e.g., Jest, Junit, Python unittest).
- Analyze functional and statement test coverage.
  - Recommendation: functional coverage is  $\geq 90\%$ , statement coverage  $\geq 60\%$
- [Extra Credit of 10 Pts] Integrate your unit tests with CI framework (e.g., Github Actions) to automatically trigger your unit.
- What to submit
  - Description of 5 P1 features and the list of corresponding source files in the Github
  - Github link of test cases directory
  - Description of functional and statement coverage.
  - Description of your integrated CI workflow (optional)
- Grading rubrics
  - Do the test cases cover 5 P1 features?
  - Coverage of unit testing cases
    - functional coverage  $\geq 90\%$ , statement coverage  $\geq 60\%$
  - Integration with CI is implemented? (extra credit of 10 Pts)
- Our 5 P1 features to be tested are...
  - User able to rate restaurant experience (Search Summary)
  - User able to login (login app)
  - User able to sign up (signup app)
  - User able to select allowed preferences (preference app)
  - User is given a randomly selected restaurant from YELP API based on select preferences (searchsummary app)
- Github links of test cases directory:
  - Search Summary
    - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/5ae61112329a8406360603997a2ecca2ece42543/application/prototype/searchsummary/tests.py>
  - Login and Sign Up
    - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/master/application/prototype/login/tests.py>

- Preferences
  - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/master/application/prototype/preferences/tests.py>
- Functional and statement coverage:
  - 75% overall statement coverage - coverage.py
  - screenshot:



Name	Last commit message
..	
htmlcov	Add coverage report
login	redo integration test for another day
preferences	trying to merge my preference fixes
prototype	Adding AWS ec2 information / load balancing
searchhistory	added searchhistory tests.py

htmlcov/index.html will show the below screenshot. This is done on every pull request.

## Coverage report: 75%

coverage.py v7.2.5, created at 2023-05-03 20:26-0700

Module	statements	missing	excluded	coverage
login/__init__.py	0	0	0	100%
login/apps.py	4	0	0	100%
login/migrations/__init__.py	0	0	0	100%
login/models.py	1	0	0	100%
login/tests.py	30	2	0	93%
login/urls.py	3	0	0	100%
login/views.py	18	1	0	94%
manage.py	12	2	0	83%
preferences/__init__.py	0	0	0	100%
preferences/apps.py	4	0	0	100%
preferences/forms.py	14	0	0	100%
preferences/migrations/0001_initial.py	7	0	0	100%
preferences/migrations/0002_history_image_url_history_user_rating.py	4	0	0	100%
preferences/migrations/__init__.py	0	0	0	100%
preferences/models.py	21	0	0	100%
preferences/tests.py	29	0	0	100%
preferences/urls.py	3	0	0	100%
preferences/views.py	64	43	0	33%
prototype/__init__.py	0	0	0	100%
prototype/settings.py	19	0	0	100%
prototype/urls.py	3	0	0	100%
searchhistory/__init__.py	0	0	0	100%
searchhistory/apps.py	4	0	0	100%
searchhistory/migrations/__init__.py	0	0	0	100%
searchhistory/models.py	1	0	0	100%
searchhistory/tests.py	26	0	0	100%
searchhistory/urls.py	3	0	0	100%
searchhistory/views.py	10	0	0	100%
searchsummary/__init__.py	0	0	0	100%
searchsummary/apps.py	4	0	0	100%
searchsummary/migrations/__init__.py	0	0	0	100%
searchsummary/models.py	1	0	0	100%
searchsummary/tests.py	67	0	0	100%
searchsummary/urls.py	3	0	0	100%
searchsummary/views.py	56	31	0	45%
signup/__init__.py	0	0	0	100%
signup/apps.py	4	0	0	100%
signup/migrations/__init__.py	0	0	0	100%
signup/models.py	1	0	0	100%
signup/tests.py	1	0	0	100%
signup/urls.py	3	0	0	100%
signup/views.py	36	21	0	42%
userprofile/__init__.py	0	0	0	100%
userprofile/apps.py	4	0	0	100%
userprofile/migrations/__init__.py	0	0	0	100%
userprofile/models.py	1	0	0	100%
userprofile/tests.py	1	0	0	100%
userprofile/urls.py	3	0	0	100%
userprofile/views.py	37	25	0	32%
<b>Total</b>	<b>502</b>	<b>125</b>	<b>0</b>	<b>75%</b>

coverage.py v7.2.5, created at 2023-05-03 20:26-0700

-

## II. Integration Test

For at least 10 P1 feature, perform the below:

- Define the integration test cases according to the format (refer to slides):
    - The format should include test case id; test case description; dates tested, test scenario with steps to follow, prerequisites, test data and expected result.
  - Perform testing and record the testing results.
    - Describe Test results (PASS or FAIL) for each test case.
    - For failed cases (e.g., bugs), record them in the bug tracker system (e.g., Jira, or Github Issues).
  - Describe how many P1 features are tested.
  - What to submit
    - Description of test cases and test results for each of 10+ P1 features.
    - URL of the bug tracker system to record your bugs.
  - Grading rubrics
    - How many P1 features were tested?
    - How many P1 features were correctly passed the integration tests?
- Our 10 P1 features are:
- Views
    - Mobile
    - Desktop
  - Login / Sign In
    - Add users
    - Users able to login
  - Preferences
    - users able to add preferences to get random restaurant based on Yelp API
  - UI
    - Consistent UI by having cat images in our app to create a fun and positive atmosphere
  - Pages
    - Login app (login page)
    - Sign in app (sign up page)
    - Preferences app (preferences app)
    - Search Summary app (summary app)

