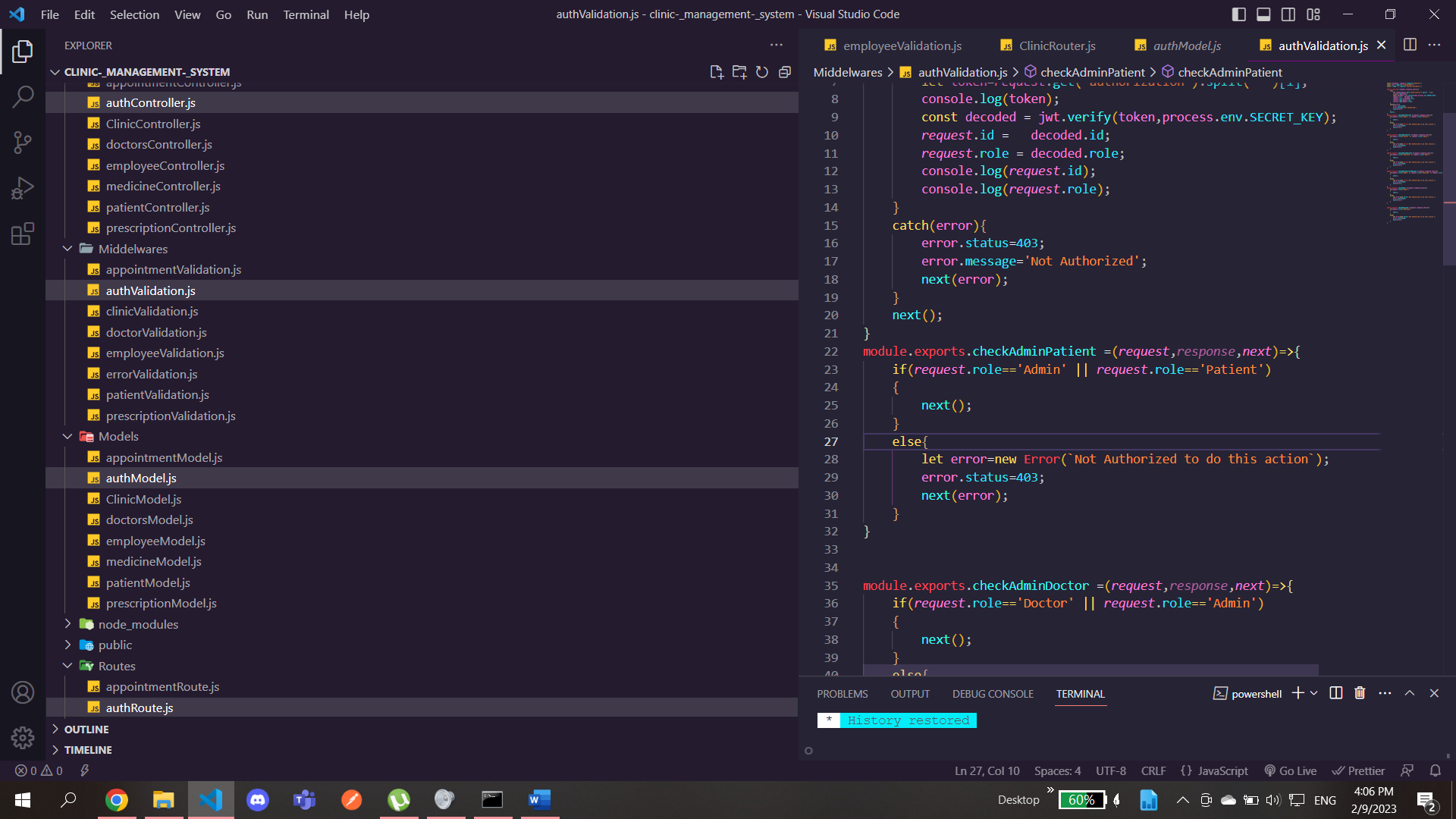
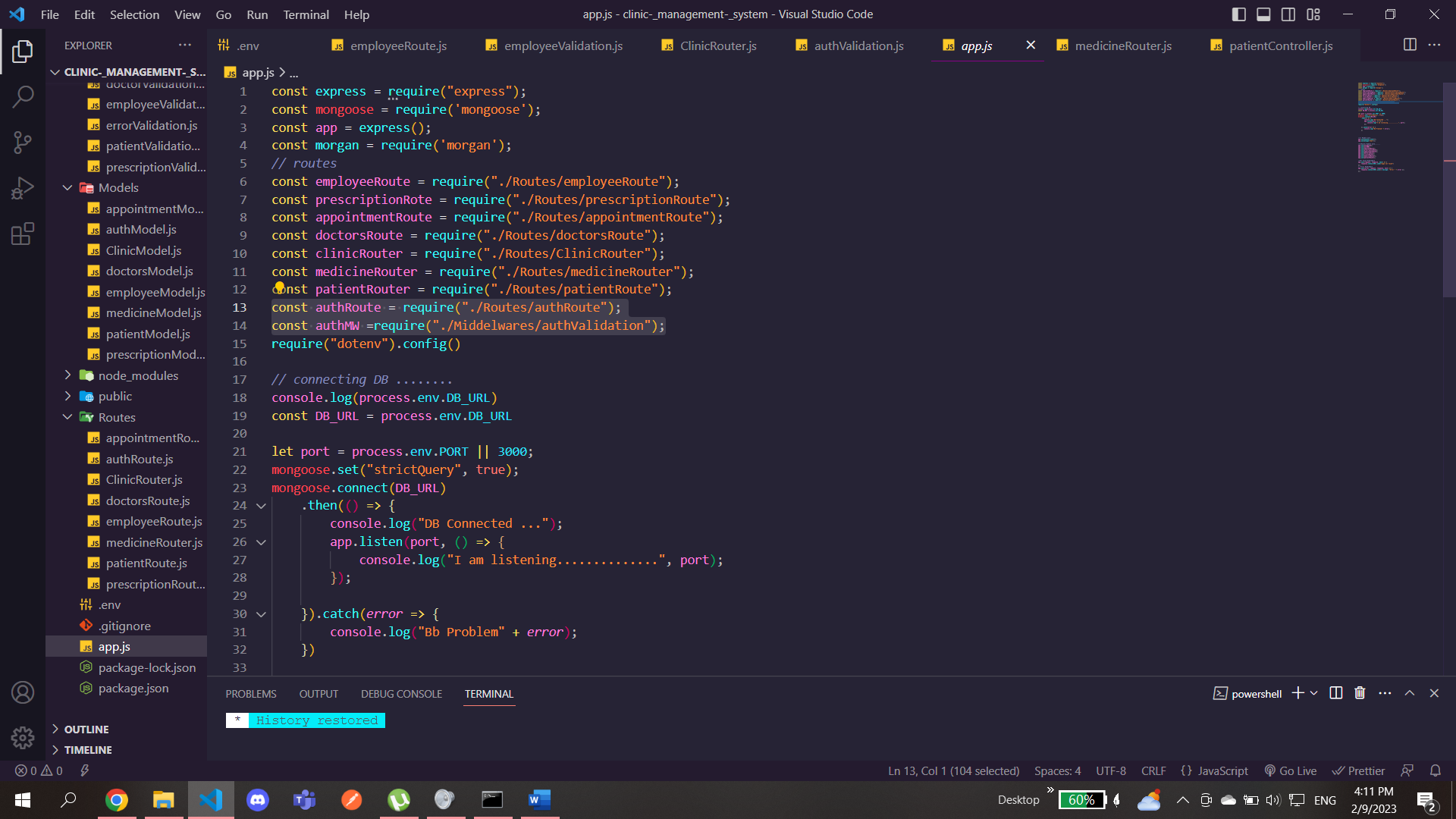
Authentication and Authorization

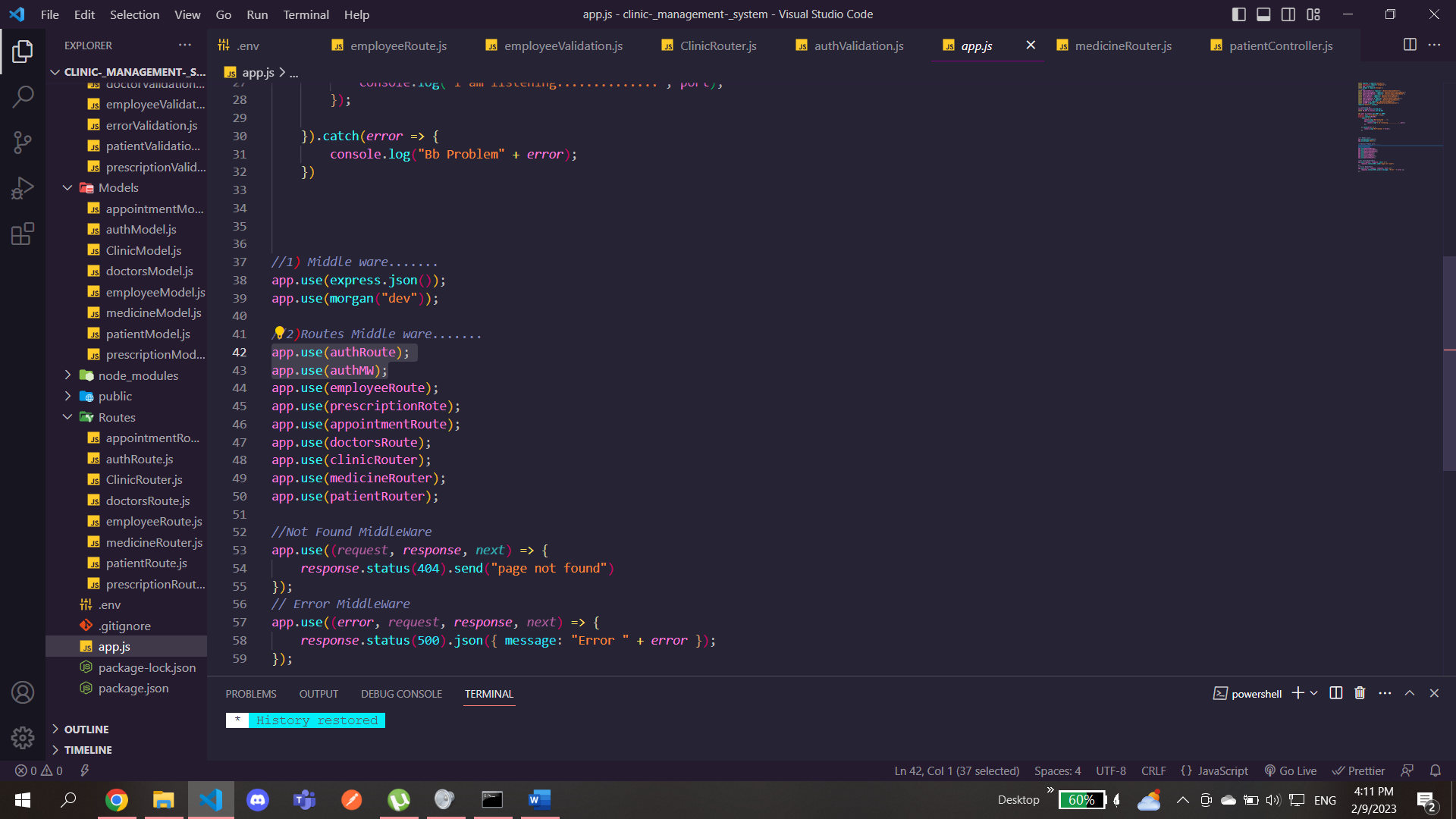
First step is the creation of the whole process files including the schema, controller, middleware and route and of course don’t forget to install the packages

[bcrypt for hashing and salting the password, jsonwebtoken for creating a token for login]



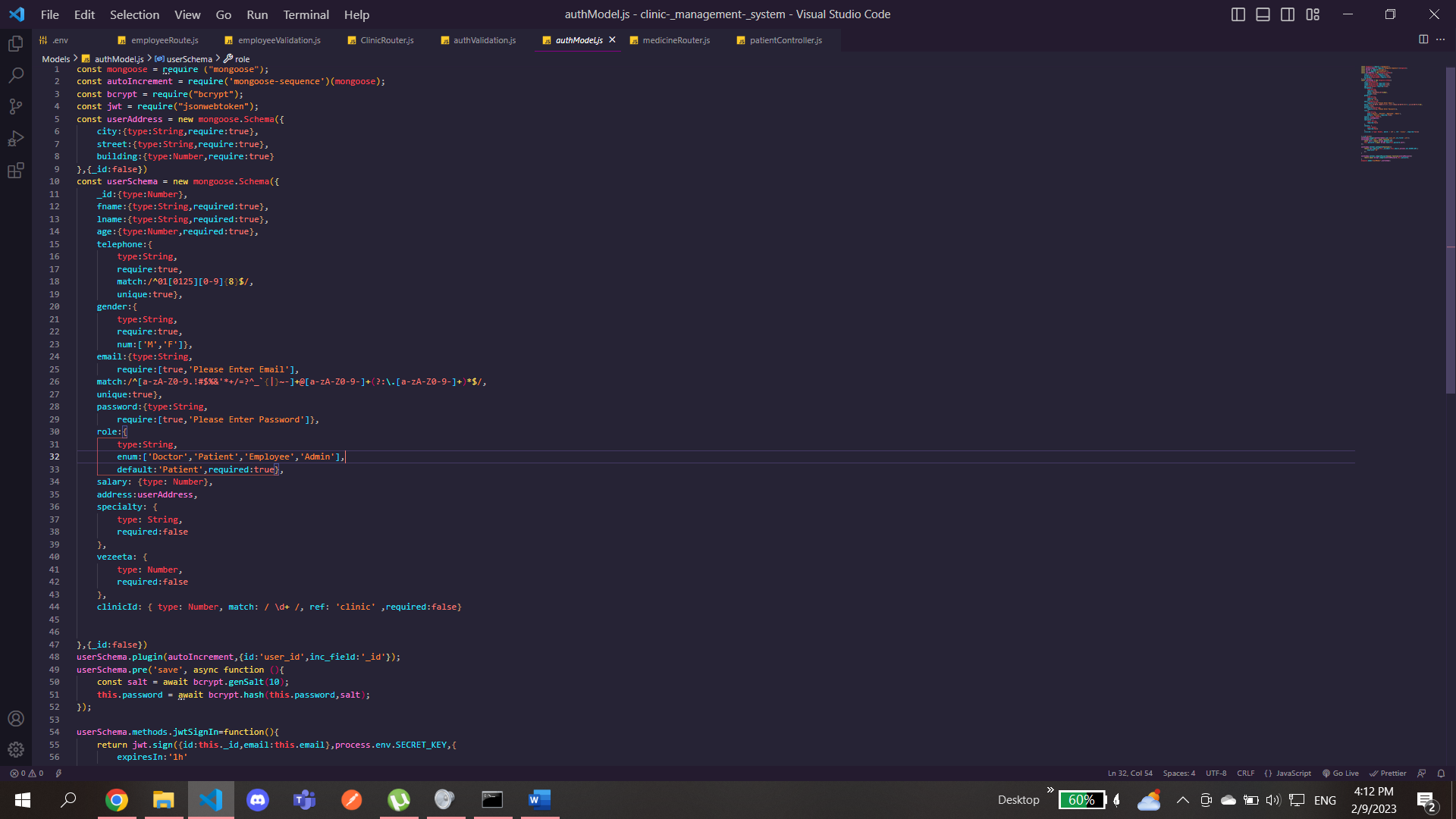
Secondly we import router and the middleware files in the app.js which runs the server



And make the server use them 

Never forget to place the authroute before the middleware or there will be some issues during the runtime by this we have locked the authorization for all users except for the ones we are going to give in the middleware

Start with the schema file as it will inflict on all the other files

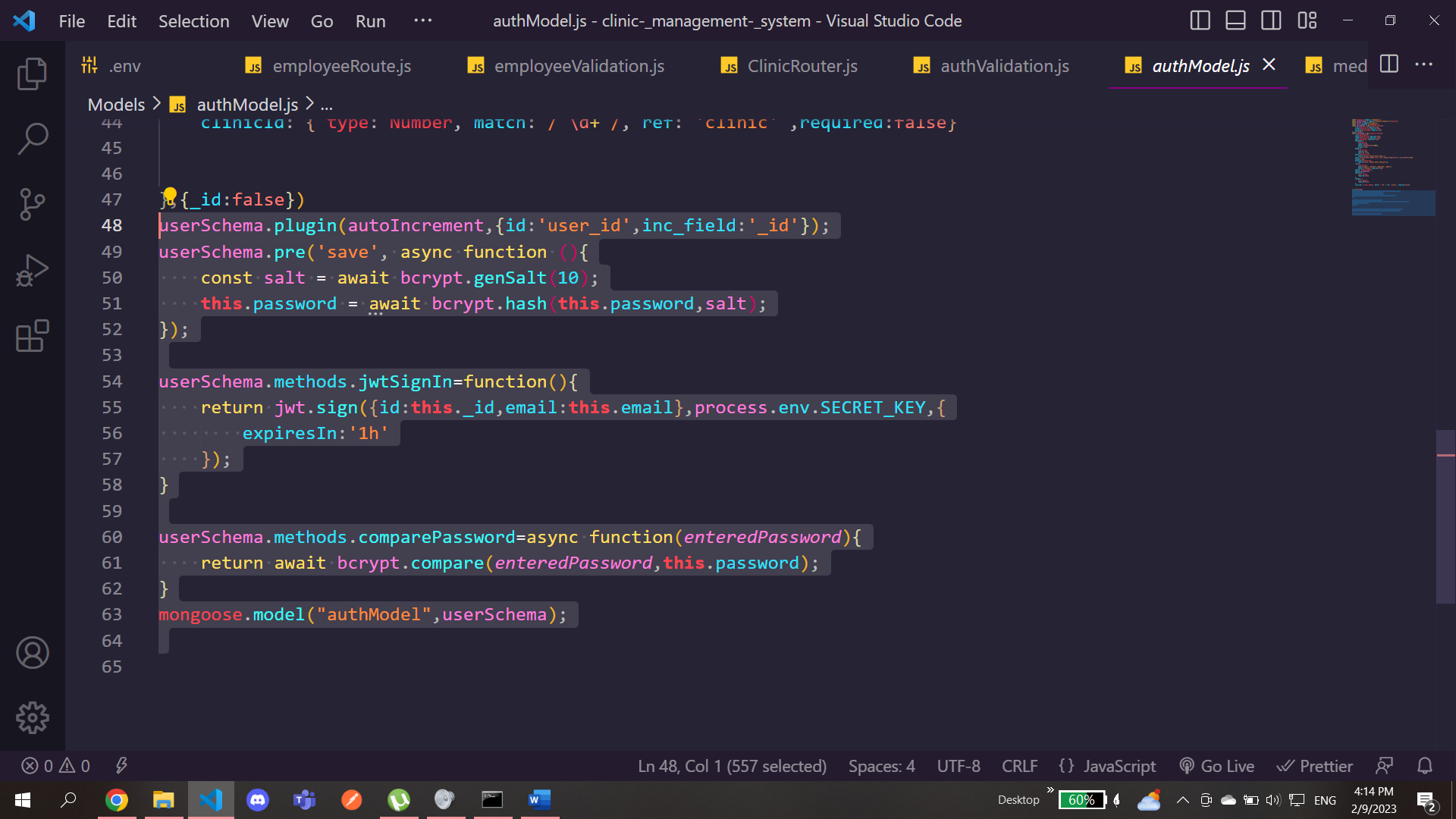


As we see we created the fields for the registration process and also include the jsonwebtoken and the bcrypt packages

Now we create a function for hashing the password which is called pre hitting the save button in postman and also we added to the hash salting to enhance the security

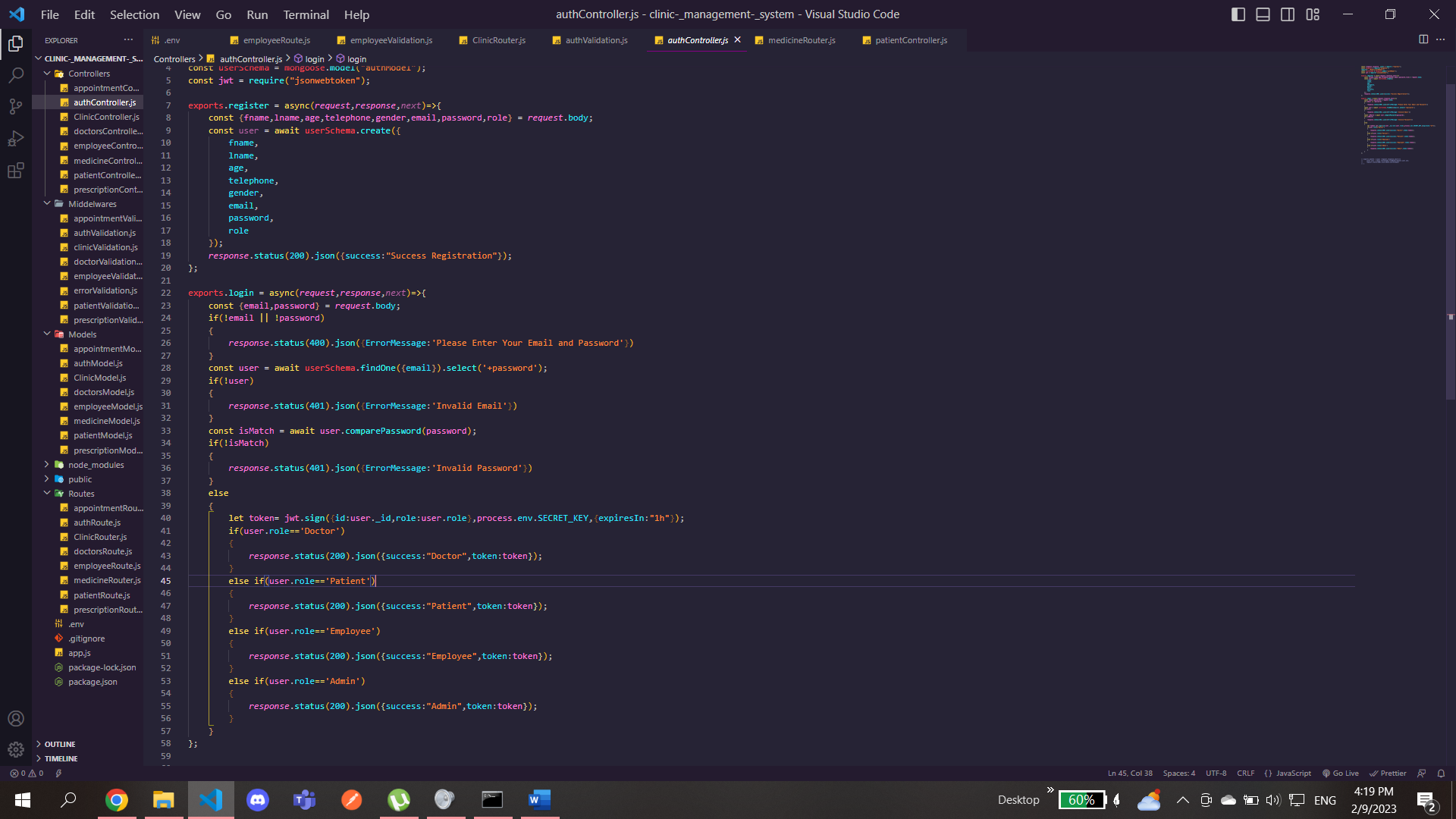
After that we create the sign in method using the jsonwebtoken package which will create a token for login with the email and post them in the console of postman

And also create a method to compare the password you typed if it is the same as the one you registered with



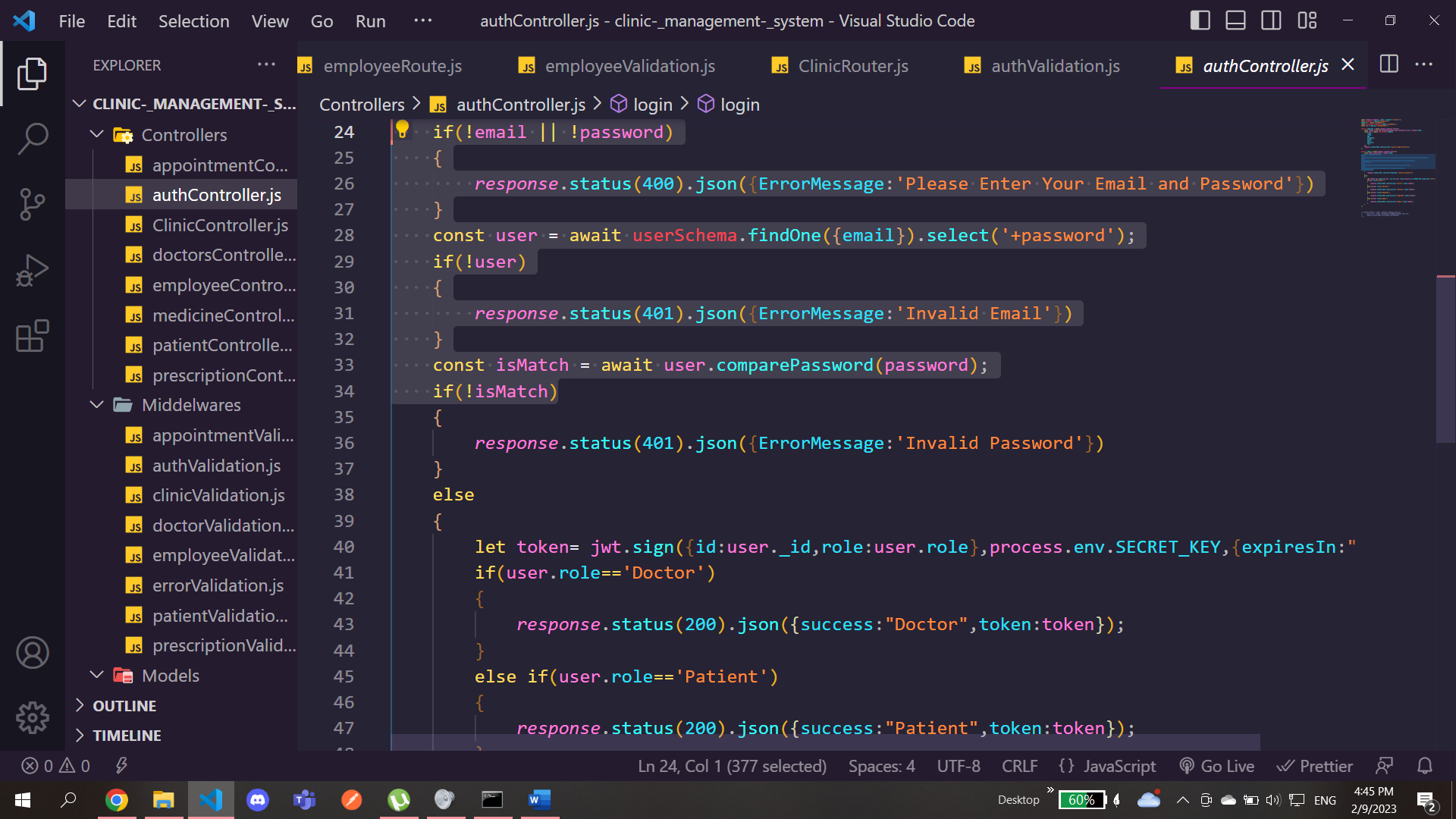
As you can see in the controller file in the register function we destructed the fields of the schema and made it request.body just for a shortcut so you don’t have to type request.body.fieldname each time of course we imported the schema file in here

