DateConverter.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import androidx.room.TypeConverter
import java.util.Date

class DateConverter {
    @TypeConverter
    fun fromTimestamp(value: Long?): Date? {
        return if (value == null) null else Date(value)
    }

    @TypeConverter
    fun dateToTimestamp(date: Date): Long {
        return date.time
    }
}
```

DetailScreen.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.activityViewModels
import androidx.lifecycle.lifecycleScope
import com.bumptech.glide.Glide
import edu.vt.cs3714.retrofitrecyclerviewguide.databinding.FragmentDetailScreenBinding
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch

/**
  * DetailScreen fragment class
  */
class DetailScreen : Fragment() {
    private var binding: FragmentDetailScreenBinding? = null
    private val model: MovieViewModel by activityViewModels()
    var movie: MovieItem? = null
    private lateinit var movieDao: MovieItemDao

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = FragmentDetailScreenBinding.inflate(layoutInflater)
        movie = arguments?.getParcelable("movie_key")
        movieDao = MovieRoomDatabase.getDatabase(requireContext().applicationContext).movieDao()
        if (movie!!.liked) {
            binding?.likeButton?.text = "Liked"
        } else {
            binding?.likeButton?.text = "Unliked"
        }

        binding?.likeButton?.setOnClickListener {
            movie?.let {
                it.liked = !it.liked
            }

            // Update the database
            lifecycleScope.launch(Dispatchers.IO) {
                movieDao.updateLikeStatus(movie!!.id, movie!!.liked)
            }

            // Update UI should be on the Main thread
            lifecycleScope.launch(Dispatchers.Main) {
                if (movie!!.liked) {
                    binding?.likeButton?.text = "Liked"
                } else {
                    binding?.likeButton?.text = "Unliked"
                }
            }
        }
    }

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

        model.allMovies.observe(viewLifecycleOwner) { movies ->
            // Update the RecyclerView adapter here
            movies?.let {
                Log.d("title type", "${this.arguments?.getString("title")!!::class.simpleName}")
                Log.d("release date", "${this.arguments?.getString("release")}")
                val posterPath = this.arguments?.getString("poster_path")
                if (!posterPath.isNullOrEmpty()) {
                    Glide.with(this).load(resources.getString(R.string.picture_base_url) +
posterPath).into(binding?.detailScreenPoster!!)
                }
                binding?.detailScreenTitle?.text = this.arguments?.getString("title")
                binding?.detailScreenRelease?.text = "Release date: ${this.arguments?.getString("release_date")}"
                binding?.detailScreenOverview?.text = "Overview: ${this.arguments?.getString("overview")}"
            }
        }
        return binding?.root
    }
}
```

ListScreen.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.ImageView
import android.widget.SearchView
import android.widget.Spinner
import android.widget.TextView
import android.widget.Toast
import androidx.core.os.bundleOf
import androidx.fragment.app.activityViewModels
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.bumptech.glide.request.RequestOptions
import edu.vt.cs3714.retrofitrecyclerviewguide.databinding.FragmentListScreenBinding
import kotlinx.coroutines.*
import org.w3c.dom.Text
import java.text.SimpleDateFormat
import java.util.Locale

/**
 * ListScreen fragment class
 */
class ListScreen : Fragment(), AdapterView.OnItemSelectedListener {
    private val movies = ArrayList<MovieItem>()
//    private lateinit var job: Job
    private val apiKey by lazy {
        resources.getString(R.string.api_key)
    }
    private val retrofitService by lazy {
        RetrofitService.create(resources.getString(R.string.base_url))
    }
    private var binding: FragmentListScreenBinding? = null
    private val model: MovieViewModel by activityViewModels()
    var spinner: Spinner? = null
    val adapter = MovieListAdapter()

    fun movies() : ArrayList<MovieItem> {
        return movies
    }

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

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    binding = FragmentListScreenBinding.inflate(inflater, container, false)
    spinner = binding!!.sortOptions
    spinner!!.onItemSelectedListener = this@ListScreen
    val recyclerView = binding?.movieList
    recyclerView?.adapter = adapter
    recyclerView?.layoutManager = LinearLayoutManager(requireContext())

    model.allMovies.observe(viewLifecycleOwner) { movies ->
        // Update the RecyclerView adapter here
        movies?.let {
            adapter.setMovies(it)
        }
    }

