Frontend Code Listing

package.json –

```json
{
  "name": "frontend",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^15.2.0",
    "@angular/common": "^15.2.0",
    "@angular/compiler": "^15.2.0",
    "@angular/core": "^15.2.0",
    "@angular/forms": "^15.2.0",
    "@angular/platform-browser": "^15.2.0",
    "@angular/platform-browser-dynamic": "^15.2.0",
    "@angular/router": "^15.2.0",
    "@auth0/auth0-angular": "^2.2.1",
    "axios": "^1.6.2",
    "bootstrap": "^5.3.2",
    "font-awesome": "^4.7.0",
    "popper": "^1.0.1",
    "rxjs": "~7.8.0",
    "tslib": "^2.3.0",
    "wikipedia": "^2.1.1",
    "zone.js": "~0.12.0"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^15.2.10",
    "@angular/cli": "~15.2.10",
    "@angular/compiler-cli": "^15.2.0",
    "@types/jasmine": "~4.3.0",
    "@types/node": "^20.10.3",
    "jasmine-core": "~4.5.0",
    "karma": "~6.4.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage": "~2.2.0",
    "karma-jasmine": "~5.1.0",
    "karma-jasmine-html-reporter": "~2.0.0",
    "typescript": "~4.9.4"
  }
}
```

angular.json –

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "frontend": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/frontend",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": [
              "zone.js"
            ],
            "tsConfig": "tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css",
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "node_modules/font-awesome/css/font-awesome.css"
            ],
            "scripts": [
              "node_modules/bootstrap/dist/js/bootstrap.bundle.min.js"
            ]
          },
          "configurations": {
            "production": {
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "500kb",
                  "maximumError": "1mb"
                },
                {
                  "type": "anyComponentStyle",
```

```json
                  "maximumWarning": "2kb",
                  "maximumError": "4kb"
                }
              ],
              "outputHashing": "all"
            },
            "development": {
              "buildOptimizer": false,
              "optimization": false,
              "vendorChunk": true,
              "extractLicenses": false,
              "sourceMap": true,
              "namedChunks": true
            }
          },
          "defaultConfiguration": "production"
        },
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "configurations": {
            "production": {
              "browserTarget": "frontend:build:production"
            },
            "development": {
              "browserTarget": "frontend:build:development"
            }
          },
          "defaultConfiguration": "development"
        },
        "extract-i18n": {
          "builder": "@angular-devkit/build-angular:extract-i18n",
          "options": {
            "browserTarget": "frontend:build"
          }
        },
        "test": {
          "builder": "@angular-devkit/build-angular:karma",
          "options": {
            "polyfills": [
              "zone.js",
              "zone.js/testing"
            ],
            "tsConfig": "tsconfig.spec.json",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css"
            ],
            "scripts": []
          }
        }
      }
    }
  }
}
```

styles.css –

```css
body {
    background-color: #E3E2DF !important;
    color: black !important;
}

.btn-primary, .btn-primary:hover, .btn-primary:active, .btn-primary:visited {
    background-color: #9A1750 !important;
    border-color: #9A1750 !important;
}

.movie-content {
    border-radius: 5px;
}
```

main.ts –

```typescript
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';


platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

index.html –

```html
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>CinePal</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>

<body>
  <app-root></app-root>
</body>

</html>
```

web.service.ts –

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import axios from 'axios';
import { Buffer } from 'buffer'
import { ActorService } from './actor.service';
import { MovieService } from './movie.service';
import { UserService } from './user.service';

interface LoginResponse {
    token: string;
}

@Injectable()
export class WebService {
    movieService: MovieService;
    actorService: ActorService;
    userService: UserService;

    constructor(public http: HttpClient) {
        this.movieService = new MovieService(http);
        this.actorService = new ActorService(http);
        this.userService = new UserService(http);
    }


    login(username: string, password: string) {
        const auth = 'Basic ' + Buffer.from(username + ':' +
password).toString('base64');
        return
axios.get<LoginResponse>('http://localhost:5000/api/v1.0/login', { headers: {
'Authorization': auth } })
    }
}
```

user.service.ts –

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { User } from './types/user.type';
import { UrlResponse } from './types/urlResponse.type';

export class UserService {
    token = sessionStorage.getItem("token") || '';

    constructor(public http: HttpClient) { }

    getUsers(pageNumber: number): Observable<User[]> {
        return
this.http.get<User[]>(`http://localhost:5000/api/v1.0/users?pn=${pageNumber}`)
    }

    getUserByUsername(username: string): Observable<User[]> {
        return
this.http.get<User[]>(`http://localhost:5000/api/v1.0/users?username=${username}`)
    }

    getUserByName(name: string): Observable<User[]> {
        return
this.http.get<User[]>(`http://localhost:5000/api/v1.0/users?name=${name}`)
    }

    addNewUser(username: string, name: string, password: string, email: string,
admin: boolean): Observable<UrlResponse> {
        const formData = new FormData();

        formData.append("username", username);
        formData.append("name", name);
        formData.append("password", password);
        formData.append("email", email);
        formData.append("admin", admin.toString());
        formData.append("favourites", [].toString());
        formData.append("watchlist", [].toString());

        return this.http.post<UrlResponse>(`http://localhost:5000/api/v1.0/users`,
formData)
    }

    editExistingUser(userId: string, username: string, name: string, password:
string, email: string, admin: boolean, favourites: Array<string>, watchlist:
Array<string>, following: Array<string>): Observable<UrlResponse> {
        const formData = new FormData();

        formData.append("username", username);
        formData.append("name", name);
        formData.append("password", password);
        formData.append("email", email);
        formData.append("admin", admin.toString());
        formData.append("favourites", favourites.toString());
        formData.append("watchlist", watchlist.toString());
        formData.append("following", following.toString());

        return
this.http.put<UrlResponse>(`http://localhost:5000/api/v1.0/users/${userId}`,
formData, { headers: { 'x-access-token': this.token } })
    }

    deleteUser(id: string): Observable<{}> {
        return
this.http.delete<{}>(`http://localhost:5000/api/v1.0/users/${id}`, { headers:
{ 'x-access-token': this.token } })
    }

    addToFavourites(userId: string, movieId: string): Observable<UrlResponse>
{
        const formData = new FormData();
        formData.append("movie_id", movieId);

        return
this.http.put<UrlResponse>(`http://localhost:5000/api/v1.0/users/${userId}/fav
ourites`, formData, { headers: { 'x-access-token': this.token } })
    }

    addToWatchlist(userId: string, movieId: string): Observable<UrlResponse> {
        const formData = new FormData();
        formData.append("movie_id", movieId);

        return
this.http.put<UrlResponse>(`http://localhost:5000/api/v1.0/users/${userId}/wat
chlist`, formData, { headers: { 'x-access-token': this.token } })
    }

    removeFromFavourites(userId: string, movieId: string):
Observable<UrlResponse> {
        return
this.http.delete<UrlResponse>(`http://localhost:5000/api/v1.0/users/${userId}/
favourites/${movieId}`, { headers: { 'x-access-token': this.token } })
    }
    removeFromWatchlist(userId: string, movieId: string):
Observable<UrlResponse> {
        return
this.http.delete<UrlResponse>(`http://localhost:5000/api/v1.0/users/${userId}/
watchlist/${movieId}`, { headers: { 'x-access-token': this.token } })
    }

}
```

movie.service.ts –

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Movie } from './types/movie.type';
import { Review } from './types/review.type';
import { UrlResponse } from './types/urlResponse.type';

export class MovieService {
    token = sessionStorage.getItem("token") || '';

    constructor(public http: HttpClient) { }

    getMovies(page_number: number): Observable<Movie[]> {
        return
this.http.get<Movie[]>(`http://localhost:5000/api/v1.0/movies?pn=${page_number
}`)
    }

    getMoviesMultipleArgs(args: string): Observable<Movie[]> {
        return
this.http.get<Movie[]>(`http://localhost:5000/api/v1.0/movies${args}`)
    }

    getMovieById(id: string): Observable<Movie> {
        return
this.http.get<Movie>(`http://localhost:5000/api/v1.0/movies?id=${id}`)
    }

    getMovieByTitle(title: string): Observable<Movie[]> {
        return
this.http.get<Movie[]>(`http://localhost:5000/api/v1.0/movies?title=${title}`)
    }

    getMovieByDecade(decade: string): Observable<Movie[]> {
        return
this.http.get<Movie[]>(`http://localhost:5000/api/v1.0/movies?decade=${decade}
`)
    }
    getMovieByGenre(genre: string): Observable<Movie[]> {
        return
this.http.get<Movie[]>(`http://localhost:5000/api/v1.0/movies?genre=${genre}`)
    }

    getAllReviews(id: string): Observable<Review[]> {
        return
this.http.get<Review[]>(`http://localhost:5000/api/v1.0/movies/${id}/reviews`)
    }

    addNewReview(movieId: string, comment: string, stars: string, username:
string): Observable<UrlResponse> {
        const formData = new FormData();

        formData.append("username", username);
        formData.append("comment", comment);
        formData.append("stars", stars);

        return
this.http.post<UrlResponse>(`http://localhost:5000/api/v1.0/movies/${movieId}/
reviews`, formData, { headers: { 'x-access-token': this.token } })
    }

    editExistingReview(movieId: string, comment: string, stars: string,
username: string, reviewId: string): Observable<UrlResponse> {
        const formData = new FormData();

        formData.append("username", username);
        formData.append("comment", comment);
        formData.append("stars", stars);

        return
this.http.put<UrlResponse>(`http://localhost:5000/api/v1.0/movies/${movieId}/r
eviews/${reviewId}`, formData, { headers: { 'x-access-token': this.token } })
    }

    deleteReview(movieId: string, reviewId: string): Observable<{}> {
        return
this.http.delete<{}>(`http://localhost:5000/api/v1.0/movies/${movieId}/reviews
/${reviewId}`, { headers: { 'x-access-token': this.token } })
    }

    addNewMovie(newTitle: string, newExtract: string, newYear: number,
newCast: string, newGenres: string, newThumbnail: string):
Observable<UrlResponse> {
        const formData = new FormData();
        formData.append("title", newTitle);
        formData.append("extract", newExtract);
        formData.append("year", newYear.toString());
        formData.append("cast", newCast);
        formData.append("genres", newGenres);
        formData.append("thumbnail", newThumbnail);

        return
this.http.post<UrlResponse>(`http://localhost:5000/api/v1.0/movies`, formData,
{ headers: { 'x-access-token': this.token } })
    }

    editMovie(id: string, newTitle: string, newExtract: string, newYear:
number, newCast: string, newGenres: string, newThumbnail: string):
Observable<UrlResponse> {
        const formData = new FormData();

