\*\*เหตุผลที่ต้องบันทึก token ลงฐานข้อมูล (user\_tokens):\*\*

1. \*\*ควบคุมการใช้งาน token:\*\*

- สามารถ revoke (ยกเลิก) token ได้ เช่น ตอน logout, เปลี่ยนรหัสผ่าน, หรือ admin ต้องการบังคับ logout ทุก device

- ตรวจสอบว่า token ที่ client ส่งมา ยังถูกต้องและไม่ถูกยกเลิก

2. \*\*รองรับ multi-device:\*\*

- 1 user อาจ login ได้หลายอุปกรณ์ (แต่ละอุปกรณ์มี token ของตัวเอง)

- สามารถจัดการ token แต่ละอันแยกกันได้

3. \*\*เพิ่มความปลอดภัย:\*\*

- ป้องกันการใช้ token เก่า (replay attack)

- ตรวจสอบประวัติการใช้งานย้อนหลังได้

4. \*\*Audit & Monitoring:\*\*

- ตรวจสอบว่า user เคย login จากที่ไหน เมื่อไร ใช้ token อะไร

5. \*\*รองรับ refresh token หรือ session management:\*\*

- ถ้าต้องการระบบ refresh token หรือ session แบบละเอียด จำเป็นต้องเก็บ token ในฐานข้อมูล

---

\*\*สรุป:\*\*

การบันทึก token ช่วยให้ระบบปลอดภัย ควบคุม session ได้ละเอียด และตรวจสอบย้อนหลังได้

เหมาะกับระบบที่ต้องการความปลอดภัยสูงและรองรับ multi-device/multi-session

pethzero: import { Controller, Post, Body } from '@nestjs/common';

import \* as bcrypt from 'bcrypt';

@Controller('api/dummy')

export class DummyController {

@Post('hash-password')

async hashPassword(@Body('password') password: string) {

if (!password) {

return { error: 'Password is required' };

}

const hash = await bcrypt.hash(password, 10);

return { hash };

}

}

กัน error หน่อย คือเรามีส่ง csrfProtection ด้วย

GitHub Copilot: ถ้าคุณมีการใช้ CSRF Protection (เช่น middleware หรือ guard ที่เช็ค CSRF token) แล้วอาจเกิด error ได้เมื่อ token ไม่ถูกต้อง หรือไม่มี token ส่งมา

แนะนำให้จับ error และตอบกลับให้เหมาะสม เช่น ใช้ try-catch:

````typescript

import { Controller, Post, Body, Req, HttpException, HttpStatus } from '@nestjs/common';

import \* as bcrypt from 'bcrypt';

@Controller('api/dummy')

export class DummyController {

@Post('hash-password')

async hashPassword(@Body('password') password: string, @Req() req) {

try {

if (!password) {

return { error: 'Password is required' };

}

// ถ้ามีการเช็ค CSRF ด้วย middleware แล้ว error จะถูก throw ออกมา

const hash = await bcrypt.hash(password, 10);

return { hash };

} catch (error) {

// ตรวจสอบว่าเป็น CSRF error หรือไม่

if (error.code === 'EBADCSRFTOKEN' || error.message?.includes('csrf')) {

throw new HttpException('Invalid CSRF token', HttpStatus.FORBIDDEN);

}

throw new HttpException('Internal server error', HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

}

````

\*\*หมายเหตุ:\*\*

- ถ้าใช้ `csurf` middleware, error ที่เกิดขึ้นจะมี code เป็น `EBADCSRFTOKEN`

- คุณอาจต้องปรับแต่งตามวิธีที่คุณ implement CSRF ในโปรเจกต์ของคุณ

pethzero: เราส่ง http://localhost:3000/api/dummy/hash-password

[Nest] 13300 - 23/07/2568 23:33:13 ERROR [ExceptionsHandler] ForbiddenError: invalid csrf token

at csrf (E:\PROJECT\kornpassorn-care\backend\node\_modules\csurf\index.js:112:19)

at <anonymous> (E:\PROJECT\kornpassorn-care\backend\src\main.ts:30:5)

at Layer.handleRequest (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\lib\layer.js:152:17)

at trimPrefix (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:342:13)

at E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:297:9

at processParams (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:582:12)

at next (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:291:5)

at E:\PROJECT\kornpassorn-care\backend\node\_modules\body-parser\lib\read.js:132:5

at AsyncResource.runInAsyncScope (node:async\_hooks:214:14)

at invokeCallback (E:\PROJECT\kornpassorn-care\backend\node\_modules\raw-body\index.js:238:16) {

code: 'EBADCSRFTOKEN'

}

มีวิธีแก้ไขข้อผิดพลาดนี้ไหม

GitHub Copilot: ปัญหานี้เกิดจาก \*\*csurf middleware\*\* ตรวจสอบ CSRF token แล้วไม่พบ token ที่ถูกต้องใน request

\*\*วิธีแก้ไข\*\*:

### 1. ต้องขอ CSRF token ก่อน

- เรียก `GET /api/auth/csrf-token` ก่อน เพื่อรับค่า token

- ตัวอย่าง response:

```json

{ "csrfToken": "xxx..." }

```

### 2. ส่ง CSRF token กลับไปใน request

- เวลาส่ง POST `/api/dummy/hash-password`

ให้แนบ CSRF token ไปใน header หรือ body หรือ cookie (ขึ้นกับ config csurf)

- ตัวอย่าง (header):

```

XSRF-TOKEN: xxx... // หรือชื่อ header ตามที่ csurf ใช้

```

### 3. ตัวอย่างการทดสอบด้วย curl

```bash

# ขอ CSRF token

curl -c cookie.txt http://localhost:3000/api/auth/csrf-token

# ส่ง hash-password พร้อม CSRF token

curl -b cookie.txt -H "XSRF-TOKEN: <token ที่ได้>" -H "Content-Type: application/json" -d '{"password":"123456"}' http://localhost:3000/api/dummy/hash-password

```

### หมายเหตุ

- ถ้าใช้ Postman ให้แนบ cookie และ header ให้ถูกต้อง

- ถ้า frontend, ต้องแนบ token ทุกครั้งที่เรียก API ที่ป้องกัน CSRF

\*\*สรุป:\*\*

ต้องขอ CSRF token ก่อน แล้วแนบ token ไปกับ request ทุกครั้งที่เรียก API ที่ถูกป้องกันด้วย csurf

ถ้าไม่แนบ token จะเจอ error `ForbiddenError: invalid csrf token`

pethzero: const isMatch = await bcrypt.compare(password, user.passwordHash);

admin123 $2b$10$abc123hashedAdminPassword false

ทำไหม false ละ

เรา const hash = await bcrypt.hash(password, 10); เลยนะ

GitHub Copilot: สาเหตุที่ `bcrypt.compare(password, user.passwordHash)` คืนค่า `false` ทั้งที่ใช้ `const hash = await bcrypt.hash(password, 10);`

\*\*เพราะ\*\*:

- ทุกครั้งที่เรียก `bcrypt.hash(password, 10)` จะได้ hash ใหม่ (salt ใหม่)

- ถ้า hash password แล้วเอา hash ไปเทียบกับ password เดิมโดย hash ใหม่อีกครั้ง จะไม่ตรงกัน

\*\*วิธีที่ถูกต้อง:\*\*

- ตอนสมัคร/สร้าง user: hash password แล้วเก็บ hash ใน DB

- ตอน login: ใช้ `bcrypt.compare(plainPassword, hashFromDB)`

ไม่ต้อง hash password ใหม่ก่อนเทียบ

\*\*ตัวอย่างผิด (จะไม่ตรงกัน):\*\*

```typescript

const hash1 = await bcrypt.hash('admin123', 10);

const hash2 = await bcrypt.hash('admin123', 10);

console.log(hash1 === hash2); // false

```

\*\*ตัวอย่างถูก:\*\*

```typescript

const hash = await bcrypt.hash('admin123', 10);

// ...บันทึก hash ลง DB...

const isMatch = await bcrypt.compare('admin123', hash); // true

```

\*\*สรุป:\*\*

อย่า hash password ใหม่ก่อน compare ให้ใช้ compare กับ hash ที่บันทึกไว้ใน DB เท่านั้น

pethzero: ไม่ได้บันทึกนะ คือ

const user = await this.databaseService.findUserByUsername(username);

console.log('Validating user:', username);

console.log('User found:', user);

if (!user) return null;

const isMatch = await bcrypt.compare(password, user.passwordHash);

console.log(password, user.passwordHash, isMatch);

if (!isMatch) return null;

return user;

}

password = 'admin123'

passwordHash =$2b$10$effUa3IemwzgKSQITCwg9eEq9cMTeeLJIdMevn5RYy3JBMSSc8Bxq

GitHub Copilot: จากข้อมูลที่ให้มา

- password = `'admin123'`

- passwordHash = `'$2b$10$effUa3IemwzgKSQITCwg9eEq9cMTeeLJIdMevn5RYy3JBMSSc8Bxq'`

- ใช้ `bcrypt.compare(password, user.passwordHash)` แล้วได้ `false`

\*\*สาเหตุที่เป็นไปได้:\*\*

1. \*\*passwordHash ไม่ได้มาจากการ hash `'admin123'`\*\*

อาจจะ hash จาก password อื่น หรือมีการ hash ซ้ำหลายรอบ

2. \*\*มีการ trim/เปลี่ยนแปลง password ก่อน hash หรือ compare\*\*

เช่น มี space, ตัวอักษรพิเศษ, หรือ encoding ไม่ตรงกัน

3. \*\*passwordHash ถูกแก้ไข/คัดลอกผิด\*\*

เช่น ขาดบางส่วน หรือมีอักขระเกิน

\*\*วิธีตรวจสอบ:\*\*

1. ทดสอบ hash ด้วยตัวเอง:

```typescript

const bcrypt = require('bcrypt');

const password = 'admin123';

const hash = '$2b$10$effUa3IemwzgKSQITCwg9eEq9cMTeeLJIdMevn5RYy3JBMSSc8Bxq';

bcrypt.compare(password, hash).then(console.log); // ควรได้ true ถ้าตรงกัน

```

2. ลอง hash `'admin123'` ใหม่ แล้ว compare กับ hash เดิม:

```typescript

const newHash = await bcrypt.hash('admin123', 10);

console.log(newHash); // จะไม่เหมือน hash เดิม แต่ compare กับ password เดิมควรได้ true

```

\*\*ข้อควรระวัง:\*\*

- อย่า hash password ใหม่แล้วเอาไป compare กับ hash เดิม

- ต้องใช้ hash ที่ได้จากการ hash ครั้งแรกเท่านั้น

\*\*สรุป:\*\*

- ถ้า `bcrypt.compare('admin123', '$2b$10$effUa3IemwzgKSQITCwg9eEq9cMTeeLJIdMevn5RYy3JBMSSc8Bxq')` ได้ `false` แปลว่า hash นี้ไม่ได้มาจาก `'admin123'`

- ให้ลอง hash `'admin123'` ใหม่ แล้วใช้ hash ที่ได้ไปเก็บใน DB แทน

ถ้าอยากเช็ค hash ว่าตรงกับ password จริงไหม ลองรันโค้ดนี้ใน Node.js:

```javascript

const bcrypt = require('bcrypt');

bcrypt.compare('admin123', '$2b$10$effUa3IemwzgKSQITCwg9eEq9cMTeeLJIdMevn5RYy3JBMSSc8Bxq').then(console.log);

```

ถ้าได้ `false` แปลว่า hash ไม่ตรงกับ password นี้แน่นอน

pethzero: ได้เราแต่เราเจอ {

"message": "CSRF token not available"

}

คือเรารัน http://localhost:3000/api/auth/csrf-token

{

"csrfToken": "w0USzQFW-qjKq8g4YXunnit6f6B1WP4tx8dM"

}

แล้ว postman ทำยังไงต่อ

GitHub Copilot: หลังจากที่คุณได้ CSRF token จาก `GET /api/auth/csrf-token` แล้ว

\*\*ขั้นตอนใน Postman สำหรับ login:\*\*

1. \*\*เรียก `GET /api/auth/csrf-token`\*\*

- เก็บค่า `csrfToken` ที่ได้จาก response

- เก็บ cookie ที่ได้ (Postman จะเก็บให้อัตโนมัติถ้าใช้ "Cookies" tab)

2. \*\*เรียก `POST /api/auth/login`\*\*

- ใส่ body เป็น JSON เช่น

```json

{

"username": "your\_username",

"password": "your\_password"

}

```

- เพิ่ม header:

```

XSRF-TOKEN: <ค่าที่ได้จากข้อ 1>

```

หรือชื่อ header ตามที่ csurf กำหนด (เช่น `x-csrf-token` หรือ `csrf-token`)

- ตรวจสอบว่า cookie จากข้อ 1 ถูกส่งไปด้วย (Postman จะส่งให้อัตโนมัติถ้าอยู่ใน session เดียวกัน)

3. \*\*ถ้ายังเจอ "CSRF token not available"\*\*

- ตรวจสอบว่า request header ชื่อถูกต้อง

- ตรวจสอบว่า cookie ถูกส่งไปด้วย

- ตรวจสอบว่า backend ใช้ชื่อ header ตรงกับที่ csurf config ไว้หรือไม่ (เช่น `XSRF-TOKEN` หรือ `x-csrf-token`)

---

\*\*สรุปตัวอย่างใน Postman:\*\*

- Step 1: GET `/api/auth/csrf-token` → copy ค่า token

- Step 2: POST `/api/auth/login`

- Body: JSON ตามปกติ

- Header: `XSRF-TOKEN: <token>`

- Cookie: Postman จะส่งให้อัตโนมัติ

ถ้ายังไม่ได้ ให้แจ้งชื่อ header ที่ backend csurf ใช้ หรือแนบ config csurf เพิ่มเติมครับ

pethzero: curl --location 'http://localhost:3000/api/auth/login' \

--header 'Content-Type: application/json' \

--header 'Cookie: \_csrf=e0\_5d-obn-0LDjoLNZyBo2jj; token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI2ZGY4YzUwYS0zNDc3LTQ3OTQtOGQwZi1hNTJkZjczYTlkNzQiLCJ1c2VybmFtZSI6ImFkbWluIiwicm9sZSI6ImFkbWluIiwiaWF0IjoxNzUzMjg4ODk0LCJleHAiOjE3NTMyOTA2OTR9.sSu6UgMw5zqhGtfG3e6rjVU0unjAJIYr0Ha\_xHhW89M' \

--data '{

"username": "admin",

"password": "admin123"

}'

ต้องแปลงแบบไหน

GitHub Copilot: ถ้าคุณใช้ \*\*csurf\*\* middleware แบบ default (หรือแบบที่ตั้งชื่อ cookie ว่า `\_csrf`)

\*\*คุณต้องส่งค่า CSRF token ใน header ด้วย\*\* (ไม่ใช่แค่ใน cookie)

\*\*ตัวอย่าง curl ที่ถูกต้อง:\*\*

````bash

curl --location 'http://localhost:3000/api/auth/login' \

--header 'Content-Type: application/json' \

--header 'Cookie: \_csrf=e0\_5d-obn-0LDjoLNZyBo2jj' \

--header 'X-CSRF-Token: e0\_5d-obn-0LDjoLNZyBo2jj' \

--data '{

"username": "admin",

"password": "admin123"

}'

````

\*\*สรุป:\*\*

- ส่ง cookie `\_csrf` (Postman จะส่งให้อัตโนมัติถ้าอยู่ใน session เดียวกัน)

- ส่ง header `X-CSRF-Token` (หรือ `x-csrf-token` ขึ้นกับ config) โดยใช้ค่าเดียวกับที่ได้จาก `/api/auth/csrf-token`

\*\*หมายเหตุ:\*\*

- ไม่จำเป็นต้องส่ง cookie `token` ใน request login (เพราะยังไม่ได้ login)

- ถ้าใช้ชื่อ header อื่น ให้ดูที่ config csurf ของคุณ

\*\*สรุปขั้นตอน:\*\*

1. GET `/api/auth/csrf-token` → ได้ค่า token เช่น `e0\_5d-obn-0LDjoLNZyBo2jj`

2. POST `/api/auth/login`

- ใส่ cookie `\_csrf=<token>`

- ใส่ header `X-CSRF-Token: <token>`

- ใส่ body เป็น JSON

\*\*ตัวอย่าง header ที่ต้องมี:\*\*

```

Cookie: \_csrf=e0\_5d-obn-0LDjoLNZyBo2jj

X-CSRF-Token: e0\_5d-obn-0LDjoLNZyBo2jj

```

\*\*เท่านี้ก็จะไม่เจอ "CSRF token not available" แล้ว\*\*

pethzero: @Post('login')

async login(@Body() body: LoginDto, @Req() req: Request, @Res() res: Response) {

const user = await this.authService.validateUser(body.username, body.password);

console.log(user);

if (!user) return res.status(401).json({ message: 'Invalid credentials' });

const token = this.authService.generateJwt({

sub: user.id,

username: user.username,

role: user.role,

});

console.log('CSRF Token:', req.csrfToken());

res.cookie('token', token, {

httpOnly: true,

secure: false,

sameSite: 'strict',

});

if (req.csrfToken) {

return res.json({ csrfToken: req.csrfToken() });

}

return res.status(500).json({ message: 'CSRF token not available' });

}

@Get('csrf-token')

getCsrfToken(@Req() req: Request) {

if (req.csrfToken) {

return { csrfToken: req.csrfToken() }; // ส่งคืน CSRF token

} else {

return { message: 'CSRF protection not applied properly' }; // แจ้งเตือนถ้า csrfToken() ไม่สามารถใช้งานได้

}

}

มันดค่า Token ยังไง แบบว่าเรา get ไปแล้วใช่ไหม

GitHub Copilot: การทำงานของ CSRF token ใน flow นี้คือ:

1. \*\*GET `/api/auth/csrf-token`\*\*

- เมื่อ client (เช่น Postman หรือ frontend) เรียก endpoint นี้

- ถ้า middleware csurf ทำงานถูกต้อง `req.csrfToken()` จะมีค่า

- Controller จะ return `{ csrfToken: <token> }`

- \*\*Client ต้องเก็บค่า token นี้ไว้\*\* (เช่น ในตัวแปร, localStorage, หรือ cookie)

2. \*\*POST `/api/auth/login`\*\*

- เวลาจะ login ต้องแนบ CSRF token ที่ได้จากข้อ 1 ไปกับ request

- โดยปกติ csurf จะตรวจสอบ token จาก header (`X-CSRF-Token` หรือ `x-csrf-token`) หรือจาก body/cookie ตาม config

- ถ้า token ถูกต้อง จะ login ได้

- ถ้า token ไม่ถูกต้อง จะ error

\*\*สรุป:\*\*

- \*\*GET\*\* `/api/auth/csrf-token` เพื่อขอ token

- \*\*POST\*\* `/api/auth/login` พร้อมแนบ token ที่ได้ (ใน header หรือ body)

- ฝั่ง backend ไม่จำเป็นต้อง return token ซ้ำใน response ของ login (return แค่ access token ก็พอ)

- ที่คุณเขียน `return res.json({ csrfToken: req.csrfToken() });` ใน login จริงๆ ไม่จำเป็น (ควร return access token แทน)

---

\*\*ตัวอย่าง flow ที่แนะนำ:\*\*

1. GET `/api/auth/csrf-token`

→ ได้ `{ csrfToken: "xxx" }`

2. POST `/api/auth/login`

- Header: `X-CSRF-Token: xxx`

- Body: `{ "username": "...", "password": "..." }`

- ได้ response: `{ access\_token: "..." }`

---

\*\*สรุป:\*\*

- CSRF token ต้องขอแยกก่อน แล้วแนบไปกับ request ที่ต้องการป้องกัน CSRF

- ไม่ต้อง return CSRF token ใน response ของ login อีก

pethzero: เขียน

curl --location 'http://localhost:3000/api/auth/login' \

--header 'Content-Type: application/json' \

--header 'Cookie: \_csrf=fKdVC6Jv-kOThnKEKK68nLAJZmCvg18XmkT8' \

--header 'X-CSRF-Token: fKdVC6Jv-kOThnKEKK68nLAJZmCvg18XmkT8' \

--data '{

"username": "admin",

"password": "admin123"

}'

{

"csrfToken": "fKdVC6Jv-kOThnKEKK68nLAJZmCvg18XmkT8"

}

ทำไหมยังไหมได้

GitHub Copilot: สาเหตุที่ยังไม่ได้ แม้จะส่งทั้ง cookie และ header แล้ว อาจเกิดจาก:

1. \*\*csurf middleware ไม่ถูก apply กับ route นี้\*\*

ตรวจสอบว่าได้ใช้ csurf middleware กับ `/api/auth/login` หรือไม่

2. \*\*ตำแหน่งการ apply middleware\*\*

ถ้าใช้กับ global หรือเฉพาะบาง route ต้องแน่ใจว่า `/api/auth/login` อยู่ใน scope ที่มี csurf

3. \*\*ชื่อ cookie/header ไม่ตรงกับ config\*\*

ถ้า config csurf ใช้ชื่ออื่น (ไม่ใช่ `\_csrf` หรือ `X-CSRF-Token`) ต้องใช้ชื่อเดียวกัน

4. \*\*Session/cookie ไม่ตรงกัน\*\*

ต้องใช้ session/cookie เดียวกับที่ได้ CSRF token (เช่น ต้องใช้ `-c` และ `-b` ใน curl)

---

### ตัวอย่าง curl ที่ถูกต้อง (พร้อมเก็บ cookie)

\*\*1. ขอ CSRF token และเก็บ cookie\*\*

```bash

curl -c cookie.txt http://localhost:3000/api/auth/csrf-token

```

จะได้ค่า token เช่น `"csrfToken": "fKdVC6Jv-kOThnKEKK68nLAJZmCvg18XmkT8"`

\*\*2. ส่ง login พร้อม cookie และ header\*\*

```bash

curl -b cookie.txt \

--header "Content-Type: application/json" \

--header "X-CSRF-Token: fKdVC6Jv-kOThnKEKK68nLAJZmCvg18XmkT8" \

--data '{"username":"admin","password":"admin123"}' \

http://localhost:3000/api/auth/login

```

---

\*\*สรุป:\*\*

- ต้องใช้ cookie เดียวกับที่ได้ CSRF token

- ต้องแน่ใจว่า csurf middleware ถูก apply กับ route

- ต้องใช้ชื่อ header/cookie ให้ตรงกับ config

ถ้ายังไม่ได้ ให้เช็ค error log ฝั่ง backend เพิ่มเติม หรือแนบ config csurf ที่ใช้งานมาด้วยครับ

pethzero: หรือ เพราะเราต้องให้ ได้เฉพาะ anular เพรากัน โจมตีจาก postman อะไรพวกนี้ ฉันเข้าใจถูกใช่ไหม

GitHub Copilot: คุณเข้าใจถูกต้องครับ

\*\*CSRF\*\* (Cross-Site Request Forgery) ถูกออกแบบมาเพื่อป้องกัน "browser-based attack" เท่านั้น

- CSRF จะป้องกันไม่ให้ browser อื่น (เช่น malicious site) ส่ง request มายัง API ของคุณโดยที่ user ไม่รู้ตัว

- \*\*CSRF ไม่ได้ป้องกันเครื่องมืออย่าง Postman หรือ curl\*\* เพราะเครื่องมือเหล่านี้ให้คุณควบคุม header/cookie เองได้

\*\*สรุป:\*\*

- ถ้า API ของคุณจะถูกเรียกจาก frontend (Angular) เท่านั้น และ user ใช้งานผ่าน browser จริง

- CSRF จะช่วยป้องกันการโจมตีจากเว็บอื่นที่พยายามส่ง request มายัง API ของคุณ

- แต่ถ้าใครใช้ Postman/curl แล้วแนบ cookie + CSRF token ถูกต้อง ก็จะผ่านได้ (เพราะถือว่าเป็น "user ที่ตั้งใจ" ไม่ใช่ browser ที่โดนหลอก)

\*\*หมายเหตุ:\*\*

- ถ้าอยากป้องกัน Postman/curl จริง ๆ ต้องใช้วิธีอื่น เช่น ตรวจสอบ User-Agent, IP, หรือ require OAuth/2FA

- CSRF ไม่ได้ออกแบบมาเพื่อป้องกัน API testing tool

\*\*สรุปอีกที:\*\*

- CSRF ป้องกัน browser-based attack

- ไม่ได้ป้องกัน Postman/curl

- ใช้ถูกต้องแล้วสำหรับ Angular frontend ครับ

pethzero: CSRF ไม่ได้ป้องกันเครื่องมืออย่าง Postman หรือ curl เพราะเครื่องมือเหล่านี้ให้คุณควบคุม header/cookie เองได้

แล้วทำไมยัง post ไม่ได้

GitHub Copilot: สาเหตุที่คุณยัง \*\*POST ไม่ได้\*\* ทั้งที่ CSRF ไม่ได้ป้องกัน Postman/curl โดยตรง มีความเป็นไปได้ดังนี้:

1. \*\*csurf ยังตรวจสอบ token อยู่\*\*

ถึงจะใช้ Postman/curl ได้ แต่คุณ \*\*ต้องส่ง cookie และ header ให้ถูกต้อง\*\* ตามที่ csurf ต้องการ

- ถ้าไม่ส่ง cookie หรือ header ที่ถูกต้อง csurf ก็จะ block เหมือนกัน

2. \*\*config csurf ใน main.ts\*\*

คุณตั้ง `skipPaths = ['/api/auth/login', '/api/auth/guest']`

หมายความว่า route `/api/auth/login` \*\*ไม่ถูกตรวจสอบ CSRF\*\*

ดังนั้น ถ้า POST `/api/auth/login` แล้ว error ไม่ใช่เพราะ CSRF

แต่ถ้า POST ไป path อื่นที่ไม่อยู่ใน skipPaths จะถูกตรวจสอบ CSRF

3. \*\*ปัญหาอื่นที่ไม่เกี่ยวกับ CSRF\*\*

- อาจเป็นปัญหา validation, auth, หรือ logic อื่นใน controller/service

---

### วิธีตรวจสอบ

- ถ้า POST `/api/auth/login` แล้ว error เป็น \*\*CSRF\*\* (เช่น `invalid csrf token`)

→ แปลว่า path นี้ยังถูกตรวจสอบ CSRF อยู่

→ ให้เช็คว่า path ที่ส่ง POST ตรงกับ `/api/auth/login` จริงหรือไม่ (เช่น `/api/auth/login/` กับ `/api/auth/login` อาจไม่เหมือนกันในบาง config)

- ถ้า error อื่น ให้ดู error message ที่ backend log

---

### ข้อควรระวัง

- ถ้าใช้ global prefix (`api`) แล้ว path ที่ส่งต้องเป็น `/api/auth/login`

- ถ้า path ไม่ตรงกับที่อยู่ใน `skipPaths` จะถูกตรวจสอบ CSRF

---

### สรุป

- \*\*Postman/curl\*\* สามารถ bypass CSRF ได้ถ้า config ให้ path นั้น skip CSRF

- ถ้า POST ไม่ได้ ให้ดู error message ที่ backend log

- ตรวจสอบ path ที่ส่ง POST ว่าตรงกับที่ skip จริงหรือไม่

- ตรวจสอบว่า error ที่ได้เป็น CSRF หรือ error อื่น

ถ้าอยากให้ช่วยวิเคราะห์เพิ่ม กรุณาส่ง error message ที่เจอขณะ POST `/api/auth/login` ครับ

pethzero: ถ้าออกแบบ ระบบ login ควรจะใช้ csrfToken ไหม หรือมใช้ตอนเรียกตัวอื่นแบ สุมมติปกติ เรามี table คนไข้ เหมา่ะแบบไหนยังไงอธิบายที

GitHub Copilot: \*\*สรุปสั้น:\*\*

- \*\*ระบบ login\*\* (POST /auth/login) \*\*ไม่จำเป็นต้องใช้ CSRF token\*\*

- \*\*CSRF token\*\* ควรใช้กับ API ที่ "เปลี่ยนแปลงข้อมูล" (เช่น POST/PUT/DELETE) หลังจาก login แล้ว

- ตัวอย่างที่ควรใช้ CSRF: สั่งจ่ายยา, แก้ไขข้อมูลคนไข้, ลบข้อมูล ฯลฯ

- \*\*ไม่ควรใช้ CSRF กับ login/guest\*\* เพราะ user ยังไม่ได้รับ session/cookie ที่จะป้องกัน CSRF

---

### อธิบาย

#### 1. \*\*CSRF คืออะไร\*\*

- ป้องกันการโจมตีที่เกิดจาก browser อื่น (เช่น เว็บปลอม) ส่ง request มายัง API ที่ user login อยู่

- ใช้ได้ผลกับ request ที่ browser ส่งโดยอัตโนมัติ (เช่น cookie/session)

#### 2. \*\*ควรใช้ CSRF กับ endpoint ไหน\*\*

- \*\*ควรใช้\*\*:

- ทุก endpoint ที่ "เปลี่ยนแปลงข้อมูล" (POST/PUT/PATCH/DELETE)

- เช่น `/api/patient/create`, `/api/patient/update`, `/api/appointment/delete`

- \*\*ไม่ควรใช้\*\*:

- endpoint ที่ "ไม่ต้องการ session/cookie" เช่น login, guest login, public API

#### 3. \*\*login ไม่ควรใช้ CSRF\*\*

- เพราะ user ยังไม่มี session/cookie

- ถ้าใช้ CSRF กับ login จะทำให้ flow ซับซ้อนโดยไม่จำเป็น

- ส่วนใหญ่ระบบ auth จะใช้แค่ username/password + rate limit + brute force protection

#### 4. \*\*ตัวอย่าง flow ที่เหมาะสม\*\*

1. \*\*POST /auth/login\*\*

- ไม่ต้องใช้ CSRF

- รับ username/password → ออก token/cookie

2. \*\*GET /auth/csrf-token\*\*

- หลัง login แล้ว (browser มี cookie แล้ว)

- ขอ CSRF token สำหรับใช้กับ request ที่เปลี่ยนแปลงข้อมูล

3. \*\*POST /patient/create\*\*

- ต้องแนบ CSRF token

- Backend ตรวจสอบ CSRF ก่อนดำเนินการ

---

### \*\*สรุปแนวทางที่เหมาะสม\*\*

- \*\*Login:\*\* ไม่ต้องใช้ CSRF

- \*\*API ที่แก้ไขข้อมูล:\*\* ใช้ CSRF

- \*\*API ที่อ่านข้อมูล (GET):\*\* ไม่ต้องใช้ CSRF

---

\*\*อ้างอิง:\*\*

- [OWASP CSRF Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\_Request\_Forgery\_Prevention\_Cheat\_Sheet.html)

- [NestJS CSRF Example](https://docs.nestjs.com/security/csrf)

---

\*\*ตอบโจทย์:\*\*

- ระบบ login ไม่ต้องใช้ CSRF

- ใช้ CSRF กับ API ที่เปลี่ยนแปลงข้อมูลหลัง login (เช่น table คนไข้)

- ป้องกันการโจมตีจาก browser อื่นที่พยายามเปลี่ยนแปลงข้อมูลของ user

pethzero: ถ้าให้ ออกแบบ login ใหม่ยกระบบ แบบมือาชีพ ระดับขายได้

GitHub Copilot: การออกแบบระบบ \*\*login\*\* สำหรับ production ที่ปลอดภัยและยืดหยุ่น (ระดับขายได้) ควรคำนึงถึงทั้ง \*\*security\*\*, \*\*scalability\*\*, และ \*\*user experience\*\*

ตัวอย่างแนวทางที่เหมาะสมสำหรับระบบที่ใช้ NestJS + JWT + CSRF + Cookie:

---

## 1. \*\*Authentication Flow\*\*

### 1.1 \*\*Login (POST /auth/login)\*\*

- \*\*Input:\*\* username, password (ผ่าน HTTPS เท่านั้น)

- \*\*Validation:\*\* ตรวจสอบข้อมูล, ป้องกัน brute-force (rate limit, lockout)

- \*\*Password:\*\* เก็บ hash ด้วย bcrypt/scrypt/argon2

- \*\*Response:\*\*

- ออก JWT (access token) และ refresh token (ถ้าใช้)

- ส่ง JWT กลับผ่าน \*\*httpOnly cookie\*\* (ไม่ให้ JS access)

- \*\*ไม่ต้องใช้ CSRF กับ login\*\*

### 1.2 \*\*CSRF Token (GET /auth/csrf-token)\*\*

- หลัง login แล้ว (browser มี cookie แล้ว)

- ส่ง CSRF token กลับ (เช่น `{ csrfToken: "..." }`)

- Frontend ต้องแนบ token นี้ใน header ทุก request ที่เปลี่ยนแปลงข้อมูล

### 1.3 \*\*Logout (POST /auth/logout)\*\*

- ลบ cookie JWT

- (ถ้าใช้ refresh token ให้ revoke ด้วย)

---

## 2. \*\*Session & Token Management\*\*

- ใช้ \*\*JWT\*\* สำหรับ session (access token อายุสั้น เช่น 15-30 นาที)

- ใช้ \*\*refresh token\*\* (อายุยาวกว่า) สำหรับ renew access token

- เก็บ refresh token ใน httpOnly cookie หรือ database (สำหรับ revoke)

- ทุก request ที่ต้อง login ให้ตรวจสอบ JWT (ผ่าน AuthGuard)

---

## 3. \*\*CSRF Protection\*\*

- ใช้ \*\*csurf\*\* middleware เฉพาะกับ API ที่เปลี่ยนแปลงข้อมูล (POST/PUT/PATCH/DELETE)

- ไม่ต้องใช้กับ login/logout/csrf-token

- CSRF token ส่งผ่าน header (เช่น `X-CSRF-Token`)

- ตรวจสอบ CSRF เฉพาะเมื่อ JWT (หรือ session) ถูกต้อง

---

## 4. \*\*Security Best Practices\*\*

- \*\*HTTPS\*\* เท่านั้น

- \*\*Rate limiting\*\* ทุก endpoint สำคัญ (login, register)

- \*\*Brute-force protection\*\* (เช่น lock account หลังพยายามผิดหลายครั้ง)

- \*\*Audit log\*\* การ login/logout ทุกครั้ง

- \*\*CORS\*\* เฉพาะ origin ที่อนุญาต

- \*\*SameSite cookie\*\*: ใช้ `'strict'` หรือ `'lax'`

- \*\*httpOnly cookie\*\* สำหรับ token

- \*\*ไม่ return error detail\*\* ที่บอกว่า username หรือ password ผิด

---

## 5. \*\*User Experience\*\*

- รองรับ multi-device login (optional)

- รองรับ forgot password/reset password (ผ่าน email/OTP)

- รองรับ 2FA (optional)

- รองรับ social login (Google, Facebook, etc.) (optional)

---

## 6. \*\*ตัวอย่างโครงสร้าง API\*\*

| Method | Path | Description | CSRF | Auth Required |

|--------|---------------------|----------------------------|------|--------------|

| POST | /auth/login | Login | ❌ | ❌ |

| POST | /auth/logout | Logout | ❌ | ✅ |

| GET | /auth/csrf-token | Get CSRF token | ❌ | ✅ |

| POST | /users | Create user | ✅ | ✅ |

| PUT | /users/:id | Update user | ✅ | ✅ |

| DELETE | /users/:id | Delete user | ✅ | ✅ |

| GET | /users/me | Get current user profile | ❌ | ✅ |

---

## 7. \*\*ตัวอย่างโค้ด (Login Controller)\*\*

````typescript

@Post('login')

async login(@Body() body: LoginDto, @Res() res: Response) {

const user = await this.authService.validateUser(body.username, body.password);

if (!user) return res.status(401).json({ message: 'Invalid credentials' });

const accessToken = this.authService.generateJwt({ sub: user.id, username: user.username, role: user.role });

const refreshToken = this.authService.generateRefreshToken({ sub: user.id });

res.cookie('token', accessToken, {

httpOnly: true,

secure: true,

sameSite: 'strict',

maxAge: 1000 \* 60 \* 30, // 30 min

});

res.cookie('refreshToken', refreshToken, {

httpOnly: true,

secure: true,

sameSite: 'strict',

maxAge: 1000 \* 60 \* 60 \* 24 \* 7, // 7 days

});

return res.json({ message: 'Login successful' });

}

````

---

## 8. \*\*สรุปแนวคิด\*\*

- \*\*Login:\*\* ไม่ต้องใช้ CSRF, ใช้ JWT + httpOnly cookie

- \*\*หลัง login:\*\* ใช้ CSRF กับทุก endpoint ที่เปลี่ยนแปลงข้อมูล

- \*\*Security:\*\* ครบถ้วนทั้ง JWT, CSRF, HTTPS, rate limit, brute-force, audit log

