Table of Contents

Project description

This project provides REST APIs for basic CRUD operations on customer reviews and consumes an external web-service for product information in high level it demonstrates below features

1. New customers can signup in the system.
2. Existing customers can login to submit reviews.
3. Logged in customers can submit new reviews for any particular product.
4. Update existing reviews posted by self.
5. Deletes reviews posted by self.
6. Fetch review's summary for any particular product giving (average review score and total reviews)
7. Fetch all reviews for a particular product

Technology used

- Java 11 (Programming language)
- Spring boot v2.5.3 (Application development framework)
- Maven - v3.8.1 (Dependency and build management)
- Security - Spring security with JWT tokens.
- Encryption - BCryptPasswordEncoder for encrypting passwords
- H2 in memory DB
- Lombok - v1.18.20 (Annotation based code generation)
- Swagger v3.0.0 - (REST API documentation)
- ModelMapper (Objects Mapping)
- Spring boot devtools (Development ease)
- Junit 5 (unit testing)
- Docker (deployment)

Security

For this project i have used JWT (json web token) based security and leveraged spring security features to protect and secure writable APIs

Just to explain security feature at high level

1. Users who needs to submit / update / delete reviews need to first signup in the system using username , password and email.
2. once signed up user can use username and password to login.
3. If login is successful in response my API will return a token currently valid for 10hrs (configurable) that token can be passed in subsequent requests to use writable APIs.

Encryption

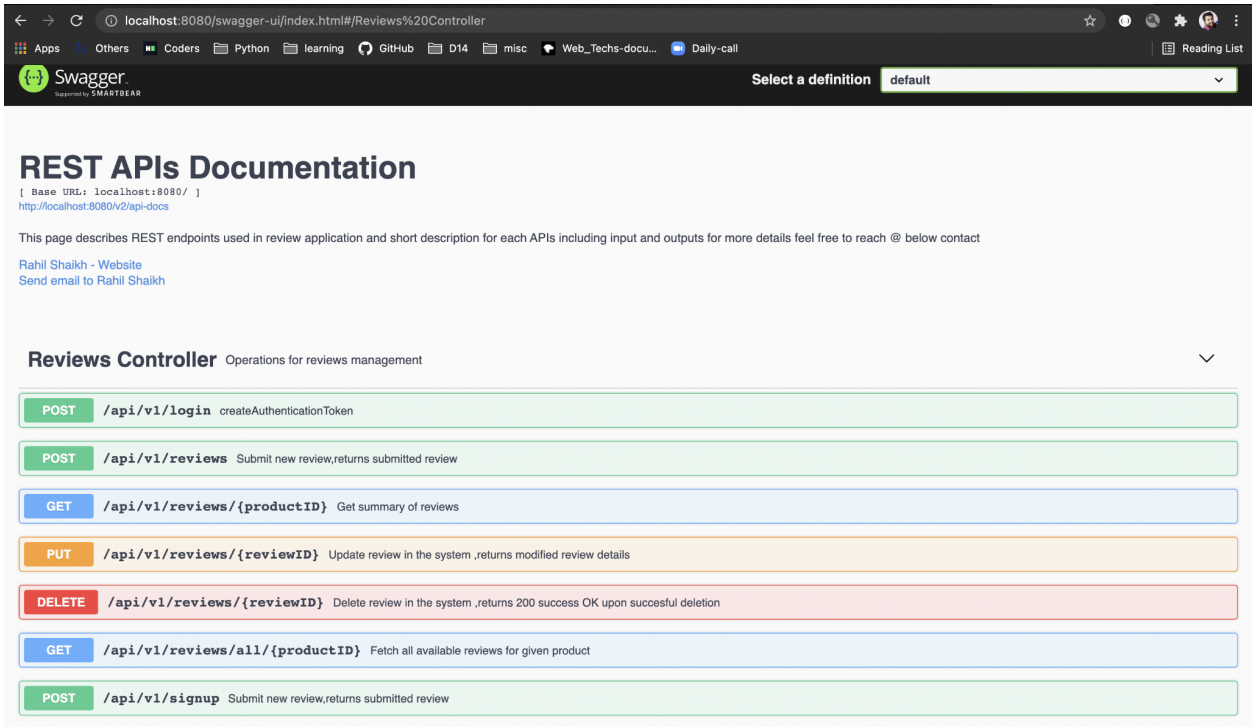
Currently i am encrypting user's passwords before saving to DB using `BCryptPasswordEncoder` which uses strong hashing function.

Exception handling

I am following Global Exception handling mechanism in class GlobalExceptionHandler
It provides a centralized Exception and error handling mechanism and provides graceful failure and response handling.

API documentation

I am using Swagger which follows open API specification for REST documentation
it can be accessed at URL http://{HOST}:{PORT}/swagger-ui.html
(sample screenshot attached below)
apart from this code level documentation wherever needed is is done in javadoc comments



Database

Current i am using H2 in memory DB and seeding some test users and products using Data.sql and schema.sql at application startup

Logging

For logging i am using Slf4J and log configuration is kept in **logback-spring.xml** file
As per current configuration it generates logs folder under current directory
And rolling policy is daily or 10mb file size , rolling is done under archived folder
(above configurations are changeable as per need)

Testing

Unit testing

For unit testing i am using Junit5 and mockito .All my tests are located under folder **src/test/java**
(note : due to time constraint i couldn't cover much test cases but just demonstrated basics)

Execution steps

For execution below softwares and tools are needed i included installation instructions for each of those

1. **Java 11 (required)**
   Installation instructions :
   MacOS : https://dzone.com/articles/installing-openjdk-11-on-macos
   Windows : https://www.openlogic.com/openjdk-downloads
   Linux : https://openjdk.java.net/install/

2. **Maven 3.8.1 (optional)**
   This is required only if you want to build the code
   Installation instructions :
   MacOS : https://maven.apache.org/install.html
   Windows :https://maven.apache.org/download.cgi
   Linux : https://maven.apache.org/download.cgi

3. **Postman REST client (optional)**
   This is optional but recommended for ease of use
   Installation instructions :
   MacOS / Windows / Linux : https://www.postman.com/downloads/

Step 2 :

Once above softwares are installed please download the following jars

1. product-service.jar
2. review-service.jar

Download-location

https://github.com/mrshaikh4u/product-reviewAPIs-SpringBoot/tree/master/executable-jars

Step 3 :

Open 2 terminal tabs (mac / Linux)

Open 2 command prompts (windows)

Navigate to location where you downloaded jars from step 2

Tab 1 : run command > java -jar product-service.jar

Tab 2 : run command > java -jar review-service.jar

Now your 2 services are up and running you can expect to see below messages in each tab

Step 3:

Please download postman collection from below location

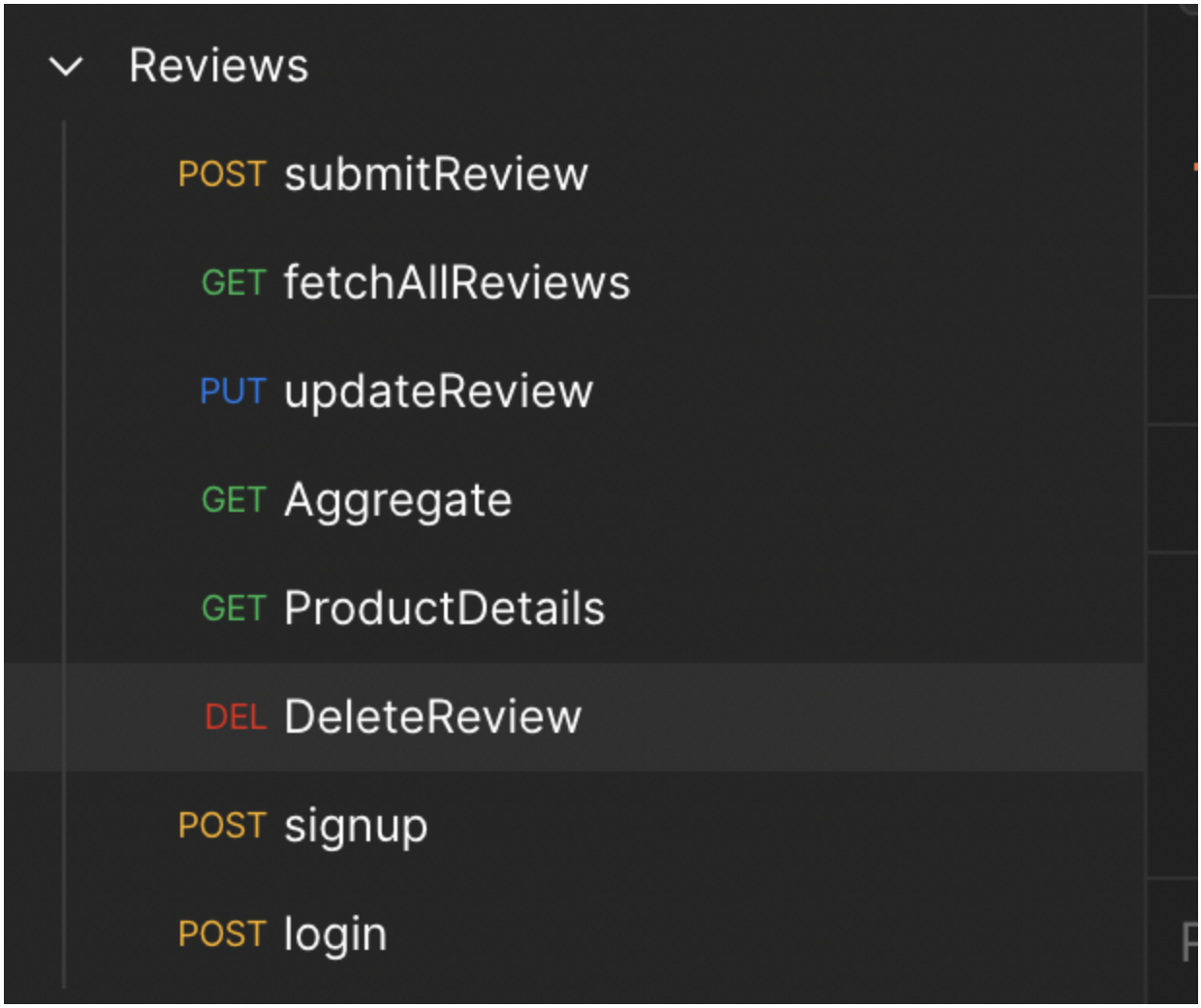
https://www.getpostman.com/collections/3c46397e99310af5062f

