Project: Resonate

Team Members: Jerry Tejada, Daniel Koo, Isaac Taylor, Tori Yi

Proposal:

An online platform where music writers, producers, performers can come together to create music from anywhere in the world. Writers can upload sheet music/chord progressions, producers can add tracks or ideas, performers can play the parts available. Allows for private and shareable collaborative folders and groups between different types of users to create music as an end product. Allows for private groups as a "premium" feature or open-source music projects where anyone can participate (think github).

Performers could participate by completing a part of the producer's music. When uploaded, the producers can choose to work with different musician's styles and audio files. Listeners could "Like" certain parts, resulting in "popular" track for a specific part such as drums, main vocal. The separate parts of the music could have a "Producer's Choice" track, and a "
Community's Choice" track. *Create, Collaborate, Resonate.*

Deployment Document

Team Members: Jerry Tejada, Daniel Koo, Isaac Taylor, Tori Yi

1. **Download Eclipse from http://www.eclipse.org/**
2. **Install Tomcat Version 9 from http://tomcat.apache.org/download-90.cgi**
3. **Install MySQL from https://dev.mysql.com/downloads/mysql/**
4. **Install MySQL Workbench from https://dev.mysql.com/downloads/workbench/**
5. **Clone repo from https://github.com/dkoodev/csci201_resonate**
    a. The libraries used include jQuery and BootStrap these are included when the repo is cloned
6. **Load the Resonate project into Eclipse**
    a. File->Open Project from File System
7. **Find the location in which you cloned the repo. Navigate to csci201_resonate/Resonate/src/com/resonate/Config.java**
    a. Update the “pathToProject” string, redirecting it to your new location.
8. **Open MySQL Workbench to load the database.**
    a. Connect to the mysql database.
    b. Click the “Open Sql” icon, and browse to csci201_resonate/Resonate.sql
    c. Run the script
9. **Right click “index.jsp” in Eclipse, click “Run as…” and go to “Run on server” to deploy project.**
10. **Progress to click Sign Up to create your own login!**

High Level Requirements

0) **<u>The Top Navigation Bar</u>**
Purpose: This is a global bar across all pages, providing navigation links between pages for users to follow.

Visibility: The visibility will change, not per page, but based on whether or not the user is logged in.

Components:
- Logo on the top left corner
  - Redirects to the promotional page.
- Browse Projects Button
  - Even when user is not logged in, guests can listen to the music created by the community. Their selection/preference of music, however, will not be saved.
- Login Button
  - This button will reveal a small box with textboxes and buttons to allow user log in.
  - A mouse-over will reveal a form to login. Failure will lead to a login page to try again, success will lead to the Account Page (#6).

When a user logs in, the following buttons will replace the login button:
- My Projects
  - Leads to a list of projects the in which the user is currently involved.
- Account
  - This button will not say “My Account”, it will be the user’s profile photo.

1) **<u>Promotional Page</u>**
Purpose: This page will introduce users to the concept of RESONATE, displaying promotional information with links that allow users to either log in or sign up.

Visibility: Guests that are not logged in.

Components:
- Navigation Bar (Defined in #0)
- A blurb about what the application does below the sign up.
- Sign Up
  - Will be in the middle of the page and when clicked will lead to the sign-up page (page #2)

2) <u>Sign-up Page</u>

Purpose: This page is dedicated to creating an account in Resonate. Information such as username, email, password will be collected to create the account.

Visibility: Users that are not logged in.

Components:

- Username
  - This name will be associated with your account and used to sign in to your account. This username will also be used to replace the login button on the top right corner in the navigation bar for personalization.
- Email
  - A confirmation email will be sent stating that the user created an account.
  - A verification link will be attached to connect the email to the account.
- Password
  - The password must be inputted along with the username to sign in.
- Confirmation Password
  - User must re-enter password to avoid any mistakes made in the first input of the password.
- Create Account Button
  - Submits all the information and checks if any of the inputted information is not acceptable.
  - If information is all acceptable, leads to “Browse Projects” (page #3) with logged in state.

3) <u>Browse Projects</u>

Purpose: The browse projects page will allow the user to view previous projects, individual tracks, and collaborations posted by other users.

Visibility: Every user

Components:

- Navigation Bar
- Search Bar (Beneath the Navigation Bar)
  - Search terms can be used to find what the user is trying to find. Could be the name of a song, user, or mix.
- Options Bar
  - Choices will be listed as genres and once one is chosen, search hits are sorted by the chosen option
  - Does not change the number of results, but only sorts the results.
- Filters Bar (Left sidebar)

    - Search hits are filtered by the chosen filter. In our web service, the filters could be by genre, music type, complete/incomplete, etc.
    - Changes the number of results that are being shown.
- Results
    - Having the Options Bar as the top bar and the Filters Bar as the left side bar, the results are displayed in the middle as blocks.
    - The blocks each represent various projects, users, or mixes of songs
    - If the user clicks on a result, such as a song, it leads to the "Audition page" for that song (page #5)
- Most Popular
    - Right sidebar that will list the most popular projects, including songs, mixes, instrumentals.

4) <u>My Projects Page</u>
Purpose: The browse projects page will allow the user to view previous projects, individual tracks, and collaborations posted by other users.

Visibility: Logged in users

Components:
- Navigation Bar
    -
- List of projects involved with the user, including songs, mixes, and instrumentals created.

5) <u>"Audition Stage" Page</u>
Purpose: This is where users can upload and "audition" their interpretation/part of a song. This is also where the songs are displayed and chosen by the user to be played in their particular mix.

Visibility: All Users

Components:
- Navigation Bar
    - Same as 0)
- Can view all the interpretations of a particular song
    -
- Different sounds tracks can be grouped by their roles, such as "Drums", "Guitar", "Vocals", etc.
- There will be two featured tracks, a "Song Writer's Choice" and "Community Choice."
- Each sound track can be selected to be in your "Mix".
- User can vote for which track is the best in its particular role.

- Each role will have multiple tracks. The tracks will be sorted in “Top Voted,” meaning it received the most votes from the community. This sorted list will be sorted live, meaning if the number of votes for a track changes after the page has loaded, the sorted list will reflect that change.
- Play button to play the song with all the tracks chosen.
    - HTML <button> with an onclick function

6) <u>Account Page</u>

Purpose: This page will allow you to handle your account settings. You can change your email, username, profile picture, password, etc.

Visibility: Logged in users

Components:

- Navigation Bar
- An account overview section which will display your profile information
- Edit profile button that allows you to change your account’s settings

Technical Specifications

The Technical specifications include hardware requirements, software requirements, benchmarks to meet, estimate of hours, estimate of cost.

Technologies that will be used in this project:

Front-end: HTML, CSS, Javascript, AJAX, JQuery, Bootstrap

Back-end: Java, Jsp, Mongodb or MySQL

Tasks 0-6 are delineated to match the high-level specification details, 7+ are tasks needed to support the first 7. Each task will not necessarily be completed in the order provided.

