Song Wars - User Guide

Song Wars - Team 17

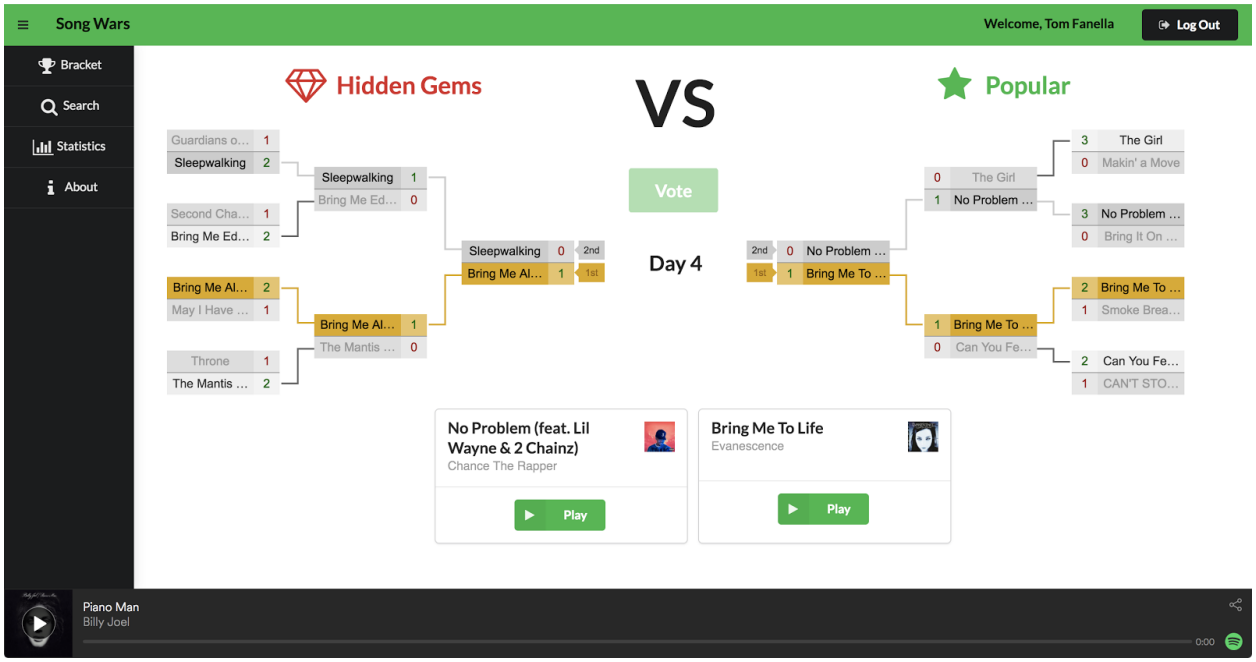
UI Usability:

Required Resources:

- Modern Updated Browser
- Spotify Desktop App
- Website URL
    - http://songwars-frontend-bugged.s3-website-us-west-2.amazonaws.com
- MYSQL Workbench CE
    - user : root
    - password : papaya17
    - hostname : song-wars-mysql.clhr6lfeqd3c.us-west-2.rds.amazonaws.com
    - port : 3306
      “USE song_wars_mysql;”

