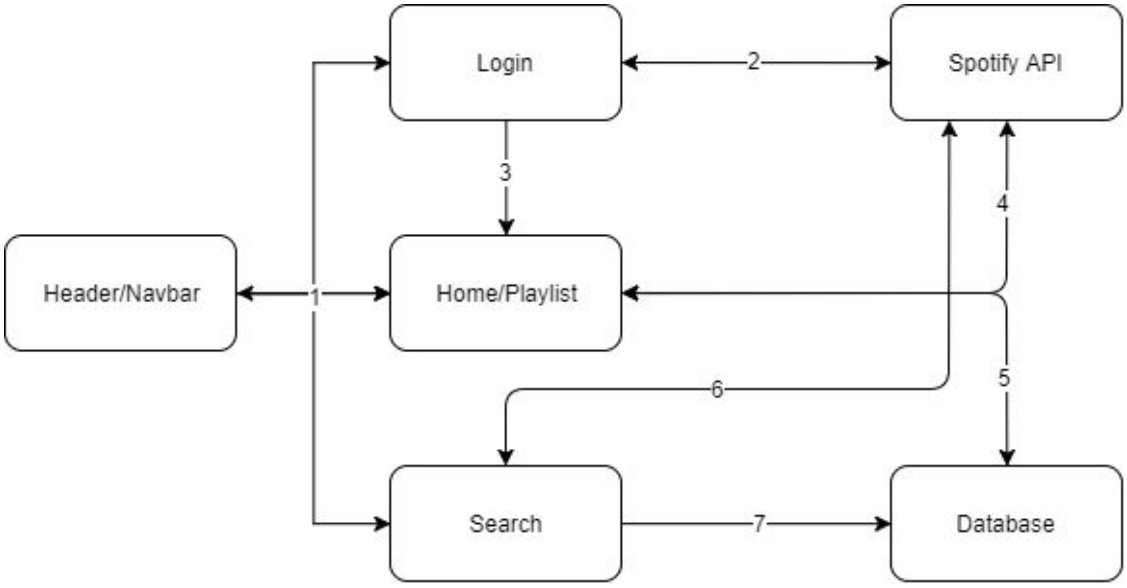
Incremental and Regression Testing

Brian Duffy, Adam Johnston, Akhil Agrawal, Shivan Desai

1. Classification of Components

a. Define all components



1. **Navigates between components**
2. **Authenticate User and Retrieve Auth Token**
3. **Redirect After Login**
4. **Check if Playlist Exists, Update with New Tracks**
5. **Request and Retrieve Tracks for Playlist**
6. **Search for and Retrieve Tracks to Add to Playlist**
7. **Add Tracks to Database**

Input/Output/Dependencies

- **Header Component**
  - **Inputs**
    - **Button Press**
  - **Outputs:**
    - **Local Redirection**
  - **Dependencies (Called by):**
    - **App Component**
  - **Dependencies (Calls):**
    - **Login, Home, Search Components**
- **Login Component**
  - **Inputs:**

        - **Button Press to Login**
    - **Outputs:**
        - **Redirects to Home Component**
        - **Auth Token**
    - **Dependencies (Called by)**
        - **Header Component**
    - **Dependencies (Calls)**
        - **Home Component**
        - **Spotify API**
- **Home Component**
    - **Inputs:**
        - **Button press to play a song**
    - **Outputs:**
        - **A List of Tracks to Play**
        - **Streaming of Track Data (through Spotify)**
    - **Dependencies (Called by):**
        - **Header and Login Components**
    - **Dependencies (Calls):**
        - **Spotify API**
        - **Database**
- **Search Component**
    - **Inputs:**
        - **Query string for track searches**
        - **Submit button**
    - **Outputs:**
        - **List of Tracks**
        - **Buttons to Add Tracks**
    - **Dependencies (Called by):**
        - **Header Component**
    - **Dependencies (Calls):**
        - **Spotify API**
        - **Database**

b. Which Form of Incremental Testing

We are using Bottom-Up Testing rather than Top-Down, as our test cases contain "Drivers" that call our submodules. We have already created our submodules and are testing their interactions and outputs through the use of said drivers. We do this instead of calling submodules that do not exist, which would be replaced with stubs in Top-Down

testing. As an example, we have a test case making sure the Search component is retrieving results correctly. The Search feature is implemented, so we use a Driver with preset search values to make the search call to query the Spotify API and retrieve the data.

