|  |  |
| --- | --- |
| **Ex. No: 13 08.05.2025** | **SPOTIFY ANALYTICS** |

**AIM:**

To create a spotify analytics dashboard and perform CRUD operation using Flask and Python.

**SPOTIFY ANALYTICS DASHBOARD:**

* It is web-based application built using Flask, Bootstrap, SQL, JS that provides user’s spotify track data.
* The dashboard displayed various statistics such as the most popular data, average of popularity and popularity distribution
* This project aims to demonstrate data visualization and analytics techniques in a user-friendly dashboard form.

**CREATING ‘SPOTIFYTRACKS’ TABLE:**

SQL> CREATE TABLE spotifytracks(

1. id INTEGER AUTO INCREMENT PRIMARY KEY,
2. trackName VARCHAR(255) NOT NULL,
3. artist VARCHAR(255) NOT NULL,
4. album VARCHAR(255) NOT NULL,
5. popularity VARCHAR(255) NOT NULL,
6. durationMinutes FLOAT NOT NULL,
7. trackId VARCHAR(255) NOT NULL);

Table created.

# FRONT-END CODING: index.html (Home page)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Spotify Track Dashboard</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body class="bg-light">

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Spotify Dashboard</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bstarget="#navbarNav">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav me-auto">

<li class="nav-item">

<a class="nav-link active" href="/">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/add-track">Add Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/delete-track">Delete Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/update-track">Update Song Track</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container my-5">

<div class="text-center mb-5">

<h1 class="display-5 fw-bold">Spotify Analytics</h1>

<p class="text-muted">Overview of track popularity and duration analytics</p> </div>

<div class="card mb-4 shadow-sm">

<div class="card-body">

<h4 class="card-title"> Most Popular Track</h4>

<p class="mb-0"><strong>{{ top\_track[0] }}</strong> by {{ top\_track[1] }}</p> <small class="text-muted">Popularity: {{ top\_track[3] }}</small> </div>

</div>

<div class="row mb-4">

<div class="col-md-6">

<div class="card shadow-sm">

<div class="card-body">

<h5 class="card-title"> Average Popularity</h5>

<p class="fs-4">{{ avg\_popularity | round(2) }}</p>

</div>

</div>

</div>

<div class="col-md-6">

<div class="card shadow-sm">

<div class="card-body">

<h5 class="card-title"> Popularity Distribution</h5>

<canvas id="popularityChart" height="150"></canvas>

</div>

</div>

</div>

</div>

<div class="card mb-4 shadow-sm">

<div class="card-body">

<h5 class="card-title">⏱ Tracks Longer than 4 Minutes</h5>

<div class="table-responsive">

<table class="table table-bordered align-middle">

<thead class="table-light">

<tr><th>Name</th><th>Artist</th><th>Duration (min)</th></tr>

</thead>

<tbody>

{% for track in long\_tracks %}

<tr>

<td>{{ track[0] }}</td>

<td>{{ track[1] }}</td>

<td>{{ track[2] }}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

<div class="card shadow-sm">

<div class="card-body">

<h5 class="card-title"> All Tracks</h5>

<div class="table-responsive">

<table class="table table-striped align-middle">

<thead class="table-light">

<tr>

<th>ID</th><th>Track</th><th>Artist</th><th>Album</th><th>Popularity</th><th> Duration</th>

</tr>

</thead>

<tbody>

{% for track in all\_tracks %}

<tr>

<td>{{ track[0] }}</td>

<td>{{ track[1] }}</td>

<td>{{ track[2] }}</td>

<td>{{ track[3] }}</td>

<td>{{ track[4] }}</td>

<td>{{ track[5] }}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

</div>

<script>

const ctx = document.getElementById('popularityChart').getContext('2d'); const chart = new Chart(ctx, { type: 'bar', data: {

labels: {{ popularity\_distribution | map(attribute=0) | list | tojson }}, datasets: [{ label: 'Track Count',

data: {{ popularity\_distribution | map(attribute=1) | list | tojson }}, backgroundColor: '#4caf50'

}]

