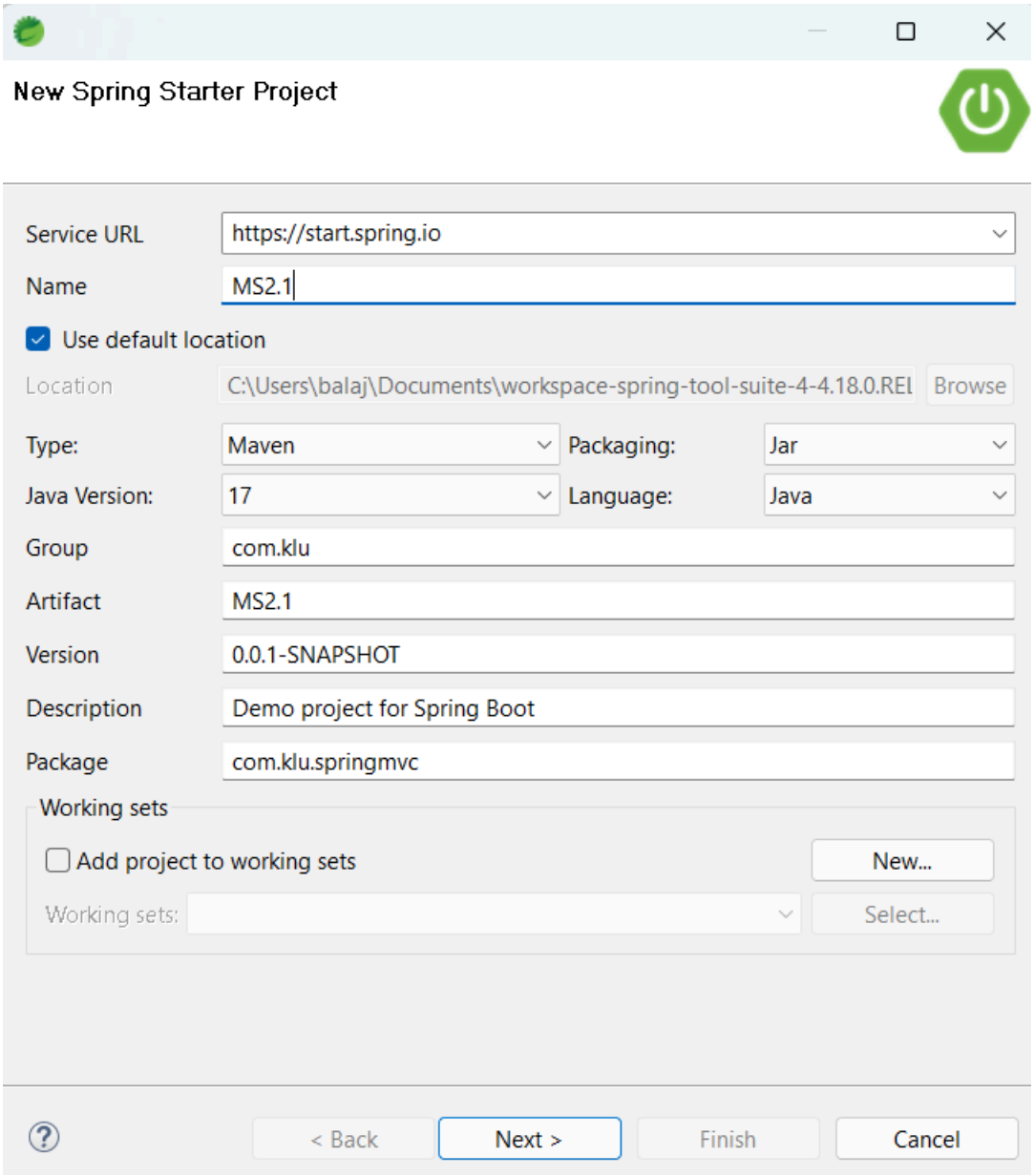
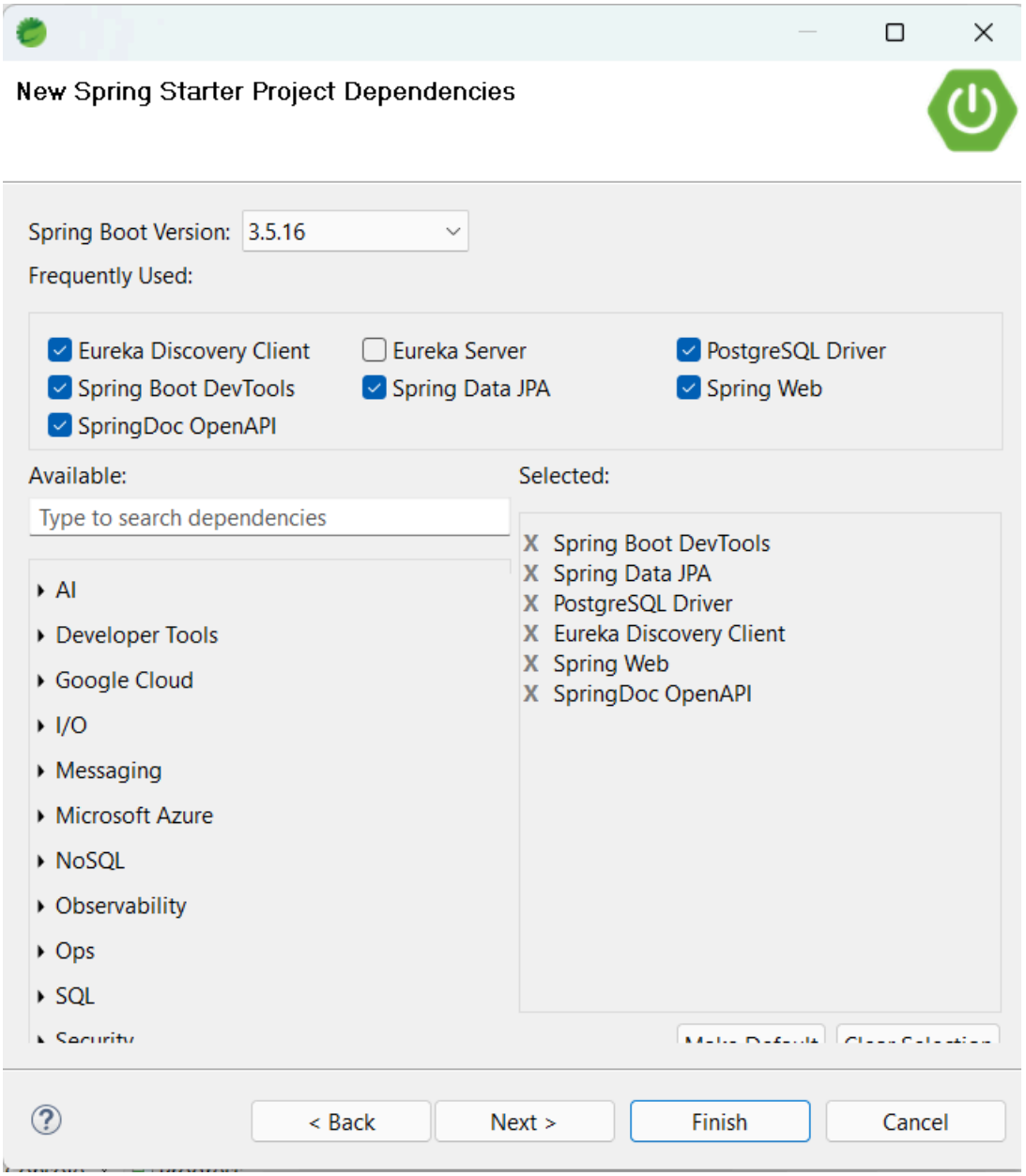
Part 1: creating microservices (Eureka server, MS2.1 (addition), MS2.2 (multiplication), MS2.3 (multiplication), MS-API-GATEWAY (with load balancer) in sts4 (with db properties ready) and checking it with addition and multiplication

Create MS1.1 and MS2.1 with the following,





Open application.properties of MS1.1 and have below code,

```
spring.application.name=MS1
server.port = 8001

spring.datasource.url = jdbc:postgresql://localhost:5432/mydb
spring.datasource.username = postgres
spring.datasource.password = root
spring.datasource.driver-class-name = org.postgresql.Driver

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto = update
```

```
spring.jpa.show-sql = true

# Eureka Server URL
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

# Register this service with Eureka
eureka.client.register-with-eureka=true

# Fetch registry from Eureka
eureka.client.fetch-registry=true

# Display IP address in Eureka Dashboard (Optional)
eureka.instance.prefer-ip-address=true
```

Similarly, open application.properties of MS2.1 and have below code,

```
spring.application.name=MS2

server.port = 8002

spring.datasource.url = jdbc:postgresql://localhost:5432/mydb
spring.datasource.username = postgres
spring.datasource.password = root
spring.datasource.driver-class-name = org.postgresql.Driver

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql = true

# Eureka Server URL
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

# Register this service with Eureka
eureka.client.register-with-eureka=true

# Fetch registry from Eureka
eureka.client.fetch-registry=true

# Display IP address in Eureka Dashboard (Optional)
eureka.instance.prefer-ip-address=true
```

Go to Apllication.java and add

@EnableDiscoveryClient

In both the files,

To get import statements automatically for this line, place cursor on this word ("@EnableEurekaServer") and press "Cntl + Shft + o"

```
package com.klu.springmvc;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class Application {

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

}
```

Create MS1Controller class in MS1.1 and have below,

```
package com.klu.springmvc;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/ms1")
public class MS1Controller {

   @GetMapping("/add")
   public String add(@RequestParam("a") int a, @RequestParam("b") int b)
   {
      int result = a + b;
      return "MS 1.1 - Additon = " + result;
   }

}
```

