Final Project Paper

Introduction:

The primary objective of our mobile application will be to design a basic music player with no advertisements or bloatware. Current existing audio players heavily rely on streaming features that not all users can take advantage of, or serve so many ads that it hinders the experience. We will accomplish this by designing two main pages, one for a music library that lists files with a list view, and another with a screen for playing audio with controls. The main features of this app are as follows:

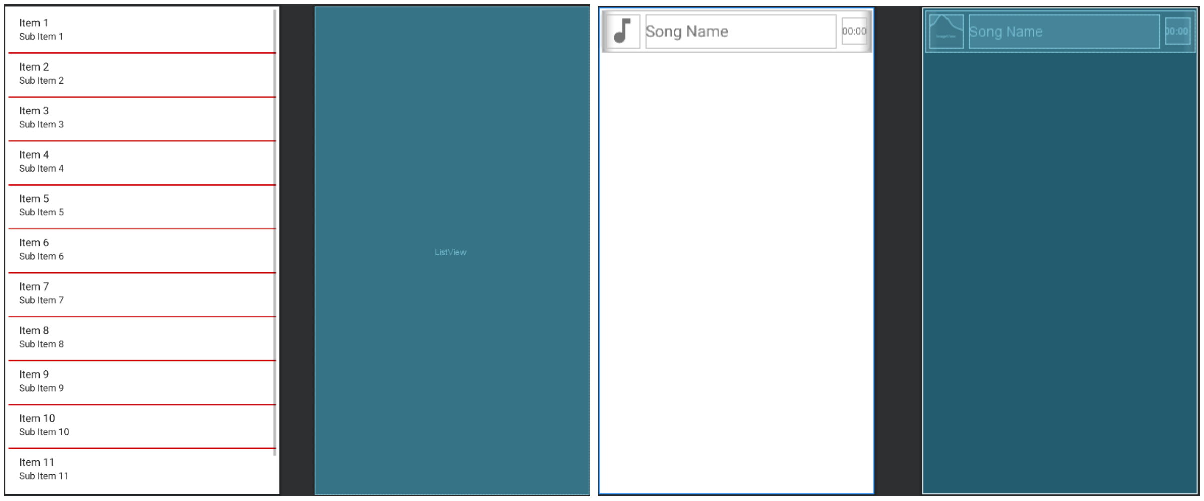
1. Availability of files on local storage. The app will request permissions to access local files in order to retrieve songs currently on the device.
2. Playing screen showing song that is currently playing with controls.
3. Continue playing music in the background.
4. Add a visualizer for audio to make the application more interesting.

We are interested in doing this project because while people oftentimes use streaming services nowadays, many others lack the sufficient internet connection and data plan to do so. Many existing music players serve constant advertisements and have clunky hard to use interfaces. We plan to make a basic one that serves no advertisements and has a list of music, and a simple player screen. We plan to use a third party library for audio visualization found here: https://github.com/gauravk95/audio-visualizer-android. Aside from that, most features we use will be stock Android ones.

Our target is any Android device running API level 24 and higher. The target SDK is 30. The envisioned user group is anyone who stores files locally on their device instead of relying on streaming services.

