Software Design Document

Mulink

Justin K. Joseph

Harrisburg University

Contents

Date

1. Introduction

1.1 Purpose

This document outlines the system design of the application ‘MuLink’. While this serves as an outline, the design of the application is subject to change and has already gone through many during development.

1.2 Intended Audience

This document is intended for anyone interested in the development of MuLink. The end product was initially intended for users of mainstream music streaming platforms that have trouble finding songs on certain platforms, constantly making them switch back and forth between the platforms to listen to their favorite songs. This functionality was partially inspired by the music bots that were available on the social platform Discord.

After technical and logistic issues, the scope of the project had to be reconsidered. MuLink was thereafter designed to be more of a playlist manager, which modifies the intended audience of the end product to anyone looking to better organize their songs across music platforms.

1.3 References

[0]	S. Dudeja, *Why is YouTube Taking Down Popular Discord Music Bots?* 2021 [Online]. Available: https://fossbytes.com/youtube-taking-down-discord-music-bots/. Accessed: April 4, 2025.

Date

[1] Spotify. “Spotify Developer Policy”. [Online]. Available: https://developer.spotify.com/policy (accessed April 4, 2025).

[2] The YouTube API Services. “YouTube API Services Terms of Service”. [Online]. Available: https://developers.google.com/youtube/terms/api-services-terms-of-service (accessed April 4, 2025).

[3] *YouTube API Services - Developer Policies*. The YouTube API Services Team [Online]. Available: https://developers.google.com/youtube/terms/developer-policies (accessed April 4, 2025).

2. System Overview

2.1 Purpose

MuLink is an application intended to integrate multiple streaming services to allow for mixed functionality across all user-linked platforms. The scope of this was initially broad, but limitations reduced this scope mainly just to playlist management across services.

2.2 Functionality

Initially, the functionality of MuLink was planned to allow for playing music, creating playlists, and tracking listening statistics using songs across linked platforms. YouTube, Spotify, and SoundCloud were planned to be supported for this.

In the planning and research phase of this project, it was discovered that while Spotify and YouTube appear to track the listening statistics of users within their own websites, they do not provide any way to access this data through their APIs. This feature was therefore scrapped.

More importantly, planning and research also explored concerns of if music playback functionality would be allowed under the terms of services of supported platforms. While this project was partially inspired by Discord music bots, the same bots were taken down due to violations of YouTube's terms of service[0]. While the specific criteria under which the terms of service were violated in the music bots remains unclear, some notable criteria under which MuLink could possibly violate terms of service include Section III, part 5 of Spotify's developer terms of service[1], as well as Section 12 of YouTube's API terms of services[2] and Section C, part 5 of YouTube's developer policies[3].

With the change of scope in the project, MuLink was repurposed to support functionality for creating playlists. Functionality to populate the playlists to those of all linked platforms was also planned. Playlist data was saved locally, which also was meant to support the ability for users to import and export pre-generated playlist data.

3. System Architecture

MuLink is an application meant to be run on Android devices. The software will be created using the Android Studio, and the main software will be written in Java. It will be deployed as an Android APK and will include all necessary files for operation with the exception of configuration files that contain keys to Youtube and Spotify APIs. These keys can be obtained through developer accounts on the respective platforms.

Internet connection and permissions will be required to use any feature that involves sending playlist data to platforms or searching songs to insert into playlists. Internet permission can be granted by finding the app through the Android settings menu, navigating to the privacy category, and changing allowed permissions. Playlist data will be saved locally and will not require internet functionalities to view. However, adding new songs to playlists will require internet.

Date

MuLink is still in development, and this document is not entirely reflective of the final product.

