CONSULTADD ASSIGNMENT
SPRING BOOT APPLICATION

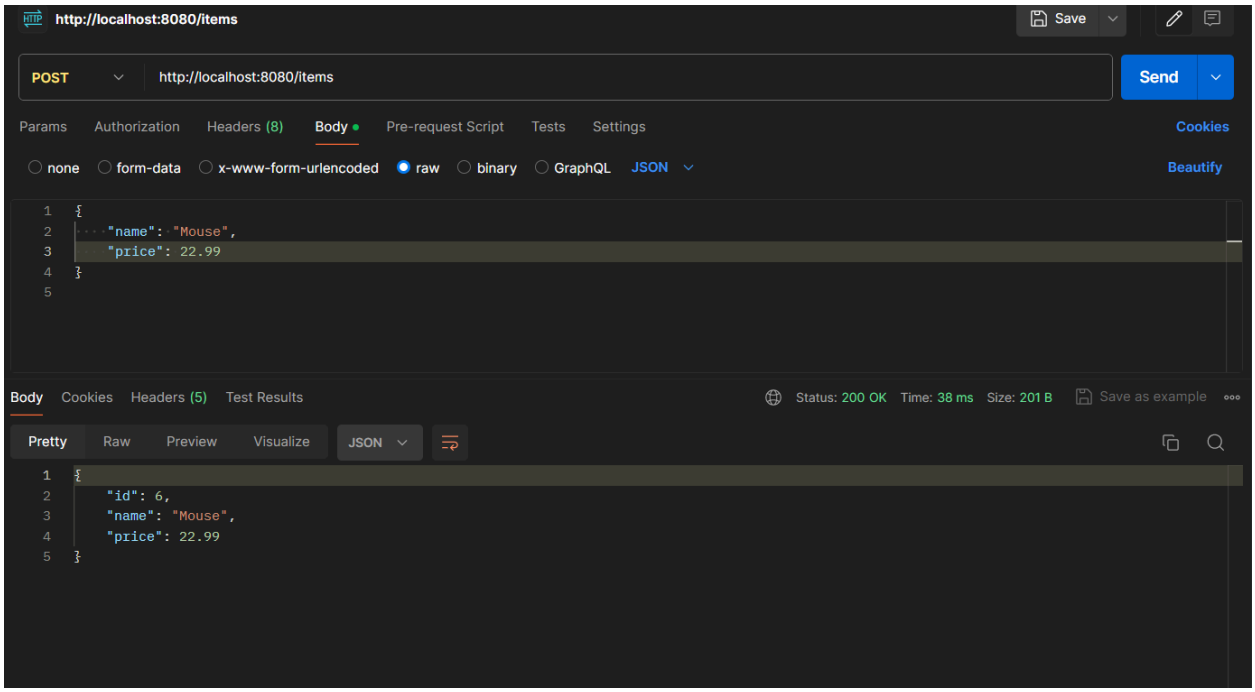
API Documentation:

1. **Add a new item**
   URL: /items
   HTTP Method: POST
   Request Body:

```
{
   "name": "Mouse",
   "price": 22.99
}
```

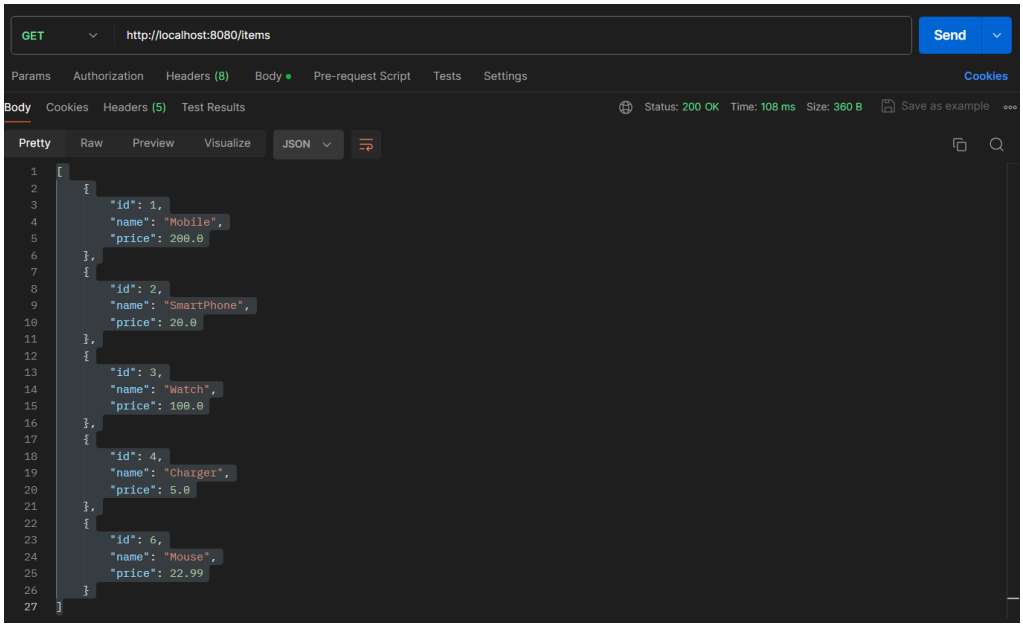
   Response:

```
{
   "id": 6,
   "name": "Mouse",
   "price": 22.99
}
```



