1. Emotion-Based Playlist Generator **(& Player?)** *[Pick #1]*

My project would create music playlists depending on the user's present mood. The user would offer a brief textual account of their feelings, and I use natural language processing (NLP) to examine the user's mood. The application would create a playlist from a pre-curated music collection that fit the sentiment (e.g., happy, sad, energetic). This project most likely would be a desktop program running Tkinter.

Users and Stakeholders:

The main users would be music aficionados who wish to customize their listening experience depending on their mood. Stakeholders would be Music therapists or streaming service providers intrigued in emotion-driven music curating could be secondary stakeholders.

Problem to Solve:

This app would offer an emotionally responsive music experience, therefore enabling users to improve or alter their mood through music specifically suited to their present emotions.

Biggest Problem to Implement:

The biggest challenge in implementing this project would be accurately tagging and curating the music library with appropriate emotional labels. Ensuring that the sentiment analysis correctly interprets the user’s mood and matches it to the right songs could also be challenging, as emotions are complex and can be nuanced.

What Part Can I See Myself Doing:

I can see myself handling the sentiment analysis and playlist generation parts of the project independently, given my familiarity with Python libraries like TextBlob and Tkinter. For the music curation and integration with a music player, I might seek some guidance or collaborate with others, especially if integrating with existing music APIs like Spotify or Apple Music.

Workflow:

The user would type a brief description of their present mood. Using sentiment analysis, I would assess the general feeling. The program would then create a playlist depending on the identified

feeling that the user could listen to using their chosen music player or a music player integrated into the app.

Data and Processing:

To handle user input, I would run a sentiment analysis library—like TextBlob or VADER. The playlist would be created by matching a set of preselected songs labeled with related emotions to the emotion itself.

Results and Presentation:

The program would produce a well-chosen playlist either as a text list of song titles or together with a music player to automatically queue the tracks.

2. FinStats - A Stock Price Analytics Platform *[Pick #2]*

I would make a platform for analyzing stock prices called FinStats that gives people detailed statistics and data visualizations for stocks around the world. Users will be able to look up any stock on the site and get a lot of information about how it has done, such as past prices, trend analysis, and key financial metrics. The goal is to make an interface that is easy for investors and financial experts to use and understand so they can look at stock market trends.

The app will probably be made as a web app with Flask as the server and a JavaScript framework like React as the frontend. The platform would get data from financial APIs (probably the free ones), process it, and show it in charts and tables that users could interact with. The project could potentially grow over time to include more advanced features, such as predictive analytics or custom investment suggestions based on what users like.

Users and Stakeholders:

Individual investors, financial analysts, and portfolio managers who need quick access to stock price data and figures would be the main users who use FinStats. Secondary stakeholders could be fintech companies that could add FinStats to their own platforms to improve analytics or educational schools that use the platform to teach finance students.

Problem to solve:

To make smart choices, investors and analysts often need detailed and up-to-date information about the stock market. FinStats would help by giving users a central place to quickly access and examine stock data. This would cut down on the time spent looking for information in many places and allow for better investment decisions.

Biggest Problem to Implement:

The biggest challenge in implementing this project would be ensuring that the platform can handle real-time data processing and display it efficiently. Additionally, managing the accuracy and reliability of the data retrieved from free APIs could pose a challenge, especially if the data sources have usage limits or provide incomplete information.

What Part Can I See Myself Doing:

I can see myself handling the backend development, including setting up the Flask server and integrating with financial APIs to retrieve and process stock data. For the frontend development, particularly the creation of interactive visualizations using a JavaScript framework like React, I may need some help or additional resources to ensure the interface is both functional and visually appealing.

Workflow:

Individuals could use a search bar to look for a specific stock or scroll through groups such as market areas. Real-time data from financial APIs would be pulled into the platform and shown in a dashboard with different charts, tables, and key measures. Users could change how the dashboard looked by adding filters for date groups, comparing stocks, or seeing different financial indicators.

Data and Processing:

APIs like Alpha Vantage, Yahoo Finance, and others like them would be used to get financial info for the platform. Stock prices, trading volume, market capitalization, and other important financial measures would be in the data. The platform would use this information to make charts that show things like line charts for past prices, bar charts for comparing volume, and pie charts for showing how the portfolio is split up. Basic statistical analysis, like volatility indices or moving averages, would also be part of this.

Results and Presentation:

People would be able to see and change charts, tables, and other visual features on an interactive web interface that shows the results. You could look into the performance of each stock in great depth.

3. Movie Success Predictor [Pick #2, no pick #3]

My project would develop a web application that predicts the potential success of upcoming movies based on various factors such as cast, director, genre, release date, and social media buzz. The app would scrape data from multiple sources, including IMDb, Rotten Tomatoes, Twitter, and Reddit, gathering information on past movies, current trends, and audience sentiment. Using this data, I would employ machine learning algorithms to predict box office performance and critical reception of upcoming movies, providing an engaging tool for movie enthusiasts to speculate on new releases.

Users and Stakeholders:

The main users would be movie enthusiasts, bloggers, and entertainment industry professionals interested in predicting the success of upcoming films. Secondary stakeholders might include marketing agencies or studios that could use the tool to gauge public interest and sentiment before a movie’s release.

Problem to Solve:

Predicting a movie's success before its release can be challenging due to the numerous factors involved. This app would provide a data-driven way to make educated guesses about a movie's potential success, helping users and industry professionals gain insights into the factors that contribute to a film's performance.

Biggest Problem to Implement:

The biggest challenge in implementing this project would be effectively scraping and integrating data from various sources, as each platform might have different structures, and accessing comprehensive data could be subject to rate limits or restrictions. Additionally, training a machine learning model that accurately predicts movie success would require a robust dataset and careful feature selection to avoid overfitting or underfitting.

What Part Can I See Myself Doing:

I can see myself managing the data scraping and integration aspects of the project, as well as developing the machine learning model using regression analysis or decision trees. For the frontend development, particularly creating a visually appealing and interactive interface, I might need some help or additional resources, especially if incorporating advanced visualizations or ensuring a seamless user experience.

Workflow:

The user would input the title of an upcoming movie or select it from a list. The app would then scrape data from various sources, including IMDb for cast and crew details, Rotten Tomatoes for reviews of similar movies, and social media platforms like Twitter and Reddit for public sentiment. The machine learning model would analyze this data to generate a prediction score for the movie's box office performance and critical reception.

Data and Processing:

The app would use data scraped from IMDb, Rotten Tomatoes, Twitter, and Reddit, focusing on factors like cast, director, genre, and social media buzz. I would use machine learning techniques such as regression analysis or decision trees to process this data and predict box office revenue and critic scores. The model would be trained on historical data to ensure accuracy.

Results and Presentation:

The app would present the prediction as a score (on a scale of 10) indicating the movie's potential box office success and critical reception. It would also provide a breakdown of the factors contributing to the prediction, such as cast popularity, genre trends, and social media sentiment. The results would be displayed in an interactive and visually appealing interface, allowing users to explore the data in detail.