HouseMusic

CMPT 276 Group Project

By: Chris Horan, Shresth Kapila, Shiyi Chen, Derrick Cham, Harry Preet Singh

https://github.com/Chris-Horan/HOUSEMUSIC

Color Codes: 0th Iteration, 1st Iteration, 2nd Iteration ,3rd Iteration

Documentation Compiled by - Harry Preet Singh

House Music
Requirements and Specification Document
07/09/2020 , Version 2

Abstract

HouseMusic is a webapp that allows users to record and loop their own recordings to make a short music sample. Our application will be lightweight and portable to allow for quick and easy editing, but is free for all users. Although the main scope of the project is to create music loops, some other features include real-time collaboration, saving your project, adding clips from your device, recording your own voice, and exporting as an mp3. Our app also features user login support, which allows the users to save and load their music projects.

Customer

The target audience will be those who are interested in creating their own music but don't have experience working with recording software. This app is a mix of both entertainment and education.While anybody of any skill level can log in and create music in seconds, more experienced music creators can also use this app to create music loops to use in projects. With this app multiple users can quickly make a short sample of their idea before committing to a full project if they so desire to.

Competitive Analysis

Although other similar applications such as exist, they do not allow for real-time collaboration between users, while ours does. Examples:
Splice Beatmaker (https://splice.com/sounds/beatmaker)

Splice Features :

- Easily search and download millions of loops, one-shots, and presets.Drag and drop them into your DAW.
- Exclusive artist packs

- 100% royalty free
- Individual samples

We believe that the scope of this project is appropriate, as it incorporates multiple APIs, includes socket.io real-time functionality, implements a browser based login system, and has at least as many epics as people in our group. We also believe that the amount of work required in this proposal is enough for five group members. We intend to use **Howler.js API** to implement our basic audio features (https://howlerjs.com/).

User Stories

1. **User Sign Up:**
   Actors: User

   Triggers/Preconditions: A user has requested the signup page for the app and comes across the signup interface

   Action/Postconditions : The code will take a username and password from the user and send it to our database. If the username is different from every other username then, it will return the home page of the new user. It will also have a confirmation field for password that will confirm the given password.

   Acceptance test :
   - **a.** Attempt to create a unique user. Check that the new user is added to the user database
     - i. **Test** - If the database already has username named Chris, if we try to add Chris again, the result will be "Username already exist"
   - **b.** Attempt to create a redundant user. Check that the entry is rejected and that the user is informed.
   - **c.** Attempt to create a user without satisfying requirements. Check that the entry is rejected and that the user is informed.
     - i. **Test** - Username of length(4,20), so if you try to test username like "cmp" **Results** : "Username should be of length 4 to 20".

   Iteration: 1

2. **Admin/User Login:**
   Actors: Admin or User
   2
   Triggers/Preconditions): An admin or existing user would like to log in to the

application.

Action/Postconditions: The existing user or admin enters their credentials correctly and submits the login form. They are then taken to the homepage for their profile.

Acceptance test :

a. Attempt login with accurate credentials. Check that the login is accepted.
   i. **Test** - Search the database if the username exists.
   ii. **Result** - If a user exists, direct the user to the main app or if the admin exists, direct the admin to the admin panel.

b. Attempt login with inaccurate credentials. Check that the login is not accepted and inform the user.
   i. **Test** - Search the database if the username exists.
   ii. **Result** - if not it will not give the access to either of the pages.

Iteration: 1

3. **Admin Adds New Admin:**

Actors: Admin

Triggers/Preconditions): An admin would like to add a new user with admin capabilities using the admin panel.

Action/Postconditions: The existing admin enters the username, email, and password of the new admin. The new admin is then added to the user database as user type admin.

Acceptance test :

a. Attempt to create a unique admin. Check that the new admin is added to the user database.

i. Test value: username: 'abcdef' (Username is not taken by any other user)
Email: 'abcdef@test.com' (Email is not taken by any other user)
Password: 'password'
Result: user 'abcdef' added to database as user type admin and the existing admin is notified.

b. Attempt to create an admin with a previously used username. Check that the entry is rejected and that the existing admin is informed.

i. **Test value**: **username**: ‘abcdef’ (Username is taken by another user)

Email: ‘abcdefgh@test.com’ (Email is not taken by any other user)

Password: ‘password’

Result: user ‘abcdef’ is not added to the database and the admin is notified that a user with this username already exists.

c. Attempt to create an admin without satisfying requirements. Check that the entry is rejected and that the existing admin is informed.

