**Movie Tracker App**

Uses firebase as backend for data service in a CRUD application that allows addition of Movies that has genre and rating characteristics. Keep note, this app may not be optimized, It is better to separate components (edit-movie, new-movie, home) further into services (review world of heroes/to do list to confirm)

**General files including firebase service**

**Index.html**

Only used to display <head> tag info, which is just essentially browser tab name and icon.

<body> initializes the <app-root> which is the app.component.ts. It initializes the app.component.html and it’s styling.

**Styles.css**

This is where you can import the bootstrap.css and @angular/material themes. Global style. Also adds a line for background color.

**Package.json**

Stores general project descriptors, console terminal script starts like npm start, dependencies & dev dependencies along with their version.

**Environment.ts**

Where firebase credentials are added

**App.module.ts**

Where all modules are imported and defined including, but not limited to: browser, Custom\_elements\_schema, forms, router and routes, all user generated components auto added, firebase modules & environment, browser animations and tools like buttons/dialogs/slider.

Declarations: user generated components, not included in this app are directives and pipes

Imports: modules

Providers: services, once included is accessible throughout entire app

Bootstrap: auto generated to AppComponent

Schemas: added custom\_elements\_schema for this project

Exports class file name, AppModule

**App.component.html**

In this project it is used to separate the side bar and main area of the web app. Initializes <router-outlet> which is essentially the app.route.ts file, acts as a placeholder that Angular dynamically fills based on the current router state. Since first page does not have any routes in URL, the app.route.ts tells it to load the home component in the area of the webpage. Side bar is a function of routes using routerLink.

**App.component.ts**

Nothing is generally put in here. I believe it just used to initiate app. Html and scss files.

**App.routes.ts**

Import all components that you want to open a new ‘page’. In the export, you define an array of routes for each component and the name of it’s URL path. You can link buttons and such using routerlink within the html file of the preceding dependency.

**Firebase.service.ts**

This fine defines a series of functions to interact with the firebase database. Figure out what public vs private means in constructor tag. Functions listed are create, get movie, get movies (whole list), update movie, delete movie, search movies, search movies by rating.

See file for what functions are used and where they are called.

**Home component**

**Home.component.html**

Used to present view overall app view. We stack the search by name, filter by rating, and movie list on top of each other. Functions used in this file are referenced in the home.component.ts file.

First input area is the search. Within the div class input-group, you define an input area with form control. Using [(ngModel)] you two-way bind what is entered with a property value in the TS file that is used in a function to first convert what is entered to all lower case then searches the firebase. Searches are based on each key as shown in the (keyup) tag. Search button at end of input form is just there for appearance.

Second area is the filter by rating. Uses the mat slider which one way binds the value to ratingValue which is not really needed for what we are using It for but good practice for filtering. Executes rangeChange() by listening to where you are moving the slider. (change) basically says for any change in the slider run the rangeChange() for that value. Matslider uses hammerJS and gesture configs, make sure to import into app.module.ts and polyfills.ts

Third area displays the current movie list in firebase. Using ngfor you can display all items within database. Specific databinding is used for firebase documents

**Home.component.ts**

Used to define the HomeComponent within modules and routing. Defines variables used in functions below and defines it’s data types and initial values if necessary (define within export and before constructor). Constructor initializes firebase and router services. ngOnInIt runs first function getData to go to firebase, stores value into object instance result

See file for full function descriptions. Functions include: getting data to store into object instance, router navigation to view details, searching by name, searching by rating (ratingchange), and a function that combines both name and rating search.

**New Movie component**

**New-movie.component.html**

Just displays add a new movie page. Allows 3 form groups for name, genre, and rating. Also includes constant check for errors or incomplete forms and responds back with “form-name is required.” Also includes a dirty check for all forms, If one form is incorrect or incomplete ADD button will not be enabled for use.

**New-movie.component.ts**

Declares properties and values for validation message object (before constructor). Validator is a module as well BTW. Defines basis for creating the form, submitting the form, and resetting the form after submission. LEARN ANGULAR FORMS.

**Edit Movie component**

**Edit-movie.component.html**

Similar to new-movie, basically the same thing, just slightly modified to be used in edit-movie.

**edit-movie.component.ts**

Similar to new movie but ngOnInIt does not create a blank form, instead it pulls and subscribes it to the newly created form. Button on home page routes to the specific movie’s edit component. See file for functions it performs, should be pretty easy to understand.

**Edit-movie.resolver.ts**

Helps grab files before a the component is loaded to display and be used. Need to read more, basically a service.

**Adding Genre Drop Down Menu**

Needed to input MatSelectModule in app.module.ts first, may need to import into components too but not sure…. Seems like you don’t right now. Validation message probably not necessary (it’s wrong for mat select I think).