CSC 648/848 Milestone 4: Testing

Section 01 Team 02 - Ensight

Pragati Makani: Team Lead, Frontend
Sarah Abusaif: Frontend Lead
Alexander Del Rio: Backend Lead
Mohammad Dahbour: Scrum Master, Backend
Christian Montalvo: Github Master

11/29/23 —

1. QA Testing

a. Unit Testing

i. Description of 5 P1 features and Github URL:

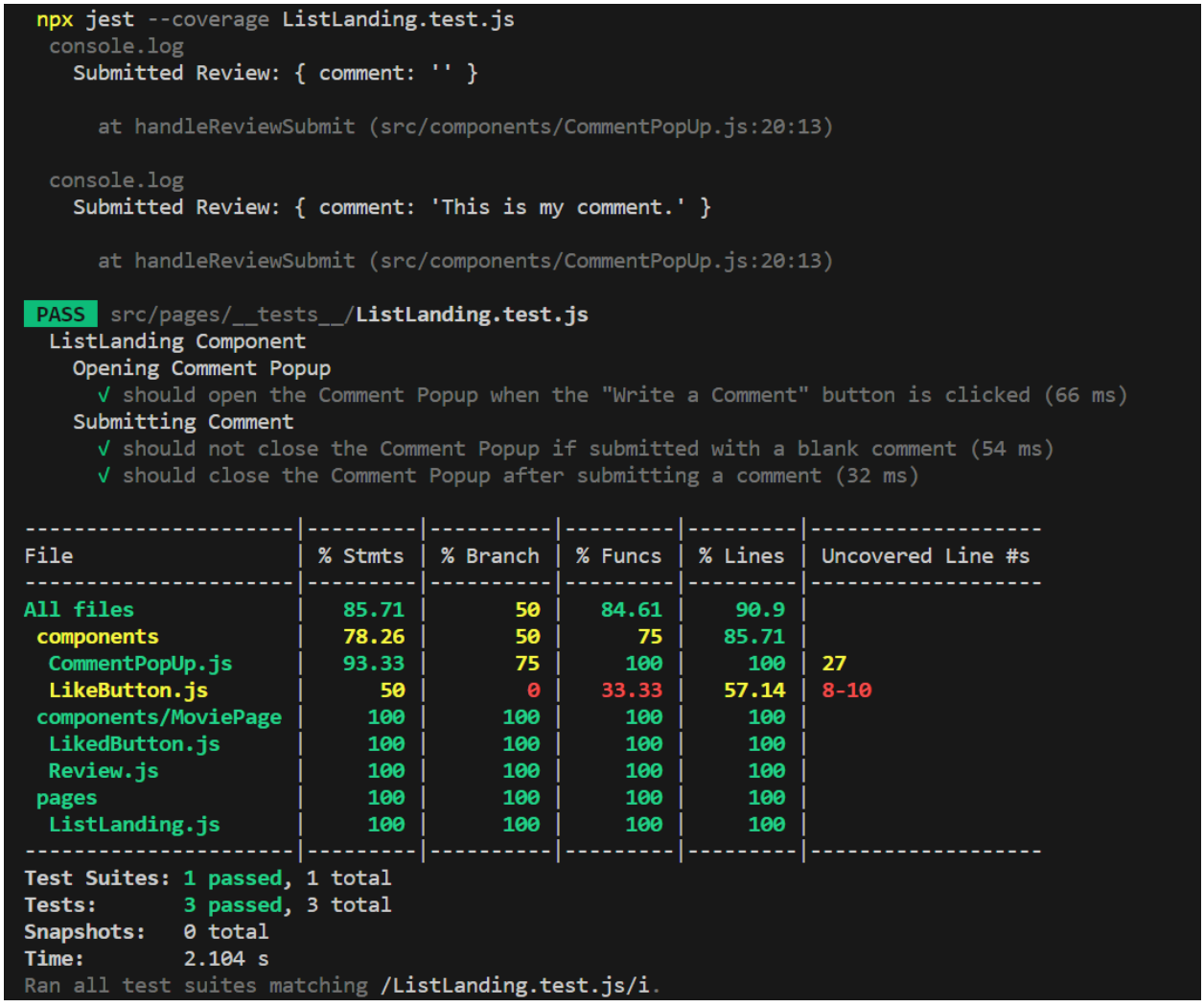
1. **Comment Popup:** Enables users to leave comments on other user's lists. When the user first lands on the ListLanding page, there is a button with the text: "Write a Comment." Clicking on the button will make the comment pop-up appear and the user can type their comment and submit it. If the user attempts to submit without typing anything, then the pop-up won't close.
   Github:
   https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/pages/__tests__/ListLanding.test.js
2. **Like Button:** Enables users to 'like' a specific review, movie, list, diary, etc. This is typically represented by

a heart icon which serves as a button that users can click or tap to like. When liked by a user, the heart icon becomes filled in, to indicate that the like has been recorded.
Github: https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/components/Tabs/__tests__/Diary.test.js

3. **Follow Button:** Enables users to follow other users. This feature is interactive and involves a change in the button's appearance to reflect the current state of the relationship—whether the user is following or not.
Github: https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/pages/__tests__/DisplayUser.test.js
4. **Rating Popup:** Enables users to rate a movie of their choice. When pressing on a movie poster from Browse or DisplayMovie, they will be taken to the MovieLanding page. On this page, users can click on the Rate text and rate the movie on a scale from 1 to 5 by clicking on the stars. This is interactive as clicking on a star will fill in that start along with any other previous stars.
Github: https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/pages/__tests__/MovieLanding.test.js
5. **Header Search Bar:** Enables users to search for a term using the search bar in the Header. After typing a prompt users can click on the search button and it will take them from whatever page they were on to

the Browse page. If the user attempts to search without typing anything, then they won't be redirected and will remain on the same page.
Github: https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/components/__tests__/Header.test.js

ii. Description of functional and statement coverage.



```
npx jest --coverage ListLanding.test.js
 console.log
   Submitted Review: { comment: '' }

     at handleReviewSubmit (src/components/CommentPopUp.js:20:13)

 console.log
   Submitted Review: { comment: 'This is my comment.' }

     at handleReviewSubmit (src/components/CommentPopUp.js:20:13)

 PASS  src/pages/__tests__/ListLanding.test.js
  ListLanding Component
    Opening Comment Popup
      √ should open the Comment Popup when the "Write a Comment" button is clicked (66 ms)
    Submitting Comment
      √ should not close the Comment Popup if submitted with a blank comment (54 ms)
      √ should close the Comment Popup after submitting a comment (32 ms)

----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |   85.71 |       50 |   84.61 |    90.9 |
 components           |   78.26 |       50 |      75 |   85.71 |
  CommentPopUp.js     |   93.33 |       75 |     100 |     100 | 27
  LikeButton.js       |      50 |        0 |   33.33 |   57.14 | 8-10
 components/MoviePage |     100 |      100 |     100 |     100 |
  LikedButton.js      |     100 |      100 |     100 |     100 |
  Review.js           |     100 |      100 |     100 |     100 |
 pages                |     100 |      100 |     100 |     100 |
  ListLanding.js      |     100 |      100 |     100 |     100 |
----------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        2.104 s
Ran all test suites matching /ListLanding.test.js/i.
```

```
npx jest --coverage Diary.test.js
 PASS  src/components/Tabs/__tests__/Diary.test.js
  Diary Component
    Liking Diary Button
      ✓ should increases the like count by one and toggles checked state when like button is clicked (39 ms)
    Unliking Diary Button
      ✓ should increase the like count by 1 then subtract by 1 when checking and uncheching the like button (10 ms)

----------------|---------|----------|---------|---------|-------------------
File            | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------|---------|----------|---------|---------|-------------------
All files       |   87.17 |       50 |    90.9 |   85.71 |
 components     |   84.21 |       40 |     100 |   83.33 |
  LikeButton.js |     100 |       50 |     100 |     100 | 9
  SlideIn.js    |   72.72 |     37.5 |     100 |   72.72 | 10-12
 components/Tabs|      90 |       60 |   83.33 |   88.23 |
  Diary.js      |      90 |       60 |   83.33 |   88.23 | 22,89
----------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.63 s, estimated 4 s
Ran all test suites matching /Diary.test.js/i.
```

```
npx jest --coverage DisplayUser.test.js
 PASS  src/pages/__tests__/DisplayUser.test.js
  DisplayUser Component
    Rendering DisplayUserResults Component
      ✓ should have a follow button for user 1 (39 ms)