Then we create all these fields using create method and will be saved in authModels collection



In the login function we compared the values of the email and the password to see if they match the entered one or not by importing the compare method from the schema we can compare the used password and the registered one due to the encryption process we’ve done that

For the email we don’t need the compare function just find the email you typed in if you found it in the registered collection then its ok

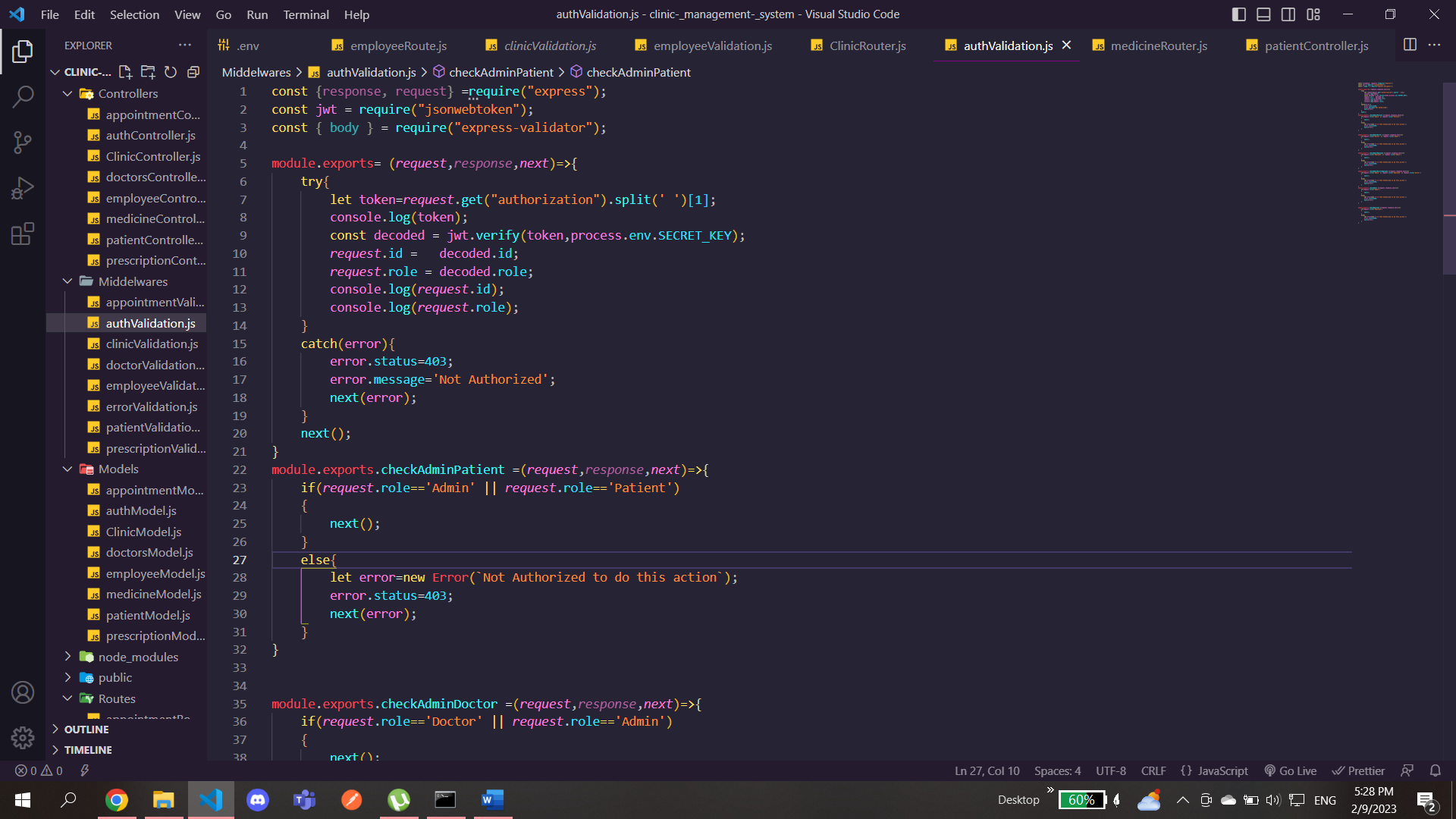


Then if all is valid we check for the role of the entered email and password to see if it’s a doctor, patient, employee or admin this brings us to the middleware folder