    /*binding?.searchView?.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
        override fun onQueryTextSubmit(query: String?): Boolean {
            query?.let { searchTerm ->
                val filteredMovies = movies.filter { it.title.contains(searchTerm, ignoreCase = true) }
                adapter.setMovies(filteredMovies)
            }

            return true
        }

        override fun onQueryTextChange(newText: String?): Boolean {
            val moviesString = movies.joinToString(separator = "") { it.title }
            newText?.let { query ->
                val filteredMovies = movies.filter { it.title.contains(query, ignoreCase = true) }
                adapter.setMovies(filteredMovies)
                val filteredMoviesString = filteredMovies.joinToString(separator = "") { it.title }
                Log.d("filtered movies", "list: $filteredMoviesString")
            }
            return true
        }

    })*/

    binding?.refresh?.setOnClickListener{
        // Display a short toast message
        Toast.makeText(context, "Refresh clicked", Toast.LENGTH_SHORT).show()
        model.refreshMovies(1)
    }

    ArrayAdapter.createFromResource(
        requireContext(),
        R.array.sorting_options,    // Your spinner item layout
        android.R.layout.simple_spinner_item
    ).also { adapter ->
```

```
            adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
            spinner!!.adapter = adapter
        }

        Log.d("ListScreen", "Movies: $movies")
        return binding?.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        // Assuming you added a Switch in your layout with the id "switchFilterLiked"
        binding?.filter?.setOnCheckedChangeListener { _, isChecked ->
            if (isChecked) {
                // Remove observer from allMovies to avoid multiple observers
                model.allMovies.removeObservers(viewLifecycleOwner)

                model.likedMovies.observe(viewLifecycleOwner, { movies ->
                    // Assuming you have a method in your adapter called 'submitList' or 'updateList' to update the
dataset.
                    adapter.setMovies(movies)
                })
            } else {
                // Remove observer from likedMovies to avoid multiple observers
                model.likedMovies.removeObservers(viewLifecycleOwner)

                model.allMovies.observe(viewLifecycleOwner, { movies ->
                    // Assuming you have a method in your adapter called 'submitList' or 'updateList' to update the
dataset.
                    adapter.setMovies(movies)
                })
            }
        }
    }

    override fun onItemSelected(parent: AdapterView<*>?, view: View?, position: Int, id: Long) {
        val selectedItem = parent?.getItemAtPosition(position).toString()
        Toast.makeText(context, "Selected: $selectedItem", Toast.LENGTH_SHORT).show()

        // If you want to perform any action based on the selected item:
        when (selectedItem) {
            "Title" -> {
                // Sort movies by title or any other action you want
                movies.sortBy { it.title }
                binding?.movieList?.adapter?.notifyDataSetChanged()
            }
            "Rating" -> {
                // Sort movies by rating or any other action you want
                movies.sortBy { it.vote_average }
                binding?.movieList?.adapter?.notifyDataSetChanged()
            }
            // Add more conditions if needed
        }
```

```
}

override fun onNothingSelected(parent: AdapterView<*>?) {
    TODO("Not yet implemented")
}


/**
  * A RecyclerView adapter class. Provides the list of items to be displayed there.
  */
inner class MovieListAdapter : RecyclerView.Adapter<MovieListAdapter.MovieViewHolder>() {
    inner class MovieViewHolder(val view: View) : RecyclerView.ViewHolder(view) {
        fun bindItems(movieItem: MovieItem) {
            itemView.setOnClickListener {
                Log.d("retrofit_demo", "list tap ")
                // Convert the date to string
                val dateFormat = SimpleDateFormat("MMMM dd, yyyy", Locale.US)
                val dateString = dateFormat.format(movieItem.release_date)
                val bundle = bundleOf(
                    "poster_path" to movieItem.poster_path,
                    "title" to movieItem.title,
                    "release_date" to dateString,
                    "overview" to movieItem.overview
                )
                bundle.putParcelable("movie_key", movieItem)
                Log.d("release date type", "${(movieItem.release_date)::class.simpleName ?: "Unknown"}")
                binding?.root?.findNavController()?.navigate(R.id.action_listScreen_to_detailScreen, bundle)
            }
        }
    }

    internal fun setMovies(movies: List<MovieItem>) {
        this@ListScreen.movies.clear()
        this@ListScreen.movies.addAll(movies)
        notifyDataSetChanged()
    }

