# Movie Search App Report

A simple movie search app using google design guidelines.

## Goals in Development

My goal was to create an extensible movie search application that could grow with age

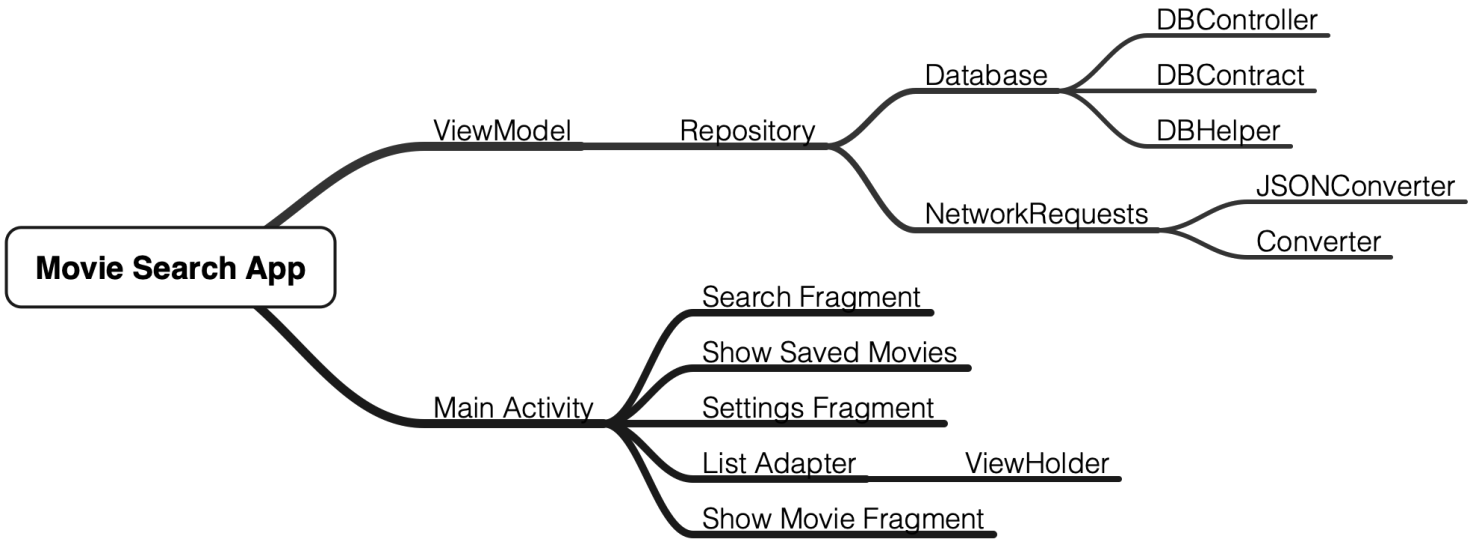
## Target Audience

My targeted audience is anyone who wants to keep track of the movies they're watching.

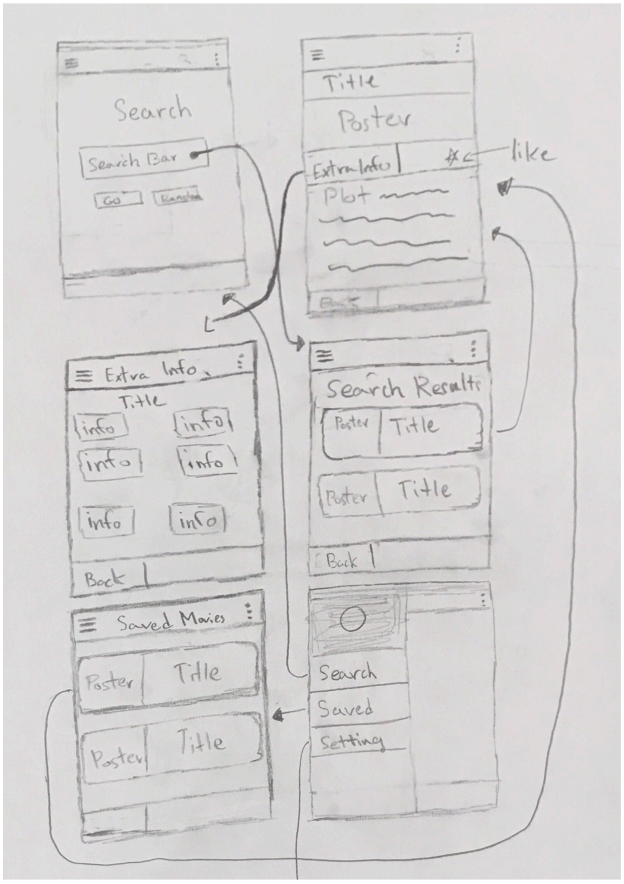
- Age Group: Everybody
- Gender: Male and Female
- Occupation: Any
- Genre: The genre is Movies

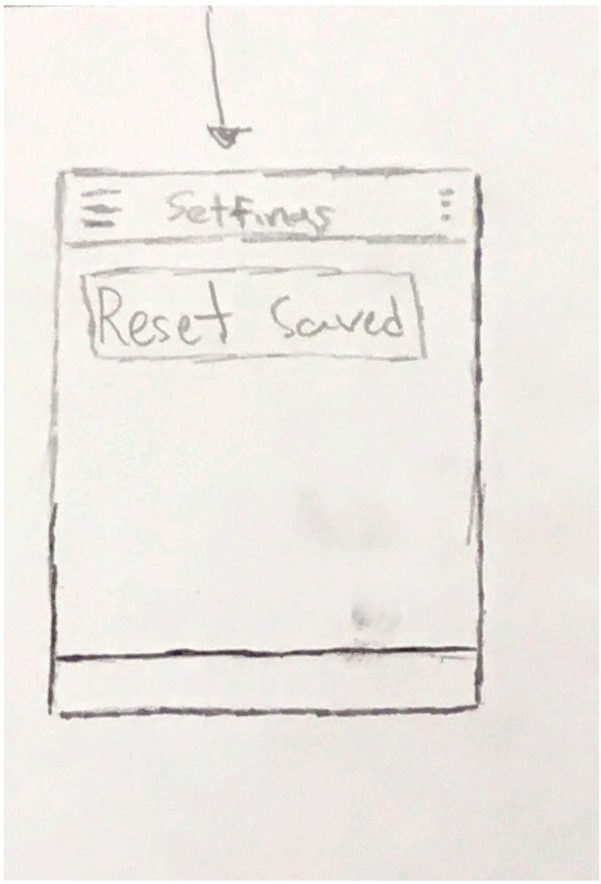
## App Architecture

This is how I planned the architecture of my application. It took a lot of planning and replanning to create the final product.

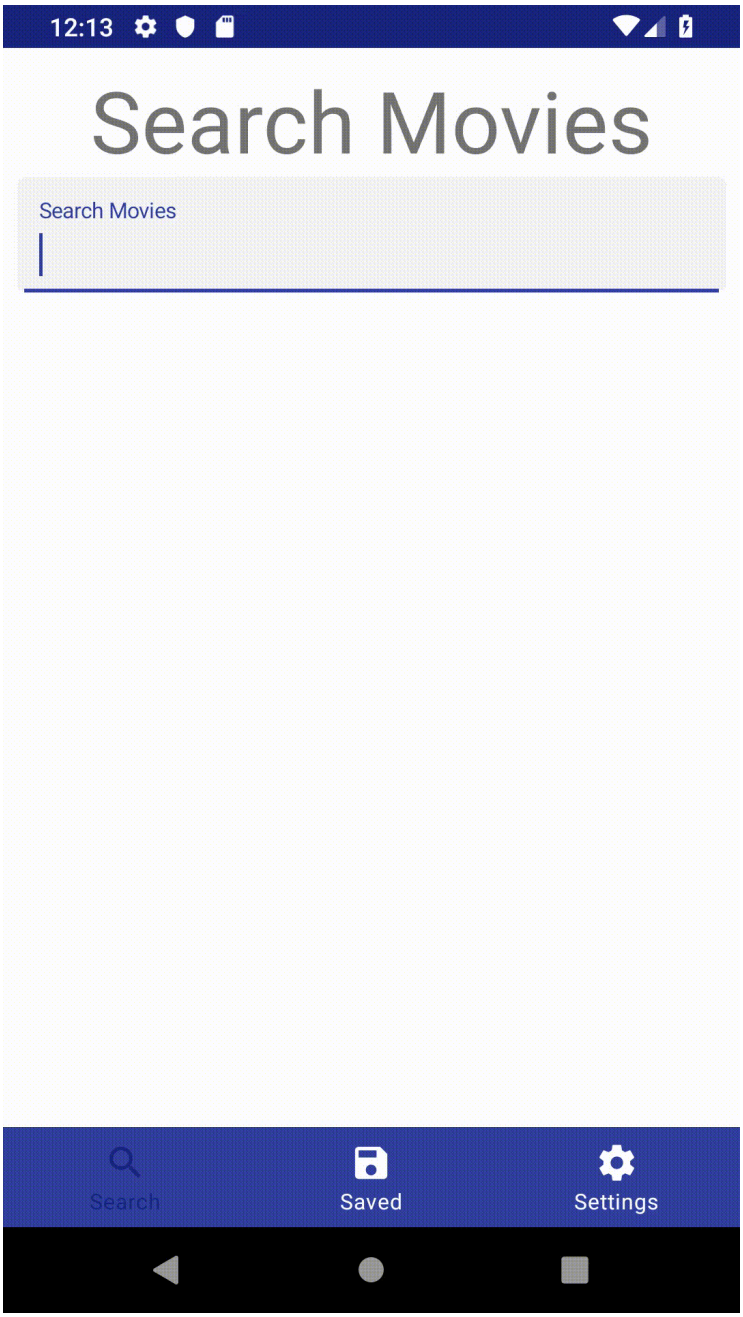

Movie Search App
ViewModel
Repository
Database
DBController
DBContract
DBHelper
NetworkRequests
JSONConverter
Converter
Main Activity
Search Fragment
Show Saved Movies
Settings Fragment
List Adapter
ViewHolder
Show Movie Fragment

# Wireframes


Search
Search Bar
Go
Title
Poster
Extra Info
#c
like
Plot
Extra Info
Title
info
info
info
info
info
info
Back
Search Results
Poster
Title
Poster
Title
Back
Saved Movies
Poster
Title
Poster
Title
Search
Saved
Setting


