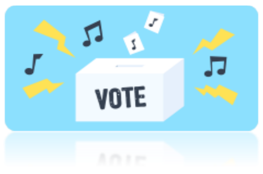


Table of Contents

1. Introduction

The general idea of our system is that an organisation can make a list of songs. Each organiser will have an ID that links to the list so that people can access it and vote which songs they like. One person can vote for five songs in the playlist, when one of their voted songs is played, they will get one vote back. The system will calculate the votes and display it to the organiser. The vote will automatically be refreshed after 12 hours of the vote.

There are three main categories of users, such as:

a. Establishment

We define establishments as bars, clubs, restaurants and cafes. They will benefit from our system because they will have their customers participate and make them feel more involved. That way, it will improve customer satisfaction.

b. Event Organiser

We define event organisers as festival organisers. The organisers can make a list of songs of some artists. After doing so, they will get the event-ID and publish it in advance to their customers with time limitation. By viewing the result of the votes, they will have more guidance towards which songs are the most popular among the customers to achieve a better rating and retention rate for the next event.

c. Customer

We define customers as people who vote songs from the list. By using our system, they will feel more involved in the particular event. They also will feel happier because their voted songs are played.

2. Description

2.1. Focus on properties

The app has some properties as follows:

1. Login

   In the 'Login' page, people can login by filling in their usernames, their passwords and an ID number from an organiser whose playlist they want to see and vote the songs in the playlist.

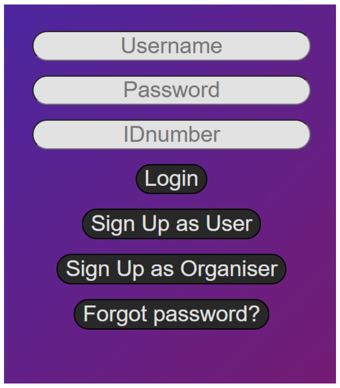


Figure 1. Login page.

2. Sign up as user

   In the 'Sign up as user' page, people can register their account by filling in their choice of username as well as password.

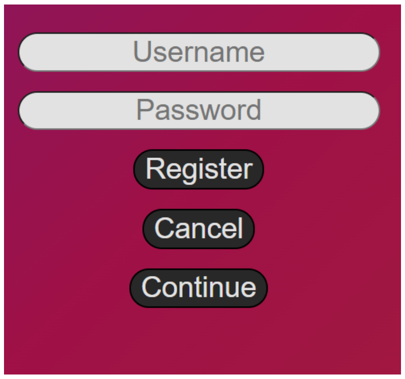


Figure 2. Sign up as user page.

3. Sign up as organiser

   In the ‘Sign up as organiser’ page, people can register their account by filling in their choice of username as well as password.

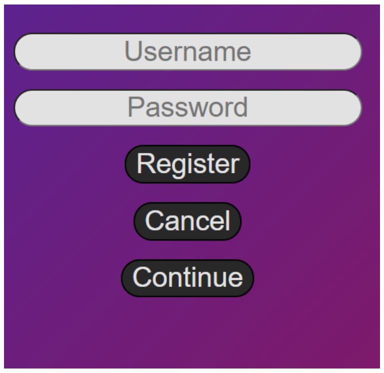


Figure 3. Sign up as organiser page.

4. Playlist

   There are 2 different pages for the ‘Playlist’ page, one is from the organiser’s point of view and the other one is from the user’s point of view.

