CSCI 201 Group Project
Testing Documentation
Group Members:

Jay Doshi Tommy Weaver Bryce Check
Thomas Chen Dayu Jiang

Introduction

The following document will go over how our app will be tested through the validation and verification portions. The validation will be presented as a whole for the app and we will use ourselves and friends to cover portions of both white and black box testing. The verification portion will be partitioned as such; Suggestions, Front-End and Back-End. Each of the sections will describe the verification portion through providing a list of test cases and the purpose behind each of the tests.

Validation

In order to validate our application we will design automated tests for stress and white box testing. We will our ask users to provide qualitative and quantitative feedback. We will ask users to rank how well they think our recommendations worked for different suggestion engines, front end designs, as well as how it ranks as compared to other similar services like Spotify's Discover feature or Pandora Radio. Most of these tests will be qualitative, but the ranking will be formed into quantitative answers, something to the accord of "Much better, better, similar, worse, much worse" as options for ranking our service compared to others. We believe these tests will help form a solid validation test.

Suggestions

Test 1

White Box Testing - Supply a song that exists in the database and make sure it is able to make a playlist of similar songs. If no songs can be matched to that song then the user should be notified with a message that the playlist builder cannot find similar songs.

Test 2

White Box Testing - Supply a song that exists and that sounds like similar to some songs in the database and see if a playlist can be built from those using the music comparison algorithm we have.

Test 3

Unit Testing - Take the raw audio data of two songs which are similar and see if the comparison method will match the songs in a playlist.

Test 4

Black Box Testing - Have users rate our suggestion engine as Much better, better, the same, worse or much worse than Spotify's Discover feature or Pandora Radio.

Front-End

Test 1

White Box Testing - Search for a song that does not exist in the database. The user should be notified that the song doesn't exist in the database.

Test 2

Black Box Testing - A song is supplied and the correct output is displayed to the user which is either the cannot find in database messages and cannot make playlist message or a valid playlist which is able to be listened to by the user.

Test 3

White-Box Testing - Test the feed view by searching for a song on one phone, another phone has the feed view open and the user should show up on the feed.

Test 4

White-Box Testing - Test the profile view by logging in and out.

Test 5

Unit Testing - When given data from the back-end, front-end feed page should display the information like a feed. One unit of feed should include the song name, song artist, song album, song image, user name, and user image.

Test 6

White Box Testing - When a user listens to a song, all users should be about to see the feed on the feed page instantly, the newest feed should appear on the top.

Test 7

Unit Testing - When a user clicks on the user image/user name on the feed, he/she should be taken to that user's profile page..

Test 8

Unit Testing - When a user clicks on the song image/song name on the feed, he/she should be given to option to add the song to the current playlist or to create a new playlist for this song and play it.

Test 9

Black Box Testing – Playlists should be display in profile page, with playlist name and the song image of first few songs in the playlist.

Test 10

Unit Testing - When clicking the playlist, user should be able to listen to the songs in the playlist in order.

Test 11

Unit Testing - User should be able to add songs to the current playlist. The added song should be at the end of the playlist. The added song should be played according to playlist order.

Test 12

White Box Testing - Have users give qualitative feedback on the design of our app as to improve future iterations.

Back-End

Test 1

White Box Testing - Make sure that a user is created when the register endpoint is hit on the backend. This ensures that names, email, username, passhash, and salt are all input into the Users table in the database.

Test 2

White Box Testing - Make sure that a login works correctly. This involves the proper retrieval of the password hash from the database as well as the salt from the database. Successful hashing of the salt and password provided by the user at login shows that the SELECT queries for logging in are properly working. Make sure that the login rejects incorrect passwords.

Test 3

White Box Testing - When a user searches for a song that doesn't exist, make sure that the database abstraction returns just the fact that the song doesn't exist. This could be done by making sure that the amount of returned rows from the database is not equal to 0.

Test 4

White Box Testing - When a user searches for a song that does exist, make sure the proper song information, MP3 file, and other metadata are returned from the database for the correct song.

Test 5

White Box Testing - When a user adds a song to the playlist, make sure the playlists table is updated accordingly with the new entry at the bottom of the playlists table with the proper information.

Test 6

White Box Testing - When a user requests to see his/her playlists, make sure only the playlists the the user created or has decided to follow are returned.

Test 7

White Box Testing - Make sure that when a user deletes a song or playlist from his/her library, then the change is reflected in the database and what the user sees.

Test 8

Unit Testing - Supply an MP3/MPEG4 file to the conversion method and ensure that raw song audio data is the output.