Setfings
Reset Saved

# App In Action




# JSON Request Example

This is how a JSON request from *The Open Movie Database* looks like.

```
{
    "Title": "Fantastic Beasts and Where to Find Them",
    "Year": "2016",
    "Rated": "PG-13",
    "Released": "18 Nov 2016",
    "Runtime": "133 min",
```

```
    "Genre": "Adventure, Family, Fantasy",
    "Director": "David Yates",
    "Writer": "J.K. Rowling",
    "Actors": "Eddie Redmayne, Sam Redford, Scott Goldman, Tim Bentinck",
    "Plot": "The adventures of writer Newt Scamander in New York's secret community of
 witches and wizards seventy years before Harry Potter reads his book in school.",
    "Language": "English, Khmer",
    "Country": "UK, USA",
    "Awards": "Won 1 Oscar. Another 14 wins & 51 nominations.",
    "Poster": "https://m.media-amazon.com/images/M/MV5BMjMxOTM1OTI4MV5BMl5BanBnXkFtZTg
wODE5OTYxMDI@._V1_SX300.jpg",
    "Ratings": [
        {
            "Source": "Internet Movie Database",
            "Value": "7.3/10"
        },
        {
            "Source": "Rotten Tomatoes",
            "Value": "74%"
        },
        {
            "Source": "Metacritic",
            "Value": "66/100"
        }
    ],
    "Metascore": "66",
    "imdbRating": "7.3",
    "imdbVotes": "335,901",
    "imdbID": "tt3183660",
    "Type": "movie",
    "DVD": "28 Mar 2017",
    "BoxOffice": "$234,018,657",
    "Production": "Warner Bros. Pictures",
    "Website": "http://www.fantasticbeasts.com/",
    "Response": "True"
}
```

# Movie Model

```
// My movie model. Includes most of the fields in OMDDBApi
data class Movie(
        var title: String,

        var year: String,

        var rated: String,

        var released: String,

        var runtime: String,

        var genre: String,

        var director: String,

        var writer: String,

        var actors: String,

        var plot: String,

        var language: String,

        var country: String,

        var awards: String,

        var poster: String,

        var metascore: String,
        var imdbRating: String,
        var imdbVotes: String,

        var imdbID: String,

        var type: String,

        var response: String,
        var saved: Boolean
)
```

# Network

- NetworkRequests
    - Converter
        - JSONConverter

## Abstract Converter

```
// abstract converter class that converts a T value to a V value and a V value to a T value
abstract class Converter<T, V> {
    abstract fun doForward(input: T): V
    abstract fun doBackward(input: V): T
}
```

## JSONConverter

```
class JSONConverter() {
    // given a omdb api json object, converts it to a movie object
    fun getMovie(c: JSONObject): Movie

     // given a jsonarray of movie titles, converts them into a movie and returns a movie list
    // converts them through a converter object that does the conversion of an individual itself
    fun getMovieList(c: JSONObject, movieTitleConverter: Converter<String, Movie?>): List<Movie>
}
```

# NetworkRequests

```
// does all network requests for getting a movie
// is given the private application directory
class NetworkRequests(private val filesDir: File) {
    // instantiates a json converter object
    private var jsonConverter: JSONConverter = JSONConverter()
    // a photo counter that gives a unique name to each poster on the device
    private var posterCounter = 0

    // given a url to an image, returns a bitmap of the of said image
    // uses input streams and BitmapFactory
    private fun getPhoto(url: String): Bitmap?

    // given a bitmap, creates a copy of that bitmap on the device
    // uses the private directory of app on device
    // returns a Uri pointing to that image on device
    private fun addFile(bitmap: Bitmap): Uri

    // creates an api request url for OMDb API.
    // stores my api key
    // uses Uri Builder
    // takes a map of different url options
    private fun movieRequestCreator(options: Map<String, String>): String

    // returns the contents of a url request.
    // uses URL to create an input stream
    // uses a buffer
    // uses InputStreamReader
    private fun getString(url: String): String

    // Returns an async callable object of List<Movie>
    // checks if response is false of the string
    // uses my request creator
    // uses jsonConverter. Gives my JsonConverter a special converter for mvoies
    // downloads a local version of the movies posters
    // checks to see if the poster is downloadable
    // converts the movie.posters url into a uri
    fun getMovies(query: String): Callable<List<Movie>>

    // special converter class
    // converts a string into a movie
    // uses a combination of JsonConverter and NetWorkRequests
    // also checks for whether the language of the movie is english
    // can also turn a movie object into a string
    inner class TitleMovieConverter : Converter<String, Movie?>()
}
```

# Database

- MovieDBController
    - MovieDBHelper
    - MovieContract

## Contract

```
object MovieContract {
    // Movie database entry labels represented as final values
    object MovieEntry : BaseColumns {
        const val TABLE_NAME = "movie"
        const val _ID = "id"

        const val COLUMN_NAME_TITLE = "title"

        const val COLUMN_NAME_YEAR = "year"

        const val COLUMN_NAME_RATED = "rated"

        const val COLUMN_NAME_RELEASED = "released"

        const val COLUMN_NAME_RUNTIME = "runtime"

        const val COLUMN_NAME_GENRE = "genre"

        const val COLUMN_NAME_DIRECTOR = "director"

        const val COLUMN_NAME_WRITER = "writer"

        const val COLUMN_NAME_ACTORS = "actor"

        const val COLUMN_NAME_PLOT = "plot"

        const val COLUMN_NAME_LANGUAGE = "language"

        const val COLUMN_NAME_COUNTRY = "country"

        const val COLUMN_NAME_AWARDS = "awards"

        const val COLUMN_NAME_POSTER = "poster"

        const val COLUMN_NAME_METASCORE = "metascore"
        const val COLUMN_NAME_IMDB_RATING = "rating"
        const val COLUMN_NAME_IMDB_VOTES = "imdb_votes"

        const val COLUMN_NAME_IMDBID = "imdb_id"
```

```
        const val COLUMN_NAME_TYPE = "type"


        const val COLUMN_NAME_SAVED = "saved"

    }
}
```

# My Database Creator

```
class MovieDBHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, null,
 DATABASE_VERSION) {

    // create the different database fields
    override fun onCreate(db: SQLiteDatabase)

    // This database is only a cache for online data, so its upgrade policy is
    // to simply to discard the data and start over
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int)

    // here is where I store the different values needed to create the database
    companion object {
        // If you change the database schema, you must increment the database version.
        const val DATABASE_VERSION = 1
        const val DATABASE_NAME = "movies2.db"
        private const val SQL_CREATE_ENTRIES =
                "CREATE TABLE ${MovieContract.MovieEntry.TABLE_NAME} (" +
                        "${MovieContract.MovieEntry.COLUMN_NAME_ACTORS} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_AWARDS} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_COUNTRY} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_DIRECTOR} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_GENRE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_IMDBID} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_LANGUAGE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_METASCORE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_IMDB_RATING} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_IMDB_VOTES} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_PLOT} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_POSTER} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_RATED} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_RELEASED} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_RUNTIME} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_SAVED} INTEGER," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_TITLE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_TYPE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_WRITER} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_YEAR} TEXT, " +
                        "ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL)"

        private const val SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS ${MovieContract.M
ovieEntry.TABLE_NAME}"
    }
}
```

## Database Controller

```
// I give the db controller my MovieDBHelper with the expectation that the database is
 already made
class MovieDBController(private val dbHelper: MovieDBHelper) {

    // Here I delete all cached movies
    fun deleteAllNotSaved()

    // Here is where I get all of the movies in the database. I also included options
for selecting specific movies.
    fun getAll(selection: String? = null, where: Array<String>? = null): List<Movie>

    // here is where I get the specific movies in the database. Includes options for w
here I get them from and a sorting option
    fun get(where: String? = null, sort: String? = null, values: Array<String>? = null
): Movie

    // returns a boolean checking if a movie is in the database
    private fun hasMovie(movie: Movie): Boolean

        // inserts a movie to the database
    private fun insert(movie: Movie): Long

    // a special insert function that inserts movies into a database
    // it also returns the movies in the database simultaneously
    // important because the list needs to reflect the search, and not all movies save
d
    // but also needs to return the saved versions if saved and in list of searched mo
vies
    fun insertAll(movies: List<Movie>): List<Movie>

// changes the boolean of a movie in database to be saved
    fun save(movie: Movie)
}
```

# Repository

- NetworkRequests
- DBController

# Repository

```
// I give my data repository both MovieDBController and NetworkRequests Objects
class MovieRepository(private var dbController: MovieDBController, private var network
Requests: NetworkRequests) {

    // calls my database controller function that returns a list of saved movies
    fun getSavedMovies(): List<Movie>

    // returns all of the movies in the database by calling a function in the database
 controller
    fun getMoviesAll(): List<Movie>

    // calls a functiom in the network request object that search's and returns a list
 of movies
    // adds these movies to the database using database controller
    fun addMovies(query: String): List<Movie>?

    // calls a function it the database to change a movie to saved
    fun saveMovie(movie: Movie)

    // deletes all movies in the database
    fun deleteAll()
}
```

# Models

- Movie
- SearchViewModel

## SearchViewModel

```
class SearchViewModel(application: Application) : AndroidViewModel(application) {
    // instantiates the repository and database, database controller, and the network
requests
    private val repository: MovieRepository = MovieRepository(MovieDBController(MovieD
BHelper(application.baseContext)), NetworkRequests(application.baseContext.filesDir))

    // a collection of mutable live data that reflect the different functions of the a
pp
    private val savedMutableLiveData = MutableLiveData<List<Movie>>()
    private val allMutableLiveData = MutableLiveData<List<Movie>>()
    private val selectedMutableLiveData = MutableLiveData<Movie>()

        // updates the mutable live data list of saved movies on a separate thread
        // returns a LiveData version of mutable live data
    fun getSavedMovies(): LiveData<List<Movie>>

    // returns a live data version of mutable list of all movies in a search request
    fun getAllMovies(): LiveData<List<Movie>>


    // searches for movies using repository on a separate thread
    // also updates the mutable live data list of all movies
    fun findMovies(title: String)

    // searches for movies using repository on a separate thread
    // also updates the mutable live data list of all movies
    fun findMovies(title: String)

    // saves the movie on a separate thread
    // updates mutable data list of saved movies
    fun saveMovie(movie: Movie)
}
```

# User Interfaces

- Main Activity
- Saved Movies Fragment
- Search Fragment
- Settings Fragment
- View Movie Fragment
- Card View

## Main Activity

- Logic

```
// checks and loads the specific fragment based that is pressed in the bottom navigati
ons bar
// loads the search fragment by default
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        // sets the default view
        supportFragmentManager.beginTransaction()
                    .replace(R.id.container, SearchFragment.newInstance())
                    .commitNow()

        bottom_navigation.setOnNavigationItemSelectedListener {
                show(Fragment)
        }
    }

    // shows a fragment given to it
    // returns true
    private fun show(fragment: Fragment): Boolean
}
```

- User Interface

```
        <com.google.android.material.bottomnavigation.BottomNavigationView
                        android:id="@+id/bottom_navigation"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:layout_gravity="bottom"
                        android:background="@color/colorPrimary"
                        app:itemIconTint="@drawable/bottom_navigation_colors"
                        app:itemTextColor="@drawable/bottom_navigation_colors"
                        app:menu="@menu/main_menu"
                        app:layout_constraintBottom_toBottomOf="parent"/>
        </androidx.coordinatorlayout.widget.CoordinatorLayout>
```


