


Web API Design with Spring Boot Week 2 Coding Assignment

Points possible: 70


Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25



Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Here's a friendly tip: as you watch the videos, code along with the videos. This will help you with the homework. When a screenshot is required, look for the icon:  You will keep adding to this project throughout this part of the course. When it comes time for the final project, use this project as a starter.

Project Resources: <https://github.com/promineotech/Spring-Boot-Course-Student-Resources>


Coding Steps:

- 1) In the project you started last week, use Lombok to add an info-level logging statement in the controller implementation method that logs the parameters that were input to the method. Remember to add the `@Slf4j` annotation to the class.
- 2) Start the application (not an integration test). Use a browser to navigate to the application passing the parameters required for your selected operation. (A browser, used in this manner, sends an HTTP GET request to the server.) Produce a screenshot showing the browser navigation bar and the log statement that is in the IDE console showing that the controller method was reached (as in the video). 

- 3) With the application still running, use the browser to navigate to the OpenAPI documentation. Use the OpenAPI documentation to send a GET request to the server with a valid model and trim level. (You can get the model and trim from the provided `data.sql` file.) Produce a screenshot showing the curl command, the request URL, and the response headers. 
- 4) Run the integration test and show that the test status is green. Produce a screenshot of the test class and the status bar. 
- 5) Add a method to the test to return a list of expected Jeep (model) objects based on the model and trim level you selected. You can get the expected list of Jeeps from the file `src/test/resources/flyway/migrations/V1.1__Jeep_Data.sql`. So, for example, using the model Wrangler and trim level "Sport", the query should return two rows:


	Row 1	Row 2
Model ID	WRANGLER	WRANGLER
Trim Level	Sport	Sport
Num Doors	2	4
Wheel Size	17	17
Base Price	\$28,475.00	\$31,975.00

The method should be named `buildExpected()`, and it should return a List of Jeep. The video put this method into a support superclass but you can include it in the main test class if you want.

- 6) Write an AssertJ assertion in the test to assert that the actual list of jeeps returned by the server is the same as the expected list. Run the test. Produce a screenshot showing...
 - a) The test with the assertion.
 - b) The JUnit status bar (should be red).
 - c) The method returning the expected list of Jeeps. 
- 7) Add a service layer in your application as shown in the videos:
 - a) Add a package named `com.promineotech.jeep.service`.
 - b) In the new package, create an interface named `JeepSalesService`.
 - c) In the same package (service), create a class named `DefaultJeepSalesService` that implements the `JeepSalesService` interface. Add the class-level annotation, `@Service`.
 - d) Inject the service interface into `DefaultJeepSalesController` using the `@Autowired` annotation. The instance variable should be private, and the variable should be named `jeepSalesService`.


- e) Define the `fetchJeeps` method in the interface. Implement the method in the service class. Call the method from the controller (make sure the controller returns the list of Jeeps returned by the service method). The method signature looks like this:

```
List<Jeep> fetchJeeps(JeepModel model, String trim);
```

- f) Add a Lombok info-level log statement in the service implementation showing that the service was called. Print the parameters passed to the method. Let the method return `null` for now.
- g) Run the test again. Produce a screenshot showing the service class implementation, the log line in the console, and the red status bar. 
- 8) Add the database dependencies described in the video to the POM file (MySQL driver and Spring Boot Starter JDBC). To find them, navigate to <https://mvnrepository.com/>. Search for `mysql-connector-j` and `spring-boot-starter-jdbc`. In the POM file you don't need version numbers for either dependency because the version is included in the Spring Boot Starter Parent.
- 9) Create `application.yaml` in `src/main/resources`. Add the `spring.datasource.url`, `spring.datasource.username`, and `spring.datasource.password` properties to `application.yaml`. The url should be the same as shown in the video (`jdbc:mysql://localhost:3306/jeep`). The password and username should match your setup. If you created the database under your root user, the username is "root", and the password is the root user password. If you created a "jeep" user or other user, use the correct username and password.

Be careful with the indentation! YAML allows hierarchical configuration but it reads the hierarchy based on the indentation level. The keyword "spring" MUST start in the first column. It should look similar to this when done:

```
spring:
  datasource:
    username: username
    password: password
    url: jdbc:mysql://localhost:3306/jeep
```

- 10) Start the application (the real application, not the test). Produce a screenshot that shows `application.yaml` and the console showing that the application has started with no errors. 
- 11) Add the H2 database as dependency. Search for the dependency in the Maven repository like you did above. Search for "h2" and pick the latest version. Again, you don't need the version number, but the scope should be set to "test".
- 12) Create `application-test.yaml` in `src/test/resources`. Add the setting `spring.datasource.url` that points to the H2 database. It should look like this:

```
spring:
  datasource:
    url: jdbc:h2:mem:jeep
```

You do not need to set the username and password because the in-memory H2 database does not require them.

