ASSIGNMENT

B. LIKHITHA

Spring Boot REST API – Basic Level (15 Problem Statements)

With only Web Starter

1. Welcome Message API

Problem Statement:

Create a REST API that returns a welcome message from the service layer.

Endpoint:

GET /welcome

Logic:

Service returns "Welcome to Spring Boot REST API"

2. Simple Calculator – Addition

Problem Statement:

Create an API to add two numbers.

Endpoint:

GET /add?a=10&b=20

Logic:

Service performs addition and returns result.

3. Simple Calculator – Even or Odd

Problem Statement:

Check whether a given number is **even or odd**.

Endpoint:

GET /evenodd/{num}

Logic:

Service checks num % 2.

4. Age Category API

Problem Statement:

Determine age category: Child, Adult, or Senior Citizen.**Endpoint:**

GET /age/{age}

Logic:

• <18 → Child

• 18–59 → Adult

• ≥60 → Senior

5. Grade Calculation API

Problem Statement:

Calculate grade based on marks.

Endpoint:

GET /grade/{marks}

Logic:

• ≥90 → A

• ≥75 → B

• ≥60 → C

• else → Fail

6. Temperature Conversion API

Problem Statement:

Convert Celsius to Fahrenheit.

Endpoint:

GET /celsius-to-fahrenheit/{celsius}

Logic:

(celsius * 9/5) + 32

7. String Reverse API

Problem Statement:

Reverse a given string.**Endpoint:**

GET /reverse/{text}

Logic:

Use StringBuilder.reverse().

8. Palindrome Check API

Problem Statement:

Check whether a string is a palindrome.

Endpoint:

GET /palindrome/{word}

Logic:

Compare original and reversed string.

9. Factorial Calculation API

Problem Statement:

Calculate factorial of a number.

Endpoint:

GET /factorial/{num}

Logic:

Loop-based factorial calculation.

10. Prime Number Check API

Problem Statement:

Check whether a number is prime.

Endpoint:

GET /prime/{num}

Logic:

Check divisibility up to √n.

11. Discount Calculation API

Problem Statement:

Calculate final price after discount.**Endpoint:**

GET /discount?price=1000&discount=10

Logic:

price - (price * discount / 100)

12. Login Validation API

Problem Statement:

Validate username and password.

Endpoint:

POST /login

Logic:

• Username = admin

• Password = admin123

• Return success or failure

13. Simple Interest Calculator API

Problem Statement:

Calculate simple interest.

Endpoint:

GET /interest?p=1000&r=10&t=2

Logic:

(p * r * t) / 100

14. Word Count API

Problem Statement:

Count number of words in a sentence.

Endpoint:

POST /wordcount

Logic:

Split string by spaces.**15. Maximum of Three Numbers API**

Problem Statement:

Find maximum of three numbers.

Endpoint:

GET /max?a=10&b=20&c=15

Logic:

Use conditional comparisons.

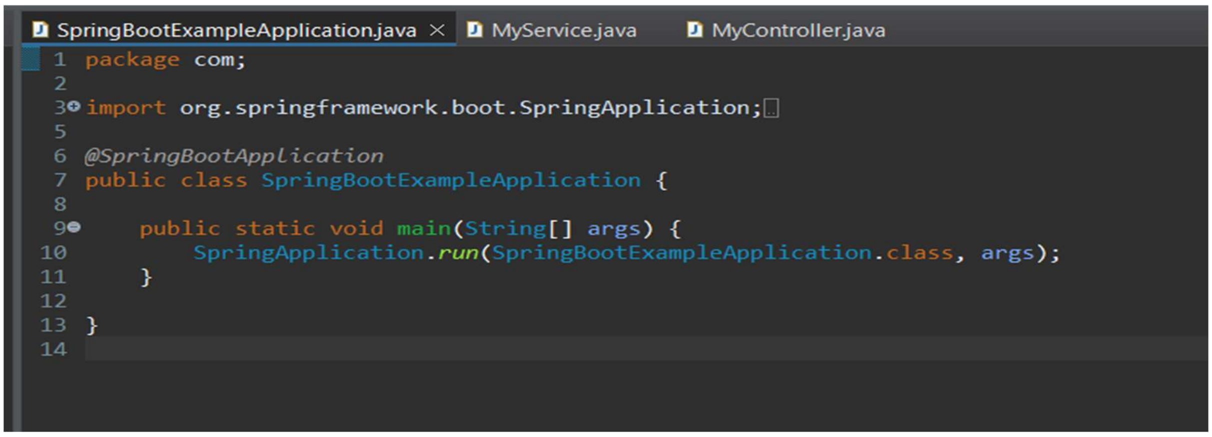
Architecture (Same for All)

controller

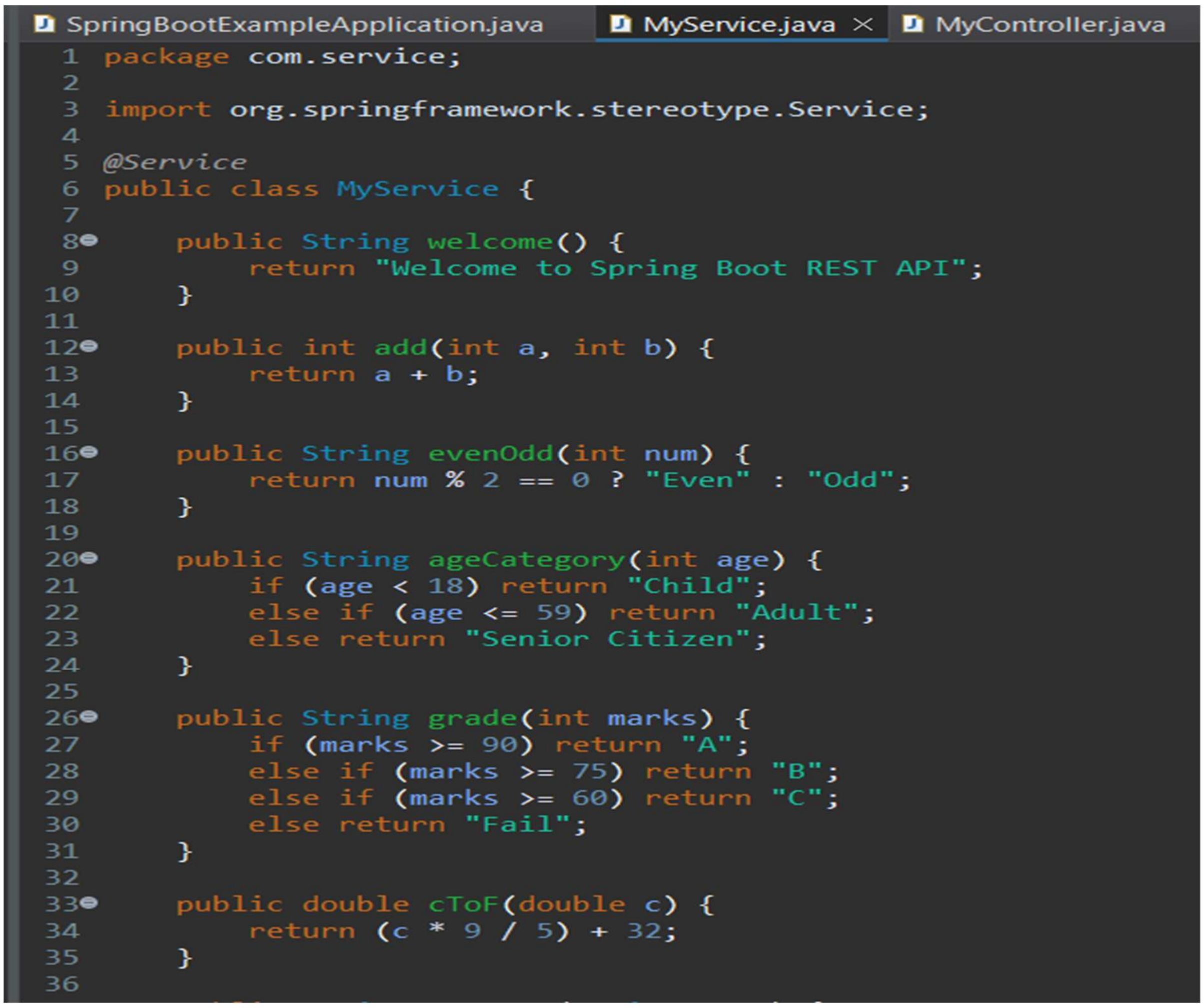
└── RestController

service

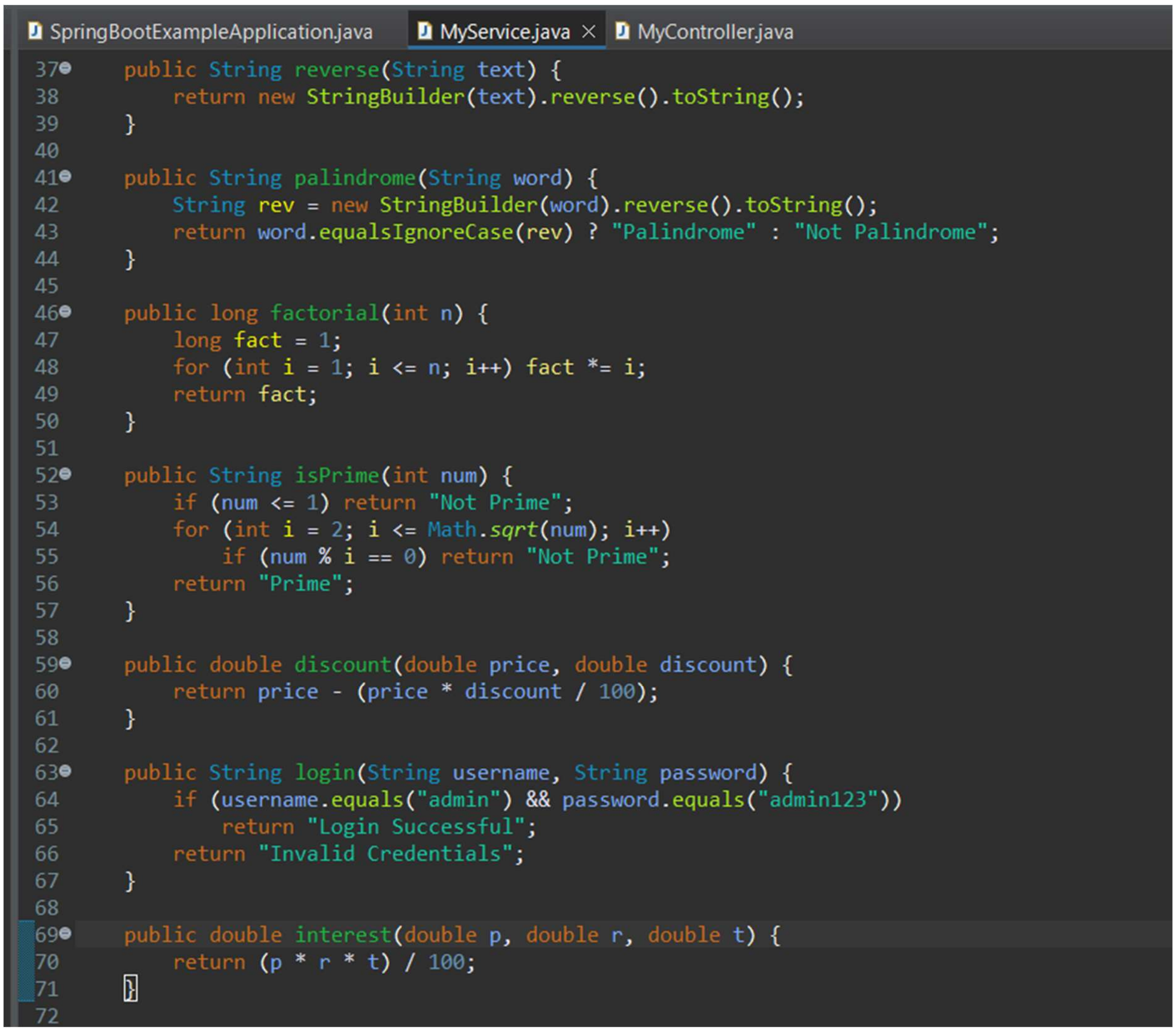
└── Business logic


SpringBootExampleApplication.java   MyService.java   MyController.java

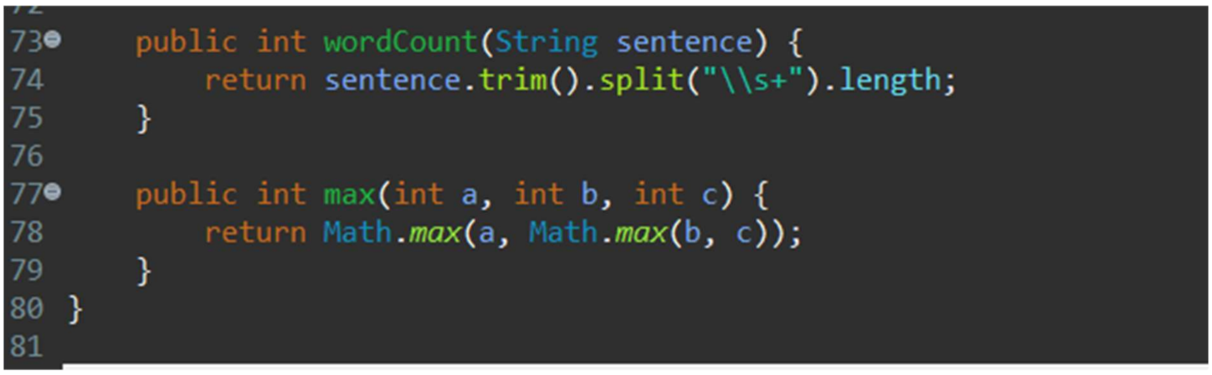
```java
package com;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class SpringBootExampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootExampleApplication.class, args);
    }

}
```


SpringBootExampleApplication.java   MyService.java   MyController.java

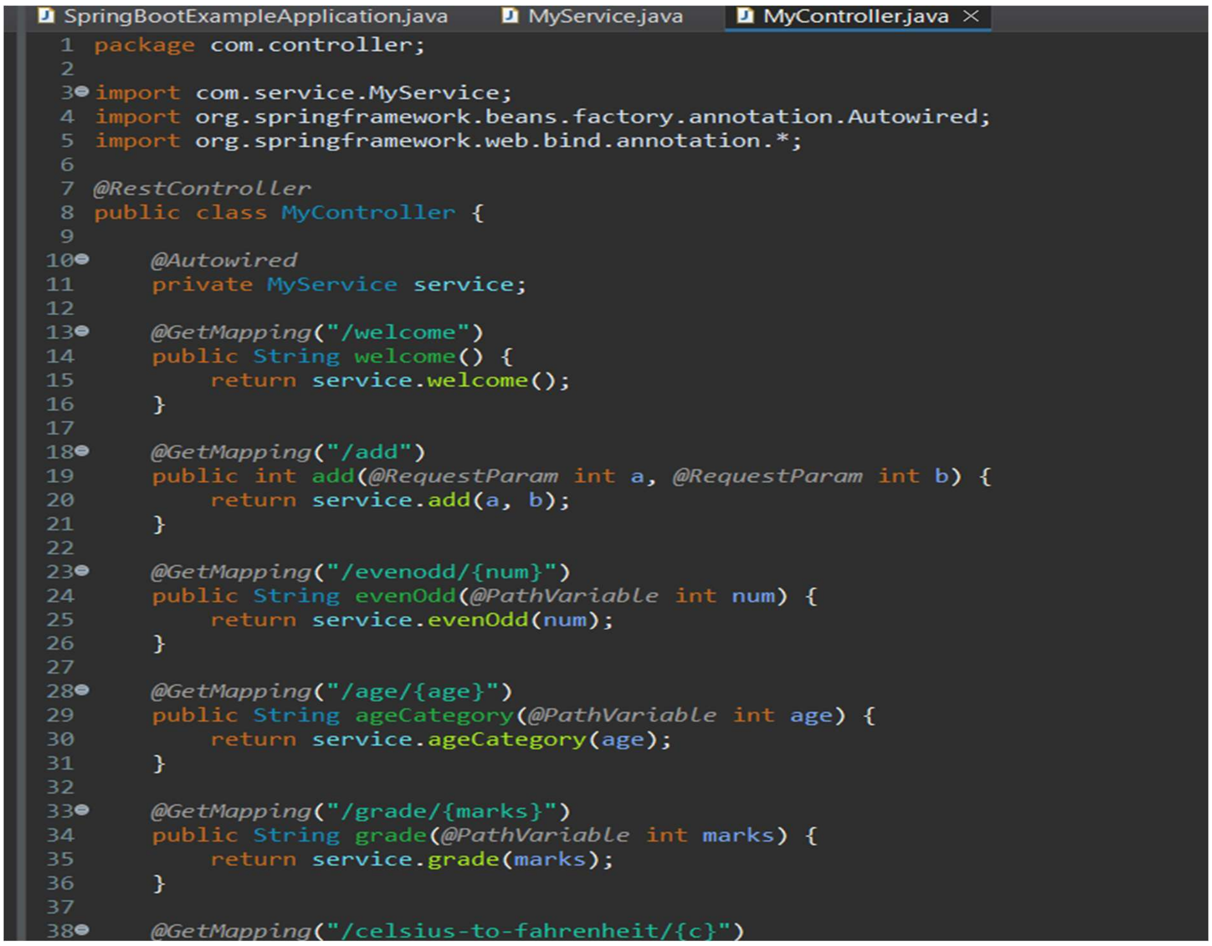
```java
package com.service;

import org.springframework.stereotype.Service;

@Service
public class MyService {

    public String welcome() {
        return "Welcome to Spring Boot REST API";
    }

    public int add(int a, int b) {
        return a + b;
    }

    public String evenOdd(int num) {
        return num % 2 == 0 ? "Even" : "Odd";
    }

    public String ageCategory(int age) {
        if (age < 18) return "Child";
        else if (age <= 59) return "Adult";
        else return "Senior Citizen";
    }

    public String grade(int marks) {
        if (marks >= 90) return "A";
        else if (marks >= 75) return "B";
        else if (marks >= 60) return "C";
        else return "Fail";
    }

    public double cToF(double c) {
        return (c * 9 / 5) + 32;
    }

```



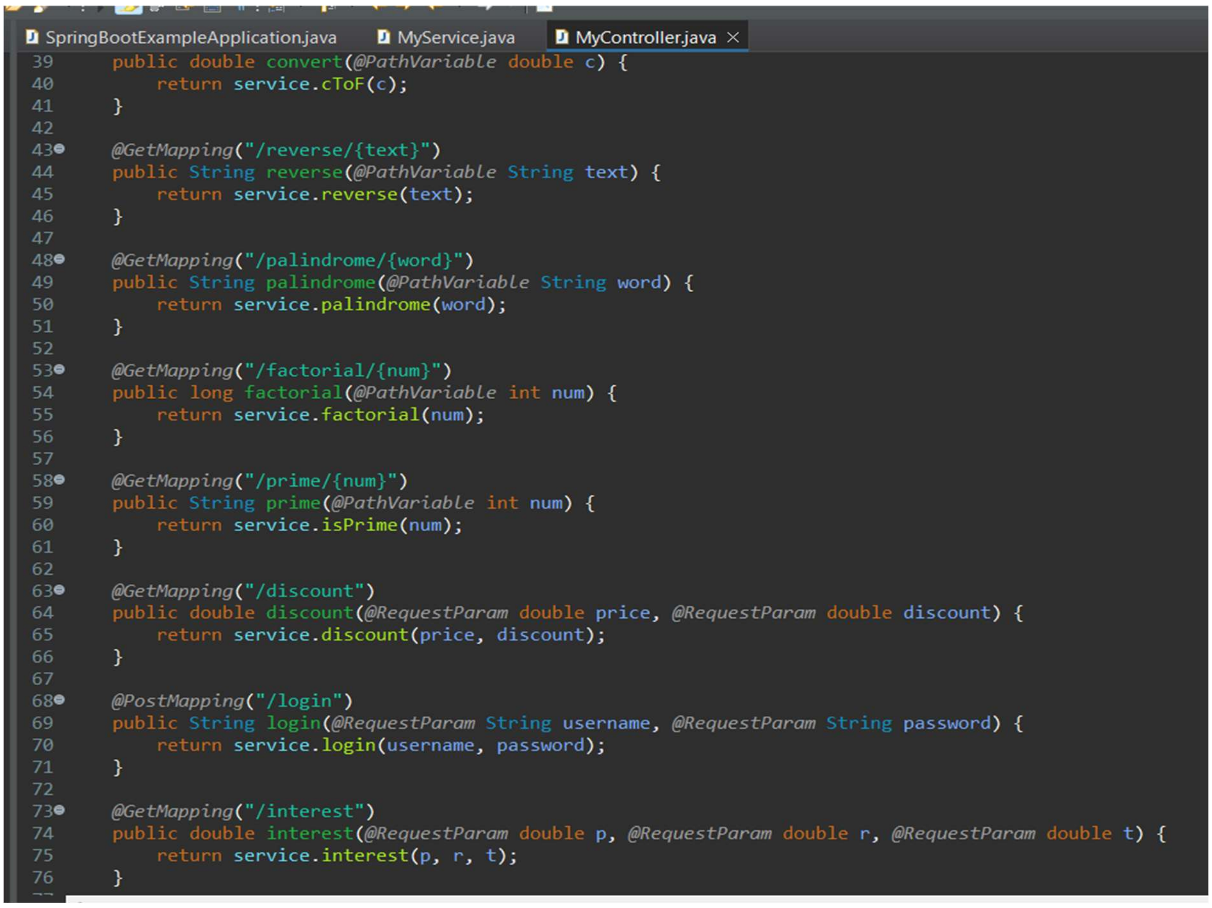
```java
    public String reverse(String text) {
        return new StringBuilder(text).reverse().toString();
    }

    public String palindrome(String word) {
        String rev = new StringBuilder(word).reverse().toString();
        return word.equalsIgnoreCase(rev) ? "Palindrome" : "Not Palindrome";
    }

    public long factorial(int n) {
        long fact = 1;
        for (int i = 1; i <= n; i++) fact *= i;
        return fact;
    }

    public String isPrime(int num) {
        if (num <= 1) return "Not Prime";
        for (int i = 2; i <= Math.sqrt(num); i++)
            if (num % i == 0) return "Not Prime";
        return "Prime";
    }

    public double discount(double price, double discount) {
        return price - (price * discount / 100);
    }

    public String login(String username, String password) {
        if (username.equals("admin") && password.equals("admin123"))
            return "Login Successful";
        return "Invalid Credentials";
    }

    public double interest(double p, double r, double t) {
        return (p * r * t) / 100;
    }

```



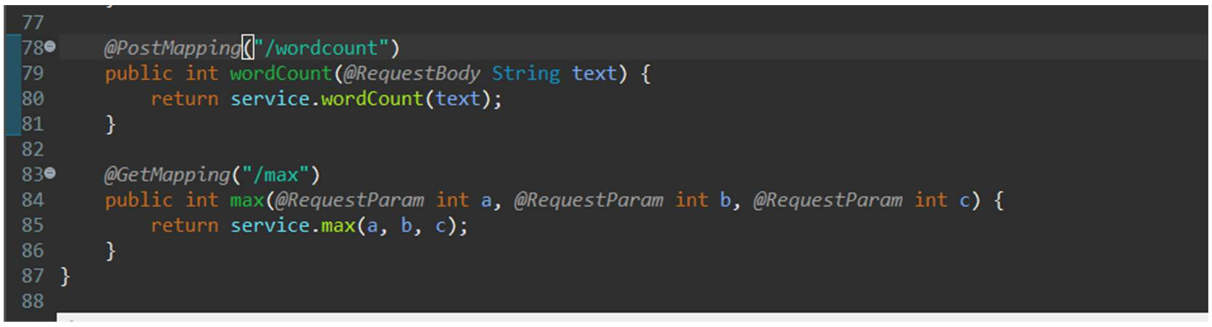
```java
    public int wordCount(String sentence) {
        return sentence.trim().split("\\s+").length;
    }

    public int max(int a, int b, int c) {
        return Math.max(a, Math.max(b, c));
    }
}

```



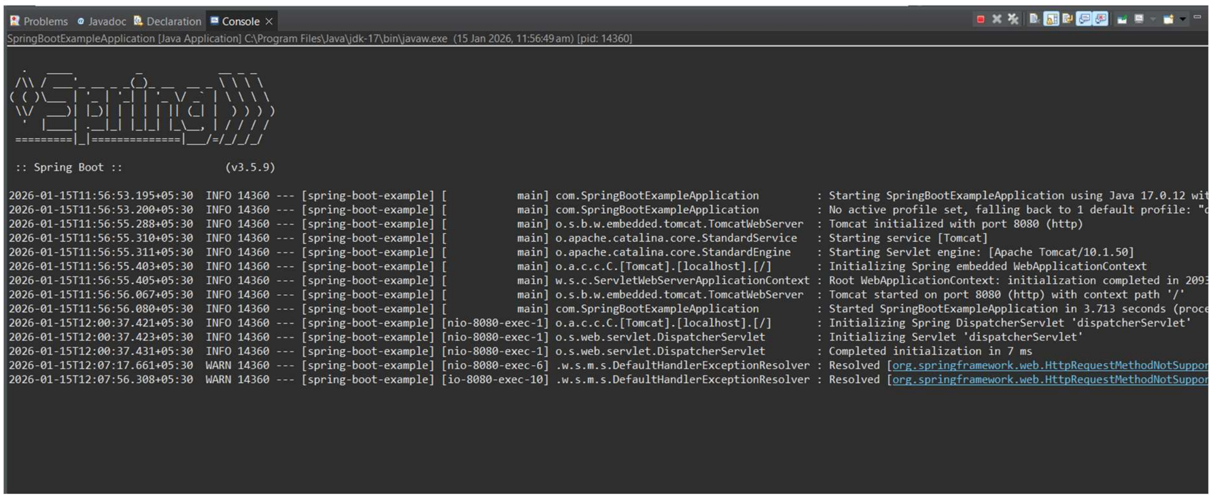
```java
package com.controller;

import com.service.MyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
public class MyController {

    @Autowired
    private MyService service;

    @GetMapping("/welcome")
    public String welcome() {
        return service.welcome();
    }

    @GetMapping("/add")
    public int add(@RequestParam int a, @RequestParam int b) {
        return service.add(a, b);
    }

    @GetMapping("/evenodd/{num}")
    public String evenOdd(@PathVariable int num) {
        return service.evenOdd(num);
    }

    @GetMapping("/age/{age}")
    public String ageCategory(@PathVariable int age) {
        return service.ageCategory(age);
    }

    @GetMapping("/grade/{marks}")
    public String grade(@PathVariable int marks) {
        return service.grade(marks);
    }

    @GetMapping("/celsius-to-fahrenheit/{c}")
```



```java
    public double convert(@PathVariable double c) {
        return service.cToF(c);
    }

    @GetMapping("/reverse/{text}")
    public String reverse(@PathVariable String text) {
        return service.reverse(text);
    }

    @GetMapping("/palindrome/{word}")
    public String palindrome(@PathVariable String word) {
        return service.palindrome(word);
    }

    @GetMapping("/factorial/{num}")
    public long factorial(@PathVariable int num) {
        return service.factorial(num);
    }

    @GetMapping("/prime/{num}")
    public String prime(@PathVariable int num) {
        return service.isPrime(num);
    }

    @GetMapping("/discount")
    public double discount(@RequestParam double price, @RequestParam double discount) {
        return service.discount(price, discount);
    }

    @PostMapping("/login")
    public String login(@RequestParam String username, @RequestParam String password) {
        return service.login(username, password);
    }

    @GetMapping("/interest")
    public double interest(@RequestParam double p, @RequestParam double r, @RequestParam double t) {
        return service.interest(p, r, t);
    }
```

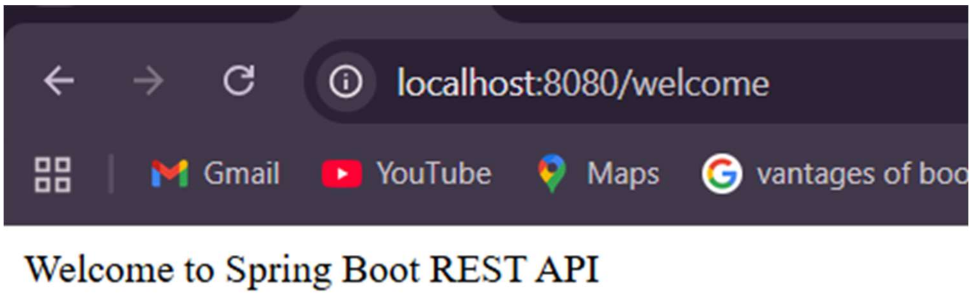


```java

    @PostMapping("/wordcount")
    public int wordCount(@RequestBody String text) {
        return service.wordCount(text);
    }

    @GetMapping("/max")
    public int max(@RequestParam int a, @RequestParam int b, @RequestParam int c) {
        return service.max(a, b, c);
    }
}

```



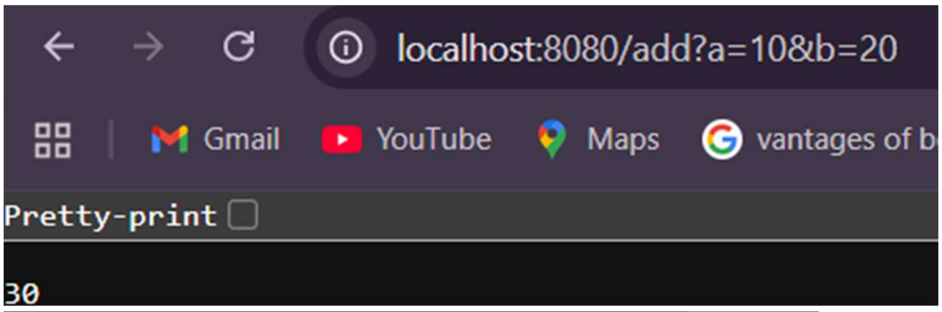
SpringBootExampleApplication [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (15 Jan 2026, 11:56:49 am) [pid: 14360]

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.9)

2026-01-15T11:56:53.195+05:30  INFO 14360 --- [spring-boot-example] [           main] com.SpringBootExampleApplication         : Starting SpringBootExampleApplication using Java 17.0.12 wit
2026-01-15T11:56:53.200+05:30  INFO 14360 --- [spring-boot-example] [           main] com.SpringBootExampleApplication         : No active profile set, falling back to 1 default profile: "d
2026-01-15T11:56:55.288+05:30  INFO 14360 --- [spring-boot-example] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2026-01-15T11:56:55.310+05:30  INFO 14360 --- [spring-boot-example] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2026-01-15T11:56:55.311+05:30  INFO 14360 --- [spring-boot-example] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.50]
2026-01-15T11:56:55.403+05:30  INFO 14360 --- [spring-boot-example] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2026-01-15T11:56:55.405+05:30  INFO 14360 --- [spring-boot-example] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2093
2026-01-15T11:56:56.067+05:30  INFO 14360 --- [spring-boot-example] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2026-01-15T11:56:56.080+05:30  INFO 14360 --- [spring-boot-example] [           main] com.SpringBootExampleApplication         : Started SpringBootExampleApplication in 3.713 seconds (proce
2026-01-15T12:00:37.421+05:30  INFO 14360 --- [spring-boot-example] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2026-01-15T12:00:37.423+05:30  INFO 14360 --- [spring-boot-example] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2026-01-15T12:00:37.431+05:30  INFO 14360 --- [spring-boot-example] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 7 ms
2026-01-15T12:07:17.661+05:30  WARN 14360 --- [spring-boot-example] [nio-8080-exec-6] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.HttpRequestMethodNotSuppor
2026-01-15T12:07:56.308+05:30  WARN 14360 --- [spring-boot-example] [io-8080-exec-10] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.HttpRequestMethodNotSuppor
```



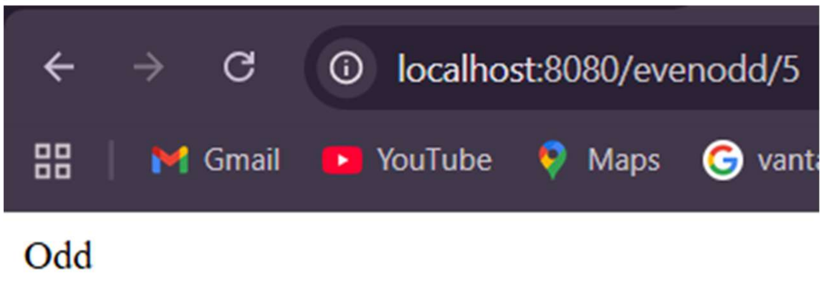
localhost:8080/welcome

Welcome to Spring Boot REST API



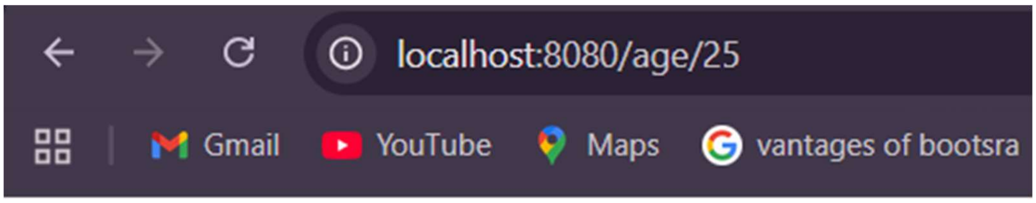
localhost:8080/add?a=10&b=20

```
30
```

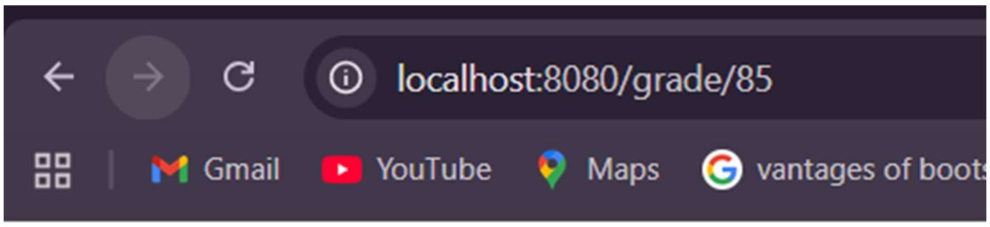


localhost:8080/evenodd/5

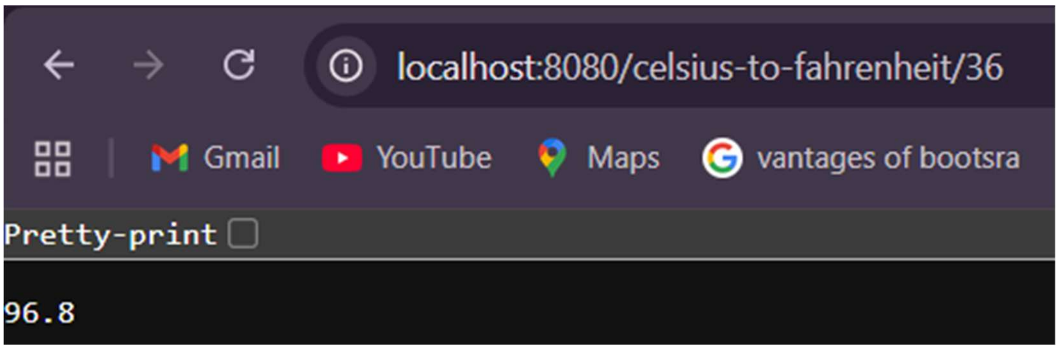
Odd


localhost:8080/age/25

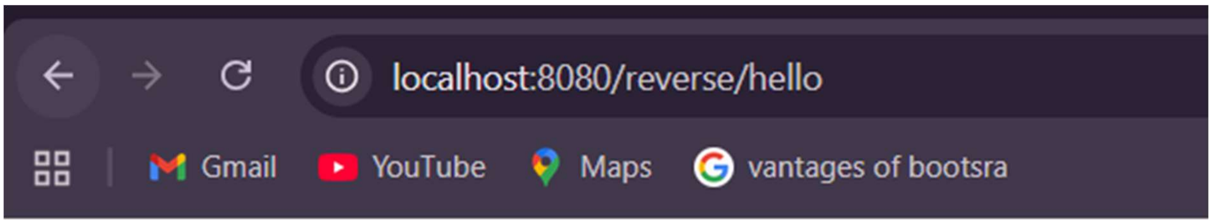
Adult


localhost:8080/grade/85

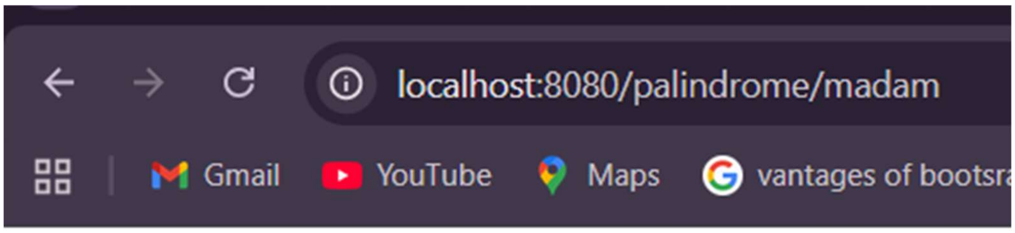
B


localhost:8080/celsius-to-fahrenheit/36

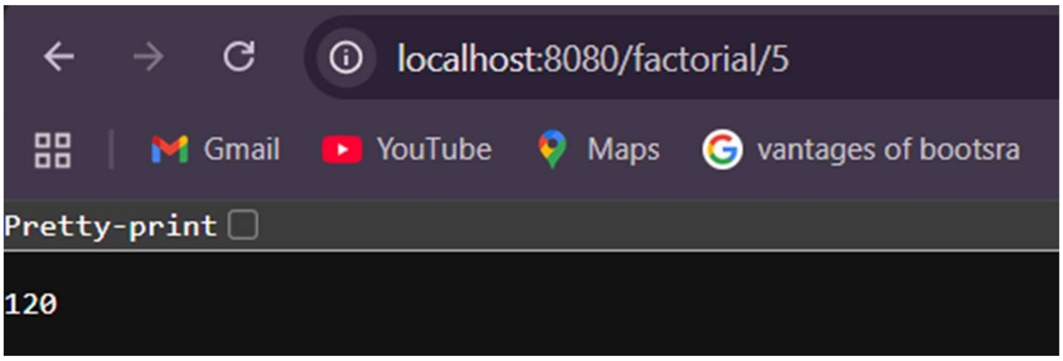
```
96.8
```


localhost:8080/reverse/hello

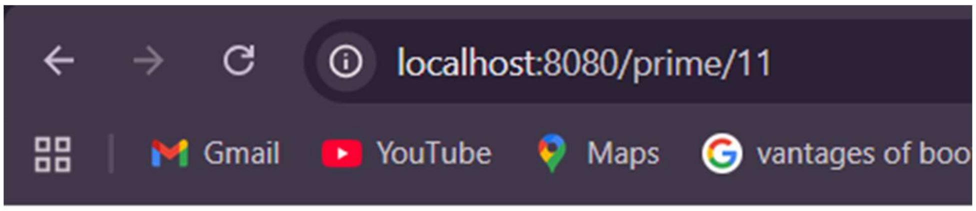
olleh


localhost:8080/palindrome/madam

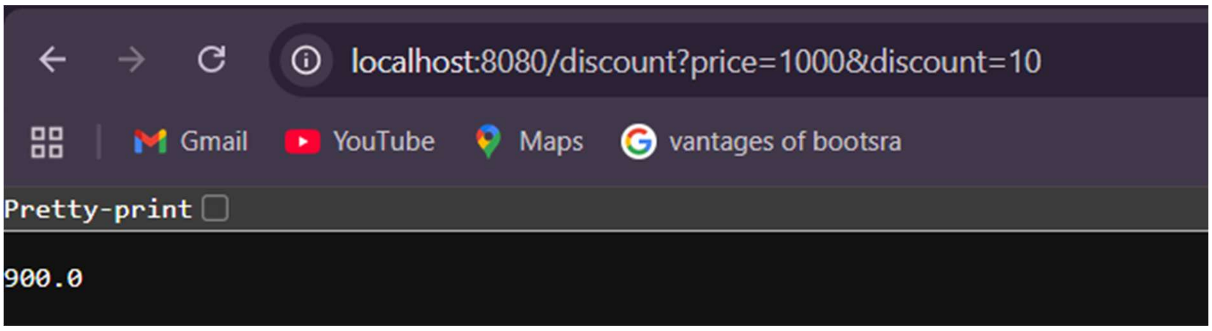
Palindrome


localhost:8080/factorial/5

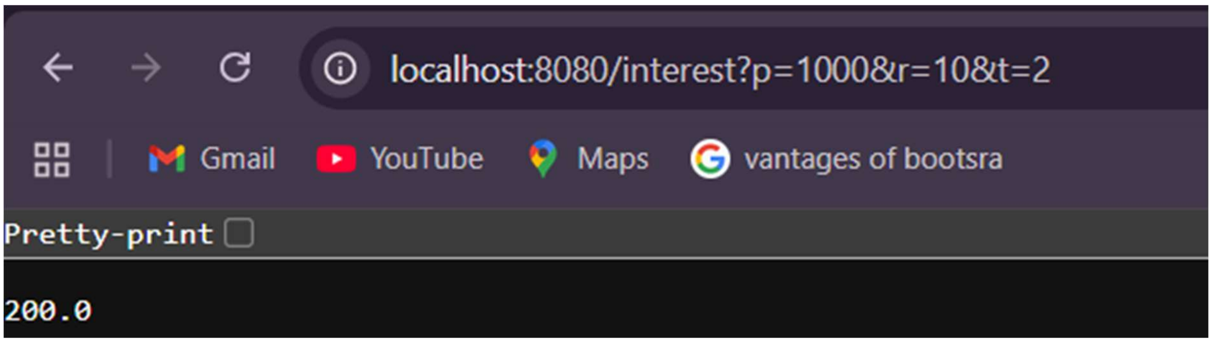
```
120
```


localhost:8080/prime/11

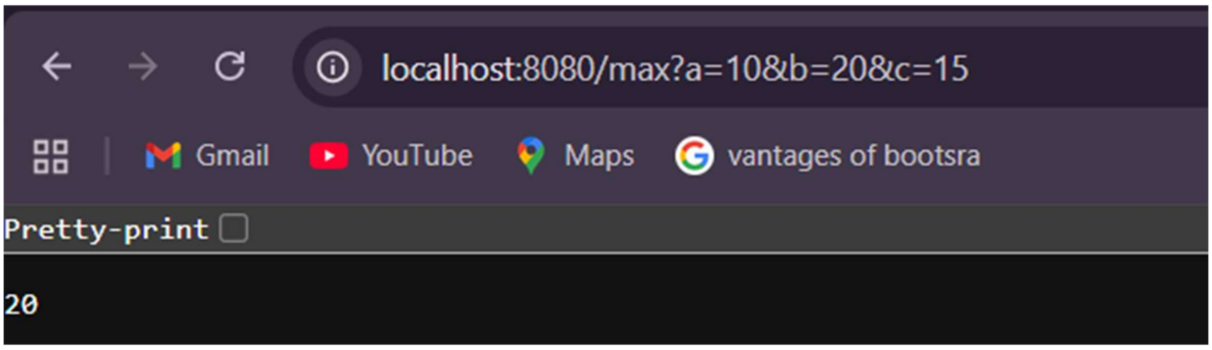
Prime


localhost:8080/discount?price=1000&discount=10

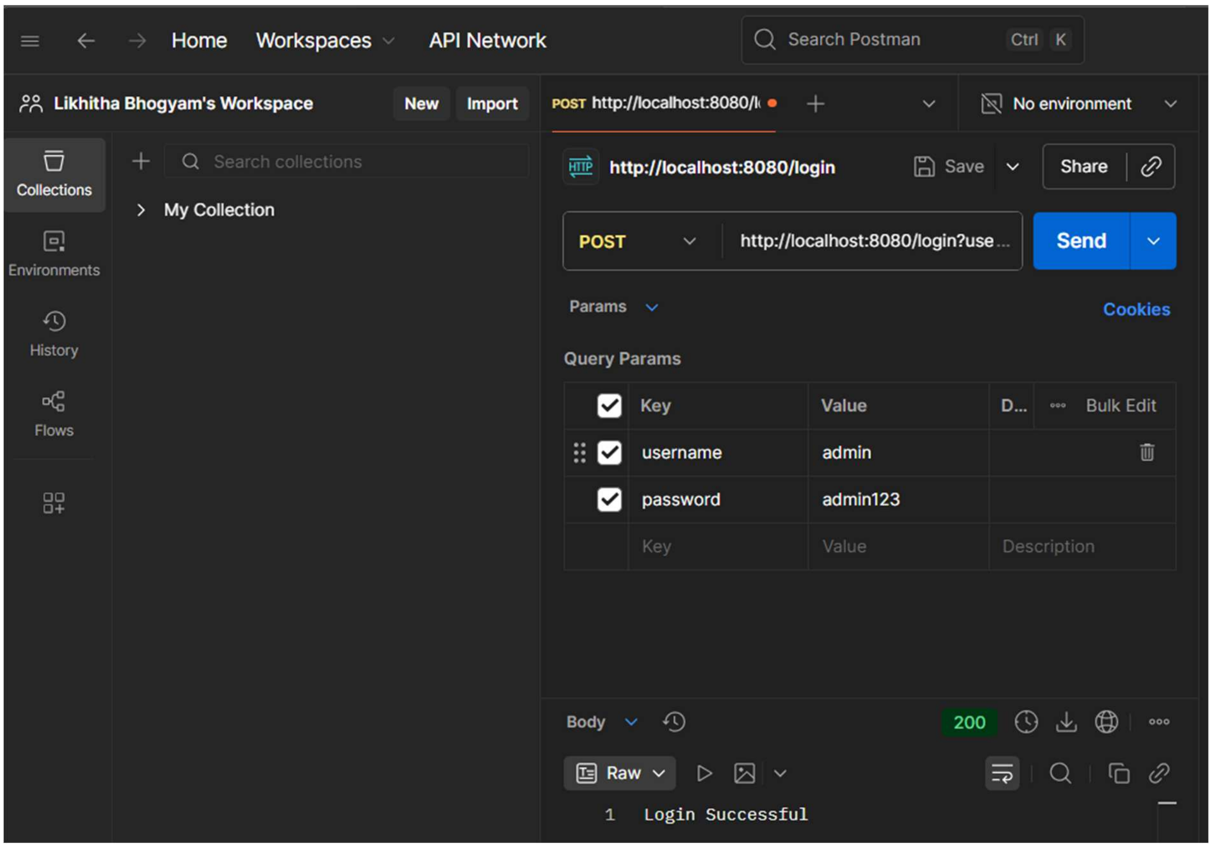
```
900.0
```


localhost:8080/interest?p=1000&r=10&t=2