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

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MovieViewHolder {
        val v = LayoutInflater.from(parent.context).inflate(R.layout.card_view, parent, false)
        return MovieViewHolder(v)
    }

    override fun onBindViewHolder(holder: MovieViewHolder, position: Int) {
        Glide.with(this@ListScreen)
            .load(resources.getString(R.string.picture_base_url) + movies[position].poster_path)
            .apply(RequestOptions().override(128, 128))
            .into(holder.view.findViewById(R.id.poster))

        holder.view.findViewById<TextView>(R.id.title).text = movies[position].title
```

```
            holder.view.findViewById<TextView>(R.id.rating).text = movies[position].vote_average.toString()

            holder.bindItems(movies[position])
        }
    }
}
```

MainActivity.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import android.content.Context
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.inputmethod.InputMethodManager
import android.widget.Button
import android.widget.SearchView
import android.widget.TextView
import android.widget.Toast
import androidx.core.os.bundleOf
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.bumptech.glide.request.RequestOptions
import edu.vt.cs3714.retrofitrecyclerviewguide.R
import edu.vt.cs3714.retrofitrecyclerviewguide.databinding.ActivityMainBinding
import edu.vt.cs3714.retrofitrecyclerviewguide.databinding.FragmentListScreenBinding
import kotlinx.coroutines.*
import retrofit2.HttpException

class MainActivity : AppCompatActivity(){
    private var binding: ActivityMainBinding? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding?.root)
    }
}
```

MovieItem.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import android.os.Parcelable
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.android.parcel.Parcelize
import java.util.*

@Parcelize
@Entity(tableName = "movie_table")
data class MovieItem(
    @PrimaryKey @ColumnInfo(name = "id") var id: Long,
    @ColumnInfo(name = "vote_count") var vote_count: Long,
    @ColumnInfo(name = "vote_average") var vote_average: Float,
    @ColumnInfo(name = "title") var title: String,
    @ColumnInfo(name = "popularity") var popularity: Float,
    @ColumnInfo(name = "poster_path") var poster_path: String,
    @ColumnInfo(name = "overview") var overview: String,
    @ColumnInfo(name = "release_date") var release_date: Date,
    @ColumnInfo(name = "liked") var liked: Boolean = false
) : Parcelable

//Example JSON record from MovieDB
/*
*
{
    "adult": false,
    "backdrop_path": "/iGCHoFp4VUwMoolO2C2u12AgiKl.jpg",
    "belongs_to_collection": null,
    "budget": 0,
    "genres": [
    {
        "id": 18,
        "name": "Drama"
    }
    ],
    "homepage": null,
    "id": 515916,
    "imdb_id": "tt8254556",
    "original_language": "nl",
    "original_title": "Girl",
    "overview": "A 15-year-old girl, born in a boy's body, dreams of becoming a ballerina and will push her body to
its limits in order for her dream to succeed.",
    "popularity": 31.534,
    "poster_path": "/a6WKjZ1eHrKV8u1DYqjW4yUPuC.jpg",
    "release_date": "2018-09-27",
    "revenue": 0,
    "runtime": 105,
    "spoken_languages": [
    {
        "iso_639_1": "nl",
        "name": "Nederlands"
    },
    {
        "iso_639_1": "en",
        "name": "English"
    },
    {
        "iso_639_1": "fr",
        "name": "FranÃ§ais"
    }
    ],
    "status": "Released",
    "tagline": "",
    "title": "Girl",
    "video": false,
    "vote_average": 7.7,
    "vote_count": 140
}
* */
```

MovieItemDao.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import androidx.lifecycle.LiveData
import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query

@Dao
interface MovieItemDao {

    @Query("SELECT * FROM movie_table order BY release_date DESC")
    fun getAllMovies(): LiveData<List<MovieItem>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertMovie(movie: MovieItem)

    @Query("DELETE FROM movie_table")
    fun deleteAll()

    @Query("UPDATE movie_table SET liked = :status WHERE id = :movieId")
    suspend fun updateLikeStatus(movieId: Long, status: Boolean)

    @Query("SELECT * FROM movie_table WHERE liked = 1")
    fun getLikedMovies(): LiveData<List<MovieItem>>
}
```

MovieItemRepository.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import androidx.annotation.WorkerThread
import androidx.lifecycle.LiveData

class MovieItemRepository(private val movieDao: MovieItemDao) {
    val allMovies: LiveData<List<MovieItem>> = movieDao.getAllMovies()

    fun getLikedMovies(): LiveData<List<MovieItem>> {
        return movieDao.getLikedMovies()
    }