Produce a screenshot showing application-test.yaml1.

Screenshots of Code:

The screenshot displays a Spring Boot application interface with three main sections:

- Code Editor:** Shows the `DefaultJeepSalesController` class with a `fetchJeeps` method that logs the model and trim values.
- Console:** Displays the application startup logs, including the Spring Boot version (v2.7.0) and the initialization of the Tomcat web server.
- REST Client:** A form for testing the `fetchJeeps` endpoint. The URL is `localhost:8080/jeeps?model=WRANGLER&trim=Sport`. The parameters are `model=WRANGLER` and `trim=Sport`. The response shows a list of Jeeps with their model and trim values.

```
1 package com.promineotech.jeepp.controller;
2
3 import java.util.List;
4 import org.springframework.web.bind.annotation.RestController;
5 import com.promineotech.jeepp.entity.Jeep;
6 import com.promineotech.jeepp.entity.JeepModel;
7 import lombok.extern.slf4j.Slf4j;
8
9 @RestController
10 @Slf4j
11 public class DefaultJeepSalesController implements JeepSalesController {
12
13     @Override
14     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
15         log.info("model={}, trim={}", model, trim);
16         return null;
17     }
18 }
19
20
```

Spring Boot v2.7.0

2022-05-25 19:04:27.112 INFO 31892 --- [restartedMain] com.promineotech.jeepp.JeepSales : Starting JeepSales using Java 17.0.2 on Felix-MacBook-Pro-local
2022-05-25 19:04:27.113 INFO 31892 --- [restartedMain] com.promineotech.jeepp.JeepSales : No active profile set, falling back to 1 default profile: "default"
2022-05-25 19:04:27.144 INFO 31892 --- [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-prop
2022-05-25 19:04:27.144 INFO 31892 --- [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging
2022-05-25 19:04:27.763 INFO 31892 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-05-25 19:04:27.769 INFO 31892 --- [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-25 19:04:27.770 INFO 31892 --- [restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.63]
2022-05-25 19:04:27.811 INFO 31892 --- [restartedMain] o.s.c.c.c.(Tomcat).[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-25 19:04:27.811 INFO 31892 --- [restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 667 ms
2022-05-25 19:04:28.257 INFO 31892 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-05-25 19:04:28.279 INFO 31892 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-05-25 19:04:28.287 INFO 31892 --- [restartedMain] com.promineotech.jeepp.JeepSales : Started JeepSales in 1.367 seconds (JVM running for 1.949)
2022-05-25 19:08:08.112 INFO 31892 --- [nio-8080-exec-1] o.a.c.c.C.(Tomcat).[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-05-25 19:08:08.112 INFO 31892 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-05-25 19:08:08.113 INFO 31892 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2022-05-25 19:08:08.137 INFO 31892 --- [nio-8080-exec-1] c.p.j.c.DefaultJeepSalesController : model=WRANGLER, trim=Sport