## Search Fragment

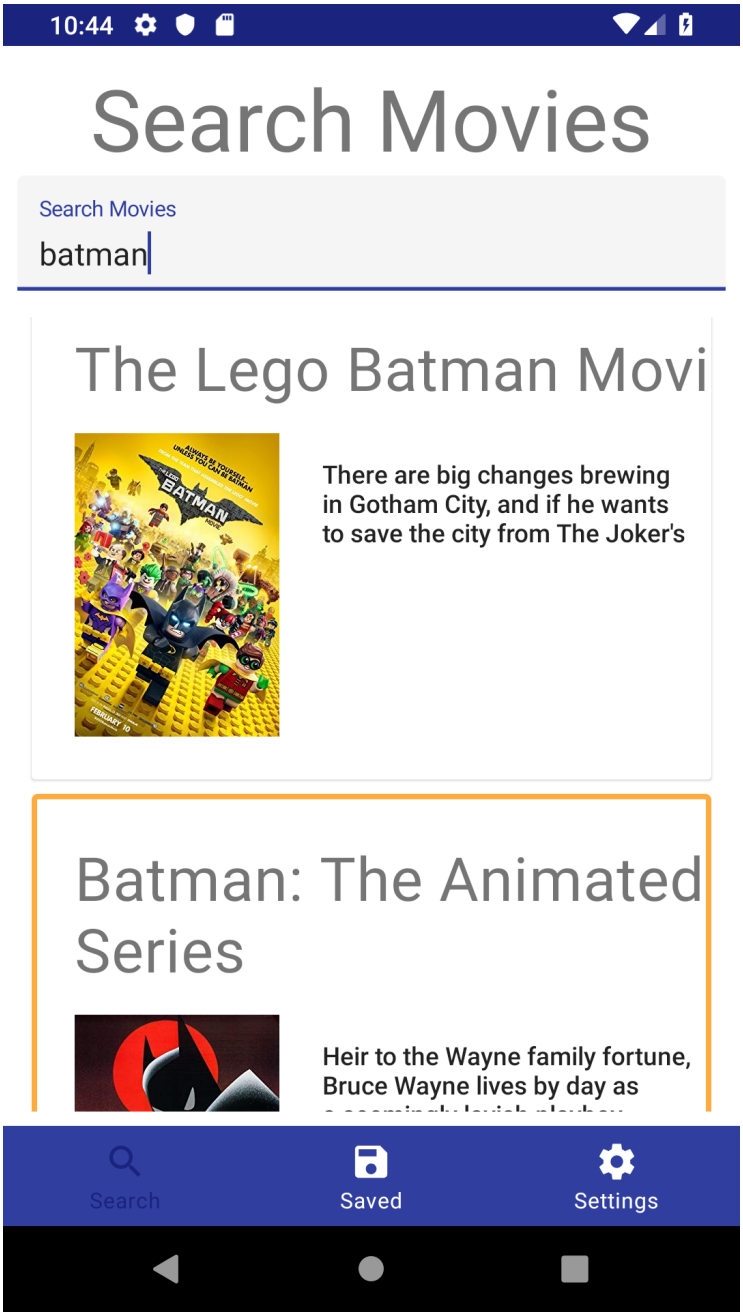
- Logic

```
// changes the attribute of src to use the setImageBitmap
@BindingMethods(value = [BindingMethod()])
// observes the search live data and updates a Recycler view
// saves the states of the search box
// addes a text changed listener to the text box
class SearchFragment : Fragment() {
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedIn
stanceState: Bundle?): View? {
        // updates list on change
        mViewModel.getAllMovies().observe(this, Observer {})

        // updates saved search text if changed
        binding.editText.text = SpannableStringBuilder(savedInstanceState.getString("s
earchText"))
```

```
        // adds a custom text watcher
        binding.editText.addTextChangedListener(mViewModel.watchText())
    }

    // saves text on changed state
    override fun onSaveInstanceState(outState: Bundle) {
    }

        // sets a binding adapter for imageUrl
        // sets an image to a uri and checks to make sure image uri isn't empty
        @BindingAdapter("imageUrl")
        fun setImageUrl(view: View, imageUri: String)

        // sets the background border of a saved movie in the recycler based on it's s
aved state
        @BindingAdapter("backgroundSaved")
        fun setBackgroundSaved(view: View, saved: Boolean)
}
```

- User Interface

```
    <!-- search box -->
     <com.google.android.material.textfield.TextInputEditText/>
    <!-- list of saved movies -->
    <androidx.recyclerview.widget.RecyclerView/>
```




## Saved Movies Fragment

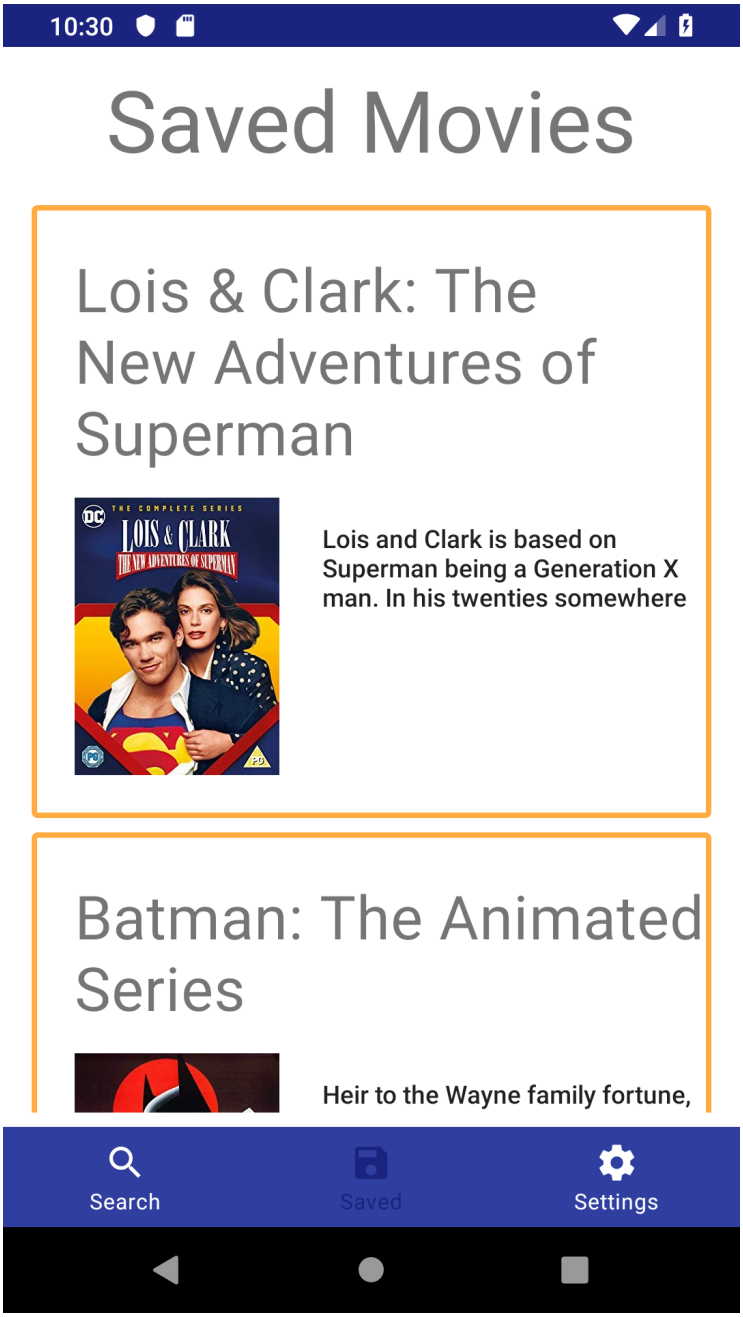
- Logic

```
/**
 * Shows a RecyclerView with a list of saved movies
*/
class SavedMovies : Fragment() {
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        mViewModel.getSavedMovies().observe(this, Observer {
            myDataset.clear()
            myDataset.addAll(it)
            myAdapter.notifyDataSetChanged()
```

```
        })
    }
}
```

- User Interface

```
<androidx.recyclerview.widget.RecyclerView      android:id="@+id/saveMovieList"/>
```




## Saved Movie Fragment

- Logic

```
/**
```

```
 * listens to check to see if a movie is selected
 * listens to see if someone saves a movie and saves the movie
 * binds a movie to the view
 */
class ViewMovieFragment : Fragment() {
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedIn
stanceState: Bundle?): View? {
        model.getSelectMovie().observe(this, Observer {
            binding.movie = it
            binding.savedCheckbox.setOnCheckedChangeListener { buttonView, isChecked -
>
                it.saved = isChecked
                model.saveMovie(it)
            }
        }
    }
}
```

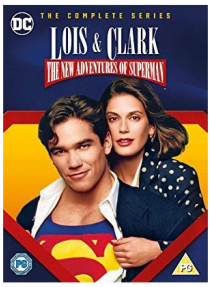
- User Interface

```
<layout>

    <data> <variable name="movie" type="Movie"/> </data>

    <ScrollView>
        <androidx.constraintlayout.widget.ConstraintLayout>
            <TextView android:text="@{movie.title, default=TITLE}" />

            <CheckBox android:checked="@{movie.saved}"/>

            <LinearLayout>
                <TextView android:text="@{movie.genre, default=GENRE}"/>
                <TextView android:text="@{movie.rated, default=RATED}"/>
                <TextView android:text="@{movie.type, default=TYPE}"/>
            </LinearLayout>

            <TextView android:text='@{movie.plot, default=PLOT}'/>
            <ImageView imageUrl="@{movie.poster}"/>

        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>


</layout>
```

# Movie

## Lois & Clark: The New Adventures of Superman

**Save Movie**

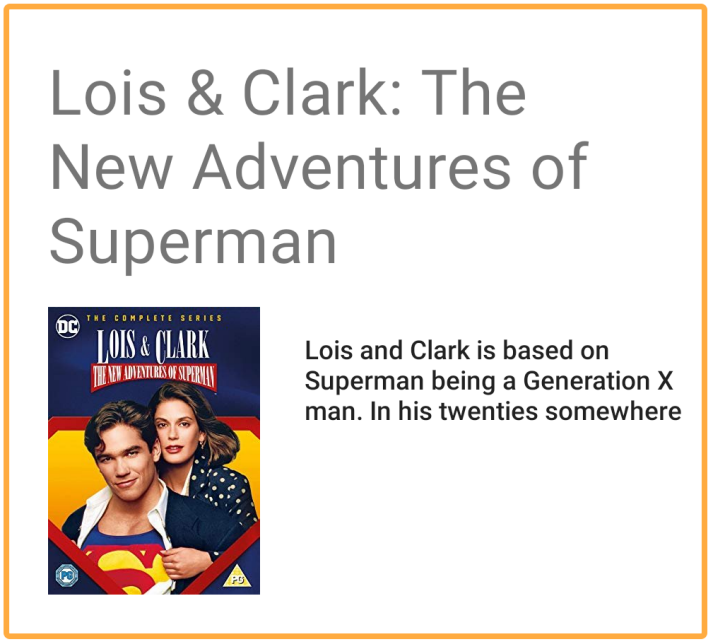



Adventure, Comedy, Drama TV-PG series

Lois and Clark is based on Superman being a Generation X man. In his twenties somewhere Clark must experience life as a pre-thirties pupil. Lois, as always, is by his side at the Daily Planet, adding that oh-so-ever romantic side to his life. The relationship between Lois and Clark, is as always, a platonic but on the edge of mad love, type of experience.

