Angular Frontend Development Guide with Node.js Backend

This document provides a detailed step-by-step process for building the frontend of a web application using Angular, with Node.js + MongoDB as the backend. The guide includes project setup, authentication, CRUD operations, routing, and integration with backend APIs.

Step 1: Install Angular CLI

Run `npm install -g @angular/cli`. Verify with `ng version`. This installs the Angular framework globally.

Step 2: Create Angular Project

Use `ng new frontend-app`. Navigate into the folder and start the dev server with `ng serve`. Access app at http://localhost:4200.

Step 3: Setup Angular Modules & Routing

Enable routing during project creation or manually configure it. Define routes in `app-routing.module.ts`.

Step 4: Install Angular Material

Run `ng add @angular/material` to use Material UI components like forms, tables, and buttons.

Step 5: Create Authentication Module

Generate components: signup, signin, signout. Create an `AuthService` to interact with your backend APIs.

Step 6: Token Handling

Use HttpInterceptor to attach JWT tokens to each request and manage user authentication state.

Step 7: Create Posts Module

Generate components: PostList, PostDetail, PostForm. Create `PostService` to handle CRUD operations with backend.

Step 8: Protect Routes

Use Angular route guards to restrict access to authenticated users only.

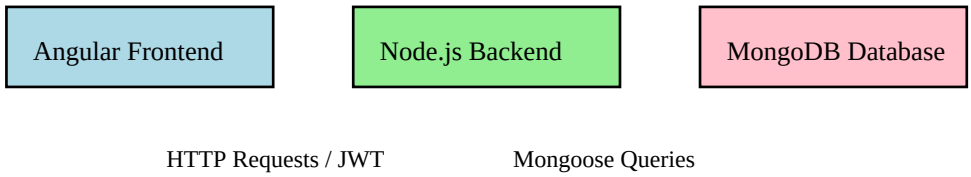
Step 9: Build UI

Use Angular Material components (tables, dialogs, forms) to design user-friendly UI for posts and authentication.

Step 10: Run & Test

Run `ng serve`, test authentication flow and CRUD operations with backend APIs.

System Flow Diagram



Code Snippets

AuthService

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private apiUrl = 'http://localhost:8000/api/auth';
  constructor(private http: HttpClient) {}
  signup(data: any): Observable<any> { return this.http.post(`${this.apiUrl}/signup`, data); }
  signin(data: any): Observable<any> { return this.http.post(`${this.apiUrl}/signin`, data); }
  logout() { localStorage.removeItem('token'); }
  saveToken(token: string) { localStorage.setItem('token', token); }
  getToken() { return localStorage.getItem('token'); }
}
```

TokenInterceptor

```
import { Injectable } from '@angular/core';
import { HttpInterceptor, HttpRequest, HttpHandler } from '@angular/common/http';
import { AuthService } from './auth.service';

@Injectable()
export class TokenInterceptor implements HttpInterceptor {
  constructor(private auth: AuthService) {}
  intercept(req: HttpRequest<any>, next: HttpHandler) {
    const token = this.auth.getToken();
    if (token) {
```

```
      const cloned = req.clone({ headers: req.headers.set('Authorization', `Bearer ${token}`) });
      return next.handle(cloned);
    }
    return next.handle(req);
  }
}
```

PostService

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({ providedIn: 'root' })
export class PostService {
  private apiUrl = 'http://localhost:8000/api/posts';
  constructor(private http: HttpClient) {}
  getAllPosts(): Observable<any> { return this.http.get(`${this.apiUrl}/all-posts`); }
  getPost(id: string): Observable<any> { return this.http.get(`${this.apiUrl}/single-posts?id=${id}`
  createPost(data: any): Observable<any> { return this.http.post(`${this.apiUrl}/create-posts`, data
  updatePost(data: any): Observable<any> { return this.http.put(`${this.apiUrl}/update-posts`, data)
  deletePost(id: string): Observable<any> { return this.http.delete(`${this.apiUrl}/delete-posts?id=$
}
```

AuthGuard

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { AuthService } from './auth.service';

@Injectable({ providedIn: 'root' })
export class AuthGuard implements CanActivate {
  constructor(private auth: AuthService, private router: Router) {}
  canActivate(): boolean {
    if (this.auth.getToken()) return true;
    this.router.navigate(['/signin']);
    return false;
  }
}
```