JUKEBOX

Project Goal: Develop a mobile app that can generate a playlist based off of multiple user preferences and stream the composite playlist from a host device.

Design Stage

1. Determine specific app features[1].
    a. Guest Side
        i. Collect Music Preferences.
        ii. Send Music Preferences To Host
        iii. Rate Quality Of Current Generated Playlist
        iv. Request More Of Specific Genre or Music Feature(ie, danceable)
    b. Host Side
        i. Compile Guest Music Preferences
        ii. Manually Add Playlist Features
        iii. Monitor Changes In Guest Pool And Update Accordingly
        iv. Generate Playlist Based On Compiled Preferences
        v. Use Playlist Recommendation API to generate playlist
        vi. Play Playlist
    c. Other
2. Decide on infrastructure
    a. Playlist Recommendation API
        i. Spotify
            1. Pros: Robust and well documented with a convenient playlist Seed Object design for generating playlists.

[1] Features required for Proof Of Concept are highlighted

2. Cons: Seed API does not seem accessible from Spotify mobile SDK's. In fact they seem to disapprove of broadcasting "listen together" apps and their mobile suites are limited.
    a. https://github.com/Daltron/Spartan
    b. Above link provides a wrapper between ios and spotify web api
3. Implementation: Our app will need to scan guest Spotify apps for information on their libraries/playlists and generate seed objects themselves. The host in turn will need to hit the Spotify REST API without the aid of a convenient mobile SDK. As far as I know, apps can issue Post and Get requests to endpoints directly so this should be achievable.

ii. Pandora

1. Pros: Available API's are cleaner than Spotify.
2. Cons: Pandora API is not publicly available. Only available options are hacky 3rd party reverse engineering.
3. Implementation: Need to do more research - don't think this one is viable.

iii. Apple Music

1. Pros: Robust and well documented with Recommendations API embedded and pre-generated for each user. Has easy mobile plugin
2. Cons: Only available for ios machines.
3. Implementation: Using the musickit plugin, our app can easily interface with the Apple Music API for each step of the playlist recommendation process.

b. App Language

i. Python

1. There exist several services for deploying mobile apps in Python. Unclear if this would be feasible for this kind of project.
2. Pros: Cross Platform Support and low initial learning curve
3. Cons: Limited advanced functionality and higher learning curve on the later more complicated features.

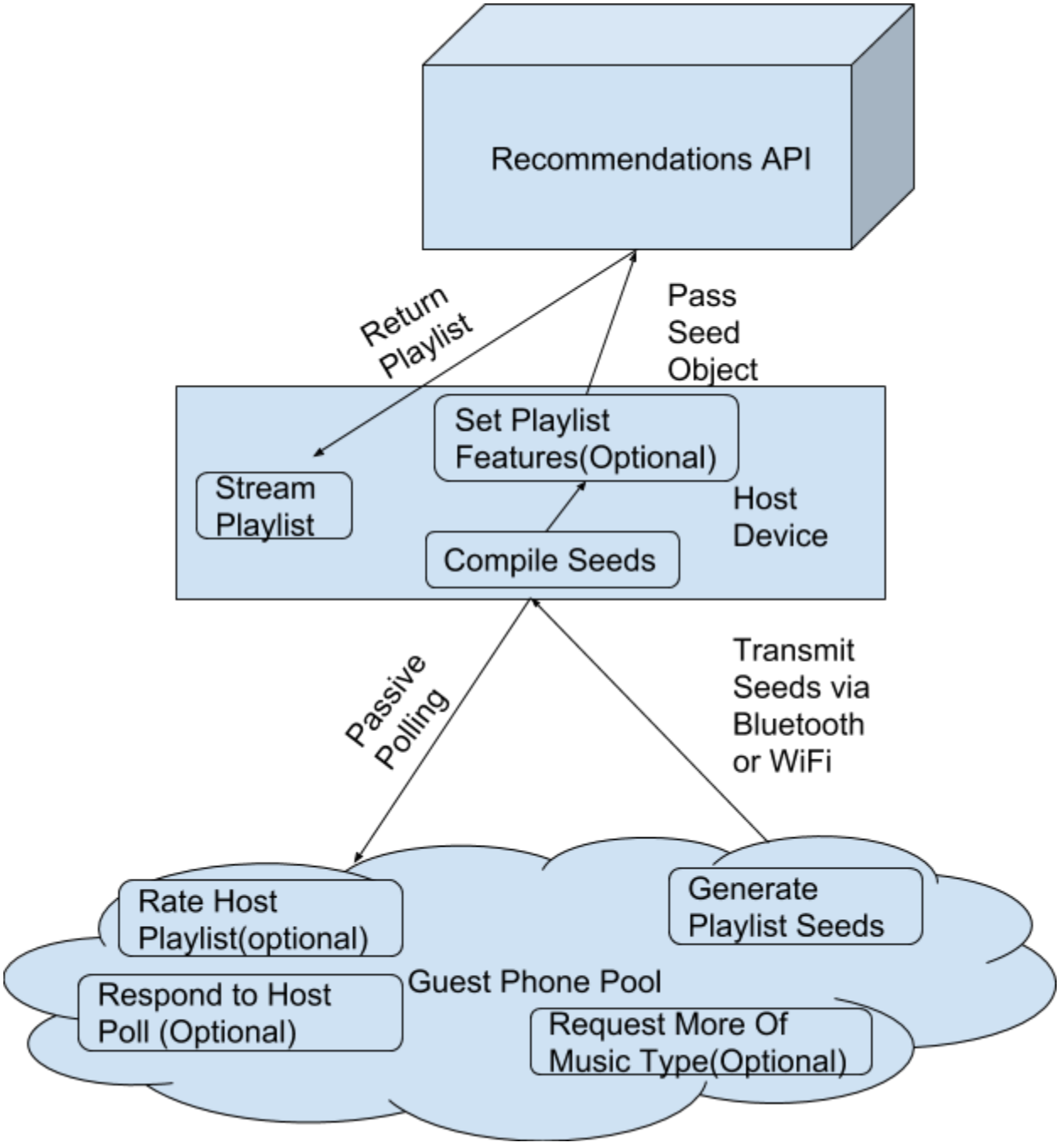
ii. Swift

1. ios native language. If we go the Apple Music route this might be most effective.

iii. Java

1. Android native.

General Design Overview:



Development Stage:

Proof of Concept Requirements:

1. App collects user music data.
2. Passes user music data to host.
3. Host does something with that data and plays something in response.

Detailed Breakdown:

Build Guest Side

1. Make an app that can read from user music service
2. Generate some kind of Preferences Object from user music library.
3. Optional: Allow users to select specific playlists to be scanned by the app.

Build Host Side

1. Compile Preference Objects into a single Preference Object.
2. Generate a Playlist from combined preference Object.
3. Optional: Build the host side as part of the app.

Build Networking Connections

1. Have app send Preference Objects to Host via standard network request protocols.

2. Convert host into a server that listens for incoming Preferences Objects.
3. Optional: Enable Bluetooth sharing for app.
4. Optional: Use Playlist Recommendation API when generating the new playlist. This will likely require formatting the Preference Objects in a particular way so if we want to try and do this, best to decide early.