Easy JWT Authentication & Authorization with Spring Security | Step-by-Step Guide

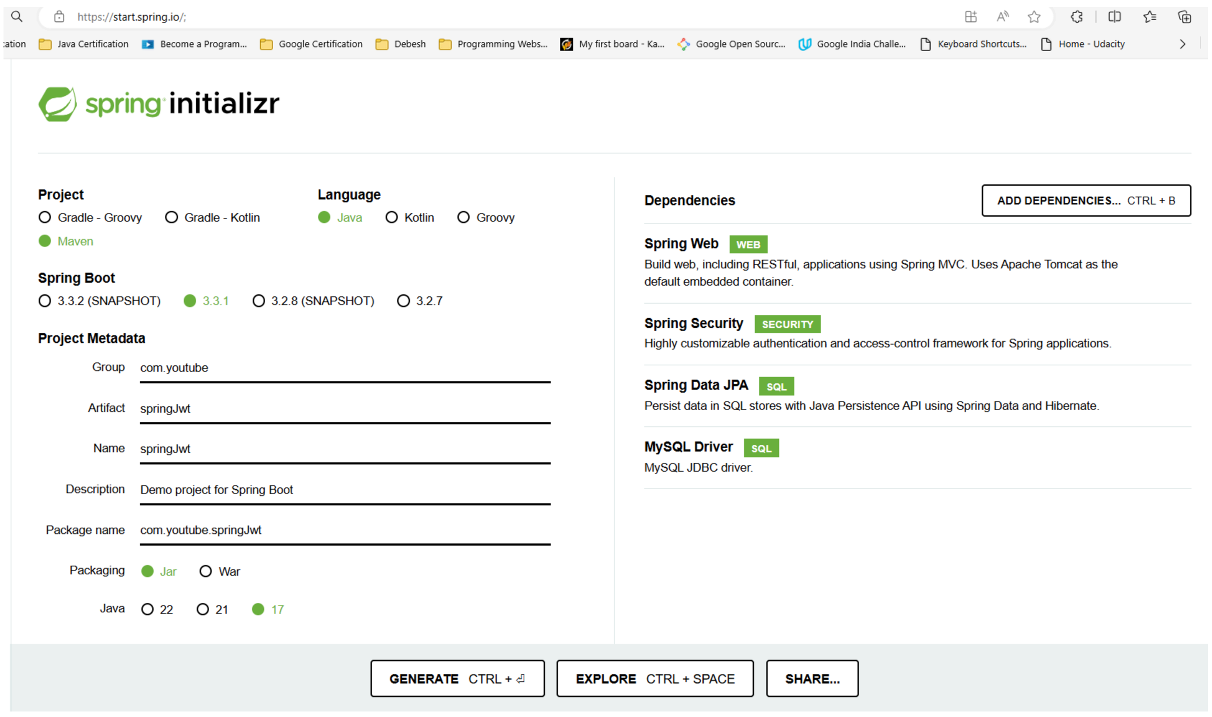
References:

1. [Easy JWT Authentication & Authorization with Spring Security | Step-by-Step Guide (youtube.com)]()
2. jwt.io
3. Spring Initializer

JWT holdes → Header . Payload . Signature

Back-end Part:

Step-1: Open Spring Initializer



https://start.spring.io/;

ation | Java Certification | Become a Program... | Google Certification | Debesh | Programming Webs... | My first board - Ka... | Google Open Sourc... | Google India Challe... | Keyboard Shortcuts... | Home - Udacity

spring initializr

Project
Gradle - Groovy | Gradle - Kotlin | [✓ Maven]

Language
[✓ Java] | Kotlin | Groovy

Spring Boot
3.3.2 (SNAPSHOT) | [✓ 3.3.1] | 3.2.8 (SNAPSHOT) | 3.2.7

Project Metadata

Group	com.youtube
Artifact	springJwt
Name	springJwt
Description	Demo project for Spring Boot
Package name	com.youtube.springJwt
Packaging	[✓ Jar] War
Java	22 21 [✓ 17]

Dependencies — ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | SHARE...