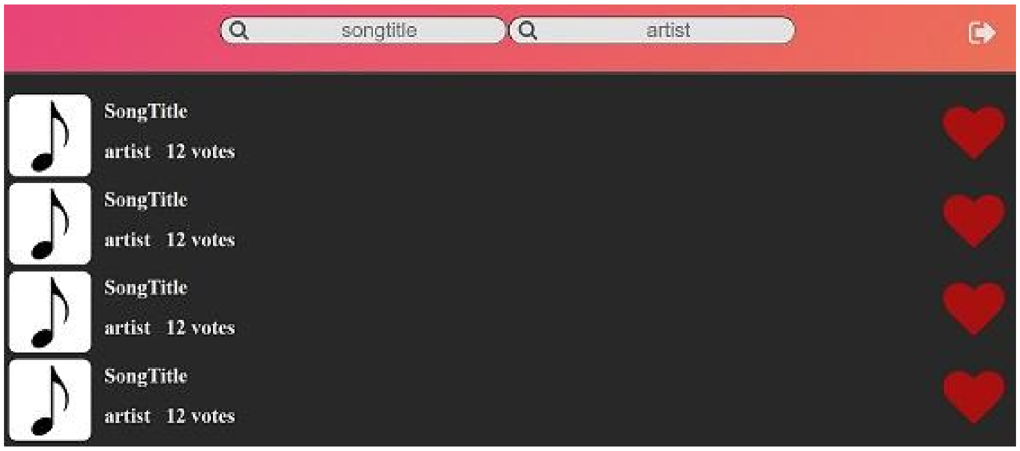


Figure 4. User’s playlist page.

5. Add song

Only organiser who can access to the ‘Add song’ page, because users are not allowed to add any songs to the playlist.

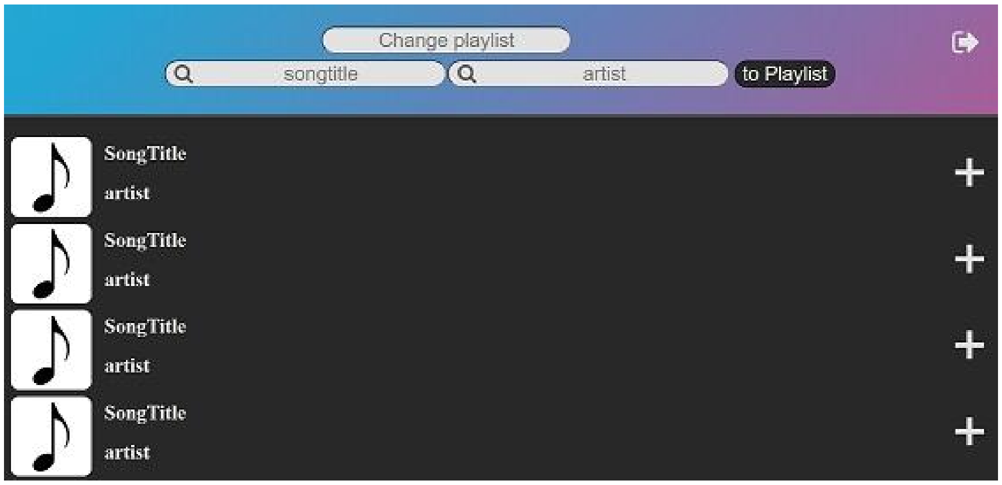


Figure 5. Organiser's playlist - add song page.

6. Remove song

The same as the “Add song” page, but here the organiser can remove songs from the playlist.