0) **Top Navigation Bar** (1 hour)

Components:

- Logo on the top left corner
  - HTML <img> tag with an <a> tag that redirects to the promotional page (#1)
- Browse Projects Button
  - HTML <button> tag with <a> linking to Browse Projects Page (#3)
  - Selection/preference of music will not be saved when in the not logged in state
- Login Button
  - HTML <button> tag with <a> links to either sign up or log in page
  - Hovering over this button will reveal a small box containing HTML textboxes and buttons allowing the user to enter their login credentials
    - Failure will lead to a login page to try again, success will lead to the Account Page (#6)

When a user logs in, the following buttons will replace the login button:

- My Projects
  - Leads to a list of projects the in which the user is currently involved.
- Account
  - This button will not say “My Account”, it will be the user’s profile photo.
  - This will be accomplished through AJAX/Jsp with an HtmlSession object. If the user has an active session, their account will be queried in the database and the information will populate the buttons.

1) **Promotional Page** (1 hour)

Components:

- Navigation Bar (Defined in #0)
- Html page (hard-coded) displaying a description of the application
- Sign Up Button
    - HTML <button> with link to the Sign-up page
    - When clicked will lead to the sign-up page (page #2)

2) **Sign Up Page** (2 hours)

Components:

- HTML<form> with text inputs, checked in real time with javascript:
    - Username: HTML input type - text
    - Email: HTML input type - email
        - Confirmation email will be sent to this address which will verify and link the email to the account
    - Password & Confirm Password: HTML input type - password
        - Password limitations: 8-14 characters
- Create Account Button
    - Submits all the information to the server and checks if any of the inputted information is not acceptable or does not match
    - If information is all acceptable, leads to “Browse Projects” (page #3) with logged in state.
        - HTML submit button, AJAX will confirm data by querying the database, then forward if successful.

3) **Browse Projects Page** (4 hours)

Components:

- Navigation Bar
- Search Bar (Beneath the Navigation Bar)
    - Search terms can be used to find what the user is trying to find. Could be the name of a song, user, or mix.
    - A form can be used along with a servlet to do form validation and giving back a response from the database. The results will be injected into the original html using ajax and jquery.
- Options Bar
    - Choices will be listed as genres and once one is chosen, search hits are sorted by the chosen option
    - Does not change the number of results, but only sorts the results.
    - JQuery can be used to sort the results.
- Filters Bar (Left sidebar)
    - Search hits are filtered by the chosen filter. In our web service, the filters could be by genre, music type, complete/incomplete, etc.

        - Changes the number of results that are being shown.
        - JQuery can be used to filter the results.
    - Results
        - Having the Options Bar as the top bar and the Filters Bar as the left side bar, the results are displayed in the middle as blocks.
        - The blocks each represent various projects, users, or mixes of songs
        - If the user clicks on a result, such as a song, it leads to the “Audition page” for that song (page #5)
    - Most Popular
        - Right sidebar that will list the most popular projects, including songs, mixes, instrumentals.
        - This will be a HTML div that contains components that are updated live. This could also be a multithreaded backend checking for popular tracks.

4) **<u>My Projects Page</u>** (4 hours)

Components:

- Navigation Bar (Defined in #0)
- List of projects involved with the user, including songs, mixes, and instrumentals created.
    - Lists with <a> links to the projects, retrieved from the database by JSP

5) **<u>“Audition Stage” Page</u>** (6 hours)

Components:

- Navigation Bar (defined in #0)
- Can view all the interpretations of a particular song
    - Main body will have a container that can hold lists of “audition” objects/cards. Each card will have a checkbox to indicate that the certain track was selected. A volume slider will allow simple volume control for the track.
- Different sounds tracks can be grouped by their roles, such as “Drums”, “Guitar”, “Vocals”, etc.
    - Each “audition” card will be grouped together by a HTML <div> tag. They will be separated visually using CSS, perhaps a bootstrap component.
- There will be two featured tracks, a “Song Writer’s Choice” and “Community Choice.”
    - For each “group” of roles, there will be two tracks at the top, the two featured tracks will have different CSS to indicate that they were voted for the most.
- Each sound track can be selected to be in your “Mix”.
    - HTML Form with checkbox will be used to add a track to the current mix of tracks.

- User can vote for which track is the best in its particular role.
    - Two HTML buttons will represent the “upvote” and the “downvote”. This will also be a form submitted to increase the vote count for the particular track.
- Each role will have multiple tracks. The tracks will be sorted in “Top Voted,” meaning it received the most votes from the community. This sorted list will be sorted live, meaning if the number of votes for a track changes after the page has loaded, the sorted list will reflect that change.
    - The live count of the votes will be a multithreaded component of the project. There will be a thread in the backend checking for updates on the vote count, and when it does find it, it will update the card to have the accurate vote count.
- Play button to play the song with all the tracks chosen.
    - HTML <button> tag that enables the tracks.
- Download button to download all the different sound tracks that are checked for further editing.

6) **Account Page** (1 hour)

Components:

- Navigation Bar
    - Same as 0)
- An account overview section which will display your profile information
    - A list of all the user’s information
- Edit profile button that allows you to change your account’s settings
    - HTML <button> with <a> link to edit profile which leads back to 2)

7) **Database** (1 hour)

The database will support each of the pages above, with the following tables:

a) Accounts: id$^+$, username, password, email, name, photo, bio (etc)
b) Auditions: id$^+$, accountid*, projectid*, name, type, instrument, filename
c) Projects: id$^+$, ownerid*, name, genre, private (bool), users (This is the only one that needs a list of users involved, hence \`users\`. The rest will be linked through keys and queries.)

* These fields are linked to fields in other tables, eg. accountid/ownerid = \`id\` in \`Accounts\`
+ These fields are unique keys.

