# Overview

Product Application has been written with the help of Spring Boot Starter project. I am using JPA with in memory h2 database. It is built on java 11 and maven 4. The data is loaded from data.csv

# Endpoints and their information

As per requirement there is 1 endpoint for my application which are mentioned as below. I am using request body which I think better from security point of view and there is no url length limit The url is Get-<http://localhost:8080/product> and it support request in json only

|  |  |
| --- | --- |
| Example of Request | Detail |
| {      "type": "phone",      "min\_Price": 100,      "max\_Price": 500,      "city": "Stockholm",      "property": "color",      "color": "silver",      "gbLimitMin": 10,      "gbLimitMax": 50  } | “type” – “phone” or “subscription”  “min\_Price” -Minimum price of products which needs to be searched. Only number is allowed  “max\_Price” -Maximum price of products which needs to be searched. Only number is allowed  “city” -City where Store is present.  “property”- color or gb\_limit as selection  “color” to select color of phone product  “gbLimitMin” – Minimum gb limit for subscription product  “gbLimitMax” – Maximum gb limit for subscription product |

There are many validations in system including normal validation, hibernate validations to validate the models

# Testing

I am junit Jupiter together with springbootstartertest and mockito to test my application. There are around **50** testcase which checks all different aspects including basic, negative, exception etc. Currently Unit Testing coverage currently stands at approx. 85%. Here is the attached report



# Some Facts About Application

1. Global Exception handler has been used to send the error back gracefully
2. Transactions are marked with default Read Committed isolation level. This will handle concurrent transactions and will give good performance. The transactions which are read only are marked with @Transactional(readOnly = **true**)
3. The code is written by following Clean Code Principles and many other principles such as SOLID, DRY,DIE etc.
4. There are many design patterns used in building this functionality.
5. Sonarlint and checkstyle are taken care

# How to run the Application

As mentioned no additional configuration is required to run the application

## Build the Application

You can build the project by using maven. Below are the steps

* 1. Unzip the attached project
  2. Go to the location of src folder where project is unzipped
  3. Run the command (ensure mvn is available)

mvn clean install

## Run the Application

To run the application, there are below 2 ways

1. Using mvn spring boot
   1. Unzip the attached project
   2. Go to the location of where project is unzipped
   3. Run the command (ensure mvn is available)

mvn spring-boot:run

1. Build the project and run the executable jar
   1. Unzip the attached project
   2. Go to the location of where project is unzipped
   3. Build the project by running below command

mvn clean install

* 1. Go to the target folder and run below to start the application

java -jar products-0.0.1-SNAPSHOT.jar

## Run the tests

Test will be running only with separate profile named as testProfile. Below are the steps

1. Unzip the attached project
2. Go to the location of where project is unzipped
3. Run the test by below command

mvn test -PtestProfile

Tests will not be run in other profiles, so this makes build and deployment faster

## Containerize the application

Please follow below steps

1. Go to src folder where the project is unzipped.
2. Build the project
3. There is DockerFile which will be used for building the image
4. Ensure docker is already installed, if not install it
5. Build the image by opening cmd from src folder and running below command

docker build -t productapp .

1. Run the container by opening cmd from src folder

docker run -p 8080:8080 -t productapp