Building Applications With Spring Boot

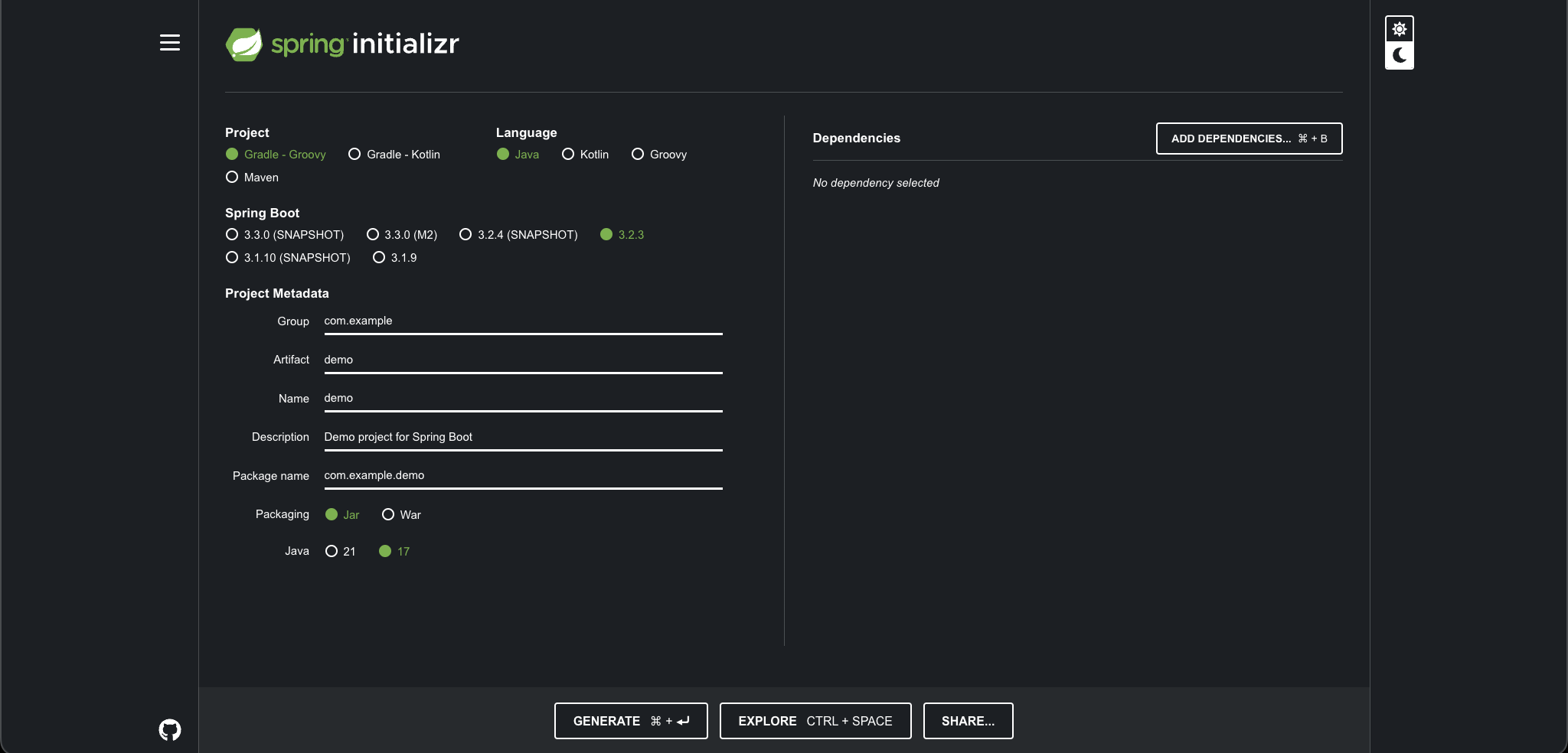
This document contains information on how to set up Spring Boot applications using Spring Initializr. Additionally, it has sections on some best practices for application development and integrating MongoDB and MySQL databases with Spring Boot.

# Spring Initializr

Spring Initializr is a web-based tool provided by the Spring team to bootstrap Spring Boot projects. It allows developers to quickly generate a basic project structure with the necessary dependencies and configurations to start building Spring-based applications.

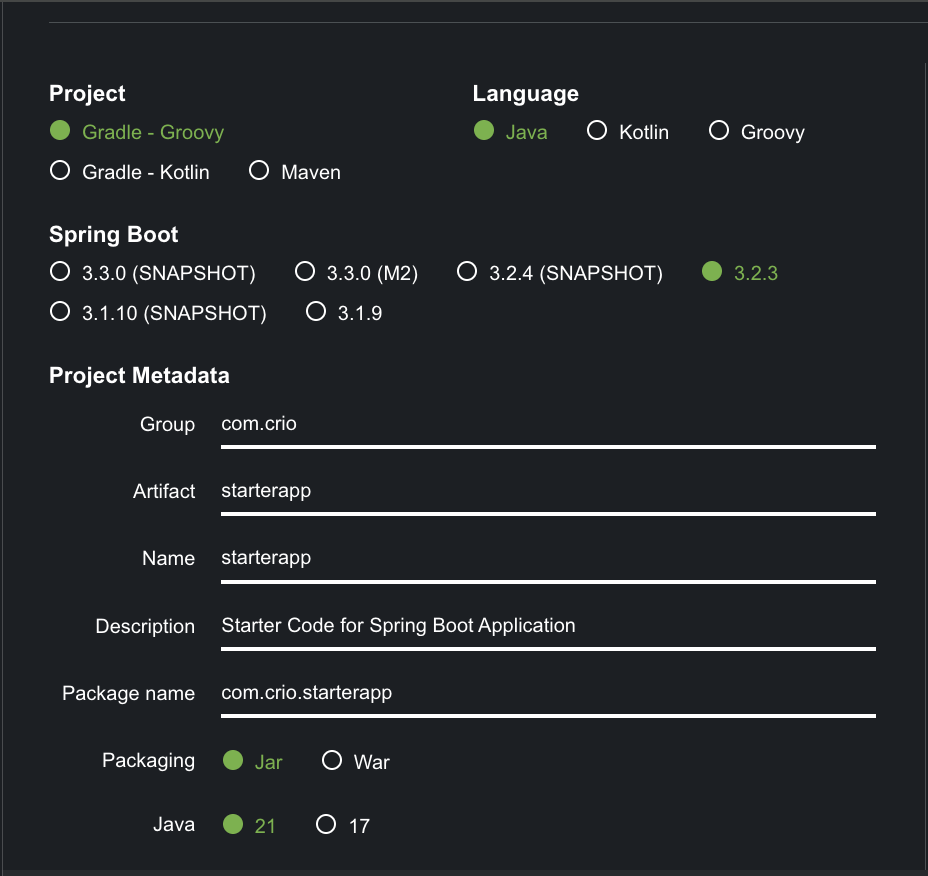
To get started with Spring Initializr, navigate to [Spring Initializr](https://start.spring.io/)

Ensure that you are seeing the homepage as seen in the screenshot below.



## Configuring The Project

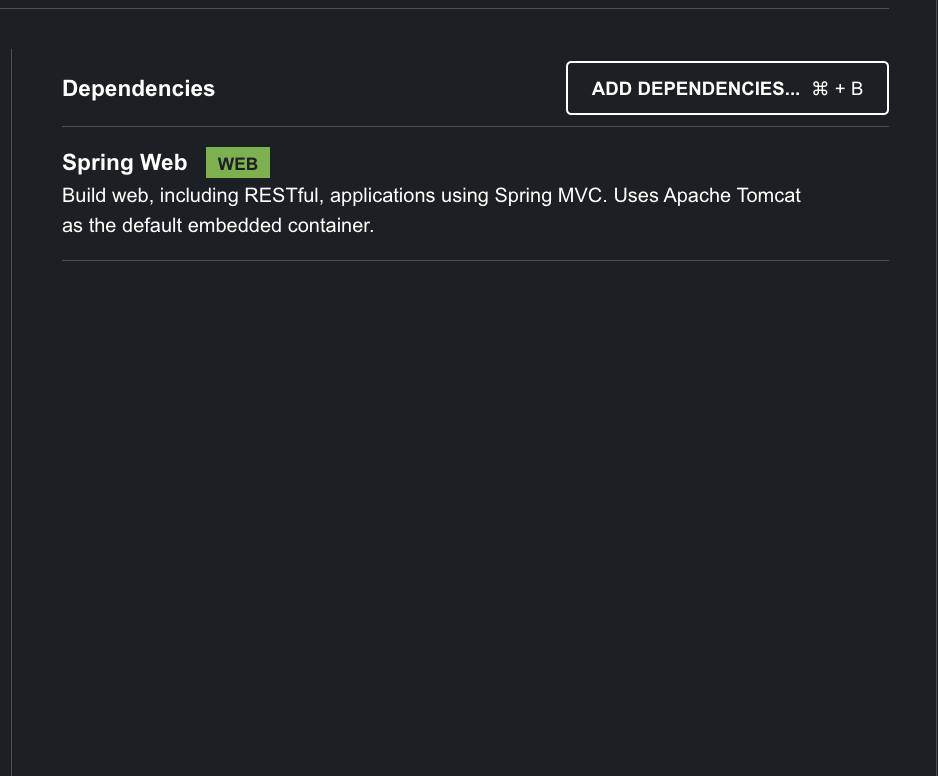
To configure the project and generate stub code with Spring Initializr, fill in the configuration form as seen in the screenshots below.