},

options: { responsive: true,

plugins: { legend: { display: false },

tooltip: { enabled: true }

}, scales: {

y: {

beginAtZero: true,

ticks: { stepSize: 1 }

}

}

}

});

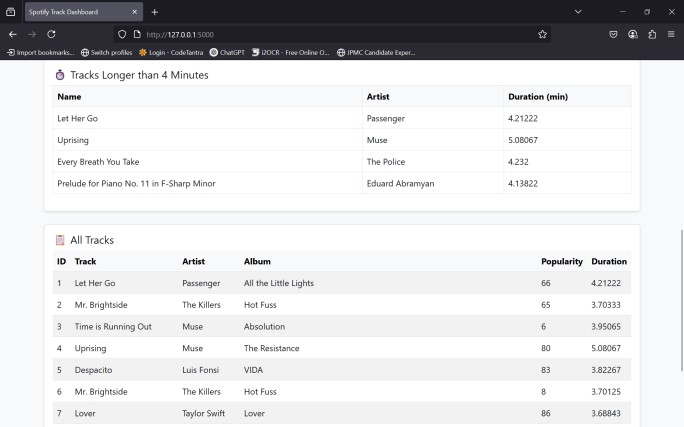
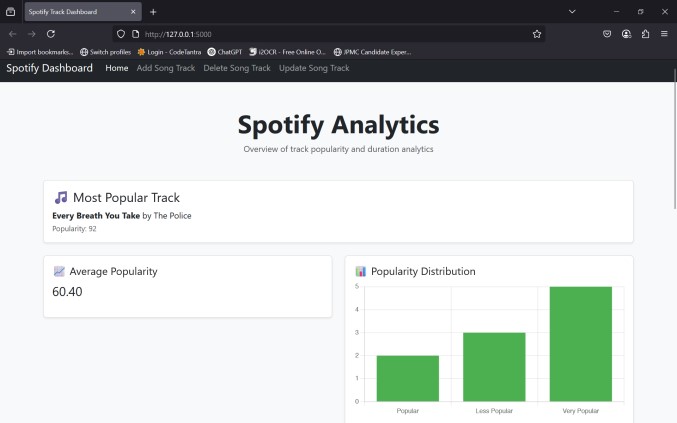
</script>

<script

src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script> </body>

</html>

# Output (Home page)



# add\_track.html (Adding track page)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Add Spotify Track</title>

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"> </head>

<body class="bg-light">

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Spotify Dashboard</a>

<div class="collapse navbar-collapse">

<ul class="navbar-nav me-auto">

<li class="nav-item">

<a class="nav-link" href="/">Home</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="/add-track">Add Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/delete-track">Delete Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/update-track">Update Song Track</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container py-5">

<div class="row justify-content-center">

<div class="col-md-8 col-lg-6">

<div class="card shadow-sm border-0">

<div class="card-body">

<h3 class="card-title mb-4 text-center"> Add a New Spotify Track</h3>

<form method="POST" action="/add-track">

<div class="mb-3">

<label for="trackUrl" class="form-label">Spotify Track URL</label> <input type="url" class="form-control" id="trackUrl" name="track\_url" placeholder="e.g. https://open.spotify.com/track/..." required>

