NEW HORIZON PUBLIC SCHOOL & PENGUIN KIDS

COMPUTER SCIENCE

INVESTIGATORY PROJECT

YOUTTIFY

on demand music streaming app

BY -

NAME : AAYUSH SRIVASTAVA
STD : XII-A
ROLL NO : 41

NAME : AKHIL PADMANABHAN
STD : XII-A
ROLL NO : 16

NAME : PRANAV SUNIL
STD : XII-A
ROLL NO : 33

2020-2021

INDEX

ACKNOWLEDGEMENT

I wish to express my deep gratitude and sincere thanks to my Respected Principal Dr.Prashant Bukkawar of New Horizon Public School, New Panvel for his encouragement and for all the facilities that he provided for this project work on "**Youttify**" which also encouraged me to do a lot of research work and learn about new things. I extend my hearty thanks to my subject teacher Mrs.Rohini.

I take this opportunity to express my deep sense of gratitude for her invaluable guidance, useful suggestions and constant encouragement, which has sustained my efforts at all stages of this project work.I can't forget to offer my sincere thanks to my parents and friends who helped me to carry out this project work and thank them for their valuable advice and support, which I received from time to time.

CERTIFICATE

This is to certify that Mst. AKHIL C PADMANABHAN a student of class XII has successfully completed the Computer Science investigatory project as prescribed by the CBSE for academic session 2020-2021.

Signature of	Signature of	Signature of
Subject Teacher	Principal	External Examiner

SCHOOL STAMP

INTRODUCTION

YOUTTIFY is an online music stream web-app where user can listen to music as well as watch the music video and the user can create his/her own liked music playlist.

This application demonstrates the how python shines in the field of web-development. YOUTTIFY uses DJANGO, python web framework as its core server side operational unit. Youttify is an audio/video streaming application where the UI is completely inspired the famous music service provider SPOTIFY.

YOUTTIFY derives all the music i.e audio/video from the YOUTUBE API. It uses DJANGO auth for unique user identification and an SQL based database to keep the records of the user and their respective playlist.

 https://youttify.pythonanywhere.com/

 https://github.com/techakhilc47/spotify-clone-django

FILE STRUCTURE

- **main**
  - admin.py
  - models.py
  - urls.py
  - views.py
- **templates**
  - login.html
  - player.html
  - playlist.html
  - search.html
  - signup.html
- **static**
  - formStyle.css
  - player.css
  - playlist.css
- **youtify**
  - settings.py
  - urls.py
- manage.py
- card.json
- cardupdate.py
- db.sqlite3

SOURCE CODE

youtify\urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
]
```

main\urls.py

```
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')
]
```

urls.py handles the web page redirections. It listen for the web requests and return the desired response. It also validates the request by calling the reapective view functions.

youtify\settings.py

```
.....................................

ALLOWED_HOSTS = ['*']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main',
]

.....................................

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 = 'youtify.wsgi.application'

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


.....................................


STATIC_URL = '/static/'

STATIC_URL = '/static/'

STATICFILES_DIRS =[
    os.path.join(BASE_DIR, "static"),
]
```

settings.py contains configuration values that your web app needs to work; database settings, logging configuration, where to find static files, API keys if you work with external APIs, and a bunch of other stuff.

* only important lines are displayed above

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 = 'kSFJGEHDCrQ'
    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')



        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)


    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 = 'kSFJGEHDCrQ'
    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):
  try:
    search = request.GET.get('search')
    song = YoutubeSearch(search, max_results=8).to_dict()
    song_li = [song[:4],song[4:]]
    # 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]
        print
        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'])
```

In the views.py there are python function that takes a Web request and returns a Web response. Here we have implemented the logic required for the smooth functioning of the web app. This includes the login/signup authentication, adding user to the database, fetching the homepage contents, fetching the user playlists and implementation of the search feature.