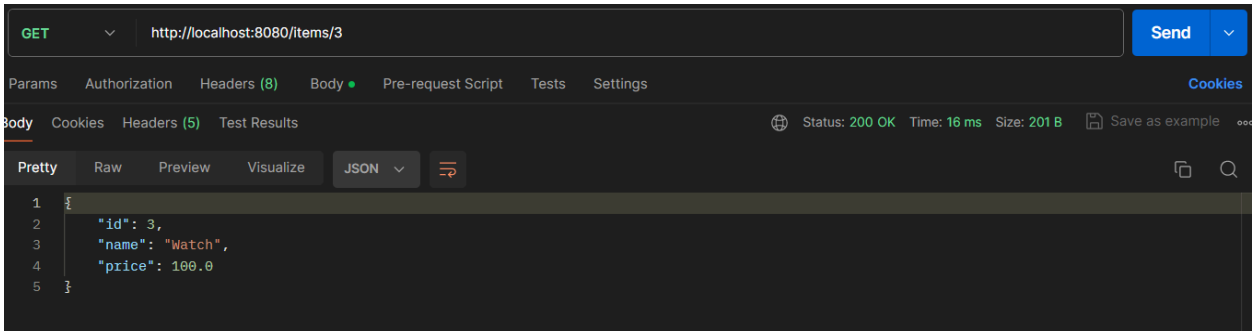
2. **Get all items**
   URL: /items
   HTTP Method: GET
   Request Body: NA
   Response:
   [

```
    {
        "id": 1,
        "name": "Mobile",
        "price": 200.0
    },
    {
        "id": 2,
        "name": "SmartPhone",
        "price": 20.0
    },
    {
        "id": 3,
        "name": "Watch",
        "price": 100.0
    },
    {
        "id": 4,
        "name": "Charger",
        "price": 5.0
    },
    {
        "id": 6,
        "name": "Mouse",
        "price": 22.99
    }
]
```

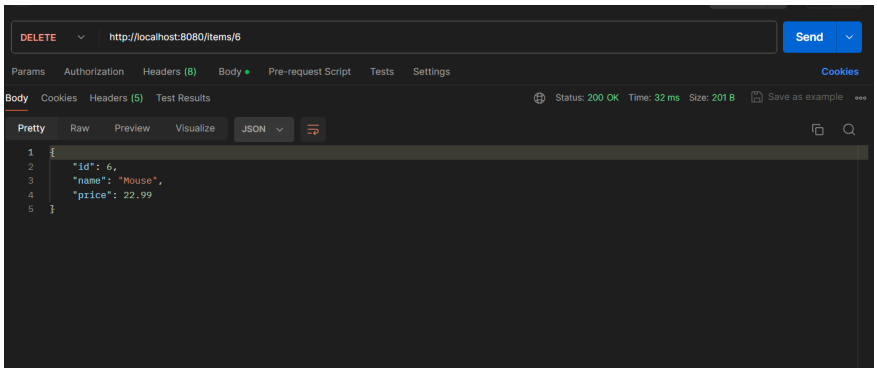
3. **Get an item by ID**
   URL: /items/{id}
   HTTP Method: GET
   Request Body: NA, we give path variable id value as 1 or etc
   Response:

```
{
   "id": 3,
   "name": "Watch",
   "price": 100.0
}
```