- \*\*User Experience:\*\* รองรับฟีเจอร์ที่จำเป็นและขยายได้

---

\*\*ถ้าต้องการตัวอย่างโค้ดหรือ flow เพิ่มเติมในแต่ละส่วน แจ้งได้เลยครับ\*\*

pethzero: งั้นปรับนี้ให้หมดหน่อย

GitHub Copilot: แนวทางปรับปรุง \*\*ระบบ login\*\* ให้เหมาะสมและปลอดภัย (ตามมาตรฐาน production):

---

### 1. \*\*ไม่ต้องใช้ CSRF กับ login/guest\*\*

- ไม่ต้อง return `{ csrfToken: ... }` ใน response ของ login/guest

- ให้ return เฉพาะ access token หรือ message

---

### 2. \*\*login (POST /auth/login)\*\*

- ตรวจสอบ user

- ออก JWT

- ส่ง JWT กลับผ่าน httpOnly cookie

- return เฉพาะ message หรือข้อมูล user ที่จำเป็น

---

### 3. \*\*getCsrfToken (GET /auth/csrf-token)\*\*

- ใช้สำหรับขอ CSRF token หลัง login แล้วเท่านั้น

---

### 4. \*\*loginAsGuest\*\*

- ไม่ต้องใช้ CSRF

- return access token

---

\*\*ตัวอย่างโค้ดที่ปรับปรุงแล้ว:\*\*

````typescript

import { Controller, Post, Req, Res, Body, Get } from '@nestjs/common';

import { AuthService } from './auth.service';

import { Response, Request } from 'express';

import { LoginDto } from './dto/login.dto';

@Controller('auth')

export class AuthController {

constructor(private authService: AuthService) { }

@Post('login')

async login(@Body() body: LoginDto, @Res() res: Response) {

const user = await this.authService.validateUser(body.username, body.password);

if (!user) return res.status(401).json({ message: 'Invalid credentials' });

const token = this.authService.generateJwt({

sub: user.id,

username: user.username,

role: user.role,

});

res.cookie('token', token, {

httpOnly: true,

secure: false, // set true in production (HTTPS)

sameSite: 'strict',

});

return res.json({ message: 'Login successful' });

}

@Get('csrf-token')

getCsrfToken(@Req() req: Request) {

if (req.csrfToken) {

return { csrfToken: req.csrfToken() };

} else {

return { message: 'CSRF protection not applied properly' };

}

}

@Post('guest')

async loginAsGuest(@Res() res: Response) {

const result = this.authService.loginAsGuest();

res.cookie('token', result.access\_token, {

httpOnly: true,

secure: false,

sameSite: 'strict',

});

return res.json({ access\_token: result.access\_token });

}

}

````

---

\*\*สรุป:\*\*

- login/guest ไม่ต้อง return CSRF token

- getCsrfToken ใช้สำหรับขอ CSRF token หลัง login

- ทุก endpoint ที่เปลี่ยนแปลงข้อมูล (หลัง login) ให้แนบ CSRF token ใน header

ถ้าต้องการปรับ service/module เพิ่มเติม แจ้งได้เลย!

pethzero: คือ เรามี database แบบ

CREATE TABLE public.patients (

id SERIAL PRIMARY KEY,

patient\_code VARCHAR(20) UNIQUE NOT NULL,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

gender VARCHAR(10), -- male, female, other

date\_of\_birth DATE,

phone VARCHAR(20),

email VARCHAR(100),

address TEXT,

blood\_type VARCHAR(5), -- A, B, AB, O

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT now(),

updated\_at TIMESTAMP DEFAULT now()

);

CREATE TABLE public.medical\_records (

id SERIAL PRIMARY KEY,

patient\_id INT REFERENCES public.patients(id) ON DELETE CASCADE,

visit\_date TIMESTAMP NOT NULL,

symptoms TEXT,

diagnosis TEXT,

treatment TEXT,

doctor\_name VARCHAR(100),

note TEXT,

created\_at TIMESTAMP DEFAULT now()

);

เบื้องต้นอยากให้ backend ช่วย สร้าง module ที่เกี่ยวกับ ข้อมูลคนไข้ ประวัติการรักษา

GitHub Copilot: ตัวอย่างการออกแบบ \*\*NestJS module\*\* สำหรับข้อมูลคนไข้ (`patients`) และประวัติการรักษา (`medical\_records`)

จะประกอบด้วย entity, service, controller, และ module ครบถ้วน พร้อมโครงสร้าง CRUD เบื้องต้น

---

### 1. \*\*สร้าง Entity\*\*

````typescript

import { Entity, Column, PrimaryGeneratedColumn, OneToMany } from 'typeorm';

import { MedicalRecord } from './medical-record.entity';

@Entity({ name: 'patients' })

export class Patient {

@PrimaryGeneratedColumn()

id: number;

@Column({ unique: true, length: 20 })

patient\_code: string;

@Column({ length: 100 })

first\_name: string;

@Column({ length: 100 })

last\_name: string;

@Column({ length: 10, nullable: true })

gender: string;

@Column({ type: 'date', nullable: true })

date\_of\_birth: Date;

@Column({ length: 20, nullable: true })

phone: string;

@Column({ length: 100, nullable: true })

email: string;

@Column({ type: 'text', nullable: true })

address: string;

@Column({ length: 5, nullable: true })

blood\_type: string;

@Column({ default: true })

is\_active: boolean;

@Column({ type: 'timestamp', default: () => 'now()' })

created\_at: Date;

@Column({ type: 'timestamp', default: () => 'now()' })

updated\_at: Date;

@OneToMany(() => MedicalRecord, (record) => record.patient)

medical\_records: MedicalRecord[];

}

````

````typescript

import { Entity, Column, PrimaryGeneratedColumn, ManyToOne, JoinColumn } from 'typeorm';

import { Patient } from './patient.entity';

@Entity({ name: 'medical\_records' })

export class MedicalRecord {

@PrimaryGeneratedColumn()

id: number;

@Column()

patient\_id: number;

@ManyToOne(() => Patient, (patient) => patient.medical\_records, { onDelete: 'CASCADE' })

@JoinColumn({ name: 'patient\_id' })

patient: Patient;

@Column({ type: 'timestamp' })

visit\_date: Date;

@Column({ type: 'text', nullable: true })

symptoms: string;

@Column({ type: 'text', nullable: true })

diagnosis: string;

@Column({ type: 'text', nullable: true })

treatment: string;

@Column({ length: 100, nullable: true })

doctor\_name: string;

@Column({ type: 'text', nullable: true })

note: string;

@Column({ type: 'timestamp', default: () => 'now()' })

created\_at: Date;

}

````

---

### 2. \*\*สร้าง Service\*\*

````typescript

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { Patient } from '../../database/entities/patient.entity';

@Injectable()

export class PatientService {

constructor(

@InjectRepository(Patient)

private patientRepository: Repository<Patient>,

) {}

findAll() {

return this.patientRepository.find();

}

findOne(id: number) {

return this.patientRepository.findOne({ where: { id } });

}

create(data: Partial<Patient>) {

const patient = this.patientRepository.create(data);

return this.patientRepository.save(patient);

}

update(id: number, data: Partial<Patient>) {

return this.patientRepository.update(id, data);

}

remove(id: number) {

return this.patientRepository.delete(id);

}

}

````

````typescript

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { MedicalRecord } from '../../database/entities/medical-record.entity';

@Injectable()

export class MedicalRecordService {

constructor(

@InjectRepository(MedicalRecord)

private recordRepository: Repository<MedicalRecord>,

) {}

findAll() {

return this.recordRepository.find({ relations: ['patient'] });

}

findByPatient(patient\_id: number) {

return this.recordRepository.find({ where: { patient\_id }, relations: ['patient'] });

}

create(data: Partial<MedicalRecord>) {

const record = this.recordRepository.create(data);

return this.recordRepository.save(record);

}

update(id: number, data: Partial<MedicalRecord>) {

return this.recordRepository.update(id, data);

}

remove(id: number) {

return this.recordRepository.delete(id);

}

}

````

---

### 3. \*\*สร้าง Controller\*\*

````typescript

import { Controller, Get, Post, Body, Param, Put, Delete } from '@nestjs/common';

import { PatientService } from './patient.service';

@Controller('patients')

export class PatientController {

constructor(private readonly patientService: PatientService) {}

@Get()

findAll() {

return this.patientService.findAll();

}

@Get(':id')

findOne(@Param('id') id: number) {

return this.patientService.findOne(id);

}

@Post()

create(@Body() data: any) {

return this.patientService.create(data);

}

@Put(':id')

update(@Param('id') id: number, @Body() data: any) {

return this.patientService.update(id, data);

}

@Delete(':id')

remove(@Param('id') id: number) {

return this.patientService.remove(id);

}

}

````

````typescript

import { Controller, Get, Post, Body, Param, Put, Delete } from '@nestjs/common';

import { MedicalRecordService } from './medical-record.service';

@Controller('medical-records')

export class MedicalRecordController {

constructor(private readonly recordService: MedicalRecordService) {}

@Get()

findAll() {

return this.recordService.findAll();

}

@Get('patient/:patient\_id')

findByPatient(@Param('patient\_id') patient\_id: number) {

return this.recordService.findByPatient(patient\_id);

}

@Post()

create(@Body() data: any) {

return this.recordService.create(data);

}

@Put(':id')

update(@Param('id') id: number, @Body() data: any) {

return this.recordService.update(id, data);

}

@Delete(':id')

remove(@Param('id') id: number) {

return this.recordService.remove(id);

}

}

````

---

### 4. \*\*สร้าง Module\*\*

````typescript

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { Patient } from '../../database/entities/patient.entity';

import { PatientService } from './patient.service';

import { PatientController } from './patient.controller';

@Module({

imports: [TypeOrmModule.forFeature([Patient])],

providers: [PatientService],

controllers: [PatientController],

exports: [PatientService],

})

export class PatientModule {}

````

````typescript

// filepath: src/modules/medical-record/medical-record.module.ts

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { MedicalRecord } from '../../database/entities/medical-record.entity';

import { MedicalRecordService } from './medical-record.service';

import { MedicalRecordController } from './medical-record.controller';

@Module({

imports: [TypeOrmModule.forFeature([MedicalRecord])],

providers: [MedicalRecordService],

controllers: [MedicalRecordController],

exports: [MedicalRecordService],

})

export class MedicalRecordModule {}

````

---

### 5. \*\*เพิ่มเข้า AppModule\*\*

```typescript

// ...existing code...

import { PatientModule } from './modules/patient/patient.module';

import { MedicalRecordModule } from './modules/medical-record/medical-record.module';

@Module({

imports: [

// ...existing code...

PatientModule,

MedicalRecordModule,

],

// ...existing code...

})

export class AppModule {}

```

---

\*\*หมายเหตุ:\*\*

- สามารถปรับปรุง DTO, validation, และ auth guard เพิ่มเติมได้ตามต้องการ

- สามารถใช้ relations ใน TypeORM เพื่อ join ข้อมูลระหว่าง patient กับ medical record ได้

---

ถ้าต้องการ scaffold ไฟล์จริงใน workspace แจ้งได้เลย!

Similar code found with 1 license type

pethzero: แต่ละหัวข้อ อยากให้สร้าง path ได้ไหมเราไม่รู้ว่าต้องสร้างตรงไหน

GitHub Copilot: แน่นอนครับ! ด้านล่างนี้คือ \*\*สรุปไฟล์และ path ที่ควรสร้าง\*\* สำหรับระบบข้อมูลคนไข้และประวัติการรักษา (ตามมาตรฐาน NestJS + TypeORM)

\*\*คุณสามารถ copy ไปสร้างไฟล์ใน path ที่ระบุได้เลย\*\*

---

### 1. Entity

````typescript

import { Entity, Column, PrimaryGeneratedColumn, OneToMany } from 'typeorm';

import { MedicalRecord } from './medical-record.entity';

@Entity({ name: 'patients' })

export class Patient {

@PrimaryGeneratedColumn()

id: number;

@Column({ unique: true, length: 20 })

patient\_code: string;

@Column({ length: 100 })

first\_name: string;

@Column({ length: 100 })

last\_name: string;

@Column({ length: 10, nullable: true })

gender: string;

@Column({ type: 'date', nullable: true })

date\_of\_birth: Date;

@Column({ length: 20, nullable: true })

phone: string;

@Column({ length: 100, nullable: true })

email: string;

@Column({ type: 'text', nullable: true })

address: string;

@Column({ length: 5, nullable: true })

blood\_type: string;

@Column({ default: true })

is\_active: boolean;

@Column({ type: 'timestamp', default: () => 'now()' })

created\_at: Date;

@Column({ type: 'timestamp', default: () => 'now()' })

updated\_at: Date;

@OneToMany(() => MedicalRecord, (record) => record.patient)

medical\_records: MedicalRecord[];

}

````

````typescript

import { Entity, Column, PrimaryGeneratedColumn, ManyToOne, JoinColumn } from 'typeorm';

import { Patient } from './patient.entity';

@Entity({ name: 'medical\_records' })

export class MedicalRecord {

@PrimaryGeneratedColumn()

id: number;

@Column()

patient\_id: number;

@ManyToOne(() => Patient, (patient) => patient.medical\_records, { onDelete: 'CASCADE' })

@JoinColumn({ name: 'patient\_id' })

patient: Patient;

@Column({ type: 'timestamp' })

visit\_date: Date;

@Column({ type: 'text', nullable: true })

symptoms: string;

@Column({ type: 'text', nullable: true })

diagnosis: string;

@Column({ type: 'text', nullable: true })

treatment: string;

@Column({ length: 100, nullable: true })

doctor\_name: string;

@Column({ type: 'text', nullable: true })

note: string;

@Column({ type: 'timestamp', default: () => 'now()' })

created\_at: Date;

}

````

---

### 2. Service

````typescript

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { Patient } from '../../database/entities/patient.entity';

@Injectable()

export class PatientService {

constructor(

@InjectRepository(Patient)

private patientRepository: Repository<Patient>,

) {}

findAll() {

return this.patientRepository.find();

}

findOne(id: number) {

return this.patientRepository.findOne({ where: { id } });

}

create(data: Partial<Patient>) {

const patient = this.patientRepository.create(data);

return this.patientRepository.save(patient);

}

update(id: number, data: Partial<Patient>) {

return this.patientRepository.update(id, data);

}

remove(id: number) {

return this.patientRepository.delete(id);

}

}

````

````typescript

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { MedicalRecord } from '../../database/entities/medical-record.entity';

@Injectable()

export class MedicalRecordService {

constructor(

@InjectRepository(MedicalRecord)

private recordRepository: Repository<MedicalRecord>,

) {}

findAll() {

return this.recordRepository.find({ relations: ['patient'] });

}

findByPatient(patient\_id: number) {

return this.recordRepository.find({ where: { patient\_id }, relations: ['patient'] });

}

create(data: Partial<MedicalRecord>) {

const record = this.recordRepository.create(data);

return this.recordRepository.save(record);

}

update(id: number, data: Partial<MedicalRecord>) {

return this.recordRepository.update(id, data);

}

remove(id: number) {

return this.recordRepository.delete(id);

}

}

````

---

### 3. Controller

````typescript

import { Controller, Get, Post, Body, Param, Put, Delete } from '@nestjs/common';

import { PatientService } from './patient.service';

@Controller('patients')

export class PatientController {

constructor(private readonly patientService: PatientService) {}

@Get()

findAll() {

return this.patientService.findAll();

}

@Get(':id')

findOne(@Param('id') id: number) {

return this.patientService.findOne(id);

}

@Post()

create(@Body() data: any) {

return this.patientService.create(data);

}

@Put(':id')

update(@Param('id') id: number, @Body() data: any) {

return this.patientService.update(id, data);

}

@Delete(':id')

remove(@Param('id') id: number) {

return this.patientService.remove(id);

}

}

````

````typescript

import { Controller, Get, Post, Body, Param, Put, Delete } from '@nestjs/common';

import { MedicalRecordService } from './medical-record.service';

@Controller('medical-records')

export class MedicalRecordController {

constructor(private readonly recordService: MedicalRecordService) {}

@Get()

findAll() {

return this.recordService.findAll();

}

@Get('patient/:patient\_id')

findByPatient(@Param('patient\_id') patient\_id: number) {

return this.recordService.findByPatient(patient\_id);

}

@Post()

create(@Body() data: any) {

return this.recordService.create(data);

}

@Put(':id')

update(@Param('id') id: number, @Body() data: any) {

return this.recordService.update(id, data);

}

@Delete(':id')

remove(@Param('id') id: number) {

return this.recordService.remove(id);

}

}

````

---

### 4. Module

````typescript

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { Patient } from '../../database/entities/patient.entity';

import { PatientService } from './patient.service';

import { PatientController } from './patient.controller';

@Module({

imports: [TypeOrmModule.forFeature([Patient])],

providers: [PatientService],

controllers: [PatientController],

exports: [PatientService],

})

export class PatientModule {}

````

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

````typescript

.ts

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { MedicalRecord } from '../../database/entities/medical-record.entity';

import { MedicalRecordService } from './medical-record.service';

import { MedicalRecordController } from './medical-record.controller';

@Module({

imports: [TypeOrmModule.forFeature([MedicalRecord])],

providers: [MedicalRecordService],

controllers: [MedicalRecordController],

exports: [MedicalRecordService],

})

export class MedicalRecordModule {}

````

---

### 5. เพิ่มใน AppModule

````typescript

// ...existing code...

import { PatientModule } from './modules/patient/patient.module';

import { MedicalRecordModule } from './modules/medical-record/medical-record.module';

@Module({

imports: [

// ...existing code...

PatientModule,

MedicalRecordModule,

],

// ...existing code...

})

export class AppModule {}

````

---

\*\*สรุป path ที่ต้องสร้าง:\*\*

- `src/database/entities/patient.entity.ts`

- `src/database/entities/medical-record.entity.ts`

- `src/modules/patient/patient.service.ts`

- `src/modules/patient/patient.controller.ts`

- `src/modules/patient/patient.module.ts`

- `src/modules/medical-record/medical-record.service.ts`

- `src/modules/medical-record/medical-record.controller.ts`

- `src/modules/medical-record/medical-record.module.ts`

- เพิ่ม import ใน app.module.ts

ถ้าต้องการ scaffold ไฟล์จริง แจ้งได้เลยครับ!

Similar code found with 1 license type

pethzero: ทำ code ให้หน่อยพร้อมการใช้งานคู่กับ CSRF เปลี่ยนแปลงข้อมูล" (POST/PUT/PATCH/DELETE)

GitHub Copilot: ````typescript

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { Patient } from '../../database/entities/patient.entity';

import { PatientService } from './patient.service';

import { PatientController } from './patient.controller';

@Module({

imports: [TypeOrmModule.forFeature([Patient])],

providers: [PatientService],

controllers: [PatientController],

exports: [PatientService],

})

export class PatientModule {}

````

````typescript

\modules\patient\patient.service.ts

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { Patient } from '../../database/entities/patient.entity';

@Injectable()

export class PatientService {

constructor(

@InjectRepository(Patient)

private patientRepository: Repository<Patient>,

) {}

findAll() {

return this.patientRepository.find();

}

findOne(id: number) {

return this.patientRepository.findOne({ where: { id } });

}

create(data: Partial<Patient>) {

const patient = this.patientRepository.create(data);

return this.patientRepository.save(patient);

}

update(id: number, data: Partial<Patient>) {

return this.patientRepository.update(id, data);

}

remove(id: number) {

return this.patientRepository.delete(id);

}

}

````

````typescript

import { Controller, Get, Post, Body, Param, Put, Delete, Req, Res, HttpStatus, UseGuards } from '@nestjs/common';

import { PatientService } from './patient.service';

import { Request, Response } from 'express';

// สมมติว่าคุณมี AuthGuard และ CSRF middleware อยู่แล้ว

// ถ้าใช้ NestJS CSRF middleware ให้แน่ใจว่า path นี้ถูก apply CSRF

@Controller('patients')

export class PatientController {

constructor(private readonly patientService: PatientService) {}

@Get()

async findAll(@Res() res: Response) {

const data = await this.patientService.findAll();

return res.status(HttpStatus.OK).json(data);

}

@Get(':id')

async findOne(@Param('id') id: number, @Res() res: Response) {

const data = await this.patientService.findOne(Number(id));

if (!data) return res.status(HttpStatus.NOT\_FOUND).json({ message: 'Patient not found' });

return res.status(HttpStatus.OK).json(data);

}

@Post()

async create(@Body() data: any, @Req() req: Request, @Res() res: Response) {

// CSRF token จะถูกตรวจสอบโดย middleware ก่อนถึง controller

const created = await this.patientService.create(data);

return res.status(HttpStatus.CREATED).json(created);

}

@Put(':id')

async update(@Param('id') id: number, @Body() data: any, @Req() req: Request, @Res() res: Response) {

const result = await this.patientService.update(Number(id), data);

if (result.affected === 0) return res.status(HttpStatus.NOT\_FOUND).json({ message: 'Patient not found' });

return res.status(HttpStatus.OK).json({ message: 'Patient updated' });

}

@Delete(':id')

async remove(@Param('id') id: number, @Req() req: Request, @Res() res: Response) {

const result = await this.patientService.remove(Number(id));

if (result.affected === 0) return res.status(HttpStatus.NOT\_FOUND).json({ message: 'Patient not found' });

return res.status(HttpStatus.OK).json({ message: 'Patient deleted' });

}

}

````

\*\*หมายเหตุการใช้งาน CSRF:\*\*

- ให้แน่ใจว่า main.ts ของคุณ apply csurf middleware กับ path `/api/patients` (POST/PUT/DELETE)

- ทุก request ที่เปลี่ยนแปลงข้อมูล (POST/PUT/DELETE) ต้องแนบ CSRF token ใน header (`X-CSRF-Token`)

- ถ้าใช้ AuthGuard สามารถเพิ่ม `@UseGuards(AuthGuard())` ได้ตามต้องการ

\*\*ตัวอย่างการเรียกใช้งาน (curl):\*\*

```bash

curl -b cookie.txt -H "X-CSRF-Token: <token>" -H "Content-Type: application/json" -d '{"first\_name":"John",...}' http://localhost:3000/api/patients

```

Similar code found with 1 license type

pethzero: {

"message": "Cannot POST /api/patients",

"error": "Not Found",

"statusCode": 404

}

ทำไหม

GitHub Copilot: สาเหตุที่เจอ

```json

{

"message": "Cannot POST /api/patients",

"error": "Not Found",

"statusCode": 404

}

```

\*\*เพราะ:\*\*

1. \*\*Global Prefix:\*\* ใน `main.ts` คุณตั้ง `app.setGlobalPrefix('api')`

2. \*\*Path ใน Controller:\*\* `@Controller('patients')`

3. \*\*ดังนั้น path ที่ถูกต้องคือ:\*\* `/api/patients`

\*\*แต่ถ้ายังเจอ 404:\*\*

- ตรวจสอบว่าได้ import `PatientModule` ใน `AppModule` แล้วหรือยัง

- ตรวจสอบว่า server ได้ restart หลังเพิ่มไฟล์ใหม่หรือไม่

- ตรวจสอบว่า method ที่ใช้เป็น `POST` และ path `/api/patients` ตรงกับ controller

\*\*Checklist แก้ไข:\*\*

1. ใน `src/app.module.ts` ต้องมี

```typescript

import { PatientModule } from './modules/patient/patient.module';

// ...

@Module({

imports: [

// ...อื่นๆ

PatientModule,

],

})

export class AppModule {}

```

2. Restart server ทุกครั้งหลังเพิ่ม/แก้ไข module

3. เรียก POST ที่ path `/api/patients` (ไม่ใช่ `/patients` เฉยๆ)

\*\*ตัวอย่าง curl ที่ถูกต้อง:\*\*

```bash

curl -X POST http://localhost:3000/api/patients \

-H "Content-Type: application/json" \

-H "X-CSRF-Token: <token>" \

-b cookie.txt \

-d '{"patient\_code":"P001","first\_name":"John","last\_name":"Doe"}'

```

\*\*สรุป:\*\*

- ถ้า import module ถูกต้อง, path `/api/patients` จะใช้งานได้

- ถ้ายังไม่ได้ ให้ดู log ที่ terminal อาจมี error ตอน start app

- ถ้าแก้แล้วยังไม่ได้ แจ้ง error log เพิ่มเติมได้เลย

pethzero: [Nest] 12528 - 24/07/2568 00:27:38 ERROR [TypeOrmModule] Unable to connect to the database. Retrying (1)...

TypeORMError: Entity metadata for Patient#medical\_records was not found. Check if you specified a correct entity object and if it's connected in the connection options.

at <anonymous> (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\metadata-builder\src\metadata-builder\EntityMetadataBuilder.ts:1128:23)

at Array.forEach (<anonymous>)

at EntityMetadataBuilder.computeInverseProperties (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\metadata-builder\src\metadata-builder\EntityMetadataBuilder.ts:1118:34)

at <anonymous> (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\metadata-builder\src\metadata-builder\EntityMetadataBuilder.ts:160:18)

at Array.forEach (<anonymous>)

at EntityMetadataBuilder.build (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\metadata-builder\src\metadata-builder\EntityMetadataBuilder.ts:159:25)

at ConnectionMetadataBuilder.buildEntityMetadatas (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\connection\src\connection\ConnectionMetadataBuilder.ts:106:11)

at process.processTicksAndRejections (node:internal/process/task\_queues:105:5)

at async DataSource.buildMetadatas (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\data-source\src\data-source\DataSource.ts:733:13)

at async DataSource.initialize (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\data-source\src\data-source\DataSource.ts:264:13)

GitHub Copilot: ปัญหานี้เกิดจาก TypeORM \*\*หา entity `MedicalRecord` ไม่เจอ\*\*

เพราะใน entity `Patient` มี relation ไปที่ `MedicalRecord` (`medical\_records: MedicalRecord[]`),

แต่คุณยัง \*\*ไม่ได้ register `MedicalRecord` entity ใน TypeORM config\*\* (ใน `databaseConfig()` หรือใน `TypeOrmModule.forRoot`)

---

## วิธีแก้ไข

### 1. \*\*เพิ่ม MedicalRecord entity ใน TypeORM config\*\*

ให้ import และเพิ่ม entity นี้ใน database config ของคุณ เช่น

````typescript

import { Patient } from '../database/entities/patient.entity';

import { MedicalRecord } from '../database/entities/medical-record.entity';

// ...existing code...

export default () => ({

// ...existing code...

entities: [

Patient,

MedicalRecord,

// ...entities อื่นๆ...

],

// ...existing code...

});

````

หรือถ้าใช้ glob pattern:

```typescript

entities: [\_\_dirname + '/../\*\*/\*.entity.{js,ts}'],

```

ให้แน่ใจว่า path นี้ครอบคลุมทั้ง `patient.entity.ts` และ `medical-record.entity.ts`

---

### 2. \*\*Restart server\*\*

หลังจากแก้ไขแล้ว ให้ stop และ start server ใหม่

---

### 3. \*\*ตรวจสอบ MedicalRecord entity\*\*

ให้แน่ใจว่าไฟล์ `medical-record.entity.ts` มีการ export class `MedicalRecord` ถูกต้อง

และ import `Patient` จาก path ที่ถูกต้อง

---

\*\*สรุป:\*\*

- ต้อง register ทุก entity ที่มี relation กันใน TypeORM config

- ถ้าใช้ glob pattern ให้แน่ใจว่า path ครอบคลุมทุก entity

- Restart server หลังแก้ไข

ถ้ายัง error แนบไฟล์ `database.config.ts` หรือ config TypeORM มาให้ดูเพิ่มได้เลยครับ!

pethzero: เวลาระบบเกิด error ระบบ จะเก็บแบบไหนกัน ที่ใช้ interceptor บ้าง ใช้เนื่องโอกาสตอนไหน

GitHub Copilot: \*\*การเก็บ error ในระบบ NestJS\*\*

โดยทั่วไปจะใช้ \*\*Exception Filter\*\* สำหรับจัดการ error (เช่น log, แปลง response, ส่งแจ้งเตือน)

ส่วน \*\*Interceptor\*\* จะเหมาะกับการ log request/response, วัด performance, หรือดักจับข้อมูลก่อน/หลัง handler

---

### 1. \*\*Interceptor ใช้กับอะไร\*\*

- \*\*LoggingInterceptor\*\* (เช่นตัวอย่างของคุณ)

- ใช้ log ทุก request/response (เช่น method, path, response time)

- ใช้กับ useGlobalInterceptors หรือเฉพาะ controller/route ก็ได้

- \*\*ไม่เหมาะกับการเก็บ error โดยละเอียด\*\* (แต่สามารถ log error ได้ถ้าใช้ catchError ใน RxJS)

\*\*ตัวอย่าง LoggingInterceptor ที่ log error ด้วย:\*\*

````typescript

import { Injectable, NestInterceptor, ExecutionContext, CallHandler } from '@nestjs/common';

import { Observable, tap, catchError } from 'rxjs';

import { throwError } from 'rxjs';

@Injectable()

export class LoggingInterceptor implements NestInterceptor {

intercept(context: ExecutionContext, next: CallHandler): Observable<any> {

const req = context.switchToHttp().getRequest();

const method = req.method;

const url = req.url;

const now = Date.now();

return next.handle().pipe(

tap(() => console.log(`[${method}] ${url} - ${Date.now() - now}ms`)),

catchError((err) => {

console.error(`[${method}] ${url} - ERROR:`, err.message);

return throwError(() => err);

}),

);

}

}

````

---

### 2. \*\*Exception Filter ใช้กับอะไร\*\*

- ใช้จัดการ error ที่เกิดขึ้นในระบบ (เช่น log ลงไฟล์/DB, ส่ง response ที่ standardized)

- ใช้กับ useGlobalFilters หรือเฉพาะ controller/route ก็ได้

\*\*ตัวอย่าง Exception Filter:\*\*

```typescript

import { ExceptionFilter, Catch, ArgumentsHost, HttpException } from '@nestjs/common';

import { Response } from 'express';

@Catch(HttpException)

export class HttpExceptionFilter implements ExceptionFilter {

catch(exception: HttpException, host: ArgumentsHost) {

const ctx = host.switchToHttp();

const response = ctx.getResponse<Response>();

const status = exception.getStatus();

const message = exception.message;

// Log error ที่นี่ (เช่น log ลงไฟล์/DB)

console.error(`[ERROR] ${status}: ${message}`);

response.status(status).json({

statusCode: status,

message,

timestamp: new Date().toISOString(),

});

}

}

```

---

### 3. \*\*สรุปการเลือกใช้\*\*

- \*\*Interceptor:\*\*

- Log request/response, วัดเวลา, ดักจับข้อมูลก่อน/หลัง handler

- ใช้ log error ได้ แต่ไม่ละเอียดเท่า Exception Filter

- \*\*Exception Filter:\*\*

- ใช้จัดการ error โดยตรง (log, แปลง response, แจ้งเตือน)

- เหมาะกับการเก็บ error ที่เกิดขึ้นในระบบ

\*\*โอกาสใช้งาน:\*\*

- Interceptor: ใช้ log ทุก request/response, performance, หรือ log error แบบเบื้องต้น

- Exception Filter: ใช้ log error, ส่ง response error ที่ standardized, แจ้งเตือน admin

---

\*\*แนะนำ:\*\*

- ใช้ Exception Filter สำหรับ error logging/handling

- ใช้ Interceptor สำหรับ log request/response หรือวัด performance

Similar code found with 2 license types

pethzero: LoggingInterceptor HttpExceptionFilter ต้องไปใส้อะไรยังไงตอยไหยดี แบบไม่ใช้ post get เขียน code พลาดแต่ก็มี ดักerror กันระบบล่ม

GitHub Copilot: การใช้งาน \*\*LoggingInterceptor\*\* และ \*\*HttpExceptionFilter\*\* ใน NestJS

เพื่อให้ระบบ log ทุก request/response และดักจับ error ทุกกรณี (รวมถึง error จาก code ที่เขียนผิด ไม่ใช่แค่ POST/GET)

\*\*ควรใช้แบบ global\*\* ใน main.ts ดังนี้

---

### 1. สร้างไฟล์ Interceptor และ Filter (คุณมีอยู่แล้ว)

- logging.interceptor.ts

- http-exception.filter.ts

---

### 2. ใส่ใน main.ts (Global)

````typescript

import { NestFactory } from '@nestjs/core';

import { AppModule } from './app.module';

import { LoggingInterceptor } from './common/interceptors/logging.interceptor';

import { HttpExceptionFilter } from './common/filters/http-exception.filter';

// ...existing import...

async function bootstrap() {

const app = await NestFactory.create(AppModule);

// ...middleware, csurf, cors, etc...