        formData.append("title", newTitle);
        formData.append("extract", newExtract);
        formData.append("year", newYear.toString());
        formData.append("cast", newCast);
        formData.append("genres", newGenres);
        formData.append("thumbnail", newThumbnail);

        return
this.http.put<UrlResponse>(`http://localhost:5000/api/v1.0/movies/${id}`,
formData, { headers: { 'x-access-token': this.token } })
    }

    deleteMovie(id: string): Observable<{}> {
        return
this.http.delete<{}>(`http://localhost:5000/api/v1.0/movies/${id}`, { headers:
{ 'x-access-token': this.token } })
    }
}
```

actor.service.ts –

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Actor } from './types/actor.type';
import { UrlResponse } from './types/urlResponse.type';

export class ActorService {
    token = sessionStorage.getItem("token") || '';

    constructor(public http: HttpClient) { }
    getActors(page_number: number): Observable<Actor[]> {
        return
this.http.get<Actor[]>(`http://localhost:5000/api/v1.0/actors?pn=${page_number
}`)
    }

    getActorById(id: string): Observable<Actor> {
        return
this.http.get<Actor>(`http://localhost:5000/api/v1.0/actors?id=${id}`)
    }

    getActorByName(name: string): Observable<Actor[]> {
        return
this.http.get<Actor[]>(`http://localhost:5000/api/v1.0/actors?name=${name}`)
    }

    getActorByDecade(decade: string): Observable<Actor[]> {
        return
this.http.get<Actor[]>(`http://localhost:5000/api/v1.0/actors?decade=${decade}
`)
    }

    getActorsMultipleArgs(args: string): Observable<Actor[]> {
        return
this.http.get<Actor[]>(`http://localhost:5000/api/v1.0/actors${args}`)
    }

    addNewActor(name: string, birthYear: number, deathYear: number, image:
string): Observable<UrlResponse> {
        const formData = new FormData();

        formData.append("birthYear", birthYear.toString());
        if (deathYear) {
            formData.append("deathYear", deathYear.toString());
        } else formData.append("deathYear", '')

        formData.append("image", image);
        formData.append("primaryName", name);

        return
this.http.post<UrlResponse>(`http://localhost:5000/api/v1.0/actors`, formData,
{ headers: { 'x-access-token': this.token } })
    }

    deleteActor(id: string): Observable<{}> {
        return
this.http.delete<{}>(`http://localhost:5000/api/v1.0/actors/${id}`, { headers:
{ 'x-access-token': this.token } })
```

```
    }

    follow(userId: string, actorId: string): Observable<UrlResponse> {
        return
this.http.put<UrlResponse>(`http://localhost:5000/api/v1.0/users/${userId}/fol
lowing/${actorId}`, {}, { headers: { 'x-access-token': this.token } })

    }

    unfollow(userId: string, actorId: string): Observable<UrlResponse> {
        return
this.http.delete<UrlResponse>(`http://localhost:5000/api/v1.0/users/${userId}/
following/${actorId}`, { headers: { 'x-access-token': this.token } })

    }
}
```

app.module.ts –

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { WebService } from './web.service';
import { HttpClient, HttpClientModule } from '@angular/common/http';
import { HomeComponent } from './home/home.component';
import { MovieComponent } from './movie/movie.component';
import { NavbarComponent } from './navbar/navbar.component';
import { FormsModule } from '@angular/forms';
import { RouterModule } from '@angular/router';
import { ActorsComponent } from './actors/actors.component';
import { ActorComponent } from './actor/actor.component';
import { AdminComponent } from './admin/admin.component';
import { RouterTestingModule } from '@angular/router/testing';
import { AuthModule } from '@auth0/auth0-angular';
import { LoginComponent } from './login/login.component';
import { ProfileComponent } from './profile/profile.component';

var routes: any = [
  {
    path: '',
    component: HomeComponent
  },
  {
    path: 'actors',
    component: ActorsComponent
  },
  {
    path: 'movie/:id',
    component: MovieComponent
  },
  {
    path: 'actors/:id',
    component: ActorComponent
  },
  {
    path: 'profile',
    component: ProfileComponent
  },
  {
    path: 'login',
    component: LoginComponent
  },
  {
    path: 'admin',
    component: AdminComponent
  }
];
@NgModule({
  declarations: [
    AppComponent,
    NavbarComponent,
    HomeComponent,
    MovieComponent,
    ActorsComponent,
    ActorComponent,
    AdminComponent,
    LoginComponent,
    ProfileComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    RouterModule,
    FormsModule,
    RouterTestingModule,
    RouterModule.forRoot(routes),
    AuthModule.forRoot({
      domain: 'dev-loovxx4fwuzohi8k.us.auth0.com',
      clientId: 'VxdQP7K2XaDCkXyed5ikMVphkvoRLWZj',
      authorizationParams: {
        redirect_uri: window.location.origin,
      },
    })
  ],
  providers: [WebService, HttpClient],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.ts –

```
import { Component } from '@angular/core';
import { WebService } from './web.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers: [WebService]
})
export class AppComponent {
  title = 'CinePal';
}
```

app.component.html –

```
<app-navbar></app-navbar>
<router-outlet></router-outlet>
```

app.routing.module.ts –

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { ActorComponent } from './actor/actor.component';
import { ActorsComponent } from './actors/actors.component';
import { AdminComponent } from './admin/admin.component';
import { HomeComponent } from './home/home.component';
import { MovieComponent } from './movie/movie.component';

const routes: Routes = [
  {
      path: '',
      component: HomeComponent
  },
  {
      path: 'actors',
      component: ActorsComponent
  },
  {
      path: 'movie/:id',
      component: MovieComponent
  },
  {
      path: 'actors/:id',
      component: ActorComponent
  },
  {
      path: 'admin',
      component: AdminComponent
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

types/actor.type.ts –

```
export type Actor = {
    _id: string;
    primaryName: string;
    birthYear: number;
    deathYear: string;
    image: string | undefined;
}
```

types/movie.type.ts –

```
import { Actor } from "./actor.type";
import { Review } from "./review.type";

export type Movie = {
    _id: string;
    title: string;
    year: number;
    cast: Actor[] | [];
    genres: string[] | [];
    href: string;
    extract: string;
    thumbnail: string;
    thumbnail_width: number;
    thumbnail_height: number;
    reviews: Review[] | [];
}
```

types/review.type.ts –

```
export type Review = {
    _id: string;
    username: string;
    comment: string;
    stars: string;
    toggled?: boolean;
}
```

types/urlResponse.type.ts –

```
export type UrlResponse = {
    url: string;
}
```

types/user.type.ts –

```
import { Actor } from "./actor.type";
import { Movie } from "./movie.type";

export type User = {
    _id: string;
    name: string;
    username: string;
    password: string;
    email: string;
    admin: boolean;
    favourites: Movie[] | [];
    watchlist: Movie[] | [];
    following: Actor[] | [];
    toggled?: boolean;
}
```

profile/profile.component.html –

```
<div class="container" *ngIf="username && user">
    <div class="row">
        <div class="col-sm-8 m-auto text-center">
            <h2>Profile</h2>
        </div>
    </div>
    <div class="row">
        <div class="col-lg-12">
            <div class="card mb-3" style="background-color: white !important;
color: #9A1750;">
                <div class="card-header" style="text-align: center; font-size:
larger; font-weight: bold;">
                    {{user.username}}
```

```html
                </div>
                <div class="card-body">
                    <div class="row">
                        <h5>Name: {{user.name}}</h5>
                    </div>
                    <div class="row">
                        <h5>Email: {{user.email}}</h5>
                    </div>
                    <div class="row">
                        <h5>Admin: {{user.admin}}</h5>
                    </div>
                    <div class="row m-2 movie-content p-2" style="background-
color: #9A1750; color: white;">
                        <h5 class="m-2">Favourites</h5>
                        <ng-container *ngFor="let movie of user.favourites">
                            <div class="col-sm-2 m-auto">
                                <img src={{movie.thumbnail}} alt=""
style="min-height: 280px; max-height: 280px;"
                                    [routerLink]="['../movie', movie._id]">
                            </div>
                        </ng-container>
                    </div>

                    <div class="row m-2 movie-content p-2" style="background-
color: #9A1750; color: white;">
                        <h5 class="m-2">Watchlist</h5>
                        <ng-container *ngFor="let movie of user.watchlist">
                            <div class="col-sm-2 m-auto">
                                <img src={{movie.thumbnail}} alt=""
style="min-height: 280px; max-height: 280px;"
                                    [routerLink]="['../movie', movie._id]">
                            </div>
                        </ng-container>
                    </div>

                    <div class="row m-2 movie-content p-2">
                        <h5 class="m-2">Following</h5>
                        <ng-container *ngFor="let actor of user.following">
                            <div class="col-sm-4 m-auto">
                                <button class="btn btn-primary w-50 m-auto"
                                    [routerLink]="['../actors',
actor._id]">{{actor.primaryName}}</button>
                            </div>
                        </ng-container>
                    </div>
                </div>
                <div class="card-footer" style="color: white;">
                    <div class="row p-3">
```

```
                        <div class="row mb-3" style="color: #9A1750;">
                            <form #editUserForm="ngForm" appIdentityRevealed
name="editUserForm">
                                <div class="form-group">
                                    <label for="name">Name</label>
                                    <input type="text" id="name" name="name"
class="form-control" required minlength="4"
                                        [(ngModel)]="user.name"
#newNameField="ngModel" />
                                    <div *ngIf="newNameField.invalid &&
(newNameField.dirty || newNameField.touched)"
                                        class="alert alert-danger m-1">
                                        <div
*ngIf="newNameField.errors?.['required']">Name is required.
                                        </div>
                                        <div
*ngIf="newNameField.errors?.['minlength']">
                                            Name must be a minimum length of
4.
                                        </div>
                                    </div>
                                </div>

                                <div class="form-group">
                                    <label for="username">Username</label>
                                    <input type="text" id="username"
name="username" class="form-control" required
                                        minlength="4"
[(ngModel)]="user.username" #newUsernameField="ngModel" />
                                    <div *ngIf="newUsernameField.invalid &&
(newUsernameField.dirty || newUsernameField.touched)"
                                        class="alert alert-danger m-1">
                                        <div
*ngIf="newUsernameField.errors?.['required']">Username is
                                            required.</div>
                                        <div
*ngIf="newUsernameField.errors?.['minlength']">
                                            Name must be a minimum length of
4.
                                        </div>
                                    </div>
                                </div>

                                <div class="form-group">
                                    <label for="password">Password</label>
                                    <input type="password" id="password"
name="password" class="form-control" required
```

```
                                        pattern="^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$"
                                        [(ngModel)]="user.password"
#newPasswordField="ngModel" />
                                    <div *ngIf="newPasswordField.invalid &&
(newPasswordField.dirty || newPasswordField.touched)"
                                        class="alert alert-danger m-1">
                                        <div
*ngIf="newPasswordField.errors?.['required']">Password is
                                            required.</div>
                                        <div
*ngIf="newPasswordField.errors?.['pattern']">
                                            Password must be a minimum of
eight characters, at least one

                                            uppercase letter, one
                                            lowercase
                                            letter, one number and one special
character.
                                        </div>
                                    </div>
                                </div>


                                <div class="form-group">
                                    <label for="email">Email</label>
                                    <input type="email" id="email"
name="email" class="form-control"
                                        pattern="[a-z0-9._%+-]+@[a-z0-9.-
]+\.[a-z]{2,4}$" [(ngModel)]="user.email"
                                        #newEmailField="ngModel" required />
                                    <div *ngIf="newEmailField.invalid &&
(newEmailField.dirty || newEmailField.touched)"
                                        class="alert alert-danger m-1">
                                        <div
*ngIf="newEmailField.errors?.['required']">Email is required.
                                        </div>
                                        <div
*ngIf="newEmailField.errors?.['pattern']">
                                            Please enter a valid email.
                                        </div>
                                    </div>
                                </div>

                                <div class="form-group" *ngIf="admin">
                                    <label for="admin">Admin</label><br>
                                    <input class="form-check-input"
type="checkbox" id="admin" [checked]="newAdmin"
                                        (change)="newAdmin = !newAdmin"
value="">
```

```
                                <div>
                                <button class="btn btn-primary m-2"
type="button" [disabled]="editUserForm.invalid"
                                    (click)="submitEditUser(user._id,
user.username, user.name, user.password, user.email, user.favourites,
user.watchlist, user.following)">Submit</button>
                                <button class="btn btn-secondary  m-2"
type="button"
                                    (click)="editUserForm.resetForm({})">Reset
</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

profile/profile.component.ts –

```
import { Component } from '@angular/core';
import { WebService } from '../web.service';
import { Router } from '@angular/router';
import { User } from '../types/user.type';
import { Movie } from '../types/movie.type';
import { Actor } from '../types/actor.type';

@Component({
  selector: 'app-profile',
  templateUrl: './profile.component.html',
  styleUrls: ['./profile.component.css']
})
export class ProfileComponent {
  username = sessionStorage.getItem("username");
  admin = (sessionStorage.getItem("admin") === 'true');
  user!: User;
  newName = '';
  newUsername = '';
  newPassword = '';
  newEmail = '';
  newAdmin = false;
  show = true
  constructor(public webService: WebService, private router: Router) {

  }
  submitEditUser = (userId: string, username: string, name: string, password:
string, email: string, favourites: Array<Movie>, watchlist: Array<Movie>,
following: Array<Actor>) => {
    const favourite_ids = favourites.map((movie) => movie._id);
    const watchlist_ids = watchlist.map((movie) => movie._id);
    const following_ids = following.map((actor) => actor._id);

    this.webService.userService.editExistingUser(userId, username, name,
password, email, this.admin, favourite_ids, watchlist_ids,
following_ids).subscribe(
      () => {
        window.location.reload()
      }
    )
  }

  ngOnInit(): void {
    if (this.username) {
      this.webService.userService.getUserByUsername(this.username!).subscribe(
(response) => {
        this.user = response[0]
      })
    } else {
      this.router.navigate(['../'])
    }
  }
}
```

navbar/navbar.component.css –

```
a {
    color: #E3E2DF;
}

.navbar {
    background-color: #5D001E !important;
    margin-bottom: 2rem;
}

.custom-toggler .navbar-toggler-icon {
  background-image: url("data:image/svg+xml;charset=utf8,%3Csvg viewBox='0 0
32 32' xmlns='http://www.w3.org/2000/svg'%3E%3Cpath stroke='rgba(195,7,63,
0.5)' stroke-width='2' stroke-linecap='round' stroke-miterlimit='10' d='M4
8h24M4 16h24M4 24h24'/%3E%3C/svg%3E");
}
```

```
.custom-toggler.navbar-toggler {
  border-color: #E3E2DF;
}
```

navbar/navbar.component.html –

```
<nav class="navbar navbar-expand-lg sticky-top">
    <div class="container-fluid">
        <a class="navbar-brand" href="#" style="color: #E3E2DF;">CinePal</a>
        <button class="navbar-toggler ml-auto custom-toggler" type="button"
data-bs-toggle="collapse"
            data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false"
            aria-label="Toggle navigation" style="color: #E3E2DF;">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link" aria-current="page" href="#">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" aria-current="page"
href="actors">Actors</a>
                </li>
                <li class="nav-item" *ngIf="loggedIn">
                    <a class="nav-link" aria-current="page"
href="profile">Profile</a>
                </li>
                <li class="nav-item" *ngIf="admin">
                    <a class="nav-link" aria-current="page"
href="admin">Admin</a>
                </li>
            </ul>
            <ul class="navbar-nav navbar-right">
                <li class="nav-item">
                    <div *ngIf="loggedIn">
                        <button class="btn btn-primary" (click)="logOut()">
                            Logout</button>
                    </div>
                    <div *ngIf="loggedIn == false">
                        <button class="btn btn-primary"
[routerLink]="['login']">
                            Login</button>
                    </div>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

navbar/navbar.component.ts –

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '@auth0/auth0-angular';


@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent {
  loggedIn: Boolean = false;
  admin = (sessionStorage.getItem("admin") === 'true');
  constructor(public authService: AuthService, public router: Router) { }

  logOut() {
    sessionStorage.clear();
    window.location.reload()
  }

  ngOnInit(): void {
    this.loggedIn = Boolean(sessionStorage.getItem("token")?.toString())
  }

}
```

movie/movie.component.css –

```
.card {
    background-color:  #5D001E !important;
    color: white;
}
```

movie/movie.component.html –

```
<div *ngIf="movie">
    <div class="container">
        <div class="row">
            <div class="col-lg-3 mb-2">
                <img src={{movie.thumbnail}} alt="">
            </div>
            <div class="col-lg-9">
```

```
                <div class="row mb-2">
                    <div class="col-lg-6">
                        <h2>{{movie.title}}</h2>
                    </div>
                    <div class="col-sm-3 mt-2" *ngIf="username &&
!favourited">
                        <button type="button" class="btn btn-primary"
(click)="addToFavourites()">Add to
                            favourites</button>
                    </div>
                    <div class="col-sm-3 mt-2" *ngIf="username && favourited">
                        <button type="button" class="btn btn-primary"
(click)="removeFromFavourites()">Remove from
                            favourites</button>
                    </div>
                    <div class="col-sm-3 mt-2" *ngIf="username &&
!watchlisted">
                        <button type="button" class="btn btn-primary"
(click)="addToWatchlist()">Add to
                            watchlist</button>
                    </div>
                    <div class="col-sm-3 mt-2" *ngIf="username &&
watchlisted">
                        <button type="button" class="btn btn-primary"
(click)="removeFromWatchlist()">Remove from
                            watchlist</button>
                    </div>
                </div>
                <div class="row">
                    <div class="col-sm-1">
                        <h6>{{movie.year}}</h6>
                    </div>
                    <ng-container *ngFor="let genre of movie.genres">
                        <div class="col-2">
                            <h6>{{genre}}</h6>
                        </div>
                    </ng-container>
                </div>
                <p>{{movie.extract}}</p>
                <div>
                    <h4>Cast</h4>
                    <div class="row">
                        <ng-container *ngFor="let actor of movie.cast">
                            <div class="col-lg-3">
                                <button type="button" class="btn btn-primary
mt-1 mr-1"
                                    [routerLink]="['../../actors',
actor._id]">{{actor.primaryName}}</button>
```

```
                            </div>
                        </ng-container>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div class="container mt-5">
        <div class="row">
            <div class="col-lg-1 pt-2">
                <h4>Reviews</h4>
            </div>
            <div class="col-sm-3 mt-2" *ngIf="username">
                <button type="button" class="btn btn-primary"
                    (click)="this.reviewFormToggle =
!this.reviewFormToggle">Add a
                    review</button>
            </div>
        </div>
        <div class="row mb-3" *ngIf="this.reviewFormToggle">
            <div class="col-lg-4 mt-2">
                <div class="card">
                    <div class="card-body">
                        <div class="alert alert-success m-1">
                            Complete the form to enable the Submit button.
                        </div>
                        <form #reviewForm="ngForm" appIdentityRevealed>
                            <div class="form-group">
                                <label for="comment">Comment</label>
                                <textarea id="comment" name="comment"
class="form-control" required minlength="4"
                                    [(ngModel)]="reviewComment"
#comment="ngModel"></textarea>
                                <div *ngIf="comment.invalid && (comment.dirty
|| comment.touched)"
                                    class="alert alert-danger m-1">
                                    <div
*ngIf="comment.errors?.['required']">Comment is required.</div>
                                    <div
*ngIf="comment.errors?.['minlength']">
                                        Comment must be at least 4 characters
long.
                                    </div>
                                </div>
                            </div>
                            <div class="form-group">
                                <label for="stars">Stars</label>
                                <input type="number" id="stars" name="stars"
class="form-control w-50" required min="1"
                                    max="5" [(ngModel)]="reviewStars"
#stars="ngModel" />
                                <div *ngIf="stars.invalid && (stars.dirty ||
stars.touched)"
                                    class="alert alert-danger m-1">
                                    <div
*ngIf="stars.errors?.['required']">Stars is required.</div>
                                    <div *ngIf="stars.errors?.['min']">
                                        Stars must be at least 1.
                                    </div>
                                    <div *ngIf="stars.errors?.['max']">
                                        Stars must be a maximum of 5.
                                    </div>
                                </div>
                            </div>
                            <button class="btn btn-primary m-2" type="submit"
[disabled]="reviewForm.invalid"
                                (click)="submitReview()">Submit</button>
                            <button class="btn btn-secondary  m-2"
type="button"
                                (click)="reviewForm.resetForm({})">Reset</butt
on>
                        </form>
                    </div>
                </div>
            </div>
        </div>
        <div class="row mb-3" *ngIf="show">
            <ng-container *ngFor="let review of movie.reviews; let i=index;">
                <div class="col-lg-4 mt-2">
                    <div class="card">
                        <div class="card-body">
                            <h5 class="card-title m-2">{{review.comment}}</h5>
                            <h6 class="card-subtitle m-
2">{{review.stars}}</h6>
                            <p class="card-text m-2">{{review.username}}</p>
                            <div *ngIf="admin || review.username == username">
                                <button type="button" class="btn btn-primary
m-2"
                                    (click)="review.toggled =
!review.toggled">Edit
                                    Review</button>
                                <button type="button" class="btn btn-primary
m-2"
                                    (click)="deleteReview(review._id)">Delete
                                    Review</button>
                            </div>
                            <div class="row mb-3" *ngIf="review.toggled">
                                <div class="mt-2">
                                    <div class="card">
                                        <div class="card-body">
                                            <div class="alert alert-success m-
1">
                                                Complete the form to enable
the Submit button.
                                            </div>
                                            <form #editReviewForm="ngForm"
appIdentityRevealed
                                                name="editReviewForm{{i}}">
                                                <div class="form-group">
                                                    <label
for="comment">Comment</label>
                                                    <textarea id="comment"
name="comment{{i}}" class="form-control"
                                                        required minlength="4"
[(ngModel)]="review.comment"
                                                        #comment="ngModel"></t
extarea>
                                                    <div
*ngIf="comment.invalid && (comment.dirty || comment.touched)"
                                                        class="alert alert-
danger m-1">
                                                        <div
*ngIf="comment.errors?.['required']">Comment is required.
                                                        </div>
                                                        <div
*ngIf="comment.errors?.['minlength']">
                                                            Comment must be at
least 4 characters long.
                                                        </div>
                                                    </div>
                                                </div>
                                                <div class="form-group">
                                                    <label
for="stars">Stars</label>
                                                    <input type="number"
id="stars" name="stars{{i}}"
                                                        class="form-control w-
50" required min="1" max="5"
                                                        [(ngModel)]="review.st
ars" #stars="ngModel" />
                                                    <div *ngIf="stars.invalid
&& (stars.dirty || stars.touched)"
```