main\models.py

```
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}'
```

Here it contains the custom models for the user and the user's playlist information, a model is the single, definitive source of information about one's data. It contains the essential fields and behaviors of the data you're storing. Each model maps to a single database table.

main\admin.py

```
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)
```

In admin.py, the custom made models are registered so the django can identify it and have an easy access so the it could be integrated wherever required with ease.

cardupdate.py

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

PLAYLISTS = [['Accidental','https://open.spotify.com/playlist/0zJ8hCBYJOcHYuk5nMPFmB?si=U6Kyom3XQ32reSuVgl2uhA',
'PL59eqqQABruMQOP1UVcVsIid685ZdwOjf'],
['TimePass','https://open.spotify.com/playlist/6gADLrLFK1kXgEEDsENi1c', 'PL59eqqQABruMSx6VSy1hbk0hG4XwtgSuy'],
['CHILLS','https://open.spotify.com/playlist/3zs3QOLX8bA5Y5oV2dmEQw', 'PL59eqqQABruN3GyAP1PnQ6Jq-TngWjT-Y'],
['Programming & Coding Music','https://open.spotify.com/playlist/6vWEpKDjVitlEDrOmLj1Aj',
'PL59eqqQABruNew500cRvomf9U6FI8RGyl'],
['Spanish','https://open.spotify.com/playlist/75QJ1JeFaeSm8uW1zmWxb8?si=Lt4kd-RARBu2TQz35RAQiQ',
'PL59eqqQABruM3TLAGthvgW10c1R6omKwq']
]
client_credentials_manager = SpotifyClientCredentials(client_id='{client_id}', client_secret='{client_secret}')
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([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 creates the data that are to be shown on the home page. Here it takes certain number of playlist links and fetches the data of the songs inside the playlist and creates a json, further which renders it in the player.html.

templates\player.html

```
.....................................
{% for playlist in CONTAINER %}
<div class="category">
   <div class="catTitle">
      <span class='label catName underline-on-hover'>{{ playlist.0 }}</span>
      <span class='label seeall underline-on-hover'>SEE ALL >></span>
   </div>
   <div class="cardContainer">
      {% for card in playlist.1 %}
      <div class="card c1">
         <div class="albumcover">
            <img class="albumimg" src="{{ card.0 }}" alt="" srcset=""></img>
            <div class="albumplay" onclick='playSong("{{ forloop.counter0 }}","{{ playlist.2 }}")'><img
               src="https://cdn.discordapp.com/attachments/760803738848133130/769476261181120734/image_1.png"
               alt=""></img></div>
         </div>
         <span class='label albumtitle bond-to-two-lines'>{{ card.1 }}</span>
         <span class='label albumauthor bond-to-two-lines'>By {{ card.2 }}</span>
      </div>
      {% endfor %}
   </div>
</div>
{% endfor %}
.....................................
```

Here this for loop iterate over each song data from the json return from the above file. It displays each song of each playlist inserting the respective song title, song author and the album image. Below code also works simialarly.

templates\search.html

```
{% extends 'player.html' %}
{% load static %}

{% block css %}
<link rel="stylesheet" href="{% static 'search.css' %}?{% now " U " %}">
{% endblock %}

{% block home %}
{% endblock %}

{% block search %}
active
{% endblock %}

{% block main %}
    <div class="homecontainer">
        {% for playlist in CONTAINER %}
        <div class="category">
            <div class="cardContainer">
                {% for card in playlist %}
                <div class="card c1">
                    <div class="albumcover">
                        <img class="albumimg" src="{{ card.thumbnails.0 }}" alt="" srcset=""></img>
                        <div class="albumplay" onclick='playSong("{{ card.id }}","none",true)'><img
                                src="https://cdn.discordapp.com/attachments/760803738848133130/769476261181120734
/image_1.png"
                                alt=""></img></div>
                    </div>
                    <span class='label albumtitle bond-to-two-lines'>{{ card.title }}</span>
                    <span class='label albumauthor bond-to-two-lines'>By {{ card.channel }}</span>
                </div>
                {% endfor %}
            </div>
        </div>
        {% endfor %}

    </div>
{% endblock %}
```

* only important lines are displayed above

templates\playlist.html

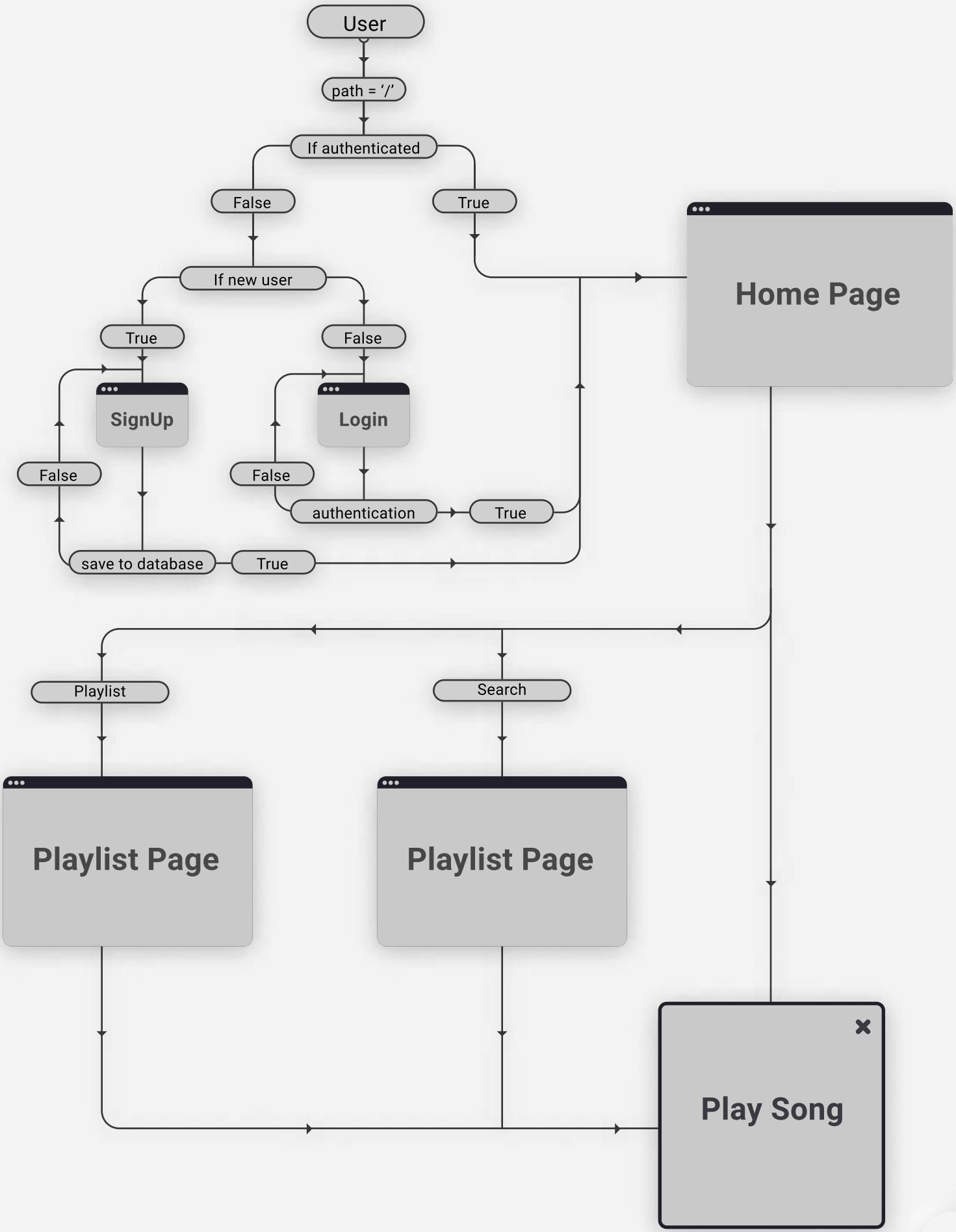
```
{% extends 'player.html' %}
{% load static %}
{% block css %}
<link rel="stylesheet" href="{% static 'playlist.css' %}?{% now " U " %}">
{% endblock %}
{% block home %}
{% endblock %}
{% block search %}
{% endblock %}
{% block liked-song %}
active
{% endblock %}
{% block main %}
<div class="playlist__banner">
   <div class="playlist__left">
      <div class="playlist__album">
         <img src="https://t.scdn.co/images/3099b3803ad9496896c43f22fe9be8c4.png" alt="" class="playlist__img">
      </div>
   </div>
   <div class="playlist__right">
      <div style='font-weight: 700;font-size: 12px;'>PLAYLIST</div>
      <div class="playlist__name">Liked Songs</div>
      <div>{{ request.user }}<span class="grey_text"> . {{user_playlist.count}} Songs<span></div>
   </div>
</div>
<div class="playlist__list">
   <div class="playlist__header" >
      <div class="playlist__headerTitle"># TITLE</div>
      <div class="playlist__headerAlbum">ALBUM</div>
      <div class="playlist__headerDate">DATE ADDED</div>
      <div class="playlist__headerDuration">
         <img src="https://img.icons8.com/pastel-glyph/344/clock--v1.png" alt="">
      </div>
   </div>
   <hr class="divider">
   <div class="playlist__songlist">
      {% for song in user_playlist %}
      <form action="" method="get">
         <div class="playlist__song playlist__row" onclick='playSong("{{ song.song_youtube_id }}","false","true")'>
            <div class="playlist__headerTitle">
               <img class="playlist__albumimg" src="{{song.song_albumsrc}}" alt =''/>
               <div class="playlist__songTitle ellipsis_one_line">
                  {{song.song_title}}
               </div>
               <textarea name="song" id="song_name_delete" cols="1" rows="1">{{song.song_title}}</textarea>
            </div>
            <div class="playlist__headerAlbum ellipsis_one_line">{{song.song_channel}}</div>
            <div class="playlist__headerDate">{{song.song_date_added}}</div>
            <div class="playlist__headerDuration">
               {{song.song_dur}}
               <button type="submit" class="delete"><img src="https://cdn.discordapp.com/attachments/760803738848133138
/777508678828951572/delete-icon-28.jpg" alt=""</button>
            </div>
         </div>
      </form>
      {% endfor %}
   </div>
</div>
{% endblock %}
```

This is an example how python django template inheritance works. playlist.html uses player.html as its base and adds the above code into at dynamically.

templates\login.html

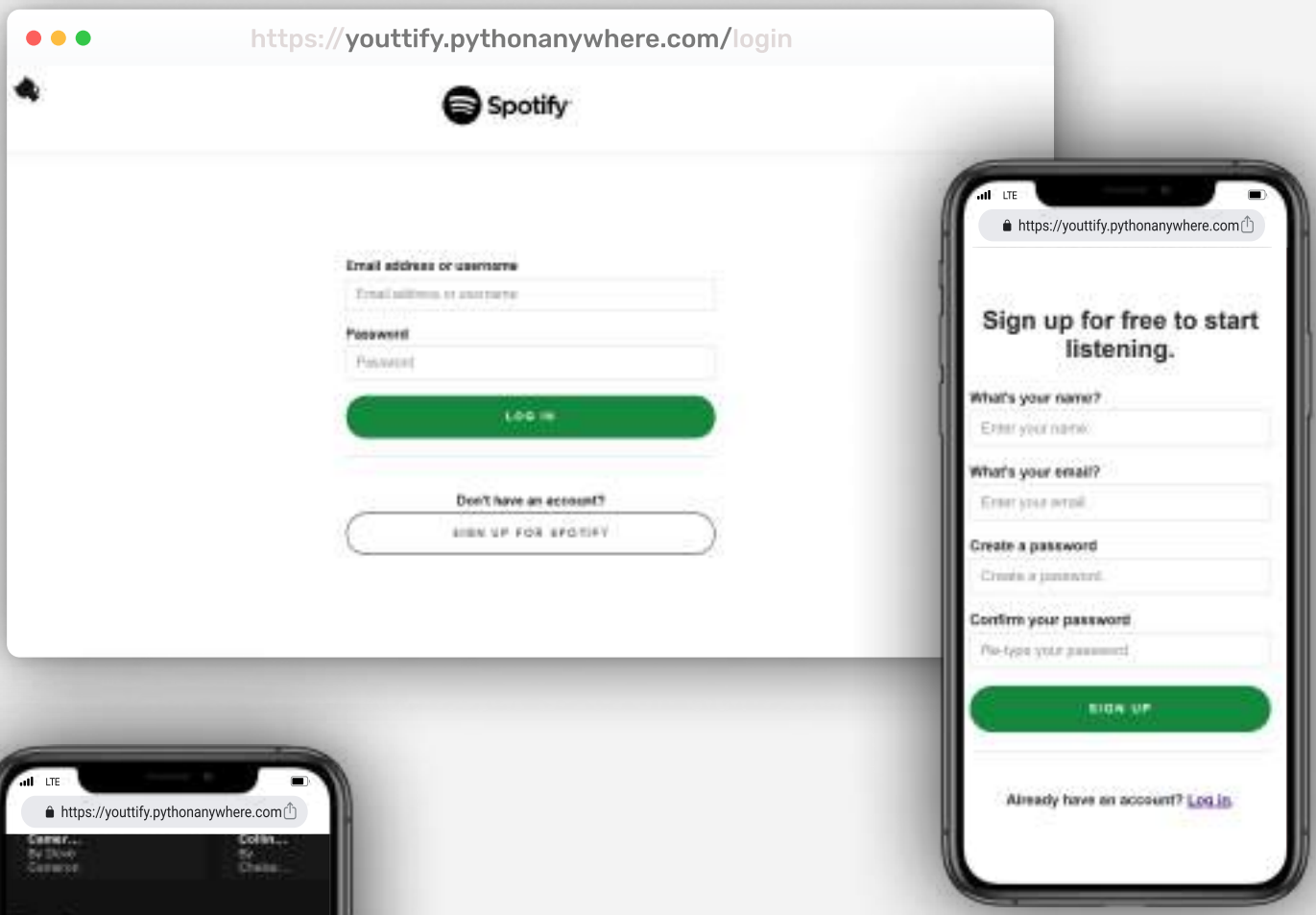
```
{% load static %}
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      {% block css %}
      <link rel="stylesheet" href="{% static "formStyle.css" %}">
      {% endblock %}
      <title>Spotify | Log-in</title>
   </head>
   <body>
      <div class="signup_header">
         <a href="" target="_blank">
         <img
            src="https://cdn.discordapp.com/attachments/768083738848133130
/768372618880483328/spotify.svg"></img>
         </a>
      </div>
      </a>
      <div class="signup_body">
         <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 spotify</a>
         </form>
      </div>
      <!-- <div class="signin_body">
         </div> -->
   </body>
   </body>
</html>
```