Create MS2Controller class in MS2.1 and have below code,

```
package com.klu.springmvc;

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

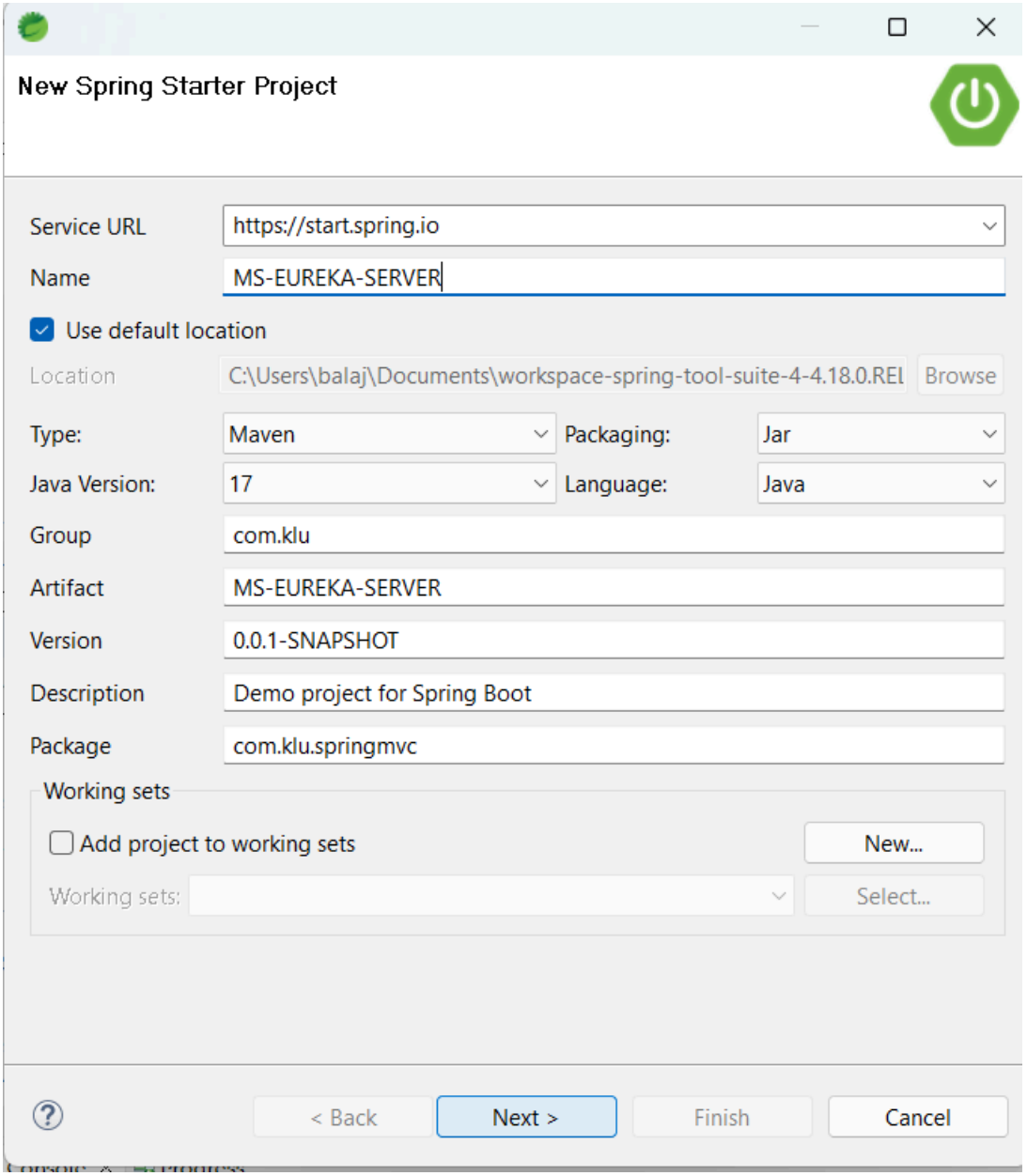
```
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/ms2")
public class MS2Controller {
    @GetMapping("/mul")
    public String add(@RequestParam("a") int a, @RequestParam("b") int b)
    {
        int result = a * b;
        return "MS 2.1 - Multiplication = " + result;
    }
}
```

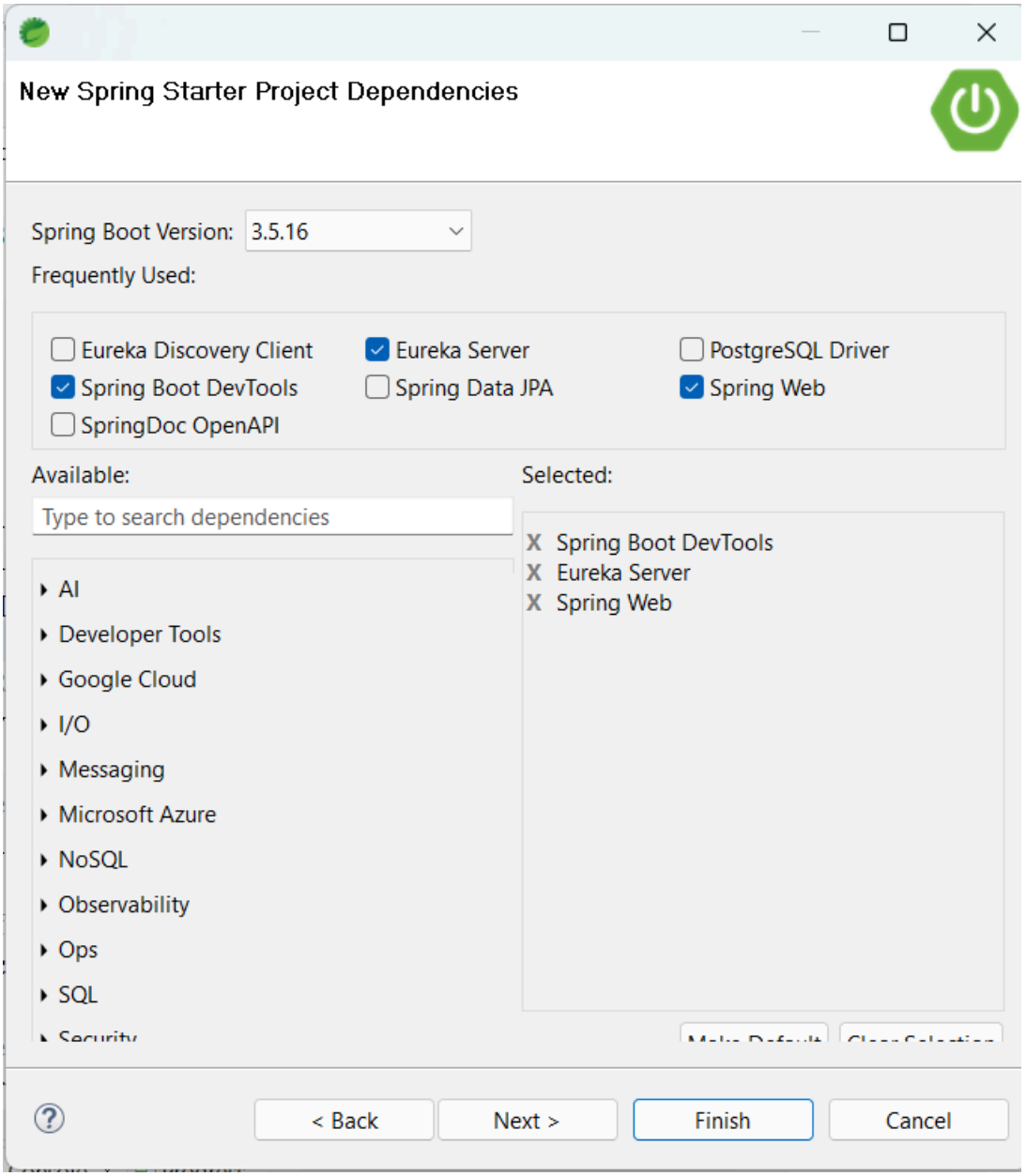
Copy MS2.1 and past as MS2.2 and modify the application.properties with only the port number as "8003".

Also modify the following line with MS2.2 in controller file:

```
return "MS 2.1 - Multiplication = " + result;
```

Create MS-EUREKA-SERVER


New Spring Starter Project
Service URL
https://start.spring.io
Name
MS-EUREKA-SERVER
Use default location
Location
C:\Users\balaj\Documents\workspace-spring-tool-suite-4-4.18.0.REl
Browse
Type:
Maven
Packaging:
Jar
Java Version:
17
Language:
Java
Group
com.klu
Artifact
MS-EUREKA-SERVER
Version
0.0.1-SNAPSHOT
Description
Demo project for Spring Boot
Package
com.klu.springmvc
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel



In MS-EUREKA-SERVER, open application.properties and have below code,

spring.application.name=MS-EUREKA-SERVER

server.port=8761

eureka.client.register-with-eureka=false

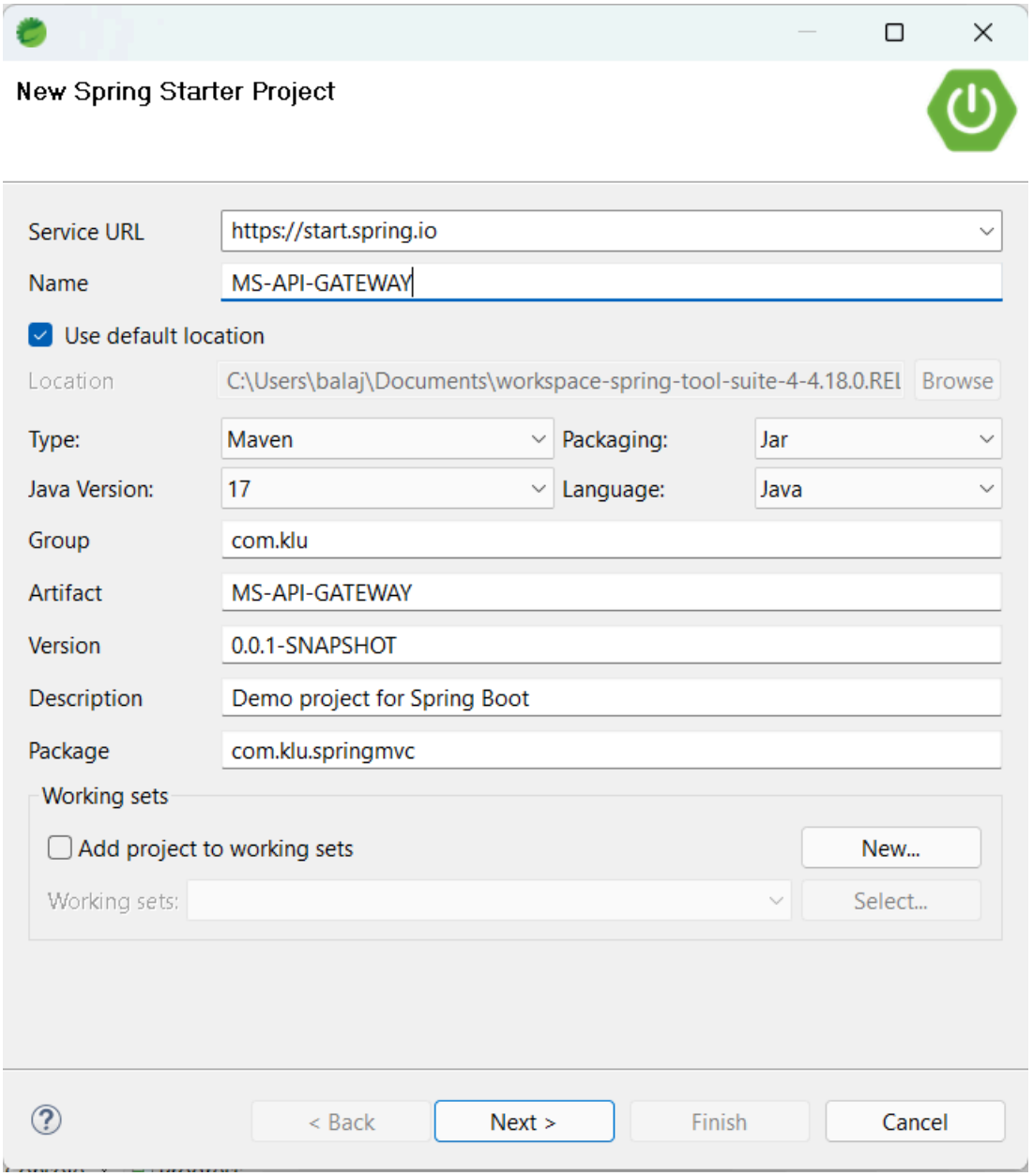
eureka.client.fetch-registry=false

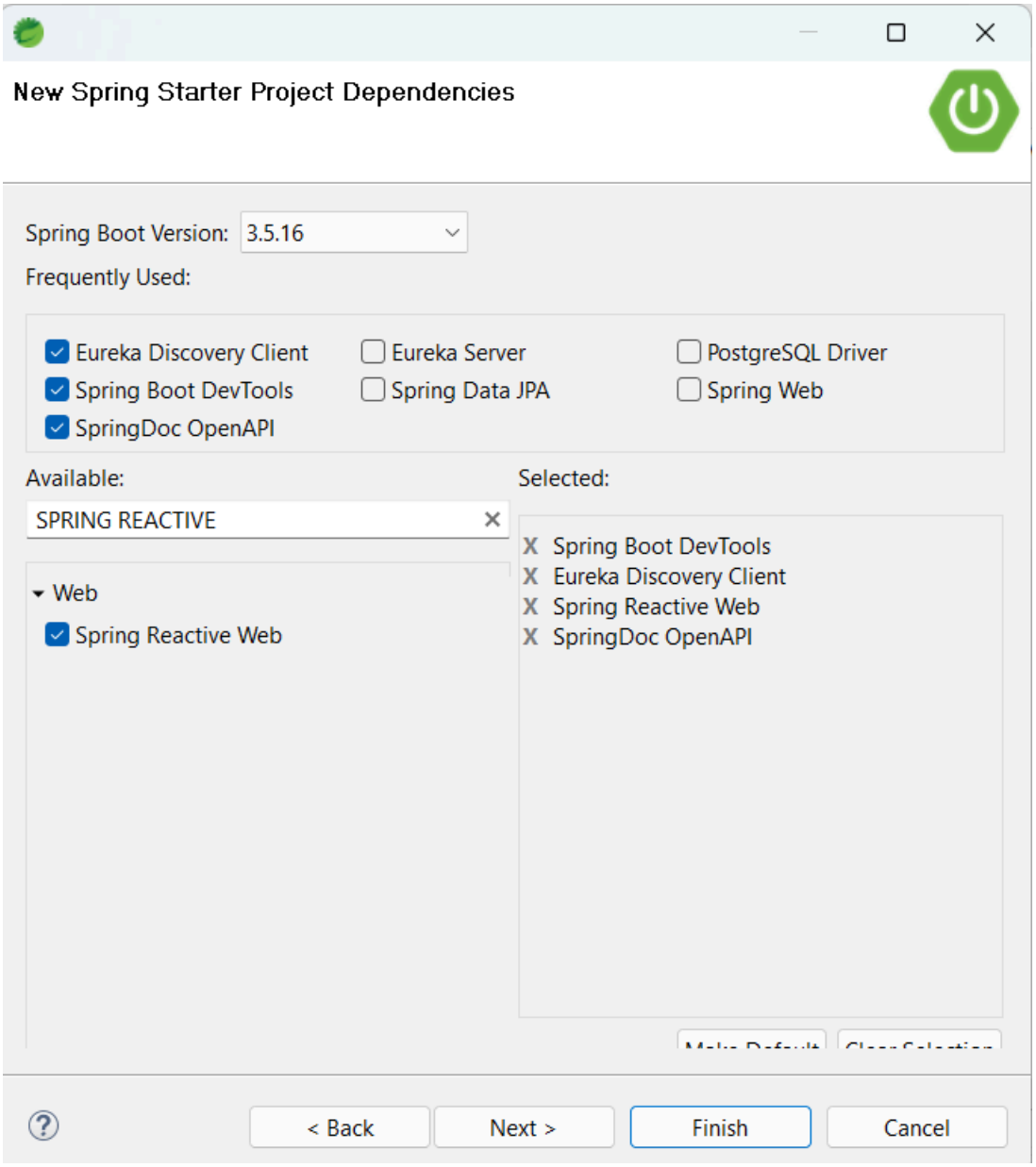
Open MsEurekaServerApplication.java and add "@EnableEurekaServer"

To get import statements automatically for this line, place cursor on this word ("@EnableEurekaServer") and press "Cntl + Shft + o"

