Capstone Stage 1

Description
- Preamble
- Problem
- Proposed Solution

Intended User

Features

User Interface Mocks
- Onboarding
- Initial Network Scan
- Navigation Drawer
- All Servers
- Discovery Status
- Folder Navigation
- Leaf Folder
- All Playlists
- Edit Playlist
- Now Playing
- Search
- Search Results
- Favorite Folders
- Widget
- Persistent Notification
- Lockscreen

Key Considerations
- How will your app handle data persistence?
- Describe any corner cases in the UX.
- Describe any libraries you'll be using and share your reasoning for including them.
- Describe how you will implement Google Play Services.

Next Steps: Required Tasks
- Task 1: Project Setup
- Task 2: Implement Database and Content Provider
- Task 3: Implement UI for Each Activity and Fragment
- Task 4: Create Samba Service to Scan Network
- Task 5: Implement a Tiny Web Server
- Task 6: Create the Node Navigation Object and Breadcrumb UI
- Task 7: Implement Firebase Analytics
- Task 8: Integrate with parts of Universal Music Player
- Task 9: Implement Google Cast

GitHub Username: DerekMorrison59

Network Music Player

Description

You’ve got lots of MP3 files available on your home network. Wouldn’t it be nice to be able to play them on your phone while you are at home? Network Music Player frees that music and makes you happy!

Preamble

Many people have music files located on shared network folders on their home network. These music files are often created by converting a CD collection into mp3 files and sometimes they are purchased and downloaded from online stores. The following illustrates how you might create a music listening app that can play files from a local network in the context of the Android Fundamentals project specifications.

Problem

The user has a number of mp3 files that are stored on local shared folders and wants an easy way to listen to this music. The user does not want to copy all these files onto a specific device in order to do this. He / she wants a way to find and play songs on the local network via WiFi .

Proposed Solution

Design an app that connects to the local network via WiFi, scans for servers / folders and mp3 files and then presents them to the user for selection. The user can play songs, make playlists, navigate around the local shared folders to pick favourites and request in-depth scanning to search for more song files.
The usual music controls are available to play / pause, next, previous and a progress indicator. Album art is displayed while a song is playing. The user can search for a song and will be presented with a list of results from across all known folders.

Intended User

People that ripped their CD library to MP3s (or purchased a large number of songs) and now want to access from their phone while at home using the WiFi network.

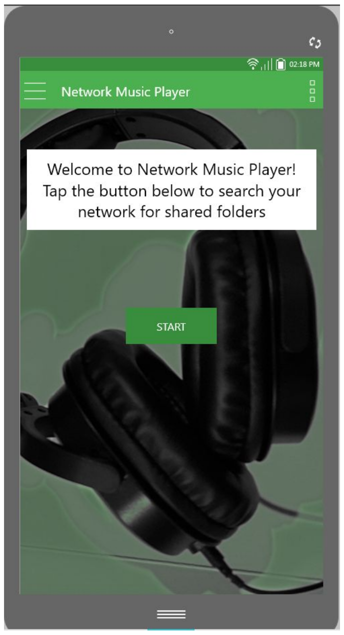
Features

- Scans local network for servers and shared folders
- Allows users to easily create playlists
- Search for songs
- Play individual songs or an entire playlist
- Returns to exactly where you left off

User Interface Mocks

These were created using MockPlus

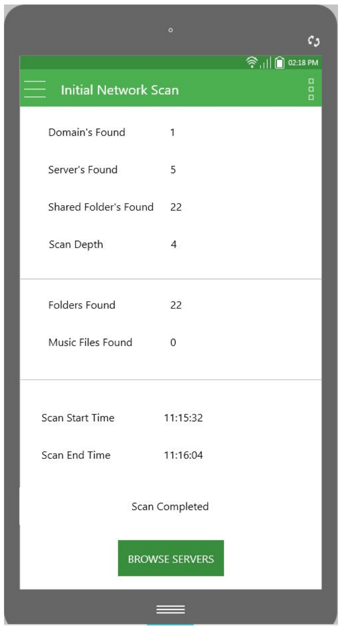
Onboarding



Welcome to Network Music Player! Tap the button below to search your network for shared folders

This is the first screen that the user will see when the app is opened for the first time. One tap and the app searches the local network for shared folders and proceeds to the next screen.