4. Data Design

4.1 Data Classes

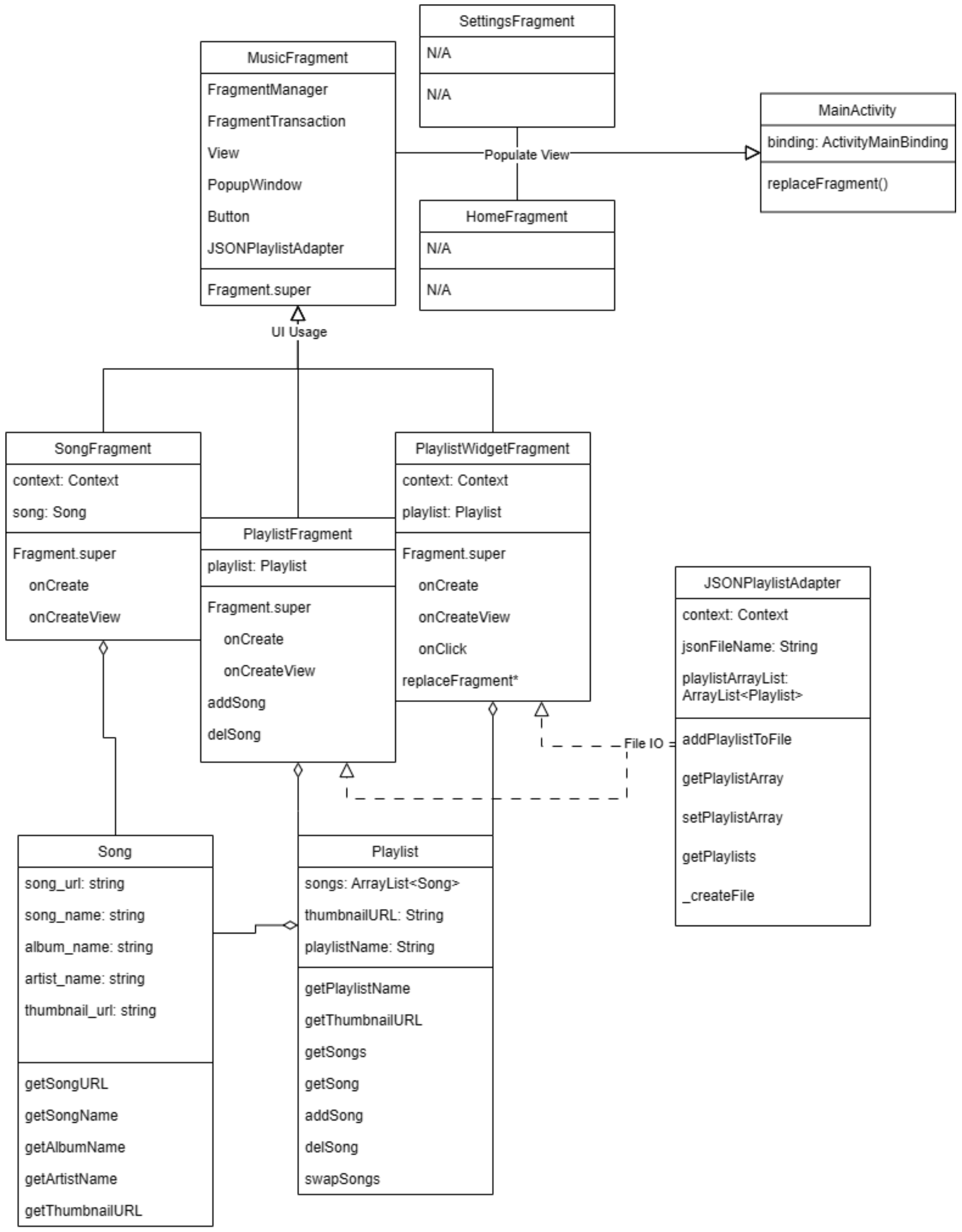


Figure 1: MuLink Data Diagram

5. Component Design

This section is meant to explain how the components of MuLink form system functionality.

5.1 User Interaction and File I/O

MuLink populates playlist data in the UI by accessing a .JSON file on local storage that contains the data. If the file does not exist, a new file is created. Any changes to the playlists made through the UI should update in the local file. While not initially included in the scope of this project, this functionality could be modified to allow for users to export and import playlist data from other users.