```java
package com.klu.springmvc;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class MsEurekaServerApplication {

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

}
```

Create MS-API-GATEWAY


New Spring Starter Project
Service URL
https://start.spring.io
Name
MS-API-GATEWAY
Use default location
Location
C:\Users\balaj\Documents\workspace-spring-tool-suite-4-4.18.0.REl
Browse
Type:
Maven
Packaging:
Jar
Java Version:
17
Language:
Java
Group
com.klu
Artifact
MS-API-GATEWAY
Version
0.0.1-SNAPSHOT
Description
Demo project for Spring Boot
Package
com.klu.springmvc
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel



Open application.properties and add below code,

```
spring.application.name=MS-API-GATEWAY
server.port=8000

# Eureka Server
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

# Register Gateway with Eureka
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true

#####################################################
# Route 1 : Addition Service
```

```
##################################################

spring.cloud.gateway.server.webflux.routes[0].id=MS1

spring.cloud.gateway.server.webflux.routes[0].uri=lb://MS1

spring.cloud.gateway.server.webflux.routes[0].predicates[0]=Path=/ms1/**

##################################################
# Route 2 : Multiplication Service
##################################################

spring.cloud.gateway.server.webflux.routes[1].id=MS2

spring.cloud.gateway.server.webflux.routes[1].uri=lb://MS2

spring.cloud.gateway.server.webflux.routes[1].predicates[0]=Path=/ms2/**
```

---

in pom.xml, relplace below code

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
```

Replace with

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
```

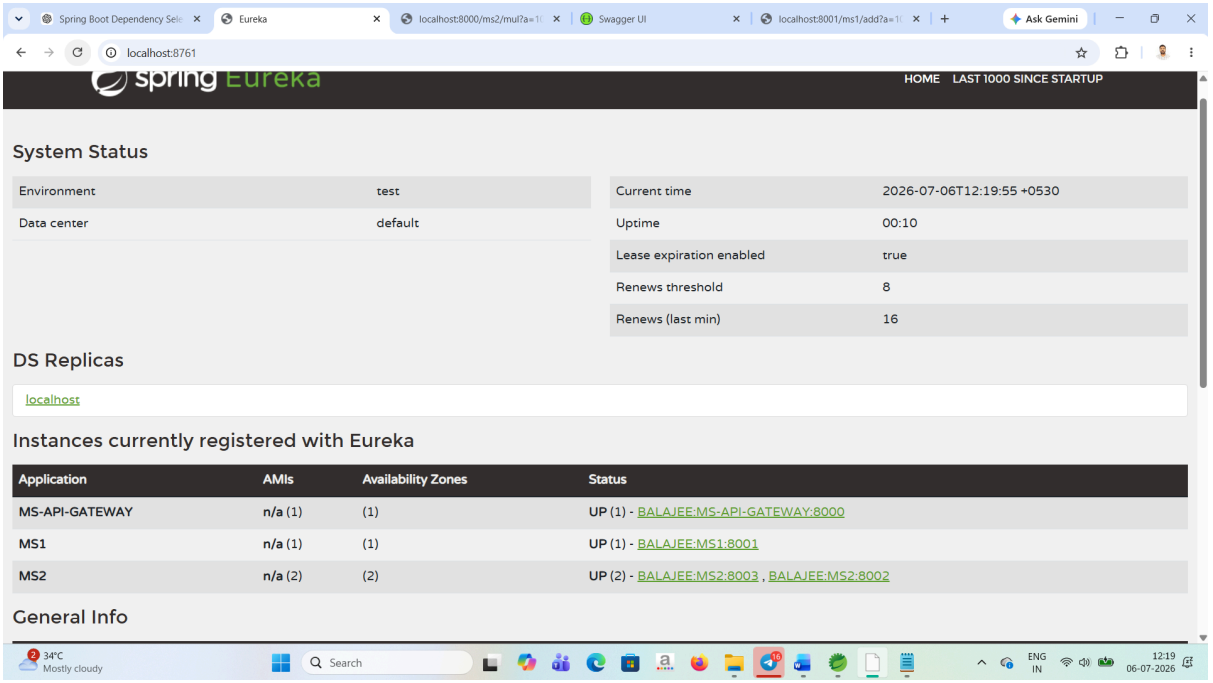
Run MS-EUREKA-SERVER

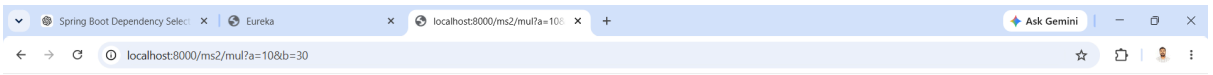
Run MS-1.1

Run MS-2.1

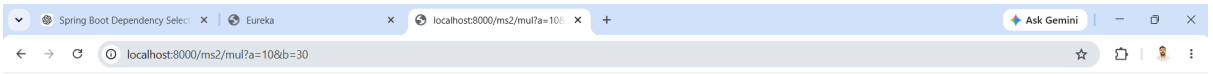
Run MS-2.2

Run MS-API-GATEWAY

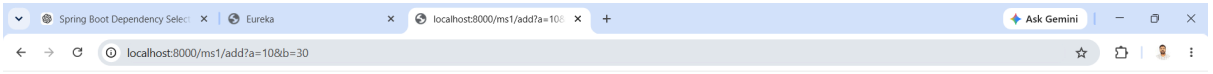




MS 2.1 - Multiplication = 300



MS 2.2 - Multiplication = 300



MS 1.1 - Additon = 40

PART 2 (JWT TOKENS, SIGNIN AND SIGNUP IN MS2.1 WITH DB OPERATIONS)

Theory:

What are we trying to achieve?

Suppose your application has users.

Before using the application, a user must

1. Register (Sign Up)
2. Login (Sign In)

3. After login, prove they are already logged in without entering the password every time.

Instead of asking for the password repeatedly, the server gives the user a **JWT Token**, which acts like a temporary ID card.

Now the user wants to: View profile , View orders , Update profile , Change password

Without JWT, the server would need to ask for the username and password every time, or maintain a server-side session.

JWT is not just about "not logging in again"

JWT also provides:

- **Authentication** – Confirms who the user is.
- **Authorization** – Can include roles (such as ADMIN or USER) so the server can decide what the user is allowed to do.
- **Statelessness** – The server doesn't need to store login sessions in memory because the token carries the necessary identity information.

Student

|

| Sign Up

▼

Database

|

| Sign In

▼

Server checks username & password

|

| Correct?

▼

Server generates JWT Token

|

▼

Client stores the token

Your JWT contains:

- Username
- User ID

- Role
- Expiration time

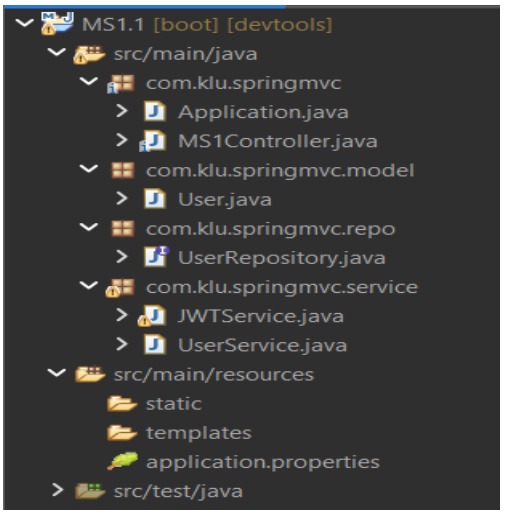
It is also **digitally signed** using your secret key.

.signWith(secretKey) adds a digital signature.

Think of it as the university's official seal on an ID card.

Practical:

In MS1.1, Create package, com.klu.springmvc.model and create class User.java with following code,



Note: com.klu.springmvc is the default package, any further package will be created with further suffix word like "model". You can create the package under src/main/java

```
package com.klu.springmvc.model;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
@Table(name = "user_tab")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    int id;
```

```
	@Column(unique = true)
	String username;
	String password;
	Integer role;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public Integer getRole() {
		return role;
	}
	public void setRole(Integer role) {
		this.role = role;
	}
	@Override
	public String toString() {
		return "User [id=" + id + ", username=" + username + ", password=" +
password + ", role=" + role + "]";
	}
}
```

Create package com.klu.springmvc.repo and create interface as "UserRepository" with below code,

```
package com.klu.springmvc.repo;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.klu.springmvc.model.User;

@Repository
```

```
public interface UserRepository extends JpaRepository<User, Integer> {

    User findByUsername(String username);

}
```

Create com.klu.springmvc.service package and have

1. **“UserService.java” class**
2. **“JWTService.java” class**

Pom.xml dependencies

```
<!-- Source: https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-api -->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-api</artifactId>
            <version>0.13.0</version>
            <scope>compile</scope>
        </dependency>
        <!-- Source: https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-impl -->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-impl</artifactId>
            <version>0.13.0</version>
            <scope>runtime</scope>
        </dependency>
        <!-- Source: https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-jackson -->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-jackson</artifactId>
            <version>0.13.0</version>
            <scope>runtime</scope>
        </dependency>
```

JWTService.java

```
package com.klu.springmvc.service;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import javax.crypto.SecretKey;
import org.springframework.stereotype.Service;
import io.jsonwebtoken.Claims;
```

```
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;

@Service
public class JWTService {
	public String generateJWT(Map<String, String> u1, String role, Integer id) {
		String key = "dfygfygvidkjvhruiyfervuoefviheruireghigh";
		SecretKey skey = Keys.hmacShaKeyFor(key.getBytes());

		Map<String, String> claim = new HashMap<>();
		claim.put("un", u1.get("username"));
		claim.put("role", role);
		claim.put("id", id.toString());

		return Jwts.builder()
		.claims(claim)
		.issuedAt(new Date())
		.setExpiration(new Date(new Date().getTime() + 86400000))
		.signWith(skey)
		.compact();
	}

	public Map<String, String> validateJWT (String token) throws Exception {
		String key = "dfygfygvidkjvhruiyfervuoefviheruireghigh";
		SecretKey skey = Keys.hmacShaKeyFor(key.getBytes());

		Claims claim = Jwts.parser()
		.verifyWith(skey)
		.build()
		.parseSignedClaims(token)
		.getPayload();

		if(claim == null || claim.getExpiration().before(new Date())) {
			throw new Exception("Token Invalid!");
		}

		Map<String, String> parsedJWT = new HashMap<>();
		parsedJWT.put("username", claim.get("un").toString());
		parsedJWT.put("role", claim.get("role").toString());
		parsedJWT.put("id", claim.get("id").toString());

		return parsedJWT;
	}
}
```

UserService.java

```
package com.klu.springmvc.service;
```

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import com.klu.springmvc.model.User;
import com.klu.springmvc.repo.UserRepository;

@Service
public class UserService {

	@Autowired
	UserRepository repo;

	@Autowired
	JWTService jwtService;

	public Object signupService(User u1) {
		Map<String, Object> response = new HashMap<>();

		try {

			User user = repo.findByUsername(u1.getUsername());
			if(user != null) {
				response.put("code", 501);
				response.put("message", "user already exist");
			}
			else {
				u1.setRole(1);
				repo.save(u1);

				response.put("code", 200);
				response.put("message", "User Registerd Successfully");
			}

			return response;
		}
		catch (Exception e) {
			response.put("code", 500);
			response.put("message", e.getMessage());
			return response;
		}
	}

	public Object signinService(Map<String, String> u1) {
		Map<String, Object> response = new HashMap<>();
```

```
                try {
                        User user = repo.findByUsername(u1.get("username"));

                        if(user == null || !user.getPassword().equals(u1.get("password"))) {
                                response.put("code", "501");
                                response.put("message", "Authentication Failed");
                        }else {
                                response.put("code", 200);
                                response.put("jwt", jwtService.generateJWT(u1,
user.getRole().toString(), user.getId()));
                        }

                        return response;
                }
                catch (Exception e) {
                        response.put("code", 500);
                        response.put("message", e.getMessage());
                        return response;
                }
        }

}
```

