

CS408 Team 17

Song Wars

Thomas Fanella, Austin Miller, Ben Maxfield, Christian Lock, Terry Lam

Product Backlog

I. Problem Statement

Currently there is no entertaining method for discovering new music through Spotify. Song Wars aims to gamify the process of finding new music by involving our users' competitive nature to find and share the next up and coming songs. We think this method will entice users to discover new music in a fun way.

II. Background information

The only way to find new music on Spotify is to browse through the ever growing number of playlists that are available or to listen to the playlists Spotify creates for you. This makes it difficult for music lovers to show underground music to fans of popular music. There is a need to compare popular music to undiscovered music.

III. Environment

For the front end we will be using the React framework in conjunction with Redux, react-router, and Semantic-UI. This will ensure our code is properly organized and the end product is efficient and easy to use. We will be using create-react-app to generate boilerplate to speed up development. For the back end we will be using AWS Lambda. We will be coding our Lambda functions in Java. We will also be taking advantage of the Spotify Developer API.

IV. Functional Requirements

| Backlog ID: | Functional Requirement: | Hours: | Status: |
|-------------|--|--------|-----------------------|
| 1 | As a user, I would like to login with Spotify. | 5 | Completed in sprint 1 |
| 2 | As a user, I would like to search for new songs. | 10 | Completed in sprint 1 |
| 3 | As a user, I would like to view the popularity and preview of each song. | 10 | Completed in sprint 2 |

| | | | |
|----|--|-----|-----------------------|
| 4 | As a user, I would like to recommend a song. | 15 | Completed in sprint 1 |
| 5 | As a user, I would like to view a bracket containing the highest recommended songs of the previous week. | 35 | Completed in sprint 2 |
| 6 | As a user, I would like to vote on two competing songs at the current stage in the bracket. | 30 | Completed in sprint 2 |
| 7 | As a user, I would like to view a history of past completed brackets. | 20 | Completed in sprint 2 |
| 8 | As a user, I would like to view statistics based on the previous weeks results. | 20 | Completed in sprint 2 |
| 11 | As a user, I would like Song Wars to automatically refresh my Spotify login session | 5 | Completed in sprint 2 |
| | TOTAL: | 150 | |

V. Non-Functional Requirements

As a User I would like...

1. to have easy and simple UI navigation
2. quick UI responsiveness (under 1 second page load on current generation computers).

As a developer I would like...

1. that the version control is well organized with with a release branch, a development branch, and a branch for each feature that gets merged into the development branch upon completion.
2. to manage user data using AWS as our back end web service
3. to prevent security and privacy breaches
4. to use Spotify's developer API to pull song data and analytics
5. to have the user stories well organized and easily viewable.
6. that the sprints be well planned out and manageable.
7. to easily communicate within the group.

VI. Use Cases

| Name | Actor Actions | System Responses |
|---------------|---------------------------|--|
| Spotify login | 1. Enter user credentials | 2. Authenticate with Spotify 3. Store an access token for API usage |

| | | |
|-----------------|--|---|
| Song search | 1. Enter keyword | 2. Submit keyword to Spotify API 3. Receive song results 4. Displays results including popularity and preview |
| Recommend song | 1. Select recommend button | 2. Send recommendation to backend 3. Backend records vote 4. Notifies user of success/fail |
| View bracket | 1. Select bracket view | 2. Requests data from backend/database 3. Display data |
| Vote on match | 1. Select vote 3. Select song to play 4. Cast vote | 2. Displays both songs with Spotify player 4. Plays requested song 5. Records vote 6. Proceeds to next match |
| View history | 1. Select history view 4. Select bracket | 2. Request data from backend/database 3. Display a list of historic brackets 5. Display specific bracket |
| View statistics | 1. Select statistics view | 2. Requests data from backend/database 3. Display data |