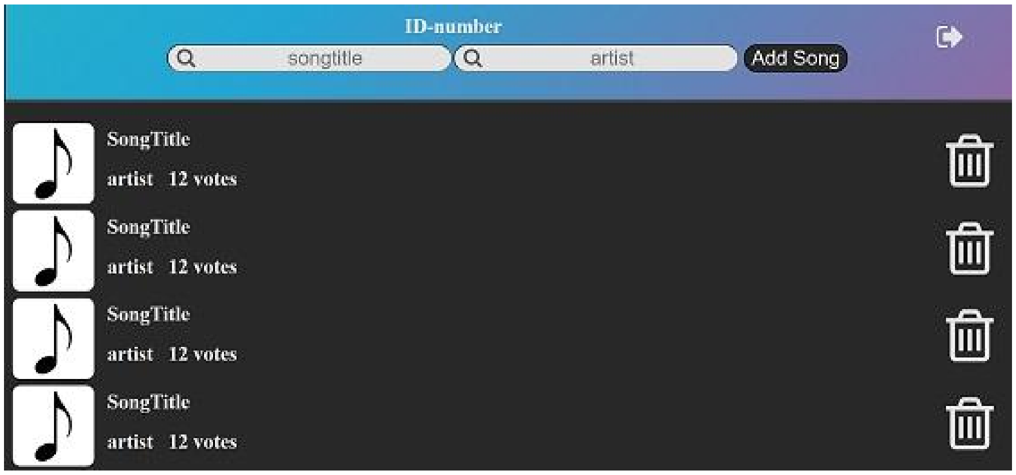


Figure 6. Organiser's playlist - remove song page.

2.2. Product justification

There are some similar applications that exist e.g. Mubo, Jukestar. The Mubo app allows users to create their own Musicbox. The Musicbox allows users to provide a Playlist. Guests can connect to the Musicbox and vote for their favourite songs using the Mubo app, however each guest has a fixed number of votes. The Jukestar app has a similar function, but it takes about 30 seconds until guests can vote on other songs again. The Jukestar app works on tokens, each guest has 12 tokens to begin with and for each vote. The vote is worth 1 token and to skip the waiting time (30seconds) to vote another song, they can pay up 2 tokens.

Our app is much simpler than the two existing apps. Our target audiences are different from theirs, we are focusing on local businesses, local events and their customers. Local businesses will have an option to interact with their customers. Event organisers will have an effective and more accurate plan. We will put a limit on five simultaneous votes. This way, votes are meaningful and users do not have to constantly check the app. Users will regain their votes after the song is played, or when they sign into a different organisation. Our app will not allow guests to add any random songs to the list to prevent confusion on the organiser's side.

As an addition to it, we thought of a last minute idea to also include the menu card that the establishments have. For event organisers, they can add promo flyers instead of menu cards. This way there is also a direct monetary gain for the organisers. However, we did not get to implement it because of the time limitation.

2.3. Specifications

In the use case diagram below, we specify the overall view of the app.

