**JWT Authentication with Spring Boot**

**Step 1: Using default username and password by generating console. Create Spring Boot Sample Application**

* **Create End Point to be secured**

**HomeController**A screenshot of a computer

AI-generated content may be incorrect.

**UserService**A screenshot of a computer

AI-generated content may be incorrect.

**User**A screenshot of a computer

AI-generated content may be incorrect.

**Response:**

A close up of a text

AI-generated content may be incorrect.

* **For Security purpose, we are using Spring security.  
  Add dependency,**A screenshot of a computer

  AI-generated content may be incorrect.
* **Rerun the project then your API will get secure.**

**Request:** [**http://localhost:8085/home/user**](http://localhost:8085/home/user) **Response:** A screenshot of a computer

AI-generated content may be incorrect.

* **Now we need to Sign in And We are getting password by console.**

**A screenshot of a computer

AI-generated content may be incorrect.  
Password: bd754765-e3c2-422b-924a-0ec84436a469**

**And By default, Username: User  
  
A screenshot of a login box

AI-generated content may be incorrect.  
Then We can login and access the API.**

* **Finally, we can login Successfully.**A screenshot of a computer error message

  AI-generated content may be incorrect.
* **Now we can access the Api.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Then We are going to Logout.**

A screenshot of a computer

AI-generated content may be incorrect.

* **After logged out, we are unable to access the API without using username and password.**

**A screenshot of a computer

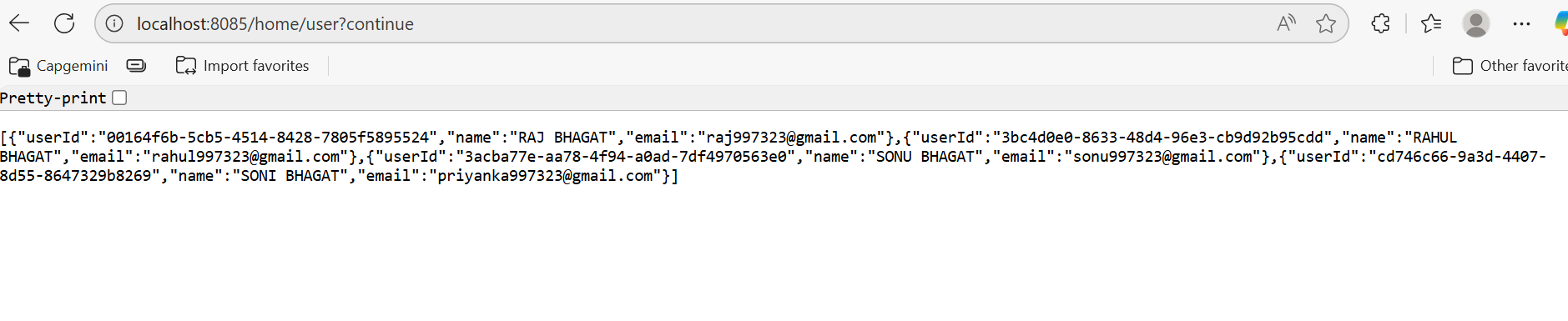
AI-generated content may be incorrect.**

**Step2: Using Properties file**

* **How to use customise Username and password. We need to add username and password in properties file.**A screenshot of a computer

  AI-generated content may be incorrect.
* **Now Rerun the application and Api by default will use the properties username and Password. And for now, we will not get password into console.**A screenshot of a computer

  AI-generated content may be incorrect.
* **Now we need to login again by default username and password using properties file.**A screenshot of a login box

  AI-generated content may be incorrect.
* **Login successful by using username: Raj And Password : Raj123**

**Step 3: Create InMemory user with UserDetailService Bean**

* **Create UserDetailService bean and write the InMemory user implementation**
* **Create CustomConfig class and create bean and also create two important bean PasswordEncoder and AuthenticationManager so that we can use later.**
* **First Need to remove username and password from properties file**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Create Configuration file And make a UserDetailsService as a Bean and then configure the user And We have to create 2 more Bean for PasswordEncoder and Authenticationmanager  
    
  A screenshot of a computer

  AI-generated content may be incorrect.**
* **For Password Encoder,**

****

* **Now Restart the application and we can login with given username and password by default spring security provide form login .  
  A screenshot of a computer

  AI-generated content may be incorrect.  
    
  A screenshot of a login box

  AI-generated content may be incorrect.**
* **We can login Successfully…**

**A screenshot of a computer error page

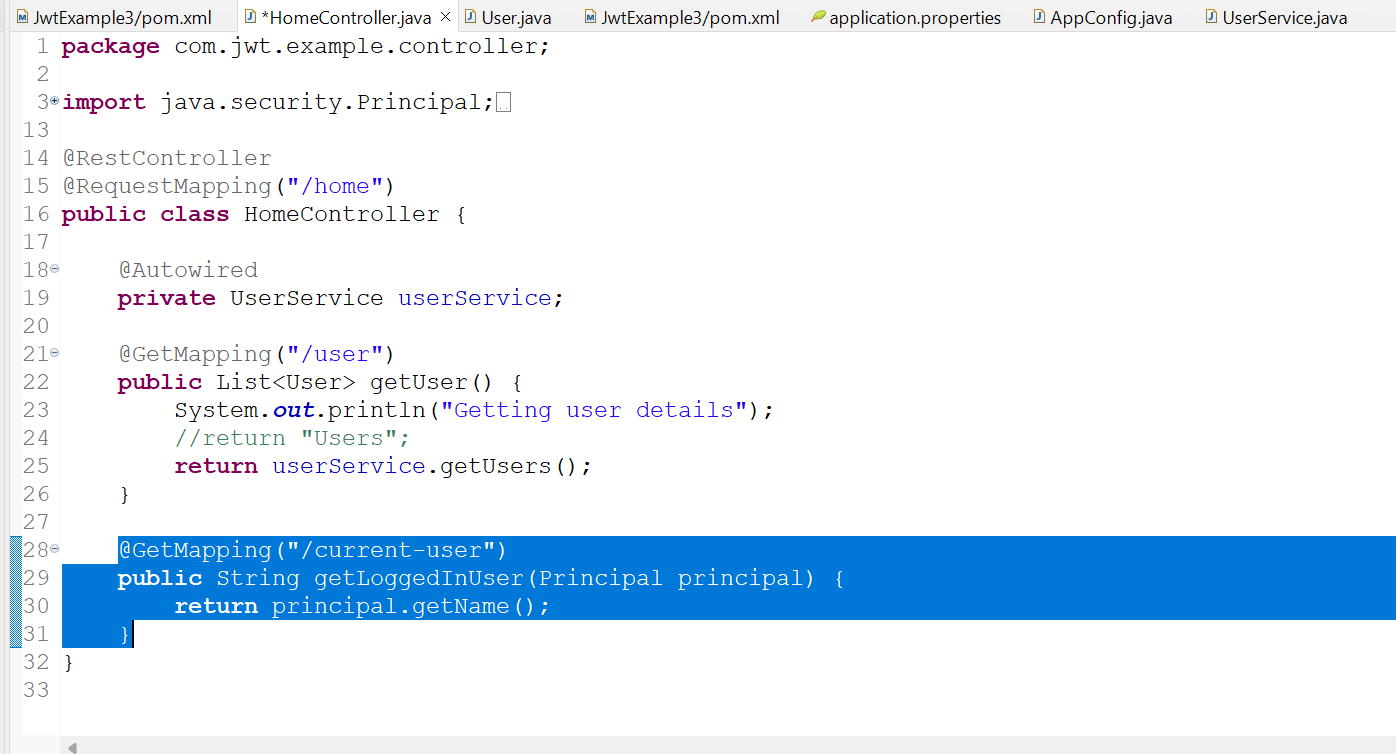
AI-generated content may be incorrect.**

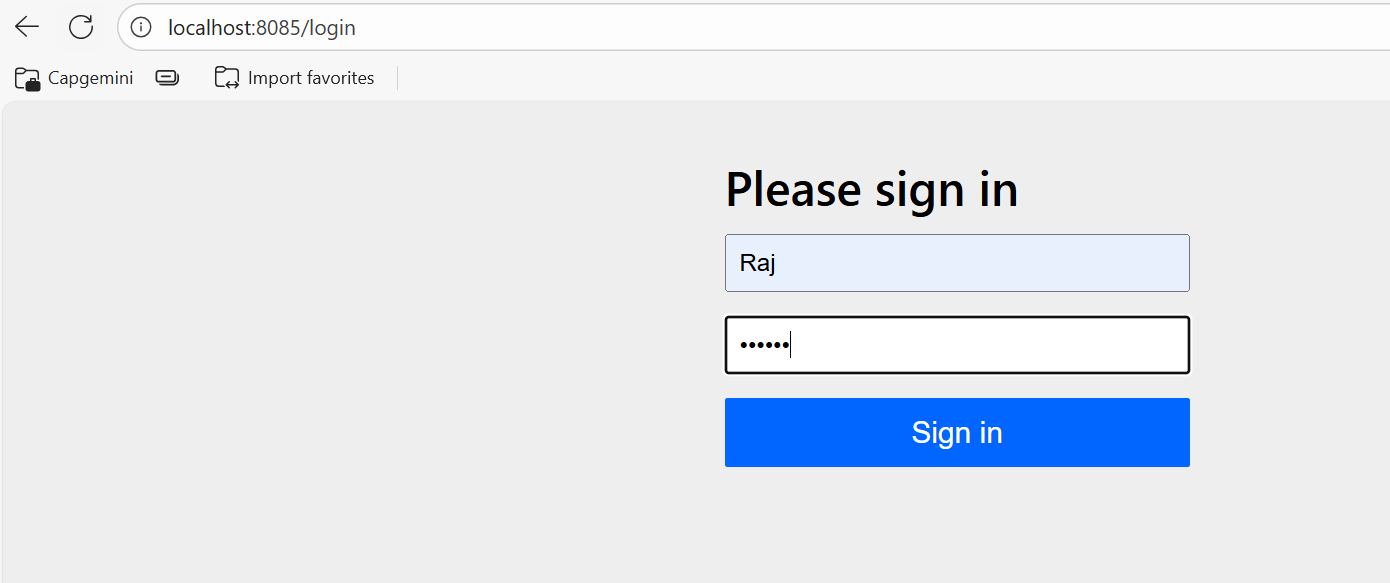
* **Now we can be able to access the API.**