Initial Network Scan

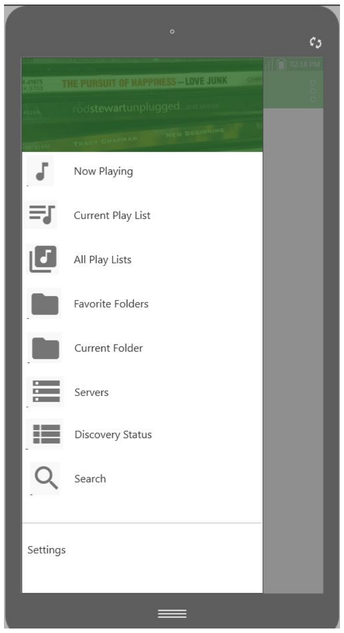


Domain's Found	1
Server's Found	5
Shared Folder's Found	22
Scan Depth	4
Folders Found	22
Music Files Found	0
Scan Start Time	11:15:32
Scan End Time	11:16:04

Scan Completed

This screen is provides the user with the results as the initial scan progresses.

Navigation Drawer

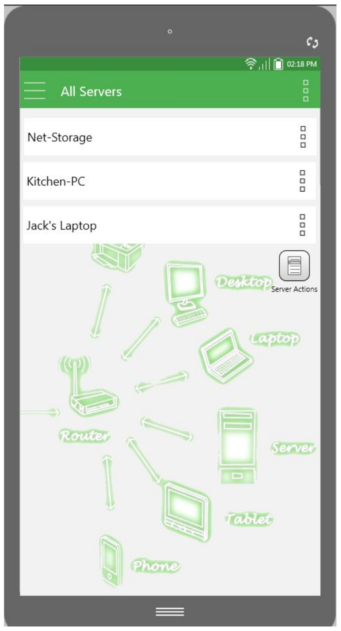


The navigation drawer provides the user with direct access to the following screens

- Now Playing
- Current Playlist
- All Playlists
- Favorite Folders

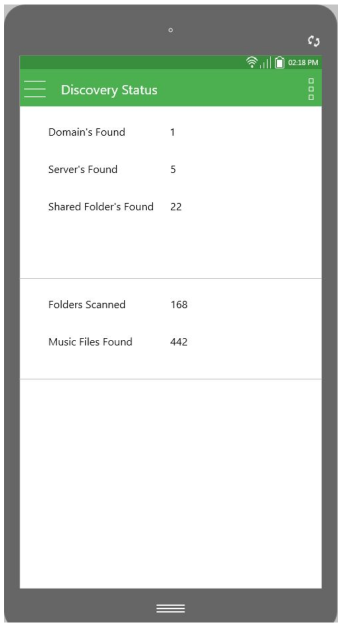
- Current Folder
- Servers
- Discovery Status
- Search
- Settings

All Servers



This screen shows the user all the servers that were found on the network. This is essentially the top of the navigation tree.

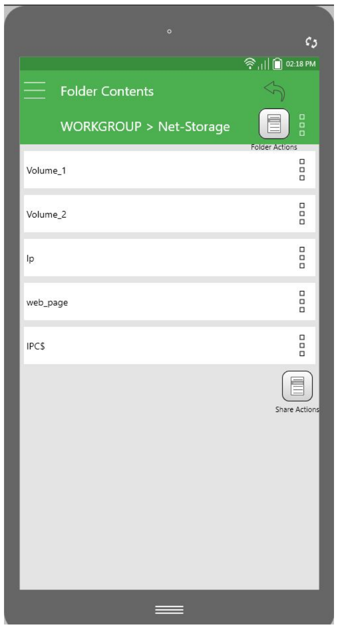
Discovery Status



Domain's Found	1
Server's Found	5
Shared Folder's Found	22
Folders Scanned	168
Music Files Found	442

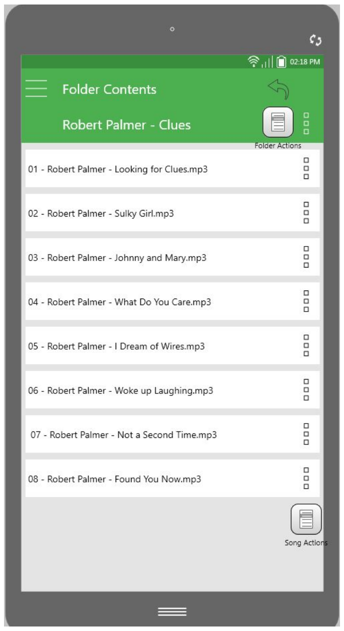
This screen shows the user details of local network that have been recorded by this app. During the Initial Network Scan the first few levels of the network were explored and saved. As the user navigates to new folders the app remembers each folder and song file found.

