**Nest JS:**

**My first Project:**

Create new project.

**Nest new project-name:**

Run the project.

Manually: **[ npm run start ] [ nest start]**

Automatically: **npm run start:dev** this automatically fetch changes.

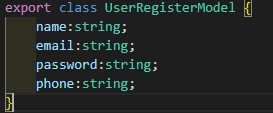
**New Module with all resources:**

Run this command for create a new module.

**Nest g resource folder-name/module-name**

**DTO file:**

DTO is just like a model file.



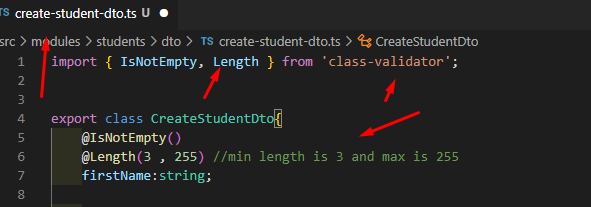
**Validations:**

**Class-validator:**

Install class-validator by using command.

Npm I class-validator

After install it we can use validators method in our **DTO** files like.

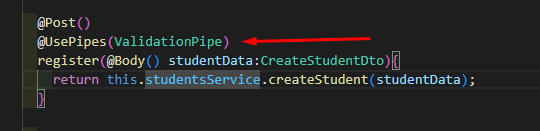


**How to use Validations:**

For use the validations we need to use **@UsePipes(ValidationPipe)** in our controller.

But before this we must need to install **class-transformer** by using command

npm i class-transformer



**Entity:**

**Note: this is use for create database, table in database like mysql etc**

Before we start we need to install some packages.

**Typeorm:**

Npm I –save typeorm mysql

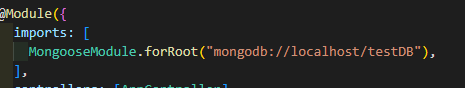
**Connection to Mongo DB:**

**Install Mongoos Packages:**

npm i @nestjs/mongoose mongoose

**Import in app.module.ts.**

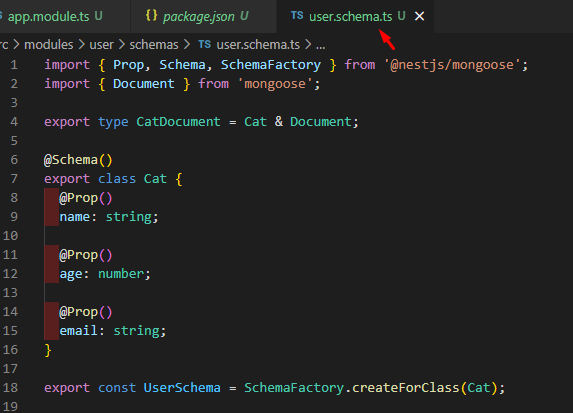
Import mongoos in app.module.ts file like.



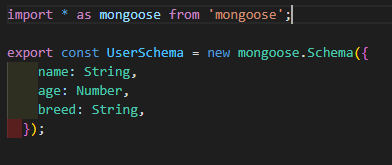
**Create Schema:**

With Mongoose, everything is derived from a Schema. Each schema maps to a MongoDB collection and defines the shape of the documents within that collection. Schemas are used to define Models. Models are responsible for creating and reading documents from the underlying MongoDB database.

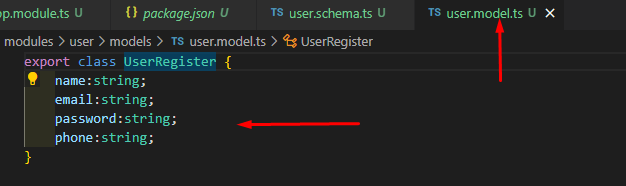
Schemas can be created with NestJS decorators, or with Mongoose itself manually. Using decorators to create schemas greatly reduces boilerplate and improves overall code readability.



Alternative code for above code is.

