iFrames Report

I used two frameworks I didn't have any knowledge about before during development of this app. Rx Swift is an open source reactive programming framework and Realm which is a data persistence framework supports multi platform. I used programmatic UI for the first time which was challenging in the beginning.

Reactive programming improve overall responsiveness, scalability and resiliency of our system thanks to the asynchronous message-driven architecture. It makes UI more dynamic and user friendly.

In the project configureSearch() function starts observing users input in the search bar and shows the results by calling function setupCellConfiguration() which fills the CollectionView according to data received in .results(let list)

For the main movie search list if search bar text is empty network class calls different api path to get user top 100 movie list, this list is saved in the Realm Data Base and persist when user opens app again. Images are not saved to the Realm directly, it's saved to documents folder and this path is called during the launch of the app. To show movie's first three genres, genre_ids array results from api call translated in to the corresponding genres.

When user clicks a movie in the movie list screen, user goes to the detail screen. This screen shows movie popularity, overview and larger image. As user scrolls to the end of overview user sees the fan art image, for not, short scrollable overviews I used a simple timer. Network class used to check connection status and when user went from offline to online another search performed by movie id which is passed in from the main list. So user can update UI with out refreshing his/her screen.

MVC design pattern is used during development of the application. MVC is an essential design pattern widely used in iOS development. This pattern makes app more maintainable and scalable for future expansions.

During the development I put my API Key to network class which was a bad exercise. Sensitive data like this should be put in another file and this file should be added to the gitignore file. Because even at the end you separate your keys, this information will be all over the place in your commit history. I spent some time reading Github documentation to fix this issue. After separating my key from the Network layer, I copied this network file and applied this long command below to clean my history about Network file:

```
git filter-branch --force --index-filter \
  "git rm --cached --ignore-unmatch iFrames/Managers/NetworkManager.swift" \
  --prune-empty --tag-name-filter cat -- --all
```

As i checked the history I realized it only removed after I moved the Network file in the Managers folder. Commits in the beginning of the project still had my api key. So I applied code above again with path: iFrames/NetworkManager.swift

Finally my commit history was clean, I added the keys.plist file to gitignore and good to go. This was a good experience for me how important it is not to commit your sensitive data EVER. Using good approaches, architecture and design patterns before writing your code can save you a lot of time in the future.

I gained a lot of experience during this project, I could have made some improvements if I had more time. My Rx code is not so clean, it gets triggered by network search results. It would be better to bind it to all states and show user custom cells like loading, no results etc. I'm happy with the search screen UI but My detail screen could have been better with good design and nice animations. I realized at the end of the project top 100 list is updated daily, so my Realm db should too. Unfortunately I didn't have time to fix it. Adding some meaningful unit tests would be great to prevent future mistakes.

I used CoreData in the pas for a simple project, its syntax was hard and you can't use it cross platform. I really loved elegance of reactive programming. I'd love to learn more about RxSwift and Realm frameworks.