localhost:8080/jeeps?model=WRANGLER&trim=Sport

Returns a List Of Jeeps given an optional model and/or trim

Parameters

Name Description

model * required
string (query)
The model name (i.e., 'WRANGLER')
RENEGADE

trim * required
string (query)
The trim level (i.e., 'Sport')
Sport

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/jeeps?model=RENEGADE&trim=Sport' \
  -H 'accept: application/json'
```

Request URL

http://localhost:8080/jeeps?model=RENEGADE&trim=Sport

Console

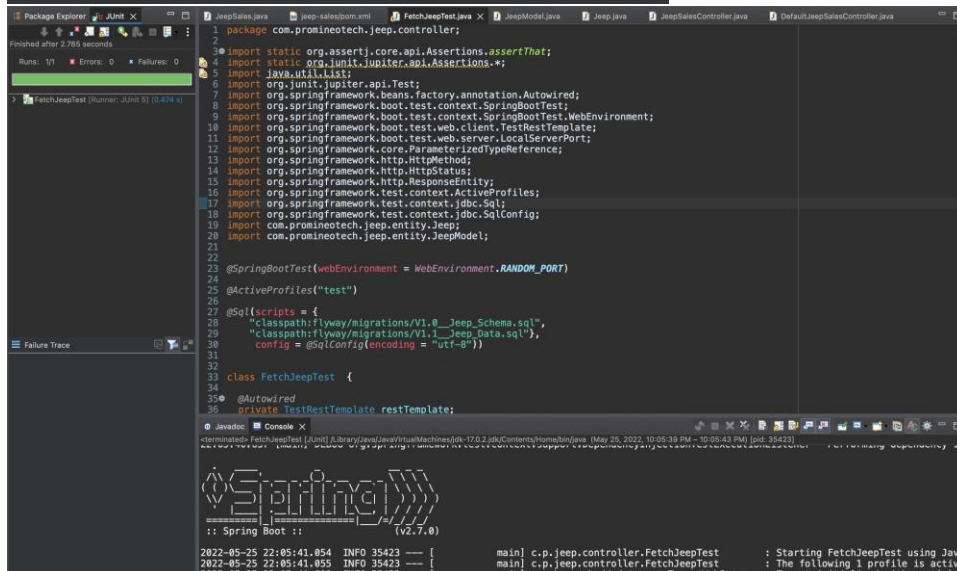
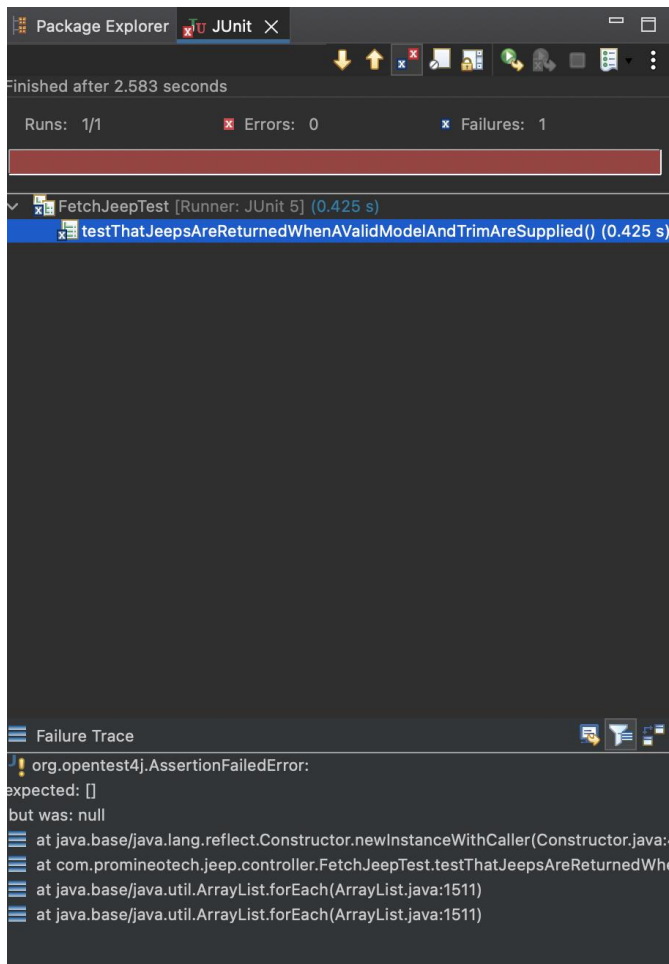
```
jeep-sales - JeepSales [Spring Boot App] [Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java (May 25, 2022, 7:04:24 PM) [pid: 31892]
2022-05-25 21:43:58.017 INFO 31892 --- [nio-8080-exec-9] c.p.j.c.DefaultJeepSalesController : model=GLADIATOR, trim=Sport
2022-05-25 21:46:05.619 INFO 31892 --- [nio-8080-exec-1] c.p.j.c.DefaultJeepSalesController : model=RENEGADE, trim=Sport
2022-05-25 21:56:43.714 INFO 31892 --- [nio-8080-exec-2] c.p.j.c.DefaultJeepSalesController : model=COMPASS, trim=Sport
```

Code Details

200

Response headers

```
connection: keep-alive
content-length: 0
date: Thu, 26 May 2022 04:56:43 GMT
keep-alive: timeout=60
```