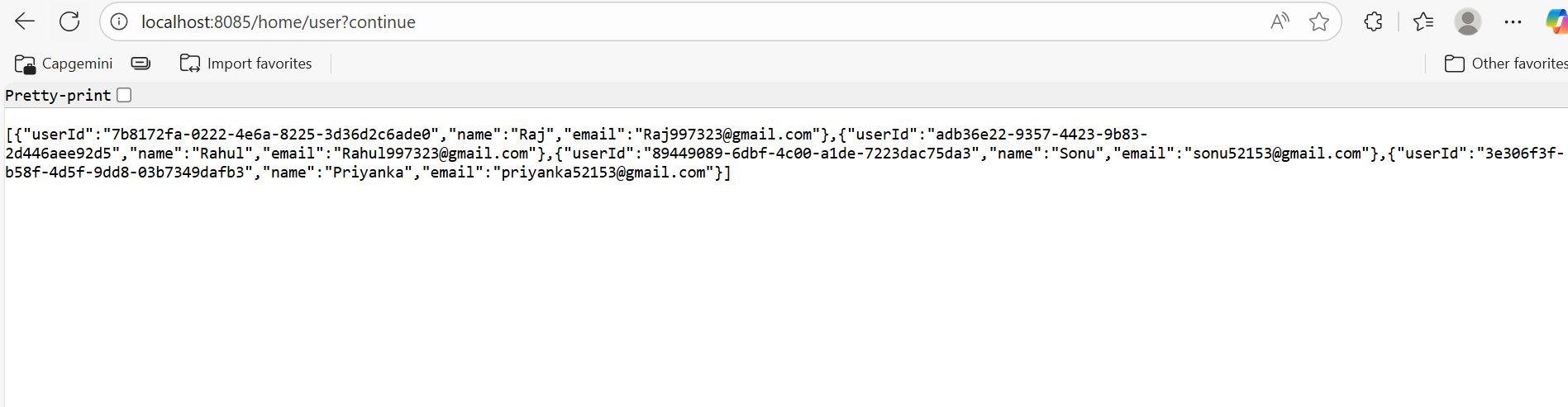
**A screenshot of a computer

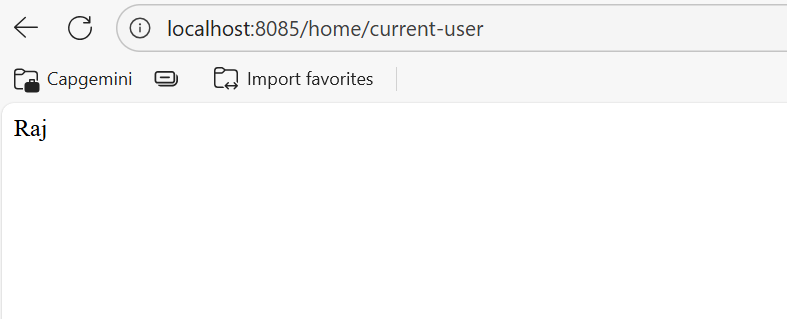
AI-generated content may be incorrect.**

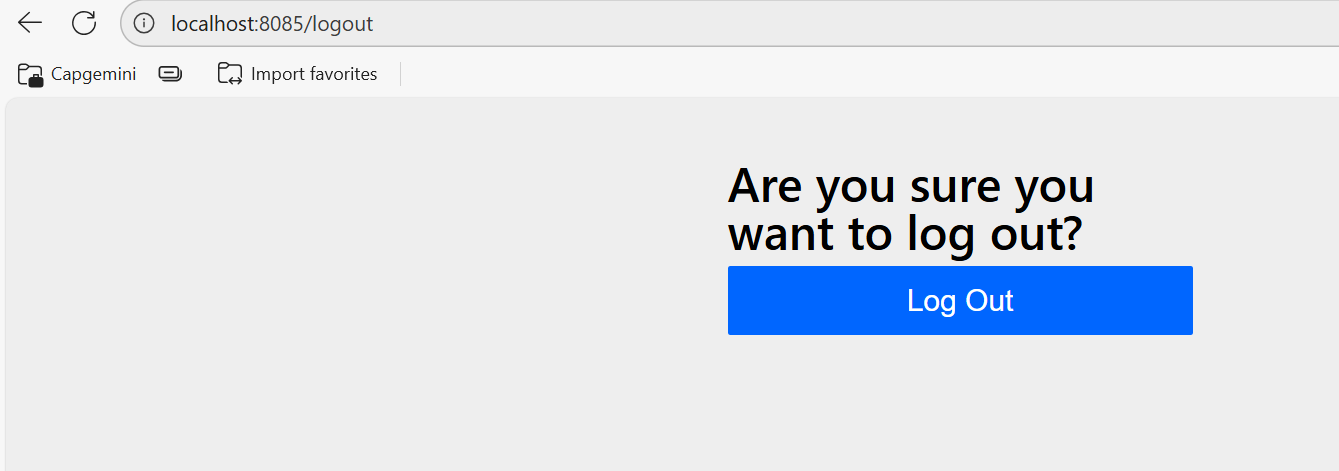
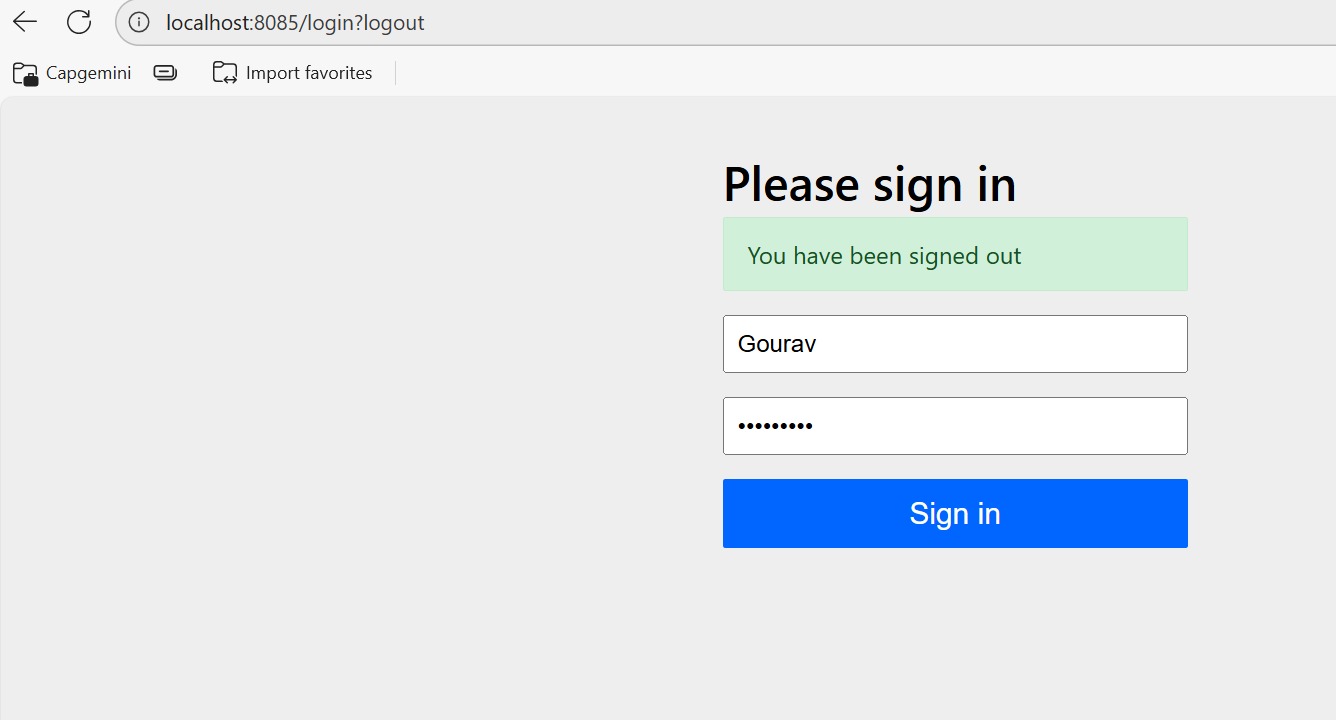
* **Now we are checking that which user has logged In.**

****

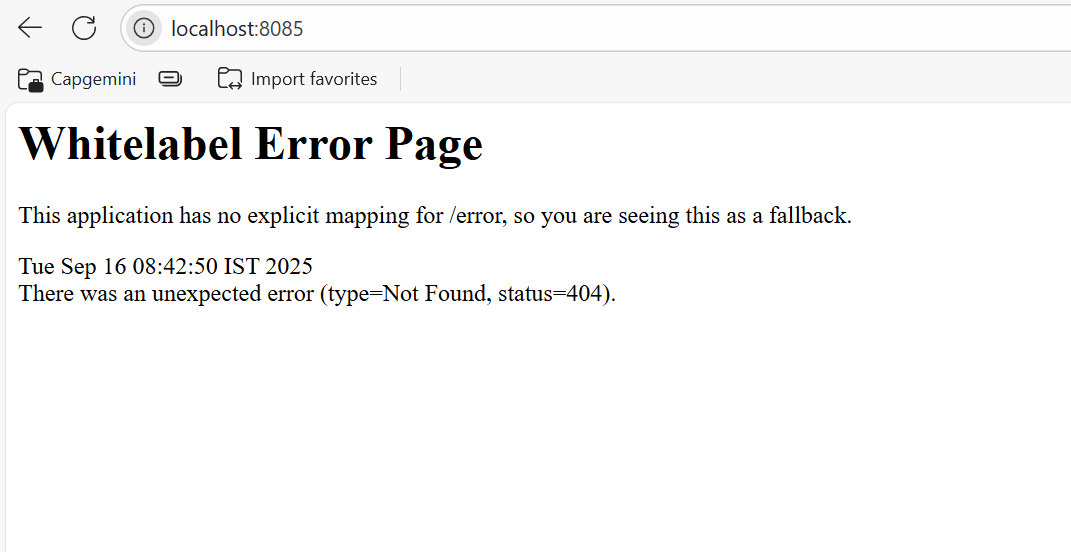
* **Now Restart the application And Enter Username and Password.  
  **
* **Response:**

****

* **Now checking current user…  
  **
* **After logging out, again we need to sign in by different user.**

**  
**

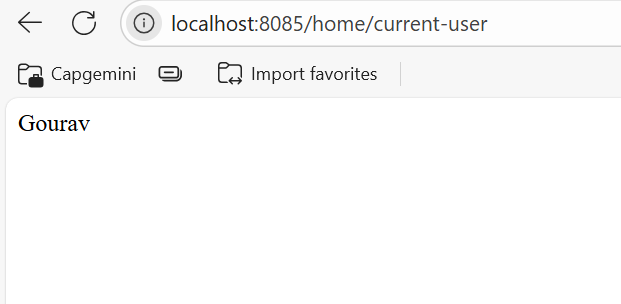
* **We can login successfully…**

****

* **We can access the API now.**

****

* **Now checking which user has been logged In.**

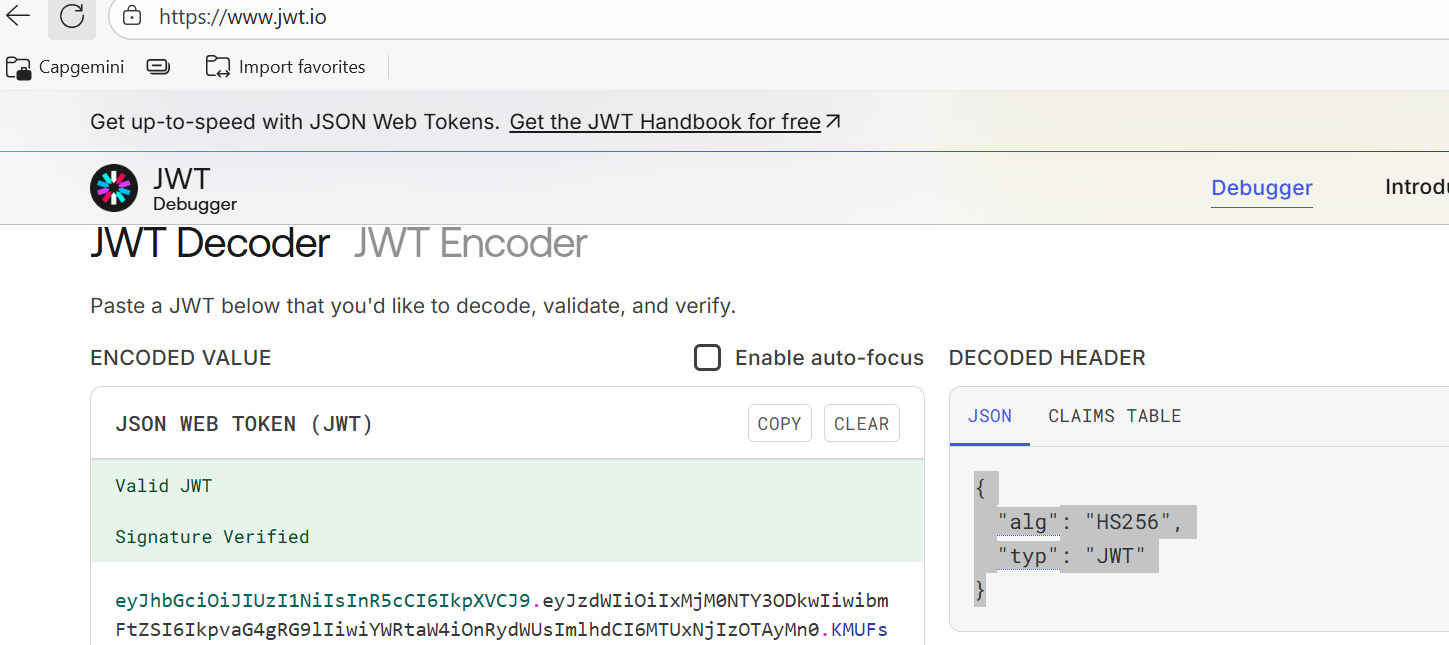
****

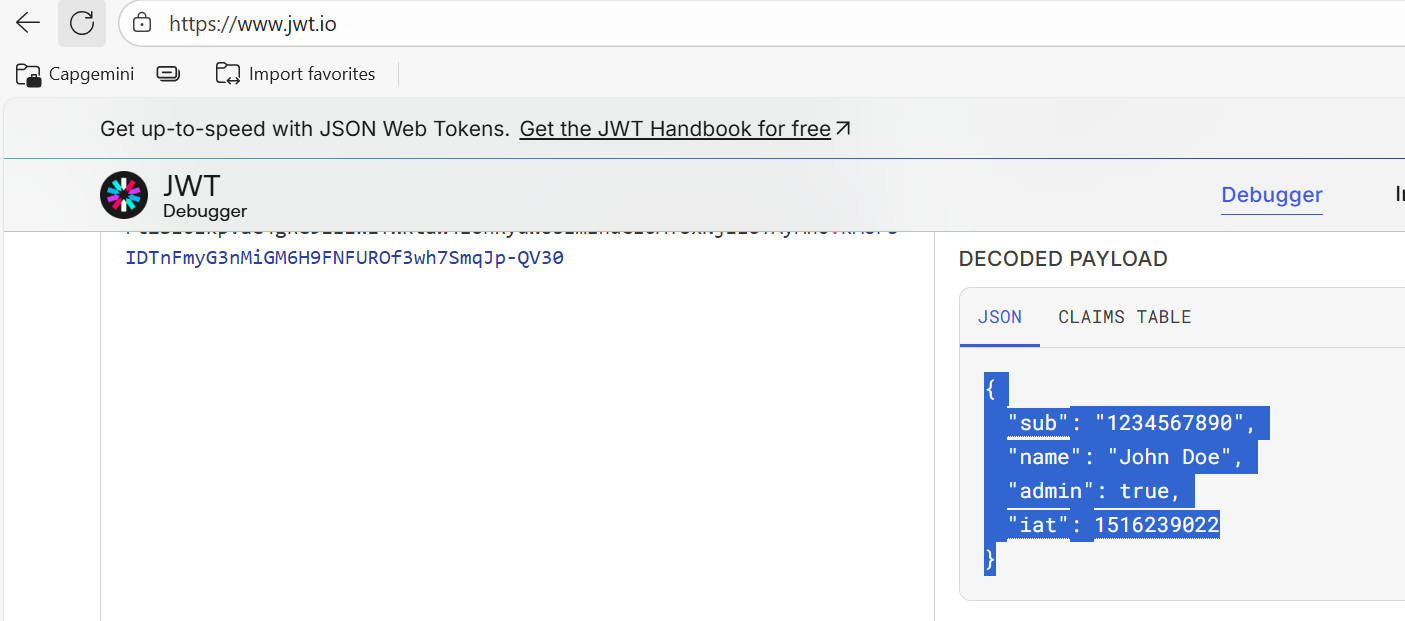
**Step 4: JWT Authentication Flow (Json Web Token)**

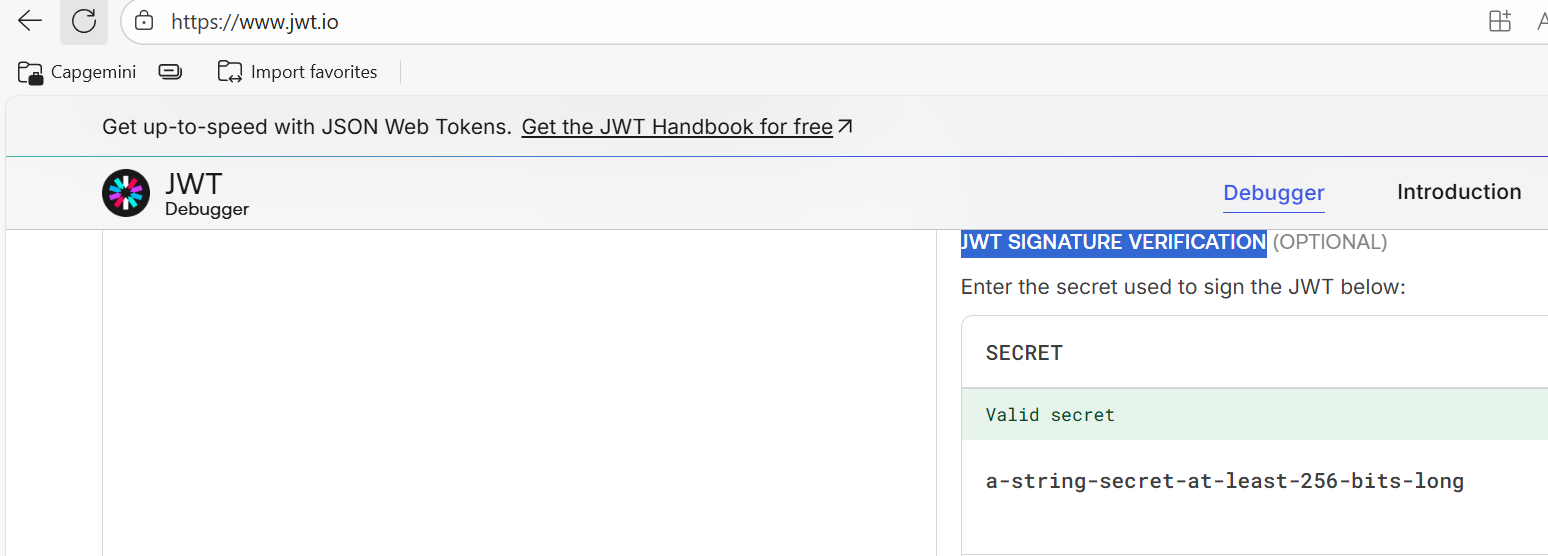
**E.g. If you go to a store and there is a huge crowd, if they are selling half the items outside, and if we want to buy something, we need to get a token by counter for any food items. Then we take the token and give it to them and based on that token they give us the items. It is the same here; we need to take the token first and using that token we can access the API.**

**Link:** [**JSON Web Tokens - jwt.io**](https://www.jwt.io/)

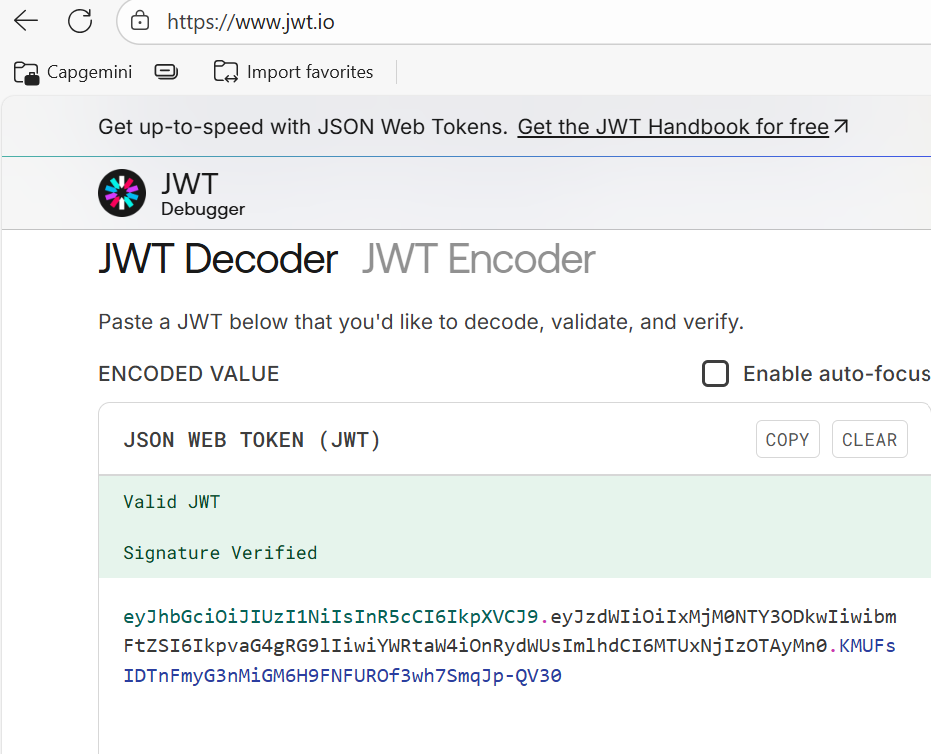
* **JWT is combination of 3 parts:  
  X.Y.Z**

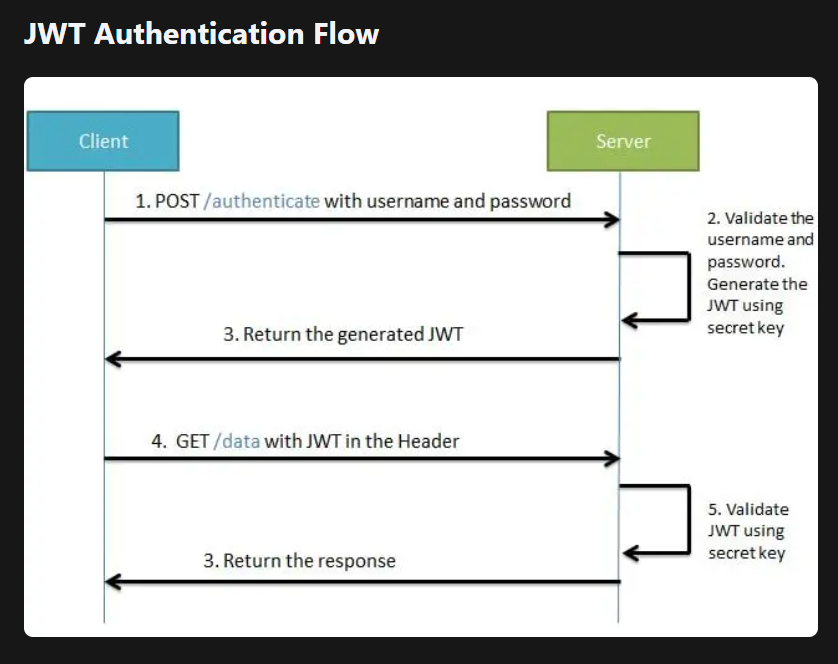
**X 🡪 Algo. + Token type  
**

**Y 🡪 Payload  
**

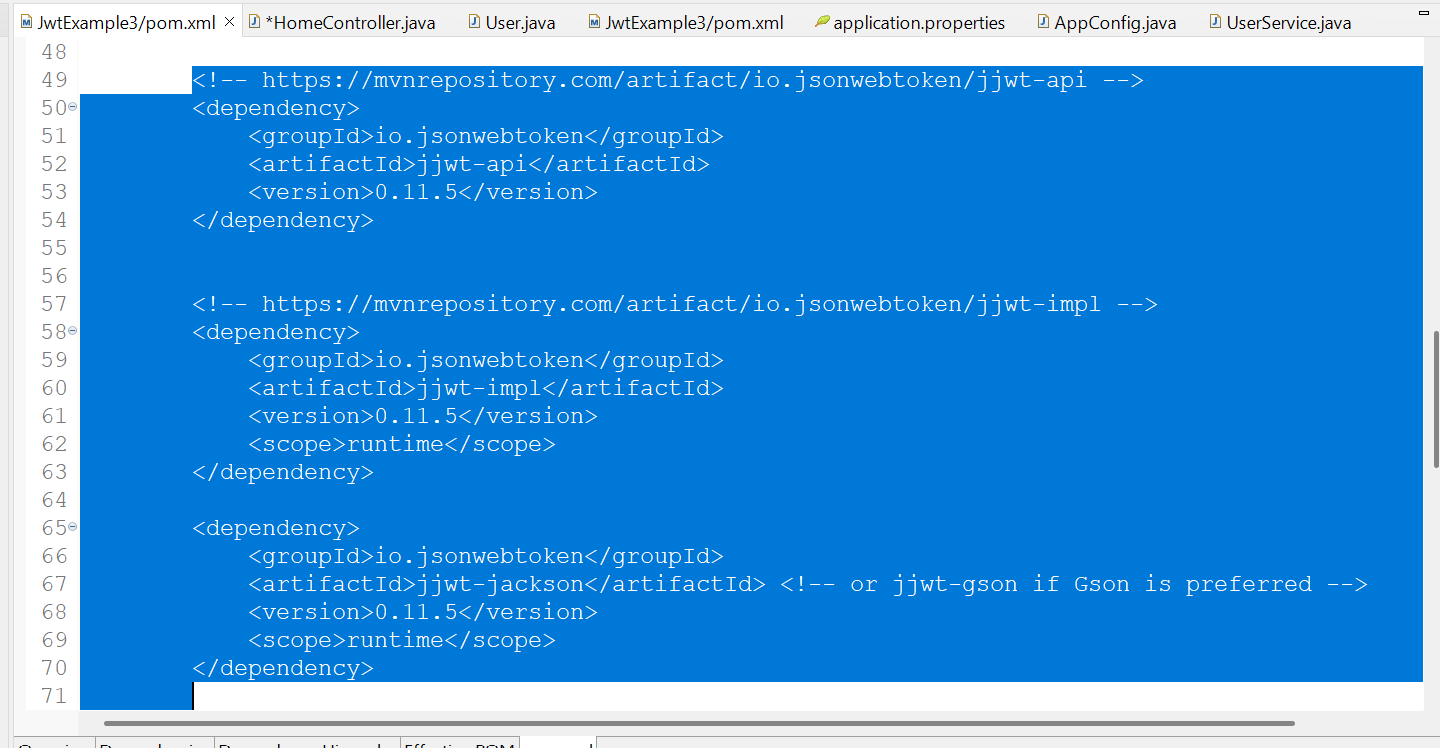
**Z 🡪 Signature Verification  
**

* **By using 3 parts, this token has been generated. Then this token will be given to the client.** **If the client accesses through this token, then they will be able to use the API. Otherwise, they will not be able to use.**

****

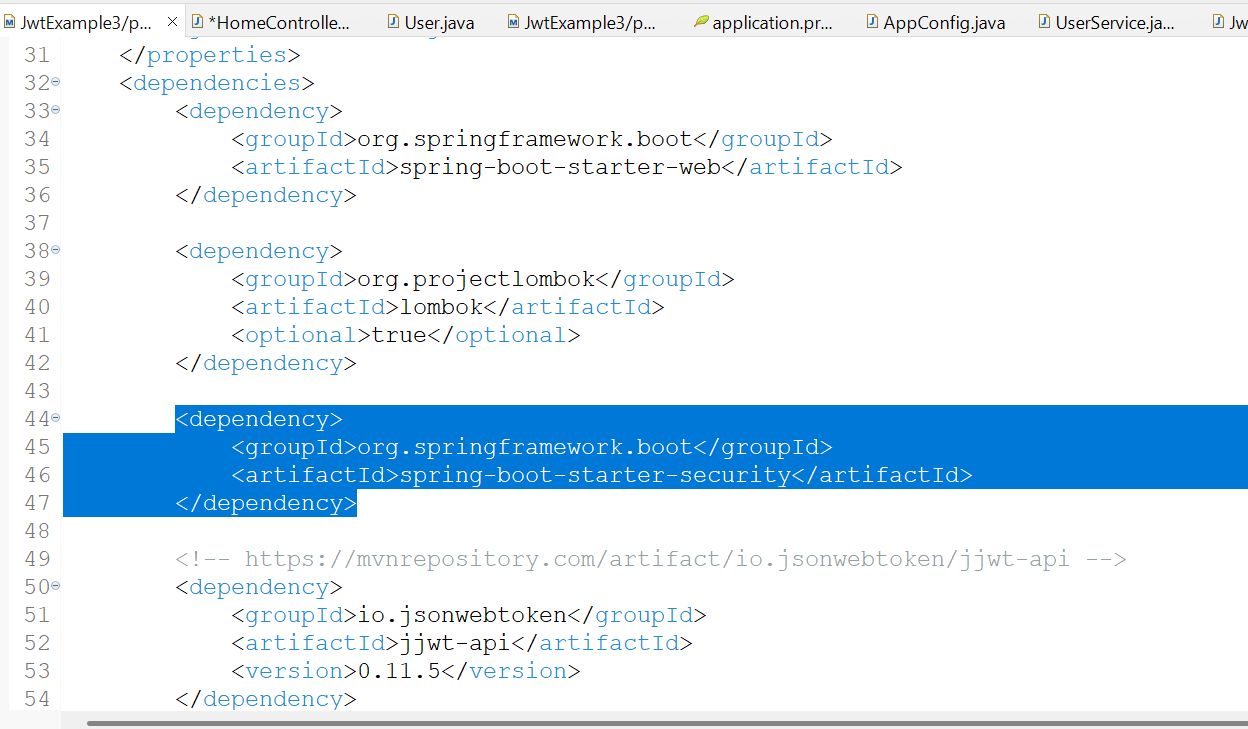
****

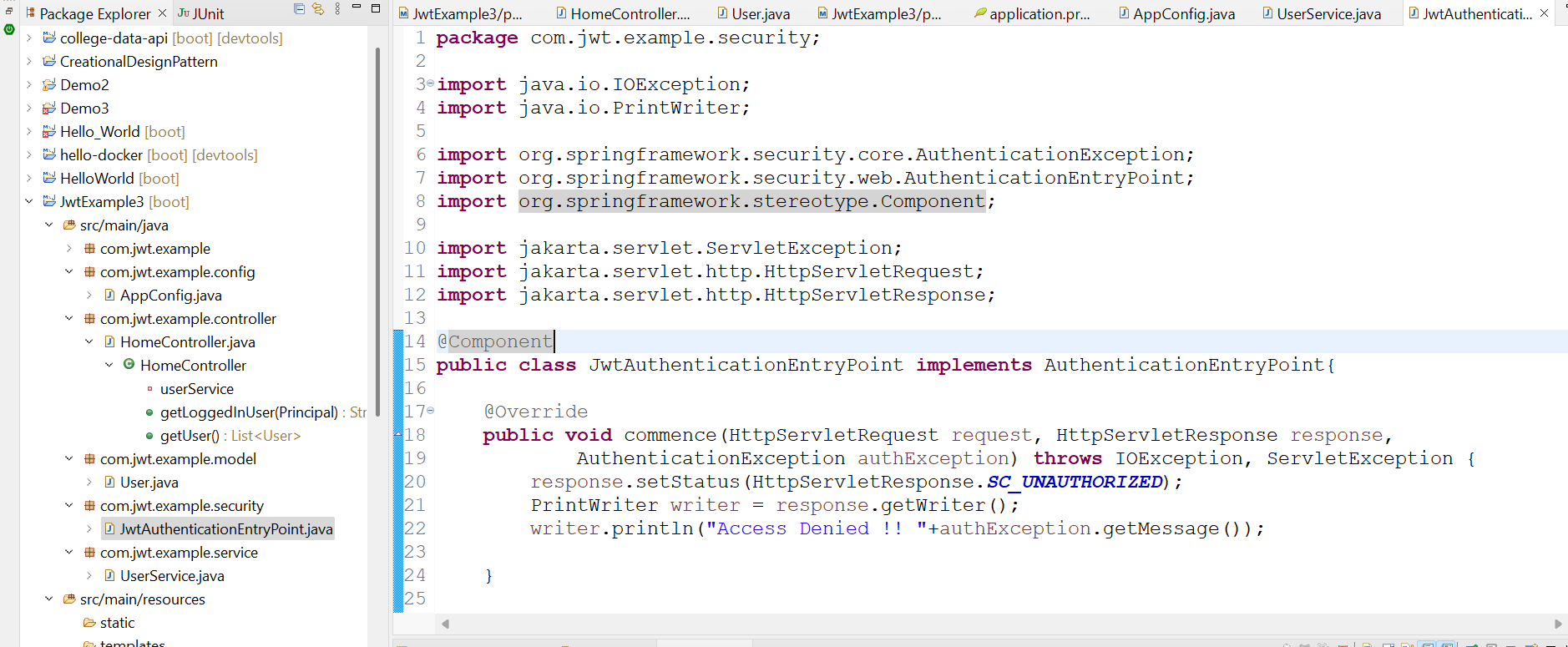


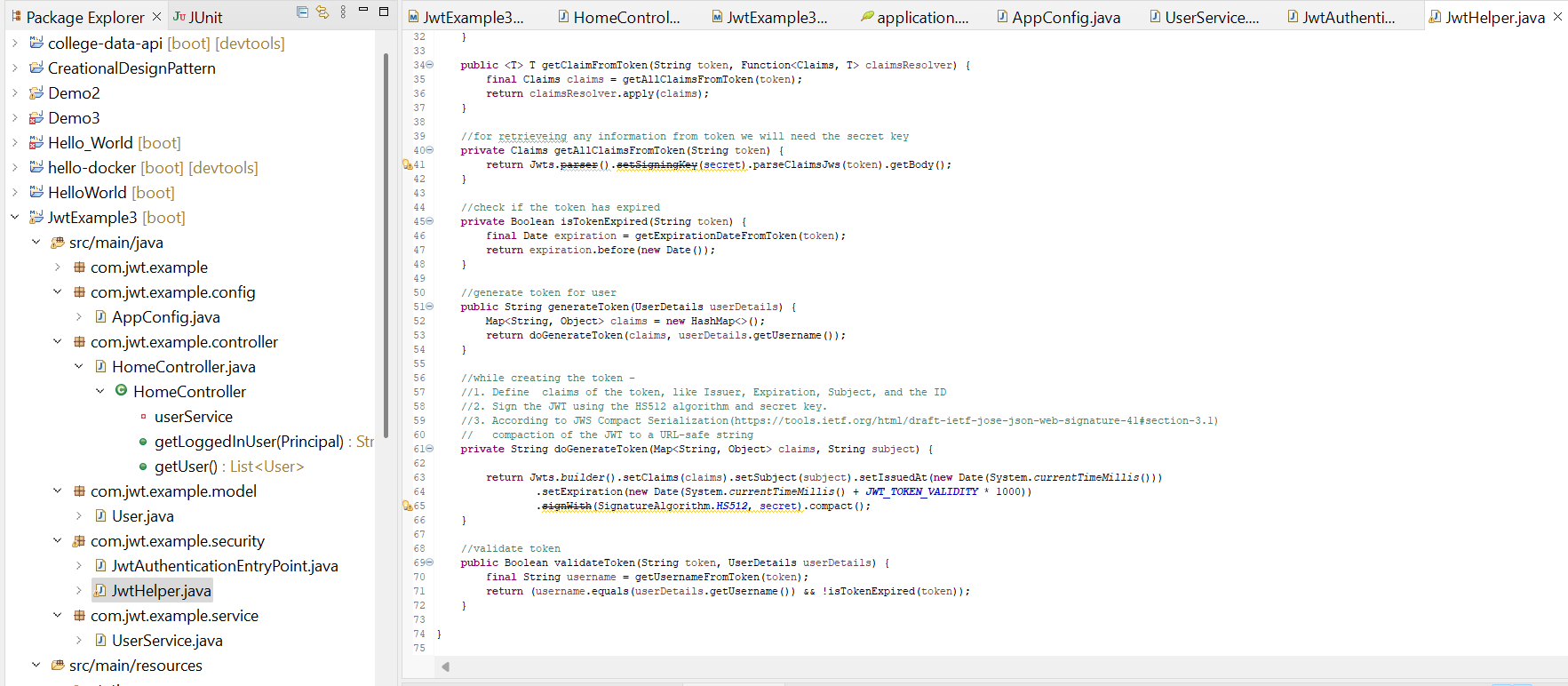
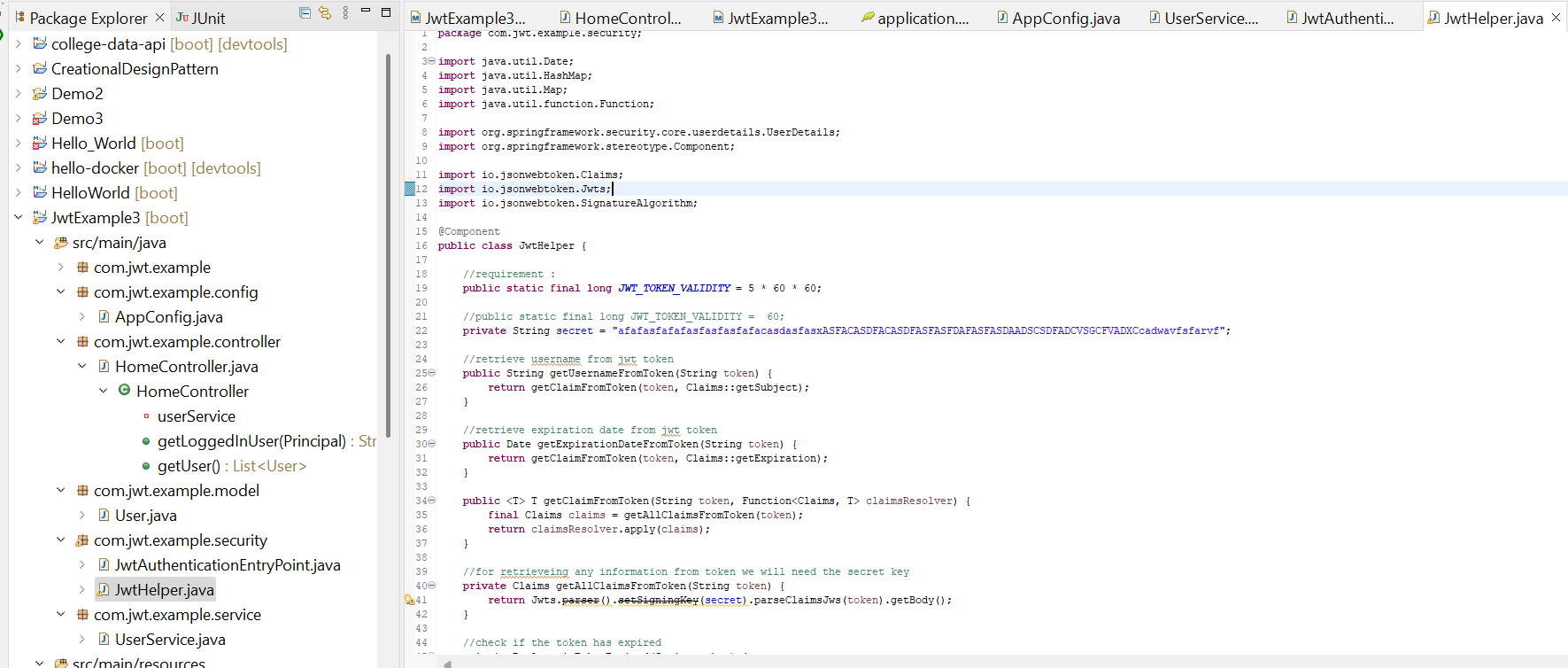
* **For JWT Dependency:** **Now Update the project then we can be able to access the classes.**