      ✓ should have a follow button for user 2 (10 ms)
      ✓ should have a follow button for user 3 (9 ms)
      ✓ should have a follow button for user 4 (7 ms)
      ✓ should have a follow button for user 5 (11 ms)
      ✓ should not have a follow button for user 6 or above (12 ms)
    Handle Toggle Follow
      ✓ should change Follow + to Following and vice versa (19 ms)

------------------------|---------|----------|---------|---------|-------------------
File                    | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------------------|---------|----------|---------|---------|-------------------
All files               |   92.85 |       90 |      80 |    92.3 |
 components             |     100 |      100 |     100 |     100 |
  FollowButton.js       |     100 |      100 |     100 |     100 |
 components/Results     |     100 |      100 |     100 |     100 |
  DisplayUserResults.js |     100 |      100 |     100 |     100 |
 pages                  |    91.3 |    83.33 |      75 |   90.47 |
  DisplayUser.js        |    91.3 |    83.33 |      75 |   90.47 | 33,77
------------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        1.897 s, estimated 2 s
Ran all test suites matching /DisplayUser.test.js/i.
```

```
 npx jest --coverage MovieLanding.test.js
  console.log
    Number of filled stars: 3

      at handleRateSubmit (src/components/RatingPopUp.js:32:15)

 PASS  src/pages/__tests__/MovieLanding.test.js
  MovieLanding Component
    Opening Rating Popup
      √ should open the Rating Popup when the "Rate" text is clicked (71 ms)
    Give a Movie a Rating
      √ should close the pop up and print the rating (70 ms)

----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |   75.32 |    46.87 |   66.66 |   78.37 |
 components           |   70.45 |       50 |      60 |    75.6 |
  LikeButton.js       |      50 |        0 |   33.33 |   57.14 | 8-10
  RatingPopUp.js      |   86.36 |       70 |     100 |   90.47 | 24,35
  ReviewPopUp.js      |   57.14 |       25 |      20 |   61.53 | 11,15,22-23,57
 components/MoviePage |     100 |      100 |     100 |     100 |
  LikedButton.js      |     100 |      100 |     100 |     100 |
  Review.js           |     100 |      100 |     100 |     100 |
  TopCast.js          |     100 |      100 |     100 |     100 |
 pages                |   77.77 |    43.75 |   66.66 |   77.77 |
  MovieLanding.js     |   77.77 |    43.75 |   66.66 |   77.77 | 29-31,46-48,53
----------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.958 s, estimated 2 s
Ran all test suites matching /MovieLanding.test.js/i.
```

```
npx jest --coverage Header.test.js
 console.log
   example

     at handleSearchClick (src/components/Header.js:35:15)

 console.log
   Input field is empty. Please enter a search term.

     at handleSearchClick (src/components/Header.js:38:15)

 PASS  src/components/__tests__/Header.test.js
  Header Component
    Search To Browse
      √ should search with a term that navigates to Browse page (88 ms)
    Search Stay
      √ should search with an empty term that does nothing (15 ms)

-------------|---------|----------|---------|---------|-------------------
File         | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------|---------|----------|---------|---------|-------------------
All files    |      75 |       50 |   33.33 |      75 |
 Header.js   |   77.77 |       75 |      40 |   77.77 | 14-15,23,27
 SideMenu.js |      50 |        0 |       0 |      50 | 6
-------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        2.152 s
```

b. Integration Testing

Test case: 1

Test Case ID	AD_001	**Test Case Descripton**	Test the Login functionality on Ensight		
Created By	Alex	**Reviewed By**	Mohammad	**Version**	1.5
QA Tester's Log	Review comments from Mohammad done in version 1.2				
Tester's Name	Alex	**Date Tested**	11-29-2023	**Test Case (Pass/Fail/Not)**	Pass

S#	Prerequisites:
1	Access to a Chrome Browser
2	
3	
4	

S#	Test data
1	username = Moe
2	Pass = something123
3	
4	

Test Scenario	Verify on entering valid username and password, the user can login

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on Login (top right of screen)	Login popup should open	As expected	Pass
3	Enter username and password	Credentials can be entered	As expected	Pass
4	Click Let's go!	User is logged in	As expected	Pass
5				