```
                                                        class="alert alert-
danger m-1">
                                                        <div
*ngIf="stars.errors?.['required']">Stars is required.</div>
                                                        <div
*ngIf="stars.errors?.['min']">
                                                            Stars must be at
least 1.
                                                        </div>
                                                        <div
*ngIf="stars.errors?.['max']">
                                                            Stars must be a
maximum of 5.
                                                        </div>
                                                    </div>
                                                </div>
                                                <div>
                                                    <button class="btn btn-
primary m-2" type="button"
                                                        [disabled]="editReview
Form.invalid"
                                                        (click)="editReview(re
view._id, review.comment, review.stars)">Submit</button>
                                                    <button class="btn btn-
secondary  m-2" type="button"
                                                        (click)="editReviewFor
m.resetForm({})">Reset</button>
                                                </div>
                                            </form>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </ng-container>
        </div>
        <div class="row mb-3" *ngIf="admin">
            <div class="col-sm-2 mt-2">
                <button type="button" class="btn btn-primary"
(click)="deleteMovie(movie._id)">Delete movie</button>
            </div>
            <div class="col-sm-2 mt-2">
                <button type="button" class="btn btn-primary"
(click)="editMovieToggle = !editMovieToggle">Edit
                    movie</button>
            </div>
        </div>

        <div class="row mb-3" *ngIf="editMovieToggle">
            <div class="card col-lg-8 m-2">
                <div class="card-header">
                    <h5>Edit movie</h5>
                </div>
                <div class="card-body">
                    <form #editMovieForm="ngForm" appIdentityRevealed
title="editMovieForm">
                        <div class="form-group">
                            <label for="title">Title</label>
                            <input type="text" id="title" name="title"
class="form-control" required minlength="4"
                                [(ngModel)]="movie.title"
#newTitleField="ngModel" />
                            <div *ngIf="newTitleField.invalid &&
(newTitleField.dirty || newTitleField.touched)"
                                class="alert alert-danger m-1">
                                <div
*ngIf="newTitleField.errors?.['required']">Title is required.</div>
                                <div
*ngIf="newTitleField.errors?.['minlength']">
                                    Title must be a minimum length of 4.
                                </div>
                            </div>
                        </div>

                        <div class="form-group">
                            <label for="extract">Extract</label>
                            <textarea id="extract" name="extract" class="form-
control" required minlength="4"
                                [(ngModel)]="movie.extract"
#newExtractField="ngModel"></textarea>
                            <div *ngIf="newExtractField.invalid &&
(newExtractField.dirty || newExtractField.touched)"
                                class="alert alert-danger m-1">
                                <div
*ngIf="newExtractField.errors?.['required']">Extract is required.</div>
                                <div
*ngIf="newExtractField.errors?.['minlength']">
                                    Extract must be a minimum length of 4.
                                </div>
                            </div>
                        </div>

                        <div class="form-group">
                            <label for="year">Year</label>
                            <input type="text" id="year" name="year"
class="form-control" required minlength="4"
                                [(ngModel)]="movie.year"
#newYearField="ngModel" />
                            <div *ngIf="newYearField.invalid &&
(newYearField.dirty || newYearField.touched)"
                                class="alert alert-danger m-1">
                                <div
*ngIf="newYearField.errors?.['required']">Year is required.</div>
                                <div
*ngIf="newYearField.errors?.['minlength']">
                                    Year must be a minimum length of 4.
                                </div>
                            </div>
                        </div>

                        <div class="form-group">
                            <label for="cast">Cast</label>
                            <input type="cast" id="cast" name="cast"
class="form-control" required [(ngModel)]="cast"
                                #newCastField="ngModel" />
                            <div *ngIf="newCastField.invalid &&
(newCastField.dirty || newCastField.touched)"
                                class="alert alert-danger m-1">
                                <div
*ngIf="newCastField.errors?.['required']">Cast is required.</div>
                            </div>
                        </div>


                        <div class="form-group">
                            <label for="genres">Genres</label>
                            <input type="genres" id="genres" name="genres"
class="form-control" [(ngModel)]="genres"
                                #newGenresField="ngModel" required />
                            <div *ngIf="newGenresField.invalid &&
(newGenresField.dirty || newGenresField.touched)"
                                class="alert alert-danger m-1">
                                <div
*ngIf="newGenresField.errors?.['required']">Genres is required.</div>
                            </div>
                        </div>

                        <div class="form-group">
                            <label for="thumbnail">Image url</label>
                            <input type="thumbnail" id="thumbnail"
name="thumbnail" class="form-control"
                                [(ngModel)]="movie.thumbnail"
#newThumbnailField="ngModel" required />
```

```
                            <div *ngIf="newThumbnailField.invalid &&
(newThumbnailField.dirty || newThumbnailField.touched)"
                                class="alert alert-danger m-1">
                                <div
*ngIf="newThumbnailField.errors?.['required']">Image url is required.</div>
                            </div>
                        </div>


                        <button class="btn btn-primary m-2" type="button"
[disabled]="editMovieForm.invalid"
                            (click)="editMovie(movie._id, movie.title,
movie.extract, movie.year, cast, genres, movie.thumbnail)">Submit</button>
                        <button class="btn btn-secondary  m-2" type="button"
                            (click)="editMovieForm.resetForm({})">Reset</butto
n>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
```

movie/movie.component.ts –

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router, ParamMap } from '@angular/router';
import { WebService } from '../web.service';
import { Movie } from '../types/movie.type';
import { Review } from '../types/review.type';
import { User } from '../types/user.type';

@Component({
  selector: 'app-movie',
  templateUrl: './movie.component.html',
  styleUrls: ['./movie.component.css']
})
export class MovieComponent implements OnInit {
  admin = (sessionStorage.getItem("admin") === 'true');
  username = sessionStorage.getItem("username") || '';
  movie!: Movie;
  id!: string;
  reviewFormToggle = false;
  reviews!: Review[];
  user!: User;
  reviewComment = '';
  reviewStars = '';
  cast = '';
  genres = '';
  newTitle = '';
  newExtract = '';
  newYear = '';
  newCast = '';
  newGenres = '';
  newThumbnail = '';
  show = true
  editMovieToggle = false;
  favourited = false;
  watchlisted = false;
  constructor(public webService: WebService, private route: ActivatedRoute,
private router: Router) {

  }

  reload = () => {
    this.show = false;
    setTimeout(() => this.show = true);
  }


  submitReview = () => {
    this.webService.movieService.addNewReview(this.id, this.reviewComment,
this.reviewStars, this.username!).subscribe(() => {
      window.location.reload()
    })
  }

  getReviews = () => {
    this.webService.movieService.getAllReviews(this.id).subscribe((response)
=> {
      this.reviews = response;
    });
  }

  editReview = (reviewId: string, newComment: string, newStars: string) => {
    this.webService.movieService.editExistingReview(this.id, newComment,
newStars, this.username, reviewId).subscribe(
      () => {
        this.webService.movieService.getMovieById(this.id).subscribe((response
) => {
          this.movie = response;
        })
      }
    )
  }

  deleteReview = (reviewId: string) => {
    this.webService.movieService.deleteReview(this.id, reviewId).subscribe(
      () => {
        this.webService.movieService.getMovieById(this.id).subscribe((response
) => {
          this.movie = response;
        })
      }
    )

  }

  editMovie = (id: string, newTitle: string, newExtract: string, newYear:
number, newCast: string, newGenres: string, newThumbnail: string) => {
    this.webService.movieService.editMovie(id, newTitle, newExtract, newYear,
newCast, newGenres, newThumbnail).subscribe(
      () => {
        window.location.reload()
      }
    )
  }

  deleteMovie = (movieId: string) => {
    if (confirm('Are you sure you want to delete this movie? (this cannot be
reversed)')) {
      this.webService.movieService.deleteMovie(movieId).subscribe(
        () => {
          this.router.navigate([`../`])
        }
      )
    }

  }

  addToFavourites = () => {
    this.webService.userService.addToFavourites(this.user._id,
this.id).subscribe(() => {
      window.location.reload()
    })
  }

  addToWatchlist = () => {
    this.webService.userService.addToWatchlist(this.user._id,
this.id).subscribe(() => {
      window.location.reload()
    })
  }

  removeFromFavourites = () => {
    this.webService.userService.removeFromFavourites(this.user._id,
this.id).subscribe(() => {
      window.location.reload()
    })
  }

  removeFromWatchlist = () => {
    this.webService.userService.removeFromWatchlist(this.user._id,
this.id).subscribe(() => {
      window.location.reload()
    })
  }

  ngOnInit(): void {
    this.route.paramMap.subscribe((params: ParamMap) => {
      this.id = params.get('id')!;
    });

    this.webService.movieService.getMovieById(this.id).subscribe((response) =>
{
      this.movie = response;
      this.cast = this.movie.cast.map((Actor) => Actor.primaryName).join(',
');
      this.genres = this.movie.genres.join(', ')
    })

    if (this.username) {
      this.webService.userService.getUserByUsername(this.username!).subscribe(
(response) => {
        this.user = response[0]
        this.user.favourites.map((movie: { _id: string; }) => {
          if (this.id == movie._id) {
            this.favourited = true
          }
        });

        this.user.watchlist.map((movie: { _id: string; }) => {
          if (this.id == movie._id) {
            this.watchlisted = true
          }
        });
      })

    }
    this.getReviews()
  }

}
```