    @WorkerThread
    fun insert(movie: MovieItem) {


        movieDao.insertMovie(movie)
    }

    @WorkerThread
    fun deleteAll() {
        movieDao.deleteAll()
    }
}
```

MovieRoomDatabase.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import androidx.room.TypeConverters
import androidx.room.migration.Migration
import androidx.sqlite.db.SupportSQLiteDatabase

@Database(entities = [MovieItem::class], version = 2, exportSchema = false)
@TypeConverters(DateConverter::class)
abstract class MovieRoomDatabase: RoomDatabase(){
    abstract fun movieDao(): MovieItemDao

    companion object {
        @Volatile
        private var INSTANCE: MovieRoomDatabase? = null
        val MIGRATION_1_2: Migration = object : Migration(1, 2) {
            override fun migrate(database: SupportSQLiteDatabase) {
                // Code to run the necessary SQL for migration
                database.execSQL("ALTER TABLE movie_table ADD COLUMN liked INTEGER NOT NULL DEFAULT 0")
            }
        }

        fun getDatabase(
            context: Context
        ): MovieRoomDatabase {
            val tempInstance = INSTANCE

            if (tempInstance != null) {
                return tempInstance
            }

            return INSTANCE ?: synchronized(this) {

                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    MovieRoomDatabase::class.java,
                    "Movie_database"
                )
                    .addMigrations(MIGRATION_1_2)
                    .build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

Movies.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

data class Movies(
    val results: List<MovieItem>,
    val total_pages: Int,
    val page: Int
)
```

MovieViewModel.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import android.app.Application
import android.util.Log
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
import io.reactivex.android.schedulers.AndroidSchedulers
import io.reactivex.disposables.Disposable
import io.reactivex.schedulers.Schedulers
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.Job
import kotlinx.coroutines.launch
import kotlin.coroutines.CoroutineContext

class MovieViewModel (application : Application) : AndroidViewModel(application) {
    private val api_key = "899f0396f49b73de1f6663573c2c2d85"
    private val api_base_url = "https://api.themoviedb.org/3/"

    private var parentJob = Job()
    private val coroutineContext: CoroutineContext
        get() = parentJob + Dispatchers.Main
    private val scope = CoroutineScope(coroutineContext)


    private var disposable: Disposable? = null

    private val repository: MovieItemRepository
    val allMovies: LiveData<List<MovieItem>>
    val likedMovies: LiveData<List<MovieItem>>

    init {
        val moviesDao = MovieRoomDatabase.getDatabase(application).movieDao()

        repository = MovieItemRepository(moviesDao)
        allMovies = repository.allMovies
        likedMovies = repository.getLikedMovies()
    }

    /**
      *
      * @param page QUERY PARAMS: required input is 1 from "themoviedb.org"
      */
    fun refreshMovies(page: Int){
        disposable =
            RetrofitService.create(api_base_url).getNowPlaying(api_key,page).subscribeOn(
                Schedulers.io()).observeOn(
                AndroidSchedulers.mainThread()).subscribe(
                {result -> showResult(result)},
                {error -> showError(error)})
    }

    private fun showError(error: Throwable?) {
        Log.d("t04","Error:"+error?.toString())
    }

    private fun showResult(result: Movies?) {

        Log.d("T04","Page:"+result?.page+"Result:"+result?.results?.last()?.release_date+ " pages "+
result?.total_pages)
        deleteAll()

        result?.results?.forEach { movie ->
            insert(movie)
        }
    }

    private fun insert(movie: MovieItem) = scope.launch(Dispatchers.IO) {
        repository.insert(movie)
    }

    private fun deleteAll() = scope.launch (Dispatchers.IO){
        repository.deleteAll()
    }
}
```

RetrofitService.kt

```
package edu.vt.cs3714.retrofitrecyclerviewguide

import io.reactivex.Observable
import com.google.gson.GsonBuilder
import com.jakewharton.retrofit2.adapter.kotlin.coroutines.CoroutineCallAdapterFactory
import kotlinx.coroutines.Deferred
import retrofit2.Retrofit
import retrofit2.adapter.rxjava2.RxJava2CallAdapterFactory
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.http.GET
import retrofit2.http.Path
import retrofit2.http.Query

/**
 * The RetrofitService handles the API requests
 *
 */
interface    RetrofitService {
    @GET("movie/now_playing?language=en-US")
    fun getNowPlaying(@Query("api_key") api_key: String, @Query("page") page: Int ): Observable<Movies>


    companion object {
        fun create(baseUrl: String): RetrofitService {

            val retrofit =
Retrofit.Builder().addCallAdapterFactory(RxJava2CallAdapterFactory.create()).addConverterFactory(
                GsonConverterFactory.create(GsonBuilder().setDateFormat("yyyy-MM-dd'T'HH:mm:ss").create()))
                .baseUrl(baseUrl)
                .build()

