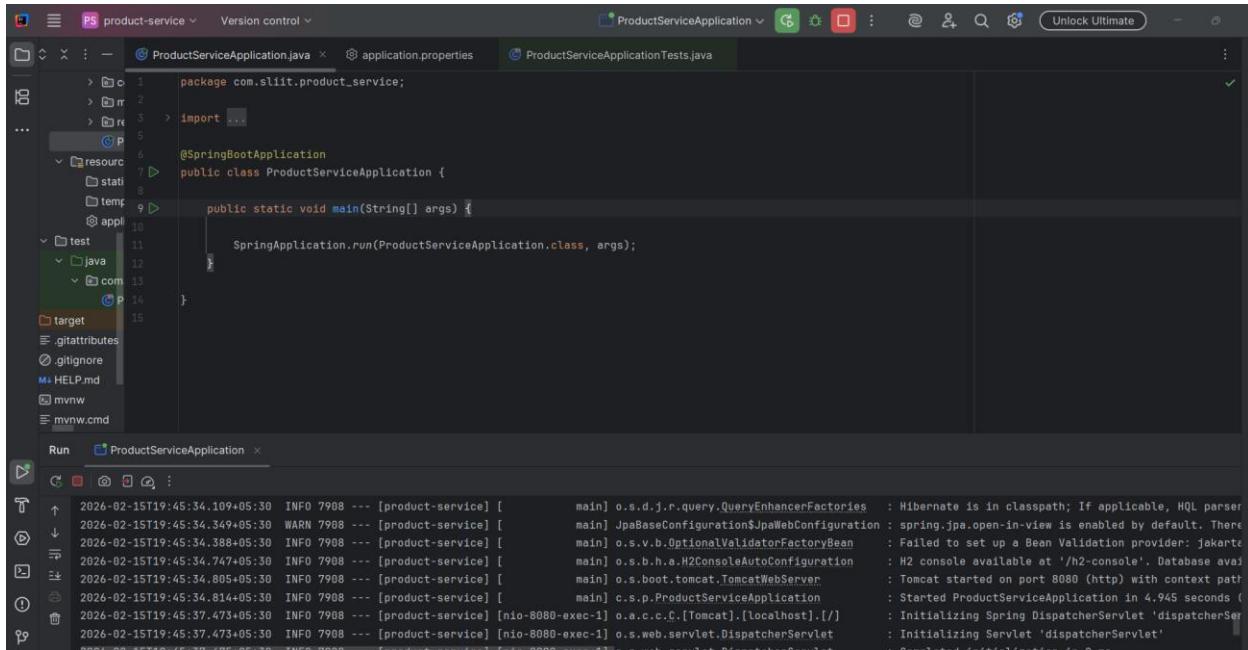


# IT22341204 – K Rangana Malmi Nadee

## Lab 03 - Current Trends in Software Engineering (SE4010)

### Run Application in IDE

- Application was executed successfully. The console shows that the embedded Tomcat server started on port 8080 and the application started without errors.



The screenshot shows the IntelliJ IDEA interface. The code editor displays `ProductServiceApplication.java` with the following code:

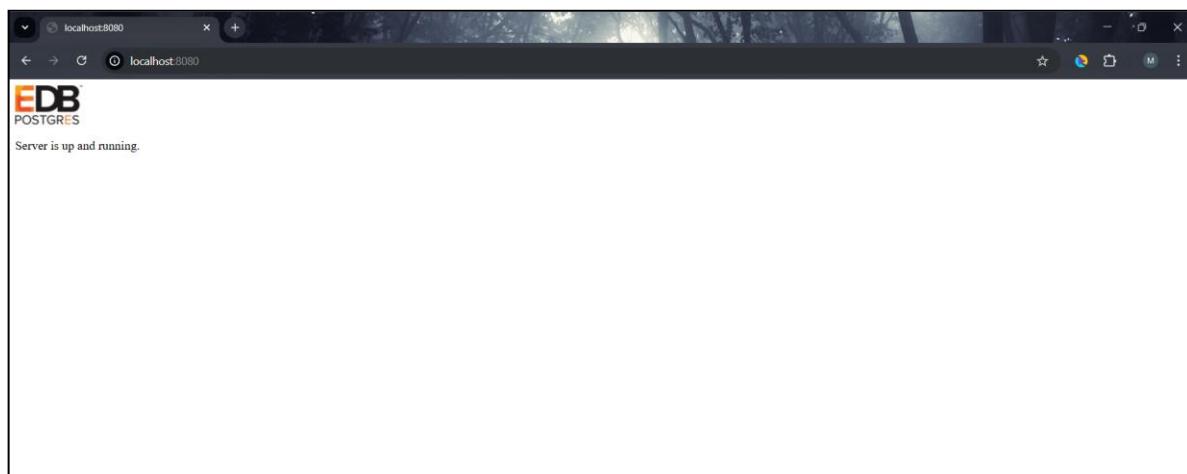
```
package com.slitt.product_service;
import ...;
import org.springframework.boot.SpringApplication;
public class ProductServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(ProductServiceApplication.class, args);
    }
}
```