Folder Navigation



This screen allows the user to explore the directory hierarchy. The current location is shown in a breadcrumb style at the top of the screen. The user can go back up using the ‘up’ button or click anywhere in the breadcrumb list to jump back multiple levels. Clicking on a directory will move focus into the selected directory and the screen will update to reflect new current location's contents.

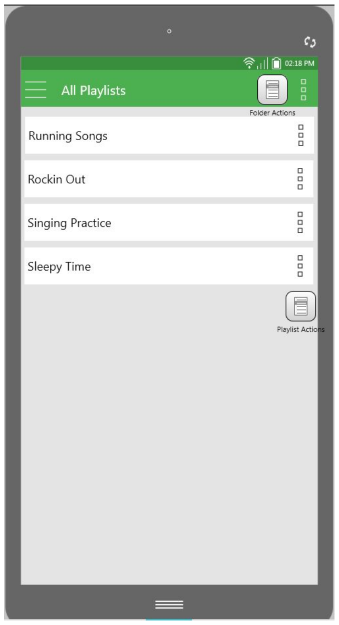
Leaf Folder



01 - Robert Palmer - Looking for Clues.mp3
02 - Robert Palmer - Sulky Girl.mp3
03 - Robert Palmer - Johnny and Mary.mp3
04 - Robert Palmer - What Do You Care.mp3
05 - Robert Palmer - I Dream of Wires.mp3
06 - Robert Palmer - Woke up Laughing.mp3
07 - Robert Palmer - Not a Second Time.mp3
08 - Robert Palmer - Found You Now.mp3

When the user reaches the end of a branch it will probably be a folder containing music files. The user can play a song or add a song or songs to a playlist. Pressing back will return to the previous node in the folder tree.

All Playlists



Running Songs

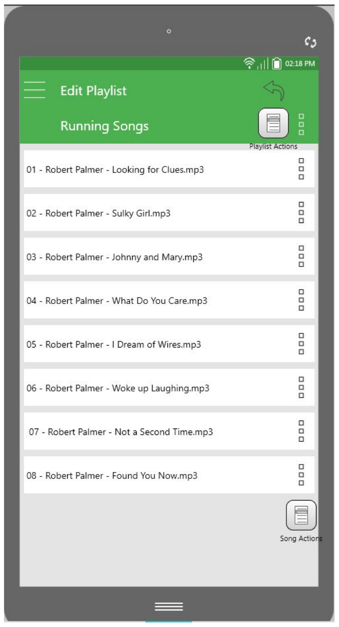
Rockin Out

Singing Practice

Sleepy Time

This screen displays all the playlists the user has created. The playlists can be copied, deleted or played.

Edit Playlist



01 - Robert Palmer - Looking for Clues.mp3

02 - Robert Palmer - Sulky Girl.mp3

03 - Robert Palmer - Johnny and Mary.mp3

04 - Robert Palmer - What Do You Care.mp3

05 - Robert Palmer - I Dream of Wires.mp3

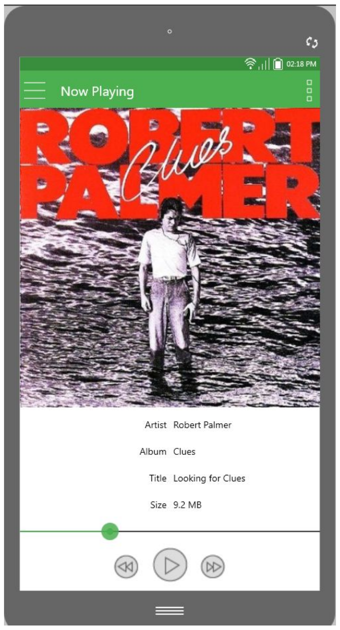
06 - Robert Palmer - Woke up Laughing.mp3