```
200.0
```


localhost:8080/max?a=10&b=20&c=15

```
20
```



http://localhost:8080/login

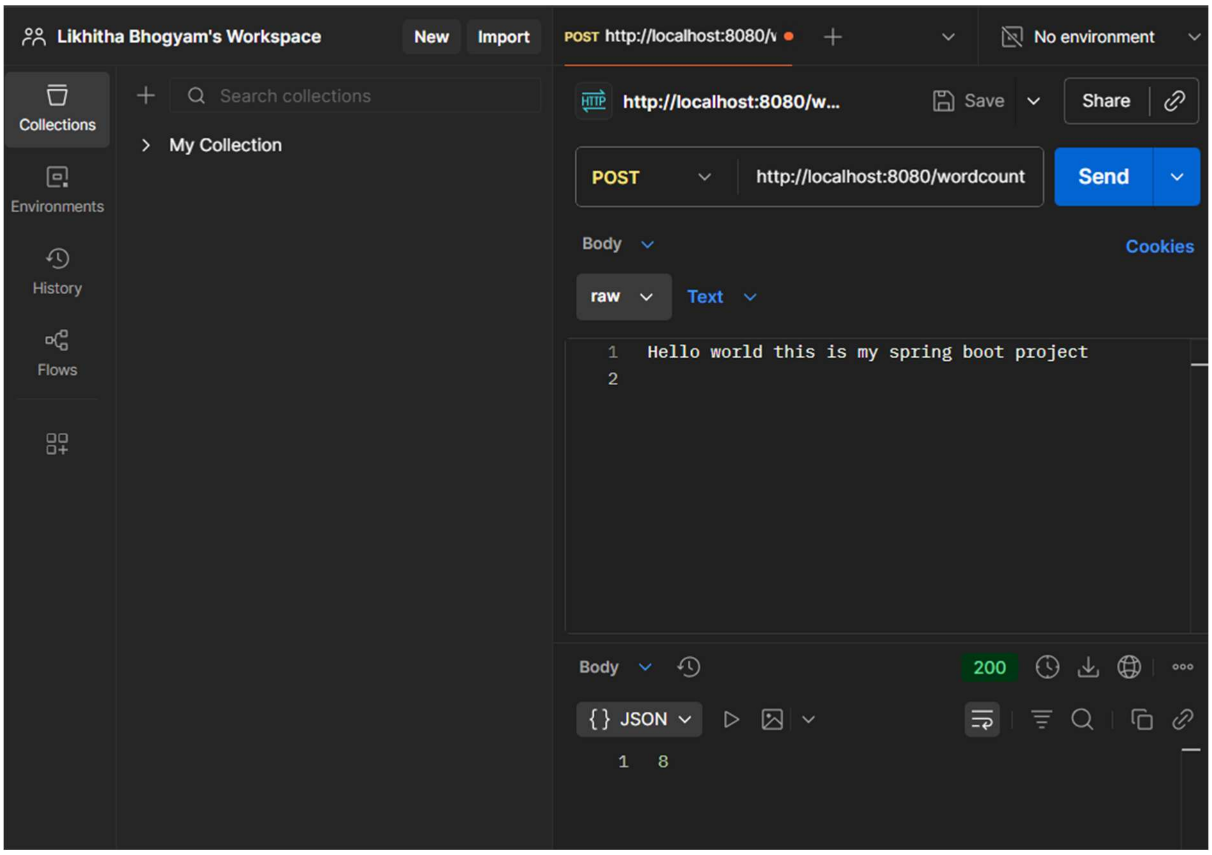
POST http://localhost:8080/login?use...

Query Params

Key	Value	Description
✓ username	admin	
✓ password	admin123	

Body 200 Raw

```
Login Successful
```



http://localhost:8080/w...

POST http://localhost:8080/wordcount

Body raw Text

```
Hello world this is my spring boot project
```

Body 200 JSON

```
8
```