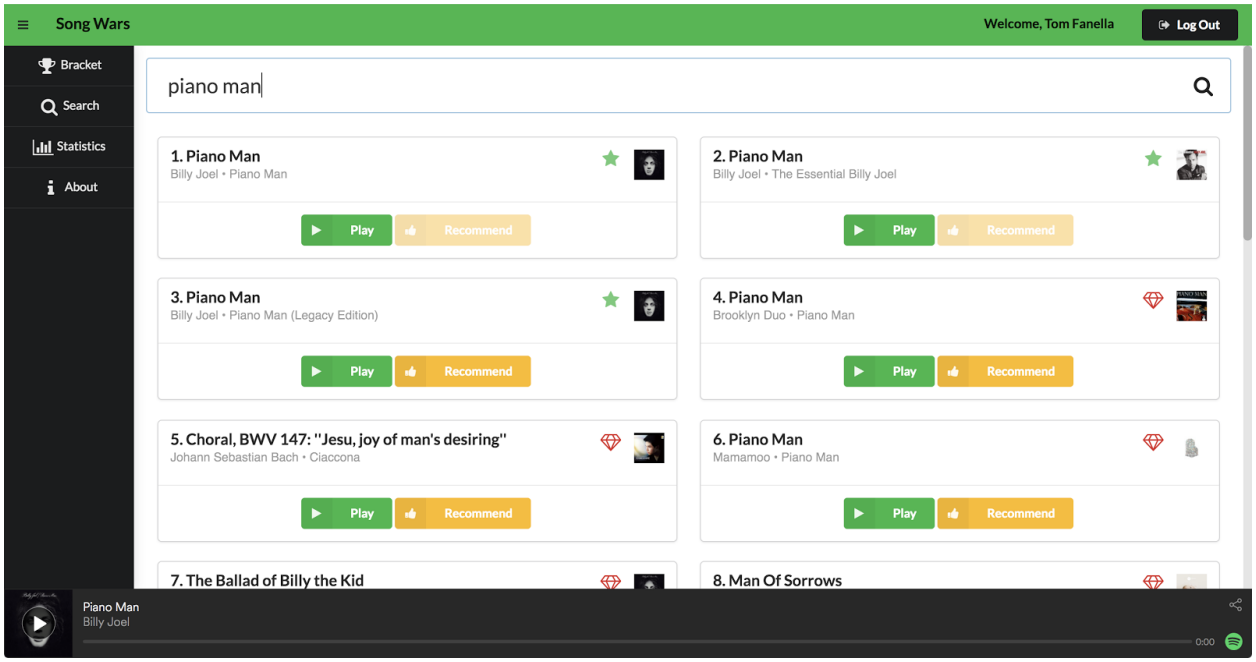
Visual Guide:

/ and /bracket



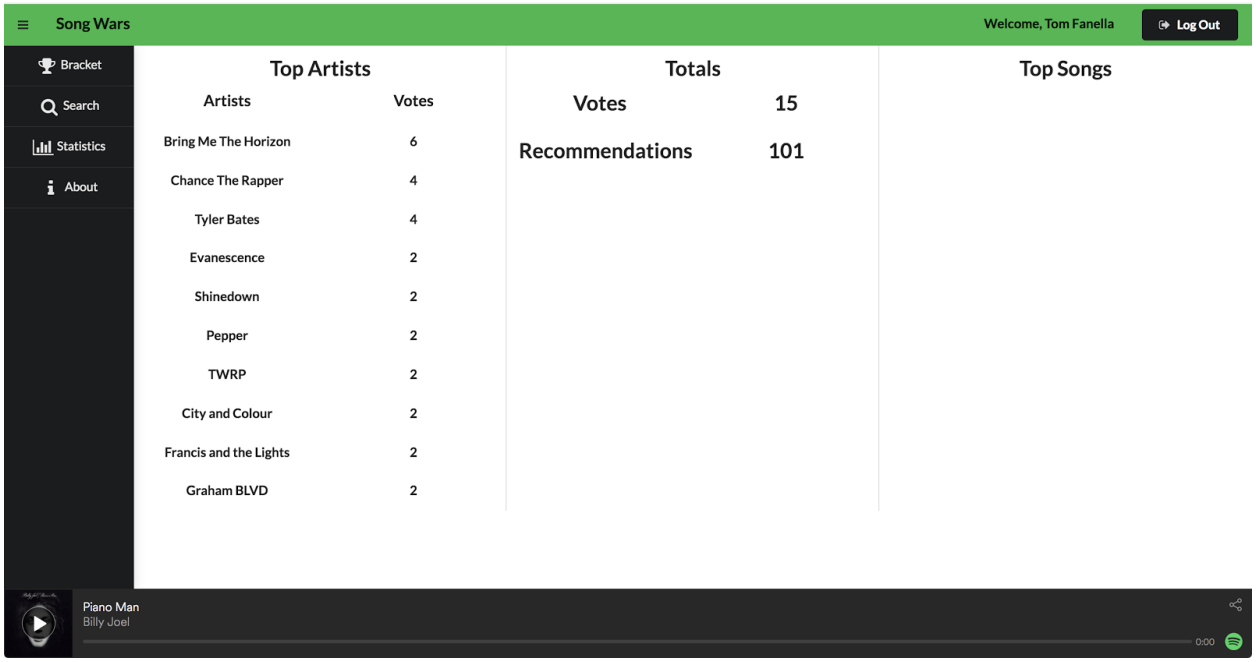
- Each of the elements should be styled like the above screenshot
- The bracket should show the results of the votes the day after all of the vote have been counted
- On the final day the winner of the bracket should be displayed in the center of the bracket
- There should be a preview displayed on the bottom for each of the matches
- The Vote button should launch the Vote view
- The play button places the song in the player widget located at the bottom bar
- The Day counter should start at 1
- As the Day counter progresses, the song elements corresponding to each day should progress through the bracket as well