Update the MS1Controller.java to below code,

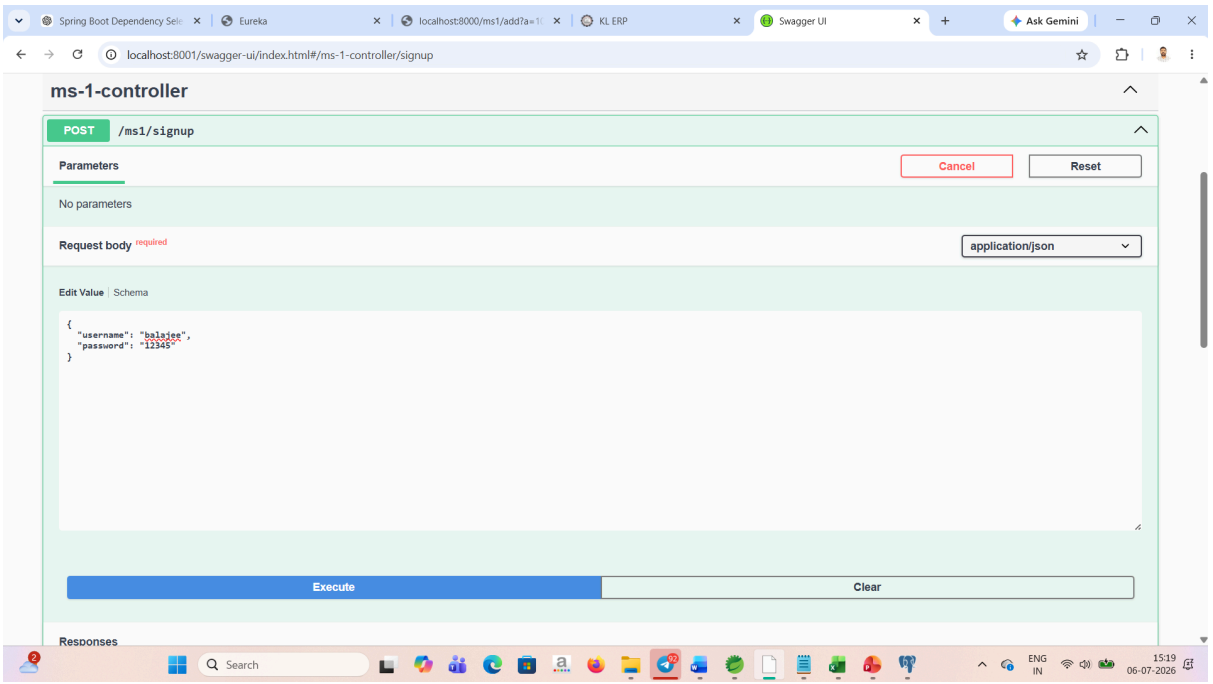
```
package com.klu.springmvc;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

//import com.example.model.Users;
import com.klu.springmvc.model.User;
import com.klu.springmvc.service.UserService;

@RestController
@RequestMapping("/ms1")
public class MS1Controller {

        @Autowired
        UserService service;

    @GetMapping("/add")
    public String add(@RequestParam("a") int a, @RequestParam("b") int b)
```

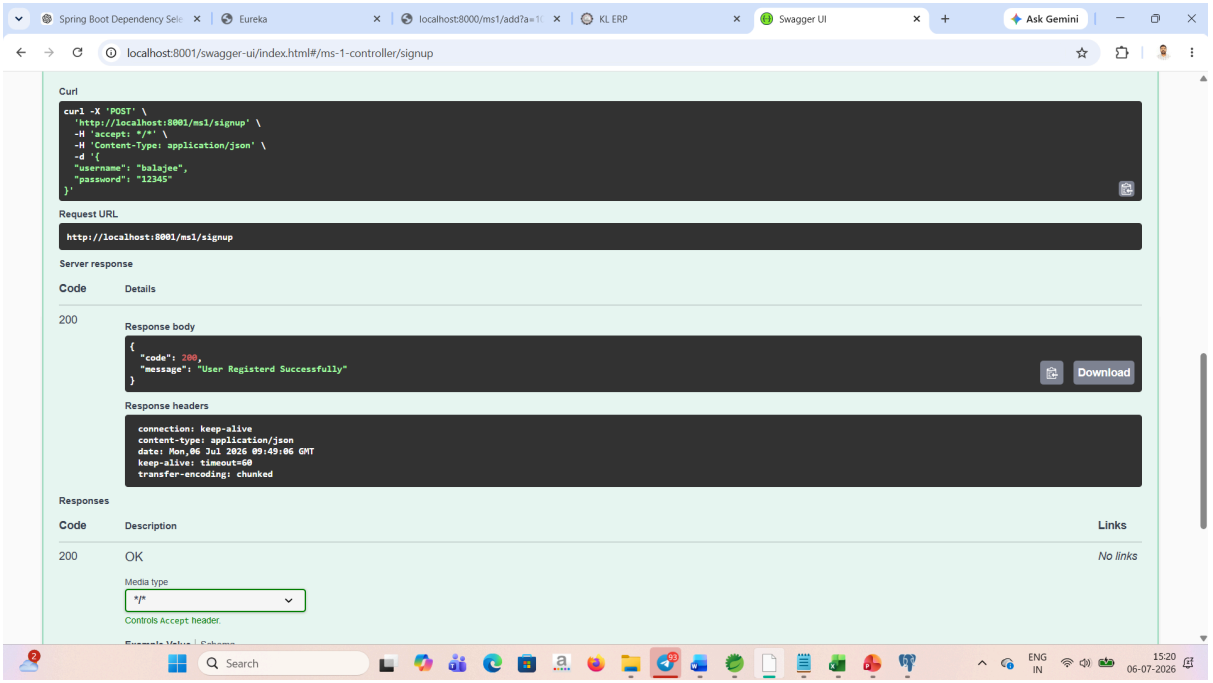
```
    {
        int result = a + b;
        return "MS 1.1 - Additon = " + result;
    }

    @PostMapping("/signin")
        public Object signin(@RequestBody Map<String, String> u1) {
                return service.signinService(u1);
        }

        @PostMapping("/signup")
        public Object signup(@RequestBody User u1) {
                return service.signupService(u1);
        }

}
```

Check the output, by "localhost:8001/swagger-ui/index.html"

Swagger	Postman
Shows APIs automatically	**Tests APIs extensively**
Documentation + Basic Testing	**Advanced Testing + Automation**
Browser-based	**Desktop/Web application**

Now signup





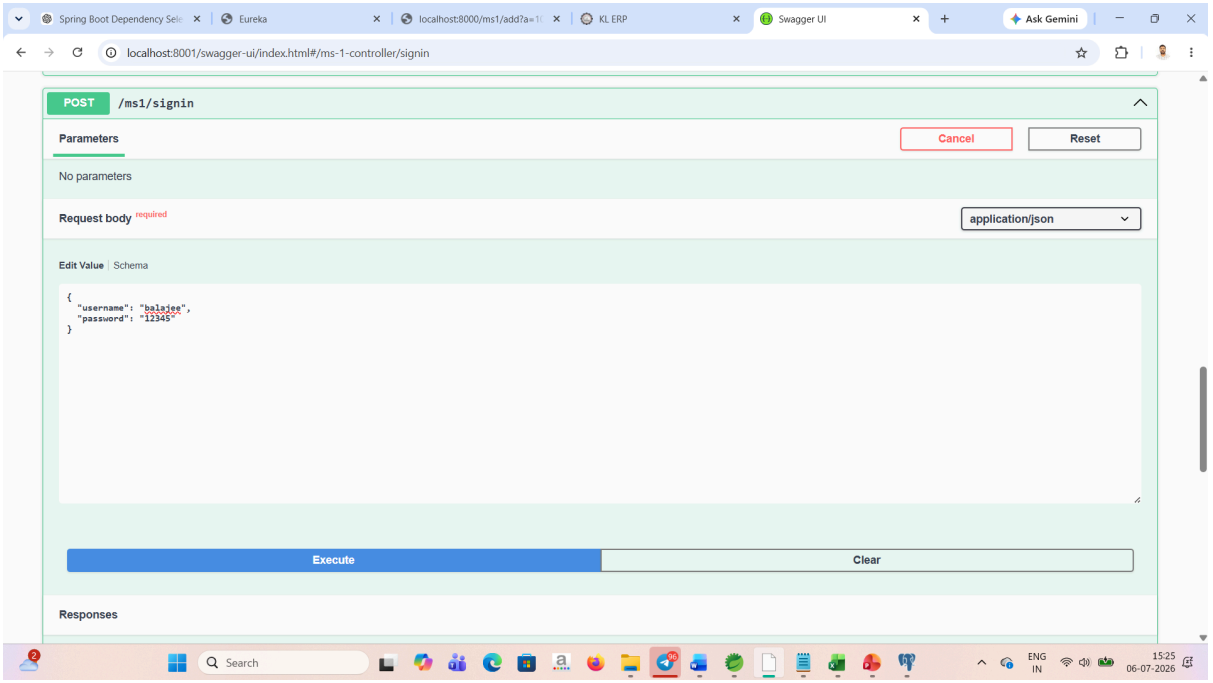
{

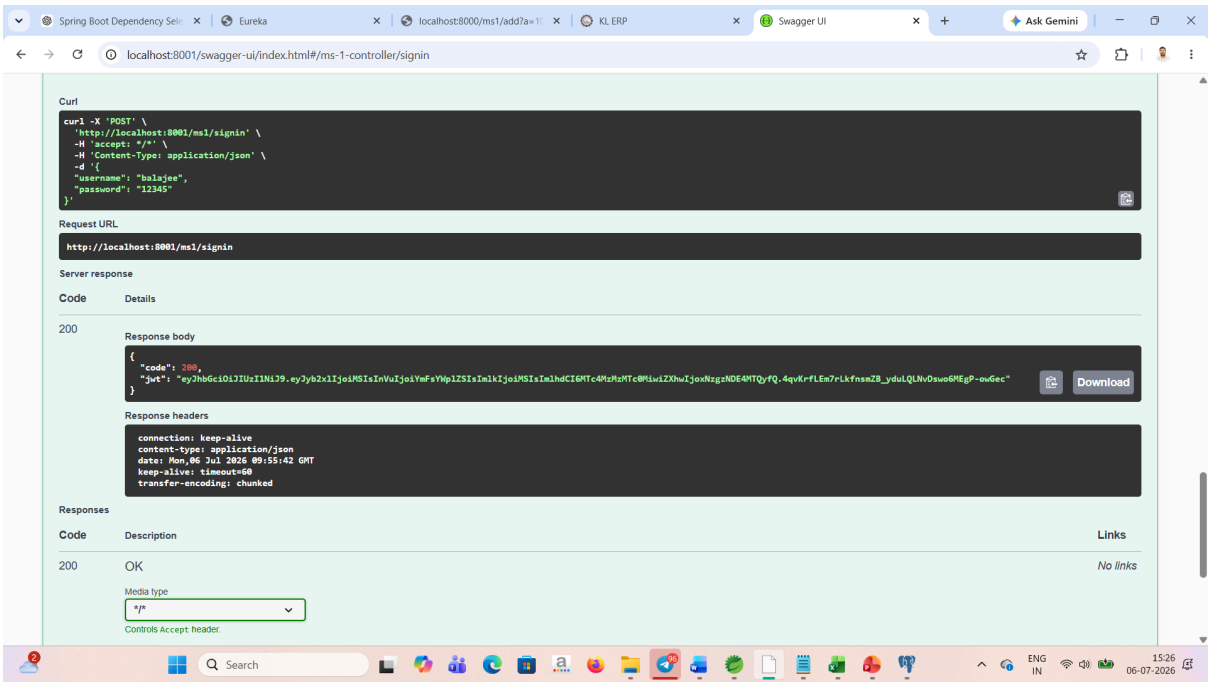
"code": 200,

"message": "User Registerd Successfully"

}

Now signin





{

"code": 200,

"jwt":
"eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVuIjoiYmFsYWplZSIsImlkIjoiMSIsImlhdCI6MTc4MzMzMTc0MiwiZXhwIjoxNzgzNDE4MTQyfQ.4qvKrfLEm7rLkfnsmZB_yduLQLNvDswo6MEgP-owGec"

}

Part 3 (CREATING TASK MICRO SERVICE IN MS2.1 FOR TASK CRUD OPERATIONS AND VALIDATE JWT TOKEN BEFORE DB OPERATIONS)

Goto MS2.1 and create package "com.klu.springmvc.model" and create class as "Task.java" and have below code.

Note: base package which is having spring boot main class file is "com.klu.springmvc".

```
package com.klu.springmvc.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Task {

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	Integer id;
	String title;
	String description;
	Integer assignedto;
	Integer priority;
	String deadline;
	Integer status;
	Integer createdby;

	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
```

```
	}
	public Integer getAssignedto() {
		return assignedto;
	}
	public void setAssignedto(Integer assignedto) {
		this.assignedto = assignedto;
	}
	public Integer getPriority() {
		return priority;
	}
	public void setPriority(Integer priority) {
		this.priority = priority;
	}
	public String getDeadline() {
		return deadline;
	}
	public void setDeadline(String deadline) {
		this.deadline = deadline;
	}
	public Integer getStatus() {
		return status;
	}
	public void setStatus(Integer status) {
		this.status = status;
	}
	public Integer getCreatedby() {
		return createdby;
	}
	public void setCreatedby(Integer createdby) {
		this.createdby = createdby;
	}

	@Override
	public String toString() {
		return "Task [id=" + id + ", title=" + title + ", description=" + description + ",
assignedto=" + assignedto
				+ ", priority=" + priority + ", deadline=" + deadline + ", status="
+ status + ", createdby="
				+ createdby + "]";
	}

}
```

Create the package "com.klu.springmvc.repo" and create interface as "TaskRepository" and have below code,

```
package com.klu.springmvc.repo;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.stereotype.Repository;
import com.klu.springmvc.model.Task;

@Repository
public interface TaskRepository extends JpaRepository<Task, Integer> {

}
```

In pom.xml add dependencies,

```
<!-- Source: https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-api -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.13.0</version>
    <scope>compile</scope>
</dependency>
<!-- Source: https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-impl -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.13.0</version>
    <scope>runtime</scope>
</dependency>
<!-- Source: https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-jackson -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.13.0</version>
    <scope>runtime</scope>
</dependency>
```

Create a package as "com.klu.springmvc.service" and create classes as "JWTService.java" (have MS1.1 JWTService.java code here) and "TaskService.java" and have below code,

```
package com.klu.springmvc.service;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import com.klu.springmvc.model.Task;
```

```
import com.klu.springmvc.repo.TaskRepository;

@Service
public class TaskService {

	@Autowired
	TaskRepository repo;

	@Autowired
	JWTService jwtService;

	public Object createTask(Task task, String token) {

		System.out.println("Task Service createTask: " + task);

		Map<String, Object> response = new HashMap<>();

		try {
			Map<String, String> parsedJWT = jwtService.validateJWT(token);

			task.setCreatedby(Integer.parseInt(parsedJWT.get("id")));
			repo.save(task);

			response.put("code", 200);
			response.put("message", "Task Created Successfully from task 1");
		}catch (Exception e) {
			response.put("code", 500);
			response.put("message", e.getMessage());
		}

		return response;
	}

	public Object getAllTasks(int page, int size, String token) {
		Map<String, Object> response = new HashMap<>();

		try {
			Map<String, String> parsedJWT = jwtService.validateJWT(token);
			Pageable pageable = PageRequest.of(page-1, size);
			Page<Task> tasks = repo.findAll(pageable);

			response.put("code", 200);
			response.put("page", page);
			response.put("size", size);
			response.put("totalpages", tasks.getTotalPages());
			response.put("tasks", tasks.getContent());
			return response;
		}catch (Exception e) {
			response.put("code", 500);
```

```
                response.put("message", e.getMessage());
                return response;
            }
        }

        public Object deleteTask(int id, String token) {
Map<String, Object> response = new HashMap<>();

            try {
                Map<String, String> parsedJWT = jwtService.validateJWT(token);

                Task task = repo.findById(id).get();
                if(task == null) {
                    throw new Exception("user not exist");
                }

                repo.deleteById(id);

                response.put("code", 200);
                response.put("message", "deleted successfully");
                return response;
            }catch (Exception e) {
                response.put("code", 500);
                response.put("message", e.getMessage());
                return response;
            }
        }

        public Object getTask(int id, String token) {
            Map<String, Object> response = new HashMap<>();

            try {
                Map<String, String> parsedJWT = jwtService.validateJWT(token);

                Task task = repo.findById(id).get();

                if(task == null) {
                    throw new Exception("User Not Found");
                }

                response.put("code", 200);
                response.put("task", task);
                return response;
            }catch (Exception e) {
                response.put("code", 500);
                response.put("message", e.getMessage());
                return response;
            }
        }
```

```
	public Object updateTask(int id, Task task, String token) {
		Map<String, Object> response = new HashMap<>();

		try {
			Map<String, String> parsedJWT = jwtService.validateJWT(token);

			Task task1 = repo.findById(id).get();
			if(task1 == null) {
				throw new Exception("user not exist");
			}

			task1.setTitle(task.getTitle());
			task1.setDescription(task.getDescription());
			task1.setAssignedto(task.getAssignedto());
			task1.setDeadline(task.getDeadline());
			task1.setPriority(task.getPriority());
			task1.setStatus(task.getStatus());
			task1.setCreatedby(Integer.parseInt(parsedJWT.get("id")));
			repo.save(task1);

			response.put("code", 200);
			response.put("message", "updated successfully");
			return response;
		}catch (Exception e) {
			response.put("code", 500);
			response.put("message", e.getMessage());
			return response;
		}
	}

}
```

Goto ms2 controller, have the below code,

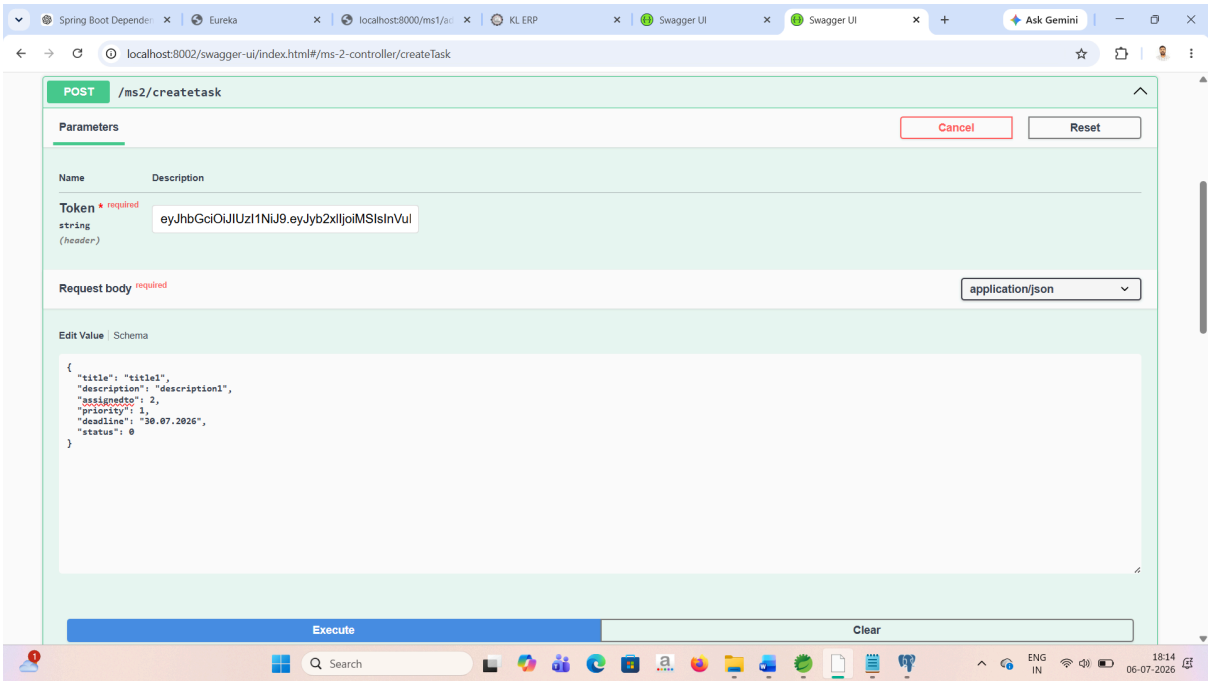
```
package com.klu.springmvc;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import org.springframework.web.bind.annotation.RestController;
import com.klu.springmvc.model.Task;

@RestController
@RequestMapping("/ms2")
public class MS2Controller {

	@Autowired
	com.klu.springmvc.service.TaskService taskService;

   @GetMapping("/mul")
   public String add(@RequestParam("a") int a, @RequestParam("b") int b)
   {
      int result = a * b;
      return "MS 2.1 - Multiplication = " + result;
   }

	@PostMapping("/createtask")
	public Object createTask(@RequestBody Task task, @RequestHeader("Token")
String token) {
		System.out.println("Task Controller createTask: " + task);
		return taskService.createTask(task, token);
	}

	@GetMapping("/getalltasks/{PAGE}/{SIZE}")
	public Object getAllTasks(@PathVariable("PAGE") int page, @PathVariable("SIZE")
int size, @RequestHeader("Token") String token) {
		return taskService.getAllTasks(page, size, token);
	}

	@DeleteMapping("/deletetask/{ID}")
	public Object deleteTask(@PathVariable("ID") int id, @RequestHeader("Token")
String token) {
		return taskService.deleteTask(id, token);
	}

	@GetMapping("/gettask/{ID}")
	public Object getTask(@PathVariable("ID") int id, @RequestHeader("Token") String
token) {
		return taskService.getTask(id, token);
	}

	@PutMapping("/updatetask/{ID}")
	public Object updateTask(@PathVariable("ID") int id, @RequestBody Task task,
@RequestHeader("Token") String token) {
		return taskService.updateTask(id, task, token);
	}
```

}

Now check the output by going to localhost:8002/swagger-ui.html

Create task



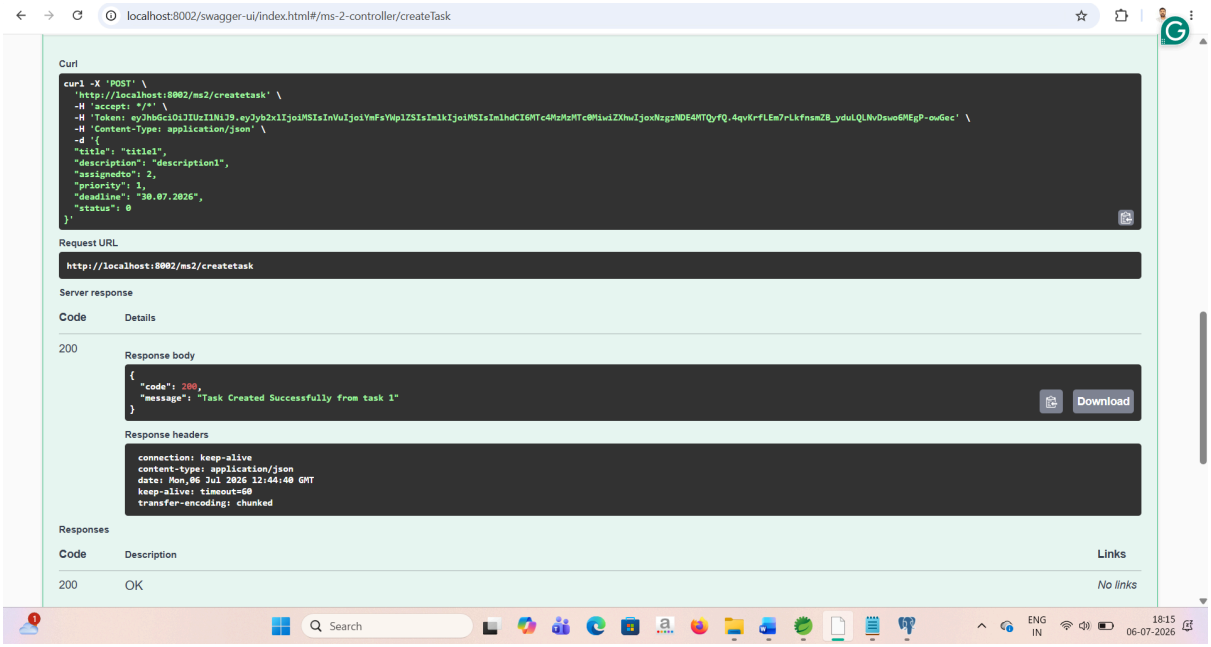
Input

Token:

eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVuIjoiYmFsYWplZSIsImlkIjoiMSIsImlhdCI6MTc4MzMzMTc0MiwiZXhwIjoxNzgzNDE4MTQyfQ.4qvKrfLEm7rLkfnsmZB_yduLQLNvDswo6MEgP-owGec

json:

```
{
  "title": "title1",
  "description": "description1",
  "assignedto": 2,
  "priority": 1,
  "deadline": "30.07.2026",
  "status": 0
}
```

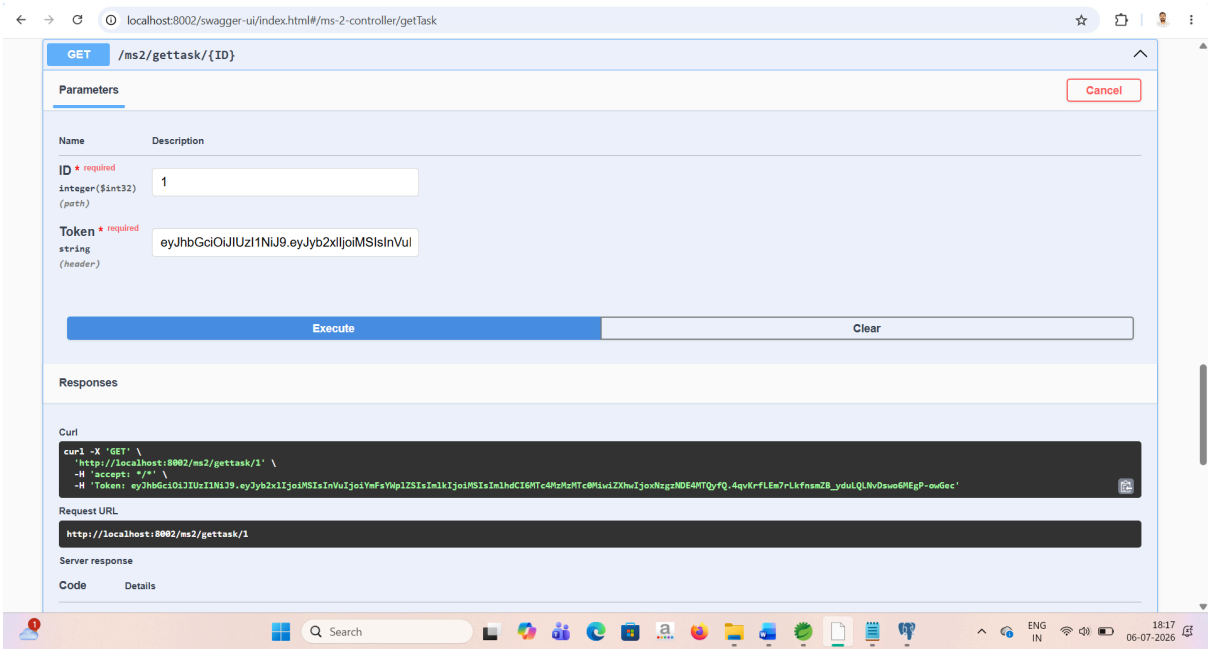


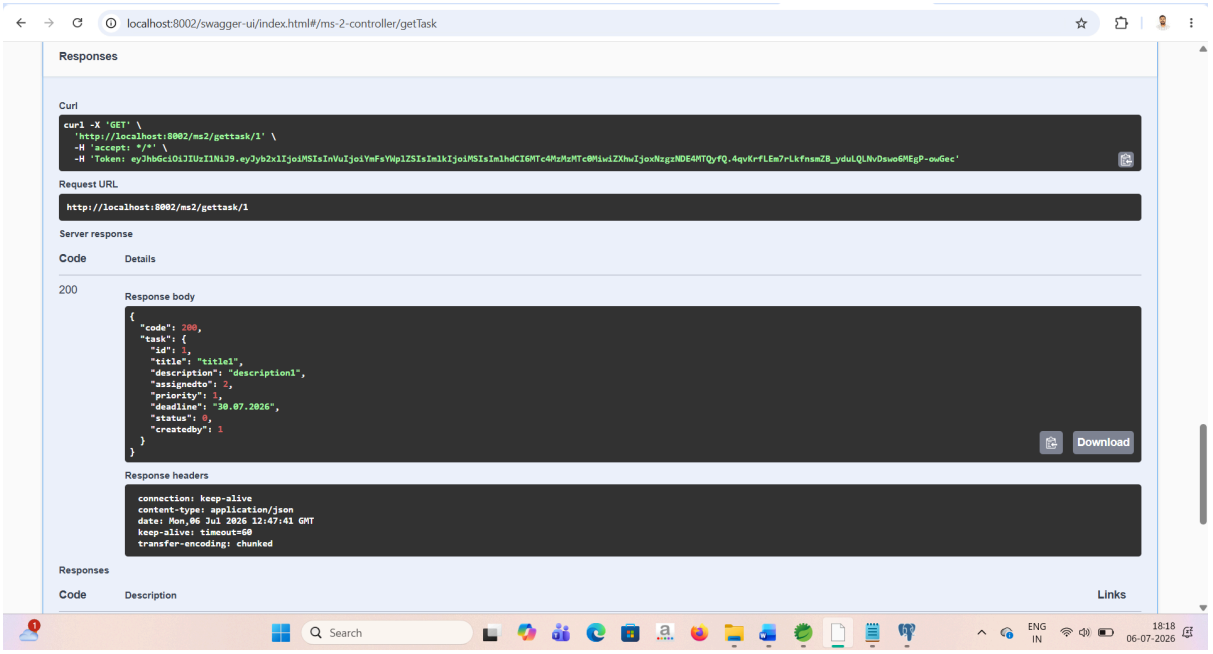

```
{
 "code": 200,
 "message": "Task Created Successfully from task 1"
}
```


	id [PK] integer	assignedto integer	createdby integer	deadline character varying (255)	description character varying (255)	priority integer	status integer	title character varying (255)
1	1	2	1	30.07.2026	description1	1	0	title1

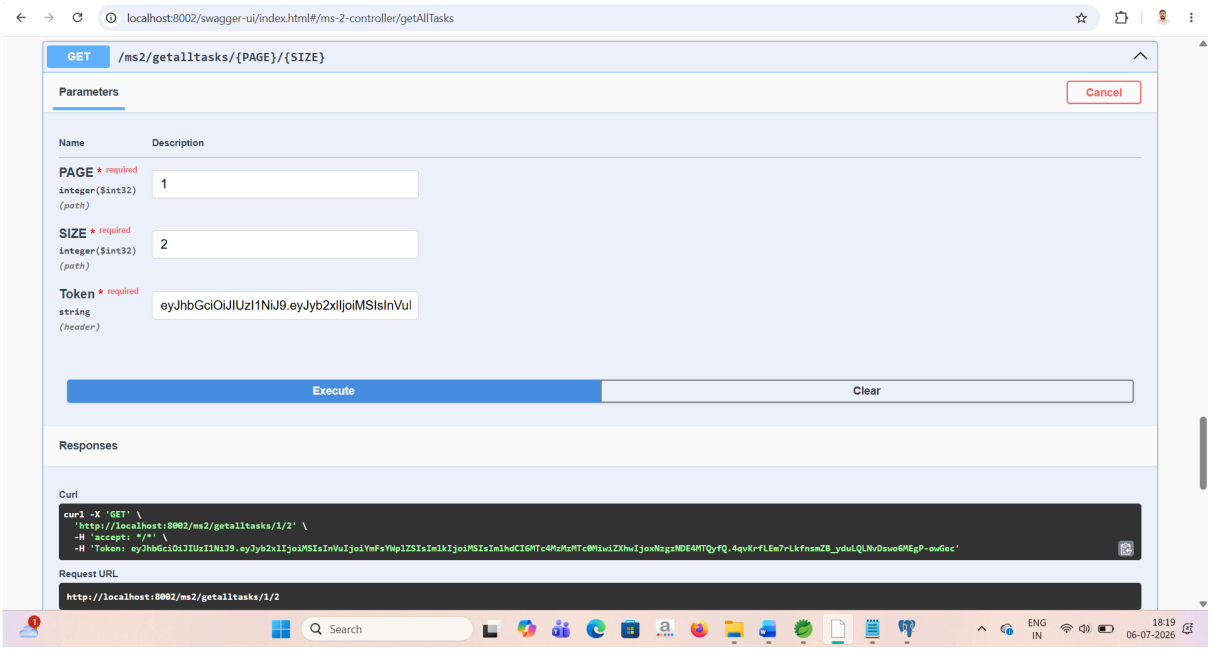
Similary, check other endpoints in task (crud operations)

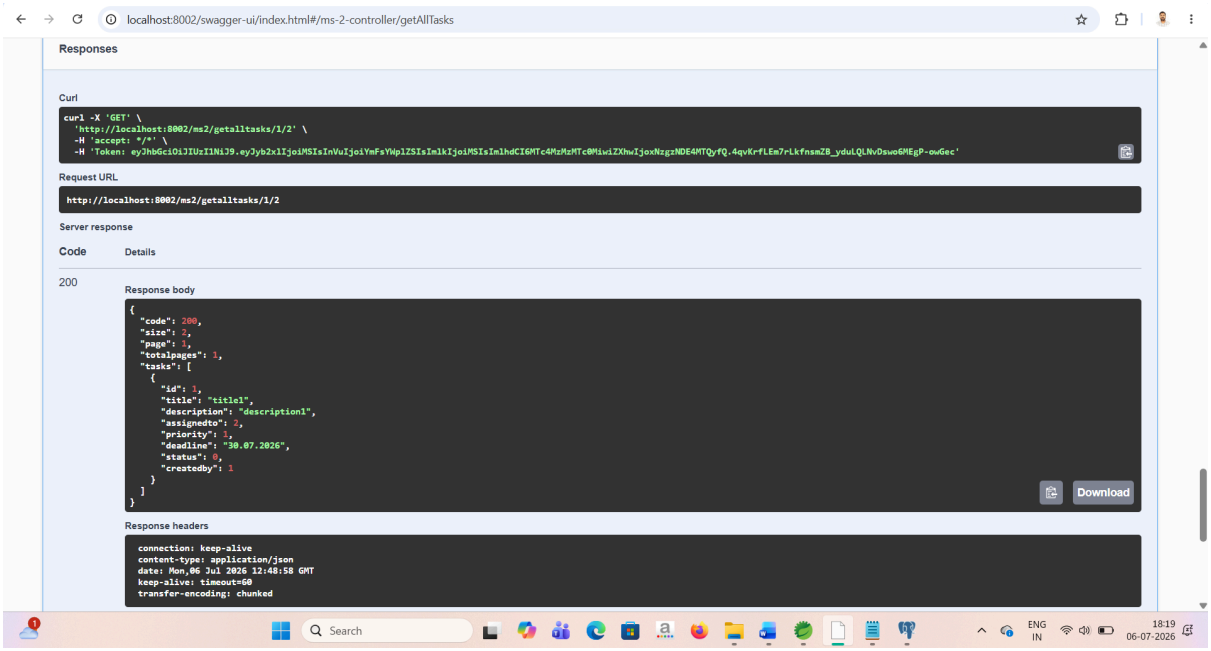
Gettask



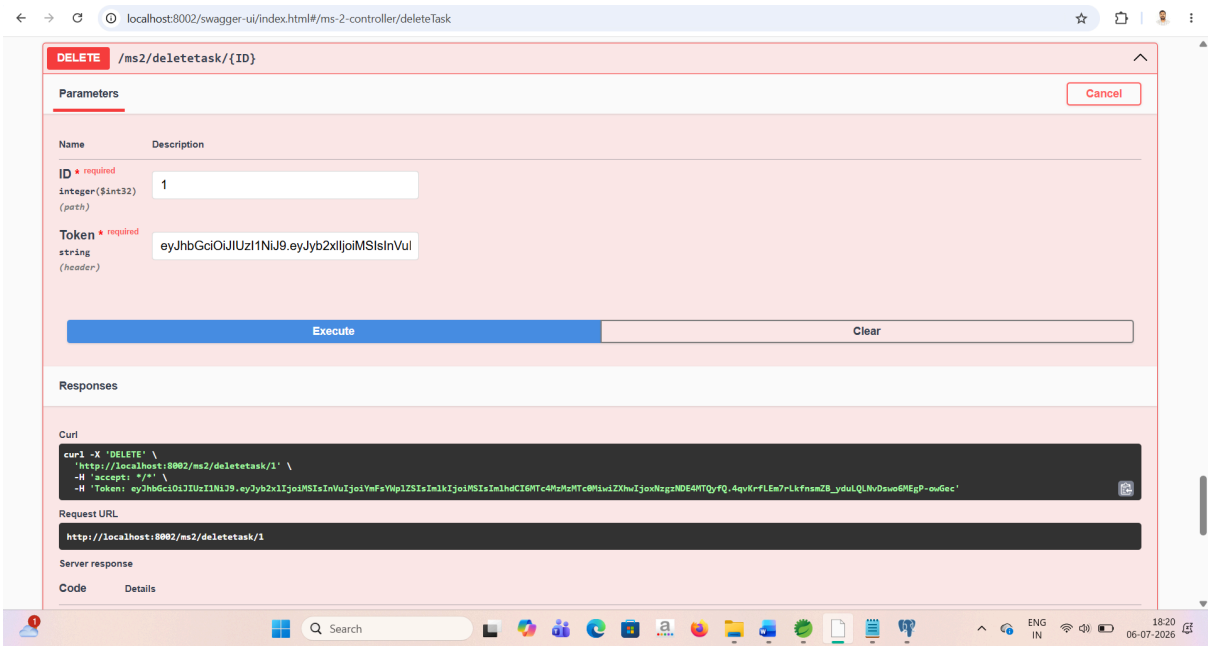

localhost:8002/swagger-ui/index.html#/ms-2-controller/getTask
Responses
Curl
curl -X 'GET' \
'http://localhost:8002/ms2/gettask/1' \
-H 'accept: */*' \
-H 'Token: eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVuIjoiYmFsYWplZSIsImlkIjoiMSIsImlhdCI6MTc4MzMzMTc0MiwiZXhwIjoxNzgzNDE4MTQyfQ.4qvKrfLEm7rLkfnsmZB_yduLQLNvDswo6MEgP-owGec'
Request URL
http://localhost:8002/ms2/gettask/1
Server response
Code
Details
200
Response body
{
"code": 200,
"task": {
"id": 1,
"title": "title1",
"description": "description1",
"assignedto": 2,
"priority": 1,
"deadline": "30.07.2026",
"status": 0,
"createdby": 1
}
}
Download
Response headers
connection: keep-alive
content-type: application/json
date: Mon,06 Jul 2026 12:47:41 GMT
keep-alive: timeout=60
transfer-encoding: chunked
Responses
Code
Description
Links
Search
ENG
IN
18:18
06-07-2026

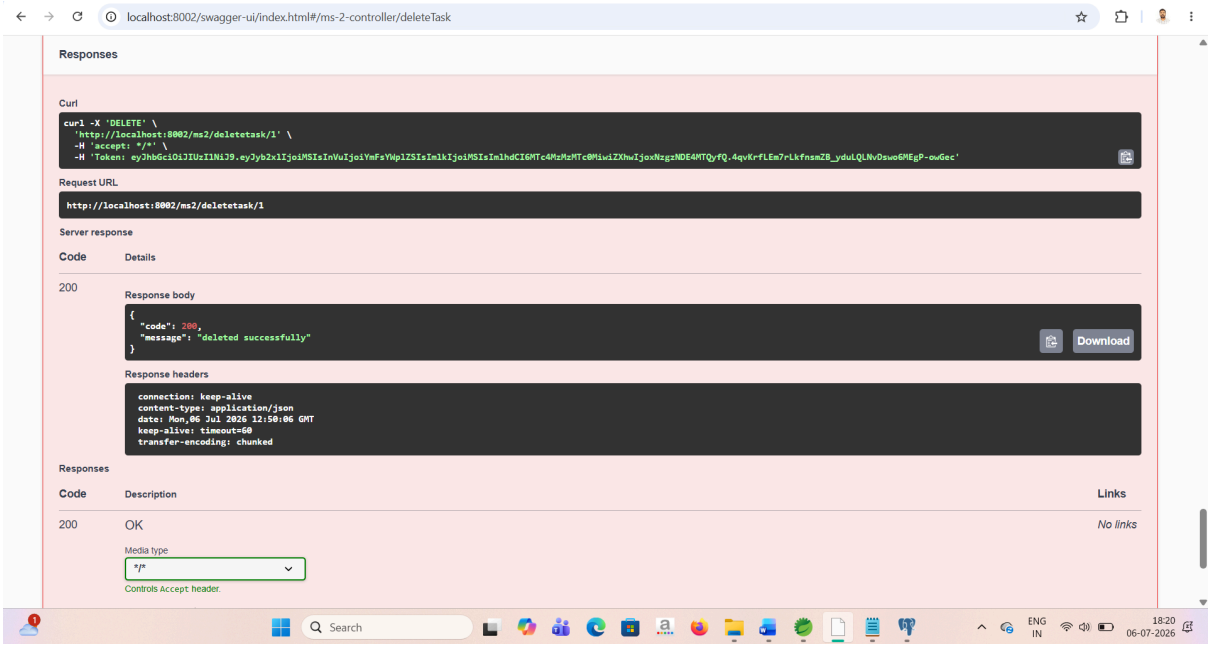
Getalltask


localhost:8002/swagger-ui/index.html#/ms-2-controller/getAllTasks
GET
/ms2/getalltasks/{PAGE}/{SIZE}
Parameters
Cancel
Name
Description
PAGE * required
integer($int32)
(path)
1
SIZE * required
integer($int32)
(path)
2
Token * required
string
(header)
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVul
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:8002/ms2/getalltasks/1/2' \
-H 'accept: */*' \
-H 'Token: eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVuIjoiYmFsYWplZSIsImlkIjoiMSIsImlhdCI6MTc4MzMzMTc0MiwiZXhwIjoxNzgzNDE4MTQyfQ.4qvKrfLEm7rLkfnsmZB_yduLQLNvDswo6MEgP-owGec'
Request URL
http://localhost:8002/ms2/getalltasks/1/2
Search
ENG
IN
18:19
06-07-2026


localhost:8002/swagger-ui/index.html#/ms-2-controller/getAllTasks
Responses
Curl
curl -X 'GET' \
'http://localhost:8002/ms2/getalltasks/1/2' \
-H 'accept: */*' \
-H 'Token: eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVuIjoiYmFsYWplZSIsImlkIjoiMSIsImlhdCI6MTc4MzMzMTc0MiwiZXhwIjoxNzgzNDE4MTQyfQ.4qvKrfLEm7rLkfnsmZB_yduLQLNvDswo6MEgP-owGec'
Request URL
http://localhost:8002/ms2/getalltasks/1/2
Server response
Code
Details
200
Response body
{
"code": 200,
"size": 2,
"page": 1,
"totalpages": 1,
"tasks": [
{
"id": 1,
"title": "title1",
"description": "description1",
"assignedto": 2,
"priority": 1,
"deadline": "30.07.2026",
"status": 0,
"createdby": 1
}
]
}
Download
Response headers
connection: keep-alive
content-type: application/json
date: Mon,06 Jul 2026 12:48:58 GMT
keep-alive: timeout=60
transfer-encoding: chunked
Search
ENG
IN
18:19
06-07-2026

Delete task


localhost:8002/swagger-ui/index.html#/ms-2-controller/deleteTask
DELETE
/ms2/deletetask/{ID}
Parameters
Cancel
Name
Description
ID * required
integer($int32)
(path)
1
Token * required
string
(header)
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVul
Execute
Clear
Responses
Curl
curl -X 'DELETE' \
'http://localhost:8002/ms2/deletetask/1' \
-H 'accept: */*' \
-H 'Token: eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVuIjoiYmFsYWplZSIsImlkIjoiMSIsImlhdCI6MTc4MzMzMTc0MiwiZXhwIjoxNzgzNDE4MTQyfQ.4qvKrfLEm7rLkfnsmZB_yduLQLNvDswo6MEgP-owGec'
Request URL
http://localhost:8002/ms2/deletetask/1
Server response
Code
Details
Search
ENG
IN
18:20
06-07-2026



Part 4 (DUPLICATING MICRO SERVICE MS2.1 TO MS2.2)

Remove the entire existing MS2.2, then duplicate the MS2.1 to MS2.2

In the controller file change the pring statements to MS2.2 from MS2.1 for identifying the difference

Copy the JWT dependencies and paste in the pom.xml (It will be already there.)

In the application.properties file from the MS2.2, update the server.port to 8003

Go to TaskService.java of both MS2.1 and MS2.2, differentiate the print statements by adding MS2.1 and MS2.2 in the respective TaskService.java files.

Part 5 (NOW TEST THE GATEWAY AND LOAD BALANCER FROM THE PORT 8000 FOR TASK SERVICE)

Create Task

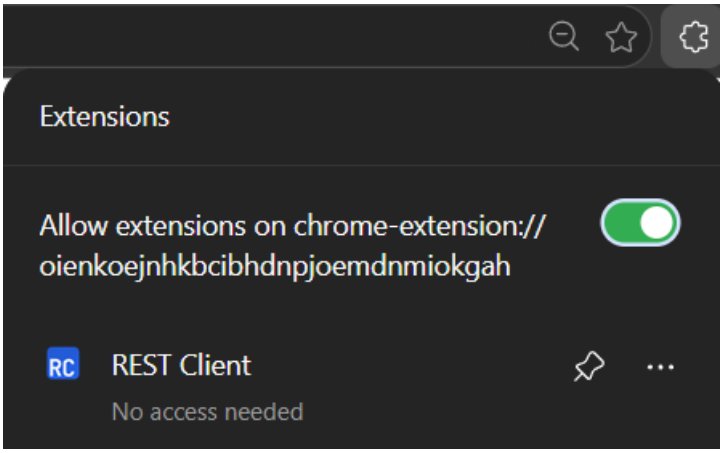
Step 1:
in google search, rest extension chrome

link:
https://chromewebstore.google.com/detail/rest-client/oienkoejnhkbcibhdnpjoemdnmiokgah

Install the extension to test the server (nodejs) separately without api gateway and reactjs integration

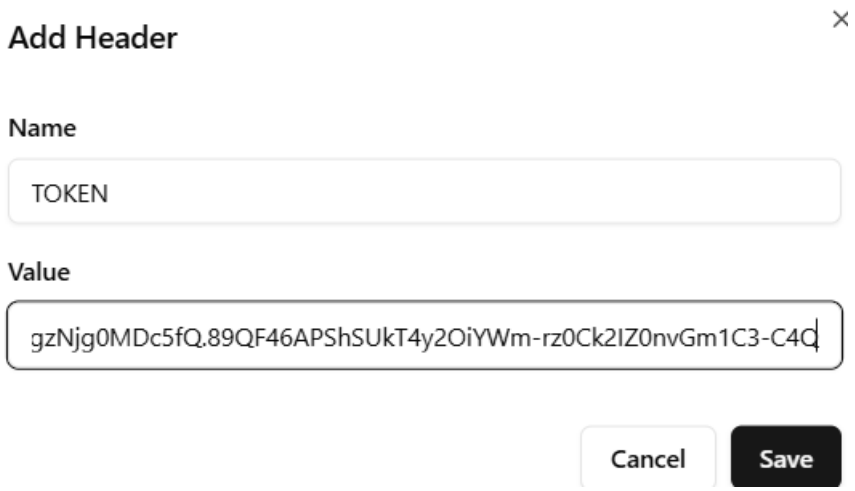
Step 2:

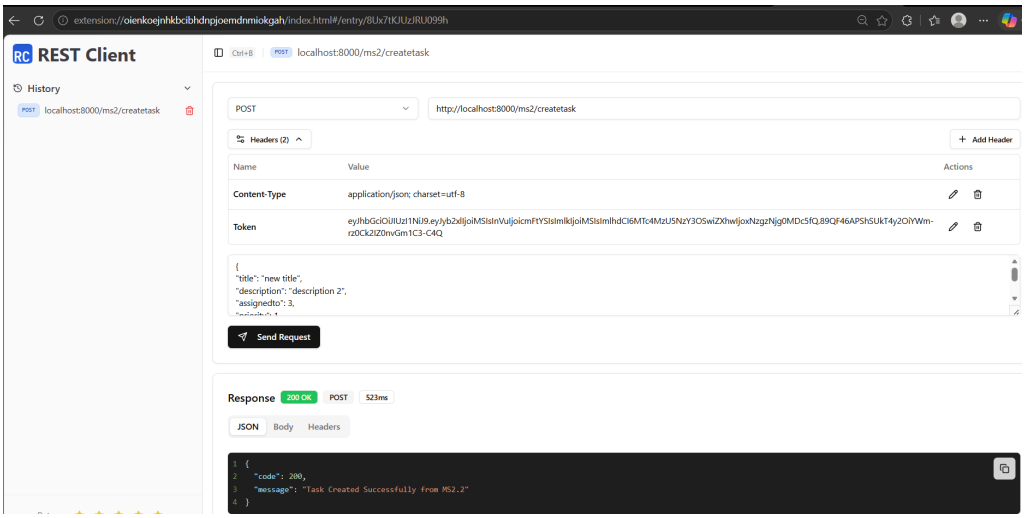
Open extension by Click on the extension button as shown in the screenshot from the browser.