**Step 5: How to implement JWT Token**

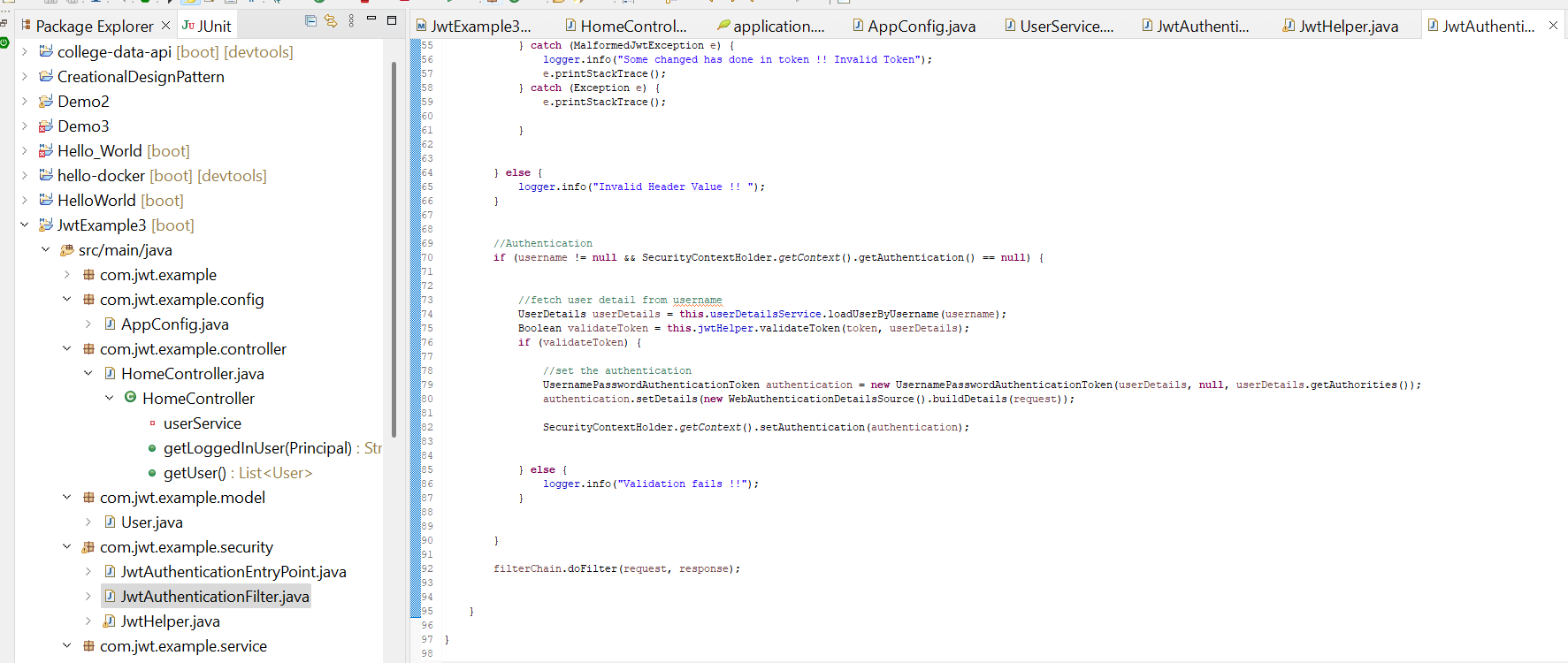
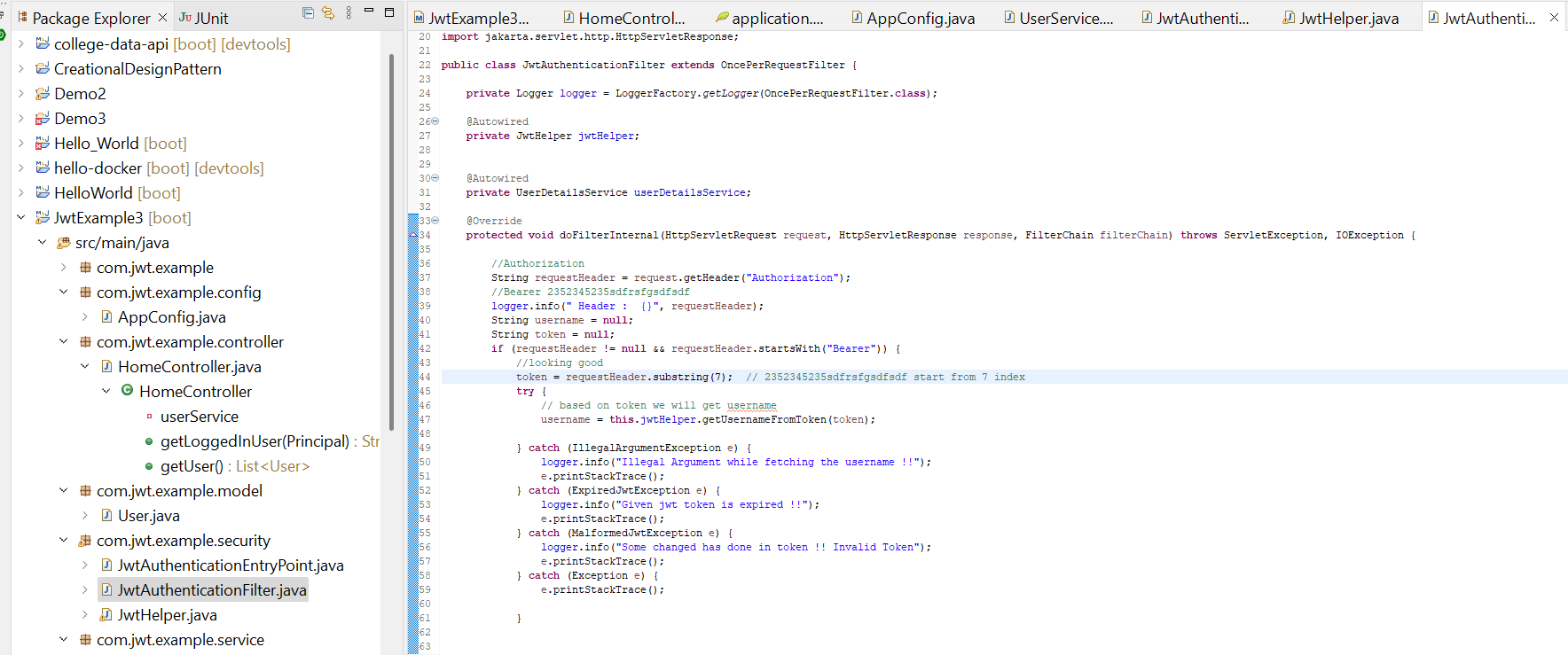
1. **Make sure spring-boot-starter-security is there in pom.xml**
2. **Create Class JWTAthenticationEntryPoint that implement AuthenticationEntryPoint. Method of this class is called whenever as exception is thrown due to unauthenticated user trying to access the resource that required authentication.**

****

****

1. **Create JWTHelper class: This class contains method related to perform operations with jwt token like generateToken, validateToken etc.Create JWTHelper class. This class contains method related to perform operations with jwt token like generateToken, validateToken etc.**
2. **Create JWTAuthenticationFilter that extends OncePerRequestFilter and override method and write the logic to check the token that is coming in header. We must write 5 important logics**

* **Get Token from request**
* **Validate Token**
* **GetUsername from token**
* **Load user associated with this token**
* **set authentication**

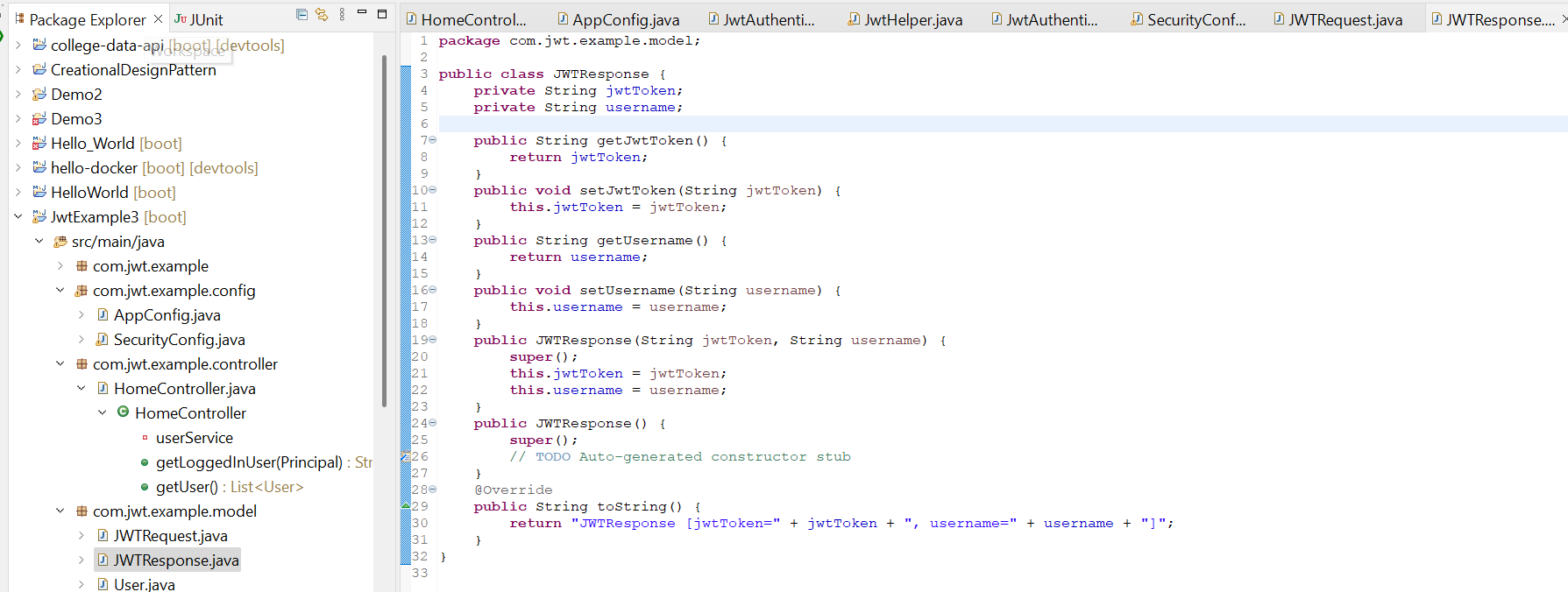


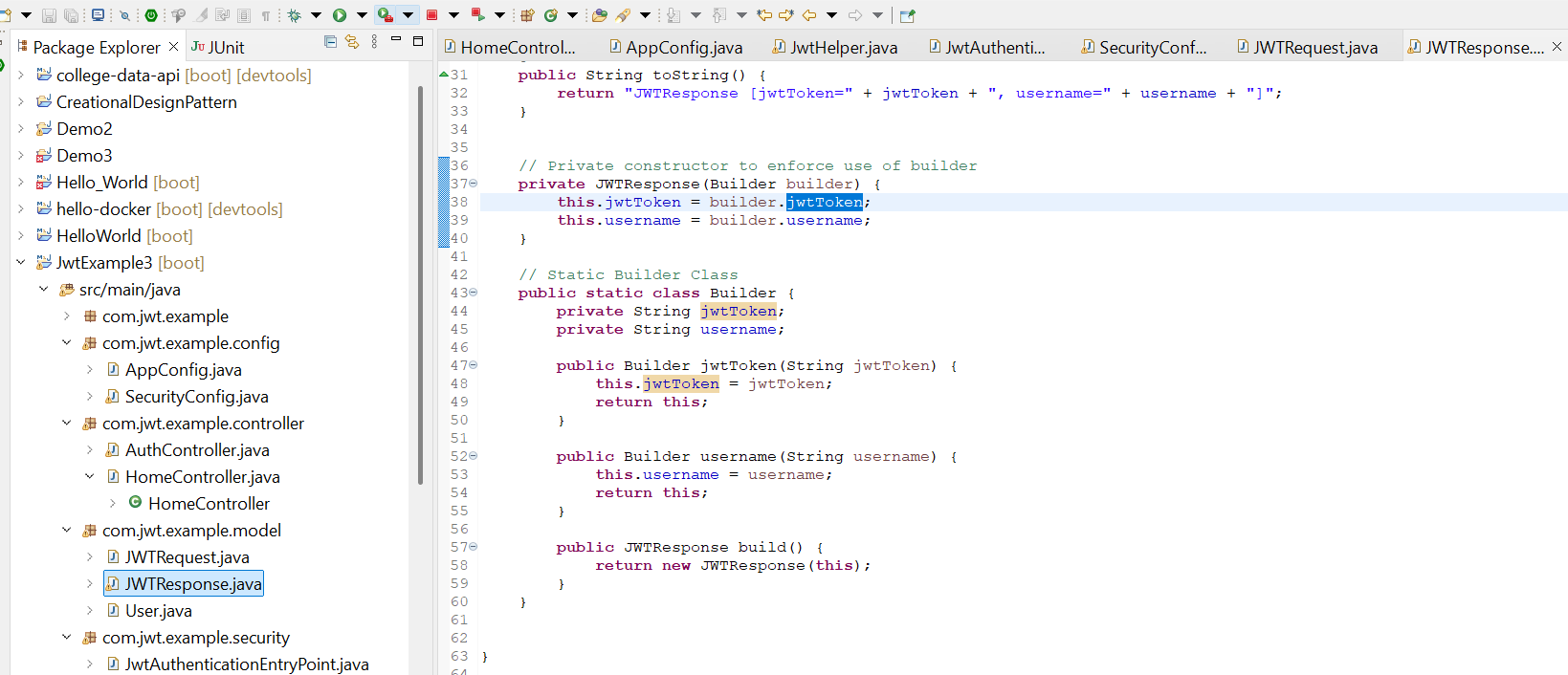
1. **Configure spring security in configuration file:**

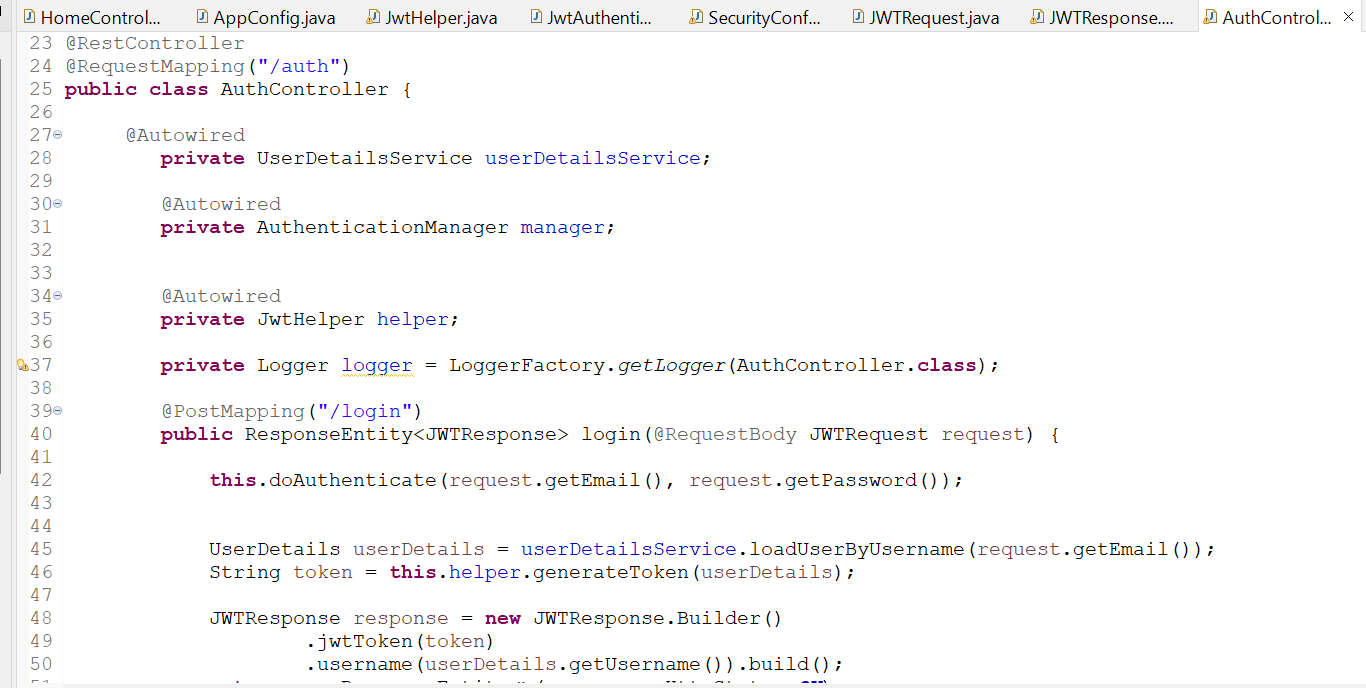
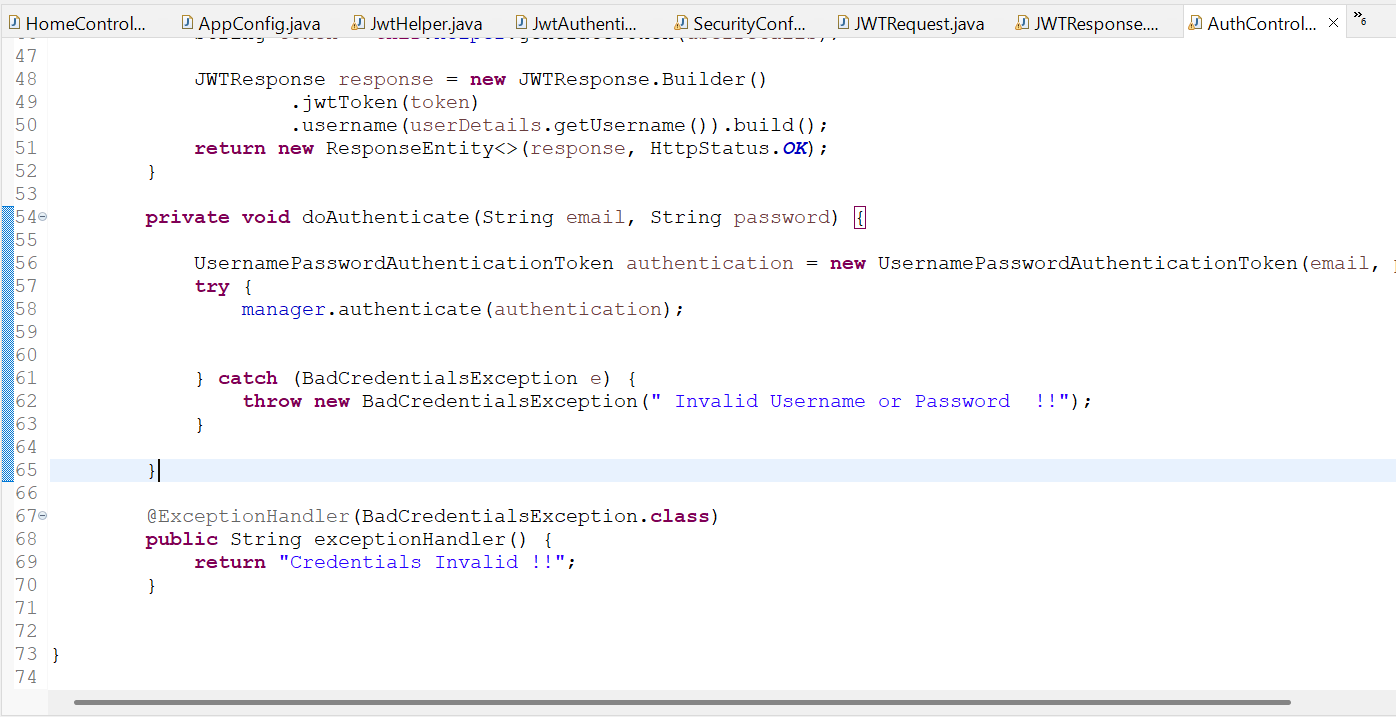
****

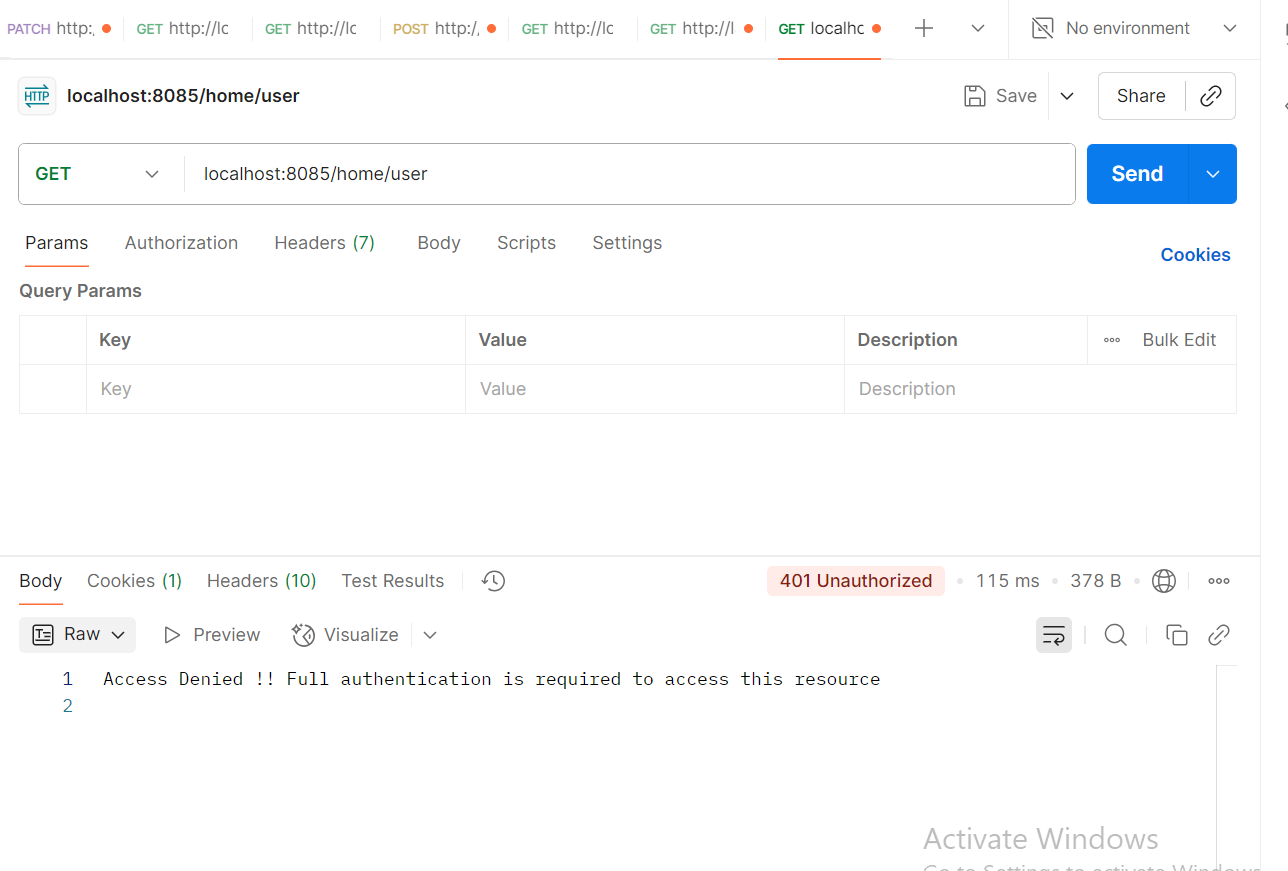
1. **Create JWTRequest and JWTResponse to receive request data and send Login success response.**