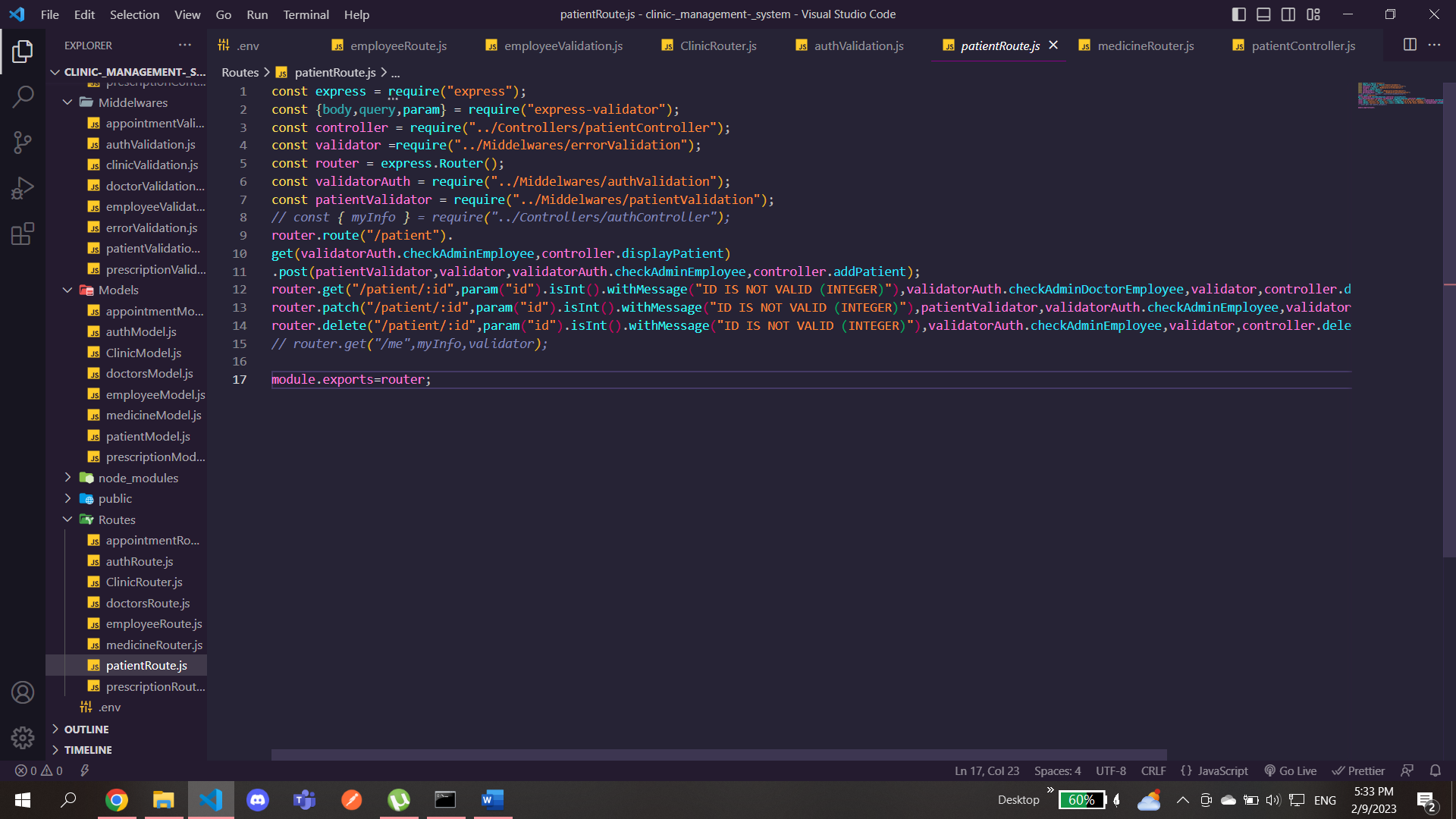
As you can see here In the middleware we created a function which takes the bearer token and separates the space in it and collects it in the array so we can extract the token only without the bearer word

Then use verify from jsonwebtoken to decode the encrypted login token to see if it is the same as the signed in one and from it find the id and the role of it

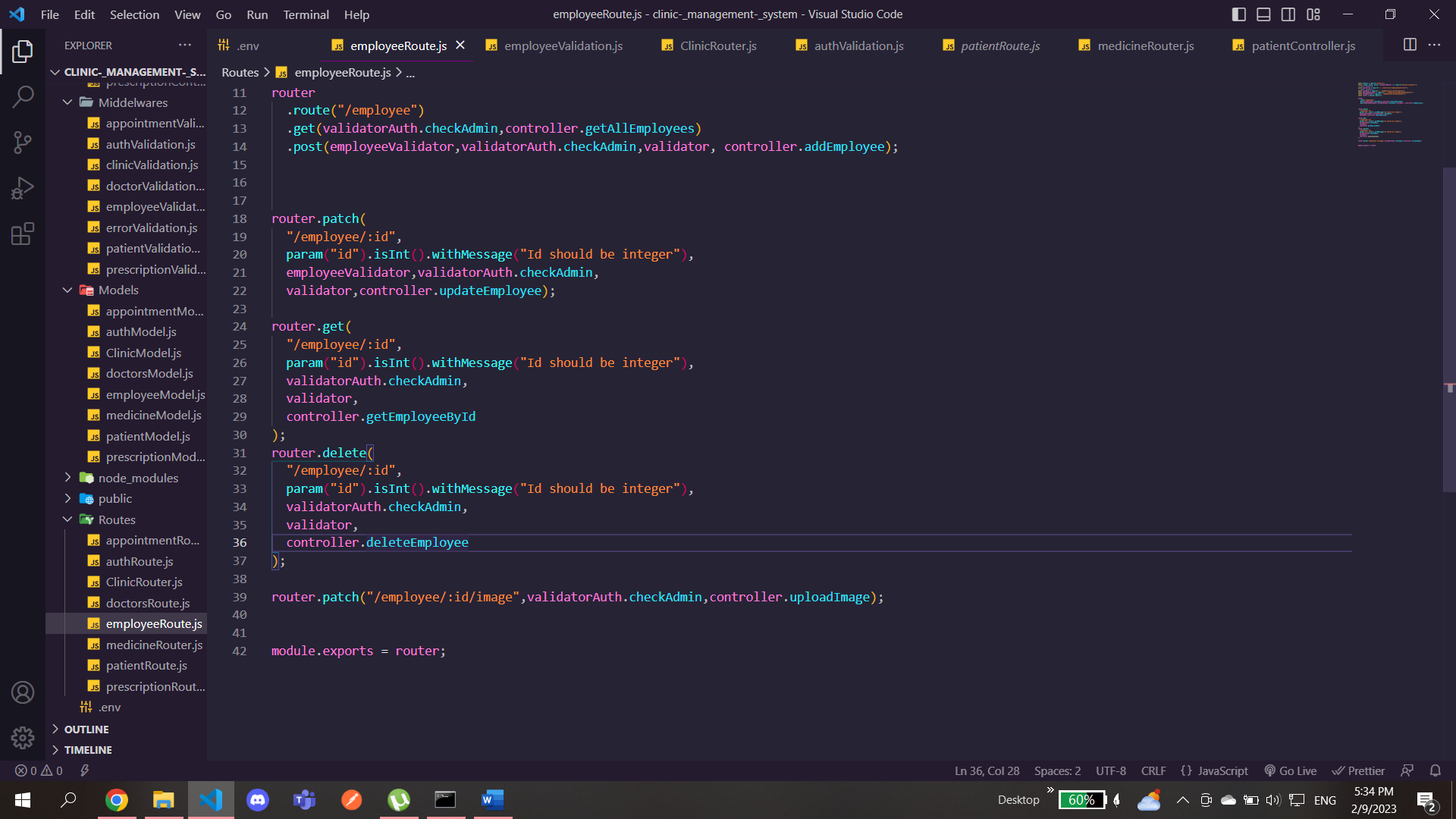
After this we create some functions in order to find the role of the user and whether he is patient, doctor, admin or employee