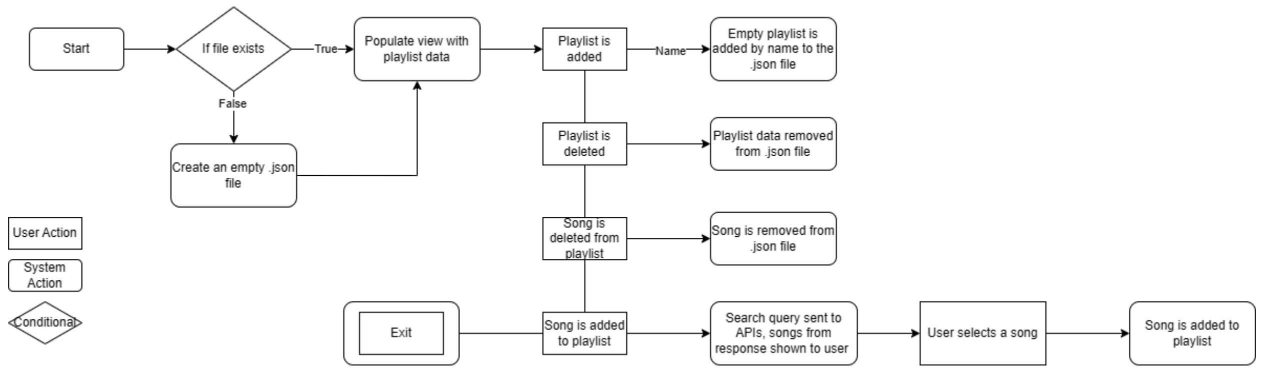

Start
If file exists
True
False
Populate view with playlist data
Create an empty .json file
Playlist is added
Name
Empty playlist is added by name to the .json file
Playlist is deleted
Playlist data removed from .json file
Song is deleted from playlist
Song is removed from .json file
Song is added to playlist
Search query sent to APIs, songs from response shown to user
User selects a song
Song is added to playlist
Exit
User Action
System Action
Conditional

Figure 2: User Interaction Component Design

6. Human Interface Design

This section is meant to show the graphical design of MuLink. This section will include initial sketches of the UI as well as updated designs where applicable.

6.1 Home Screen

This is the default fragment that MuLink displays by default. This view was initially meant to display snippets derived from listening statistics such as recently played songs, most played songs, and favorite artists. More generally, this page was meant to be an adaptable space for any general information—such as update notices—to be displayed.

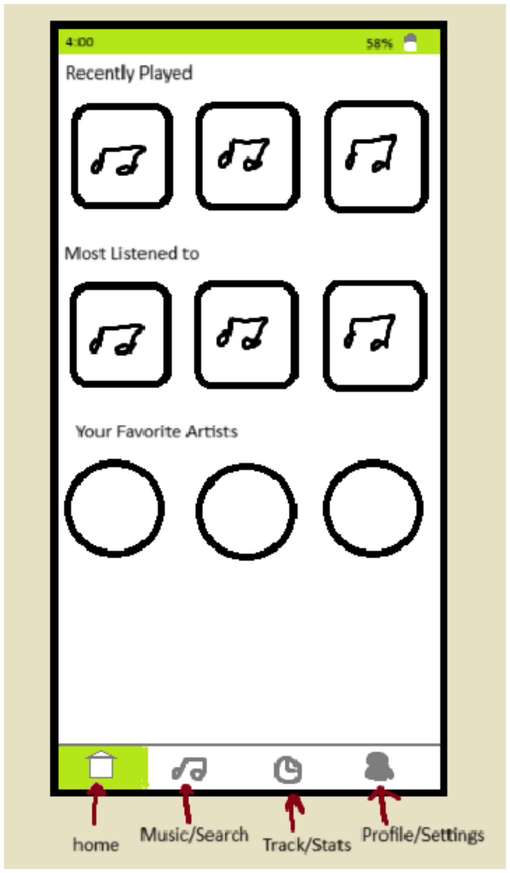


Figure 3: Initial Home Fragment Sketch

6.2 Music Fragment

The music Fragment is where the majority of functionality is located. Therefore, most of the interface is designed to be contained within this fragment.

6.2.1 Initial Sketch - Base View

The initial sketch of the music fragment was meant to show the user all of their added songs. Adding songs was to be done through the search bar at the top of the screen, which would send search for the songs using the API of supported platforms.