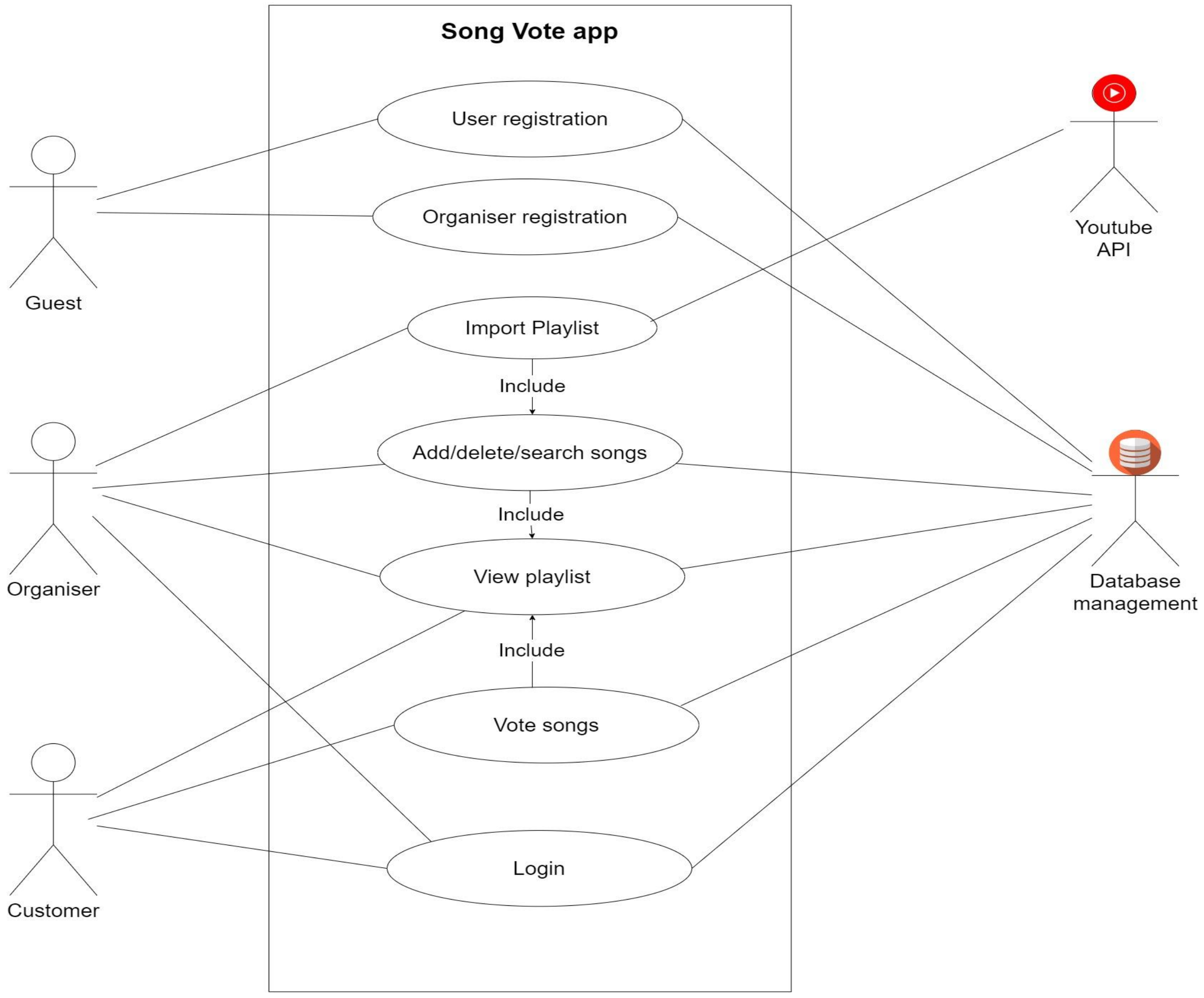


Figure 7. Use case diagram.

As seen in the diagram above, there are three different parts of function in the app. There are guests, organisers and customers. As for guests, they can only earn access to the app after the registration. After login, organisers can view their own playlist and make alterations to it. This means searching, adding and/or deleting any songs from/to their playlist. Additionally, organisers can add their menu to the system. Organiser's customers can login to the app using their username, password and organiser's ID to vote songs in the playlist.

ID and name	UC-1 Song Vote application
Primary actor	End-user Secondary actor: database management and Youtube API
Description	Guests can register as an organiser and/or as a customer. Organisers can login, import playlist and make a playlist by searching, adding and/or deleting songs. Customers can login, view organisers' playlist and vote on songs.
Trigger	Organiser indicates he/she would like to make a playlist for his/her customers to vote on. Customer indicates he/she would like to vote on songs in the playlist that the organiser makes.
Precondition	User opens a web browser.
Post-condition	POST-1 : Guests become organisers and/or customers. POST-2 : Organisers have playlists of songs. POST-3 : Customers vote on songs in the playlists.
Normal flow	**Register as user:** Step 1: User opens the web app. Step 2: User clicks on the 'Sign up as user' button. Step 3: User fills in his/her desired username. Step 4: User fills in his/her desired password. Step 5: User clicks on the 'Register' button. **Register as organiser:** Step 1: User opens the web app. Step 2: User clicks on the 'Sign up as organiser' button. Step 3: User fills in his/her desired username. Step 4: User fills in his/her desired password. Step 5: User clicks on the 'Register' button. **Make a playlist (for organisers):** Step 1: Guest user opens the web app. Step 2: Guest user fills in his/her username. Step 3: Guest user fills in his/her password. Step 4: Guest user clicks on 'Login' button. Step 5: Organiser searches for songs he/she desires. Step 6: Organiser adds songs to the playlist by clicking on the song title. Step 7: (optional) Organiser deletes songs from the playlist by clicking on the trash bin icon on the far right side of the song title. **Vote on a song (for customers):** Step 1: Guest user opens the web app. Step 2: Guest user fills in his/her username. Step 3: Guest user fills in his/her password. Step 4: Guest user fills in the organiser's ID number.

