MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report
of
Internet Technologies Lab (CSE 3262)

MUSIC STREAMING WEBAPP

SUBMITTED
BY

Saurabh Kumar Mishra
200905314
C 53

Udeet Vinod Mittal
200905406
C 64

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal.

April 2023

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
28/04/2023

CERTIFICATE

This is to certify that the project titled **Music Streaming WebApp** is a record of the bonafide work done by **Saurabh Kumar Mishra (200905314), Udeet Vinod Mittal (200905406)** submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

1. **Prof. Ancilla Pinto, Assistant Professor, CSE Dept.**

TABLE OF CONTENTS

ABSTRACT

The music streaming web application is a platform for music lovers to listen to their favorite tracks and discover new music. The application is built using Django, a Python web framework, and SQLite, a lightweight database management system. The application is designed to be user-friendly and provides a seamless experience for music streaming.

CHAPTER 1: INTRODUCTION

In today's fast-paced world, music has become an essential part of our lives. However the current music streaming apps only allow previews of the songs for free. And we believe that music deserves to be free, to all. Thus, the idea for 'Freewave' webapp was born where a user can search for any song, add to liked songs and listen to it in its entirety without any annoying ads. The UI/UX was a strong focus for us while building the app. The project's benefits include access to a wide variety of music, a user-friendly interface, and customization options.

CHAPTER 2: PROBLEM STATEMENT & OBJECTIVES

The music industry has witnessed a significant shift towards digital platforms for music distribution and consumption. While several music streaming platforms like Spotify, Apple Music, and Pandora exist, there is a need for a customizable music streaming web application that provides a unique user experience.

The project aims to address this need by developing a music streaming web application using Django, JavaScript, jQuery, SQLite, and other technologies.

The music streaming web application allows users to create an account and log in to access their personalized music library. Users can browse through different playlists and their liked song to discover new music. The application also provides a search feature to find specific tracks, albums, or artists.

The application has an intuitive user interface with a modern design. It is also responsive and can be accessed from different devices, including desktops, laptops, and mobile devices.

The project is hosted on Railway, a cloud-based platform that provides a seamless deployment experience. Railway provides version control and collaborative features, making it easy for

multiple developers to work on the project simultaneously. The application is scalable and can be easily customized to add new features and functionalities. Overall, the music streaming web application is a comprehensive platform for music streaming that provides a unique user experience.

CHAPTER 3: METHODOLOGY

The website https://freewave.up.railway.app/ is a music streaming platform that allows users to listen to a wide variety of music for free. The website's methodology involves several steps, including data storage, user authentication, and music streaming. Here's a detailed description of the website's methodology:

1. Data Storage: The website uses SQLite, a lightweight database management system, to store user data, including user information, playlists, and music metadata. SQLite is a popular choice for web applications as it provides reliable data storage and retrieval and requires minimal configuration.

2. User Authentication: To ensure data security, the website requires users to create an account and log in to access the music library. The website uses Django's built-in authentication system to authenticate users and restrict access to unauthorized users. Users can create an account by providing their email address and creating a password.

3. Music Streaming: Once authenticated, users can access the music library and browse through different genres, playlists, and artists. The website uses Django to retrieve music metadata from the SQLite database and render the music library's user interface. The website uses JavaScript and jQuery to ensure smooth music streaming and playback. The website also provides a search feature, allowing users to find specific tracks, albums, or artists. The songs that are played on the website are directly taken from YouTube using python libraries and the playlists that are displayed on the home page are taken from a playlist already made on Spotify for convenience.

4. Playlists: The website allows users to create their own playlist and add tracks to them through the like button. The website uses Django to retrieve playlist data from the SQLite database and render the playlist's user interface. Users can add or remove tracks from their playlists using the website's user interface.

5. Hosting: The website is hosted on Railway, a cloud-based platform that provides a seamless deployment experience. Railway provides version control and collaborative features, making it easy for multiple developers to work on the project simultaneously.

In summary, the website's methodology involves using Django, SQLite, JavaScript, and jQuery to provide users with a seamless music streaming experience. The website's methodology ensures data security, smooth music streaming and playback, and provides users with features like personalized playlists. The website offers a library of royalty-free music that users can stream and download for free.

The outline of the project is –

.vscode	final commit	1 hour ago
app	final commit	1 hour ago
main	final commit	1 hour ago
static	final commit	1 hour ago
staticfiles	final commit	1 hour ago
templates	final commit	1 hour ago
.cache	final commit	1 hour ago
.replit	final commit	1 hour ago
README.md	Update README.md	1 hour ago
card.json	final commit	1 hour ago
cardupdate.py	final commit	1 hour ago
db.sqlite3	final commit	1 hour ago
manage.py	final commit	1 hour ago
requirements.txt	final commit	1 hour ago

Now we will explore all the parts.

