System and Unit Test Report
FlixForum
Team : Ryan Wang, Saki Yokokawa, Gavin Poley, Julia Tober, Khin Yone
March 14th 2023

Sprint 1:

- **(8) User Story 1**: As a TV show watcher, I want to be able to communicate with other people and be a part of a community.
- **(21) User Story 2**: As a TV show watcher, I want to see some TV shows with forums so that I can get a preview of different shows available and navigate quickly to the chosen forums without searching. (about home page)

Sprint 2:

- **(13) User story 1:** As a TV show watcher, I want to see some TV shows with forums after I log in so that I can get a preview of different shows available and navigate quickly to the chosen forums without searching.
- **(21) User story 2:** As a TV show watcher, I want to be able to have a platform where I can see how people feel about new TV shows so that I can find new shows that might interest me and get real feedback about the show/episodes from other people.
- **(21) User story 3:** As a TV show watcher, I want to be able to discuss with others about a specific TV show episode so that we can communicate our thoughts about it as we are watching it.

Sprint 3

- **(21) User Story 1:** As a TV show watcher, I want to be able to discuss with others about a specific TV show episode so that we can communicate our thoughts about it as we are watching it.
- ~~**(13) User Story 2:** As a person interested in TV shows, I want to be able to look at the most popular opinion of the show to gauge the overall consensus of the show episode. I also want to be able to agree with other people's posts.~~ **(Omitted, since we didn't have time to do the ratings for TV shows)**

Sprint 4

- **(21) User Story 1:** As a TV show watcher, I want to be able to discuss with others about a specific TV show episode so that we can communicate our thoughts about it as we are watching it.
- **(34) User Story 2:** As a person interested in TV shows, I want to be able to agree with other people's posts.
- ~~**(18) User Story 3:** As a person interested in TV shows, I want to be able to look at the most popular opinion of the show to gauge the overall consensus of the show episode.~~

Highlighted user stories are the same that were pushed down because it was not completed in the original sprints. Striked through user stories were incomplete.

System Tests:

Sprint 1 User Story 1 :

Scenarios:

1. Start 'FlixForum' website; select 'Login', select 'Don't have an account? Register here'; type
    a. username = <cse115a>
    b. email = <cse115a@gmail.com>
    c. password = <cse115a>
    d. Press 'Sign up' button
    e. User should see login page
2. On Login page, type
    a. email=<cse115a@gmail.com>
    b. password=<cse115a>
    c. Press 'Log in' button
    d. Users should see the 'Log Out' button on the homepage if users are logged in successfully
3. Type 'sense 8' on search bar and Click 'sense 8' showcard
    a. Users should see "Sense 8" Forums page
4. Select:
    a. Season=<S1>
    b. Episodes=<Limbic Resonance>
5. Click 'New Post' button
    a. Users should be able to see the new post forum

Sprint 1 User Story 2 :

1. Navigate to FlixForum
2. Click the arrows to browse the three rows of TV shows previewed
3. User should be able to find popular shows on netflix, comedy, and romance TV shows

Sprint 2 User Story 2 :

1. Navigate to the FlixForum web
2. Click on the search bar in the top right corner and type in 'Sense8' and press enter
3. Click on the Sense8 showcard
4. Select season 1 from the Season drop down menu and select 'Limbic Resonance' from the episode drop down menu
5. The user should be able to view different posts from different user discussing their opinions about the episode

Sprint 2 User Story 3 :

1. navigate to FlixForum; go to Search Bar on the top right corner; type
    a. Sense8
2. select from Seasons drop down menu '1'
3. select from Episodes drop down menu 'Limbic Resonance'
4. click the *Go* button

5. view the posts made under the forum
6. click the *new post* button; type
    a. in the title textbox “why this first episode hooked me in”
    b. in the content textbox “it builds up the plot so good and the characterization is amazing”
7. click the *submit* button
8. click the *Go* button
9. User should see their post at the button of the page with the title and content they inputted

Sprint 4 User Story 2 :