            return retrofit.create(RetrofitService::class.java)
        }
    }
}
```

layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph"
        tools:layout_editor_absoluteX="1dp"
        tools:layout_editor_absoluteY="1dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>

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

layout/card_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/card_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/poster"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@mipmap/ic_launcher" />

        <TextView
            android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="."
            android:textAppearance="@style/TextAppearance.AppCompat.Large"
            app:layout_constraintStart_toEndOf="@+id/poster"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/rating"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="."
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toEndOf="@+id/poster" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.cardview.widget.CardView>
```

layout/fragment_detail_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/detail_screen_poster"
        android:layout_width="match_parent"
        android:layout_height="350dp"
        android:paddingBottom="5dp"
        app:layout_constraintBottom_toTopOf="@id/detail_screen_title"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/detail_screen_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="Movie B"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:textSize="30sp"
        app:layout_constraintBottom_toTopOf="@id/detail_screen_release"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/detail_screen_poster" />

    <TextView
        android:id="@+id/detail_screen_release"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingStart="50dp"
        android:paddingEnd="50dp"
        android:paddingBottom="15dp"
        android:text="Release date: May 25, 2017"
        android:textSize="18sp"
        app:layout_constraintBottom_toTopOf="@id/detail_screen_overview"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/detail_screen_title" />

    <TextView
```

```
        android:id="@+id/detail_screen_overview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingStart="50dp"
        android:paddingEnd="50dp"
        android:paddingBottom="15dp"
        android:text="Overview: A movie about a black sun...."
        android:textSize="18sp"
        app:layout_constraintBottom_toTopOf="@id/like_button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/detail_screen_release" />

    <Button
        android:id="@+id/like_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Unliked"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/detail_screen_overview" />


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

layout/fragment_list_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!--    <SearchView-->
    <!--        android:id="@+id/search_view"-->
    <!--        android:layout_width="0dp"-->
    <!--        android:layout_height="0dp"-->
    <!--        app:layout_constraintBottom_toTopOf="@id/movie_list"-->
    <!--        app:layout_constraintEnd_toStartOf="@id/refresh"-->
    <!--        app:layout_constraintStart_toStartOf="parent"-->
    <!--        app:layout_constraintTop_toTopOf="parent"-->
    <!--        app:showAsAction="collapseActionView|ifRoom"-->
    <!--        tools:layout_editor_absoluteX="61dp"-->
    <!--        tools:layout_editor_absoluteY="1dp" />-->

    <Button
        android:id="@+id/refresh"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawableStart="@drawable/ic_refresh"
        android:text="Refresh"
        app:layout_constraintBottom_toTopOf="@+id/movie_list"
        app:layout_constraintEnd_toStartOf="@id/filter"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Switch
        android:id="@+id/filter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Filter"
        app:layout_constraintBottom_toTopOf="@+id/movie_list"
        app:layout_constraintEnd_toStartOf="@id/sort_by"
        app:layout_constraintStart_toEndOf="@id/refresh"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/sort_by"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sort by: "
        android:textColor="@color/black"
        android:textSize="17sp"
        app:layout_constraintBottom_toTopOf="@+id/movie_list"
        app:layout_constraintEnd_toStartOf="@id/sort_options"
        app:layout_constraintStart_toEndOf="@id/filter"
        app:layout_constraintTop_toTopOf="parent" />

    <Spinner
        android:id="@+id/sort_options"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/movie_list"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@id/sort_by"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/movie_list"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/refresh" />

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

navigation/nav_graph.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/listScreen">

    <fragment
        android:id="@+id/listScreen"
        android:name="edu.vt.cs3714.retrofitrecyclerviewguide.ListScreen"
        android:label="fragment_list_screen"
        tools:layout="@layout/fragment_list_screen" >
        <action
            android:id="@+id/action_listScreen_to_detailScreen"
            app:destination="@id/detailScreen" />
    </fragment>
    <fragment
        android:id="@+id/detailScreen"
        android:name="edu.vt.cs3714.retrofitrecyclerviewguide.DetailScreen"
        android:label="fragment_detail_screen"
        tools:layout="@layout/fragment_detail_screen" />
</navigation>
```

values/strings.xml

```
<resources>
    <string name="app_name">Movie Database</string>
    <string name="base_url">https://api.themoviedb.org/3/</string>
    <string name="picture_base_url">http://image.tmdb.org/t/p/w500/</string>
    <string name="api_key">EnterYourAPIKeyHere</string>
    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
    <string-array name="sorting_options">
        <item>Title</item>
        <item>Rating</item>
    </string-array>
</resources>
```