HOMEWORK ASSIGNMENT 4

CSCI 571 – Fall 2023

Abstract

SwiftUI, JSON, MongoDB, Node.js, Google photos API, and eBay API

Marco Papa

papa@usc.edu

1. Objectives

- Become familiar with Swift language, Xcode, and iOS App Development
- Build a good-looking iOS app.

2. Background

2.1 Xcode

Xcode is an integrated development environment (IDE) containing a suite of software development tools developed by Apple for developing software for macOS and iOS. First released in 2003, the latest stable release is version 15.0.1 and is available via the Mac App Store free of charge.

Features:

- Swift 5 support
- Playgrounds
- Interface Builder
- Device simulator and testing
- User Interface Testing
- Code Coverage

The Official homepage of the Xcode is located at https://developer.apple.com/xcode/

2.2 iOS

iOS (originally iPhone OS) is a mobile operating system created and developed by Apple Inc. and distributed exclusively for Apple hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod touch. It is the second most popular mobile operating system in the world by sales, after Android.

The Official iOS home page is located at:

http://www.apple.com/iOS/

The Official iOS Developer homepage is located at:

https://developer.apple.com/ios/

2.3 Swift

Swift is a general-purpose, multi-paradigm, compiled programming language created for iOS, macOS, watchOS, tvOS and Linux development by Apple Inc. Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of existing Objective-C code written for Apple products. Swift is intended to be more resilient to erroneous code ("safer") than Objective-C and also more concise. It is built with the LLVM compiler framework included in Xcode 6 and later and uses the Objective-C runtime, which allows C, Objective-C, C++ and Swift code to run within a single program.

The Official Swift homepage is located at:

https://developer.apple.com/swift/

2.4 SwiftUI

SwiftUI is an innovative, exceptionally simple way to build user interfaces across all Apple platforms with the power of Swift. Build user interfaces for any Apple device using just one set of tools and APIs. With declarative Swift syntax that's easy to read and natural to write, SwiftUI works seamlessly with new Xcode design tools to keep your code and design perfectly in sync. Automatic support for Dynamic Type, Dark Mode, localization, and accessibility means your first line of SwiftUI code is already the most powerful UI code you've ever written.

SwiftUI was initially released at WWDC in 2019, the newer version, SwiftUI 2, was released earlier at WWDC 2020. With SwiftUI 2, developers are now able to build complete apps with swift language only for both the UI and the logic.

The Official SwiftUI homepage is located at:

https://developer.apple.com/xcode/swiftui/

The complete guide to SF Symbols is available at:

https://www.hackingwithswift.com/articles/237/complete-guide-to-sf-symbols

2.5 SF Symbols

With over 3,300 symbols, SF Symbols is a library of iconography designed to integrate seamlessly with San Francisco, the system font for Apple platforms. Symbols come in nine weights and three scales, and automatically align with text labels. They can be exported and edited in vector graphics editing tools to create custom symbols with shared design characteristics and accessibility features. SF Symbols 3 features over 600 new symbols, enhanced color customization, a new inspector, and improved support for custom symbols.

The latest version, SF Symbols 3, was released earlier this September 2021. SF Symbols 3 features over 600 new symbols, including devices, health, gaming, and more.

These new symbols are available in apps running iOS 15, iPadOS 15, macOS Monterey Beta, tvOS 15, and watchOS 8.

All of the symbols used in this homework are available in SF Symbols 2 and above.

The Official SF Symbols homepage is located at:

https://developer.apple.com/sf-symbols/

3. Prerequisites

This homework requires the use of the following components:

3.1 Download and install the latest version of Xcode (Sonoma)

To develop iOS apps using the latest technologies described in these lessons, you need a Mac computer (with latest version of macOS installed) running the latest version of Xcode. Xcode includes all the features you need to design, develop, and debug an app. Xcode also contains the iOS SDK, which extends Xcode to include the tools, compilers, and frameworks you need specifically for iOS development.

Download the latest version of Xcode on your Mac for free from the App Store.

To download the latest version of Xcode:

- Open the App Store app on your Mac (by default it's in the Dock).
- In the search field in the top-right corner, type Xcode and press the Return key.
- The Xcode app shows up as the first search result.
- Click Get and then click Install App.
- Enter your Apple ID and password when prompted.
- Xcode is downloaded into your Applications directory.

You may use any other IDE other than Xcode, but you will be on your own if problems come up.

3.2 Add your account to Xcode

When you add your Apple ID to the Xcode Accounts preferences, Xcode displays all the teams you belong to. Xcode also shows your role on the team and details about your signing identities and provisioning profiles that you'll create later in this document. If you don't belong to the Apple Developer Program, a personal team appears.

Here is detailed documentation:

https://developer.apple.com/library/iOS/documentation/IDEs/Conceptual/AppStoreDistributionTutorial/AddingYourAccounttoXcode/AddingYourAccounttoXcode.html

IMPORTANT

This homework is developed on the latest MacOS Sonoma (14). You must develop the iOS app with MacOS Sonoma. Using MacOS versions earlier than Sonoma is not allowed. If you have problems such as missing symbols (system images) using SFSymbols, failing to compile or run SwiftUI code, your best option is to install the latest MacOS.

IMPORTANT

This homework is developed with SwiftUI. We strongly recommend you to develop the app with SwiftUI. If you use storyboards, you might find some of the functionalities harder to implement, and we do not provide support in such cases.

This homework requires the use of the following components:

- Download and install Xcode 15.0.1. Xcode 15.0.1 provides the crucial functionalities you will need to develop SwiftUI apps. You can download Xcode 15.0.1 by searching "Xcode" in your mac's app store.

- Download and install SF Symbols 4, this app provides all the necessary icons for this homework. SF Symbols 1 might not contain all required icons. If you see a symbol fail to load during development, it might be because your MacOS version is too old.

- If you are new to iOS/SwiftUI Development, Section 7 Implementation Hints are going to be your best friends!

Note that with newer versions of XCode we use the Swift Package Manager. CocoaPods is only used as a dependency manager for Swift and Objective-C Cocoa projects.

3.3 Install the required libraries:

The following libraries are those you may use:

- Alamofire : To make the HTTP requests.
- Kingfisher : For rendering images.
- PromiseKit : For handling asynchronous requests.
- SwiftyJSON : To read, parse, and handle the JSON data.

NOTE: You NEED NOT confine yourself to use ONLY the above libraries. You may use any library you feel will solve the purpose.

4. High Level Design

This homework is a mobile app version of Homework 3. In this exercise, you will develop an iOS Mobile application, which allows users to search for products using the eBay APIs, get product details, add products in wishlist, and post on Facebook. You should reuse the backend service (node.js script) you developed in HW3 and follow the same API call requirements.

The main screen of this app is like that in **Figure 1.1**. The launch screen of the app is shown in Figure 1. All the implementation details and requirements will be explained in the following sections

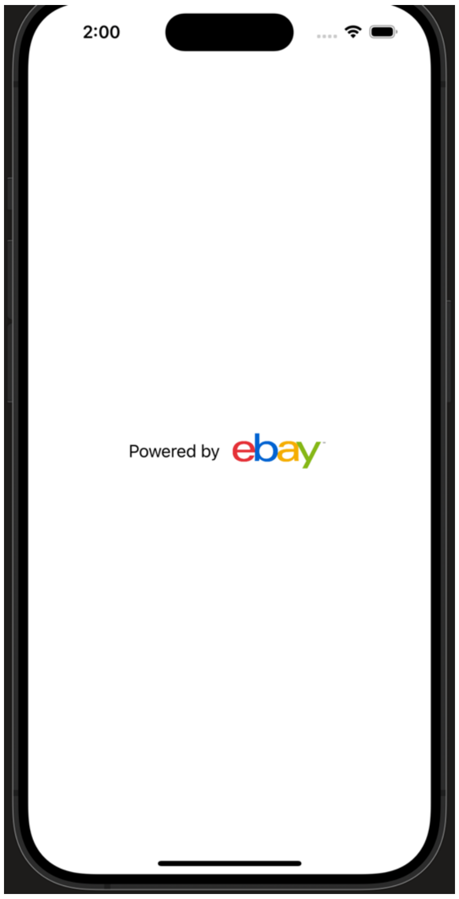


Figure 1: Launch Screen

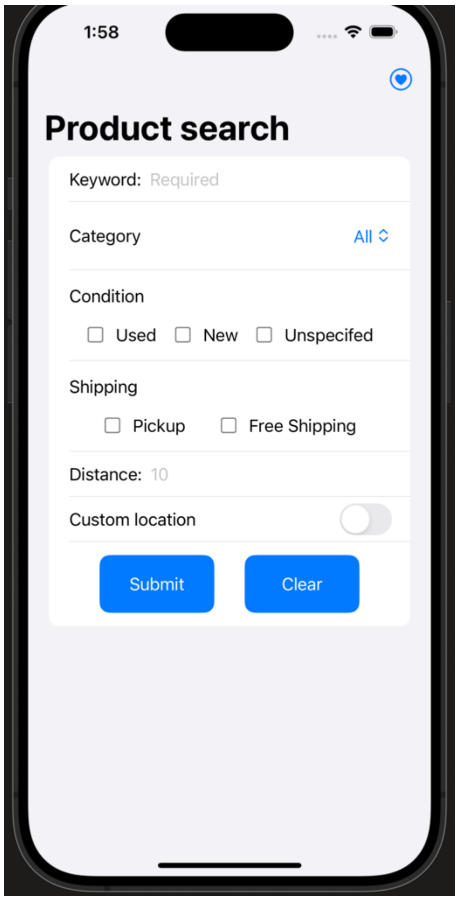


Figure 1.1: Main Screen of the app

5. Implementation

5.1 Search Form

You must replicate the Search Form, as shown in **Figure 1**. The interface consists of the following components:

- **Keyword:** A text component allowing the user to enter the keyword.
- **Category:** A text component allowing the user to choose a category. When the user taps on this field, a picker view should display at the bottom of the screen for selecting a category, as shown in **Figure 2**. Make sure you include all the categories shown below.

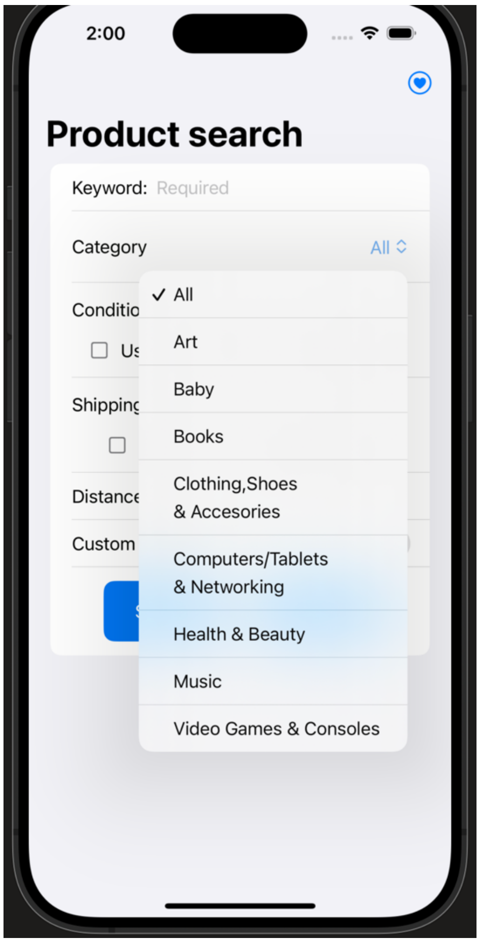
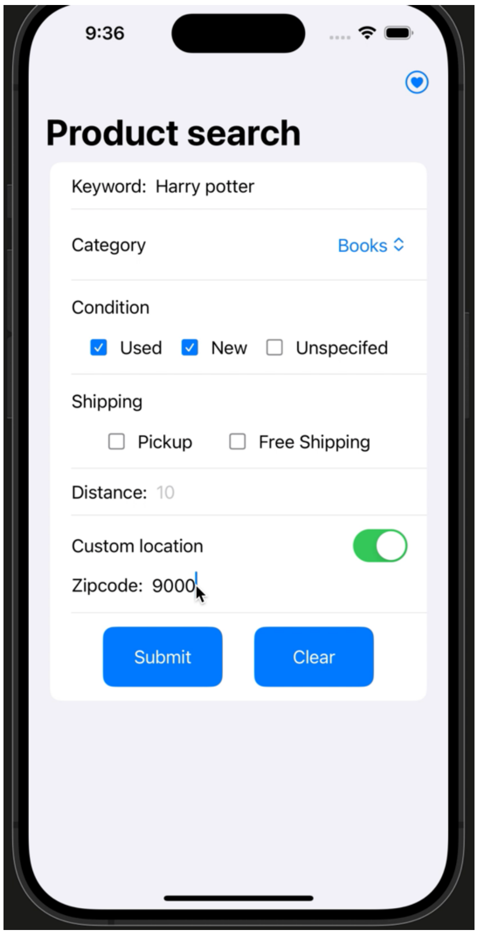


Figure 2: Choose category from a Picker View

- **Condition:** Three checkboxes, one each for **New, Used, and Unspecified**. Multiple options can be selected at once.
- **Shipping:** Two Checkboxes, one each for **Pickup, and Free Shipping**. Both options can be selected at once.
- **Distance:** A text component allowing user to enter the distance (in miles). By default, it is set to 10 miles.
- **Custom Location:** A switch that toggles between current user location (when turned OFF) and custom location (when turned ON) which is provided in the form of a zipcode.
- **Zipcode:** A text component which is shown/hidden based on the state of the switch (Custom Location). It provides the autocomplete function as shown in **Figure 3.1**. You can use the same API as Homework 3. (Hint: The results can be shown as a 'SheetView')