For the app/settings.py we have made following changes –

settings.py

app > settings.py > ...

```python
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "2rue53h9#2220t(1c$tx)&-2=*i0n138ug5)5lq$17)vpi#4at"

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ["*"]

SECURE_REFERRER_POLICY = "no-referrer-when-downgrade"
# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "whitenoise.runserver_nostatic",
    "main",
]

```

settings.py

app > settings.py > ...

```python
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]
CSRF_TRUSTED_ORIGINS = ["https://freewave.up.railway.app"]
ROOT_URLCONF = "app.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [os.path.join(BASE_DIR, "templates")],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "app.wsgi.application"


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": os.path.join(BASE_DIR, "db.sqlite3"),
    }
}


```

Now for the main webapp in the app we have following files –

models.py 2

main > models.py > ...

```python
from django.db import models


# Create your models here.
class playlist_user(models.Model):
    username = models.CharField(max_length=200)

    def __str__(self):
        return f'Username = {self.username}, Liked Songs = {list(self.playlist_song_set.all())}'

class playlist_song(models.Model):
    user = models.ForeignKey(playlist_user, on_delete=models.CASCADE)
    song_title = models.CharField(max_length=200)
    song_youtube_id =  models.CharField(max_length=20)
    song_albumsrc = models.CharField(max_length=255)
    song_dur = models.CharField(max_length=7)
    song_channel = models.CharField(max_length=100)
    song_date_added = models.CharField(max_length=12)

    def __str__(self):
      return f'Title = {self.song_title}, Date = {self.song_date_added}'
```

admin.py 2

main > admin.py

```python
from django.contrib import admin
from .models import playlist_user, playlist_song
# Register your models here.
admin.site.register(playlist_user)
admin.site.register(playlist_song)
```

urls.py 3

main > urls.py > ...

```python
from django.urls import path
from . import views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path("", views.default, name='default'),
    path("signup/", views.signup, name="signup"),
    path("login/", views.login_auth, name="login_auth"),
    path("logout/", views.logout_auth, name="logout_auth"),
    path("playlist/", views.playlist, name='your_playlists'),
    path("search/", views.search, name='search_page')
]
```

views.py 6

main > views.py > ...

```python
from django.http.response import HttpResponse
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from .models import playlist_user
from django.urls.base import reverse
from django.contrib.auth import authenticate,login,logout
from youtube_search import YoutubeSearch
import json
# import cardupdate



f = open('card.json', 'r')
CONTAINER = json.load(f)

def default(request):
    global CONTAINER
    if request.user.is_anonymous:
        return redirect('/login')

    if request.method == 'POST':

        add_playlist(request)
        return HttpResponse("")

    song = 'siwpn14IE7E'
    return render(request, 'player.html',{'CONTAINER':CONTAINER, 'song':song})

def signup(request):
    context= {'username':True,'email':True}
    if not request.user.is_anonymous:
        return redirect('/')
    if request.method == 'POST':
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
```

views.py 6

main > views.py > ...

```python
        if (username,) in User.objects.values_list("username",) :
            context['username'] = False
            return render(request,'signup.html',context)

        elif (email,) in User.objects.values_list("email",):
            context['email'] = False
            return render(request,'signup.html',context)

        playlist_user.objects.create(username=username)
        new_user = User.objects.create_user(username,email,password)
        new_user.save()
        login(request,new_user)
        return redirect('/')
    return render(request,'signup.html',context)


def login_auth(request):
    if not request.user.is_anonymous:
        return redirect('/')
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        # print(User.objects.values_list("password",))

        user = authenticate(username=username, password=password)

        if user is not None:
            # A backend authenticated the credentials
            login(request,user)
            return redirect('/')

        else:
            # No backend authenticated the credentials
            context= {'case':False}
            return render(request,'login.html',context)

```

views.py 6

main > views.py > search

```python
    context= {'case':True}
    return render(request,'login.html',context)



def logout_auth(request):
    logout(request)
    return redirect('/login')


def playlist(request):
    if request.user.is_anonymous:
        return redirect('/login')
    cur_user = playlist_user.objects.get(username = request.user)
    try:
      song = request.GET.get('song')
      song = cur_user.playlist_song_set.get(song_title=song)
      song.delete()
    except:
      pass
    if request.method == 'POST':
        add_playlist(request)
        return HttpResponse("")
    song = 'siwpn14IE7E'
    user_playlist = cur_user.playlist_song_set.all()
    # print(list(playlist_row)[0].song_title)
    return render(request, 'playlist.html', {'song':song,'user_playlist':user_playlist})


def search(request):
  if request.method == 'POST':

    add_playlist(request)
    return HttpResponse("")
  try:
```

views.py 6

main > views.py > search

```python

def search(request):
  if request.method == 'POST':

    add_playlist(request)
    return HttpResponse("")
  try:
    search = request.GET.get('search')
    song = YoutubeSearch(search, max_results=10).to_dict()
    song_li = [song[:10:2],song[1:10:2]]
    # print(song_li)
  except:
    return redirect('/')

  return render(request, 'search.html', {'CONTAINER': song_li, 'song':song_li[0][0]['id']})



def add_playlist(request):
    cur_user = playlist_user.objects.get(username = request.user)

    if (request.POST['title'],) not in cur_user.playlist_song_set.values_list('song_title', ):

        songdic = (YoutubeSearch(request.POST['title'], max_results=1).to_dict())[0]
        song__albumsrc=songdic['thumbnails'][0]
        cur_user.playlist_song_set.create(song_title=request.POST['title'],song_dur=request.POST['duration'],
        song_albumsrc = song__albumsrc,
        song_channel=request.POST['channel'], song_date_added=request.POST['date'],song_youtube_id=request.POST['songid'])