Step-2:Import the extracted project in the desired IDE [Open IntelliJ IDE.]

- Open *Intellij IDE*
- Click on *File*
- Then *OPEN*
- Click the *Project Folder*
- Click *POM.XML*
- It loded the springboot project :)

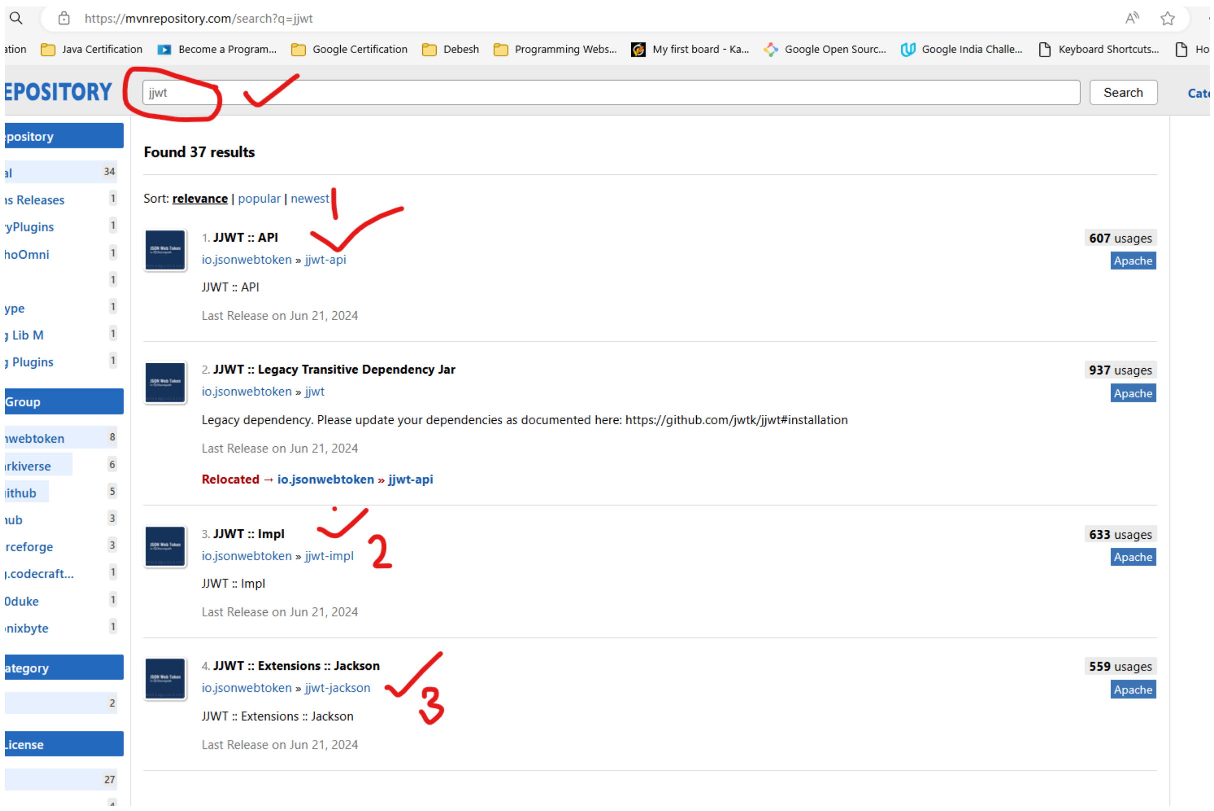
Step-3: Dependencies

These are the dependencies I added 👍

```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>

   <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <scope>runtime</scope>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
   </dependency>
   <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-test</artifactId>
```

```
        <scope>test</scope>
    </dependency>
</dependencies>
```

Add New Dependency:



https://mvnrepository.com/search?q=jjwt

tion | Java Certification | Become a Program... | Google Certification | Debesh | Programming Webs... | My first board - Ka... | Google Open Sourc... | Google India Challe... | Keyboard Shortcuts... | Ho

