Mobile Mentor

iOS Code Challenge

1. Introduction

This code challenge has been designed to evaluate your overall skillset as an iOS Developer. It will test your ability to debug, refactor, and implement new features and enhancements. The basic functionality is described below, along with the user stories and requirements for each new feature/enhancement.

The app is built entirely with programmatic views so the use of Storyboards or NIBs/XIBs is not permitted. The app uses UIViewControllers exclusively, containing multiple UITableViews but no UITableViewControllers, so if you need to include a UITableView as part of a new feature or enhancement, you will need to implement the UITableView as a subview of a UIViewController. This app utilizes the MVVM design pattern so will need to ensure that you adhere to that pattern.

The Xcode project has a Theme folder that contains definitions for the visual design of the app. If a new feature or enhancement requires the creation of a new screen or UI element, ensure that the design remains consistent.

There are two default users that are created when the app launches. The login details for the default users are stored in the UserAccountModel.

If you have any questions or need clarification on any of the instructions below, feel free to contact the Mobile App Development Manager at jeremy.barger@mobile-mentor.com.

2. Basic Overview

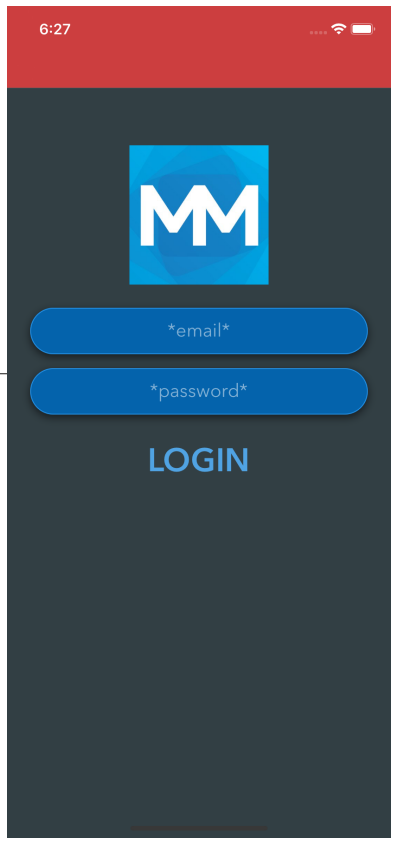
This app shoul allow users to perform basic searches on iTunes for an artist or song via the API provided by Apple. After the user enters a search term - an artist name or song title - the app makes a network call to the iTunes API, converts the data that is returned into a JSON object, parse that JSON object, store certain values from that JSON object into the SearchResultsModel, and presents the data stored in the SearchResultsModel to the user in a UITableView.

The four values from the JSON object that will need to be displayed are: “artworkUrl100”, “trackName”, “collectionName”, and “artistName”.

3. Current Functionality

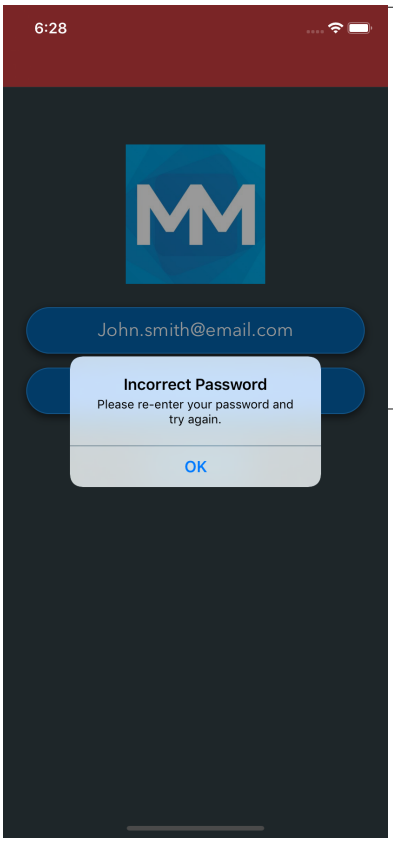
3.1. Login Screen

The LoginController is a UIViewController that is embedded in a UINavigationController. This UIViewController should contain a UIImageView, two UITextFields, and a UIButton. The UITextField for the password should be secure.