```

Now to get the playlists we have cardupdate.py which stores the data in a json file:

cardupdate.py 3

cardupdate.py > ...

```python
from spotipy.oauth2 import SpotifyClientCredentials
import spotipy
from youtube_search import YoutubeSearch

PLAYLISTS = [
    [
        "eng",
        "https://open.spotify.com/playlist/7rE1ztUke1WZP2dAGh0Fex?si=6a5acdec3b5245ce",
        "PLtD4KkuvbBvCuvhdoTimMzpTQVJxVIhpB",
    ],
      [
        "Chill soft EDM",
        "https://open.spotify.com/playlist/4018yqdYFAGJq2tn0bYFCu?si=efaa1ad5a4b14cfe",
        "PLtD4KkuvbBvCUt9TtaKMd5P0-X7gcMxgl",
    ],
]
client_credentials_manager = SpotifyClientCredentials(
    client_id="4c2d042400724ba9a95fd33593beaa97",
    client_secret="fe85c1684601493e9b6294285a62d08d",
)
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)


CONTAINER = []
for playlist in PLAYLISTS:
    Name, Link, playlistid = playlist
    playlistcard = []
    count = 0
    PlaylistLink = "http://www.youtube.com/watch_videos?video_ids="
    for i in sp.playlist_tracks(Link)["items"]:
        if count == 50:
            break
        try:
            song = i["track"]["name"] + i["track"]["artists"][0]["name"]
            songdic = (YoutubeSearch(song, max_results=1).to_dict())[0]
            playlistcard.append(
```

cardupdate.py 3

cardupdate.py > ...

```python
        if count == 50:
            break
        try:
            song = i["track"]["name"] + i["track"]["artists"][0]["name"]
            songdic = (YoutubeSearch(song, max_results=1).to_dict())[0]
            playlistcard.append(
                [
                    songdic["thumbnails"][0],
                    songdic["title"],
                    songdic["channel"],
                    songdic["id"],
                ]
            )
            PlaylistLink += songdic["id"] + ","
        except:
            continue
        count += 1

    from urllib.request import urlopen

    req = urlopen(PlaylistLink)
    PlaylistLink = req.geturl()
    print(PlaylistLink)
    PlaylistId = PlaylistLink[PlaylistLink.find("list") + 5 :]

    CONTAINER.append([Name, playlistcard, playlistid])

import json

json.dump(CONTAINER, open("card.json", "w"), indent=6)
```

This is how the playlists are stored in card.json–

card.json

card.json > ...


```
[
    [
        "Mixtape",
        [
            [
                "https://i.ytimg.com/vi/GFSiisBYZ3U/hq720.jpg?sqp=-oaymwEjCOgCEMoBSFryq4qpAxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLBmlNYcb4
                "Nitty Gritty Dirt Band - American Dream",
                "brotherhamlet",
                "GFSiisBYZ3U"
            ],
            [
                "https://i.ytimg.com/vi/Hy_xJRbTq2Q/hq720.jpg?sqp=-oaymwEjCOgCEMoBSFryq4qpAxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLCr6DB_hQ
                "American Authors - Hit It (Audio)",
                "American Authors",
                "Hy_xJRbTq2Q"
            ],
            [
                "https://i.ytimg.com/vi/5DNrb132guY/hq720.jpg?sqp=-oaymwEjCOgCEMoBSFryq4qpAxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLAm0ad_Ms
                "Jamie N Commons - Marathon",
                "Jamie N Commons",
                "5DNrb132guY"
            ],
            [
                "https://i.ytimg.com/vi/yA281OuU3rk/hq720.jpg?sqp=-oaymwEjCOgCEMoBSFryq4qpAxUIARUAAAAAGAElAADIQj0AgKJDeAE=&rs=AOn4CLA-7D_tYz
                "Nine Inch Nails - Copy of a (VEVO Presents)",
                "Nine Inch Nails",
                "yA281OuU3rk"
            ],
            [
                "https://i.ytimg.com/vi/6LRN7qUmmYY/hq720.jpg?sqp=-oaymwE9COgCEMoBSFryq4qpAy8IARUAAAAAGAElAADIQj0AgKJDeAHwAQH4Af4JgALQBYoCDA
                "Rex Orange County -  Never Enough (Official Audio)",
                "Rex Orange County",
                "6LRN7qUmmYY"
            ],
            [
```


Now for the html files, we see the templates folder –

login.html

templates > login.html > html > body > div.signup_body

```html
{% load static %}
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" type="image/png" href="{% static 'favicon.ico' %}" />
    <meta name="description" itemprop="description"
        content="Play your playlists and favourites tracks, albums, and artists of YouTube with a Spotify styled UI" />
    <meta name="keywords"
        content="convert, convert playlists, transfer, syncing, smart links, free, apple music, streaming services, spotify, youtube" />

    <link rel="stylesheet" href="{% static 'formStyle.css' %}" />
    <title>Freewave - Because music deserves to be free</title>

</head>

<body>
    <!-- <a href="https://github.com/udeet27/Music-streaming-app" target="_blank" class="github-corner" aria-label="View source on GitHub">
    <div class="signup_header"><br> <a href="https://www.github.com/udeet27"
            style="text-decoration: none; color: #222326;" target="_blank"><span class="label" style="font-size: 23px;">F
                R E E W A V E</span></a><br>

    </div>
    <div class="signup_body">

        <div class="label center-align" style="text-align: center;color: #616467;margin-top: 15px;font-size: 13.5px;">

                <img src="{% static 'freewave-logo-removebg-preview.png' %}" alt="" width="100" height="100">
            <br><br>
            Because
            music deserves to be <span
                style="text-align: center;font-style: italic;color: #15883e;margin-top: 15px;font-size: 13.5px;">free</span>
        </div>
        <br><br>
        <form class="login_form" method='POST' enctype="multipart/form-data">
```

