myFlix web app

Angular frontend development process

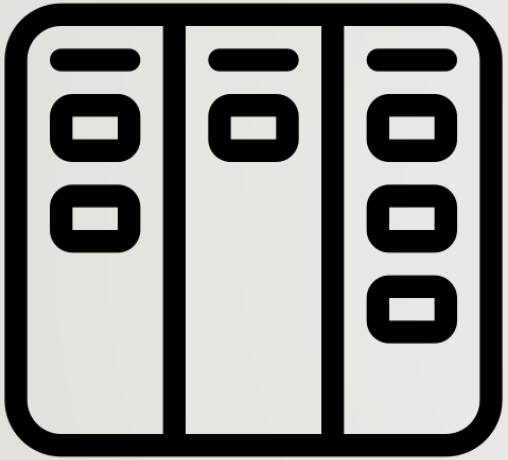
September	2023

*All usernames, passwords or other similar information shown in this presentation is fictive and for illustrative purpose only

TABLE OF CONTENTS

TABLE OF CONTENTS

01



Analyzing the project user stories, **setting up** a Kanban board and **creating** the app user flow

Skills used

Project management
Analytical thinking
Visual communication (schemas)

WHY WAS THIS STEP IMPORTANT

Analyzing the user stories is essential to pinpoint the exact features that need to be implemented in a product, and thus fulfil users expectations.

Knowing how to work with Kanban is also important, since it's a widely used project management method in various work environments. Kanban is interesting since it's based on simple visuals and allows tasks to be presented in a clear and effective way. It is therefore a very useful project management and monitoring tool, especially in coding.

WHAT WAS THE GOAL

Analyzing project each user story (2) and note down the technical requirements and functionalities needed to meet the desired results.

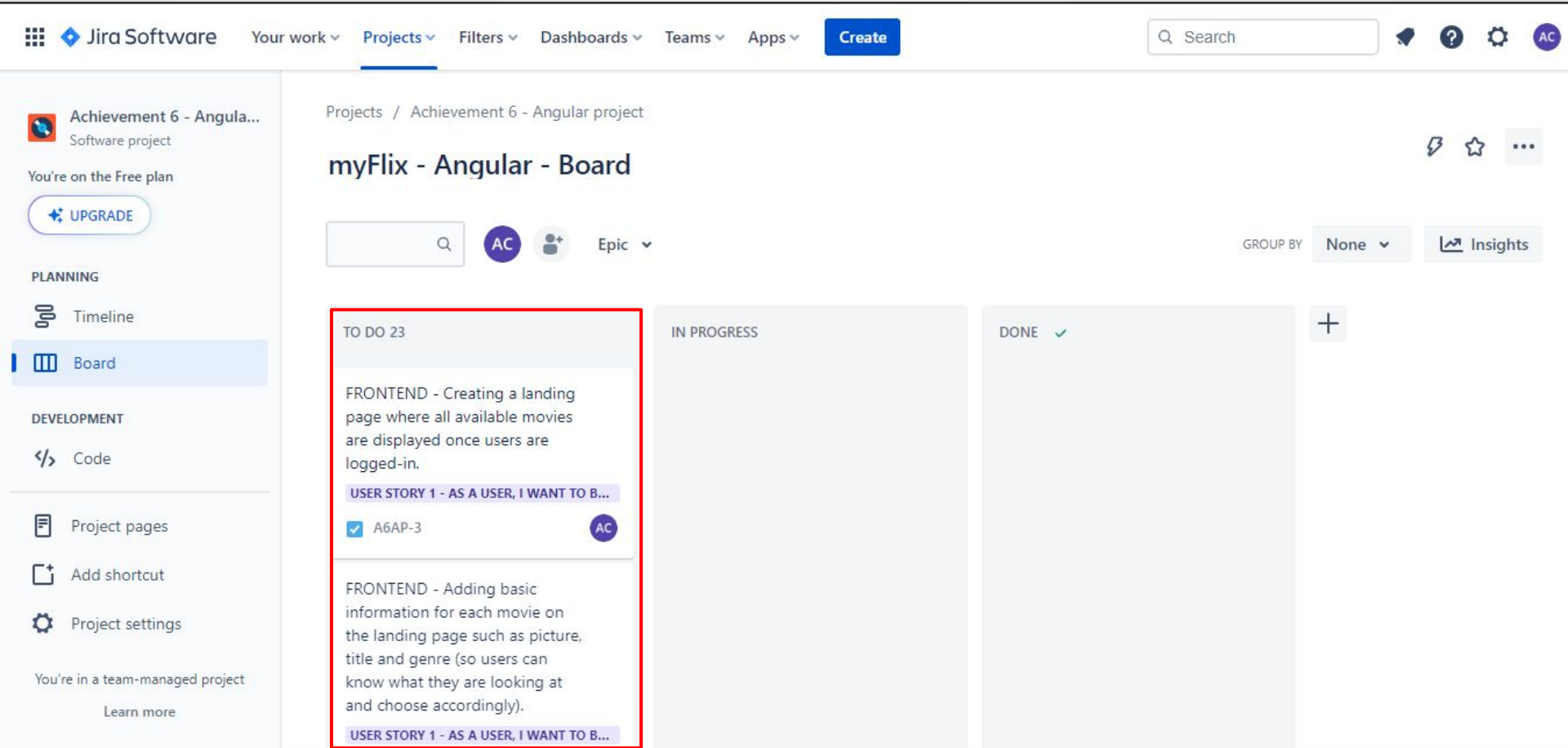
Migrating the technical product requirements extract from the project's user stories into Kanban card tasks (in Jira) to structure and orient the project from the beginning to the end (see images on next slides).

- Adding story points to each card based on the expected effort required.

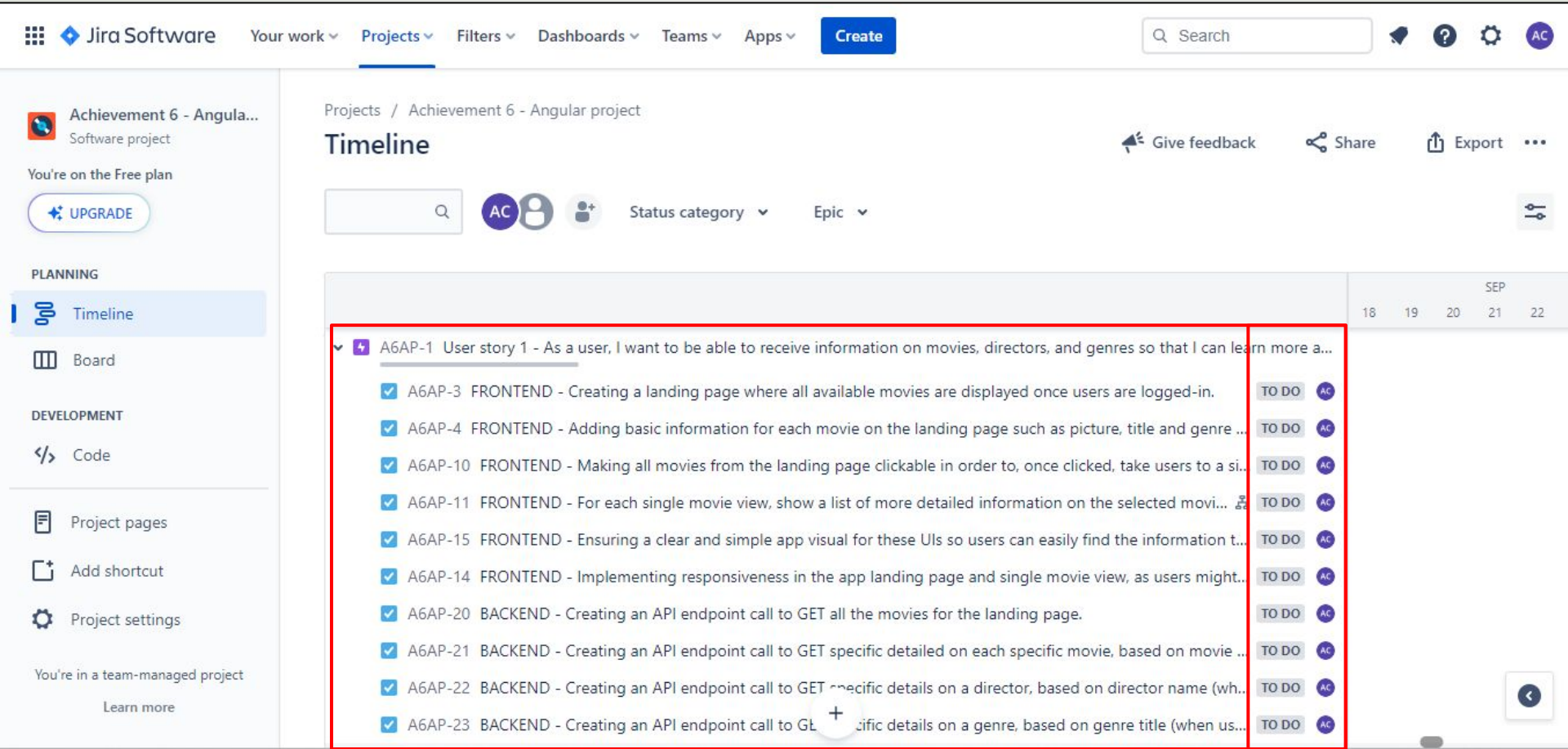
Creating the user flow on myFlix Angular (see images on next slides).

- The goal was to, based on the user stories, showing the steps users have to complete to achieve their end goal while using the web app.

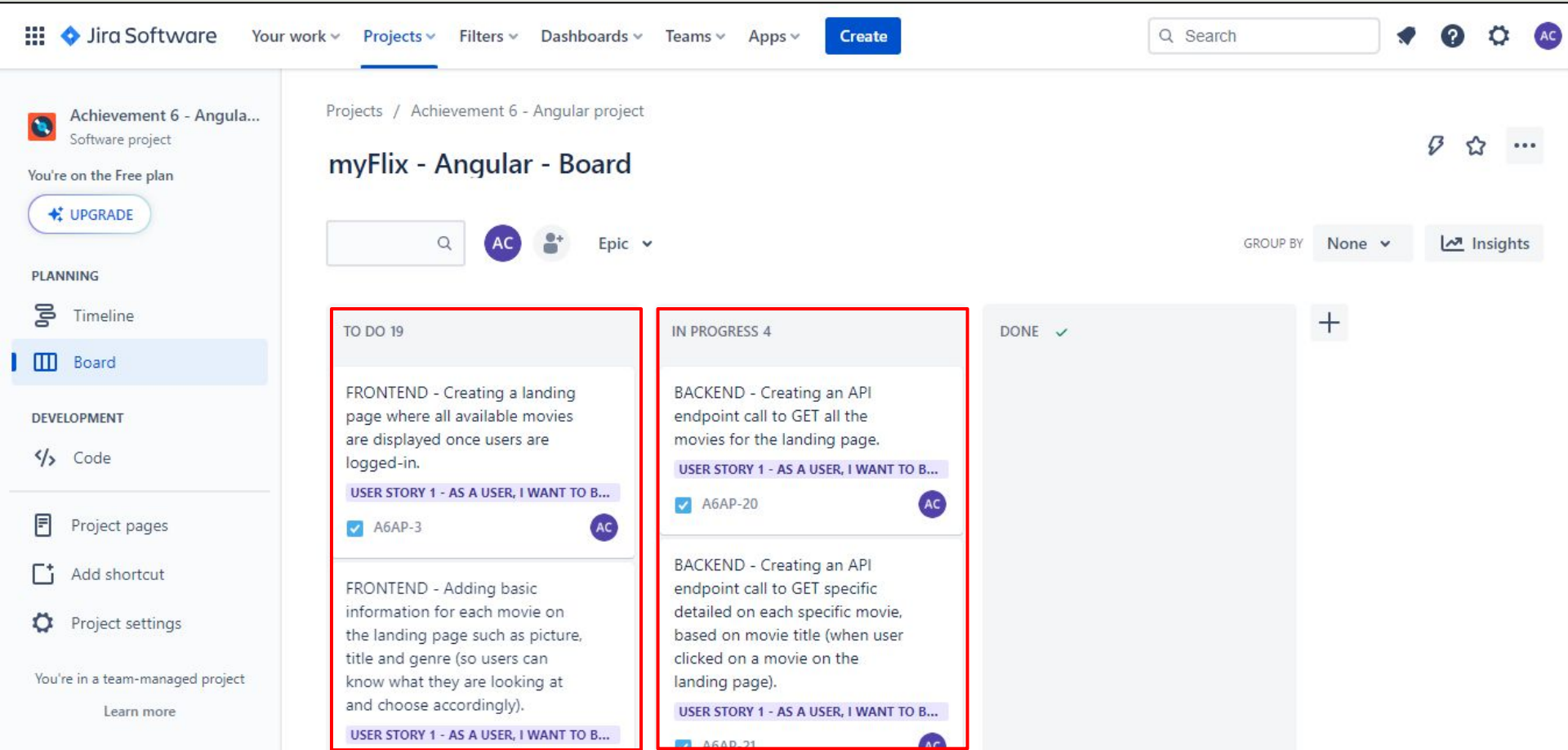
KANBAN BOARD AT THE BEGINNING OF THE PROJECT



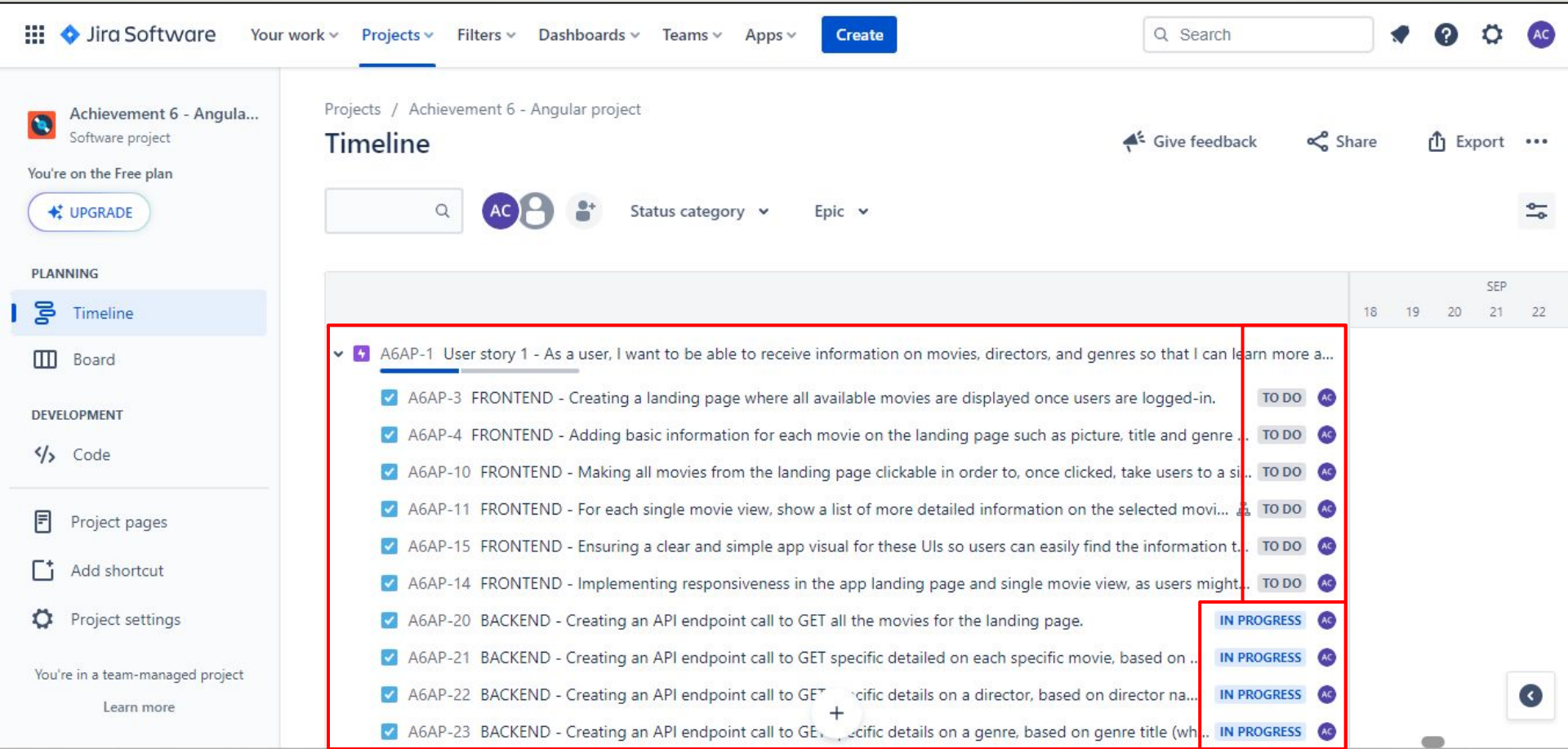
KANBAN TIMELINE AT THE BEGINNING OF THE PROJECT



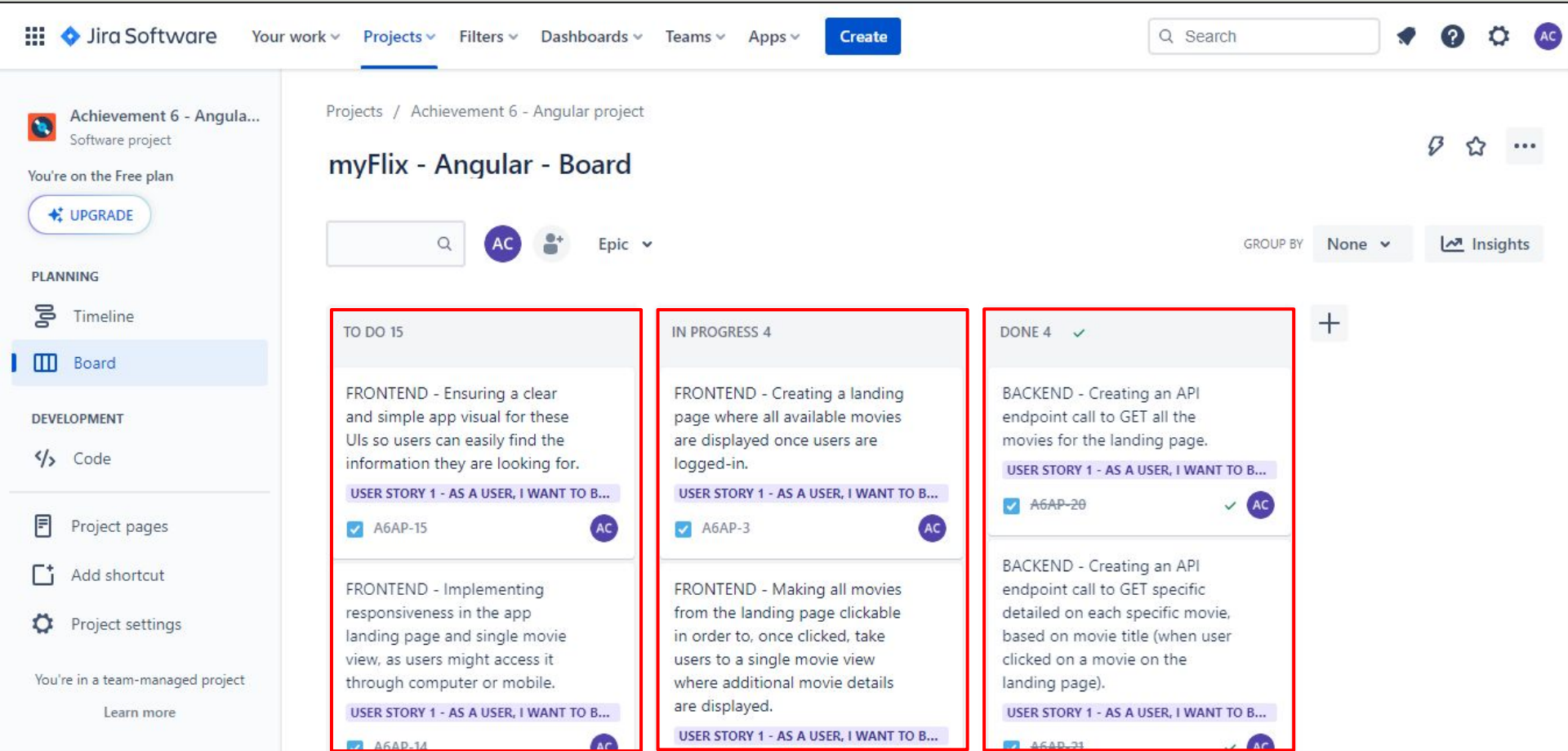
KANBAN BOARD IN PROGRESS



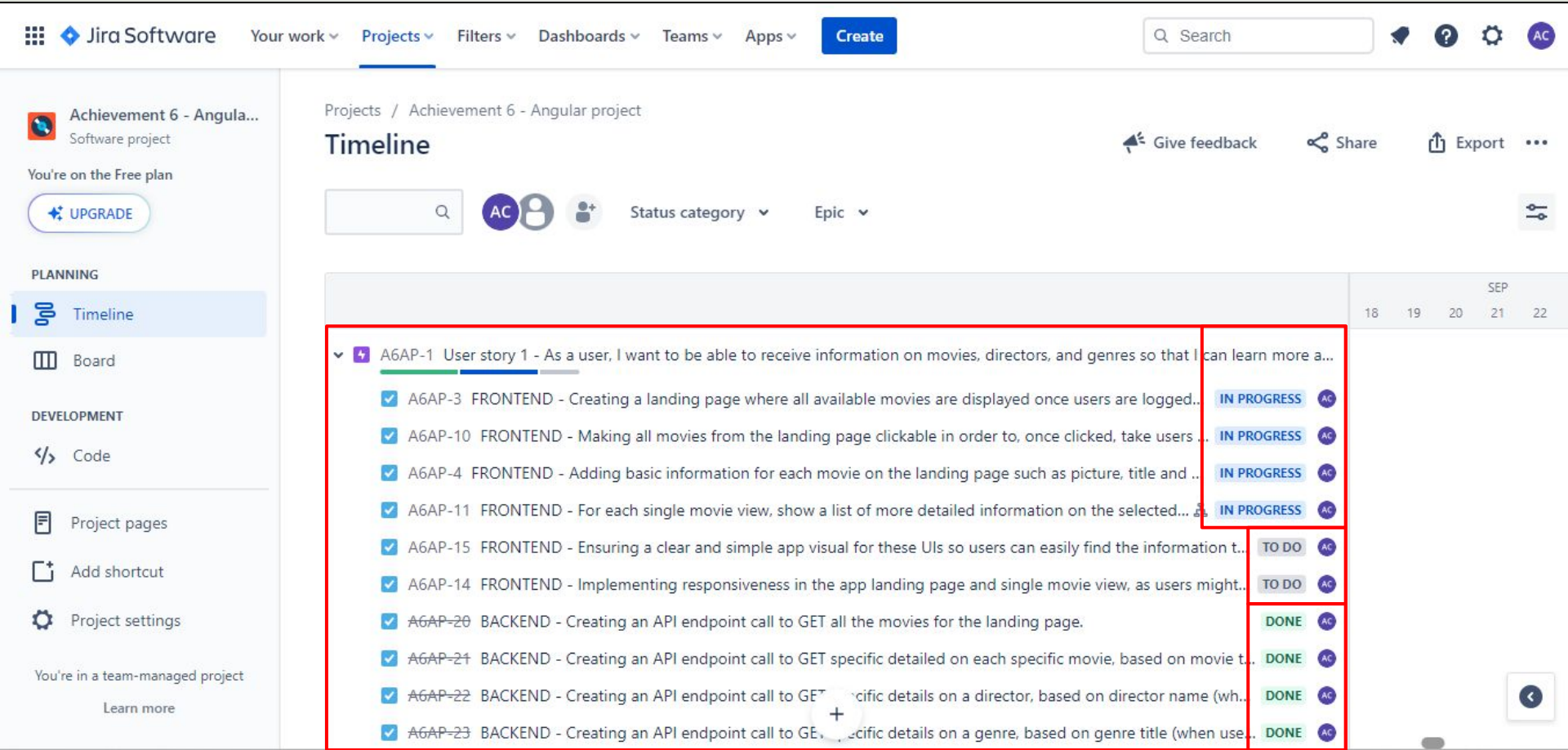
KANBAN TIMELINE IN PROGRESS



KANBAN BOARD IN PROGRESS

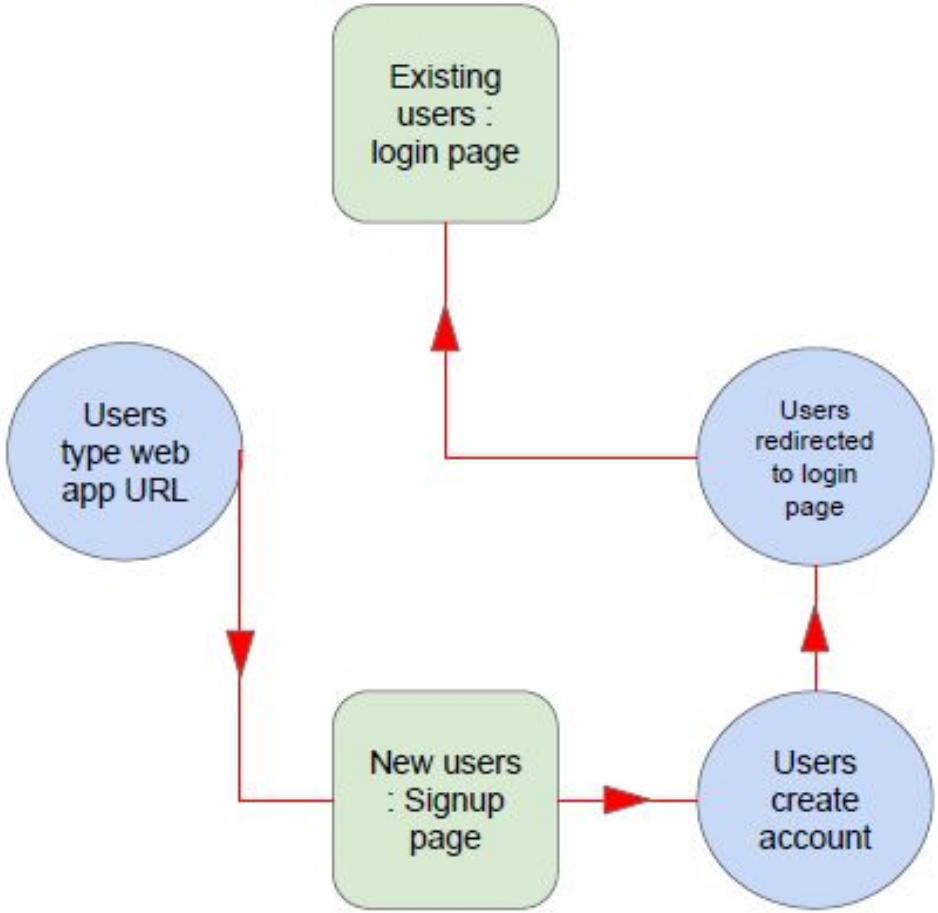


KANBAN TIMELINE IN PROGRESS



User flow 1

As a user, I want to be able to create a profile so I can save data about my favorite movies.



Legend

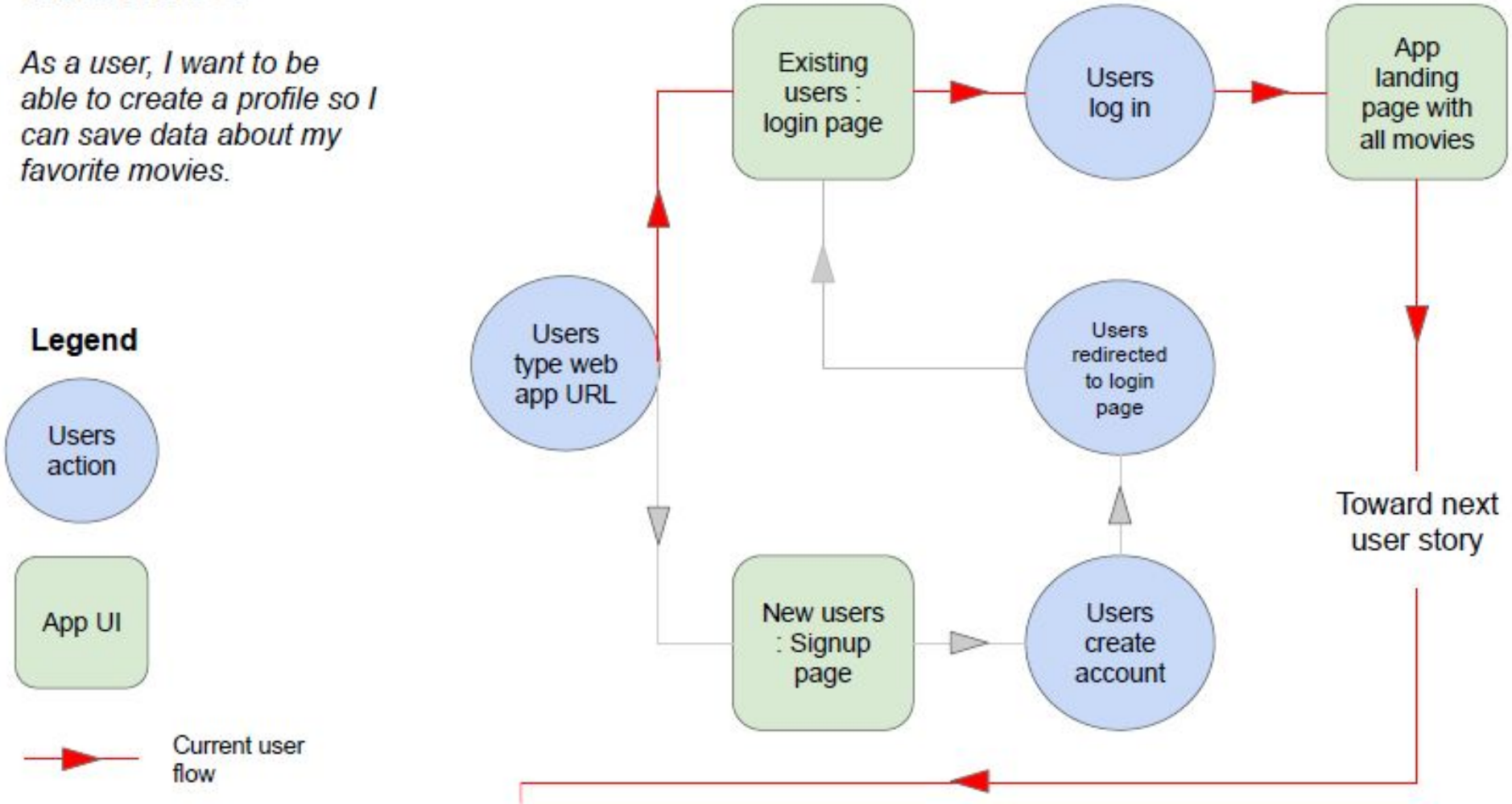
Users action

App UI

Current user flow

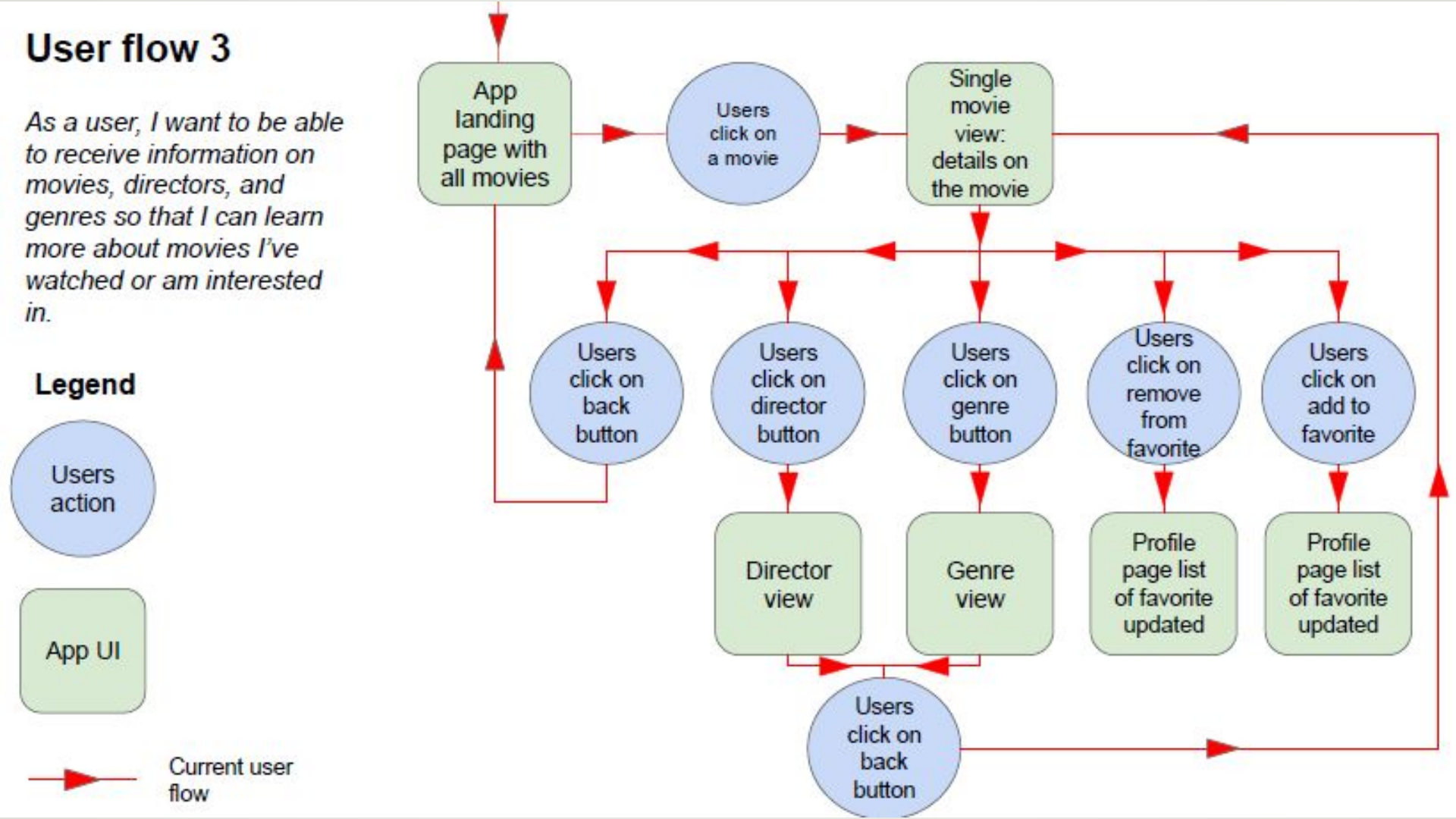
User flow 2

As a user, I want to be able to create a profile so I can save data about my favorite movies.



User flow 3

As a user, I want to be able to receive information on movies, directors, and genres so that I can learn more about movies I've watched or am interested in.



CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

This first step was familiar to me, because I already had experience in handling project management software (e.g.: Asana) as well as analyzing user stories.

DECISION MADE

I chose the project management tool used for the project (in this case - Jira). I also decided how to formulate and present the tasks for the projects in the Jira's cards, and added their story points.

02

Creating the project using Angular CLI, **configuring** the files and **creating** an Angular service file to connect with the API

Skills used

Research
Problem-solving
Debugging

WHY WAS THIS STEP IMPORTANT

Organizing the development and the coding environment for efficient and well-oriented work from the beginning is one of the keys to ensure smooth workflow and limit avoidable time loss due to inefficient project initialization.

As for the Angular service, it was essential to create it to ensure communication between the frontend and the backend of the web app.

WHAT WAS THE GOAL

Creating the project by running *$ ng new myFlix-Angular-client* in the Angular CLI (which is similar to running *create-react-app* in React, except for Angular in this case).

Setting up and configuring Angular HttpClient (with Angular *HttpClientModule*) to ensure the web app is able to communicate and fetch data from the movie API.

Writing codes to load the data from the API:

- In the newly *fetch-api-data.service.ts* created, I've added the necessary imports (HttpClient, HttpHeaders, HttpErrorResponse, etc.).

- I then implemented in this same file the services / codes used to make the calls from myFlix to each endpoint in the backend API (see examples on the next slides).

CODE CALLING THE ENDPOINT ALLOWING TO GET ALL MOVIES

```
getAllMovies(): Observable<any> {
  const token = localStorage.getItem('token');
  return this.http.get(apiUrl + 'movies', {
    headers: new HttpHeaders(
      {
        Authorization: 'Bearer ' + token,
      })
  }).pipe(
    map(this.extractResponseData),
    catchError(this.handleError)
  );
}
```

CODE CALLING THE ENDPOINT ALLOWING TO ADD FAVORITE MOVIES

```
userAddFavoriteMovie(Username: string, MovieID: string): Observable<any> {
  const token = localStorage.getItem('token');
  const headers = new HttpHeaders({
    Authorization: 'Bearer ' + token,
  });
  const requestBody = {};
  return this.http.post(apiUrl + 'users/' + Username + '/movies/' + MovieID, requestBody, { headers }
    .pipe(
      map(this.extractResponseData),
      catchError(this.handleError)
    );
}
```

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

At first, I realized that there was a missing endpoint in the backend, considering the list of calls from the frontend that needed to be implemented. It was therefore impossible to call this specific endpoint from the frontend.

To fix this, I went back to my backend code, and I created a new endpoint. I carried out various tests to ensure that it was working properly. I was then able to connect my frontend service to this new endpoint, thus ensuring to keep meeting the project requirements.

03

Creating user registration and user log in components / UIs

Skills used

Research
Problem-solving
Code writing
Debugging

WHY WAS THIS STEP IMPORTANT

Implementing and rendering a sign up and a log logic / views was important to ensure users have a mean to access the content displayed inside the web app (and reserved to authenticated users).

WHAT WAS THE GOAL

Creating a sign up and log in component using *$ ng generate component user-registration-form* and *$ ng generate component user-login-form*.

Implementing the codes in the newly TypeScript (.ts) file created for the sign up and log in process respectively.

(suite on next slide)

WHAT WAS THE GOAL (SUITE)

Updating the respective HTML file for the sign up and log in respectively, to ensure a simple and clear view is rendered in users' welcome page for both components.

- Angular's binding syntax and directives, such as *NgModel*, have also been added in the HTML to establish communication between the TypeScript file and the HTML files for the sign up and log in process respectively.

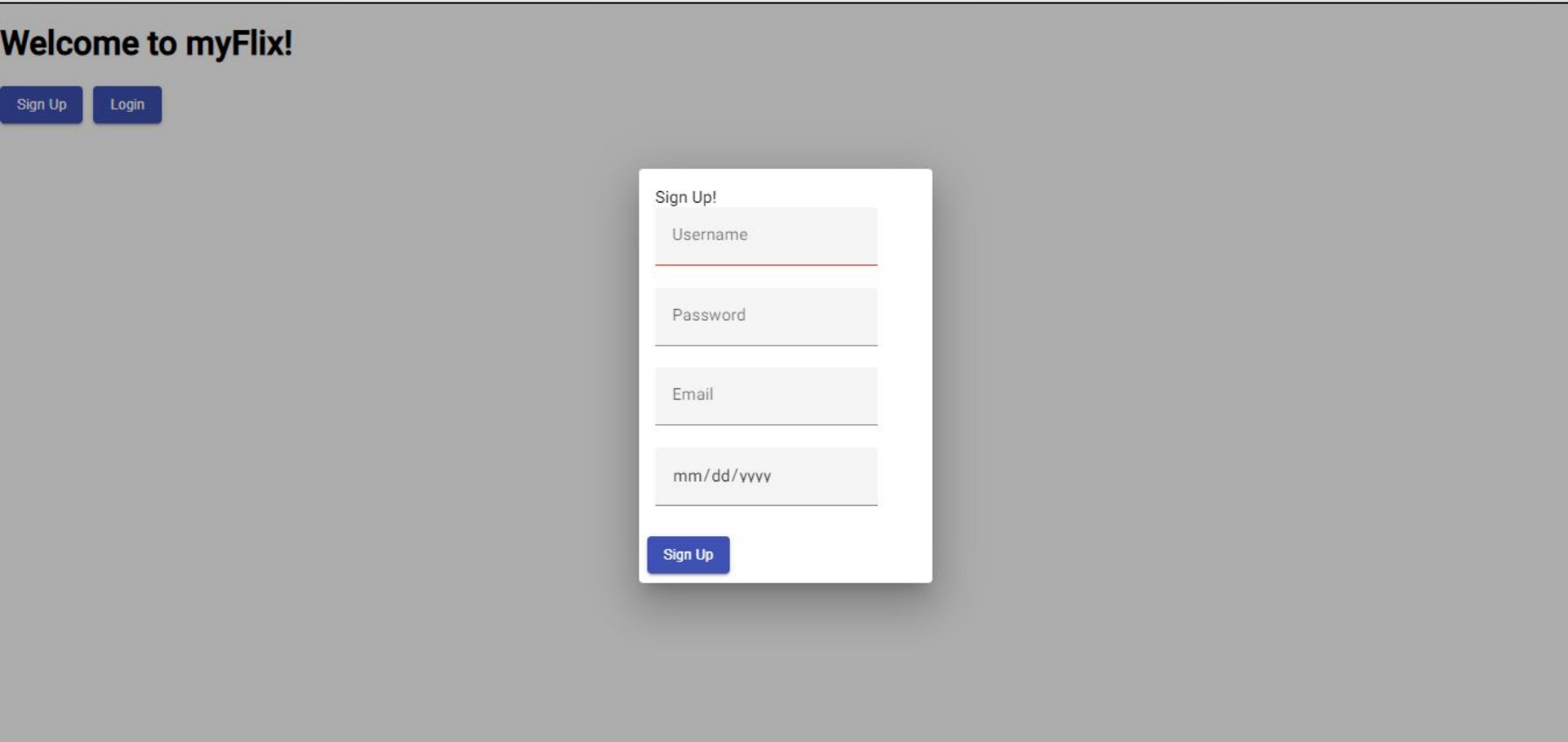
Polishing up the UIs for the welcome page / sign up view / log in view with Angular Material.

- For example, sign up form and log in form in users screen are shown using dialog box from Angular Material (similar to modal in Bootstrap) (see next images).

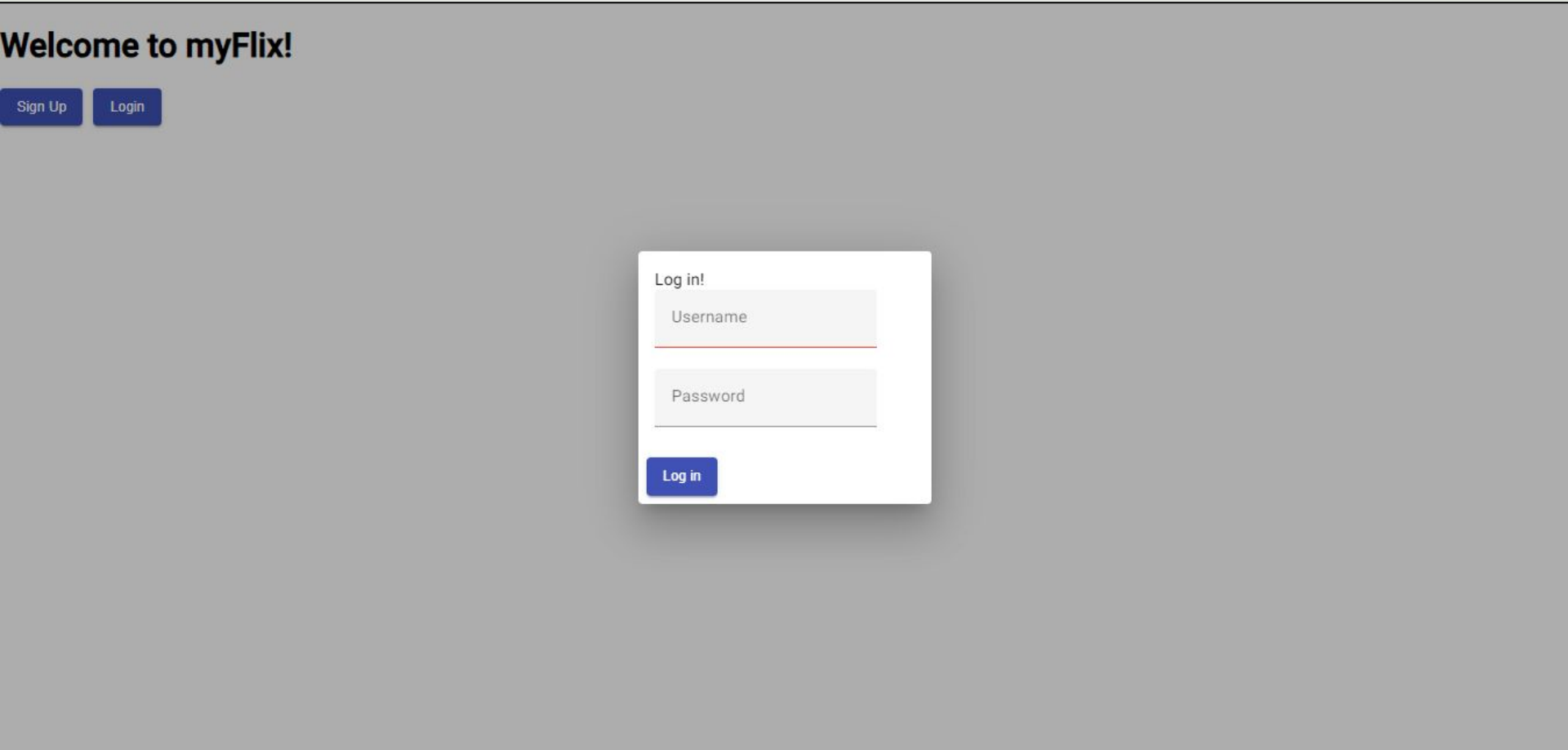
Running tests to ensure users' sign up and log in process are working as expected and that, once logged-in, current user and related token are stored correctly in localStorage.

FIRST RENDERED VIEW - WELCOME PAGE (NO STYLINGS)

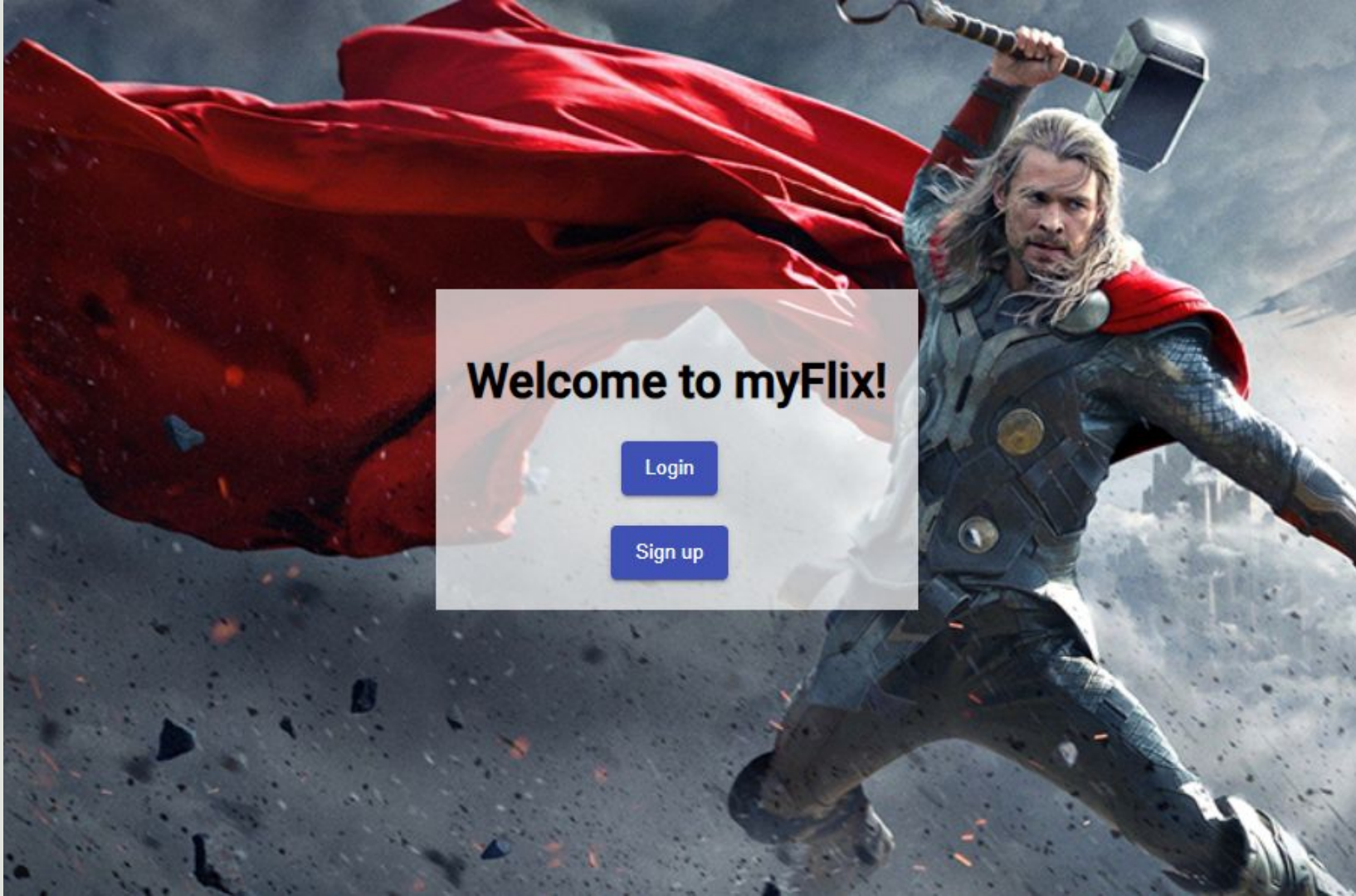
FIRST RENDERED VIEW - SIGN UP FORM (NO STYLING)



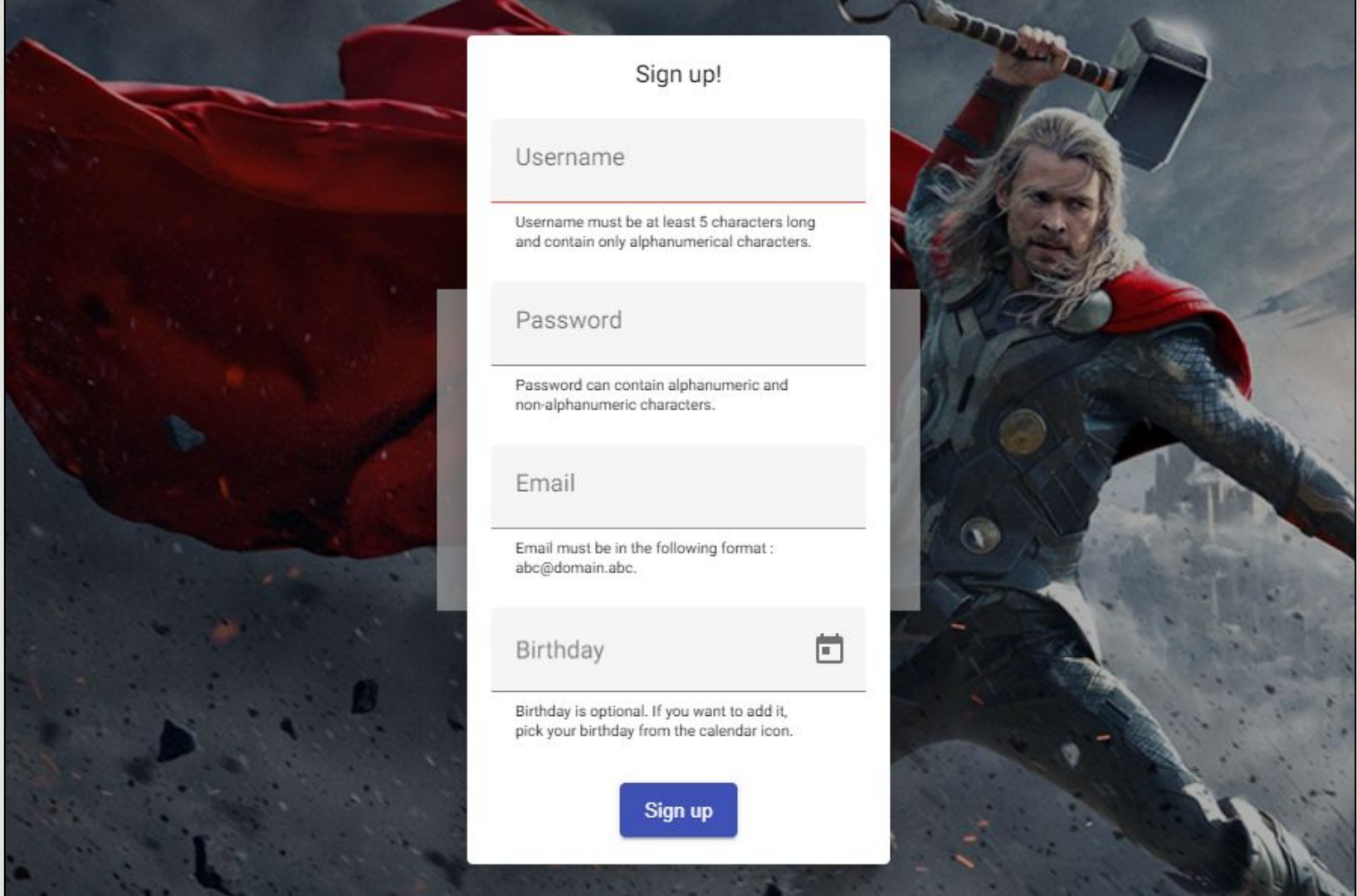
FIRST RENDERED VIEW - LOG IN FORM (NO STYLING)



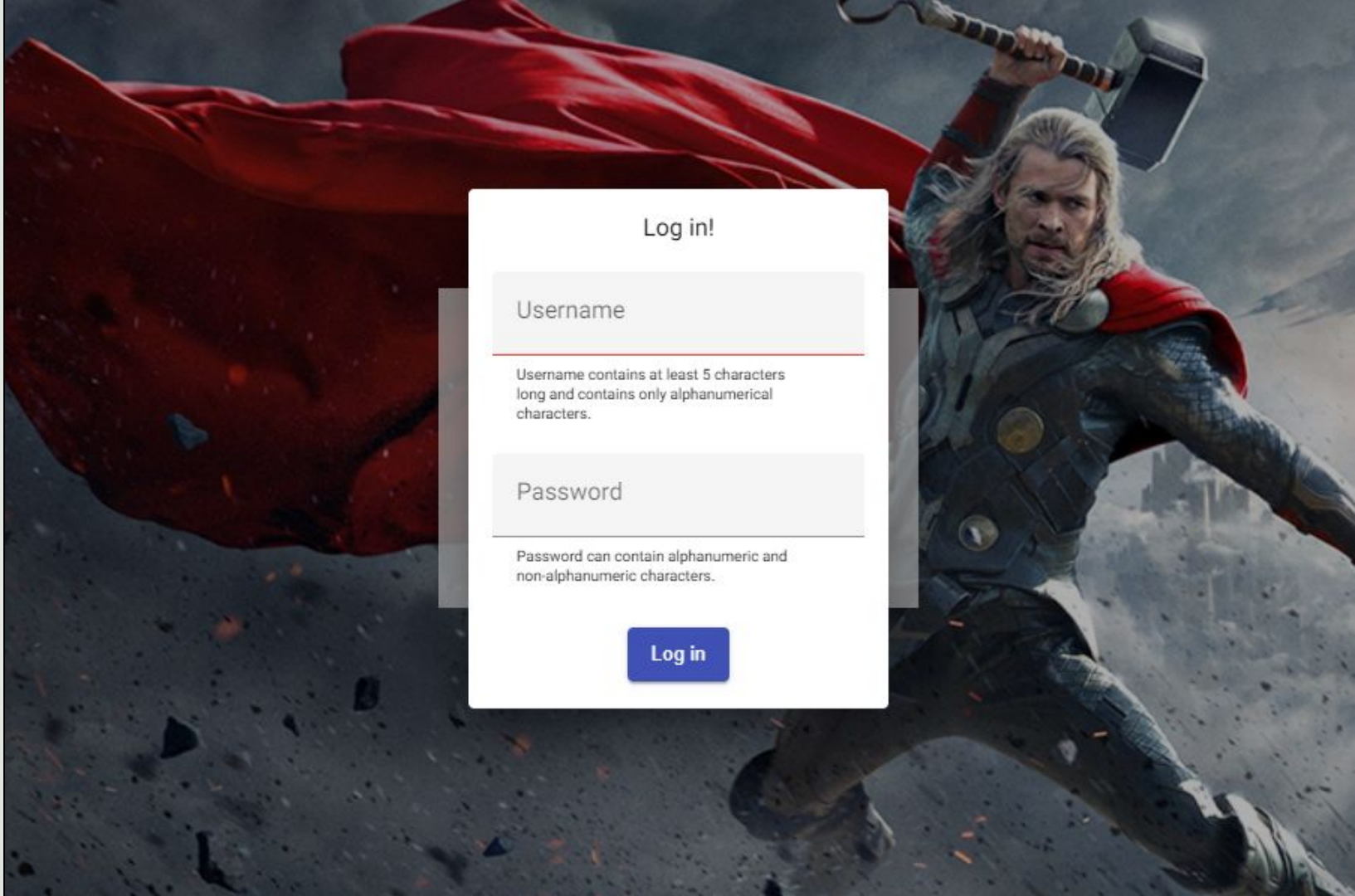
WELCOME PAGE (WITH STYLING)



SIGN UP FORM (WITH STYLING)



LOG IN FORM (WITH STYLING)



04

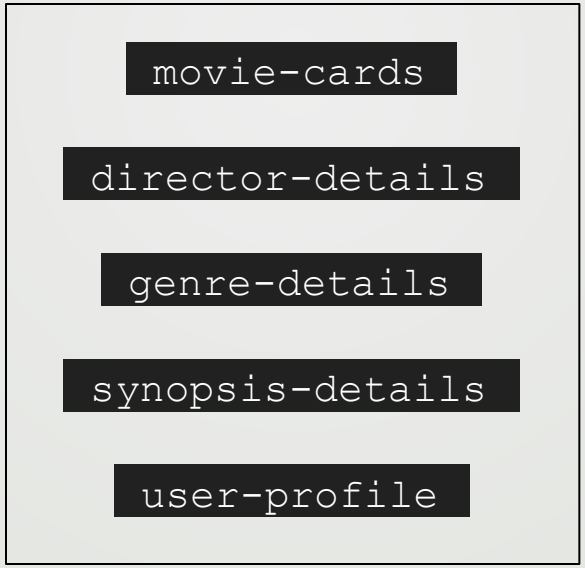
Implementing the app main view and profile view (for logged-in users) and routing between UIs

Skills used

Research
Problem-solving
Code writing
Debugging

WHY WAS THIS STEP IMPORTANT

This step was important to implement the actual core codes and views of the app. Without it, the app wouldn't have had any movies to display and no profile page on which users could consult their information and update it, which would have make the product pointless.



WHAT WAS THE GOAL

Creating the main view (displaying all movie cards) on which users land after logging in successfully, to do so, I've:

- Created the main view component
- Wrote codes in the main view TypeScript file to fetch the movies from the API
- Wrote codes in the main view HTML file to display the movies fetched from the API on the users' screens
- Added an action bar on each movie card with different links to display, in dialogs and based on user interaction, movie director details, movie genre details and movie synopsis. A favorite button has also been added (see images on next slides)
- Ensuring responsiveness and consistent styling / color palette with Angular Material for a good experience on any device sizes.

Creating the navigation bar. To do so, I've:

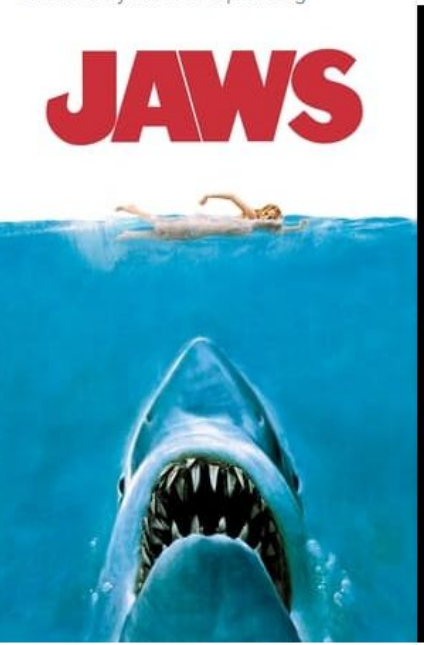
- Added the myFlix logo
- Listed all available pages to navigate
- Added the Logout function

FIRST RENDERED VIEW - MAIN VIEW ON LARGE SCREENS (NO STYLING)

myFlix

Menu

Jaws
Directed by: Steven Spielberg

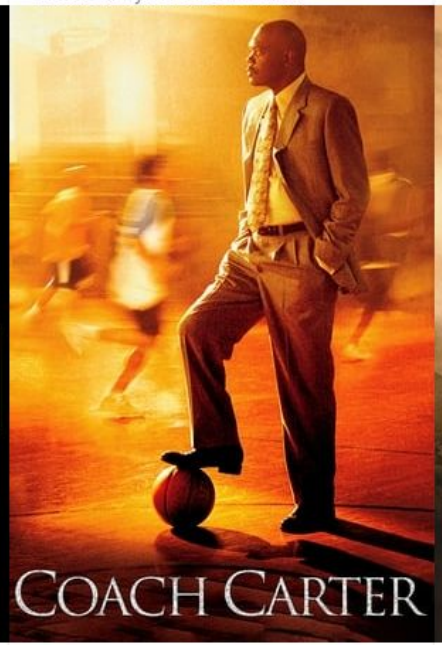


Genre Director Synopsis ♡

Jurassic Park
Directed by: Steven Spielberg

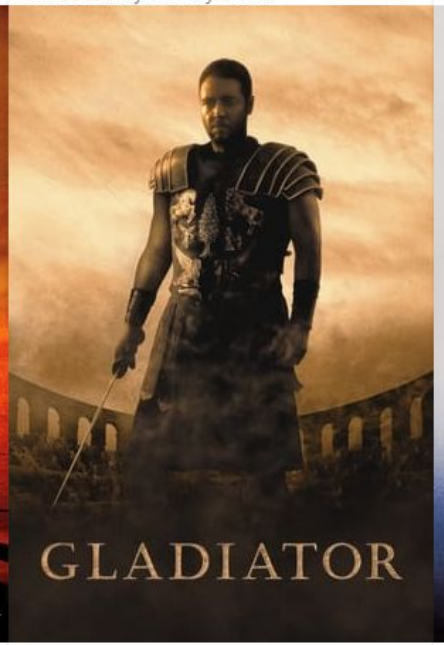
Genre Director Synopsis ♡

Coach Carter
Directed by: Thomas Carter



Genre Director Synopsis ♡

Gladiator
Directed by: Ridley Scott

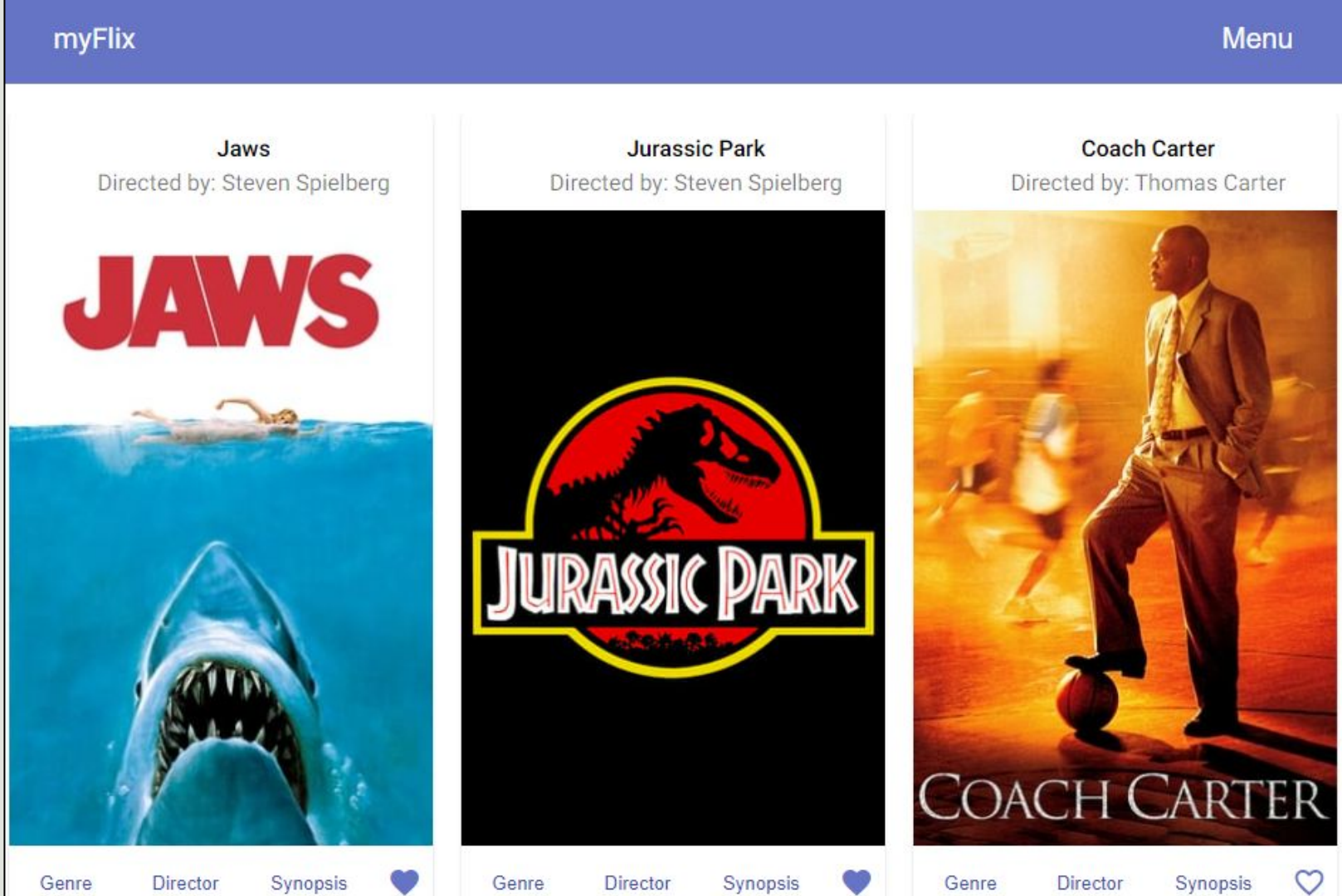


Genre Director Synopsis ♡

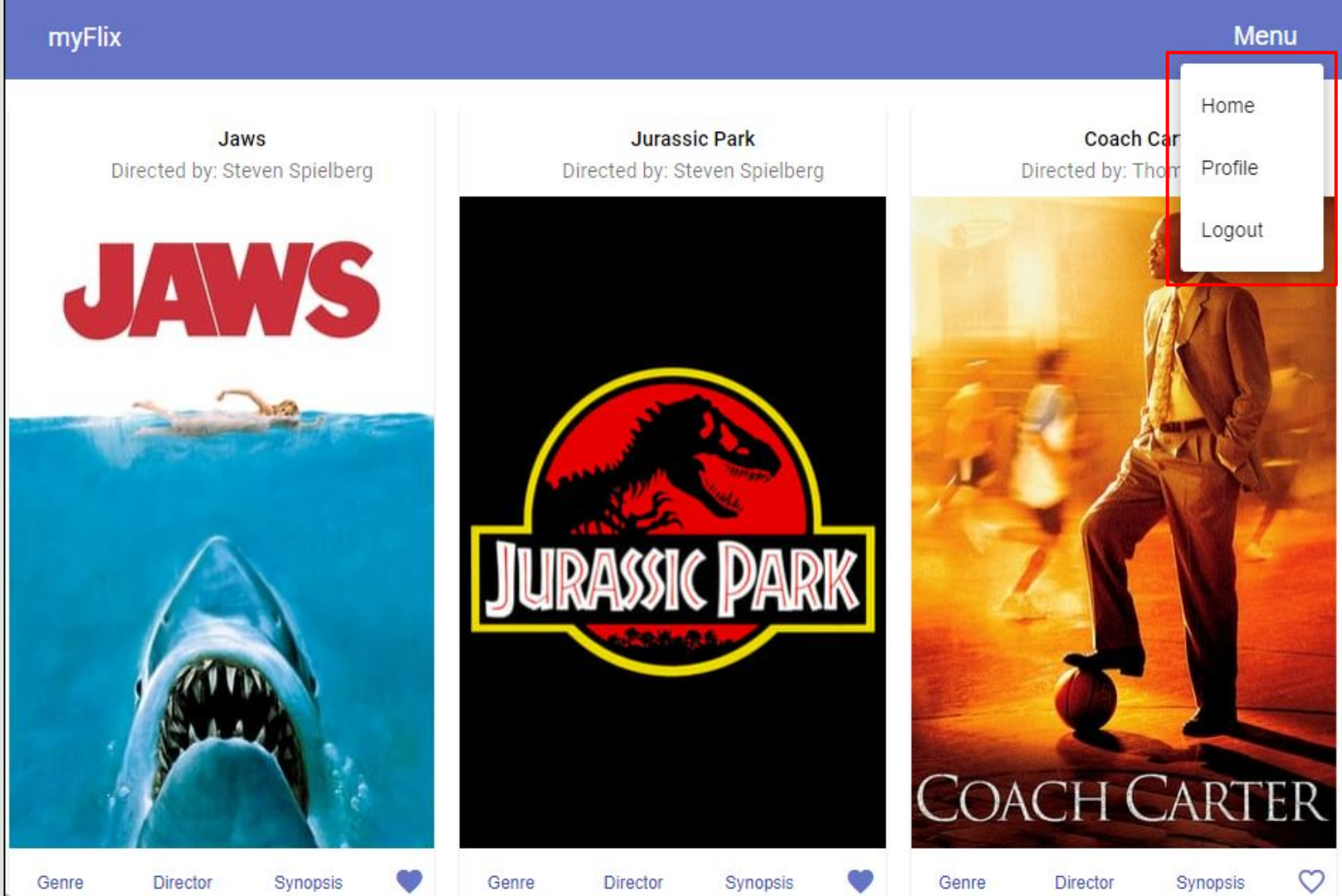
Silence of the Lambs
Directed by: Jonatha

Genre Director Sy

MAIN VIEW ON LARGE SCREENS (WITH STYLING)



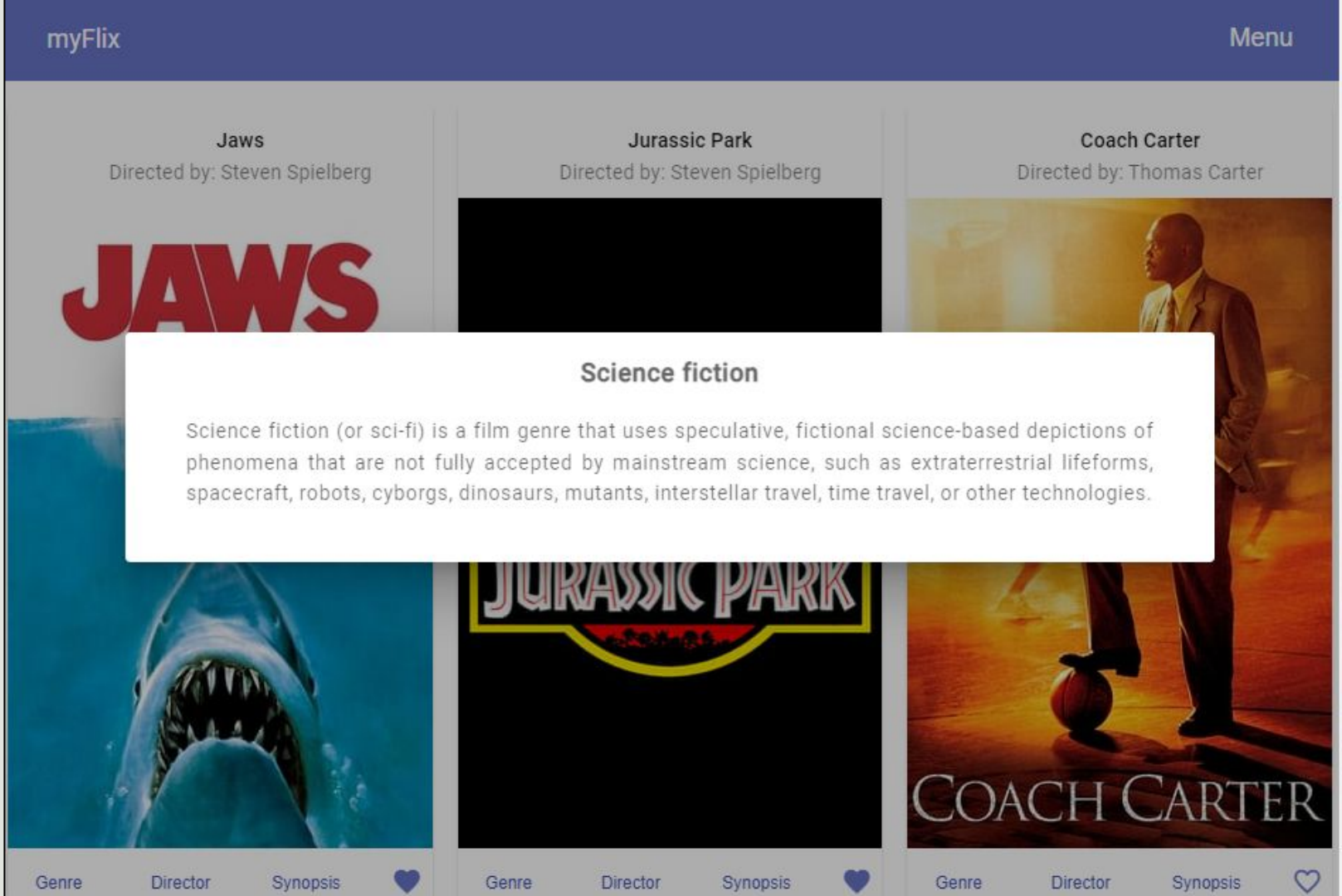
MAIN VIEW (WITH STYLING)



MAIN VIEW

-

MOVIE GENRE DETAILS



MAIN VIEW

-

MOVIE DIRECTOR DETAILS

myFlix

Menu

Jaws

Directed by: Steven Spielberg

Genre Director Synopsis

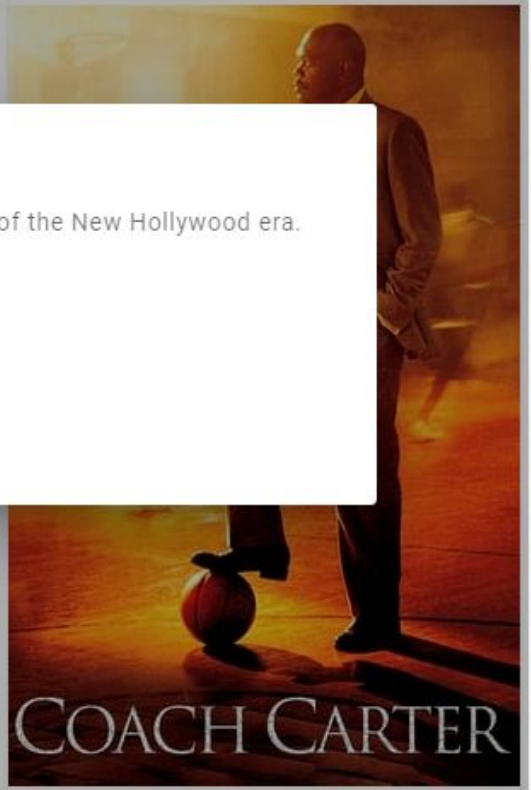
Jurassic Park

Directed by: Steven Spielberg

Genre Director Synopsis

Coach Carter

Directed by: Thomas Carter



Genre Director Synopsis

Steven Spielberg

Biography: Steven Spielberg is an American filmmaker and a major figure of the New Hollywood era.

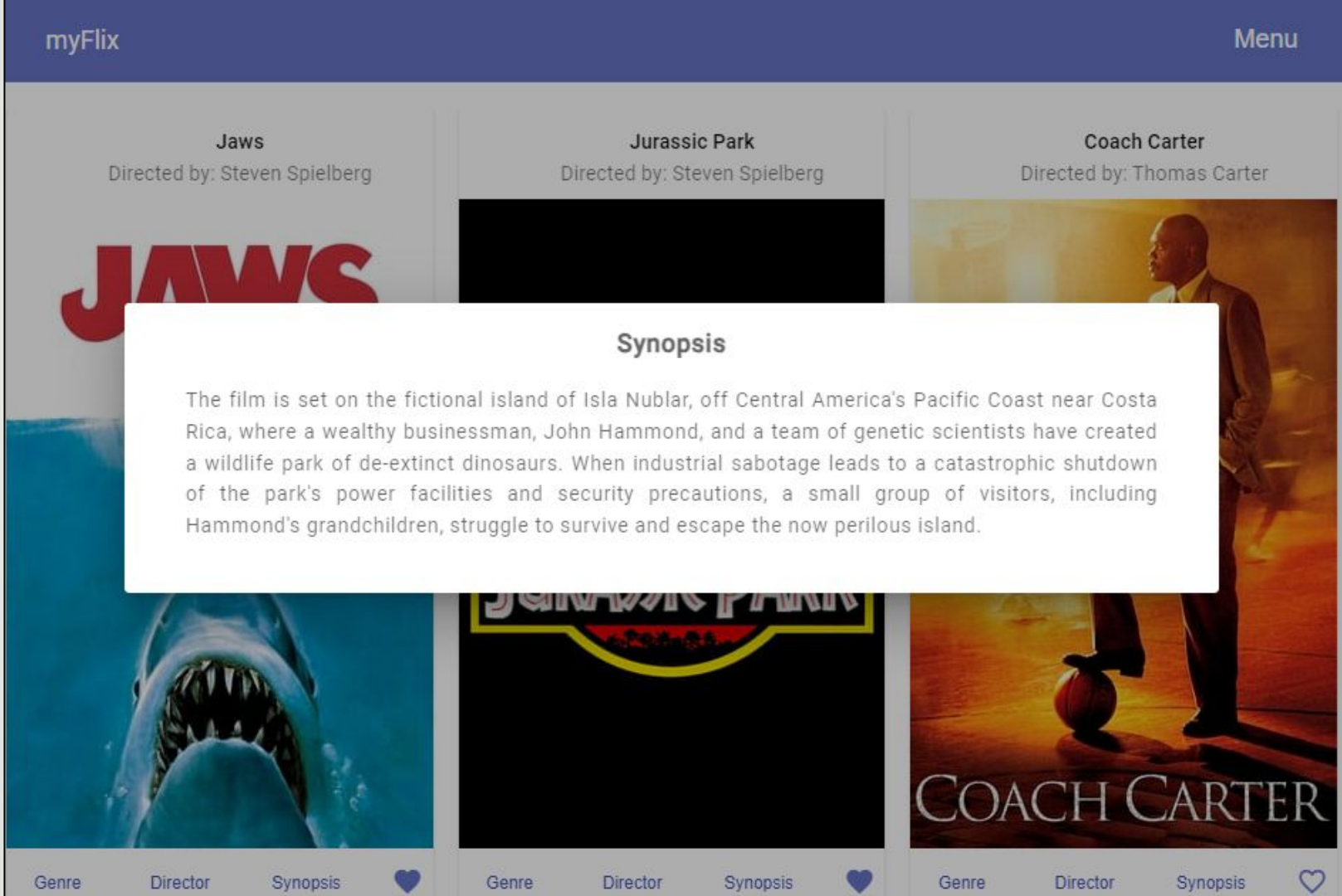
Date of birth: 12/18/46

Date of death: NA

MAIN VIEW

-

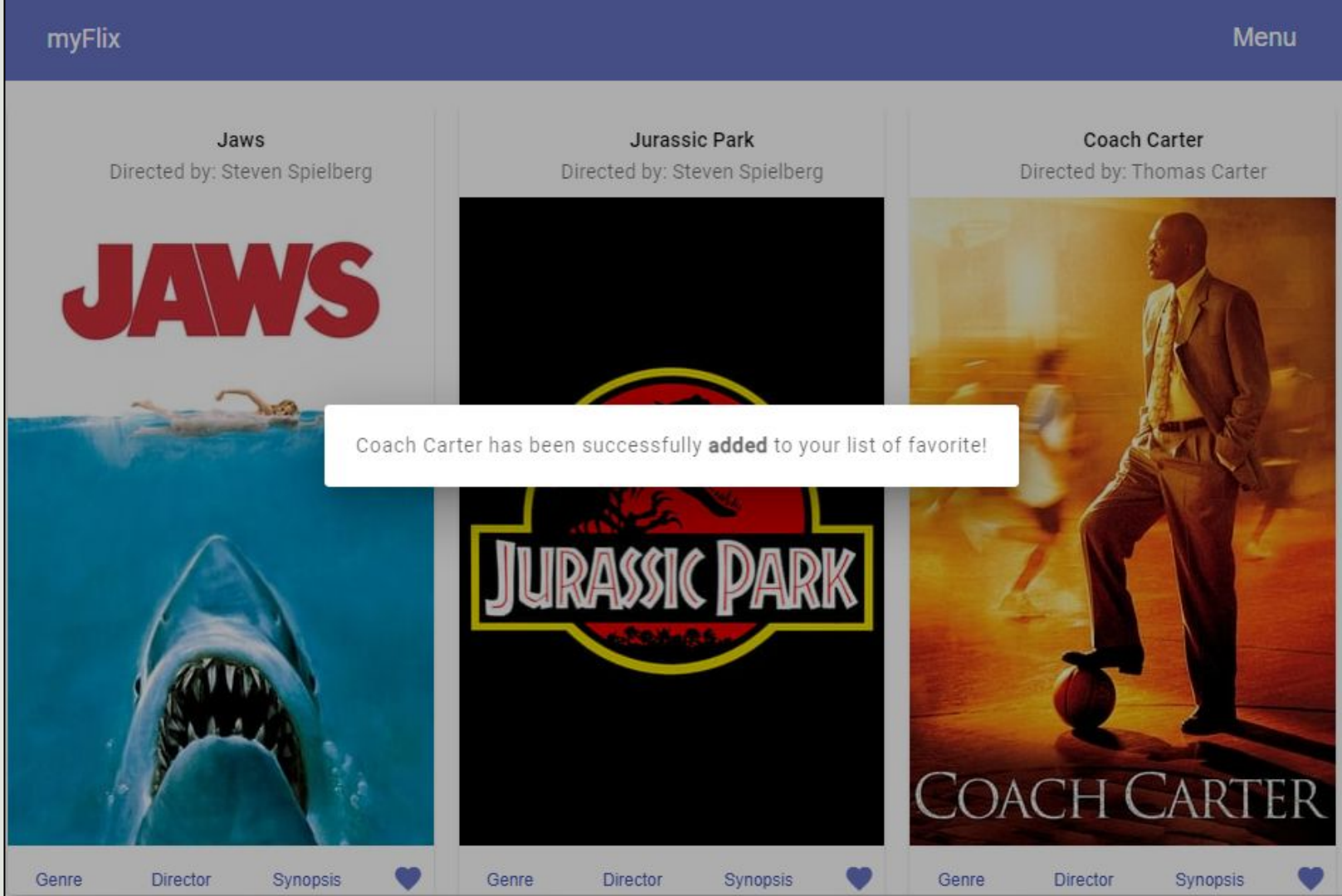
MOVIE SYNOPSIS DETAILS



MAIN VIEW

-

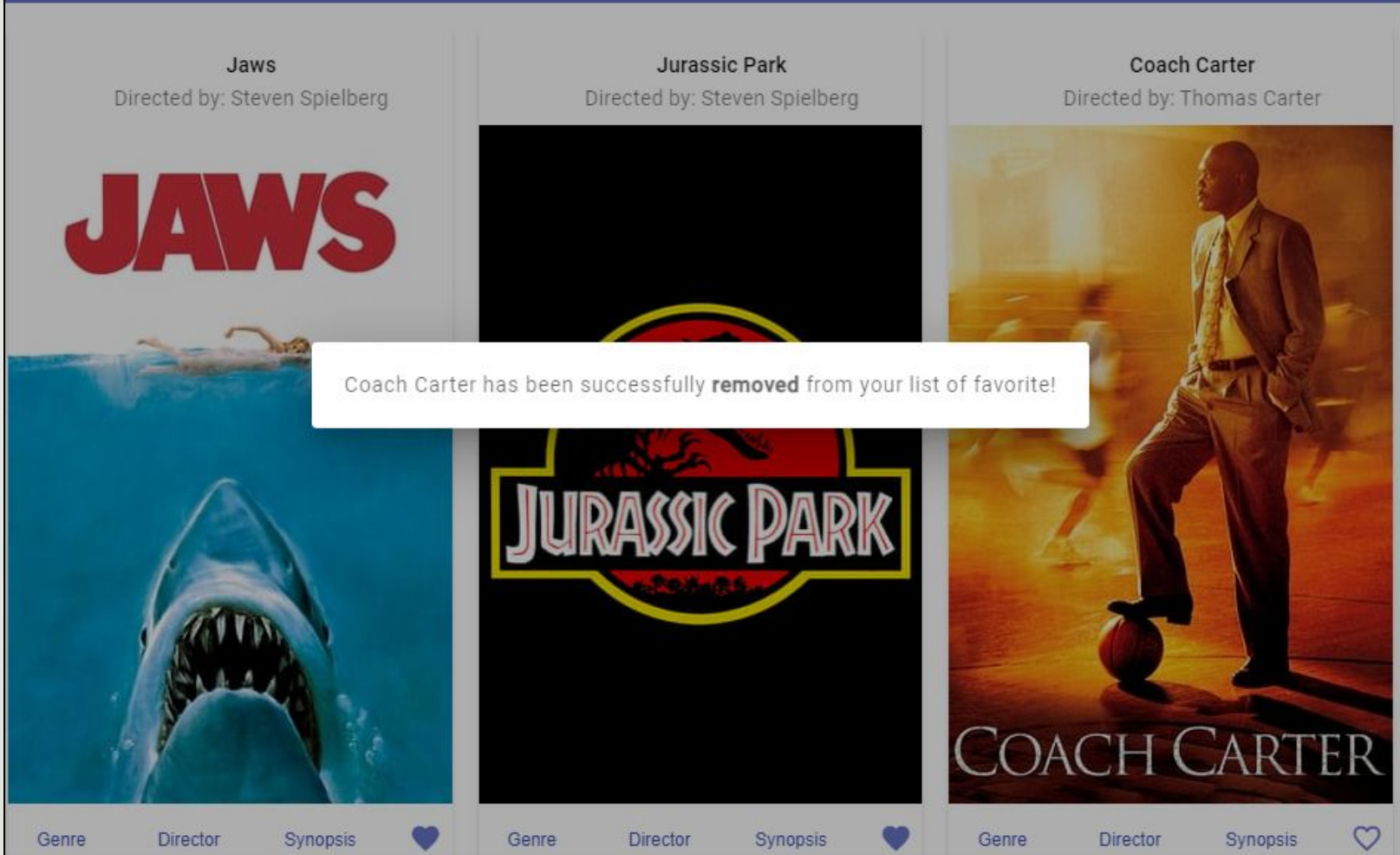
ADD TO FAVORITE



MAIN VIEW

-

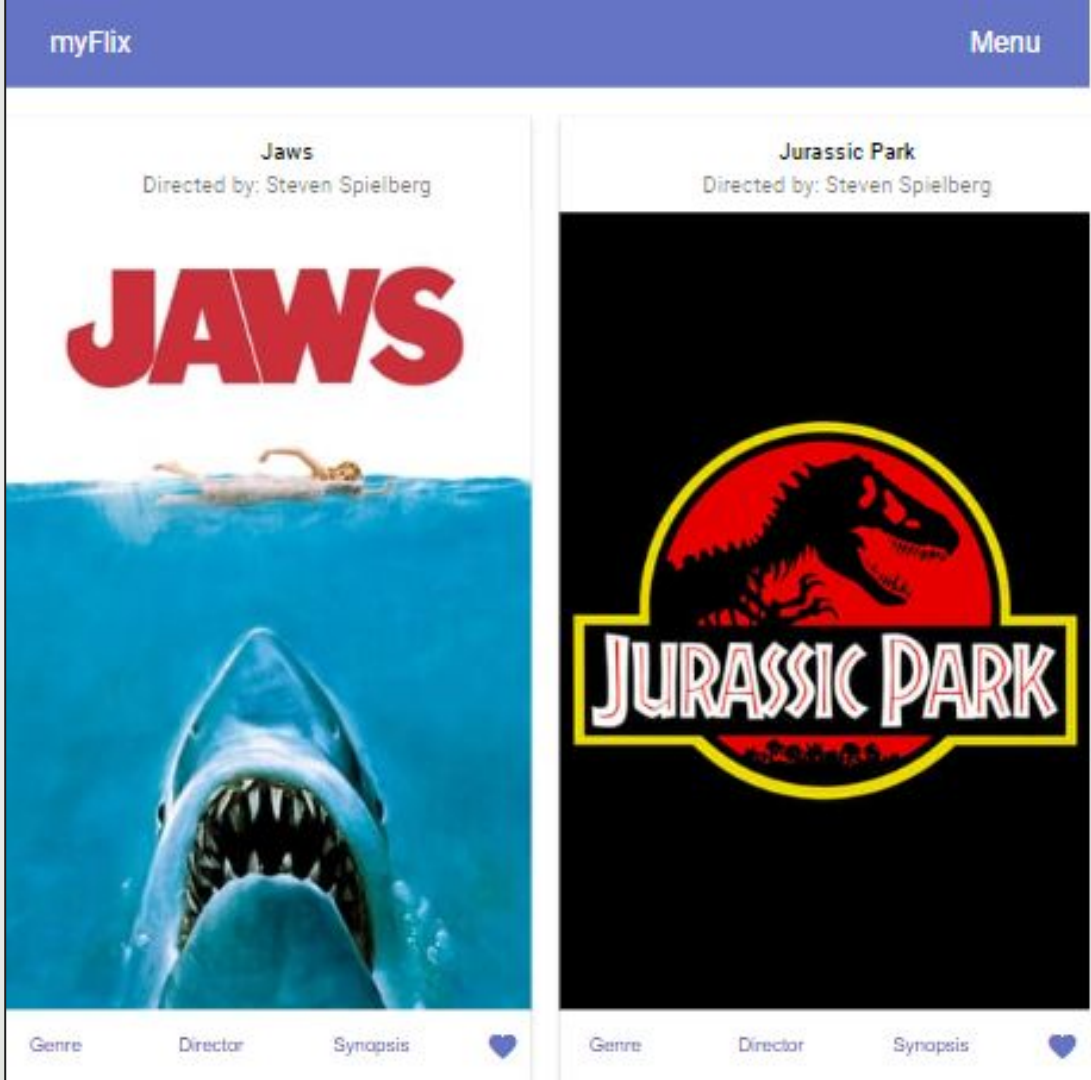
REMOVE FROM FAVORITE



MAIN VIEW

-

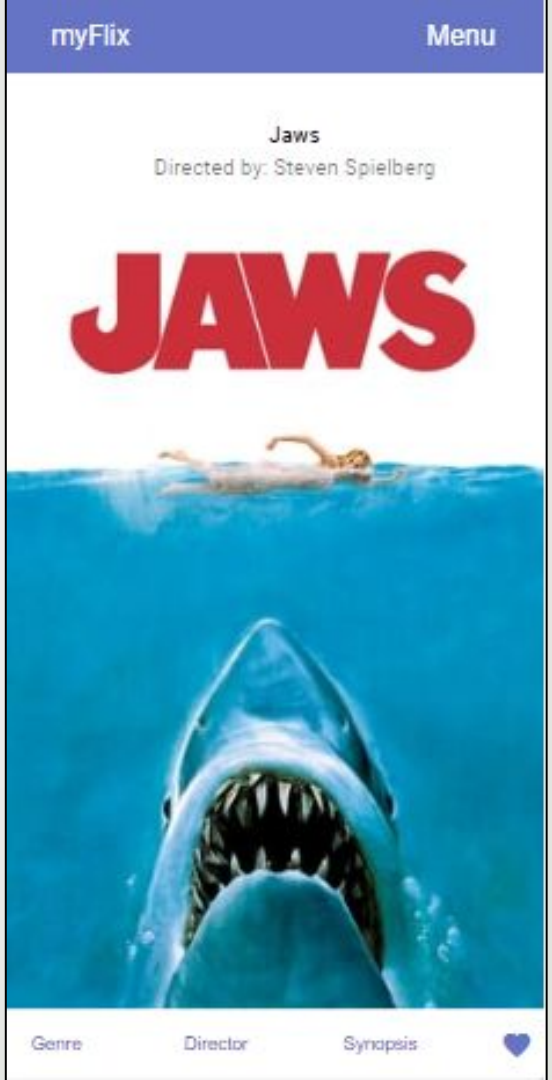
RESPONSIVE FOR MEDIUM SCREENS



MAIN VIEW

-

RESPONSIVE FOR SMALL SCREENS



WHAT WAS THE GOAL (SUITE)

Creating the profile page for users to be able to consult their info and update their account. To do so, I:

- Repeated some steps done in the main view UI creation (creation of a component and wrote code in TypeScript and HTML documents)
- Implemented logic to display user information, allow user to update their account (username, password, email and birthday) or delete it, and show movies that have been added to the favorite. I also added some codes to ensure users can remove movies from their favorite from within their profile page, and update the list of favorite accordingly in real time.
- Implemented logic to ensure that users actions (movie categorized as favorite) remains between session, so that users can go back on their account and always see / access the movies they’ve listed in a previous session as favorite.
- Ensuring responsiveness and consistent styling / color palette with Angular Material for a good experience on any device sizes.

FIRST RENDERED VIEW - PROFILE PAGE (NO STYLING)

User Profile

Name: Testuser

Email: Testuser@gmail.com

Favorite movies:

FIRST RENDERED VIEW - PROFILE PAGE (NO STYLING)

User Profile

Name: Testuser

Email: Testuser@gmail.com

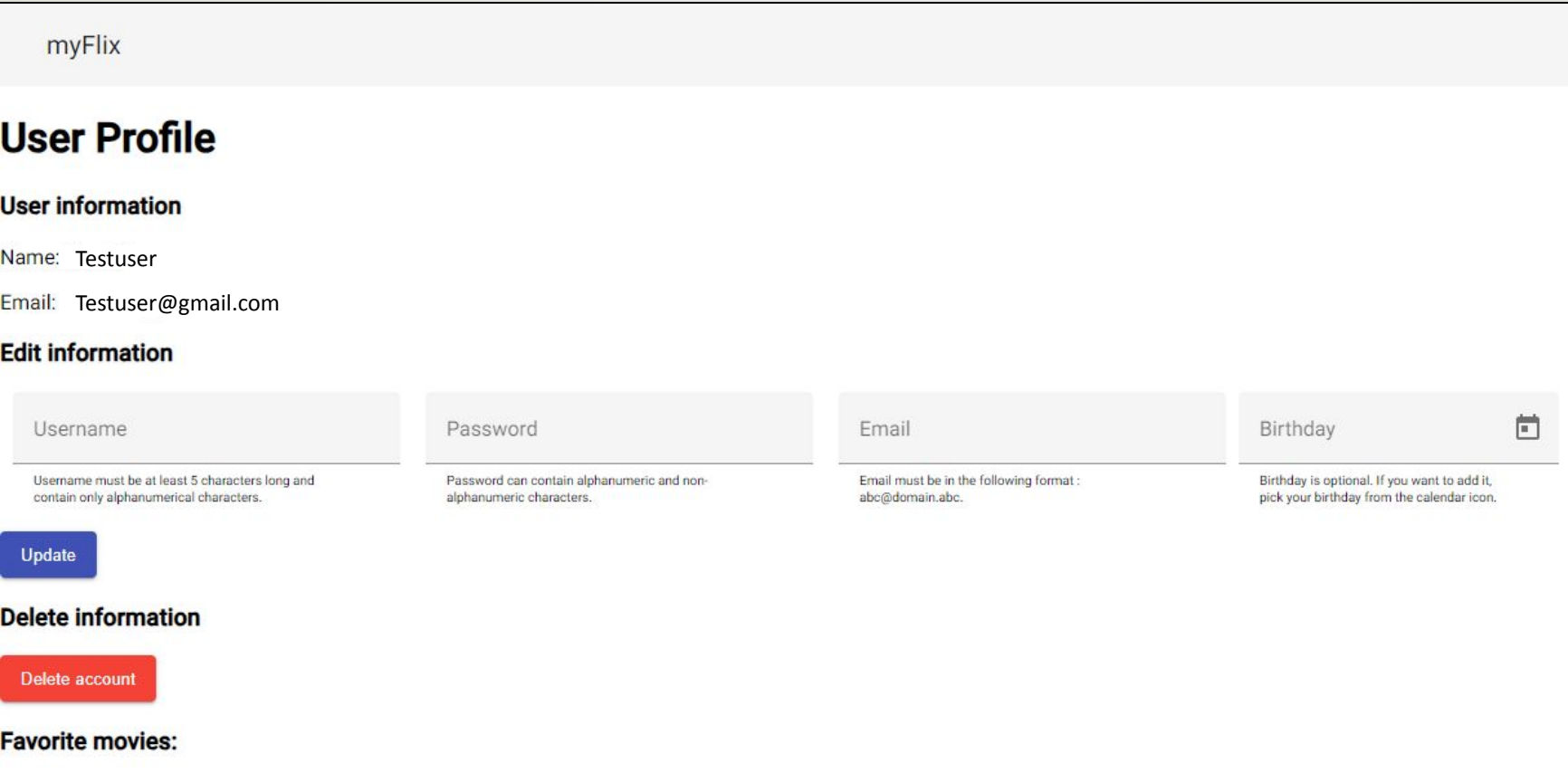
Favorite movies:

Jurassic Park

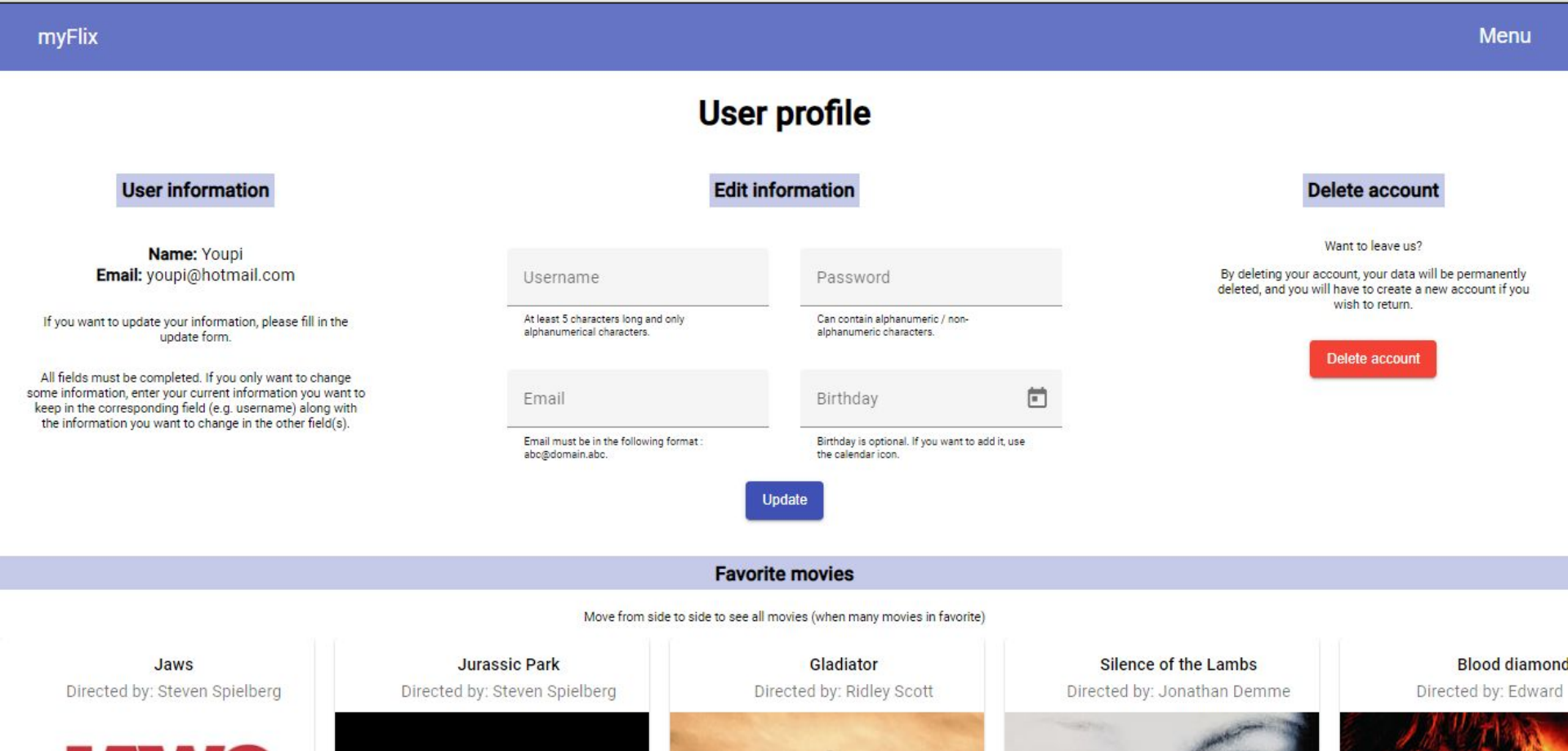
Directed by: Steven Spielberg

remove from favorite

FIRST RENDERED VIEW - PROFILE PAGE (NO STYLING)



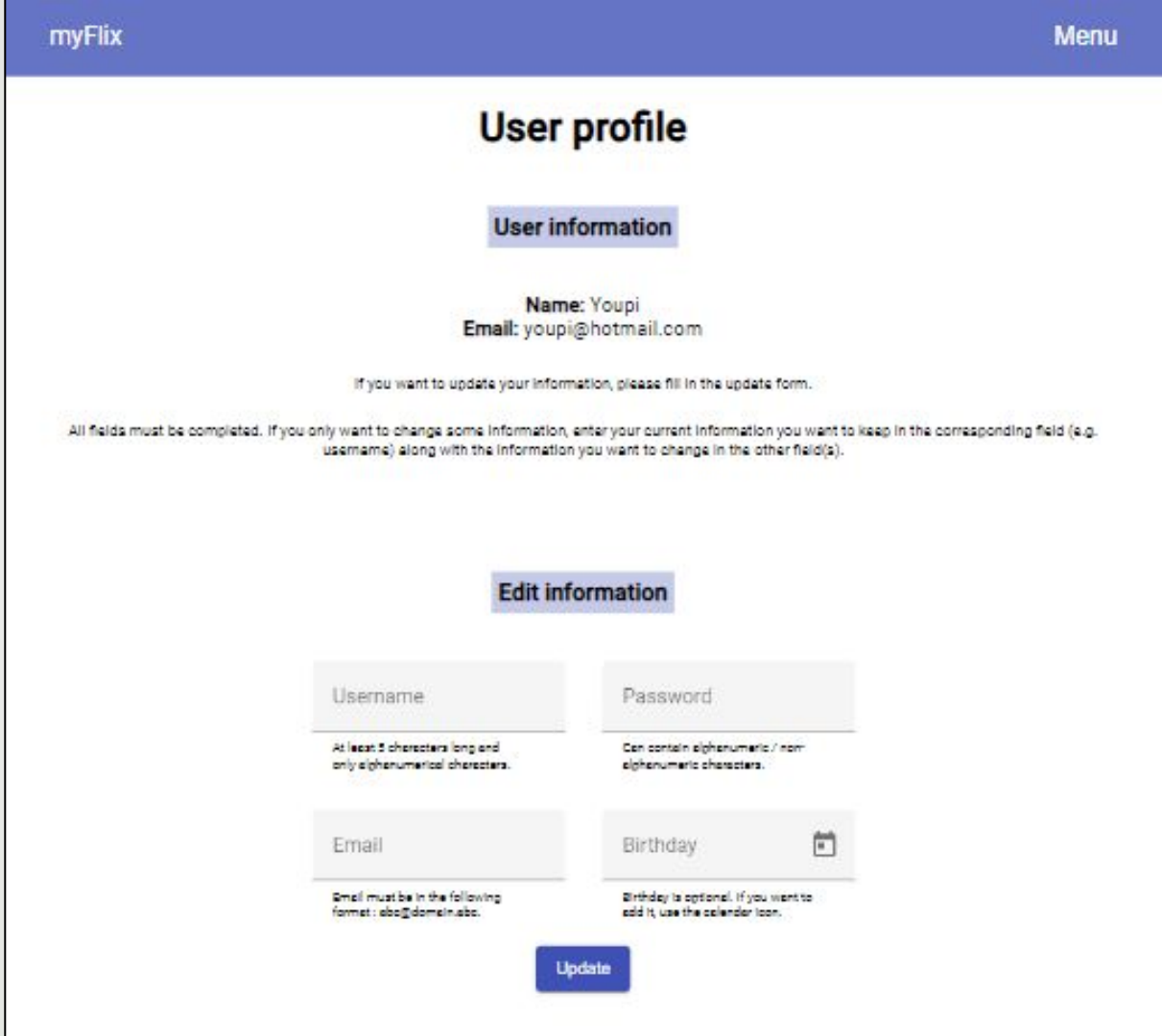
PROFILE PAGE ON LARGE SCREENS (WITH STYLING)



PROFILE PAGE

-

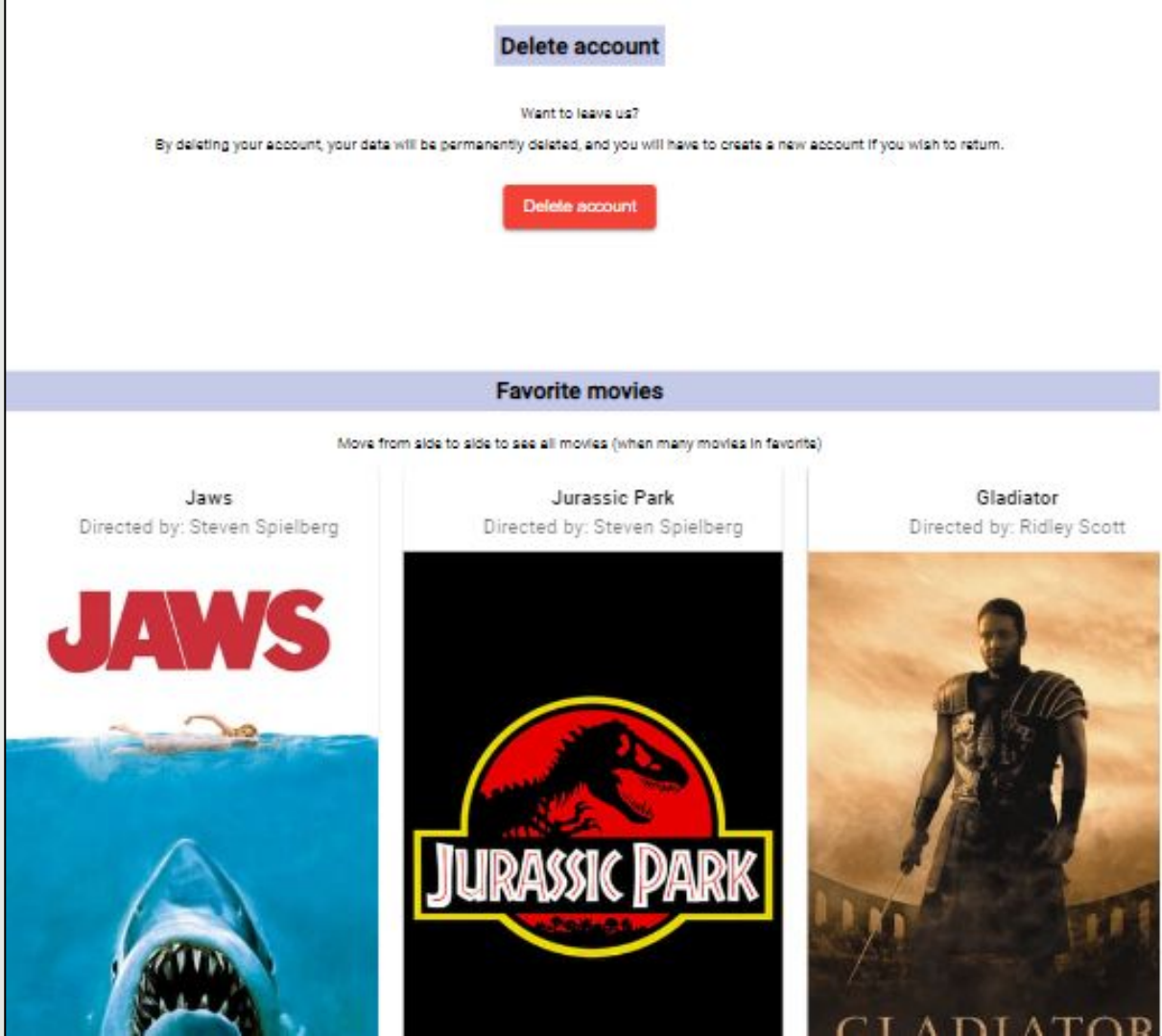
RESPONSIVE FOR MEDIUM SCREENS (PART 1)



PROFILE PAGE

-

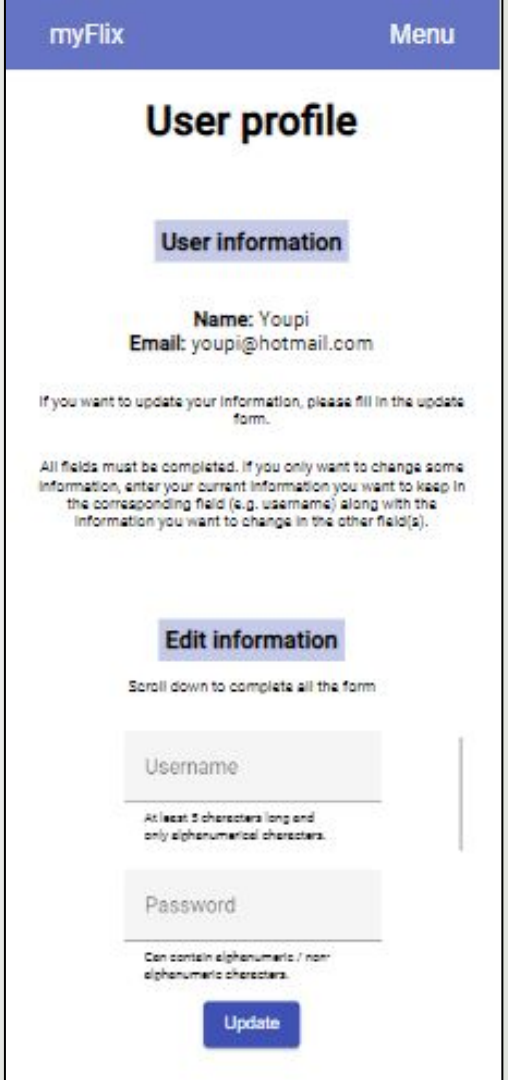
RESPONSIVE FOR MEDIUM SCREENS (PART 2)



PROFILE PAGE

-

RESPONSIVE FOR SMALL SCREENS (PART 1)



PROFILE PAGE

-

RESPONSIVE FOR SMALL SCREENS (PART 2)

WHAT WAS THE GOAL (SUITE)

Implementing routing between the different views (with RouterModule) to redirect users when certain actions are triggered (e.g: users are redirected to the welcome view when they delete their account). The three routes implemented are:

- /welcome
- /movies
- /user-profile

DECISIONS MADE

I chose how to structure and present the visuals (font, color, buttons, dialogs, etc.) of the welcome view, the main view and the profile page, at all levels of responsiveness.

05

Final code **revision, testing** and design **last adjustments**

Skills used

Critical thinking
Synthetization

WHY WAS THIS STEP IMPORTANT

Ensuring that the codes are optimized, that the web app passes all the tests and that the final product is in its best visual version was a logic thing to do for a sound deployment.

WHAT WAS THE GOAL

Reviewing each code to make sure everything was optimized as much as possible in order to facilitate future modifications, additions or adjustments to the web app.

Carrying out several tests on all the functionalities of the app to ensure that it works perfectly.

Adjusting some final design details to ensure professional looking.

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

I positioned myself from the point of view of future colleagues who could work on my project. How can I make my project and my codes as clear as possible to promote its easy appropriation? I reviewed each file to bring improvements in certain places where I thought it could be useful.

09

github-pages

Deploying the web app

Skills used

-

WHY WAS THIS STEP IMPORTANT

This step was essential to ensure that the app could be used and accessed by everyone.

WHAT WAS THE GOAL

Deploying the app to gh-pages and make it available for everyone.

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

Since I had previously implemented a CORS protection measure in my backend code, I had to add the new gh-pages domain of myFlix Angular in my backend's list of authorized domains, this ensuring that the API would authorize the requests routed to it instead of blocking them.

10

README.md

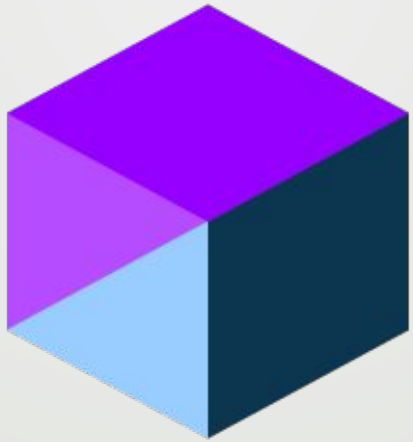
Creating all the necessary documentation on the web app

Skills used

Content writing
Technical writing
Communication

WHY WAS THIS STEP IMPORTANT

Adequate and complete documentation on a project, from the comments inside codes to the creation of a README document, usually greatly facilitate future updates, better and shared understanding, and project appropriation by new members, among other things. It is therefore important to carry out such task.



WHAT WAS THE GOAL

Preparing two of my projects for handoff to other developers - my myFlix backend API and my myFlix frontend Angular. To do so, I:

- Inserted meaningful comments where necessary in my projects files
- Generated documentation for myFlix backend using JSDoc (see image next slides)
- Generated documentation for myFlix Angular frontend using TypeDoc (see image next slides)
- Created a README on the project's Github repository (see image next slides)

Writing the README document on the project Github repository.

```
/**
 * @fileoverview passport.js
 * @description This file contains the logic that handles two authentication strategies: local (usernar
*/

const passport = require('passport'),
    LocalStrategy = require('passport-local').Strategy,
    Models = require('./models.js'),
    passportJWT = require('passport-jwt');


let Users = Models.User,
    JWTStrategy = passportJWT.Strategy,
    ExtractJWT = passportJWT.ExtractJwt;


passport.use(new LocalStrategy(
    {
        usernameField: 'Username',
        passwordField: 'Password'
    },
    (username, password, callback) => {
        console.log(username + '  ' + password);
        Users.findOne({ Username: username })
            .then((user) => {
```

- myFlix BACKEND -

COMMENTS WRITTEN IN JSDOC FORMAT TO GENERATE JSDOC DOCUMENTATION AUTOMATICALLY (SEE NEXT IMAGE)

myFlix JSDoc DOCUMENTATION AUTOMATICALLY GENERATED

Home

auth.js

This file contains the API logic for user login, including authentication using Passport and emission of JWT tokens upon users successful login. It has one POST endpoint : /login.

Source: auth.js, line 1

index.js

This file constitutes the main API element. All endpoints are defined there, as well as their specific parameters, what they require and what information they return. Endpoints summary (12 in total) :

-6 GET
-1 PUT
-3 POST
-2 DELETE

This file also contains the codes to import other project files (eg: auth.js which contains the logic for log in or passport.js which contains the logic related to JWT) as well as codes that configure certain additional elements, such as cross-origin resource sharing (CORS).

```
/**
 * @module fetch-api-service
 * @remarks
 * This component regroups all the logic and function to call the different API's endpoints. Specific
 *
 * This component include several requests/functions:
 *
 * -6 GET requests
 * -1 PUT request
 * -3 POST requests
 * -2 DELETE requests
 */

import { Injectable } from '@angular/core';
import { catchError } from 'rxjs/operators';
import { HttpClient, HttpHeaders, HttpErrorResponse } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { map } from 'rxjs/operators';


/**
* Declaring the api url that will provide data for the client
*/
const apiUrl = 'https://my-weekend-movie-app-53a46e3377d7.her
@Injectable({
  providedIn: 'root'
})
```

- myFlix FRONTEND -

COMMENTS WRITTEN IN TYPEDOC FORMAT TO GENERATE TYPEDOC DOCUMENTATION AUTOMATICALLY (SEE NEXT IMAGE)

myFlix TYPEDOC DOCUMENTATION AUTOMATICALLY GENERATED

my-flix-angular-client

- my-flix-angular-client
  - app-root
    - AppComponent
  - favorite-movie-add
    - AddFavoriteDetailsCompo
  - favorite-movie-remove
    - RemoveFavoriteDetailsCor
  - fetch-api-service
    - FetchApiDataService
  - get-all-movies
    - MovieCardComponent
  - get-information-on-director
    - DirectorDetailsComponent
  - get-information-on-genre
    - GenreDetailsComponent
  - get-information-on-synopsis
    - SynopsisDetailsComponen

Module get-information-on-director

› Settings

Remarks

This component allows to display information about the director of each movie in a dialog prompt (when clicked by users). To do so, it takes the information about the director of X movie object, and renders it in the html.

Example

```
In the TypeScript file, this is the data extracted from a movie object and
required for this component to work:

data: {
   Name: string,
   Bio: string,
   Birth: string,
   Death: string
 }

These are then accessed within the html (e.g: {{ data.Name }} to display
director's name).
```

Param

README SAMPLE

-

FULL VERSION ON GITHUB

README.md

myFlix web app documentation (frontend - Angular)

Content

Projet description

myFlix web app has been created to serve as a reference in the domain of visual entertainment. Users can create an account and then log into *myFlix* to have access to information about different movies. They can search for movies and create lists of favorites. *myFlix* has been built in two parts: the frontend (here) and the backend (see this repository for the backend part of *myFlix*).

The objective of this part of the project (frontend) was to develop an easy-to-use and responsive web app using Angular for the best user-experience whenever they are accessing *myFlix* to read details about different movies.

myFlix frontend development can be breakdown in the five following points:

- **Who** — The users of *myFlix* web app, so movie enthusiasts who enjoy reading information about different

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

Finding the right balance between giving the right level of information, while remaining as synthetic as possible.

DECISIONS MADE

I wrote the README documentation from A to Z, in terms of content, presentation and structure.