8) **Design technical notes** (0 hours)

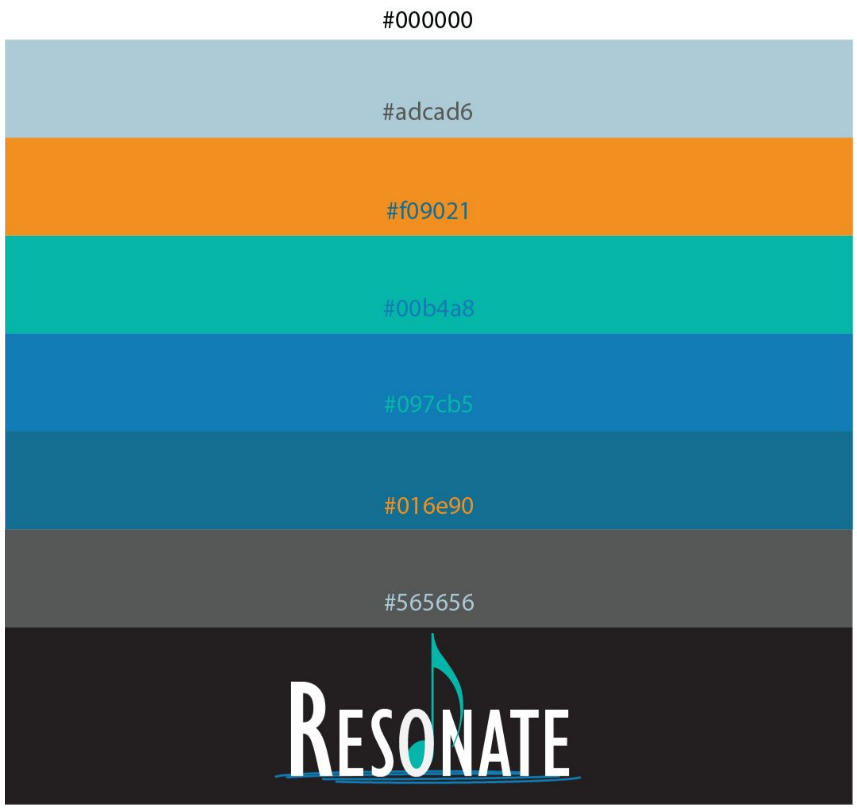
The content here is not for a specific part of the program, but rather an overall technical reference to create a cohesive design between pages.

Fonts:

Titles: Ubuntu (Google Fonts)
Paragraphs: Open Sans (Google Fonts)
Logo: Gill Sans MT Condensed (Adobe Typekit)

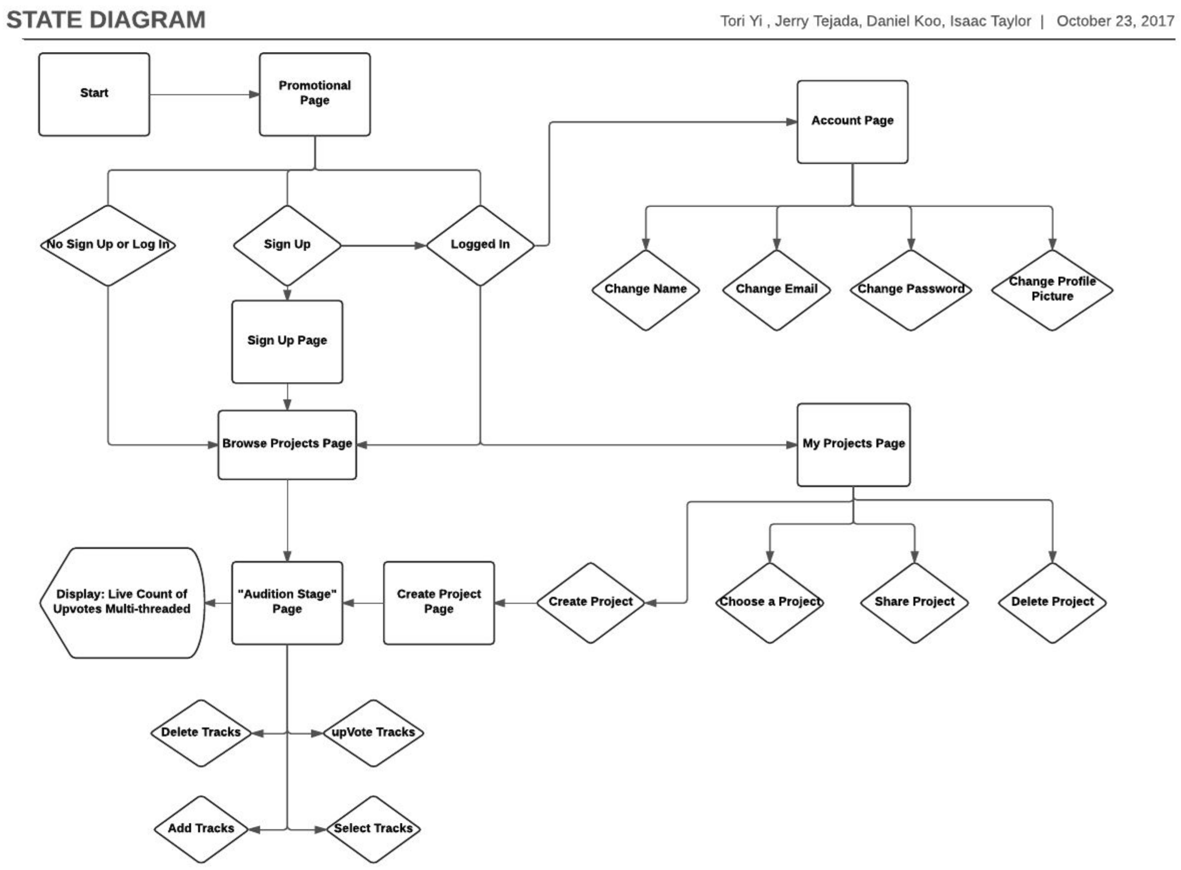
Inspiration References: Spotify, LeetCode

Color Pallette:

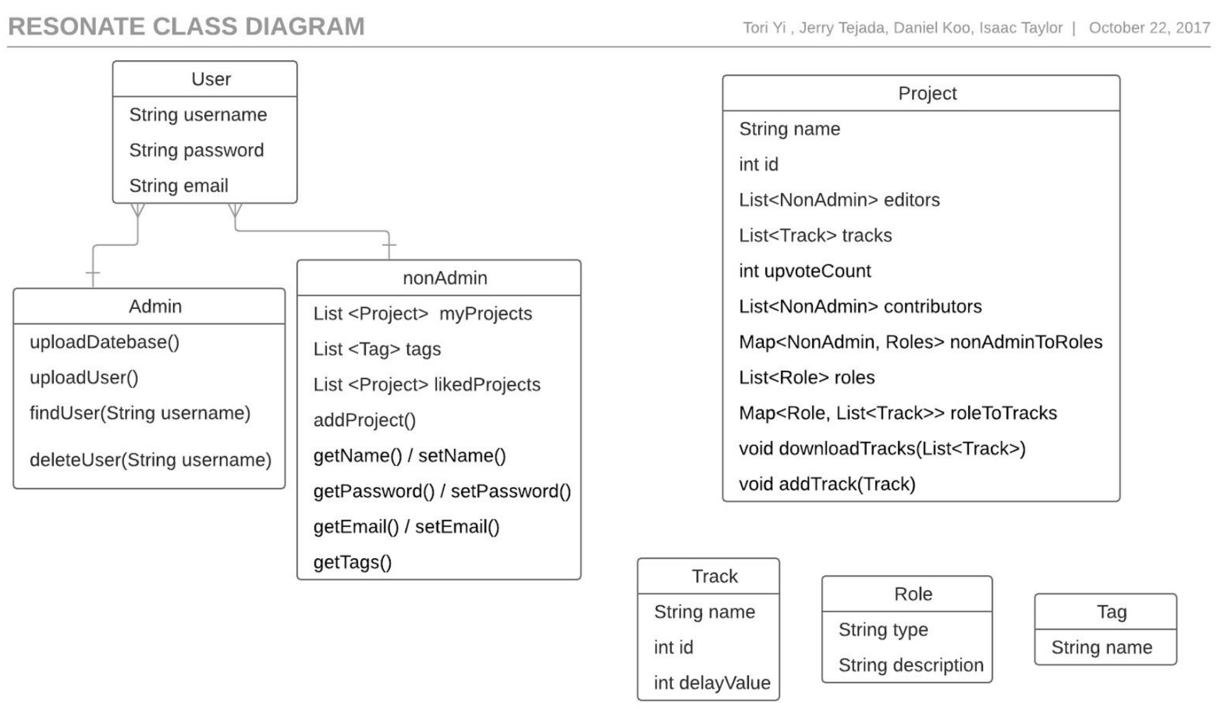


Detailed Design Document

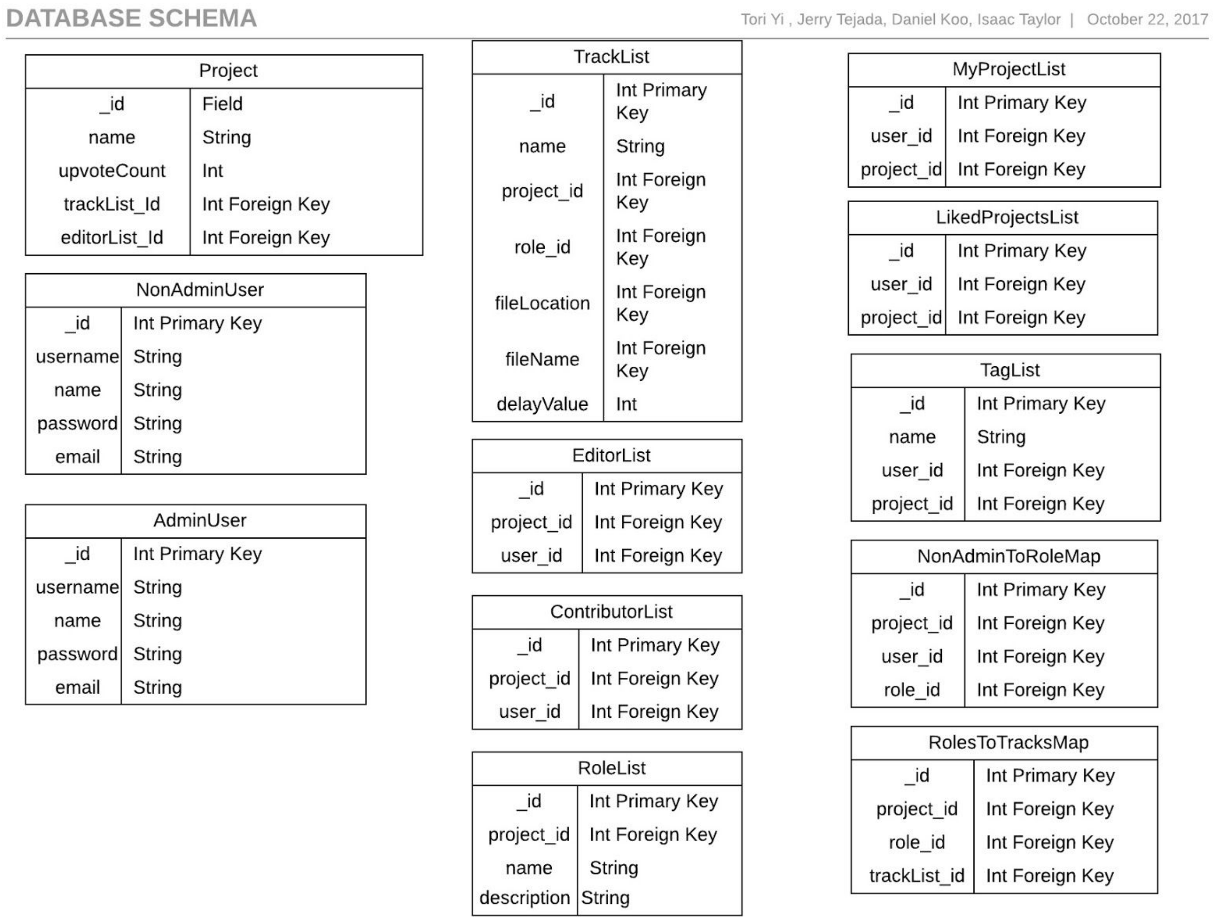
State Diagram:



Class Diagram:



Database Schema:



DATABASE SCHEMA

Tori Yi , Jerry Tejada, Daniel Koo, Isaac Taylor | October 22, 2017

Project	
_id	Field
name	String
upvoteCount	Int
trackList_Id	Int Foreign Key
editorList_Id	Int Foreign Key

NonAdminUser	
_id	Int Primary Key
username	String
name	String
password	String
email	String

AdminUser	
_id	Int Primary Key
username	String
name	String
password	String
email	String

TrackList	
_id	Int Primary Key
name	String
project_id	Int Foreign Key
role_id	Int Foreign Key
fileLocation	Int Foreign Key
fileName	Int Foreign Key
delayValue	Int

EditorList	
_id	Int Primary Key
project_id	Int Foreign Key
user_id	Int Foreign Key

ContributorList	
_id	Int Primary Key
project_id	Int Foreign Key
user_id	Int Foreign Key

RoleList	
_id	Int Primary Key
project_id	Int Foreign Key
name	String
description	String

MyProjectList	
_id	Int Primary Key
user_id	Int Foreign Key
project_id	Int Foreign Key

LikedProjectsList	
_id	Int Primary Key
user_id	Int Foreign Key
project_id	Int Foreign Key

TagList	
_id	Int Primary Key
name	String
user_id	Int Foreign Key
project_id	Int Foreign Key

NonAdminToRoleMap	
_id	Int Primary Key
project_id	Int Foreign Key
user_id	Int Foreign Key
role_id	Int Foreign Key

RolesToTracksMap	
_id	Int Primary Key
project_id	Int Foreign Key
role_id	Int Foreign Key
trackList_id	Int Foreign Key

Method/Variable Descriptions:

- User Class
    - String username, String password, String email
        - The username, password, and email Strings will store the username, password, and email that the user entered when first signing up for the site
- Admin Class (Inherits from User)
    - Methods
        - uploadDatabase() : Reads the database information from a file and uploads the corresponding content onto the screen
        - uploadUser() : Reads a new user's information from the signup page and adds them to the database