/search



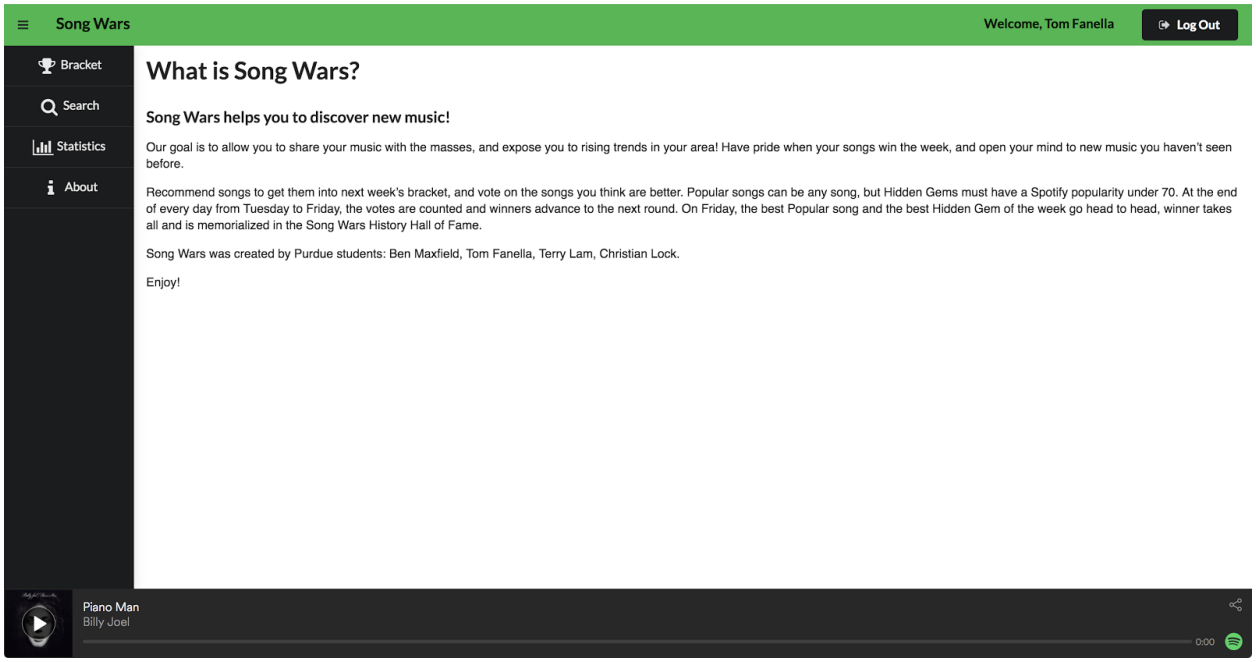
- Typing in the search box should instantly search
- Search results should be in sorted order
- Search results should display the name, album, artist, album art, and popularity classification
- Search results should contain a Play and Recommend button
- The play button places the song in the player widget
- The recommend button recommends the song for next week's bracket
- When no results are found a message should be displayed
- We assume users will recommend enough songs to complete the requirements of filling a full bracket.