* Ensure that the build tool is selected as **Gradle - Groovy** and the language as **Java**.
* The Spring Boot version can be the latest available version. Avoid using M2 and Snapshot as these are not stable versions.
* In the project metadata, Group refers to the domain name, reversed. Artifact is the application name, which is the same as the name. The Description contains a short description of the application and the Package name is the combination of Group and Artifact.
* Ensure that packaging is set to Jar while choosing the latest Java version.

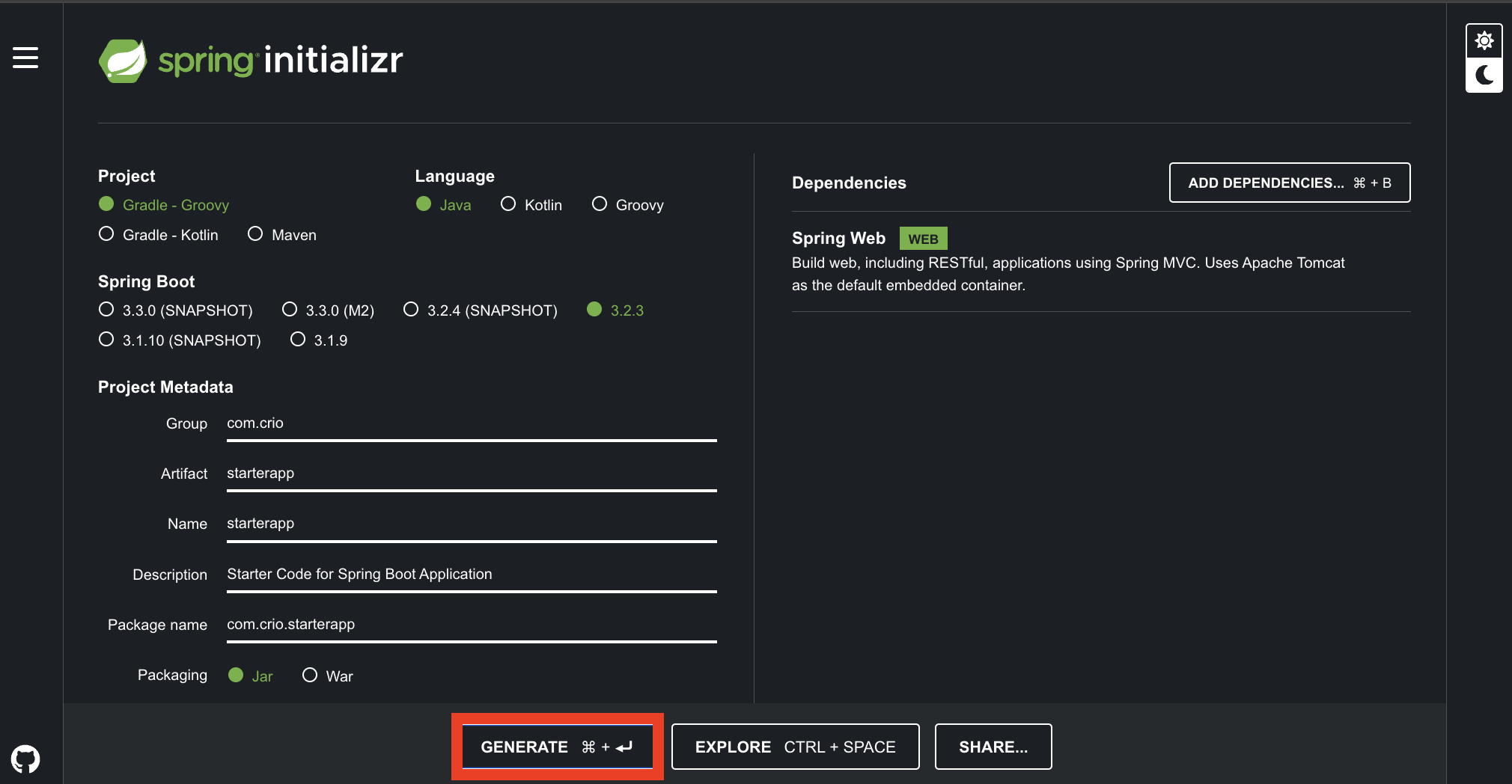
## Adding Dependencies

To get started, **Spring Web** is the only dependency that is required. Spring Web is used to build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Add Spring Web Dependency as seen in the screenshot below.

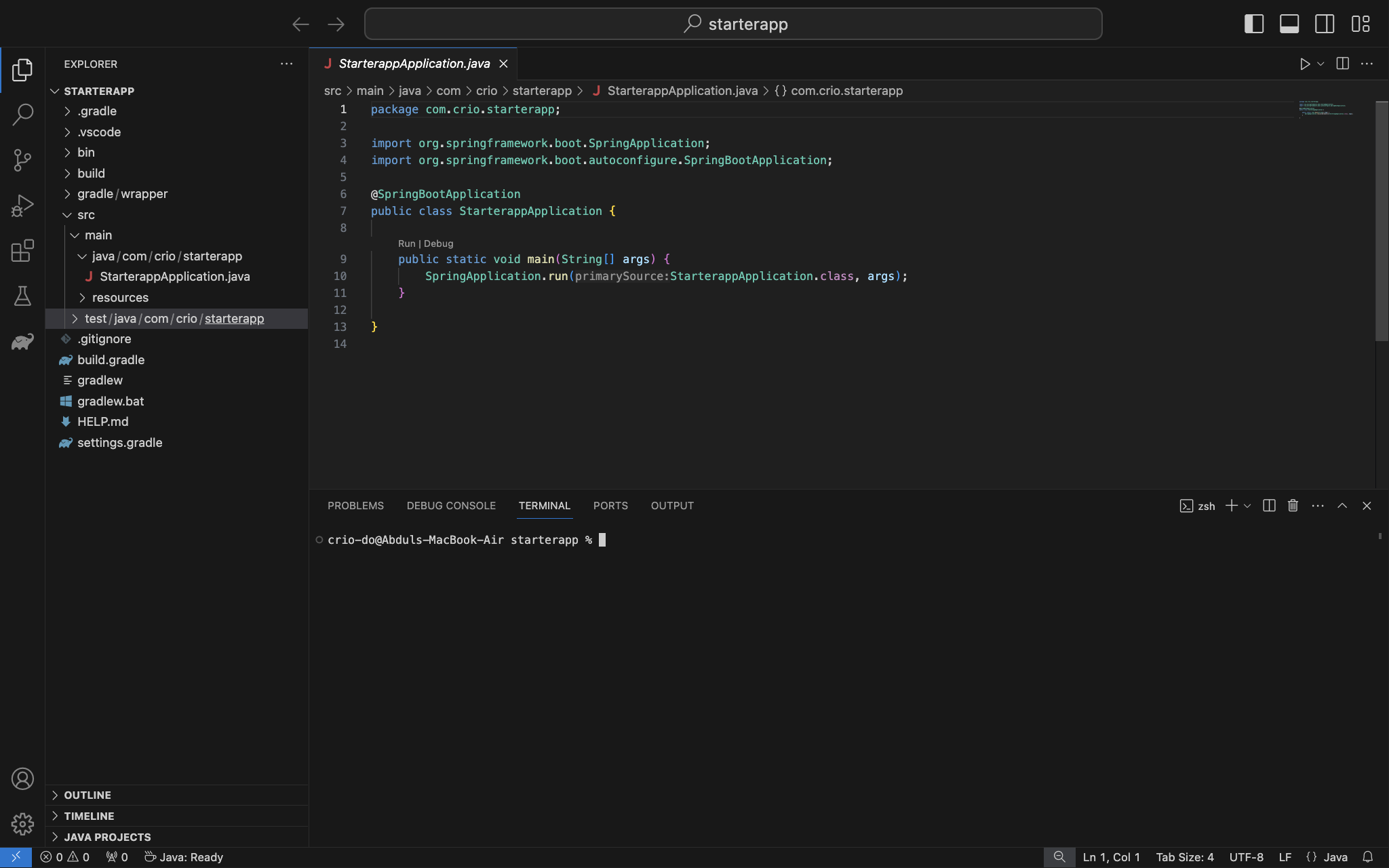
## Generating The Files

After configuring the project and adding dependencies, generate the files by clicking the “GENERATE” button as seen in the screenshot below.