I. Test values: username: ‘??abcdef’ (Username is not taken by any other user)
Email: ‘abcdef@test.com’ (Email is not taken by any other user)
Password: ‘password’
Result: user ‘abcdef’ is not added to the database and the admin is notified that the username does not meet the requirements.

Iteration: 1

4. **Admin Removes User or Admin:**
Actors: Admin

Triggers/Preconditions): An admin would like to remove a user or admin.

Action/Postconditions: The admin enters the username of the user or admin that they would like to remove. The user or admin in question is then deleted from the database.

Acceptance test :

a. Attempt to remove a valid user/admin. Check that the removal is completed.
   i. **Test Value: username = ‘removable’** (Assumed that this a valid username that hasn’t been deleted yet)
   ii. This user will be deleted from the database, which can be verified by the ‘search’ feature.

b. Attempt to remove a non-existent user/admin. Check that the removal was not possible and inform the admin with an error message.
   i. **Test Value: username = ‘fakeusername’** (Assumed that this an invalid username that does not exist)
   ii. The admin will be notified with an error message saying “User does not Exist” and the database is not modified.

c. Attempt to remove a user/admin without providing any information at all (ie. leaving the textbox blank). Check that the removal was not possible and give an error message to the admin.
   i. **Test Value: username = "**
   ii. The admin will be notified with an error message saying "User does not Exist" and the database is not modified.

Iteration: 1

5. **Admin Change User Password:**
   Actors: Admin

   Triggers/Preconditions): An admin would like to change a particular user.

   Action/Postconditions: The admin enters the username of the user that they want to change the password. The user password is then changed from the database.
   Acceptance test :
   a. Attempt to enter the invalid user. Check that the user does not exist and inform the admin.
      i. **Test value** -**Username**: password12
         Password: changePassword
         The **result** will be "**User does not exist**", the database is not changed.
   b. Attempt to enter the valid user. Check that the user exists and inform the admin the password is changed for the corresponding user.
      i. **Test value -** **Username**: chris
         Password: password098765
         The **result** will be "**Password changed successfully**", and the database is updated.

   Iteration: 1

6. **Admin Search for User:**
   Actors: Admin

   Triggers/Preconditions): An admin would like to search a particular user by entering the username.

   Action/Postconditions: The admin enters the username of the user that they want to search. The email and password of the user will be shown in the search session of the admin panel.

Acceptance test :

a. Attempt to enter the invalid user. Search for that user and return “no results” in the search session.

   i. **Test Value:**

   -**username: ccdef123** (This is a invalid username.)
   -The searching result will show the text “No results”.

b. Attempt to enter the valid user. Search for that user and return all information(username, email and password) in a table format in the search session.

   i. **Test Value**:

   - **username: chris567** (This is a valid username.)
   -The searching result will return the username, email and password of this particular user .

Iteration: 1

Future User Stories

1. **User or Admin Forgets Login Information and Needs to Change Password**

2. **User Creates a Music Track:**
   Actors: User

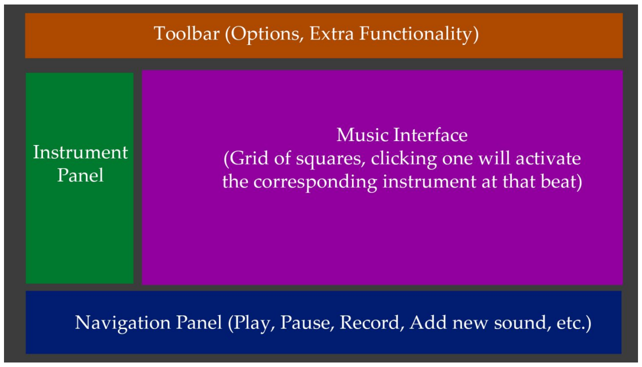
3. **User Records a Music Track:**
   Actors: User

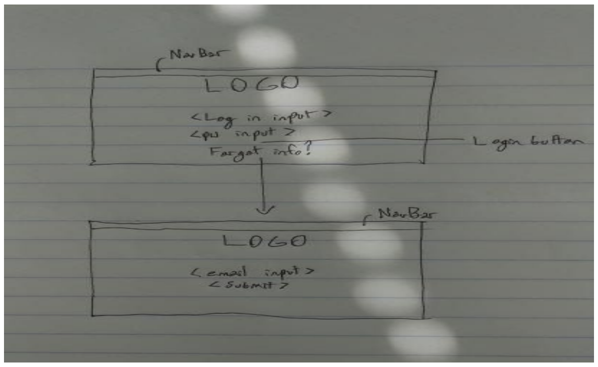
4. **User Saves Project:**
   Actors: User