2. Incremental and Regression Testing

Automation

- In order to do incremental and regression testing, we used Jasmine and Karma. Jasmine is a testing framework for angular, which allowed us to setup a series of tests for Karma to run. The creators describe the Karma environment as “a place where developers can just write the code and get instant feedback from their tests.” This means we were able to:
    1) Have our tests run automatically through Karma
    2) Get immediate information on whether our test cases passed or failed
    3) And if they failed, why they failed

Home Component	Incremental Testing		
Defect #	Description	Severity	How To Correct
1	We noticed that repeats are still being added to the playlist from the Search component	3	We need to add a check to make sure that the track being added hasn’t already been added. (check the song id)
2	On the spotify database, new empty playlists are added.	1	We are attempting to use a switch map to make sure that we are actually adding to one playlist, and not creating more.

Home Component	Regression Testing		
Defect #	Description	Severity	How To Correct

1	We need to make sure that the things we add from the search component are updated to the home component as well as the database.	1	We did a bit of incremental testing to see if we were getting the data. We were getting data correctly. Displaying it however we were not, so used an ng-container to store the tracks and display them.
2	Even when there is a track in the playlist, we could still see the error message of “no tracks found”	3	We made that element visible based on the fact that there was valid data in the data structure that locally stores the playlist.

Login Component	Incremental Testing		
Defect #	Description	Severity	How To Correct
1	Login is still a persistent pain. We continually have to login very often, regardless of the component we are testing.	2	We need to implement Authorization Code Flow. We weren't able to do so within a week because we needed to add more functionality.
2	When a user ends up making a request, for some queries they need to be logged in.	1	We have two options, we can implement Authorization Code Flow, or we can implement a method that logs them in if a request is not processed due to a 401 error.

Login Component	Regression Testing		
Defect #	Description	Severity	How To Correct
1	Upon cancellation of	2	We partially fixed this by

	the login, they aren't redirected or provided an error message.		adding scopes to the login process. The scopes we require do not need private information. So Spotify auto accepts those scopes.
2	We still have yet to figure out how to keep the page from navigating to the home page without finishing authentication.	2	In testing we can use something called a spy which has the ability to check weather a certain function has been called of a class. Potentially there is something like that outside of the testing environment. If so then we can use it to see when the function window.close is called then we navigate to the home component.

Search Component	Incremental Testing		
Defect #	Description	Severity	How To Correct
1	When adding a component from the search results to the home component, we are getting duplicates.	2	Just like in the home component, we need to do some sort of database call to see if the track somebody wants to add is not already there. (if the id's are the same, reject them adding that track).
2	We have found out during testing that our tests are unable to sign in but for some reason we are still able to retrieve search data despite not being logged in	2	We need to figure out a way to asynchronously test and see if the window for login is displayed and actually render the login page of spotify. Currently it opens the window and nothing is showing up. This

	visibly.		however doesn't happen outside of testing.

Search Component	Regression Testing		
Defect #	Description	Severity	How To Correct
1	Users could search without being logged in.	2	If the user gets a 401 error, we log them in and try again.
2	Users could search invalid characers.	2	We added a filter for alpha numeric characters to reduce this from happening. We don't know the full extent of what constitutes as an invalid character.

Header Component	Incremental Testing		
Defect #	Description	Severity	How To Correct
1	We needed to connect all components to one another.	1	We ended up using a header which injects child components into a tag called <router-outlet>.
2	We have noticed that upon clicking a header link that corresponds to an already displayed component, the page is reloaded.	3	We fixed this by using the Router link module of angular.

Header Component	Regression Testing		
Defect #	Description	Severity	How To Correct
1	We had issues early on with redirection to components.	1	We implemented some tests to solidify our method of navigating between components. What we used was a router outlet and a router linker which injects html components into a custom html tag by name.
2	We had tested component redirection between only two components. But not complex tests.	3	We have developed a few tests to prove the functionality of basic navigation. Now we need to make some more complex tests.