## Next Steps - Running The Server

After generating the files, extract the .zip file and open it using VS Code. Ensure that it looks similar to the screenshot below.

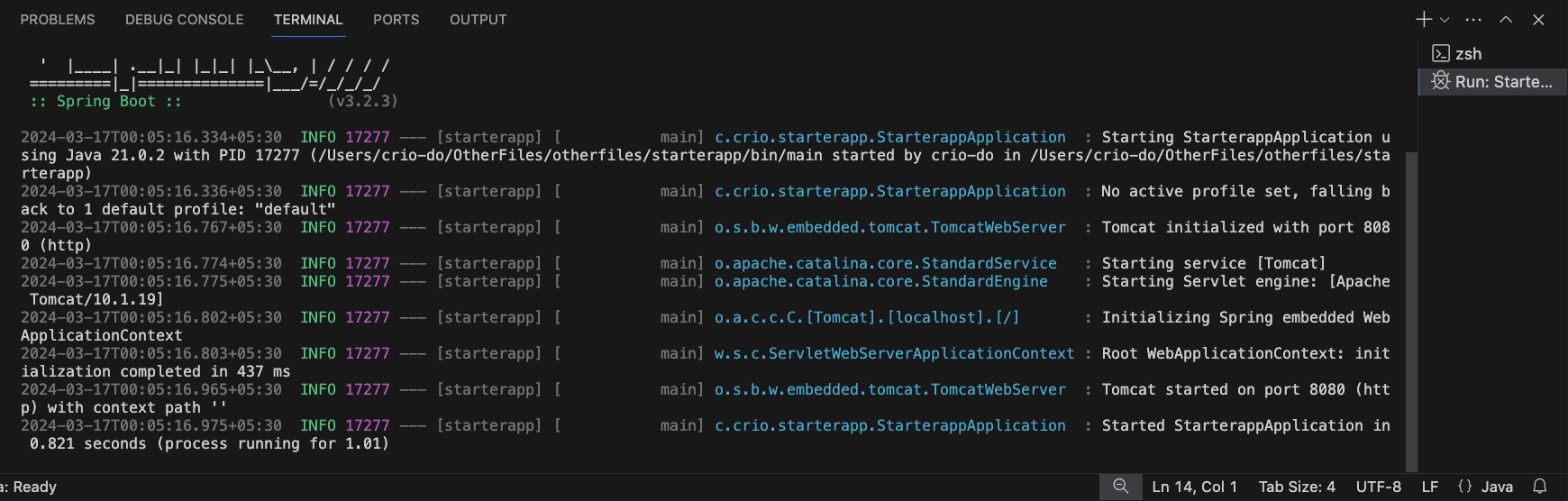


After setting up the files, run the server by running the StarterappApplication.java file

Alternatively, you can start the server by running the following command in the terminal:

./gradlew bootrun

Ensure the output looks like the screenshot below. The server must start on the specified port, the default port number is 8080.

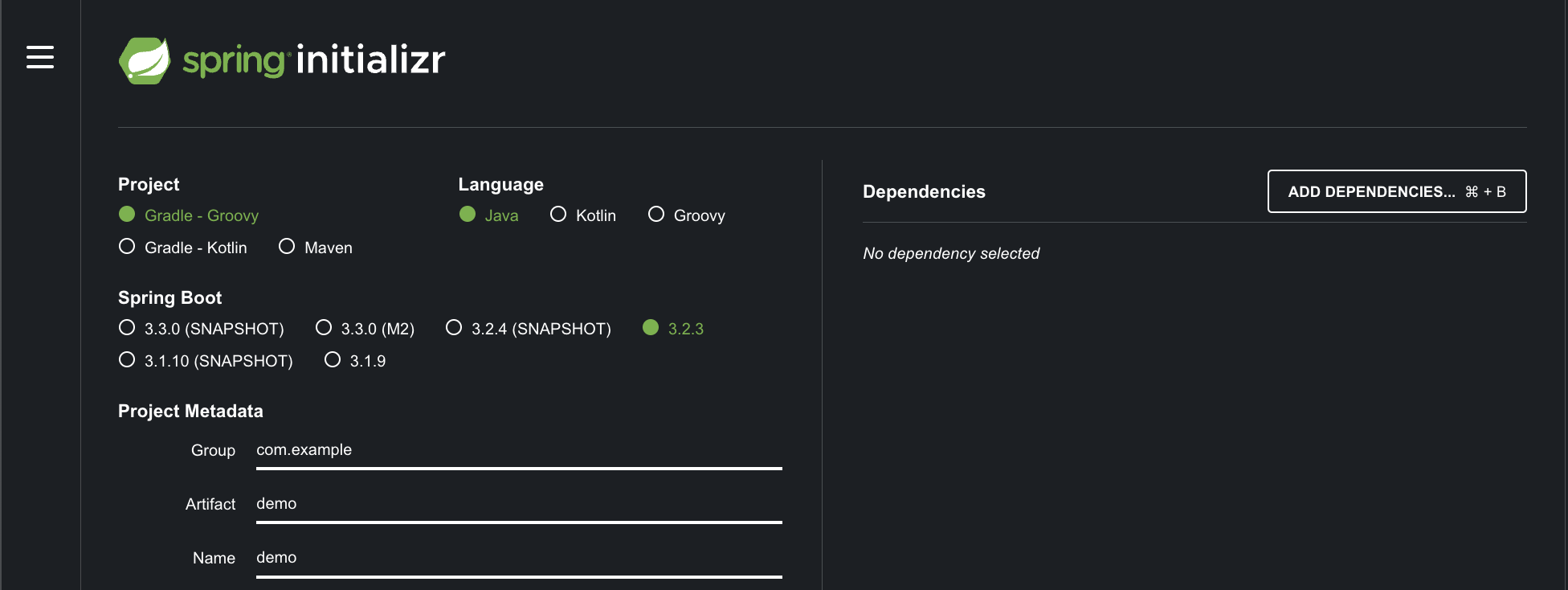
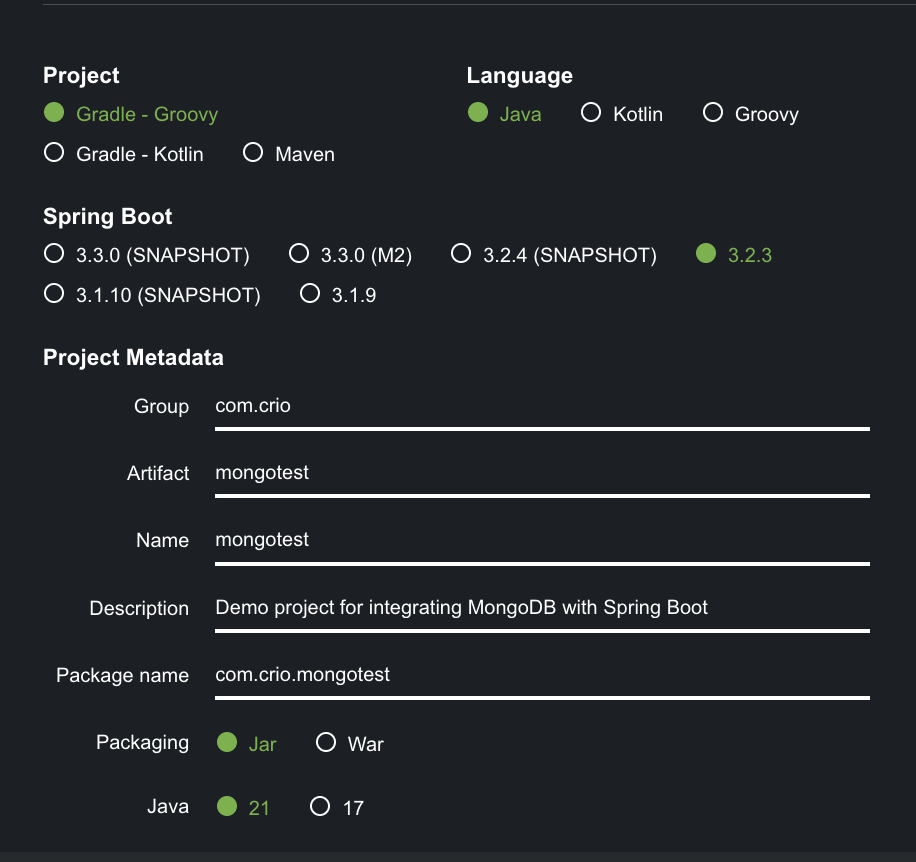
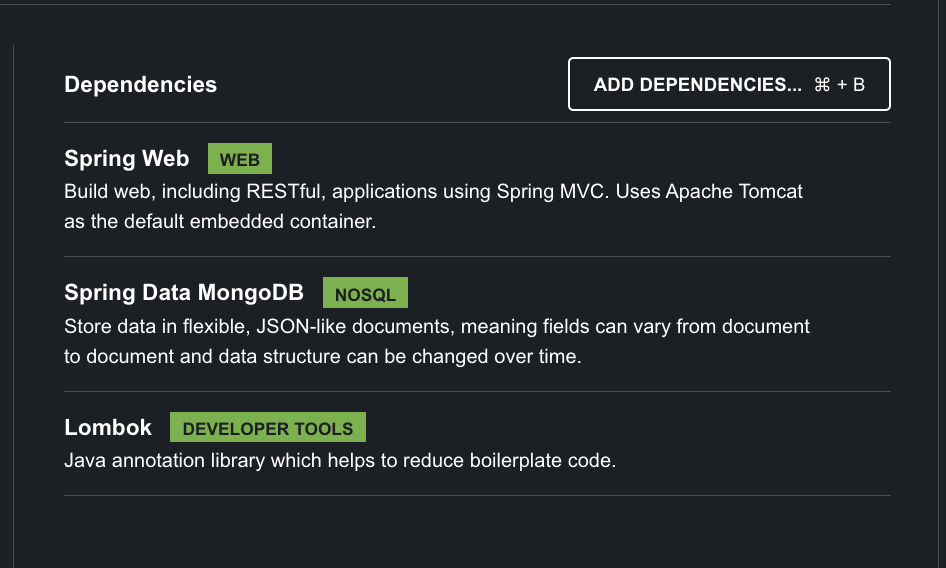
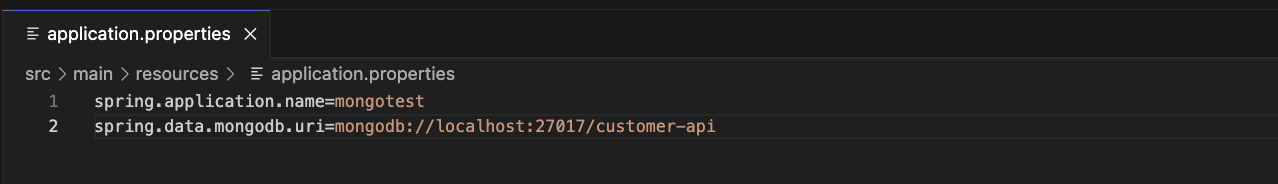