5. **User Deletes Project:**
   Actors: User

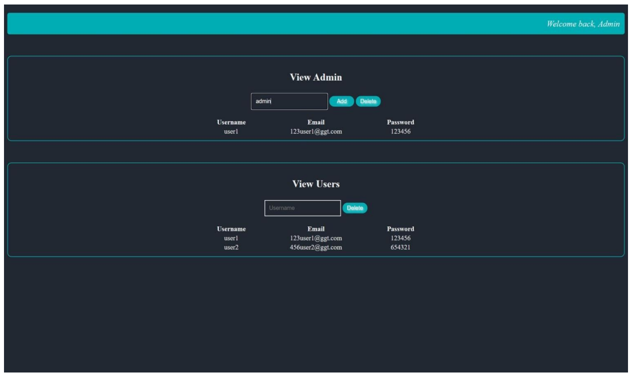
User Interface Requirements

<u>Initial UI Mockups</u>


Toolbar (Options, Extra Functionality)
Instrument Panel
Music Interface
(Grid of squares, clicking one will activate the corresponding instrument at that beat)
Navigation Panel (Play, Pause, Record, Add new sound, etc.)



<u>First and Second Drafts</u>

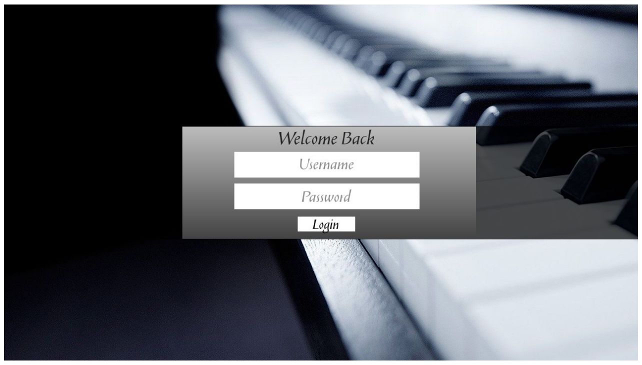


View Admin

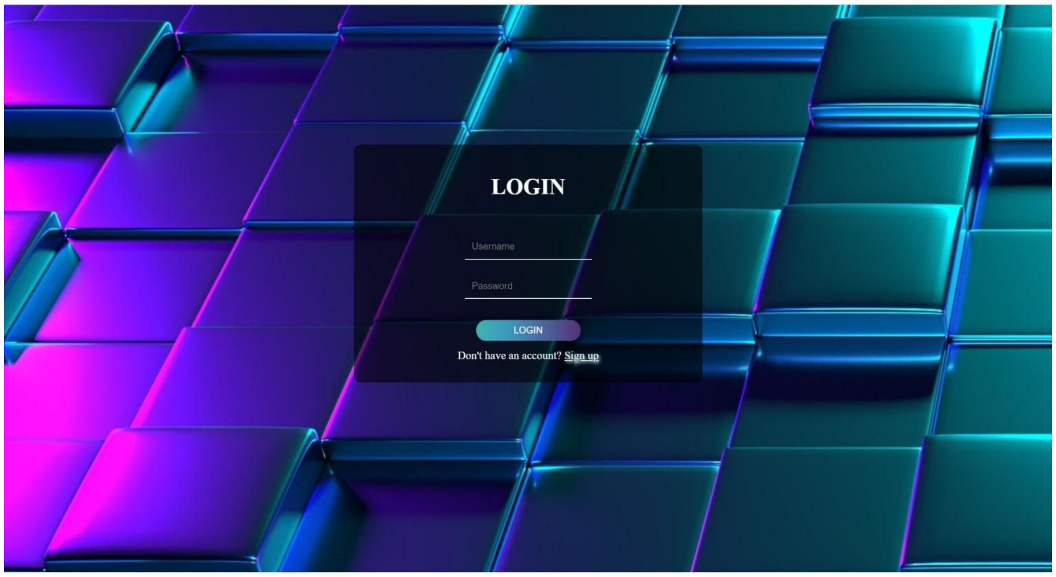
Username	Email	Password
user1	123user1@ggt.com	123456

View Users

Username	Email	Password
user1	123user1@ggt.com	123456
user2	456user2@ggt.com	654321



<u>Final</u>


LOGIN
Username
Password
LOGIN
Don't have an account? Sign up