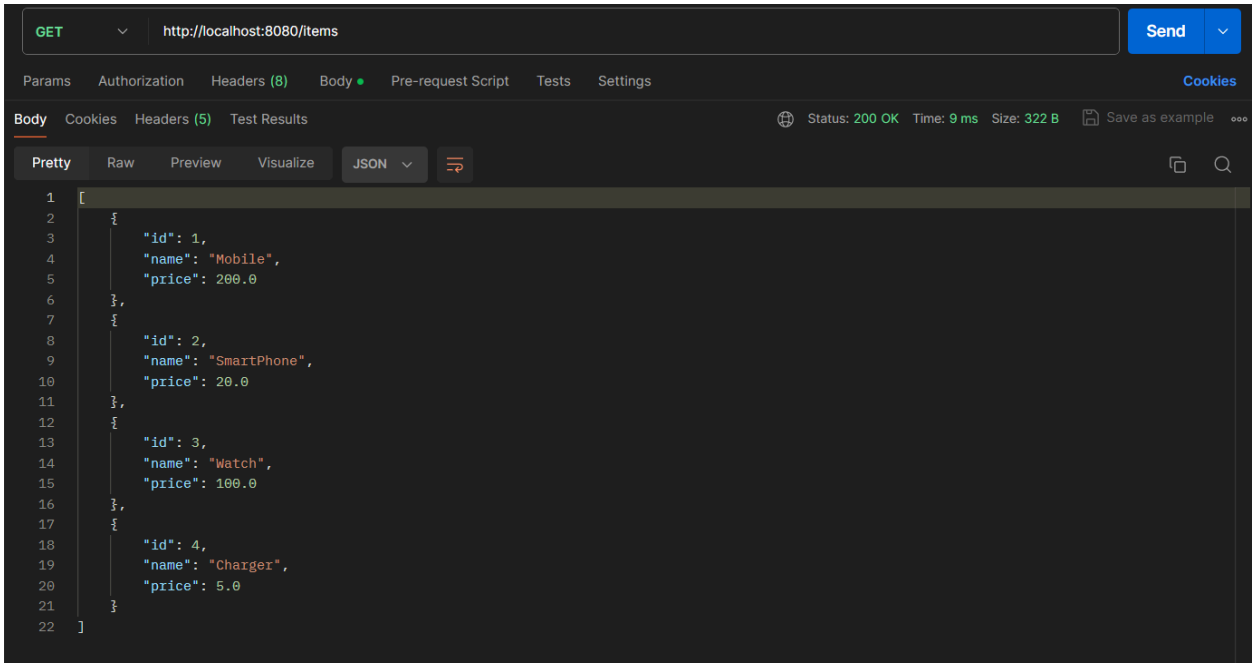


4. **Delete an item by ID**
   URL: /items/{id}
   HTTP Method: DELETE
   Request Body: NA, we give path variable id value as 1 or etc
   Response:

```
{
   "id": 6,
   "name": "Mouse",
   "price": 22.99
}
```



After deleting, the below updated list of items:

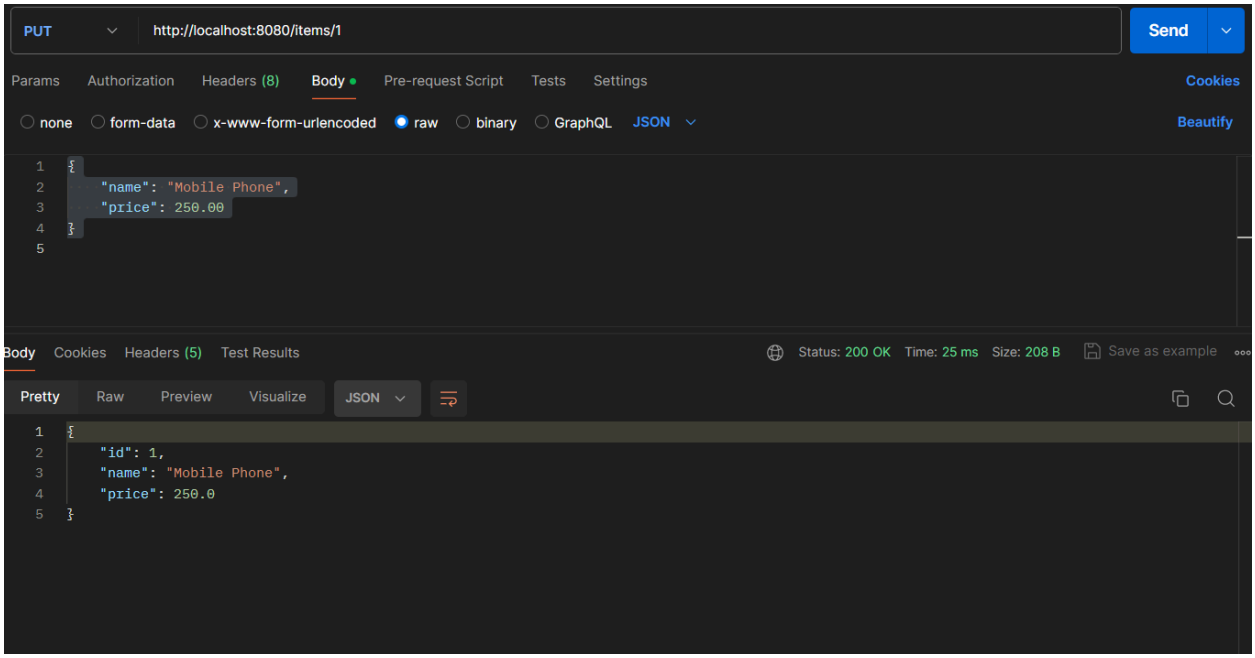


5. **Update an item**
   URL: /items/{id}
   HTTP Method: PUT
   Request Body: We give path variable id value as 1 or etc