# Spring Boot Folder Structure (Best Practices)

A well-defined folder structure provides a clear layout of where to find different components of your application. It makes it easier for developers to locate specific files, packages, and resources, reducing the time spent searching for code. To learn more about folder structure and best practices in Spring Boot, refer to this article: [Spring Boot Folder Structure (Best Practices)](https://malshani-wijekoon.medium.com/spring-boot-folder-structure-best-practices-18ef78a81819)

# Integrating MongoDB with Spring Boot

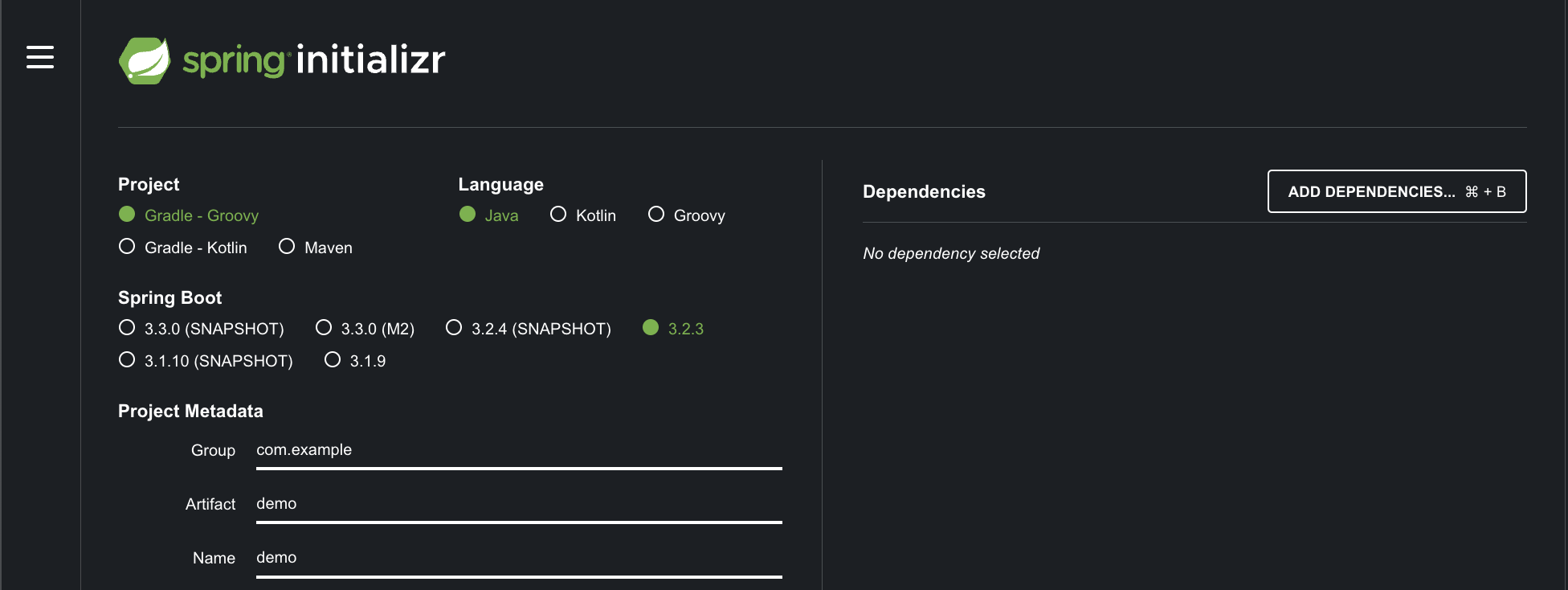
In this section, let us create a new Spring Boot project and integrate it with MongoDB while following the best practices. To get started, follow these steps:

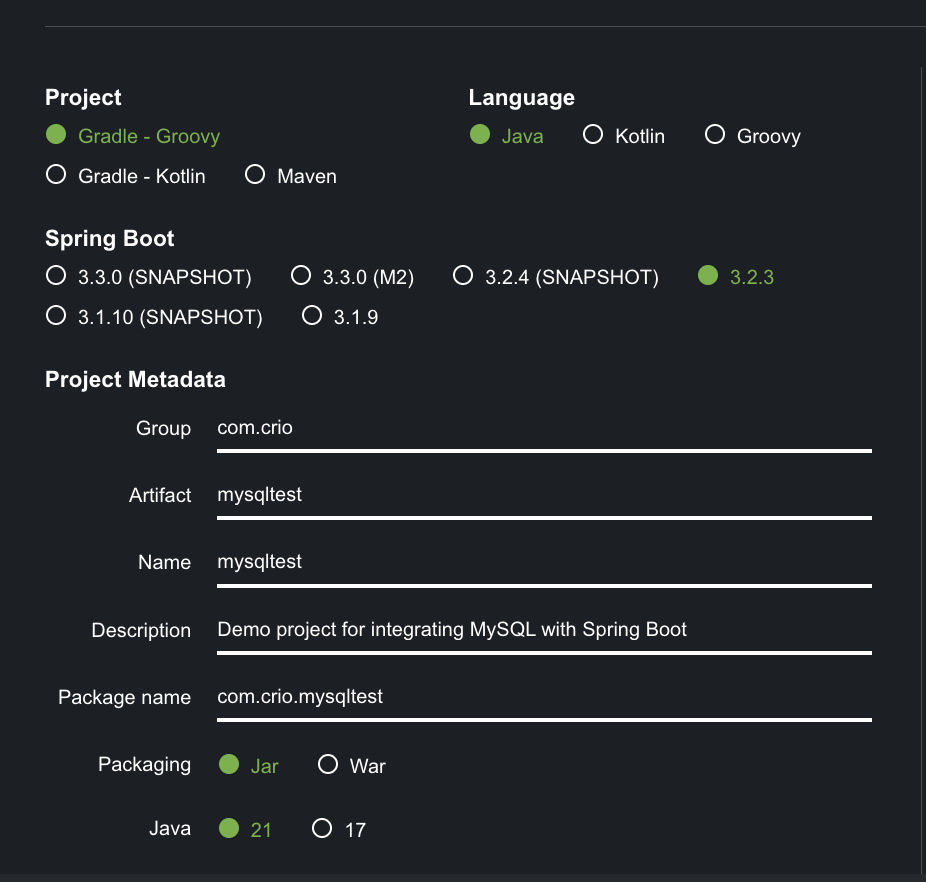
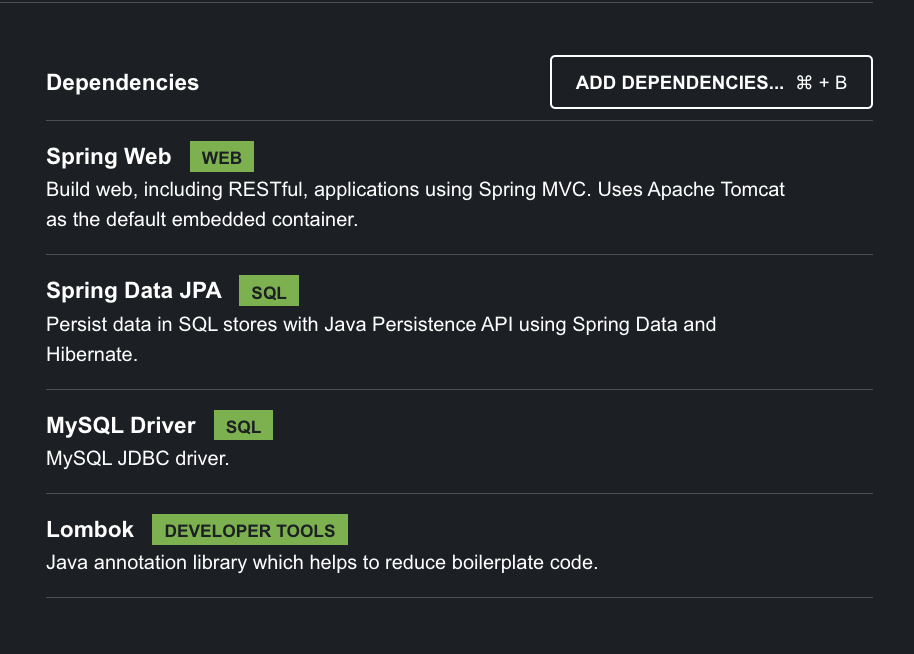
1. Navigate to the Spring Initializr Website: [Spring Initializr](https://start.spring.io/)
2. Configure the app as we did previously
3. In the dependencies section, add the following dependencies:
   1. Spring Web: Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
   2. Spring Data MongoDB: Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.
   3. Lombok: Java annotation library which helps to reduce boilerplate code.
4. Generate and extract the downloaded .zip file
5. To integrate MongoDB with Spring Boot, open the extracted folder in VS Code and navigate to: **src/main/resources/application.properties**
6. In the application.properties file, make the following addition: spring.data.mongodb.uri=mongodb://localhost:27017/customer-api
7. Here, “customer-api” is the name of the database, which will be created if it does not exist
8. Before running the Spring Boot server, ensure the MongoDB server is running as expected. ([Refer to the setup documentation](https://docs.google.com/document/d/1LbRboQXtkjvto8ftQnX0JnwjQsy96nECqyTimeMX7Fg/edit#heading=h.27vh9qp3iig9))
9. Run the src/main/java/com/crio/mongotest/MongotestApplication.java file and ensure that the Mongo driver starts as seen in the screenshot below
10. Create the layers and extend the MongoRepository interface as required. To learn more, navigate to the official documentation: [Getting Started | Accessing Data with MongoDB](https://spring.io/guides/gs/accessing-data-mongodb)

# Integrating MySQL with Spring Boot

In this section, let us create a new Spring Boot project and integrate it with MySQL while following the best practices. To get started, follow these steps:

1. Navigate to the Spring Initializr Website: [Spring Initializr](https://start.spring.io/)



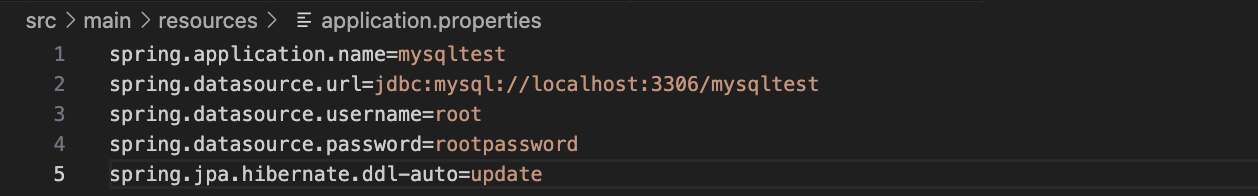
1. Configure the app as we did previously
2. In the dependencies section, add the following dependencies:
   1. Spring Web: Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
   2. Spring Data JPA: Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
   3. MySQL Driver: A driver library that facilitates the JDBC connection
   4. Lombok: Java annotation library which helps to reduce boilerplate code.
3. Generate and extract the downloaded .zip file
4. To integrate MySQL with Spring Boot, open the extracted folder in VS Code and navigate to: **src/main/resources/application.properties**
5. In the application.properties file, make the following additions:

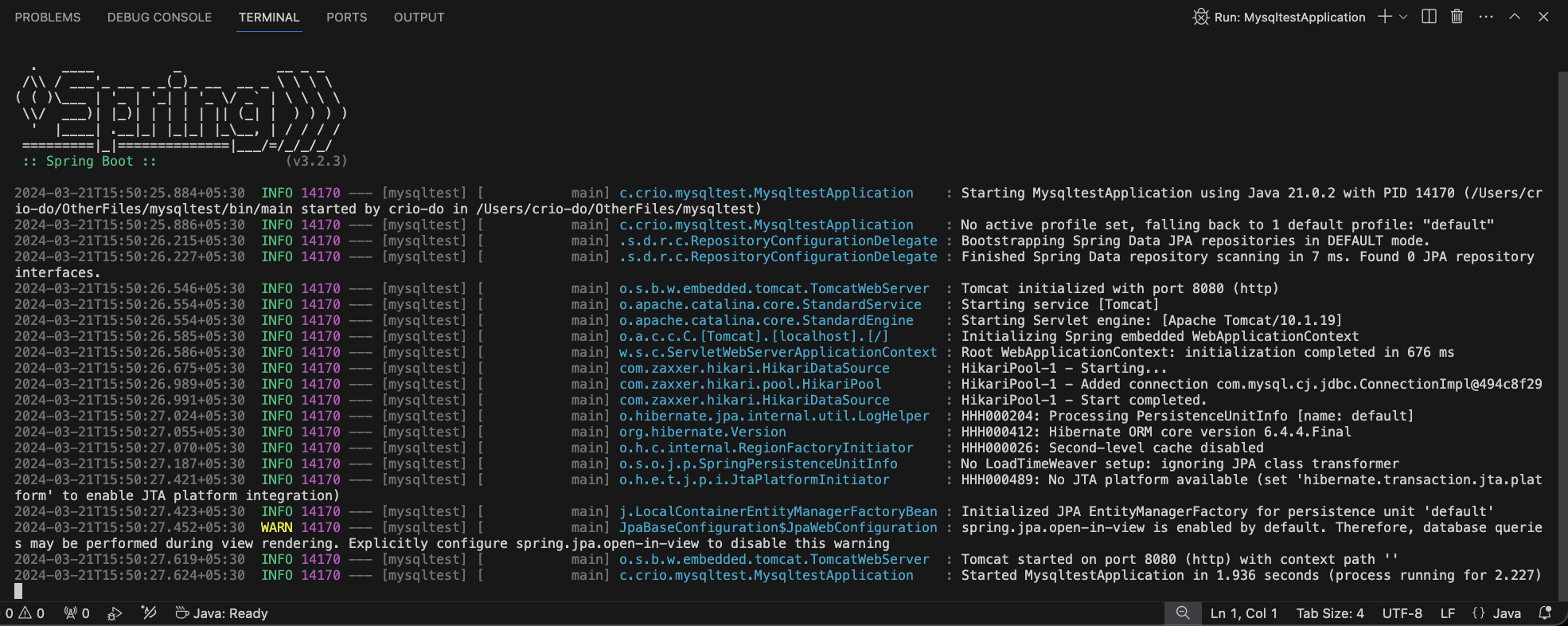
spring.datasource.url=jdbc:mysql://localhost:3306/mysqltemp