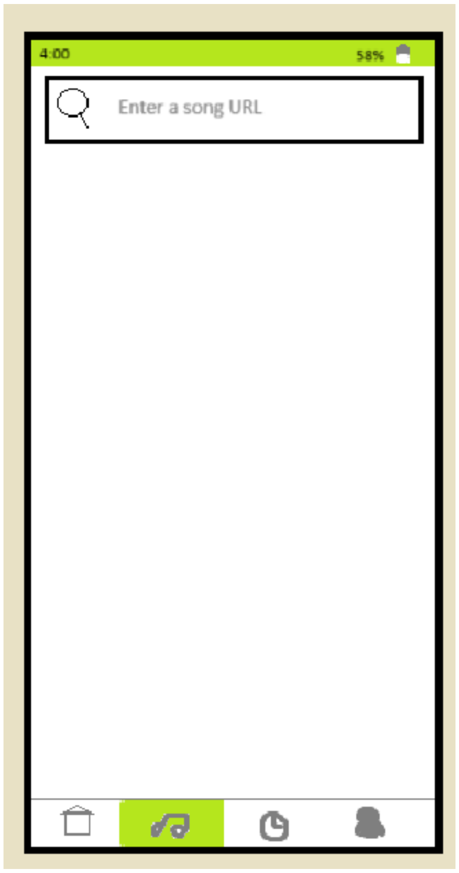


Figure 4: Initial Music Fragment Sketch - Base View

6.2.2 Initial Sketch - Song View

Selecting a song from the base music fragment would repopulate the view with the Song View Fragment. This view would allow users to play the songs in the app, see the album thumbnail, and redirect to the platforms that the song could be found on.

Date

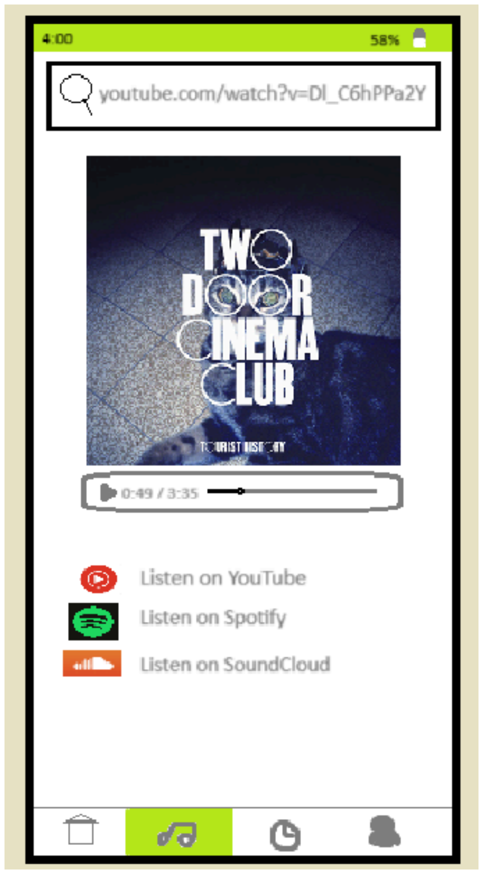
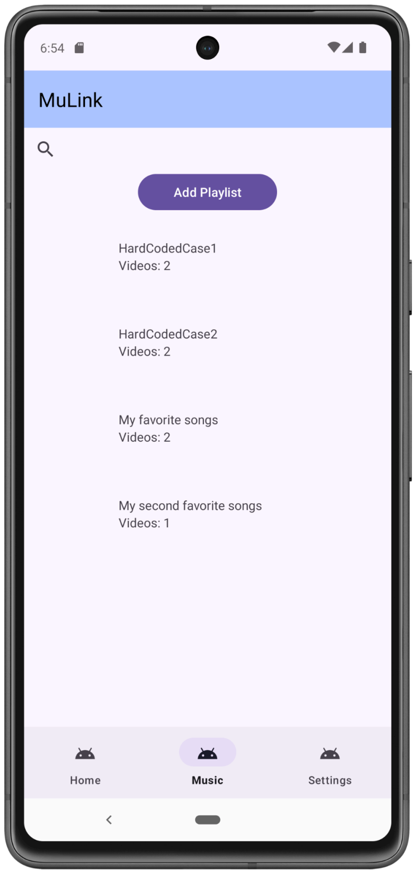


Figure 5: Initial Music Fragment Sketch - Song View

6.2.3 Current Design - Base View

After MuLink's change in scope, the design for the base view of the music fragment changed to show the user's playlists. While playlists can have thumbnails to represent them, the playlists in this example do not have a thumbnail.

Date



HardCodedCase1
Videos: 2

HardCodedCase2
Videos: 2

My favorite songs
Videos: 2

My second favorite songs
Videos: 1

Figure 6: Current Music Fragment Design - Base View

6.2.4 Current Design - Playlist View

With the playlists being shown in the updated base view, it was appropriate to include a view to see the songs within each playlist. The playlist view was made to show every song within the playlist. While not included in this example, this view was also meant to be the point through which users add and delete songs to their playlists.

