COS80020 - Creating Data Driven Mobile Applications

HD2 – Jai Lafferty 101610772

The repo containing my Distinction App can be found at:
https://github.com/CreDataDrivMobApp/task-d-JaiLaff

Design Communication Perspective

Introduction

There were a few key things I wanted to achieve out of the Album Master application. I put a focus on setting clear roles for each of my classes and screens. Due to the memory and storage space of a mobile device, it was imperative that the app was made with some sense of direction in the design stage. It was hence key to design an application whose classes, navigation flow and data was thought out and planned in advance. I also needed to be wary of the APIs used and their limitations in respect to how I planned on using them within the app.

Class Structure

As instructed during this semester, I made an effort to make the class roles compliant with the Model-View-Controller design pattern. As the View portion of this pattern resides in the Storyboard file provided, the Swift classes were to be used for the Model and the Controller (mostly the controller). For this section of the report I will be primarily be focusing on the controller classes as there is a section dedicated to the Model further into the report. Figure 1 illustrates the classes and their very general roles within the context of the app. There were three key topics to cover with the classes, the View Controllers used for controlling the actual screen of the app, the Model used for temporarily storing data, and the Data Controllers used for interfacing with either online APIs or Core Data within the app. The choice to have dedicated classes for controlling data was due to the desire to be able to access the same methods from multiple screens within the app. For instance, both the tableView and the album details screen require use of the iTunes Search API and would be logical for both screens to use methods contained within the same object (dataParser). This allows for changes in only a single class should the usage of an API or the Core Data model change.