## My Custom Card View

```
<layout>
    <data>
        <variable name="movie"
                  type="edu.wccnet.searchappwithcaching.model.Movie"/>
    </data>
    <com.google.android.material.card.MaterialCardView>

        <androidx.constraintlayout.widget.ConstraintLayout>

            <ImageView imageUrl="@{movie.poster}"/>
            <TextView android:text="@{movie.title, default=TITLE}"/>

            <TextView android:text='@{movie.plot}'/>

        </androidx.constraintlayout.widget.ConstraintLayout>

    </com.google.android.material.card.MaterialCardView>
</layout>
```




# My Custom Movie List Adapter

- View Holder
- Custom Click Listener

```
// A custom movie list adapter that takes a dataset and a custom click listener
class MyMovieListAdapter(private val movieDataset: List<Movie>, private val customOnCl
```

```
ickListener: CustomOnClickListener) : androidx.recyclerview.widget.RecyclerView.Adapte
r<MyMovieListAdapter.MyViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        var layout: LayoutInflater = LayoutInflater.from(parent.context)
        var movieSearchBinding: MovieCardBinding = MovieCardBinding.inflate(layout, pa
rent, false)
        return MyViewHolder(movieSearchBinding)
    }

    override fun getItemCount(): Int {
        return movieDataset.size
    }

    override fun onBindViewHolder(p0: MyViewHolder, p1: Int) {
        p0.bind(movieDataset[p1])
    }

    // A custom view holder that binds that data of a movie and also gives it's contai
ner a custom click listener
    inner class MyViewHolder(var binding: MovieCardBinding) : androidx.recyclerview.wi
dget.RecyclerView.ViewHolder(binding.root) {
        fun bind(movie: Movie) {
            binding.movie = movie
            binding.root.setOnClickListener { customOnClickListener.onItemClick(movie)
 }
        }
    }

}
```

# Implementation Of Click Listener

```
// My implementation of a click listener that takes a fragment manager and a viewmodel
// Selects a movie and also shows the movie
class MyCustomOnClickListener(private val fragmentManager: FragmentManager, private va
l mViewModel: SearchViewModel) : CustomOnClickListener() {
    override fun onItemClick(movie: Movie) {
        mViewModel.selectMovie(movie)
        fragmentManager.beginTransaction().replace(R.id.container, ViewMovieFragment.n
ewInstance()).addToBackStack("MovieFragmentView").commit()
    }
}
```

## Settings Fragment

- Logic

```
/**
 * Deletes all movies and alerts user of deletion using Snackbar
 */
class SettingsFragment : Fragment() {
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        binding.clearData.setOnClickListener {
            mViewModel.deleteAllMovies()
            snackbar.show() // shows alert that user deleted all movies
        }
    }
}
```

- User Interface

```
 <!--simple Button-->
<com.google.android.material.button.MaterialButton/>
```


10:49
Settings
CLEAR ALL SAVED MOVIES
Search
Saved
Settings

# Technologies Used

- Fragments
- ViewModel
- Singleton Design
- LiveData
- Databinding
- Kotlin
- RecyclerView
- SnackBar
- Coordinator Layout
- Threads
- Callable
- Uri
- Threadable
- InputStreams
- BitmapFactory
- JSONObject
- Google Material Framework

# Future Roadmap

While this app is basic, it's a **solid** foundation enables me to start adding other features. A couple potential features could be a *watched section* that has movies that you've already watched, while there could also be a *want to watch section* featuring movies you want to watch. The search list could be delineated by whether it's a movie or show. The search area could also feature a random search button. Another potential possibility could be a general watching section that would be there mainly for tv shows.

The future of this app is bright. I've architectured it in such a way that adding incremental upgrades is a small matter. It means adding a couple methods here and there, modifying the XML, and testing the new features.

Here's a list of future development:
- Tablet Development
- Movie/TV Delineation
- Progress Animation
- Change Saved to Watching
- Create a Finished Movies/TV area
- Set which episode currently watching for TV
- More Async
- Add material animations

# Possible Revenue

I should be able to use *The Open Movie Database* freely for a commercial venture. I wouldn't want to use ads as revenue. This stems from my own experience with applications that used ads. I have long abandoned them. Instead, I'd want to sell either features through in-app purchases or the app it-self. If this application is successful, that would be my goal.

# Appendix

## MovieRepository

```
package edu.wccnet.searchappwithcaching.controller

import android.net.Uri
import android.util.Log
import edu.wccnet.searchappwithcaching.MovieDBController
import edu.wccnet.searchappwithcaching.model.Movie
import edu.wccnet.searchappwithcaching.network.NetworkRequests
import java.io.File
import java.util.concurrent.Executors
import kotlin.concurrent.thread

// I give my data repository both MovieDBController and NetworkRequests Objects
class MovieRepository(private var dbController: MovieDBController, private var network
Requests: NetworkRequests) {

    // calls my database controller function that returns a list of saved movies
    fun getSavedMovies(): List<Movie> {
        Log.e("MovieRepository", "getting saved db movies")
        return dbController.getAll("saved=?", arrayOf("1"))
    }

    // returns all of the movies in the database by calling a function in the database
 controller
    fun getMoviesAll(): List<Movie> {
        return dbController.getAll()
    }

    // calls a functiom in the network request object that search's and returns a list
 of movies
    // adds these movies to the database using database controller
    fun addMovies(query: String): List<Movie>? {
        dbController.deleteAllNotSaved()
        val future = Executors.newSingleThreadExecutor().submit(networkRequests.getMov
ies(query))
        var movies = future.get()
        Log.e("MovieRepository", "Getting request movies")

        movies.forEach { Log.e("MovieRepository", it.toString()) }
        if (movies != null)
            return dbController.insertAll(movies)
        else
            return ArrayList<Movie>()
    }
```

```
    // calls a function it the database to change a movie to saved
    fun saveMovie(movie: Movie) {
        dbController.save(movie)
    }

    // deletes all movies in the database
    fun deleteAll() {
        thread {
            dbController.getAll().forEach {
                if(Uri.parse(it.poster) != Uri.EMPTY)
                    File(Uri.decode(it.poster)).delete()
            }
            dbController.deleteAll()
        }


    }

}
```

# MovieContract

```
package edu.wccnet.searchappwithcaching

import android.provider.BaseColumns

object MovieContract {
    // Movie database entry labels represented as final values
    object MovieEntry : BaseColumns {
        const val TABLE_NAME = "movie"
        const val _ID = "id"

        const val COLUMN_NAME_TITLE = "title"

        const val COLUMN_NAME_YEAR = "year"

        const val COLUMN_NAME_RATED = "rated"

        const val COLUMN_NAME_RELEASED = "released"

        const val COLUMN_NAME_RUNTIME = "runtime"

        const val COLUMN_NAME_GENRE = "genre"

        const val COLUMN_NAME_DIRECTOR = "director"

        const val COLUMN_NAME_WRITER = "writer"

        const val COLUMN_NAME_ACTORS = "actor"
```

```
        const val COLUMN_NAME_PLOT = "plot"

        const val COLUMN_NAME_LANGUAGE = "language"

        const val COLUMN_NAME_COUNTRY = "country"

        const val COLUMN_NAME_AWARDS = "awards"

        const val COLUMN_NAME_POSTER = "poster"

        const val COLUMN_NAME_METASCORE = "metascore"
        const val COLUMN_NAME_IMDB_RATING = "rating"
        const val COLUMN_NAME_IMDB_VOTES = "imdb_votes"

        const val COLUMN_NAME_IMDBID = "imdb_id"

        const val COLUMN_NAME_TYPE = "type"


        const val COLUMN_NAME_SAVED = "saved"

    }
}
```

# MovieDBController

```
package edu.wccnet.searchappwithcaching

import android.content.ContentValues
import edu.wccnet.searchappwithcaching.model.Movie
import edu.wccnet.searchappwithcaching.ui.search.MovieDBHelper

// I give the db controller my MovieDBHelper with the expectation that everything is a
lready made
class MovieDBController(private val dbHelper: MovieDBHelper) {

    // Here I delete all unsaved movies. Used for caching
    fun deleteAllNotSaved() {
        val db = dbHelper.writableDatabase
        getAll("${MovieContract.MovieEntry.COLUMN_NAME_SAVED}=?", arrayOf("0"))
        db.delete(MovieContract.MovieEntry.TABLE_NAME, "${MovieContract.MovieEntry.COL
UMN_NAME_SAVED}=?", arrayOf("0"))
    }

    // here I delete all movies (including the saved movies). used to reset the saved
movie database
    fun deleteAll() {
        val db = dbHelper.writableDatabase
```

```
        db.delete(MovieContract.MovieEntry.TABLE_NAME, null, null)
    }


    // Here is where I get all of the movies in the database. I also included options
for selecting specific movies.
    fun getAll(selection: String? = null, where: Array<String>? = null): List<Movie> {
        val db = dbHelper.readableDatabase

        val cursor = db.query(
                MovieContract.MovieEntry.TABLE_NAME,   // The table to query
                null,             // The array of columns to return (pass null to get
all)
                selection,              // The columns for the WHERE clause
                where,          // The values for the WHERE clause
                null,                   // don't group the rows
                null,                   // don't filter by row groups
                null                // The sort order
        )
        val movies = ArrayList<Movie>()

        with(cursor) {
            while (moveToNext()) {
                movies.add(Movie(
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_TITLE)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_YEAR)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_RATED)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_RELEASED)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_RUNTIME)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_GENRE)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_DIRECTOR)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_WRITER)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_ACTORS)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_PLOT)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_LANGUAGE)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_COUNTRY)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_AWARDS)),
```

```
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_POSTER)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_METASCORE)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_IMDB_RATING)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_IMDB_VOTES)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_IMDBID)),
                        getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUM
N_NAME_TYPE)),
                        "",
                        (getInt(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_
NAME_SAVED)) == 1)
                ))

            }
        }
        return movies
    }

    // here is where I get the specific movies in the database. Includes options for w
here I get them from and a sorting option
    fun get(where: String? = null, sort: String? = null, values: Array<String>? = null
): Movie {
        val db = dbHelper.readableDatabase

        val cursor = db.query(
                MovieContract.MovieEntry.TABLE_NAME,   // The table to query
                null,             // The array of columns to return (pass null to get
all)
                where,              // The columns for the WHERE clause
                values,          // The values for the WHERE clause
                null,                   // don't group the rows
                null,                   // don't filter by row groups
                sort// The sort order
        )
        cursor.moveToFirst()
        with(cursor) {
            return Movie(
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_TITLE)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_YEAR)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_RATED)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_RELEASED)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
```

```
ME_RUNTIME)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_GENRE)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_DIRECTOR)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_WRITER)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_ACTORS)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_PLOT)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_LANGUAGE)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_COUNTRY)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_AWARDS)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_POSTER)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_METASCORE)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_IMDB_RATING)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_IMDB_VOTES)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_IMDBID)),
                    getString(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NA
ME_TYPE)),
                    "",
                    (getInt(getColumnIndexOrThrow(MovieContract.MovieEntry.COLUMN_NAME
_SAVED)) == 1)
            )
        }
    }


    // returns a boolean checking if a movie is in the database
    private fun hasMovie(movie: Movie): Boolean {
        val db = dbHelper.readableDatabase
        val cursor = db.query(
                MovieContract.MovieEntry.TABLE_NAME,   // The table to query
                null,             // The array of columns to return (pass null to get
all)
                "${MovieContract.MovieEntry.COLUMN_NAME_TITLE}=?",             // The
 columns for the WHERE clause
                arrayOf(movie.title),          // The values for the WHERE clause
                null,                   // don't group the rows
                null,                   // don't filter by row groups
                null               // The sort order
```

```
        )
        return cursor.moveToFirst()
    }

    // inserts a movie to the database
    private fun insert(movie: Movie): Long {
        // Gets the data repository in write mode
        val db = dbHelper.writableDatabase

        // Create a new map of values, where column names are the keys
        val m = ContentValues()
        m.put(MovieContract.MovieEntry.COLUMN_NAME_ACTORS, movie.actors)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_AWARDS, movie.awards)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_COUNTRY, movie.country)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_DIRECTOR, movie.director)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_GENRE, movie.genre)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_IMDBID, movie.imdbID)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_LANGUAGE, movie.language)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_METASCORE, movie.metascore)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_IMDB_RATING, movie.imdbRating)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_IMDB_VOTES, movie.imdbVotes)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_PLOT, movie.plot)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_POSTER, movie.poster)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_RATED, movie.rated)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_RELEASED, movie.released)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_RUNTIME, movie.runtime)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_SAVED, movie.saved)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_TITLE, movie.title)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_TYPE, movie.type)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_WRITER, movie.writer)
        m.put(MovieContract.MovieEntry.COLUMN_NAME_YEAR, movie.year)

        return db.insert(MovieContract.MovieEntry.TABLE_NAME, null, m)
    }

    // a special insert function that inserts movies into a database
    // it also returns the movies in the database simultaneously
    // important because the list needs to reflect the search, and not all movies save
d
    // but also needs to return the saved versions if saved and in list of searched mo
vies
    fun insertAll(movies: List<Movie>): List<Movie> {
        movies.forEach {
            if (!hasMovie(it))
                insert(it)
        }
        var databaseMoviesBasedOnSearch = mutableListOf<Movie>()
        movies.forEach {
            databaseMoviesBasedOnSearch.add(get("${MovieContract.MovieEntry.COLUMN_NAM
E_TITLE}=?", "${MovieContract.MovieEntry.COLUMN_NAME_SAVED}=?", arrayOf(it.title, "1")
```

```
))
            return@forEach
        }
        return databaseMoviesBasedOnSearch
    }

    // changes the boolean of a movie in database to be saved
    fun save(movie: Movie) {
        val db = dbHelper.writableDatabase

        val values = ContentValues().apply {
            put(MovieContract.MovieEntry.COLUMN_NAME_SAVED, movie.saved)
            put(MovieContract.MovieEntry.COLUMN_NAME_POSTER, movie.poster)
        }

        val selection = "${MovieContract.MovieEntry.COLUMN_NAME_TITLE} LIKE ?"
        val selectionArgs = arrayOf(movie.title)
        db.update(
                MovieContract.MovieEntry.TABLE_NAME,
                values,
                selection,
                selectionArgs)
    }
}
```