The run console at the bottom shows the application's log output:

```
2026-02-15T19:45:34,109+05:30 INFO 7908 --- [product-service] [main] o.s.d.j.r.query.QueryEnhancerFactories : Hibernate is in classpath; if applicable, HQL parser
2026-02-15T19:45:34,349+05:30 WARN 7908 --- [product-service] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. There
2026-02-15T19:45:34,388+05:30 INFO 7908 --- [product-service] [main] o.s.v.b.OptionalValidatorFactoryBean : Failed to set up a Bean Validation provider: jakarta
2026-02-15T19:45:34,747+05:30 INFO 7908 --- [product-service] [main] o.s.b.h.a.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database avail
2026-02-15T19:45:34,805+05:30 INFO 7908 --- [product-service] [main] o.s.boot.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path
2026-02-15T19:45:34,814+05:30 INFO 7908 --- [product-service] [main] c.s.p.ProductServiceApplication : Started ProductServiceApplication in 4.945 seconds ((
2026-02-15T19:45:37,473+05:30 INFO 7908 --- [product-service] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2026-02-15T19:45:37,473+05:30 INFO 7908 --- [product-service] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2026-02-15T19:45:37,475+05:30 INFO 7908 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Completed initialization in 2 ms
```

Open browser and visit :

URL → <http://localhost:8080>

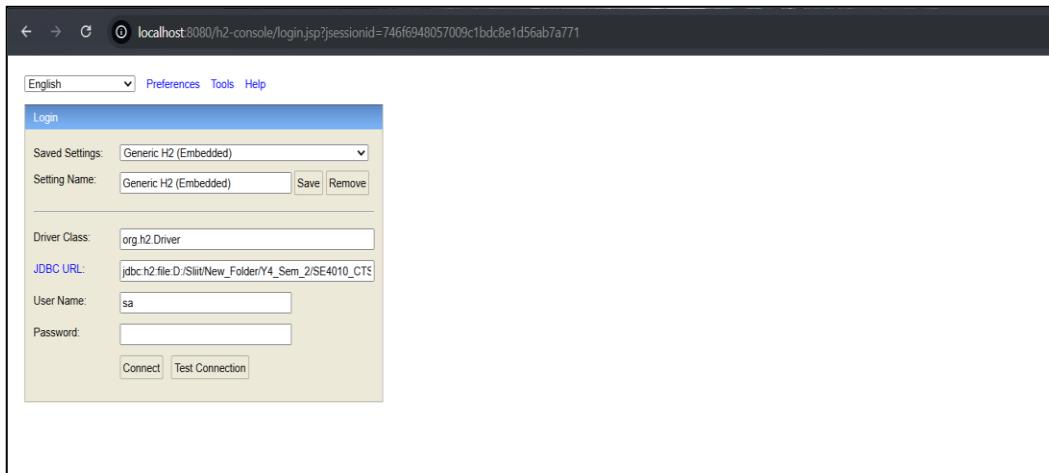


## Access H2 Console

- The H2 in-memory database console was accessed successfully.

Open browser and visit :

URL → <http://localhost:8080/h2-console>



## H2 Product Table

A screenshot of the H2 Console interface showing the "PRODUCT" table. The left sidebar shows the database structure with "jdbc:h2:file:D:/Slit/New\_Folder/Y4\_Sem\_2/SE4010\_CTS" as the connection, "INFORMATION\_SCHEMA" and "Users" under "SCHEMAS", and "H2 2.4.240 (2025-09-22)" under "VERSION". The main area displays SQL statements being run:

```
Select * From Product
Insert Into Product (NAME, PRICE) Values('Laptop', 200000);
Insert Into Product (NAME, PRICE) Values ('Smartphone', 80000);
Insert Into Product (NAME, PRICE) Values ('Tablet', 40000);
Insert Into Product (NAME, PRICE) Values('Smartwatch', 20000);
Insert Into Product (NAME, PRICE) Values('Headphones', 1500);
Insert Into Product (NAME, PRICE) Values('Mouse', 3000);

SELECT * FROM PRODUCT WHERE ID = 3;
DELETE FROM PRODUCT WHERE ID = 6;
```