EPOSITORY | jjwt | Search | Cat

Found 37 results

Sort: **relevance** | popular | newest

1. **JJWT :: API** — io.jsonwebtoken » jjwt-api — 607 usages — Apache
JJWT :: API
Last Release on Jun 21, 2024

2. **JJWT :: Legacy Transitive Dependency Jar** — io.jsonwebtoken » jjwt — 937 usages — Apache
Legacy dependency. Please update your dependencies as documented here: https://github.com/jwtk/jjwt#installation
Last Release on Jun 21, 2024
Relocated → io.jsonwebtoken » jjwt-api

3. **JJWT :: Impl** — io.jsonwebtoken » jjwt-impl — 633 usages — Apache ✓ 2
JJWT :: Impl
Last Release on Jun 21, 2024

4. **JJWT :: Extensions :: Jackson** — io.jsonwebtoken » jjwt-jackson — 559 usages — Apache ✓ 3
JJWT :: Extensions :: Jackson
Last Release on Jun 21, 2024

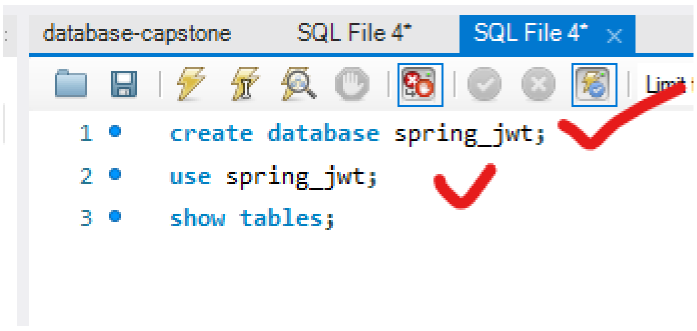
pository	
l	34
s Releases	1
yPlugins	1
hoOmni	1
	1
ype	1
Lib M	1
Plugins	1
Group	
webtoken	8
rkiverse	6
ithub	5
ub	3
rceforge	3
.codecraft...	1
0duke	1
nixbyte	1
ategory	
	2
icense	
	27

```
<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-api -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.12.5</version>
</dependency>

<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-impl
-->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.12.6</version>
    <scope>runtime</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-jackson
-->
```

```xml
<dependency>
   <groupId>io.jsonwebtoken</groupId>
   <artifactId>jjwt-jackson</artifactId>
   <version>0.12.5</version>
   <scope>runtime</scope>
</dependency>
```

Step-4: Database Connection

- Open MySQL Workbench
- Create database spring_jwt



```sql
create database spring_jwt;
use spring_jwt;
show tables;
```

- Open application.properties file and refactor it as application.yml file
  Then write the following code:

```yaml
spring:
 datasource:
   url: jdbc:mysql://localhost:3306/spring_jwt
   username: root
   password: root
   driver-class-name: com.mysql.cj.jdbc.Driver
 jpa:
   hibernate:
     ddl-auto: create-drop
   show-sql: true
   properties:
     hibernate:
       format_sql: true
   database: mysql
   database-platform: org.hibernate.dialect.MySQL8Dialect
```

Step-5: Spring Security -> Start to Work on

1. Create a user class

Create a package belong to the root package as model
src/main/java/com/youtube/springJwt/model