# MovieDBHelper

```
package edu.wccnet.searchappwithcaching.ui.search

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import edu.wccnet.searchappwithcaching.MovieContract

// My Database creator
class MovieDBHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, null,
 DATABASE_VERSION) {

    // create the different database fields
    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(SQL_CREATE_ENTRIES)
    }

    // This database is only a cache for online data, so its upgrade policy is
    // to simply to discard the data and start over
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL(SQL_DELETE_ENTRIES)
        onCreate(db)
    }
```

```
    override fun onDowngrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        onUpgrade(db, oldVersion, newVersion)
    }

    // here is where I store the different values needed to create the database
    companion object {
        // If you change the database schema, you must increment the database version.
        const val DATABASE_VERSION = 1
        const val DATABASE_NAME = "movies2.db"
        private const val SQL_CREATE_ENTRIES =
                "CREATE TABLE ${MovieContract.MovieEntry.TABLE_NAME} (" +
                        "${MovieContract.MovieEntry.COLUMN_NAME_ACTORS} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_AWARDS} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_COUNTRY} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_DIRECTOR} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_GENRE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_IMDBID} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_LANGUAGE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_METASCORE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_IMDB_RATING} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_IMDB_VOTES} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_PLOT} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_POSTER} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_RATED} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_RELEASED} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_RUNTIME} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_SAVED} INTEGER," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_TITLE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_TYPE} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_WRITER} TEXT," +
                        "${MovieContract.MovieEntry.COLUMN_NAME_YEAR} TEXT, " +
                        "ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL)"

        private const val SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS ${MovieContract.M
ovieEntry.TABLE_NAME}"
    }


}
```

# Movie

```
package edu.wccnet.searchappwithcaching.model


// My movie model. Includes most of the fields in OMDDBApi
data class Movie(
        var title: String,

        var year: String,

        var rated: String,

        var released: String,

        var runtime: String,

        var genre: String,

        var director: String,

        var writer: String,

        var actors: String,

        var plot: String,

        var language: String,

        var country: String,

        var awards: String,

        var poster: String,

        var metascore: String,
        var imdbRating: String,
        var imdbVotes: String,

        var imdbID: String,

        var type: String,

        var response: String,
        var saved: Boolean
)
```

# SearchViewModel

```
package edu.wccnet.searchappwithcaching.model

import android.app.Application
import android.text.Editable
import android.text.TextWatcher
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import edu.wccnet.searchappwithcaching.MovieDBController
import edu.wccnet.searchappwithcaching.controller.MovieRepository
import edu.wccnet.searchappwithcaching.network.NetworkRequests
import edu.wccnet.searchappwithcaching.ui.search.MovieDBHelper
import kotlin.concurrent.thread

class SearchViewModel(application: Application) : AndroidViewModel(application) {
    // instantiates the repository and database, database controller, and the network
requests
    private val repository: MovieRepository = MovieRepository(MovieDBController(MovieD
BHelper(application.baseContext)), NetworkRequests(application.baseContext.filesDir))

    // a collection of mutable live data that reflect the different functions of the a
pp
    private val savedMutableLiveData = MutableLiveData<List<Movie>>()
    private val allMutableLiveData = MutableLiveData<List<Movie>>()
    private val selectedMutableLiveData = MutableLiveData<Movie>()

    // updates the mutable live data list of saved movies on a separate thread
    // returns a LiveData version of mutable live data
    fun getSavedMovies(): LiveData<List<Movie>> {
        thread {
            savedMutableLiveData.postValue(repository.getSavedMovies())
        }
        return savedMutableLiveData
    }

    // returns a live data version of mutable list of all movies in a search request
    fun getAllMovies(): LiveData<List<Movie>> {
        return allMutableLiveData
    }

    // searches for movies using repository on a separate thread
    // also updates the mutable live data list of all movies
    fun findMovies(title: String) {
        thread {
            allMutableLiveData.postValue(repository.addMovies(title))
        }
    }

    // saves the movie on a separate thread
    // updates mutable data list of saved movies
```

```
    fun saveMovie(movie: Movie) {
        thread {
            repository.saveMovie(movie)
            savedMutableLiveData.postValue(repository.getSavedMovies())
        }
    }

    // updates the mutable live data of selected movies with the current movie
    // does this on a seperate thread
    fun selectMovie(movie: Movie) {
        thread {
            selectedMutableLiveData.postValue(movie)
        }
    }

    // returns the live data of the selected movie
    fun getSelectMovie(): LiveData<Movie> {
        return selectedMutableLiveData
    }

    // deletes all movies on a seperate thread
    fun deleteAllMovies() {
        thread { repository.deleteAll() }
    }

    // returns a custom TextWatcher
    fun watchText(): TextWatcher {
        return MovieTextWatcher()
    }

    // My Custom Text watch. Watches a text edit box and searches for a movie on text
change
    // possible future changes include overhead to check if word is legible
    private inner class MovieTextWatcher : TextWatcher {
        override fun afterTextChanged(s: Editable?) {
            if (s!!.isNotEmpty())
                findMovies(s.toString())
        }

        override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after
: Int) {
        }

        override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: I
nt) {
        }

    }
}
```

# JSONConverter

```
package edu.wccnet.searchappwithcaching.network

import edu.wccnet.searchappwithcaching.Converter
import edu.wccnet.searchappwithcaching.model.Movie
import org.json.JSONObject

class JSONConverter() {
    // given a omdb api json object, converts it to a movie object
    fun getMovie(c: JSONObject): Movie {
        return Movie(
                c.getString("Title"),

                c.getString("Year"),

                c.getString("Rated"),

                c.getString("Released"),

                c.getString("Runtime"),

                c.getString("Genre"),

                c.getString("Director"),

                c.getString("Writer"),

                c.getString("Actors"),

                c.getString("Plot"),

                c.getString("Language"),

                c.getString("Country"),

                c.getString("Awards"),

                c.getString("Poster"),

                c.getString("Metascore"),
                c.getString("imdbRating"),
                c.getString("imdbVotes"),

                c.getString("imdbID"),

                c.getString("Type"),

                c.getString("Response"),
```

```
                false)
    }

    // given a jsonarray of movie titles, converts them into a movie and returns a mov
ie list
    // converts them through a converter object that does the conversion of an individ
ual itself
    fun getMovieList(c: JSONObject, movieTitleConverter: Converter<String, Movie?>): L
ist<Movie> {
        val movies: MutableList<Movie> = mutableListOf()
        val searchResults = c.getJSONArray("Search")
        for (i in 0 until searchResults.length()) {
            val singleMovie = searchResults.getJSONObject(i)
            val title = singleMovie.getString("Title")
            when (movieTitleConverter.doForward(title) != null) {
                true -> movies += movieTitleConverter.doForward(title)!!
            }
        }
        return movies
    }
}
```

