To set up and run the `ExProductService` application, follow these steps.

1. Add Security Dependency to `pom.xml`

The provided `pom.xml` file requires the Spring Security dependency to enable the security configuration in `securityconfig.java`. You should add the following dependency to your `pom.xml` file inside the `<dependencies>` block:

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

After adding the dependency, your Maven project will be able to utilize the `SecurityFilterChain` and `UserDetailsService` beans defined in the `securityconfig.java` file.

---

2. Add Configuration file

Add Config Package file. The configuration file can be added in the Package Config as in the name of securityconfig and add the required beans as in the below example file.

https://github.com/kannanmano6cfs/SBAug2025/blob/main/ExProductService/src/main/java/com/learning/exproductservice/Config/securityconfig.java

---

3. Review the API Endpoints and Security Rules

The `ProductController.java` file defines the API endpoints. The `securityconfig.java` file enforces access control based on user roles and endpoint patterns. Here is a breakdown of the endpoints and their access requirements:

- **`GET /`**: Accessible to any authenticated user. Returns a welcome message.
- **`POST /new/v1`**: Accessible only to users with the `ADMIN` role. This endpoint adds a new product with hardcoded details to the database.
- **`POST /new/v2`**: Accessible only to users with the `ADMIN` role. This endpoint adds a new product using a JSON request body. The request body should be in the format of the `Product.java` class, with `prdname`, `prdprice`, and `prddesc` fields.
- **`GET /count`**: Accessible to any authenticated user. Returns the total number of products in the database.
- **`GET /products`**: Accessible to users with either the `ADMIN` or `MEMBER` role. Returns a list of all products in the database.
- **`GET /product/v1/{id}`**: Accessible to users with the `ADMIN`, `MEMBER`, or `USER` role. Returns a specific product by its ID. The ID is passed as a path variable.
- **`GET /product/v2`**: Accessible to users with the `ADMIN`, `MEMBER`, or `USER` role. Returns a specific product by its ID. The ID is passed as a request parameter.

- **`GET /product/v3`**: Accessible to users with the `ADMIN`, `MEMBER`, or `USER` role. Returns a list of products whose name contains the search string. The search string is passed as a request parameter.

---

4. User Credentials

The `securityconfig.java` file defines three in-memory users with specific roles. You will need to use these credentials for authentication:

- **Username**: `user` | **Password**: `user` | **Role**: `USER`
- **Username**: `admin` | **Password**: `admin` | **Role**: `ADMIN`
- **Username**: `member` | **Password**: `member` | **Role**: `MEMBER`

---

5. Run and Test the Application

After adding the security dependency, you can run the application. You must provide a username and password for each request, as the application uses HTTP Basic authentication. For example, to access `/products` as a `MEMBER`, you would use the username `member` and password `member`.