Summary

In this sample project you will build the foundation of a new shopping app for our fashion brand partner Chloé.

The app should present a Homepage where the user can select one of the four main product categories:

- Dresses
- Skirts and shorts
- Bags
- Sneakers

When the user selects a product category, the app will show a list of products available for that category, with basic information displayed for every product.
The user can also access a detail page with more information on a specific product. At any time, the user should be able to easily navigate between the main menu with the product categories, a products list, and product detail page.

Requirements

- Support for iPhone in portrait.
- Support for iOS 13.0 and greater.
- Written in Swift ≥ 5.0 making use of UIKit and Autolayout on Xcode 12 or later.
- At startup, the app displays a view with the main departments with an image for each one; you can choose any image you want to represent the departments.
- Together with each image the app will display a label for that department name.
- Tapping on one of the category images will take the user to a list of products available for that category
- Each product category list will display, for each item:
    - One thumbnail image representing the product.
    - The product title.
    - The price of the product.
- Tapping on a single product will take the user to a product detail view; this view will display:
    - One image representing the product.
    - The product model name.
    - The product macro category.
    - The product micro category.
    - The product price.
    - Any additional useful information which comes from the API.
- At any time, the app allows the user to go from a product detail view back to the category list that product belongs to, and from the category list back to main menu.

Resources you will need to use to complete the test

- The products lists representing the four main categories are available in JSON format with the Search API:

        - Dresses: https://api.yoox.biz/Search.API/1.3/CHLOE_GB/search/results.json?ave=prod&productsPerPage=50&gender=D&page=1&department=drsss&format=lite&sortRule=Ranking
        - Skirts and shorts: https://api.yoox.biz/Search.API/1.3/CHLOE_GB/search/results.json?ave=prod&productsPerPage=50&gender=D&page=1&department=skrtsnds&format=lite&sortRule=Ranking
        - Bags: https://api.yoox.biz/Search.API/1.3/CHLOE_GB/search/results.json?ave=prod&productsPerPage=50&gender=D&page=1&department=nwrrvlsbgs&format=lite&sortRule=Ranking
        - Sneakers: https://api.yoox.biz/Search.API/1.3/CHLOE_GB/search/results.json?ave=prod&productsPerPage=50&gender=D&page=1&department=shssnkrs&format=lite&sortRule=Ranking
- Each item of the Search.API provides the required information in the MicroCategory, FullPrice and ModelNames fields
- To show additional product information in the product detail page, such as colours and sizes, the Item API can be used: https://api.yoox.biz/Item.API/1.0/CHLOE_GB/item/<Code8>.json
    - Code8 is provided for each item by the Search.API
- The URL for the product images is: https://cdn.yoox.biz/<folderIdentifier>/<DefaultCode10>_<resolution>_<type>.jpg
    - folderIdentifier -> first two numbers of the *DefaultCode10* attribute, available through both the Search and Item API
    - DefaultCode10 -> value of the field with the same name in the product information
    - Resolution:
        - 8 - 45x60 pixels
        - 9 - 90x120 pixels
        - 12 - 120x160 pixels
        - 13 - 240x320 pixels
        - 17 - 320x427 pixels
        - 18 - 640x853 pixels
        - 19 - 400x533 pixels
        - 20 - 800x1067 pixels
        - 21 - 480x640 pixels
        - 22 - 960x1280 pixels
        - 23 - 1536x2048 pixels
        - 24 - 3072x4096 pixels
    - Type -> "F" for the front shot
    - Eg: http://cdn.yoox.biz/45/45577274CH_18_F.jpg

Guidelines for development

- Usage of a version control system is required, with your favourite branching model.
- Where possible, Apple frameworks should be preferred to third-party libraries; however, it is acceptable to use third-party libraries as long as you provide a reason for it.

- MVC architectural pattern is not strictly required and other solutions are well accepted.
- It is generally preferable to use the most recent technology provided by iOS to solve a problem, but it is up to you to choose what to use; just make sure you can explain why you chose a technology over another.

Bonus Tasks

Please approach the following tasks only after you have completed all the tasks listed above under "requirements".

- Try to handle asynchronous tasks (like network calls) without completely blocking the user interface.
- Manage connectivity issues or possible error returned by the APIs.
- Provide some Unit tests and/or UI tests written with the XCTest framework.
- Feel free to develop some optional parts with Swift UI as well.
- **Express your creativity! :)**

The time available to complete and return this test is five calendar days.
Please upload here a zip file containing your Xcode project comprehensive, if available, of the .git folder.

Good luck!