Test case: 2

Test Case ID	AD_002	**Test Case Descripton**	Test the Logout functionality on Ensight		
Created By	Alex	**Reviewed By**	Mohammad	**Version**	1.5
QA Tester's Log	Review comments from Mohammad done in version 1.2				
Tester's Name	Alex	**Date Tested**	11-21-2023	**Test Case (Pass/Fail/Not)**	Pass

S#	Prerequisites:
1	Access to a Chrome Browser
2	
3	
4	

S#	Test data
1	username = Moe
2	Pass = something123
3	
4	

Test Scenario	Verify on entering valid username and password, the user can login, then they can logout

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on Login (top right of screen)	Login popup should open	As expected	Pass
3	Enter username and password	Credentials can be entered	As expected	Pass
4	Click Let's go!	User is logged in	As expected	Pass
6	Click on Logout (top right of screen)	User should get logged out	As expected	Pass
5				

Test case: 3

Test Case ID	AD_003	**Test Case Descripton**	Test the Search functionality on Ensight		
Created By	Alex	**Reviewed By**	Mohammad	**Version**	1.5
QA Tester's Log	Review comments from Mohammad done in version 1.2				
Tester's Name	Alex	**Date Tested**	11-14-2023	**Test Case (Pass/Fail/Not)**	Pass

S#	Prerequisites:
1	Access to a Chrome Browser
2	
3	
4	

S#	Test data
1	search query = spider-man
2	
3	
4	

Test Scenario	Verify on entering an input as a search query into the search bar, appropriate results should be displayed

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on search bar at the top of the page	search bar should be accepting input	As expected	Pass
3	Enter search query "spider-man"	search query should be appearing as input	As expected	Pass
4	Click Search	Results of movies/lists/ and users containing the name "spider-man" should appear	As expected	Pass
6				
5				

Test case: 4

Test Case ID	AD_004	**Test Case Descripton**	Test the Filter by year functionality on Ensight		
Created By	Alex	**Reviewed By**	Mohammad	**Version**	1.5
QA Tester's Log	Review comments from Mohammad done in version 1.2				
Tester's Name	Alex	**Date Tested**	11-14-2023	**Test Case (Pass/Fail/Not)**	Pass

S#	Prerequisites:
1	Access to a Chrome Browser
2	
3	
4	

S#	Test data
1	search query = spider-man
2	
3	
4	

Test Scenario	Verify on entering an input as a search query into the search bar, appropriate results should be displayed, then check if you can filter those results by their date of release

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on search bar at the top of the page	search bar should be accepting input	As expected	Pass
3	Enter search query "spider-man"	search query should be appearing as input	As expected	Pass
4	Click Search	Results of movies/lists/ and users containing the name "spider-man should appear"	As expected	Pass
6	On left side of page, tick the checkbox for year "2010s"	Same Results, but should be limited to movies released in the 2010s	As expected	Pass
5				

Test case: 5

Test Case ID	AD_005	Test Case Descripton	Test the Filter by genre functionality on Ensight		
Created By	Alex	Reviewed By	Mohammad	Version	1.5
QA Tester's Log	Review comments from Mohammad done in version 1.2				
Tester's Name	Alex	Date Tested	11-8-2023	Test Case (Pass/Fail/Not)	Pass

S#	Prerequisites:
1	Access to a Chrome Browser
2	
3	
4	

S#	Test data
1	search query = spider-man
2	
3	
4	

Test Scenario	Verify on entering an input as a search query into the search bar, appropriate results should be displayed, then check if you can filter those results by their genre

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on search bar at the top of the page	search bar should be accepting input	As expected	Pass
3	Enter search query "spider-man"	search query should be appearing as input	As expected	Pass
4	Click Search	Results of movies/lists/ and users containing the name "spider-man should appear"	As expected	Pass
6	On left side of page, tick the checkbox for genre "Action	Same Results, but should be limited to movies of the "Action" genre	As expected	Pass
5				

