Music Player

1. Record your reasons for implementing the solution the way you did, the struggles you faced, and the problems you overcame.

I used the MVVM pattern using SwiftUI, I used it to separate the view from logic and make the application more testable. MVVM stands for Model, View, View Model. It is a structural design pattern that allows for the separation of code into 3 groups. The view is the UI. The model is the business logic. The view model sits between the view and the model and is responsible for translating data back and forth between them. I went and explored different example code on the internet that uses this pattern and showed how the view and view model are connected using model binding and commands.

The struggle part :- because I recently started working on SwiftUI so faced few difficulties in building UI, and the most difficult section was the persistence player view , conceptually I know the solution but I wasn't aware how I should implement this in SwiftUI. But after lots of struggle I found a solution and implemented using **Overlay and Geometry** in SwiftUI.

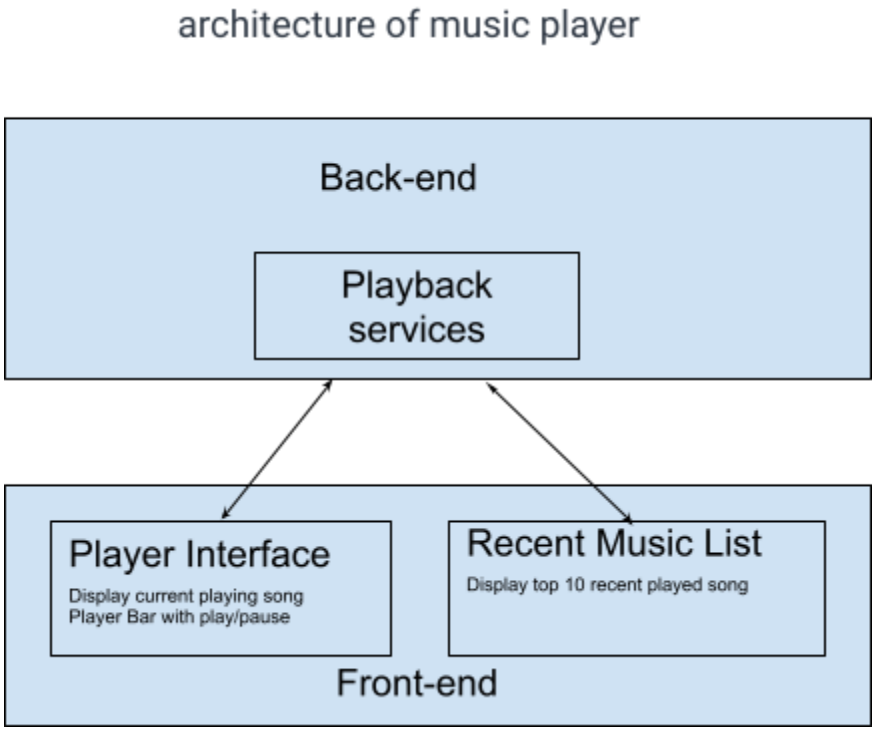
2. What shortcuts did you take that would be a bad practice in production?

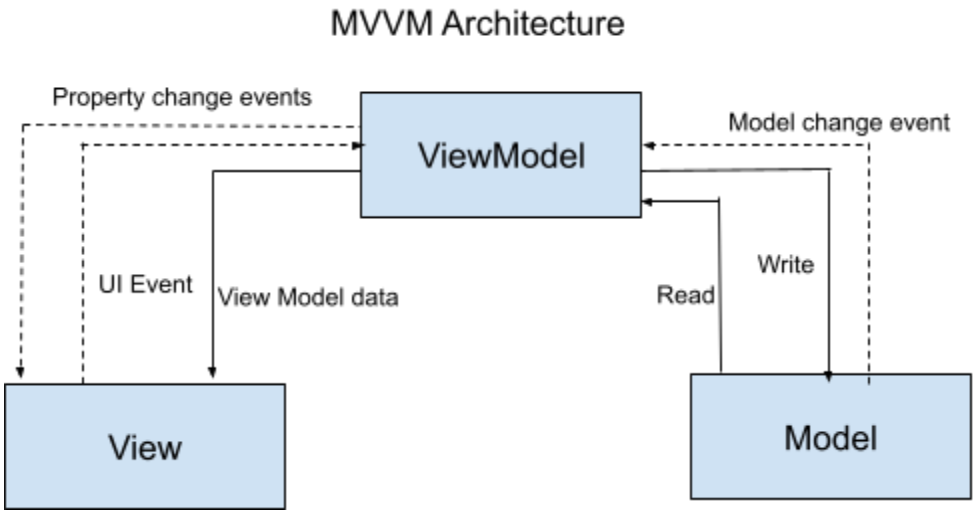
Not knowing how to optimize.
Ignoring error messages, exceptional handling.
Hardcoding values instead of making them configurable.
Blindly copy/pasting code.
Not taking the time to learn how things really work.
Ignoring proven best practices.
Not using Google enough.