1. Navigate to the FlixForum web
2. Click the login button on the top right corner of the home page type;
    a. email = <sabrina@gmail.com>
    b. password = <sabrina>
    c. Press ‘Log in’ button
3. User will be navigated back to the home page
4. Click on the search bar in the top right corner and type;
    a. Queen’s Gambit
    b. Press enter
5. Click on the Queen’s Gambit card
6. Select season 1 from the Season drop down menu
7. Select ‘Openings’’ from the episode drop down menu
8. The user should be able to view different posts from different user discussing their opinions about the episode, with the amount of likes and dislikes shown next to a corresponding button
9. Click the like/dislike button to update the total amount of likes/dislikes for that specific post (*Note: The like/dislike buttons are not working perfectly, which may cause bugs in other components of the website such as the navigation to a different forum)

Unit Test

Register/Login

- Tested by using console.log("Wrong username/password combination") when users type:
    1. Blank email address or/and password (empty input)
    2. Email address they signed up on the Register page, but used the different/wrong password
    3. Email address that does not exist on users DB (email address they have not created an account on the Register page)
- Tested by using console.log("correct") and making sure it navigates automatically to the home page after users click the 'login' button on the home page and sign in with their correct email address and password OR after users click the 'new post' button on the Forum page (before signing in) and sign in with their correct email address and password

NewPostModal

- Before working with the database, ie deploying data we built the page in increments
- First, in order to check that the data written in the textbox were correctly being grabbed, use console.log statements to see what values were being grabbed when the submit button was clicked.
- Then once we had the correct data being grabbed from the text boxes, we made sure we could get information about the user who was logged in since they were the ones who would be making the post. We used registered users in our database users table to compare the user_id in the console.log and the user_id corresponding to the email and password used to log in from the database.
- Then we also made sure we could get the forum's information: Title, Season, Episode in order to use this information to deploy to the database. We ensured that the information was correct by using our API to check the correct season_id, and the episode_id and comparing it to our data that we viewed by using console.log.
- Once we got the information about the user who is logged in and the forum's information, we started working on making the submit button and working in increments on deploying to the database also by using console.log statements and using workbench to check how the database was being updated.
- It wasn't feasible to have a row in the forums table in our database for every episode of every show, therefore we decided to only have rows in the database for forums that had a post under it.
- There were two cases:
    - 1. It is the first post under a forum
        - deploy a row into the forums table in db
        - grab the forum_id from forums table in db using the forum information we have
        - deploy a row into the posts table in db

    - 2. It isn't the first post under a forum
        - grab the forum_id from forums table in db using the forum information we grabbed
        - deploy a row into the posts table in db using forum_id
- In each step of these cases, as we were pulling from the db and pushing to the db, we used console.log statements to ensure the data we were getting were correct by comparing it to the database's stored information.
- We would check the data pulled from the db by statements such as :
    console.logging(result[0].forum_id)
- We would check the data pushed to the db were correct by using mySQL workbench and running the server to check the updates in the database as we were working.

Search Bar

- There were essentially two components regarding searching for a TV show - the search bar itself where the user entered the TV show, and the search result page where the user could then click on the different showcards to take them to the specific forum
- Firstly, we used console.logs throughout each section to ensure each portion was working properly
    - console.log('clicked enter') → keep track of whenever a user has entered a search query
    - console.log(result.state) → ensure that the search result page correctly received the input from the search bar page
    - console.log(JSON.stringify(result.state.message)) → needed to make sure that we properly converted the input into the proper format in order to use is as a normal string
    - console.log(error) → log any error that occurred
- In addition to the console logs, before the search result showcards and the API call were implemented, we would print out the query that the user entered straight inside the html in the return statement to ensure that we were able to retrieve it properly from the search bar page and that it was converted to a string correctly.
- Once we made sure that we could retrieve and convert the input string, we tested to make sure that we could make the updated API call to in order to receive the requested show
    - By using a fetch statement to make the API call, we would use catch(error) to find any error that could possibly occur when making the new API call and console log if there was one

- After we ensured that we could get the correct shows from the updated API call, everytime we made a search query, we checked to make sure the shows being displayed matched the search query we inserted. For instance, if we typed in 'Stranger Things', we confirmed that an incorrect show such as 'Sense8' wouldn't show up in the result showcards.