login.html

templates > login.html > html > body

```html
        <form class="login_form" method='POST' enctype="multipart/form-data">
            {% csrf_token %}
            <label for="login-username" class="label">
                Email address or username
            </label>

            <input type="text" class="form_input" pattern="[A-Za-z0-9.@_-e]+" name="username"
                placeholder="Email address or username" required>
            <label for="login-username" class="label">
                Password
            </label>
            <input type="password" class="form_input" name="password" placeholder="Password" required>
            {% if not case %}
            <span style="color: #f79862;
                        padding-bottom:20px;
                        text-align:center;">Username or password invalid</span>
            {% endif %}
            <button class='btn btn_log_in' type="submit">Log in</button>
            <div class="divider"></div>
            <p class='label center-align'>Don't have an account?</p>
            <a class='btn btn_sign_up' href="/signup">Sign Up for Freewave</a>
        </form><br>
        <div class="label center-align" style="font-size: 13px;color: #616467;">Made with &#9829; by <a
                href="https://www.github.com/udeet27" style="text-decoration: none;color: #15883e;"
                target="_blank">Udeet</a> & <a href="https://www.github.com/SaurabhSKM"
                style="text-decoration: none;color: #15883e;" target="_blank">Saurabh</a></div>
    </div>

    <!-- <div class="signin_body">

        </div> -->

</body>
</body>

</html>
```

The player.html file contains all the code for the main page and is the base for others-

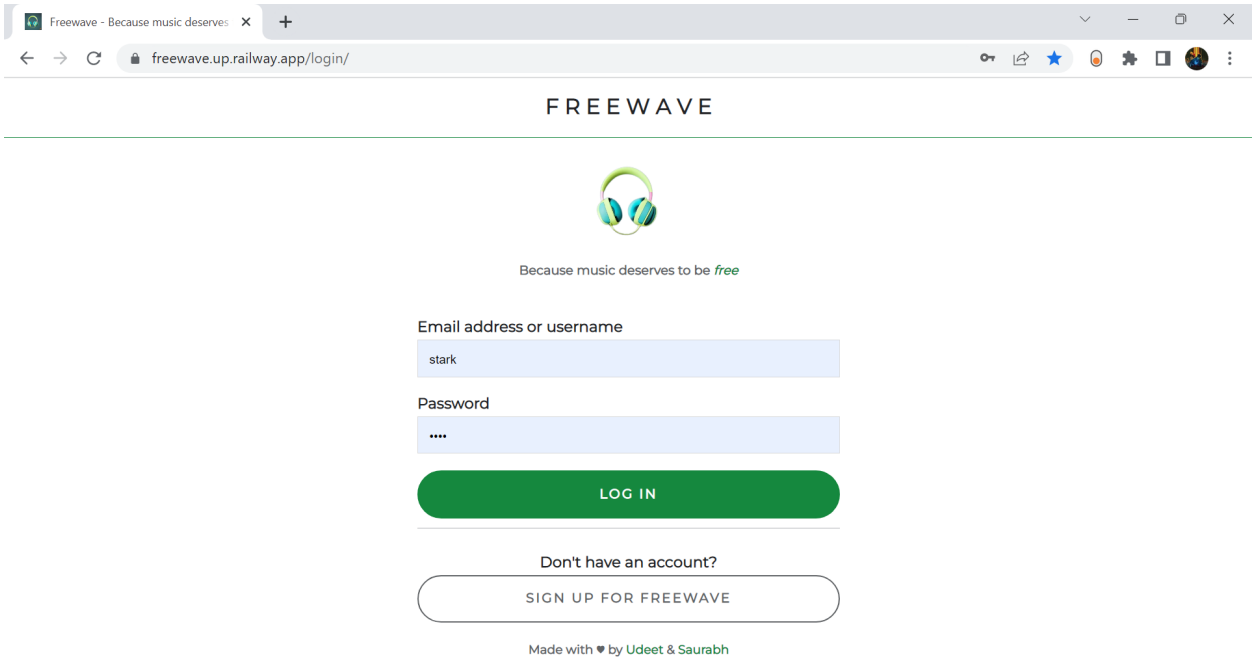
player.html ● | playlist.html | signup.html | site.webmanifest