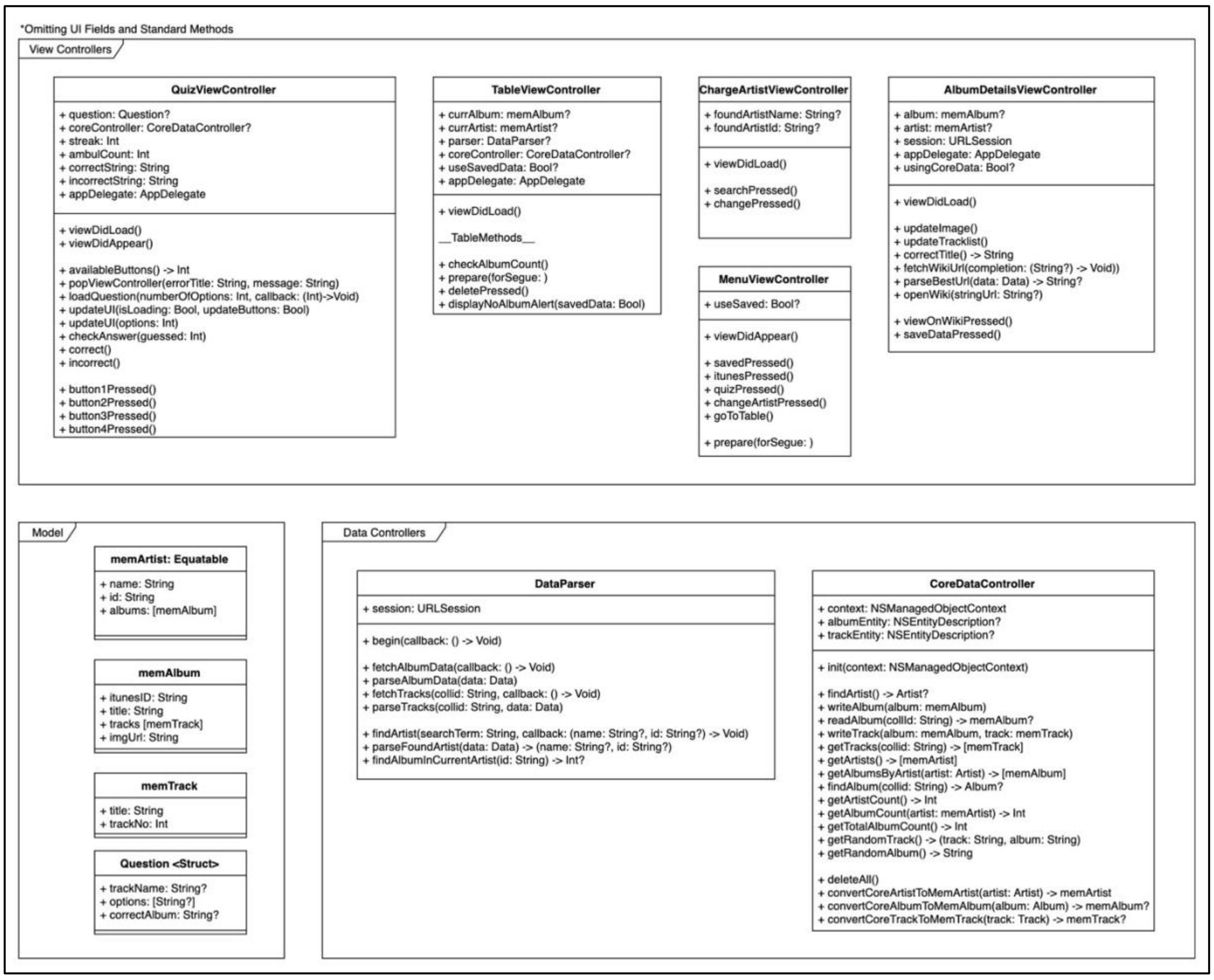

*Omitting UI Fields and Standard Methods
View Controllers
QuizViewController
+ question: Question?
+ coreController: CoreDataController?
+ streak: Int
+ ambulCount: Int
+ correctString: String
+ incorrectString: String
+ appDelegate: AppDelegate
+ viewDidLoad()
+ viewDidAppear()
+ availableButtons() -> Int
+ popViewController(errorTitle: String, message: String)
+ loadQuestion(numberOfOptions: Int, callback: (Int)->Void)
+ updateUI(isLoading: Bool, updateButtons: Bool)
+ updateUI(options: Int)
+ checkAnswer(guessed: Int)
+ correct()
+ incorrect()
+ button1Pressed()
+ button2Pressed()
+ button3Pressed()
+ button4Pressed()
TableViewController
+ currAlbum: memAlbum?
+ currArtist: memArtist?
+ parser: DataParser?
+ coreController: CoreDataController?
+ useSavedData: Bool?
+ appDelegate: AppDelegate
+ viewDidLoad()
__TableMethods__
+ checkAlbumCount()
+ prepare(forSegue:)
+ deletePressed()
+ displayNoAlbumAlert(savedData: Bool)
ChargeArtistViewController
+ foundArtistName: String?
+ foundArtistId: String?
+ viewDidLoad()
+ searchPressed()
+ changePressed()
MenuViewController
+ useSaved: Bool?
+ viewDidAppear()
+ savedPressed()
+ itunesPressed()
+ quizPressed()
+ changeArtistPressed()
+ goToTable()
+ prepare(forSegue:)
AlbumDetailsViewController
+ album: memAlbum?
+ artist: memArtist?
+ session: URLSession
+ appDelegate: AppDelegate
+ usingCoreData: Bool?
+ viewDidLoad()
+ updateImage()
+ updateTracklist()
+ correctTitle() -> String
+ fetchWikiUrl(completion: (String?) -> Void))
+ parseBestUrl(data: Data) -> String?
+ openWiki(stringUrl: String?)
+ viewOnWikiPressed()
+ saveDataPressed()
Model
memArtist: Equatable
+ name: String
+ id: String
+ albums: [memAlbum]
memAlbum
+ itunesID: String
+ title: String
+ tracks [memTrack]
+ imgUrl: String
memTrack
+ title: String
+ trackNo: Int
Question <Struct>
+ trackName: String?
+ options: [String?]
+ correctAlbum: String?
Data Controllers
DataParser
+ session: URLSession
+ begin(callback: () -> Void)
+ fetchAlbumData(callback: () -> Void)
+ parseAlbumData(data: Data)
+ fetchTracks(collid: String, callback: () -> Void)
+ parseTracks(collid: String, data: Data)
+ findArtist(searchTerm: String, callback: (name: String?, id: String?) -> Void)
+ parseFoundArtist(data: Data) -> (name: String?, id: String?)
+ findAlbumInCurrentArtist(id: String) -> Int?
CoreDataController
+ context: NSManagedObjectContext
+ albumEntity: NSEntityDescription?
+ trackEntity: NSEntityDescription?
+ init(context: NSManagedObjectContext)
+ findArtist() -> Artist?
+ writeAlbum(album: memAlbum)
+ readAlbum(collId: String) -> memAlbum?
+ writeTrack(album: memAlbum, track: memTrack)
+ getTracks(collid: String) -> [memTrack]
+ getArtists() -> [memArtist]
+ getAlbumsByArtist(artist: Artist) -> [memAlbum]
+ findAlbum(collid: String) -> Album?
+ getArtistCount() -> Int
+ getAlbumCount(artist: memArtist) -> Int
+ getTotalAlbumCount() -> Int
+ getRandomTrack() -> (track: String, album: String)
+ getRandomAlbum() -> String
+ deleteAll()
+ convertCoreArtistToMemArtist(artist: Artist) -> memArtist
+ convertCoreAlbumToMemAlbum(album: Album) -> memAlbum?
+ convertCoreTrackToMemTrack(track: Track) -> memTrack?

Fig. 1: Album Master Class Diagram

Navigation Structure

The navigation structure is extremely simple and relatively familiar for most users. It features a single menu displaying to the user almost all accessible screens. The use of a Navigation Controller and segues within the application made this very easy to implement. With the core idea behind the app being very simple, I aimed to mirror this simplicity within the navigation structure.

I identified that I was going to need almost the exact same tableView screen for both the iTunes Albums and the Saved Albums. This motivated the reuse of the tableView. Figure 2 depicts this.