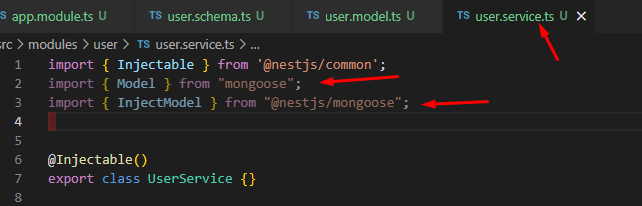


**Create a Model:**

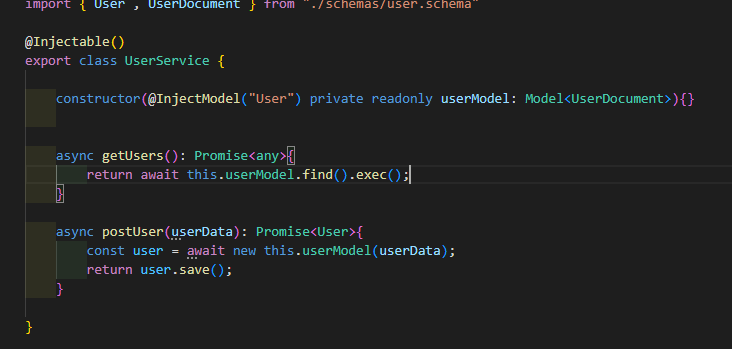


**In Service File:**

Import Model and InjectModel in service file.

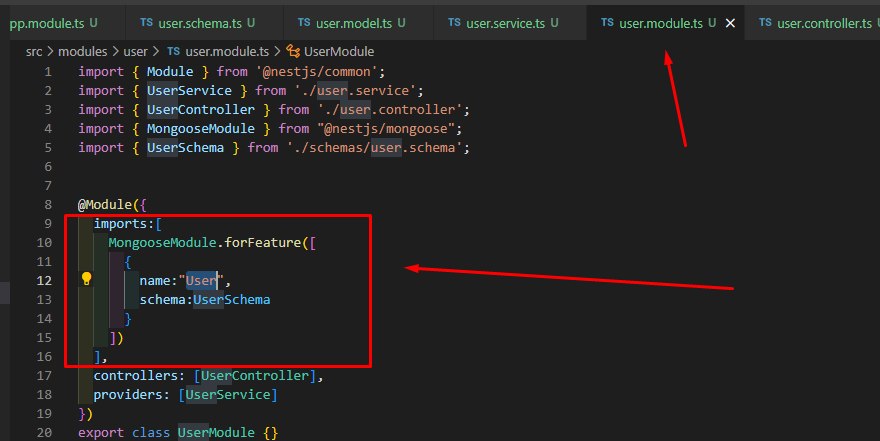


Add constructor and functions.



Here **“User”** is the module name

**Now import it with user.module.ts file.**

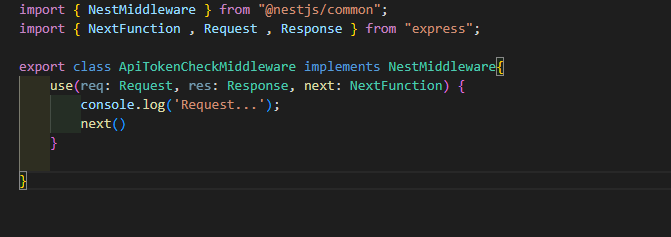


**Middleware:**

**Create new middleware:**

We can create middleware in anywhere in our app but best way is to create token in middleware folder inside src folder.

Here is the basic structure of middleware.



We can see middleware implements the class **NestMiddleware** from @nestjs/common.

**Next() Function:**

Middleware works in a pipe mean if you have three middleware like

1 first

2 second

3 third

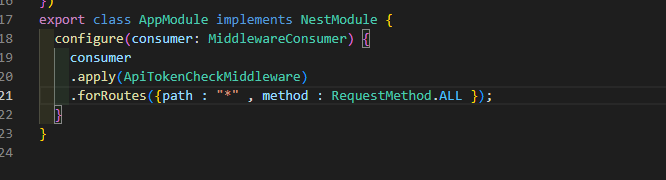
Then these three middleware execute in a pipe if there is no error in first middleware then they move to next mean second middleware by **next()** function.

**Where to use Middleware?**

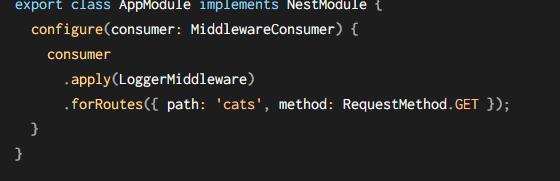
In **app.module.ts** file we implement the middleware like.

In Image **.apply(**middleware name**)** and **forRoutes()** is the routes where you want to implement middleware and **RequestMethod.All** mean on all methods like (**get , post , delete , update ,etc**).

**How to implements on all Routes:**



**How to implements on specific Route:**



**Password Hash(encrypt):**

For this we need to install some packages.