// ใส่ Interceptor และ Exception Filter แบบ global

app.useGlobalInterceptors(new LoggingInterceptor());

app.useGlobalFilters(new HttpExceptionFilter());

// ...existing code...

await app.listen(process.env.PORT || 3000);

}

bootstrap();

````

---

### 3. ผลลัพธ์

- ทุก request/response จะถูก log (รวมถึง error)

- ทุก error (ทั้งที่ throw HttpException หรือ error จาก code พลาด เช่น throw new Error) จะถูกดักจับ ไม่ทำให้ระบบล่ม และ log ออก console

- Response จะ standardized ตามที่ filter กำหนด

---

\*\*สรุป:\*\*

- ใส่ `app.useGlobalInterceptors(new LoggingInterceptor())` และ `app.useGlobalFilters(new HttpExceptionFilter())` ใน main.ts

- ไม่ต้องไปใส่ที่ controller หรือ service ทีละตัว

- ระบบจะ log และกัน error ทุกกรณี (รวมถึง error จาก code ที่เขียนผิด)

---

\*\*เพิ่มเติม:\*\*

- ถ้าอยาก log ลงไฟล์/DB ให้แก้ใน filter/interceptor ได้เลย

- ถ้าอยาก log error ที่ไม่ใช่ HttpException (เช่น throw new Error) ให้ใช้ `@Catch()` ไม่ใส่ argument หรือใส่ `Error` ใน filter

ถ้าต้องการตัวอย่าง filter ที่ดักจับ error ทุกชนิด แจ้งได้เลย!

pethzero: cmd แสดง

[Nest] 17012 - 24/07/2568 00:34:19 ERROR [ExceptionsHandler] ForbiddenError: invalid csrf token

at csrf (E:\PROJECT\kornpassorn-care\backend\node\_modules\csurf\index.js:112:19)

at <anonymous> (E:\PROJECT\kornpassorn-care\backend\src\main.ts:32:5)

at Layer.handleRequest (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\lib\layer.js:152:17)

at trimPrefix (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:342:13)

at E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:297:9

at processParams (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:582:12)

at next (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:291:5)

at E:\PROJECT\kornpassorn-care\backend\node\_modules\body-parser\lib\read.js:132:5

at AsyncResource.runInAsyncScope (node:async\_hooks:214:14)

at invokeCallback (E:\PROJECT\kornpassorn-care\backend\node\_modules\raw-body\index.js:238:16) {

code: 'EBADCSRFTOKEN'

}

[POST] /api/patients - ERROR: null value in column "patient\_code" of relation "patients" violates not-null constraint

[Nest] 17012 - 24/07/2568 00:34:31 ERROR [ExceptionsHandler] QueryFailedError: null value in column "patient\_code" of relation "patients" violates not-null constraint

at PostgresQueryRunner.query (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\driver\src\driver\postgres\PostgresQueryRunner.ts:325:19)

at process.processTicksAndRejections (node:internal/process/task\_queues:105:5)

at async InsertQueryBuilder.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\query-builder\src\query-builder\InsertQueryBuilder.ts:164:33)

at async SubjectExecutor.executeInsertOperations (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:435:42)

at async SubjectExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:137:9)

at async EntityPersistExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\EntityPersistExecutor.ts:182:21)

at async PatientController.create (E:\PROJECT\kornpassorn-care\backend\src\modules\patient\patient.controller.ts:28:21) {

query: 'INSERT INTO "patients"("patient\_code", "first\_name", "last\_name", "gender", "date\_of\_birth", "phone", "email", "address", "blood\_type", "is\_active", "created\_at", "updated\_at") VALUES (DEFAULT, $1, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT) RETURNING "id", "is\_active", "created\_at", "updated\_at"',

parameters: [

'John'

],

driverError: error: null value in column "patient\_code" of relation "patients" violates not-null constraint

at E:\PROJECT\kornpassorn-care\backend\node\_modules\pg\lib\client.js:545:17

at process.processTicksAndRejections (node:internal/process/task\_queues:105:5)

at async PostgresQueryRunner.query (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\driver\src\driver\postgres\PostgresQueryRunner.ts:254:25)

at async InsertQueryBuilder.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\query-builder\src\query-builder\InsertQueryBuilder.ts:164:33)

at async SubjectExecutor.executeInsertOperations (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:435:42)

at async SubjectExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:137:9)

at async EntityPersistExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\EntityPersistExecutor.ts:182:21)

at async PatientController.create (E:\PROJECT\kornpassorn-care\backend\src\modules\patient\patient.controller.ts:28:21) {

สร้างกัน ได้ไหม

GitHub Copilot: จาก error ที่คุณเจอ มี 2 ส่วนหลัก:

---

### 1. \*\*CSRF: ForbiddenError: invalid csrf token\*\*

\*\*สาเหตุ:\*\*

- ไม่ได้แนบ CSRF token ที่ถูกต้องใน header (`X-CSRF-Token`) หรือ cookie

- หรือ session/cookie ที่ใช้ขอ CSRF token กับที่ใช้ POST ไม่ตรงกัน

\*\*วิธีแก้:\*\*

1. \*\*GET\*\* `/api/auth/csrf-token` เพื่อขอ token (และ cookie)

2. \*\*POST\*\* `/api/patients`

- แนบ cookie ที่ได้จากข้อ 1

- แนบ header `X-CSRF-Token: <token ที่ได้>`

\*\*ตัวอย่าง curl:\*\*

```bash

curl -c cookie.txt http://localhost:3000/api/auth/csrf-token

curl -b cookie.txt -H "X-CSRF-Token: <token>" -H "Content-Type: application/json" \

-d '{"patient\_code":"P001","first\_name":"John","last\_name":"Doe"}' \

http://localhost:3000/api/patients

```

---

### 2. \*\*QueryFailedError: null value in column "patient\_code"\*\*

\*\*สาเหตุ:\*\*

- ข้อมูลที่ส่งไปไม่ครบ โดยเฉพาะ `patient\_code` ซึ่งเป็น NOT NULL

- ตัวอย่าง log:

```

parameters: [ 'John' ]

```

แปลว่า body ที่ส่งไปมีแค่ `first\_name` เท่านั้น

\*\*วิธีแก้:\*\*

- ต้องส่งข้อมูลครบอย่างน้อย 3 field: `patient\_code`, `first\_name`, `last\_name`

- ตัวอย่าง body ที่ถูกต้อง:

```json

{

"patient\_code": "P001",

"first\_name": "John",

"last\_name": "Doe"

}

```

---

### 3. \*\*ตัวอย่างการสร้างคนไข้ (POST /api/patients)\*\*

\*\*curl ตัวอย่าง:\*\*

```bash

curl -b cookie.txt -H "X-CSRF-Token: <token>" -H "Content-Type: application/json" \

-d '{"patient\_code":"P001","first\_name":"John","last\_name":"Doe"}' \

http://localhost:3000/api/patients

```

---

\*\*สรุป:\*\*

- แนบ CSRF token และ cookie ให้ถูกต้อง

- ส่งข้อมูลครบทุก field ที่จำเป็น (`patient\_code`, `first\_name`, `last\_name`)

- ถ้ายัง error แนบ request body ที่ใช้จริงมาให้ดูเพิ่มได้เลย

pethzero: [Nest] 17012 - 24/07/2568 00:34:19 ERROR [ExceptionsHandler] ForbiddenError: invalid csrf token

at csrf (E:\PROJECT\kornpassorn-care\backend\node\_modules\csurf\index.js:112:19)

at <anonymous> (E:\PROJECT\kornpassorn-care\backend\src\main.ts:32:5)

at Layer.handleRequest (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\lib\layer.js:152:17)

at trimPrefix (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:342:13)

at E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:297:9

at processParams (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:582:12)

at next (E:\PROJECT\kornpassorn-care\backend\node\_modules\router\index.js:291:5)

at E:\PROJECT\kornpassorn-care\backend\node\_modules\body-parser\lib\read.js:132:5

at AsyncResource.runInAsyncScope (node:async\_hooks:214:14)

at invokeCallback (E:\PROJECT\kornpassorn-care\backend\node\_modules\raw-body\index.js:238:16) {

code: 'EBADCSRFTOKEN'

}

[POST] /api/patients - ERROR: null value in column "patient\_code" of relation "patients" violates not-null constraint

[Nest] 17012 - 24/07/2568 00:34:31 ERROR [ExceptionsHandler] QueryFailedError: null value in column "patient\_code" of relation "patients" violates not-null constraint

at PostgresQueryRunner.query (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\driver\src\driver\postgres\PostgresQueryRunner.ts:325:19)

at process.processTicksAndRejections (node:internal/process/task\_queues:105:5)

at async InsertQueryBuilder.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\query-builder\src\query-builder\InsertQueryBuilder.ts:164:33)

at async SubjectExecutor.executeInsertOperations (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:435:42)

at async SubjectExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:137:9)

at async EntityPersistExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\EntityPersistExecutor.ts:182:21)

at async PatientController.create (E:\PROJECT\kornpassorn-care\backend\src\modules\patient\patient.controller.ts:28:21) {

query: 'INSERT INTO "patients"("patient\_code", "first\_name", "last\_name", "gender", "date\_of\_birth", "phone", "email", "address", "blood\_type", "is\_active", "created\_at", "updated\_at") VALUES (DEFAULT, $1, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT, DEFAULT) RETURNING "id", "is\_active", "created\_at", "updated\_at"',

parameters: [

'John'

],

driverError: error: null value in column "patient\_code" of relation "patients" violates not-null constraint

at E:\PROJECT\kornpassorn-care\backend\node\_modules\pg\lib\client.js:545:17

at process.processTicksAndRejections (node:internal/process/task\_queues:105:5)

at async PostgresQueryRunner.query (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\driver\src\driver\postgres\PostgresQueryRunner.ts:254:25)

at async InsertQueryBuilder.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\query-builder\src\query-builder\InsertQueryBuilder.ts:164:33)

at async SubjectExecutor.executeInsertOperations (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:435:42)

at async SubjectExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\SubjectExecutor.ts:137:9)

at async EntityPersistExecutor.execute (E:\PROJECT\kornpassorn-care\backend\node\_modules\typeorm\persistence\src\persistence\EntityPersistExecutor.ts:182:21)

at async PatientController.create (E:\PROJECT\kornpassorn-care\backend\src\modules\patient\patient.controller.ts:28:21)

ไม่อยากให้แสดง แบบ at E:\PROJECT\kornpassorn-care\backend\node\_modules\pg\lib\client.js:545:17

เอาแค่ log error

GitHub Copilot: คุณสามารถปรับ Exception Filter ของคุณให้ \*\*log เฉพาะ error message\*\* (ไม่แสดง stack trace)

และส่ง response ที่ปลอดภัย/สั้นให้ client ได้แบบนี้

````typescript

import { ExceptionFilter, Catch, ArgumentsHost, HttpException } from '@nestjs/common';

import { Response } from 'express';

@Catch()

export class HttpExceptionFilter implements ExceptionFilter {

catch(exception: any, host: ArgumentsHost) {

const ctx = host.switchToHttp();

const response = ctx.getResponse<Response>();

const status = exception instanceof HttpException ? exception.getStatus() : 500;

const message = exception.message || 'Internal server error';

// Log เฉพาะ error message (ไม่แสดง stack trace)

console.error(`[ERROR] ${status}: ${message}`);

response.status(status).json({

statusCode: status,

message,

timestamp: new Date().toISOString(),

});

}

}

````

\*\*วิธีใช้:\*\*

- ใส่ใน `main.ts` ด้วย `app.useGlobalFilters(new HttpExceptionFilter());`

\*\*ผลลัพธ์:\*\*

- Console log จะเห็นแค่ `[ERROR] 500: <error message>`

- Client จะได้ response ที่ standardized และไม่เห็น stack trace

\*\*หมายเหตุ:\*\*

- ถ้าอยาก log ลงไฟล์/DB ก็เพิ่มใน filter นี้ได้เลย

Similar code found with 1 license type

pethzero: ควรอัพเกรดอะไรเพิ่ม

GitHub Copilot: ### ข้อเสนอแนะการอัปเกรด/ปรับปรุงโปรเจกต์ของคุณ

#### 1. \*\*Entity Relation\*\*

- เปิดใช้ความสัมพันธ์ระหว่าง `Patient` กับ `MedicalRecord` (uncomment ใน entity ทั้งสองไฟล์)

- เพิ่ม `MedicalRecord` entity ใน TypeORM config (ถ้าใช้ autoLoadEntities: true แล้ว ครอบคลุมอยู่แล้ว)

#### 2. \*\*Validation & DTO\*\*

- ใช้ DTO + class-validator ในทุก controller (เช่น POST/PUT ของ patient) เพื่อป้องกันข้อมูลผิดรูปแบบ

- ตัวอย่าง:

```typescript

import { IsString, IsNotEmpty } from 'class-validator';

export class CreatePatientDto {

@IsString()

@IsNotEmpty()

patient\_code: string;

// ...field อื่นๆ

}

```

#### 3. \*\*Security\*\*

- ตั้งค่า CSRF cookie เป็น `httpOnly: true` ใน production

- ตั้งค่า CORS ให้ปลอดภัย (origin เฉพาะ production)

- ใช้ HTTPS ใน production

#### 4. \*\*Error Handling\*\*

- ใช้ Exception Filter (คุณมีแล้ว) และสามารถ log ลงไฟล์หรือส่งแจ้งเตือน (เช่น Sentry) ได้

#### 5. \*\*Authentication/Authorization\*\*

- ใช้ `@UseGuards(JwtAuthGuard)` กับทุก endpoint ที่ต้องการ auth

- เพิ่ม Role Guard ถ้าต้องการแยกสิทธิ์

#### 6. \*\*Logging\*\*

- พิจารณาใช้ logger ที่ advance เช่น [winston](https://github.com/winstonjs/winston) หรือ [pino](https://github.com/pinojs/pino) แทน console.log

#### 7. \*\*Testing\*\*

- เพิ่ม unit test และ e2e test ให้ครอบคลุมมากขึ้น

#### 8. \*\*Database Migration\*\*

- ใช้ TypeORM migration แทน synchronize: true ใน production

#### 9. \*\*API Documentation\*\*

- ติดตั้งและตั้งค่า [Swagger](https://docs.nestjs.com/openapi/introduction) (`@nestjs/swagger`) เพื่อให้มี API docs อัตโนมัติ

#### 10. \*\*Performance\*\*

- ใช้ cache (เช่น Redis) ถ้าข้อมูลอ่านบ่อย

- ใช้ pagination กับ endpoint ที่ return ข้อมูลจำนวนมาก

---

\*\*ตัวอย่างการเพิ่ม Swagger:\*\*

````typescript

// main.ts

import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';

// ...existing code...

const config = new DocumentBuilder()

.setTitle('Kornpassorn Care API')

.setDescription('API documentation')

.setVersion('1.0')

.addBearerAuth()

.build();

const document = SwaggerModule.createDocument(app, config);

SwaggerModule.setup('api/docs', app, document);

````

---

\*\*สรุป:\*\*

- เปิดใช้ relation entity, ใช้ DTO+validation, เพิ่ม security, ใช้ migration, เพิ่ม Swagger, และ logging ที่ advance

- ถ้าต้องการตัวอย่างโค้ดแต่ละข้อ แจ้งได้เลย!

pethzero: เราเห็นมี guard มันจะต้องเพิ่มยังไง

GitHub Copilot: ถ้าคุณต้องการ \*\*เพิ่ม Guard\*\* (เช่น JWT Auth Guard) เพื่อป้องกัน endpoint ให้เฉพาะผู้ที่ login แล้วเท่านั้นที่เข้าถึงได้

คุณสามารถเพิ่มได้ง่าย ๆ ดังนี้

---

### 1. \*\*สร้าง/ใช้ Guard ที่มีอยู่แล้ว\*\*

คุณมี `JwtAuthGuard` อยู่แล้ว

```typescript

import { Injectable } from '@nestjs/common';

import { AuthGuard } from '@nestjs/passport';

@Injectable()

export class JwtAuthGuard extends AuthGuard('jwt') {}

```

---

### 2. \*\*ใช้งาน Guard ใน Controller\*\*

#### \*\*แบบทั้ง Controller\*\*

```typescript

import { UseGuards } from '@nestjs/common';

import { JwtAuthGuard } from '../auth/jwt-auth.guard';

@UseGuards(JwtAuthGuard)

@Controller('patients')

export class PatientController {

// ...methods...

}

```

#### \*\*หรือเฉพาะบางเมธอด\*\*

```typescript

import { UseGuards } from '@nestjs/common';

import { JwtAuthGuard } from '../auth/jwt-auth.guard';

@Controller('patients')

export class PatientController {

constructor(private readonly patientService: PatientService) {}

@UseGuards(JwtAuthGuard)

@Get()

async findAll(@Res() res: Response) {

// ...

}

// ...method อื่นๆ...

}

```

---

### 3. \*\*ผลลัพธ์\*\*

- ทุก request ที่เข้ามา endpoint ที่มี Guard จะต้องแนบ JWT (เช่น Bearer token หรือ cookie)

- ถ้าไม่มีหรือไม่ถูกต้อง จะได้ 401 Unauthorized

---

### 4. \*\*แนะนำ\*\*

- ใช้กับทุก endpoint ที่ต้องการให้เฉพาะ user ที่ login แล้วเข้าถึงได้ (เช่น `/patients`, `/medical-records`)

- ถ้าต้องการแยกสิทธิ์ (role) เพิ่มเติม สามารถสร้าง Custom Guard หรือใช้ Decorator เพิ่มได้

---

\*\*สรุป:\*\*

- import และใช้ `@UseGuards(JwtAuthGuard)` ที่ controller หรือ method ที่ต้องการ

- ระบบจะบังคับตรวจสอบ JWT อัตโนมัติ

ถ้าต้องการตัวอย่างแบบ role guard หรือ custom guard เพิ่มเติม แจ้งได้เลย!

pethzero: E:\PROJECT\kornpassorn-care\backend\src\modules\patient เรายุบแบบว่า medicel ได้ไหม

medicel => pateinent , appointment

database => entities รวม เกี่ยวกับ ไฟล์ medeical รวมๆกัน

GitHub Copilot: ได้ครับ! คุณสามารถ \*\*ยุบโฟลเดอร์\*\* หรือ \*\*จัดกลุ่ม module\*\* ให้เหมาะกับ business domain ได้ เช่น

---

## ตัวอย่างโครงสร้างแบบ Domain-centric

```

src/

modules/

medical/

patient/

patient.controller.ts

patient.service.ts

patient.module.ts

appointment/

appointment.controller.ts

appointment.service.ts

appointment.module.ts

medical.module.ts

database/

entities/

patient.entity.ts

appointment.entity.ts

medical-record.entity.ts

...

```

- \*\*modules/medical/\*\* รวมทุกอย่างที่เกี่ยวกับ medical domain (เช่น patient, appointment, medical-record)

- \*\*database/entities/\*\* รวม entity ทุกตัวที่เกี่ยวข้อง

---

## วิธีจัดการ

1. \*\*สร้างโฟลเดอร์ใหม่\*\*

- `src/modules/medical/patient/`

- `src/modules/medical/appointment/`

- (หรือ `medical-record/` ถ้าต้องการแยก)

2. \*\*ย้ายไฟล์\*\*

- ย้าย controller/service/module ของ patient, appointment ไปไว้ในแต่ละโฟลเดอร์ย่อย

- ปรับ import path ให้ถูกต้อง

3. \*\*สร้าง medical.module.ts\*\*

- รวม patient.module, appointment.module, medical-record.module (ถ้ามี) ใน medical.module.ts

```typescript

// filepath: src/modules/medical/medical.module.ts

import { Module } from '@nestjs/common';

import { PatientModule } from './patient/patient.module';

import { AppointmentModule } from './appointment/appointment.module';

// import { MedicalRecordModule } from './medical-record/medical-record.module';

@Module({

imports: [PatientModule, AppointmentModule /\*, MedicalRecordModule\*/],

})

export class MedicalModule {}

```

4. \*\*เพิ่ม MedicalModule ใน AppModule\*\*

```typescript

import { MedicalModule } from './modules/medical/medical.module';

// ...

@Module({

imports: [

// ...other modules

MedicalModule,

],

})

export class AppModule {}

```

---

## สรุป

- \*\*ยุบรวมได้\*\* และควรทำถ้าอยากให้โครงสร้างชัดเจนตาม business domain

- \*\*entities\*\* ควรอยู่รวมกันใน `database/entities/`

- \*\*แต่ละ feature (patient, appointment, ...)\*\* แยกเป็น submodule ใน `modules/medical/`

---

\*\*ข้อดี:\*\*

- ดูแลรักษาง่าย

- ขยายระบบในอนาคตสะดวก

- เหมาะกับทีมงานหลายคน

ถ้าต้องการตัวอย่างไฟล์หรือ scaffold โครงสร้างใหม่ แจ้งได้เลย!

pethzero: medical.entity.ts = > patient.entity.ts

appointment.entity.ts

medical-record.entity.ts รวม ใน ไฟล์

GitHub Copilot: ถ้าคุณต้องการ \*\*รวม export entity หลายตัวไว้ในไฟล์เดียว\*\* (เช่น `medical.entity.ts`)

เพื่อให้ import จากที่เดียวใน module/service อื่นๆ ได้ง่าย

ตัวอย่างโค้ด:

````typescript

export \* from './patient.entity';

export \* from './appointment.entity';

export \* from './medical-record.entity';

````

\*\*วิธีใช้:\*\*

เวลาต้องการ import entity ใน module/service อื่น

สามารถ import จาก `medical.entity.ts` ได้เลย เช่น

```typescript

import { Patient, Appointment, MedicalRecord } from '../../database/entities/medical.entity';

```

\*\*ข้อดี:\*\*

- Import entity หลายตัวจากไฟล์เดียว

- จัดการ entity ที่เกี่ยวข้องกับ medical domain ได้ง่าย

\*\*หมายเหตุ:\*\*

- ไฟล์ `patient.entity.ts`, `appointment.entity.ts`, `medical-record.entity.ts` ยังต้องแยกกันอยู่ (แต่ export รวมในไฟล์เดียวได้)

- ไม่ควรเขียน entity ทั้งหมดในไฟล์เดียว (ควรแยกแต่ละ class แล้ว export รวม)

ถ้าต้องการตัวอย่างไฟล์ entity แต่ละตัว แจ้งได้เลย!

pethzero: แบบ เอา

CREATE TABLE public.patients (

id SERIAL PRIMARY KEY,

patient\_code VARCHAR(20) UNIQUE NOT NULL,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

gender VARCHAR(10), -- male, female, other

date\_of\_birth DATE,

phone VARCHAR(20),

email VARCHAR(100),

address TEXT,

blood\_type VARCHAR(5), -- A, B, AB, O

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT now(),

updated\_at TIMESTAMP DEFAULT now()

);

CREATE TABLE public.medical\_records (

id SERIAL PRIMARY KEY,

patient\_id INT REFERENCES public.patients(id) ON DELETE CASCADE,

visit\_date TIMESTAMP NOT NULL,

symptoms TEXT,

diagnosis TEXT,

treatment TEXT,

doctor\_name VARCHAR(100),

note TEXT,

created\_at TIMESTAMP DEFAULT now()

);

รวมด้วยกันเลย

GitHub Copilot: ````typescript

import { Entity, Column, PrimaryGeneratedColumn, OneToMany, ManyToOne, JoinColumn } from 'typeorm';

// Patient Entity

@Entity({ name: 'patients' })

export class Patient {

@PrimaryGeneratedColumn()

id: number;

@Column({ unique: true, length: 20 })

patient\_code: string;

@Column({ length: 100 })

first\_name: string;

@Column({ length: 100 })

last\_name: string;

@Column({ length: 10, nullable: true })

gender: string;

@Column({ type: 'date', nullable: true })

date\_of\_birth: Date;

@Column({ length: 20, nullable: true })

phone: string;

@Column({ length: 100, nullable: true })

email: string;

@Column({ type: 'text', nullable: true })

address: string;

@Column({ length: 5, nullable: true })

blood\_type: string;

@Column({ default: true })

is\_active: boolean;

@Column({ type: 'timestamp', default: () => 'now()' })

created\_at: Date;

@Column({ type: 'timestamp', default: () => 'now()' })

updated\_at: Date;

@OneToMany(() => MedicalRecord, (record) => record.patient)

medical\_records: MedicalRecord[];

}

// MedicalRecord Entity

@Entity({ name: 'medical\_records' })

export class MedicalRecord {

@PrimaryGeneratedColumn()

id: number;

@Column()

patient\_id: number;

@ManyToOne(() => Patient, (patient) => patient.medical\_records, { onDelete: 'CASCADE' })

@JoinColumn({ name: 'patient\_id' })

patient: Patient;

@Column({ type: 'timestamp' })

visit\_date: Date;

@Column({ type: 'text', nullable: true })

symptoms: string;

@Column({ type: 'text', nullable: true })

diagnosis: string;

@Column({ type: 'text', nullable: true })

treatment: string;

@Column({ length: 100, nullable: true })

doctor\_name: string;

@Column({ type: 'text', nullable: true })

note: string;

@Column({ type: 'timestamp', default: () => 'now()' })

created\_at: Date;

}

````

pethzero: Nest] 16060 - 24/07/2568 22:35:40 ERROR [TypeOrmModule] Unable to connect to the database. Retrying (1)...

TypeORMError: Entity metadata for Patient#medical\_records was not found. Check if you specified a correct entity object and if it's connected in the connection options.

GitHub Copilot: ปัญหานี้เกิดจาก TypeORM \*\*หา entity `MedicalRecord` ไม่เจอ\*\*

เพราะใน entity `Patient` (ใน `medical.entity.ts`) มี relation ไปที่ `MedicalRecord`

แต่ TypeORM ยังไม่ได้รู้จัก `MedicalRecord` entity

### วิธีแก้ไข

#### 1. ตรวจสอบการตั้งค่า TypeORM

คุณใช้ `autoLoadEntities: true` ใน `database.config.ts` แล้ว

แต่ \*\*ต้องแน่ใจว่าได้ import/export ทั้ง `Patient` และ `MedicalRecord` จากไฟล์เดียวกัน\*\*

และ module ที่ใช้ TypeOrmModule.forFeature ต้อง import ทั้งสอง entity

#### 2. ปรับ PatientModule ให้รู้จักทั้งสอง entity

```typescript

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { Patient, MedicalRecord } from '../../../database/entities/medical.entity';

import { PatientService } from './patient.service';

import { PatientController } from './patient.controller';

@Module({

imports: [TypeOrmModule.forFeature([Patient, MedicalRecord])],

providers: [PatientService],

controllers: [PatientController],

exports: [PatientService],

})

export class PatientModule {}

```

#### 3. ถ้าใช้ MedicalRecord ใน module อื่น ให้ import ด้วย

ถ้ามี module อื่นที่ใช้ `MedicalRecord` (เช่น MedicalRecordModule)

ให้ใช้ `TypeOrmModule.forFeature([MedicalRecord])` ด้วย

#### 4. Restart server

หลังแก้ไขแล้ว ให้ stop และ start server ใหม่

---

\*\*สรุป:\*\*

- ต้อง import ทั้ง `Patient` และ `MedicalRecord` ใน `TypeOrmModule.forFeature` ของทุก module ที่มี relation กัน

- ถ้าใช้ autoLoadEntities: true แล้ว แต่ยัง error ให้เช็ค import/export และ path ให้ถูกต้อง

- Restart server หลังแก้ไข

ถ้ายัง error แนบโค้ด module ที่เกี่ยวข้องมาให้ดูเพิ่มได้เลย!

pethzero: เราสามารถเขียน แบบ raw script หรือ มี insert แบบ django ไหมที่มี update\_or\_create

GitHub Copilot: NestJS/TypeORM ไม่มีฟังก์ชัน `update\_or\_create` แบบ Django โดยตรง

แต่คุณสามารถเขียน logic แบบ raw หรือ custom method ให้ได้ผลลัพธ์เดียวกัน เช่น

---

### 1. \*\*แบบ TypeORM (Service method)\*\*

````typescript

async updateOrCreate(criteria: Partial<Patient>, data: Partial<Patient>) {

let patient = await this.patientRepository.findOne({ where: criteria });

if (patient) {

await this.patientRepository.update(patient.id, data);

return this.patientRepository.findOne({ where: { id: patient.id } });

} else {

patient = this.patientRepository.create({ ...criteria, ...data });

return this.patientRepository.save(patient);

}

}

````

\*\*วิธีใช้ใน controller:\*\*

```typescript

@Post('upsert')

async upsert(@Body() body: any, @Res() res: Response) {

// สมมติ criteria คือ patient\_code

const result = await this.patientService.updateOrCreate(

{ patient\_code: body.patient\_code },

body

);

return res.status(HttpStatus.OK).json(result);

}

```

---

### 2. \*\*แบบ Raw SQL (QueryBuilder)\*\*

````typescript

import { DataSource } from 'typeorm';

// ...ใน constructor inject DataSource...

async upsertRaw(data: Partial<Patient>) {

// ใช้ ON CONFLICT (Postgres)

return this.patientRepository.query(

`

INSERT INTO patients (patient\_code, first\_name, last\_name)

VALUES ($1, $2, $3)

ON CONFLICT (patient\_code)

DO UPDATE SET first\_name = EXCLUDED.first\_name, last\_name = EXCLUDED.last\_name

RETURNING \*

`,

[data.patient\_code, data.first\_name, data.last\_name]

);

}

````

---

### \*\*สรุป\*\*

- คุณสามารถสร้างฟังก์ชัน `updateOrCreate` เองใน service ได้

- หรือใช้ raw SQL (Postgres: `ON CONFLICT ... DO UPDATE`)

- ไม่มี built-in แบบ Django แต่ logic ทำได้เหมือนกัน

ถ้าต้องการตัวอย่างแบบอื่น (bulk, transaction ฯลฯ) แจ้งได้เลย!

pethzero:

ขอตัวอย่าง postman แต่ละตัวหน่อย ในการทำ patients

GitHub Copilot: ตัวอย่าง \*\*Postman\*\* สำหรับการใช้งาน API `/api/patients` (CRUD + upsert)

\*\*ทุก POST/PUT/DELETE ต้องแนบ CSRF token และ cookie\*\*

---

### 1. \*\*ขอ CSRF Token (GET)\*\*

- \*\*Method:\*\* GET

- \*\*URL:\*\* `http://localhost:3000/api/auth/csrf-token`

- \*\*Response:\*\*

```json

{ "csrfToken": "xxxxxx" }

```

- \*\*หมายเหตุ:\*\*

- เก็บ cookie ที่ได้ (Postman จะเก็บให้อัตโนมัติถ้าอยู่ใน session เดียวกัน)

---

### 2. \*\*สร้างคนไข้ (POST /api/patients)\*\*

- \*\*Method:\*\* POST

- \*\*URL:\*\* `http://localhost:3000/api/patients`

- \*\*Headers:\*\*

- `Content-Type: application/json`

- `X-CSRF-Token: <csrfToken ที่ได้จากข้อ 1>`

- \*\*Body (raw JSON):\*\*

```json

{

"patient\_code": "P001",

"first\_name": "John",

"last\_name": "Doe",

"gender": "male",

"date\_of\_birth": "1990-01-01",

"phone": "0812345678",

"email": "john@example.com",

"address": "123 Main St",

"blood\_type": "A"

}

```

---

### 3. \*\*ดูข้อมูลคนไข้ทั้งหมด (GET /api/patients)\*\*

- \*\*Method:\*\* GET

- \*\*URL:\*\* `http://localhost:3000/api/patients`

- \*\*Headers:\*\* (ไม่ต้องใช้ CSRF)

- \*\*Response:\*\*

```json

[

{

"id": 1,

"patient\_code": "P001",

"first\_name": "John",

...

}

]

```

---

### 4. \*\*ดูข้อมูลคนไข้รายเดียว (GET /api/patients/:id)\*\*

- \*\*Method:\*\* GET

- \*\*URL:\*\* `http://localhost:3000/api/patients/1`

- \*\*Headers:\*\* (ไม่ต้องใช้ CSRF)

---

### 5. \*\*แก้ไขข้อมูลคนไข้ (PUT /api/patients/:id)\*\*

- \*\*Method:\*\* PUT

- \*\*URL:\*\* `http://localhost:3000/api/patients/1`

- \*\*Headers:\*\*

- `Content-Type: application/json`

- `X-CSRF-Token: <csrfToken>`

- \*\*Body (raw JSON):\*\*

```json

{

"first\_name": "Jane",

"last\_name": "Smith"

}

```

---

### 6. \*\*ลบคนไข้ (DELETE /api/patients/:id)\*\*

- \*\*Method:\*\* DELETE

- \*\*URL:\*\* `http://localhost:3000/api/patients/1`

- \*\*Headers:\*\*

- `X-CSRF-Token: <csrfToken>`

---

### 7. \*\*Upsert (POST /api/patients/upsert)\*\*

- \*\*Method:\*\* POST

- \*\*URL:\*\* `http://localhost:3000/api/patients/upsert`

- \*\*Headers:\*\*

- `Content-Type: application/json`

- `X-CSRF-Token: <csrfToken>`

- \*\*Body (raw JSON):\*\*

```json

{

"patient\_code": "P002",

"first\_name": "Alice",

"last\_name": "Wonder"

}

```

---

\*\*หมายเหตุ:\*\*

- ทุก request ที่เปลี่ยนแปลงข้อมูล (POST/PUT/DELETE/upsert) ต้องแนบ CSRF token และ cookie ที่ได้จากการขอ CSRF token

- ถ้าใช้ Postman ให้เรียก GET `/api/auth/csrf-token` ก่อน แล้วใช้ cookie + token เดิมในทุก request

ถ้าต้องการตัวอย่าง Postman collection (JSON) แจ้งได้เลย!

pethzero:

import { Injectable } from '@angular/core';

import { BehaviorSubject, Observable, of, throwError } from 'rxjs';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { tap, catchError, map } from 'rxjs/operators';

import { environment } from '../../environments/environment';

export interface User {

userId: number;

username: string;

role: string;

}

@Injectable({ providedIn: 'root' })

export class AuthService {

private currentUserSubject = new BehaviorSubject<User | null>(null);

currentUser$: Observable<User | null> = this.currentUserSubject.asObservable();

private csrfToken: string = '';

constructor(private http: HttpClient) {

const token = localStorage.getItem('token');

if (token) {

try {

const user = this.decodeToken(token);

this.currentUserSubject.next(user);

} catch (error) {

console.error('Invalid token in localStorage, clearing it.', error);

this.logout(); // ล้าง token ที่ไม่ถูกต้องออก

}

}

}

fetchCsrfToken() {

return this.http.get<{ csrfToken: string }>(

`${environment.apiUrl}/auth/csrf-token`,

{ withCredentials: true }

).pipe(

tap(res => this.csrfToken = res.csrfToken)

);

}

getCsrfToken(): string {

return this.csrfToken;

}

private decodeToken(token: string): User {

const payload = JSON.parse(atob(token.split('.')[1]));

return {

userId: payload.sub,

username: payload.username,

role: payload.role

};

}

loginWithCredentials(username: string, password: string): Observable<boolean> {

return this.http.post<{ csrfToken: string }>(

`${environment.apiUrl}/auth/login`,

{ username, password },

{ withCredentials: true } // ✅ สำคัญมาก ต้องเปิดเพื่อรับ cookie

).pipe(

tap(response => {

this.csrfToken = response.csrfToken;

}),

map(() => true),

catchError(() => of(false))

);

}

loginAsGuest(): Observable<boolean> {

return this.http.post<{ access\_token: string }>(

`${environment.apiUrl}/auth/guest`,

{}

).pipe(

tap(response => {

const token = response.access\_token;

localStorage.setItem('token', token);

const user = this.decodeToken(token);

this.currentUserSubject.next(user);

}),

map(() => true),

catchError(err => {

console.error('Guest login failed', err);

this.logout();

return of(false);

})

);

}

logout(): void {

localStorage.removeItem('token');

this.currentUserSubject.next(null);

}

getCurrentUser(): User | null {

return this.currentUserSubject.value;

}

}

ออกแบบถูกไหม login คือเราได้ ค่า null

GitHub Copilot: โค้ดของคุณ \*\*แนวคิดถูกต้อง\*\* ในการแยก fetch CSRF token และ login แต่มีจุดที่ควรปรับปรุงเพื่อให้ใช้งานกับ backend ที่ใช้ CSRF + JWT + Cookie ได้สมบูรณ์:

---

### 1. \*\*loginWithCredentials\*\*

- ตอนนี้คุณคาดหวังว่า login จะ return `{ csrfToken: string }`

- แต่โดยมาตรฐาน backend ควร return `{ access\_token: string }` หรือ `{ message: 'Login successful' }`

- \*\*CSRF token ควรขอแยก\*\* (หลัง login สำเร็จ) ด้วย `fetchCsrfToken()`

- ไม่ควร expect ว่า login จะ return csrfToken

---

### 2. \*\*การใช้งานที่ถูกต้อง\*\*

- 1. เรียก `loginWithCredentials` → ได้ cookie JWT (backend set httpOnly cookie)

- 2. เรียก `fetchCsrfToken()` → ได้ CSRF token เก็บไว้ใน service

- 3. ทุก request ที่เปลี่ยนแปลงข้อมูล (POST/PUT/DELETE) ให้แนบ header `X-CSRF-Token: this.csrfToken`

---

### 3. \*\*ตัวอย่าง loginWithCredentials ที่ควรเป็น\*\*

````typescript

loginWithCredentials(username: string, password: string): Observable<boolean> {

return this.http.post<{ access\_token: string }>(

`${environment.apiUrl}/auth/login`,

{ username, password },

{ withCredentials: true }

).pipe(

tap(response => {

// ถ้าต้องการเก็บ access\_token ใน localStorage (ถ้า backend ส่งกลับมา)

if (response.access\_token) {

localStorage.setItem('token', response.access\_token);

const user = this.decodeToken(response.access\_token);

this.currentUserSubject.next(user);

}

}),

map(() => true),

catchError(() => of(false))

);

}

````

---

### 4. \*\*สรุป flow ที่ถูกต้อง\*\*

1. \*\*loginWithCredentials\*\* → ได้ JWT cookie (หรือ access\_token)

2. \*\*fetchCsrfToken\*\* → ได้ CSRF token

3. \*\*ทุก request ที่ต้องการ CSRF\*\*

- แนบ header: `X-CSRF-Token: this.csrfToken`

- และ `{ withCredentials: true }`

---

### 5. \*\*สาเหตุที่ login ได้ค่า null\*\*

- เพราะ backend ไม่ได้ return `{ csrfToken: ... }` ใน login

- หรือไม่ได้ set access\_token ใน localStorage

- หรือ decodeToken ไม่ถูกต้อง

---

### \*\*สรุป\*\*

- login ไม่ควร expect csrfToken ใน response

- login เสร็จแล้วค่อย fetch CSRF token แยก

- ทุก request ที่เปลี่ยนข้อมูลต้องแนบ CSRF token ใน header

ถ้าต้องการตัวอย่างการใช้งานกับ Angular service เพิ่มเติม แจ้งได้เลย!

pethzero: อยากให้ webui ดีไซหา ที่ เพิ่ม คน ตาง เทส api ที่ไป อัพเดท patients

GitHub Copilot: แนวทาง \*\*ดีไซน์หน้าเพิ่ม/แก้ไขคนไข้\*\* (Patient) ใน Angular (webui)

เพื่อเทส API `/api/patients` (POST/PUT) แบบที่ใช้งานจริงกับ backend

---

## 1. \*\*สร้าง Service สำหรับคนไข้\*\*

```typescript

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

import { environment } from '../../environments/environment';

export interface Patient {

id?: number;

patient\_code: string;

first\_name: string;

last\_name: string;

gender?: string;

date\_of\_birth?: string;

phone?: string;

email?: string;

address?: string;

blood\_type?: string;

}

@Injectable({ providedIn: 'root' })

export class PatientService {

constructor(private http: HttpClient) {}

createPatient(patient: Patient): Observable<Patient> {

return this.http.post<Patient>(`${environment.apiUrl}/patients`, patient);

}

updatePatient(id: number, patient: Partial<Patient>): Observable<any> {

return this.http.put(`${environment.apiUrl}/patients/${id}`, patient);

}

}

```

---

## 2. \*\*สร้างฟอร์มเพิ่ม/แก้ไขคนไข้ (Component)\*\*

```typescript

import { Component } from '@angular/core';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { PatientService, Patient } from 'src/app/core/services/patient.service';

@Component({

selector: 'app-patient-form',

standalone: true,

templateUrl: './patient-form.html',

styleUrls: ['./patient-form.scss'],

imports: [/\* ReactiveFormsModule, CommonModule, ... \*/]

})

export class PatientFormComponent {

patientForm: FormGroup;

successMessage = '';

errorMessage = '';

constructor(private fb: FormBuilder, private patientService: PatientService) {

this.patientForm = this.fb.group({

patient\_code: ['', Validators.required],

first\_name: ['', Validators.required],

last\_name: ['', Validators.required],

gender: [''],

date\_of\_birth: [''],

phone: [''],

email: [''],

address: [''],

blood\_type: ['']

});

}

submit() {

if (this.patientForm.invalid) return;

const data: Patient = this.patientForm.value;

this.patientService.createPatient(data).subscribe({

next: (res) => {

this.successMessage = 'เพิ่มคนไข้สำเร็จ!';

this.errorMessage = '';

this.patientForm.reset();

},

error: (err) => {

this.errorMessage = 'เกิดข้อผิดพลาด: ' + (err.error?.message || err.message);

this.successMessage = '';

}

});

}

}

```

---

## 3. \*\*ตัวอย่าง HTML ฟอร์ม\*\*

```html

<form [formGroup]="patientForm" (ngSubmit)="submit()" class="space-y-4 max-w-lg mx-auto bg-white p-6 rounded-xl shadow">

<div>

<label>รหัสคนไข้</label>

<input formControlName="patient\_code" class="input" required />

</div>

<div>

<label>ชื่อ</label>

<input formControlName="first\_name" class="input" required />

</div>

<div>

<label>นามสกุล</label>

<input formControlName="last\_name" class="input" required />

</div>

<div>

<label>เพศ</label>

<select formControlName="gender" class="input">

<option value="">-</option>

<option value="male">ชาย</option>

<option value="female">หญิง</option>

<option value="other">อื่นๆ</option>

</select>

</div>

<div>

<label>วันเกิด</label>

<input type="date" formControlName="date\_of\_birth" class="input" />

</div>

<div>

<label>เบอร์โทร</label>

<input formControlName="phone" class="input" />

</div>

<div>

<label>Email</label>

<input formControlName="email" class="input" />

</div>

<div>

<label>ที่อยู่</label>

<input formControlName="address" class="input" />

</div>

<div>

<label>กรุ๊ปเลือด</label>

<input formControlName="blood\_type" class="input" />

</div>

<button type="submit" class="btn btn-primary" [disabled]="patientForm.invalid">บันทึก</button>

<div \*ngIf="successMessage" class="text-green-600">{{ successMessage }}</div>

<div \*ngIf="errorMessage" class="text-red-600">{{ errorMessage }}</div>

</form>

```

---

## 4. \*\*การใช้งาน\*\*

- เพิ่ม `<app-patient-form></app-patient-form>` ในหน้า test หรือหน้า patient-list

- เมื่อ submit จะยิง API POST `/api/patients` พร้อมแนบ CSRF token อัตโนมัติ (เพราะมี csrfInterceptor อยู่แล้ว)

---

\*\*สรุป:\*\*

- สร้าง service สำหรับเรียก API

- สร้างฟอร์ม Angular สำหรับเพิ่ม/แก้ไขคนไข้

- ใช้ Reactive Form + HttpClient

- ใช้ interceptor แนบ CSRF token อัตโนมัติ

- ทดสอบเพิ่มข้อมูลคนไข้กับ backend ได้ทันที

ถ้าต้องการตัวอย่างแบบแก้ไข (edit) หรือแบบ dialog แจ้งได้เลย!