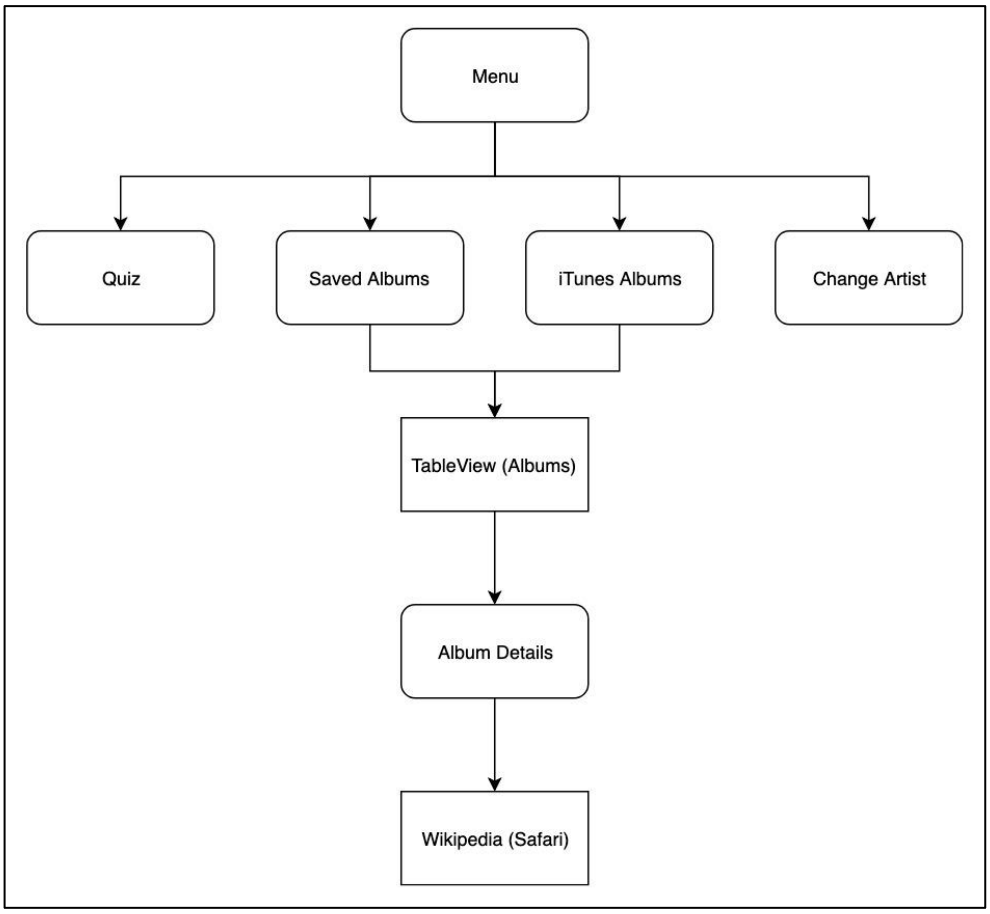


Fig. 2: Album Master Navigation Structure

Data Model

The data model used was also quite simple. I needed to locally store three entities each with a relationship to each other within Core Data. Figure 3 illustrates the relationships between Artist, Album and track entities. I also wanted to keep the attributes relatively simple, there was no need storing the iTunes Store price of an album if you were being quizzed on the track list. It is also important to note that should an additional attribute need to be added to the Core Data, only the data controller objects mentioned previously needs altering.

In addition to the Core Data entities, temporary data is stored in memory. Classes memArtist, memAlbum and memTrack were created to store the data for use with either the tableView or insertion into Core Data. I chose classes over structs as I wanted reference type objects rather than value type. This was as I planned on storing a memArtist as the current artist whose albums are to be saved/deleted etc. As the current artist was also being stored in a set of memArtists, I wanted changes made to the current artist to also be made in the set. Figure 1 highlights the structure of the classes.

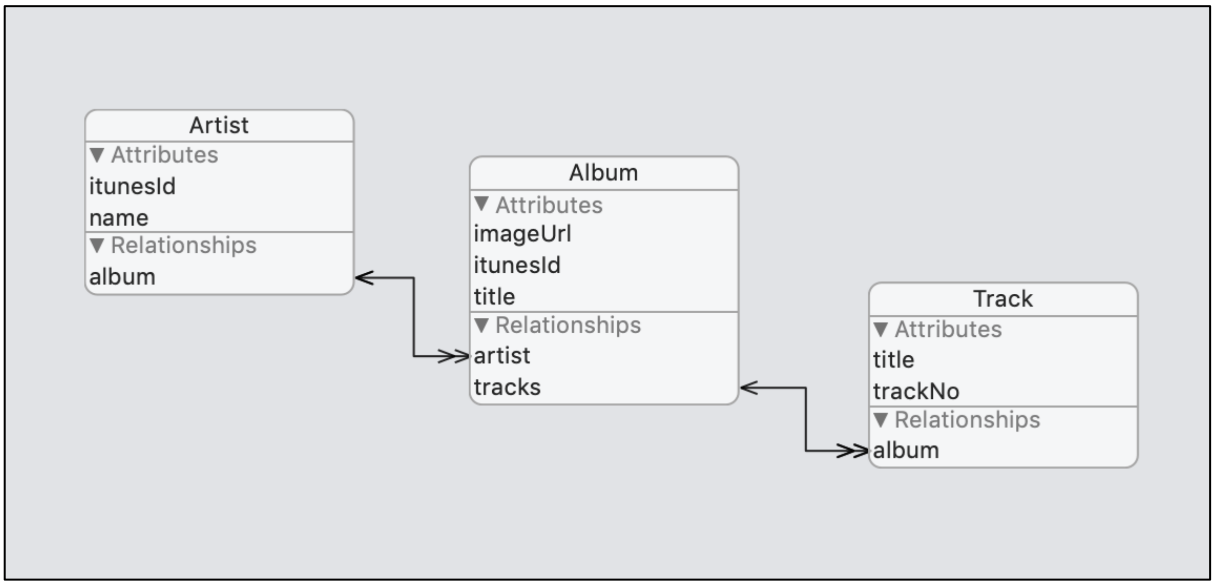


Fig. 3: Album Master Core Data Model

API Limitations & Solutions

I used both the iTunes Search API and the MediaWiki API to retrieve data for my app. The basic actions were to retrieve artist, album and track data from the iTunes Search API while searching and selecting the best Wikipedia page for a given album's data using the MediaWiki API.

Primarily, the iTunes Search API is a public API; Anyone is free to send queries to Apple. To combat overloading their servers with queries, users are rate limited to a certain number of API calls over a period of time, in the case of the iTunes Search API the limit is 20 calls per minute[1]. The MediaWiki API however, being an open source project, provides a set of etiquette guidelines where they state that "there is no hard and fast limit on read requests, but we ask that you be considerate…"[2] As I only called this API from a single button I had no need to consider frequent calling of MediaWiki API. In order to comply with the iTunes API's rate limit and not impact upon the users' experience, attributes such as the track list and albums by the current artist are stored in memory for retrieval should the user wish to revisit that data. This prevented repeated calls to the API that could possibly be blocked by the rate limit.