Npm I bcrypt

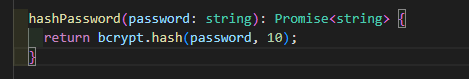
Npm I -D @types/bcrypt

**Import it:**

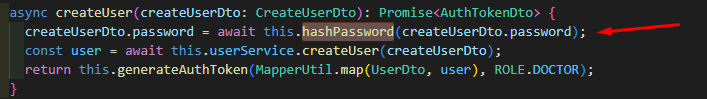
Import \* as bcrypt from ‘bcrypt’;

In required controller or service file.

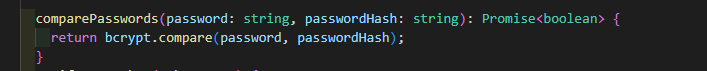
**Create function for hash password:**



**Use this function:**



**Decrypt or dehash Password funtion:**



**Passport & JWT:**

For authentication we need to install some packages.

npm install --save @nestjs/passport passport passport-local

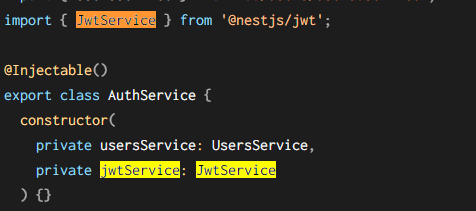
npm install --save-dev @types/passport-local

**packages for JWT:**

npm install --save @nestjs/jwt passport-jwt

npm install --save-dev @types/passport-jwt

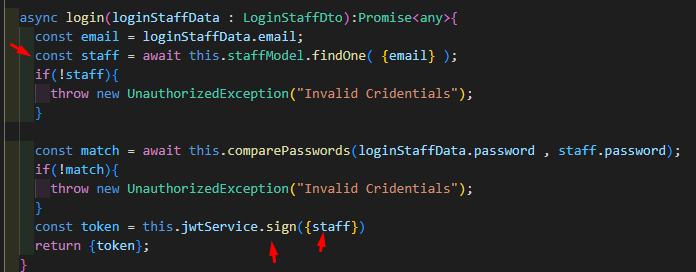
After install these packages we need to import **jwtService** in our service file.



**Generate JWT token:**

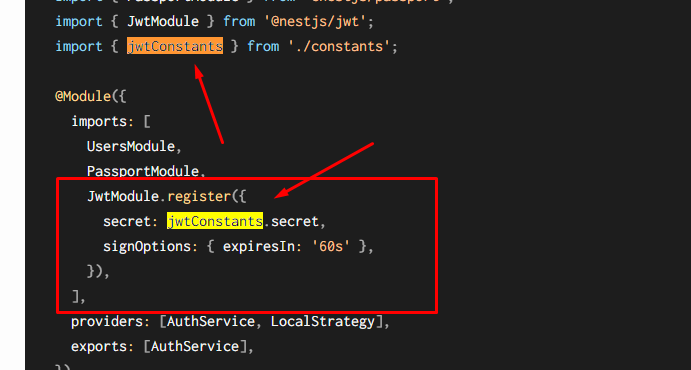
For generate jwt token we use **sign()** method of **jwtService.**

In Image we can see **staff** basically staff is the values or data that we want to store in jwt token.



**Import JWTModule in require module.**

Now we need to import JWT in required module where we use JWT service. Basically in Auth Module.

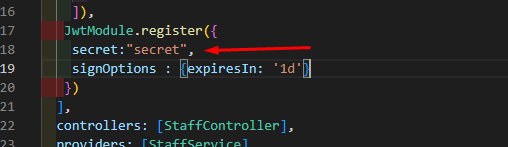


**Constants mean:**

Constants basically is a key that we use to create a JWT token.



**We can also write it directly.**



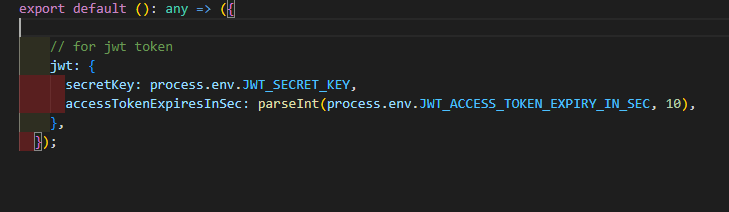
**Another method for import JWT in Auth Module:**

**Config Service:**

We can import jwt module in auth module by using config service but first we need to install it.

