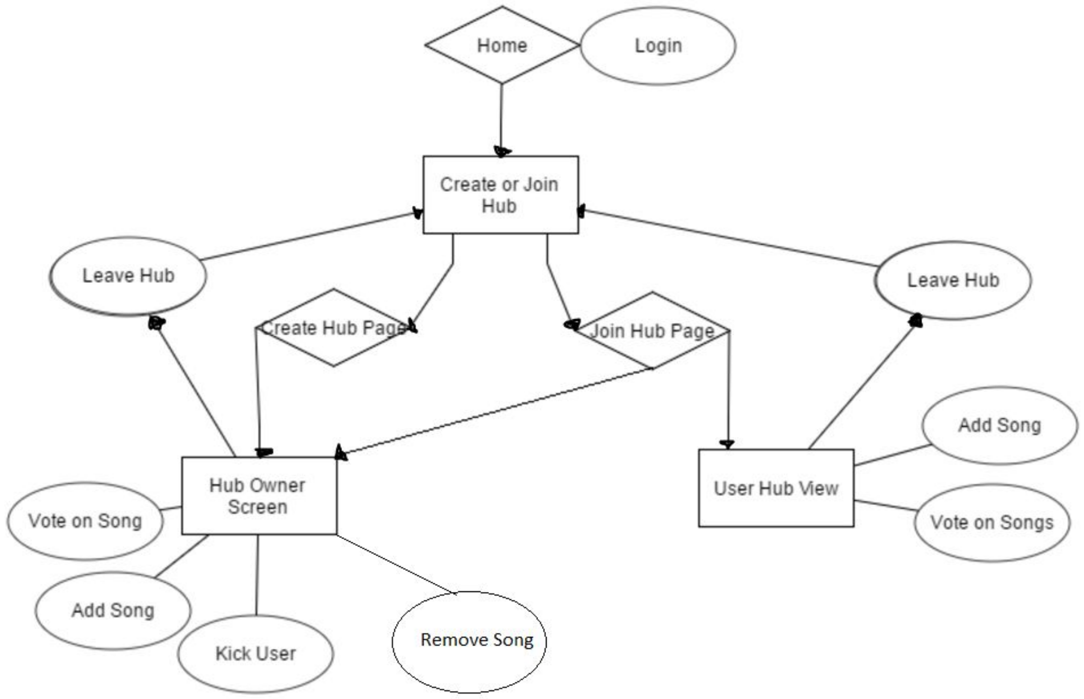
Team 10: MusiQueue-web

Incremental and Regression Testing Sprint 2

Classification of Components

Component Definitions



A user first starts off on the homepage of the app.
Then, the user has to login using his/her Google account.
Next, the user can choose to create or join a hub.
If the user creates a hub, they start off with an empty playlist to which they can add songs as well as upvote and downvote songs.
A hub owner can also remove songs from the playlist and kick users from their hub.
If the user chooses to join a hub, they can search for an existing hub by name and go to the user hub view.
On the user hub view, a user can add songs and vote on songs.
A user can also leave the hub that they are on to return to the create/join hub page.

Form of Incremental Testing

We used bottom-up testing for our incremental testing. We did this by creating unit tests for each component to see if they functioned properly isolated from everything else. Then, we added components together as they passed all of their isolation tests and tested the new system.

Incremental Testing Log

Product	**Home, Login/Logout**		
Date	10/19/2017		
Author	Vincent Maggioli		
Defect #	**Description**	**Severity**	**How Corrected**
1	Theme did not match our original color format once implemented	1	Rather than use a standard theme, create a custom theme and import it into the project to match the required light blue layout
2	Angular theme not showing up on site after importing	1	Themes only work on Angular Material components, they don't just apply to the standard html elements. Assign colors to Material elements in HTML and the theme works fine
Product	**Hub View**		
Date	10/21/2017		
Author	Sairam Galla, Brian Hanford		
Defect #	**Description**	**Severity**	**How Corrected**
1	A user should only be able to upvote or downvote a song once.	1	Keep track of each user's vote for each song on the database after the user votes for that song. If the user has already upvoted or downvoted a song, clicking that button again does nothing.

2	The user should be able to change their vote to nothing even after they upvote or downvote a song.	1	Change the "vote" value for the song to "null" if the upvote or downvote button is pressed after it has already been pressed before. And bring the overall rank of the song down by one.
3	The user cannot see whether they have upvoted or downvoted a song.	1	Change the color of the upvote or downvote button to blue when the song has been upvoted or downvoted respectively.

Product	**Home, Login/Logout**		
Date	10/21/2017		
Author	Samuel Kuhns		
Defect #	**Description**	**Severity**	**How Corrected**
1	Theme not correctly loading for my branch and taints input	2	Find a new way to install the inputs
2	ID still not correctly put into text box when revisiting page	1	Fix call to get currentUser.username into text box

Regression Testing Log

Product	**Hub View**		
Date	10/21/2017		
Author	Sairam Galla, Brian Hanford, Neil Amin		
Defect #	**Description**	**Severity**	**How Corrected**

1	After implementing allowing the user to only upvote or downvote for each song once, the playlist was not reordering.	2	Changing the vote on the backend was done outside the "once" database reference. The database references are executed on a different thread, so this results in the vote being updated before the rank of the song gets updated and the playlist never reorders. Moving the vote update inside the "once" function fixes this.
2	The color of the upvote or downvote button never changes even after upvoting or downvoting.	1	We need to return true or false based upon the "vote" value in the database. However, it is not possible to return values from database references even by using callback functions or Promises because the database references are executed on different threads and the value is returned before the database thread is executed. We tried changing the color to blue by using the toggle buttons provided by Angular.
3	The upvote and downvote buttons didn't initially increase and decrease the rank of each song by one respectively. It was changing the rank at random.	2	We should not subscribe to "vote" value in the database because the data keeps changing and we only want a one-time poll to get the "vote" value here. So changing the subscribe to a "once" database reference fixed this.
4	using firebase listeners for every database call caused all things to rerun code every time there was a change in the database. Objects would be created multiple times and recreated after deletion	3	Change database access methods to not constantly be listening for changes if they were meant to be single database lookup/addition
5	Hub name not properly being displayed on user or main.	1	Set hub name when page is loaded.