/statistics



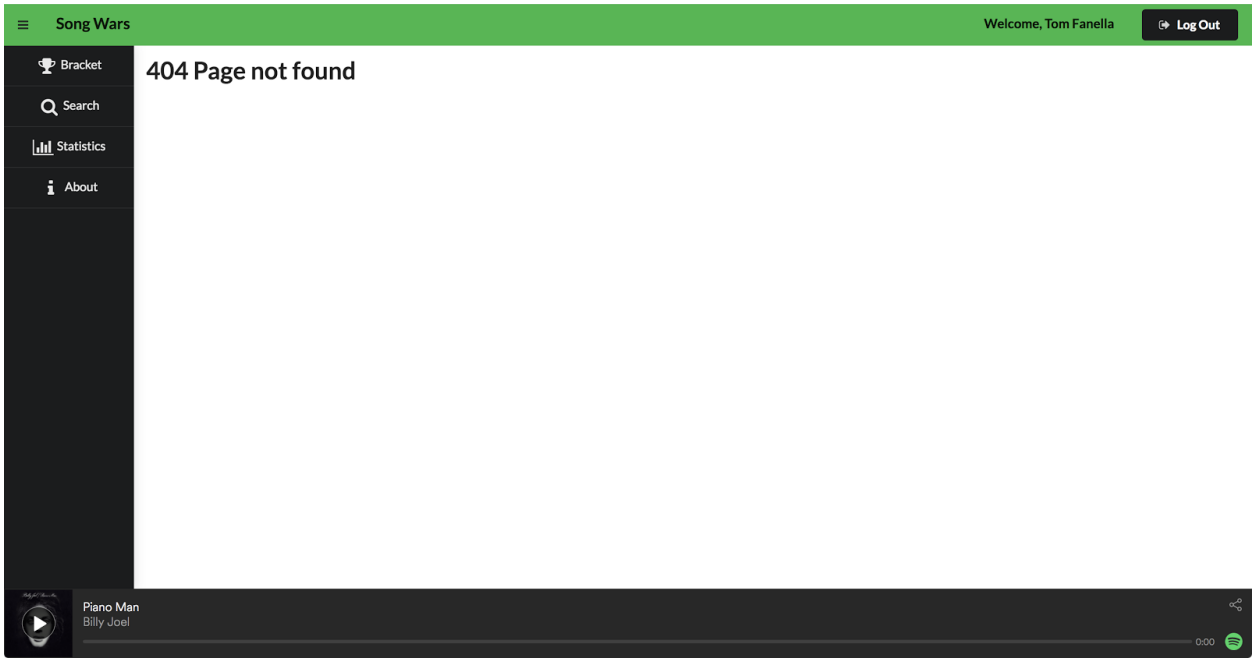
- The statistics page should display statistics about the application to the user
- The statistics page will update on a timed schedule
- If a user wants to manually update the statistics page (see: /admin/stats/update - POST)

/about



- The about page should display some information about the application

/{unmatched}



- A 404 Page should be shown for any route that is not defined in the application

Cross-site abilities

- The Login button should redirect to the spotify login page and store the user's data in the browser
- The Logout button should redirect to the homepage and remove the user's data from the browser
- When a play button is in the player, click the play button in the lower left hand corner to play the song (you must have the Spotify Desktop app installed)
- Clicking any of the links in the sidebar should link to that page
- Clicking the menu bar in the top left corner should toggle the sidebar
- The default song should be either the song the user is currently listening to or the most recent winner

API:

Buggy API Endpoint: https://tluquld4j9.execute-api.us-west-2.amazonaws.com/Release

/user/validate - GET

Intent:

Uses code to retrieve access token and refresh token from Spotify. Relevant user information is saved in the users table, and some of which is returned to user so they can make Spotify API requests client side.

Expected Inputs:

Querysting parameters:

"code": The code returned from Spotify's client-side authentication endpoint. Used to get the access and refresh tokens on server side.

Body:

Empty

Expected Outputs:

Expected Response Codes:

200 - OK - Returns the expected response body below.

