TINY-TUNES

Music Player Microservice

MAY 18, 2021

SURENDRA DURA
2020436@student.cct.ie

Table of Contents

Introduction:

Tiny-Tunes is my application name where I am going to make a music player microservice application. My application going to have the 4 services which are as follows:

1. Search-and-play songs
2. Create playlists
3. Record stats about a song
4. Create an automated playlist engine

The application is made in the Fastify node js(Fastify, Fast and low overhead web framework, for Node.js, 2021) in the backend service and the html and the JavaScript is used in the Frontend. The application is a microservices and it was deployed in the Kubernetes using the Docker.

The below figure is my interface of all 4 services:

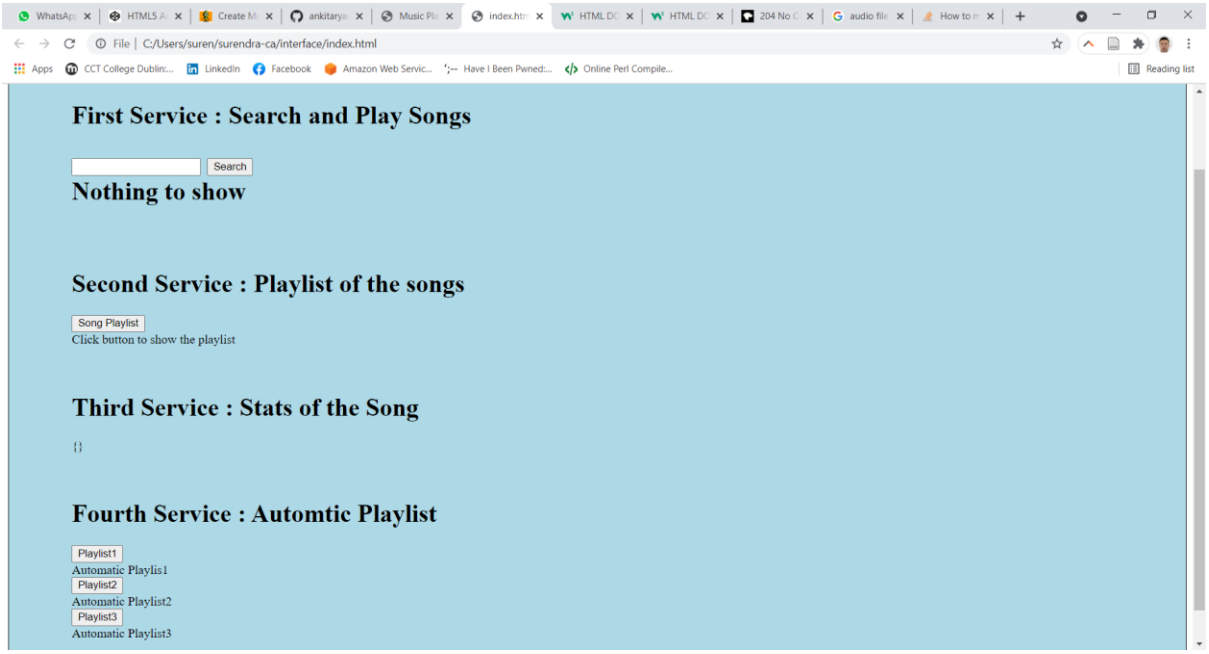


Fig: 4 Services Interface

The below figure is my normal diagram, but I am going to use on top of the docker and Kubernetes in all the four service:

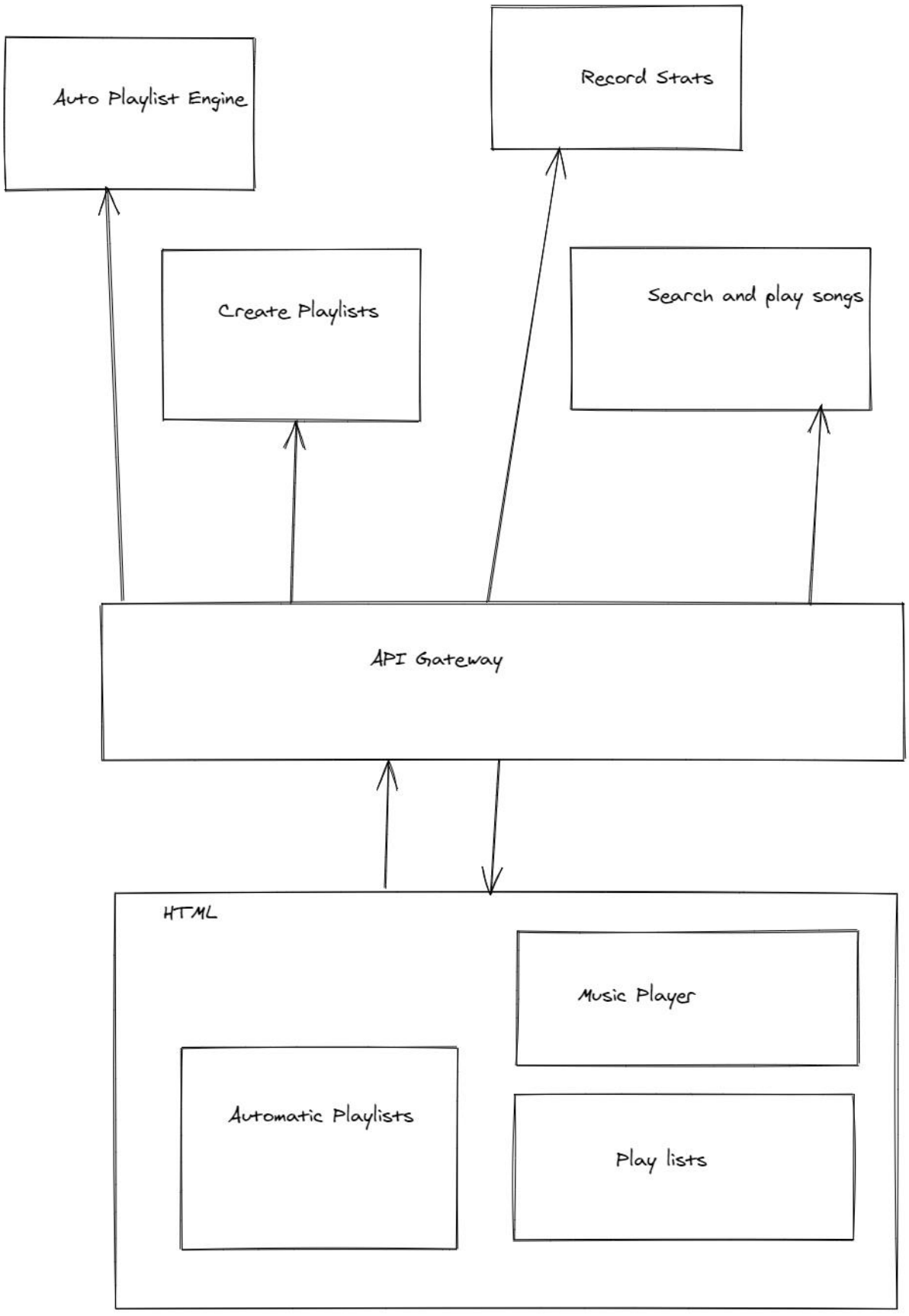


Fig: Tiny-Tunes Service(Gonzalez, 2021)

Search-and-Play Songs:

This is my first service where I made a search the songs using their name and then play the songs. For this I used the Fastify website to make a server(*Fastify, Fast and low overhead web framework, for Node.js*, 2021). On the other hand, I used the David code (Gonzalez, 2021) which he have in the google drive.

For this service, I made a normal service to run the Fastify and then after that I made two routes inside the index.js file (which is my node js file for search and play songs) which are search and the songs. The search is the get request of Fastify where I can search the songs and another song is also get request of Fastify server where I can play the music. Both get request need the song data to fetch the file, so I made a songs folder(*MP3Juices - Free MP3 Downloads*, 2021) inside the search and play songs service and input the list of songs in a mp3 file. Then, I made a local json list of song inside my index.js file.

The dependencies need for this service are:

1. Node js -> npm init
2. Fastify -> npm install --save fastify
3. Type -> npm i -D typescript @types/node
4. Fastify-Cors -> npm install --save fastify-cors
5. Run the file -> node index.js

The first code helps me to build the node js package.json file where I will store the node js libraries and dependencies. The Fastify help me to run the server. The type helps me to work on the content type file like audio. The Fastify-Cors(*fastify/fastify-cors*, 2021) help to fetch the file in the html. The last code helps me to run the file so that my server will run in the localhost port 3000.

After running the server, I can see the result by using the following link which are show below with the screen shot:

This link is for the searching the song: http://localhost:3000/search?query=song

This link is for the playing the song: http://localhost:3000/songs/song_1.mp3

The first screenshot is for the searching the song and the second screenshot is for playing the song.

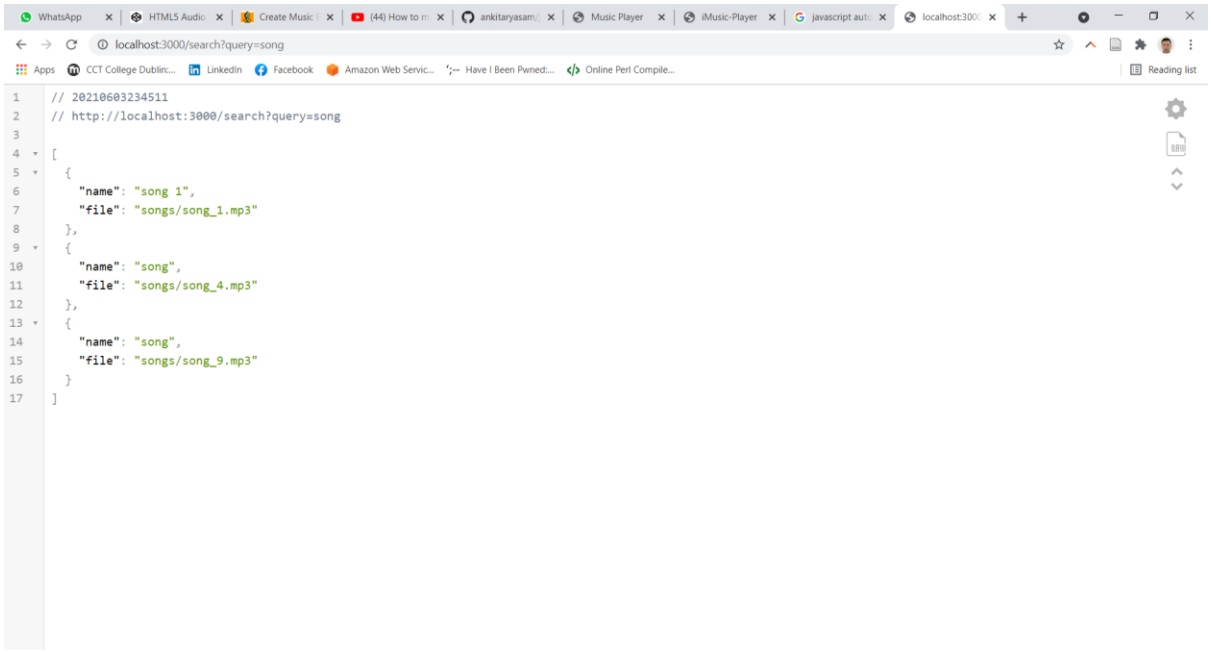


Fig: Searching the Song

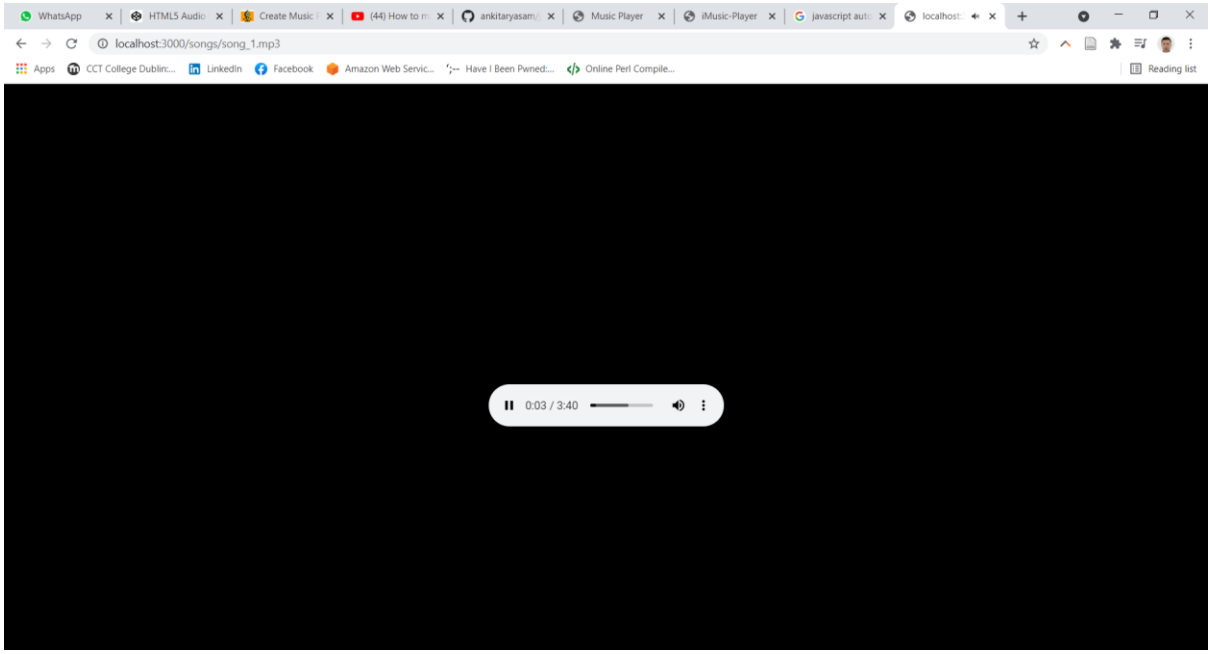


Fig: Playing the song

After that I made an interface folder and make a index.html and main.js file so that I can show the file in Frontend using the HTML and JavaScript.

The below screen shot show me that I can search and play the song.

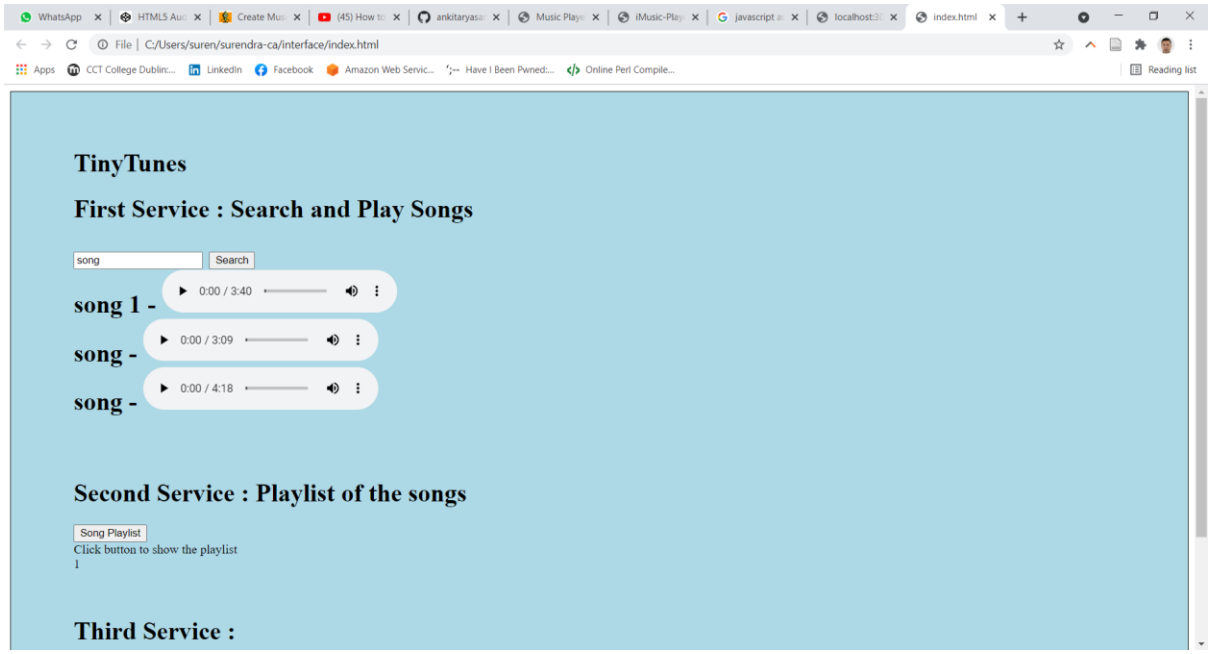


Fig: Search and play the song

Docker:

Then, I make a Dockerfile inside the file and make layer and build a docker container “search-and-play” by using this code “docker build ./ -t search-and-play”. Then, I run the container using this code “docker run -it -p 3000:3000 search-and-play”. Then, When I went to the index.html and run the file then I got the same result as above. But using this code, I cannot use the same command line, so I am going to use the detached mode which code is “docker run -d -p 3000:3000 search-and-play”. It helps me to run the container in detached mode.

Kubernetes:

Then, I made a deployment.yaml file and input the code and made a search-and-play-deployment using replicas 3 which make me 3 pods. Then, I made a search-and-play-service with the type of Load balancer then I got the result down in the screen shot.

Commands used for Kubernetes:

Deployment -> Kubectl apply -f deployment.yaml

Service -> Kubectl apply -f service.yaml

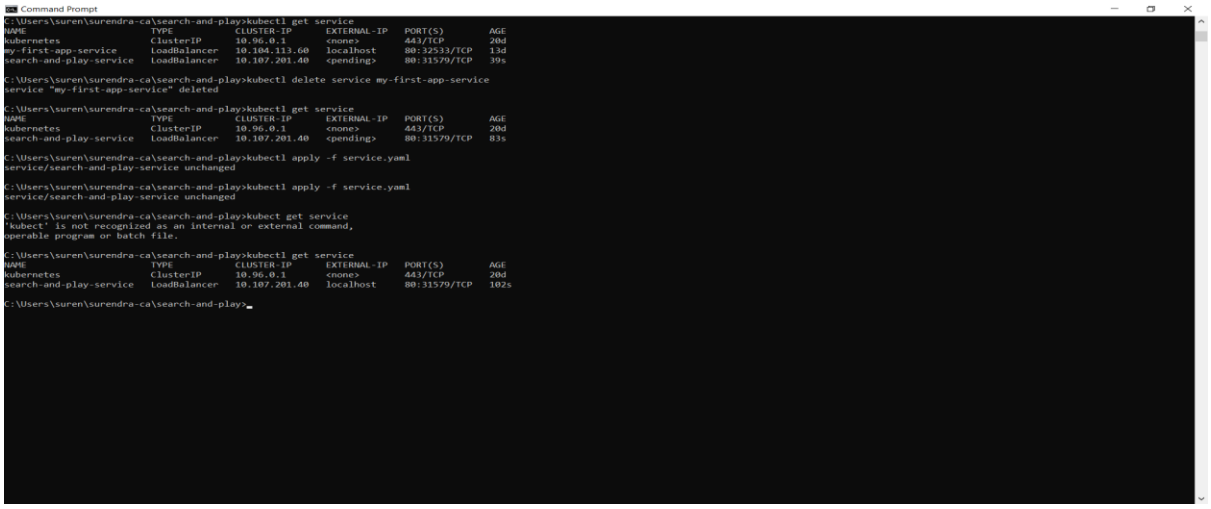


Fig: Working Pods, Deployment and Service

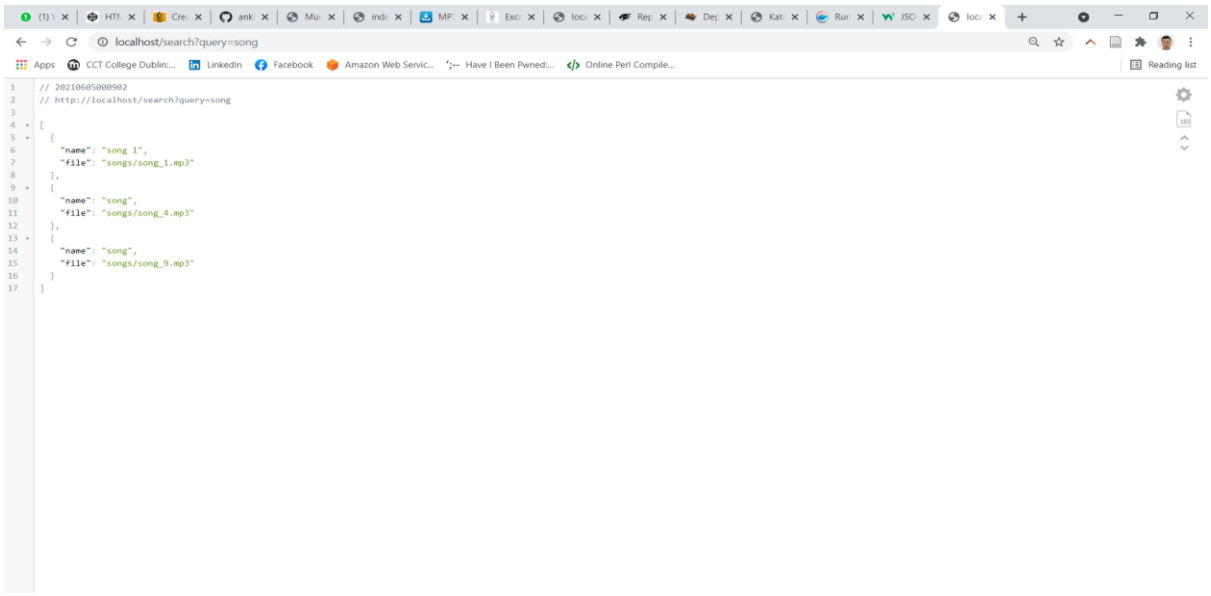


Fig: Working service

Create Playlists:

This is my second service creating playlists where I am going to show the list of song and play them. For this service, I used the same dependencies from the search-and-play songs. In this service also, I put the songs in the songs folder and then I made a local json file inside my main page called index.js and then I made a route with the playlist and then I called all the songs which I have. For this service, I used port 3001 because I already have search-and-play service in 3001. The below I have the screen shot of the service.

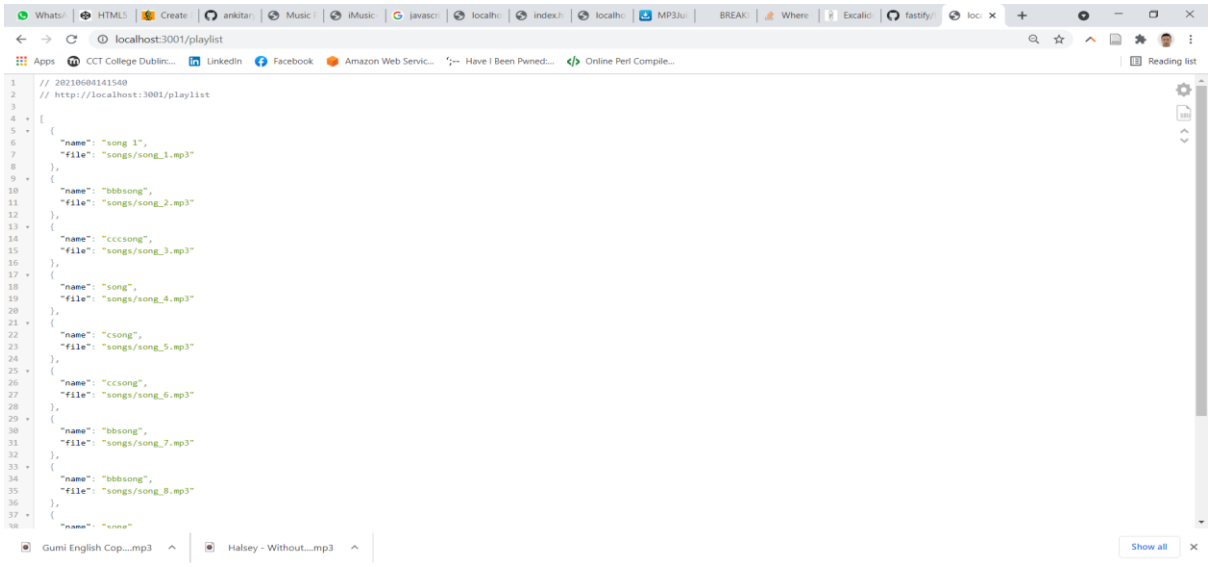


Fig: List of song

Then, I made a frontend using the html and JavaScript and the below I have a screenshot of playlist.

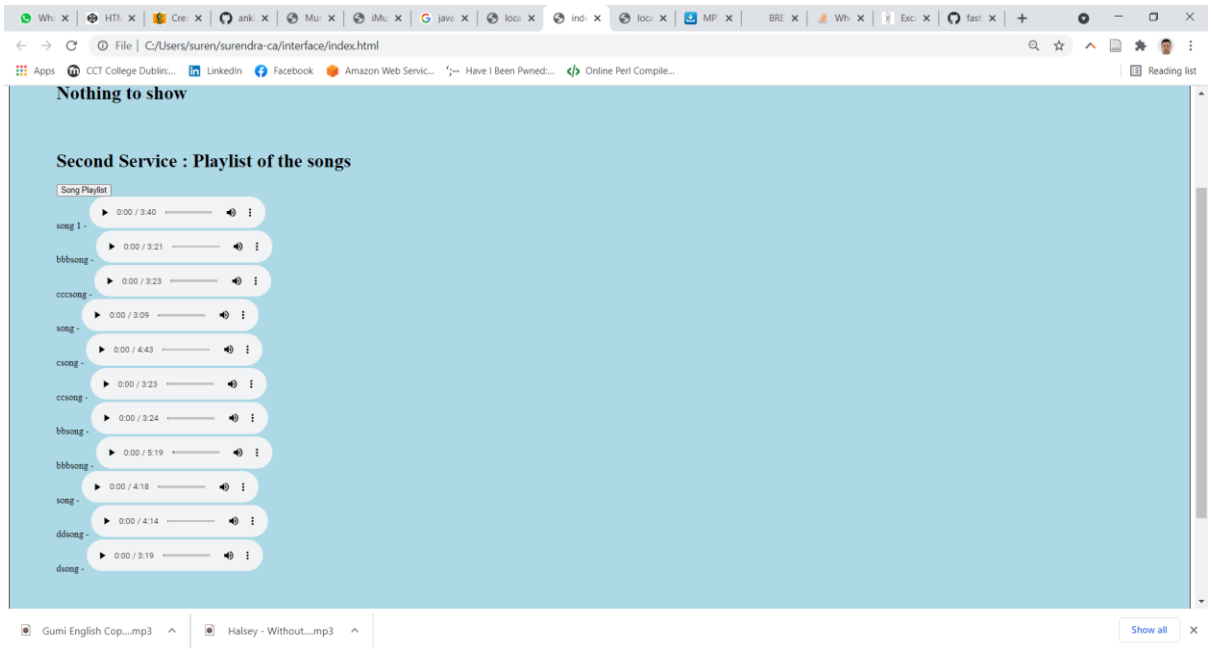


Fig: Playlist in the frontend

Docker:

Then, I made a Dockerfile inside this service folder and build a docker container using this code “docker build ./ -t playlist” and run the docker container using this code “docker run -d -p 3001:3001 playlist”, I got the same result as above screenshot.