Similar code found with 1 license type

pethzero: คือ อยากให้ เริ่มโดยการสร้างตั้งแต่ folder file เลย

GitHub Copilot: แนวทาง \*\*สร้างหน้าเพิ่ม/แก้ไขคนไข้ (Patient)\*\* สำหรับ Angular webui ของคุณ

\*\*ตั้งแต่สร้างโฟลเดอร์, service, component, และเชื่อมต่อ API จริง\*\*

---

## 1. \*\*สร้างโฟลเดอร์และไฟล์\*\*

สร้างโฟลเดอร์และไฟล์ดังนี้ (ใน `src/app/pages/people/`):

```

src/app/pages/people/

patient-form/

patient-form.ts

patient-form.html

patient-form.scss

```

---

## 2. \*\*สร้าง PatientService\*\*

````typescript

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

import { environment } from '../../environments/environment';

export interface Patient {

id?: number;

patient\_code: string;

first\_name: string;

last\_name: string;

gender?: string;

date\_of\_birth?: string;

phone?: string;

email?: string;

address?: string;

blood\_type?: string;

}

@Injectable({ providedIn: 'root' })

export class PatientService {

constructor(private http: HttpClient) {}

createPatient(patient: Patient): Observable<Patient> {

return this.http.post<Patient>(`${environment.apiUrl}/patients`, patient);

}

updatePatient(id: number, patient: Partial<Patient>): Observable<any> {

return this.http.put(`${environment.apiUrl}/patients/${id}`, patient);

}

}

````

---

## 3. \*\*สร้าง Component ฟอร์ม\*\*

### patient-form.ts

````typescript

import { Component } from '@angular/core';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { PatientService, Patient } from 'src/app/core/services/patient.service';

@Component({

selector: 'app-patient-form',

standalone: true,

templateUrl: './patient-form.html',

styleUrls: ['./patient-form.scss'],

imports: []

})

export class PatientFormComponent {

patientForm: FormGroup;

successMessage = '';

errorMessage = '';

constructor(private fb: FormBuilder, private patientService: PatientService) {

this.patientForm = this.fb.group({

patient\_code: ['', Validators.required],

first\_name: ['', Validators.required],

last\_name: ['', Validators.required],

gender: [''],

date\_of\_birth: [''],

phone: [''],

email: [''],

address: [''],

blood\_type: ['']

});

}

submit() {

if (this.patientForm.invalid) return;

const data: Patient = this.patientForm.value;

this.patientService.createPatient(data).subscribe({

next: () => {

this.successMessage = 'เพิ่มคนไข้สำเร็จ!';

this.errorMessage = '';

this.patientForm.reset();

},

error: (err) => {

this.errorMessage = 'เกิดข้อผิดพลาด: ' + (err.error?.message || err.message);

this.successMessage = '';

}

});

}

}

````

---

### patient-form.html

````html

<form [formGroup]="patientForm" (ngSubmit)="submit()" class="space-y-4 max-w-lg mx-auto bg-white p-6 rounded-xl shadow">

<div>

<label>รหัสคนไข้</label>

<input formControlName="patient\_code" class="input" required />

</div>

<div>

<label>ชื่อ</label>

<input formControlName="first\_name" class="input" required />

</div>

<div>

<label>นามสกุล</label>

<input formControlName="last\_name" class="input" required />

</div>

<div>

<label>เพศ</label>

<select formControlName="gender" class="input">

<option value="">-</option>

<option value="male">ชาย</option>

<option value="female">หญิง</option>

<option value="other">อื่นๆ</option>

</select>

</div>

<div>

<label>วันเกิด</label>

<input type="date" formControlName="date\_of\_birth" class="input" />

</div>

<div>

<label>เบอร์โทร</label>

<input formControlName="phone" class="input" />

</div>

<div>

<label>Email</label>

<input formControlName="email" class="input" />

</div>

<div>

<label>ที่อยู่</label>

<input formControlName="address" class="input" />

</div>

<div>

<label>กรุ๊ปเลือด</label>

<input formControlName="blood\_type" class="input" />

</div>

<button type="submit" class="btn btn-primary" [disabled]="patientForm.invalid">บันทึก</button>

<div \*ngIf="successMessage" class="text-green-600">{{ successMessage }}</div>

<div \*ngIf="errorMessage" class="text-red-600">{{ errorMessage }}</div>

</form>

````

---

### patient-form.scss

````scss

.input {

width: 100%;

padding: 0.5rem;

border: 1px solid #e0e0e0;

border-radius: 0.5rem;

margin-bottom: 0.5rem;

}

.btn.btn-primary {

background: #1976d2;

color: white;

padding: 0.5rem 2rem;

border-radius: 0.5rem;

border: none;

font-weight: 600;

cursor: pointer;

}

````

---

## 4. \*\*เพิ่ม route ใน people-routing.module.ts\*\*

```typescript

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { PatientFormComponent } from './patient-form/patient-form';

const routes: Routes = [

{ path: 'add', component: PatientFormComponent },

// ...อื่นๆ

];

@NgModule({

imports: [RouterModule.forChild(routes)],

exports: [RouterModule]

})

export class PeopleRoutingModule {}

```

---

## 5. \*\*ใช้งาน\*\*

- ไปที่ `/patient/add` จะเห็นฟอร์มเพิ่มคนไข้

- กรอกข้อมูล กดบันทึก → ยิง API `/api/patients` (POST) พร้อม CSRF token อัตโนมัติ (ถ้าใช้ csrfInterceptor)

---

\*\*สรุป:\*\*

- สร้างโฟลเดอร์/ไฟล์ตามตัวอย่าง

- สร้าง service, component, html, scss

- เพิ่ม route

- ทดสอบเพิ่มคนไข้กับ backend ได้ทันที

ถ้าต้องการตัวอย่างแบบแก้ไข (edit) หรือ list แจ้งได้เลย!

Similar code found with 1 license type

pethzero: ทำไมเราถึงกด เข้าสู่ระบบ 2 รอบเหมือนจะ error รอบแรก

GitHub Copilot: ปัญหาที่คุณต้อง \*\*กด "เข้าสู่ระบบ" 2 รอบ\*\* ถึงจะเข้าได้ มักเกิดจากสาเหตุเหล่านี้ (ใน Angular + CSRF + Cookie):

---

### \*\*1. ยังไม่มี CSRF token ใน memory ตอน login ครั้งแรก\*\*

- เมื่อคุณเปิดหน้า login แล้วกดเข้าสู่ระบบทันที

- ถ้า `AuthService` ยังไม่ได้ fetch CSRF token (`csrfToken` ยังว่าง)

- interceptor จะไม่แนบ `X-CSRF-Token` ใน request

- backend จะ reject ด้วย error CSRF

- พอกดรอบสอง CSRF token ถูก fetch แล้ว จึง login ผ่าน

---

### \*\*2. Flow ที่ถูกต้องควรเป็น\*\*

1. \*\*เมื่อเข้าแอปหรือหน้า login:\*\*

- ควร fetch CSRF token ทันที (เช่น ใน `ngOnInit` ของหน้า login หรือใน AuthService)

2. \*\*เมื่อ login:\*\*

- CSRF token พร้อมใช้งาน interceptor จะแนบ header ให้ทันที

---

### \*\*3. วิธีแก้ไข\*\*

\*\*เพิ่มการ fetch CSRF token ในหน้า login ก่อน login จริง\*\*

```typescript

import { Component, OnInit } from '@angular/core';

// ...existing import...

export class LoginComponent implements OnInit {

// ...existing code...

ngOnInit(): void {

// เรียก fetch CSRF token ทันทีที่เข้า component

this.auth.fetchCsrfToken().subscribe();

}

login(): void {

this.auth.loginWithCredentials(this.username, this.password).subscribe(success => {

if (success) {

this.router.navigate(['/dashboard']);

} else {

this.errorMessage = 'ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง';

}

});

}

// ...existing code...

}

```

\*\*หรือ\*\*

- ให้ AuthService fetch CSRF token อัตโนมัติใน constructor (แต่จะเปลือง request ทุก reload)

- หรือให้ Interceptor ดัก request แรกแล้ว fetch CSRF token ให้อัตโนมัติ (ซับซ้อนขึ้น)

---

### \*\*4. สรุป\*\*

- ปัญหาเกิดจาก CSRF token ยังไม่ถูกโหลดใน memory ตอน login ครั้งแรก

- แก้โดย fetch CSRF token ใน `ngOnInit` ของหน้า login ทุกครั้งก่อน login

- หลังจากนี้จะกด "เข้าสู่ระบบ" แค่ครั้งเดียวก็เข้าได้

---

\*\*ถ้าต้องการตัวอย่างโค้ดเต็มหรือวิธีแก้แบบอื่น แจ้งได้เลย!\*\*

pethzero: ดูเหมือนจะเดี่ยวกับ admin-login นะ แก้ไขยังไงดี /login ได้อยู่แล้ว

GitHub Copilot: สาเหตุที่ \*\*/login ใช้งานได้\*\* แต่ \*\*/admin/login\*\* มีปัญหา

เพราะในไฟล์ `admin-login.ts` คุณเรียก `this.auth.loginWithCredentials(...)` แล้ว \*\*ไม่ได้ subscribe\*\* (Observable จะไม่ถูก execute ถ้าไม่ subscribe)

แต่ใน `/login` (user) คุณ subscribe แล้ว

---

### วิธีแก้ไข (admin-login.ts)

\*\*เปลี่ยนจาก:\*\*

```typescript

login() {

const success = this.auth.loginWithCredentials(this.username, this.password);

if (success) {

this.router.navigate(['/dashboard']);

} else {

this.errorMessage = 'ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง';

}

}

```

\*\*เป็น:\*\*

```typescript

login() {

this.auth.loginWithCredentials(this.username, this.password).subscribe(success => {

if (success) {

this.router.navigate(['/dashboard']);

} else {

this.errorMessage = 'ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง';

}

});

}

```

\*\*และสำหรับ guest:\*\*

```typescript

loginAsGuest() {

this.auth.loginAsGuest().subscribe(success => {

if (success) {

this.router.navigate(['/dashboard']);

} else {

this.errorMessage = 'ไม่สามารถเข้าสู่ระบบ guest ได้';

}

});

}

```

---

\*\*สรุป:\*\*

- ต้อง subscribe ทุกครั้งที่ใช้ Observable จาก service

- แบบนี้จะทำงานเหมือน `/login` และแก้ปัญหา `/admin/login` ได้ทันที

---

\*\*เพิ่มเติม:\*\*

- ถ้าอยากแยกสิทธิ์ admin จริง ๆ ให้ตรวจสอบ role หลัง login ด้วย (เช่น redirect เฉพาะ admin)

pethzero: แก้ไขปัญหายังไงดี

GitHub Copilot: ### ปัญหา

- คุณมี path `/login` และ `/admin-login` (หรือ `/admin/login`)

- แต่ในไฟล์ route มีทั้งแบบ \*\*lazy load\*\* (`loadComponent`) และแบบ \*\*children\*\*

- ใน login-routing.module.ts ใช้ path `login-user` กับ `admin-login`

- ใน app.routes.ts ใช้ path `/login` และ `/admin-login` (แบบ root)

- หน้า `/admin-login` กับ `/admin/login` อาจสับสนหรือ route ไม่ตรงกัน

---

### \*\*แนวทางแก้ไข\*\*

#### 1. \*\*แนะนำให้ใช้ path เดียวกันทั้งระบบ\*\*

- ถ้าอยากให้ admin login ที่ `/admin/login`

- ให้แก้ route เป็น `/admin/login`

- ถ้าอยากให้ admin login ที่ `/admin-login`

- ให้ใช้ `/admin-login` ให้ตรงกันทุกที่

#### 2. \*\*แก้ไขใน app.routes.ts\*\*

```typescript

{

path: 'login',

canActivate: [LoginRedirectGuard],

loadComponent: () => import('./pages/login/user/login').then(m => m.LoginComponent)

},

{

path: 'admin/login', // เปลี่ยนเป็น admin/login

canActivate: [LoginRedirectGuard],

loadComponent: () => import('./pages/login/admin/admin-login').then(m => m.AdminLoginComponent)

},

```

#### 3. \*\*แก้ไขลิงก์ในหน้า login\*\*

ใน login.html (user)

```html

<a routerLink="/admin/login" class="text-blue-600 hover:underline">

สำหรับผู้ดูแลระบบ (admin portal)

</a>

```

ใน admin-login.html

(ถ้าต้องการลิงก์กลับ user login)

```html

<a routerLink="/login" class="text-blue-600 hover:underline">

สำหรับผู้ใช้ทั่วไป (user portal)

</a>

```

#### 4. \*\*ไม่จำเป็นต้องใช้ login-routing.module.ts ถ้าใช้แบบ loadComponent\*\*

- ถ้าใช้แบบ lazy load ด้วย module ให้ใช้ children

- ถ้าใช้แบบ loadComponent ให้ลบ login-routing.module.ts ออกได้

---

### \*\*สรุป\*\*

- ใช้ path `/login` สำหรับ user, `/admin/login` สำหรับ admin

- ให้ path ใน route และลิงก์ใน html ตรงกัน

- ไม่ต้องมี path `/admin-login` หรือ `/login-user` ซ้ำซ้อน

- ตรวจสอบว่าใน browser พิมพ์ `/admin/login` แล้วเจอหน้า admin login จริง

---

\*\*ตัวอย่างสั้นๆ ใน app.routes.ts:\*\*

```typescript

{

path: 'login',

loadComponent: () => import('./pages/login/user/login').then(m => m.LoginComponent)

},

{

path: 'admin/login',

loadComponent: () => import('./pages/login/admin/admin-login').then(m => m.AdminLoginComponent)

},

```

---

\*\*สรุป:\*\*

- ให้ใช้ path เดียวกันทั้งใน route และลิงก์

- ไม่ต้องมีหลายไฟล์ routing module สำหรับ login ถ้าใช้ loadComponent

- ปัญหาจะหมดไป และ `/login` กับ `/admin/login` จะใช้งานได้ถูกต้อง

ถ้าต้องการตัวอย่าง route หรือแก้ไขไฟล์ไหนเพิ่มเติม แจ้งได้เลย!

pethzero:

▲ [WARNING] NG8103: The `\*ngIf` directive was used in the template, but neither the `NgIf` directive nor the `CommonModule` was imported. Use Angular's built-in control flow @if or make sure that either the `NgIf` directive or the `CommonModule` is included in the `@Component.imports` array of this component. [plugin angular-compiler]

src/app/pages/people/patient-form/patient-form.html:45:8:

45 │ <div \*ngIf="errorMessage" class="text-red-600">{{ errorMessage }...

╵ ~~~~

Error occurs in the template of component PatientFormComponent.

src/app/pages/people/patient-form/patient-form.ts:9:15:

9 │ templateUrl: './patient-form.html',

╵ ~~~~~~~~~~~~~~~~~~~~~

X [ERROR] TS2339: Property 'getCsrfToken' does not exist on type 'AuthService'. [plugin angular-compiler]

src/app/core/interceptors/csrf-interceptor.ts:8:32:

8 │ const csrfToken = authService.getCsrfToken();

╵ ~~~~~~~~~~~~

X [ERROR] NG8002: Can't bind to 'formGroup' since it isn't a known property of 'form'. [plugin angular-compiler]

src/app/pages/people/patient-form/patient-form.html:1:6:

1 │ <form [formGroup]="patientForm" (ngSubmit)="submit()" class="space-...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component PatientFormComponent.

src/app/pages/people/patient-form/patient-form.ts:9:15:

9 │ templateUrl: './patient-form.html',

GitHub Copilot: ปัญหาที่เจอเกิดจาก

1. \*\*ไม่ได้ import CommonModule\*\* (สำหรับ `\*ngIf`)

2. \*\*ไม่ได้ import ReactiveFormsModule\*\* (สำหรับ `[formGroup]`)

3. \*\*AuthService ไม่มีเมธอด `getCsrfToken()`\*\* (ใน `csrf-interceptor.ts`)

---

### วิธีแก้ไข

#### 1. เพิ่ม CommonModule และ ReactiveFormsModule ใน `@Component.imports`

```typescript

import { Component } from '@angular/core';

import { FormBuilder, FormGroup, Validators, ReactiveFormsModule } from '@angular/forms';

import { CommonModule } from '@angular/common';

import { PatientService } from 'src/app/core/services/patient.service';

@Component({

selector: 'app-patient-form',

standalone: true,

templateUrl: './patient-form.html',

styleUrls: ['./patient-form.scss'],

imports: [CommonModule, ReactiveFormsModule]

})

export class PatientFormComponent {

// ...existing code...

}

```

---

#### 2. เพิ่มเมธอด `getCsrfToken()` ใน AuthService

```typescript

getCsrfToken(): string {

return this.csrfToken;

}

```

---

### สรุป

- เพิ่ม `CommonModule` และ `ReactiveFormsModule` ใน `imports` ของ component

- เพิ่มเมธอด `getCsrfToken()` ใน AuthService

\*\*หลังแก้ไขนี้ error จะหายและฟอร์มจะใช้งานได้ปกติ\*\*

pethzero: ถ้าเราติดตั้ง

npm install --save @nestjs/swagger

ระบบต้องปรับอะไรบ้าง

GitHub Copilot: หลังจากติดตั้ง

```sh

npm install --save @nestjs/swagger swagger-ui-express

```

\*\*คุณต้องปรับดังนี้เพื่อเปิดใช้งาน Swagger ใน NestJS:\*\*

---

### 1. \*\*เพิ่มโค้ดใน main.ts\*\*

````typescript

import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';

// ...existing import...

async function bootstrap() {

const app = await NestFactory.create(AppModule);

// ...existing code...

// --- เพิ่มส่วนนี้สำหรับ Swagger ---

const config = new DocumentBuilder()

.setTitle('Kornpassorn Care API')

.setDescription('API documentation')

.setVersion('1.0')

.addBearerAuth()

.build();

const document = SwaggerModule.createDocument(app, config);

SwaggerModule.setup('api/docs', app, document);

// --- จบส่วน Swagger ---

await app.listen(process.env.PORT || 3000);

}

bootstrap();

````

---

### 2. \*\*(แนะนำ) ตกแต่ง DTO และ Controller ด้วย Decorator ของ Swagger\*\*

ตัวอย่าง:

```typescript

import { ApiProperty } from '@nestjs/swagger';

export class CreatePatientDto {

@ApiProperty({ example: 'P001' })

patient\_code: string;

@ApiProperty({ example: 'John' })

first\_name: string;

// ...field อื่นๆ...

}

```

และใน Controller:

```typescript

import { ApiTags, ApiBearerAuth } from '@nestjs/swagger';

@ApiTags('patients')

@ApiBearerAuth()

@Controller('patients')

export class PatientController { ... }

```

---

### 3. \*\*เข้าใช้งาน Swagger UI\*\*

- รันเซิร์ฟเวอร์ แล้วเปิดเบราว์เซอร์ไปที่

```

http://localhost:3000/api/docs

```

---

\*\*สรุป:\*\*

- เพิ่มโค้ด Swagger ใน main.ts

- (แนะนำ) ตกแต่ง DTO/Controller ด้วย decorator ของ Swagger

- เข้าใช้งานที่ `/api/docs`

ถ้าต้องการตัวอย่าง DTO หรือ Controller ที่ตกแต่งด้วย Swagger เพิ่มเติม แจ้งได้เลย!

pethzero: นี้คือ folder ขอให้ดูใหม่และปรับปรุง

GitHub Copilot: จากโครงสร้างโฟลเดอร์และไฟล์ใน #, คุณติดตั้ง `@nestjs/swagger` แล้ว

\*\*แนวทางปรับปรุงเพื่อเปิดใช้งาน Swagger API Docs ในโปรเจกต์ของคุณ:\*\*

---

### 1. \*\*เพิ่ม Swagger ใน main.ts\*\*

เพิ่มโค้ดนี้ในไฟล์ `src/main.ts` ก่อน `await app.listen(...)`:

````typescript

import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';

// ...existing import...

async function bootstrap() {

const app = await NestFactory.create(AppModule);

// ...existing code...

// Swagger setup

const config = new DocumentBuilder()

.setTitle('Kornpassorn Care API')

.setDescription('API documentation')

.setVersion('1.0')

.addBearerAuth()

.build();

const document = SwaggerModule.createDocument(app, config);

SwaggerModule.setup('api/docs', app, document);

await app.listen(process.env.PORT || 3000);

}

bootstrap();

````

---

### 2. \*\*(แนะนำ) ตกแต่ง DTO และ Controller ด้วย Swagger Decorator\*\*

ตัวอย่างใน DTO:

````typescript

// filepath: src/modules/auth/dto/login.dto.ts

import { ApiProperty } from '@nestjs/swagger';

export class LoginDto {

@ApiProperty({ example: 'admin' })

username: string;

@ApiProperty({ example: 'password123' })

password: string;

}

````

ตัวอย่างใน Controller:

````typescript

import { ApiTags, ApiBearerAuth } from '@nestjs/swagger';

@ApiTags('patients')

@ApiBearerAuth()

@Controller('patients')

export class PatientController { ... }

````

---

### 3. \*\*เข้าใช้งาน Swagger UI\*\*

- รันเซิร์ฟเวอร์ แล้วเปิดเบราว์เซอร์ไปที่

```

http://localhost:3000/api/docs

```

---

### 4. \*\*(Optional) ปรับปรุงโครงสร้างโฟลเดอร์\*\*

- โครงสร้างปัจจุบันของคุณเหมาะสมแล้วสำหรับ NestJS (modules, entities, common, config, etc.)

- ถ้าต้องการแยก API docs หรือ custom decorator เพิ่มเติม สามารถสร้างใน `src/common/` ได้

---

\*\*สรุป:\*\*

- เพิ่ม Swagger ใน main.ts

- (แนะนำ) ตกแต่ง DTO/Controller ด้วย decorator

- เข้าใช้งานที่ `/api/docs`

- โครงสร้างโฟลเดอร์ของคุณเหมาะสมกับ NestJS แล้ว

ถ้าต้องการตัวอย่างการตกแต่ง Swagger เพิ่มเติม แจ้งได้เลย!

Similar code found with 1 license type

pethzero: http://localhost:3000/api/docs เอาชื่อ docs เท่ๆ กว่านี้ได้ไหม หรือมันคือ มาตราญาน

GitHub Copilot: คุณสามารถเปลี่ยน path `/api/docs` เป็นอะไรก็ได้ตามต้องการ ไม่จำเป็นต้องเป็น `/docs`

ตัวอย่างชื่อที่นิยมใช้ เช่น

- `/api/swagger`

- `/api/api-docs`

- `/api/openapi`

- `/api/documentation`

- `/api/explorer`

\*\*วิธีเปลี่ยน:\*\*

เช่น ถ้าอยากใช้ `/api/api-docs`

```typescript

SwaggerModule.setup('api/api-docs', app, document);

```

หรือถ้าอยากเท่ๆ เช่น `/api/explorer`

```typescript

SwaggerModule.setup('api/explorer', app, document);

```

\*\*สรุป:\*\*

- เปลี่ยนได้อิสระ ไม่กระทบมาตรฐาน

- เลือกชื่อที่เหมาะกับทีม/องค์กร/branding ได้เลย

\*\*ตัวอย่าง:\*\*

```typescript

SwaggerModule.setup('api/explorer', app, document);

```

แล้วเข้าใช้งานที่

```

http://localhost:3000/api/explorer

```

pethzero: เขียนให้มัน รอบรับ swager ยังไงได้บ้าง

GitHub Copilot: คุณตั้งค่า Swagger ใน main.ts ถูกต้องแล้ว (Swagger UI ที่ `/api/openapi`)

แต่ถ้าต้องการให้ Swagger รองรับการใช้งาน (เช่น ทดสอบ API ได้จริง) \*\*โดยไม่ติด CSRF\*\*

ควร \*\*skip CSRF สำหรับ path Swagger\*\* ด้วย

---

### วิธีปรับปรุงให้ Swagger ใช้งานได้ (เช่น Try it out ได้)

\*\*1. เพิ่ม path Swagger (`/api/openapi`, `/api-json`) ใน skipPaths\*\*

```typescript

const skipPaths = [

'/api/auth/login',

'/api/auth/guest',

'/api/openapi', // Swagger UI

'/api-json', // Swagger JSON (ถ้าใช้)

];

```

\*\*2. ปรับ middleware ให้รองรับ path ที่ขึ้นต้นด้วย `/api/openapi`\*\*

```typescript

app.use((req, res, next) => {

if (

skipPaths.includes(req.path) ||

req.path.startsWith('/api/openapi') || // รองรับ /api/openapi และ static asset ของ Swagger

req.path.startsWith('/api-json')

) {

return next();

}

csrfProtection(req, res, next);

});

```

---

### ตัวอย่างโค้ดเต็ม (เฉพาะส่วนที่เกี่ยวข้อง)

````typescript

const skipPaths = [

'/api/auth/login',

'/api/auth/guest',

'/api/openapi',

'/api-json',

];

app.use((req, res, next) => {

if (

skipPaths.includes(req.path) ||

req.path.startsWith('/api/openapi') ||

req.path.startsWith('/api-json')

) {

return next();

}

csrfProtection(req, res, next);

});

````

---

### หมายเหตุ

- Swagger UI จะสามารถ "Try it out" ได้โดยไม่ติด CSRF

- ถ้าอยากให้ปลอดภัยใน production ให้เปิด CSRF เฉพาะ API จริง ไม่รวม Swagger

---

\*\*สรุป:\*\*

- เพิ่ม path Swagger ใน skipPaths

- หรือใช้ `req.path.startsWith('/api/openapi')` เพื่อข้าม CSRF

- Swagger จะใช้งานได้เต็มที่ (Try it out ได้)

pethzero: /api/openapi คือก็อยากทดสอบ csrf-token

GitHub Copilot: ถ้าคุณต้องการให้ Swagger (`/api/openapi`) สามารถ \*\*ทดสอบ API ที่ต้องใช้ CSRF token\*\* ได้ (เช่น POST/PUT/DELETE)

\*\*แต่ยังคงปกป้อง CSRF สำหรับ API จริง\*\*

ให้ \*\*ไม่ skip `/api/csrf-token`\*\* ใน `skipPaths`

และเพิ่ม endpoint `/api/auth/csrf-token` สำหรับขอ CSRF token

---

### วิธีใช้งานกับ Swagger

1. \*\*Swagger UI\*\* จะสามารถเรียก `/api/auth/csrf-token` ได้ (ไม่ต้อง skip ใน skipPaths)

2. ทุก request ที่ต้องการ CSRF (POST/PUT/DELETE)

- ให้ copy ค่า `csrfToken` ที่ได้จาก `/api/auth/csrf-token`

- ใส่ใน header `X-CSRF-Token` ตอนกด "Try it out" ใน Swagger

---

### ตัวอย่างการตั้งค่า (main.ts)

```typescript

const skipPaths = [

'/api/auth/login',

'/api/auth/guest',

'/api/openapi',

'/api-json',

// \*\*\* ไม่ต้องใส่ '/api/auth/csrf-token' ใน skipPaths \*\*\*

];

app.use((req, res, next) => {

if (

skipPaths.includes(req.path) ||

req.path.startsWith('/api/openapi') ||

req.path.startsWith('/api-json')

) {

return next();

}

csrfProtection(req, res, next);

});

```

---

### วิธีทดสอบ CSRF ใน Swagger

1. ไปที่ `/api/openapi`

2. เรียก `GET /api/auth/csrf-token` → จะได้ `{ "csrfToken": "xxxx" }`

3. เวลา "Try it out" กับ API ที่ต้องการ CSRF (เช่น POST/PUT/DELETE)

- ใส่ header:

```

X-CSRF-Token: xxxx

```

- ถ้า API ต้องใช้ cookie ด้วย ให้แนบ cookie ที่ได้จาก `/api/auth/csrf-token` ด้วย (Swagger UI จะส่ง cookie ให้อัตโนมัติถ้าอยู่ใน session เดียวกัน)

---

\*\*สรุป:\*\*

- อย่า skip `/api/auth/csrf-token` ใน skipPaths

- ให้ Swagger UI เรียก `/api/auth/csrf-token` ได้

- ทุก request ที่ต้องการ CSRF ให้ใส่ header `X-CSRF-Token` ด้วยค่า token ที่ได้

\*\*แบบนี้จะทดสอบ CSRF-protected API ผ่าน Swagger ได้สมบูรณ์\*\* - ถ้า API ต้องใช้ cookie ด้วย ให้แนบ cookie ที่ได้จาก `/api/auth/csrf-token` ด้วย (Swagger UI จะส่ง cookie ให้อัตโนมัติถ้าอยู่ใน session เดียวกัน)

---

\*\*สรุป:\*\*

- อย่า skip `/api/auth/csrf-token` ใน skipPaths

- ให้ Swagger UI เรียก `/api/auth/csrf-token` ได้

- ทุก request ที่ต้องการ CSRF ให้ใส่ header `X-CSRF-Token` ด้วยค่า token ที่ได้

\*\*แบบนี้จะทดสอบ CSRF-protected API ผ่าน Swagger ได้สมบูรณ์\*\*

pethzero: เกิดข้อผิดพลาด: Http failure response for http://localhost:3000/api/patients: 0 Unknown Error

ccess to XMLHttpRequest at 'http://localhost:3000/api/patients' from origin 'http://localhost:4200' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

[NEW] Explain Console errors by using Copilot in Edge: click

to explain an error.