# NetworkRequests

```
package edu.wccnet.searchappwithcaching.network

import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.net.Uri
import android.util.Log
import edu.wccnet.searchappwithcaching.Converter
import edu.wccnet.searchappwithcaching.model.Movie
import org.json.JSONObject
import java.io.File
import java.io.FileOutputStream
import java.io.InputStreamReader
import java.net.URL
import java.util.concurrent.Callable

// does all network requests for getting a movie
// is given the private application directory
class NetworkRequests(private val filesDir: File) {
    // instantiates a json converter object
    private var jsonConverter: JSONConverter = JSONConverter()
    // a photo counter that gives a unique name to each poster on the device
    private var posterCounter = 0

    // given a url to an image, returns a bitmap of the of said image
    // uses input streams and BitmapFactory
```

```
    private fun getPhoto(url: String): Bitmap? {
        try {
            val connection = URL(url).openConnection()
            connection.connect()
            val `is` = connection.getInputStream()
            return BitmapFactory.decodeStream(`is`)
        } catch (e: Exception) {
            Log.e("NetworkRequests", e.toString())
            return null
        }

    }

    // given a bitmap, creates a copy of that bitmap on the device
    // uses the private directory of app on device
    // returns a Uri pointing to that image on device
    private fun addFile(bitmap: Bitmap): Uri {
        posterCounter+=1
        var fileName = "poster ${posterCounter}.jpg"
        val file = File(filesDir, fileName)
        FileOutputStream(file).use { bitmap.compress(Bitmap.CompressFormat.JPEG, 85, i
t) }
        return Uri.fromFile(file)
    }

    // creates an api request url for OMDb API.
    // stores my api key
    // uses Uri Builder
    // takes a map of different url options
    private fun movieRequestCreator(options: Map<String, String>): String {
        val builder = Uri.Builder().scheme("http").authority("www.omdbapi.com").append
Path("").appendQueryParameter("apikey", "75be55f8")
        for ((key, value) in options) {
            builder.appendQueryParameter(key, value)
        }
        return builder.toString()
    }

    // returns the contents of a url request.
    // uses URL to create an input stream
    // uses a buffer
    // uses InputStreamReader
    private fun getString(url: String): String {
        val connection = URL(url).openConnection()
        connection.connect()
        val bufferSize = 1024
        var buffer = CharArray(bufferSize)
        val out = StringBuilder()
        val inputStreamReader = InputStreamReader(connection.inputStream)
        while (true) {
```

```
            val rsz = inputStreamReader.read(buffer, 0, buffer.size)
            if (rsz < 0)
                break
            out.append(buffer, 0, rsz)
        }

        Log.e("NetworkRequests", out.toString())

        return out.toString()
    }

    // Returns an async callable object of List<Movie>
    // checks if response is false of the string
    // uses my request creator
    // uses jsonConverter. Gives my JsonConverter a special converter for mvoies
    // downloads a local version of the movies posters
    // checks to see if the poster is downloadable
    // converts the movie.posters url into a uri
    fun getMovies(query: String): Callable<List<Movie>> {
        return Callable {
            var obj = JSONObject(getString(movieRequestCreator(mapOf("s" to query))))
            if (obj.getString("Response") == "False") {
                return@Callable ArrayList<Movie>()
            }
            var movieList = jsonConverter.getMovieList(obj, TitleMovieConverter())
            movieList.forEach {
                val posterBitmap = getPhoto(it.poster)
                if (posterBitmap != null)
                    it.poster = addFile(posterBitmap).toString()
                else
                    it.poster = Uri.EMPTY.toString()
            }

            return@Callable movieList
        }
    }

    // special converter class
    // converts a string into a movie
    // uses a combination of JsonConverter and NetWorkRequests
    // also checks for whether the language of the movie is english
    // can also turn a movie object into a string
    inner class TitleMovieConverter : Converter<String, Movie?>() {
        override fun doForward(input: String): Movie? {
            val jsonObject = JSONObject(getString(movieRequestCreator(mapOf("t" to inp
ut, "plot" to "full"))))
            return when (jsonObject.getString("Response") == "False" || jsonObject.get
String("Language") != "English") {
                true -> null
                false -> jsonConverter.getMovie(jsonObject)
```

```
            }

        }

        override fun doBackward(input: Movie?) = input!!.title
    }


}
```

# MainActivity

```
package edu.wccnet.searchappwithcaching.ui.search

import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import edu.wccnet.searchappwithcaching.R
import kotlinx.android.synthetic.main.main_activity.*
import kotlinx.android.synthetic.main.main_activity.view.*


// checks and loads the specific fragment based that is pressed in the bottom navigati
ons bar
// loads the search fragment by default
class MainActivity : AppCompatActivity(), ViewMovieFragment.OnFragmentInteractionListe
ner, SavedMovies.OnFragmentInteractionListener, SettingsFragment.OnFragmentInteraction
Listener {
    override fun onFragmentInteraction(uri: Uri) {
        TODO("not implemented") //To change body of created functions use File | Setti
ngs | File Templates.
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)
        if (savedInstanceState == null) {
            supportFragmentManager.beginTransaction()
                    .replace(R.id.container, SearchFragment.newInstance())
                    .commitNow()
        }

        bottom_navigation.setOnNavigationItemSelectedListener {
            it.setChecked(true)
            when (it.itemId) {
                R.id.search -> showSearch()
                R.id.saved -> showSaved()
                R.id.settings -> showSettings()
```

```
                else -> false
            }
        }
    }

    // shows a fragment given to it
    // returns true
    private fun show(fragment: Fragment): Boolean {
        supportFragmentManager.beginTransaction().replace(R.id.container, fragment).co
mmitNow()
        return true
    }

    // calls show with a Search Fragment
    private fun showSearch(): Boolean {
        return show(SearchFragment.newInstance())
    }

    // calls show with a Saved Fragment
    private fun showSaved(): Boolean {
        return show(SavedMovies.newInstance())
    }

    // calls show with a Settings Fragment
    private fun showSettings(): Boolean {
        return show(SettingsFragment.newInstance())
    }
}
```

# SavedMoviesFragment

```
package edu.wccnet.searchappwithcaching.ui.search

import android.content.Context
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import androidx.recyclerview.widget.LinearLayoutManager
import edu.wccnet.searchappwithcaching.MyMovieListAdapter
import edu.wccnet.searchappwithcaching.R
import edu.wccnet.searchappwithcaching.databinding.FragmentSavedMoviesBinding
import edu.wccnet.searchappwithcaching.model.Movie
import edu.wccnet.searchappwithcaching.model.SearchViewModel
```

```
import edu.wccnet.searchappwithcaching.MyCustomOnClickListener
import androidx.databinding.adapters.TextViewBindingAdapter.setText
import android.widget.TextView
import androidx.core.content.ContextCompat
import androidx.databinding.BindingAdapter




// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

/**
 * Shows a RecyclerView with a list of saved movies
 * Activities that contain this fragment must implement the
 * [SavedMovies.OnFragmentInteractionListener] interface
 * to handle interaction events.
 * Use the [SavedMovies.newInstance] factory method to
 * create an instance of this fragment.
 *
 */
class SavedMovies : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null
    private var listener: OnFragmentInteractionListener? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let {
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }



    // listens for a saved movie changes and updates the recycler view object
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
        val binding: FragmentSavedMoviesBinding = DataBindingUtil.inflate(
                inflater, R.layout.fragment_saved_movies, container, false)
        val mViewModel = ViewModelProviders.of(activity!!).get(SearchViewModel(activit
y!!.application)::class.java)
        var myDataset = ArrayList<Movie>()
        var myAdapter = MyMovieListAdapter(myDataset, MyCustomOnClickListener(fragment
Manager!!, mViewModel))
```

```
        var myLayout = LinearLayoutManager(this.context)
        binding.saveMovieList.apply {
            layoutManager = myLayout
            adapter = myAdapter

        }
        mViewModel.getSavedMovies().observe(this, Observer {
            myDataset.clear()
            myDataset.addAll(it)
            myAdapter.notifyDataSetChanged()
        })
        return binding.root
    }

    // TODO: Rename method, update argument and hook method into UI event
    fun onButtonPressed(uri: Uri) {
        listener?.onFragmentInteraction(uri)
    }

    override fun onAttach(context: Context) {
        super.onAttach(context)
        if (context is OnFragmentInteractionListener) {
            listener = context
        } else {
            throw RuntimeException(context.toString() + " must implement OnFragmentInt
eractionListener")
        }
    }

    override fun onDetach() {
        super.onDetach()
        listener = null
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     *
     *
     * See the Android Training lesson [Communicating with Other Fragments]
     * (http://developer.android.com/training/basics/fragments/communicating.html)
     * for more information.
     */
    interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        fun onFragmentInteraction(uri: Uri)
    }
```

```
    companion object {
        /**
         * Use this factory method to create a new instance of
         * this fragment using the provided parameters.
         *
         * @param param1 Parameter 1.
         * @param param2 Parameter 2.
         * @return A new instance of fragment SavedMovies.
         */
        // TODO: Rename and change types and number of parameters
        @JvmStatic
        fun newInstance() = SavedMovies()
    }
}
```

# SearchFragment

```
package edu.wccnet.searchappwithcaching.ui.search

import android.net.Uri
import android.os.Bundle
import android.text.SpannableStringBuilder
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.cardview.widget.CardView
import androidx.databinding.BindingAdapter
import androidx.databinding.BindingMethod
import androidx.databinding.BindingMethods
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import androidx.recyclerview.widget.LinearLayoutManager
import edu.wccnet.searchappwithcaching.MyMovieListAdapter
import edu.wccnet.searchappwithcaching.R
import edu.wccnet.searchappwithcaching.databinding.SearchFragmentBinding
import edu.wccnet.searchappwithcaching.model.Movie
import edu.wccnet.searchappwithcaching.model.SearchViewModel
import edu.wccnet.searchappwithcaching.MyCustomOnClickListener

// changes the attribute of src to use the setImageBitmap
@BindingMethods(value = [BindingMethod(
        type = android.widget.ImageView::class,
        attribute = "android:src",
        method = "setImageBitmap"
)])
```

```
// observes the search live data and updates a Recycler view
// saves the states of the search box
// addes a text changed listener to the text box
class SearchFragment : Fragment(), ViewMovieFragment.OnFragmentInteractionListener {
    lateinit var binding: SearchFragmentBinding
    override fun onFragmentInteraction(uri: Uri) {
        TODO("not implemented") //To change body of created functions use File | Setti
ngs | File Templates.
    }


    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {

        binding = DataBindingUtil.inflate(
                inflater, R.layout.search_fragment, container, false)
        val mViewModel = ViewModelProviders.of(activity!!).get(SearchViewModel(activit
y!!.application)::class.java)

        var movieData = ArrayList<Movie>()
        var myAdapter = MyMovieListAdapter(movieData, MyCustomOnClickListener(fragment
Manager!!, mViewModel))
        var viewManager = LinearLayoutManager(this.context)
        binding.recycler.apply {
            layoutManager = viewManager
            adapter = myAdapter
        }
            mViewModel.getAllMovies().observe(this, Observer {
            movieData.clear()
            movieData.addAll(it)
            myAdapter.notifyDataSetChanged()

        })
        if (savedInstanceState != null) {
            binding.editText.text = SpannableStringBuilder(savedInstanceState.getStrin
g("searchText"))
        }
        binding.editText.addTextChangedListener(mViewModel.watchText())

        return binding.root
    }

    override fun onSaveInstanceState(outState: Bundle) {
        outState.putString("searchText", binding.editText.text.toString())
        super.onSaveInstanceState(outState)
    }


    override fun onActivityCreated(savedInstanceState: Bundle?) {
```

```
        super.onActivityCreated(savedInstanceState)
        // TODO: Use the ViewModel
    }

    companion object {
        fun newInstance() = SearchFragment()


        // sets a binding adapter for imageUrl
        // sets an image to a uri
        // checks to make sure that the image uri isn't empty
        @BindingAdapter("imageUrl")
        @JvmStatic
        fun setImageUrl(view: View, imageUri: String) {
            if (Uri.parse(imageUri) != Uri.EMPTY) {
                if (view is ImageView)
                    view.setImageURI(Uri.parse(imageUri))
            }
        }

        // sets the background of a saved movie in the recycler based on it's saved st
ate
        @BindingAdapter("backgroundSaved")
        @JvmStatic
        fun setBackgroundSaved(view: View, saved: Boolean) {
            if (saved && view is CardView) {
                 view.background = view.context.getDrawable(R.drawable.border_secondar
y)
            }
        }
    }
}
```