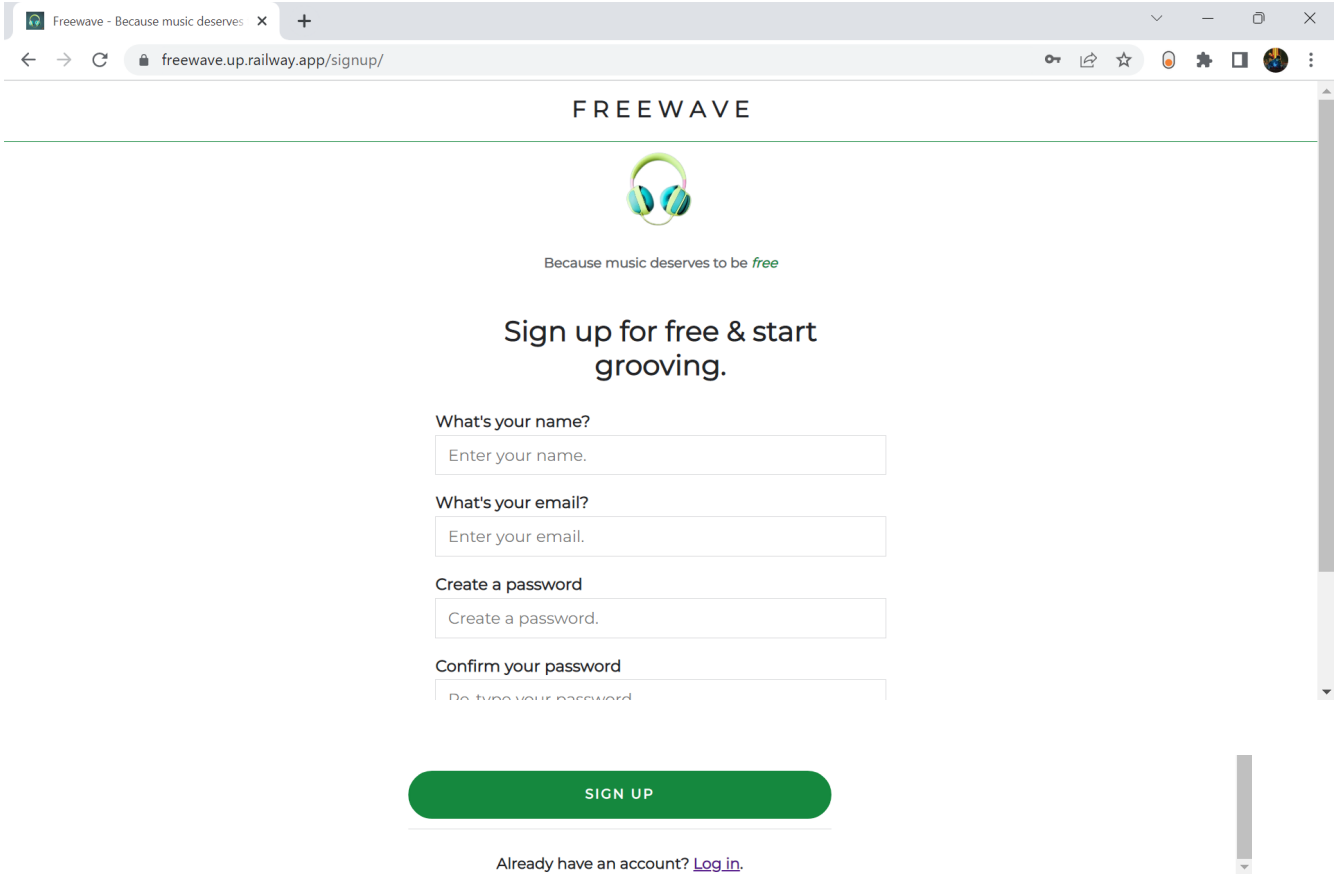
templates > player.html > html > body > script