        - findUser(String username) : searches for a user in the database using their username and returns the corresponding User object
        - deleteUser(String username) : deletes a user from the database
- NonAdmin Class (inherits from User)
    - Variables
        - List<Project> myProjects : A list that holds all the Project objects that a user has created or is currently collaborating in
        - List<Tag> tags : A list of Tag objects associated with the user (Tags described below).
        - List<Project> likedProjects : A list of Project objects that the user has "liked"
    - Methods
        - addProject() : This will add a project to either the user's myProjects list or likedProjects list
        - getName(), getPassword(), getEmail(), getTags()
            - Getter functions that return the user's corresponding String or List variable
- Project Class
    - Variables
        - String name, int id: The name of the project and an id linked to the project used for searching and accessing
        - List<NonAdmin> editors : A list that holds all the people who can change the properties of the project
        - List<Track> tracks : A list that holds all the tracks related to a project
        - Int upvoteCount : A variable that holds how many "likes" a project gets
        - List<NonAdmin> contributors: A list that holds all the users (minus the admin) who are participating/contributing to a project
        - List<Role> roles : A list of Role objects (Role object described below)
        - Map<NonAdmin, Role> nonAdminToRoles : A map that links a NonAdmin to their respective role in the project
        - Map<Role, List<Track> > roleToTracks : A map that links the user's role to a list of the tracks they are using in the project
    - Methods

        - downloadTracks(List<Track> tracks) : Downloads each track that is being used in the project
        - addTrack(Track t) : Adds a Track object to the database so that it can be used/edited by the user
- Track object
    - String name : String that holds the name of the track
    - int Id : int value that associates an ID number with the track used for searching and accessing
    - int delayValue : Value in milliseconds to fix alignment issues with different music tracks. This is to prevent out of sync music playback when using cross-platform browsers and etc.
- Role object
    - A Role is the respective “role” that a user has in the project. Types of roles are producer, songwriter, singer, pianist, guitarist, etc. (these are stored in the String type variable)
    - String description : The way that a particular user chooses to describe the project, given that different users are allowed describe the project in their own words
- Tag object
    - Tag objects will be used to link certain words to a user’s account in order to suggest content we think the user will enjoy based on previous searches and their own past projects

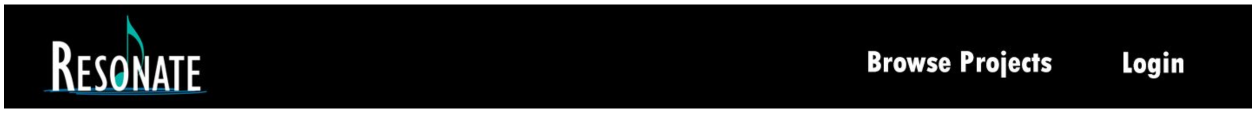
Hardware Requirements: To run this web application the user will need a computer with internet access.

Software Requirements: A web browser (Including but not limited to Google Chrome, Safari, Mozilla Firefox)

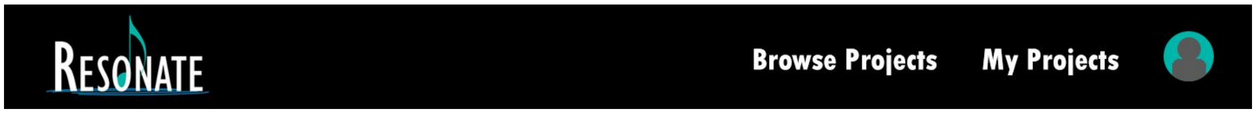
Graphic User Interface:

0) Top Navigation Bar

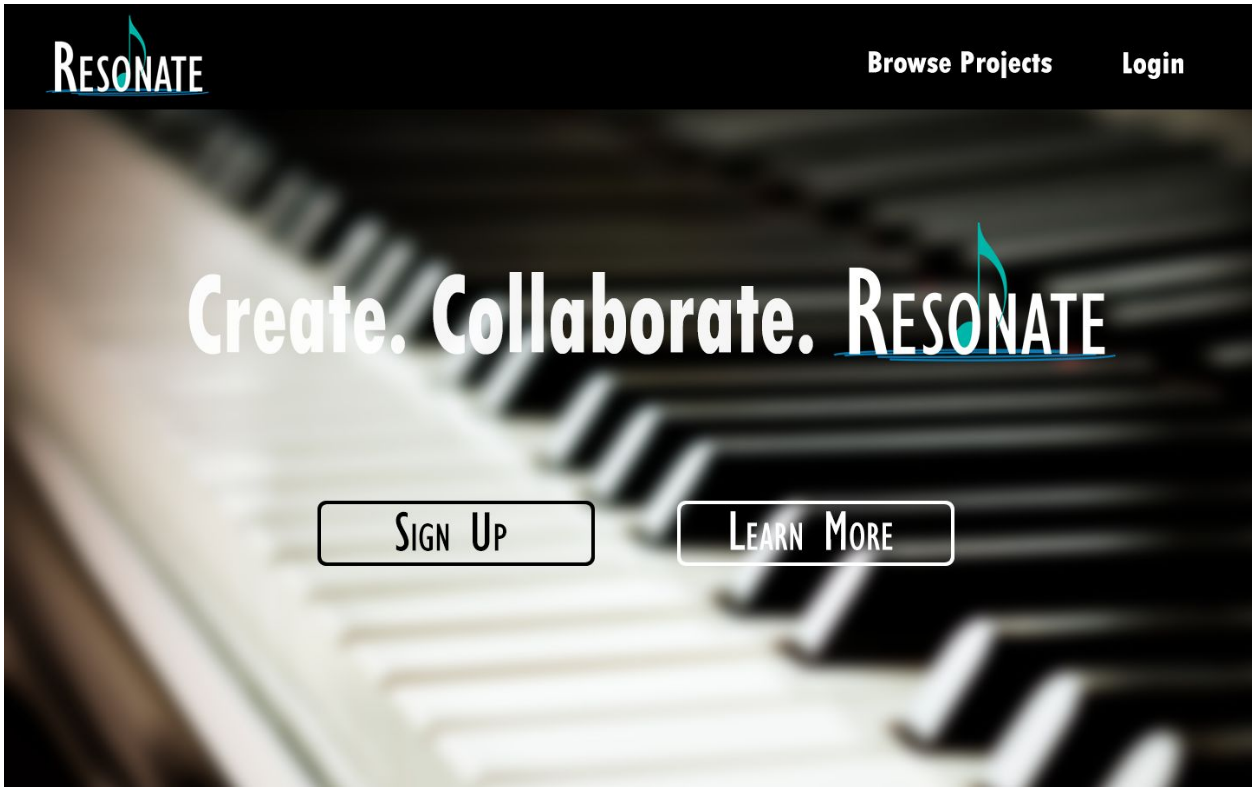
a) Not logged in:



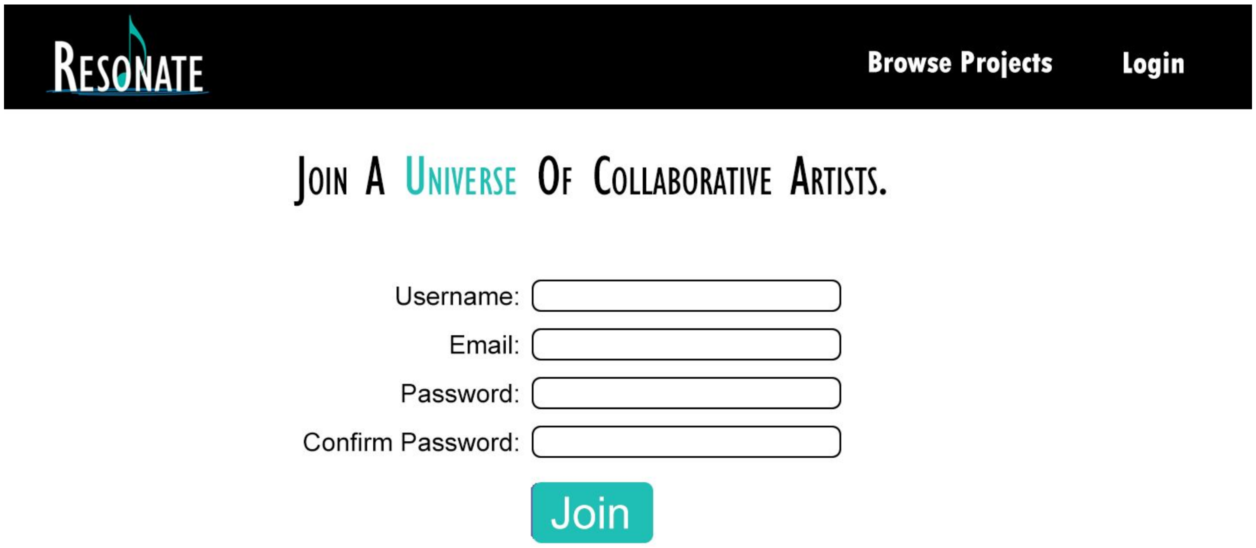
b) Logged in:



