Assignment 2- Android

For the choice of database, the same database from the previous assignment is chosen. Retrofit2 is used to fetch the data from the database API. For the use of Retrofit an interface called **APIFetch** is created, in this interface two methods are declared: *getTrending* and *getQuery*; both methods have required query fields, both has the API-key as one, the last takes a query string in addition. In order to keep all the code related to fetching data self-contained, a singleton class, called **APIHelper**, is created. This class constructs the retrofit object using the **APIFetch** interface. This class also has a method for resolving the response from the API, this method takes two callBacks, first is for defining a successful response, secund is for a failed response.

Going from view to view is done with different Activity classes, to split up the responsibility. Fragments have not been used as no part of the app is used more than one place, hoverer if a part was used more than one place fragments would have been used.

The app implements MVVM. The model is **MovieModel** which is implemented as a data class with attributes based on the values present in the response from the API. The views are both activity\_main and activity\_movie as they both use the model. **MovieViewModel** is implemented as the ViewModel, as the adding and removing comes in bulk, it contains a method for adding many, getting all and removing all.

For displaying the data movies on in the **MainActivity** a **RecyclerView** is used with a corresponding adaptor, called **MovieAdapter**, this adapter is initialized to have access to **MovieViewModel**, so the **RecyclerView** can display the movies stored in the **ViewModel**.

A **SearchView** is used for getting the query string needed to use the *getQuery* method, whenever an input occurs in the **SearchView**, a request is sent to the API with the current value in the **SearchView**, the **MovieViewModel** is then updated with the resulting values and the **RecyclerView** is notified that its dataset has been updated. If the **SearchView** text value is an empty string, the getTrending method is called instead of getQuery.

Each **ItemView** in the **RecyclerView** has an OnClickListener that starts the **MovieActivity** with the movie of the selected **ItemView** in the Intent. This movie is then retrieved from the Intent and is used for populating view.

Any hardcoded strings like the API-url and API-key, are stored in recourses in strings.xml, so anywhere in the code where any of there values are used, they er done so by getting them from the context *getString* method. By storing these values like this, it ensures that if one of them changes, for some reason, it would only be one place that needed to be updated.