Learn more

Don't show again

error-interceptor.ts:9 [HTTP ERROR] HttpErrorResponse {headers: \_HttpHeaders, status: 0, statusText: 'Unknown Error', url: 'http://localhost:3000/api/patients', ok: false, …}

overrideMethod @ hook.js:608

(anonymous) @ error-interceptor.ts:9

(anonymous) @ catchError.js:10

OperatorSubscriber2.\_this.\_error @ OperatorSubscriber.js:25

Subscriber2.error @ Subscriber.js:43

Subscriber2.\_error @ Subscriber.js:67

Subscriber2.error @ Subscriber.js:43

onError @ module-CBsxN\_3E.mjs:2469

invokeTask @ zone.js:431

(anonymous) @ debug\_node-DTOmNMDH.mjs:16160

onInvokeTask @ debug\_node-DTOmNMDH.mjs:16160

invokeTask @ zone.js:430

onInvokeTask @ debug\_node-DTOmNMDH.mjs:16483

invokeTask @ zone.js:430

runTask @ zone.js:161

invokeTask @ zone.js:515

invokeTask @ zone.js:1141

globalCallback @ zone.js:1172

globalZoneAwareCallback @ zone.js:1205

Zone - XMLHttpRequest.addEventListener:error

onScheduleTask @ debug\_node-DTOmNMDH.mjs:16154

scheduleTask @ zone.js:411

onScheduleTask @ zone.js:273

scheduleTask @ zone.js:411

scheduleTask @ zone.js:207

scheduleEventTask @ zone.js:233

(anonymous) @ zone.js:1498

(anonymous) @ module-CBsxN\_3E.mjs:2522

Observable2.\_trySubscribe @ Observable.js:38

(anonymous) @ Observable.js:32

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ switchMap.js:14

OperatorSubscriber2.\_this.\_next @ OperatorSubscriber.js:15

Subscriber2.next @ Subscriber.js:34

(anonymous) @ innerFrom.js:51

Observable2.\_trySubscribe @ Observable.js:38

(anonymous) @ Observable.js:32

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ switchMap.js:10

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ catchError.js:9

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ finalize.js:5

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

doInnerSub @ mergeInternals.js:19

outerNext @ mergeInternals.js:14

OperatorSubscriber2.\_this.\_next @ OperatorSubscriber.js:15

Subscriber2.next @ Subscriber.js:34

(anonymous) @ innerFrom.js:51

Observable2.\_trySubscribe @ Observable.js:38

(anonymous) @ Observable.js:32

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

mergeInternals @ mergeInternals.js:53

(anonymous) @ mergeMap.js:14

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ filter.js:6

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ map.js:6

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

submit @ patient-form.ts:36

PatientFormComponent\_Template\_form\_ngSubmit\_0\_listener @ patient-form.html:1

executeListenerWithErrorHandling @ debug\_node-DTOmNMDH.mjs:12874

wrapListenerIn\_markDirtyAndPreventDefault @ debug\_node-DTOmNMDH.mjs:12857

ConsumerObserver2.next @ Subscriber.js:96

Subscriber2.\_next @ Subscriber.js:63

Subscriber2.next @ Subscriber.js:34

(anonymous) @ Subject.js:41

errorContext @ errorContext.js:19

Subject2.next @ Subject.js:31

emit @ debug\_node-DTOmNMDH.mjs:16030

onSubmit @ forms.mjs:5630

FormGroupDirective\_submit\_HostBindingHandler @ forms.mjs:5727

executeListenerWithErrorHandling @ debug\_node-DTOmNMDH.mjs:12874

wrapListenerIn\_markDirtyAndPreventDefault @ debug\_node-DTOmNMDH.mjs:12857

(anonymous) @ dom\_renderer-Frqw9gM5.mjs:709

invokeTask @ zone.js:431

(anonymous) @ debug\_node-DTOmNMDH.mjs:16160

onInvokeTask @ debug\_node-DTOmNMDH.mjs:16160

invokeTask @ zone.js:430

onInvokeTask @ debug\_node-DTOmNMDH.mjs:16483

invokeTask @ zone.js:430

runTask @ zone.js:161

invokeTask @ zone.js:515

invokeTask @ zone.js:1141

globalCallback @ zone.js:1172

globalZoneAwareCallback @ zone.js:1205

patient-form.ts:36 POST http://localhost:3000/api/patients net::ERR\_FAILED 500 (Internal Server Error)

scheduleTask @ zone.js:2182

scheduleTask @ zone.js:417

onScheduleTask @ debug\_node-DTOmNMDH.mjs:16155

scheduleTask @ zone.js:411

onScheduleTask @ zone.js:273

scheduleTask @ zone.js:411

scheduleTask @ zone.js:207

scheduleMacroTask @ zone.js:230

scheduleMacroTaskWithCurrentZone @ zone.js:723

(anonymous) @ zone.js:2221

proto.<computed> @ zone.js:1038

(anonymous) @ module-CBsxN\_3E.mjs:2535

Observable2.\_trySubscribe @ Observable.js:38

(anonymous) @ Observable.js:32

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ switchMap.js:14

OperatorSubscriber2.\_this.\_next @ OperatorSubscriber.js:15

Subscriber2.next @ Subscriber.js:34

(anonymous) @ innerFrom.js:51

Observable2.\_trySubscribe @ Observable.js:38

(anonymous) @ Observable.js:32

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ switchMap.js:10

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ catchError.js:9

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ finalize.js:5

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

doInnerSub @ mergeInternals.js:19

outerNext @ mergeInternals.js:14

OperatorSubscriber2.\_this.\_next @ OperatorSubscriber.js:15

Subscriber2.next @ Subscriber.js:34

(anonymous) @ innerFrom.js:51

Observable2.\_trySubscribe @ Observable.js:38

(anonymous) @ Observable.js:32

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

mergeInternals @ mergeInternals.js:53

(anonymous) @ mergeMap.js:14

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ filter.js:6

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

(anonymous) @ map.js:6

(anonymous) @ lift.js:10

(anonymous) @ Observable.js:27

errorContext @ errorContext.js:19

Observable2.subscribe @ Observable.js:23

submit @ patient-form.ts:36

PatientFormComponent\_Template\_form\_ngSubmit\_0\_listener @ patient-form.html:1

executeListenerWithErrorHandling @ debug\_node-DTOmNMDH.mjs:12874

wrapListenerIn\_markDirtyAndPreventDefault @ debug\_node-DTOmNMDH.mjs:12857

ConsumerObserver2.next @ Subscriber.js:96

Subscriber2.\_next @ Subscriber.js:63

Subscriber2.next @ Subscriber.js:34

(anonymous) @ Subject.js:41

errorContext @ errorContext.js:19

Subject2.next @ Subject.js:31

emit @ debug\_node-DTOmNMDH.mjs:16030

onSubmit @ forms.mjs:5630

FormGroupDirective\_submit\_HostBindingHandler @ forms.mjs:5727

executeListenerWithErrorHandling @ debug\_node-DTOmNMDH.mjs:12874

wrapListenerIn\_markDirtyAndPreventDefault @ debug\_node-DTOmNMDH.mjs:12857

(anonymous) @ dom\_renderer-Frqw9gM5.mjs:709

invokeTask @ zone.js:431

(anonymous) @ debug\_node-DTOmNMDH.mjs:16160

onInvokeTask @ debug\_node-DTOmNMDH.mjs:16160

invokeTask @ zone.js:430

onInvokeTask @ debug\_node-DTOmNMDH.mjs:16483

invokeTask @ zone.js:430

runTask @ zone.js:161

invokeTask @ zone.js:515

invokeTask @ zone.js:1141

globalCallback @ zone.js:1172

globalZoneAwareCallback @ zone.js:1205

GitHub Copilot: จาก error ที่คุณเจอ

- `No 'Access-Control-Allow-Origin' header is present on the requested resource.`

- `Http failure response for http://localhost:3000/api/patients: 0 Unknown Error`

- `net::ERR\_FAILED 500 (Internal Server Error)`

\*\*สาเหตุหลัก:\*\*

1. \*\*CORS\*\*: Backend ยังไม่ได้เปิด CORS หรือเปิดไม่ถูกต้อง

2. \*\*CSRF\*\*: ถ้า CORS ถูกต้องแล้ว อาจติด CSRF (เช่น ไม่ได้แนบ CSRF token หรือ cookie)

---

## วิธีแก้ไข

### 1. ตรวจสอบการตั้งค่า CORS ใน backend

ใน main.ts ของ NestJS \*\*ต้องมีแบบนี้\*\* (และอยู่ก่อน listen):

````typescript

app.enableCors({

origin: 'http://localhost:4200',

credentials: true,

});

````

### 2. ตรวจสอบว่า frontend ส่ง cookie และ CSRF token

- ทุก request ที่เปลี่ยนข้อมูล (POST/PUT/DELETE) ต้องแนบ

- header: `X-CSRF-Token: <token>`

- option: `{ withCredentials: true }` ใน HttpClient

### 3. ตรวจสอบว่า backend ตอบ header CORS ถูกต้อง

- ลองเรียก API `/api/patients` ด้วย Postman หรือ curl ดูว่าได้ response หรือไม่

- ถ้าไม่ได้ ให้ดู log ฝั่ง backend ว่ามี error อะไร

### 4. ตรวจสอบว่า backend ไม่ crash

- ดู log terminal ของ backend ว่ามี error อื่นหรือไม่ (เช่น database, validate, etc.)

---

## Checklist

- [x] `app.enableCors({ origin: 'http://localhost:4200', credentials: true })`

- [x] frontend ส่ง `{ withCredentials: true }` ทุก request

- [x] แนบ CSRF token ใน header ทุก POST/PUT/DELETE

- [x] backend ไม่ crash และตอบ 200/201

---

## ตัวอย่างการตั้งค่า (main.ts)

````typescript

app.enableCors({

origin: 'http://localhost:4200',

credentials: true,

});

````

---

\*\*ถ้ายัง error:\*\*

- แนบ log ฝั่ง backend เพิ่มเติม

- ลองเรียก API ด้วย Postman แล้วดู response header ว่ามี `Access-Control-Allow-Origin` หรือไม่

- ตรวจสอบว่า frontend กับ backend ใช้ port และ protocol ตรงกัน (เช่น http ทั้งคู่)

---

\*\*สรุป:\*\*

- ปัญหานี้เกิดจาก CORS หรือ CSRF

- ให้แก้ไขตาม checklist ข้างบน

- ถ้ายังไม่หาย แจ้ง log backend เพิ่มเติมได้เลย

pethzero: [ERROR] 500: invalid csrf token

[ERROR] 500: invalid csrf token

วิธีแก้ไข patient-form:1 Access to XMLHttpRequest at 'http://localhost:3000/api/patients' from origin 'http://localhost:4200' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

GitHub Copilot: ### ปัญหา

- \*\*CORS:\*\* `No 'Access-Control-Allow-Origin' header is present...`

- \*\*CSRF:\*\* `[ERROR] 500: invalid csrf token`

---

## วิธีแก้ไข (ทีละขั้น)

### 1. ตรวจสอบลำดับการเรียก `app.enableCors()`

\*\*ต้องเรียก `app.enableCors()` ก่อน middleware อื่นๆ\*\*

\*\*แก้ไขให้เป็นแบบนี้:\*\*

````typescript

async function bootstrap() {

const app = await NestFactory.create(AppModule);

app.enableCors({

origin: 'http://localhost:4200',

credentials: true,

});

app.use(cookieParser());

app.use(json());

app.setGlobalPrefix('api');

// ...csrf, skipPaths, app.use(...)...

// ...Swagger, pipes, interceptors, filters...

await app.listen(process.env.PORT || 3000);

}

````

---

### 2. ตรวจสอบ CSRF Token ฝั่ง Frontend

- ก่อน submit ฟอร์ม ต้องเรียก `/api/auth/csrf-token` (GET) เพื่อรับ token และ cookie

- ทุก POST/PUT/DELETE ต้องแนบ header `X-CSRF-Token` และ `{ withCredentials: true }`

- ตรวจสอบว่า Interceptor ของคุณแนบ token ถูกต้อง

---

### 3. ตรวจสอบ skipPaths

- \*\*อย่าใส่\*\* `/api/auth/csrf-token` ใน skipPaths (ถูกต้องแล้ว)

- Swagger (`/api/openapi`) ควรอยู่ใน skipPaths

---

### 4. ตรวจสอบ Cookie

- Frontend ต้องส่ง cookie ที่ได้จาก `/api/auth/csrf-token` กลับไปทุก request (Angular: `{ withCredentials: true }`)

---

## สรุปโค้ดตัวอย่าง (main.ts)

````typescript

async function bootstrap() {

const app = await NestFactory.create(AppModule);

// 1. เปิด CORS ก่อน middleware อื่น

app.enableCors({

origin: 'http://localhost:4200',

credentials: true,

});

app.use(cookieParser());

app.use(json());

app.setGlobalPrefix('api');

const csrfProtection = csurf({

cookie: {

httpOnly: false, // ⚠️ production ควร true

sameSite: 'strict',

secure: false,

},

});

const skipPaths = [

'/api/auth/login',

'/api/auth/guest',

'/api/openapi',

'/api-json',

];

app.use((req, res, next) => {

if (

skipPaths.includes(req.path) ||

req.path.startsWith('/api/openapi') ||

req.path.startsWith('/api-json')

) {

return next();

}

csrfProtection(req, res, next);

});

// ...Swagger, pipes, interceptors, filters...

await app.listen(process.env.PORT || 3000);

}

bootstrap();

````

---

## สรุป

- \*\*CORS ต้องเปิดก่อน middleware อื่น\*\*

- \*\*Frontend ต้องแนบ CSRF token และ cookie ทุก request\*\*

- \*\*Backend ต้องไม่ crash และตอบ header CORS ทุกกรณี\*\*

---

\*\*หลังแก้ไขนี้ CORS และ CSRF จะทำงานถูกต้อง\*\*

ถ้ายัง error แนบ log backend เพิ่มเติมได้เลย!

pethzero: ตอนนี้ เหลือ patient-form.ts:36 POST http://localhost:3000/api/patients 500 (Internal Server Error) เกิดข้อผิดพลาด: invalid csrf token

ต้องแก้ไขยังไงในการใช้งาน

GitHub Copilot: ปัญหา \*\*500 (Internal Server Error): invalid csrf token\*\*

เกิดจาก \*\*CSRF token ที่แนบไปกับ request ไม่ถูกต้อง หรือไม่มี\*\*

(แม้ CORS จะผ่านแล้ว)

---

## วิธีแก้ไข (ทีละขั้น)

### 1. \*\*ลำดับการขอ CSRF token\*\*

1. \*\*ก่อน submit ฟอร์ม\*\* ต้องเรียก `GET /api/auth/csrf-token`

- จะได้ทั้ง cookie (เช่น `csrfSecret`) และ response `{ csrfToken: "xxxx" }`

2. \*\*เก็บ token นี้ไว้ใน AuthService\*\*

3. \*\*ทุก POST/PUT/DELETE\*\*

- แนบ header `X-CSRF-Token: xxxx` (ใช้ Interceptor)

- แนบ cookie (Angular: `{ withCredentials: true }`)

---

### 2. \*\*ตรวจสอบ Interceptor ฝั่ง Angular\*\*

ในไฟล์ `csrf-interceptor.ts`

ต้องแน่ใจว่าแนบ token และเปิด withCredentials ทุก request ที่เปลี่ยนข้อมูล

```typescript

export const csrfInterceptor: HttpInterceptorFn = (req, next) => {

const authService = inject(AuthService);

const csrfToken = authService.getCsrfToken();

let headers = req.headers;

if (csrfToken && ['POST', 'PUT', 'DELETE'].includes(req.method)) {

headers = headers.set('X-CSRF-Token', csrfToken);

}

const cloned = req.clone({

headers,

withCredentials: true, // ต้องเปิดให้ browser ส่ง cookie

});

return next(cloned);

};

```

---

### 3. \*\*ตรวจสอบการเรียก fetchCsrfToken()\*\*

\*\*ก่อน\*\* submit ฟอร์ม (หรือหลัง login สำเร็จ)

\*\*ต้องเรียก\*\* `authService.fetchCsrfToken().subscribe();`

เพื่อให้ AuthService เก็บ token ล่าสุด

ตัวอย่างใน `ngOnInit()` ของฟอร์ม:

```typescript

ngOnInit(): void {

this.auth.fetchCsrfToken().subscribe();

}

```

---

### 4. \*\*ตรวจสอบ Cookie\*\*

- หลังเรียก `/api/auth/csrf-token` แล้ว

- ดูใน devtools > Application > Cookies ว่ามี cookie ที่ backend set มาหรือไม่

- ทุก request POST/PUT/DELETE ต้องส่ง cookie นี้กลับไป (Angular: `{ withCredentials: true }`)

---

### 5. \*\*สรุป flow ที่ถูกต้อง\*\*

1. \*\*login\*\* (หรือ guest) → ได้ cookie JWT

2. \*\*fetch CSRF token\*\* (`GET /api/auth/csrf-token`) → ได้ cookie + token

3. \*\*submit ฟอร์ม\*\* (POST/PUT/DELETE) → แนบ header `X-CSRF-Token` + cookie

---

## ตัวอย่างการใช้งานในฟอร์ม

```typescript

@Component({

// ...

})

export class PatientFormComponent implements OnInit {

// ...

constructor(private fb: FormBuilder, private patientService: PatientService, private auth: AuthService) {

// ...

}

ngOnInit(): void {

this.auth.fetchCsrfToken().subscribe();

}

submit() {

if (this.patientForm.invalid) return;

const data: Patient = this.patientForm.value;

this.patientService.createPatient(data).subscribe({

next: () => {

this.successMessage = 'เพิ่มคนไข้สำเร็จ!';

this.errorMessage = '';

this.patientForm.reset();

},

error: (err) => {

this.errorMessage = 'เกิดข้อผิดพลาด: ' + (err.error?.message || err.message);

this.successMessage = '';

}

});

}

}

```

---

## \*\*สรุป\*\*

- ต้องเรียก `fetchCsrfToken()` ก่อน submit ฟอร์มทุกครั้ง (หรือหลัง login)

- Interceptor ต้องแนบ `X-CSRF-Token` และ `{ withCredentials: true }`

- ตรวจสอบว่า cookie ถูกส่งไปกับ request จริง

\*\*ถ้าทำครบนี้แล้ว ยัง error แนบ log network (request/response headers) มาให้ดูเพิ่มได้เลย!\*\*

pethzero: สุดยอดอได้แล้วแต่ว่าเราอยากจะ ปรับปรุง layout อยากให้ css แบบ https://primeng.org/

GitHub Copilot: ดีใจด้วยที่ระบบใช้งานได้แล้ว!

ถ้าคุณอยาก \*\*ปรับปรุง layout และ CSS ให้ดูสวยแบบ [PrimeNG](https://primeng.org/)\*\*

สามารถทำได้ 2 แนวทางหลัก:

---

## แนวทางที่ 1: \*\*ใช้ PrimeNG จริงในโปรเจกต์\*\*

1. \*\*ติดตั้ง PrimeNG และ PrimeIcons\*\*

```sh

npm install primeng primeicons

```

2. \*\*import CSS ของ PrimeNG ใน `styles.scss`\*\*

```scss

@import 'primeng/resources/themes/lara-light-blue/theme.css'; // หรือ theme อื่น

@import 'primeng/resources/primeng.min.css';

@import 'primeicons/primeicons.css';

```

3. \*\*เปลี่ยน component layout ให้ใช้ของ PrimeNG\*\*

- เช่น `<p-sidebar>`, `<p-menubar>`, `<p-button>`, `<p-card>`, `<p-inputText>` ฯลฯ

- ตัวอย่าง sidebar:

```html

<p-sidebar [(visible)]="visibleSidebar">

<h1>เมนู</h1>

<ul>

<li><a routerLink="/dashboard">Dashboard</a></li>

<li><a routerLink="/patient">Patient</a></li>

</ul>

</p-sidebar>

```

4. \*\*ดูตัวอย่างและโค้ดเพิ่มเติมที่ [PrimeNG Docs](https://primeng.org/components/menubar)\*\*

- คุณสามารถ mix PrimeNG กับ Angular Material ได้ (แต่ควรเลือกใช้หลักๆ แค่ 1 library)

---

## แนวทางที่ 2: \*\*ปรับ CSS เดิมให้คล้าย PrimeNG\*\*

- ดูตัวอย่าง CSS ของ PrimeNG (เช่น theme `lara-light-blue`)

- ปรับสี, border-radius, shadow, spacing, font ให้คล้าย PrimeNG

- ใช้ utility class เช่น `.p-card`, `.p-button`, `.p-inputtext` (หรือสร้างเอง)

- ตัวอย่างปรับปุ่ม:

```scss

.btn {

background: #2196f3;

color: #fff;

border-radius: 6px;

border: none;

padding: 0.5rem 1.5rem;

font-weight: 500;

box-shadow: 0 2px 8px rgba(33,150,243,0.08);

transition: background 0.2s;

&:hover {

background: #1769aa;

}

}

```

---

## \*\*แนะนำ\*\*

- ถ้าอยากได้ความสวยและ component หลากหลายแบบ PrimeNG จริง \*\*ควรติดตั้ง PrimeNG\*\*

- ถ้าอยากปรับแค่ theme ให้คล้าย PrimeNG โดยไม่เพิ่ม dependency ก็ปรับ CSS เดิมตามตัวอย่างข้างบน

---

\*\*ถ้าต้องการตัวอย่างโค้ด PrimeNG sidebar, menubar, หรือฟอร์ม แจ้งได้เลย!\*\*

หรือถ้าต้องการแนะนำ theme PrimeNG ที่เหมาะกับงานโรงพยาบาล/คลินิก แจ้งได้เช่นกันครับ!

pethzero: แค่อยากให้ปรับ css คล้ายๆ สีเรียบหรู ระดับ โลก ที่ทำัน

GitHub Copilot: แนวทางปรับ CSS layout ของคุณให้ \*\*เรียบหรู สไตล์โลกสมัยใหม่\*\* (inspired by PrimeNG, Material, และ SaaS dashboard ระดับโลก):

---

## 1. \*\*สีหลัก (Primary Color Palette)\*\*

- ใช้โทนฟ้า-กรมท่า-เทา-ขาว (เรียบหรู, สบายตา, เหมาะกับงานสุขภาพ)

- ตัวอย่าง palette:

- Primary: `#1976d2` (ฟ้าเข้ม)

- Accent: `#42a5f5` (ฟ้าอ่อน)

- Background: `#f5f7fa` (เทาอ่อน)

- Surface: `#fff` (ขาว)

- Text: `#1e293b` (เทาเข้ม)

- Border: `#e0e0e0` (เทาอ่อน)

---

## 2. \*\*ตัวอย่างปรับ sidebar.scss\*\*

````scss

.sidebar-list {

background: #fff;

border-right: 1px solid #e0e0e0;

min-height: 100vh;

padding: 1.5rem 0 1rem 0;

box-shadow: 2px 0 8px 0 rgba(25, 118, 210, 0.04);

}

.menu-name {

font-size: 1.6rem;

font-weight: 900;

text-align: center;

color: #1976d2;

background: linear-gradient(90deg, #e3f2fd 0%, #fff 100%);

border-radius: 16px;

margin: 0 1rem 1.5rem 1rem;

padding: 1rem 0;

letter-spacing: 1px;

text-transform: uppercase;

box-shadow: 0 2px 8px rgba(25, 118, 210, 0.06);

}

.menu-head {

font-size: 0.9rem;

font-weight: 600;

color: #90a4ae;

padding: 0.75rem 1.5rem 0.25rem;

background: transparent;

}

.detail-item {

display: flex;

align-items: center;

font-size: 1rem;

color: #1e293b;

padding: 0.7rem 1.5rem;

border-left: 3px solid transparent;

border-radius: 8px 0 0 8px;

background: transparent;

transition: all 0.18s cubic-bezier(.4,0,.2,1);

.label-text {

white-space: nowrap;

font-weight: 500;

}

&:hover, &.active-link {

background: linear-gradient(90deg, #e3f2fd 0%, #fff 100%);

border-left-color: #1976d2;

color: #1976d2;

font-weight: 600;

box-shadow: 0 2px 8px rgba(25, 118, 210, 0.04);

}

}

mat-divider {

margin: 0.5rem 0;

background: #e0e0e0;

}

mat-expansion-panel {

background: #fff;

box-shadow: none;

border-left: 2px solid transparent;

mat-expansion-panel-header {

padding: 0.25rem 1.5rem;

font-weight: 500;

font-size: 1rem;

color: #1976d2;

background: #f5f7fa;

border-radius: 8px;

}

mat-expansion-panel-header.mat-expanded {

background: #e3f2fd;

border-left-color: #42a5f5;

}

}

mat-icon {

font-size: 1.2rem;

margin-right: 0.5rem;

color: #1976d2;

}

````

---

## 3. \*\*ตัวอย่างปรับ header.scss\*\*

````scss

.header-toolbar {

background: linear-gradient(90deg, #1976d2 0%, #42a5f5 100%);

color: #fff;

height: 60px;

box-shadow: 0 2px 8px rgba(25, 118, 210, 0.08);

.header-title {

font-size: 1.3rem;

font-weight: 700;

letter-spacing: 1px;

color: #fff;

}

}

````

---

## 4. \*\*ตัวอย่างปรับ full-layout.scss\*\*

````scss

.sidenav-container {

height: 100vh;

background: #f5f7fa;

}

.sidenav {

width: 250px;

background: #fff;

box-shadow: 2px 0 8px 0 rgba(25, 118, 210, 0.04);

}

.sidenav-content {

background: #f5f7fa;

}

main.content {

padding: 32px 24px 24px 24px;

background: #f5f7fa;

min-height: calc(100vh - 60px);

border-radius: 16px;

box-shadow: 0 2px 8px rgba(25, 118, 210, 0.03);

}

````

---

## 5. \*\*Tips เพิ่มเติม\*\*

- ใช้ \*\*box-shadow\*\* เบาๆ, \*\*border-radius\*\* 8-16px, \*\*gradient\*\* อ่อนๆ

- ใช้ \*\*font-weight\*\* 600-900 สำหรับหัวข้อ

- ใช้ \*\*transition\*\* กับ hover ทุกปุ่ม/เมนู

- ใช้ \*\*mat-icon\*\* สี primary (`#1976d2`) หรือ accent (`#42a5f5`)

- พื้นหลังเน้นขาว/เทาอ่อน, ข้อความเข้ม

---

## 6. \*\*ตัวอย่างธีม PrimeNG ที่แนะนำ\*\*

- Lara Light Blue (`@import 'primeng/resources/themes/lara-light-blue/theme.css';`)

- Saga Blue

- Arya Blue

---

\*\*สรุป:\*\*

- ปรับสีหลักเป็นฟ้า-ขาว-เทา

- ใช้ gradient, shadow, border-radius, font-weight

- ดูตัวอย่างข้างบนแล้วปรับในไฟล์ sidebar.scss, header.scss, full-layout.scss ได้เลย

ถ้าอยากได้ตัวอย่าง sidebar หรือ header แบบ PrimeNG จริง (HTML+CSS) เพิ่มเติม แจ้งได้เลย!

pethzero: ทำไฟล์ อนิเมชะ้น loading ให้หน่้อยได้ไหม

GitHub Copilot: ตัวอย่าง \*\*ไฟล์ Animation Loading (Angular + SCSS)\*\*

สามารถนำไปใช้เป็น component loading ทั่วไปในโปรเจกต์ของคุณได้เลย

---

### 1. สร้างไฟล์ `src/app/ui/loading-spinner/loading-spinner.ts`

````typescript

import { Component, Input } from '@angular/core';

@Component({

selector: 'app-loading-spinner',

standalone: true,

templateUrl: './loading-spinner.html',

styleUrls: ['./loading-spinner.scss']

})

export class LoadingSpinnerComponent {

@Input() size = 48; // px

@Input() color = '#1976d2';

}

````

---

### 2. สร้างไฟล์ `src/app/ui/loading-spinner/loading-spinner.html`

````html

<div class="spinner-container">

<div class="spinner" [ngStyle]="{'width.px': size, 'height.px': size, 'border-top-color': color}"></div>

</div>

````

---

### 3. สร้างไฟล์ `src/app/ui/loading-spinner/loading-spinner.scss`

````scss

.spinner-container {

display: flex;

align-items: center;

justify-content: center;

min-height: 64px;

}

.spinner {

border: 4px solid #e0e0e0;

border-top: 4px solid #1976d2;

border-radius: 50%;

width: 48px;

height: 48px;

animation: spin 0.8s linear infinite;

transition: border-top-color 0.2s;

}

@keyframes spin {

0% { transform: rotate(0deg);}

100% { transform: rotate(360deg);}

}

````

---

### 4. วิธีใช้งานในหน้าอื่น

```html

<app-loading-spinner [size]="64" color="#42a5f5"></app-loading-spinner>

```

หรือแบบ default

```html

<app-loading-spinner></app-loading-spinner>

```

---

\*\*สรุป:\*\*

- สร้าง 3 ไฟล์ใน `src/app/ui/loading-spinner/`

- ใช้ `<app-loading-spinner>` ได้ทุกที่ในโปรเจกต์

- ปรับขนาด/สีได้ด้วย `[size]` และ `[color]`

pethzero:

X [ERROR] NG8002: Can't bind to 'ngStyle' since it isn't a known property of 'div'. [plugin angular-compiler]

src/app/ui/loading-spinner/loading-spinner.html:2:23:

2 │ ...="spinner" [ngStyle]="{'width.px': size, 'height.px': size, 'bor...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component LoadingSpinnerComponent.

src/app/ui/loading-spinner/loading-spinner.ts:6:15:

6 │ templateUrl: './loading-spinner.html',

╵ ~~~~~~~~~~~~~~~~~~~~~~~~

GitHub Copilot: ปัญหานี้เกิดจาก \*\*ยังไม่ได้ import CommonModule\*\* ใน component แบบ standalone

ซึ่ง CommonModule จะรวม directive อย่าง `\*ngIf`, `\*ngFor`, `ngStyle`, `ngClass` ฯลฯ

---

\*\*วิธีแก้ไข:\*\*

เพิ่ม `CommonModule` ใน `imports` ของ `@Component` ใน loading-spinner.ts

````typescript

import { Component, Input } from '@angular/core';

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-loading-spinner',

standalone: true,

templateUrl: './loading-spinner.html',

styleUrls: ['./loading-spinner.scss'],

imports: [CommonModule] // ✅ เพิ่มตรงนี้

})

export class LoadingSpinnerComponent {

@Input() size = 48; // px

@Input() color = '#1976d2';

}

````

\*\*หลังแก้ไขนี้ error `Can't bind to 'ngStyle'` จะหายไป\*\*

pethzero: อยากให้ card box มันสวยๆ พื้นหลัง สีพาเทล เทาๆ แบบ ดูดี

GitHub Copilot: ตัวอย่างการปรับ \*\*card box\*\* ใน dashboard ให้ดูเรียบหรู พื้นหลังสีพาเทลเทาๆ สไตล์โลกสมัยใหม่

(ใช้ร่วมกับ Angular Material ได้เลย)

---

### 1. ปรับ SCSS ของ card ใน dashboard.scss

````scss

.dashboard-container {

background-color: #f3f4f6; // เทาอ่อน

min-height: 100vh;

padding: 1.5rem;

}

mat-card {

background: linear-gradient(135deg, #f7fafc 0%, #e3e9f3 100%);

box-shadow: 0 4px 24px 0 rgba(25, 118, 210, 0.07), 0 1.5px 4px 0 rgba(0,0,0,0.03);

border-radius: 1.25rem; // 20px

padding: 1.5rem 1.25rem;

transition: box-shadow 0.2s, transform 0.2s;

border: 1px solid #e0e7ef;

&:hover {

box-shadow: 0 8px 32px 0 rgba(25, 118, 210, 0.12), 0 2px 8px 0 rgba(0,0,0,0.06);

transform: translateY(-2px) scale(1.02);

}

}

mat-card-title {

color: #374151;

font-size: 1.1rem;

font-weight: 700;

margin-bottom: 0.5rem;

letter-spacing: 0.5px;

}

mat-card-content p {

margin: 0;

font-size: 2.2rem;

font-weight: 800;

color: #1976d2;

letter-spacing: 1px;

}

@media (max-width: 640px) {

mat-card {

padding: 1rem 0.5rem;

border-radius: 1rem;

}

}

````

---

### 2. ปรับ HTML (optional, เพิ่มคลาสหรือ icon ได้)

```html

<!-- ...ใน dashboard.html... -->

<mat-card class="stat-card">

<mat-card-title>

<mat-icon style="vertical-align: middle; color: #90caf9; margin-right: 8px;">groups</mat-icon>

คนไข้ทั้งหมด

</mat-card-title>

<mat-card-content>

<p class="stat-number text-blue-600">{{ stats.totalPatients }}</p>

</mat-card-content>

</mat-card>

```

\*หมายเหตุ: เพิ่ม `<mat-icon>` ได้ถ้าอยากให้ดู modern ขึ้น\*

---

### 3. ตัวอย่างสีพาเทลอื่นๆ (เลือกใช้ได้)

- ฟ้าอ่อน: `#e3e9f3`

- เทาอ่อน: `#f7fafc`

- ม่วงพาเทล: `#ede7f6`

- เขียวพาเทล: `#e3fcec`

- เหลืองพาเทล: `#fffde7`

---

### 4. ผลลัพธ์

- Card ดูนุ่มนวล มีมิติ พื้นหลังพาเทล

- Hover แล้วลอยขึ้นเล็กน้อย

- Font หนา ดู modern

- ใช้ร่วมกับ Material ได้เลย

---

\*\*ถ้าอยากได้ตัวอย่างสีหรือ gradient เพิ่มเติม แจ้งได้เลย!\*\*

pethzero:

X [ERROR] NG8001: 'mat-icon' is not a known element:

1. If 'mat-icon' is an Angular component, then verify that it is included in the '@Component.imports' of this component.

2. If 'mat-icon' is a Web Component then add 'CUSTOM\_ELEMENTS\_SCHEMA' to the '@Component.schemas' of this component to suppress this message. [plugin angular-compiler]

src/app/pages/dashboard/dashboard.html:7:8:

7 │ <mat-icon style="vertical-align: middle; color: #90caf9; ma...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component DashboardComponent.

src/app/pages/dashboard/dashboard.ts:22:15:

22 │ templateUrl: './dashboard.html',

╵ ~~~~~~~~~~~~~~~~~~

GitHub Copilot: ปัญหา

- Error: `'mat-icon' is not a known element`

- เพราะคุณใช้ `<mat-icon>` แต่ยัง \*\*ไม่ได้ import MatIconModule\*\* ใน `@Component.imports` (ถ้าใช้ standalone) หรือใน module

---

## วิธีแก้ไข (แบบ Standalone Component)

\*\*เพิ่ม MatIconModule ใน imports ของ DashboardComponent\*\*

````typescript

import { Component } from '@angular/core';

import { MatCardModule } from '@angular/material/card';

import { MatIconModule } from '@angular/material/icon'; // เพิ่มบรรทัดนี้

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-dashboard',

standalone: true,

templateUrl: './dashboard.html',

styleUrls: ['./dashboard.scss'],

imports: [

CommonModule,

MatCardModule,

MatIconModule, // เพิ่มตรงนี้

// ...module อื่นๆ ที่ใช้ใน template

]

})

export class DashboardComponent { /\* ... \*/ }

````

---

## หมายเหตุ

- ถ้าใช้ใน module-based (NgModule) ให้ import `MatIconModule` ใน `imports` ของ module นั้นด้วย

---

\*\*หลังเพิ่ม MatIconModule แล้ว error จะหาย และไอคอนจะแสดงผลปกติ\*\*

pethzero: ปรับหน้านี้รวมทั้ง table เราใช้ tailwind นะ

GitHub Copilot: ตัวอย่างการ \*\*ปรับหน้า dashboard ทั้งหมด (รวม table)\*\* ให้ดูเรียบหรู สไตล์โลกสมัยใหม่ ด้วย Tailwind CSS

(เน้นสีพาเทล, card มีมิติ, table สวยอ่านง่าย)

---

````html

<div class="dashboard-container min-h-screen p-6 bg-gray-100">

<!-- สรุปสถิติ -->

<div class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-2 lg:grid-cols-4 gap-6">

<!-- การ์ด: คนไข้ทั้งหมด -->

<div class="rounded-2xl bg-gradient-to-br from-white via-blue-50 to-gray-100 shadow-lg p-6 flex flex-col items-start hover:shadow-xl transition">

<div class="flex items-center mb-2">

<span class="inline-flex items-center justify-center w-10 h-10 rounded-full bg-blue-100 mr-3">

<span class="material-icons text-blue-500 text-2xl">groups</span>

</span>

<span class="text-lg font-bold text-gray-700">คนไข้ทั้งหมด</span>

</div>

<div class="text-3xl font-extrabold text-blue-600">{{ stats.totalPatients }}</div>

</div>

<!-- การ์ด: คนไข้ใหม่วันนี้ -->

<div class="rounded-2xl bg-gradient-to-br from-white via-green-50 to-gray-100 shadow-lg p-6 flex flex-col items-start hover:shadow-xl transition">

<div class="flex items-center mb-2">

<span class="inline-flex items-center justify-center w-10 h-10 rounded-full bg-green-100 mr-3">

<span class="material-icons text-green-500 text-2xl">person\_add</span>

</span>

<span class="text-lg font-bold text-gray-700">คนไข้ใหม่วันนี้</span>

</div>

<div class="text-3xl font-extrabold text-green-600">{{ stats.newPatientsToday }}</div>

</div>

<!-- การ์ด: นัดหมายวันนี้ -->

<div class="rounded-2xl bg-gradient-to-br from-white via-purple-50 to-gray-100 shadow-lg p-6 flex flex-col items-start hover:shadow-xl transition">

<div class="flex items-center mb-2">

<span class="inline-flex items-center justify-center w-10 h-10 rounded-full bg-purple-100 mr-3">

<span class="material-icons text-purple-500 text-2xl">event</span>

</span>

<span class="text-lg font-bold text-gray-700">นัดหมายวันนี้</span>

</div>

<div class="text-3xl font-extrabold text-purple-600">{{ stats.appointmentsToday }}</div>

</div>

<!-- การ์ด: อยู่ในห้องตรวจ -->

<div class="rounded-2xl bg-gradient-to-br from-white via-red-50 to-gray-100 shadow-lg p-6 flex flex-col items-start hover:shadow-xl transition">

<div class="flex items-center mb-2">

<span class="inline-flex items-center justify-center w-10 h-10 rounded-full bg-red-100 mr-3">

<span class="material-icons text-red-500 text-2xl">local\_hospital</span>

</span>

<span class="text-lg font-bold text-gray-700">อยู่ในห้องตรวจ</span>

</div>

<div class="text-3xl font-extrabold text-red-600">{{ stats.inExamRoom }}</div>

</div>

</div>

<!-- กราฟ -->

<div class="charts-container grid grid-cols-1 sm:grid-cols-2 gap-6 mt-8">

<div class="rounded-2xl bg-white shadow-lg p-6">

<div class="text-gray-800 text-lg font-semibold mb-4">สัดส่วนเพศคนไข้</div>

<canvas baseChart [data]="chartData" [options]="chartOptions" [type]="chartType"

class="dashboard-chart"></canvas>

</div>

<div class="rounded-2xl bg-white shadow-lg p-6">

<div class="text-gray-800 text-lg font-semibold mb-4">ยอดผู้ป่วยรายเดือน</div>

<canvas baseChart [data]="chartData2" [options]="chartOptions2" [type]="chartType2"

class="dashboard-chart"></canvas>

</div>

</div>

<!-- ตารางรายชื่อคนไข้ -->

<div class="mt-10">

<div class="bg-white rounded-2xl shadow-lg overflow-x-auto">

<table class="min-w-full divide-y divide-gray-200">

<thead class="bg-gray-50">

<tr>

<th class="px-6 py-3 text-left text-xs font-bold text-gray-500 uppercase tracking-wider">รหัส</th>

<th class="px-6 py-3 text-left text-xs font-bold text-gray-500 uppercase tracking-wider">ชื่อ</th>

<th class="px-6 py-3 text-left text-xs font-bold text-gray-500 uppercase tracking-wider">นามสกุล</th>

<th class="px-6 py-3 text-left text-xs font-bold text-gray-500 uppercase tracking-wider">เพศ</th>

<th class="px-6 py-3 text-left text-xs font-bold text-gray-500 uppercase tracking-wider">วันเกิด</th>

<th class="px-6 py-3 text-left text-xs font-bold text-gray-500 uppercase tracking-wider">เบอร์โทร</th>

<th class="px-6 py-3"></th>

</tr>

</thead>

<tbody class="bg-white divide-y divide-gray-100">

<tr \*ngFor="let patient of patients">

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-700">{{ patient.patient\_code }}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-700">{{ patient.first\_name }}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-700">{{ patient.last\_name }}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-700">{{ patient.gender }}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-700">{{ patient.date\_of\_birth | date }}</td>

<td class="px-6 py-4 whitespace-nowrap text-sm text-gray-700">{{ patient.phone }}</td>

<td class="px-6 py-4 whitespace-nowrap text-right text-sm">

<button class="inline-flex items-center px-3 py-1.5 bg-blue-50 text-blue-700 rounded-lg hover:bg-blue-100 transition">

<span class="material-icons text-base mr-1">edit</span> แก้ไข

</button>

</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

````

---

\*\*หมายเหตุ:\*\*

- ใช้ `<span class="material-icons">...</span>` สำหรับไอคอน (Google Fonts: Material Icons)

- ถ้ายังไม่ได้เพิ่ม Material Icons ให้เพิ่มใน `index.html`:

```html

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

```

- ปรับสีและขนาดได้ตามต้องการ

---

\*\*ผลลัพธ์:\*\*

- Card, กราฟ, ตาราง ดูเรียบหรู พื้นหลังพาเทล

- Responsive, อ่านง่าย, สวยแบบ SaaS dashboard

ถ้าอยากปรับเพิ่มหรือมี section อื่น แจ้งได้เลย!

Similar code found with 1 license type

pethzero: Application bundle generation failed. [0.101 seconds]

▲ [WARNING] TS-998113: PatientList is not used within the template of DashboardComponent [plugin angular-compiler]

src/app/pages/dashboard/dashboard.ts:22:61:

22 │ ..., MatCardModule, BaseChartDirective, PatientList, MatIconModule,],

╵ ~~~~~~~~~~~

X [ERROR] NG9: Property 'patients' does not exist on type 'DashboardComponent'. [plugin angular-compiler]

src/app/pages/dashboard/dashboard.html:76:37:

76 │ <tr \*ngFor="let patient of patients">

╵ ~~~~~~~~

Error occurs in the template of component DashboardComponent.

src/app/pages/dashboard/dashboard.ts:23:15:

23 │ templateUrl: './dashboard.html',

╵ ~~~~~~~~~~~~~~~~~~

GitHub Copilot: ### ปัญหา

- `[ERROR] NG9: Property 'patients' does not exist on type 'DashboardComponent'.`

- เพราะใน template มี `\*ngFor="let patient of patients"` แต่ใน `DashboardComponent` \*\*ยังไม่มี property `patients`\*\*

---

### วิธีแก้ไข

\*\*เพิ่ม property `patients` ใน DashboardComponent\*\*

````typescript

import { Component } from '@angular/core';

// ...imports...

@Component({

selector: 'app-dashboard',

standalone: true,

templateUrl: './dashboard.html',

styleUrls: ['./dashboard.scss'],

imports: [

// ...module อื่นๆ...

]

})

export class DashboardComponent {

// ...properties อื่นๆ...

patients = [

{

patient\_code: 'P001',

first\_name: 'สมชาย',

last\_name: 'ใจดี',

gender: 'ชาย',

date\_of\_birth: new Date(1990, 1, 1),

phone: '0812345678'

},

{

patient\_code: 'P002',

first\_name: 'สมหญิง',

last\_name: 'ใจงาม',

gender: 'หญิง',

date\_of\_birth: new Date(1992, 5, 10),

phone: '0898765432'

}

// ...เพิ่ม mock data หรือดึงจาก service ก็ได้...

];

// ...methods อื่นๆ...

}

````

---

\*\*ถ้าคุณจะดึงข้อมูลจริงจาก service\*\*

ให้ประกาศ `patients: Patient[] = [];`

แล้วไปโหลดข้อมูลใน `ngOnInit()` เช่น

```typescript

ngOnInit() {

this.patientService.getAll().subscribe(data => this.patients = data);

}

```

---

\*\*สรุป:\*\*

- ต้องมี property `patients` ใน DashboardComponent

- จะเป็น mock data หรือดึงจาก service ก็ได้

- Error นี้จะหายทันทีเมื่อเพิ่ม property นี้

ถ้าต้องการตัวอย่างการดึงข้อมูลจาก API จริง แจ้งได้เลย!

pethzero: ไม่ใช่ คือ

GitHub Copilot: ขออธิบายและแนะนำแนวทางที่ถูกต้องสำหรับการ \*\*แสดงตารางรายชื่อคนไข้ในหน้า Dashboard\*\* โดยใช้ Tailwind (ไม่ใช้ Angular Material Table) และ \*\*แยกตารางจริงไว้ที่ PatientList\*\*:

---

### 1. \*\*ถ้าอยากให้ Dashboard เรียกใช้ตารางจาก PatientList\*\*

\*\*ใน dashboard.html\*\*

แทนที่จะเขียน `<table>` เอง ให้ใช้ component ที่สร้างไว้แล้วแบบนี้:

```html

<!-- ...ส่วนอื่นๆ ของ dashboard... -->

<!-- ตารางรายชื่อคนไข้ -->

<div class="mt-10">

<app-patient-list></app-patient-list>

</div>

```

\*\*ใน dashboard

pethzero: คิดว่า ขาด folder แบบไหน struce ที่ระดับ มาตราฐานมีกัน

GitHub Copilot: จากโครงสร้างปัจจุบันของคุณ \*\*ถือว่าดีมากแล้ว\*\* สำหรับ Angular + NestJS ที่แยก `core`, `layout`, `pages`, `models`, `shared`, `ui`

แต่ถ้าอยากให้ \*\*ครบมาตรฐานระดับองค์กร/production\*\* (inspired by Angular style guide, Nx, และ enterprise project)

สามารถเพิ่ม/ปรับปรุงได้ดังนี้

---

## 1. \*\*แนะนำโฟลเดอร์ที่ควรมีเพิ่ม\*\*

### \*\*a. `shared/`\*\*

- สำหรับ component, pipe, directive, util ที่ใช้ซ้ำหลายที่ (generic, ไม่ขึ้นกับ business logic)

- ตัวอย่าง: `shared/pipes/`, `shared/directives/`, `shared/components/`

### \*\*b. `features/` หรือ `modules/`\*\*

- ถ้าโปรเจกต์ใหญ่ แยก business feature เป็น module เช่น `features/patient/`, `features/appointment/`

- จะช่วยให้ scale และ test ง่าย

### \*\*c. `assets/`\*\*

- สำหรับไฟล์ static เช่น รูป โลโก้ ไอคอน ฟอนต์ (แต่ปกติ Angular CLI จะมี `src/assets` อยู่แล้ว)

### \*\*d. `state/` หรือ `store/`\*\*

- ถ้าใช้ state management (เช่น NgRx, Akita, SignalStore)

- แยก logic store, action, reducer, selector ไว้ที่เดียว

### \*\*e. `types/` หรือ `interfaces/`\*\*

- สำหรับ type/interface กลางที่ใช้ข้ามหลายโมดูล (ถ้าไม่อยากปะปนกับ model/business)

### \*\*f. `services/` (ในแต่ละ feature)\*\*

- ปัจจุบันคุณมี `core/services/` แล้ว

- ถ้าแต่ละ feature มี service เฉพาะ ควรแยกไว้ใน feature นั้น

### \*\*g. `guards/`, `interceptors/`, `resolvers/` (ใน core หรือ shared)\*\*

- คุณมี `core/guards/` และ `core/interceptors/` แล้ว ดีมาก

### \*\*h. `environments/`\*\*

- คุณมีแล้ว

---

## 2. \*\*ตัวอย่างโครงสร้างที่แนะนำ\*\*

```

src/

app/

core/

guards/

interceptors/

services/

constants/

layout/

header/

sidebar/

full-layout/

shared/

components/

pipes/

directives/

utils/

features/

patient/

patient-list/

patient-form/

patient.service.ts

appointment/

...

models/

users/

stats/

state/ // (ถ้าใช้ state management)

ui/

button/

loading-spinner/

pages/

dashboard/

login/

not-found/

...

environments/

assets/

```

---

## 3. \*\*สิ่งที่คุณมีแล้วและดีมาก\*\*

- `core/` สำหรับ logic กลาง

- `layout/` สำหรับ UI หลัก

- `pages/` สำหรับ route/page

- `models/` สำหรับ business model

- `ui/` สำหรับ component UI เล็กๆ

- `shared/` (มีแล้วแต่ยังไม่เห็นรายละเอียด)

---

## 4. \*\*สรุป\*\*

- โครงสร้างคุณดีมากแล้วสำหรับ Angular app

- ถ้าอยาก scale หรือให้ทีมใหม่เข้าใจง่ายขึ้น

- เพิ่ม `shared/`, `features/`, `state/`, `types/` ตามข้างบน

- ถ้าใช้ Nx หรือ monorepo จะมี `libs/` สำหรับ library ที่ share ข้าม project

---

\*\*ถ้าต้องการตัวอย่างไฟล์ index.ts, barrel file, หรือแนวทางแบ่ง feature module เพิ่มเติม แจ้งได้เลย!\*\*

pethzero: เรามี asset ควรเอาไว้ ข้าวในไหท shared สร้างตัวอย่างได้ไหม

GitHub Copilot: \*\*คำตอบ:\*\*

ไฟล์ asset (เช่น รูป โลโก้ ฟอนต์ ไอคอน) \*\*ควรเก็บไว้ใน `src/assets/`\*\* ไม่ควรเก็บใน `shared/`

เพราะ `shared/` สำหรับ component, pipe, directive, หรือ util ที่ใช้ซ้ำในโค้ด Angular

แต่ \*\*ถ้าคุณอยากสร้าง shared component สำหรับแสดง asset\*\* (เช่น โลโก้บริษัท)

ให้สร้าง component ใน `shared/components/` แล้วอ้างอิง asset จาก `assets/`

---

### ตัวอย่าง: สร้าง Shared Logo Component

\*\*1. โฟลเดอร์\*\*

```

src/app/shared/components/logo/

logo.ts

logo.html

logo.scss

```

\*\*2. ไฟล์ logo.ts\*\*

````typescript

import { Component, Input } from '@angular/core';

@Component({

selector: 'app-logo',

standalone: true,

templateUrl: './logo.html',

styleUrls: ['./logo.scss']

})

export class LogoComponent {

@Input() size = 48; // px

}

````

\*\*3. ไฟล์ logo.html\*\*

````html

<img [src]="'assets/picture/logo.png'" [style.width.px]="size" [style.height.px]="size" alt="Logo" class="rounded-xl shadow" />

````

\*\*4. ไฟล์ logo.scss\*\*

````scss

/\* filepath: src/app/shared/components/logo/logo.scss \*/

img {

display: block;

object-fit: contain;

}

````

\*\*5. วิธีใช้ในหน้าอื่น\*\*

```html

<app-logo [size]="64"></app-logo>

```

---

\*\*สรุป:\*\*

- asset จริง (ไฟล์ภาพ ฯลฯ) อยู่ใน `src/assets/picture/`

- component ที่ใช้ซ้ำ (เช่น โลโก้) อยู่ใน `src/app/shared/components/`

- อ้างอิง asset ด้วย path `assets/picture/logo.png` ใน template

\*\*แบบนี้จะเป็นมาตรฐานและดูแลง่ายในระยะยาว\*\*

pethzero: state คือ และ ui ui สร้างแบบ มาตราฐานที่ ทุกคนชอบทำกันได้ไหม เราใช้ Angular Material UI

GitHub Copilot: ### คำอธิบาย

#### \*\*1. state คืออะไร?\*\*

- `state` (หรือ state management) คือการจัดการข้อมูลสถานะของแอป เช่น user login, รายการข้อมูล, loading, error ฯลฯ

- ใน Angular ถ้าโปรเจกต์ใหญ่ นิยมใช้ state management เช่น \*\*NgRx\*\*, \*\*SignalStore\*\*, \*\*Akita\*\*, หรือ \*\*Component Store\*\*

- โฟลเดอร์ `state/` หรือ `store/` จะเก็บ logic ที่เกี่ยวกับ state เช่น actions, reducers, selectors, stores

\*\*ตัวอย่างโครงสร้าง state (NgRx):\*\*

```

src/app/state/

patient/

patient.actions.ts

patient.reducer.ts

patient.effects.ts

patient.selectors.ts

patient.models.ts

app.state.ts

```

หรือถ้าใช้ SignalStore/ComponentStore ก็จะมี store class ในนี้

---

#### \*\*2. ui/ui คืออะไร?\*\*

- โฟลเดอร์ `ui/` สำหรับ \*\*UI component เล็กๆ ที่ใช้ซ้ำ\*\* (เช่น ปุ่ม, spinner, logo, modal, badge ฯลฯ)

- เป็น component ที่ "dumb" (stateless, ไม่ผูก business logic) และ \*\*ควรใช้ Angular Material\*\* หรือ custom style ได้

- ทุกคนในทีมจะเข้าใจตรงกันว่า ถ้าอยากใช้ปุ่ม/loader/alert ที่หน้าตาเหมือนกัน ให้ import จาก `ui/`

\*\*ตัวอย่างโครงสร้าง ui:\*\*

```

src/app/ui/

button/

button.ts

button.html

button.scss

loading-spinner/

loading-spinner.ts

loading-spinner.html

loading-spinner.scss

logo/

logo.ts

logo.html

logo.scss

...

```

- แต่ละ component เป็น standalone (Angular 16+), import CommonModule, Material module ที่จำเป็น

---

#### \*\*3. ตัวอย่าง ui component มาตรฐาน (Angular Material)\*\*

\*\*ปุ่ม (Button)\*\*

````typescript

import { Component, Input } from '@angular/core';

import { MatButtonModule } from '@angular/material/button';

@Component({

selector: 'app-button',

standalone: true,

imports: [MatButtonModule],

template: `<button mat-raised-button [color]="color"><ng-content></ng-content></button>`

})

export class ButtonComponent {

@Input() color: 'primary' | 'accent' | 'warn' = 'primary';

}

````

\*\*โลโก้ (Logo)\*\*

```html

-->

<img [src]="'assets/picture/logo.png'" [style.width.px]="size" [style.height.px]="size" alt="Logo" class="rounded-xl shadow" />

```

\*\*โหลดดิ้ง (Loading Spinner)\*\*

ดูตัวอย่างใน # (ของคุณดีอยู่แล้ว)

---

#### \*\*4. สรุปแนวทางมาตรฐาน\*\*

- `state/` สำหรับ logic จัดการข้อมูล (NgRx, SignalStore, ฯลฯ)

- `ui/` สำหรับ component UI เล็กๆ ที่ใช้ซ้ำ (stateless, ใช้ Angular Material ได้)

- `shared/` สำหรับ pipe, directive, หรือ component ที่ generic มากๆ

- ทุก component ใน `ui/` ควร standalone, import Material module ที่จำเป็น, และไม่ผูก business logic

---

\*\*แนวทางนี้เป็นมาตรฐานที่ทีมใหญ่และองค์กรนิยมใช้กัน\*\*

ถ้าต้องการตัวอย่างโครงสร้างหรือ component เพิ่มเติม แจ้งได้เลย!

Similar code found with 1 license type

pethzero: src/app/ui/

button/

button.ts

button.html

button.scss

button คิดแบบดีๆ ให้หน่อยสิ

GitHub Copilot: ตัวอย่าง \*\*ปุ่ม (Button) แบบมาตรฐานระดับองค์กร\*\* สำหรับ Angular + Angular Material

- รองรับ type, color, loading, disabled, icon, fullWidth

- ใช้ใน `src/app/ui/button/`

---

### 1. `button.ts`

````typescript

import { Component, Input } from '@angular/core';

import { MatButtonModule } from '@angular/material/button';

import { MatIconModule } from '@angular/material/icon';

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-button',

standalone: true,

imports: [CommonModule, MatButtonModule, MatIconModule],

templateUrl: './button.html',

styleUrls: ['./button.scss']

})

export class ButtonComponent {

@Input() type: 'button' | 'submit' | 'reset' = 'button';

@Input() color: 'primary' | 'accent' | 'warn' | undefined = 'primary';

@Input() disabled = false;

@Input() loading = false;

@Input() icon?: string; // ชื่อไอคอน material

@Input() fullWidth = false;

}

````

---

### 2. `button.html`

````html

<button

mat-raised-button

[color]="color"

[type]="type"

[disabled]="disabled || loading"

[class.full-width]="fullWidth"

>

<mat-icon \*ngIf="icon && !loading" class="mr-2">{{ icon }}</mat-icon>

<span \*ngIf="!loading"><ng-content></ng-content></span>

<mat-progress-spinner

\*ngIf="loading"

diameter="20"

mode="indeterminate"

color="primary"

class="loading-spinner"

></mat-progress-spinner>

</button>

````

---

### 3. `button.scss`

````scss

button.full-width {

width: 100%;

}

button {

font-weight: 600;

letter-spacing: 0.5px;

border-radius: 8px;

min-height: 40px;

min-width: 80px;

display: inline-flex;

align-items: center;

justify-content: center;

transition: box-shadow 0.18s;

}

mat-icon {

font-size: 20px;

margin-right: 0.5rem;

}

.loading-spinner {

margin-left: 0.5rem;

}

````

---

### 4. วิธีใช้งาน

```html

<app-button color="primary" icon="add">เพิ่มข้อมูล</app-button>

<app-button color="accent" [loading]="isSaving" fullWidth>บันทึก</app-button>

<app-button color="warn" icon="delete" [disabled]="true">ลบ</app-button>

```

---

\*\*ข้อดี:\*\*

- ใช้ซ้ำได้ทุกที่

- รองรับ loading, icon, fullWidth

- ใช้ Material Design

- ปรับแต่งง่าย

\*\*แนวทางนี้เป็นมาตรฐานที่ทีมใหญ่ใช้กัน\*\*

pethzero: อยากทำ ไฟล์หนึ่งแบบว่า ถ้าคลิก

{

path: 'dashboard',

loadComponent: () => import('./pages/dashboard/dashboard').then(m => m.DashboardComponent),

canActivate: [AuthGuard],

data: { roles: ['guest', 'user', 'admin'] } // ให้ guest, user, admin เข้าได้

},

path: 'dashboard',

จะมี ไฟล์ที่ เก็บ pagemenu แสดงที่ sidebar เราจะกำหนด แบบว่า /dashboard

pagemenu มี

{

link:aaa

name:aaa

ตัวลูก:[]

expen:true flase

}

menu ปรับเปรียบใหม่

GitHub Copilot: แนวทางที่ “ดีและยืดหยุ่น” สำหรับ Angular (และองค์กรนิยมใช้) คือ

\*\*แยกไฟล์ config สำหรับ page menu\*\* แล้ว import ไปใช้ใน SidebarComponent

เพื่อให้ sidebar เปลี่ยนเมนูตาม route หรือ role ได้ง่าย

---

### 1. สร้างไฟล์เมนู config

\*\*ตัวอย่าง: `src/app/core/constants/page-menu.ts`\*\*

````typescript

export interface PageMenuItem {

link: string;

name: string;

icon?: string;

children?: PageMenuItem[];

expen?: boolean;

roles?: string[]; // ถ้าต้องการ filter ตาม role

}

export const PAGE\_MENUS: Record<string, PageMenuItem[]> = {

'/dashboard': [

{

link: '/dashboard',

name: 'แดชบอร์ด',

icon: 'dashboard',

expen: false,

},

{

link: '/dashboard/overview',

name: 'ภาพรวม',

icon: 'insights',

expen: false,

},

{

link: '/dashboard/reports',

name: 'รายงาน',

icon: 'bar\_chart',

expen: true,

children: [

{ link: '/dashboard/reports/daily', name: 'รายวัน', icon: 'today' },

{ link: '/dashboard/reports/monthly', name: 'รายเดือน', icon: 'date\_range' }

]

}

],

'/patient': [

{

link: '/patient',

name: 'รายชื่อคนไข้',

icon: 'people',

expen: false,

},

{

link: '/patient/add',

name: 'เพิ่มคนไข้',

icon: 'person\_add',

expen: false,

}

]

// เพิ่มเมนูอื่นๆ ตาม path ได้

};

````

---

### 2. ปรับ SidebarComponent ให้อ่านเมนูจาก config

\*\*ใน SidebarComponent:\*\*

````typescript

import { PAGE\_MENUS, PageMenuItem } from '../../core/constants/page-menu';

// ...existing code...

export class SidebarComponent {

// ...existing code...

pageMenu: PageMenuItem[] = [];

ngOnInit() {

// สมมุติว่าคุณใช้ currentPath หรือ route ปัจจุบัน

const mainPath = this.currentPath.split('/')[1] ? '/' + this.currentPath.split('/')[1] : '/dashboard';

this.pageMenu = PAGE\_MENUS[mainPath] || [];

}

// ...existing code...

}

````

---

### 3. ใช้ pageMenu ใน template

\*\*ใน sidebar.html:\*\*

```html

<!-- ... -->

<ng-container \*ngFor="let item of pageMenu">

<a [routerLink]="item.link" class="sidebar-link" \*ngIf="!item.children">

<mat-icon>{{ item.icon }}</mat-icon>

<span>{{ item.name }}</span>

</a>

<mat-expansion-panel \*ngIf="item.children">

<mat-expansion-panel-header>

<mat-icon>{{ item.icon }}</mat-icon>

<span>{{ item.name }}</span>

</mat-expansion-panel-header>

<ng-container \*ngFor="let child of item.children">

<a [routerLink]="child.link" class="sidebar-link">

<mat-icon>{{ child.icon }}</mat-icon>

<span>{{ child.name }}</span>

</a>

</ng-container>

</mat-expansion-panel>

</ng-container>

<!-- ... -->

```

---

### 4. ข้อดีของแนวทางนี้

- \*\*แยก logic menu ออกจาก component\*\* (แก้ไขง่าย, scale ได้)

- \*\*รองรับลูกเมนู (children)\*\* และ expen (ขยาย/ยุบ)

- \*\*รองรับ role\*\* (เพิ่ม filter ได้)

- \*\*เปลี่ยนเมนูตาม path ได้ทันที\*\*

---

\*\*สรุป:\*\*

- สร้างไฟล์ page-menu.ts สำหรับ config เมนู

- SidebarComponent อ่านเมนูจาก config ตาม path

- Template ใช้ pageMenu วนแสดงเมนู

- ปรับแต่งได้ง่ายและเป็นมาตรฐาน

ถ้าต้องการตัวอย่าง filter ตาม role หรือ dynamic menu เพิ่มเติม แจ้งได้เลย!

pethzero: Application bundle generation failed. [0.188 seconds]

X [ERROR] NG8001: 'mat-progress-spinner' is not a known element:

1. If 'mat-progress-spinner' is an Angular component, then verify that it is included in the '@Component.imports' of this component.

2. If 'mat-progress-spinner' is a Web Component then add 'CUSTOM\_ELEMENTS\_SCHEMA' to the '@Component.schemas' of this component to suppress this message. [plugin angular-compiler]

src/app/ui/button/button.html:10:2:

10 │ <mat-progress-spinner

╵ ~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component ButtonComponent.

src/app/ui/button/button.ts:10:15:

10 │ templateUrl: './button.html',

╵ ~~~~~~~~~~~~~~~

Application bundle generation failed. [0.164 seconds]

X [ERROR] NG8001: 'mat-progress-spinner' is not a known element:

1. If 'mat-progress-spinner' is an Angular component, then verify that it is included in the '@Component.imports' of this component.

2. If 'mat-progress-spinner' is a Web Component then add 'CUSTOM\_ELEMENTS\_SCHEMA' to the '@Component.schemas' of this component to suppress this message. [plugin angular-compiler]

src/app/ui/button/button.html:10:2:

10 │ <mat-progress-spinner

╵ ~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component ButtonComponent.

src/app/ui/button/button.ts:10:15:

10 │ templateUrl: './button.html',

╵ ~~~~~~~~~~~~~~~

GitHub Copilot: ปัญหา

- Error: `'mat-progress-spinner' is not a known element`

- เพราะยัง \*\*ไม่ได้ import MatProgressSpinnerModule\*\* ใน `@Component.imports` ของ ButtonComponent

---

### วิธีแก้ไข

\*\*เพิ่ม MatProgressSpinnerModule ใน imports ของ ButtonComponent\*\*

````typescript

import { Component, Input } from '@angular/core';

import { MatButtonModule } from '@angular/material/button';

import { MatIconModule } from '@angular/material/icon';

import { MatProgressSpinnerModule } from '@angular/material/progress-spinner'; // เพิ่มบรรทัดนี้

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-button',

standalone: true,

imports: [

CommonModule,

MatButtonModule,

MatIconModule,

MatProgressSpinnerModule // เพิ่มตรงนี้

],

templateUrl: './button.html',

styleUrls: ['./button.scss']

})

export class ButtonComponent {

// ...existing code...

}

````

\*\*หลังเพิ่มนี้ error จะหาย และ spinner จะแสดงผลปกติ\*\*

pethzero: อยากให้

<mat-nav-list [@slideInOut]="isSidebarOpen ? 'in' : 'out'">

<div class="menu-name">{{ menu.name }}</div>

<mat-divider></mat-divider>

<ng-container \*ngFor="let item of pageMenu">

<a [routerLink]="item.link" class="sidebar-link" \*ngIf="!item.children">

<mat-icon>{{ item.icon }}</mat-icon>

<span>{{ item.name }}</span>

</a>

<mat-expansion-panel \*ngIf="item.children">

<mat-expansion-panel-header>

<mat-icon>{{ item.icon }}</mat-icon>

<span>{{ item.name }}</span>

</mat-expansion-panel-header>

<ng-container \*ngFor="let child of item.children">

<a [routerLink]="child.link" class="sidebar-link">

<mat-icon>{{ child.icon }}</mat-icon>

<span>{{ child.name }}</span>

</a>

</ng-container>

</mat-expansion-panel>

</ng-container>

</mat-nav-list>

มันเรียก icon สวยๆ แบบ

a

b

c

c1

c2

icon อยู่กลางตัวอักษรด้วย

GitHub Copilot: คุณสามารถปรับ sidebar ให้แสดงเมนูแบบมีไอคอนสวยๆ (icon อยู่กลางตัวอักษร) และรองรับเมนูซ้อน (children) ได้แบบนี้:

---

\*\*1. ปรับ HTML ให้ไอคอนอยู่กลางตัวอักษรและรองรับเมนูซ้อน\*\*

````html

<mat-nav-list [@slideInOut]="isSidebarOpen ? 'in' : 'out'">

<div class="menu-name">{{ menu.name }}</div>

<mat-divider></mat-divider>

<ng-container \*ngFor="let item of pageMenu">

<!-- เมนูหลัก -->

<a

[routerLink]="item.link"

class="sidebar-link flex items-center gap-3 py-2 px-4 rounded-lg hover:bg-blue-50 transition"

\*ngIf="!item.children"

>

<mat-icon class="text-xl align-middle">{{ item.icon }}</mat-icon>

<span class="align-middle font-medium">{{ item.name }}</span>

</a>

<!-- เมนูที่มีลูก -->

<mat-expansion-panel \*ngIf="item.children" class="sidebar-expansion">

<mat-expansion-panel-header>

<mat-icon class="text-xl align-middle">{{ item.icon }}</mat-icon>

<span class="align-middle font-medium ml-2">{{ item.name }}</span>

</mat-expansion-panel-header>

<ng-container \*ngFor="let child of item.children">

<a

[routerLink]="child.link"

class="sidebar-link flex items-center gap-3 py-2 px-8 rounded-lg hover:bg-blue-50 transition"

>

<mat-icon class="text-base align-middle">{{ child.icon }}</mat-icon>

<span class="align-middle font-normal">{{ child.name }}</span>

</a>

</ng-container>

</mat-expansion-panel>

</ng-container>

</mat-nav-list>

````

---

\*\*2. ปรับ CSS/SCSS เพิ่มเติม (sidebar.scss)\*\*

````scss

.sidebar-link {

display: flex;

align-items: center;

gap: 0.75rem;

font-size: 1rem;

color: #1e293b;

text-decoration: none;

border-radius: 8px;

transition: background 0.18s, color 0.18s;

padding: 0.5rem 1rem;

&:hover, &.active-link {

background: linear-gradient(90deg, #e3f2fd 0%, #fff 100%);

color: #1976d2;

font-weight: 600;

}

mat-icon {

font-size: 1.3rem;

vertical-align: middle;

color: #1976d2;

}

}

.sidebar-expansion {

background: transparent;

box-shadow: none;

margin-bottom: 0.25rem;

mat-expansion-panel-header {

padding: 0.25rem 1rem;

font-weight: 500;

font-size: 1rem;

color: #1976d2;

background: #f5f7fa;

border-radius: 8px;

min-height: 40px;

mat-icon {

font-size: 1.3rem;

vertical-align: middle;

}

}

}

````

---

\*\*3. ตัวอย่างโครงสร้างเมนู (ใน page-menu.ts หรือใน SidebarComponent)\*\*

```typescript

[

{ link: '/a', name: 'A', icon: 'home' },

{ link: '/b', name: 'B', icon: 'star' },

{

name: 'C', icon: 'folder', children: [

{ link: '/c/c1', name: 'C1', icon: 'chevron\_right' },

{ link: '/c/c2', name: 'C2', icon: 'chevron\_right' }

]

}

]

```

---

\*\*ผลลัพธ์:\*\*

- ไอคอนอยู่กลางตัวอักษร (แนวเดียวกับ PrimeNG/Material)

- เมนูหลักและเมนูลูกดูสวยงาม มีระยะห่างและ hover effect

- รองรับเมนูซ้อน (children) ได้

---

\*\*ถ้าต้องการตัวอย่างเมนูหรือไอคอนเพิ่มเติม แจ้งได้เลย!\*\*

pethzero: ERROR RuntimeError: NG04002: Cannot match any routes. URL Segment: 'dashboard/reports/daily'

at Recognizer.noMatchError (router-CsukTOog.mjs:3900:12)

at router-CsukTOog.mjs:3939:20

at catchError.js:10:39

at OperatorSubscriber2.\_this.\_error (OperatorSubscriber.js:25:21)

at Subscriber2.error (Subscriber.js:43:18)

at Subscriber2.\_error (Subscriber.js:67:30)

at Subscriber2.error (Subscriber.js:43:18)

at Subscriber2.\_error (Subscriber.js:67:30)

at Subscriber2.error (Subscriber.js:43:18)

at Subscriber2.\_error (Subscriber.js:67:30)

เวลากดผิดให้ทำการ error 404 เพิม targer แบบ \_blank

GitHub Copilot: ปัญหา

- เมื่อกดเมนูที่ไม่มี route จริง (เช่น `/dashboard/reports/daily`) จะเกิด error `NG04002: Cannot match any routes. URL Segment: ...`

- คุณต้องการ:

1. ถ้า route ไม่เจอ ให้ redirect ไปหน้า 404 (not-found)

2. ถ้าเมนูนั้นไม่มี route ให้เปิดลิงก์แบบ `<a target="\_blank">` (external link)

---

## วิธีแก้ไข

### 1. เพิ่ม Wildcard Route สำหรับ 404

ใน `app.routes.ts` \*\*เพิ่มท้ายสุดของ routes:\*\*

````typescript

{

path: '\*\*',

loadComponent: () => import('./pages/not-found/not-found').then(m => m.NotFoundComponent)

}

````

---

### 2. ปรับ Sidebar ให้รองรับ target="\_blank" สำหรับ external link

\*\*ใน page-menu.ts\*\*

- เพิ่ม property เช่น `external?: boolean` ใน `PageMenuItem`

- ถ้าเป็น external link ให้ set `external: true`

\*\*ตัวอย่าง:\*\*

````typescript

export interface PageMenuItem {

link: string;

name: string;

icon?: string;

children?: PageMenuItem[];

expen?: boolean;

roles?: string[];

external?: boolean; // เพิ่มตรงนี้

}

````

\*\*ตัวอย่างเมนู:\*\*

````typescript

{

link: 'https://www.google.com',

name: 'Google',

icon: 'open\_in\_new',

external: true

}

````

---

### 3. ปรับ sidebar.html ให้รองรับ target="\_blank"

````html

<ng-container \*ngFor="let item of pageMenu">

<!-- เมนูหลัก -->

<a

\*ngIf="!item.children && !item.external"

[routerLink]="item.link"

class="sidebar-link flex items-center gap-3 py-2 px-4 rounded-lg hover:bg-blue-50 transition"

>

<mat-icon class="text-xl align-middle">{{ item.icon }}</mat-icon>

<span class="align-middle font-medium">{{ item.name }}</span>

</a>

<a

\*ngIf="!item.children && item.external"

[href]="item.link"

target="\_blank"

rel="noopener"

class="sidebar-link flex items-center gap-3 py-2 px-4 rounded-lg hover:bg-blue-50 transition"

>

<mat-icon class="text-xl align-middle">{{ item.icon }}</mat-icon>

<span class="align-middle font-medium">{{ item.name }}</span>

</a>

<!-- ...children เหมือนเดิม... -->

</ng-container>

````

---

### 4. สรุป

- เพิ่ม wildcard route สำหรับ 404

- เพิ่ม property `external` ในเมนู ถ้าต้องการเปิดลิงก์นอก

- ปรับ template ให้ใช้ `[routerLink]` สำหรับ internal, `[href] target="\_blank"` สำหรับ external

---

\*\*แบบนี้จะไม่ error และรองรับทั้ง internal/external link ใน sidebar ได้อย่างปลอดภัย\*\*

pethzero: export interface PageMenuItem {

link: string;

name: string;

icon?: string;

children?: PageMenuItem[];

expen?: boolean;

roles?: string[];

external?: boolean; // เพิ่มตรงนี้

}

ตั้งชื่อตัวแปรให้ มันสอดคล้องได้ไหม

GitHub Copilot: แนวทางตั้งชื่อ interface และตัวแปรให้ \*\*สอดคล้องและสื่อความหมายดีขึ้น\*\*

(ตามมาตรฐาน Angular/TypeScript และองค์กรนิยมใช้)

---

\*\*1. Interface:\*\*

```typescript

export interface SidebarMenuItem {

path: string;

label: string;

icon?: string;

children?: SidebarMenuItem[];

expanded?: boolean;

roles?: string[];

external?: boolean;

}

```

\*\*2. ตัวแปรรวมเมนู:\*\*

```typescript

export const SIDEBAR\_MENUS: Record<string, SidebarMenuItem[]> = {

'/dashboard': [

{

path: '/dashboard',

label: 'แดชบอร์ด',

icon: 'dashboard',

expanded: false,

},

{

path: '/dashboard/overview',

label: 'ภาพรวม',

icon: 'insights',

expanded: false,

},

{

path: '/dashboard/reports',

label: 'รายงาน',

icon: 'bar\_chart',

expanded: true,

children: [

{ path: '/dashboard/reports/daily', label: 'รายวัน', icon: 'today' },

{ path: '/dashboard/reports/monthly', label: 'รายเดือน', icon: 'date\_range' }

]

}

],

'/patient': [

{

path: '/patient',

label: 'รายชื่อคนไข้',

icon: 'people',

expanded: false,

},

{

path: '/patient/add',

label: 'เพิ่มคนไข้',

icon: 'person\_add',

expanded: false,

}

]

// เพิ่มเมนูอื่นๆ ตาม path ได้

};

```

---

\*\*สรุป:\*\*

- ใช้ชื่อ `SidebarMenuItem` สำหรับ interface

- ใช้ชื่อ `SIDEBAR\_MENUS` สำหรับตัวแปรรวมเมนู

- เปลี่ยน `link` → `path`, `name` → `label`, `expen` → `expanded`

- อ่านง่ายและสื่อความหมายดีขึ้น

pethzero: html เอาใหม่หน่อย

GitHub Copilot: แนวทางใหม่ (ให้ชื่อ key ใน menu และใน template \*\*ตรงกัน\*\* และสื่อความหมายดี)

- ใช้ `path`, `label`, `icon`, `children`, `expanded`, `external`

- รองรับเมนูหลัก, เมนูย่อย, external link

- ใช้กับ interface และ PAGE\_MENUS ที่คุณตั้งไว้

---

````html

<mat-nav-list [@slideInOut]="isSidebarOpen ? 'in' : 'out'">

<div class="menu-name">{{ menu?.label || 'เมนู' }}</div>

<mat-divider></mat-divider>

<ng-container \*ngFor="let item of pageMenu">

<!-- เมนูหลัก (internal link) -->

<a

\*ngIf="!item.children && !item.external"

[routerLink]="item.path"

class="sidebar-link flex items-center gap-3 py-2 px-4 rounded-lg hover:bg-blue-50 transition"

>

<mat-icon class="text-xl align-middle">{{ item.icon }}</mat-icon>

<span class="align-middle font-medium">{{ item.label }}</span>

</a>

<!-- เมนูหลัก (external link) -->

<a

\*ngIf="!item.children && item.external"

[href]="item.path"

target="\_blank"

rel="noopener"

class="sidebar-link flex items-center gap-3 py-2 px-4 rounded-lg hover:bg-blue-50 transition"

>

<mat-icon class="text-xl align-middle">{{ item.icon }}</mat-icon>

<span class="align-middle font-medium">{{ item.label }}</span>

</a>

<!-- เมนูที่มีลูก -->

<mat-expansion-panel \*ngIf="item.children" class="sidebar-expansion" [expanded]="item.expanded">

<mat-expansion-panel-header>

<mat-icon class="text-xl align-middle">{{ item.icon }}</mat-icon>

<span class="align-middle font-medium ml-2">{{ item.label }}</span>

</mat-expansion-panel-header>

<ng-container \*ngFor="let child of item.children">

<a

\*ngIf="!child.external"

[routerLink]="child.path"

class="sidebar-link flex items-center gap-3 py-2 px-8 rounded-lg hover:bg-blue-50 transition"

>

<mat-icon class="text-base align-middle">{{ child.icon }}</mat-icon>

<span class="align-middle font-normal">{{ child.label }}</span>

</a>

<a

\*ngIf="child.external"

[href]="child.path"

target="\_blank"

rel="noopener"

class="sidebar-link flex items-center gap-3 py-2 px-8 rounded-lg hover:bg-blue-50 transition"

>

<mat-icon class="text-base align-middle">{{ child.icon }}</mat-icon>

<span class="align-middle font-normal">{{ child.label }}</span>

</a>

</ng-container>

</mat-expansion-panel>

</ng-container>

</mat-nav-list>

````

\*\*ข้อดี:\*\*

- ใช้ key ตรงกับ page-menu.ts

- รองรับทั้ง internal/external link ทั้งเมนูหลักและลูก

- ไอคอนอยู่กลางตัวอักษร สวยงาม

- ใช้กับ Angular Material ได้ทันที

pethzero: lication bundle generation failed. [0.163 seconds]

▲ [WARNING] NG8107: The left side of this optional chain operation does not include 'null' or 'undefined' in its type, therefore the '?.' operator can be replaced with the '.' operator. [plugin angular-compiler]

src/app/layout/sidebar/sidebar.html:2:34:

2 │ <div class="menu-name">{{ menu?.label || 'เมนู' }}</div>

╵ ~~~~~

Error occurs in the template of component SidebarComponent.

src/app/layout/sidebar/sidebar.ts:24:15:

24 │ templateUrl: './sidebar.html',

╵ ~~~~~~~~~~~~~~~~

X [ERROR] NG9: Property 'label' does not exist on type '{ name: string; heads: ({ label: string; expandable: boolean; link: string; icon: string; roles: string[]; details?: undefined; } | { label: string; expandable: boolean; roles: string[]; details: { label: string; icon: string; link: string; }[]; link?: undefined; icon?: undefined; })[]; }'. [plugin angular-compiler]

src/app/layout/sidebar/sidebar.html:2:34:

2 │ <div class="menu-name">{{ menu?.label || 'เมนู' }}</div>

╵ ~~~~~

Error occurs in the template of component SidebarComponent.

src/app/layout/sidebar/sidebar.ts:24:15:

24 │ templateUrl: './sidebar.html',

╵ ~~~~~~~~~~~~~~~~

GitHub Copilot: จาก error นี้

- `Property 'label' does not exist on type '{ name: string; ... }'`

- และ warning: `menu?.label` ใช้ optional chain ทั้งที่ menu ไม่มีวันเป็น null/undefined

\*\*สาเหตุ:\*\*

- ตัวแปร `menu` ใน SidebarComponent มี property ชื่อ `name` ไม่ใช่ `label`

---

### วิธีแก้ไข

\*\*เปลี่ยนใน sidebar.html:\*\*

```html

<div class="menu-name">{{ menu?.name || 'เมนู' }}</div>

```

หรือถ้า menu ไม่มีวันเป็น null:

```html

<div class="menu-name">{{ menu.name || 'เมนู' }}</div>

```

---

\*\*สรุป:\*\*

- ใช้ `menu.name` แทน `menu.label`

- ถ้า menu อาจเป็น undefined ให้ใช้ `menu?.name`

- ถ้าแน่ใจว่ามีเสมอ ใช้ `menu.name` ได้เลย

\*\*หลังแก้ไขนี้ error จะหาย\*\*

pethzero: [WARNING] NG8107: The left side of this optional chain operation does not include 'null' or 'undefined' in its type, therefore the '?.' operator can be replaced with the '.' operator. [plugin angular-compiler]

src/app/layout/sidebar/sidebar.html:2:34:

2 │ <div class="menu-name">{{ menu?.name || 'เมนู' }}</div>

╵ ~~~~

Error occurs in the template of component SidebarComponent.

src/app/layout/sidebar/sidebar.ts:24:15:

24 │ templateUrl: './sidebar.html',

╵ ~~~~~~~~~~~~~~~~

แก้ที

Component update sent to client(s).

GitHub Copilot: ให้เปลี่ยน

```html

<div class="menu-name">{{ menu?.name || 'เมนู' }}</div>

```

\*\*เป็น\*\*

```html

<div class="menu-name">{{ menu.name || 'เมนู' }}</div>

```

หรือถ้า menu ไม่มีวันเป็น null/undefined ใช้แค่

```html

<div class="menu-name">{{ menu.name }}</div>

```

\*\*แบบนี้ warning จะหาย และโค้ดจะตรงกับ type ของ menu\*\*

pethzero: อยากให้ปรับ icon คน

GitHub Copilot: ถ้าต้องการ \*\*ปรับไอคอน "คน" (account\_circle)\*\* ใน user menu ที่ header ให้ดูเด่น/สวยขึ้น

สามารถปรับขนาด สี หรือเปลี่ยนเป็นไอคอนอื่นได้

### ตัวอย่างการปรับ (แนะนำ)

\*\*1. ปรับขนาดและสีใน SCSS\*\*

ใน user-menu.scss มี class `.menu-head-icon` อยู่แล้ว

ถ้าอยากให้เด่นขึ้น เช่น เปลี่ยนสีพื้นหลังเป็นฟ้าอ่อน วงกลม:

````scss

.menu-head-icon {

font-size: 36px;

width: 44px !important;

height: 44px !important;

color: #1976d2;

background: #e3f2fd;

border-radius: 50%;

box-shadow: 0 2px 8px rgba(25, 118, 210, 0.08);

display: flex;

align-items: center;

justify-content: center;

}

````

\*\*2. ปรับใน HTML (user-menu.html)\*\*

ไม่ต้องแก้ไขอะไรเพิ่ม เพราะใช้ class เดิม

```html

<button mat-icon-button [matMenuTriggerFor]="userMenu" aria-label="User menu">

<mat-icon class="menu-head-icon">account\_circle</mat-icon>

</button>

```

\*\*3. ถ้าอยากเปลี่ยนเป็นไอคอนอื่น\*\*

เปลี่ยนชื่อไอคอนใน `<mat-icon>` เช่น `person`, `person\_outline`, `supervisor\_account` ฯลฯ

```html

<mat-icon class="menu-head-icon">person</mat-icon>

```

---

\*\*ผลลัพธ์:\*\*

- ไอคอนคนจะใหญ่ขึ้น วงกลม พื้นหลังฟ้า ดูเด่นและสวยแบบ dashboard สมัยใหม่

- สามารถเปลี่ยนสีหรือขนาดได้ตามต้องการ

ถ้าอยากได้แบบมีรูปโปรไฟล์ (avatar) จริง แจ้งได้เลย!

pethzero: ดูเหมือน ไม่กึ่งกลาง havor มันดูเพี้ยน

GitHub Copilot: เพื่อให้ \*\*icon คน (account\_circle) ใน user menu กึ่งกลางสมบูรณ์\*\* ทั้งแนวตั้ง-แนวนอน และ hover ไม่เพี้ยน

ให้ปรับทั้งปุ่มและไอคอนดังนี้

---

### 1. ปรับปุ่ม icon ใน user-menu.html

```html

<button mat-icon-button [matMenuTriggerFor]="userMenu" aria-label="User menu" class="user-menu-btn">

<mat-icon class="menu-head-icon">account\_circle</mat-icon>

</button>

```

---

### 2. ปรับ SCSS เพิ่มเติม

```scss

.user-menu-btn {

display: flex;

align-items: center;

justify-content: center;

padding: 0;

border-radius: 50%;

background: transparent;

transition: background 0.18s;

height: 48px;

width: 48px;

min-width: 48px;

&:hover, &:focus {

background: #e3f2fd !important;

box-shadow: 0 2px 8px rgba(25, 118, 210, 0.10);

}

}

.menu-head-icon {

font-size: 36px;

width: 44px !important;

height: 44px !important;

color: #1976d2;

background: #e3f2fd;

border-radius: 50%;

box-shadow: 0 2px 8px rgba(25, 118, 210, 0.08);

display: flex;

align-items: center;

justify-content: center;

margin: 0 auto;

transition: background 0.18s, color 0.18s;

}

```

---

\*\*ผลลัพธ์:\*\*

- ไอคอนคนจะอยู่กึ่งกลางปุ่มเสมอ

- Hover แล้วพื้นหลังฟ้าอ่อน ไม่เพี้ยน

- ดูสวยแบบ Material/PrimeNG

---

\*\*ถ้ายังไม่กึ่งกลาง แจ้ง browser หรือแนบ screenshot เพิ่มเติมได้เลย!\*\*

pethzero: Angular Material Table อยากให้ออกแบบ table ที่มี serverside true false

set ค่าต่างๆ ได้แบบ https://datatables.net/ อยากให้ เลียนแบบและปรังปรุงทันสมัย ใน Angular Material Table สร้างต้นแบบไว้ folder ไหนดี

GitHub Copilot: แนวทางมาตรฐานสำหรับ Angular (และองค์กรนิยมใช้)

\*\*ควรสร้างต้นแบบ Table ที่ reusable ไว้ใน `src/app/ui/table/`\*\*

- เป็น component ที่รับ input config (columns, data, serverSide, etc.)

- ใช้ Angular Material Table (`MatTableModule`) + Tailwind/SCSS

- รองรับ serverSide (true/false), paging, sorting, filter, loading, customizable columns ฯลฯ

- ใช้ pattern แบบ DataTables (แต่ทันสมัยกว่า)

---

## 1. \*\*โครงสร้างไฟล์\*\*

```

src/app/ui/table/

smart-table.ts

smart-table.html

smart-table.scss

smart-table.model.ts

```

---

## 2. \*\*ตัวอย่าง interface config (smart-table.model.ts)\*\*

````typescript

export interface SmartTableColumn {

field: string;

header: string;

sortable?: boolean;

filterable?: boolean;

width?: string;

type?: 'text' | 'number' | 'date' | 'custom';

cellTemplate?: any;

}

export interface SmartTableConfig {

columns: SmartTableColumn[];

serverSide?: boolean;

pageSizeOptions?: number[];

defaultPageSize?: number;

loading?: boolean;

showActions?: boolean;

}

````

---

## 3. \*\*component (smart-table.ts)\*\*

````typescript

import { Component, Input, Output, EventEmitter } from '@angular/core';

import { MatTableModule } from '@angular/material/table';

import { MatPaginatorModule, PageEvent } from '@angular/material/paginator';

import { MatSortModule, Sort } from '@angular/material/sort';

import { CommonModule } from '@angular/common';

import { SmartTableConfig } from './smart-table.model';

@Component({

selector: 'app-smart-table',

standalone: true,

imports: [CommonModule, MatTableModule, MatPaginatorModule, MatSortModule],

templateUrl: './smart-table.html',

styleUrls: ['./smart-table.scss']

})

export class SmartTableComponent {

@Input() config!: SmartTableConfig;

@Input() data: any[] = [];

@Input() total = 0;

@Input() loading = false;

@Output() pageChange = new EventEmitter<PageEvent>();

@Output() sortChange = new EventEmitter<Sort>();

@Output() filterChange = new EventEmitter<string>();

displayedColumns: string[] = [];

ngOnInit() {

this.displayedColumns = this.config.columns.map(col => col.field);

if (this.config.showActions) this.displayedColumns.push('actions');

}

onPage(event: PageEvent) {

this.pageChange.emit(event);

}

onSort(event: Sort) {

this.sortChange.emit(event);

}

}

````

---

## 4. \*\*template (smart-table.html)\*\*

````html

<div class="relative">

<mat-progress-bar \*ngIf="loading" mode="indeterminate"></mat-progress-bar>

<table mat-table [dataSource]="data" matSort (matSortChange)="onSort($event)" class="min-w-full">

<ng-container \*ngFor="let col of config.columns" [matColumnDef]="col.field">

<th mat-header-cell \*matHeaderCellDef mat-sort-header="{{col.sortable ? '' : null}}">

{{ col.header }}

</th>

<td mat-cell \*matCellDef="let row">

{{ row[col.field] }}

</td>

</ng-container>

<ng-container \*ngIf="config.showActions" matColumnDef="actions">

<th mat-header-cell \*matHeaderCellDef>Actions</th>

<td mat-cell \*matCellDef="let row">

<ng-content select="[actions]"></ng-content>

</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

<tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

</table>

<mat-paginator

\*ngIf="!config.serverSide"

[length]="data.length"

[pageSizeOptions]="config.pageSizeOptions || [10, 25, 50]"

[pageSize]="config.defaultPageSize || 10"

(page)="onPage($event)">

</mat-paginator>

<mat-paginator

\*ngIf="config.serverSide"

[length]="total"

[pageSizeOptions]="config.pageSizeOptions || [10, 25, 50]"

[pageSize]="config.defaultPageSize || 10"

(page)="onPage($event)">

</mat-paginator>

</div>

````

---

## 5. \*\*วิธีใช้ในหน้าอื่น\*\*

```html

<app-smart-table

[config]="tableConfig"

[data]="patients"

[total]="totalPatients"

[loading]="loading"

(pageChange)="onPage($event)"

(sortChange)="onSort($event)">

</app-smart-table>

```

---

## 6. \*\*สรุป\*\*

- สร้างไว้ใน `src/app/ui/table/`

- ใช้ซ้ำได้ทุกที่

- ปรับแต่ง config ได้เหมือน DataTables

- รองรับ serverSide/ClientSide, loading, actions, sorting, paging

\*\*ถ้าต้องการตัวอย่างการเชื่อมต่อ API หรือ custom cell แจ้งได้เลย!\*\*

Similar code found with 1 license type

pethzero: {

path: 'patient',

loadChildren: () => import('./features/patient/patient-routing.module').then(m => m.PeopleRoutingModule),

canActivate: [AuthGuard],

data: { roles: ['guest', 'user', 'admin'] } // ให้ user และ admin เข้าได้

},

const routes: Routes = [

{ path: 'patient-list', component: PatientList },

{ path: 'patient-form', component: PatientFormComponent },

];

เราอยากให้แบบว่า จะมี patient-questionnaire ที่จะสร้า่ง คือ แบบสอบถามปรึกษา

อยากได้ Postgest database แบบ หน้าตา

GitHub Copilot: ถ้าคุณต้องการ \*\*ออกแบบตาราง (table) สำหรับ "แบบสอบถามปรึกษา" (patient-questionnaire)\*\* ในฐานข้อมูล PostgreSQL

และต้องการแนวทางที่เหมาะกับ Angular/NestJS (รองรับการขยายในอนาคต)

นี่คือตัวอย่าง schema ที่นิยมใช้ใน production

---

## 1. ตัวอย่าง Table: `patient\_questionnaire`

````sql

-- ตารางหลักสำหรับเก็บแบบสอบถามแต่ละชุด

CREATE TABLE patient\_questionnaire (

id SERIAL PRIMARY KEY,

patient\_id INTEGER NOT NULL REFERENCES patient(id),

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW(),

status VARCHAR(32) DEFAULT 'draft', -- draft, submitted, reviewed, etc.

note TEXT

);

````

---

## 2. Table: `questionnaire\_question`

````sql

-- เก็บคำถามแต่ละข้อ (master)

CREATE TABLE questionnaire\_question (

id SERIAL PRIMARY KEY,

code VARCHAR(64) UNIQUE NOT NULL,

text TEXT NOT NULL,

type VARCHAR(32) NOT NULL, -- เช่น 'text', 'choice', 'number', 'date'

choices TEXT[], -- ถ้าเป็น choice จะเก็บตัวเลือก

is\_required BOOLEAN DEFAULT false

);

````

---

## 3. Table: `patient\_questionnaire\_answer`

````sql

-- เก็บคำตอบของแต่ละ patient questionnaire

CREATE TABLE patient\_questionnaire\_answer (

id SERIAL PRIMARY KEY,

questionnaire\_id INTEGER NOT NULL REFERENCES patient\_questionnaire(id),

question\_id INTEGER NOT NULL REFERENCES questionnaire\_question(id),

answer\_text TEXT, -- เก็บคำตอบ (text/number/choice)

answered\_at TIMESTAMP DEFAULT NOW()

);

````

---

## 4. ความสัมพันธ์

- 1 patient มีได้หลาย questionnaire

- 1 questionnaire มีได้หลาย answer (แต่ละ answer อ้างอิง question)

---

## 5. ตัวอย่างข้อมูล

\*\*questionnaire\_question\*\*

| id | code | text | type | choices | is\_required |

|----|--------------|------------------------|---------|---------------------------|-------------|

| 1 | symptom | อาการที่ปรึกษา | text | NULL | true |

| 2 | fever | มีไข้หรือไม่ | choice | {'ไม่มี','ต่ำ','สูง'} | false |

| 3 | duration | เป็นมากี่วัน | number | NULL | false |

\*\*patient\_questionnaire\*\*

| id | patient\_id | created\_at | status | note |

|----|------------|---------------------|----------|-----------|

| 1 | 101 | 2025-07-27 10:00:00 | draft | NULL |

\*\*patient\_questionnaire\_answer\*\*

| id | questionnaire\_id | question\_id | answer\_text | answered\_at |

|----|------------------|------------|-------------|-----------------------|

| 1 | 1 | 1 | ปวดหัว | 2025-07-27 10:01:00 |

| 2 | 1 | 2 | ต่ำ | 2025-07-27 10:01:10 |

| 3 | 1 | 3 | 2 | 2025-07-27 10:01:20 |

---

## 6. ข้อดีของโครงสร้างนี้

- \*\*ยืดหยุ่น:\*\* เพิ่ม/ลดคำถามได้ ไม่ต้องแก้ schema

- \*\*รองรับหลายแบบสอบถาม:\*\* ถ้าอนาคตมีหลายฟอร์ม แค่เพิ่มใน `questionnaire\_question`

- \*\*เหมาะกับ Angular/NestJS:\*\* mapping ง่าย, query ง่าย, ขยายได้

---

## 7. ถ้าต้องการเก็บ "template" ของแบบสอบถามหลายชุด

เพิ่ม table `questionnaire\_template` แล้วให้ `questionnaire\_question` อ้างอิง template\_id

---

\*\*สรุป:\*\*

- ใช้ 3 table หลัก: patient\_questionnaire, questionnaire\_question, patient\_questionnaire\_answer

- ยืดหยุ่น, ขยายง่าย, เหมาะกับงาน consult/แบบสอบถาม

ถ้าต้องการตัวอย่าง entity NestJS หรือ Angular form สำหรับแบบสอบถาม แจ้งได้เลย!

pethzero: แล้วก็

{

path: 'patient',

loadChildren: () => import('./features/patient/patient-routing.module').then(m => m.PeopleRoutingModule),

canActivate: [AuthGuard],

data: { roles: ['guest', 'user', 'admin'] } // ให้ user และ admin เข้าได้

},

const routes: Routes = [

{ path: 'patient-list', component: PatientList },

{ path: 'patient-form', component: PatientFormComponent },

];

patient-list และ patient-form จะมีแค่ user และ admin เท่านั้นที่เข้าได้เขียนได้แบบไหน

เราอยากให้แบบว่า จะมี patient-questionnaire ที่จะสร้า่ง คือ แบบสอบถามปรึกษา

GitHub Copilot: คุณสามารถ \*\*จำกัดสิทธิ์เข้า route\*\* (เช่น patient-list, patient-form) ให้เฉพาะ user และ admin ได้ โดยใช้ Angular Route Guard + data roles แบบนี้

---

### 1. ตัวอย่าง patient-routing.module.ts

````typescript

import { Routes } from '@angular/router';

import { PatientList } from './patient-list/patient-list';

import { PatientFormComponent } from './patient-form/patient-form';

import { AuthGuard } from '../../core/guards/auth.guard';

export const routes: Routes = [

{

path: 'patient-list',

component: PatientList,

canActivate: [AuthGuard],

data: { roles: ['user', 'admin'] }

},

{

path: 'patient-form',

component: PatientFormComponent,

canActivate: [AuthGuard],

data: { roles: ['user', 'admin'] }

},

// เพิ่ม route สำหรับ patient-questionnaire ได้ที่นี่

{

path: 'patient-questionnaire',

loadComponent: () => import('./patient-questionnaire/patient-questionnaire').then(m => m.PatientQuestionnaireComponent),

canActivate: [AuthGuard],

data: { roles: ['user', 'admin'] }

}

];

````

---

### 2. ตัวอย่าง AuthGuard (core/guards/auth.guard.ts)

```typescript

import { Injectable } from '@angular/core';

import { CanActivate, ActivatedRouteSnapshot, Router } from '@angular/router';

import { AuthService } from '../services/auth.service';

@Injectable({ providedIn: 'root' })

export class AuthGuard implements CanActivate {

constructor(private auth: AuthService, private router: Router) {}

canActivate(route: ActivatedRouteSnapshot): boolean {

const user = this.auth.currentUserValue;

const allowedRoles = route.data['roles'] as string[];

if (user && allowedRoles.includes(user.role)) {

return true;

}

// redirect to not-found or login

this.router.navigate(['/not-found']);

return false;

}

}

```

---

### 3. สรุป

- ใส่ `canActivate: [AuthGuard]` และ `data: { roles: [...] }` ในแต่ละ route

- ใน AuthGuard ตรวจสอบ role ของ user กับ roles ที่กำหนดใน data

- เพิ่ม route ใหม่ (เช่น patient-questionnaire) ได้ใน routing module เดียวกัน

---

\*\*แบบนี้เฉพาะ user และ admin เท่านั้นที่เข้า patient-list, patient-form, patient-questionnaire ได้\*\*

ถ้าต้องการตัวอย่าง patient-questionnaire component เพิ่มเติม แจ้งได้เลย!

Similar code found with 3 license types

pethzero: // auth-guard.ts

import { Injectable } from '@angular/core';

import { AuthService } from '../services/auth.service';

import {

CanActivate,

CanActivateChild,

CanMatch,

ActivatedRouteSnapshot,

Route,

Router,

UrlSegment

} from '@angular/router';

@Injectable({ providedIn: 'root' })

export class AuthGuard implements CanActivate, CanActivateChild, CanMatch {

constructor(private auth: AuthService, private router: Router) { }

canActivate(route: ActivatedRouteSnapshot): boolean {

return this.checkAccess(route.data?.['roles']);

}

canActivateChild(childRoute: ActivatedRouteSnapshot): boolean {

return this.checkAccess(childRoute.data?.['roles']);

}

canMatch(route: Route, segments: UrlSegment[]): boolean {

return this.checkAccess(route.data?.['roles']);

}

private checkAccess(allowedRoles?: string[]): boolean {

const user = this.auth.getCurrentUser();

console.log(user)

if (!user) {

this.router.navigate(['/login']);

return false;

}

if (!allowedRoles || allowedRoles.length === 0) {

return true;

}

if (allowedRoles.includes(user.role)) {

return true;

}

this.router.navigate(['/unauthorized']);

return false;

}

}

อันนี้ของเดิม เหมือนจะดีแล้วนะแต่ต้องปรับปรุง อีกนิดนะ

GitHub Copilot: แนวทางปรับปรุง AuthGuard ของคุณให้ \*\*ปลอดภัยและยืดหยุ่นขึ้น\*\*

- รองรับ async (เช่น กรณี getCurrentUser เป็น Observable/Promise ในอนาคต)

- ป้องกัน redirect loop (ถ้าอยู่หน้า /unauthorized หรือ /login แล้ว)

- เพิ่ม comment และ logging ที่เหมาะสม

---

```typescript

import { Injectable } from '@angular/core';

import { AuthService } from '../services/auth.service';

import {

CanActivate,

CanActivateChild,

CanMatch,

ActivatedRouteSnapshot,

Route,

Router,

UrlSegment

} from '@angular/router';

@Injectable({ providedIn: 'root' })

export class AuthGuard implements CanActivate, CanActivateChild, CanMatch {

constructor(private auth: AuthService, private router: Router) {}

canActivate(route: ActivatedRouteSnapshot): boolean {

return this.checkAccess(route.data?.['roles']);

}

canActivateChild(childRoute: ActivatedRouteSnapshot): boolean {

return this.checkAccess(childRoute.data?.['roles']);

}

canMatch(route: Route, segments: UrlSegment[]): boolean {

return this.checkAccess(route.data?.['roles']);

}

private checkAccess(allowedRoles?: string[]): boolean {

const user = this.auth.getCurrentUser();

// Logging สำหรับ debug

// console.log('[AuthGuard] user:', user, 'allowedRoles:', allowedRoles);

if (!user) {

// ป้องกัน redirect loop

if (this.router.url !== '/login') {

this.router.navigate(['/login']);

}

return false;

}

if (!allowedRoles || allowedRoles.length === 0) {

return true;

}

if (allowedRoles.includes(user.role)) {

return true;

}

// ป้องกัน redirect loop

if (this.router.url !== '/unauthorized') {

this.router.navigate(['/unauthorized']);

}

return false;

}

}

```

\*\*ข้อดี:\*\*

- ป้องกัน redirect ซ้ำซ้อน

- รองรับ roles แบบ flexible

- พร้อมขยายเป็น async ได้ในอนาคต

- Logging ชัดเจน (comment ไว้ให้เปิดใช้ตอน debug)

\*\*ถ้าต้องการรองรับ Observable/Promise เพิ่มเติม แจ้งได้เลย!\*\*

pethzero: X [ERROR] NG8001: 'mat-progress-bar' is not a known element:

1. If 'mat-progress-bar' is an Angular component, then verify that it is included in the '@Component.imports' of this component.

2. If 'mat-progress-bar' is a Web Component then add 'CUSTOM\_ELEMENTS\_SCHEMA' to the '@Component.schemas' of this component to suppress this message. [plugin angular-compiler]

src/app/ui/table/smart-table.html:2:2:

2 │ <mat-progress-bar \*ngIf="loading" mode="indeterminate"></mat-prog...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component SmartTableComponent.

src/app/ui/table/smart-table.ts:12:15:

12 │ templateUrl: './smart-table.html',

╵ ~~~~~~~~~~~~~~~~~~~~

X [ERROR] TS-992008: Could not find stylesheet file './smart-table.scss'. [plugin angular-compiler]

src/app/ui/table/smart-table.ts:13:14:

13 │ styleUrls: ['./smart-table.scss']

╵ ~~~~~~~~~~~~~~~~~~~~

Application bundle generation failed. [0.121 seconds]

X [ERROR] NG8001: 'mat-progress-bar' is not a known element:

1. If 'mat-progress-bar' is an Angular component, then verify that it is included in the '@Component.imports' of this component.

2. If 'mat-progress-bar' is a Web Component then add 'CUSTOM\_ELEMENTS\_SCHEMA' to the '@Component.schemas' of this component to suppress this message. [plugin angular-compiler]

src/app/ui/table/smart-table.html:2:2:

2 │ <mat-progress-bar \*ngIf="loading" mode="indeterminate"></mat-prog...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component SmartTableComponent.

src/app/ui/table/smart-table.ts:12:15:

12 │ templateUrl: './smart-table.html',

╵ ~~~~~~~~~~~~~~~~~~~~

X [ERROR] TS-992008: Could not find stylesheet file './smart-table.scss'. [plugin angular-compiler]

src/app/ui/table/smart-table.ts:13:14:

13 │ styleUrls: ['./smart-table.scss']

╵ ~~~~~~~~~~~~~~~~~~~~

GitHub Copilot: ### สาเหตุและวิธีแก้ไข

#### 1. Error: `'mat-progress-bar' is not a known element`

\*\*สาเหตุ:\*\*

ยังไม่ได้ import `MatProgressBarModule` ใน `imports` ของ `SmartTableComponent`

\*\*วิธีแก้ไข:\*\*

เพิ่ม `MatProgressBarModule` ใน imports

````typescript

import { MatProgressBarModule } from '@angular/material/progress-bar';

// ...existing imports...

@Component({

selector: 'app-smart-table',

standalone: true,

imports: [

CommonModule,

MatTableModule,

MatPaginatorModule,

MatSortModule,

MatProgressBarModule // <-- เพิ่มตรงนี้

],

templateUrl: './smart-table.html',

styleUrls: ['./smart-table.scss']

})

export class SmartTableComponent {

// ...existing code...

}

````

---

#### 2. Error: `Could not find stylesheet file './smart-table.scss'`

\*\*สาเหตุ:\*\*

ยังไม่มีไฟล์ `smart-table.scss` ในโฟลเดอร์นี้

\*\*วิธีแก้ไข:\*\*

สร้างไฟล์เปล่าๆ หรือใส่ style พื้นฐาน

````scss

/\* Smart Table styles \*/

:host {

display: block;

}

````

---

\*\*สรุป:\*\*

- เพิ่ม `MatProgressBarModule` ใน imports ของ component

- สร้างไฟล์ `smart-table.scss` (จะว่างหรือใส่ style ก็ได้)

\*\*หลังแก้ไขนี้ error จะหาย\*\*

pethzero: Application bundle generation complete. [0.492 seconds]

Page reload sent to client(s).

Application bundle generation failed. [0.612 seconds]

X [ERROR] NG8002: Can't bind to 'ngStyle' since it isn't a known property of 'div'. [plugin angular-compiler]

src/app/ui/loading-spinner/loading-spinner.html:2:23:

2 │ ...="spinner" [ngStyle]="{'width.px': size, 'height.px': size, 'bor...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component LoadingSpinnerComponent.

src/app/ui/loading-spinner/loading-spinner.ts:6:15:

6 │ templateUrl: './loading-spinner.html',

╵ ~~~~~~~~~~~~~~~~~~~~~~~~

GitHub Copilot: ปัญหา

- Error: `Can't bind to 'ngStyle' since it isn't a known property of 'div'.`

- เพราะ \*\*ยังไม่ได้ import CommonModule\*\* ใน `imports` ของ LoadingSpinnerComponent (สำหรับ directive อย่าง ngStyle)

---

### วิธีแก้ไข

\*\*เพิ่ม CommonModule ใน imports ของ LoadingSpinnerComponent\*\*

````typescript

import { Component, Input } from '@angular/core';

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-loading-spinner',

standalone: true,

templateUrl: './loading-spinner.html',

styleUrls: ['./loading-spinner.scss'],

imports: [CommonModule] // ✅ เพิ่มตรงนี้

})

export class LoadingSpinnerComponent {

@Input() size = 48;

@Input() color = '#1976d2';

}

````

\*\*หลังเพิ่มนี้ error จะหาย และ ngStyle จะทำงานได้\*\*

pethzero:

Page reload sent to client(s).

Application bundle generation failed. [0.847 seconds]

X [ERROR] NG8002: Can't bind to 'ngStyle' since it isn't a known property of 'div'. [plugin angular-compiler]

src/app/ui/loading-spinner/loading-spinner.html:2:23:

2 │ ...="spinner" [ngStyle]="{'width.px': size, 'height.px': size, 'bor...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component LoadingSpinnerComponent.

src/app/ui/loading-spinner/loading-spinner.ts:6:15:

6 │ templateUrl: './loading-spinner.html',

╵ ~~~~~~~~~~~~~~~~~~~~~~~~

GitHub Copilot: คุณแก้ถูกต้องแล้วที่เพิ่ม `CommonModule` ใน `imports` ของ LoadingSpinnerComponent

แต่ถ้ายัง error แบบนี้ แสดงว่า \*\*Angular เวอร์ชันที่ใช้ อาจยังไม่รองรับ imports ใน @Component (standalone) แบบนี้\*\*

หรือมีปัญหา cache/compile

\*\*วิธีแก้ไขที่ได้ผลแน่นอน:\*\*

1. \*\*ตรวจสอบว่าไฟล์นี้ถูก import ใน module อื่นหรือไม่\*\*

- ถ้าใช้แบบ standalone component ให้แน่ใจว่า component ที่เรียก `<app-loading-spinner>`

ก็ต้อง import LoadingSpinnerComponent ใน `imports` ของมันด้วย

2. \*\*ลองลบ node\_modules แล้วติดตั้งใหม่\*\*

```sh

rm -rf node\_modules

npm install

```

3. \*\*ถ้าใช้ Angular < 15\*\*

- ต้องประกาศ LoadingSpinnerComponent ใน `declarations` ของ module และ import `CommonModule` ใน module นั้น

4. \*\*ถ้าใช้ Angular 15+ (standalone component)\*\*

- ตรวจสอบ component ที่เรียก `<app-loading-spinner>` ว่า import LoadingSpinnerComponent ใน `imports` ของมันด้วย เช่น

```typescript

imports: [

...,

LoadingSpinnerComponent

]

```

\*\*สรุป:\*\*

- โค้ดใน LoadingSpinnerComponent ถูกต้องแล้ว

- ให้เช็ก component แม่ที่เรียกใช้ ว่า import LoadingSpinnerComponent ใน `imports` ด้วย

- ถ้ายัง error แจ้ง Angular version และตัวอย่าง component ที่เรียกใช้ spinner มาได้เลยครับ

pethzero:

Page reload sent to client(s).

Application bundle generation failed. [0.847 seconds]

X [ERROR] NG8002: Can't bind to 'ngStyle' since it isn't a known property of 'div'. [plugin angular-compiler]

src/app/ui/loading-spinner/loading-spinner.html:2:23:

2 │ ...="spinner" [ngStyle]="{'width.px': size, 'height.px': size, 'bor...