	Step 5: Guest user clicks on ‘Login’ button. Step 6: Customer searches/scrolls for songs he/she desires to vote on. Step 7: Customer votes a song by clicking on heart icon button. Step 8: (optional) Customer undoes his/her vote by clicking the heart icon button again.
Alternative flow	n/a.
Exception	**Register as user and organiser:** Give an error message for used username for registration. Ask the guest user to enter a valid and not-used username for registration. **Make a playlist (for organisers):** Give an error message for unknown artists/songs when searching artists/songs. Ask the organisers to enter a valid artists’ name or songs’ title. **Vote on a song (for customers):** The heart icon button is not responding due to a malfunction of the system and the system will give an error message to the users.
Priority	High.
Frequency of use	**For Organisers:** Expect a high frequency of this use case during the first few months, because the organisers would like to update their playlist to make their customers interested. **For Customers:** Expect a high frequency of this use case during the first visits to the organisers’ establishments, because customers are curious. However, only those who find the application cool will continue to use it for different establishments.
Assumptions	**Register as user:** Registered correct information about the user. **Register as organiser:** Registered correct information about the organiser. **Make a playlist (for organisers):** Imported correct playlist from youtube music API. **Vote on a song (for customers):** Imported correct playlist based on organiser’s ID number.
Other information	n/a.

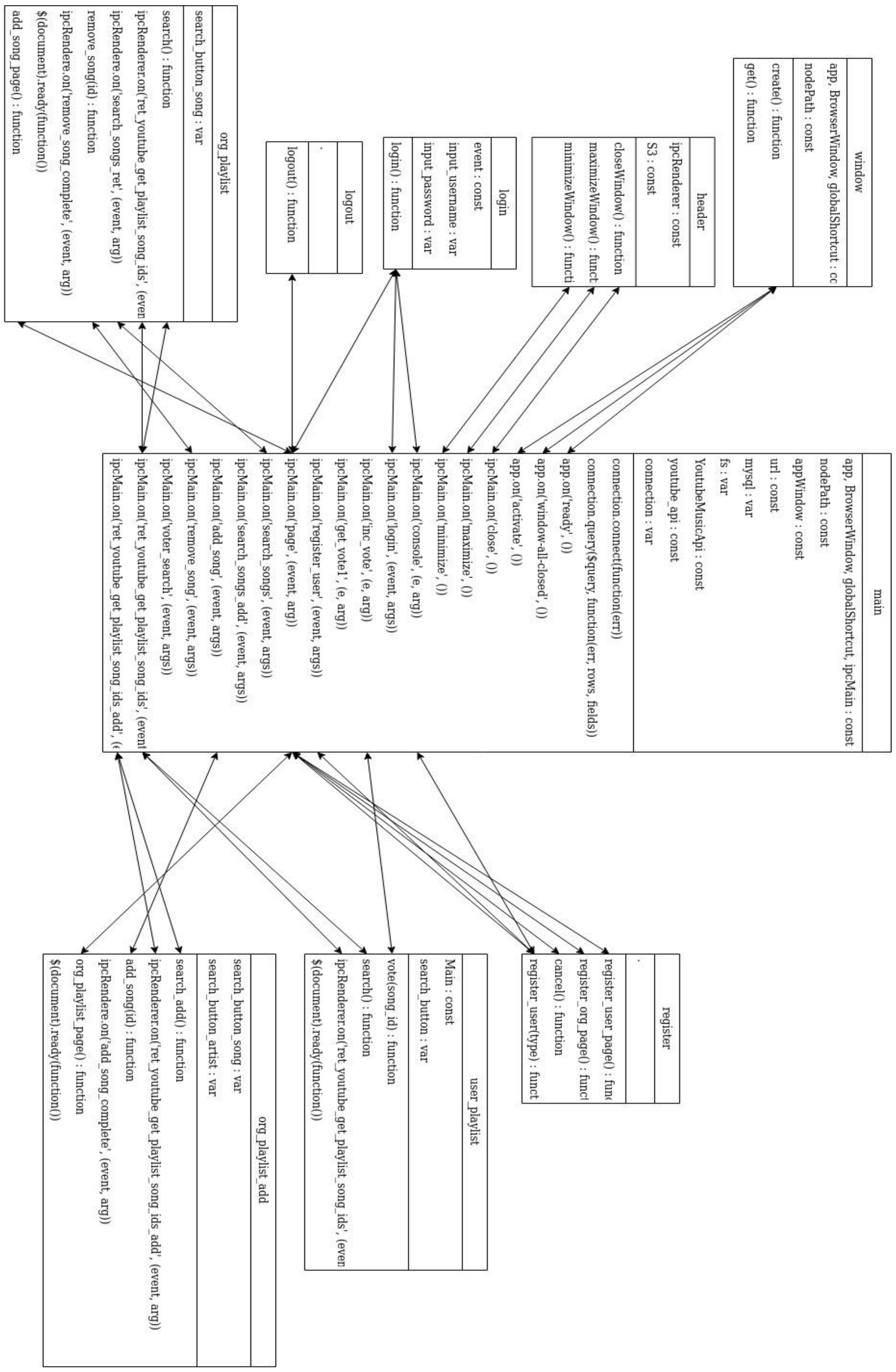
Table 1. Use case template.