400 - BadRequest - If input is malformed and caught before it causes an InternalServerError.

500 - InternalServerError - SQL errors and other unexpected occurrences.

Expected Response Body:

{

“name” : “Firstname Lastname”,

“access_token” : “spotifyaccesstokenhere”,

“user_id” : “spotifyuseridhere”

}

/user/validate/update - POST

/brackets/current - GET

Intent:

Retrieves the current week’s bracket data, which includes every participating song as well as their data. This data includes the song-id, artist name, album, their position in the bracket as well as other pieces of information. For every round that is currently happening/has happened, data will be returned for it. For future rounds/the song winner, data will not be returned until previous rounds have finished/winner has been calculated.

Expected inputs: None

Expected Outputs:

Expected Response Codes:

200 - OK - Returns the expected response body below.

500 - InternalServerError - SQL errors and other unexpected occurrences.

Expected Response Body:

```
{
  “round”: *currentRound
  "bracket_id":"some_bracketId",
  "LeftSide":[
    [
```

```
                {
                  "Option1":{
                    "id":"1IFRVS4t1oII0XG9RBWdKH",
                    "name":"The Girl",
                    "popularity":64,
                    "preview_url":"spotify:track:1IFRVS4t1oII0XG9RBWdKH",
                    "album_name":"Bring Me Your Love",

"album_image":"https://i.scdn.co/image/de87ede7d6a041e35ab7c3c55f37448784c24fb7",
                    "artists_name":"City and Colour",
                    "votes":3,
                    "round":1,
                    "position":1
                  },
                  "Option2":{
                    "id":"1Ea64ROQHsFNdB1U0EqEgf",
                    "name":"Makin' a Move",
                    "popularity":42,
                    "preview_url":"spotify:track:1Ea64ROQHsFNdB1U0EqEgf",
                    "album_name":"Guardians of the Zone",

"album_image":"https://i.scdn.co/image/3745aadfbe24053dbcf104184170535f1993eeec",
                    "artists_name":"TWRP",
                    "votes":0,
                    "round":1,
                    "position":2
                  }... *same format repeated for each song matchup (Option1, Option2)
                   RightSide: [ { …} ], *follows format of left side for each matchup
                   Finals: [ {...} ], *data for two songs, a single matchup
                   Winner: {...} *data for the winner
```


/brackets/current/{bracket-id}/vote - GET

Intent:

Retrieves a randomly and sufficiently sorted set of song matchups that the user is yet to vote on. Songs should be random enough that if most users don’t complete all votes, there won’t be an uneven distribution of votes to the matchups. Matchups returned may be empty if there are no votes left.

Expected Inputs:

Path Parameters:

{bracket-id} - should be the bracket ID of the bracket the user wants to vote on.

Querystring Parameters:

user_id - the Spotify user ID returned with all successful Song Wars API calls.

access_token - Current Spotify access token returned from initial log in and later token updates.

Body:

Empty

Expected Outputs:

Expected Response Codes:

200 - OK - Returns the expected response body below.

400 - BadRequest - If input is malformed and caught before it causes an InternalServerError.

403 - Forbidden - User attempts to make request without valid login credentials (user id and access token).

444 - VotingClosed - User attempts to get votes on days that the bracket is not open. The bracket is only open on Tuesday, Wednesday, Thursday, and Friday.

500 - InternalServerError - SQL errors and other unexpected occurrences.

