**Xfinity – Comcast**

**Introduction**

The document is to understand the exercise application by Comcast

**Implementation Approach**

* As the two-app version which were going to use the shared code, the best option was to put the code in the external framework and use the framework by call the initializers from respective apps.
* Created **BaseApp Framework** as **CocoaTouchFramework** to implement common functionality between the two apps.
* Used **Carthage** as dependency manager to integrate external dependencies in the project.
* Used **UISplitViewController** to have UI functionality for iPad and iPhone.
* Created separate layer to make service call and separate the functionality.
* Created extension for **UIViewController** to have a generic method for showing alerts.
* Create extension of **String** to have a functionality of splitting the string by hyphen.

**Architecture Used**

* Used **MVVM** **architecture** to have separate layers between **view, view model and model** classes to separate the UI and Logical functionality.

**Patterns Used**

* Used factory pattern to initialize factory for two apps having same functionality

**External Library Used**

* **SDWebImage –** Used for **automatic image** **caching** for characters.

**Modules in Application**

* **BaseApp Framework** – Cocoa touch framework which contains common code to be used inboth apps.
* **The Wire Character Viewer** – Application to be designed.
* **Simpsons Character Viewer** – Application to be designed.

**How to run the project**

1. Open the project using file.
2. Select BaseApp scheme and build.
3. Change scheme to **The Wire Character Viewer** or **Simpsons Character Viewer.**
4. Run the app.

**Base App**

* **Helper –** Has files/class for functionality throughout the application.
* **Main Application –** To implement the functional required in the exercise.
* **Model Class –** To have the model classes for the service call.

**Helper**

* **Application Manager –** Purpose of this class is to initialize the app with the correct app type and give the initial controller to the child apps.
* **Constants –** To store the constants such as default strings/name of controller/storyboards to be used throughout the app.
* **Character Request Manager –** To make the service call with the appropriate URL and return the content. The method in the call can receive any url and return content.
* **Bundle Helper –** To determine the bundle of the base app framework.
* **UIViewController+Alert –** Generic utility file to show any kind of alert. (Used in app for service failiure)
* **String+SplitHelper –** Generic utility for parsing content of the service and splitting using hypen for title and detail text.

**Main Application**

* **CharacterList –** Contains all the views/storyboards, view controller, view models to be used to show character list.
* **CharaterDetail –** Contains all the views/storyboards, view controller, view models to be used to show character detail screen.
* **AppTypeFactory –** As functionality for both the apps was same the best pattern could be used was factory

The group contains a base factory class names **CharacterFactory.**

Also has two sub-factories **SimpsonFactory** and **WireCharacterFactory** which inherit the properties of **CharacterFactory.**

**CharacterFactory –** Initializes the property such as url, app type and has method to make service call and get data.

**Model Classes –** Has class such as CharacterModel, RelatedTopics, Icon to store the content of the service.

Also wrote **Unit Test** for the view models for controllers in target **BaseAppTests**