3. Design

3.1. Global design

For the global design, we use HTML/CSS as the front-end program, JavaScript and SQL as the back-end and through Electron we connect the back-end and front-end program. We made the basic design in HTML, such as the field where users can type in their inputs, the buttons for login, register as user, register as organiser and forgot password in the Login page. Then we made the design look better through the use of CSS.

3.2. Detailed design *Figure 8. Class diagram.*


org_playlist
search_button_song : var
search() : function
ipcRenderer.on('ret_youtube_get_playlist_song_ids', (even
ipcRendere.on('search_songs_ret', (event, arg))
remove_song(id) : function
ipcRendere.on('remove_song_complete', (event, arg))
$(document).ready(function())
add_song_page() : function
logout
.
logout() : function
login
event : const
input_username : var
input_password : var
login() : function
header
ipcRenderer : const
S3 : const
closeWindow() : function
maximizeWindow() : funct
minimizeWindow() : functi
window
app, BrowserWindow, globalShortcut : cc
nodePath : const
create() : function
get() : function
main
app, BrowserWindow, globalShortcut, ipcMain : const
nodePath : const
appWindow : const
url : const
mysql : var
fs : var
YoutubeMusicApi : const
youtube_api : const
connection : var
connection.connect(function(err))
connection.query($query, function(err, rows, fields))
app.on('ready', ())
app.on('window-all-closed', ())
app.on('activate', ())
ipcMain.on('close', ())
ipcMain.on('maximize', ())
ipcMain.on('minimize', ())
ipcMain.on('console', (e, arg))
ipcMain.on('login', (event, args))
ipcMain.on('inc_vote', (e, arg))
ipcMain.on('get_vote1', (e, arg))
ipcMain.on('register_user', (event, args))
ipcMain.on('page', (event, arg))
ipcMain.on('search_songs', (event, args))
ipcMain.on('search_songs_add', (event, args))
ipcMain.on('add_song', (event, args))
ipcMain.on('remove_song', (event, args))
ipcMain.on('voter_search', (event, args))
ipcMain.on('ret_youtube_get_playlist_song_ids', (event
ipcMain.on('ret_youtube_get_playlist_song_ids_add', (
register
.
register_user_page() : func
register_org_page() : funct
cancel() : function
register_user(type) : funct
user_playlist
Main : const
search_button : var
vote(song_id) : function
search() : function
ipcRenderer.on('ret_youtube_get_playlist_song_ids', (even
$(document).ready(function())
org_playlist_add
search_button_song : var
search_button_artist : var
search_add() : function
ipcRenderer.on('ret_youtube_get_playlist_song_ids_add', (event, arg))
add_song(id) : function
ipcRendere.on('add_song_complete', (event, arg))
org_playlist_page() : function
$(document).ready(function())