Then select the POST, enter url, add the token by click on the Add Header Button, enter the body text as shown in the screen shot.

provide url as post method: http://localhost:8000/ms2/createtask





json header

Token

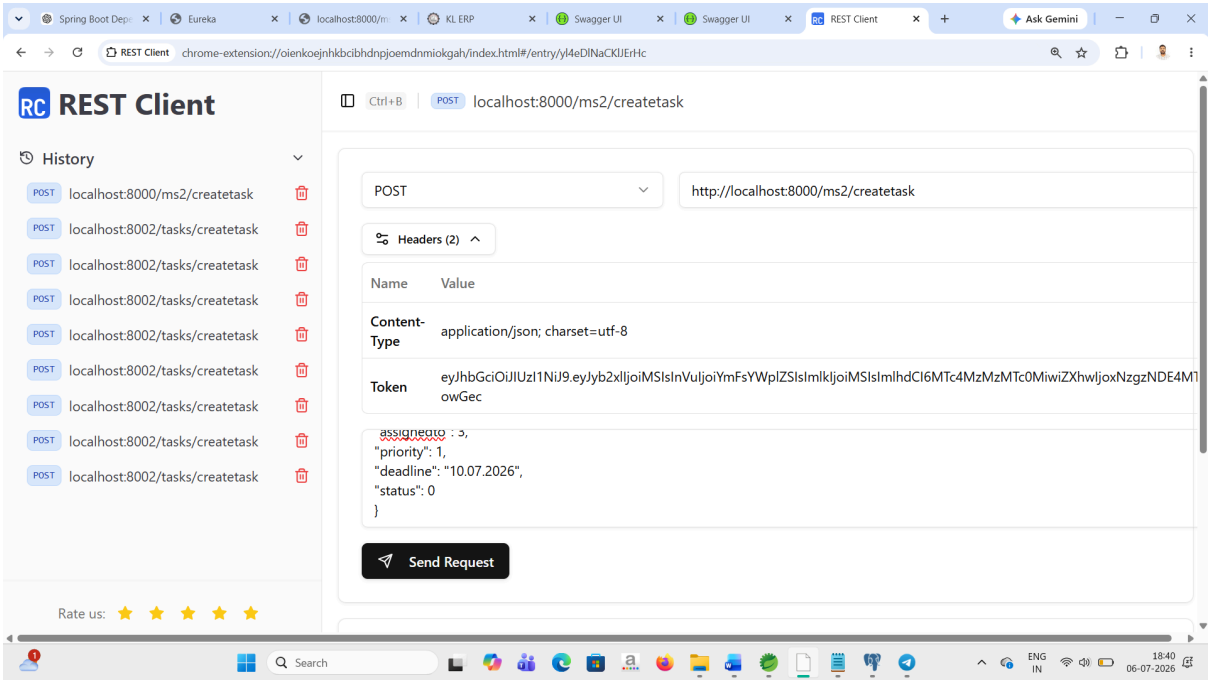
eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjoiMSIsInVuIjoiYmFsYWplZSIsImlkIjoiMSIsImlhdCI6MTc4MzMzMTc0MiwiZXhwIjoxNzgzNDE4MTQyfQ.4qvKrfLEm7rLkfnsmZB_yduLQLNvDswo6MEgP-owGec

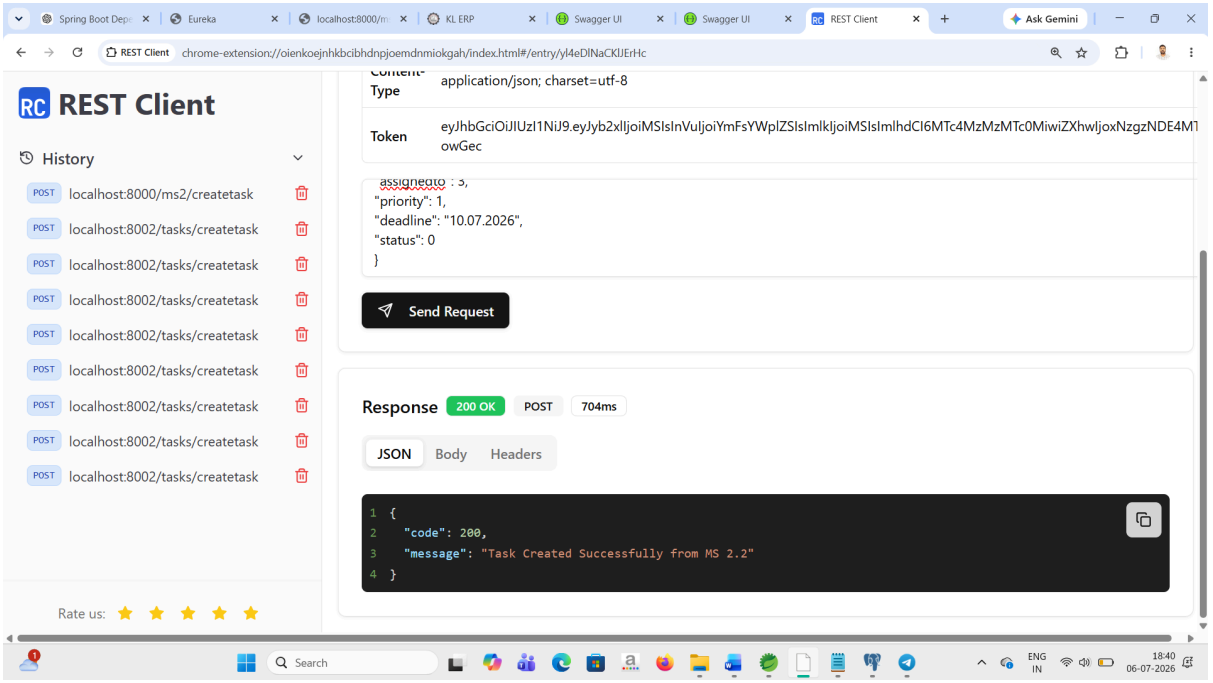
json req body data

```
{
"title": "new title",
"description": "description 1",
"assignedto": 3,
"priority": 1,
"deadline": "10.07.2026",
"status": 0
}
```

click **send** button

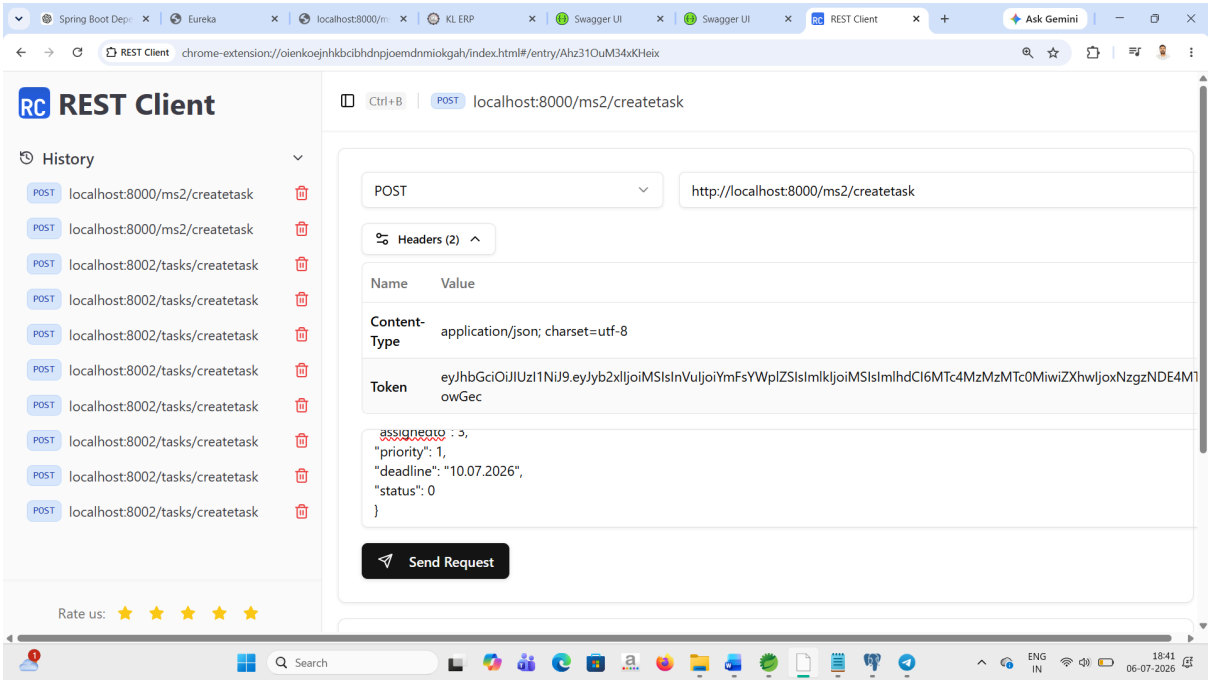
output:

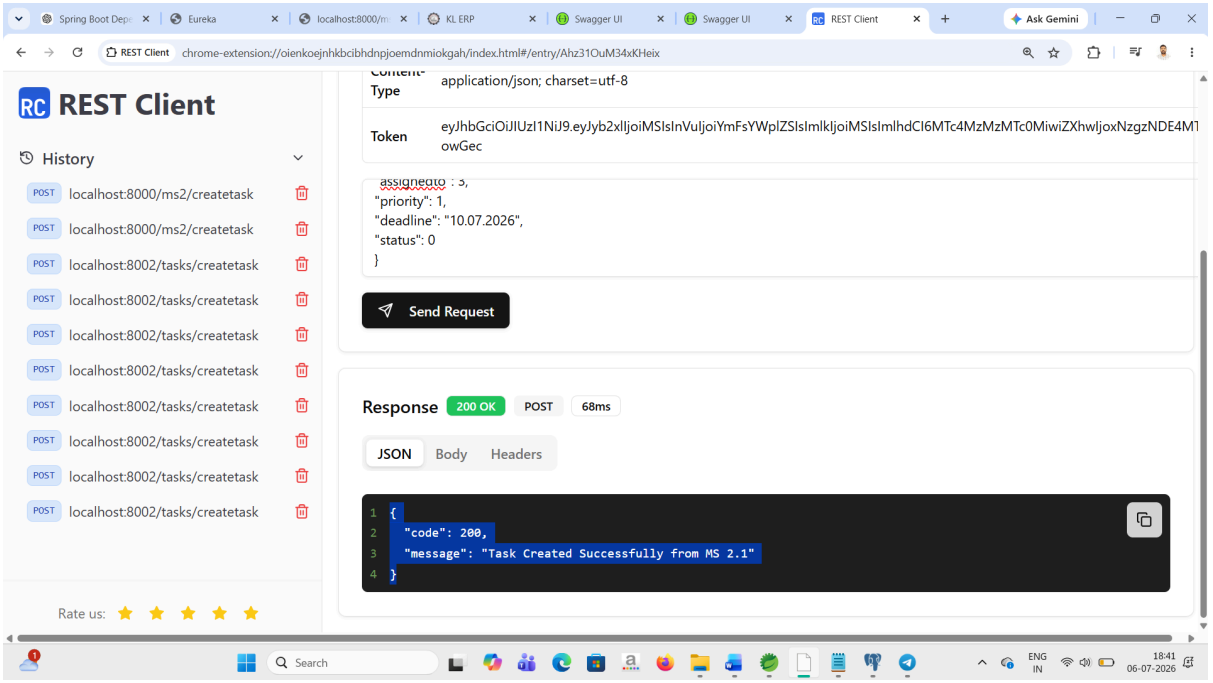





```
{
  "code": 200,
  "message": "Task Created Successfully from MS 2.2"
}
```

Again click send button for second output






```
{
  "code": 200,
  "message": "Task Created Successfully from MS 2.1"
}
```