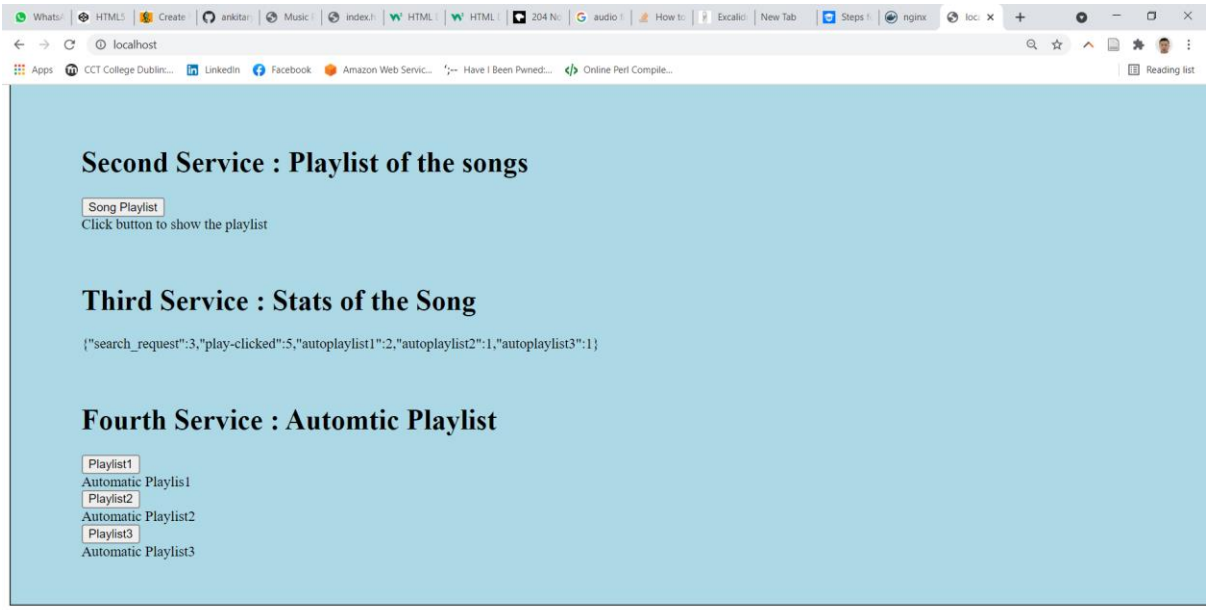
Kubernetes:

Then again, I made a deployment.yaml and service.yaml for the Kubernetes where I have a playlist pods and the service running. In this part also, I made type of Load balancer service.

Note: At this stage I am just showing the individual service to use the docker and the Kubernetes tools.

Record Stats about a song:

This is my third service where I am going to record and stats of the music player. For this service also, I have dependencies which I used before.



From this service, I used the code from the moodle (Gonzalez, 2021) and I display the count when the button is clicked on this interface. But I havenot count the song play and pause button to be count.

Docker:

Then, I made a Dockerfile inside this service folder and build a docker container using this code “docker build ./ -t stats” and run the docker container using this code “docker run -d -p 3002:3002 stats”, I got the same result as above screenshot.

Kubernetes:

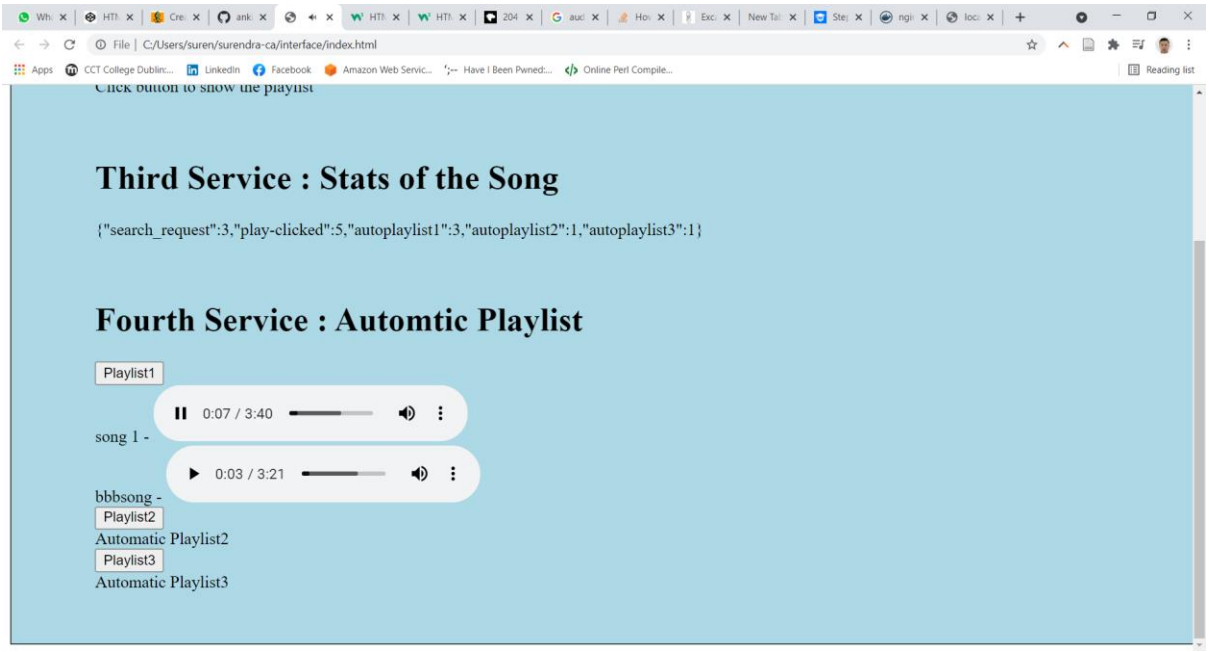
Then again, I made a deployment.yaml and service.yaml for the Kubernetes where I have a playlist pods and the service running. In this part also, I made type of Load balancer service.

Note: At this stage I am just showing the individual service to use the docker and the Kubernetes tools.

Create an automated playlist engine:

This is my last service where I have a list of songs and they will be automatically play when the one button is clicked. The below I have a screenshot of it.

I used the same code from the playlists service and make a 3 json list of song and 3 path so each path get the each json object. The song are autoplay suing the html audio autoplay function but I have problem here that all the song play at the same time.



Docker:

Then, I made a Dockerfile inside this service folder and build a docker container using this code “docker build ./ -t automatic-play” and run the docker container using this code “docker run -d -p 3003:3003 automatic-play”, I got the same result as above screenshot.

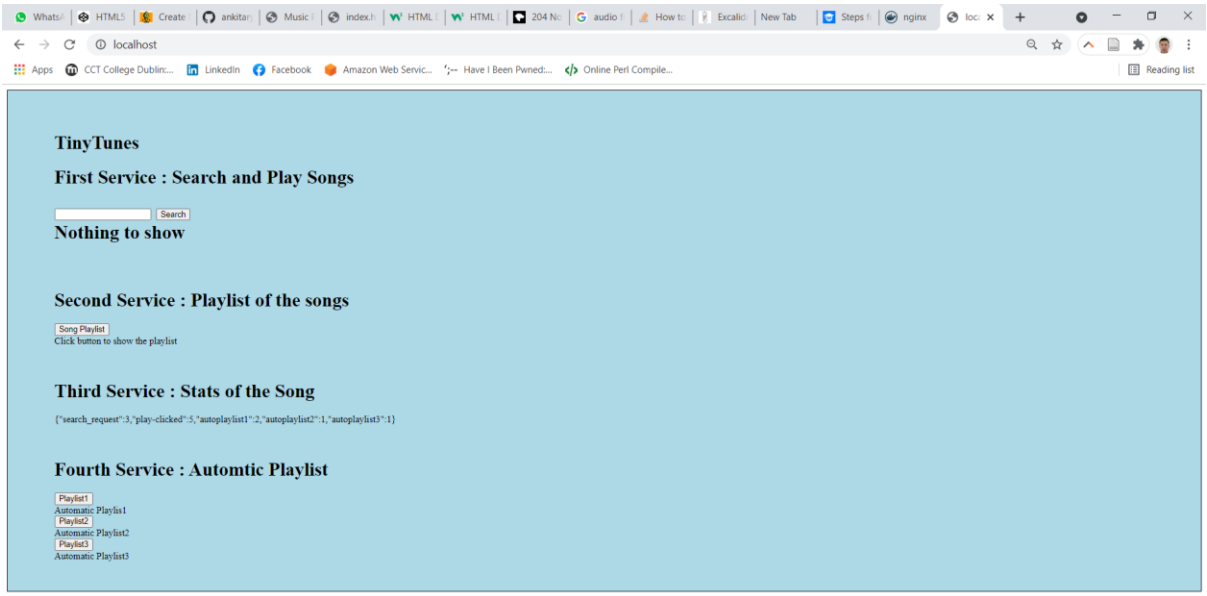
Kubernetes:

Then again, I made a deployment.yaml and service.yaml for the Kubernetes where I have a playlist pods and the service running. In this part also, I made type of Load balancer service.

Note: At this stage I am just showing the individual service to use the docker and the Kubernetes tools.

Frontend:

This is the folder which I have the frontend part(*Steps for Deploying a Static HTML Site with Docker and Nginx,* 2021) of my services for working: I used the nginx to make the Dockerfile. The below is the screenshot of my page.



Docker:

Then, I made a Dockerfile inside this service folder and build a docker container using this code “docker build -t html-server-image:v1 .” and run the docker container using this code “docker run -d -p 80:80 html-server-image:v1”, I got the same result as above screenshot.

Kubernetes:

Then again, I made a deployment.yaml and service.yaml for the Kubernetes where I have a playlist pods and the service running. In this part also, I made type of Load balancer service.

Note: At this stage I am just showing the individual service to use the docker and the Kubernetes tools.

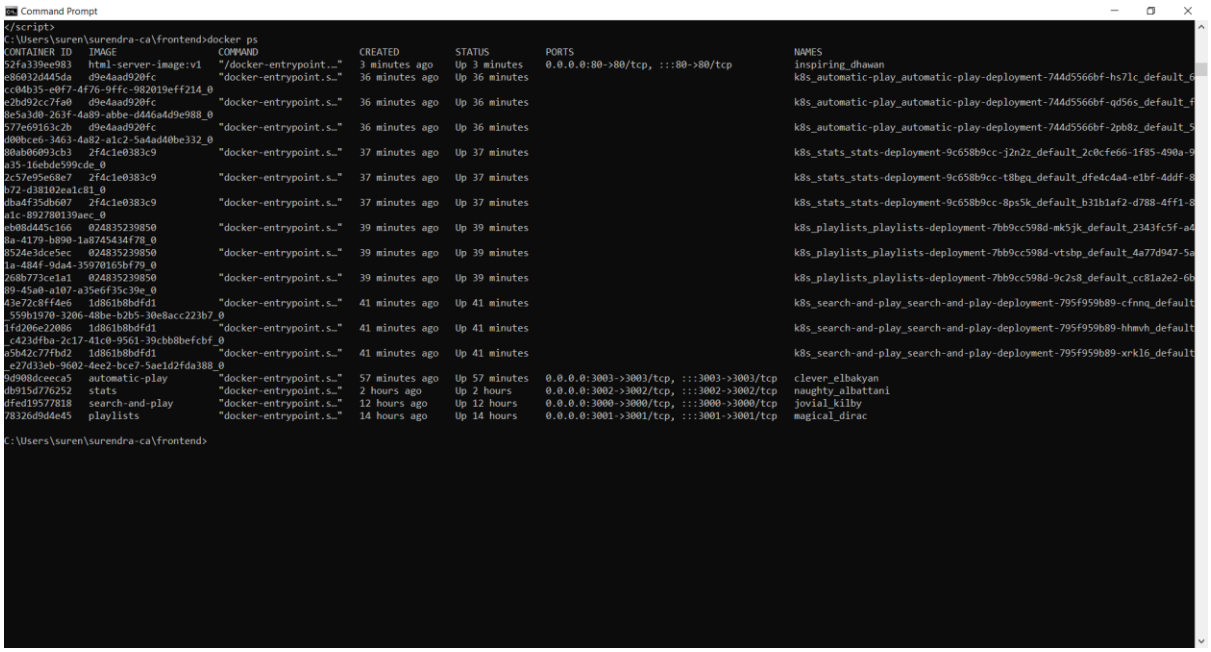


Fig: frontend Docker Running called html-server-images:v1

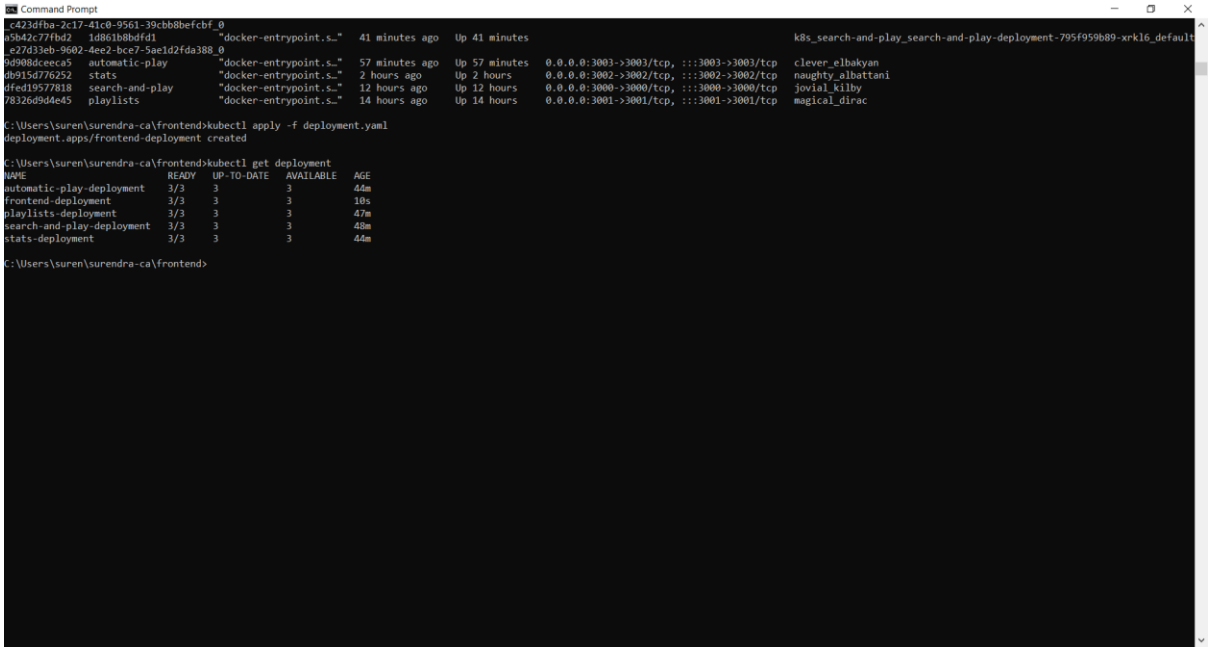
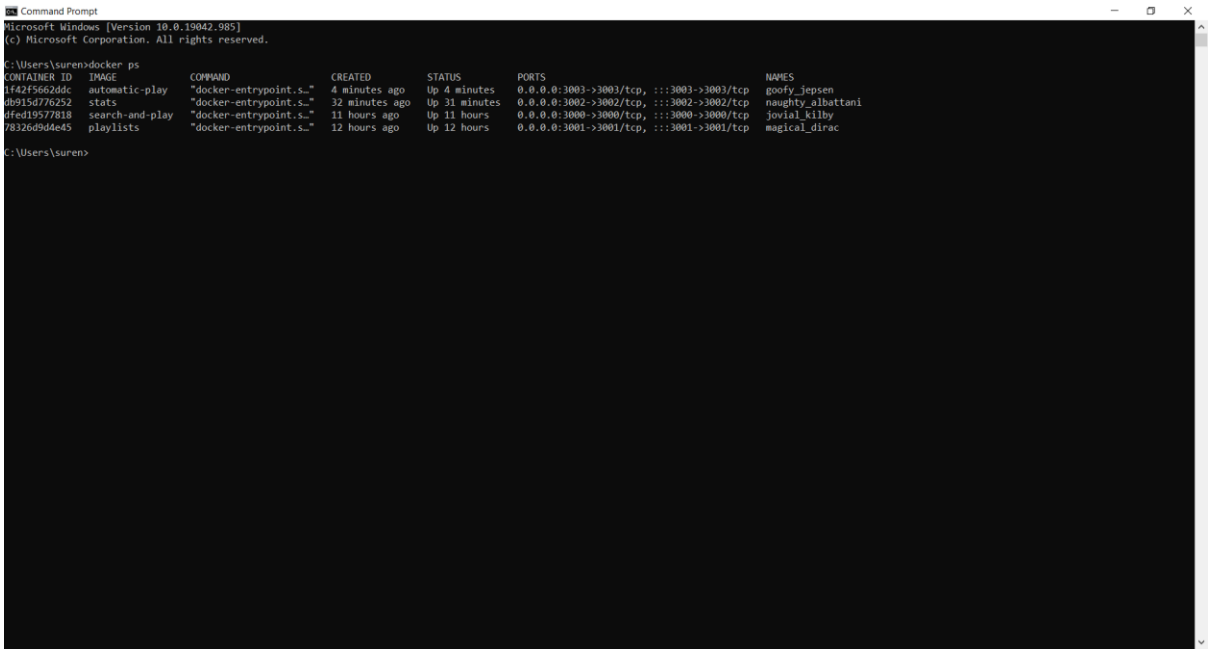


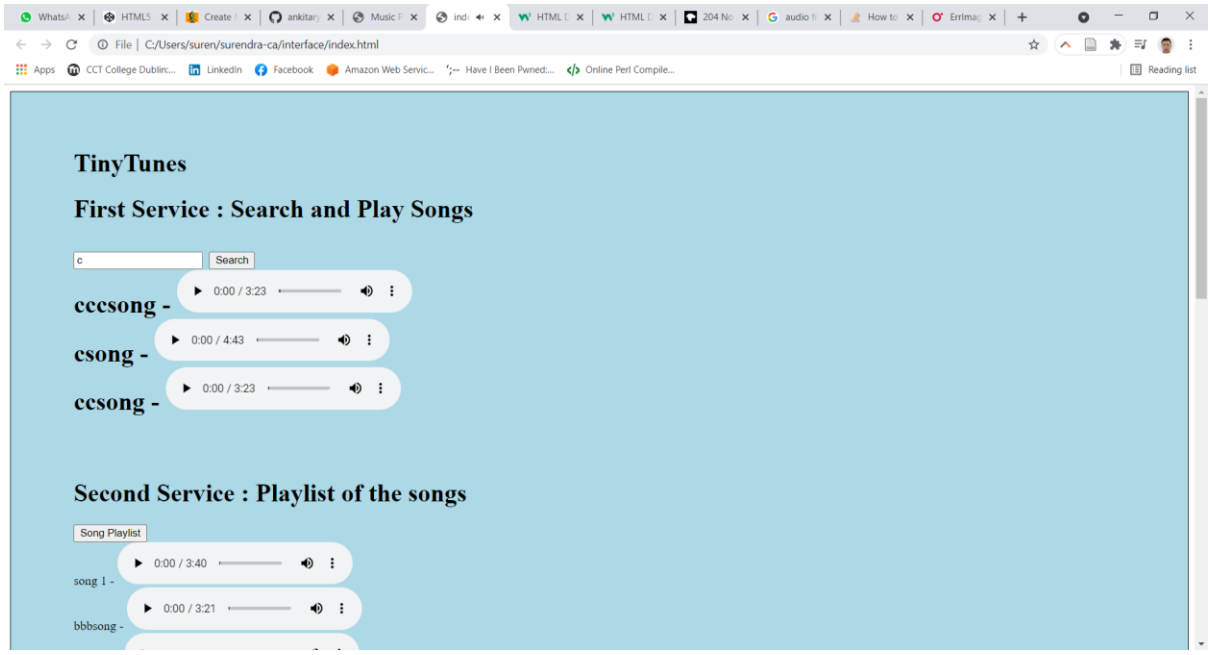
Fig: Frontend Deployment

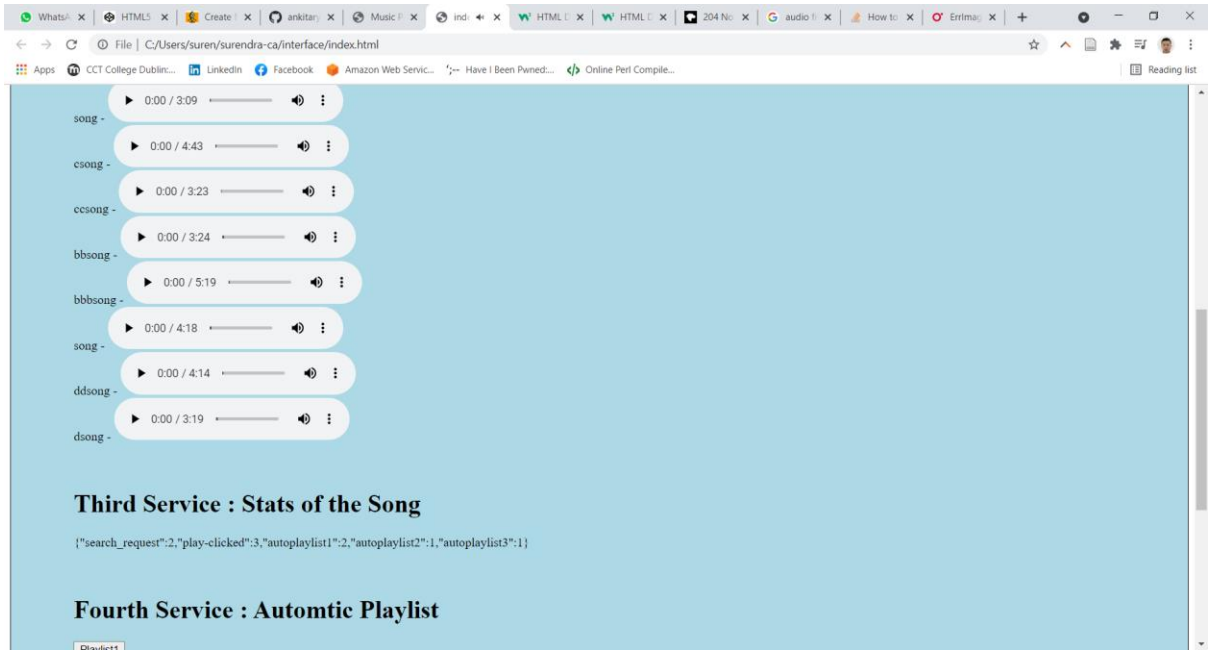
Appendix:

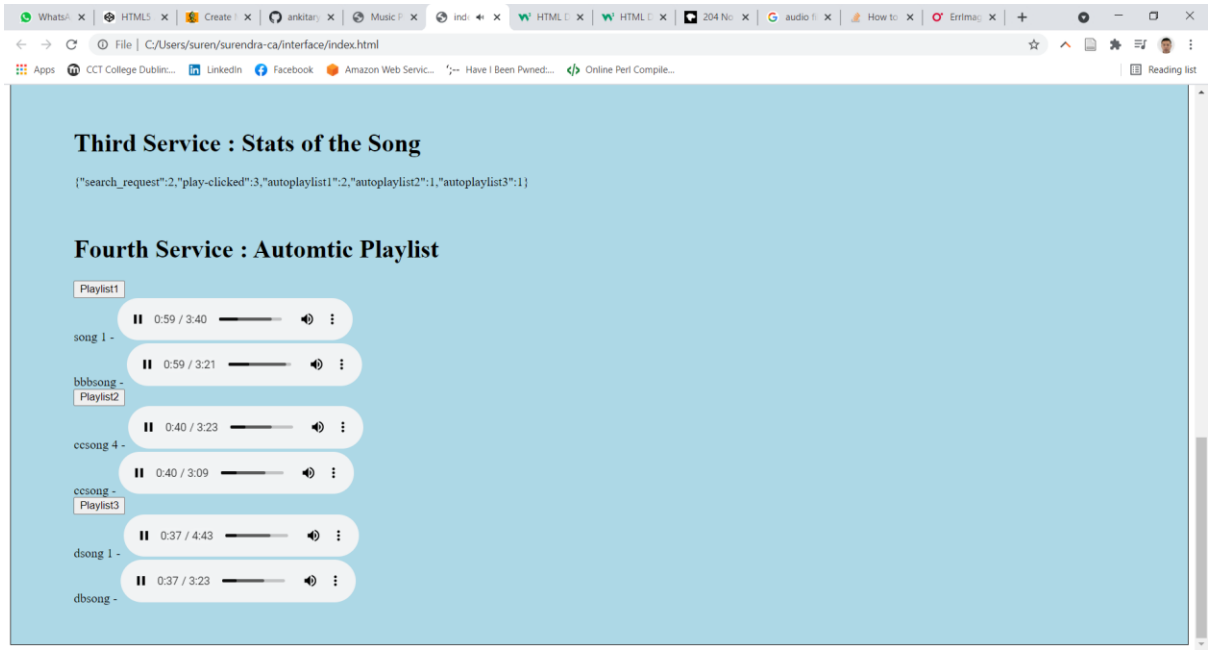
Docker Running Containers:



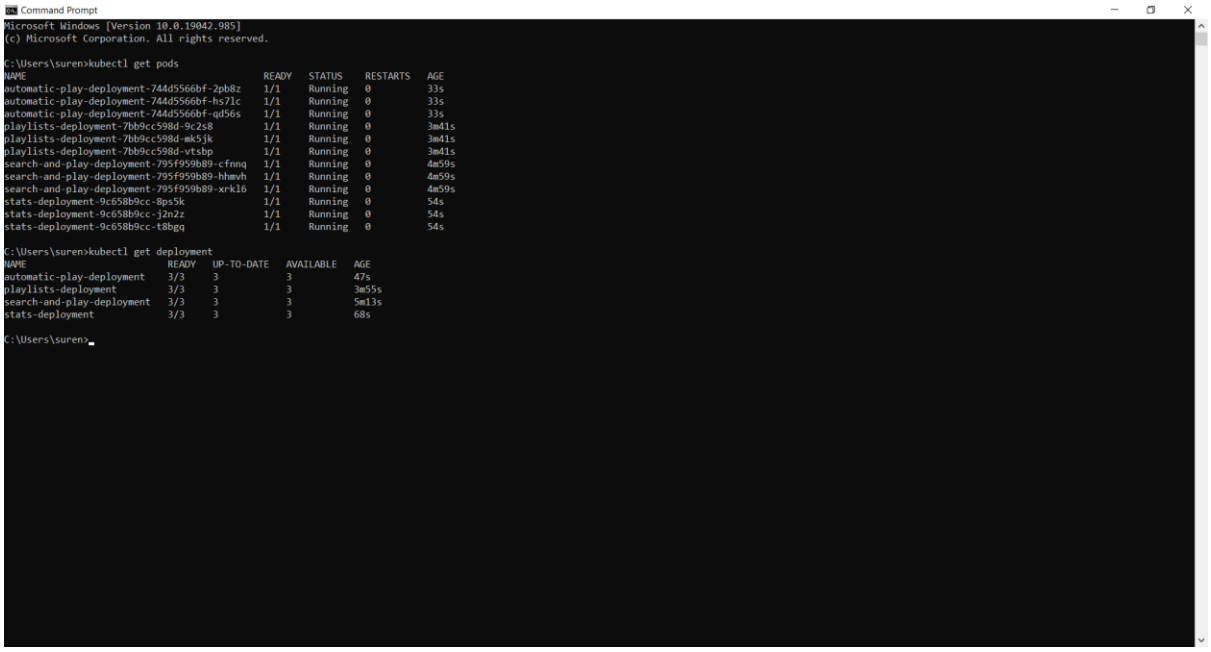
Frontend showing running:







Running Pods and Deployment of 4 Services:



References

Fastify, Fast and low overhead web framework, for Node.js (2021) *Fastify*. Available at: https://www.fastify.io/ (Accessed: 2 June 2021).

fastify/fastify-cors (2021) *GitHub*. Available at: https://github.com/fastify/fastify-cors (Accessed: 4 June 2021).

Gonzalez, D. (2021) *Cloud Services & Integration [Wed 12:00] – dgonzalez@cct.ie - Google Drive*. Available at: https://drive.google.com/drive/folders/1-bMtWxYjQssLye5Nzkfxvdo8IgFkQveh (Accessed: 2 June 2021).

MP3Juices - Free MP3 Downloads (2021). Available at: https://www.mp3juices.cc/ (Accessed: 4 June 2021).

Steps for Deploying a Static HTML Site with Docker and Nginx (2021). Available at: http://www.dailysmarty.com/posts/steps-for-deploying-a-static-html-site-with-docker-and-nginx (Accessed: 5 June 2021).