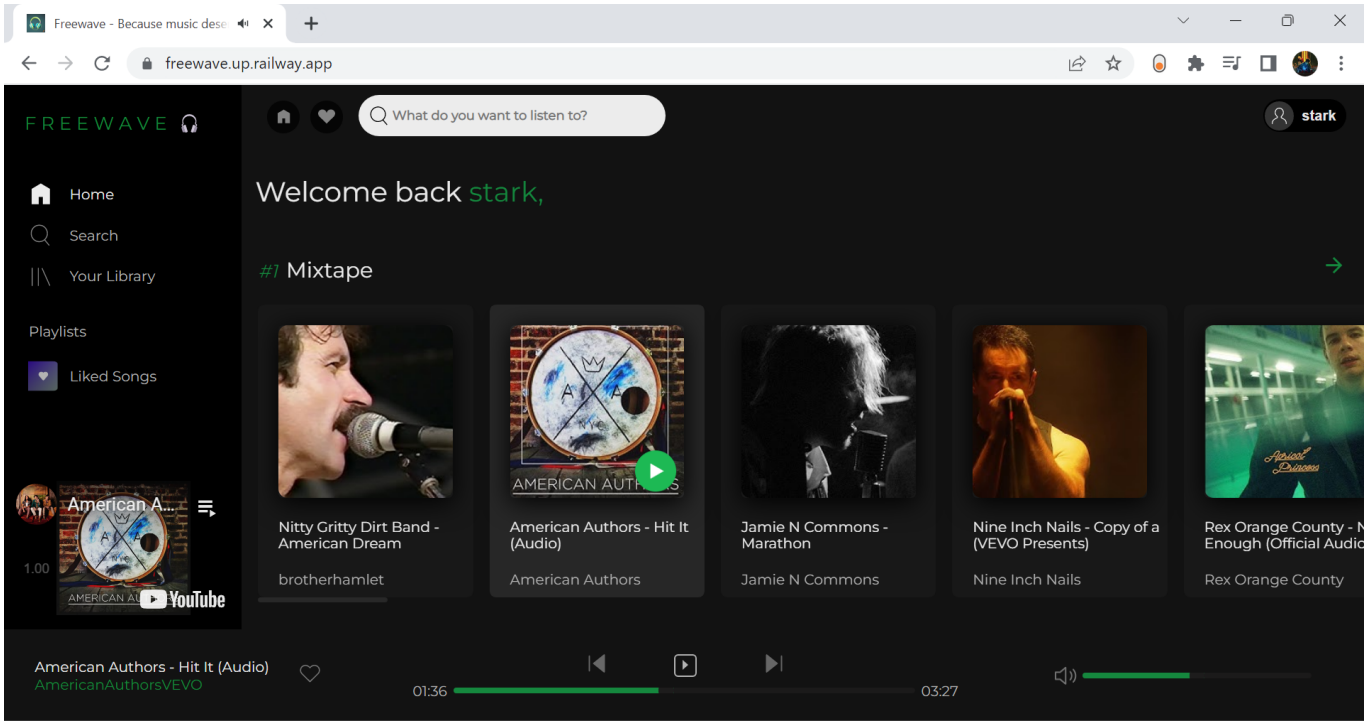
```html
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" type="image/png" href="{% static 'favicon.ico' %}" />
    <meta name="description" itemprop="description"
        content="Play your playlists and favourites tracks, albums, and artists of YouTube with a Spotify styled UI" />
    <meta name="keywords"
        content="convert, convert playlists, transfer, syncing, smart links, free, apple music, streaming services, spotify, youtube" />
    <link rel="stylesheet" href="{% static 'player.css' %}?{% now ' U ' %}"> {% block css %} {% endblock %}
    <title>Freewave - Because music deserves to be free</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js">
    </script>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/simplebar@latest/dist/simplebar.css">
    <link href='https://unpkg.com/css.gg@2.0.0/icons/css/play-button-r.css' rel='stylesheet'>
    <link rel="stylesheet"
        href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
    <script src="https://cdn.jsdelivr.net/npm/simplebar@latest/dist/simplebar.min.js"></script>
</head>

<body>

    <div class="grid-container">
        <div class="sidebar">
            <a href="https://www.github.com/udeet27/Music-streaming-app" style="text-decoration: none; color: #b3b3b3;"
                target="_blank"><span class="label" style="font-size: 18px; margin-top: 24px;margin-left: 20px;margin-bottom: 20px;color: #15883e;
                    R E E W A V E  🎧
                    <!-- <img src="{% static 'freewave-logo-removebg-preview.png' %}" alt="" width="100"
                    height="100" style="margin-left: 15px;"> -->
                </span>
            </a>