```
1 package com.promineotech.jee.service;
2
3 import java.util.List;
4
5 @Slf4j
6 public class DefaultJeepSaleService implements JeepSalesService {
7
8     @Override
9     public List<Jeep> fetchJeeps(JeepModel model, String trim) {
10         log.info("The fetchjeeps method was called with model={} and trim={}", model, trim);
11         return null;
12     }
13 }
14
15
16
17
18
19
```

Finished after 1.693 seconds

Runs: 1/1

Errors: 1

Failures: 0

FetchJeepTest [Runner: JUnit 5] (0.000 s)

testThatJeepsAreReturnedWhenAValidModelAndTrimAreSupplied() (0.000 s)

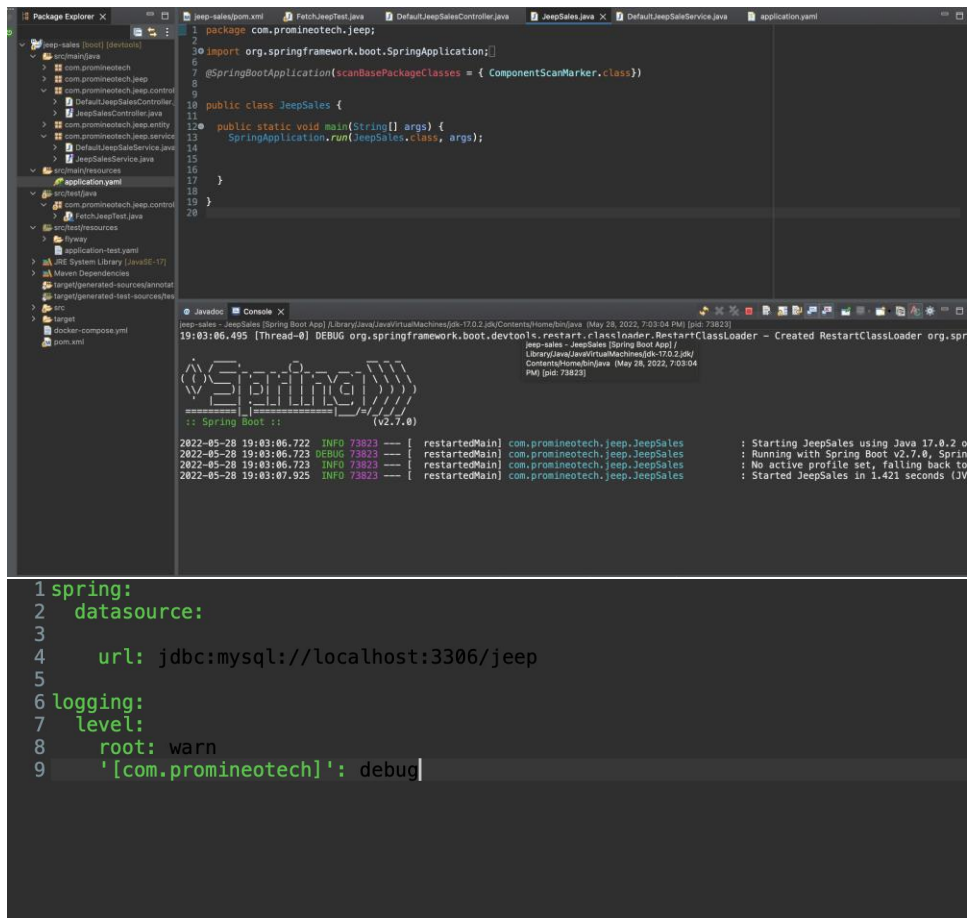
```
<terminated> FetchJeepTest [JUnit] /Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java (May 28, 2022, 5:55:37 PM - 5:55:41 PM) [pid: 70658]
at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:786) ~[spring-beans-5.3.18.jar:5.3.18]
at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:863) ~[spring-context-5.3.18.jar:5.3.18]
at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:583) ~[spring-context-5.3.18.jar:5.3.18]
at org.springframework.boot.web.servlet.context.ServletWebServerApplicationContext.refresh(ServletWebServerApplicationContext.java:147) ~[spring-boot-2.7.0.jar:2.7.0]
at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:734) ~[spring-boot-2.7.0.jar:2.7.0]
at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:408) ~[spring-boot-2.7.0.jar:2.7.0]
at org.springframework.boot.SpringApplication.run(SpringApplication.java:308) ~[spring-boot-2.7.0.jar:2.7.0]
at org.springframework.boot.test.context.SpringBootContextLoader.loadContext(SpringBootContextLoader.java:132) ~[spring-boot-test-2.7.0.jar:2.7.0]
at org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate.loadContextInternal(DefaultCacheAwareContextLoaderDelegate.java:43) ~[spring-test-5.3.18.jar:5.3.18]
at org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate.loadContext(DefaultCacheAwareContextLoaderDelegate.java:144) ~[spring-test-5.3.18.jar:5.3.18]
... 72 common frames omitted
Caused by: org.springframework.beans.factory.NoSuchBeanDefinitionException: No qualifying bean of type 'com.promineotech.jeeptest.service.JeeptestService'
at org.springframework.beans.factory.support.DefaultListableBeanFactory.raiseNoMatchingBeanFound(DefaultListableBeanFactory.java:1895) ~[spring-beans-5.3.18.jar:5.3.18]
at org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.java:1311) ~[spring-beans-5.3.18.jar:5.3.18]
at org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.java:1259) ~[spring-beans-5.3.18.jar:5.3.18]
at org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.resolveFieldValue(AutowiredAnnotationBeanPostProcessor.java:658) ~[spring-beans-5.3.18.jar:5.3.18]
... 92 common frames omitted