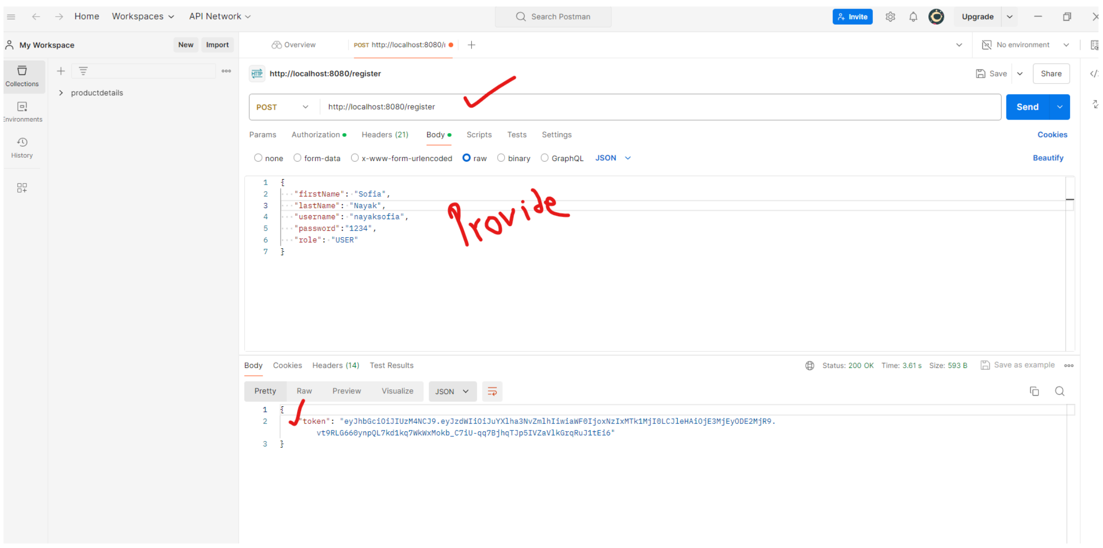
```
package com.youtube.springJwt.model;
   public class User {}
```

Step-6: Search for 256 bit secret key generation

- Click: [Human-Readable Encryption Keys (asecuritysite.com)]
- Choose: 256 bit
- Copy the Hex Key:
  e5fca0035f9aaa2c6bf4641e81bd5d35bcc046ac5b6b6fcbf950d7699d1c257b

POSTMAN API Checking:

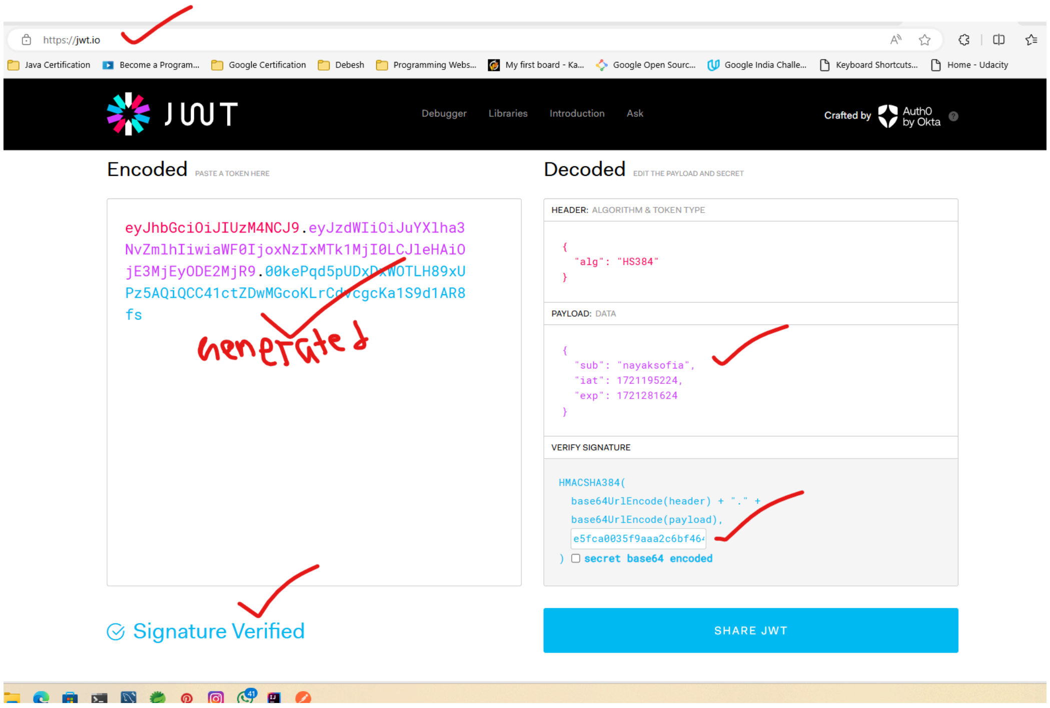
Token Generated:


POST http://localhost:8080/register

```
{
   "firstName": "Sofia",
   "lastName": "Nayak",
   "username": "nayaksofia",
   "password":"1234",
   "role": "USER"
}
```

Status: 200 OK Time: 3.61 s Size: 593 B

```
{
  "token": "eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJuYXlha3NvZmlhIiwiaWF0IjoxNzIxMTk1MjI0LCJleHAiOjE3MjEyODE2MjR9.
      vt9RLG66OynpQL7kd1kq7WkWxMokb_C7iU-qq7BjhqTJp5IVZaVlkGrqRuJ1tEi6"
}
```



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJuYXlha3
NvZmlhIiwiaWF0IjoxNzIxMTk1MjI0LCJleHAiO
jE3MjEyODE2MjR9.00kePqd5pUDxDxWOTLH89xU
Pz5AQiQCC41ctZDwMGcoKLrCdvcgcKa1S9d1AR8
fs
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS384"
}
```

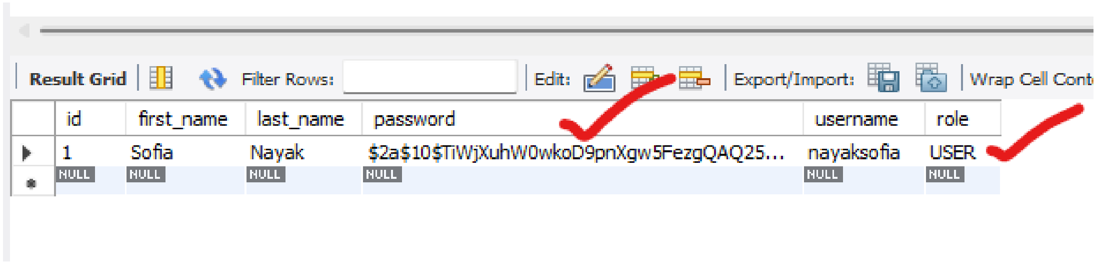
PAYLOAD: DATA

```
{
  "sub": "nayaksofia",
  "iat": 1721195224,
  "exp": 1721281624
}
```

VERIFY SIGNATURE

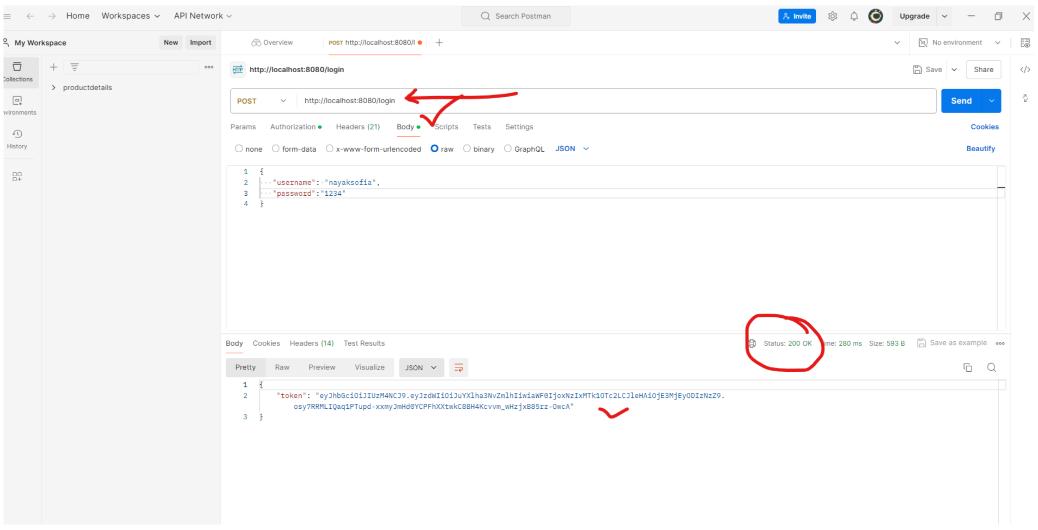
```
HMACSHA384(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  e5fca0035f9aaa2c6bf46
) ☐ secret base64 encoded
```

In Database the table created:



	id	first_name	last_name	password	username	role
▶	1	Sofia	Nayak	$2a$10$TiWjXuhW0wkoD9pnXgw5FezgQAQ25...	nayaksofia	USER
✱	NULL	NULL	NULL	NULL	NULL	NULL

Let’s Login To the system:



POST http://localhost:8080/login

Params | Authorization | Headers (21) | Body | Scripts | Tests | Settings

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
  "username": "nayaksofia",
  "password":"1234"
}
```

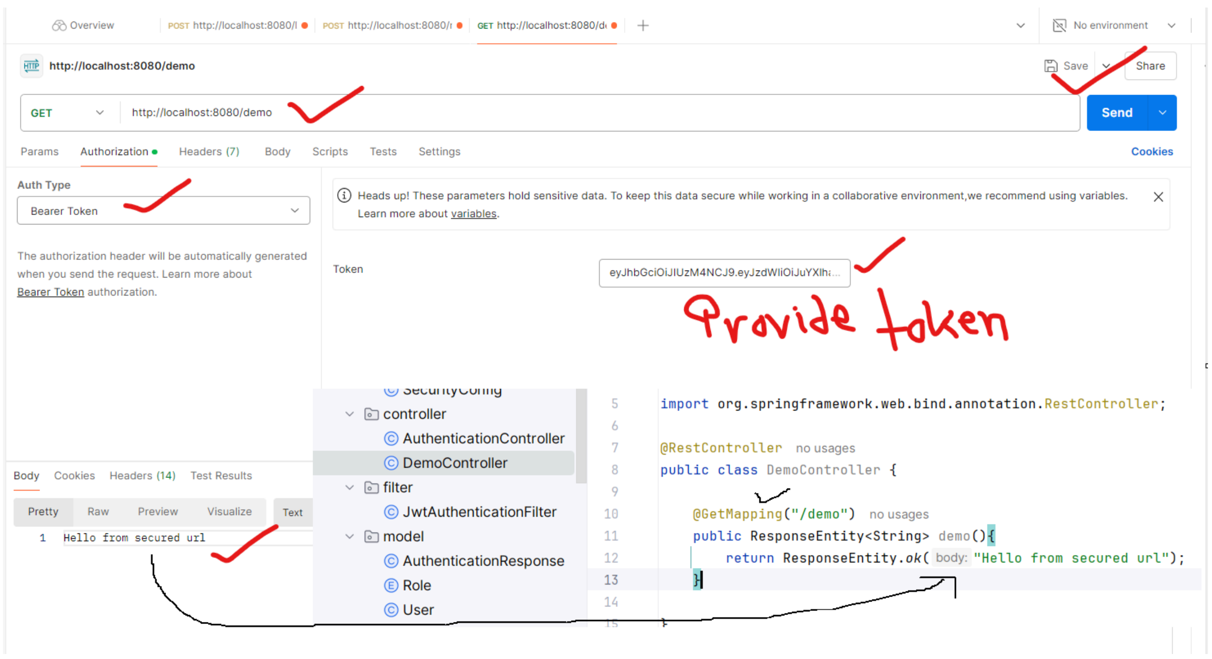
Body | Cookies | Headers (14) | Test Results — Status: 200 OK

```
{
    "token": "eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJuYXlha3NvZmlhIiwiaWF0IjoxNzIxMTk1OTc2LCJleHAiOjE3MjEyODIzNzZ9.
        osy7RRMLIQaq1PTupd-xxmyJmHd0YCPFhXXtwkC8BH4Kcvvm_wHzjxB85rz-OwcA"
}
```

Endpoints to check Authentication:

USER AUTHENTICATION

```
{
   "firstName":"Sofia",
   "lastName": "Nayak",
   "username": "nayaksofia",
   "password":"12345",
   "role":"USER"
}
```



http://localhost:8080/demo

GET http://localhost:8080/demo

Auth Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token: eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJuYXIh...

Body · Cookies · Headers (14) · Test Results — Pretty · Raw · Preview · Visualize · Text

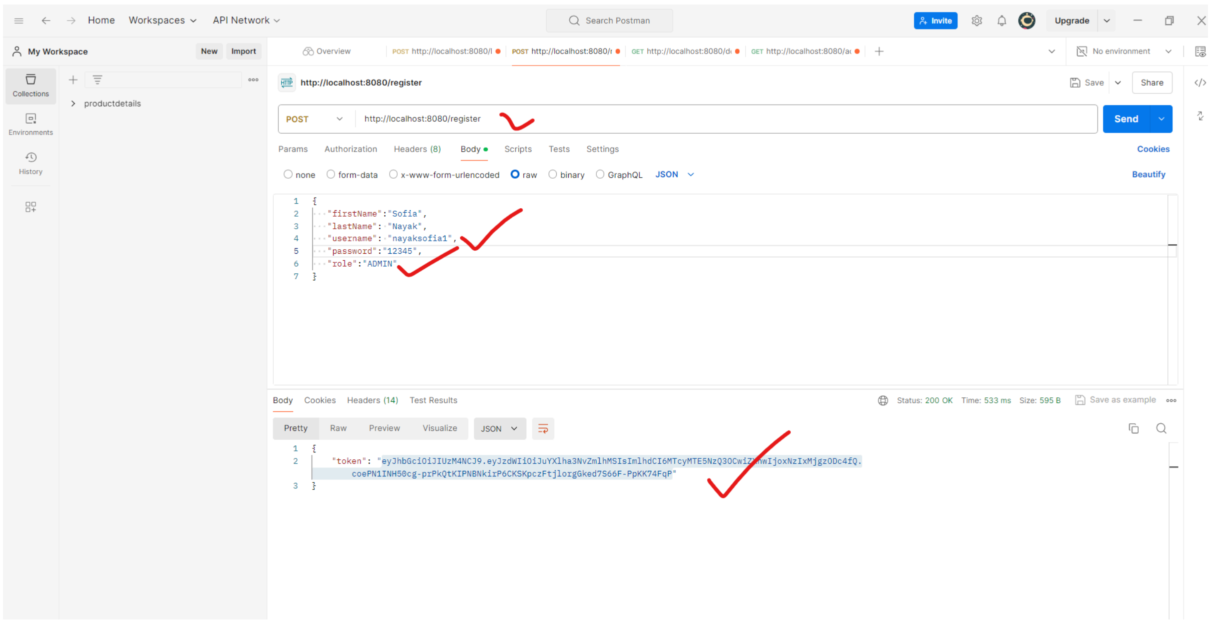
```
1   Hello from secured url
```

```
SecurityConfig
controller
    AuthenticationController
    DemoController
filter
    JwtAuthenticationFilter
model
    AuthenticationResponse
    Role
    User
```

```
import org.springframework.web.bind.annotation.RestController;

@RestController  no usages
public class DemoController {

    @GetMapping("/demo")  no usages
    public ResponseEntity<String> demo(){
        return ResponseEntity.ok( body: "Hello from secured url");
    }

}
```

For Admin:

```
{
   "firstName":"Sofia",
   "lastName": "Nayak",
   "username": "nayaksofia1",
   "password":"12345",
   "role":"ADMIN"
}
```

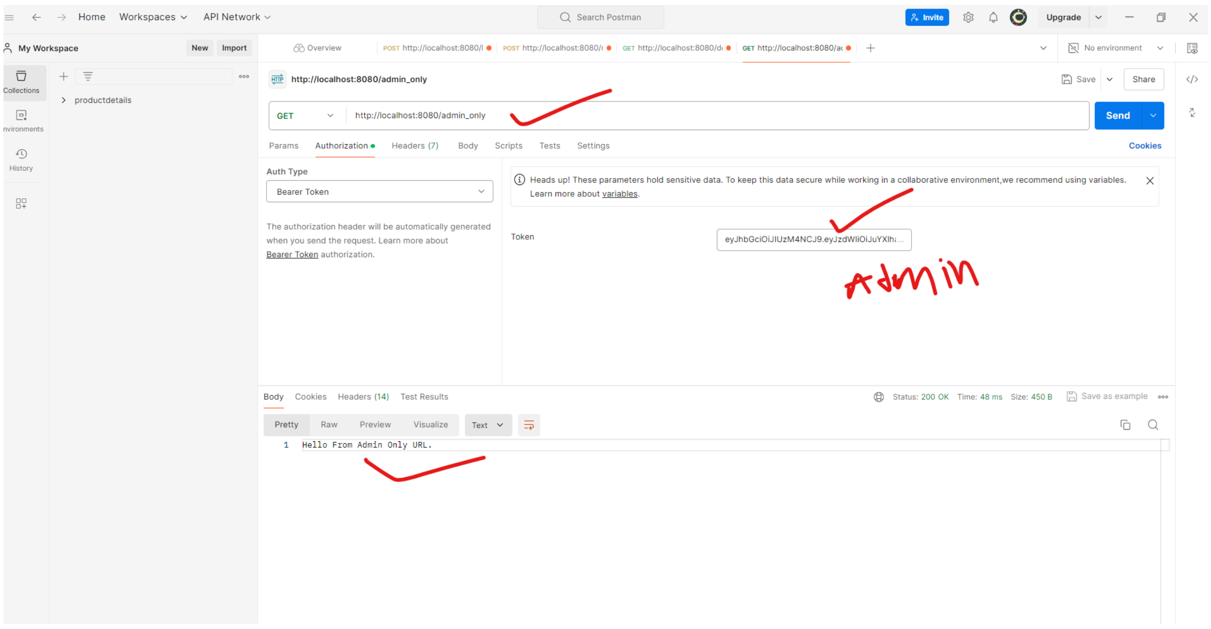


POST http://localhost:8080/register

```
{
    "firstName":"Sofia",
    "lastName": "Nayak",
    "username": "nayaksofia1",
    "password":"12345",
    "role":"ADMIN"
}
```

Status: 200 OK Time: 533 ms Size: 595 B

```
{
    "token": "eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJuYXlha3NvZmlhMSIsImlhdCI6MTcyMTE5NzQ3OCwiZXhwIjoxNzIxMjgzODc4fQ.
        coePN1INH50cg-prPkQtKIPNBNkirP6CKSKpczFtjlorgGked7S66F-PpKK74FqP"
}
```



GET http://localhost:8080/admin_only

Authorization: Auth Type: Bearer Token

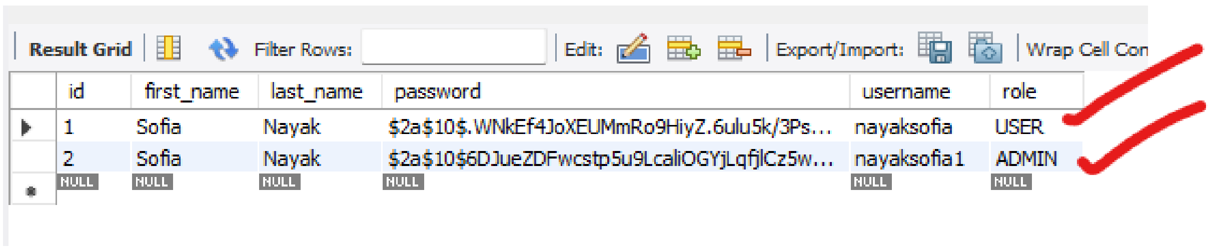
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token: eyJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJuYXlh...

Admin

Status: 200 OK Time: 48 ms Size: 450 B

```
Hello From Admin Only URL.
```



	id	first_name	last_name	password	username	role
▶	1	Sofia	Nayak	$2a$10$.WNkEf4JoXEUMmRo9HiyZ.6ulu5k/3Ps...	nayaksofia	USER
	2	Sofia	Nayak	$2a$10$6DJueZDFwcstp5u9LcaliOGYjLqfjlCz5w...	nayaksofia1	ADMIN
	NULL	NULL	NULL	NULL	NULL	NULL

Completed Successsfully.