Test case: 6

Test Case ID	AD_006	Test Case Descripton	Test the Filter by rating functionality on Ensight		
Created By	Alex	Reviewed By	Mohammad	Version	1.5
QA Tester's Log	Review comments from Mohammad done in version 1.2				
Tester's Name	Alex	Date Tested	11-8-2023	Test Case (Pass/Fail/Not)	Pass

S#	Prerequisites:
1	Access to a Chrome Browser
2	
3	
4	

S#	Test data
1	search query = spider-man
2	
3	
4	

Test Scenario	Verify on entering an input as a search query into the search bar, appropriate results should be displayed, then check if you can filter those results by their ratings

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on search bar at the top of the page	search bar should be accepting input	As expected	Pass
3	Enter search query "spider-man"	search query should be appearing as input	As expected	Pass
4	Click Search	Results of movies/lists/ and users containing the name "spider-man should appear"	As expected	Pass
6	On left side of page, tick the button for Rating "Lowest"	Same Results, but should be sorted from lowest rated movies to highest	As expected	Pass
5				

Test case: 7

Test Case ID	AD_007	Test Case Descripton	Test the create a movie list functionality on Ensight		
Created By	Alex	Reviewed By	Mohammad	Version	1.5
QA Tester's Log	Review comments from Mohammad done in version 1.2				
Tester's Name	Alex	Date Tested	11-29-2023	Test Case (Pass/Fail/Not)	Pass

S#	Prerequisites:
1	Access to a Chrome Browser
2	
3	
4	

S#	Test data
1	username = Moe
2	Pass = something123
3	search query = "spider-man"
4	

Test Scenario	Verify on entering valid username and password, the user can login, then add movies to thier movie list and create list with a title and description

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on Login (top right of screen)	Login popup should open	As expected	Pass
3	Enter username and password	Credentials can be entered	As expected	Pass
4	Click Let's go!	User is logged in	As expected	Pass
5	Click on menu side bar	side bar open with tabs to chose from	As expected	Pass
6	Click on the Profile option	should be redirected to your user profile	As expected	Pass
7	Click on the Lists tab	should view your created lists with an option to create a new one	As expected	Pass
8	Click on the "+" icon	Should display a search bar that prompts you to search for movies, along with a space to add a title and another space to add a description for your list	As expected	Pass
9	Type your search query then hit search	Results of movies containing the name "spider-man should appear"	As expected	Pass
10	Click on 5 different movies	All selected movies should be displayed in the staging area to the right	As expected	Pass
11	Add a title and description	Input should be accepted for the title and description of your movie list	As expected	Pass
12	Click on Create	The movie list should be created and a preview of it should be displayed on your profile page under the lists tab	As expected	Pass

Test case: 8

Test Case ID	AD_008	Test Case Descripton	Test registering user accounts		
Created By	Alex	Reviewed By	Mohammad	Version	0.1
QA Tester's Log					
Tester's Name	Alex	Date Tested	11-29-2023	Test Case (Pass/Fail/Not)	Pass

S#	Prerequisites:
1	Access to Chrome Browser
2	
3	
4	

S#	Test data
1	username = Moe
2	Pass = something123
3	
4	

Test Scenario	Verify on entering valid username and password, the user can login

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on Login (top right of screen)	Login page should open	As expected	Pass
3	Click on Sign up	Registration form should display	As expected	Pass
4	Enter desired credentials and click confirm	Account is created, user is logged in and given an auth token, and user is rerouted to home page	As expected	Pass

Test case: 9

Test Case ID	AD_009	Test Case Descripton	Test Movie Detail pages load correctly		
Created By	Alex	Reviewed By	Mohammad	Version	0.1

QA Tester's Log					
Tester's Name	Alex	Date Tested	11-29-2023	Test Case (Pass/Fail/Not)	Pass

S#	Prerequisites:
1	Access to Chrome Browser
2	
3	
4	

S#	Test data
1	movie_id=872585
2	
3	
4	

Test Scenario	Verify Movie landing page for specific movie is loaded with the details from the database

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space/MovieLanding/872585	Page for 'Oppenheimer' should open	As expected	Pass
2	Verify details of the movie have been loaded	Title, release-date, genres, description, rating data, credits data is loaded	Data is loaded from database as expected, but not all data in the database is how it should be in the end	Pass (mostly)
3	Click watch trailer button	Embedded YouTube trailer should open	As expected	Pass
4	Click hide trailer button	Embedded YouTube trailer should close	As expected	Pass
5				