9:36

Product search

Keyword: Harry potter

Category: Books

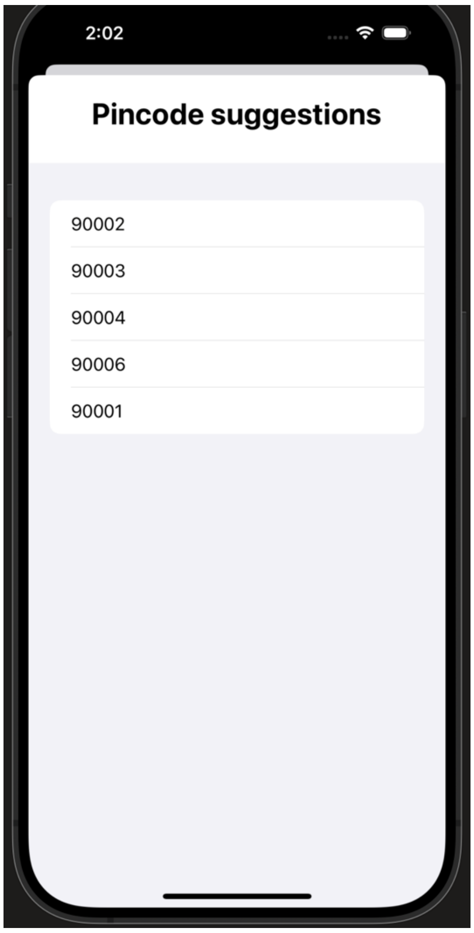
Condition: [✓ Used] [✓ New] Unspecifed

Shipping: Pickup, Free Shipping

Distance: 10

Custom location: on

Zipcode: 9000


2:02

Pincode suggestions

90002
90003
90004
90006
90001

Figure 3: Giving initial Zipcode characters **Figure 3.1: Autocomplete for Zipcode**

- A **SEARCH button** to get the input information of each field, after validation. If the validation is successful, then the products would be fetched from the server. However, if the validations are unsuccessful, appropriate messages should be displayed and no further requests would be made to the server.
- A **CLEAR button** to clear the input fields and set them to default values if applicable. It should remove all error messages (if present).

5.1.1 Current Location

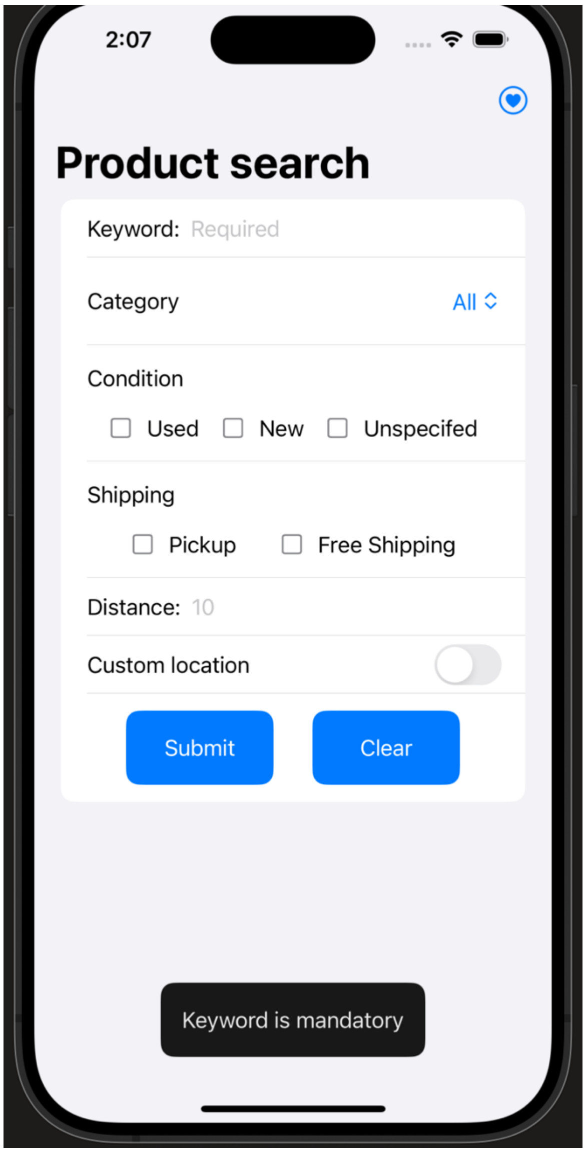
When Custom Location toggle is OFF, the current location should be used. The current location can be obtained by using ip-api.com or the standard location service provided by apple. **Both** ways are acceptable for this HW, but in general, Apple's location service should be used if the app requires high precision locations or more frequent updates.

More details on standard location service can be found here:

https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_standard_location_service

5.1.2 Validation

The validation for empty keyword needs to be implemented. If the user does not enter anything in the text field or just enters some empty spaces, when user presses the 'Submit' button you need to display an appropriate message to indicate the error, as shown below.



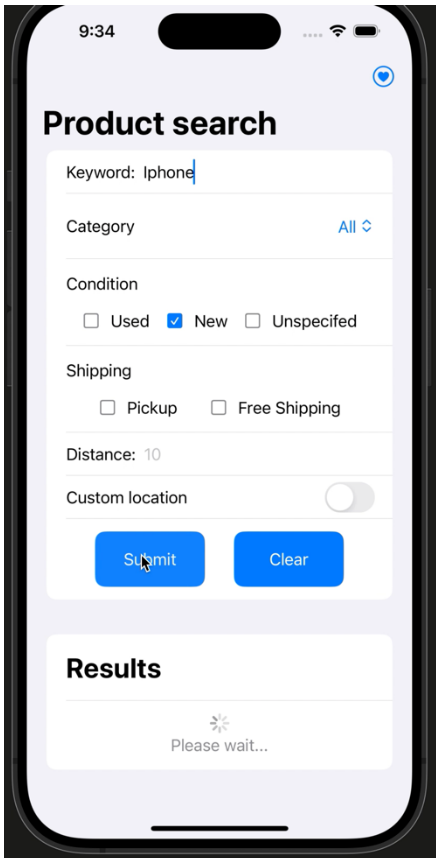
Keyword is mandatory

Figure 4: Validation for Keyword

Once the validation is successful, you should execute an HTTP request to the Node.js backend located in the GCP/AWS/Azure backend, which you should have completed in Homework 8, and then navigate to the Search Results page.

5.2 Search Results

When the user taps the ‘Submit’ button, your app should display a *loading spinner* (Figure 5) before it’s ready to show the results page. Then after it gets data from your backend, it should hide the spinner and display the result page as a table, as shown in Figure 6.



Keyword: Iphone

Category: All

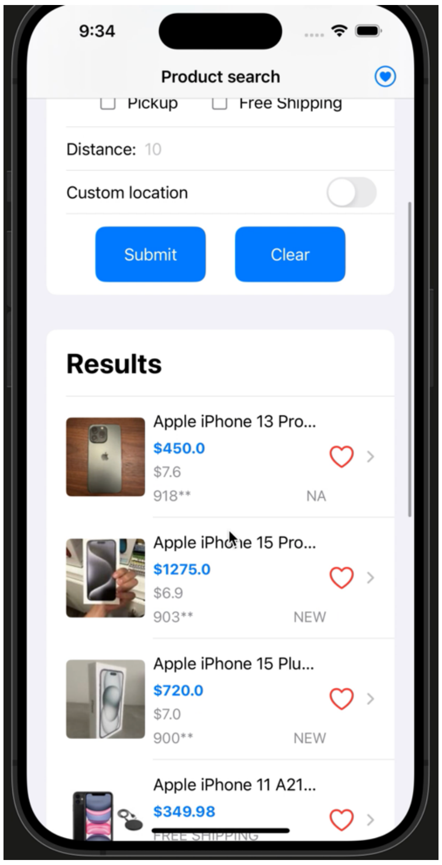
Condition: ☐ Used ☑ New ☐ Unspecifed

Shipping: ☐ Pickup ☐ Free Shipping

Distance: 10

Custom location: off

Please wait...



☐ Pickup ☐ Free Shipping

Distance: 10

Custom location: off

Apple iPhone 13 Pro... $450.0 $7.6 918** NA

Apple iPhone 15 Pro... $1275.0 $6.9 903** NEW

Apple iPhone 15 Plu... $720.0 $7.0 900** NEW

Apple iPhone 11 A21... $349.98 FREE SHIPPING

Figure 5: Display a spinner while searching

Figure 6: List of search results

Each of the **Table’s Cell** should have the following:

- Image of the product
- Title of the product
- Price of the product
- Shipping cost of the product: If shipping cost is 0, ‘FREE SHIPPING’ should be displayed instead.
- Zip code
- Condition of the product (New / Used / Refurbished /NA).
- A heart-shaped “WishList” button

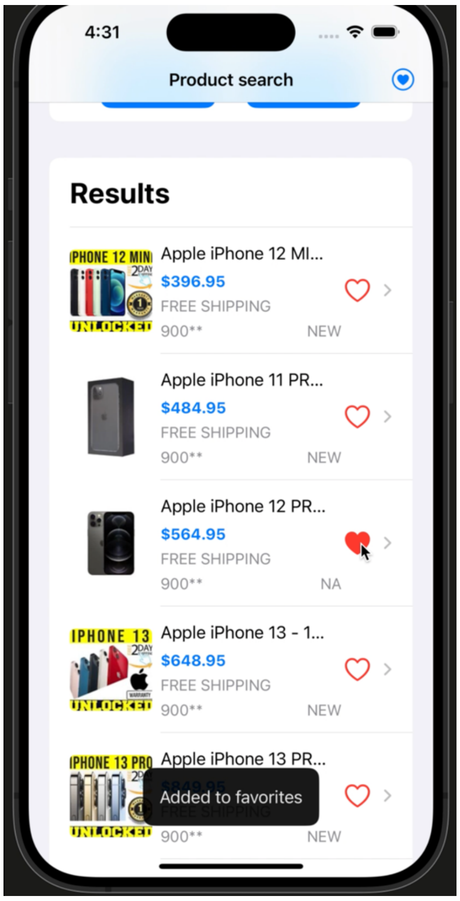
5.2.1 Condition

The condition of the product is shown according to the following mapping:

Condition ID	Condition
1. ‘1000’	NEW
2. ‘2000’ or ‘2500’	REFURBISHED
3. ‘3000’ OR ‘4000’ or ‘5000’ or ‘6000’	USED
4. Any other	NA

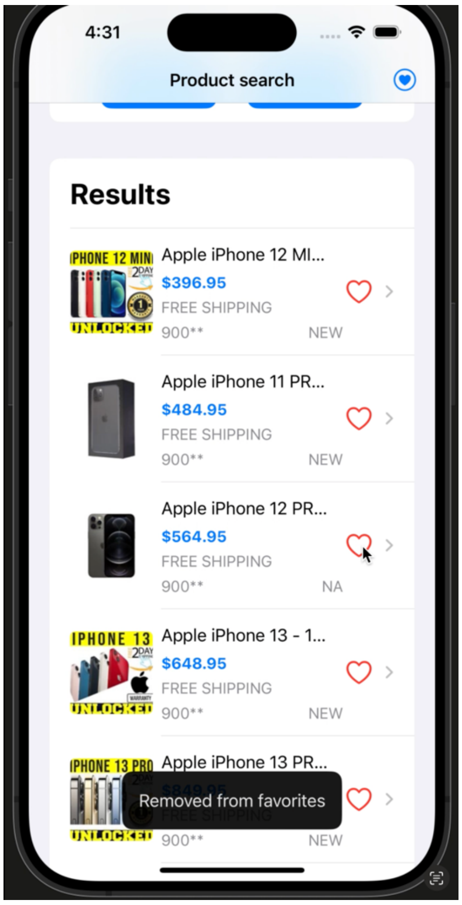
5.2.2 Wishlist button

Tapping the Wishlist button (the heart) would add the corresponding product into the Wish list and the Wishlist button should be filled with red, as shown in Figure 6. Tapping that button again would remove that product from the wish list and the button is changed back to empty button as shown in Figure 7.



Apple iPhone 12 MI... $396.95 FREE SHIPPING 900** NEW
Apple iPhone 11 PR... $484.95 FREE SHIPPING 900** NEW
Apple iPhone 12 PR... $564.95 FREE SHIPPING 900** NA
Apple iPhone 13 - 1... $648.95 FREE SHIPPING 900** NEW
Apple iPhone 13 PR... 900** NEW
Added to favorites

Figure 6: Wishlist button turning red



Apple iPhone 12 MI... $396.95 FREE SHIPPING 900** NEW
Apple iPhone 11 PR... $484.95 FREE SHIPPING 900** NEW
Apple iPhone 12 PR... $564.95 FREE SHIPPING 900** NA
Apple iPhone 13 - 1... $648.95 FREE SHIPPING 900** NEW
Apple iPhone 13 PR... 900** NEW
Removed from favorites

Figure 7: Wishlist button becoming empty on clicking again.

5.2.3 No Results

When the search query returns zero results or fails to get the results, appropriate message should be displayed as shown in figure 8.