Additionally, limitations arise should the users not be in a location with network connectivity as they would not be able to explore new albums. This was the main motivation behind the save mechanism. Rather than just flagging favourite albums but still requiring users to be connected to the internet, the saved album data is stored locally. This then allowed the quiz to be taken offline, improving usability for the users and allowing the application to retain some functionality. Further,

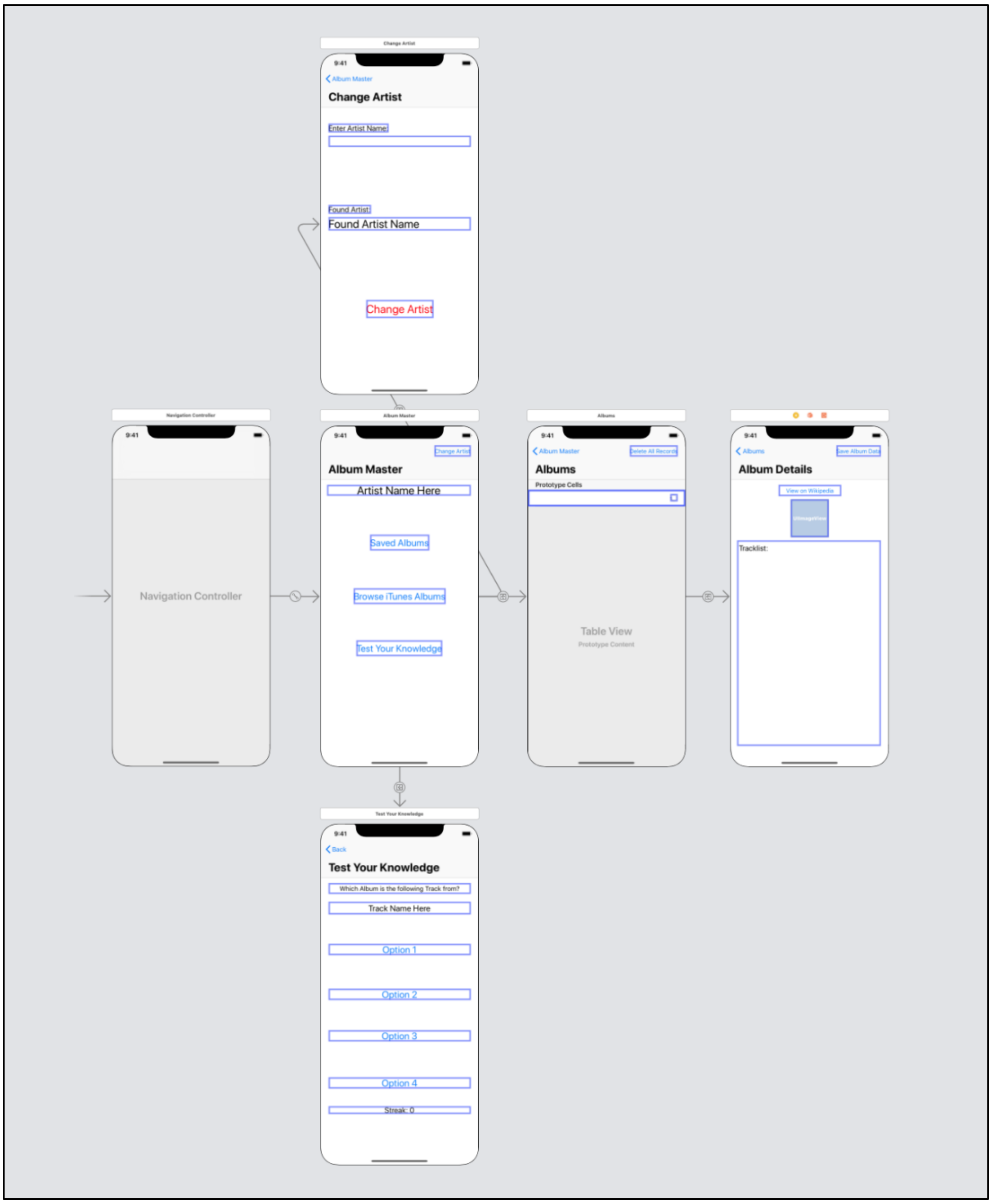
[1] Apple 2019
[2] MediaWiki 2019

as discussed in the Distinction Report, for users with limited network connectivity, a combination of completion handlers and activity indicators are used to notify the users of slow operations.