07 - Robert Palmer - Not a Second Time.mp3

08 - Robert Palmer - Found You Now.mp3

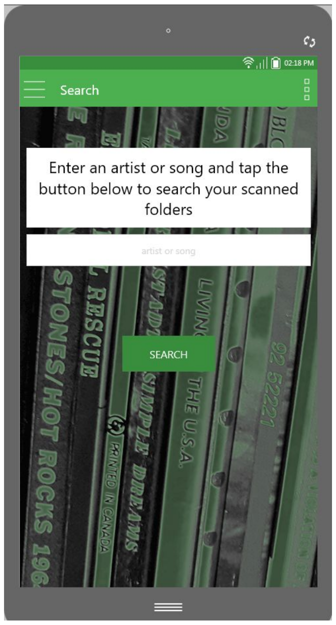
The user can select a playlist from the screen above to view and edit it’s contents. Songs can be removed or re-arranged from here. This playlist can be added to another playlist

Now Playing



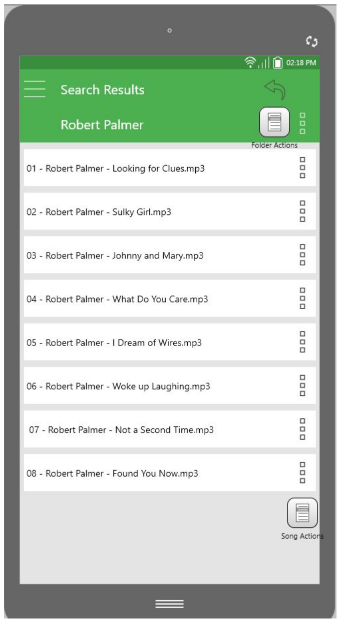
This screen shows the user details of the song that is currently being played. It provides the usual controls to pause / play and skip forward or back in the current playlist.

Search


Search
Enter an artist or song and tap the button below to search your scanned folders
artist or song
SEARCH

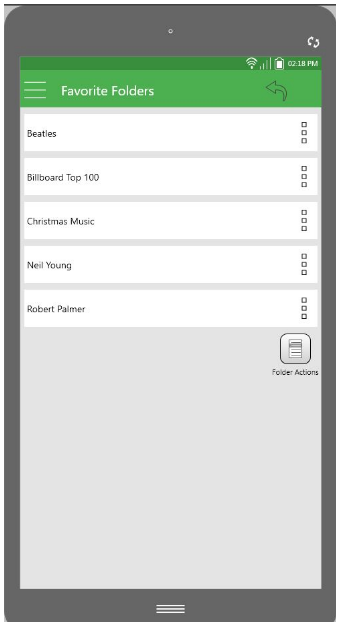
This screen allows the user to search the scanned directories for a song or artist.

Search Results


02:18 PM
Search Results
Robert Palmer
Folder Actions
01 - Robert Palmer - Looking for Clues.mp3
02 - Robert Palmer - Sulky Girl.mp3
03 - Robert Palmer - Johnny and Mary.mp3
04 - Robert Palmer - What Do You Care.mp3
05 - Robert Palmer - I Dream of Wires.mp3
06 - Robert Palmer - Woke up Laughing.mp3
07 - Robert Palmer - Not a Second Time.mp3
08 - Robert Palmer - Found You Now.mp3
Song Actions

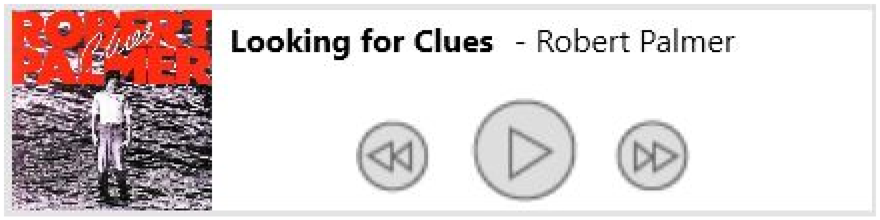
This screen shows the user the search results. From here the songs can be played or added to a playlist. Pressing back returns to the Search screen