WORKING

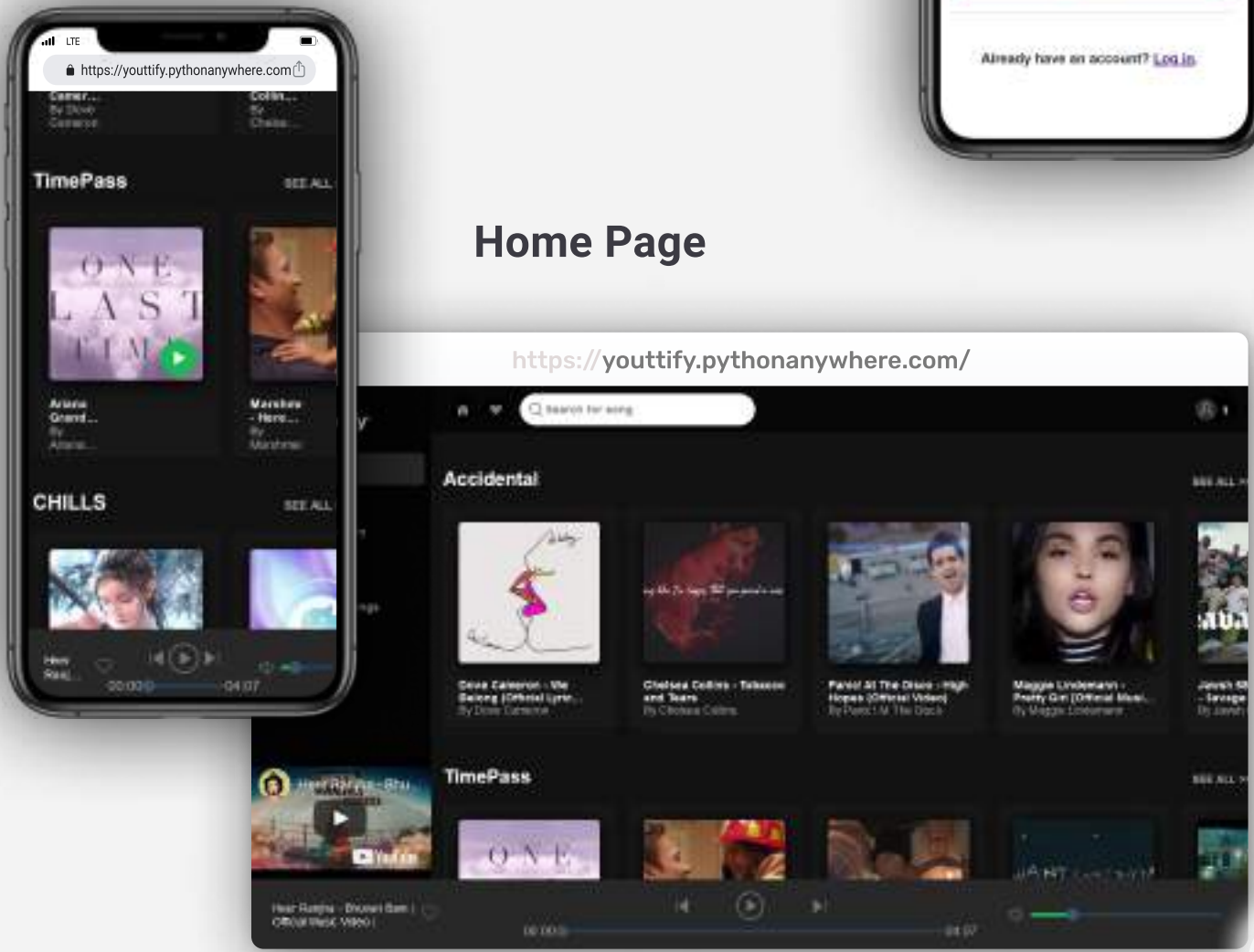

User
path = '/'
If authenticated
False
True
Home Page
If new user
True
False
SignUp
Login
False
False
authentication
True
save to database
True
Playlist
Search
Playlist Page
Playlist Page
Play Song