3. What would you have done with more time? We know you have a life. :-)

I do social work specially during weekends or holidays and spend time with my family, friends and kids.

4. Please include an architecture diagram.





Player Flow

On after App launch, you will see two Tabs at the bottom of the screen first one is **Home View**, it will show you the main home screen where you can see current playing song thumb , song name and artist name on center of the screen and as well as on player bar , on player bar user can play and pause the song. Second tabs is to view all top recently played songs .

The Model

The application contains one model classes. It class holds the properties of a Song.
The model also contains a `PlayingMusic_Base` struct that holds a collection of songs that make up the list of songs to be played.

Services

The application contains one service called `APIService`. This service loads the songs into the song list from the provided API URL path.

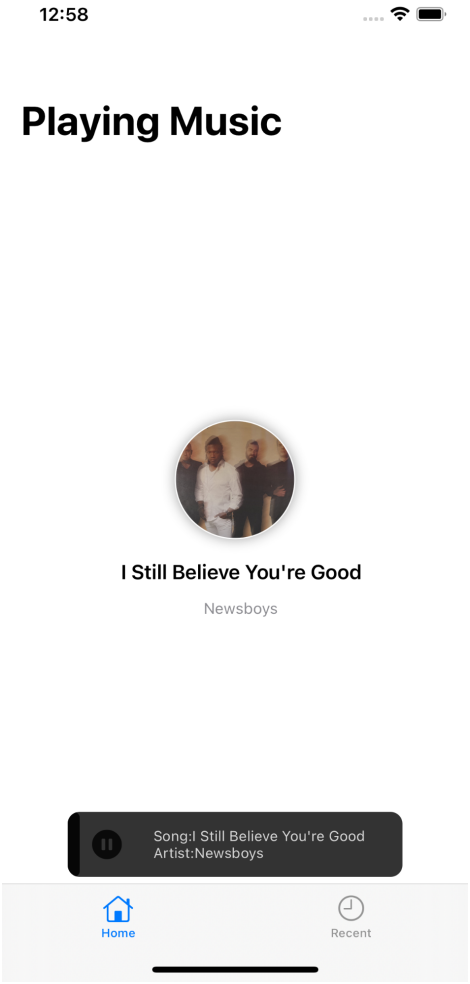
The View

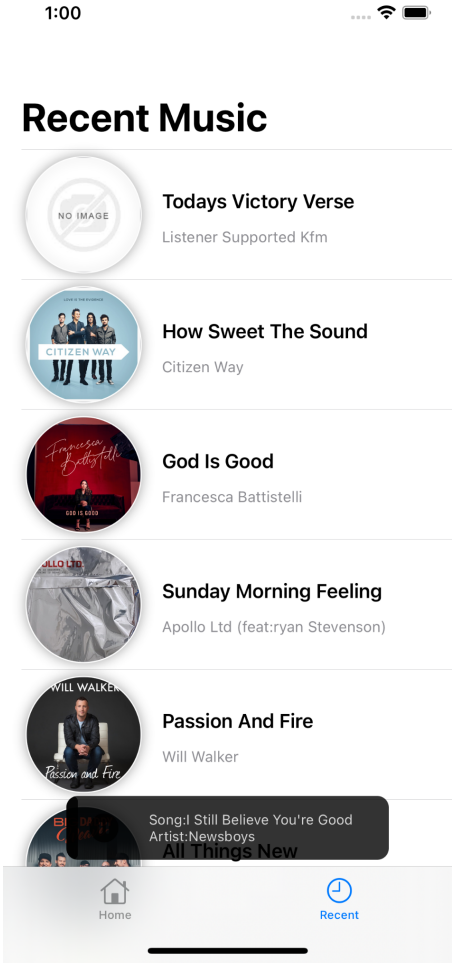
The view is made from SwiftUI code. It uses standard swift ui components like View, Text, VStack, List etc. this view shows the data from the model view.

View Model

The final piece is the view model. The view model contains everything needed to drive the view. This includes properties displayed in the view and the commands that are executed by the user through the UI. This application has two view so the view model is named `PlayingMusicModel and RecentMusicModel`

App Screenshots





Technology and Methodology which used

1. MVVM - MVVM stands for Model, View, View Model. It is a structural design pattern that allows for the separation of code into 3 groups
2. SwiftUI - building user interface
3. CocoaPods - CocoaPods is a dependency manager

Libraries used

1. SwiftUI
2. AVKit
3. UIKit
4. GoogleMobileAds

TODO:

1. Music background play
2. Show ads before stream music .