3.3. Design justification

Our design is overall simple and clear. Just like any playlist you can scroll through the songs. The login and registration pages also do not have much except for the input bars, which are rounded for a softer look, and the buttons needed plus a satisfying rainbow gradient background that we also used as a fixed top (when scrolling stays at the top) in our playlist pages. In addition, we did not make the white and black we used pure and the background for playlists dark, so that your eyes will not hurt while staring at it. Because the transition between this light fixed top and the dark playlist backgrounds was very sharp, we made the border of our fixed top a lighter black shade. For the add, remove and vote buttons we used icons instead of text, because they are simpler, take up less space and more easily grab the users attention. When putting your mouse on the buttons, except the vote one, the border or icon slowly turns turquoise. This is a way for users to know that their mouse is on the button. The slow transition gives off a more relaxed vibe matching with the slow transition of the gradient backgrounds. For the vote button, the heart, we made it pulse and light up when hovering to make it look a bit more like a real heart of a person that is excited about a song before clicking. It turns red after the user clicks on it, so that he/she knows he/she has voted for the song. Lastly, to make it obvious when input bars are search bars we put the search icon inside the bar.

4. Evaluation

As a group we learned a lot, we used lots of programs, websites and toolkits:

Electron & Node Js: Framework and used to install modules

Visual Studio: Used for making the html and css code for the website

Bootstrap: Tried to use for making the website more easily

Codecademy: Used for learning html, css and bootstrap

Blender: Used for creating a background, which we did not get to use

Google docs: Used for making the Project Report

Discord: For communication

And lots of other online resources to answer small problems we encountered.

As a group we all learned something new, for some it was back-end focussed and others more front-end.

Development process:

The communication in the Discord group was very nice, we are all friends so we could talk about R&D anytime. Sometimes we planned to discuss parts of the project during dinner or thought of cool ideas during lunch. This made the collaboration pleasant. Everyone was also really quick to respond on Discord. All of us were pretty motivated as well, because we liked the things that we were doing, be that programming, designing or brainstorming. As for designing, Codecademies tutorials were lovely and graspable. The demo helped us in seeing that we had done a good job on our application and got us the feedback we needed for the finishing touches. All in all, we are satisfied with the result of our project.

Unsolved issues:

At first we wanted to be able to convert the application as a mobile app after making it as a website. For this we tried to use Bootstrap, which uses a grid pane to more easily transform the website into an app. Sadly, we did not gain enough experience to change the CSS code into Bootstrap and get everything in the right place again. We decided for the time we had left that it was better to spend it on making the pages prettier instead. Maybe for the future we will make it into an app too.

Another unsolved issue was the implementation for the ‘Forgot password?’ button. We did not have time left for this after creating the rest and wanted to do this at last, because it is not necessary for the functionality of our application.

Future:

In the future we can now all work more efficiently on a project. We know that keeping things simple in the beginning is a good habit and that it is not wise to switch to another toolkit close to the end of a development process. We learned these lessons by breaking them first: we wanted to make the application work by using a QR-code instead of an ID-number and we tried making an app for mobile and PC at the same time.