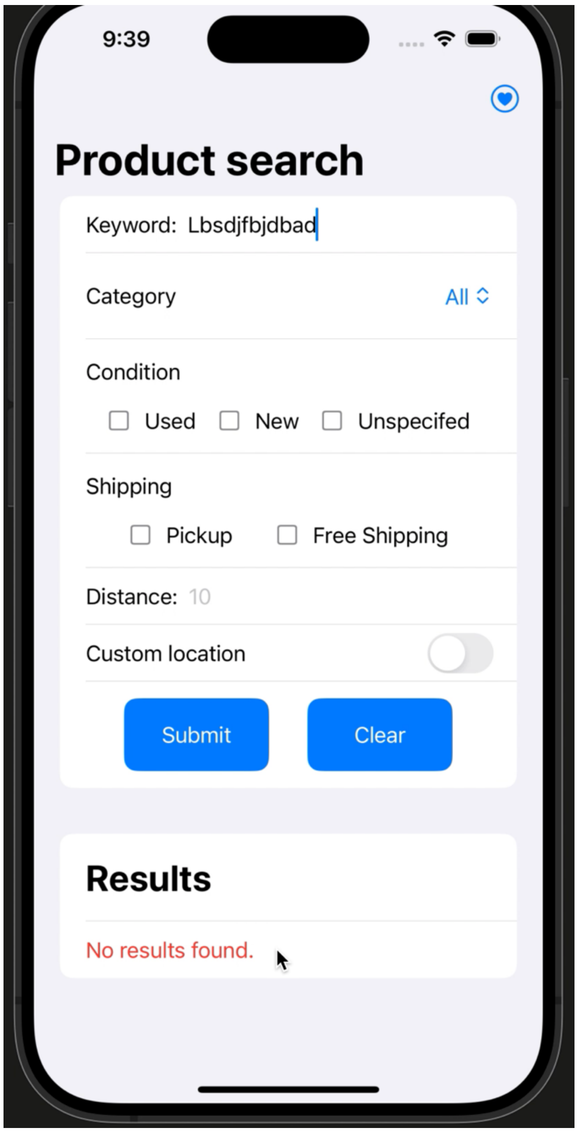


Figure 8: Message on No Results

5.3 Product Details

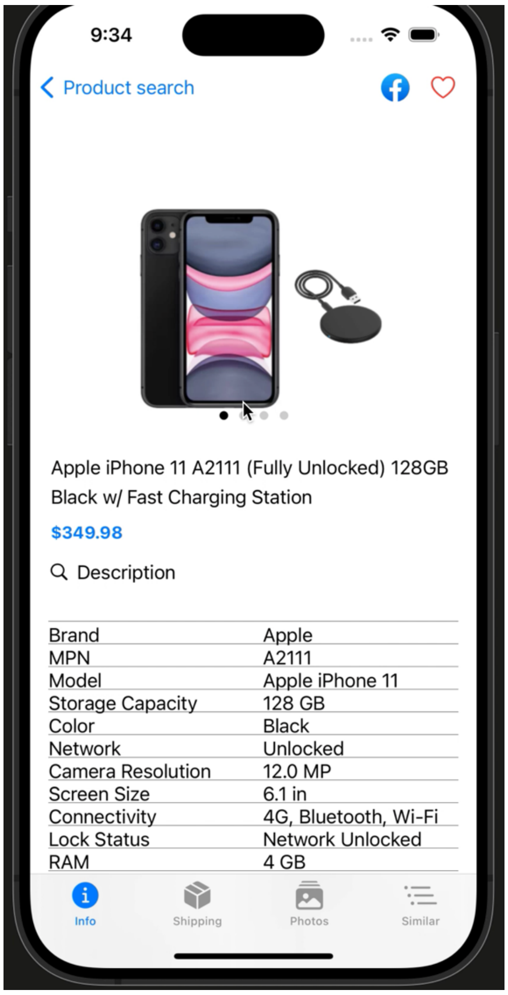
Tapping on a cell in the results table should show details of that product with four tabs: Product Info, Shipping Info, Product photos from google and Similar Products (Figure 9).

<u>Note:</u> A spinner should be shown before you are ready to display information in each tab.

<u>Note:</u> The icons representing each tab are taken from SF Symbols, and you are expected to use the same.

All the four tabs share the same **Navigation Bar on the top**. The navigation bar should include the following elements:

- **Back button** which navigates back to the search results list.
- **Share button (Facebook icon)** to share the product details on Facebook. Once the button is tapped, a web page should be opened to allow the user to share the product information on Facebook, as shown in Figure 10. **See homework reference video to know how it works and the format of the Facebook content.**


9:34

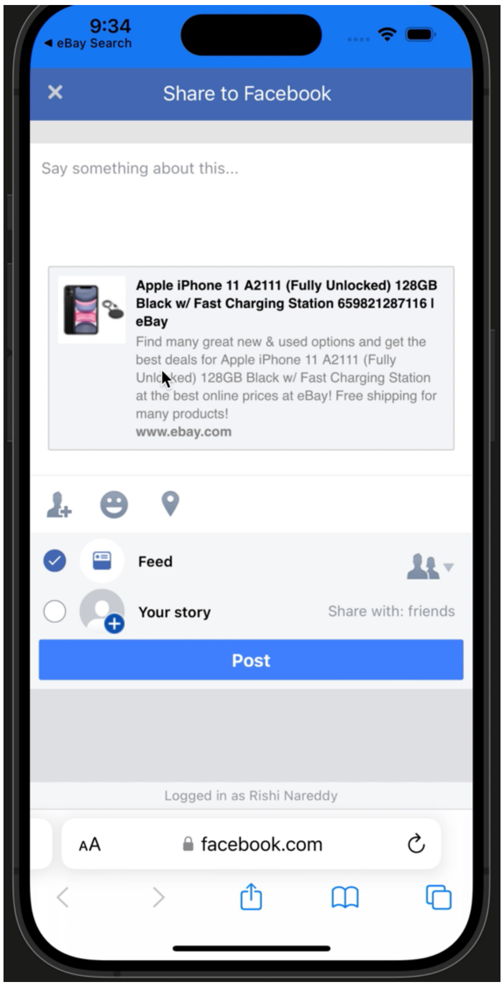
< Product search

Apple iPhone 11 A2111 (Fully Unlocked) 128GB Black w/ Fast Charging Station

$349.98

Description

Brand	Apple
MPN	A2111
Model	Apple iPhone 11
Storage Capacity	128 GB
Color	Black
Network	Unlocked
Camera Resolution	12.0 MP
Screen Size	6.1 in
Connectivity	4G, Bluetooth, Wi-Fi
Lock Status	Network Unlocked
RAM	4 GB


9:34
◂ eBay Search

Share to Facebook

Say something about this...

Apple iPhone 11 A2111 (Fully Unlocked) 128GB Black w/ Fast Charging Station 659821287116 | eBay
Find many great new & used options and get the best deals for Apple iPhone 11 A2111 (Fully Unlocked) 128GB Black w/ Fast Charging Station at the best online prices at eBay! Free shipping for many products!
www.ebay.com

Feed
Your story — Share with: friends
Post

Logged in as Rishi Nareddy

facebook.com

Figure 9: Product Details with 4 tabs **Figure 10: Share Product on Facebook**

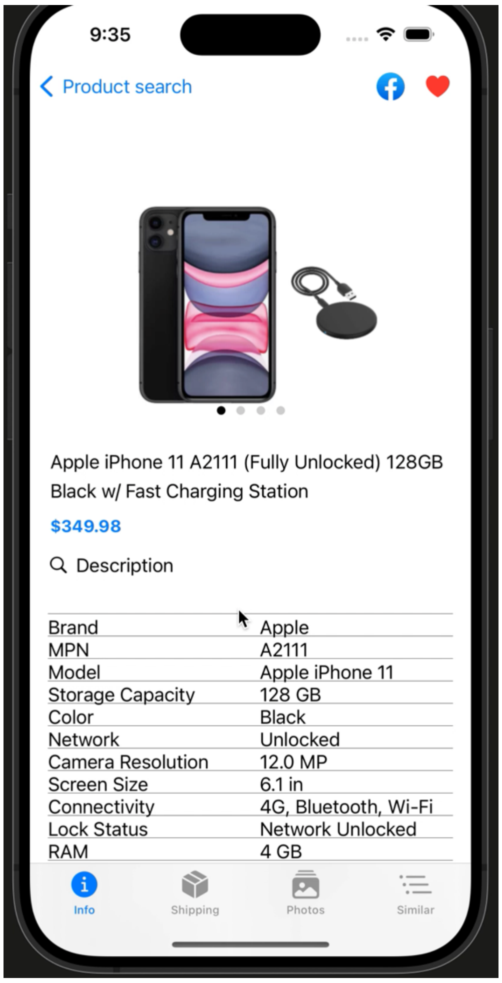
- **Wishlist button** to add/remove the product to/from the wish list.

5.3.1 Product Info Tab

The product info tab contains the following items (see Figure 10):

- A 'ScrollView' and a 'Carousel' to **horizontally** display the product images.
- Title

- Price
- Description about the product: All the elements of the 'NameValueList' in 'ItemSpecifics'.



Apple iPhone 11 A2111 (Fully Unlocked) 128GB Black w/ Fast Charging Station

$349.98

Description

Brand	Apple
MPN	A2111
Model	Apple iPhone 11
Storage Capacity	128 GB
Color	Black
Network	Unlocked
Camera Resolution	12.0 MP
Screen Size	6.1 in
Connectivity	4G, Bluetooth, Wi-Fi
Lock Status	Network Unlocked
RAM	4 GB

Figure 10: Product Info Tab

5.3.2 Shipping Tab

The Shipping tab is divided vertically into 3 sections as shown in Figure 11:

Section 1: Seller Information

Seller information contains the following:

- Store Name: Hyperlink with the name of the store name linked to the storeURL from the 'StoreName' and 'StoreURL' attributes of 'Storefront' property of Item.
- Feedback Score: 'FeedbackScore' of the 'Seller' attribute.
- Popularity: 'PositiveFeedbackPercent' of the 'Seller' attribute.