Date

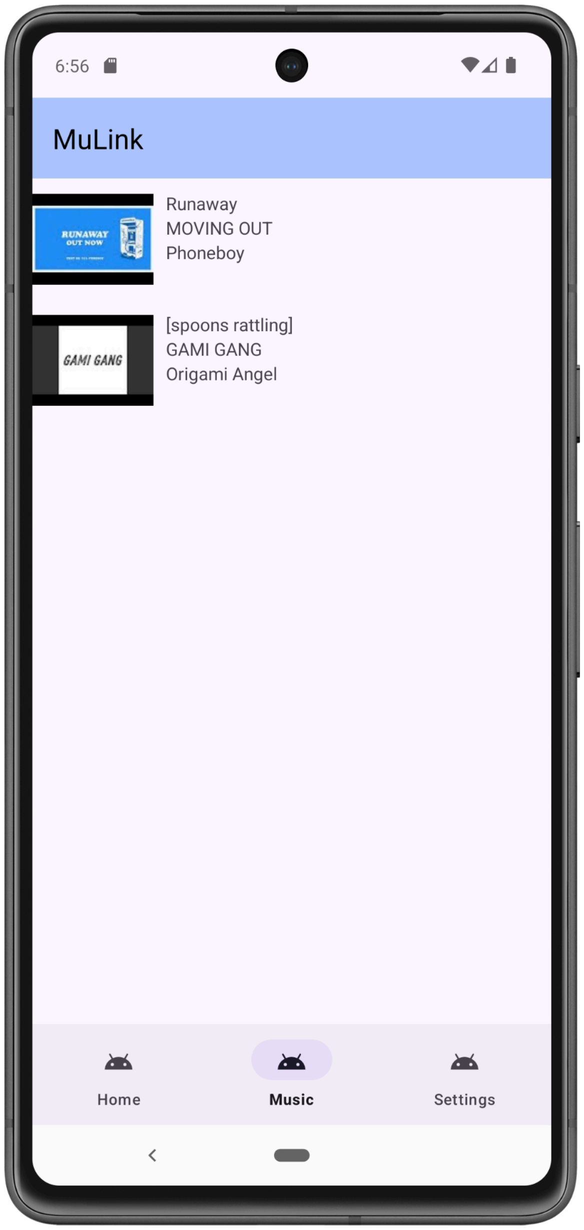


Figure 7: Current Music Fragment Design - Playlist View

6.3 Statistics Fragment

Date

The statistics fragment was originally slated to track and show listening data to users. This feature was cut from development plans due to platforms not supporting this through their APIs.

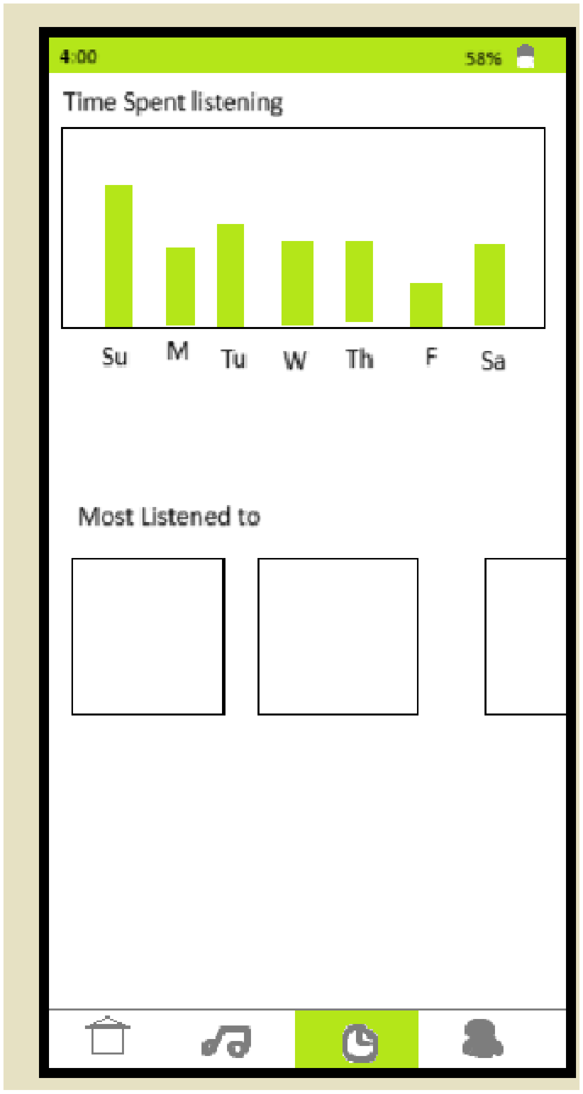


Figure 8: Statistics Fragment