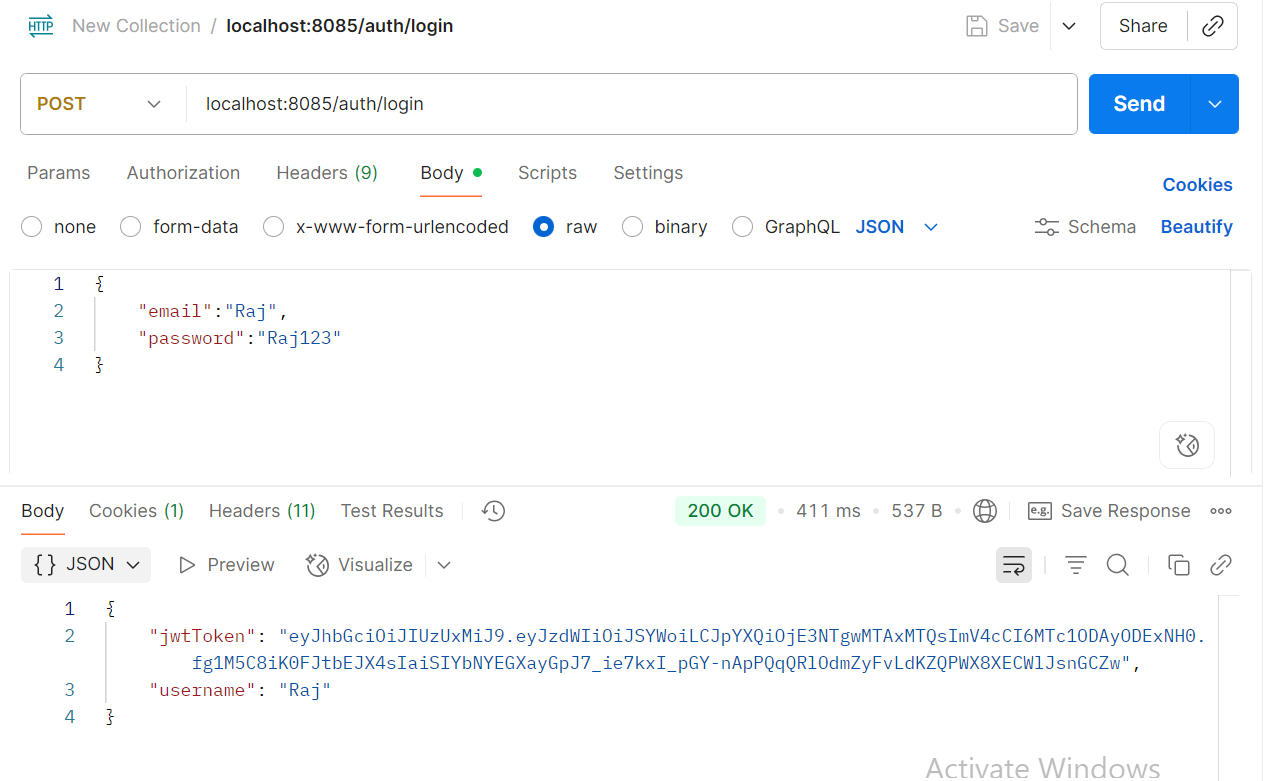
****

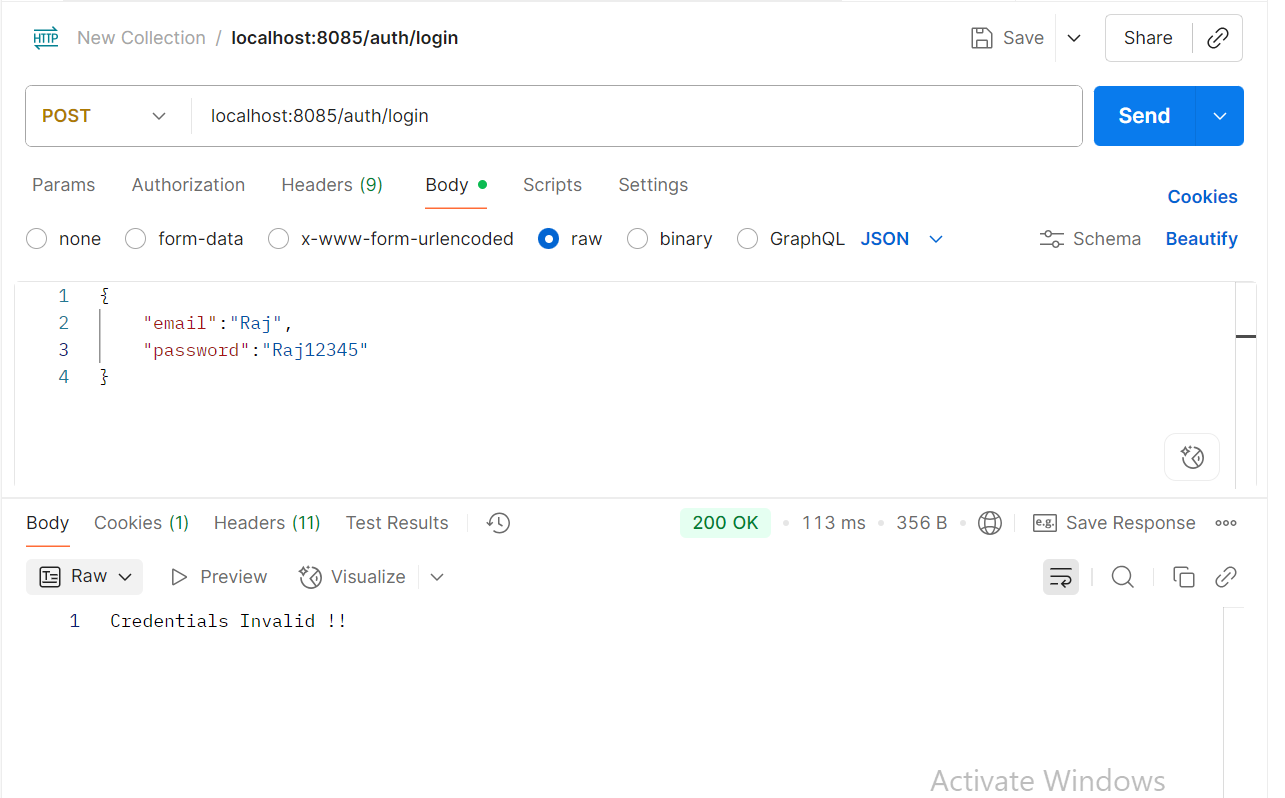
****

****

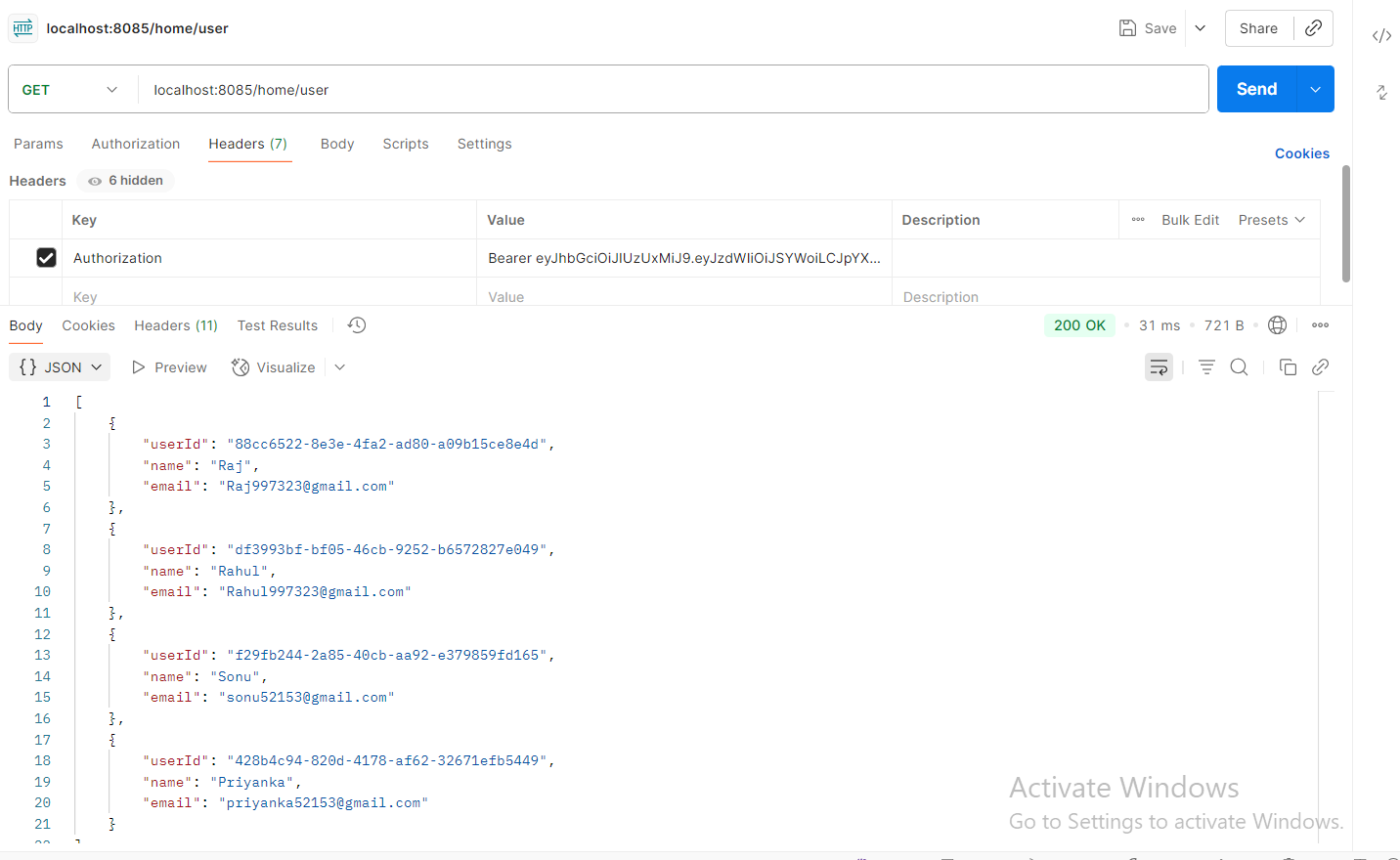
1. **Create login API to accept username and password and return token if username and password is correct.  
     
   **
2. **Test Application**

****

* **Generate Token using JWTRequest Field – email & password**
* **If Password is incorrect then getting Credentials Invalid !!**

****

* **Finally Validate the API using token.**

****