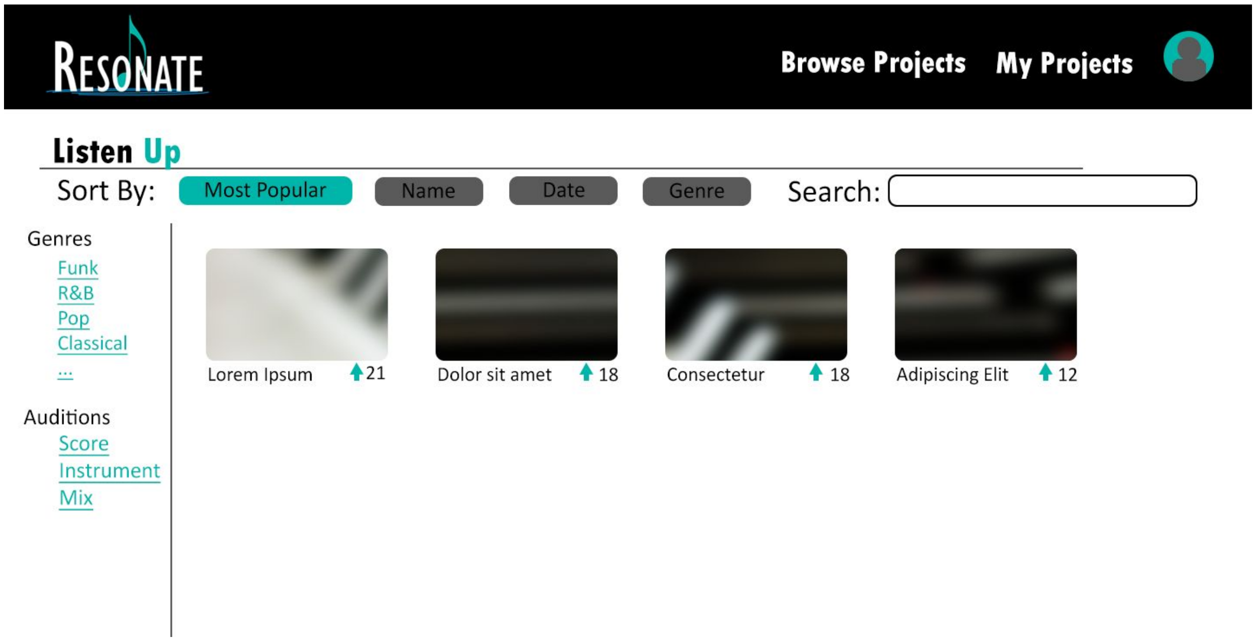
1) Promotional Page



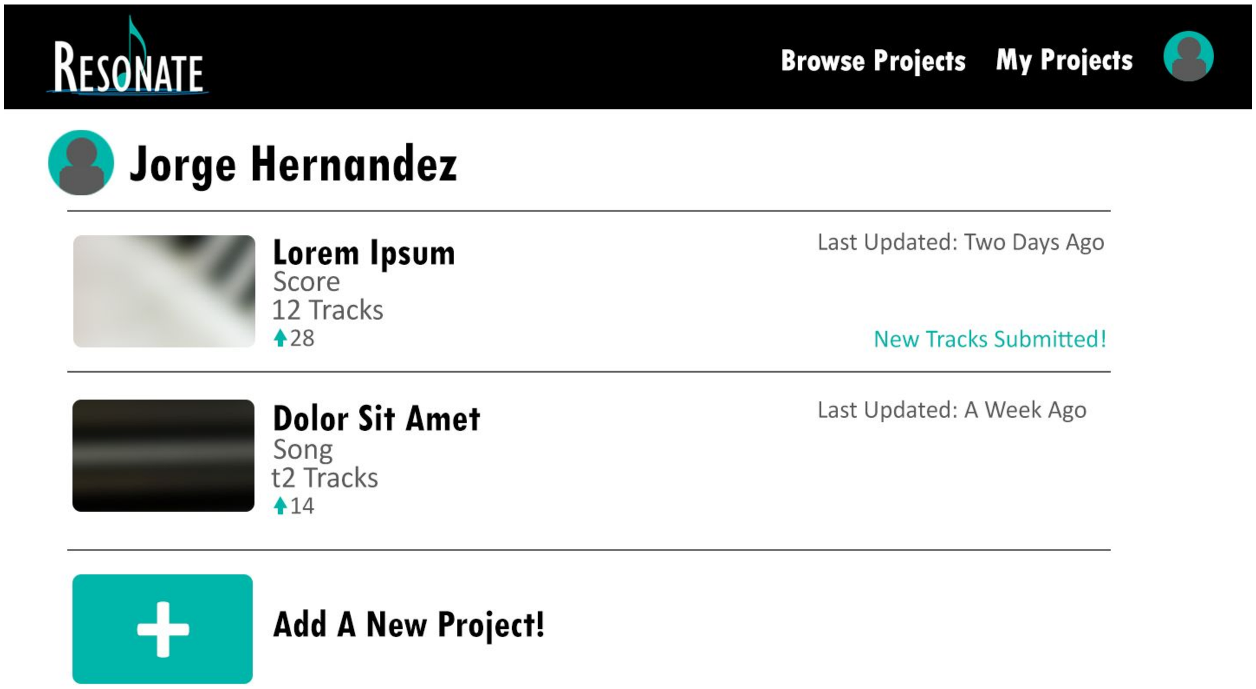
2) Sign up Page



3) Browse projects page



4) My projects page


RESONATE

Browse Projects My Projects

Jorge Hernandez

Lorem Ipsum
Score
12 Tracks
28

Last Updated: Two Days Ago

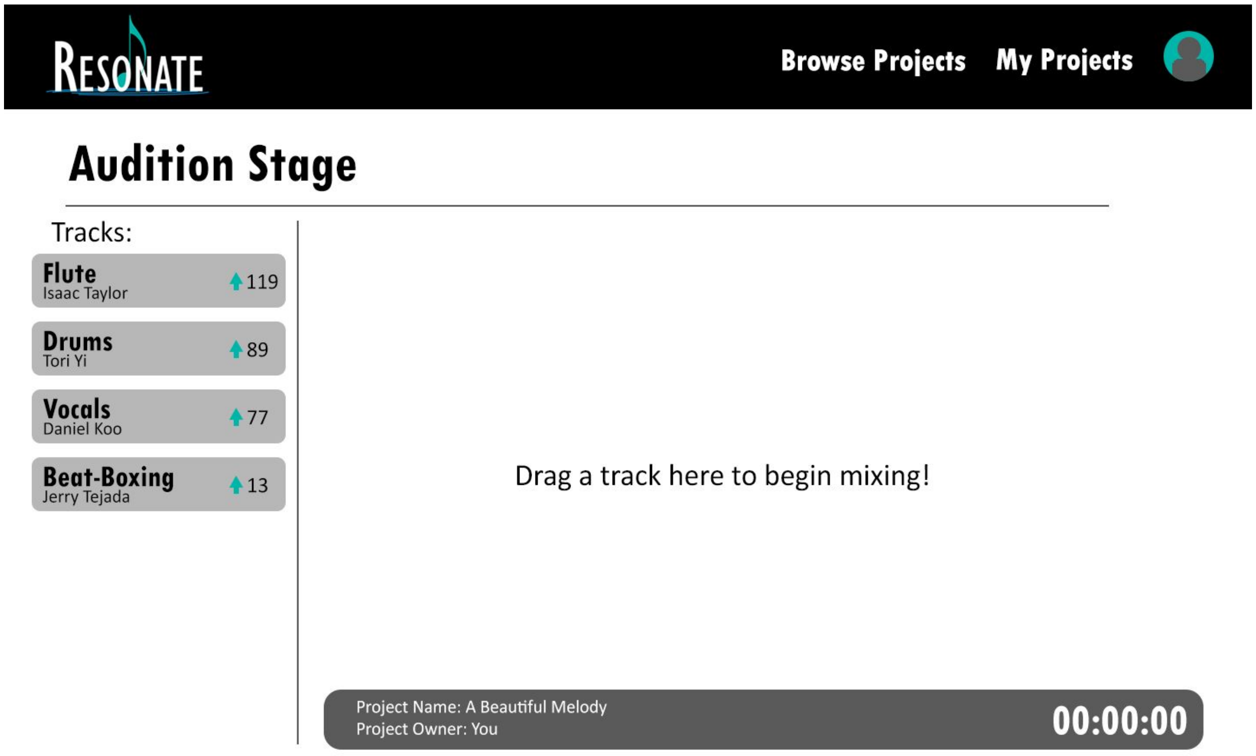
New Tracks Submitted!

Dolor Sit Amet
Song
t2 Tracks
14

Last Updated: A Week Ago

Add A New Project!

5) “Audition Stage” page


RESONATE

Browse Projects My Projects

Audition Stage

Tracks:

Flute
Isaac Taylor
119

Drums
Tori Yi
89

Vocals
Daniel Koo
77

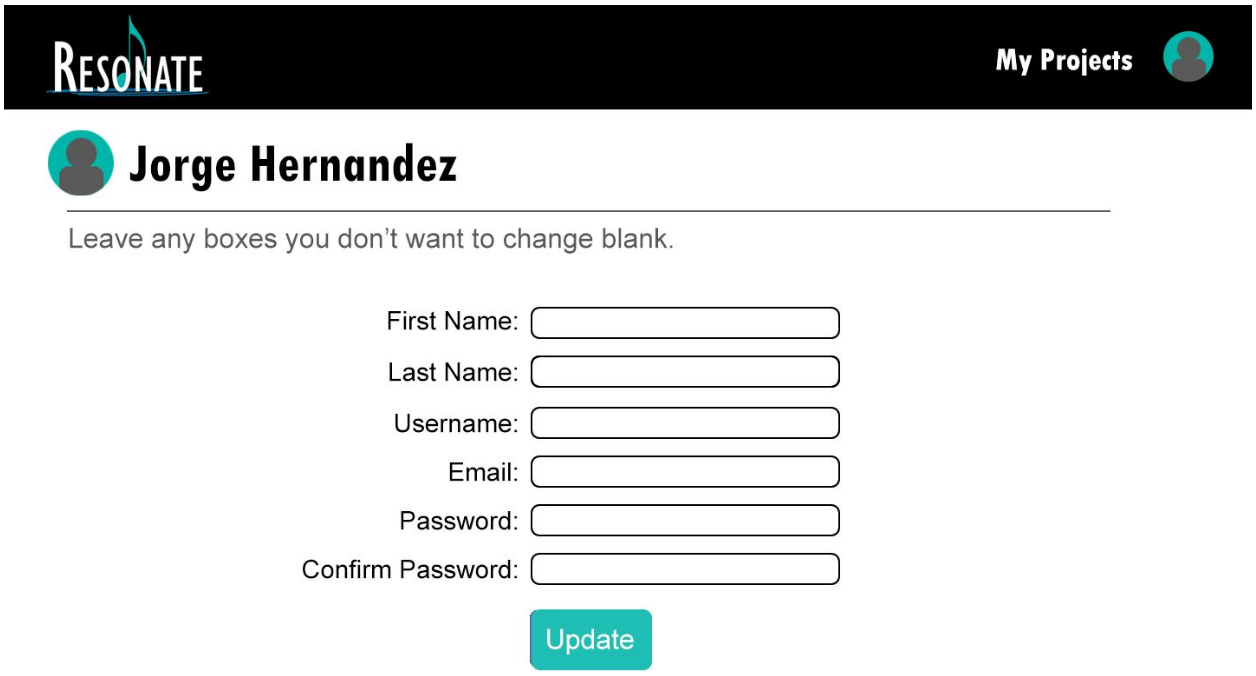
Beat-Boxing
Jerry Tejada
13

Drag a track here to begin mixing!

