Singh, Viresh
Viresh.Singh@Live.com | 9902704000
https://www.linkedin.com/in/viresh-singh-33935835/

Coding Assignment

Problem Statement

To download the JSON data from remote server and show the objects in a list. Allow user to navigate to view the details of particular item by tapping on it. If an item has image to be downloaded, download and show the image.

Expected Output Parameters

- Coding standards
- UI performance
- Error Handling
- Test Cases

Solution Description

Environment Specification

Application developed on below environment:
XCode: 10.1
Swift: 4.2
Mac OS: Mojave 10.14
Source Code Location: https://github.com/VireshS/DataLoader

1. **Implemented Functional Features:**
   - Download the movies JSON data from remote server and show them up in the table view
   - while showing movie title and image (if configured).
   - Allow user to scroll the table and tap on any movie to go to details page
   - Show all additional associated details of the movie
   - When image is being downloaded, app will show the progress and if image is not available it will show default image.

2. Implemented Unit Test Cases

Test cases are written to demonstrate the idea of how it can be tested. So, as of now app does not have 100% coverage.
However, test coverage can be increased if required.
Below are the high-level test cases which are written:

- Try to parse invalid JSON into Movie object/list
- Try to parse single movie object having valid key names and data
- Try to parse array of movies object having valid key names and data
- Try to parse array of movies object having in valid key names (with leading and trailing spaces) and data
- Testing performance of parsing huge json.

3. Implemented UI Test Cases [Implemented as Extra feature]

- Sample test for testing if all images are loaded in the list and if user is able to navigate to the details page. And also, to verify if basic details of the movie are listed on details page

4. Bonus Features/Non-Functions/Performance Sensitive use cases:

- Asynchronous execution of all JSON and image download operations and parsing of JSON/Image data to native objects using OperationQueue and Operation sub classing architecture.
- Lazy and On demand download of the movie images based on when they actually required.
- Configuration to control how many images can be downloaded at a time rather than invoking them all together
- Storing the Images into NSCache to avoid memory pressure at times.
- Dedicated detailed error handling.
- Application can handle the json keys with any number of leading or trailing space together with cases differences.
- Modularized architecture having single responsibility classes and modules

5. Improvement Opportunities:

- As this application uses the test version of JSON response, it relies on traditional way of handling and parsing the JSON. But in real world scenarios, most of the time those JSON keys remains consistently same. So, Codable protocol can be used to parse the data. Codable was introduced in swift which works on modern encoding/decoding mechanism of data.
- Download progress for images can be shown if required.
- Error handling can be extended to handled detailed standard http and business errors.
- Images can be also stored in document directory or local db (depending upon data security concerns) to have the tertiary store to look up if some images are trashed from cache when system is under resource pressure to avoid downloading it again over network.
- Application can validate and verify the server public key to become immune to MIM attacks.

6. Reusability:

- Developed custom NetworkOperationsQueue, DataDownloader Operation and the Image caching classes can be directly reused in other applications too.
- Application error class can be reused to have efficient error handling

7. Application Security

- As of now application does not store any information on disk in any way like UserDefaults, Local DB, Keychain, document directory etc.
- Application does not expose any extension, interface or deeplink to be invoked or establish any kind of data sharing with external entity.
- Application relies on default OS Server Trust Evaluation policy, hence does not allow communication with hosts having invalid, expired or self-signed SSL/TLS certificate
- However, application is still **vulnerable for Man In Middle attack** as it does not verify the Server public key.