Favorite Folders



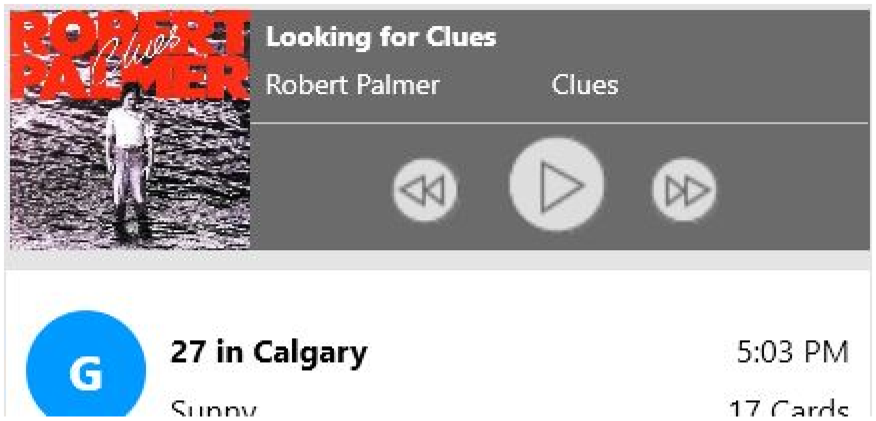
This screen shows all the folders the user has marked as a favorite. From here the user can explore any of the folders marked as Favorite.

Widget



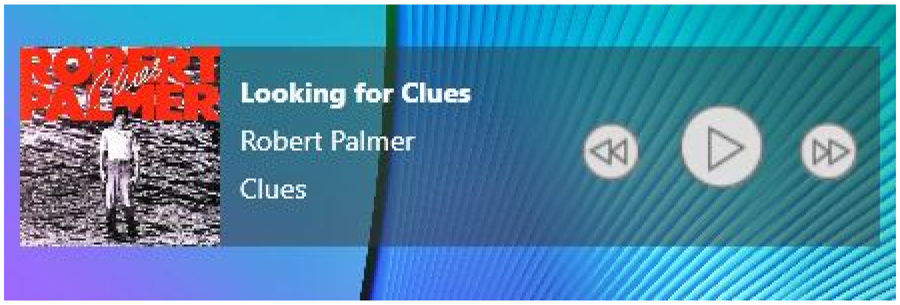
This image shows the widget provided with this app. The user can see basic information about the current song and interact with the player to pause / play, skip back or forward one song.

Persistent Notification



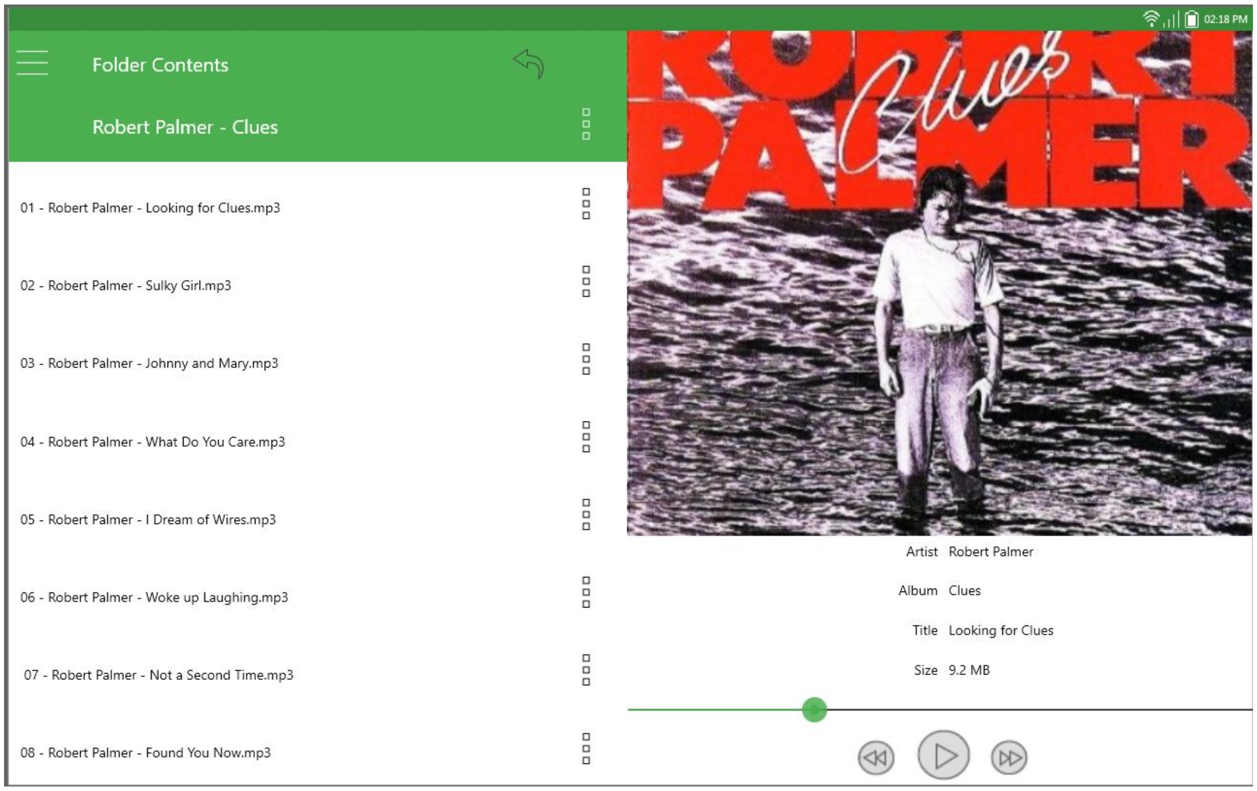
This image shows a persistent Notification provided by this app when a song is playing or recently paused. The user can see basic information about the current song and interact with the player to pause / play, skip back or forward one song.