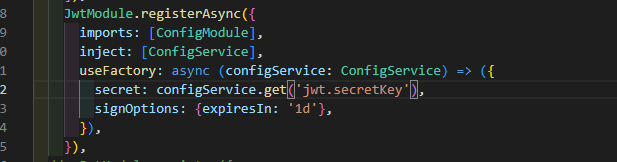
8u0

**Create app.config.ts file in src/core folder.**



Here we can see we write our secret key and expiry time from **.env** file.

**Now use this in Auth Module.**

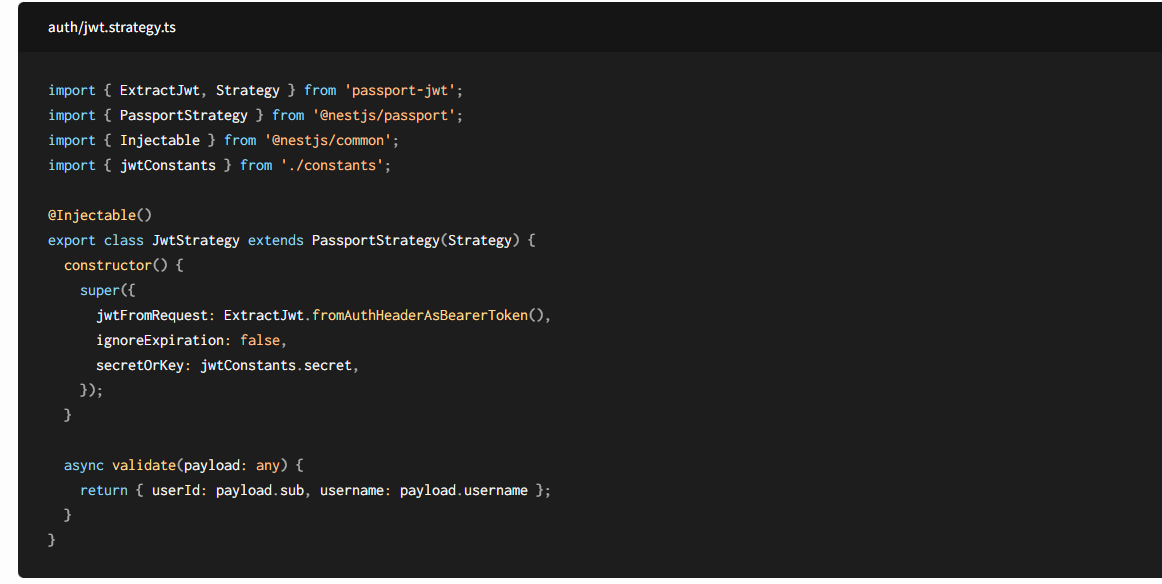


**Authenticate user by using passport.**

How to authenticate user by using **passport in nestjs?**

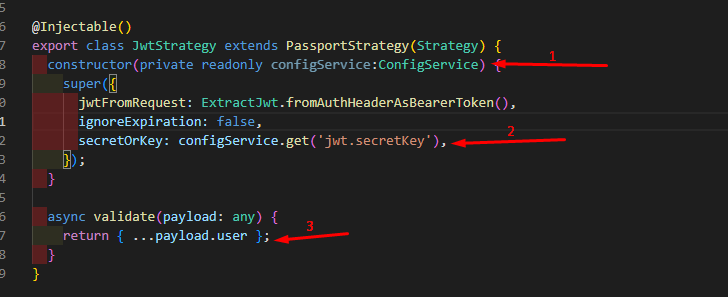
**Create jwt.strategy.ts file in auth\_module/strategy.**

<https://docs.nestjs.com/security/authentication> link of complete tutorial.



From above link we can see the structure of file. So we will use same code.

**Our file code is:**

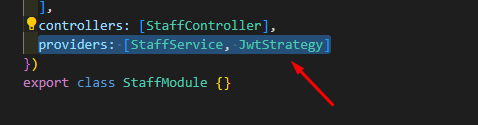


1: import configService file.

2: Use our secret key that we create in app.config.ts file.

3: Return payload.

**Add this class in auth.module.ts:**



**JWT Auth Guard:**

https://docs.nestjs.com/security/authentication

**Create new file in auth\_module/guard.**

From tutorial.



**How to use Auth Guard:**

Just add @UseGuards(JwtAuthGuard) in controller request where you want to add restrictions.

