In [57]:
```
import pandas as pd

# Load your dataset
df = pd.read_csv('top 100 streamed_songs.csv')
```

In [58]:
```
df
```

Out[58]:

	id	name	duration	energy	key	loudness	mode	speechiness	acousticnes
0	4ZtFanR9U6ndgddUvNcjcG	Good 4 U Olivia Rodrigo	2.97	0.664	9	-5.044	1	0.1540	0.3350
1	5fxyZf6m2xHeSrOzUfcJrq	Stay The Kid LAROI & Justin Bieber	2.30	0.506	8	-11.275	1	0.0589	0.3790
2	5nujrmhLynf4yMoMtj8AQF	Levitating Dua Lipa feat. DaBaby	3.38	0.825	6	-3.787	0	0.0601	0.0088
3	4iJyoBOLtHqaGxP12qzhQI	Peaches Justin Bieber feat. Daniel Caesar & Gi...	3.30	0.696	0	-6.181	1	0.1190	0.3210
4	1SC5rEoYDGUK4NfG82494W	Montero (Call Me By Your Name) Lil Nas X	2.30	0.503	8	-6.725	0	0.2200	0.2930
...	...	...	...	...	...	...	...	...	
95	4iJyoBOLtHqaGxP12qzhQI	Peaches (feat. Daniel Caesar & Giveon) Justin ...	3.30	0.696	0	-6.181	1	0.1190	0.3210
96	1FSWSs9CL01RCYxXtm08Rf	Dance With Me Tonight Olly Murs	3.37	0.748	11	-5.922	0	0.0589	0.3050
97	54bFM56PmE4YLRnqpW6Tha	Therefore I Am Billie Eilish	2.91	0.340	11	-7.773	0	0.0697	0.2180
98	5jsw9uXEGuKyJzs0boZ1bT	Stitches Shawn Mendes	3.45	0.754	1	-6.684	1	0.0615	0.0151
99	1SC5rEoYDGUK4NfG82494W	Montero (Call Me By Your Name) Lil Nas X	2.30	0.503	8	-6.725	0	0.2200	0.2930

100 rows × 14 columns

In [59]:
```
# drop columns we dont wanna use

df = df.drop(columns=['id'])
```

In [60]:
```
from sklearn.preprocessing import StandardScaler

# Check for missing values
print(df.isnull().sum())

# Drop missing values
df = df.dropna()

# Select features for recommendation
features = ['duration', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrum

# Scale the features
scaler = StandardScaler()
df[features] = scaler.fit_transform(df[features])
```

```
name                0
duration            0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
danceability        0
dtype: int64
```

In [61]:
```
from sklearn.metrics.pairwise import cosine_similarity

# Compute cosine similarity matrix
similarity_matrix = cosine_similarity(df[features])

# Convert the matrix to a DataFrame for easier manipulation
similarity_df = pd.DataFrame(similarity_matrix, index=df['name'], columns=df['name'])
```

In [62]:
```
def get_recommendations(song_name, similarity_df, df, top_n=10):
    try:
        # Check if the song is in the dataset
        if song_name not in similarity_df.index:
            return f"Song '{song_name}' not found in the dataset."

        # Get similarity scores for the song
        similarity_scores = similarity_df[song_name].sort_values(ascending=False)

        # Get the top N similar songs
        top_songs = similarity_scores.iloc[1:top_n + 1].index

        # Return the details of the recommended songs
        return df[df['name'].isin(top_songs)][['name']]
    except KeyError as e:
        return f"An error occurred: {e}. The song '{song_name}' may not exist in the datas
    except Exception as e:
        return f"An unexpected error occurred: {e}"
```

In [63]:
```
# Example usage
recommended_songs = get_recommendations('Good 4 U Olivia Rodrigo', similarity_df, df, top_
print(recommended_songs)
```

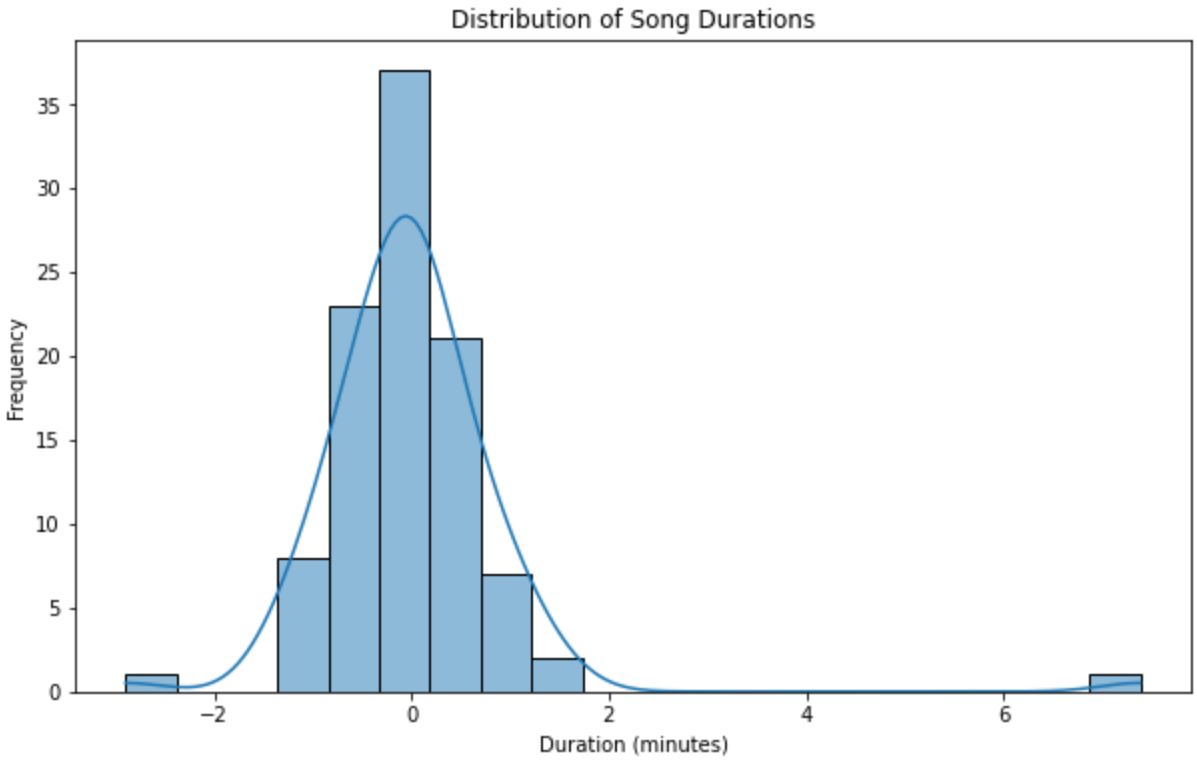
```
                                name
12                      Rapstar Polo G
28         LA CANCIÓN J Balvin & Bad Bunny
36              Tusa KAROL G & Nicki Minaj
42    Head & Heart (feat. MNEK) Joel Corry
70  Heat Waves - Diplo Remix Glass Animals
```

In [64]:
```
# Test the function
print(get_recommendations('Stitches Shawn Mendes', similarity_df, df, top_n=5))
```

```
                                              name
8                      Blinding Lights The Weeknd
48                     Save Your Tears The Weeknd
49  The Woo (feat. 50 Cent & Roddy Ricch) Pop Smoke
64       Uptown Funk (feat. Bruno Mars) Mark Ronson
78  Somebody That I Used To Know Gotye feat. Kimbra
```

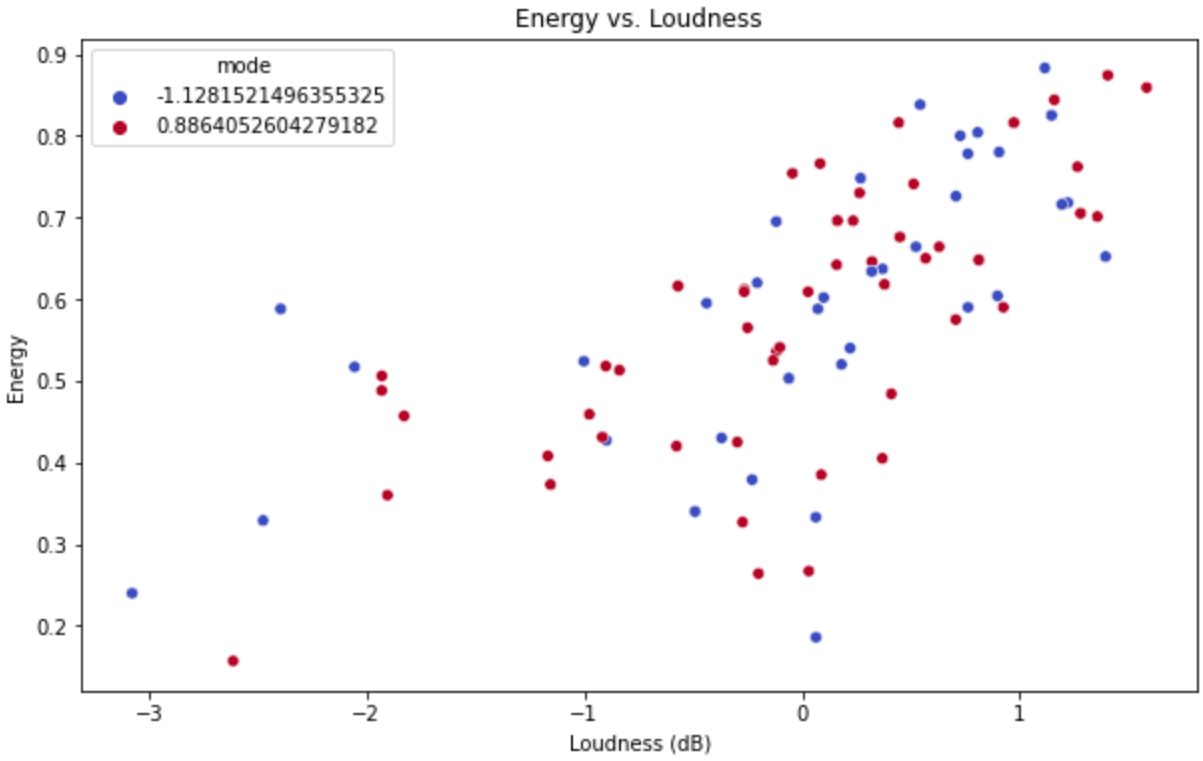
In [71]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram of song durations
plt.figure(figsize=(10, 6))
sns.histplot(df['duration'], kde=True, bins=20)
plt.title('Distribution of Song Durations')
plt.xlabel('Duration (minutes)')
plt.ylabel('Frequency')
plt.show()
```



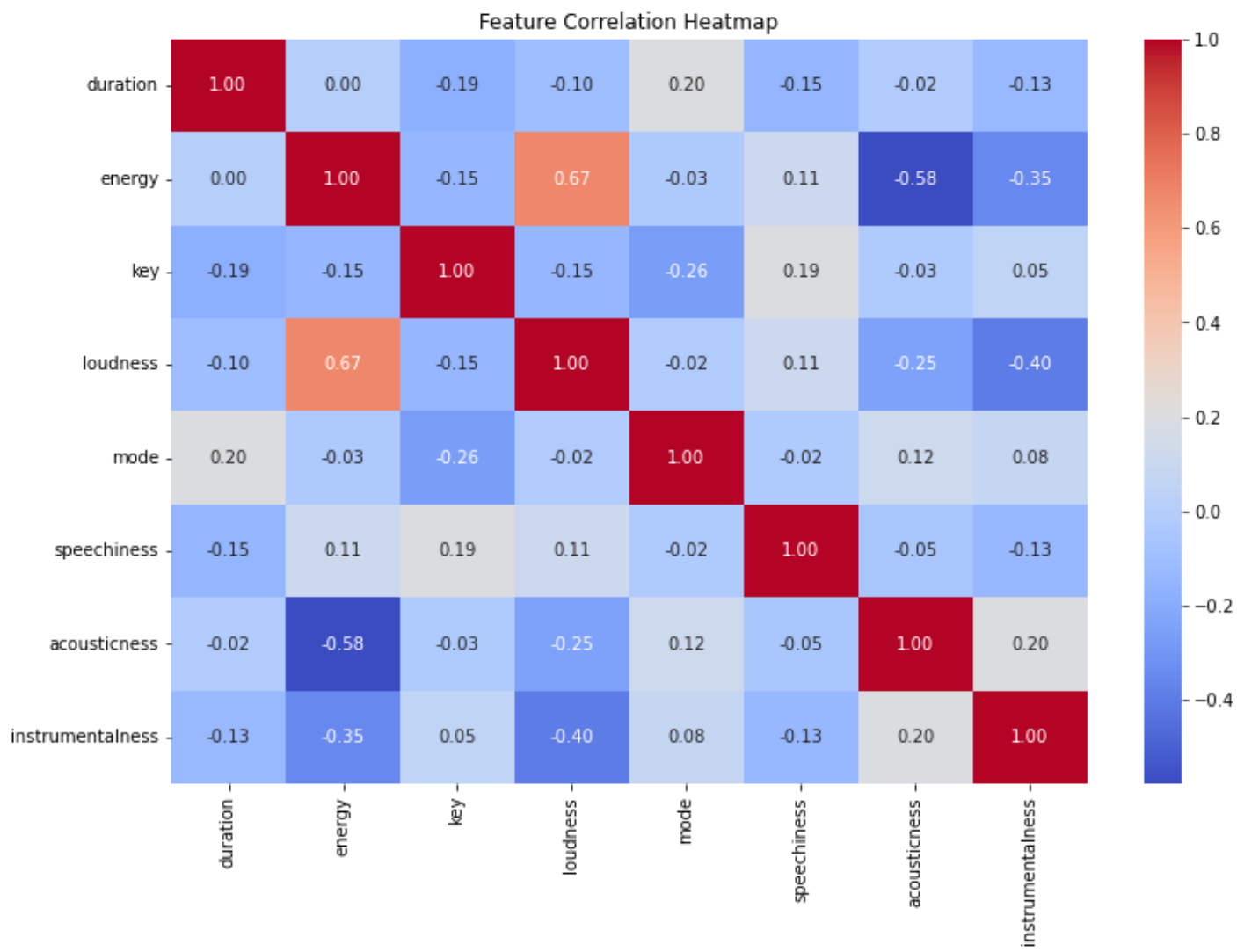
In [67]:
```
# Scatter plot of energy vs. loudness, mode is if the song is in major or minor, checks to
# have higher energy levels
plt.figure(figsize=(10, 6))
sns.scatterplot(x='loudness', y='energy', data=df, hue='mode', palette='coolwarm')
```

```
plt.title('Energy vs. Loudness')
plt.xlabel('Loudness (dB)')
plt.ylabel('Energy')
plt.show()
```



In [69]:

```
# Correlation heatmap
plt.figure(figsize=(12, 8))
correlation_matrix = df[['duration', 'energy', 'key', 'loudness', 'mode', 'speechiness',
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Feature Correlation Heatmap')
plt.show()
```


Feature Correlation Heatmap
duration
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
1.00 0.00 -0.19 -0.10 0.20 -0.15 -0.02 -0.13
0.00 1.00 -0.15 0.67 -0.03 0.11 -0.58 -0.35
-0.19 -0.15 1.00 -0.15 -0.26 0.19 -0.03 0.05
-0.10 0.67 -0.15 1.00 -0.02 0.11 -0.25 -0.40
0.20 -0.03 -0.26 -0.02 1.00 -0.02 0.12 0.08
-0.15 0.11 0.19 0.11 -0.02 1.00 -0.05 -0.13
-0.02 -0.58 -0.03 -0.25 0.12 -0.05 1.00 0.20
-0.13 -0.35 0.05 -0.40 0.08 -0.13 0.20 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

In [ ]: