vibecheck Component Diagram

- Mobile App
- Infrastructure
- Machine Learning

User Login

- Use of **Spotify** user credentials to request access tokens from the Spotify API
- This screen also verifies the current tokens every time the user opens the app, if they are not valid, a new set of tokens is requested
- The user is not allowed to continue unless they are verified

Landing Screen

- Main screen of the application
- The user is asked to enter a "vibe" or sentence

Vibecheck screen

- Queries the Docker container
- The results from the query are displayed
- The user is able to play any of the songs displayed by using the Spotify API
- The user is also able to share the current playlist as a social media link
- The ability to save the playlist to the users's Spotify account is also provided

Queries Azure Container

Docker Container

- Flask endpoint for Mobile App queries
- ML model index for fast retrieval

ML model is contained inside the containerized app

ML model

- Trains model that is used for flask endpoint
- Use annoy index to make quicker predictions
- Generates artifact for flask endpoint

Model trains using data from the database

Data Mining Virtual Machine

- Short term Storage
- Mines data from the internet such as song lyrics and various pieces of metadata

Caches data

Redis Cache

- Short term Storage
- Has persistence in memory
- Fast read and writing of data

Archives data

Mongo DB

- Long term storage of data
- Used for data to train ML models

Reference

- Docker
- Redis
- MongoDB
- Flask
- Pytorch
- Azure
- Allen NLP
- Annoy
- React Native
- Spotify API
- TypeScript
- Python
- Jupyter Notebook

A project developed and designed by: **Jacob Danovitch, Omar Garcia Flores, Nathan Embaye & Faris Mohamed**