Lockscreen



This image shows the lockscreen widget that appears when a song is playing. The user can see basic information about the current song and interact with the player to pause / play, skip back or forward one song.

Tablet Version



01 - Robert Palmer - Looking for Clues.mp3
02 - Robert Palmer - Sulky Girl.mp3
03 - Robert Palmer - Johnny and Mary.mp3
04 - Robert Palmer - What Do You Care.mp3
05 - Robert Palmer - I Dream of Wires.mp3
06 - Robert Palmer - Woke up Laughing.mp3
07 - Robert Palmer - Not a Second Time.mp3
08 - Robert Palmer - Found You Now.mp3

Artist	Robert Palmer
Album	Clues
Title	Looking for Clues
Size	9.2 MB

This screen provides an example of how the previous screens (as Fragments) can be used together on a tablet with more space Above shows the folder view on the left and now playing on the right. The user could see direct action from tapping a song in the left list and having it start playing on the right.

Key Considerations

How will your app handle data persistence?

Meta data for the files discovered on the network will be stored in a SQLite database.
An IntentService will be used to retrieve file data from the network and store it in the database.
A Content Provider will provide access to the database for various screens.
Shared Preferences will be used to remember the current settings like song now playing, screen being displayed and selected playlist. (app will use these to return to the state where the user left off)

Playlists will be stored in the SQLite database
Additional song metadata will be optionally retrieved from the internet using the last.fm api.

Describe any corner cases in the UX.

The user has access to all entry points in the UI from the Navigation Drawer. Several of the screens are endpoints and therefore do not offer further navigation themselves. The user can use the Navigation Drawer to immediately jump to another screen. For example, if the user was viewing the All Servers screen it would be possible to go directly to the Now Playing screen using the Navigation Drawer.

Describe any libraries you'll be using and share your reasoning for including them.

Picasso to handle the loading and caching of images
JCIFS - Java CIFS/Samba library used to connect to the local network and scan for Servers, File Shares, Directories and music files (https://jcifs.samba.org/)

Describe how you will implement Google Play Services.

- Firebase Analytics - monitor specific actions (ex. time for initial scan) and how long the app is used
- Google Cast v3 - Cast the audio stream to a nearby cast device

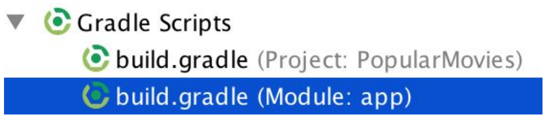
Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Add Picasso and JCIFS libraries to the project.

To add Picasso to the project, start by modifying the build.gradle file for your app. These modifications happen in the build.gradle file for the module's directory, *not* the project root directory (it is the file highlighted in blue in the screenshot below).