login/login.component.html –

```
<div class="container">
    <div class="col-sm-5 m-auto">
        <div *ngIf="loginError" class="alert alert-danger">
            Invalid credentials
        </div>
        <form #loginForm="ngForm" appIdentityRevealed name="loginForm">
            <div class="form-group">
                <label for="username">Username</label>
                <input type="text" id="username" name="username" class="form-
control" required minlength="4"
                    [(ngModel)]="username" #userNameField="ngModel" />
                <div *ngIf="userNameField.invalid && (userNameField.dirty ||
userNameField.touched)"
                    class="alert alert-danger m-1">
                    <div *ngIf="userNameField.errors?.['required']">Username
is required.</div>
                    <div *ngIf="userNameField.errors?.['minlength']">
                        Name must be a minimum length of 4.
                    </div>
                </div>
            </div>

            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" id="password" name="password"
class="form-control" required
                    [(ngModel)]="password" #passwordField="ngModel" />
                <div *ngIf="passwordField.invalid && (passwordField.dirty ||
passwordField.touched)"
                    class="alert alert-danger m-1">
                    <div *ngIf="passwordField.errors?.['required']">Password
is required.</div>
                    <div *ngIf="passwordField.errors?.['pattern']">
                        Password must be a minimum of eight characters, at
least one uppercase letter, one
                        lowercase
                        letter, one number and one special character.
                    </div>
                </div>
            </div>
            <button class="btn btn-primary m-2" type="button"
[disabled]="loginForm.invalid"
                (click)="login(username, password)">Login</button>
        </form>
    </div>
    <div class="col-sm-5 m-auto">
        <button class="btn btn-primary m-2" type="button"
(click)="signUpToggle = !signUpToggle">Sign up</button>
    </div>
    <div class="col-sm-5 m-auto" *ngIf="signUpToggle">
        <div class="card-header" align="center">
            <h4>Sign up</h4>
        </div>
        <div class="card-body">
            <form #newUserForm="ngForm" appIdentityRevealed
name="newUserForm">
                <div class="form-group">
                    <label for="name">Name</label>
                    <input type="text" id="name" name="name" class="form-
control" required minlength="4"
                        [(ngModel)]="newName" #newNameField="ngModel" />
                    <div *ngIf="newNameField.invalid && (newNameField.dirty ||
newNameField.touched)"
                        class="alert alert-danger m-1">
                        <div *ngIf="newNameField.errors?.['required']">Name is
required.</div>
                        <div *ngIf="newNameField.errors?.['minlength']">
                            Name must be a minimum length of 4.
                        </div>
                    </div>
                </div>

                <div class="form-group">
                    <label for="username">Username</label>
                    <input type="text" id="username" name="username"
class="form-control" required minlength="4"
                        [(ngModel)]="newUsername" #newUsernameField="ngModel"
/>
                    <div *ngIf="newUsernameField.invalid &&
(newUsernameField.dirty || newUsernameField.touched)"
                        class="alert alert-danger m-1">
                        <div
*ngIf="newUsernameField.errors?.['required']">Username is required.</div>
                        <div *ngIf="newUsernameField.errors?.['minlength']">
                            Name must be a minimum length of 4.
                        </div>
                    </div>
                </div>

                <div class="form-group">
                    <label for="password">Password</label>
                    <input type="password" id="password" name="password"
class="form-control" required
                        pattern="^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$"
                        [(ngModel)]="newPassword" #newPasswordField="ngModel"
/>
                    <div *ngIf="newPasswordField.invalid &&
(newPasswordField.dirty || newPasswordField.touched)"
                        class="alert alert-danger m-1">
                        <div
*ngIf="newPasswordField.errors?.['required']">Password is required.</div>
                        <div *ngIf="newPasswordField.errors?.['pattern']">
                            Password must be a minimum of eight characters, at
least one uppercase letter, one
                            lowercase
                            letter, one number and one special character.
                        </div>
                    </div>
                </div>


                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="email" id="email" name="email" class="form-
control"
                        pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$"
[(ngModel)]="newEmail"
                        #newEmailField="ngModel" required />
                    <div *ngIf="newEmailField.invalid && (newEmailField.dirty
|| newEmailField.touched)"
                        class="alert alert-danger m-1">
                        <div *ngIf="newEmailField.errors?.['required']">Email
is required.</div>
                        <div *ngIf="newEmailField.errors?.['pattern']">
                            Please enter a valid email.
                        </div>
                    </div>
                </div>
                <button class="btn btn-primary m-2" type="button"
[disabled]="newUserForm.invalid"
                    (click)="submitNewUser(newUsername, newName, newPassword,
newEmail)">Submit</button>
                <button class="btn btn-secondary  m-2" type="button"
(click)="newUserForm.resetForm({})">Reset</button>
            </form>
        </div>
    </div>
</div>
```

login/login.component.ts –

```
import { Component } from '@angular/core';
import { WebService } from '../web.service';
import { User } from '../types/user.type';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {
  username = '';
  password = '';
  signUpToggle = false;
  user!: User[];
  loggedIn = false;
  loginError = false;
  newName = '';
  newUsername = '';
  newPassword = '';
  newEmail = '';

  constructor(public webService: WebService) { }

  submitNewUser = (username: string, name: string, password: string, email:
string) => {
    this.webService.userService.addNewUser(username, name, password, email,
false).subscribe(
      () => {
        this.login(username, password);
      }
    )
  }

  login = async (username: string, password: string) => {
    const res = await this.webService.login(username, password)
      .catch((err) => {
        this.loginError = true;
      })

    if (res) {
      sessionStorage.setItem("token", res.data.token);
      this.webService.userService.getUserByUsername(username).subscribe((respo
nse) => {
        this.user = response;
        sessionStorage.setItem("admin", this.user[0].admin.toString());
        sessionStorage.setItem("username", this.user[0].username);
        location.replace('');
      })
    }
  }


}
```

home/home.component.css –

```
.card {
    background-color: #E3E2DF;
    margin-bottom: 2rem;
}

.grid {
    display: grid;
    grid-template-columns: 1fr 1fr; /* Create 2 columns and auto-place rows */
}
```

home/home.component.html –

```
<div class="container mb-2">
    <div class="row">
        <div class="col-sm-12 mb-2" align="center">
            <h2>Search</h2>
        </div>
    </div>
    <div class="row">
        <div class="d-flex justify-content-center m-auto">
            <form class="col-sm-8 d-flex">
                <div class="input-group">
                    <input id="search" type="search" class="form-control"
name="search"
                        placeholder="Search for a movie..."
(keyup.enter)="getSearchResult()"
                        [(ngModel)]="movieSearchTitle">
                    <button class="btn btn-primary"
(click)="getSearchResult()"><i class="fa fa-search"></i></button>
                </div>
                <button class="btn btn-primary" (click)="resetSearch()"
style="margin-left: 2px;">Reset</button>
            </form>
        </div>
    </div>
    <div class="row pt-4">
        <div class="d-flex justify-content-center m-auto">
            <div class="col-lg-3 text-center">
```

```
                <div class="dropdown">
                    <button class="btn btn-primary dropdown-toggle m-2"
type="button" id="dropdownMenuButton"
                        data-bs-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
                        Decade
                    </button>
                    <div class="dropdown-menu" aria-
labelledby="dropdownMenuButton">
                        <ng-container *ngFor="let decade of this.decadeList">
                            <div class="form-check" style="margin-left:
1rem;">
                                <input id="decade-{{decade}}" [value]='decade'
type="radio" name="decades"
                                    class="form-check-input"
[(ngModel)]="checkedDecade">
                                <label class=" form-check-label"
for="dropdownCheck2">
                                    {{decade}}
                                </label>
                            </div>
                        </ng-container>
                    </div>
                </div>
            </div>
            <div class="col-lg-3 text-center">
                <div class="dropdown">
                    <button class="btn btn-primary dropdown-toggle m-2"
type="button" id="dropdownMenuButton"
                        data-bs-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
                        Genre
                    </button>
                    <div class="dropdown-menu" aria-
labelledby="dropdownMenuButton">
                        <ng-container *ngFor="let genre of this.genreList">
                            <div class="form-check" style="margin-left:
1rem;">
                                <input id="genre-{{genre}}" [value]='genre'
type="radio" name="genres"
                                    class="form-check-input"
[(ngModel)]="checkedGenre">
                                <label class="form-check-label"
for="dropdownCheck2">
                                    {{genre}}
                                </label>
                            </div>
                        </ng-container>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div class="row pt-2">
        <button class="btn btn-primary w-25 m-auto" type="button"
(click)="applyFilter()"> Apply Filter </button>
    </div>
</div>
<div class="container">
    <div class="row">
        <div class="col-sm-12 mb-2" align="center">
            <h2>Top Picks</h2>
        </div>
    </div>
    <div class="row" *ngIf="show">
        <ng-container *ngFor="let movie of moviesList">
            <div class="col-lg-6">
                <div class="card mb-3">
                    <div class="card-header"
                        style="color: #9A1750; text-align: center; font-size:
larger; font-weight: bold;">
                        {{ movie.title }} ({{movie.year}})
                    </div>
                    <div class="card-body" style="display: flex;">
                        <img src={{movie.thumbnail}} alt="" style="margin:
auto; min-height: 280px; max-height: 280px;">
                    </div>
                    <div class="card-footer" style="color: white;">
                        <div class="row p-3">
                            <button class="btn btn-primary w-50 m-auto"
[routerLink]="['movie', movie._id]">More
                                Info</button>
                        </div>
                    </div>
                </div>
            </div>
        </ng-container>
    </div>
    <div class="row pb-5" *ngIf="this.movieSearchTitle.length == 0">
        <div class="col-lg-12 d-flex">
            <div class="col-sm-4">
                <button class="btn btn-primary w-100" (click)="getPrevPage()"
*ngIf="this.pageNo > 1">Previous</button>
            </div>
            <div class="col-sm-4">
                <div class="text-center">
```

```
                    <ng-container *ngFor="let page of pageList"><button
class="btn btn-primary"
                            style="margin-right: 4px; margin-top: 2px;"
                            (click)="getSpecificPage(page)">{{page}}</button>
                    </ng-container>
                </div>
            </div>
            <div class="col-sm-4">
                <button class="btn btn-primary w-100"
(click)="getNextPage()">Next</button>
            </div>
        </div>
    </div>
</div>
```

home/home.component.ts –

```
import { Component } from '@angular/core';
import { WebService } from '../web.service';
import { Movie } from '../types/movie.type';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {
  movieSearchTitle = '';
  moviesList!: Movie[];
  pageList = [1, 2, 3, 4, 5, 300, 600]
  checkedDecade = '';
  checkedGenre = '';
  genreList = ["Horror", "Thriller", "Comedy", "Action", "Drama", "Science
Fiction", "Western", "Supernatural"];
  decadeList = ["1960", "1970", "1980", "1990", "2000", "2010", "2020"];
  pageNo = 1;
  show = true;

  constructor(public webService: WebService) { }

  reload = () => {
    this.show = false;

    setTimeout(() => this.show = true);
  }

  getMovieByDecade = (decade: string) => {
    this.webService.movieService.getMovieByDecade(decade).subscribe((response)
=> {
      this.moviesList = response;
      this.reload();
    })
  }

  getMovieByGenre = (genre: string) => {
    this.webService.movieService.getMovieByGenre(genre).subscribe((response)
=> {
      this.moviesList = response;
      this.reload();
    })
  }

  applyFilter = () => {
    this.webService.movieService.getMoviesMultipleArgs(`?decade=${this.checked
Decade}&genre=${this.checkedGenre}`).subscribe((response) => {
      this.moviesList = response;
      this.reload();
    })
  }

  getSearchResult = () => {
    this.webService.movieService.getMovieByTitle(this.movieSearchTitle).subscr
ibe((response) => {
      this.moviesList = response;
      this.reload();
    })
  }

  resetSearch = () => {
    this.movieSearchTitle = '';
    this.webService.movieService.getMovies(this.pageNo).subscribe((response)
=> {
      this.moviesList = response;
      this.reload();
    })
  }

  buildArgs = () => {
    let args = `?pn=${this.pageNo}`;
    if (this.checkedDecade) {
      args += `&decade=${this.checkedDecade}`;
    };
    if (this.checkedGenre) {
      args += `&genre=${this.checkedGenre}`;
    }
    return args;
  }

  getPrevPage = () => {
    if (this.pageNo > 1) {
      this.pageNo -= 1;
      const args = this.buildArgs();
      this.webService.movieService.getMoviesMultipleArgs(args).subscribe((response) => {
        this.moviesList = response;
        this.reload();
      })
    }
  }

  getNextPage = () => {
    this.pageNo += 1;
    const args = this.buildArgs();
    this.webService.movieService.getMoviesMultipleArgs(args).subscribe((response) => {
      this.moviesList = response;
      this.reload();
    })
  }

  getSpecificPage = (page: number) => {
    this.pageNo = page;
    const args = this.buildArgs();
    this.webService.movieService.getMoviesMultipleArgs(args).subscribe((response) => {
      this.moviesList = response;
      this.reload();
    })
  }

  ngOnInit() {
    this.webService.movieService.getMovies(this.pageNo).subscribe((response) => {
      this.moviesList = response;
    })
  }
}
```

admin/admin.component.css –

```
.card {
    background-color:  #5D001E !important;
    color: white;
}
```

admin/admin.component.html –

```
<div class="container mt-4" *ngIf="admin">
    <h1 class="text-center">Admin Dashboard</h1>
    <div class="row m-1">
        <div class="card col-lg-5 m-auto">
            <div class="card-header" align="center">
                <h4>Add a user</h4>
            </div>
            <div class="card-body">
                <form #newUserForm="ngForm" appIdentityRevealed
name="newUserForm">
                    <div class="form-group">
                        <label for="name">Name</label>
                        <input type="text" id="name" name="name" class="form-
control" required minlength="4"
                            [(ngModel)]="newName" #newNameField="ngModel" />
                        <div *ngIf="newNameField.invalid &&
(newNameField.dirty || newNameField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newNameField.errors?.['required']">Name is required.</div>
                            <div *ngIf="newNameField.errors?.['minlength']">
                                Name must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="username">Username</label>
                        <input type="text" id="username" name="username"
class="form-control" required minlength="4"
                            [(ngModel)]="newUsername"
#newUsernameField="ngModel" />
                        <div *ngIf="newUsernameField.invalid &&
(newUsernameField.dirty || newUsernameField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newUsernameField.errors?.['required']">Username is required.</div>
```

```
                            <div
*ngIf="newUsernameField.errors?.['minlength']">
                                Name must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="password">Password</label>
                        <input type="password" id="password" name="password"
class="form-control" required
                            pattern="^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$"
                            [(ngModel)]="newPassword"
#newPasswordField="ngModel" />
                        <div *ngIf="newPasswordField.invalid &&
(newPasswordField.dirty || newPasswordField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newPasswordField.errors?.['required']">Password is required.</div>
                            <div *ngIf="newPasswordField.errors?.['pattern']">
                                Password must be a minimum of eight
characters, at least one uppercase letter, one
                                lowercase
                                letter, one number and one special character.
                            </div>
                        </div>
                    </div>


                    <div class="form-group">
                        <label for="email">Email</label>
                        <input type="email" id="email" name="email"
class="form-control"
                            pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$"
[(ngModel)]="newEmail"
                            #newEmailField="ngModel" required />
                        <div *ngIf="newEmailField.invalid &&
(newEmailField.dirty || newEmailField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newEmailField.errors?.['required']">Email is required.</div>
                            <div *ngIf="newEmailField.errors?.['pattern']">
                                Please enter a valid email.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
```

```
                        <label for="admin">Admin</label><br>
                        <input class="form-check-input" type="checkbox"
id="admin" [checked]="newAdmin"
                            (change)="newAdmin = !newAdmin">
                    </div>
                    <button class="btn btn-primary m-2" type="button"
[disabled]="newUserForm.invalid"
                        (click)="submitNewUser(newUsername, newName,
newPassword, newEmail, newAdmin)">Submit</button>
                    <button class="btn btn-secondary  m-2" type="button"
                        (click)="newUserForm.resetForm({})">Reset</button>
                </form>
            </div>
        </div>

        <div class="card col-lg-5 m-auto">
            <div class="card-header" align="center">
                <h4>Add a movie</h4>
            </div>
            <div class="card-body">
                <form #newMovieForm="ngForm" appIdentityRevealed
title="newMovieForm">
                    <div class="form-group">
                        <label for="title">Title</label>
                        <input type="text" id="title" name="title"
class="form-control" required minlength="4"
                            [(ngModel)]="newTitle" #newTitleField="ngModel" />
                        <div *ngIf="newTitleField.invalid &&
(newTitleField.dirty || newTitleField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newTitleField.errors?.['required']">Title is required.</div>
                            <div *ngIf="newTitleField.errors?.['minlength']">
                                Title must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="extract">Extract</label>
                        <input type="text" id="extract" name="extract"
class="form-control" required minlength="4"
                            [(ngModel)]="newExtract"
#newExtractField="ngModel" />
                        <div *ngIf="newExtractField.invalid &&
(newExtractField.dirty || newExtractField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newExtractField.errors?.['required']">Extract is required.</div>
                            <div
*ngIf="newExtractField.errors?.['minlength']">
                                Extract must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="year">Year</label>
                        <input type="text" id="year" name="year" class="form-
control" required minlength="4"
                            [(ngModel)]="newYear" #newYearField="ngModel" />
                        <div *ngIf="newYearField.invalid &&
(newYearField.dirty || newYearField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newYearField.errors?.['required']">Year is required.</div>
                            <div *ngIf="newYearField.errors?.['minlength']">
                                Year must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="cast">Cast</label>
                        <input type="cast" id="cast" name="cast" class="form-
control" required [(ngModel)]="newCast"
                            #newCastField="ngModel" />
                        <div *ngIf="newCastField.invalid &&
(newCastField.dirty || newCastField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newCastField.errors?.['required']">Cast is required.</div>
                        </div>
                    </div>


                    <div class="form-group">
                        <label for="genres">Genres</label>
                        <input type="genres" id="genres" name="genres"
class="form-control" [(ngModel)]="newGenres"
                            #newGenresField="ngModel" required />
                        <div *ngIf="newGenresField.invalid &&
(newGenresField.dirty || newGenresField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newGenresField.errors?.['required']">Genres is required.</div>
```

```
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="thumbnail">Image url</label>
                        <input type="thumbnail" id="thumbnail"
name="thumbnail" class="form-control"
                            [(ngModel)]="newThumbnail"
#newThumbnailField="ngModel" required />
                        <div *ngIf="newThumbnailField.invalid &&
(newThumbnailField.dirty || newThumbnailField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newThumbnailField.errors?.['required']">Image url is required.</div>
                        </div>
                    </div>


                    <button class="btn btn-primary m-2" type="button"
[disabled]="newMovieForm.invalid"
                        (click)="submitNewMovie(newTitle, newExtract, newYear,
newCast, newGenres, newThumbnail)">Submit</button>
                    <button class="btn btn-secondary  m-2" type="button"
                        (click)="newMovieForm.resetForm({})">Reset</button>
                </form>
            </div>
        </div>
    </div>
    <div class="row m-1">
        <div class="card col-lg-4 m-auto">
            <div class="card-header" align="center">
                <h4>Add an actor</h4>
            </div>
            <div class="card-body">
                <form #newActorForm="ngForm" appIdentityRevealed
name="newActorForm">
                    <div class="form-group">
                        <label for="newActorName">Name</label>
                        <input type="text" id="newActorName"
name="newActorName" class="form-control" required
                            minlength="4" [(ngModel)]="newActorName"
#newActorNameField="ngModel" />
                        <div *ngIf="newActorNameField.invalid &&
(newActorNameField.dirty || newActorNameField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newActorNameField.errors?.['required']">Name is required.</div>
                            <div
*ngIf="newActorNameField.errors?.['minlength']">
```

```
                                Name must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="birthYear">Birth Year</label>
                        <input type="number" id="birthYear" name="birthYear"
class="form-control" required minlength="4"
                            [(ngModel)]="birthYear"
#newBirthYearField="ngModel" />
                        <div *ngIf="newBirthYearField.invalid &&
(newBirthYearField.dirty || newBirthYearField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newBirthYearField.errors?.['required']">Birth Year is required.</div>
                            <div
*ngIf="newBirthYearField.errors?.['minlength']">
                                Name must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="deathYear">Death Year</label>
                        <input type="number" id="deathYear" name="deathYear"
class="form-control" minlength="4"
                            [(ngModel)]="deathYear"
#newDeathYearField="ngModel" />
                        <div *ngIf="newDeathYearField.invalid &&
(newDeathYearField.dirty || newDeathYearField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newDeathYearField.errors?.['minlength']">
                                Name must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <div class="form-group">
                        <label for="image">Image Url</label>
                        <input type="text" id="image" name="image"
class="form-control" required minlength="4"
                            [(ngModel)]="image" #newImageField="ngModel" />
                        <div *ngIf="newImageField.invalid &&
(newImageField.dirty || newImageField.touched)"
                            class="alert alert-danger m-1">
                            <div
*ngIf="newImageField.errors?.['required']">Image url is required.</div>
                            <div *ngIf="newImageField.errors?.['minlength']">
                                Name must be a minimum length of 4.
                            </div>
                        </div>
                    </div>

                    <button class="btn btn-primary m-2" type="button"
[disabled]="newActorForm.invalid"
                        (click)="submitNewActor(newActorName, birthYear,
deathYear, image)">Submit</button>
                    <button class="btn btn-secondary  m-2" type="button"
                        (click)="newActorForm.resetForm({})">Reset</button>
                </form>
            </div>
        </div>
    </div>
    <div class="row m-1">
        <div class="card col-md-12 m-auto">
            <div class="row">
                <div class="card-header col-sm-12 mb-2" align="center">
                    <h4>Search for a user</h4>
                </div>
            </div>
            <div class="row m-2">
                <div class="d-flex justify-content-center m-auto">
                    <form class="col-sm-8 d-flex">
                        <div class="input-group">
                            <input id="search" type="userSearch" class="form-
control" name="userSearch"
                                placeholder="Search for a user..."
(keyup.enter)="getUserSearchResult(userSearchName)"
                                [(ngModel)]="userSearchName">
                            <button class="btn btn-primary"
(click)="getUserSearchResult(userSearchName)"><i
                                    class="fa fa-search"></i></button>
                        </div>
                    </form>
                </div>
            </div>
            <div class="row" *ngIf="show">
                <ng-container *ngFor="let user of userList; let i=index">
                    <div class="col-lg-12">
                        <div class="card mb-3" style="background-color: white
!important; color: #9A1750;">
                            <div class="card-header" style="text-align:
center; font-size: larger; font-weight: bold;">
```

```
                                {{user.username}}
                            </div>
                            <div class="card-body">
                                <div class="row">
                                    <h5>Name: {{user.name}}</h5>
                                </div>
                                <div class="row">
                                    <h5>Email: {{user.email}}</h5>
                                </div>
                                <div class="row">
                                    <h5>Admin: {{user.admin}}</h5>
                                </div>

                                <div class="row m-2 movie-content p-2"
style="background-color: #9A1750; color: white;">
                                    <h5 class="m-2">Favourites</h5>
                                    <ng-container *ngFor="let movie of
user.favourites">
                                        <div class="col-sm-2 m-auto">
                                            <img src={{movie.thumbnail}}
alt=""
                                                style="min-height: 280px; max-
height: 280px;"
                                                [routerLink]="['../movie',
movie._id]">
                                        </div>
                                    </ng-container>
                                </div>

                                <div class="row m-2 movie-content p-2"
style="background-color: #9A1750; color: white;">
                                    <h5 class="m-2">Watchlist</h5>
                                    <ng-container *ngFor="let movie of
user.watchlist">
                                        <div class="col-sm-2 m-auto">
                                            <img src={{movie.thumbnail}}
alt=""
                                                style="min-height: 280px; max-
height: 280px;"
                                                [routerLink]="['../movie',
movie._id]">
                                        </div>
                                    </ng-container>
                                </div>


                                <div class="row m-2 movie-content p-2">
                                    <h5 class="m-2">Following</h5>
                                    <ng-container *ngFor="let actor of 
user.following">
                                        <div class="col-sm-4 m-auto">
                                            <button class="btn btn-primary w-
50 m-auto"
                                                [routerLink]="['../actors', 
actor._id]">{{actor.primaryName}}</button>
                                        </div>
                                    </ng-container>
                                </div>
                            </div>
                            <div class="card-footer" style="color: white;">
                                <div class="row p-3">
                                    <div class="row m-2">
                                        <button type="button" class="btn btn-
primary w-50 m-auto"
                                            (click)="user.toggled = 
!user.toggled">Edit user</button>
                                    </div>
                                    <div class="row m-2">
                                        <button type="button" class="btn btn-
primary w-50 m-auto"
                                            (click)="deleteUser(user._id)">Del
ete user</button>
                                    </div>
                                    <div class="row mb-3" *ngIf="user.toggled" 
style="color: #9A1750;">
                                        <form #editUserForm="ngForm" 
appIdentityRevealed name="editUserForm">
                                            <div class="form-group">
                                                <label for="name">Name</label>
                                                <input type="text" 
id="name{{i}}" name="name" class="form-control"
                                                    required minlength="4" 
[(ngModel)]="newName"
                                                    #newNameField="ngModel" />
                                                <div 
*ngIf="newNameField.invalid && (newNameField.dirty || newNameField.touched)"
                                                    class="alert alert-danger 
m-1">
                                                    <div 
*ngIf="newNameField.errors?.['required']">Name is required.
                                                    </div>
                                                    <div 
*ngIf="newNameField.errors?.['minlength']">
                                                        Name must be a minimum 
length of 4.
                                                    </div>
                                                    </div>
                                                </div>

                                                <div class="form-group">
                                                    <label
for="username">Username</label>
                                                    <input type="text"
id="username{{i}}" name="username"
                                                        class="form-control"
required minlength="4"
                                                        [(ngModel)]="newUsername"
#newUsernameField="ngModel" />
                                                    <div
*ngIf="newUsernameField.invalid && (newUsernameField.dirty ||
newUsernameField.touched)"
                                                        class="alert alert-danger
m-1">
                                                        <div
*ngIf="newUsernameField.errors?.['required']">Username is
                                                            required.</div>
                                                        <div
*ngIf="newUsernameField.errors?.['minlength']">
                                                            Name must be a minimum
length of 4.
                                                        </div>
                                                    </div>
                                                </div>

                                                <div class="form-group">
                                                    <label
for="password">Password</label>
                                                    <input type="password"
id="password{{i}}" name="password"
                                                        class="form-control"
required
                                                        pattern="^(?=.*[a-
z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$"
                                                        [(ngModel)]="newPassword"
#newPasswordField="ngModel" />
                                                    <div
*ngIf="newPasswordField.invalid && (newPasswordField.dirty ||
newPasswordField.touched)"
                                                        class="alert alert-danger
m-1">
                                                        <div
*ngIf="newPasswordField.errors?.['required']">Password is
                                                            required.</div>
                                                        <div
*ngIf="newPasswordField.errors?.['pattern']">
                                                            Password must be a
minimum of eight characters, at least one
                                                            uppercase letter, one
                                                            lowercase
                                                            letter, one number and
one special character.
                                                        </div>
                                                    </div>
                                                </div>


                                                <div class="form-group">
                                                    <label
for="email">Email</label>
                                                    <input type="email"
id="email{{i}}" name="email" class="form-control"
                                                        pattern="[a-z0-9._%+-
]+@[a-z0-9.-]+\.[a-z]{2,4}$"
                                                        [(ngModel)]="newEmail"
#newEmailField="ngModel" required />
                                                    <div
*ngIf="newEmailField.invalid && (newEmailField.dirty ||
newEmailField.touched)"
                                                        class="alert alert-danger
m-1">
                                                        <div
*ngIf="newEmailField.errors?.['required']">Email is required.
                                                        </div>
                                                        <div
*ngIf="newEmailField.errors?.['pattern']">
                                                            Please enter a valid
email.
                                                        </div>
                                                    </div>
                                                </div>


                                                <div class="form-group">
                                                    <label
for="admin">Admin</label><br>
                                                    <input class="form-check-
input" type="checkbox" id="admin{{i}}"
                                                        [checked]="newAdmin"
(change)="newAdmin = !newAdmin">
                                                </div>
                                                <button class="btn btn-primary m-
2" type="button"
```

```
                                                [disabled]="editUserForm.inval
id"
                                                (click)="submitEditUser(user._
id, newUsername, newName, newPassword, newEmail, newAdmin, user.favourites,
user.watchlist, user.following)">Submit</button>
                                            <button class="btn btn-
secondary  m-2" type="button"
                                                (click)="editUserForm.resetFor
m({})">Reset</button>
                                        </form>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </ng-container>
            </div>
        </div>
    </div>
</div>
```

admin/admin.component.ts –

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { WebService } from '../web.service';
import { User } from '../types/user.type';
import { Actor } from '../types/actor.type';
import { Movie } from '../types/movie.type';

@Component({
  selector: 'app-admin',
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.css']
})
export class AdminComponent {
  userList!: User[];
  newName = '';
  newUsername = '';
  newPassword = '';
  newEmail = '';
  userSearchName = '';
  newAdmin = false;
  newTitle = '';
  newExtract = '';
  newYear!: number;
  newCast!: string;
  newGenres!: string;
  newThumbnail = '';
  newActorName = '';
  birthYear!: number;
  deathYear!: number;
  image = '';
  show = true
  pageNo = 1;
  editUserToggle = false;
  admin = (sessionStorage.getItem("admin") === 'true');


  constructor(public webService: WebService, private router: Router) {
  }

  reload = () => {
    this.show = false;

    setTimeout(() => this.show = true);
  }

  submitNewUser = (username: string, name: string, password: string, email:
string, admin: boolean,) => {
    this.webService.userService.addNewUser(username, name, password, email,
admin).subscribe(
      () => {
        this.webService.userService.getUsers(this.pageNo)
        window.location.reload()
      }
    )
  }

  submitEditUser = (userId: string, username: string, name: string, password:
string, email: string, admin: boolean, favourites: Array<Movie>, watchlist:
Array<Movie>, following: Array<Actor>) => {
    const favourite_ids = favourites.map((movie) => movie._id);
    const watchlist_ids = watchlist.map((movie) => movie._id);
    const following_ids = following.map((actor) => actor._id)

    this.webService.userService.editExistingUser(userId, username, name,
password, email, admin, favourite_ids, watchlist_ids,
following_ids).subscribe(
      () => {
        this.webService.userService.getUsers(this.pageNo)
        window.location.reload()
      }
    )
  }
  submitNewMovie = (newTitle: string, newExtract: string, newYear: number,
newCast: string, newGenres: string, newThumbnail: string) => {
    this.webService.movieService.addNewMovie(newTitle, newExtract, newYear,
newCast, newGenres, newThumbnail).subscribe(
      (response) => {
        this.router.navigate([`../movie/${response.url.split('movies/')[1]}`])
      }
    )
  }

  submitNewActor = (name: string, birthYear: number, deathYear: number, image:
string) => {
    this.webService.actorService.addNewActor(name, birthYear, deathYear,
image).subscribe(
      () => {
        window.location.reload()
      }
    )
  }

  getUserSearchResult = (searchName: string) => {
    this.webService.userService.getUserByName(searchName).subscribe((response)
=> {
      this.userList = response;
    })
    this.reload()
  }

  deleteUser = (id: string) => {
    if (confirm('Do you want to submit?')) {
      this.webService.userService.deleteUser(id).subscribe(
        () => {
          this.webService.userService.getUsers(this.pageNo)
          window.location.reload()
        }
      )
    }
  }

  ngOnInit() {
    if (this.admin) {
      this.webService.userService.getUsers(this.pageNo).subscribe((response)
=> {
        this.userList = response;
      })
    } else {
      this.router.navigate(['../'])
    }
    }

