Team 9 Final Project Plan

Kevin Bu and Sherry Feng

Vision

To clarify the overall picture of our product: our application will be a crowdsourced song generator, in which users can get randomly generated song recommendations from a pre-seeded database by paying one coin. A user can filter song recommendations on various aspects, including: genre, length, uploads, or popularity ratio (calculated by upvotes/downvotes). The user can add song data into the database, yielding two coins for each song. Thus, this database, besides a pre-seeded collection of 1000 songs, will be entirely crowdsourced. Users can save, upvote, and/or downvote songs, which will then be added to their respective user database. One's playlist of saved songs can then be filtered by genre, length, and alphabetical order.

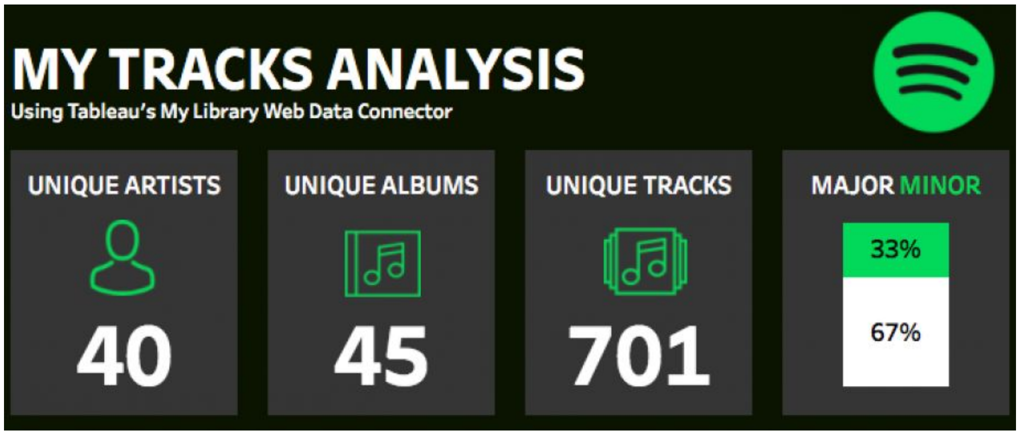
Page Outline

- On all pages: header/footer with app info (includes admin login area)
  - Admin authentication will reveal an additional button to view entire song database
- Page 1
  - Button to proceed as guest (allows generation of random songs, but not filtering or saving songs)
  - Form for account creation
  - Button for login
- Page 2, only opens after authentication
  - If authenticated, option to view profile on top right
    - Changing/resetting password capability
  - View saved music on left side of page
  - Button that allows user to see saved, upvoted, and downvoted songs
  - Button that allows a user to add song data (artist and song name) to earn coins
  - Button that allows a user to get a randomly generated song in accordance to their filtering criteria (costs one coin)
    - Filtering criteria: genre, length, uploads, popularity ratio (upvotes to downvotes)
    - User can choose to save the song and/or upvote/downvote it

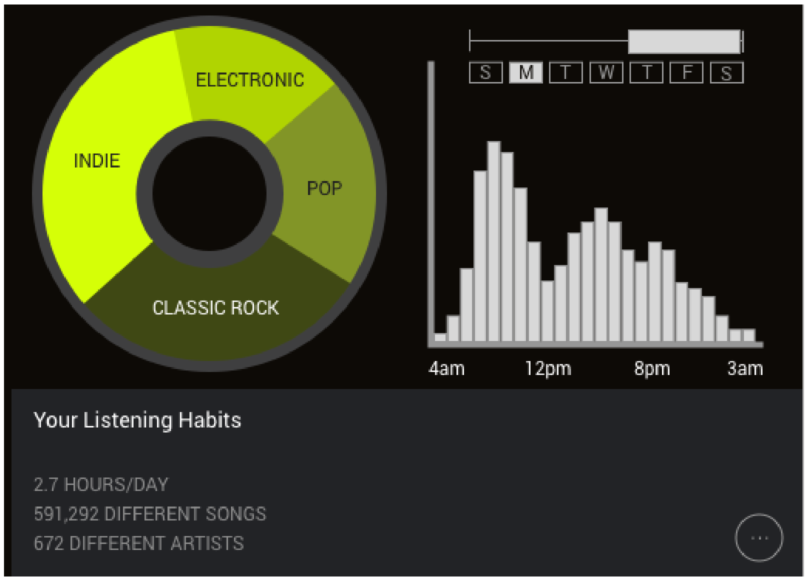
Methods we need to write

- User input
  - Accepting user login data: use input fields in a form, store in Firebase

    - Validating user song input: catch error messages from Spotify's get track method (https://developer.spotify.com/web-api/user-guide/#response-status-codes)
    - Storing user song input: storing select data (artist, song, genre, url) from Spotify's get track method (https://developer.spotify.com/web-api/get-track/) into a JSON object and pushing that into Firebase
    - Tracking upvotes/downvotes for song: tracking upvote/downvote numbers as an attribute to the object in Firebase
- Song Output
    - Fetching song from database: get child from Firebase, given the user filters on genre, length, uploads, or popularity ratio
    - Opening link to song from database: each song will have an URL to its song on Spotify, this can either be displayed on the page as a String or opened in a div/modal
    - Skipping to next song in database: go to next child in Firebase
    - Checking coin limit: check user data for number of coins, ensure it is not 0
    - Checking guest permissions: count number of songs guest has listened to, when it hits 5, redirect guest to a signup page
- Data storage
    - Seeding initial database with similar songs: manual input of data
    - Add to database with Spotify track: Use user input and the Spotify search method (https://developer.spotify.com/web-api/search-item/)
    - Remove song from database after enough downvotes: remove child from Firebase if upvote/downvote ratio exceeds 1/2
- User preferences
    - Display the songs that user has upvoted/downvoted: the songs each user likes or dislikes is stored in user data, we can iterate through these objects to display them on the screen
    - Compare like/dislike ratio to other average users. Requires this to be computed from a global variable that sums up all likes and dislikes from all users.
    - Visualization: sort the songs alphabetically or chronologically or by duration. Requires a filtering function for our Firebase data
    - Example of user visualization: this could be done for the inputs or liked tracks that the user has



- https://www.interworks.com/sites/default/files/styles/featured/public/blog/Screen%20Shot%202017-04-21%20at%202.05.28%20PM_0.png?itok=LUJewGgb



- https://cdn-images-1.medium.com/max/800/1*olhYYmEsc4KfQ_T_y8zjmQ.png

Different User Permissions

- Guest: cannot save any song data, meaning they cannot save or filter songs-- can only generate random songs.
- User: can listen to songs and access them (given coin limits) and filter liked songs
- Admin: can view entire database, coin limits don't apply. A user can only be an admin if they have a special code for it (CS290).