CS 4720 Girafarig Final Project Documentation

Developer List

- Dennis Huang (dlh4fx)
- Jason Valenzuela (jev4zs)

Device Name & Platform

- Device name is Girafarig, and it is an Android device

Project Title

- Move-N-Groove

Project Pitch

The purpose of this app is to play songs from a playlist based on a user's actions. The app has four different playlists to choose from: Day, Night, Fast, and Slow. The playlists are from Spotify, and a premium Spotify account is required to use the app. Once logged in, the user can perform an action to trigger a change of the playlist. For the Day and Night playlists, a light sensor is used to see if the user is in a well lit area. If yes, the Day playlist is suggested, otherwise the Night one is. For Fast and Slow, the GPS is used to determine the user's speed. If the user crosses a certain threshold of speed, then the Fast playlist is suggested, otherwise the Slow one is. The app suggests playlists by creating an alert dialog that shows what the app has detected the user doing. The user can select "yes" to start playing the suggested playlist or no to not. The user can then pause, skip forward, and skip backward on a playlist. The app also allows the user to manually choose a specific song on any of the playlists by accessing a playlist's info page.

Platform Justification

Android was chosen as the platform because the partners felt more comfortable with it than iOS. Both had plenty of experience with Java and felt that for a project of a larger scale

such as this one, it was best to stick with what they already knew. Previously working together with XCode to develop an iOS app proved to be rather frustrating, as only one partner had a Macbook. The other partner had to find time to go to Rice 340, which was constantly reserved for other classes. Furthermore, the lack of documentation available online for Swift 8 made it hard to find answers to implement simple functions as novices in iOS development. The partners found the convenience of Android Studio and the more robust documentation of Android to be more appealing than iOS.

Key Features

- Spotify Authentication
    - When the user first accesses the main screen of the app, a pop up will emerge asking the user to login with a Spotify account. Once logged in, the user can then start playing playlists.
- GPS implementation
    - The app will prompt the user to allow permissions for Location Services. Once granted, the app will use the device's GPS to determine speed of the user. The speed is used to determine whether the fast or slow playlist should be suggested.
- Light sensor implementation
    - The app has a button called "Light Sensor". When clicked, the device's light sensor will calculate the amount of light the device currently detects. This value is used to determine whether the day or night playlist should be suggested.
- Item info
    - At the right side of each playlist's row, a little "i" icon can be seen. When clicked, this brings up a new page that talks about the details of the playlist. At the bottom, it lists all songs in a playlist. The user can manually select a song to play

by tapping on the text of the song. A back button is located at the very bottom to return to the main screen.

- Controlling a playlist
    - When a song is being played, the song's name, artist, and album name will be displayed. The user can pause, skip forwards, and skip backwards on a playlist. To play a different playlist, the user can simply tap the row of the desired playlist. These features were all implementations of functions available in the Spotify API for Android development.
- Playlist suggestions
    - When the user performs a certain action (moving quickly/slowly, being in a lit/dark area) an alert will pop up asking the user if he/she desires to play a different playlist. Selecting yes will start the playlist, no will make the alert go away.

Testing Methodologies

Testing this app mainly involved trying to perform an action to see if the app will respond the expected way. This included playing a song correctly, skipping to a specific song, checking to make sure the right metadata was being displayed, etc. If anything was wrong, the code would be changed to correct the error. If any unexpected error occurred, such as a crash, the app was run with the debugger attached. The debugger usually printed a clear trace on where the error occurred, which was then promptly fixed.

Usage

The app **REQUIRES** a premium Spotify account to be used to login. The Spotify API for Android needs a premium account in order to work properly. Please contact either Jason or Dennis for a premium account info if one is not available.

The GPS works a lot better when outdoors. It had been suggested to use the NETWORK_PROVIDER, but that does not return a speed. The GPS_PROVIDER was used instead. When testing that feature, go outside and start walking around quickly to see if an alert pops up. The GPS updates every 10 seconds.

The light sensor takes a little bit of time to adjust between light and dark. To test for the night playlist, cover up the top of the tablet (where the camera is) and wait for the tablet's screen to dim a bit, then tap the light sensor button. For day playlist, place the tablet in a well lit area and wait for the screen to brighten back up.

Lessons Learned

Developing the app proved to be challenging in a number of different ways. One of the challenges was implementing the GPS. Getting permission handling was rather difficult, and the app would crash frequently when the app attempted to access the GPS without permission. It was discovered that checking for permission occurred after attempting to access the GPS, so moving the permission handling before that fixed the problems. Another challenge with the GPS was to calculate speed. The NETWORK_PROVIDER of the GPS would not return speed, so the GPS_PROVIDER was used instead. The speed was calculated by putting the latitude and longitude in an equation that returned the desired values. The GPS also did not work well indoors, so testing this feature had to be done outdoors.

Another lesson learned was connecting an app to Spotify. The Spotify API for ANDROID is still under development, and it is constantly updated. The documentation was not as robust as expected, which made authentication more difficult. There were examples of how to authenticate online, but some example code was outdated. The correct code was written after contacting Spotify's API Android team for some help. Once authenticated, an instance of a

Spotify player class was created in order to start playing songs. Getting a song to play correctly was a big achievement, and it made the rest of the interfacing with Spotify a little easier.

Another challenge was to moderate the number of alerts the user would receive. Some features, such as the light sensor, would update every second, and after each update, a new alert dialog would pop up on the screen. This would continue until the app became overwhelmed and crashed to the home screen. To fix this problem, boolean variables were used to check which alert would pop up on the screen. The boolean would then be set to true to indicate that this alert was currently being created and no further instances would be required unless the user changed to a different state. Once the user changed to a different playlist, then the boolean variable was set to false, indicating that the alert could be created again.

Wireframe

The wireframe proved to be useful in brainstorming for the project. It allowed the partners to get a good idea on what they wanted the app to look like, and it also let them determine which features of the app were most prominent to display. However, some ideas on the wireframe were not able to be implemented. An initial idea was to display a miniature Spotify player in an activity, but it turns out the Android Spotify API does not have any visuals. Thus, a bare-bone music player was created for the app.

Having a wireframe made placing activities very clear. Visualizing how activities would overlap each other made making segways easy to do. Once completed for one activity, it was not difficult to repeat the task for other activities. The program used to create the wireframe was Balsamiq.