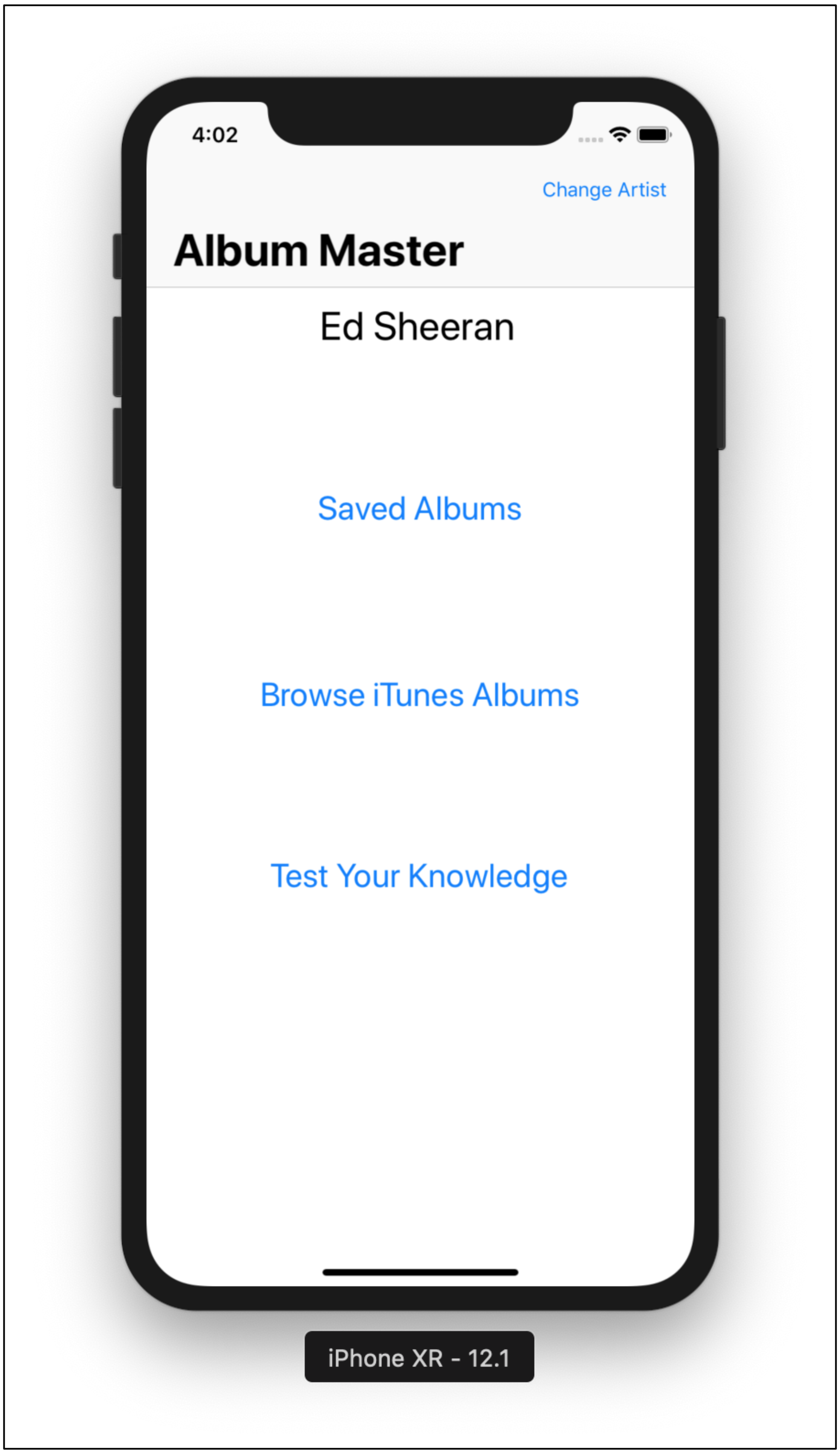
Conclusion

A lot of thought was put into the design of the application. The presented areas required consideration of the user's interests, the iOS conventions and the API limitations and etiquette. While the UI of the application may seem relatively uninspired, the work on the Model and the View is robust enough that the UI could take many forms whilst retaining solid functionality.

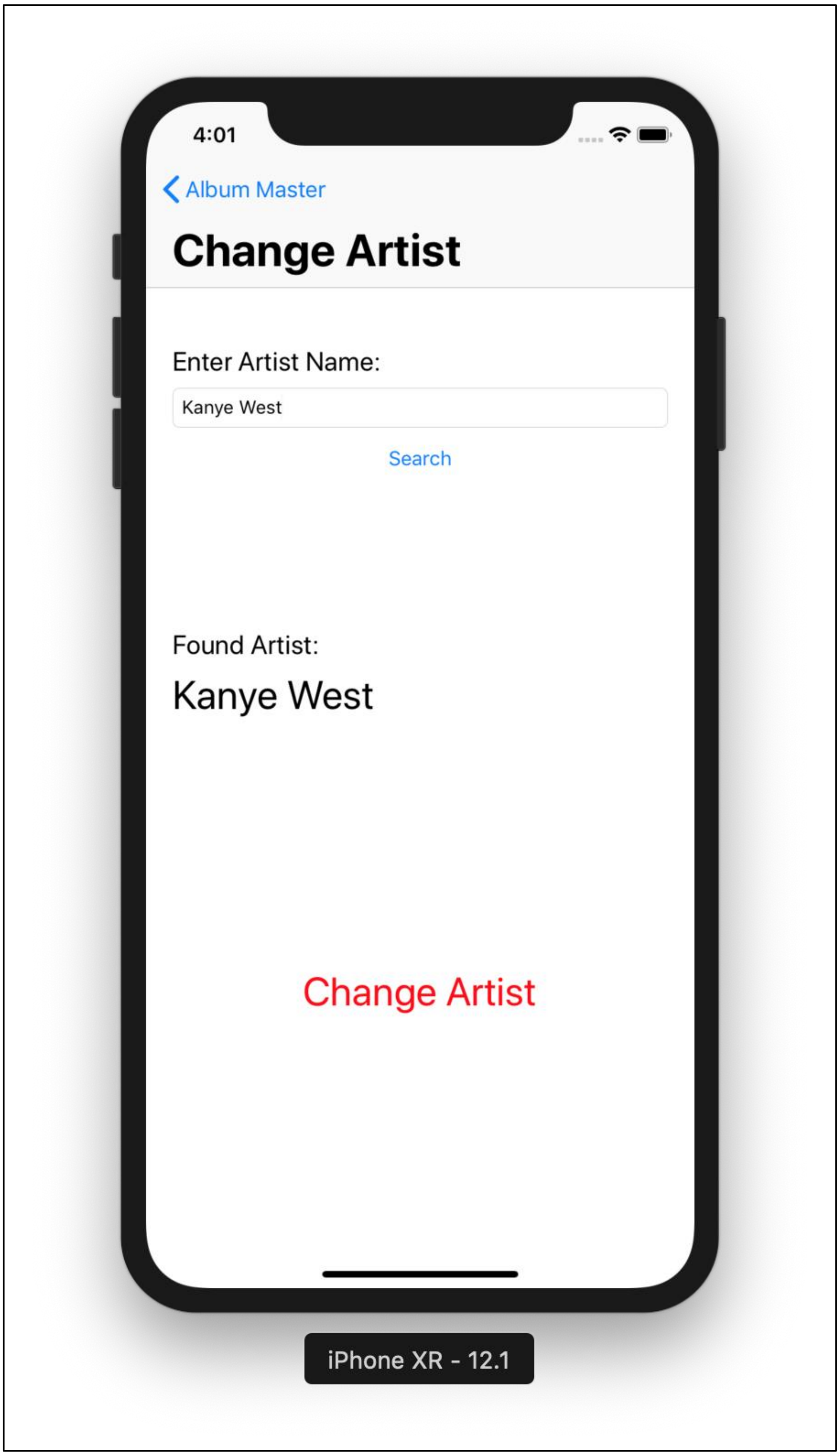
Appendix



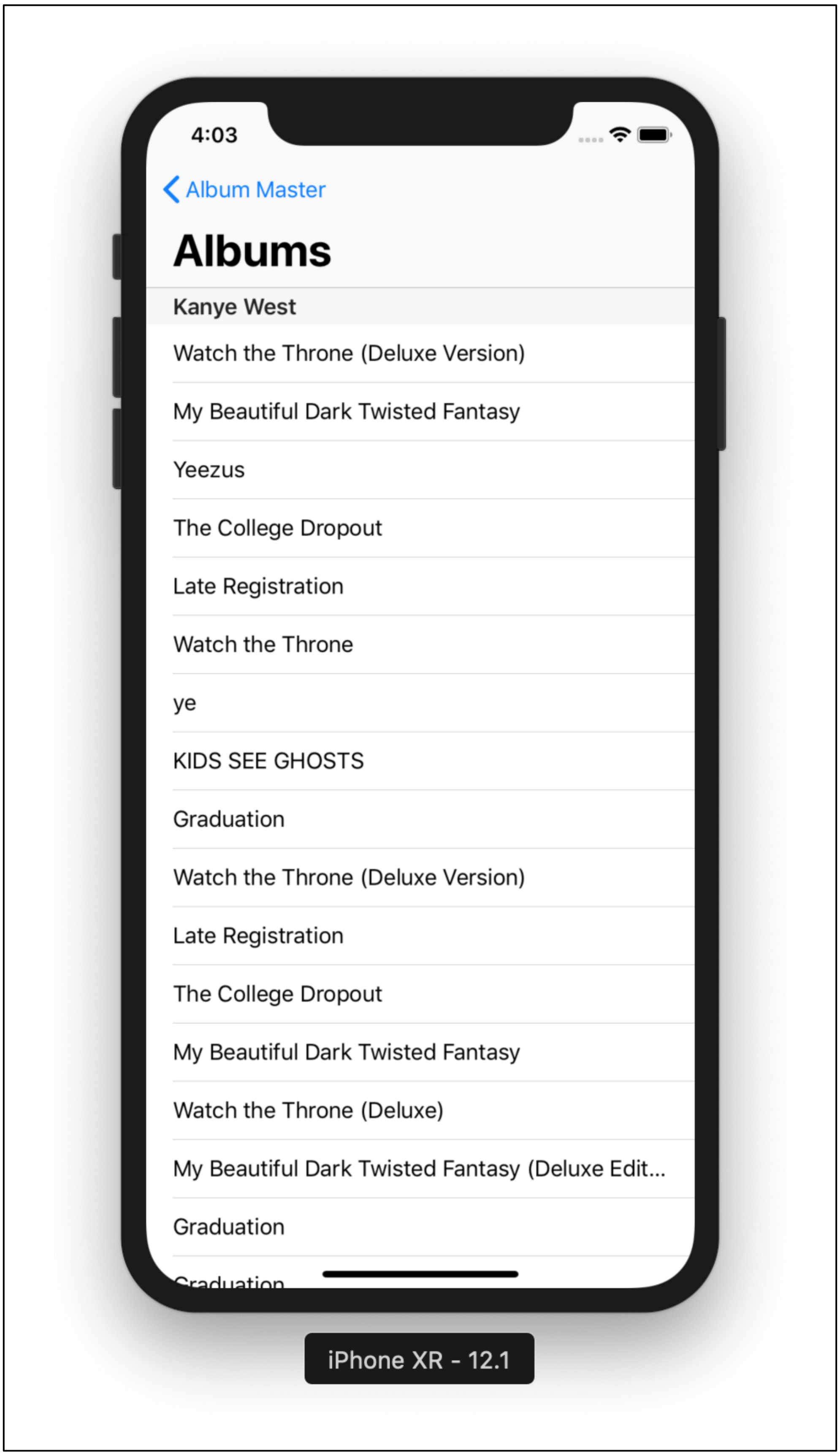
Album Master Storyboard



Album Master Menu



Album Master Change Artist



Albums

Kanye West

Watch the Throne (Deluxe Version)

My Beautiful Dark Twisted Fantasy

Yeezus

The College Dropout

Late Registration

Watch the Throne

ye

KIDS SEE GHOSTS

Graduation

Watch the Throne (Deluxe Version)