The results of the "Select \* From Product" query are shown in a table:

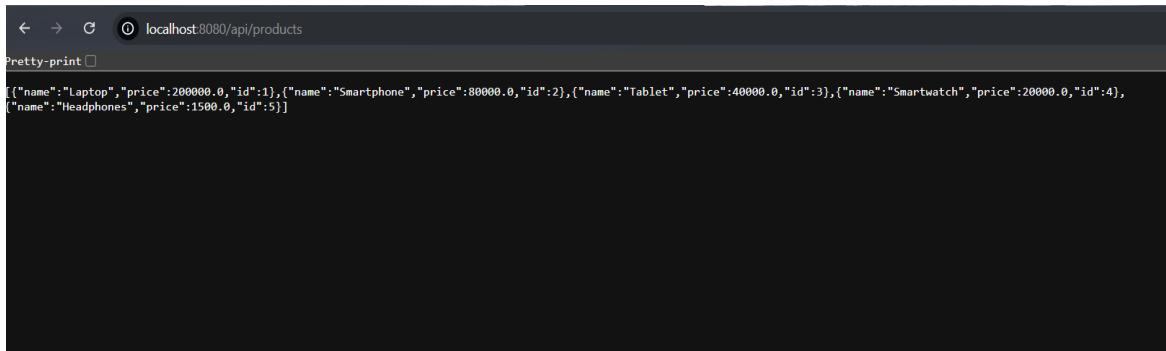
ID	NAME	PRICE
1	Laptop	200000.0
2	Smartphone	80000.0
3	Tablet	40000.0
4	Smartwatch	20000.0
5	Headphones	1500.0

(5 rows, 1 ms)

This endpoint returns all products in JSON format, confirming that the GET API works correctly.

Open browser and visit :

URL → <http://localhost:8080/api/products>



A screenshot of a web browser window. The address bar shows "localhost:8080/api/products". Below the address bar, there is a "Pretty-print" checkbox. The main content area displays a JSON array of five products:

```
[{"name": "Laptop", "price": 20000.0, "id": 1}, {"name": "Smartphone", "price": 8000.0, "id": 2}, {"name": "Tablet", "price": 40000.0, "id": 3}, {"name": "Smartwatch", "price": 20000.0, "id": 4}, {"name": "Headphones", "price": 1500.0, "id": 5}]
```