Test case: 10

Test Case ID	AD_010	Test Case Descripton	Add movie to the user's favorites		
Created By	Alex	Reviewed By	Mohammad	Version	0.1

QA Tester's Log					
Tester's Name	Alex	Date Tested	11-29-2023	Test Case (Pass/Fail/Not)	Pass

S#	Prerequisites:
1	Access to Chrome Browser
2	
3	
4	

S#	Test data
1	username = Moe
2	Pass = something123
3	movie_id=872 585
4	

Test Scenario	Log in user and navigate to a movie landing page then click the heart to add to the user's favorites

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://ensight.space	Site should open	As expected	Pass
2	Click on Login (top right of screen)	Login popup should open	As expected	Pass
3	Enter username and password	Credentials can be entered	As expected	Pass
4	Click Let's go!	User is logged in	As expected	Pass
5	Navigate to http://ensight.space/MovieLanding/872585	Page for 'Oppenheimer' should open with a heart next to title	As expected	Pass
6	Click on heart	Heart should fill in and 'Oppenheimer' should be added to the logged in user's favorites	As expected	Pass
7	Reload Page	heart should be filled in since the movie is already stored as a favorite for this user in the database	As expected	Pass

2. Coding Practices

For Django REST Framework (Python):

Chosen Coding Style:
For our Django REST Framework backend, we have adopted the PEP 8 -- Style Guide for Python Code. PEP 8 provides guidelines and best practices on how to write Python code. It covers various aspects of coding style, such as indentation, tab width, maximum line length, imports organization, whitespace usage, naming conventions, and more.

Enforcement Tools:
black: An uncompromising code formatter that takes our code and rewrites it in a way that adheres to the style guide, ensuring consistency.

Usage:
Developers are also encouraged to run Black locally before pushing their code.

Example Source Files:
Git Branch: "feature-filters"

1. views.py
(https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/feature-filters/application/ensight/app/views.py)
2. models.py
(https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/feature-filters/application/ensight/app/models.py)
3. get_movie_data.py
(https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/feature-filters/get_movie_data.py)

4. serializers.py
(https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/feature-filters/application/ensight/app/serializers.py)
5. urls.py
(https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/feature-filters/application/ensight/app/urls.py)

Enforcing the Style Guide:

1. **Installation:** Install Black in your Django project
2. **Configuration:** Black doesn't require complex configurations. To customize line length (default is 88), create a pyproject.toml file
3. **Integration in the Development Workflow:** Configured VS Code to use Black for formatting using plugins which allowed us to format our code with Black automatically.
4. **Running Black:** Formatted the entire codebase by running "*black .*" in our project root.

For ReactJS (JavaScript):

Chosen Coding Style:
For our ReactJS frontend development, we have adopted the Google JavaScript Style Guide. This style guide is detailed and covers all aspects of JavaScript coding conventions, including but not limited to variable declarations, function and class naming conventions, file structure, and use of language features such as async/await, promises, and events. It also includes specific style rules for using JSX in React.

Enforcement Tools:
Prettier: We use Prettier for code formatting. While Prettier does not cover all style rules, it ensures that the code format is consistent across the entire codebase.

Usage:
Our team members were expected to configure their code editors to use Prettier, which will help them adhere to the style guide during development.

Example Source Files:
Git Branch: "Frontend"

1. **MovieLanding.js:** https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/pages/MovieLanding.js
2. **DisplayUser.js:** https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/pages/DisplayUser.js
3. **DiaryFocus.js:** https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/components/Tabs/ProfileTabsContent/DiaryFocus.js
4. **ListLanding.js:** https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/pages/ListLanding.js
5. **Browse.js:** https://github.com/CSC-648-SFSU/csc648-01-fa23-team02/blob/Frontend/application/ensight/src/pages/Browse.js

Enforcing the Style Guide:

1. **Installation:** Install Prettier in your project as a dev dependency:
2. **Configuration:** Create a .prettierrc file in your project root for custom configurations.
3. **Integration in the Development Workflow:** Configured code editor (VS Code) to format on save with Prettier by using the extension.
4. **Running Prettier:** Manually format codebase by running "*npx prettier --write .*" in our project root.