```

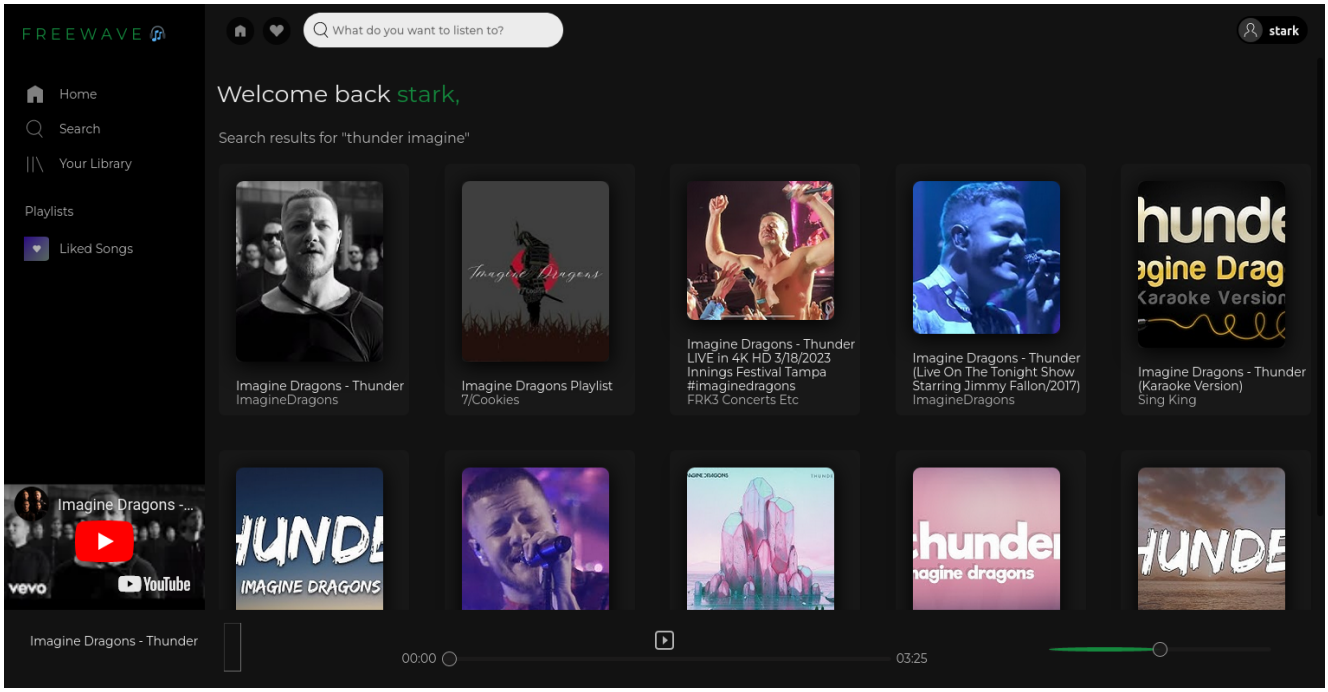
CHAPTER 4: RESULTS & SNAPSHOTS-

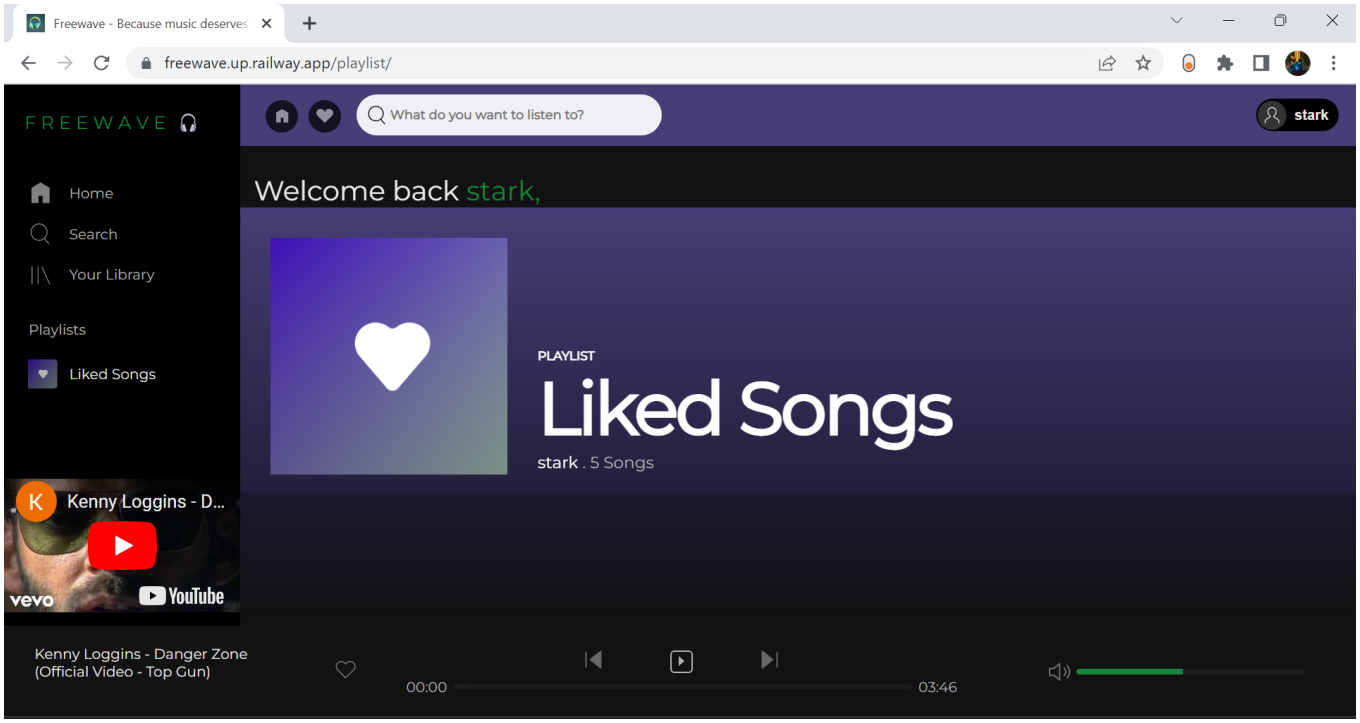
https://github.com/udeet27/Music-streaming-app