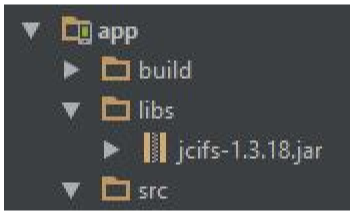
Gradle Scripts
build.gradle (Project: PopularMovies)
build.gradle (Module: app)

In your app/build.gradle file, add:

```
repositories {
    mavenCentral()
}
```

Next, add compile 'com.squareup.picasso:picasso:2.5.2' to the dependencies block.

To add the JCIFS library to the project start by downloading the jar file from https://jcifs.samba.org/ and adding it into the libs directory of the app.



```
app
  build
  libs
    jcifs-1.3.18.jar
  src
```

Next, add the following line to the dependencies block of the app's build.gradle file

```
compile files('libs/jcifs-1.3.18.jar')
```

Task 2: Implement Database and Content Provider

- Create SQLite database to store music file metadata and playlists
- Create some basic Android Tests to ensure the database works properly
- Create a Content Provider for the database

Task 3: Implement UI for Each Activity and Fragment

Define project colors and font sizes and inherit from an AppCompat style

Build the UI layouts for the following screens:
- Onboarding
- Initial Network Scan results
- All Servers

- Folder Navigation
- Leaf Folder
- All Playlists
- Edit Playlists
- Now Playing
- Search
- Search Results
- Favorite Folders

Task 4: Create Samba Service to Scan Network

Implement a service to scan local network and add the results to the database

- Create an Intent Service that receives a node ID, queries the network for child nodes and then does a bulk insert to the database.

Task 5: Implement a Tiny Web Server

Implement a service to serve local network files (via samba) using NanoHTTPD (https://github.com/NanoHttpd/nanohttpd)

- Create a local web server (127.0.0.1) that accepts requests from the music player and serves the requested music file over http.

Task 6: Create the Node Navigation Object and Breadcrumb UI

Provides the user with a system for navigating the node tree (folders and sub-folders).
It will be provide the following features:

- Keep track of the current node
- Maintain a list of previous nodes (a path that goes back to the root node)
- Method to go back one level (previous node)
- Method to jump to any of the previous nodes (one tap to go to any previous node)
- Method to add a new node to the end of the list (new current location)

Build a breadcrumb UI fragment to work with the Node Navigation object. This fragment will show the current path with the current node visible and the other nodes floating off to the left

Task 7: Implement Firebase Analytics

Prior to adding any Firebase feature to the project it is necessary to sign up and establish the app on the website, use the directions here:
https://firebase.google.com/docs/android/setup

The detailed steps to add Firebase Analytics to a project are well documented on the website (https://firebase.google.com/docs/analytics/android/start/). Basically it consists of:

Adding a dependency to the root-level build.gradle file

- `compile 'com.google.gms:google-services:3.0.0'`

Adding a dependency to the app-level build.gradle file

- `compile 'com.google.firebase:firebase-core:9.4.0'`

Declaring a FirebaseAnalytics object at the top of MainActivity

- `private FirebaseAnalytics mFirebaseAnalytics;`

Initializing it in the onCreate() method

- `mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);`

Creating a Bundle and load it with parameters

- `Bundle bundle = new Bundle();`
- `bundle.putString(FirebaseAnalytics.Param.ITEM_ID, id);`
- `bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, name);`
- `bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "image");`

Calling logEvent and specify the Event type and the bundle

- `mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);`

Task 8: Integrate with parts of Universal Music Player

Use the example code from Universal Music Player to understand how to design the connections and interactions of the key parts of a modern music playing app. Specifically, how the following classes are used:

- MediaPlayer
- MediaBrowserCompat
- MediaBrowserServiceCompat
- MediaControllerCompat
- MediaDescriptionCompat
- MediaSessionCompat
- MediaSessionCallback
- MediaMetadataCompat
- PlaybackStateCompat
- BroadcastReceiver

Task 9: Implement Google Cast

Make the app capable of casting the playing music to a Chromecast device. The detailed steps to add Google Cast to a project are well documented on the website (https://developers.google.com/cast/docs/android_sender_setup).

Add Google Play Services to the Project by ensuring the dependencies block contains

- `compile 'com.android.support:appcompat-v7:23.4.0'`
- `compile 'com.android.support:mediarouter-v7:23.4.0'`
- `compile 'com.google.android.gms:play-services-cast-framework:9.2.0'`