# SettingsFragment

```
package edu.wccnet.searchappwithcaching.ui.search

import android.content.Context
import android.net.Uri
import android.os.Bundle
import android.renderscript.ScriptGroup
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProviders
import edu.wccnet.searchappwithcaching.R
import edu.wccnet.searchappwithcaching.databinding.FragmentSettingsBinding
```

```
import edu.wccnet.searchappwithcaching.model.SearchViewModel
import com.google.android.material.snackbar.Snackbar



// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

/**
 * Deletes all movies and alerts user of deletion using Snackbar
 * Activities that contain this fragment must implement the
 * [SettingsFragment.OnFragmentInteractionListener] interface
 * to handle interaction events.
 * Use the [SettingsFragment.newInstance] factory method to
 * create an instance of this fragment.
 *
 */
class SettingsFragment : Fragment() {
    private var param1: String? = null
    private var param2: String? = null
    private var listener: OnFragmentInteractionListener? = null
    lateinit var binding: FragmentSettingsBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let {
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
        binding = DataBindingUtil.inflate(
                inflater, R.layout.fragment_settings, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val mViewModel = ViewModelProviders.of(activity!!).get(SearchViewModel(activit
y!!.application)::class.java)
        binding.clearData.setOnClickListener {
            mViewModel.deleteAllMovies()
            val snackbar = Snackbar.make(activity!!.findViewById(R.id.placeSnackbar),
"Deleted All Saved Movies", Snackbar.LENGTH_LONG)
            snackbar.setAnchorView(activity!!.findViewById<View>(R.id.bottom_navigatio
```

```
n))
            snackbar.show()
        }
    }

    // TODO: Rename method, update argument and hook method into UI event
    fun onButtonPressed(uri: Uri) {
        listener?.onFragmentInteraction(uri)
    }

    override fun onAttach(context: Context) {
        super.onAttach(context)
        if (context is OnFragmentInteractionListener) {
            listener = context
        } else {
            throw RuntimeException(context.toString() + " must implement OnFragmentInt
eractionListener")
        }
    }

    override fun onDetach() {
        super.onDetach()
        listener = null
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     *
     *
     * See the Android Training lesson [Communicating with Other Fragments]
     * (http://developer.android.com/training/basics/fragments/communicating.html)
     * for more information.
     */
    interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        fun onFragmentInteraction(uri: Uri)
    }

    companion object {
        /**
         * Use this factory method to create a new instance of
         * this fragment using the provided parameters.
         *
         * @param param1 Parameter 1.
         * @param param2 Parameter 2.
         * @return A new instance of fragment SettingsFragment.
         */
```

```
        // TODO: Rename and change types and number of parameters
        @JvmStatic
        fun newInstance() = SettingsFragment()
    }
}
```

# ViewMovieFragment

```
package edu.wccnet.searchappwithcaching.ui.search

import android.content.Context
import android.net.Uri
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import edu.wccnet.searchappwithcaching.R
import edu.wccnet.searchappwithcaching.databinding.FragmentViewMovieBinding
import edu.wccnet.searchappwithcaching.model.SearchViewModel


// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

/**
 * listens to check to see if a movie is selected
 * listens to see if someone saves a movie and saves the movie
 * Activities that contain this fragment must implement the
 * [ViewMovieFragment.OnFragmentInteractionListener] interface
 * to handle interaction events.
 * Use the [ViewMovieFragment.newInstance] factory method to
 * create an instance of this fragment.
 *
 */
class ViewMovieFragment : androidx.fragment.app.Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null
    private var listener: OnFragmentInteractionListener? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        arguments?.let {
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
        var root: View? = null
        val binding = DataBindingUtil.inflate<FragmentViewMovieBinding>(inflater, R.la
yout.fragment_view_movie, container, false)
        val model = ViewModelProviders.of(activity!!).get(SearchViewModel::class.java)

        model.getSelectMovie().observe(this, Observer {
            binding.movie = it
            binding.savedCheckbox.setOnCheckedChangeListener { buttonView, isChecked -
>
                Log.e("ViewModelFragment", isChecked.toString())
                it.saved = isChecked
                model.saveMovie(it)
            }
            root = binding.root
        })

        return binding.root
    }

    // TODO: Rename method, update argument and hook method into UI event
    fun onButtonPressed(uri: Uri) {
        listener?.onFragmentInteraction(uri)
    }

    override fun onAttach(context: Context) {
        super.onAttach(context)
        if (context is OnFragmentInteractionListener) {
            listener = context
        } else {
            throw RuntimeException(context.toString() + " must implement OnFragmentInt
eractionListener")
        }
    }

    override fun onDetach() {
        super.onDetach()
        listener = null
    }

    /**
     * This interface must be implemented by activities that contain this
```

```
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     *
     *
     * See the Android Training lesson [Communicating with Other Fragments]
     * (http://developer.android.com/training/basics/fragments/communicating.html)
     * for more information.
     */
    interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        fun onFragmentInteraction(uri: Uri)
    }

    companion object {
        /**
         * Use this factory method to create a new instance of
         * this fragment using the provided parameters.
         *
         * @param param1 Parameter 1.
         * @param param2 Parameter 2.
         * @return A new instance of fragment ViewMovieFragment.
         */
        // TODO: Rename and change types and number of parameters
        @JvmStatic
        fun newInstance() = ViewMovieFragment()
    }
}
```

# Converter

```
package edu.wccnet.searchappwithcaching

// abstract converter class that converts a T value to a V value and a V value to a T
value
abstract class Converter<T, V> {
    abstract fun doForward(input: T): V
    abstract fun doBackward(input: V): T
}
```

# CustomOnClickListener

```
package edu.wccnet.searchappwithcaching

import edu.wccnet.searchappwithcaching.model.Movie


// Abstract custom click listener that takes a movie
abstract class CustomOnClickListener {
    abstract fun onItemClick(movie: Movie)
}
```

# MyCustomOnClickListener

```
package edu.wccnet.searchappwithcaching

import androidx.fragment.app.FragmentManager
import edu.wccnet.searchappwithcaching.model.Movie
import edu.wccnet.searchappwithcaching.model.SearchViewModel
import edu.wccnet.searchappwithcaching.ui.search.ViewMovieFragment

// My implementation of a click listener that takes a fragment manager and a viewmodel
// Selects a movie and also shows the movie
class MyCustomOnClickListener(private val fragmentManager: FragmentManager, private va
l mViewModel: SearchViewModel) : CustomOnClickListener() {
    override fun onItemClick(movie: Movie) {
        mViewModel.selectMovie(movie)
        fragmentManager.beginTransaction().replace(R.id.container, ViewMovieFragment.n
ewInstance()).addToBackStack("MovieFragmentView").commit()
    }
}
```

# MyMovieListAdapter

```
package edu.wccnet.searchappwithcaching

import android.view.LayoutInflater
import android.view.ViewGroup
import edu.wccnet.searchappwithcaching.databinding.MovieCardBinding
import edu.wccnet.searchappwithcaching.model.Movie

// A custom movie list adapter that takes a dataset and a custom click listener
class MyMovieListAdapter(private val movieDataset: List<Movie>, private val customOnCl
ickListener: CustomOnClickListener) : androidx.recyclerview.widget.RecyclerView.Adapte
r<MyMovieListAdapter.MyViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        var layout: LayoutInflater = LayoutInflater.from(parent.context)
        var movieSearchBinding: MovieCardBinding = MovieCardBinding.inflate(layout, pa
rent, false)
        return MyViewHolder(movieSearchBinding)
    }

    override fun getItemCount(): Int {
        return movieDataset.size
    }

    override fun onBindViewHolder(p0: MyViewHolder, p1: Int) {

        p0.bind(movieDataset[p1])

    }

    // A custom view holder that binds that data of a movie and also gives it's contai
ner a custom click listener
    inner class MyViewHolder(var binding: MovieCardBinding) : androidx.recyclerview.wi
dget.RecyclerView.ViewHolder(binding.root) {
        fun bind(movie: Movie) {
            binding.movie = movie
            binding.executePendingBindings()
            binding.root.setOnClickListener { customOnClickListener.onItemClick(movie)
 }
        }
    }

}
```

# border_secondary

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <stroke android:width="3dp"  android:color="@color/colorAccent"/>
    <corners android:radius="2dp"/>
</shape>
```

# fragment_saved_movies

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schema
s.android.com/tools"
>
    <androidx.constraintlayout.widget.ConstraintLayout android:layout_width="match_par
ent" android:layout_height="match_parent">
        <androidx.appcompat.widget.AppCompatTextView android:layout_width="wrap_conten
t"
                                                     android:layout_height="wrap_conte
nt"
                                                     style="@style/TextAppearance.Mate
rialComponents.Headline3"
                                                     android:text="Saved Movies"
                                                     android:id="@+id/appCompatTextVie
w"
                                                     android:layout_marginTop="8dp"
                                                     app:layout_constraintTop_toTopOf=
"parent"
                                                     android:layout_marginStart="8dp"
                                                     app:layout_constraintStart_toStar
tOf="parent"
                                                     android:layout_marginEnd="8dp"
                                                     app:layout_constraintEnd_toEndOf=
"parent"
                                                     android:textAppearance="@android:
style/TextAppearance.Material.WindowTitle"/>
        <androidx.recyclerview.widget.RecyclerView
                android:layout_width="0dp"
                android:layout_height="0dp"
                app:layout_constraintEnd_toEndOf="parent"
                android:layout_marginEnd="8dp" app:layout_constraintStart_toStartOf="p
arent"
                android:layout_marginStart="8dp"
                android:id="@+id/saveMovieList"
                app:layout_constraintTop_toBottomOf="@+id/appCompatTextView" android:l
ayout_marginTop="8dp"
                android:layout_marginBottom="8dp" app:layout_constraintBottom_toBottom
Of="parent"
                app:layout_constraintHorizontal_bias="0.0" app:layout_constraintVertic
al_bias="0.508"/>
    </androidx.constraintlayout.widget.ConstraintLayout>

</layout>
```

# fragment_settings

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schema
s.android.com/tools"
        tools:context="edu.wccnet.searchappwithcaching.ui.search.SettingsFragment">

    <!--simple button-->

    <androidx.constraintlayout.widget.ConstraintLayout android:layout_width="match_par
ent"
                                                       android:layout_height="match_pa
rent">
        <com.google.android.material.button.MaterialButton android:layout_width="wrap_
content"
                                                           android:layout_height="wrap
_content"
                                                           android:id="@+id/clearData"
                                                           android:text="Clear All Sav
ed Movies"
                                                           android:layout_marginTop="1
6dp"
                                                           style="@style/Widget.Materi
alComponents.Button.OutlinedButton"
                                                           app:layout_constraintTop_to
BottomOf="@+id/appCompatTextView"
                                                           app:layout_constraintStart_
toStartOf="parent"
                                                           android:layout_marginStart=
"16dp"/>
        <androidx.appcompat.widget.AppCompatTextView android:layout_width="wrap_conten
t"
                                                     android:layout_height="wrap_conte
nt"
                                                     style="@style/TextAppearance.Mate
rialComponents.Headline3"
                                                     android:text="Settings"
                                                     android:id="@+id/appCompatTextVie
w"
                                                     android:layout_marginTop="8dp"
                                                     app:layout_constraintTop_toTopOf=
"parent"
                                                     android:layout_marginStart="8dp"
                                                     app:layout_constraintStart_toStar
tOf="parent"
                                                     android:layout_marginEnd="8dp"
                                                     app:layout_constraintEnd_toEndOf=
"parent"
                                                     android:textAppearance="@android:
style/TextAppearance.Material.WindowTitle"/>
    </androidx.constraintlayout.widget.ConstraintLayout>
```