spring.datasource.username=root

spring.datasource.password=rootpassword

spring.jpa.hibernate.ddl-auto=update

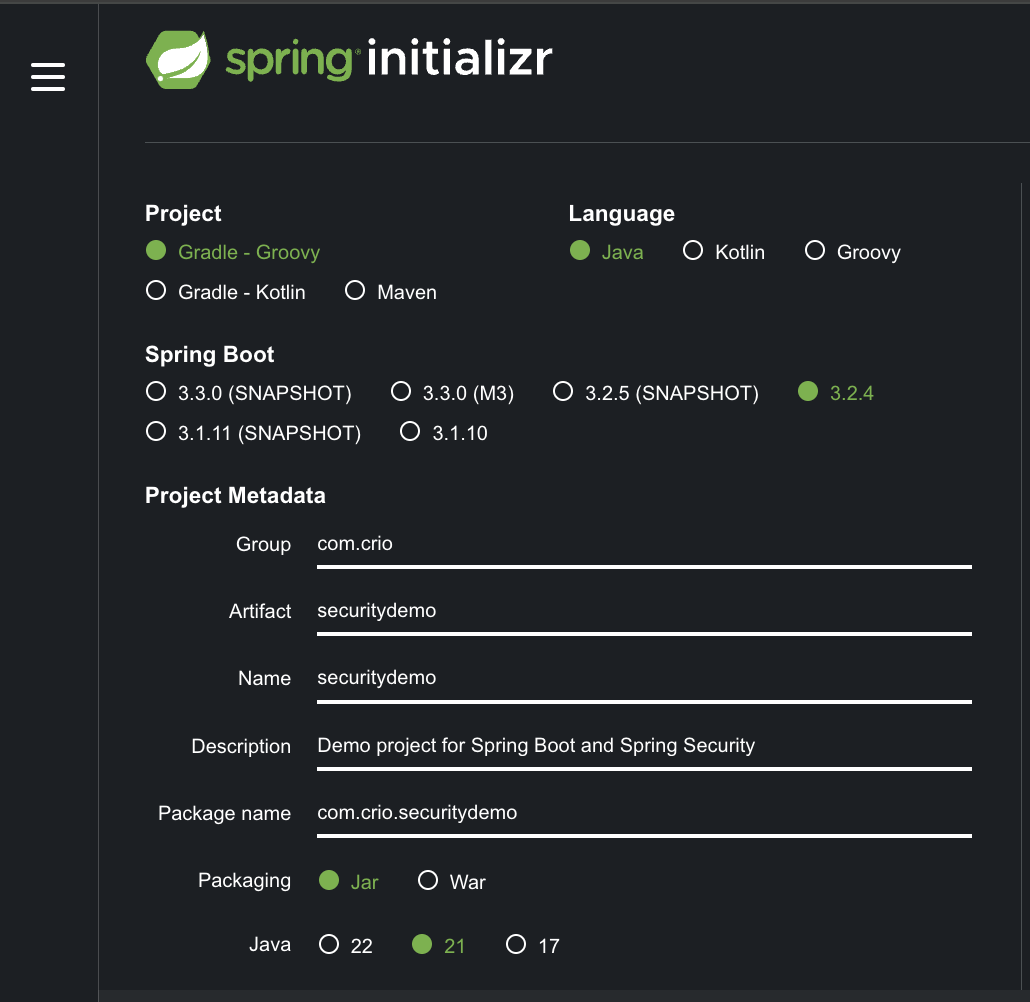


1. Here, “mysqltest” is the name of the database
2. Ensure that you assign **your** username and password values
3. Before running the Spring Boot server, ensure that the MySQL server is running as expected. ([Refer to the setup documentation](https://docs.google.com/document/d/1LbRboQXtkjvto8ftQnX0JnwjQsy96nECqyTimeMX7Fg))
4. If “mysqltest” database does not exist, Spring Boot will throw an error
5. To create the database, initiate the MySQL shell:
   1. For macOS, type the following command in the terminal:  
      mysql -u root -p  
      Enter the MySQL server password when prompted
   2. For windows, search for “MySQL Command Line Client”, if you do not have it installed, installing by referring to the [setup documentation](https://docs.google.com/document/d/1LbRboQXtkjvto8ftQnX0JnwjQsy96nECqyTimeMX7Fg)  
      Enter the MySQL server password when prompted
   3. To create the database, type the following command in the MySQL shell:  
      CREATE DATABASE mysqltest;
   4. Ensure that you are getting an output as seen in the screenshot below
6. Run the src/main/java/com/crio/mysqltest/MysqltestApplication.java file and ensure that the server starts successfully as seen in the screenshot below
7. Create the layers and extend the JpaRepository interface as required. To learn more, navigate to this documentation: [Introduction to Spring Data JPA | Baeldung](https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa)

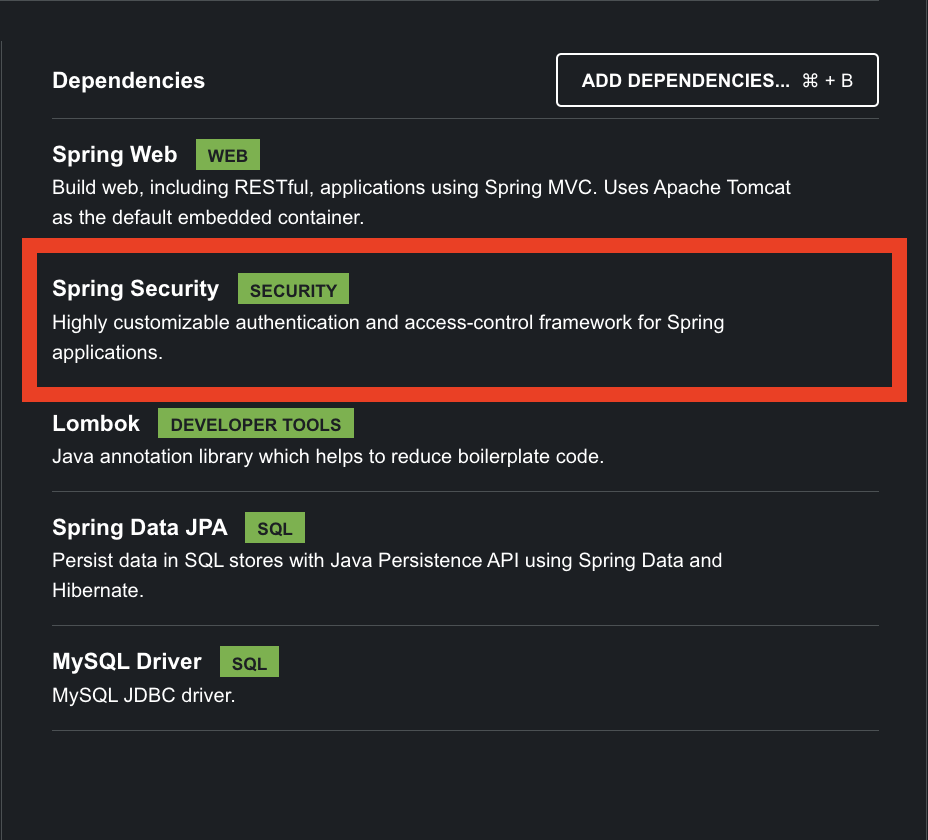
# Integrating Spring Security with Spring Boot

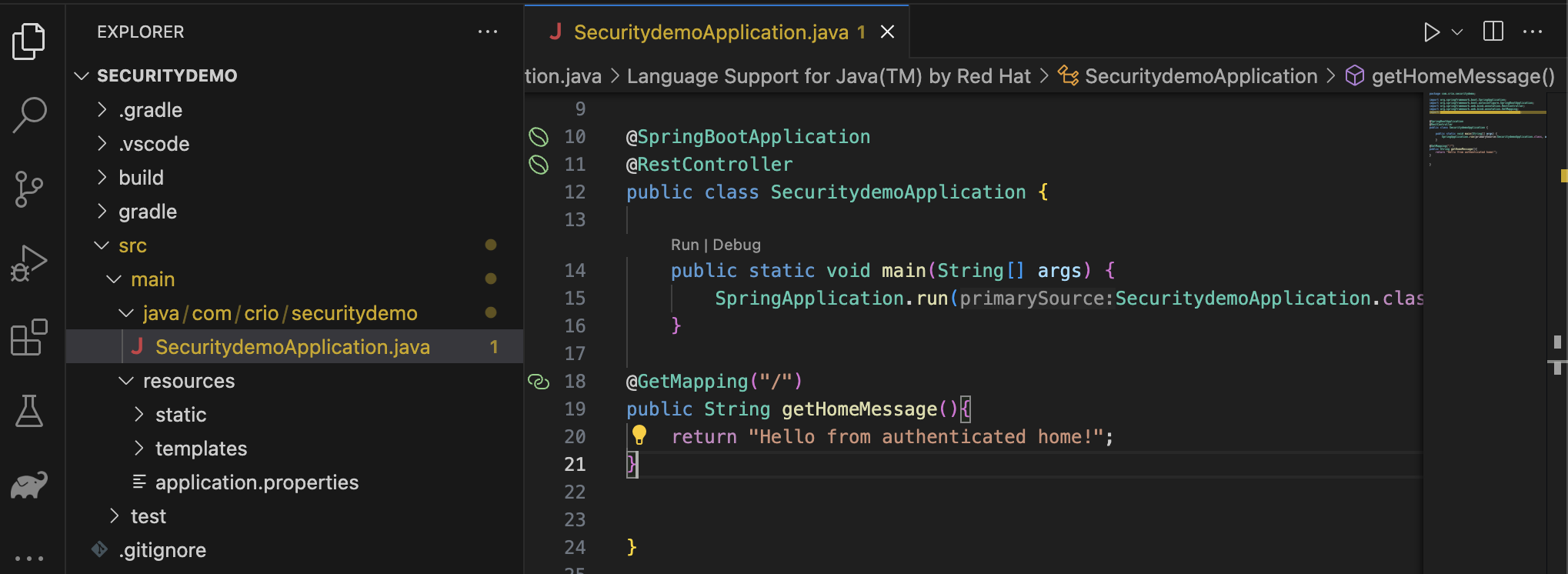
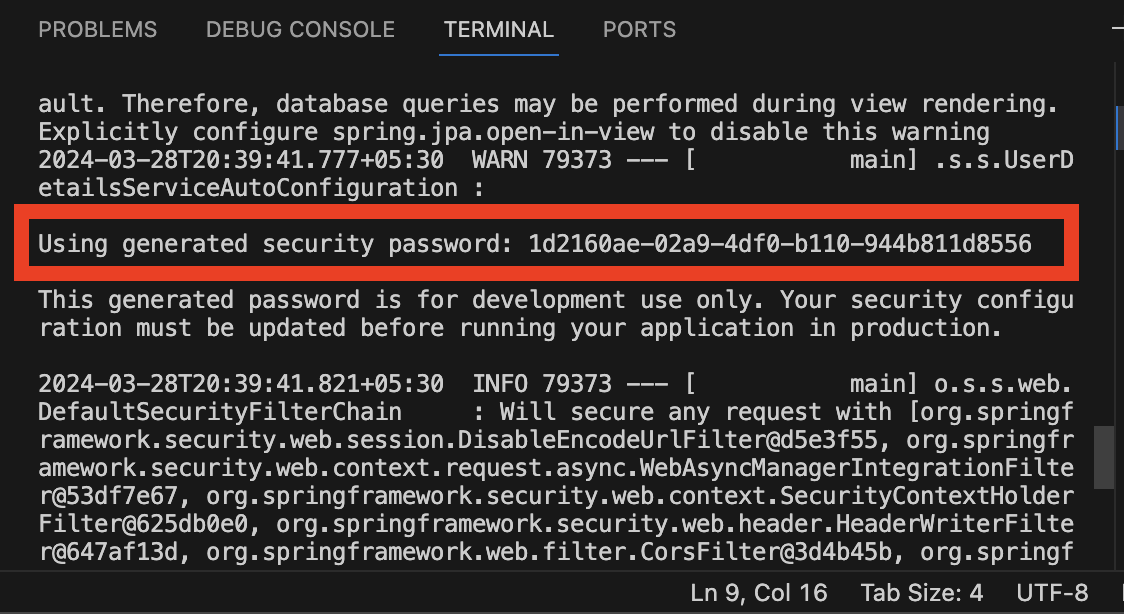
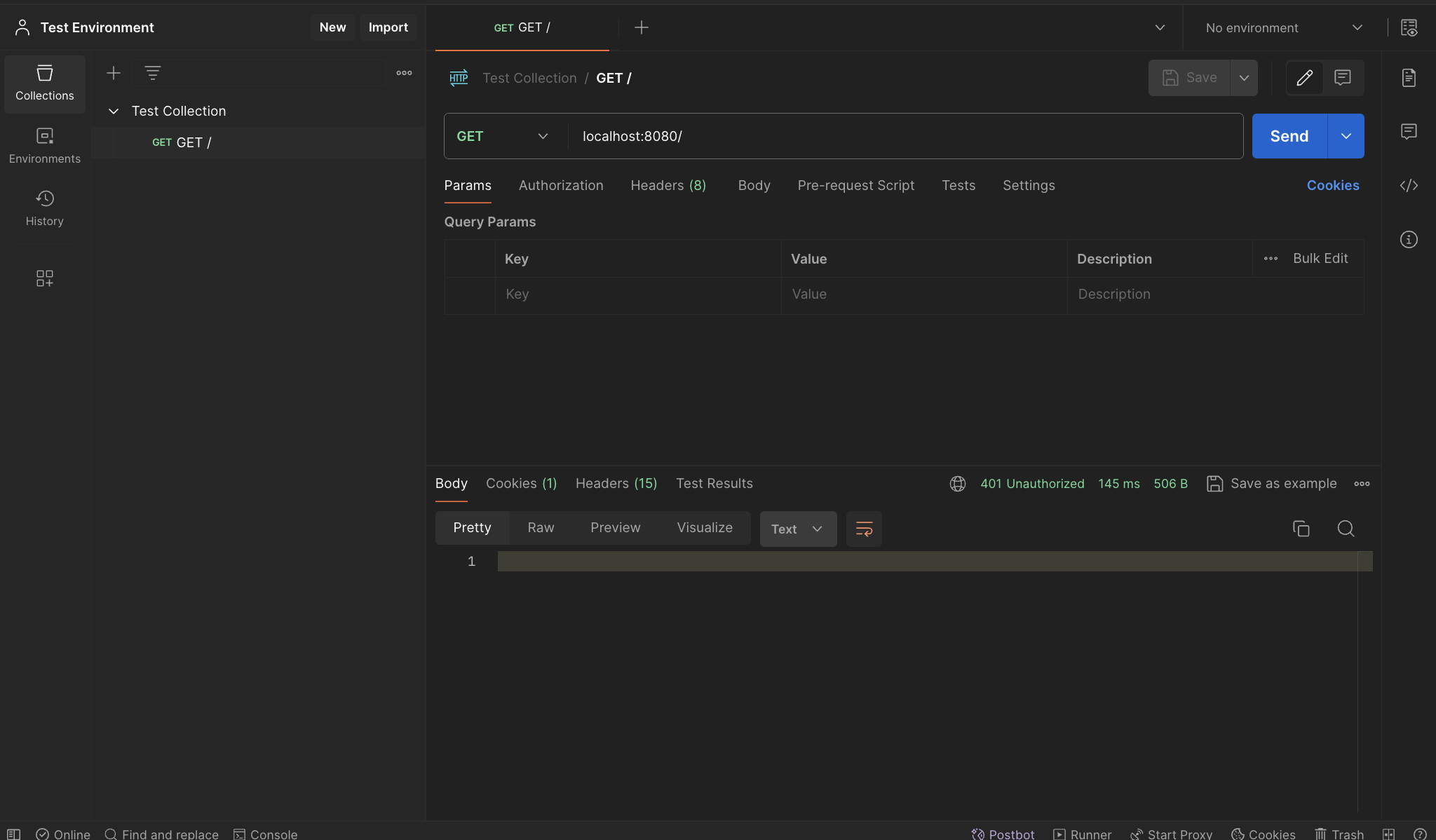
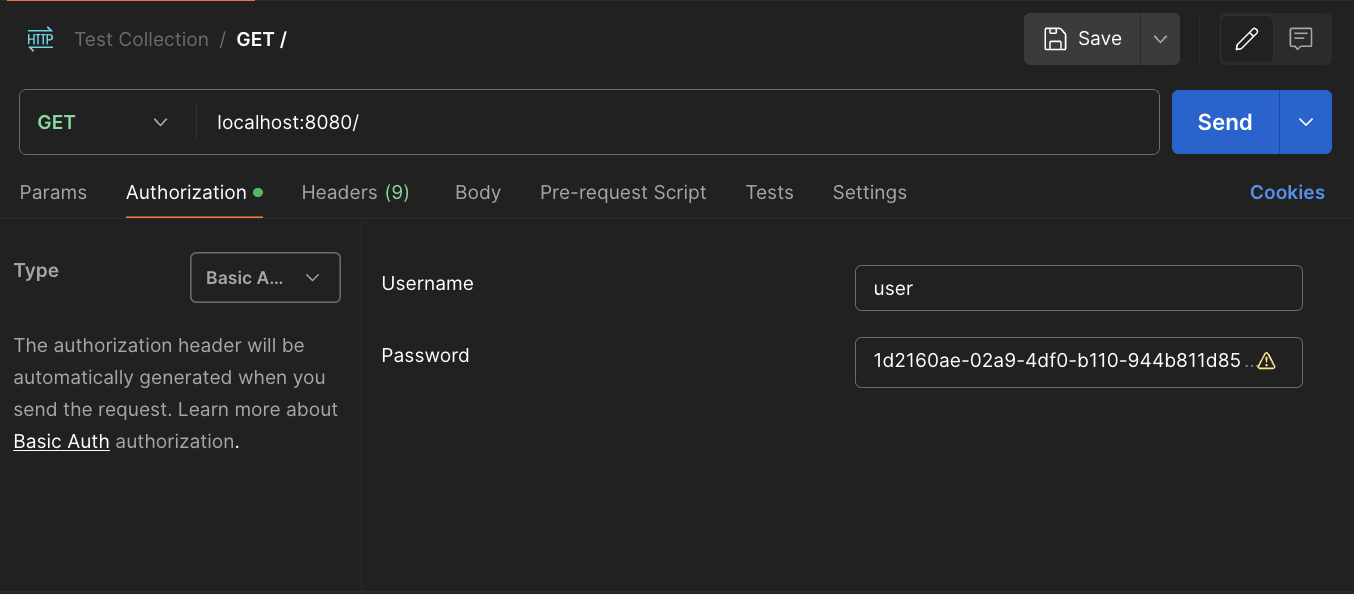
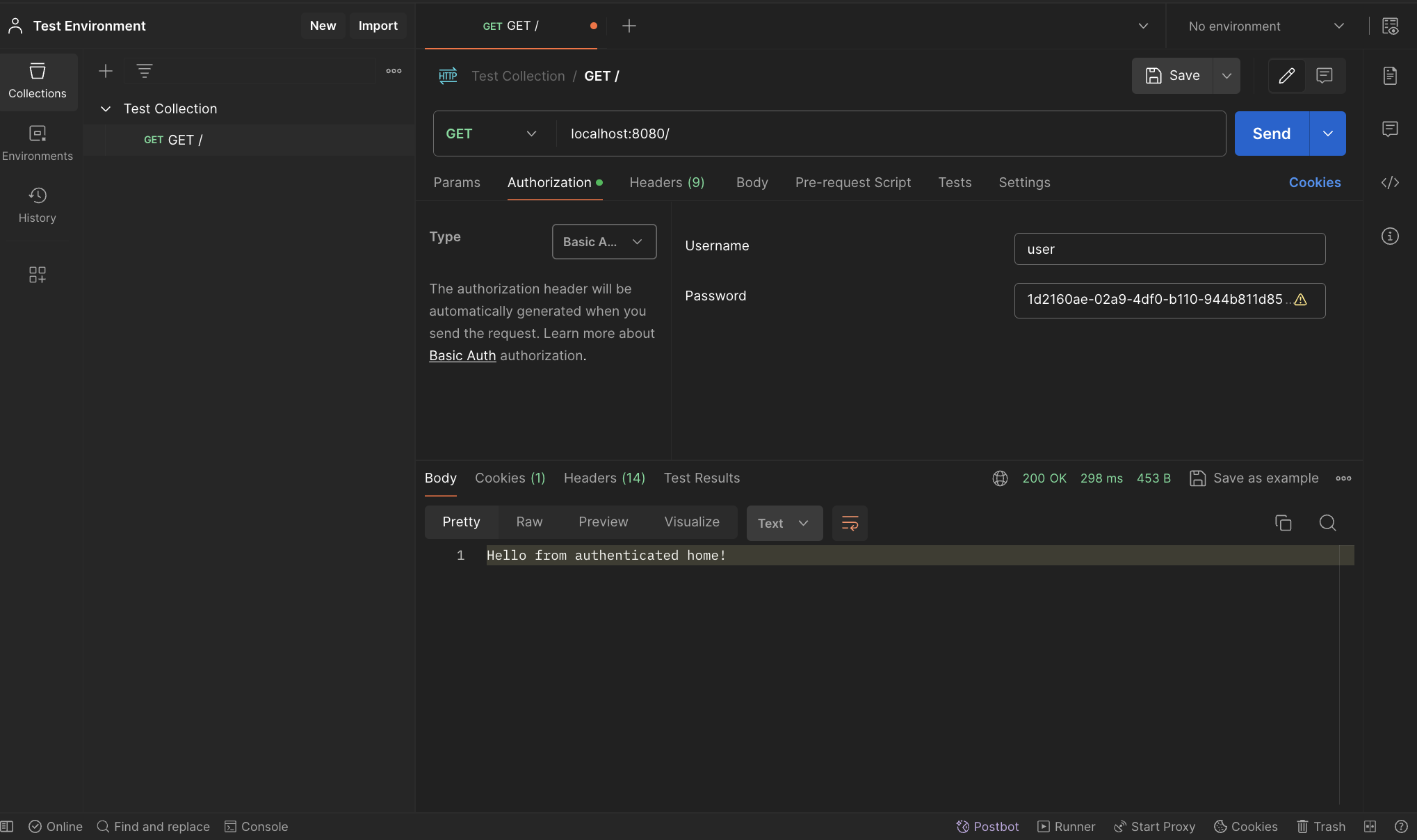
In this section, let us create a new Spring Boot project and integrate it with Spring Security. You are free to use a database of your choice. This example uses MySQL as the database. To get started, follow the steps below:

1. Navigate to the Spring Initializr Website: [Spring Initializr](https://start.spring.io/)
2. Configure the app as we did previously



1. In the dependencies section, add a dependency for Spring Security



1. Generate and extract the downloaded .zip file
2. Open the extracted folder in VS Code
3. Configure the database (Refer to the sections above)
4. Add an @RestController annotation to the main application class (src/main/java/com/crio/securitydemo/SecuritydemoApplication.java)
5. Create a GET endpoint to test Spring Security
6. Run the application - ./gradlew bootrun
7. By default, all the endpoints are secured by Spring Security
8. To access the endpoint that we have defined, find the randomly generated password in the terminal after running the Spring Application
9. Open Postman to send a GET request to the root directory (“/”)
10. Send a GET request to localhost:8080/ - It should return a “401 Unauthorized” error
11. To access the protected resource, configure the Authorization details as seen in the screenshot below:
    1. The type must be set to Basic Auth
    2. The default username for Spring Security is user
    3. The password is the randomly generated password (Previously obtained from the terminal)
12. Re-send the GET request after configuring the Authentication details. This time you should be able to view the response as seen in the screenshot below

Your application is now ready to use the power of Spring Security!