Section 2: Shipping Information

Shipping information contains the following:

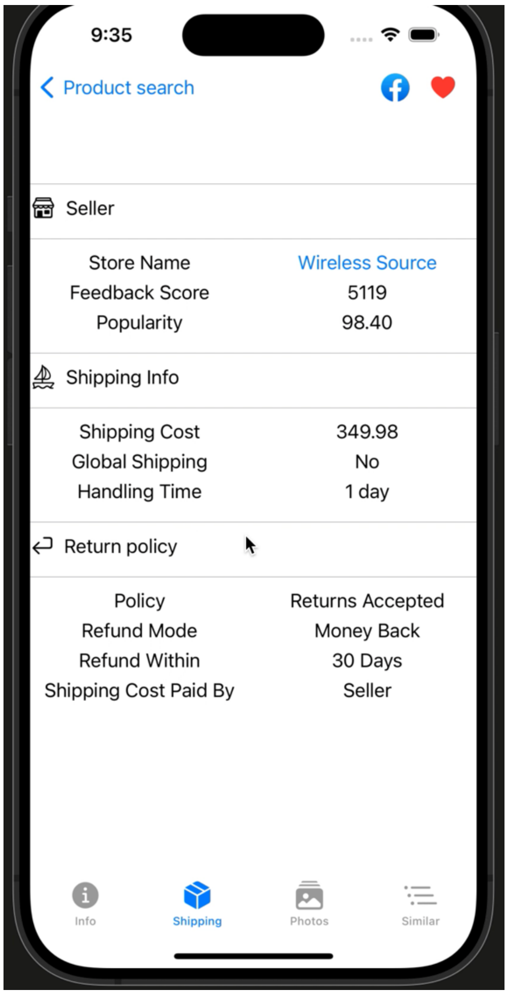
- Shipping Cost: 'FREE' if cost is 0.

- Global Shipping: ‘GlobalShipping’ property of the ‘Item’: ‘Yes’ if the value is ‘True’ and ‘No’ if ‘False’.
- Handling Time: ‘HandlingTime’ property of the ‘Item’.

Section 3: Return Policy

Return policy section contains the following:

- Policy: ‘ReturnsAccepted’ attribute of ‘ReturnPolicy’ key of the ‘Item’.
- Refund Mode: ‘Refund’ attribute of ‘ReturnPolicy’ key of the ‘Item’.
- Return Within: ‘ReturnsWithin’ attribute of ‘ReturnPolicy’ key of the ‘Item’.
- Shipping Cost Paid By: ‘ShippingCostPaidBy’ attribute of ‘ReturnPolicy’ key of the ‘Item’



Store Name	Wireless Source
Feedback Score	5119
Popularity	98.40
Shipping Cost	349.98
Global Shipping	No
Handling Time	1 day
Policy	Returns Accepted
Refund Mode	Money Back
Refund Within	30 Days
Shipping Cost Paid By	Seller

Figure 11: Shipping Tab

Error Handling:

- If any of the above-mentioned fields is missing in the API response, it should NOT be displayed.
- If ALL of the fields of a particular section are missing, the entire section should not be displayed, not even the Header of the section.

5.3.3 Google Photos Tab

Google Images are to be fetched using the title of the product and shown as ‘ScrollView’ with vertical scroll, as shown in **Figure 12**.

See homework 3 for more details of the Google Photos API.

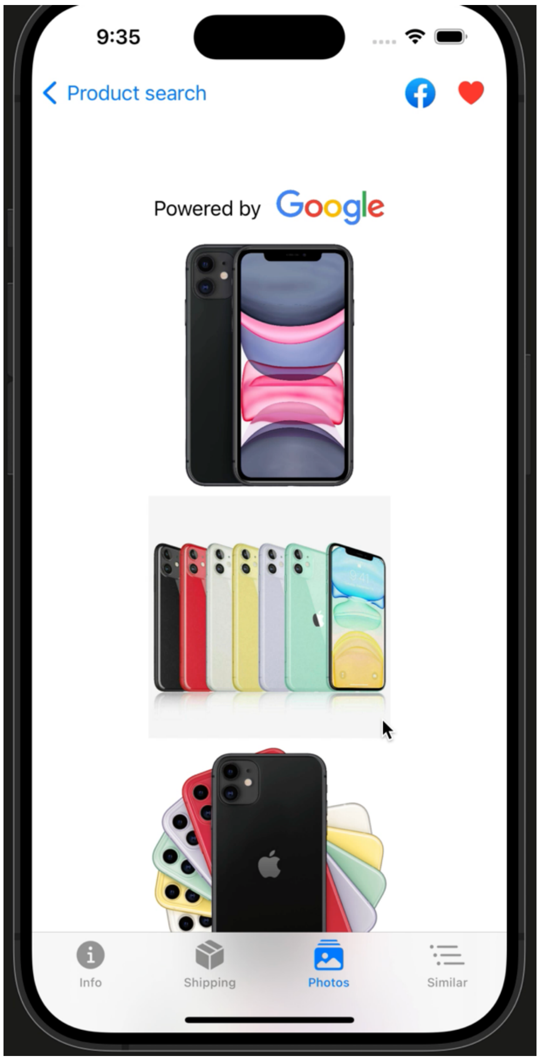


Figure 12: Google Photos Tab

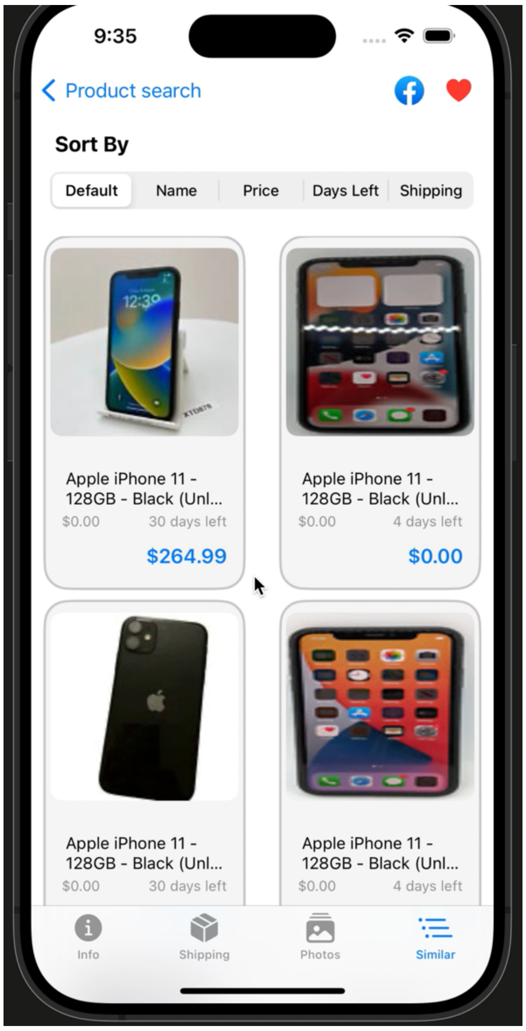
5.3.3 Similar Items Tab

As shown in Figure 13.1, you should use two segmented controls – one to switch between which parameter to sort the Similar Items on and the other to decide in what order to sort it on as shown in Figure 13.2.