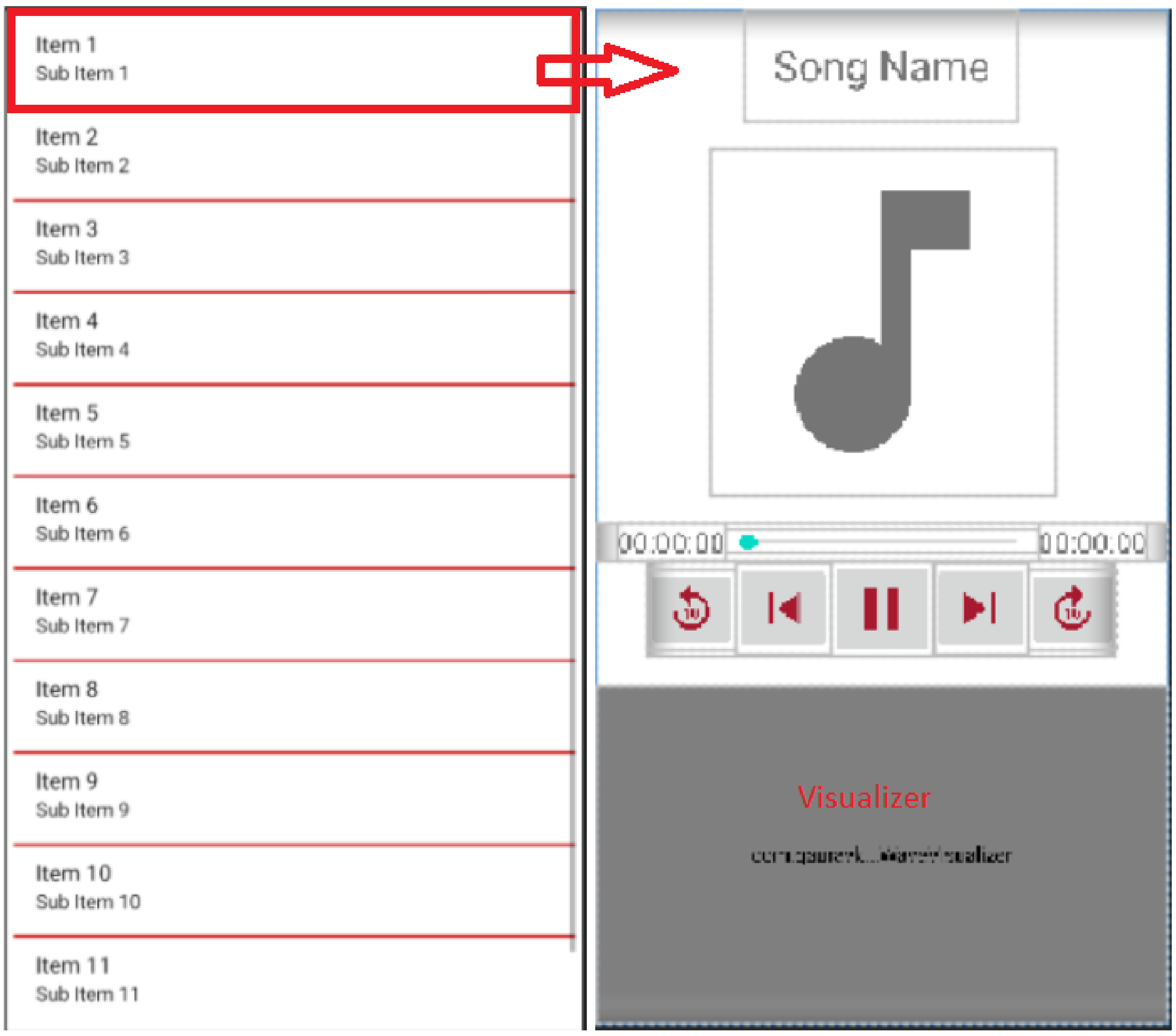
UI Mockup Design

This application will consist of a list view of songs, and a player screen with controls and an image.



Above is an example of the main view the user should see upon starting the app. This is an xml file generated and displayed in Android Studio. It displays a list of music files stored on the device. The list item view is on the right which consists of the image from the song metadata, the title of the file, and the length of the song. We will also use Gson to allow the passage of object data between activities by converting them to JSON strings and back.

The control flow is as follows:



As you can see, when a user clicks on a song, it will take them to the playing screen where they have a visualizer and basic audio controls. Upon pressing the back button from the player activity, the application will stop all audio playback and return to the song selection screen.

Functional Design

We plan on using the MVC pattern for designing our application for the play screen so data persists upon device change states. For our mobile sensor we are using the microphone to route audio being played by MediaPlayer to generate a visualization graph using this third party library https://github.com/gauravk95/audio-visualizer-android. To use this, we will add an import and create an instance of a WaveVisualizer in the XML file in our player. For the networking component we will have a button on the main page to open a WebView to download media files to play. In order to meet the persistence requirement, we plan on storing a small local file within the app files that contains the path to the last played song. Upon starting the app, if this path exists, the audio player will resume at that song.

We plan on using a third party library linked above to generate a live visual graph of the current sound being played. This will use the device microphone to record audio from the media player and generate the graph.