</div>

<button type="submit" class="btn btn-success w-100">Submit Track</button>

</form>

</div>

</div>

</div>

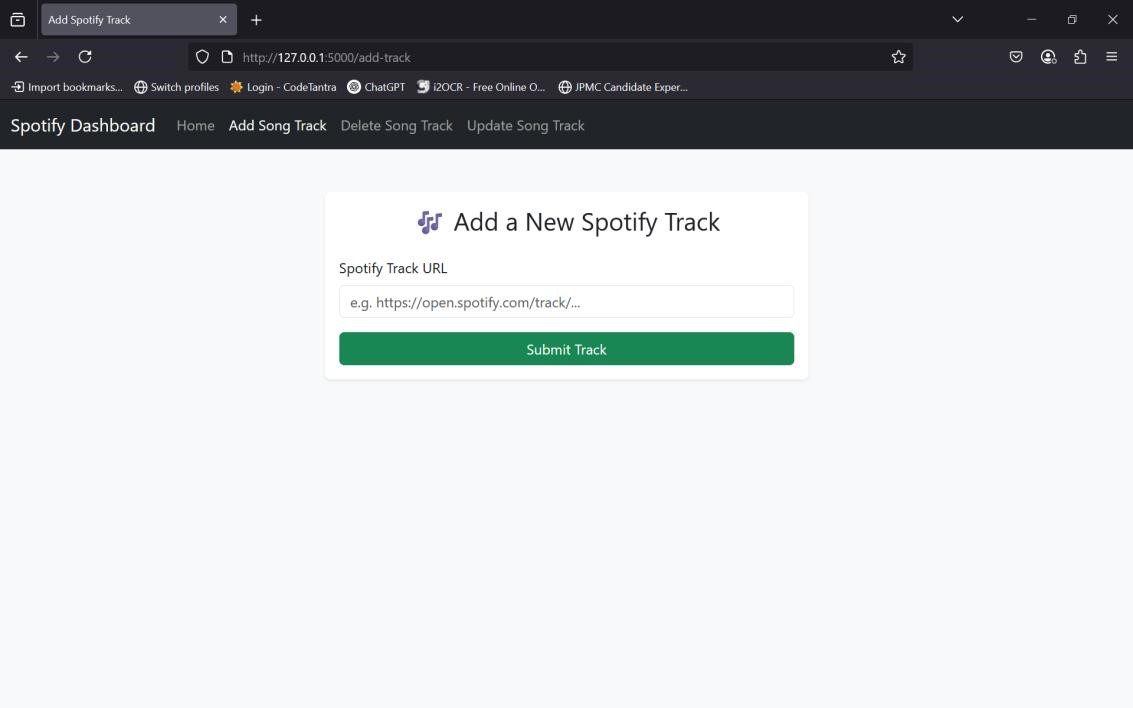
</div>

</div>

</body>

</html>

# Output (Add track page)



# delete\_track.html (Delete a track)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Add Spotify Track</title>

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body class="bg-light">

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Spotify Dashboard</a>

<div class="collapse navbar-collapse">

<ul class="navbar-nav me-auto">

<li class="nav-item">

<a class="nav-link" href="/">Home</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="/add-track">Add Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/delete-track">Delete Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/update-track">Update Song Track</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container py-5">

<div class="row justify-content-center">

<div class="col-md-8 col-lg-6">

<div class="card shadow-sm border-0">

<div class="card-body">

<h3 class="card-title mb-4 text-center"> Enter a Spotify Track to Delete</h3>

<form method="POST" action="/delete-track">

<div class="mb-3">

<label for="trackUrl" class="form-label">Spotify Track URL</label> <input type="url" class="form-control" id="trackUrl" name="track\_url" placeholder="e.g. https://open.spotify.com/track/..." required>