Late Registration

The College Dropout

My Beautiful Dark Twisted Fantasy

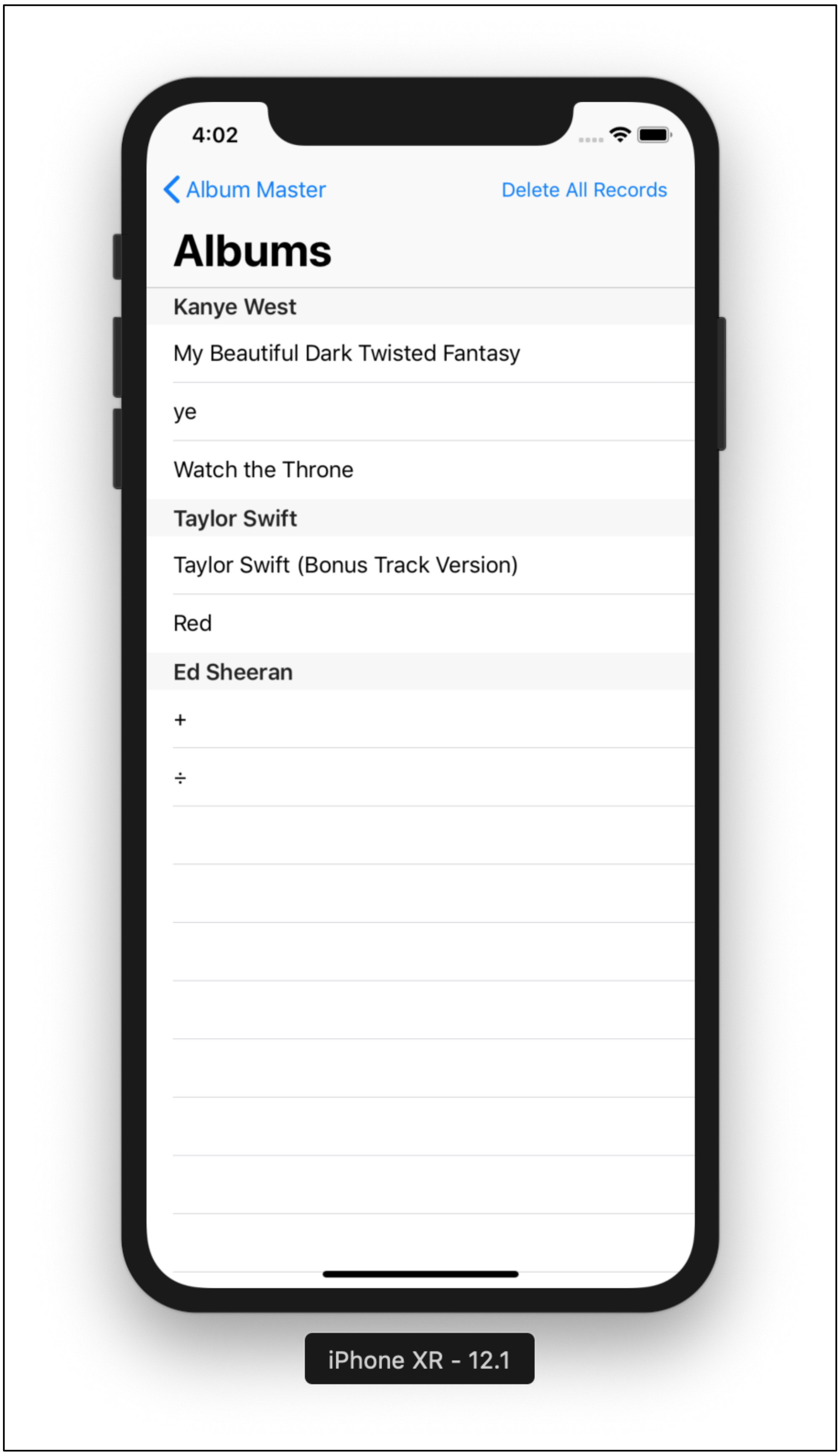
Watch the Throne (Deluxe)

My Beautiful Dark Twisted Fantasy (Deluxe Edit...

Graduation

Graduation

Album Master iTunes Albums



Albums

Kanye West

My Beautiful Dark Twisted Fantasy

ye

Watch the Throne

Taylor Swift

Taylor Swift (Bonus Track Version)