Expected Response Body:

```
{
    “name” : “Firstname Lastname”,
    “access_token” : “spotifyaccesstokenhere”,
    “Bracket_id” : “shortid”,
    “Round” : 1-5,
    “Matchups” : [
        {
            “Song1” : {
                “position” : 1-(16/2^(round-1)),
                “id” : “spotifysongidhere”,
                “name” : “spotifysongnamehere”,
                “popularity” : 1-100,
                “preview_url” : “spotifypreviewurihere”,
                “album_name” : “spotifyalbumnamehere”,
                “album_image” : “spotifyurltoalbumimagehere”,
                “artists_name” : “spotifyartistsnamehere”,
                “bracket_id” : “shortid”
            },
```

```
                    “Song2” : {
                        “position” : 1-(16/2^(round-1)),
                        “id” : “spotifysongidhere”,
                        “name” : “spotifysongnamehere”,
                        “popularity” : 1-100,
                        “preview_url” : “spotifypreviewurihere”,
                        “album_name” : “spotifyalbumnamehere”,
                        “album_image” : “spotifyurltoalbumimagehere”,
                        “artists_name” : “spotifyartistsnamehere”,
                        “bracket_id” : “shortid”
                    }
                },
                {
                    “Song1” : {
                        “position” : 1-(16/2^(round-1)),
                        “id” : “spotifysongidhere”,
                        “name” : “spotifysongnamehere”,
                        “popularity” : 1-100,
                        “preview_url” : “spotifypreviewurihere”,
                        “album_name” : “spotifyalbumnamehere”,
                        “album_image” : “spotifyurltoalbumimagehere”,
                        “artists_name” : “spotifyartistsnamehere”,
                        “bracket_id” : “shortid”
                    },
                    “Song2” : {
                        “position” : 1-(16/2^(round-1)),
                        “id” : “spotifysongidhere”,
                        “name” : “spotifysongnamehere”,
                        “popularity” : 1-100,
                        “preview_url” : “spotifypreviewurihere”,
                        “album_name” : “spotifyalbumnamehere”,
                        “album_image” : “spotifyurltoalbumimagehere”,
                        “artists_name” : “spotifyartistsnamehere”,
                        “bracket_id” : “shortid”
                    }
                },
                etc.
        ],
        “user_id” : “spotifyuseridhere”
}
```

/brackets/current/{bracket-id}/vote - POST

Intent: Records votes that are cast for a song in the database. Updates the table as well as records which song has already been voted for by the user, as to prevent multiple voting/voting for both songs in a matchup.

Expected Inputs:

- Body:
  - Access_token
  - Vote: {
    - User_id,
    - Song_id,
    - Round,
    - Position,
    - Bracket_id
  - }

Expected Output:

- Expected Response Codes:
  - 200 - Success
  - 400 - BadRequest - If input is malformed and caught before it causes an InternalServerError.
  - 403 - Forbidden - User attempts to make request without valid login credentials (user id and access token).
  - 500 - InternalServerError - SQL errors and other unexpected occurrences.
- Expected Response Body:
  - {status: success}

/brackets/history - GET

Intent: Retrieves bracket headers so that the user will be able to look up previous brackets based on the date and bracket id.

Expected Inputs:

Querystring:

access_token - Current Spotify access token returned from initial log in and later token updates.

Expected Outputs:

Expected Response Codes:

403 - Forbidden - User attempts to make request without valid login credentials (user id and access token).

Expected Response Body:

```
{
Headers : [
        {
                bracket_id,
                Date
        } … *an array of objects that contain individual header data
}
```

/brackets/history/{bracket-id} - GET

Intent: Retrieve bracket data similarly to how bracket data is retrieved for brackets/current. Returns an object containing participating songs and their individual data for a bracket that was completed during past weeks. Each bracket is assigned to a unique bracket-id, which is used to query the database for all the data associated to that bracket_id

Expected Input:

Path Parameters:

{bracket-id} - should be the bracket ID of the bracket the user wants to retrieve.

Querystring Parameters:

Access_token - Current Spotify access token returned from initial log in and later token

Body:

None

Expected Output:

Expected Response Codes:

200 - OK - Returns the expected response body below.

500 - InternalServerError - SQL errors and other unexpected occurrences.