}
```

actors/actors.component.html –

```
<div class="container mb-2">
    <div class="row">
        <div class="col-sm-12 mb-2" align="center">
            <h2>Search</h2>
        </div>
    </div>
    <div class="row">
        <div class="d-flex justify-content-center">
            <form class="col-sm-6 d-flex">
                <div class="input-group">
                    <input id="search" type="search" class="form-control"
name="search"
                        placeholder="Search for an actor..."
(keyup.enter)="getSearchResult()"
                        [(ngModel)]="actorSearchName">
                    <button type=" button" class="btn btn-primary"
(click)="getSearchResult()"><i
                            class="fa fa-search"></i></button>
                </div>
                <button class="btn btn-primary" (click)="resetSearch()"
style="margin-left: 2px;">Reset</button>
            </form>
        </div>
    </div>
    <div class="row pt-4">
        <div class="d-flex justify-content-center m-auto">
            <div class="col-lg-3 text-center">
                <div class="dropdown">
                    <button class="btn btn-primary dropdown-toggle m-2"
type="button" id="dropdownMenuButton"
                        data-bs-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
                        Decade
                    </button>
                    <div class="dropdown-menu" aria-
labelledby="dropdownMenuButton">
                        <ng-container *ngFor="let decade of this.decadeList">
                            <div class="form-check" style="margin-left:
1rem;">
                                <input id="decade-{{decade}}" [value]='decade'
type="radio" name="decades"
                                    class="form-check-input"
[(ngModel)]="checkedDecade">
                                <label class=" form-check-label"
for="dropdownCheck2">
```

```
                                    {{decade}}
                                </label>
                            </div>
                        </ng-container>
                    </div>
                </div>
            </div>
        </div>
        <div class="row pt-2 m-auto">
            <button class="btn btn-primary w-25 m-auto" type="button"
(click)="applyFilter()"> Apply Filter </button>
        </div>
        <div class="container">
            <div class="row">
                <div class="col-sm-12 mb-2" align="center">
                    <h2>Top Picks</h2>
                </div>
            </div>
            <div class="row" *ngIf="show">
                <ng-container *ngFor="let actor of actorList">
                    <div class="col-lg-6">
                        <div class="card mb-3">
                            <div class="card-header"
                                style="color: #9A1750; text-align: center;
font-size: larger; font-weight: bold;">
                                {{ actor.primaryName }} ({{actor.birthYear}})
                            </div>
                            <div class="card-body" style="display: flex;">
                                <img src={{actor.image}} alt="" style="margin:
auto; max-height: 350px;">
                            </div>
                            <div class="card-footer" style="color: white;">
                                <div class="row p-3">
                                    <button class="btn btn-primary w-50 m-
auto" [routerLink]="[actor._id]">More
                                        Info</button>
                                </div>
                            </div>
                        </div>
                    </div>
                </ng-container>
            </div>
            <div class="row pb-5" *ngIf="this.actorSearchName.length == 0">
                <div class="col-lg-12" style="display: flex;">
                    <div class="col-sm-4">
                        <button class="btn btn-primary w-100"
(click)="getPrevPage()"
                            *ngIf="this.pageNo > 1">Previous</button>
                    </div>
                    <div class="col-sm-4">
                        <div class="text-center">
                            <ng-container *ngFor="let page of
pageList"><button class="btn btn-primary"
                                    style="margin-right: 4px; margin-top:
2px;"
                                    (click)="getSpecificPage(page)">{{page}}</
button>
                            </ng-container>
                        </div>
                    </div>
                    <div class="col-sm-4">
                        <button class="btn btn-primary w-100"
(click)="getNextPage()">Next</button>
                    </div>
                </div>
            </div>
        </div>
```

actors/actors.component.ts –

```
import { Component } from '@angular/core';
import { WebService } from '../web.service';
import { Actor } from '../types/actor.type';

@Component({
  selector: 'app-actors',
  templateUrl: './actors.component.html',
  styleUrls: ['./actors.component.css']
})
export class ActorsComponent {
  actorSearchName = '';
  actorList!: Actor[];
  pageList = [1, 2, 3, 4, 5, 300, 600]
  decadeList = [1900, 1910, 1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990,
2000, 2010, 2020];
  checkedDecade = '';
  pageNo = 1;
  show: boolean = true

  constructor(public webService: WebService) {
  }


  reload = () => {
    this.show = false;
    setTimeout(() => this.show = true);
  }
  resetSearch = () => {
    this.actorSearchName = ''
    this.webService.actorService.getActors(this.pageNo).subscribe((response)
=> {
      this.actorList = response
      this.reload()

    })
  }

  getSearchResult = () => {
    this.webService.actorService.getActorByName(this.actorSearchName).subscrib
e((response) => {
      this.actorList = response
      this.pageNo = 1;
      this.reload()
    })
  }

  buildArgs = () => {
    let args = `?pn=${this.pageNo}`
    if (this.checkedDecade) {
      args += `&decade=${this.checkedDecade}`
    }
    return args
  }

  getPrevPage = () => {
    if (this.pageNo > 1) {
      this.pageNo -= 1
      const args = this.buildArgs()
      this.webService.actorService.getActorsMultipleArgs(args).subscribe((resp
onse) => {
        this.actorList = response
        this.reload()
      })
    }
  }

  getNextPage = () => {
    this.pageNo += 1
    const args = this.buildArgs()
    this.webService.actorService.getActorsMultipleArgs(args).subscribe((respon
se) => {
      this.actorList = response
      this.reload()
    })
  }

  getSpecificPage = (page: number) => {
    this.pageNo = page
    const args = this.buildArgs()
    this.webService.actorService.getActorsMultipleArgs(args).subscribe((respon
se) => {
      this.actorList = response
      this.reload()
    })
  }

  applyFilter = () => {
    this.webService.actorService.getActorByDecade(this.checkedDecade).subscrib
e((response) => {
      this.actorList = response
      this.pageNo = 1;
      this.reload()
    })
  }


  ngOnInit() {
    this.webService.actorService.getActors(this.pageNo).subscribe((response)
=> {
      this.actorList = response;
    })
  }

}
```

actor/actor.component.html –

```
<div class="container" *ngIf="actor">
    <div class="row">
        <div class="col-lg-3 mb-2">
            <img style="max-width: 200px;" src={{image}} alt="">
        </div>
        <div class="col-lg-9">
            <div class="row mb-2">
                <div class="col-lg-6">
                    <h2>{{actor.primaryName}}</h2>
                </div>
                <div class="col-sm-8 mt-2" *ngIf="username && !following">
                    <button type="button" class="btn btn-primary"
(click)="follow()">Follow</button>
                </div>
                <div class="col-sm-8 mt-2" *ngIf="username && following">
```

```
                    <button type="button" class="btn btn-primary"
(click)="unfollow()">Unfollow</button>
                </div>
            </div>
            <div class="row">
                <div class="col-sm-1">
                    <h6>{{actor.birthYear}}</h6>
                </div>
            </div>
        </div>
        <p>{{extract}}</p>
    </div>
    <div class="row mb-3" *ngIf="admin">
        <div class="col-sm-2 mt-2">
            <button type="button" class="btn btn-primary"
(click)="deleteActor(actor._id)">Delete actor</button>
        </div>
    </div>
</div>
```

actor/actor.component.ts –

```
import { Component } from '@angular/core';
import { ActivatedRoute, ParamMap, Router } from '@angular/router';
import { WebService } from '../web.service';
import wiki from 'wikipedia'
import { Actor } from '../types/actor.type';
import { User } from '../types/user.type';

@Component({
  selector: 'app-actor',
  templateUrl: './actor.component.html',
  styleUrls: ['./actor.component.css']
})
export class ActorComponent {
  id!: string;
  actor!: Actor;
  user!: User;
  username = sessionStorage.getItem("username");
  admin = (sessionStorage.getItem("admin") === 'true');
  following = false;
  extract = '';
  image = '';
  constructor(public webService: WebService, private route: ActivatedRoute,
private router: Router) {
  }

  wikiSearch = async (name: string) => {
    return (await wiki.search(name)).results[0].title
  }

  getContent = async (title: string) => {
    return (await wiki.summary(title)).extract

  }

  getImage = async (title: string) => {
    return ((await wiki.summary(title)).originalimage.source)
  }

  deleteActor = (actorId: string) => {
    if (confirm('Are you sure you want to delete this actor? (this cannot be
reversed)')) {
      this.webService.actorService.deleteActor(actorId).subscribe(
        () => {
          this.router.navigate([`../`])
        }
      )
    }

  }

  follow = () => {
    this.webService.actorService.follow(this.user._id, this.id).subscribe(()
=> {
      window.location.reload()
    })
  }

  unfollow = () => {
    this.webService.actorService.unfollow(this.user._id, this.id).subscribe(()
=> {
      window.location.reload()
    })
  }

  ngOnInit(): void {
    this.route.paramMap.subscribe((params: ParamMap) => {
      this.id = params.get('id')!;
    });

    this.webService.actorService.getActorById(this.id).subscribe(async
(response) => {
      const actor: Actor = response;
      this.actor = actor
      const search = await this.wikiSearch(`Actor ${actor.primaryName}`)

      this.extract = await this.getContent(search)

      this.image = await this.getImage(search)
    })

    if (this.username) {
      this.webService.userService.getUserByUsername(this.username!).subscribe(
(response) => {
        this.user = response[0]

        this.user.following.map((actor: { _id: string; }) => {
          if (this.id == actor._id) {
            this.following = true
          }
        });
      })

    }
  }
}
```