Project Name: A Beautiful Melody
Project Owner: You

00:00:00

6) Account Page



Jorge Hernandez

Leave any boxes you don’t want to change blank.

First Name:	
Last Name:	
Username:	
Email:	
Password:	
Confirm Password:	

Testing Document

Top Navigation Bar

White Box Tests:

Test Case 1: After login has completed, the username and password strings are passed through a findUser function.

- Success if user is returned
- Success if user is not found in database and user is notified of it.

Black Box Tests:

Test Case 2: Clicking on the logo

- When clicked, user should be redirected to the main page

Test Case 3: Clicking on the “Browse Projects” Button

- When clicked, user should be redirected to the “Browse Projects” page

Test Case 4: Clicking login button

- When clicked, a window should appear in the top right corner with fields to enter username criteria

Test Case 5: Login functionality - User enters a username that does not exist

- Successful if user is redirected to login page, invalid credentials message appears

Test Case 6: Login functionality - User enters a valid username but password that does not match

- Successful if user is redirected to login page, invalid credentials message appears

Test Case 7: Login functionality - User enters a valid username and valid password

- Successful if user is redirected to the user’s my accounts page

Test Case 8: Login functionality - When logged in, clicking on the “My Projects” button

- Successful if user is redirected to the “My Projects” page

Test Case 9: Login functionality - When logged in, mouse over the Account picture.

- When clicked, window appears with buttons to redirect to “My Account” page or to “Log out”

Test Case 10: Login functionality - When logged in, click the Account picture.

- When clicked, successful if it redirects the user to their “My Account” page.

Stress Tests:

Test Case 11: Input into the login box: Username: “Bob”, Password: “bob; DROP * from NonAdminUsers;”

- Only successful if your accounts still exist afterwards

Promotional Page

Black Box Tests:

Test Case 12: Page loads successfully

- Successful if the page loads with all of the correct information

Sign-up Page

White Box Tests:

Test Case 13: After signup has completed, the username, email, and password strings are passed through a createUser function.

- Success if user is created
- Failure if user is not added in database

Black Box Tests:

Test Case 14: Inputting username

- Successfully checks whether or not the username already exists

Test Case 15: Inputting email

- Checks whether or not user has entered a valid email address
  - Searches for “@” symbol
- Also checks whether or not that email already exists in the database

Test Case 16: Inputting password and confirm password

- Successfully checks if the password in both fields are equal
- Error message appears on screen if Strings don’t match

Test Case 17: Clicking the Sign up button

- Should redirect to the sign-up page

Stress Tests:

Test Case 18:

- Input a large number of characters into a text box
  - Should not be able to sign up with the inputted information

Test Case 19:

- Use characters that do not have a corresponding ASCII value
  - Should not be able to sign up

Browse Projects Page

Black Box Tests:

Test Case 20: Click a sort option on the top row after “Sort By:”

- Only successful if the projects listed reorder according to the sort button pressed, eg: “Most Popular” should put the top voted projects at the top.

Test Case 21: Click a filter on the left side

- Only successful if projects disappear that should not be a part of the filtered list, eg: “Pop” Genre clicked only shows Projects in the Pop genre

Test Case 22: Scroll to the end of the page.

- Only successful if more projects appear or reached EOL.

White Box Tests:

Test Case 23: On page load, a Socket is opened to keep track of new projects and votes

- Only successful if socket loads and votes etc update in realtime (via jquery).

Test Case 24: Click a project

- Only successful if it redirects you to the project’s page.

Test Case 25: Open a project in a new tab, vote on it.

- Only successful if the vote count of the project goes up on this page, and, if it raises it above another project, resorts when in “Most popular” sort mode.

Stress Tests:

Test Case 26: Clicking quickly back and forth between multiple sort and filter buttons.

- Only successful if the boxes don’t begin flying around the page or stop moving

Test Case 27: Simultaneously adding new projects and voting on existing projects (scripted)

- Only successful if all the new projects appear and are sorted properly in realtime.

My Projects Page

White Box Tests:

Test Case 29: When the page loads, the user is pass to the getProjects function

- Successful if the correct projects are passed back in the form of a List.

Black Box Tests:

Test Case 30: Reaching the page

- Successful if the correct projects load.

Test Case 31: Clicking on one of the projects

- Successful if the “Audition Stage” page is loaded with the correct project.

“Audition Stage” Page

White Box Tests:

Test Case 32: When the page loads, the project is passed through a findProject function

- Successful if the correct project is passed back and the tracks are sorted by "Top Voted"

Test Case 33: Clicking on the "Vote" button for each track
- Successful if the vote is processed into the database and the change is reflected on the tracks

Test Case 34: Clicking on "download" button for selected tracks.
- Successful if the page is multithreaded and still is live and music is compiled and downloaded to the client.

Test Case 35: Clicking on "save as my mix" button for selected tracks
- Successful if the mix was added to the list of projects that the user is associated with in the database.

Black Box Tests:

Test Case 36: Reaching the page
- Successful if the correct project load.

Test Case 37: Clicking on the checkboxes
- Successful if the particular tracks are selected.

Test Case 38: Clicking on the "play" button
- Successful if the selected tracks are played at the same time.

Test Case 39: Clicking on the "download" button
- Successful if the selected tracks downloaded in a compiled format

Test Case 40: Clicking on the "vote" button
- Successful if the vote is updated on the track

Test Case 41: Save as my mix button
- Successful if the mix is saved as a project in my projects page

Account Page

White Box Tests:

Test Case 42: When the page loads, the project is passed through a updateInformation function
- Successful if the correct updated information is returned

Test Case 43: When new profile picture is uploaded
- Successful if the image is formatted and the file is saved in the database

Test Case 44: When new password is added
- Successful if the password is saved to the database.

Test Case 45: When new username is added
- Successful if the username is saved to the database

Test Case 46: When new email is added
- Successful if the email is saved to the database.

Black Box Tests:

Test Case 47: Information display

- Should display all the information submitted by the user in the correct fields (ex: Username: toriyi, Email: toriyi@usc.edu, Password: HelloWorld)

Test Case 48: Clicking on edit profile button

- Successful if able to input modified information into corresponding text box and the modified information is saved and displayed on the page

Stress Tests:

Test Case 49:

- Should not be able to input characters that do not have corresponding ASCII characters when modifying information

Test Case 50:

- Should not be able to input too many characters into a text box