Singh, Viresh

Viresh.Singh@Live.com | +91 9902704000

TECHNOLOGY SPECIALIST – MOBILITY, HONEYWELL

https://www.linkedin.com/in/singhviresh/

Environment Specification

XCode: 10.1, Swift: 4.2, Deployment Target iOS 12

Source Code Location: https://github.com/VireshS/SeatGeekAPIClient

1. High Level Functional Features:

- Download the sports events data for provided query parameter.
- Showing image, title, time and location for each sports event
- If primary performer image is not available for an event, show the secondary performer image
- Allowing navigation to view details of a particular event
- Marking or unmarking a sport event as favorite and showing its favorite status
- 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 Non-functional requirements

- Downloading image for a sports event asynchronously in background
- Caching the downloaded images in memory (not persistently) to reuse just in case if more than one event is using the same image or if same sports event is listed for various search query or even when user performs the search for same query again
- Optimized network data downloads with restricted to only 4 downloads at a time for any image download and 1 download at a time for any search query download.
- Lazy and On demand download of the movie images based on when they actually required.

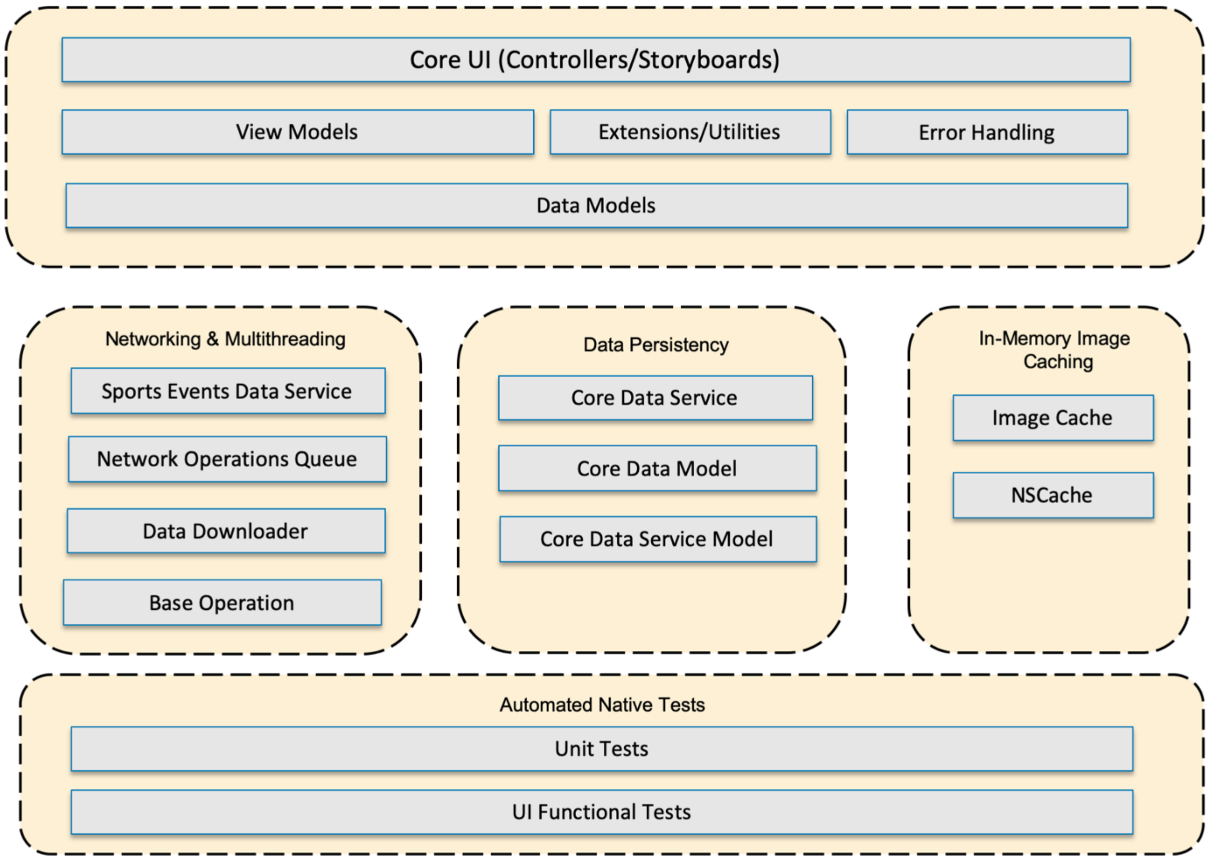
3. Technology Background

- Multithread application for various network and json parsing task in background threads to keep the main thread light and active.
- Lightweight in memory image cache to fully coordinate with application memory pressure
- Using Coredata to persist the favorite sports event information across app life cycles

4. Design and Architecture

- Uses MVC/ MVVM, Single responsibility, Thread Pool principles
- Uses layered service architecture or lying down the single responsibility modules with their well-defined responsibilities
- Refer details below for architecture of the application:

Application Layered Architecture



5. Bonus Functionality

- Image download progress will be shown whenever image download in progress
- Error handling for network related errors

6 . Implemented Unit Test Cases

Below are the high-level test cases which are written:

a. Networking

- Handling invalid json for Sports events API response
- Handling empty json for Sports events API response
- Handling valid json for Sports events API response and validating the overall json serialization
- Try to parse array of movies object having in valid key names (with leading and trailing spaces) and data

b. CoreData

- Storing favorite sports event information
- Performing CRUD operations for sports event for marking them as favorite or non-favorite

c. Storyboards

- Checking if able to load the designated ViewControllers
- Checking if root controller is embeded in navigation controller and has default initial controller
- Checking if root controller is the correct controller and not some other controller

7. Implemented UI Test Cases

- Checking initial screen elements to be available
- Performing automated search for a keyword
- Validating the results populated for the search query
- Tapping first item in list and navigating to details page

8. Known Issues

- Sometimes for few images progress bar might be shown infinitely

9. Improvement Opportunities:

- Unit test cases can be extended to test View model and over all integrates Test with Actual API call or mocked API response
- Application relies on traditional way of handling and parsing the JSON. Codable protocol can be used to parse the data. Codable was introduced in swift which works on modern encoding/decoding mechanism of data. SwiftyJson is an another 3rd party library which can be used for parsing json.
- 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.
- If its our own api and server, Application can validate and verify the server public key to become immune to MIM attacks.

10. Reusability:

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

11. Application Security

- Application stores the ID for each Sports Event which is marked as favorite in plain text in core data. However, application does not save any other data in any form on disk or keychain or UserDefaults or 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.