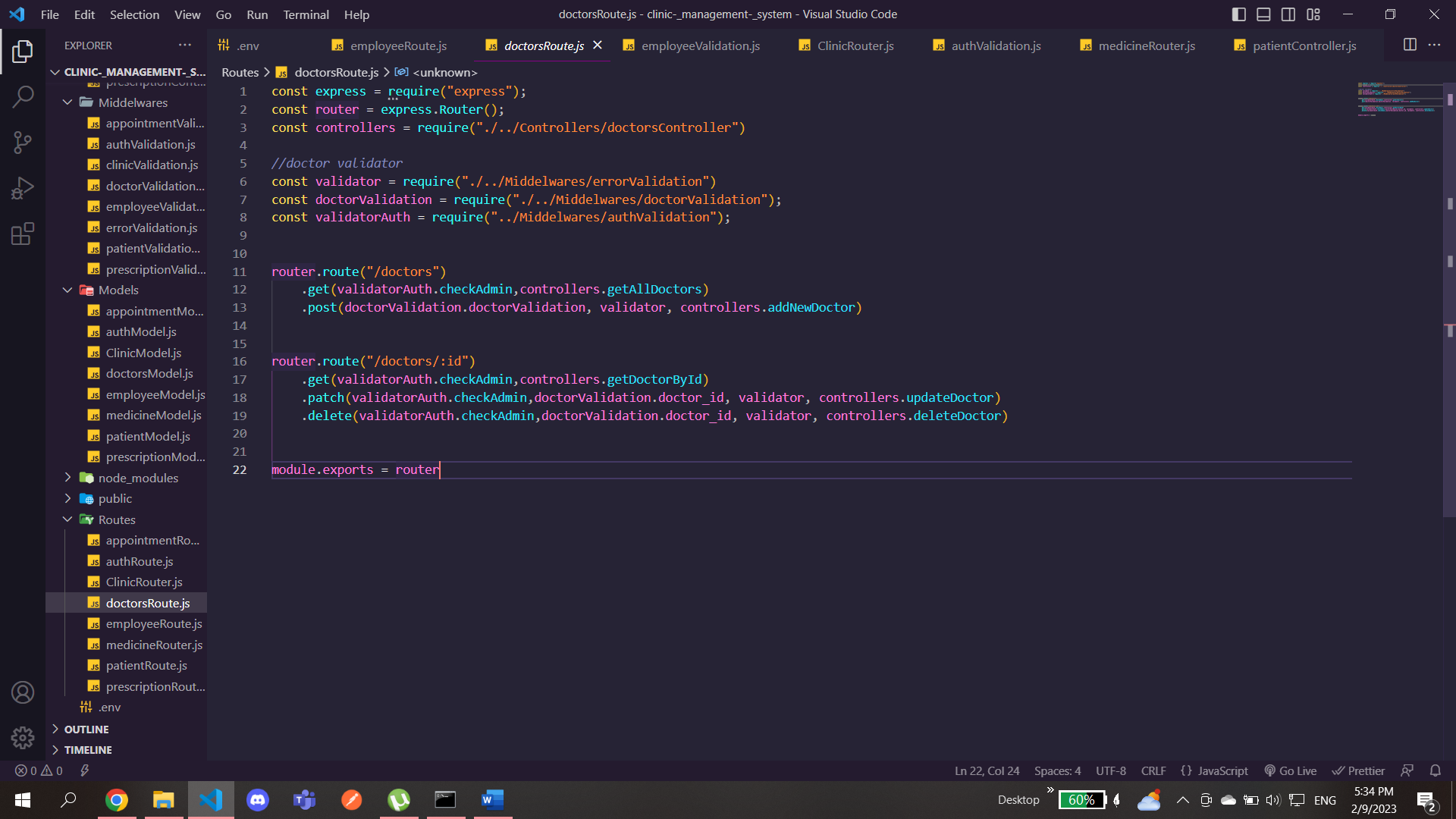


After this we just need to import the middleware in each router of the users and give him the needed function for authorization



For the above example here we have given authorization for the admin or employee to display all the patients





in the above example we’ve only given the admin the authorization to get, delete or update the employee or the doctor