Expected Response Body:


```
{
  "bracket_id":"some_bracketId",
  "LeftSide":[
    [
      {
        "Option1":{
          "id":"1IFRVS4t1oll0XG9RBWdKH",
          "name":"The Girl",
          "popularity":64,
          "preview_url":"spotify:track:1IFRVS4t1oll0XG9RBWdKH",
          "album_name":"Bring Me Your Love",
          "album_image":"https://i.scdn.co/image/de87ede7d6a041e35ab7c3c55f37448784c24fb7",
          "artists_name":"City and Colour",
          "votes":3,
          "round":1,
          "position":1
        },
        "Option2":{
          "id":"1Ea64ROQHsFNdB1U0EqEgf",
          "name":"Makin' a Move",
          "popularity":42,
          "preview_url":"spotify:track:1Ea64ROQHsFNdB1U0EqEgf",
          "album_name":"Guardians of the Zone",
          "album_image":"https://i.scdn.co/image/3745aadfbe24053dbcf104184170535f1993eeec",
          "artists_name":"TWRP",
          "votes":0,
          "round":1,
          "position":2
        }... *same format repeated for each song matchup (Option1, Option2)
          RightSide: [ { ...} ], *follows format of left side for each matchup
          Finals: [ {...} ], *data for two songs, a single matchup
          Winner: {...} *data for the winner
```


/song/recommend - POST

Intent:

Recommends song for next week's bracket. If a song hasn't been recommended before, then the data supplied fills the table for that song. Otherwise, the song is identified by id and its recommendations count is increased by one.

Expected Input:

Querystring - Current Spotify access token returned from initial log in and later token updates.

Body:

Empty

Expected Output:

Response Codes:

200 - Success

400 - BadRequest - If input is malformed and caught before it causes an InternalServerError.

403 - Forbidden - User attempts to make request without valid login credentials (user id and access token).

500 - InternalServerError - SQL errors and other unexpected occurrences.

Response Body:

```
{
    "status" : "success"
}
```

/stats - GET

Intent:

Returns all available statistics in the stats table. This includes, vote count, recommendation count (for this bracket, from last week), the top artists by number of recommended songs in all brackets, and the top songs by number of wins.

Expected Input:

Querystring Parameters:

user_id - the Spotify user ID returned with all successful Song Wars API calls.

access_token - Current Spotify access token returned from initial log in and later token updates.

Body:

Empty

Expected Output:

Response Codes:

200 - OK - Returns the expected response body below.

400 - BadRequest - If input is malformed and caught before it causes an InternalServerError.

500 - InternalServerError - SQL errors and other unexpected occurrences.

Response Body:

```
{
    “recommendations” : #,
    “votes” : #,
    “top_artists” : [
        {
            “name” : “artistname1”,
            “count” : #
        },
        {
            “name” : “artistname2”,
            “count” : #
        },
        … total 10x
    ],
    “top_songs” : [
        {
            “name” : “songname1”,
            “count” : #
        },
        {
            “name” : “songname2”,
            “count” : #
        },
        … total 10x
    ],
    “access_token” : “spotifyaccesstokenhere”,
    “user_id” : “spotifyuseridhere”
}
```

/admin/migrate/brackets - POST

Intent:

In production, this endpoint wouldn't exist, and the code would run in a timed manner, once per week, on Monday at midnight, however for testing purposes we open this to you. The purpose of this is to move on from last week's bracket and start a new bracket from the recommendations taken from the last week. This also updates in stats the number of recommendations for the new bracket. The sequence of events here is to take top recommendations for songs below or equal to a popularity of 70, and grab the next top 10 with unrestricted popularity rating and fill the last_week_bracket table at round one. Songs in each side of the bracket are shuffled (still no cross-pollination however). Afterward, last_week_bracket, users_last_week_bracket, users_recommendations, and recommendations tables are emptied.

Expected Input:

> Empty

Expected Output:

> Empty

/admin/migrate/round - POST

Intent:

In production, this endpoint wouldn't exist, and the code would run in a timed manner, once on Tuesday midnight, Wednesday midnight, Thursday midnight, and Friday midnight, however for testing purposes we open this to you. The purpose of this is to move the winners of the last round, to the next round, increasing the round by one.

Expected Input:

> Empty

Expected Output:

> Empty

/admin/stats/update - POST

Intent:

Updates all statistics into stats table, so that updated statistics will be returned by /stats - GET. Updates everything but the number of recommendations, which is done when brackets are migrated. In production, this endpoint wouldn't exist, and the code would run in a timed manner.

Expected Input:

None

Expected Output:

None