Ascending/Descending option should not be displayed when user selects "Default".

The Items are shown in the form of grid.
Each of the **Grid Cell** should have the following:

- Image of the product
- Title of the product
- Price of the product
- Shipping cost of the product
- Days Left



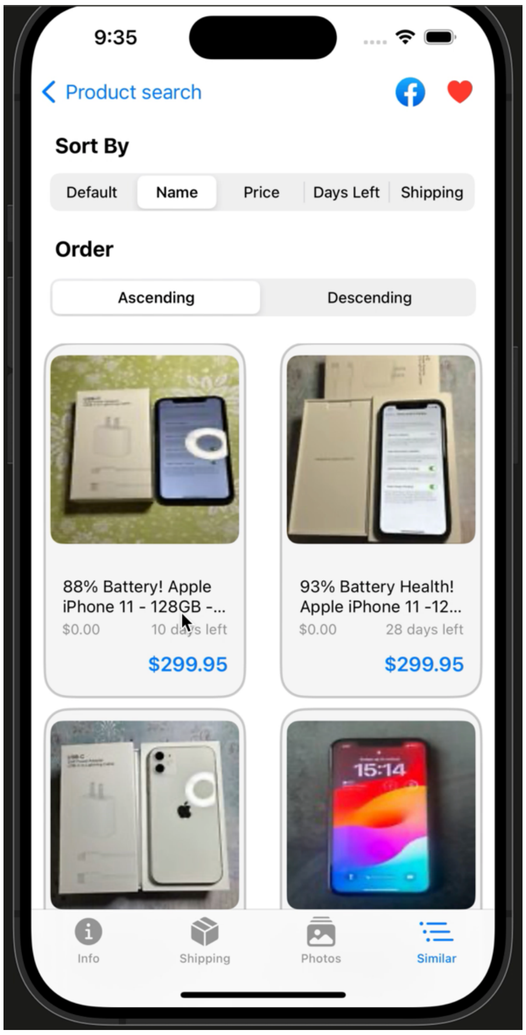


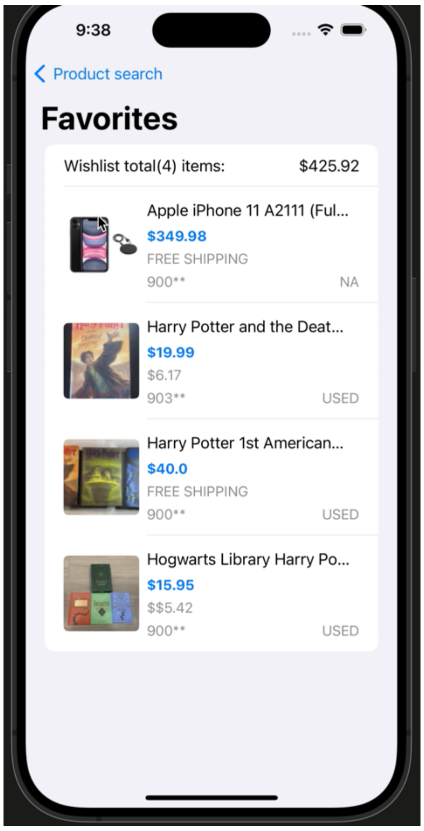
Figure 13.1: Similar Items Tab

Figure 13.2: Similar Items with Ascending/Descending

5.4 Wish List

Use a clickable favorite icon at the top-right corner of the main screen to switch between the Search screen and WishList screen. The Wish List products should be displayed in a table. Each of the table cells has the exact same structure as the search results as shown in **Figure 14,** but has **<u>No Items in wishList</u>** as shown in **Figure 15** if there are no items in Wish List. The total number of items in the WishList and sum of price of all the items should be displayed. (see **Figure 14**)

Use **"MongoDB"** to store WishList data.



Wishlist total(4) items: $425.92

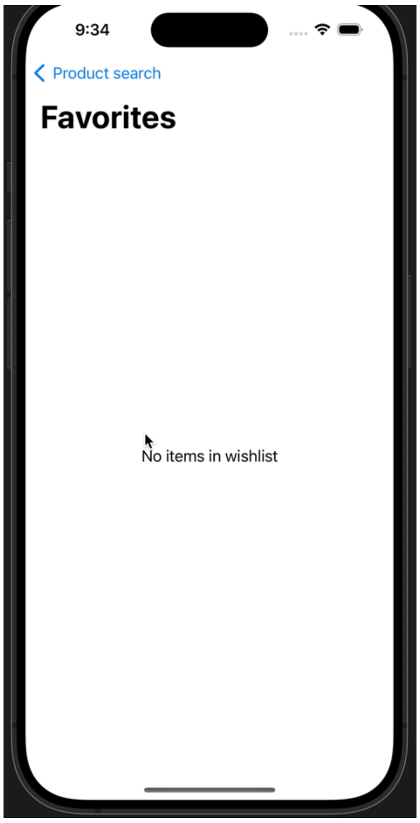
Apple iPhone 11 A2111 (Ful...
$349.98
FREE SHIPPING
900** NA

Harry Potter and the Deat...
$19.99
$6.17
903** USED

Harry Potter 1st American...
$40.0
FREE SHIPPING
900** USED

Hogwarts Library Harry Po...
$15.95
$$5.42
900** USED

Figure 14: Wish List

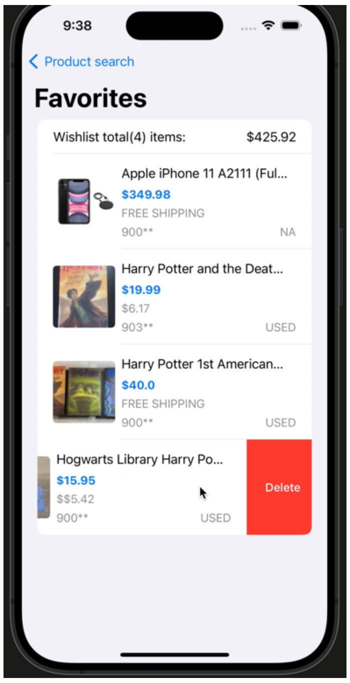


No items in wishlist

Figure 15: No Items in Wish List

5.4.2 Wish List Delete

The user should be able to remove a product by left-swiping and clicking the delete as shown in **Figure 16.1**, and left-swipe completely to delete as shown in **Figure 16.2**


9:38
Product search
Favorites
Wishlist total(4) items: $425.92
Apple iPhone 11 A2111 (Ful...
$349.98
FREE SHIPPING
900**
NA
Harry Potter and the Deat...
$19.99
$6.17
903**
USED
Harry Potter 1st American...
$40.0
FREE SHIPPING
900**
USED
Hogwarts Library Harry Po...
$15.95
$$5.42
900**
USED
Delete

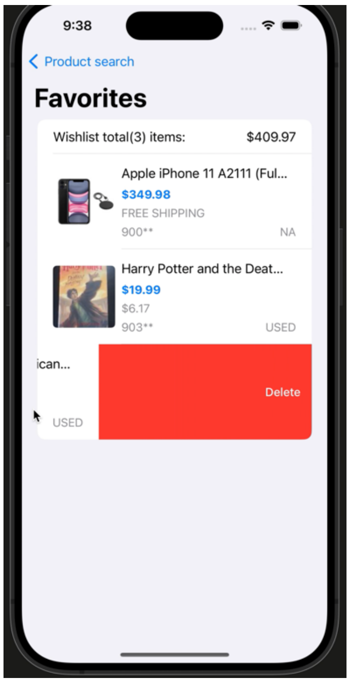


Figure 16.1: Swipe and click delete to remove a product

Figure 16.2: Swipe to remove a product

5.6 Additional

For things not specified in the document, grading guideline, or the video, you can make your own decisions. But keep in mind about the following points:

- You can only make HTTP requests to your backend (Node.js app on AWS/GCP/Azure) or use Apple Map SDK for iOS.
- All HTTP requests should be asynchronous.

