**📌 Full Learning + Implementation Plan**

**1️⃣ Week 1: Setup, Components & Routing**

**🎯 Learning Goals**

✅ **Angular Setup & CLI Commands**  
✅ **Project Structure** (app.module.ts, index.html, styles.scss)  
✅ **Components & Data Binding** ({{}}, [], (), [(ngModel)])  
✅ **Routing & Navigation**

**🛠 Project Implementation**

✅ Install Angular, Bootstrap & Create a New Project

sh

CopyEdit

npm install -g @angular/cli

ng new movie-database

cd movie-database

npm install bootstrap

✅ Configure Bootstrap in angular.json  
✅ Create **3 main pages**

sh

CopyEdit

ng generate component pages/home

ng generate component pages/movie-detail

ng generate component pages/favorites

✅ Implement **Navbar & Routing** using RouterModule  
✅ Create movies array and display static movie data using \*ngFor

**2️⃣ Week 2: Fetching API Data & UI Design**

**🎯 Learning Goals**

✅ **HTTP Requests (HttpClientModule)**  
✅ **Observables & RxJS (subscribe(), map())**  
✅ **Styling with Bootstrap**

**🛠 Project Implementation**

✅ Register for **TMDb API Key**  
✅ Create **Movie Service** to fetch data

sh

CopyEdit

ng generate service services/movie

✅ Fetch **popular movies** and display on the home page  
✅ Use Bootstrap Cards to style movie posters  
✅ Implement **Search Functionality** using API

**3️⃣ Week 3: Authentication & State Management**

**🎯 Learning Goals**

✅ **Forms (ngModel, FormGroup, FormControl)**  
✅ **LocalStorage for session management**  
✅ **Route Guards (CanActivate)**

**🛠 Project Implementation**

✅ Create **Login & Signup Pages**  
✅ Store user session in localStorage  
✅ Protect **Favorites Page** using Route Guard  
✅ Show **"Login Required"** message if user is not authenticated

**4️⃣ Week 4: Favorites Feature & Movie Details**

**🎯 Learning Goals**

✅ **State Management (BehaviorSubject, localStorage)**  
✅ **Movie Details Page (ActivatedRoute)**

**🛠 Project Implementation**

✅ Create **"Add to Favorites"** button for each movie  
✅ Save favorites in localStorage  
✅ Create **Favorites Page** to display saved movies  
✅ Implement **Movie Details Page** to fetch movie details dynamically

**5️⃣ Week 5: Movie Recommendation System (ML Integration)**

**🎯 Learning Goals**

✅ **Machine Learning Basics** (Collaborative Filtering, Content-Based Filtering)  
✅ **Python Flask API for Recommendations**  
✅ **Connecting Angular with Flask API**

**🛠 Project Implementation**

✅ Train a **Movie Recommendation Model** using Python  
✅ Use **TMDb Movie Dataset** (or any public dataset)  
✅ Deploy **Flask API** to serve recommendations  
✅ Call Flask API from Angular using HttpClient  
✅ Show **Recommended Movies** on the Movie Details page

**6️⃣ Week 6: Final Improvements & Deployment**

**🎯 Learning Goals**

✅ **Animations & UI Enhancements**  
✅ **Optimizing API Calls (Caching, Lazy Loading)**  
✅ **Deploying on Firebase or Netlify**

**🛠 Project Implementation**

✅ Add **Loading Spinners**  
✅ Improve UI with **Angular Material**  
✅ Deploy Flask API to **Render or AWS Lambda**  
✅ Deploy Angular App to **Firebase, Vercel, or Netlify**

**📌 Super Detailed Weekly Tasks**

**⏳ Full Timeline:**

| **Week** | **Learning** | **Implementation** |
| --- | --- | --- |
| **Week 1** | Angular Basics, Components, Routing | Create pages, navbar, and static movie list |
| **Week 2** | API Requests (HttpClient), RxJS | Fetch movies from TMDb API, Design UI |
| **Week 3** | Forms, Authentication, Guards | Add login/signup, Protect pages |
| **Week 4** | State Management, Movie Details Page | Add Favorites feature, Dynamic movie details |
| **Week 5** | ML & Flask API for Recommendations | Train Movie Recommender, Connect with Angular |
| **Week 6** | UI Enhancements, Deployment | Improve UI, Deploy Flask API & Angular App |

**📌 Technologies Used**

* **Frontend:** Angular, Bootstrap, RxJS
* **Backend (for ML):** Python, Flask, Pandas, Scikit-learn
* **Database:** TMDb API, LocalStorage
* **Deployment:** Firebase, Vercel, Netlify, Render