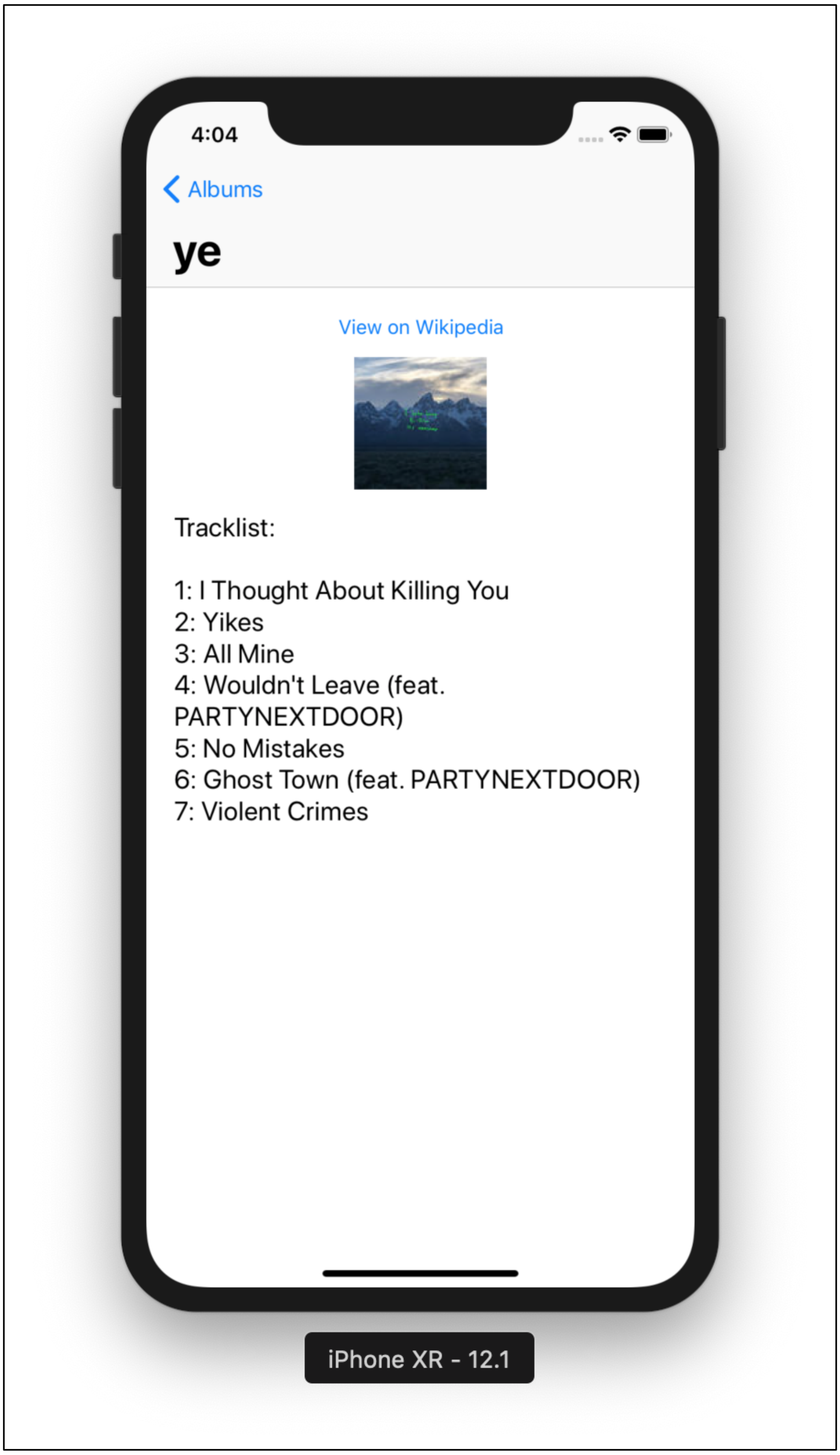
Red

Ed Sheeran

+

÷

Album Master Saved Albums



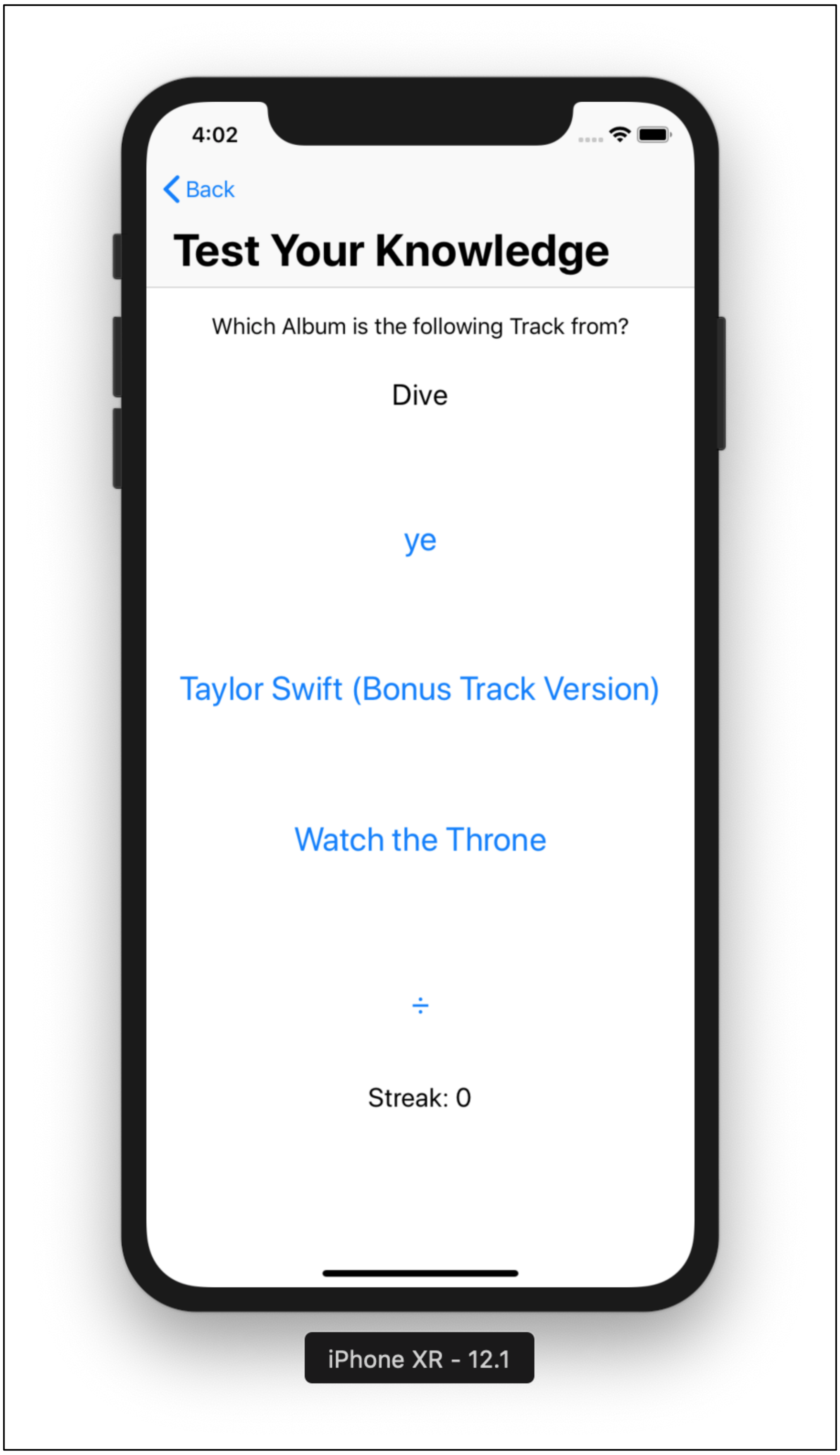
ye

View on Wikipedia

Tracklist:

1: I Thought About Killing You
2: Yikes
3: All Mine
4: Wouldn't Leave (feat. PARTYNEXTDOOR)
5: No Mistakes
6: Ghost Town (feat. PARTYNEXTDOOR)
7: Violent Crimes

Album Master Album Details


4:02

Back

Test Your Knowledge

Which Album is the following Track from?

Dive

ye

Taylor Swift (Bonus Track Version)

Watch the Throne

÷

Streak: 0

iPhone XR - 12.1

Album Master Quiz

REFERENCES

Apple 2019, *Best Way to Search for Content,* iTunes Affiliate Resources, Viewed 7 June 2019, <https://affiliate.itunes.apple.com/resources/blog/best-way-to-search-for-content/>

MediaWiki 2019, *API:Etiquette,* MediaWiki API Documentation, Viewed 7 2019, <https://www.mediawiki.org/wiki/API:Etiquette>