- After ensuring that everyone with the API call was working properly, the rest of the code to actually display the showcards and make them clickable to the correct forum is the same code and logic used in the home page where there were separate testing strategies used there

Show Cards

- To check that information was being grabbed from the Netflix API before implementing the Cards, we used console.log() inside our first fetch call and checked json that was retrieved by doing console.log(json.titles) which provided a list of all the shows using the API.
- Once the data was ensured to be correct from the database, we put the json results into a useState() function to hold the grabbed values, then mapped the data inside the list to be console logged using the format assigned by the API, with calls
    - console.log("title", show.jawSummary.title);
    - console.log("showID", show.summary.id);
- This data is verified to be able to loop over the list and create Cards to hold the show title and image, which is verified by the correct show title and image appearing on the card
- Now we need to make sure that we could navigate to the forum page and have the data passed so when the card is clicked it will use navigate() to move to the forum page and use local storage to pass the show data including the title, image, id and more
- To verify that the correct show data was updated and passed to local storage, when the Card is clicked and the forum page is navigated, variables pull from the local storage for example
    - Const Title = localStorage.getItem('title');
    - console.log(Title);
- Which will show that the correct show data was updated to be used for the Forum page, but with more variables that are passed to be used for the forums portion of the application
- Once the testing of static cards finished, we moved to having the cards be on a carousel like the Netflix UI, so we implemented a function to handle when the arrows are clicked by inputting left or right when clicked
- This was tested by implementing console logs inside the handleClick function that would update whenever the left or right button is clicked which would show that the buttons will update the position reference of the cards in the row, using
    - console.log(listPos);
- To make sure that the cards actually moved we connected the buttons and made a reference to the affected row and manually clicked the arrows, which showed that the cards all moved depending on the left and right buttons
- After these tests with just one row of cards, we implemented 2 more rows with the same behavior barring changes to the variables and testing manually showed that it works exactly the same as the first row.

Forum

- For the forum, it builds off the Show Cards passing data so we had to make sure that data was correctly passed to the forum page, so when a Show Card is clicked it console.log() what is associated with the card and navigates to the Forum page we grab from the local storage and console.log() the data to see if the data matches.
- Once it is ensured that our Forum page can get the data from the clicked Show Card, we display the title and picture from the card on the page, then we immediately pass the show id that was grabbed to a fetch request to the API that will grab all the seasons related to the show, which we console logged the json with
    - console.log(json[0]);
- This printed out all the seasons related to that show and we could check if the number of seasons matched what the API website showed.
- Once we got the seasons working, we implemented it into a dropdown menu that would update a value holding the season id, which we tested with a console.log() to ensure that selecting a season was being updated
    - console.log(currSeason);
- After we verified that seasons were correctly being assigned and selected, we pass the season id into our second fetch to the API which will grab all the episodes of the current season, which we tested by printing out the results of the episodes which came out as a list and matched up with the show's data
    - console.log(json[0].episodes);
- We confirm that correlated episodes to the current season are being grabbed correctly, so we put it in a drop down and tested the output to make sure that our episode selector was updating by console logging the current episode id. We also made sure that you can't pick any episodes until the season has been selected
- After these two tasks, we had to test that we could grab posts from our database using the provided season_id and episode_id, so there is a GO button that will call on a function called forum() that will enact an Axios GET request that will grab the posts.
- To make sure that the posts only came from the season and episode, we console log the results episode and season id with a console log of what was sent
    - console.log("posts retrieved", response.data)
    - console.log("season id, season_number_forum)
    - console.log("episode id, episode_number_forum)
- Once we could verify that the posts were retrieved from the database by comparing the results to what was showing up in our database workbench, we then map the list of posts we retrieved into cards that display downwards, which will now display the user, title, content, and the likes and dislikes of a post.

Like Button

- Since the logic for the like/dislike button is quite extensive, there were many console logs and manual database checking that was necessary to ensure that each component was working properly
- In the beginning, there was only a local value to store the likes and dislikes value
- This was to ensure that the activeBtn logic was working correctly detached from the database.

- Once we confirmed this, we started to implement the storing and retrieval of likes/dislikes from the database (still with only one global like and dislike value).

    - console.log(‘current btnStatus: ‘, btnStatus);

- After we verified that the overall global like/button worked, we connected it to the DB
- To test this, the printed out the Axios post calls to make sure the forum was getting the correct data from the DB
    - console.log(response.data[0].like)
- In addition, we also logged which Axios Post scenario we were currently in to determine if we correctly setting the activeBtn state

- **PLEASE NOTE:** The like/dislike button functionality is not working completely as documented in the Working Prototype Known Problems Report. These errors come not in functionality of the button, but in how it interfaces with the databases.