KUMU CODING CHALLENGE:

Objective:

Create an application that contains at least one dependency. This application should display a list of items obtained from an iTunes Search API and show a detailed view of each item. The URL you must obtain your data from is located on these two URL's:

https://itunes.apple.com/search?term=star&country=au&media=movie&all

(iTunes Web Service Documentation: https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/#searching)

Evaluation of Criteria for this task.

1. Readability – Class and method names should clearly show their intent and responsibility.
2. Maintainability - SOLID principles and design pattern.
3. Scalability – Your software should easily accommodate possible future requirement changes.

Requirements:

Your code must be written in Swift. Using SwiftUI is a plus

General:

1. Your list should show the following details from the API mentioned above:
    - Track Name
    - Artwork (there are multiple size formats, pick one that you consider is appropriate for the task)
    - Price
    - Genre
2. App should have a working search bar
3. App should have the ability to mark an item as their favorite, which they can view in a Favorites section.

In addition, the detail view should show "Long Description" for the given item

Each row should have a placeholder image if for some reason the URL is unable to be loaded. Images must not be duplicated when scrolling.

Persistence (Room Persistence Implementation):

Your app should demonstrate the ability to save data and reuse it when the user opens the app again. Your local storage must have the "Room Persistence" implementation.

- This functionality must be very clear in your app
- Your implementation should be explained in a README.md or in code comments

Suggestions:

- A date when the user previously visited, shown in the list header
- Save the last screen that the user was on. When the app restores, it should present the user with that screen.

You can save other data. Add any supporting information in the same manner as above.

Architecture:

The app should demonstrate at least one common design pattern of your choice (MVP, MVVM, Navigation Architecture). You can provide any additional information on your choice of pattern in README.md file or as code comments (for example, why did you choose that particular one, any benefits, and disadvantages of using it)

Documentation:
The app should use internal documentation by means of /// comments above methods, properties etc, and // MARK: - format for extensions and code sections.

UI and design:

You're free to experiment with how you style your app's UI elements. Your approach should be consistent and follow human interface guidelines for your platform.

Submission:

The completed app can be submitted by publishing it to your Github account as a public repository.

Should you have any questions, please let us know.