Authentication

When the LOGIN button is tapped, the app will need to successfully authenticate the user before allowing them access to the app. This authentication will need to compare the text that is entered in the UITextFields against the user data that is stored in the UserAccountModel.



Incorrect Password

If the user enters an incorrect password, the user should receive an error message via a UIAlertController.

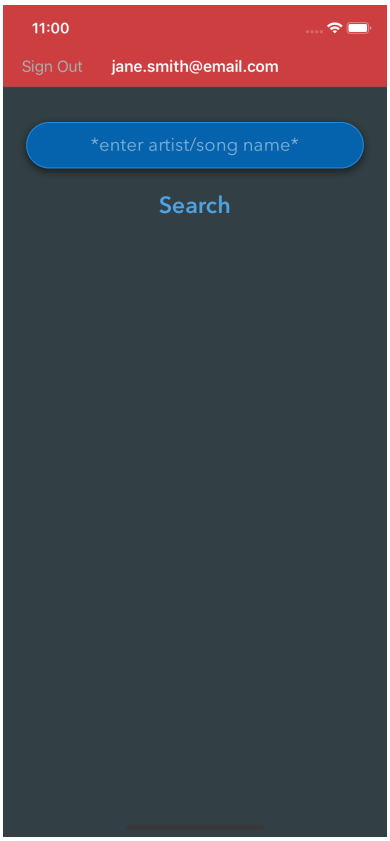
When the user taps the “OK” button, the text in the UITextField for the password should be cleared and this UITextField should become active. The text in the UITextField for the email address should remain unchanged.

Account Not Found

If the user attempts to login with an email address that has not been registered, the user should receive an error message via a UIAlertController that provides them with only the option to “Cancel”.

If the user taps “Cancel” then the UIAlertController should be dismissed, the UITextFields should be cleared, and the UITextField for the email address should become active.

3.2. Search Screen

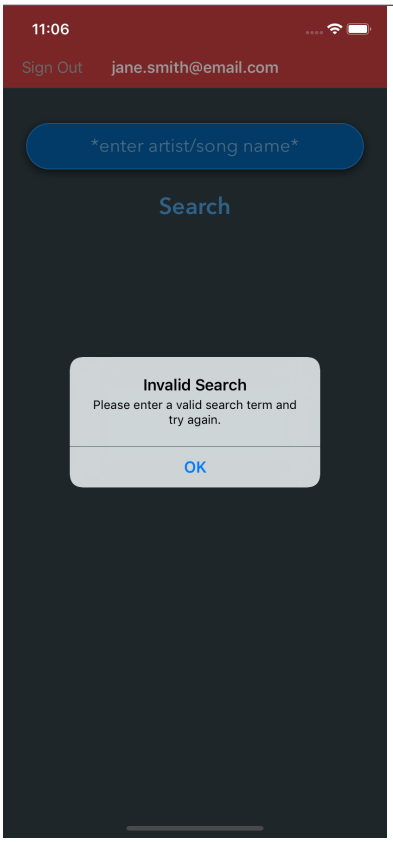


After a successful login, the user should be presented with the SearchController. The SearchController is a UIViewController that is embedded in a UINavigationController. The SearchController should contain a UITextField, a UIButton, and a UITableView.

The title of the navigationController for the SearchController should populate with the name of the user that is currently signed in.

The UITableVIew should not be visible unless there is data stored in the SearchHistoryModel.

The app should validate that the user has entered a valid search term when the user taps the “Search” button. If the search term is valid, the search term should be captured in the SearchHistoryViewModel and the navigationController should push the ResultsController.



Invalid Search Term

If the UITextField for the search term is empty, the user should be presented with an error message via a UIAlertController. When the user taps “OK”, the UITextField for the search term should become active.

Search History

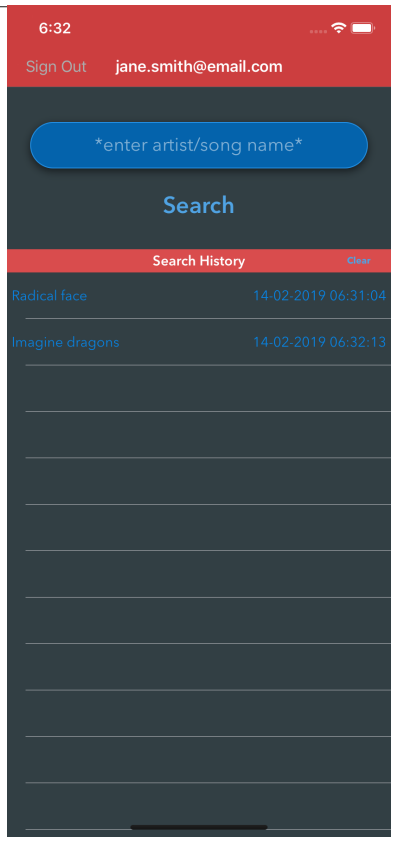
Once the user has performed a valid search, the UITableView should appear that shows the search history for the logged in user. The UITableView should contain a HeaderView that returns the SearchHistoryHeaderView and a UIButton.

Each row of the UITableView will contain the data that was captured for an individual search and will be displayed in ascending order.

The UITableViewCell should contain two UILabels.

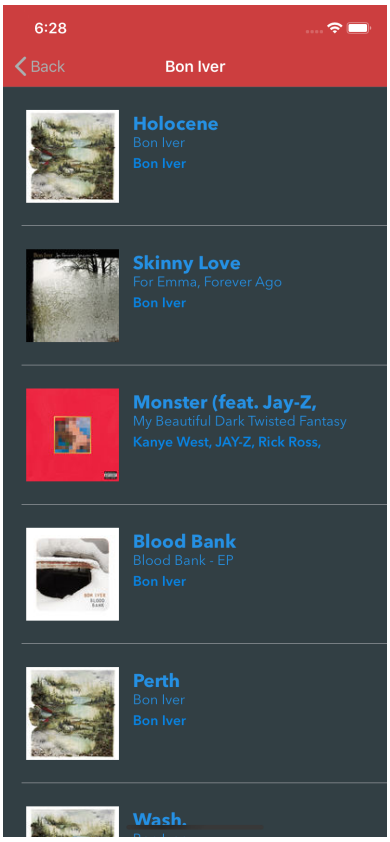
One UILabel will display the search term that was captured; the other UILabel will display the date and time that the search was performed.

When the “Clear” button is tapped, the data should be cleared from the SearchHistoryModel and the UITableView should no longer be visible.



Cell Selection

When a user selects a cell, the UITextField for the search term should be populated with the historical search term displayed in the cell and a new search should be executed, meaning the app should initiate a new search action by making an API call to iTunes via the Network Manager.



3.4. ResultsController

The ResultsController is a UIViewController that contains a UITableView that uses the ResultsTableViewCell for each row.

The ResultsTableViewCell should contain a UIImageView and three UILabels. The UIImageView will display the album image artwork and the UILabels will display the track name, the album name, and the artist name.

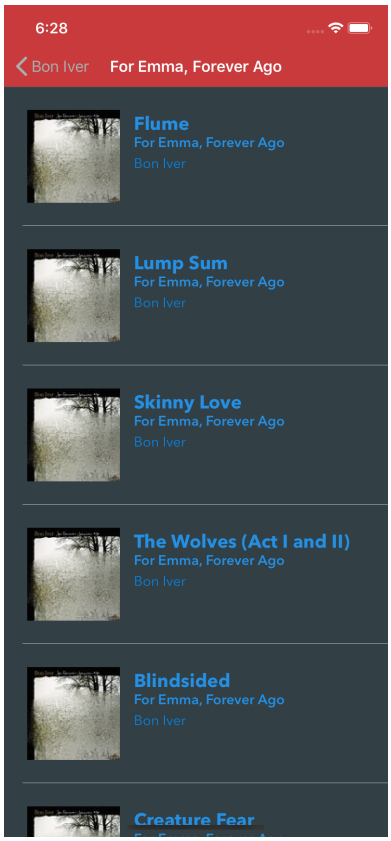
The title for the navigationController should display the term that was searched for by the user.

The selectionStyle for the UITableViewCell should be set to none.

Cell Selection

When the user selects a cell, the app will make another API call to iTunes, via the Network Manager, to query the name of the album. The data returned from iTunes will contain all of the information for the individual tracks that are included in the album, which will need to be converted to a JSON object and parsed. The values from the parsed JSON object should be stored in the SearchResultsViewModel. The values stored in the SearchResultsViewModel will be presented to the user in a UITableView that is a subview of the AlbumController, which will be pushed by the navigationController when the Network Manager has completed the API call.

3.5. AlbumController



The AlbumController is a UIViewController that should contain a UITableViewController that uses the ResultsTableViewCell for each row. The tracks should be displayed in chronological order, so the first row will contain the information for the first track of the album and the last row will contain the information for the last track of the album.

The ResultsTableViewCell should contain a UIImageView and three UILabels. The UIImageView will display the album image artwork and the UILabels will display the track name, the album name, and the artist name.

The title for the navigationController should display the name of the album.

The selectionStyle for the UITableViewCell should be set to none and nothing should happen when the user selects a cell.

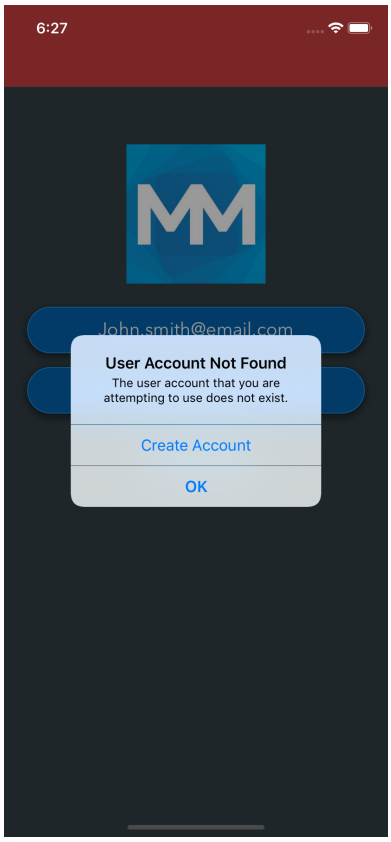
4. Debugging

The app is currently nonfunctional because of a number of bugs that have been reported after the latest release. Unfortunately, we have not yet been able to compile a list of bugs so you will need to start at the beginning, attempting to restore the functionality that is described in this documentation.

5. Enhancements

The following enhancements have been requested:

5.1. LoginController



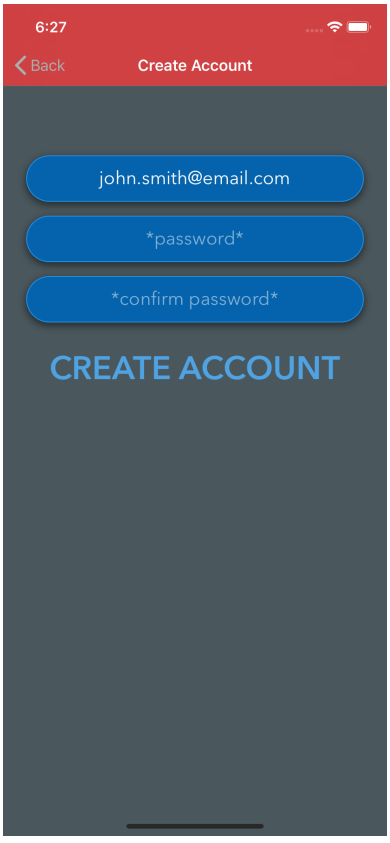
If the user attempts to login with an email address that has not been registered, currently the user receives an error message via a UIAlertController that provides them with only one option: Cancel.

This needs to be modified so that the user is presented with two options: Cancel or Create Account.

If the user taps “Cancel” then the UIAlertController should be dismissed, the UITextFields should be cleared, and the UITextField for the email address should become active.

If the user taps “Create Account” the text in the UITextField for the email address should be captured in the UserAccountViewModel and the user should be presented with the CreateAccountController.

5.2. Create Account Screen



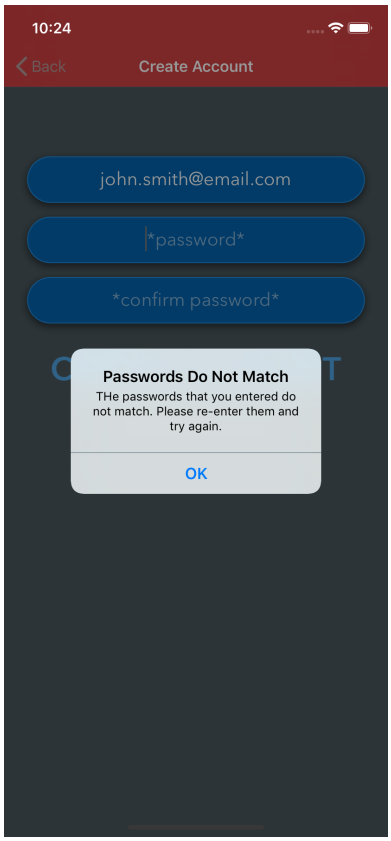
The CreateAccountController is a UIViewController that will be pushed onto the current navigationController. This UIViewController should contain three UITextFields and a UIButton.

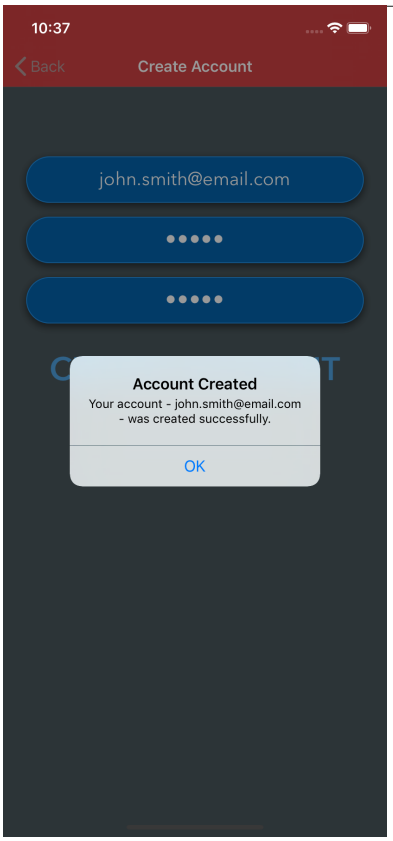
The UITextField for the email address should be automatically populated with the email address that was stored in the UserAccountViewModel when the user tapped “Create Account” from the LoginController. The user should not be able to interact with or alter the text in the UITextField for the email address.

The UITextFields for the password and password confirmation should be secure.

Passwords Do Not Match

When the user taps the “Create Account” button, the text entered into the UITextFields for the password and the password confirmation will need to be compared. If the text does not match, the user should be notified via a UIAlertController. When the user taps “OK”, the UITextFields for the password and the password confirmation should be cleared and the UITextField for the password should become active.





Account Created

When the user taps the “Create Account” button, if the password comparison passed, the user’s account should be created, meaning that the user email and password should be stored in the UserAccountModel.

The user should be presented with a message that their account was successfully created via a UIAlertController. The message should include the email address that was used to register. When the user taps the “OK” button, the the current navigationController should pop the CreateAccountController and the user should be presented with the SearchController, a UIViewController that is embedded in a UINavigationController.

5.2. AlbumController

Currently, the AlbumController displays a UITableView that contains the album image artwork in each cell. This UIViewController needs to be updated so that the album image artwork appears once - at the top of the UIViewController - with the UITableView situation below it. The final result should look similar to that of Apple Music.

The UITableViewCell needs to be updated so that only the UILabel for the track name is displayed. This UILabel needs to be updated so that it is the same width as the screen, the track name is centered in the label, and the track number is prefixed. Unfortunately, there is no screen mockup to present so you will need to do your best to ensure that the UI design is followed.