╵ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component LoadingSpinnerComponent.

src/app/ui/loading-spinner/loading-spinner.ts:6:15:

6 │ templateUrl: './loading-spinner.html',

╵ ~~~~~~~~~~~~~~~~~~~~~~~~

GitHub Copilot: ปัญหานี้เกิดจาก \*\*component ที่เรียกใช้ `<app-loading-spinner>` ยังไม่ได้ import CommonModule หรือ LoadingSpinnerComponent ใน imports\*\*

แม้ว่า LoadingSpinnerComponent จะ import CommonModule แล้ว แต่ component แม่ (ที่เรียกใช้ spinner) ก็ต้อง import ด้วย

---

### วิธีแก้ไข

\*\*1. ถ้าใช้แบบ standalone component:\*\*

- ใน component ที่เรียก `<app-loading-spinner>` (เช่นหน้า page หรือ component อื่น)

- ต้องเพิ่ม `LoadingSpinnerComponent` ใน `imports` ของ component นั้น

ตัวอย่าง:

````typescript

// ใน component ที่เรียกใช้ spinner

import { LoadingSpinnerComponent } from 'src/app/ui/loading-spinner/loading-spinner';

@Component({

// ...

standalone: true,

imports: [

// ...อื่นๆ

LoadingSpinnerComponent

],

// ...

})

export class SomeParentComponent { }

````

---

\*\*2. ถ้าใช้ใน module-based (NgModule):\*\*

- ให้ import `CommonModule` ใน module ที่ประกาศ LoadingSpinnerComponent

---

\*\*สรุป:\*\*

- Component ที่เรียก `<app-loading-spinner>` ต้อง import LoadingSpinnerComponent ใน `imports` ด้วย

- ไม่เช่นนั้น directive อย่าง `ngStyle` จะไม่ถูกประกาศใน scope ของ component นั้น

\*\*หลังแก้ไขนี้ error จะหายทันที\*\*

pethzero: CREATE TABLE IF NOT EXISTS public.user\_tokens

(

id uuid NOT NULL DEFAULT gen\_random\_uuid(),

user\_id uuid,

token text COLLATE pg\_catalog."default" NOT NULL,

created\_at timestamp with time zone DEFAULT now(),

expired\_at timestamp with time zone,

revoked boolean DEFAULT false,

CONSTRAINT user\_tokens\_pkey PRIMARY KEY (id)

);

CREATE TABLE IF NOT EXISTS public.login\_logs

(

id serial NOT NULL,

user\_id uuid,

login\_time timestamp with time zone DEFAULT now(),

ip\_address character varying(45) COLLATE pg\_catalog."default",

user\_agent text COLLATE pg\_catalog."default",

success boolean,

fail\_reason text COLLATE pg\_catalog."default",

CONSTRAINT login\_logs\_pkey PRIMARY KEY (id)

);

ALTER TABLE IF EXISTS public.login\_logs

ADD CONSTRAINT login\_logs\_user\_id\_fkey FOREIGN KEY (user\_id)

REFERENCES public.users (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

ALTER TABLE IF EXISTS public.user\_tokens

ADD CONSTRAINT user\_tokens\_user\_id\_fkey FOREIGN KEY (user\_id)

REFERENCES public.users (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

พวกนี้ ต้องใช้งานแบบบไหนยังไง กับ backend สร้างอะไรตอนไหน

GitHub Copilot: ตารางเหล่านี้ (`user\_tokens`, `login\_logs`) คือ \*\*ตารางระบบสำหรับจัดการ session/token และบันทึก log การเข้าสู่ระบบ\*\*

เหมาะกับระบบ authentication ที่ใช้ JWT หรือ session token ใน backend (NestJS/Express)

\*\*สรุปการใช้งานและสร้าง/บันทึกข้อมูลแต่ละตาราง:\*\*

---

## 1. ตาราง `user\_tokens`

\*\*ใช้สำหรับ:\*\*

- เก็บ token (เช่น JWT หรือ refresh token) ที่ออกให้ user แต่ละคน

- ใช้ตรวจสอบว่า token นี้ยัง valid หรือถูก revoke แล้วหรือไม่ (เช่น ตอน logout, เปลี่ยนรหัสผ่าน)

\*\*สร้างข้อมูลตอนไหน:\*\*

- \*\*หลัง login สำเร็จ:\*\*

- สร้าง token ใหม่ (JWT หรือ refresh token)

- บันทึก token นี้ลงใน `user\_tokens` พร้อม user\_id, created\_at, expired\_at

- \*\*ตอน logout หรือ revoke:\*\*

- อัปเดต `revoked = true` หรือ ลบ record นี้ทิ้ง

\*\*ตัวอย่างโค้ด (NestJS):\*\*

```typescript

// หลัง login สำเร็จ

await userTokenRepo.save({

userId: user.id,

token: jwtToken,

createdAt: new Date(),

expiredAt: new Date(Date.now() + 86400 \* 1000), // ตัวอย่าง 1 วัน

revoked: false

});

// ตอน logout

await userTokenRepo.update({ token }, { revoked: true });

```

---

## 2. ตาราง `login\_logs`

\*\*ใช้สำหรับ:\*\*

- เก็บ log ทุกครั้งที่มีการ login (สำเร็จหรือไม่สำเร็จ)

- ใช้ตรวจสอบความปลอดภัย, audit, หรือวิเคราะห์พฤติกรรมผู้ใช้

\*\*สร้างข้อมูลตอนไหน:\*\*

- \*\*ทุกครั้งที่มีการ login (สำเร็จหรือ fail):\*\*

- บันทึก user\_id, เวลา, ip, user\_agent, success (true/false), fail\_reason (ถ้ามี)

\*\*ตัวอย่างโค้ด (NestJS):\*\*

```typescript

// หลัง login สำเร็จหรือ fail

await loginLogRepo.save({

userId: user?.id,

loginTime: new Date(),

ipAddress: req.ip,

userAgent: req.headers['user-agent'],

success: isSuccess,

failReason: isSuccess ? null : 'Invalid password'

});

```

---

## 3. การสร้างตาราง (Migration/Init)

- สร้างตารางเหล่านี้ \*\*ครั้งเดียว\*\* ตอน setup database (ใช้ migration หรือรัน SQL ตรง)

- หลังจากนั้น NestJS จะใช้ ORM (TypeORM) map entity กับตารางนี้อัตโนมัติ

---

## 4. การใช้งานใน backend

- \*\*เวลามี user login:\*\*

- ตรวจสอบ username/password

- ถ้าสำเร็จ → สร้าง token, save ลง `user\_tokens`, log ลง `login\_logs`

- ถ้า fail → log ลง `login\_logs` (success=false, fail\_reason)

- \*\*เวลาตรวจสอบ token:\*\*

- ตรวจสอบว่า token นี้ยังอยู่ใน `user\_tokens` และ `revoked = false`

- \*\*เวลาผู้ใช้ logout:\*\*

- set `revoked = true` ใน `user\_tokens` สำหรับ token นี้

---

## 5. สรุป

- สร้างตารางตอน setup database (migration/init)

- บันทึกข้อมูลทุกครั้งที่ login/logout หรือมีการออก token ใหม่

- ใช้ร่วมกับระบบ auth ของ backend เพื่อความปลอดภัยและตรวจสอบย้อนหลัง

---

\*\*ถ้าต้องการตัวอย่าง entity TypeORM หรือ service สำหรับจัดการ 2 ตารางนี้ แจ้งได้เลย!\*\*

pethzero: เอางงี้ทำที่หน้าบ้าน หรือหลังบ้าน เริ่มต้นต้องทำยังไงดี สร้างอะไร ทำไฟล์ที่ไหย

GitHub Copilot: ### สรุปแนวทางการเริ่มต้น \*\*ระบบ Login/Token/Log\*\* (user\_tokens, login\_logs)

\*\*ควรเริ่มที่ "หลังบ้าน (backend)" ก่อน\*\* แล้วจึงเชื่อมต่อกับ "หน้าบ้าน (frontend)"

เพราะ backend คือจุดที่สร้าง/ตรวจสอบ token และบันทึก log จริง

---

## 1. \*\*หลังบ้าน (backend/NestJS) ต้องทำอะไรบ้าง\*\*

\*\*1.1 สร้าง Entity/Model สำหรับ user\_tokens และ login\_logs\*\*

- สร้างไฟล์ entity ใน `backend/src/modules/auth/entities/` หรือ `backend/src/modules/user/entities/`

- ตัวอย่าง:

- `user-token.entity.ts`

- `login-log.entity.ts`

\*\*1.2 สร้าง Service/Repository สำหรับบันทึก token และ log\*\*

- สร้าง service ใน auth

- ตัวอย่าง:

- `auth.service.ts` (เพิ่ม logic save token/log)

- `auth.controller.ts` (endpoint login/logout)

\*\*1.3 เพิ่ม logic ใน service\*\*

- หลัง login สำเร็จ → save token ลง user\_tokens, save log ลง login\_logs

- หลัง logout → revoke token (update revoked = true)

- ทุกครั้งที่ login (สำเร็จ/ล้มเหลว) → save log

\*\*1.4 Migration หรือ SQL สำหรับสร้างตาราง\*\*

- สร้าง migration หรือรัน SQL ที่คุณมีใน PostgreSQL

---

## 2. \*\*หน้าบ้าน (frontend/Angular) ต้องทำอะไรบ้าง\*\*

\*\*2.1 สร้างหน้า Login\*\*

- มีอยู่แล้ว (`src/app/pages/login/`)

- ส่ง username/password ไป backend

\*\*2.2 รับ token จาก backend\*\*

- เก็บ token ใน localStorage/cookie

- ส่ง token ไปกับทุก request (ผ่าน interceptor)

\*\*2.3 แสดง error หรือ redirect ตามผลลัพธ์ login\*\*

---

## 3. \*\*ไฟล์ที่ควรสร้าง/แก้ไข (backend)\*\*

```

backend/

src/

modules/

auth/

auth.controller.ts

auth.service.ts

entities/

user-token.entity.ts

login-log.entity.ts

```

- ถ้าใช้ TypeORM: สร้าง entity ตาม schema ที่คุณมี

- เพิ่ม logic ใน service/controller

---

## 4. \*\*ไฟล์ที่ควรสร้าง/แก้ไข (frontend)\*\*

```

webui/

src/

app/

core/

services/

auth.service.ts // เรียก login/logout

pages/

login/

... (มีอยู่แล้ว)

```

- auth.service.ts: เรียก API login/logout, เก็บ token

---

## 5. \*\*เริ่มต้นแบบ Step-by-step\*\*

1. \*\*สร้าง/แก้ไข entity + migration ใน backend\*\*

2. \*\*เพิ่ม logic บันทึก token/log ใน service backend\*\*

3. \*\*ทดสอบ login/logout ผ่าน Postman หรือ Swagger\*\*

4. \*\*เชื่อมต่อจาก frontend (Angular)\*\*

5. \*\*ทดสอบ flow จริง\*\*

---

\*\*สรุป:\*\*

- เริ่มที่ backend: สร้าง entity, service, migration, logic

- จากนั้นค่อยเชื่อมต่อ frontend

- ทุกอย่างเกี่ยวกับ token/log ต้องเริ่มที่ backend ก่อน

ถ้าต้องการตัวอย่าง entity/service หรือไฟล์ migration แจ้งได้เลย!

pethzero: backend.2 สร้าง Service/Repository สำหรับบันทึก token และ log

สร้าง service ใน auth

ตัวอย่าง:

auth.service.ts (เพิ่ม logic save token/log)

auth.controller.ts (endpoint login/logout)

1.3 เพิ่ม logic ใน service

หลัง login สำเร็จ → save token ลง user\_tokens, save log ลง login\_logs

หลัง logout → revoke token (update revoked = true)

ทุกครั้งที่ login (สำเร็จ/ล้มเหลว) → save log ขอตัวอย่าง

GitHub Copilot: ตัวอย่างการเพิ่ม logic \*\*บันทึก token และ log\*\* ใน auth.service.ts และ auth.controller.ts

(สมมุติใช้ TypeORM และมี entity `UserToken`, `LoginLog` แล้ว)

---

### 1. สร้าง entity (ถ้ายังไม่มี)

```typescript

import { Entity, PrimaryGeneratedColumn, Column, ManyToOne, CreateDateColumn } from 'typeorm';

import { User } from '../../database/entities/user.entity';

@Entity('user\_tokens')

export class UserToken {

@PrimaryGeneratedColumn('uuid')

id: string;

@ManyToOne(() => User)

user: User;

@Column()

token: string;

@CreateDateColumn()

created\_at: Date;

@Column({ type: 'timestamp', nullable: true })

expired\_at: Date;

@Column({ default: false })

revoked: boolean;

}

```

```typescript

import { Entity, PrimaryGeneratedColumn, Column, CreateDateColumn, ManyToOne } from 'typeorm';

import { User } from '../../database/entities/user.entity';

@Entity('login\_logs')

export class LoginLog {

@PrimaryGeneratedColumn()

id: number;

@ManyToOne(() => User, { nullable: true })

user: User;

@CreateDateColumn()

login\_time: Date;

@Column({ length: 45, nullable: true })

ip\_address: string;

@Column({ type: 'text', nullable: true })

user\_agent: string;

@Column({ default: false })

success: boolean;

@Column({ type: 'text', nullable: true })

fail\_reason: string;

}

```

---

### 2. เพิ่มใน auth.module.ts

```typescript

import { UserToken } from './entities/user-token.entity';

import { LoginLog } from './entities/login-log.entity';

TypeOrmModule.forFeature([User, UserToken, LoginLog]),

```

---

### 3. เพิ่ม logic ใน auth.service.ts

```typescript

import { UserToken } from './entities/user-token.entity';

import { LoginLog } from './entities/login-log.entity';

import { Repository } from 'typeorm';

import { InjectRepository } from '@nestjs/typeorm';

// ...ใน constructor...

constructor(

// ...เดิม...

@InjectRepository(UserToken) private userTokenRepo: Repository<UserToken>,

@InjectRepository(LoginLog) private loginLogRepo: Repository<LoginLog>,

// ...

) {}

// บันทึก log ทุกครั้งที่ login (สำเร็จ/ล้มเหลว)

async logLogin(user: User | null, success: boolean, req: any, failReason?: string) {

await this.loginLogRepo.save({

user: user || null,

ip\_address: req.ip,

user\_agent: req.headers['user-agent'],

success,

fail\_reason: failReason || null,

});

}

// หลัง login สำเร็จ

async saveToken(user: User, token: string, expiredAt: Date) {

await this.userTokenRepo.save({

user,

token,

expired\_at: expiredAt,

revoked: false,

});

}

// logout

async revokeToken(token: string) {

await this.userTokenRepo.update({ token }, { revoked: true });

}

```

---

### 4. ใช้ใน auth.controller.ts

```typescript

@Post('login')

async login(@Body() body: LoginDto, @Req() req: Request, @Res() res: Response) {

const user = await this.authService.validateUser(body.username, body.password);

if (!user) {

await this.authService.logLogin(null, false, req, 'Invalid credentials');

return res.status(401).json({ message: 'Invalid credentials' });

}

const token = this.authService.generateJwt({

sub: user.id,

username: user.username,

role: user.role,

});

// บันทึก token

const expiredAt = new Date(Date.now() + 24 \* 60 \* 60 \* 1000); // ตัวอย่าง 1 วัน

await this.authService.saveToken(user, token, expiredAt);

// บันทึก log สำเร็จ

await this.authService.logLogin(user, true, req);

res.cookie('token', token, {

httpOnly: true,

secure: false,

sameSite: 'strict',

});

return res.json({ access\_token: token });

}

@Post('logout')

async logout(@Req() req: Request, @Res() res: Response) {

const token = req.cookies['token'];

if (token) {

await this.authService.revokeToken(token);

}

res.clearCookie('token');

return res.json({ message: 'Logged out' });

}

```

---

\*\*สรุป:\*\*

- หลัง login สำเร็จ: save token, log login success

- หลัง login fail: log login fail

- หลัง logout: revoke token

- ทุกครั้งที่ login/logout มีการบันทึก log/token ตามมาตรฐาน

\*\*สามารถนำไปปรับใช้กับไฟล์ใน # ได้ทันที\*\*

pethzero:

src/modules/auth/auth.service.ts:66:7 - error TS2769: No overload matches this call.

Overload 1 of 4, '(entities: DeepPartial<LoginLog>[], options?: SaveOptions | undefined): Promise<(DeepPartial<LoginLog> & LoginLog)[]>', gave the following error.

Object literal may only specify known properties, and 'user' does not exist in type 'DeepPartial<LoginLog>[]'.

66 user: user || null,

~~~~

[01:22:06] Found 1 error. Watching for file changes.

GitHub Copilot: ปัญหา

- error: `'user' does not exist in type 'DeepPartial<LoginLog>[]'`

- สาเหตุ: ใน entity `LoginLog` ฟิลด์ user อาจไม่ได้ประกาศเป็น property หรือ typeORM mapping ไม่ถูกต้อง

---

### วิธีแก้ไข

\*\*1. ตรวจสอบ entity `LoginLog`\*\*

ต้องมี property `user` และต้องใช้ `@ManyToOne` แบบนี้:

````typescript

import { Entity, PrimaryGeneratedColumn, Column, CreateDateColumn, ManyToOne } from 'typeorm';

import { User } from './user.entity';

@Entity('login\_logs')

export class LoginLog {

@PrimaryGeneratedColumn()

id: number;

@ManyToOne(() => User, { nullable: true })

user: User;

@CreateDateColumn()

login\_time: Date;

@Column({ length: 45, nullable: true })

ip\_address: string;

@Column({ type: 'text', nullable: true })

user\_agent: string;

@Column({ default: false })

success: boolean;

@Column({ type: 'text', nullable: true })

fail\_reason: string;

}

````

\*\*2. ถ้า user เป็น userId (uuid) ให้ใช้ user\_id แทน\*\*

ถ้า entity ของคุณใช้ `user\_id: string` (ไม่ใช่ relation) ให้เปลี่ยนใน save เป็น

```typescript

await this.loginLogRepo.save({

user: user ? user.id : null,

ip\_address: req.ip,

user\_agent: req.headers['user-agent'],

success,

fail\_reason: failReason || null,

});

```

และใน entity:

```typescript

@Column({ type: 'uuid', nullable: true })

user: string;

```

\*\*3. ถ้าใช้ relation (แบบที่แนะนำ) ให้แน่ใจว่า import User entity ถูกต้อง\*\*

---

### สรุป

- ตรวจสอบว่า entity `LoginLog` มี property `user` และเป็น relation หรือไม่

- ถ้าเป็น relation ใช้ `@ManyToOne(() => User, ...) user: User`

- ถ้าเป็นแค่ user\_id ให้ใช้ `user: user ? user.id : null`

\*\*หลังแก้ไขนี้ error จะหาย\*\*

ถ้ายัง error แนบ entity `login-log.entity.ts` มาได้เลยครับ

pethzero: CREATE TABLE IF NOT EXISTS public.user\_tokens

(

id uuid NOT NULL DEFAULT gen\_random\_uuid(),

user\_id uuid,

token text COLLATE pg\_catalog."default" NOT NULL,

created\_at timestamp with time zone DEFAULT now(),

expired\_at timestamp with time zone,

revoked boolean DEFAULT false,

CONSTRAINT user\_tokens\_pkey PRIMARY KEY (id)

);

CREATE TABLE IF NOT EXISTS public.login\_logs

(

id serial NOT NULL,

user\_id uuid,

login\_time timestamp with time zone DEFAULT now(),

ip\_address character varying(45) COLLATE pg\_catalog."default",

user\_agent text COLLATE pg\_catalog."default",

success boolean,

fail\_reason text COLLATE pg\_catalog."default",

CONSTRAINT login\_logs\_pkey PRIMARY KEY (id)

);

GitHub Copilot: \*\*สาเหตุของ error:\*\*

- ใน entity `LoginLog` ฟิลด์ user เป็น relation (`user: User`)

- แต่ใน SQL table `login\_logs` ใช้ชื่อ column ว่า `user\_id`

- TypeORM จะ map relation เป็น `userId` โดยอัตโนมัติ ถ้าไม่ได้ระบุชื่อ column

- ถ้าไม่ระบุชื่อ column ด้วย `@JoinColumn({ name: 'user\_id' })` จะเกิด mapping mismatch

---

## วิธีแก้ไขให้ตรงกับ schema

\*\*1. เพิ่ม @JoinColumn ใน entity\*\*

````typescript

import { Entity, PrimaryGeneratedColumn, Column, CreateDateColumn, ManyToOne, JoinColumn } from 'typeorm';

import { User } from './user.entity';

@Entity('login\_logs')

export class LoginLog {

@PrimaryGeneratedColumn()

id: number;

@ManyToOne(() => User, { nullable: true })

@JoinColumn({ name: 'user\_id' }) // <-- เพิ่มบรรทัดนี้

user: User;

@CreateDateColumn()

login\_time: Date;

@Column({ length: 45, nullable: true })

ip\_address: string;

@Column({ type: 'text', nullable: true })

user\_agent: string;

@Column({ default: false })

success: boolean;

@Column({ type: 'text', nullable: true })

fail\_reason: string;

}

````

\*\*2. เวลา save log ให้ส่ง user object (ไม่ใช่ user.id)\*\*

```typescript

await this.loginLogRepo.save({

user: user || null,

ip\_address: req.ip,

user\_agent: req.headers['user-agent'],

success,

fail\_reason: failReason || null,

});

```

---

\*\*สรุป:\*\*

- เพิ่ม `@JoinColumn({ name: 'user\_id' })` ที่ relation

- ส่ง user object ตอน save log

- แบบนี้จะ mapping ตรงกับ schema ที่ใช้ `user\_id` ในฐานข้อมูล

\*\*หลังแก้ไขนี้ error จะหาย และ log จะถูกบันทึกถูกต้อง\*\*

pethzero: [01:24:56] File change detected. Starting incremental compilation...

src/modules/auth/auth.service.ts:66:7 - error TS2769: No overload matches this call.

Overload 1 of 4, '(entities: DeepPartial<LoginLog>[], options?: SaveOptions | undefined): Promise<(DeepPartial<LoginLog> & LoginLog)[]>', gave the following error.

Object literal may only specify known properties, and 'user' does not exist in type 'DeepPartial<LoginLog>[]'.

66 user: user || null,

~~~~

GitHub Copilot: ปัญหา

- error: `'user' does not exist in type 'DeepPartial<LoginLog>[]'`

- สาเหตุ: TypeORM ไม่รู้จัก property `user` ใน entity `LoginLog` (อาจ import หรือ mapping ผิด)

---

### วิธีแก้ไข

\*\*1. ตรวจสอบว่า entity `LoginLog` import `User` ถูกต้อง และมี property `user` แบบ relation\*\*

```typescript

import { Entity, PrimaryGeneratedColumn, Column, CreateDateColumn, ManyToOne, JoinColumn } from 'typeorm';

import { User } from './user.entity';

@Entity('login\_logs')

export class LoginLog {

@PrimaryGeneratedColumn()

id: number;

@ManyToOne(() => User, { nullable: true })

@JoinColumn({ name: 'user\_id' })

user: User;

@CreateDateColumn()

login\_time: Date;

@Column({ length: 45, nullable: true })

ip\_address: string;

@Column({ type: 'text', nullable: true })

user\_agent: string;

@Column({ default: false })

success: boolean;

@Column({ type: 'text', nullable: true })

fail\_reason: string;

}

```

\*\*2. ตรวจสอบว่าใน `auth.module.ts` หรือที่ใช้ TypeORM มีการ import entity นี้ถูกต้อง\*\*

```typescript

TypeOrmModule.forFeature([User, UserToken, LoginLog])

```

\*\*3. ถ้า entity ถูกต้องแล้ว ให้ลบไฟล์ build เก่า แล้ว build ใหม่\*\*

```sh

rm -rf dist

npm run build

```

\*\*4. ถ้ายัง error ให้ลองเปลี่ยนเป็น userId (string) ชั่วคราว\*\*

```typescript

// ใน entity

@Column({ type: 'uuid', nullable: true })

user\_id: string;

// ใน logLogin

await this.loginLogRepo.save({

user\_id: user ? user.id : null,

ip\_address: req.ip,

user\_agent: req.headers['user-agent'],

success,

fail\_reason: failReason || null,

});

```

---

### สรุป

- ตรวจสอบ entity `LoginLog` ว่ามี property `user` แบบ relation และ import `User` ถูกต้อง

- ถ้ายัง error ให้ใช้ `user\_id: string` แทน relation ชั่วคราว

- ลบ build เก่าแล้ว build ใหม่

\*\*ถ้ายัง error แนบ entity `login-log.entity.ts` และ `user.entity.ts` มาได้เลยครับ\*\*

pethzero: คือ เรามีทั้ง users login\_logs user\_tokens

GitHub Copilot: คุณออกแบบถูกต้องแล้วครับ!

\*\*สรุปความสัมพันธ์และการใช้งาน\*\*

---

### 1. ตาราง users

- เก็บข้อมูลผู้ใช้หลัก (id, username, passwordHash, email, role ฯลฯ)

### 2. ตาราง login\_logs

- เก็บ log ทุกครั้งที่มีการ login (สำเร็จ/ล้มเหลว)

- มีฟิลด์ user\_id (relation ไป users.id), login\_time, ip, user\_agent, success, fail\_reason

### 3. ตาราง user\_tokens

- เก็บ token ที่ออกให้ user แต่ละคน (เช่น JWT หรือ refresh token)

- มีฟิลด์ user\_id (relation ไป users.id), token, created\_at, expired\_at, revoked

---

### \*\*การใช้งานในระบบ\*\*

- \*\*เวลาผู้ใช้ login:\*\*

- ตรวจสอบ username/password จาก `users`

- ถ้าสำเร็จ → สร้าง token, save ลง `user\_tokens`, save log ลง `login\_logs` (success = true)

- ถ้า fail → save log ลง `login\_logs` (success = false, fail\_reason)

- \*\*เวลาตรวจสอบ token:\*\*

- ตรวจสอบว่า token นี้ยังอยู่ใน `user\_tokens` และ revoked = false

- \*\*เวลาผู้ใช้ logout:\*\*

- set revoked = true ใน `user\_tokens` สำหรับ token นี้

---

### \*\*ตัวอย่าง entity และการ mapping (TypeORM/NestJS)\*\*

- `User`

- `LoginLog`

- `UserToken`

\*\*ทุก entity มี relation ไป users ด้วย @ManyToOne และ @JoinColumn({ name: 'user\_id' })\*\*

---

### \*\*สรุป\*\*

- โครงสร้างนี้เป็นมาตรฐานสำหรับระบบ auth ที่ปลอดภัย

- สามารถขยายต่อยอดได้ เช่น เพิ่ม refresh token, ตรวจสอบประวัติ login, บังคับ logout ทุก device ฯลฯ

\*\*ถ้าต้องการตัวอย่าง query หรือ service เพิ่มเติม แจ้งได้เลย!\*\*

pethzero: [01:24:56] File change detected. Starting incremental compilation...

src/modules/auth/auth.service.ts:66:7 - error TS2769: No overload matches this call.

Overload 1 of 4, '(entities: DeepPartial<LoginLog>[], options?: SaveOptions | undefined): Promise<(DeepPartial<LoginLog> & LoginLog)[]>', gave the following error.

Object literal may only specify known properties, and 'user' does not exist in type 'DeepPartial<LoginLog>[]'.

66 user: user || null,

~~~~

GitHub Copilot: ปัญหา

- error: `'user' does not exist in type 'DeepPartial<LoginLog>[]'`

- สาเหตุ: TypeORM ไม่รู้จัก property `user` ใน entity `LoginLog` (อาจ import ผิด หรือ entity ไม่ตรงกับฐานข้อมูล)

---

### วิธีแก้ไข

\*\*1. ตรวจสอบ entity `LoginLog` ว่ามี property `user` แบบ relation และ import `User` ถูกต้อง\*\*

```typescript

import { Entity, PrimaryGeneratedColumn, Column, CreateDateColumn, ManyToOne, JoinColumn } from 'typeorm';

import { User } from './user.entity';

@Entity('login\_logs')

export class LoginLog {

@PrimaryGeneratedColumn()

id: number;

@ManyToOne(() => User, { nullable: true })

@JoinColumn({ name: 'user\_id' })

user: User;

@CreateDateColumn()

login\_time: Date;

@Column({ length: 45, nullable: true })

ip\_address: string;

@Column({ type: 'text', nullable: true })

user\_agent: string;

@Column({ default: false })

success: boolean;

@Column({ type: 'text', nullable: true })

fail\_reason: string;

}

```

\*\*2. ตรวจสอบว่าใน `auth.module.ts` หรือที่ใช้ TypeORM มีการ import entity นี้ถูกต้อง\*\*

```typescript

TypeOrmModule.forFeature([User, UserToken, LoginLog])

```

\*\*3. ลบไฟล์ build เก่า แล้ว build ใหม่\*\*

```sh

rm -rf dist

npm run build

```

\*\*4. ถ้ายัง error ให้ลองใช้ userId แทน relation ชั่วคราว\*\*

```typescript

// ใน entity

@Column({ type: 'uuid', nullable: true })

user\_id: string;

// ใน logLogin

await this.loginLogRepo.save({

user\_id: user ? user.id : null,

ip\_address: req.ip,

user\_agent: req.headers['user-agent'],

success,

fail\_reason: failReason || null,

});

```

---

### สรุป

- ตรวจสอบ entity `LoginLog` ว่ามี property `user` แบบ relation และ import `User` ถูกต้อง

- ถ้ายัง error ให้ใช้ `user\_id: string` แทน relation ชั่วคราว

- ลบ build เก่าแล้ว build ใหม่

\*\*หลังแก้ไขนี้ error จะหาย\*\*

ถ้ายัง error แนบ entity `login-log.entity.ts` และ `user.entity.ts` มาได้เลยครับ

pethzero:

src/modules/auth/auth.service.ts:65:26 - error TS2769: No overload matches this call.

Overload 1 of 4, '(entities: DeepPartial<LoginLog>[], options?: SaveOptions | undefined): Promise<(DeepPartial<LoginLog> & LoginLog)[]>', gave the following error.

Object literal may only specify known properties, and 'user\_id' does not exist in type 'DeepPartial<LoginLog>[]'.

Overload 2 of 4, '(entity: DeepPartial<LoginLog>, options?: SaveOptions | undefined): Promise<DeepPartial<LoginLog> & LoginLog>', gave the following error.

Type 'string | null' is not assignable to type 'string | undefined'.

Type 'null' is not assignable to type 'string | undefined'.

65 await this.loginLogRepo.save({

GitHub Copilot: ปัญหา

- error: `'user\_id' does not exist in type 'DeepPartial<LoginLog>'`

- สาเหตุ: entity `LoginLog` ใช้ relation เป็น `user: User` (ไม่ใช่ `user\_id: string`)

- และ TypeORM ไม่ยอมรับ `null` สำหรับ relation ถ้า type เป็น `string | undefined`

---

### วิธีแก้ไข (แบบ relation)

\*\*ใน entity `LoginLog`\*\*

```typescript

@ManyToOne(() => User, { nullable: true })

@JoinColumn({ name: 'user\_id' })

user: User;

```

\*\*ใน service ให้ส่ง user object หรือ undefined (ไม่ใช่ user\_id หรือ null)\*\*

```typescript

async logLogin(user: User | null, success: boolean, req: any, failReason?: string) {

await this.loginLogRepo.save({

user: user || undefined, // ใช้ undefined แทน null

ip\_address: req.ip,

user\_agent: req.headers['user-agent'],

success,

fail\_reason: failReason || undefined,

});

}

```

---

### สรุป

- ต้องส่ง `user: User | undefined` ไม่ใช่ `user\_id`

- ถ้าไม่มี user ให้ใช้ `undefined` (ไม่ใช่ null)

- แบบนี้ TypeORM จะ map ไปที่ `user\_id` ในฐานข้อมูลให้อัตโนมัติ

\*\*หลังแก้ไขนี้ error จะหาย\*\*

pethzero: async saveToken(user: User, token: string, expiredAt: Date) {

await this.userTokenRepo.save({

user,

token,

expired\_at: expiredAt,

revoked: false,

});

}

// logout

async revokeToken(token: string) {

await this.userTokenRepo.update({ token }, { revoked: true });

}

}

แค่ save ใช่ไหม

GitHub Copilot: ใช่ครับ!

ฟังก์ชัน `saveToken` ที่คุณเขียนนี้ \*\*เพียงพอสำหรับการบันทึก token ลงตาราง user\_tokens\*\* ด้วย TypeORM

โดยจะสร้าง record ใหม่ทุกครั้งที่ login สำเร็จ

```typescript

async saveToken(user: User, token: string, expiredAt: Date) {

await this.userTokenRepo.save({

user, // relation ไป users

token, // JWT หรือ refresh token

expired\_at: expiredAt,

revoked: false,

});

}

```

\*\*หมายเหตุ:\*\*

- ถ้า user เดิม login หลายรอบ จะมีหลาย token ในตาราง (เหมาะกับระบบ multi-device)

- ถ้าอยากให้ 1 user มี token เดียว ให้ query หา token เดิมแล้ว update/revoke ก่อน save ใหม่

\*\*ส่วน revokeToken\*\*

- ใช้สำหรับอัปเดต revoked = true ตอน logout

\*\*สรุป:\*\*

- ใช้ save แบบนี้ถูกต้องแล้ว

- ถ้าต้องการ logic เพิ่มเติม (เช่น revoke token เก่าก่อน save ใหม่) สามารถเพิ่มได้ภายหลัง

pethzero: ทำไหมต้อง บันทึก token ด้วย

GitHub Copilot: \*\*เหตุผลที่ต้องบันทึก token ลงฐานข้อมูล (user\_tokens):\*\*

1. \*\*ควบคุมการใช้งาน token:\*\*

- สามารถ revoke (ยกเลิก) token ได้ เช่น ตอน logout, เปลี่ยนรหัสผ่าน, หรือ admin ต้องการบังคับ logout ทุก device

- ตรวจสอบว่า token ที่ client ส่งมา ยังถูกต้องและไม่ถูกยกเลิก

2. \*\*รองรับ multi-device:\*\*

- 1 user อาจ login ได้หลายอุปกรณ์ (แต่ละอุปกรณ์มี token ของตัวเอง)

- สามารถจัดการ token แต่ละอันแยกกันได้

3. \*\*เพิ่มความปลอดภัย:\*\*

- ป้องกันการใช้ token เก่า (replay attack)

- ตรวจสอบประวัติการใช้งานย้อนหลังได้

4. \*\*Audit & Monitoring:\*\*

- ตรวจสอบว่า user เคย login จากที่ไหน เมื่อไร ใช้ token อะไร

5. \*\*รองรับ refresh token หรือ session management:\*\*

- ถ้าต้องการระบบ refresh token หรือ session แบบละเอียด จำเป็นต้องเก็บ token ในฐานข้อมูล

---

\*\*สรุป:\*\*

การบันทึก token ช่วยให้ระบบปลอดภัย ควบคุม session ได้ละเอียด และตรวจสอบย้อนหลังได้

เหมาะกับระบบที่ต้องการความปลอดภัยสูงและรองรับ multi-device/multi-session

pethzero: ควบคุมการใช้งาน token:

สามารถ revoke (ยกเลิก) token ได้ เช่น ตอน logout, เปลี่ยนรหัสผ่าน, หรือ admin ต้องการบังคับ logout ทุก device

ตรวจสอบว่า token ที่ client ส่งมา ยังถูกต้องและไม่ถูกยกเลิก

รองรับ multi-device:

1 user อาจ login ได้หลายอุปกรณ์ (แต่ละอุปกรณ์มี token ของตัวเอง)

สามารถจัดการ token แต่ละอันแยกกันได้

เพิ่มความปลอดภัย:

ป้องกันการใช้ token เก่า (replay attack)

ตรวจสอบประวัติการใช้งานย้อนหลังได้

Audit & Monitoring:

ตรวจสอบว่า user เคย login จากที่ไหน เมื่อไร ใช้ token อะไร

รองรับ refresh token หรือ session management:

ถ้าต้องการระบบ refresh token หรือ session แบบละเอียด จำเป็นต้องเก็บ token ในฐานข้อมูล

ใน5 ที่กล่าวมา ต้องสร้างอะไรเพิ่ตรงไหนไหม ที่หน้าบ้าน หรือหลังบ้าน

GitHub Copilot: \*\*ทั้ง 5 ข้อที่กล่าวมา\*\*

> - ควบคุมการใช้งาน token

> - รองรับ multi-device

> - เพิ่มความปลอดภัย

> - Audit & Monitoring

> - รองรับ refresh token หรือ session management

\*\*ทั้งหมดนี้ "ต้องทำที่หลังบ้าน (backend)" เป็นหลัก\*\*

โดยเฉพาะ logic ที่เกี่ยวกับการบันทึก, ตรวจสอบ, และ revoke token

\*\*หน้าบ้าน (frontend)\*\* แค่เก็บ/ส่ง token และแสดงผลลัพธ์ตามที่ backend ตอบกลับ

---

### สิ่งที่ต้องสร้าง/เพิ่มใน "หลังบ้าน (backend)"

1. \*\*Entity/ตาราง user\_tokens\*\*

- มีแล้ว (ตามที่คุณสร้างไว้)

2. \*\*Service/Repository สำหรับบันทึกและตรวจสอบ token\*\*

- มีแล้วใน auth.service.ts (`saveToken`, `revokeToken`)

3. \*\*Logic ตรวจสอบ token ทุกครั้งที่มี request\*\*

- ใน JWT strategy หรือ middleware ต้องเช็คว่า token ที่ client ส่งมา

- ยังอยู่ใน user\_tokens

- revoked = false

- (option) expired\_at > now

4. \*\*เพิ่ม endpoint สำหรับ admin revoke ทุก token ของ user\*\*

- เช่น `/api/auth/revoke-all/:userId`

5. \*\*(ถ้าต้องการ refresh token)\*\*

- เพิ่มฟิลด์หรือ table สำหรับ refresh token

- เพิ่ม logic ออก refresh token, ตรวจสอบ, และ revoke

---

### สิ่งที่ต้องทำใน "หน้าบ้าน (frontend)"

- เก็บ token ที่ได้จาก backend (localStorage/cookie)

- ส่ง token ไปกับ request (ผ่าน interceptor)

- ถ้า backend ตอบว่า token หมดอายุ/ถูก revoke → logout อัตโนมัติ

---

### สรุป

- \*\*ทุก logic สำคัญเกี่ยวกับ token, revoke, audit, multi-device, refresh token\*\*

\*\*ต้องทำที่ backend\*\*

- \*\*frontend\*\* แค่เก็บ/ส่ง token และแสดงผลลัพธ์

---

\*\*ถ้าต้องการตัวอย่างโค้ด middleware ตรวจสอบ token หรือ endpoint สำหรับ admin revoke ทุก token แจ้งได้เลย!\*\*

# revoke-all

revoke-all ใน context นี้คือ endpoint สำหรับ "ยกเลิก token ทั้งหมดของ user"

(ใช้สำหรับ admin หรือระบบที่ต้องการบังคับ logout ทุก session ของ user เป้าหมาย)

ตัวอย่างการใช้งาน

URL:

POST /auth/revoke-all/:userId

เช่น:

POST /auth/revoke-all/123e4567-e89b-12d3-a456-426614174000

ผลลัพธ์:

Token ทั้งหมดของ user ที่มี id ตรงกับ userId จะถูก set revoked = true ในฐานข้อมูล

(user ทุก device จะถูกบังคับ logout)

สรุป

revoke-all = "ยกเลิก token ทั้งหมด" ของ user ที่ระบุ

ใช้สำหรับ admin หรือระบบที่ต้องการควบคุม session ของ user

ปลอดภัย เหมาะกับระบบที่ต้องการบังคับ logout ทุกอุปกรณ์/ทุก session ทันที