OUTPUT

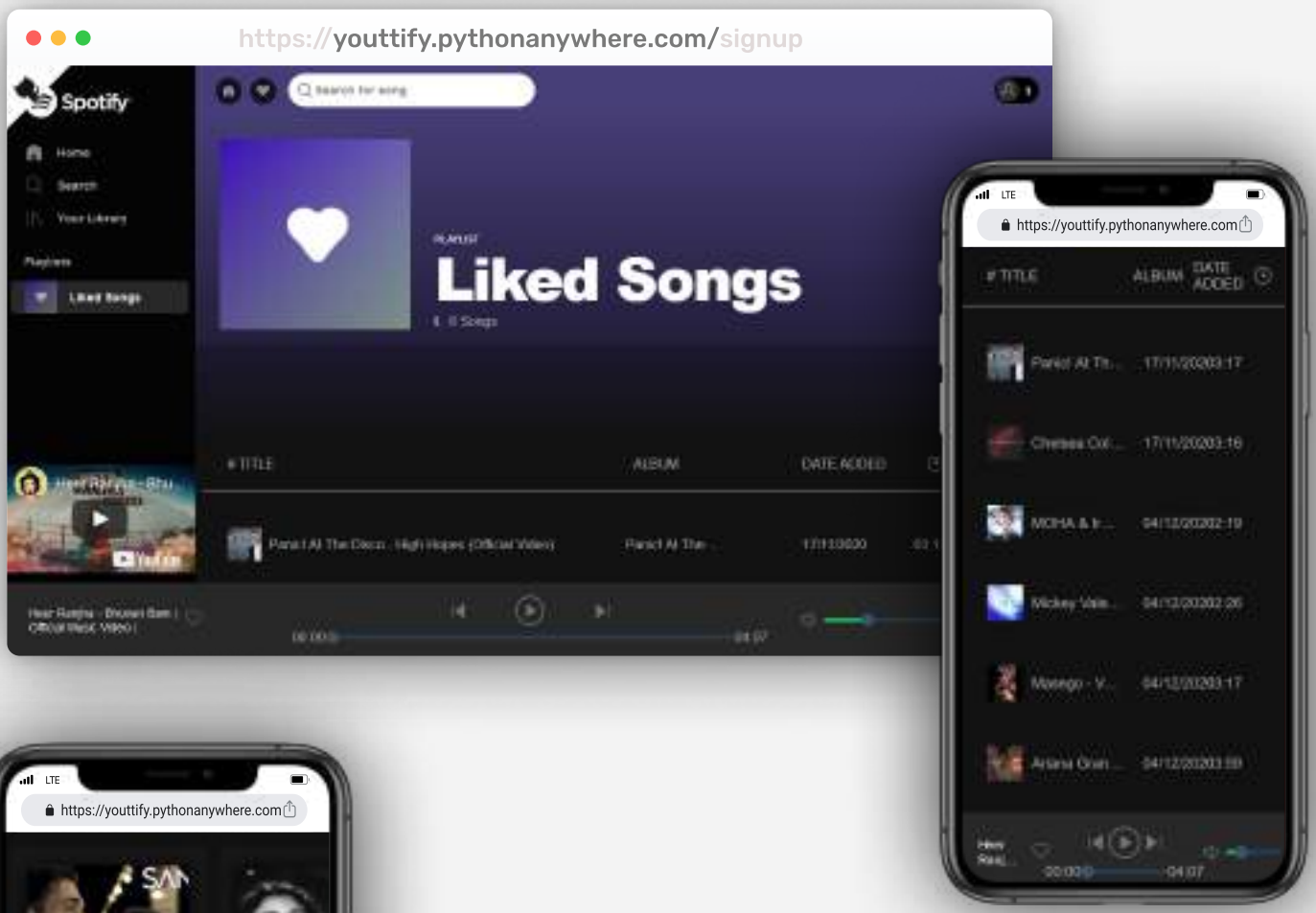
Login / SignUp Page



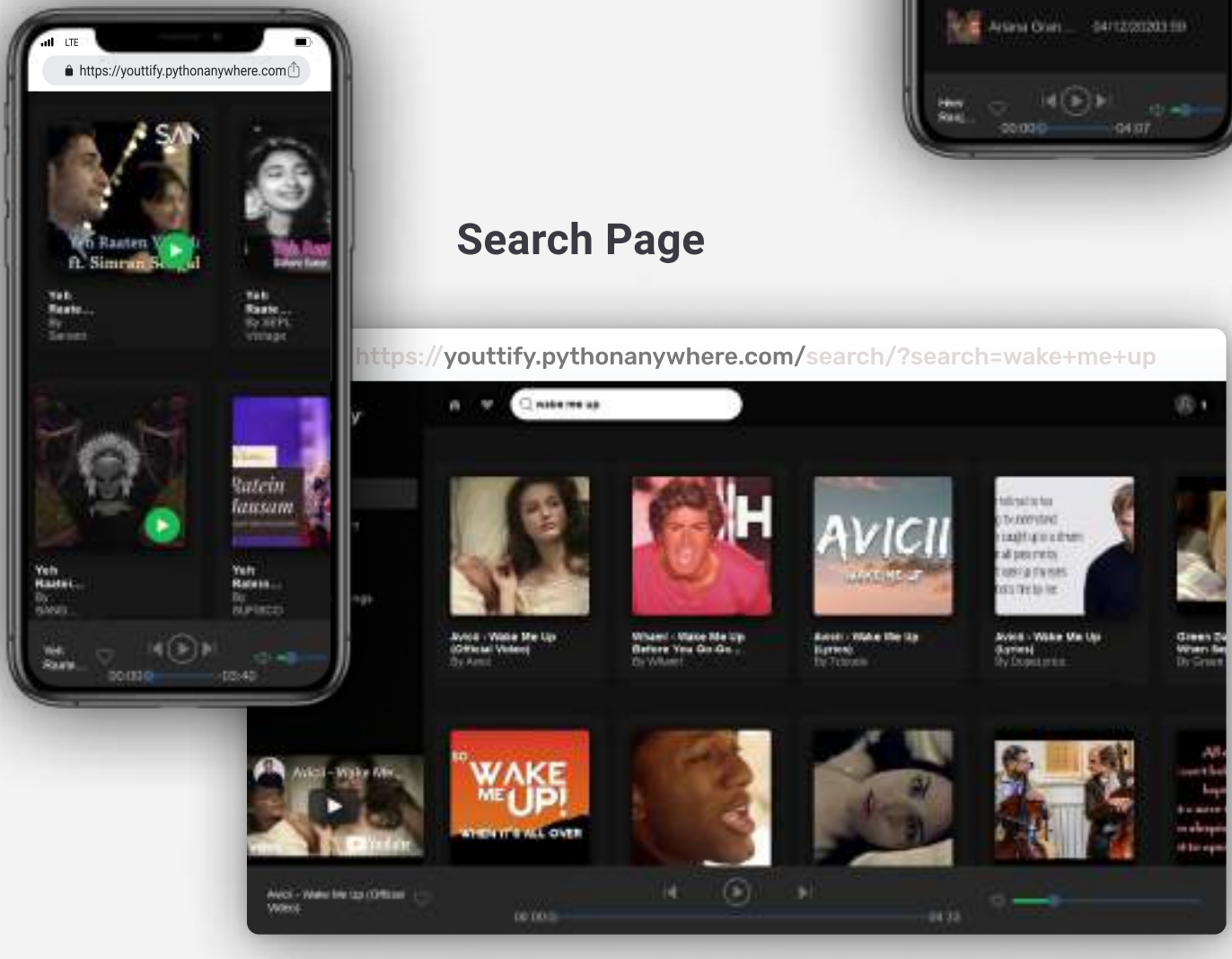
Home Page



Playlist Page



Search Page



SOFTWARE

Softwares used

- VSCODE - primary code editor
- Live Share - for collabrative work
- Discord - for communication and team work

Languages Used

- Python
- HTML
- CSS
- JavaScript

Testing and Debugging

- Repl - live server and live collabrative work

Hosting

- PYTHONANYWHERE

BIBLIOGRAPHY

- PYTHONANYWHERE documentation
- Django documentation
- Stackoverflow
- Youtube API documentation
- SpotifyAPI documentation
- Jquery documentation