```
</layout>
```

# fragment_view_movie

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schema
s.android.com/tools"
        tools:context="edu.wccnet.searchappwithcaching.ui.search.ViewMovieFragment">
    <data>
        <variable name="movie" type="edu.wccnet.searchappwithcaching.model.Movie"/>
    </data>

    <ScrollView android:layout_width="fill_parent" android:layout_height="fill_parent"
>
        <androidx.constraintlayout.widget.ConstraintLayout android:layout_width="match
_parent"
                                                           android:layout_height="wrap
_content">
            <TextView android:layout_width="0dp" android:layout_height="wrap_content"
                      android:text="@{movie.title, default=TITLE}"
                      style="@style/TextAppearance.MaterialComponents.Headline5"
                      app:layout_constraintStart_toStartOf="parent"
                      android:layout_marginStart="24dp"
                      android:id="@+id/title"
                      android:textAppearance="@android:style/TextAppearance.Material"
                      app:layout_constraintTop_toBottomOf="@id/Tlabel"
                      android:layout_marginEnd="24dp"
                      app:layout_constraintEnd_toEndOf="parent"
                      app:layout_constraintHorizontal_bias="1.0" android:layout_margin
Top="14dp"
                      android:layout_marginBottom="14dp" app:layout_constraintBottom_t
oTopOf="@+id/savedCheckbox"/>
            <TextView
                    android:text="Movie"
                    android:layout_width="wrap_content"
                    style="@style/TextAppearance.MaterialComponents.Headline2"
                    android:layout_height="wrap_content" android:id="@+id/Tlabel" andr
oid:layout_marginEnd="8dp"
                    app:layout_constraintEnd_toEndOf="parent" android:layout_marginSta
rt="8dp"
                    app:layout_constraintStart_toStartOf="parent"
                    android:layout_marginBottom="15dp"
            />
            <androidx.appcompat.widget.AppCompatCheckBox android:layout_width="wrap_co
ntent"
                                                         android:layout_height="wrap_c
ontent"
                                                         android:checked="@{movie.save
d}"
```

```
                                                    android:id="@+id/savedCheckbo
x"
                                                    app:layout_constraintStart_to
EndOf="@+id/saveLabel"
                                                    app:layout_constraintTop_toBo
ttomOf="@id/title"
                                                    android:layout_marginStart="1
4dp"
                                                    android:layout_marginTop="14d
p"/>
            <LinearLayout android:layout_width="match_parent" android:layout_height="w
rap_content"
                          android:id="@+id/details" app:layout_constraintTop_toBottomO
f="@id/imageView"
                          android:layout_marginTop="14dp" app:layout_constraintStart_t
oStartOf="parent"
                          android:layout_marginStart="24dp" app:layout_constraintEnd_t
oEndOf="parent"
                          android:layout_marginEnd="24dp">
                <TextView android:layout_width="wrap_content" android:layout_weight="1
"
                          android:layout_height="wrap_content" android:text="@{movie.g
enre, default=GENRE}"/>
                <TextView android:layout_weight="1" android:layout_width="wrap_content
"
                          android:layout_height="wrap_content" android:text="@{movie.r
ated, default=RATED}"/>
                <TextView android:layout_weight="1" android:layout_width="wrap_content
"
                          android:layout_height="wrap_content" android:text="@{movie.t
ype, default=TYPE}"/>
            </LinearLayout>

            <TextView
                    android:text='@{movie.plot, default=PLOT}'
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:id="@+id/plot"
                    app:layout_constraintStart_toStartOf="parent"
                    android:layout_marginStart="24dp"
                    style="@style/TextAppearance.MaterialComponents.Body1"
                    app:layout_constraintEnd_toEndOf="parent"
                    android:layout_marginEnd="24dp"
                    app:layout_constraintTop_toBottomOf="@id/details"
                    android:layout_marginTop="14dp"/>
            <TextView
                    android:text="Save Movie"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content" android:id="@+id/saveLabel"
                    style="@style/TextAppearance.MaterialComponents.Headline6"
```

```
                    app:layout_constraintTop_toBottomOf="@+id/title"
                    app:layout_constraintStart_toStartOf="parent" android:layout_margi
nStart="24dp"
                    android:layout_marginTop="14dp"/>
            <ImageView
                    imageUrl="@{movie.poster}"
                    android:src="@drawable/ic_save_black_24dp"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:id="@+id/imageView"
                    app:layout_constraintTop_toBottomOf="@+id/saveLabel"
                    app:layout_constraintStart_toStartOf="parent"
                    android:layout_marginStart="24dp" app:layout_constraintEnd_toEndOf
="parent"
                    android:layout_marginEnd="24dp" android:layout_marginTop="14dp"/>

        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>


</layout>
```

# main_activity

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".ui.search.MainActivity">
        <androidx.coordinatorlayout.widget.CoordinatorLayout android:layout_width="mat
ch_parent"
                                                             android:layout_height="0d
p" android:id="@+id/container"
                                                             app:layout_constraintTop_
toTopOf="parent"
                                                             app:layout_constraintBott
om_toTopOf="@id/placeSnackbar"
                                                             app:layout_constraintVert
ical_weight="8">

        </androidx.coordinatorlayout.widget.CoordinatorLayout>


        <androidx.coordinatorlayout.widget.CoordinatorLayout android:layout_width="mat
ch_parent"
```

```
                                                    android:layout_height="wr
ap_content"
                                                    android:id="@+id/placeSna
ckbar"
                                                    android:layout_marginBott
om="0dp"
                                                    app:layout_constraintBott
om_toBottomOf="parent"
                                                    app:layout_constraintTop_
toBottomOf="@id/container">
                <com.google.android.material.bottomnavigation.BottomNavigationView
                        android:id="@+id/bottom_navigation"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:layout_gravity="bottom"
                        android:background="@color/colorPrimary"
                        app:itemIconTint="@drawable/bottom_navigation_colors"
                        app:itemTextColor="@drawable/bottom_navigation_colors"
                        app:menu="@menu/main_menu"
                        app:layout_constraintBottom_toBottomOf="parent"/>
        </androidx.coordinatorlayout.widget.CoordinatorLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

# movie_card

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable name="movie"
                  type="edu.wccnet.searchappwithcaching.model.Movie"/>
    </data>
    <com.google.android.material.card.MaterialCardView
            style="@style/Widget.MaterialComponents.CardView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="@dimen/mtrl_card_spacing"
            backgroundSaved="@{movie.saved}"
            android:layout_marginTop="@dimen/mtrl_card_spacing"
            android:layout_marginRight="@dimen/mtrl_card_spacing">

        <androidx.constraintlayout.widget.ConstraintLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

            <ImageView
                    android:layout_width="0dp"
                    imageUrl="@{movie.poster}"
```

```
                    android:layout_height="wrap_content"
                    android:id="@+id/movieView"
                    android:adjustViewBounds="false"
                    android:cropToPadding="true"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@id/textView2"
                    android:layout_marginStart="24dp" android:layout_marginTop="14dp"
                    android:layout_marginBottom="24dp"
                    app:layout_constraintBottom_toBottomOf="parent"/>
            <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@{movie.title, default=TITLE}"
                    style="@style/TextAppearance.MaterialComponents.Headline4"
                    android:id="@+id/textView2"
                    android:layout_marginTop="24dp" app:layout_constraintTop_toTopOf="
parent"
                    app:layout_constraintStart_toStartOf="parent" android:layout_margi
nStart="24dp"/>

            <TextView
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:text='@{movie.plot, default="Mystery writer Cornelia Van G
order has rented a country house called The Oaks, which not long ago had been the scen
e of some murders committed by a strange and violent criminal known as The Bat. Meanwh
ile, the houses owner, bank president John Fleming, has recently embezzled one million
 dollars in securities, and has hidden the proceeds in the house, but he is killed bef
ore he can retrieve the money. Thus the lonely country house soon becomes the site of
many mysterious and dangerous activities."}'
                    style="@style/TextAppearance.MaterialComponents.Subtitle2"
                    app:layout_constraintTop_toBottomOf="@+id/textView2"
                    android:textAppearance="@android:style/TextAppearance.Material.Cap
tion"
                    android:layout_marginTop="27dp"
                    app:layout_constrainedWidth="true"
                    app:layout_constraintStart_toEndOf="@id/movieView"
                    android:layout_marginStart="24dp"
                    app:layout_constraintEnd_toEndOf="parent"
                    android:maxLines="3"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintHorizontal_bias="0.0"
                    app:layout_constraintVertical_bias="0.0"/>
        </androidx.constraintlayout.widget.ConstraintLayout>
        <!--<TextView android:layout_marginLeft="10dp" android:layout_width="wrap_cont
ent" android:layout_height="wrap_content" setBoolean="@{movie.saved, default=""}"/>-->

    </com.google.android.material.card.MaterialCardView>
</layout>
```

# search_fragment

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://s
chemas.android.com/apk/res-auto"
>
    <androidx.constraintlayout.widget.ConstraintLayout
            android:id="@+id/search"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
    >

        <TextView
                android:text="Search Movies"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:id="@+id/textView"
                android:textAppearance="@android:style/TextAppearance.Material.WindowT
itle"
                android:layout_marginEnd="8dp"
                style="@style/TextAppearance.MaterialComponents.Headline3"
                app:layout_constraintEnd_toEndOf="parent" android:layout_marginStart="
8dp"
                app:layout_constraintStart_toStartOf="parent" android:layout_marginTop
="8dp"
                app:layout_constraintTop_toTopOf="parent"/>
        <com.google.android.material.textfield.TextInputLayout
                android:layout_width="0dp"
                android:layout_height="wrap_content"

                style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox"
                android:hint="Search Movies"
                app:layout_constraintStart_toStartOf="parent"
                android:layout_marginStart="8dp" android:id="@+id/textInputLayout"
                app:layout_constraintTop_toBottomOf="@+id/textView" app:layout_constra
intEnd_toEndOf="parent"
                android:layout_marginEnd="8dp">

            <com.google.android.material.textfield.TextInputEditText
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content" android:id="@+id/editText"/>
        </com.google.android.material.textfield.TextInputLayout>
        <androidx.recyclerview.widget.RecyclerView
                android:layout_width="0dp"
                android:layout_height="0dp"
                app:layout_constraintEnd_toEndOf="parent"
                android:layout_marginEnd="8dp" app:layout_constraintStart_toStartOf="p
arent"
                android:layout_marginStart="8dp"
                android:id="@+id/recycler"
```

```
                app:layout_constraintTop_toBottomOf="@+id/textInputLayout" android:lay
out_marginTop="8dp"
                android:layout_marginBottom="8dp" app:layout_constraintBottom_toBottom
Of="parent"
                app:layout_constraintHorizontal_bias="0.0" app:layout_constraintVertic
al_bias="0.508"/>
    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```