## Revised Product Backlog

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | **Story** | **Initial priority** | **Task** | **Estimate** | **Implementation notes** | **Actual Effort** |
| Implementation for creating a playlist | The user should be able to create a playlist from the current library of songs in blocks of 1 - 3 hours each. | High | A linked list will be implemented that will contain the desired songs for a playlist and will be used to save the playlist. It will then be read into a queue to be played. | 1.5 hours | * A python list (similar to an array list in java) so it had many of the same features as LinkedList with better indexing * A list was used instead of linked list as there was a sufficient reason as to why it was used | 2 hours |
| The song queue will have limitations that require the total time of all songs to be between 1 to 3 hours. Otherwise the queue will not be able to be playable. | 1 hour | * To-do |  |
| The admin should have the ability to limit songs from being selected. | High | The user classified as the administrator will be able to select a button on the song library that will allow the admin to select songs for restricted access. | 2 hour | * The administrator is able to remove a song which makes the isAllowed class for the song false instead of true |  |
| When browsing the song database, users should have the ability to organize the songs from the library in a particular classification, such as; band, genre, year and future classifications. | High | A drop-down menu will be implemented for any classification necessary, with a list of options. A selection of options should accurately search the list and find songs which are of those properties. | 1 hour | * To-do |  |
| A tool bar will be implemented which contains the menus and allows for expansion for future classifications menus. | 1 hour | * A navbar was implement seeing as we have a website | 1.5 hours |
| The playlist creation tool should suggest songs from the database that are similar to the history of the users. | Low | The tool will do a smart search which will take the users previous selections of songs, dissect its traits and compare them to other songs in the database, actively displaying the results to the user. | ½ hour | * Currently all the songs are recommended | 1 hour |
| The user should have the ability to fill the playlist with a random assortment of songs given particular classifications by the user. | Low | The randomizing tool will be available somewhere near the top of the song browser GUI, with check indicating it is active | N/A | N/A |  |
| While the tool is active, a set of classifications will be available (genre, musician, mood) and are selectable. | N/A | N/A |  |
| The queue auto fills with randomly selected songs, without repeat, if possible. The songs must follow the restrictions set by the user. | N/A | N/A |  |
| Playlist, songs and users should have an array of statistics/analysis tools | The user should be able to retrieve statistical info about any song in a playlist, such as its relative frequency within a time period selected by the user. | High | Next to every song, a statistics page button will be available to bring them to a page with that song’s stats | 1 hour | * An anchor tag was used to link them to the song statistics |  |
| A relative frequency stats section will be available, and it will have a time frame selection by utilizing a Google calendar. The songs relative frequency in that time period will be displayed as a line graph with the y-axis being relative frequency in percentage and the x- axis representing time. | N/A | N/A |  |
| A log of the songs played by users should be available to other users, with customization to time frame. | Med | A list will be created for every user, with basic information of each song that they have played. | 1 hour | * To-do * A set will be used in the user object to track what songs they have played |  |
| A time frame can be selected from a drop down menu containing time frames such as; 1 day, 1 week, all time etc. | N/A | N/A |  |
| There should be a statistics page which contains stats about song, musician and genre selections per user, as well as the frequency the user accesses the song database | High | A statistics page will be automatically created for each user account created. | 1 hour | * To-do * Each user object will have all songs they played and playlists they played will be tracked * See implementation above for the log of the songs played by users |  |
| Pages of stats will be available on the user’s stats page, with a tool bar located to the left to select the desired page. The side bar must be customizable | N/A | N/A |  |
| Playlists must have the ability to be modified. | The user must be able to edit already made playlists in their saved playlists collection but must keep the time length between 1 – 3 hours. | High | Since saved playlists utilize a linked list, adding, removing and general reorganization of a playlist will done using typical list modifications algorithms such as changing required pointers from song to song. | 1 hour | * To update the playlist a new form must be submitted * This was used because it requires constant time vs. linear time | 2 hours |
| The user should have the ability to make modifications while the playlist is currently active. | High | Currently active playlists utilize a queue, so a user will have the ability to designate an upcoming song to be skipped with a button on the GUI. | 1 hour | * A button was created to link the user to the next song (skipping the current song) |  |
| The queue will accept additions to the playlist on the fly by adding any new songs to the end of the queue. | N/A | N/A |  |
| Every DJ should have a personal profile | The account should have the ability to save any of the user's playlists for future use. | High | Each user will be given a hash table upon creation of an account which will store all of the user’s playlists. The keys for each table will be the user’s unique user name in conjunction with a unique playlist name per playlist. | 2 hours | * We used an object-oriented approach along with a JSON file to store the data for each DJ * When the JSON file is loaded the user, information is stored in a dictionary * We used a flask login manager to keep track of the user that has logged in * We have a user class which has methods associated with saving data to the JSON file if a new user registers an account | 3 hours |
| A DJ profile should have the potential to be classified as an administrator | Med | For each current user, the head administrator will be able to access a ‘make admin’ option in the user’s options pane. | 2 hours | * To-do * We need to create a head admin that has privileges to make other users admins as well |  |

Total Estimated Time: 16 hours

Total Actual Time: