Team Members: Kweku Amoa, Donovan McCoy, Collins Nji, Alex Piccinich, Sergio Quiijano-Sanin, Kim Shields, Chance White, Amanda Yeung

Overview:
Our project for class is to create a web page to allow users to communicate with a chatbox in order to get song recommendations. The users will be prompted to create an account and answer some questions which will be prompted by the chatbox(these questions will likely be related to the music genre, tempo, the occasion, etc.) they can then toggle between this page, their account page(which also saves liked playlist) and then a page where they can see all the songs in their liked playlist.

High Level Description of the Project

- We believe the music industry is only going to grow overtime and with the security that the industry will not fail we feel this is a pretty safe investment. There is also a big push in the millennial and GEN X generation to support small and allow new businesses to survive. We hope that users will log onto our website and then create personalized playlists with the potential to buy their music directly from the artist and take away from the big corporations.
- We feel the music industry is dominated by two main companies and we think it's time for a new software to give more options to the consumers. By eliminating the basis (where the companies receive money) within the companies who recommend music we hope this would give each artist an equal chance to come across someone's playlist and not based on the amount of money they pay.
- As far as our customer base we would want this app to be accessible to anyone who enjoys listening to music. As far as development goes we would have a primary team responsible for updating and taking care of our servers. The secondary users- those who use the app would get in contact with the primary team in case anyone faces any issues. Music is a universal experience so we want to bring this system to everyone.
- In order to use this tool, the user only needs to have a device that can access the internet. They can expect the need to create an account(likely will require an email address, username, passwork, and a name) besides that it is very easy for the user to use.

Assigned Roles for each Team Member:

Role	Member
Requirements Analyst	Kim Shields, Alex
Test Engineer	Donnie
Data Engineer	Sergio, Amanda
Software Engineer/ Solution Architect	Chance

Deployment Engineer	Kweku Amoa
Application Programmer	Everyone
Project Manager	Collin

Requirements for Reliability

- Can we release patches while still being available 24/7? (maybe analyze which times there are less users, and push the changes then?) and/or make an announcement of when we'll be down)
- Deploy the application in a stable cloud environment and allow logging errors and requests
- Allow users to send error messages to inform developers of problems
- Keep logs of when users log in or out of the system
- Have documentation available for users in case of errors or bugs
- Develop user guides for utilizing our tool/program
- User confidentiality? (user data security w/ passwords and personal info)

Requirements for Effectiveness

- Keep the user interface simple and un-convoluted
- Delivery of music streaming, flawlessly/ no lag on our end
- Have a queue of songs ready to be played -
- Provide accurate suggestions according to the user's choices
- Cost-effective → money spent on licenses *but* profit by subscription to offset
- We can communicate the project to Spotify so that they can give us special API keys that allow us to make more calls

Requirements for Adaptability

- Allow for a database to be constantly changing with new music
- Be able to "maintain" a growth of subscription services
- Keep a list of available updates and new features/ have a release date
- Keep an API documentation showing how to add new features to the application
- Have a plan to get more API calls as more users are joining
- Being able to expand our target audience, maybe not just US, going global?
- Being a mobile app vs desktop app

Requirements for Maintainability

- Make use of comments and describing what the code is doing (documentation)
- Keep track of important changes and conversations revolving around the application
- Write quick-start documentation for using the application
- Autogenerate API docs (using docstrings) from comments in the code
- Implement the use of Version Control in order to make maintainable changes

- Have a flow-graph where we can link how the application will communicate with databases

IMPLEMENTATION

- The application will be built using the Flask framework, and it will connect to the Spotify API to get song information to display to the users. The application will also implement a chatbot which a user can interact with to provide instant song suggestions based on the user input.
- The application will be deployed in the cloud (either using Heroku or Google Cloud).
- We plan to use a version control system to manage the code base, and to have various deployment steps to ensure that the application stays running and available 24/7 even when we issue software patches.
- The application will automatically log build statuses and other loggable information to ensure that we can find error points easily in case the application fails.
- It will allow users to send error messages directly to the developers to inform them of problems that may arise during runtime
- We plan to have detailed documentation available for users in case of errors or bugs
- Develop user guides for utilizing our tool/program
- Write quick-start documentation for using the application

Screen Prototypes

Please, use the link below to view the various screen drafts for the application :
https://jamboard.google.com/d/17ml8nupfUrnoNOMFEIqPvYqsSkUULIPaZgBXnKBEYIg/viewer

HIGH LEVEL ARCHITECTURE USE CASE

This diagram is the general overview of how the app works. It shows the functions of the users and how they interact within the app along with the songs API and the components of the application it interacts with.