Full-Stack Registration/Login/Profile App - Step-by-Step Guide

# Project Overview

🔐 Features:

- User Registration (extended fields)

- User Login (JWT-based)

- Protected Profile Page

- Fully connected Angular + Node + MongoDB stack

# PART 1: BACKEND (Node.js + Express + MongoDB + JWT)

📁 Backend Project Structure:

user-auth-api/

├── config/

│ └── db.js

├── controllers/

│ └── userController.js

├── middleware/

│ └── authMiddleware.js

├── models/

│ └── User.js

├── routes/

│ └── userRoutes.js

├── .env

├── package.json

├── server.js

🧑‍🍳 Step-by-Step Setup:

**1. Create backend folder**

mkdir user-auth-api

cd user-auth-api

**2. Initialize project**

npm init -y

**3. Install dependencies**

npm install express mongoose dotenv bcryptjs jsonwebtoken cors

**4. Install dev dependencies**

npm install nodemon --save-dev

**5. Add start script in package.json**

"scripts": {

"start": "nodemon server.js"

}

**6. Create required folders & files**

mkdir config controllers middleware models routes

touch .env server.js

touch config/db.js

touch controllers/userController.js

touch middleware/authMiddleware.js

touch models/User.js

touch routes/userRoutes.js

**7. Add MongoDB and JWT configs in .env**

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_jwt\_secret

**8. Run the backend**

npm start

# PART 2: FRONTEND (Angular Standalone Components)

📁 Angular Project Structure:

angular-frontend/

├── src/

│ ├── app/

│ │ ├── home/

│ │ ├── login/

│ │ ├── register/

│ │ ├── profile/

│ │ ├── auth.guard.ts

│ │ ├── auth.service.ts

│ │ └── app.routes.ts

├── angular.json

├── package.json

🧑‍🍳 Step-by-Step Setup:

**1. Create Angular project**

ng new angular-frontend --standalone

cd angular-frontend

**2. Install Angular Material & Bootstrap**

ng add @angular/material

npm install bootstrap

**3. Update angular.json styles section**

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

"src/styles.css"

]

**4. Create components**

ng generate component home --standalone

ng generate component login --standalone

ng generate component register --standalone

ng generate component profile --standalone

**5. Create services and guards**

ng generate service auth

ng generate guard auth

**6. Set up routing in app.routes.ts**

- Configure routes for: home, login, register, profile

- Use canActivate guard for profile route

**7. Run frontend**

ng serve

# Connecting Angular to Express (Backend API)

⚙️ CORS Setup (in backend):

- In server.js:

const cors = require('cors');

app.use(cors());

🔐 AuthService (Frontend):

- Handles:

/register → Register user

/login → Login user and store JWT

/profile → Get profile using JWT

# Summary of All Terminal Commands Used

# Backend

mkdir user-auth-api

cd user-auth-api

npm init -y

npm install express mongoose dotenv bcryptjs jsonwebtoken cors

npm install nodemon --save-dev

mkdir config controllers middleware models routes

touch .env server.js

touch config/db.js

touch controllers/userController.js

touch middleware/authMiddleware.js

touch models/User.js

touch routes/userRoutes.js

npm start

# Frontend

ng new angular-frontend --standalone

cd angular-frontend

ng add @angular/material

npm install bootstrap

ng generate component home --standalone

ng generate component login --standalone

ng generate component register --standalone

ng generate component profile --standalone

ng generate service auth

ng generate guard auth

ng serve

# Final Steps to Test

1. Open backend: http://localhost:5000/api/users

2. Open frontend: http://localhost:4200

3. Test:

- Register a new user

- Login

- Access Profile → shows full data

- Navigate between pages using buttons (not navbar)