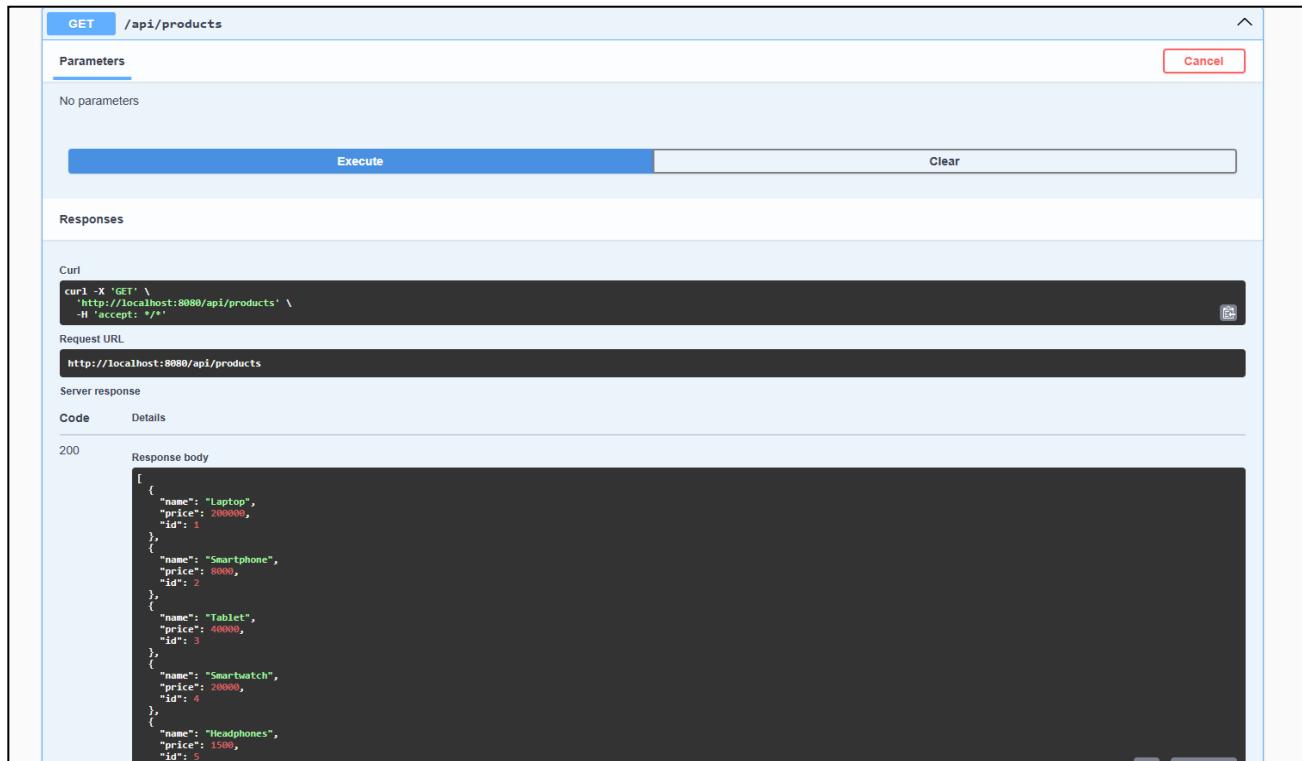
## Swagger UI – API List

- Swagger UI displays all available REST endpoints and allows interactive testing

Open browser and visit :

URL → <http://localhost:8080/swagger-ui.html>

## GET /products – Retrieve all products



A screenshot of the Swagger UI interface for the "/products" endpoint. The top navigation bar shows "GET /api/products". The "Parameters" section indicates "No parameters". Below it is an "Execute" button and a "Clear" button. The "Responses" section includes a "Curl" code block and a "Request URL" input field containing "http://localhost:8080/api/products". The "Server response" section shows a "Code" tab selected, displaying a status code of 200 and a "Details" tab. The "Response body" section displays the same JSON array of products as shown in the browser screenshot above:

```
[{"name": "Laptop", "price": 20000.0, "id": 1}, {"name": "Smartphone", "price": 8000.0, "id": 2}, {"name": "Tablet", "price": 40000.0, "id": 3}, {"name": "Smartwatch", "price": 20000.0, "id": 4}, {"name": "Headphones", "price": 1500.0, "id": 5}]
```

## POST /products – Create new product

The screenshot shows a REST API testing interface with the following details:

- Method:** POST
- Endpoint:** /api/products
- Parameters:** No parameters
- Request body (required):** application/json
- Request body content:**

```
{ "name": "Monitor", "price": 75000 }
```
- Buttons:** Execute, Clear, Cancel, Reset
- Responses:**
  - Curl:** curl -X 'POST' \ 'http://localhost:8080/api/products' \ -H 'accept: \*/\*' \ -H 'Content-Type: application/json' \ -d '{ "name": "Monitor", "price": 75000 }'
  - Request URL:** http://localhost:8080/api/products
  - Server response:**

Code	Details
200	<b>Response body:</b> <pre>{ "name": "Monitor", "price": 75000, "id": 17 }</pre> <b>Download</b>
	<b>Response headers:</b> connection: keep-alive content-type: application/json date: Sun, 15 Feb 2026 15:02:08 GMT keep-alive: timeout=60 transfer-encoding: chunked

## GET /products/{id} – Retrieve product by ID

GET /api/products/{id}

Parameters

Name	Description
id <small>required</small>	integer(\$int64) (path)

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/api/products/1' \
-H 'accept: */*'
```

Request URL

<http://localhost:8080/api/products/1>

Server response

Code Details

200 Response body

```
{
  "name": "Laptop",
  "price": 200000,
  "id": 1
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Sun,15 Feb 2026 15:04:19 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

## DELETE /products/{id} – Delete product

DELETE /api/products/{id}

Parameters

Name	Description
id <small>required</small>	integer(\$int64) (path)

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:8080/api/products/5' \
-H 'accept: */*'
```

Request URL

<http://localhost:8080/api/products/5>

Server response

Code Details

200 Response headers

```
connection: keep-alive
content-length: 0
date: Sun,15 Feb 2026 15:07:09 GMT
keep-alive: timeout=60
```

Responses

Code	Description	Links
200	OK	No links