</div>

<button type="submit" class="btn btn-success w-100">Delete Track</button>

</form>

</div>

</div>

</div>

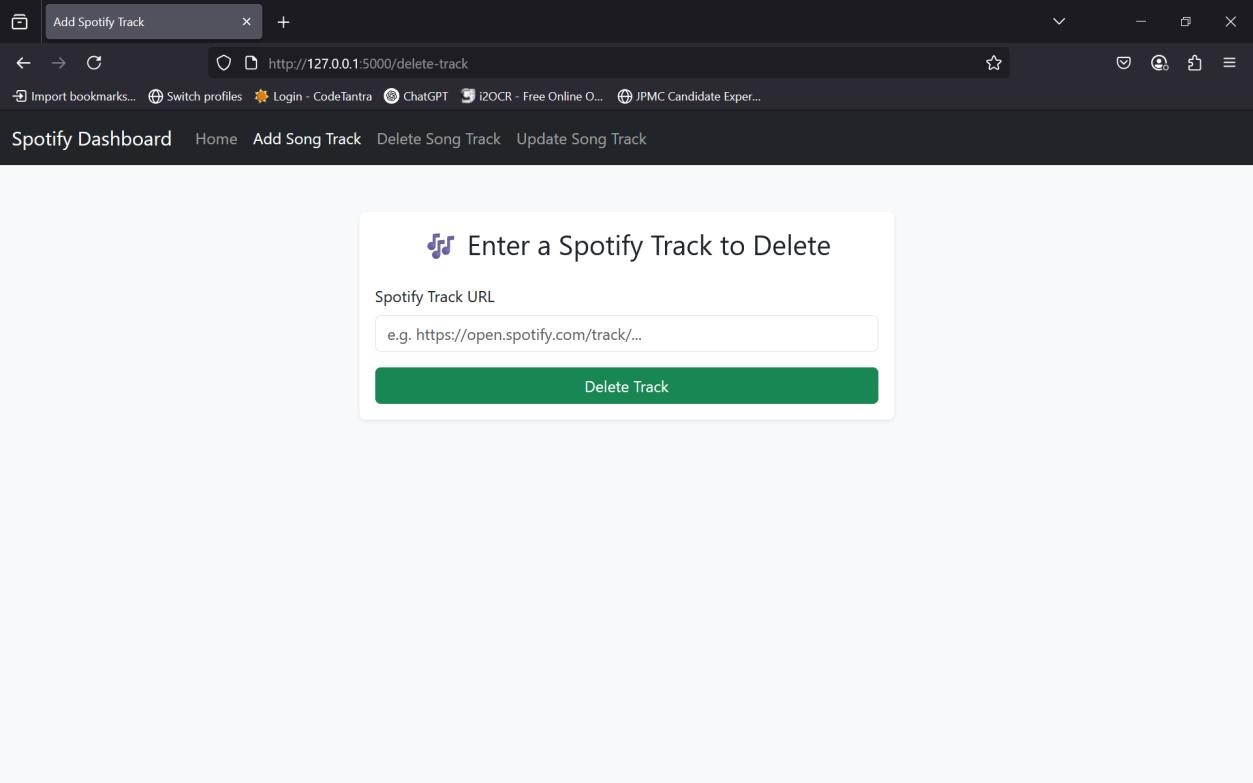
</div>

</div>

</body>

</html>

# Output (delete track page)



# update\_track.html (Update a existing track)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Add Spotify Track</title> <link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body class="bg-light">

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Spotify Dashboard</a>

<div class="collapse navbar-collapse">

<ul class="navbar-nav me-auto">

<li class="nav-item">

<a class="nav-link" href="/">Home</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="/add-track">Add Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/delete-track">Delete Song Track</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/update-track">Update Song Track</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container py-5">

<div class="row justify-content-center">

<div class="col-md-8 col-lg-6">

<div class="card shadow-sm border-0">

<div class="card-body">

<h3 class="card-title mb-4 text-center"> Enter a Spotify Track to Delete</h3>

<form method="POST" action="/update-track">

<div class="mb-3">

<label for="trackUrl" class="form-label">Spotify Track URL</label> <input type="url" class="form-control" id="trackUrl" name="track\_url" placeholder="e.g. https://open.spotify.com/track/..." required>

</div>

<div class="mb-3">

<label for="trackName" class="form-label">Track Name</label>

<input type="text" class="form-control" id="trackName" name="trackName" placeholder="Let Her Go..." required>

</div>

<button type="submit" class="btn btn-success w-100">Update Track</button>

</form>

</div>

</div>

</div>

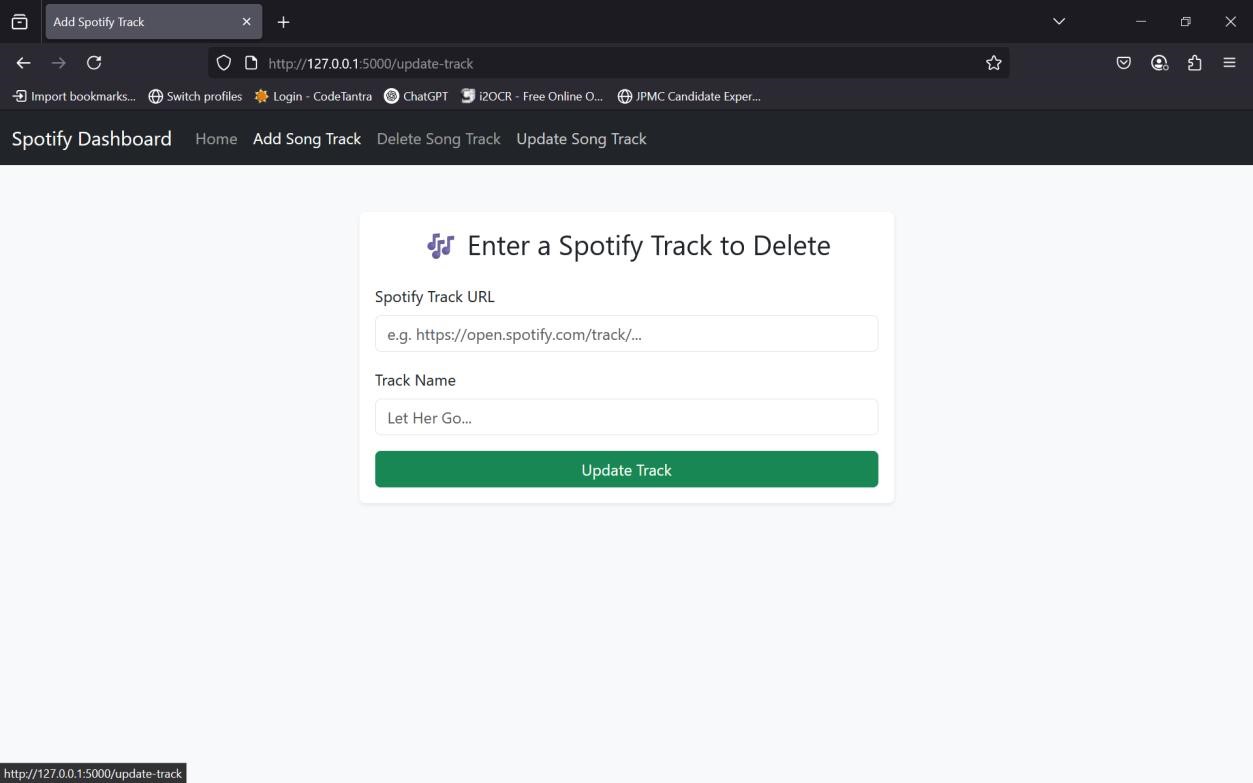
</div>

</div>

</body>

</html>

# Output (Update track page)



**BACK-END CODING**

# App.py (Backend & Query processing)

from spotipy.oauth2 import SpotifyClientCredentials from flask import Flask, render\_template, request, redirect, url\_for import spotipy import pandas as pd import matplotlib.pyplot as plt import re import json

import pymysql

sp = spotipy.Spotify(auth\_manager=SpotifyClientCredentials( client\_id='6decade811ac447aaaa63832575d7b9d',

client\_secret='15b07e4bcdc6407088ccc592155bc1bb'

))

try:

connection = pymysql.connect( host= 'localhost', user= 'root', password= 'Nitish2006', database= 'spotifyAnalytics', port= 3306

)

cursor = connection.cursor() print("DB connected successfully")

except:

print(f"Error")

app = Flask(\_\_name\_\_)

@app.route('/') def index():

# 1. All tracks

cursor.execute("SELECT \* FROM spotifytracks")

all\_tracks = cursor.fetchall()

# 2. Most popular track cursor.execute("""

SELECT trackName, artist, album, popularity

FROM spotifytracks

ORDER BY popularity DESC

LIMIT 1

""")

top\_track = cursor.fetchone()

# 3. Average popularity

cursor.execute("SELECT AVG(popularity) AS average\_popularity FROM spotifytracks") avg\_popularity = cursor.fetchone()[0]

# 4. Long tracks cursor.execute("""

SELECT trackName, artist, durationMinutes

FROM spotifytracks

WHERE durationMinutes > 4.0

""")

long\_tracks = cursor.fetchall()

# 5. Popularity range distribution cursor.execute("""

SELECT

CASE

WHEN popularity >= 80 THEN 'Very Popular'

WHEN popularity >= 50 THEN 'Popular'

ELSE 'Less Popular'

END AS popularity\_range,

COUNT(\*) AS track\_count

FROM spotifytracks

GROUP BY popularity\_range

""")

popularity\_distribution = cursor.fetchall()

return render\_template('index.html', all\_tracks=all\_tracks, top\_track=top\_track,

avg\_popularity=avg\_popularity, long\_tracks=long\_tracks, popularity\_distribution=popularity\_distribution)

@app.route('/add-track', methods=['GET', 'POST']) def add\_track(): if request.method == 'POST': trackUrl = request.form.get('track\_url') print("Track URL received:", trackUrl) trackId = re.search(r'track/([a-zA-Z0-9]+)', trackUrl).group(1) try:

track = sp.track(trackId) except: print("Error in finding track") return redirect(url\_for('index')) #print(json.dumps(track, indent=4))

trackData = {

'Track name': track['name'],

'Artist': track['artists'][0]['name'],

'Album': track['album']['name'],

'Popularity': track['popularity'],

'Duration (minutes)': track['duration\_ms'] / 60000,

'Track ID': trackId

}

query = """INSERT INTO spotifytracks (trackName, artist, album, popularity, durationMinutes, trackId)

VALUES (%s, %s, %s, %s, %s, %s)""" cursor.execute(query,(trackData['Track

name'],trackData['Artist'],trackData['Album'],trackData['Popularity'],trackData['Duration

(minutes)'],trackId))

connection.commit()

print("Inserted the track ",trackData['Track name'])

return redirect(url\_for('index'))

return render\_template('add\_track.html')

@app.route('/delete-track', methods=['GET', 'POST']) def delete\_track():

if request.method == 'POST':

trackUrl = request.form.get('track\_url') print("Track URL received:", trackUrl) trackId = re.search(r'track/([a-zA-Z0-9]+)', trackUrl).group(1) try:

track = sp.track(trackId)

except: print("Error in finding track") return redirect(url\_for('index')) #print(json.dumps(track, indent=4)) trackData = {

'Track name': track['name'],

'Artist': track['artists'][0]['name'],

'Album': track['album']['name'],

'Popularity': track['popularity'],

'Duration (minutes)': track['duration\_ms'] / 60000

}

query = """SELECT \* FROM spotifytracks WHERE trackId = %s"""

cursor.execute(query, (trackId)) if(cursor.fetchall()):

query = """DELETE FROM spotifytracks WHERE trackId = %s"""

cursor.execute(query, (trackId)) connection.commit() print("Track deleted successfully") else:

print("No tracks found")

return redirect(url\_for('index')) return render\_template('delete\_track.html')

@app.route('/update-track', methods=['GET', 'POST']) def update\_track(): if request.method == 'POST': trackUrl = request.form.get('track\_url') trackName = request.form.get('trackName') print("Track URL received:", trackUrl) trackId = re.search(r'track/([a-zA-Z0-9]+)', trackUrl).group(1) try:

track = sp.track(trackId) except: print("Error in finding track") return redirect(url\_for('index')) #print(json.dumps(track, indent=4))

trackData = {

'Track name': track['name'],

'Artist': track['artists'][0]['name'],

'Album': track['album']['name'],

'Popularity': track['popularity'],

'Duration (minutes)': track['duration\_ms'] / 60000

}

query = """SELECT \* FROM spotifytracks WHERE trackId = %s"""

cursor.execute(query, (trackId)) if(cursor.fetchall()):

query = """UPDATE spotifytracks SET trackName = %s WHERE trackId = %s"""

cursor.execute(query, (trackName, trackId)) connection.commit() print("Track updated successfully") else:

print("No tracks found")

return redirect(url\_for('index'))

return render\_template('update\_track.html')

'''filePath = "tracksUrl.txt" with open(filePath, 'r') as file: trackUrls = file.readlines()

for trackUrl in trackUrls: trackUrl = trackUrl.strip()

trackId = re.search(r'track/([a-zA-Z0-9]+)', trackUrl).group(1) track = sp.track(trackId)

#print(json.dumps(track, indent=4))

trackData = {

'Track name': track['name'],

'Artist': track['artists'][0]['name'],

'Album': track['album']['name'],

'Popularity': track['popularity'],

'Duration (minutes)': track['duration\_ms'] / 60000

}

query = """INSERT INTO spotifytracks (trackName, artist, album, popularity, durationMinutes) VALUES (%s, %s, %s, %s, %s)""" cursor.execute(query,(trackData['Track

name'],trackData['Artist'],trackData['Album'],trackData['Popularity'],trackData['Duration

(minutes)']))

connection.commit()

print("Inserted the track ",trackData['Track name']) cursor.close()

connection.close()

print("All tracks have been processed and inserted into the database.")

'''

'''print("Metadata: ")

print(json.dumps(trackData, indent=4))

df = pd.DataFrame([trackData])

df.to\_csv('spotifyTrackData.csv',index=False)

features = ['Popularity', 'Duration (minutes)']

values = [trackData['Popularity'],trackData['Duration (minutes)']] plt.figure(figsize=(8,5))

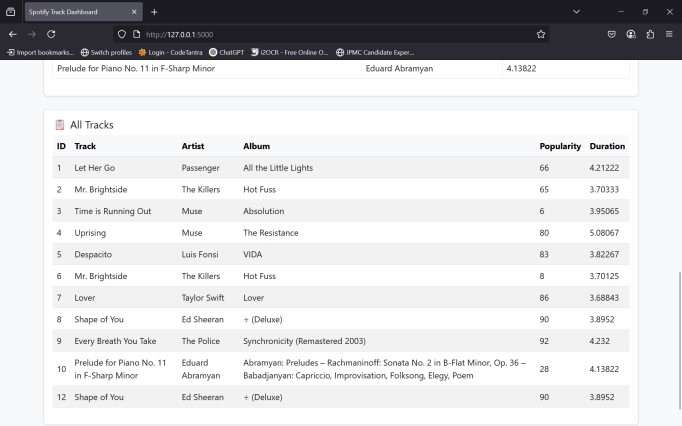
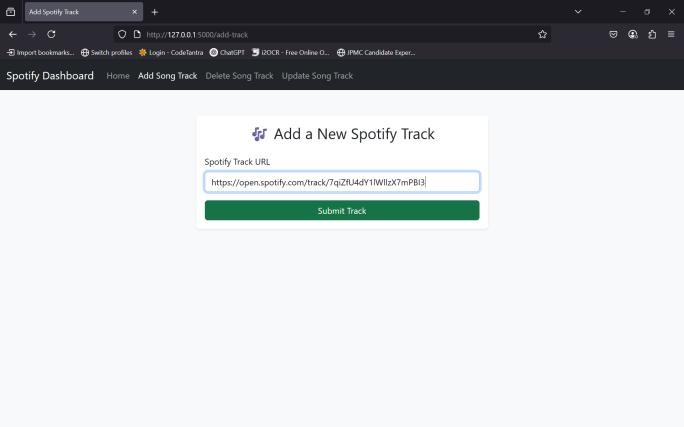
plt.bar(features, values, color='skyblue', edgecolor='black') plt.title("Track metadata") plt.ylabel('Value')

plt.show()'''

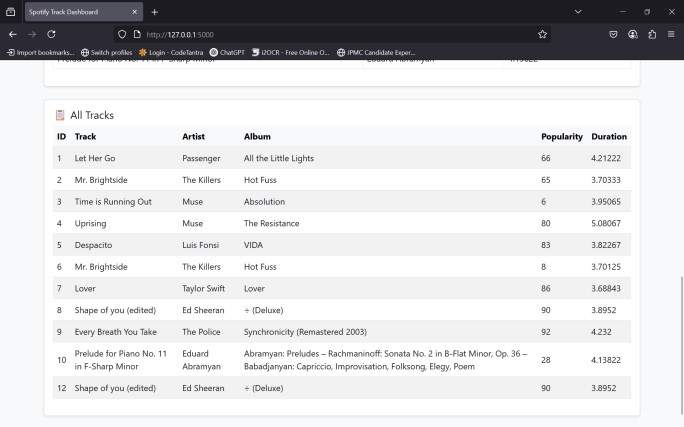
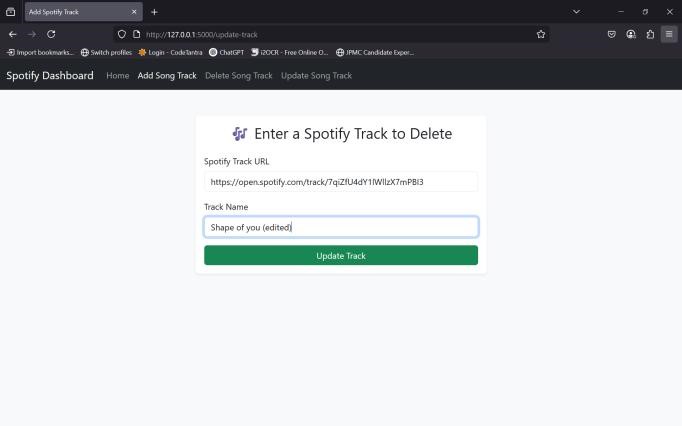
if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

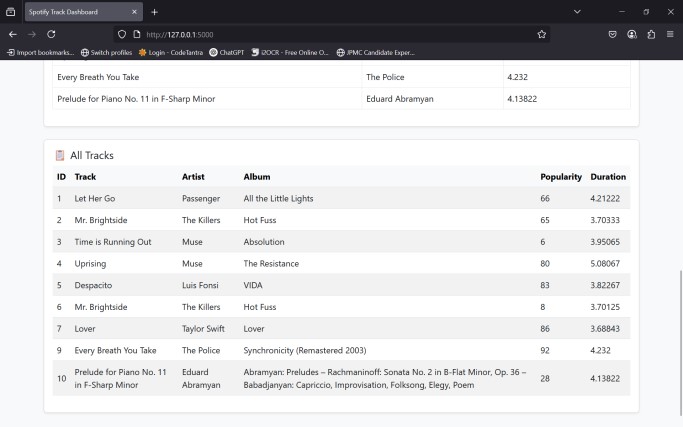
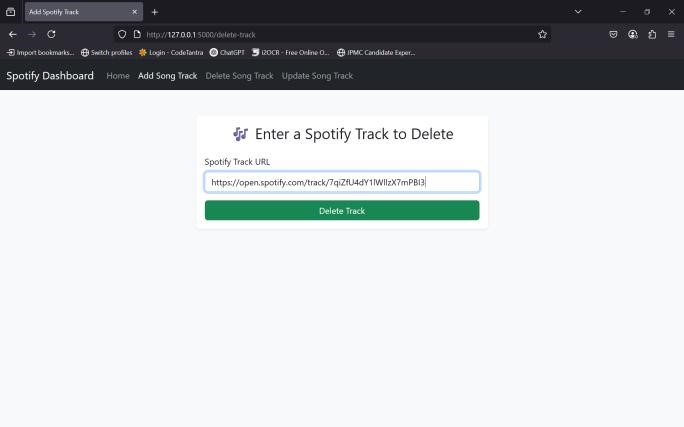
**OUTPUT:**



**Adding a new Track Listing the updated track**



# Updating a existing track Listing the updated track



# Deleting the existing track Listing the updated track

|  |  |  |
| --- | --- | --- |
| **CONTENTS** | **MARKS ALLOTED** | **MARKS OBTAINED** |
| Aim, Algorithm, SQL, PL/SQL | 30 |  |
| Execution and Result | 20 |  |
| Viva | 10 |  |
| Total | 60 |  |

**RESULT:**

Thus, a spotify analytics dashboard was created and performed CRUD operation using Flask and Python.