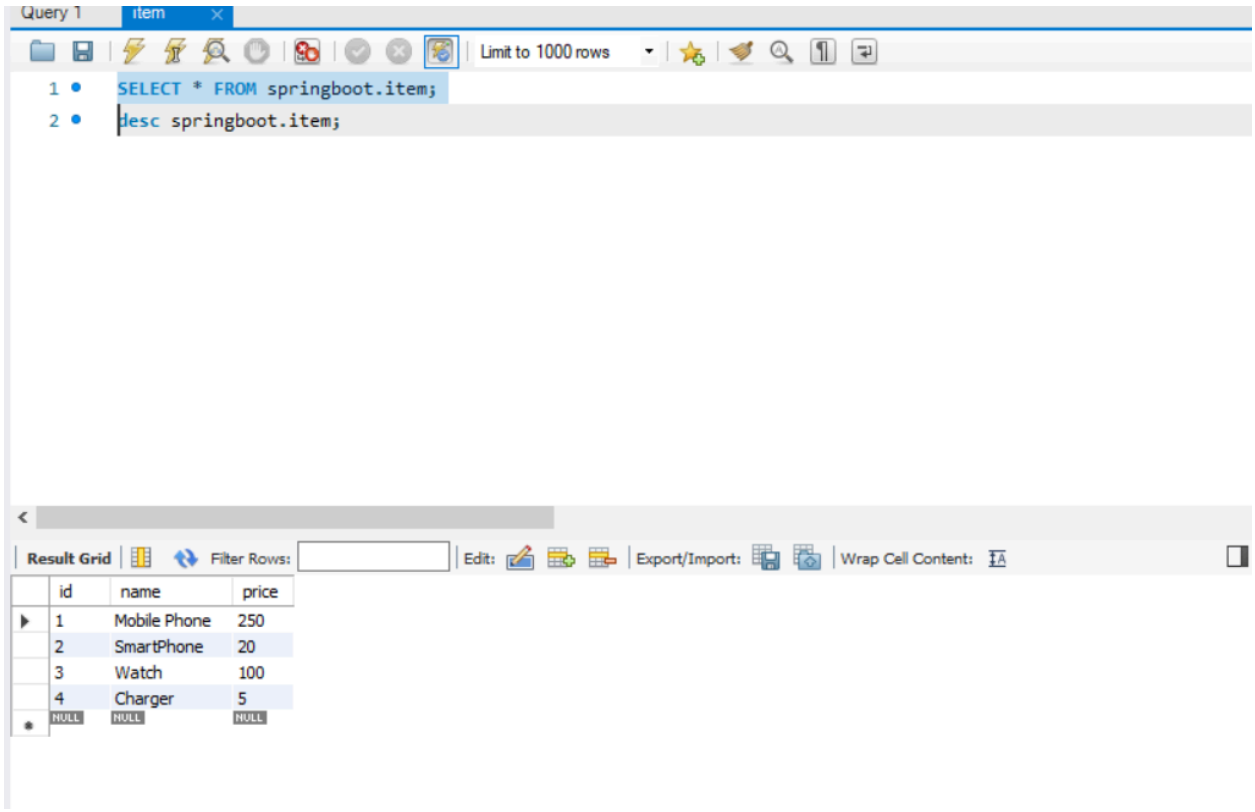
```
{
    "name": "Mobile Phone",
    "price": 250.00
}
```

   Response:

```
{
    "id": 1,
    "name": "Mobile Phone",
    "price": 250.0
}
```



Final Database output after performing all CRUD operations:



SetUp Instructions:

a. Prerequisites:
   - Java version downloaded
   - Configure the MySQL server
   - IntelliJ or Eclipse

b. Clone the git repository
   git clone <repo_url>

c. Dependencies used for this spring boot application:
   - Spring Web: used for RESTful web applications; Tomcat is the default server.
   - Spring Data JPA: Java Persistence API for persist data in SQL
   - Lombok: for getter, setter methods, constructor. This is used to reduce the boiler plate code.
   - MySQL Driver: connect with JDBC drive

d. Update the application.properties file with the SQL username, password and the database, which is used for creating tables and querying the data.

e. Create a model class annotated with @Entity to define the fields of the database table and include annotations like @Id and @Column in the project.

f. Create a repository interface extending JPARepository to handle database operations like save, findAll, etc.

g. Create a service class with the @Service annotation that contains the logic of the application.

h. Create a controller class with the @RestController that is responsible for handling the HTTP requests and map with the endpoints using @GetMapping, @PostMapping, etc.

i. Finally, run the main method and test all the endpoints working fine for the URL given and parallley verify the database whether the endpoints data, which is the output, is updating the table or not.

Usage Examples:

These are added in the API documentation section explaining the request and response received after performing the CRUD operations.