6. API Documentation

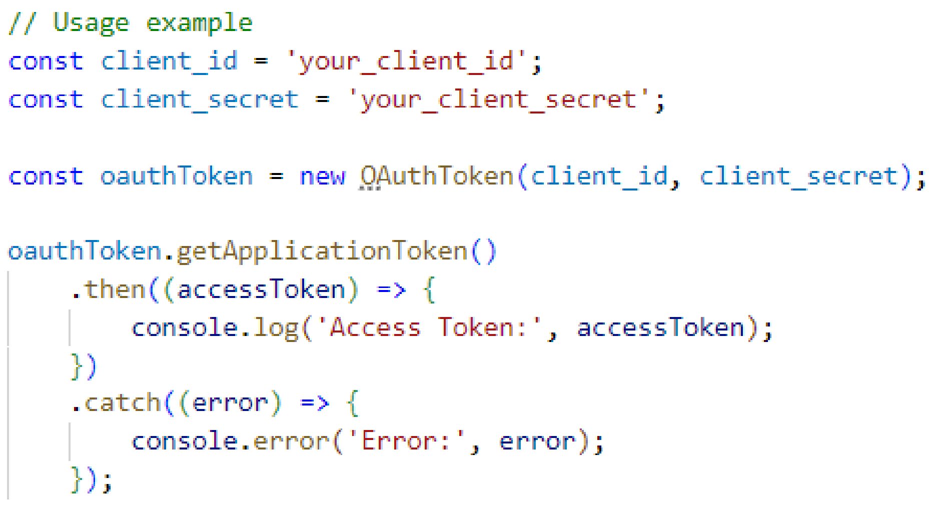
6.1 eBay API

To use eBay API, you need first to register for an eBay Account. This is the same App id used for the HW2. You can re-use it.

Note: Developers using "GetSingleItem" API calls must authenticate with an OAuth application access token in the HTTP header X-EBAY-API-IAF-TOKEN. Follow the steps below to create an OAuth token.

1. Download the *ebay_oauth_token.js* file shared with this assignment. You will need to use the **OAuthToken** class in your backend to generate OAuth application access token before calling *GetSingleItem*

2. Use the following code in your backend to generate the OAuth application access token and use it in your *GetSingleItem* call.



```
// Usage example
const client_id = 'your_client_id';
const client_secret = 'your_client_secret';

const oauthToken = new OAuthToken(client_id, client_secret);

oauthToken.getApplicationToken()
    .then((accessToken) => {
        console.log('Access Token:', accessToken);
    })
    .catch((error) => {
        console.error('Error:', error);
    });
```

6.2 Google Customized Search

This link will provide the details to get the API key:
https://developers.google.com/custom-search/json-api/v1/overview

7. Implementation Hints

7.1 Images

The images used in this homework are available in the zip file in the D2L Dropbox folder and will be linked in Piazza post for HW 4. The videos also will be uploaded on D2L and the YouTube video linked on Piazza post.

You can either replace the Assets.xcassets folder in your project with the given one or add the assets and edit Assets.xcassets folder under your project in Xcode yourself.

7.2 MongoDB

Use MongoDB to store the items in the WishList. The Setup for MongoDB is exactly the same as it is for Homework 3. Refer Homework 3 Documentation.

NOTE: Using MongoDB is a hard requirement. Failing to MongoDB will lead to a major point deduction. Refrain from using any other method such as "UserDefaults".

7.3 GET Call

You **must use a GET method** to request the backend since you are required to provide the above additional link to your homework list to let graders check whether the Node.js script code is running in the "cloud" on GCP/AWS/Azure (see 5.4 above). Please refer to the grading guidelines for details.

7.4 Good Starting Point

There will be a Piazza with links to great tutorials, here are a few core ones. Make sure to understand SwiftUI and iOS development before you start the homework. It will save you a lot of time.

Introduction to SwiftUI - WWDC 2020 - Videos:
https://developer.apple.com/videos/play/wwdc2020/10119/

SwiftUI Tutorials: https://developer.apple.com/tutorials/swiftui

View Modifiers: https://developer.apple.com/documentation/swiftui/viewmodifier

SwiftUI cheat sheet: https://jinxiansen.github.io/SwiftUI/

7.5 Displaying ProgressView

https://developer.apple.com/documentation/swiftui/progressview

7.6 Implementing Splash Screen

https://kristaps.me/blog/swiftui-launch-screen/

7.7 Working with NavigationBar and App Navigation

Adding a navigation bar - a free Hacking with iOS: SwiftUI Edition tutorial:
https://www.hackingwithswift.com/books/ios-swiftui/adding-a-navigation-bar

The Complete Guide to NavigationView in SwiftUI:
https://www.youtube.com/watch?v=nA6Jo6YnL9g

7.8 Working with TabView

Adding TabView and tabItem():
https://www.hackingwithswift.com/quick-start/swiftui/adding-tabview-and-tabitem

TabView: https://developer.apple.com/documentation/swiftui/tabview

7.9 Adding Toasts

SwiftUI: Global Overlay That Can Be Triggered From Any View:
https://stackoverflow.com/questions/56550135/swiftui-global-overlay-that-can-be-triggered-from-any-view second answer.

7.10 Open Link in browser

How to open web links in Safari - a free SwiftUI by Example tutorial:
https://www.hackingwithswift.com/quick-start/swiftui/how-to-open-web-links-in-safari

7.11 Image related

How to disable the overlay color for images inside Button and NavigationLink:
https://www.hackingwithswift.com/quick-start/swiftui/how-to-disable-the-overlay-color-for-images-inside-button-and-navigationlink

SwiftUI Image clipped to shape has transparent padding in context menu:
https://stackoverflow.com/questions/62687224/swiftui-image-clipped-to-shape-has-transparent-padding-in-context-menu

7.12 Adding a Sheet for zipcode results

How to present a new view using sheets - a free SwiftUI by Example tutorial:
https://www.hackingwithswift.com/quick-start/swiftui/how-to-present-a-new-view-using-sheets

8. Files to Submit

You should also ZIP all your source code (the .swift/ and directories exclude other files) and submit the resulting ZIP file to the DEN D2L Dropbox folder.

You will also have to submit a video of your assignment demo to the same DEN D2L Dropbox folder. You must demo your submission using Zoom. Details for how to create the video are on DEN D2L.

Demo is done on a MacBook using the simulator, and not on a physical mobile device.

IMPORTANT

All videos are part of the homework description. All discussions and explanations in Piazza related to this homework are part of the homework description, and will be accounted into grading. So, please, review all Piazza threads before finishing the assignment.