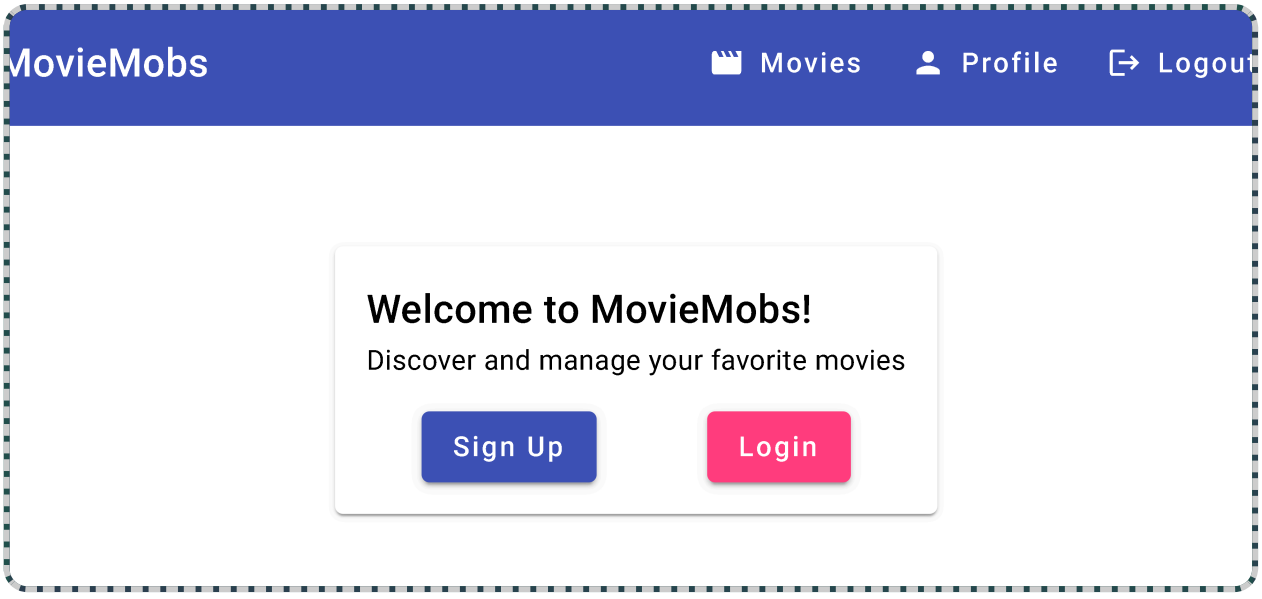


MovieMobs

A Case Study in Modernizing an Angular Application

Project Pitch:

Along my training at CareerFoundry.com I was assigned several tasks that aimed at the final building of an app related to movies, under the concept of "MyFlix" which is evidently related to other commercial movie platforms we are familiar with, worldwide. I took the liberty to rename it "MovieMobs", re-design its frameworks and visuals, and try my best, as a full-stack developer in training, to obtain a modern Angular 19 application, by independently solving complex issues caused by outdated learning materials and framework evolution. This project showcases my acquired expertise in Angular and Angular Material, as well as my ability to research, adapt, and deliver what I consider ultimately resilient, production-ready solutions.

MY ROLE

Sole Developer

Responsible for setup, UI/UX, API integration, and debugging

DURATION

3 Weeks

TECH STACK

Angular 19 (Standalone)

Angular Material 19

TypeScript & RxJS

SCSS

REST API

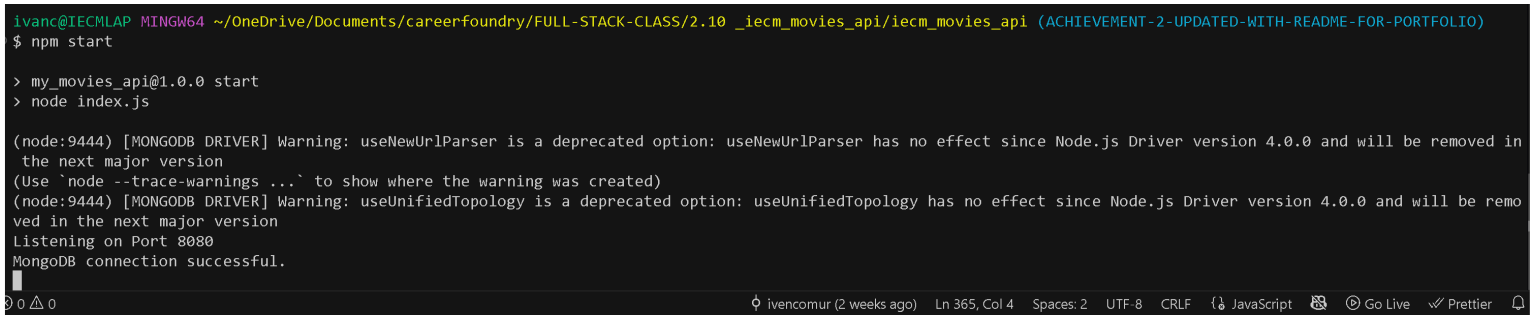
The Process & Challenges

The primary challenge was that the course materials were based on a previous version of Angular. I chose to build the application using the current standard, Angular 19. This decision created crucial learning opportunities that reflect real-world development work.

Challenge 1: Architecture: From Legacy NgModule to Modern Standalone Components

The legacy materials used the now-outdated NgModule architecture. To build a scalable and maintainable app, I refactored the entire structure to use Angular 19's modern standalone component architecture. This involved researching the latest documentation and ensuring every component was modular and self-contained, resulting in a scalable, maintainable codebase ready for future enhancements.

BEFORE (LEGACY NGMODULE)



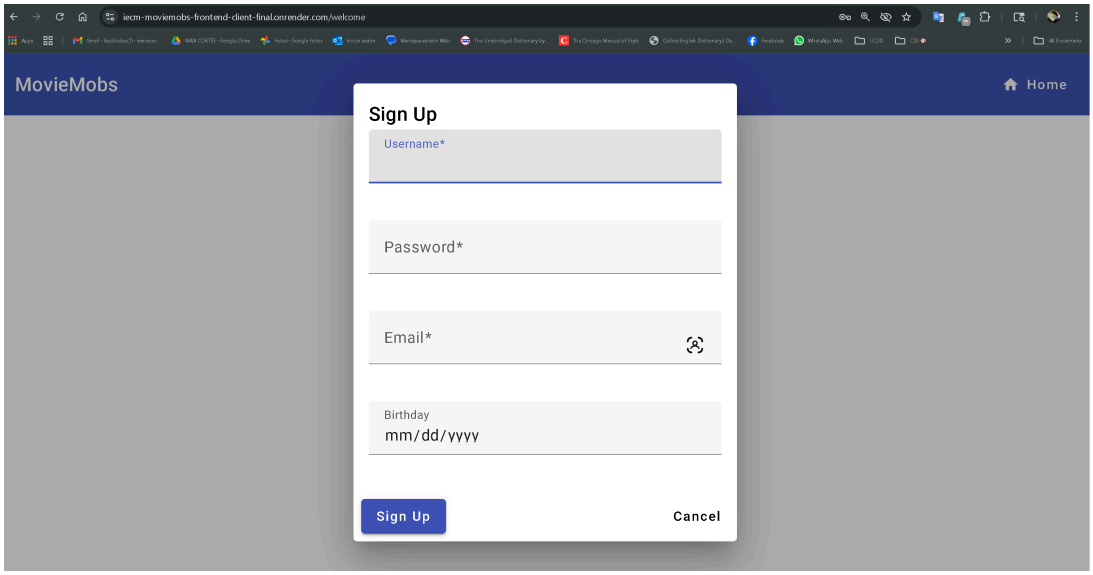
AFTER (MY SOLUTION)

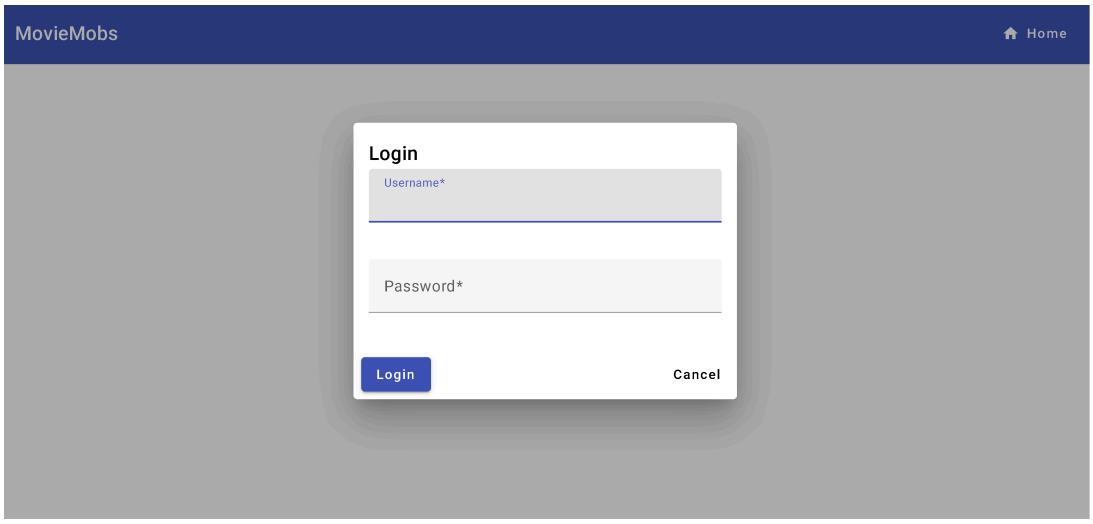
```
$ npm start

> my_movies_api@1.0.0 start
> node index.js

Listening on Port 8080
DB Connection successful
Attempting to authenticate user: IVANUSHKA
User found: IVANUSHKA
Authentication successful for user: IVANUSHKA
About to generate JWT for user: IVANUSHKA
User JSON: {
  _id: new ObjectId('68c47e9bc3dba8511c92a0a6'),
  Username: 'IVANUSHKA',
  Password: '$2b$10$hcqKXxZ41w5ZQYfWzIl51.4iVl7S6t316PFKsw3jB7DMf29eKuvGO',
  Email: 'ivan@ivanes.com',
  Birthday: 1921-01-01T00:00:00.000Z,
  FavoriteMovies: [],
  __v: 0
}
User object passed to generateJWTToken: {
  _id: new ObjectId('68c47e9bc3dba8511c92a0a6'),
  Username: 'IVANUSHKA',
  Password: '$2b$10$hcqKXxZ41w5ZQYfWzIl51.4iVl7S6t316PFKsw3jB7DMf29eKuvGO',
  Email: 'ivan@ivanes.com',
  Birthday: 1921-01-01T00:00:00.000Z,
  FavoriteMovies: [],
  __v: 0
}
Extracted username: IVANUSHKA Type: string
JWT token generated successfully
JWT Payload: {
```

Challenge 2: UI: Modernizing Angular Material Theming

Angular Material's theming syntax changed dramatically between versions, and the old mat.define-palette() function was deprecated. I studied the official migration guides and iteratively debugged the SCSS to implement a custom theme using the new mat.theme() function, resulting in a clean UI consistent with Material Design 3.

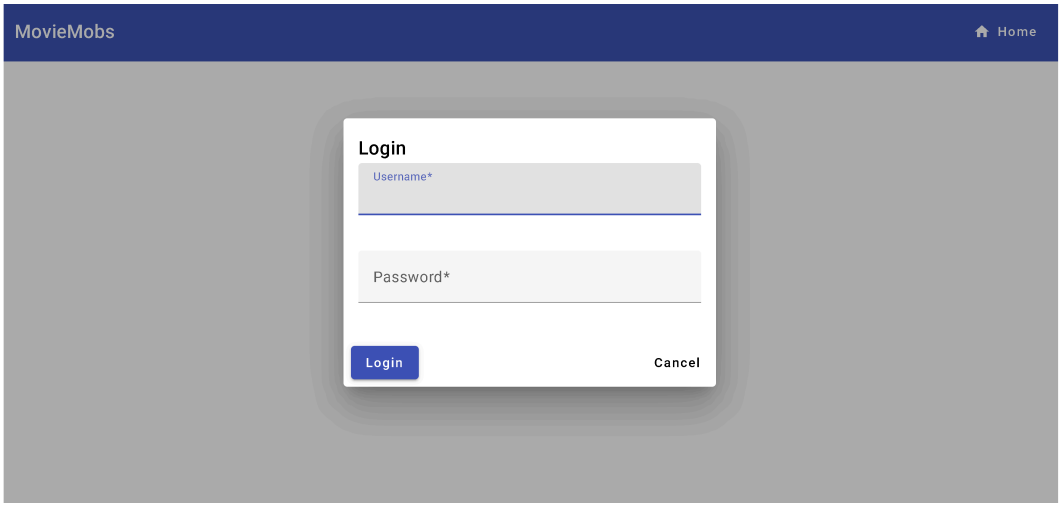
VISUAL PROOF:




MovieMobs
Home
Login
Username*
Password*
Login
Cancel

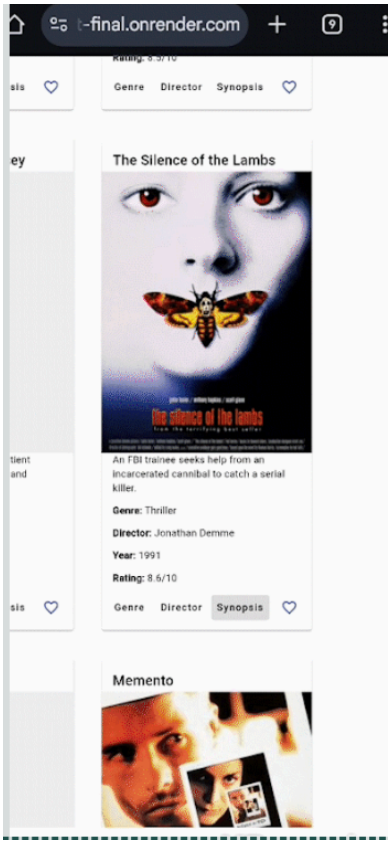
Challenge 3: API Integration & Asynchronous Handling

Runtime errors arose from changes in how animation providers were loaded and mismatches between the API's data fields and the examples. I engineered a robust solution by creating a reusable API service (fetch-api-data.service.ts) that uses RxJS to manage all asynchronous logic and TypeScript interfaces to ensure type-safe data handling from the backend. This resulted in stable backend communication with robust error handling.


MovieMobs
Home
Login
Username*
Password*
Login
Cancel

The Final Solution

The result is a fully functional and modern Angular application featuring secure user registration and login with session persistence. It is built on a strong, modular foundation that is ready for future feature development.



Key Learnings & Future Steps

Key Learnings

This project was a critical lesson in the rapid evolution of web development. It strengthened my resilience and problem-solving mindset, proving that I can successfully deliver a modern application even when faced with outdated resources and documentation.

Future Steps

The next steps are to implement the core movie features:

→ A browsable movie gallery

→ Detailed movie view pages

→ User-specific favorite lists

→ Deployment to cloud platform with CI/CD pipeline

View the Project

Frontend

View on GitHub

Live Demo

Backend

View on GitHub

Live Demo