Package Explorer | JUnit | FetchJeepTest... | Jeeptest.java | JeeptestController.java | JeeptestService.java | DefaultJeeptestService.java | JeeptestTest.java | JeeptestTestController.java | DefaultJeeptestTestController.java
Finished after 2.854 seconds
Runs: 1/1 | Errors: 0 | Failures: 1
testThatJeeptestReturnedWhenCalledModelAn

Failure Trace
org.opentest4j.AssertionFailedError:
expected:
[Jeep(modelPK=null, modelId=WRANGLER, trimLevel=Sport)]
but was:
null
at java.base/java.lang.reflect.Constructor.newInstance()
at com.promineotech.jeeptest.controller.FetchJeepTest.fetchJeep()
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

Spring
:: Spring Boot ::
(v2.7.0)

2022-05-28 18:18:16.183 INFO 71739 --- [main] c.p.jeeptest.controller.FetchJeepTest : Starting FetchJeepTest using
2022-05-28 18:18:16.184 DEBUG 71739 --- [main] c.p.jeeptest.controller.FetchJeepTest : Running with Spring Boot v2.7.0
2022-05-28 18:18:16.184 INFO 71739 --- [main] c.p.jeeptest.controller.FetchJeepTest : The following 1 profile is active
2022-05-28 18:18:17.656 INFO 71739 --- [main] c.p.jeeptest.controller.FetchJeepTest : Started FetchJeepTest in 1.93
2022-05-28 18:18:17.999 DEBUG 71739 --- [o-auto-1-exec-1] c.p.jeeptest.controller.FetchJeepTest : model=WRANGLER, trim=Sport
2022-05-28 18:18:18.000 INFO 71739 --- [o-auto-1-exec-1] c.p.jeeptest.service.DefaultJeeptestService : The fetchJeeps method was called
```

```
1 package com.promineotech.jeep;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication(scanBasePackageClasses = { ComponentScanMarker.class })
7
8 public class JeepSales {
9
10     public static void main(String[] args) {
11         SpringApplication.run(JeepSales.class, args);
12     }
13 }
14
15
16
17
18
19
20
```

```
19:03:06.495 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.spr
Spring
:: Spring Boot ::
(v2.7.0)
2022-05-28 19:03:06.722 INFO 73823 --- [ restartedMain ] com.promineotech.jeep.JeepSales : Starting JeepSales using Java 17.0.2 o
2022-05-28 19:03:06.723 DEBUG 73823 --- [ restartedMain ] com.promineotech.jeep.JeepSales : Running with Spring Boot v2.7.4.0, Sprin
2022-05-28 19:03:06.723 INFO 73823 --- [ restartedMain ] com.promineotech.jeep.JeepSales : No active profile set, falling back to
2022-05-28 19:03:07.925 INFO 73823 --- [ restartedMain ] com.promineotech.jeep.JeepSales : Started JeepSales in 1.421 seconds (JV
```

```
1 spring:
2   datasource:
3
4   url: jdbc:mysql://localhost:3306/jeep
5
6 logging:
7   level:
8     root: warn
9   '[com.promineotech]': debug
```

Screenshots of Running Application:

URL to GitHub Repository:

<https://github.com/Framos22/Spring-Boot-Assignment-2>