Design Inspection, Code Inspection and Unit Testing

Brian Duffy, Akhil Agrawal, Adam Johnston, Shivan Desai

Design Inspection

Defect #	Description	Severity	Correction
1	Spotify's Authorization Code Flow was not returning an authorization code as expected.	1	We decided to switch to the Implicit Grant Flow, which was easier to implement, and forces the user to login again after a certain time limit, which allows us to make sure they are still listening to the tracks.
2	The get request for authentication was returning useless JSON instead of the auth token we needed.	1	We opened the verification in another window using window.open(), allowing the user to okay the use of their credentials and returning us an access token
3	We were unable to communicate the auth component from one component to another.	2	We solved this by using the localStorage.setItem() method which stores data locally so we can retrieve the necessary data from other components.
4	The window used to validate credentials would not close upon finish. Calling window.close would immediately close the window before the user could verify.	1	We ended up creating another component and used it's linked html as the callback url. Then we utilized Angular's ngOnInit() which executes code before the component is displayed.
5	We had some issues retrieving data from various get requests, as it returned an Observable.	1	In order to access this observable, we had to use .subscribe and map the objects to either json or our own preset containers.

6	The track data would not update upon completion of a get request and setting of the values in the class.	3	We made sure any displaying of information was done after the full method completion.
7	We had issues finding ways to bind data across parent and children components.	2	We ended up creating a SpotifyService which is defined in each component's constructor. This allowed us to communicate to the Spotify Api by using only one service that shares the spotify data across all components.

Code Inspection

Defect #	Description	Severity	Correction
1	We defined variables with no type. TypeScript allows this.	3	We figured out that to declare variables like in JavaScript, we need to use the 'let' keyword. In addition to this we also gave variable types as opposed to leaving them undefined.
2	In order to test our web app locally without a server, we needed to somehow handle cross-origin resource sharing errors. Which are a security measure for untrusted urls.	2	We found a temporary solution called an interceptor that basically intercepts the cross-origin resource sharing errors. Once we get the server connected to our app, we will no longer need to intercept this.
3	When creating headers we called HttpHeaders().set() for	2	We did some searching through HttpHeaders() api of

	each header variable we wanted to use. Which ended up not actually working.		Angular, and found out to add additional header variables we needed to use the .append() method
4	When trying to get data from get requests, we tried using map() to put data into containers.	2	Angular 4 uses the .subscribe() which allows us easily map the json returned from the get request to specified parameters. Example, name: data['name'], This enables the use of custom containers to accommodate for the different json objects returned from Spotify.

Unit Testing Defects

We are using Jasmine as our framework for unit testing. Jasmine is known for being an automated testing framework, as it is executing unit tests automatically along with Karma. It allows the comparison of expected values with actual values and is able to run the tests whenever necessary by calling ng test.

Login Component:

This component is where get access to our web app. Here they click the login button and a new window opens where they input their Spotify username and password. They are then redirected to the Home Component.

Defect #	Description	Severity	Correction
1	Login Component does not redirect after a user clicks the login button.	2	We added a router link route to the button elements attributes. That redirects to Home Component on click.
2	Upon refusal to accept Spotify permissions of our web app	2	Not yet solved. We need to figure out a way to navigate to

	the Login component still redirects to the Home component.		a component after the user closes the Spotify authentication window.
3	We currently have implicit grant authorization implemented. The issue we have is that the user needs to continually log in for a new auth token.	3	Currently not fixed. We need to implement authorization code flow for Spotify oauth.

Home Component:

The home component is going to be where we display the songs of the playlist and the users listen to the songs in the playlist. In other words this will be our main component for user interaction.

Defect #	Description	Severity	Correction
1	Upon login components redirection to the home component, the component is not created	1	Added dependency to the app.module.ts file.
2	If the Id's are invalid, an error is thrown and the html doesn't draw	2	We used ngIf to ensure non null values are displayed and null values are not.

Radio Component:

Will later be renamed appropriately a more appropriate name. As of now the radio component has our search functionality to search for tracks in Spotify.

Defect #	Description	Severity	Correction

1	Users can enter invalid track names in the search form. No errors are thrown.	3	Currently not fixed. We need to check the contents of the get request json data to see if the array of tracks is empty.
2	Users can still do searches even if they are not logged in.	3	Not fixed. We need to redirect them to the login page if they get a 401 response from the get request.
3	Users can enter a blank input as the query term in the search.	3	Not fixed, we need to add input validations
4	If the user searches with the query term left blank before any other searches, a list of songs associated with the term ‘undefined’ are listed.	3	Not fixed, we need to add input validations
5	If user searches with a non-null query term and then searches with the query term left blank, the results of the search do not update and the server responds with a ‘400 Bad Request’ error.	3	Not fixed, we need to add input validations