Team Number: Section 011 team 4

Team Members:

Bernardo Fortunato

Bo Zhang **(?)**

Dalbir Brar

Jared Jewell

Jacob Levato

Team Name: Mr. Worldwide

Project title: Vynilla

Test Plan

Step 1: Identify critical points for project functionality. Ex: User authentication, Spotify API communication, etc.

Step 2: Determine the best way to test their functionalities.

- User auth: Ensure that a newly registered user is inserted into the database with the profile name of their choosing.
- Spotify API Communication: Ensure that, when given a search term (say, Doja Cat), the returned object contains that term.

Essentially, we just want tests to validate that, given input, we get the expected outputs, when considering any of our various tasks (API, database communication, etc).

```
//Ensure that, when registering, no fields are left empty
describe(“Register Form Validation”, () => {
        it(“Nothing is empty” () => {
                expect(req.body.name).to.not.be.empty;
```

```
            expect(req.body.username).to.not.be.empty;
            expect(req.body.email).to.not.be.empty;
            expect(req.body.password).to.not.be.empty;
            expect(req.body.confirmPass).to.not.be.empty;
            });
        it("Confirm Password and Password to match" () => {
            expect(req.body.password).to.eq(req.body.confirmPass);
            });
        });

//User Sign In
describe("Log In Form Validation", () => {
        it("Nothing is empty" () => {
            expect(req.body.email).to.not.be.empty;
            expect(req.body.password).to.not.be.empty;
            });
        it("Verify User Information" () => {
            expect(req.body.email).to.eq(returnFromDatabaseEmail);
            expect(req.body.password).to.eq(returnFromDatabasePassword);
            });
        });


// Spotify Unit Testing
//Ensure that, when querying Spotify with a search term, that search term exists in the returned
object
describe("Spotify Information", () => {
        it("Spotify song return" () => {
        //ensure search returns intended item
            expect(req.body.songname).eq(
                spotifyApi.searchTracks(req.body.songname).body.tracks.items.name
            )
        });
});
```

Step 3: Once the above bases are covered, proceed onto the next tasks and iteratively repeat the process.

Individual Contributions

- This deliverable includes a couple of lines about each team member's contribution towards the project.
- Include a link to the latest commit made by each team member on the GitHub repository.
- Share a screenshot of the project management board being maintained for this project indicating the status of the tasks at hand.

- Jake: I worked on searching for friends and creating a relationship table between users. This allows for a “Friends” portion of Vynilla. You can now search for users and send a friend request. The user who receives the friend request will see the request on their profile page and can accept or deny the friend request. Accepting the request will place the friend’s username in each of their friends list. Friends will soon be able to see what each friend was most recently listening to. I also added Jared’s work connecting to Spotify to Vynilla. When you register for Vynilla, you are prompted to connect to spotify and it now displays your spotify profile picture. (Latest Commit: https://github.com/CSCI-3308-CU-Boulder/3308SP21_011_4/commit/4f757a62f569ff25dcb23b96e47f8fc24ced5890)
- Jared: I’ve continued working on our Spotify-API integrations—I was able to create a playlist from songs supplied in an array. This involved repeatedly querying the Spotify API while iterating through the array, and storing the results in another array, then using the results to create a playlist on the user’s account, and finally, populating the array. This is an important proof-of-concept for further manipulation.

    - Update: I was able to make it less ‘hard-coded’. The user can now create a playlist via a modal, wherein they can enter a name, description, and whether it should be private or not. Then, they can search songs from the Spotify API, several results of which are displayed for the user to select from. The selected songs can be added to the playlist. This is essentially the queue functionality that we desired.
    - Update2: I merged Spotify-API-practice all by myself! King of merge conflicts!
        - https://github.com/CSCI-3308-CU-Boulder/3308SP21_011_4/commit/206e8e78bf627211896de4427377a514579c7e2d
- Dalbir: I worked on the spotify-API with Jared creating one page to set the user’s profile in the database. I worked on another page to retrieve the profile page from the database to show on the profile page. We can grab Artists, songs, and albums now from spotify.
    - https://github.com/CSCI-3308-CU-Boulder/3308SP21_011_4/commit/90b86efdd082a1e3d270a9ec909096bf6fced617
- Bernardo: I worked on the Spotify API and implemented search songs through the API and helped with the milestone documents.
    - Link: https://github.com/CSCI-3308-CU-Boulder/3308SP21_011_4/commit/f7cb8cf8cfa602b15cd99ad851b1c9449bd5b9a2

Risks

Organisational Risks : Occurs due to lack of resources to complete the project on time, lack of skilled members in the team, poor planning, etc. For example, we cannot allow ourselves to, by some mishap of planning, all be working disjointedly on the same task—at this point, we need to be evenly distributing our efforts across the remaining tasks instead of all working separately on a single task.

Technical Risks : Occurs due to untested code, improper implementation of test cases, limited test data, etc. In our case, we must ensure proper error handling, so that a variety of edge

cases cannot occur—for example, we wouldn't want a user to be able to access another user's account via some database query mishap on our end.

Business Risks :

On our end: Spotify's risk of us messing up and accessing someone's private data. Which, if we've properly handled our routing, user-authentication, handling of sensitive information, and so forth, wouldn't happen.

On their end: Content creators leaving spotify, especially smaller artists.