- Testing results (Passed):

Id	Test	Description
1	Signup and Login work together. Logging in by itself works.	Signing up brings user to preferences page. Logging in with an already-created user brings the user to the preferences page.
2	Multiple preferences searches appear in history	Going back to preferences screen and making a search adds recent search to searchhistory page.
3	Changing user profile's username and password changes user in Django	Navigating to profile screen, changing user information, then checking Django admin shows that the profile for a user changes with the text field and button press.

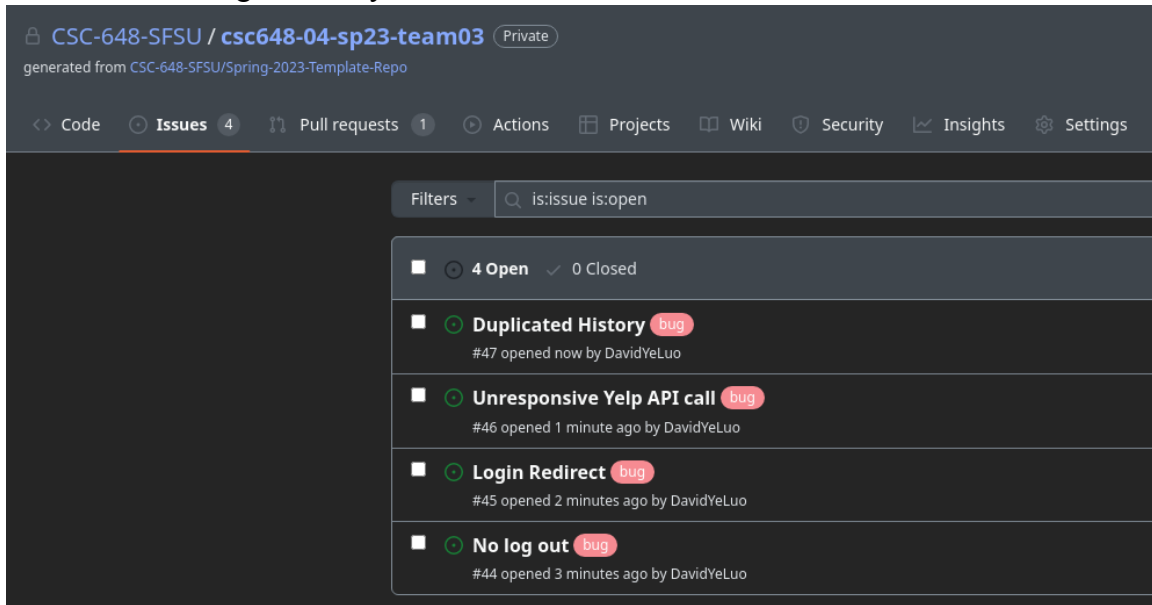
- Testing results (Failed):

Id	Test/Bug/Description	Status	To replicate
4	No log out	Have yet to implement	No button anywhere to log out. So you can not attempt to log out.
5	Login Redirect	In progress	When trying to login, django takes you to the wrong preferences page. This usually happens after using Django's logout(), or changing the user password, then logging back in.
6	Unresponsive Yelp API call	In Progress	When a search result gives an empty result, there isn't a response/notification to the user.

7	Duplicated History	In Progress	When a user generates a search from the preference page and transitions to the search summary, it works fine. However, when the user presses back, the user creates another search that is the same as the previous one.
---	--------------------	-------------	--



- URL of bug tracker system: Github issues



## **2) Coding Practices:**

### **I. A Coding Style**

- Please describe what coding style you chose (e.g., Google Python Coding Style, refer to slides), and how to enforce it by the tool or plug-in (refer to slides).
  - Please list source files related to 5 P1 features which demonstrate your coding style.
- 
- Our coding style: PEP coding standards
    - Use meaningful variable, function, and class names
    - Naming styles: lowercase with underscores (variables/functions), CamelCase (classes), UPPERCASE (constants)
    - Indent using 4 spaces
    - Limit line length to 79 characters
    - Organize imports: standard library, third-party, Django, local application
    - Use pydoc for documentation
    - Use flake8 for linting
    - Add comments for complex logic
  - Source files related to 5 P1 features:
    - Login (views.py and tests.py)
      - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/master/application/prototype/login/views.py>
      - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/master/application/prototype/searchsummary/tests.py>
    - Signup
      - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/master/application/prototype/signup/views.py>
    - Preferences
      - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/master/application/prototype/preferences/views.py>
    - Search Summary (Random Yelp Restaurant and User Rating Features)
      - <https://github.com/CSC-648-SFSU/csc648-04-sp23-team03/blob/master/application/prototype/searchsummary/views.py>