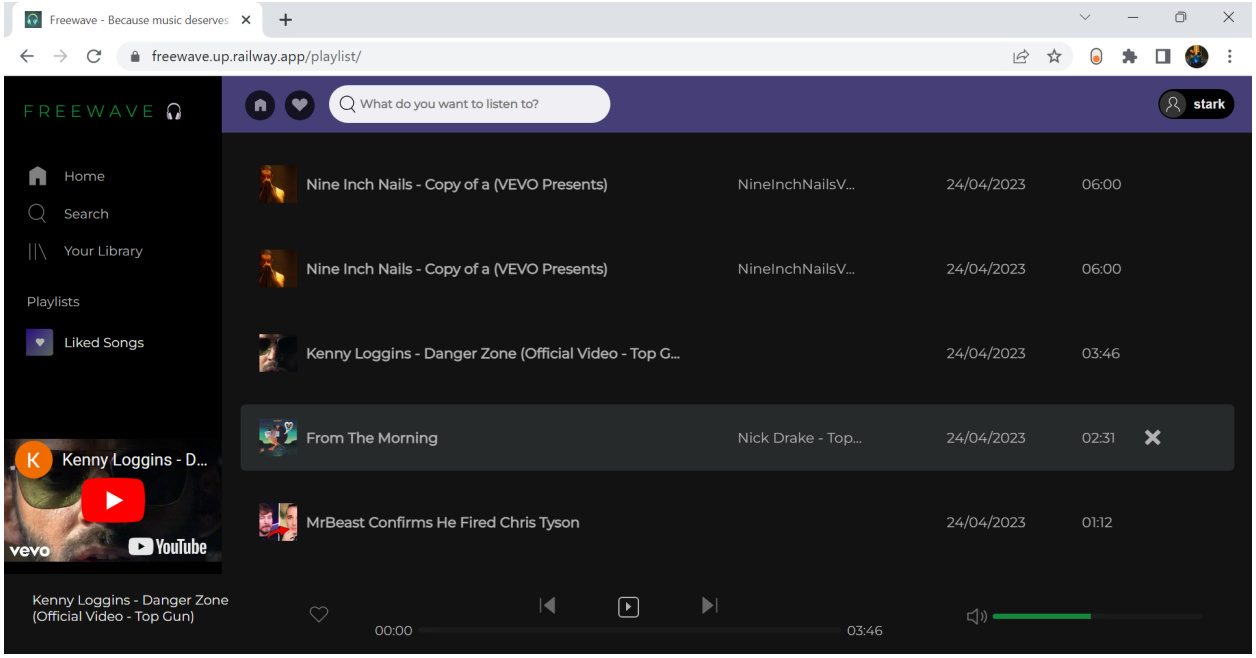



Freewave - Because music deserves
freewave.up.railway.app/signup/
FREEWAVE
Because music deserves to be free
Sign up for free & start grooving.
What's your name?
Enter your name.
What's your email?
Enter your email.
Create a password
Create a password.
Confirm your password
SIGN UP
Already have an account? Log in.

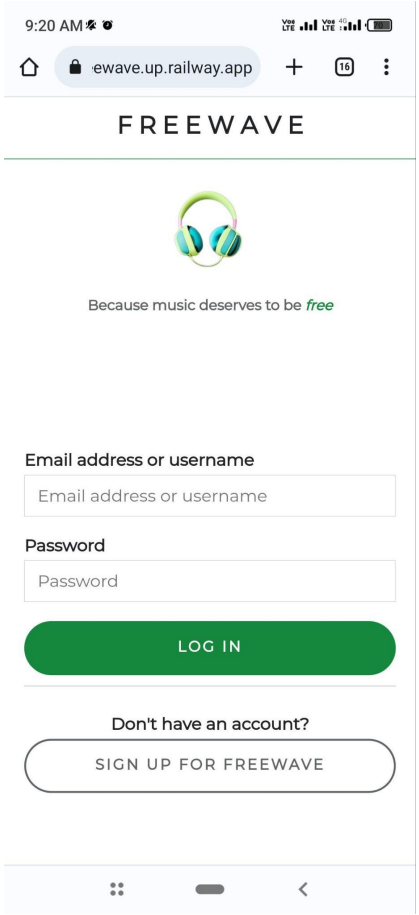

Freewave - Because music dese
freewave.up.railway.app
FREEWAVE
What do you want to listen to?
stark
Home
Search
Your Library
Playlists
Liked Songs
Welcome back stark,
#1 Mixtape
Nitty Gritty Dirt Band - American Dream
brotherhamlet
American Authors - Hit It (Audio)
American Authors
Jamie N Commons - Marathon
Jamie N Commons
Nine Inch Nails - Copy of a (VEVO Presents)
Nine Inch Nails
Rex Orange County
American Authors - Hit It (Audio)
AmericanAuthorsVEVO
01:36
03:27


FREEWAVE
What do you want to listen to?
stark
Home
Search
Your Library
Playlists
Liked Songs
Welcome back stark,
Search results for "thunder imagine"
Imagine Dragons - Thunder
ImagineDragons
Imagine Dragons Playlist
7/Cookies
Imagine Dragons - Thunder
LIVE in 4K HD 3/18/2023
Innings Festival Tampa
#imaginedragons
FRK3 Concerts Etc
Imagine Dragons - Thunder
(Live On The Tonight Show
Starring Jimmy Fallon/2017)
ImagineDragons
Imagine Dragons - Thunder
(Karaoke Version)
Sing King
Imagine Dragons -...
vevo
YouTube
Imagine Dragons - Thunder
00:00
03:25


Freewave - Because music deserves
freewave.up.railway.app/playlist/
FREEWAVE
What do you want to listen to?
stark
Home
Search
Your Library
Playlists
Liked Songs
Welcome back stark,
PLAYLIST
Liked Songs
stark . 5 Songs
Kenny Loggins - D...
vevo
YouTube
Kenny Loggins - Danger Zone
(Official Video - Top Gun)
00:00
03:46



Our website works also on mobile phones -



CHAPTER 5: CONCLUSION

Therefore we have successfully completed our Freewave webapp, a music streaming website, using the technologies learnt in the IT LAB. We hope you liked our application and any critical feedback is welcomed.

CHAPTER 6: LIMITATIONS & FUTURE WORK

Limitations-

- For now we have only added one playlist “Liked Songs” for each user.
- On the Home Dashboard only two playlists are available.
- For the authentication feature , no “forgot password” functionality has been added.
- Search results for songs do not just include the music results but other similar top results.

Future Work :

- Synced Lyrics functionality for each song .
- Functionality for Sharing song links on social media.
- Collaborative Playlists with multiple users access

CHAPTER 7: REFERENCES

https://github.com/joetats/youtube_search
https://docs.djangoproject.com/en/4.2/
https://spotipy.readthedocs.io/en/2.22.1/
https://railway.app/