โค้ดที่ให้มาเป็นโครงสร้างที่ค่อนข้างถูกต้องแล้วครับ โดยคุณได้ตั้งค่า CSRF Token และ JWT (สำหรับ session) ไว้ในตัวอย่าง แต่ยังมีบางส่วนที่อาจจะต้องเพิ่มเติมหรือปรับให้ชัดเจนขึ้นเล็กน้อยเพื่อให้มันทำงานได้อย่างสมบูรณ์ ดังนี้:

**หลักการของการใช้ CSRF Token กับ JWT**

1. **Login (Login Endpoint)**
   * เมื่อผู้ใช้ทำการ login, ระบบจะทำการตรวจสอบ username และ password.
   * หากข้อมูลถูกต้อง ระบบจะสร้าง **JWT** แล้วส่งกลับไปใน cookie และ **CSRF Token** ผ่าน response JSON.
2. **ขอ CSRF Token (Get CSRF Token Endpoint)**
   * หลังจากผู้ใช้ login เสร็จ จะต้องเรียก API /csrf-token เพื่อขอ **CSRF Token** ไปใช้ใน request ต่อไปที่มีการป้องกัน CSRF (เช่น POST, PUT, DELETE).
   * CSRF Token จะถูกใช้ในการป้องกัน Cross-Site Request Forgery โดยต้องส่งผ่าน header X-CSRF-Token ในการส่งข้อมูลที่มีการแก้ไขหรือมีผลกระทบต่อระบบ.
3. **ใช้ CSRF Token กับ API อื่นๆ**
   * เมื่อเรียก API อื่นๆ ที่มีการเปลี่ยนแปลงข้อมูล เช่น POST, PUT, DELETE จะต้องส่ง CSRF Token ใน X-CSRF-Token header ร่วมกับการส่ง JWT ใน cookie หรือ Authorization header.

**อธิบายโค้ดที่ให้มา**

**1. Login Endpoint**

@Post('login')

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

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

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

const payload = { sub: user.id, username: user.username, role: user.role };

const token = this.authService.generateJwt(payload);

res.cookie('token', token, {

httpOnly: true, // เพิ่มความปลอดภัยไม่ให้สามารถอ่าน cookie ได้จาก JavaScript

secure: false, // ใช้ `true` เมื่อใช้ https

sameSite: 'strict', // ป้องกัน CSRF ข้ามเว็บไซต์

});

// ส่ง CSRF token กลับมา

if (req.csrfToken) {

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

}

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

}

* **Login:** ตรวจสอบข้อมูลผู้ใช้ และส่ง JWT ไปเก็บใน cookie พร้อมตั้งค่าความปลอดภัย (เช่น httpOnly, sameSite).
* **CSRF Token:** หลังจาก login สำเร็จ จะส่ง csrfToken กลับไปใน response ถ้าหาก middleware CSRF (เช่น csurf) ถูกตั้งค่าไว้อย่างถูกต้อง.

**2. CSRF Token Endpoint**

@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

}

}

* **CSRF Token:** เมื่อผู้ใช้ login แล้ว จะสามารถเรียก endpoint นี้เพื่อรับ **CSRF Token** ที่จะใช้ในการป้องกัน CSRF ในการทำ request ต่อไป.

**การติดตั้งและตั้งค่า CSRF Middleware**

หากยังไม่ได้ติดตั้ง middleware สำหรับ CSRF (เช่น csurf) คุณสามารถติดตั้งและ config ได้ตามขั้นตอนนี้:

**1. ติดตั้ง csurf**

npm install csurf

**2. เพิ่ม CSRF Middleware ใน main.ts หรือ app.module.ts**

import \* as csurf from 'csurf';

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

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

async function bootstrap() {

const app = await NestFactory.create(AppModule);

// เพิ่ม CSRF middleware

app.use(csurf());

await app.listen(3000);

}

bootstrap();

**3. ตั้งค่า Cookie Parser (ถ้ายังไม่ได้ทำ)**

คุณอาจต้องใช้ cookie-parser เพื่อให้สามารถจัดการกับ cookies ใน NestJS:

npm install cookie-parser

จากนั้นเพิ่มใน main.ts:

import \* as cookieParser from 'cookie-parser';

async function bootstrap() {

const app = await NestFactory.create(AppModule);

// เพิ่ม cookie-parser เพื่อให้สามารถใช้งาน cookie ได้

app.use(cookieParser());

await app.listen(3000);

}

**ขั้นตอนการใช้งานจากฝั่ง Frontend**

1. **Login:**
   * ทำการส่ง POST request ไปที่ /auth/login พร้อมกับข้อมูล username และ password.
   * เมื่อ login สำเร็จ, คุณจะได้รับ JWT ใน cookie และ CSRF Token ใน response.
2. **ขอ CSRF Token:**
   * หลังจาก login เสร็จเรียบร้อยแล้ว, ให้ทำการส่ง GET request ไปที่ /auth/csrf-token เพื่อขอ csrfToken จากเซิร์ฟเวอร์.
3. **ใช้ CSRF Token:**
   * เมื่อมีการส่ง POST หรือ request ที่มีการเปลี่ยนแปลงข้อมูล, ให้ส่ง csrfToken ที่ได้จาก /auth/csrf-token ใน X-CSRF-Token header ร่วมกับ JWT ที่อยู่ใน cookie.

**สรุปขั้นตอนการใช้ CSRF Token**

1. **Login**: ผู้ใช้ทำการ login ผ่าน /auth/login, เซิร์ฟเวอร์จะตอบกลับพร้อมกับ JWT ใน cookie และ CSRF Token.
2. **ขอ CSRF Token**: หลังจาก login เสร็จ, เรียก /auth/csrf-token เพื่อรับ csrfToken ที่จะใช้ในการป้องกัน CSRF ใน request ถัดไป.
3. **ส่งข้อมูล**: ทุกครั้งที่ส่งข้อมูลที่มีการเปลี่ยนแปลง (เช่น POST, PUT, DELETE), คุณต้องแนบ X-CSRF-Token header พร้อม csrfToken และ JWT ใน cookie.

ถ้าในอนาคตมีคำถามเพิ่มเติมเกี่ยวกับการตั้งค่าหรือปัญหาที่เจอ, สามารถถามเพิ่มเติมได้เลยครับ!