Step 4:

Import postman collection downloaded in step 3

into postman client from file->import

After import you can expect to see as below screenshot in left pan of postman



For simplicity you can set postman environment variables as

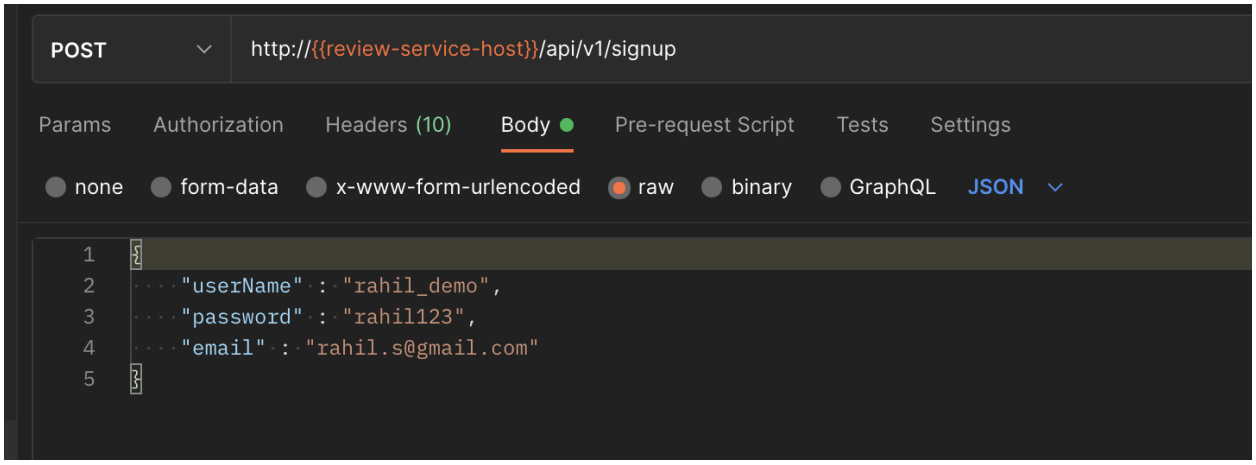
ReviewsAppENV Edit

VARIABLE	INITIAL VALUE	CURRENT VALUE
review-service-host	localhost:8080	localhost:8080
product-service-host	localhost:8081	localhost:8081

API testing :

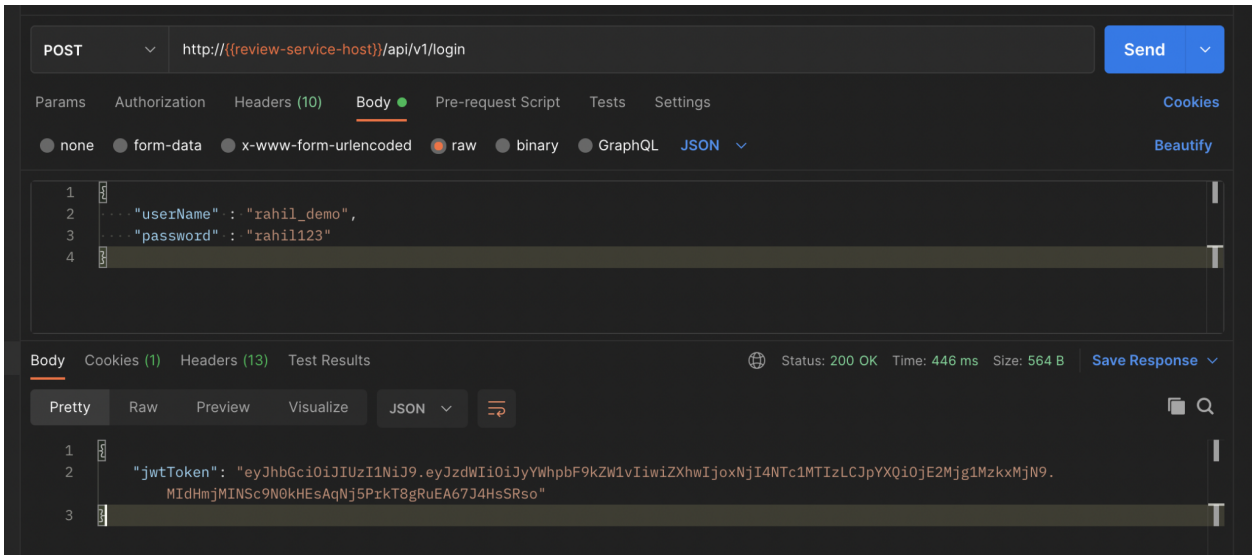
SignUp :

Run signup API as shown in screen shot



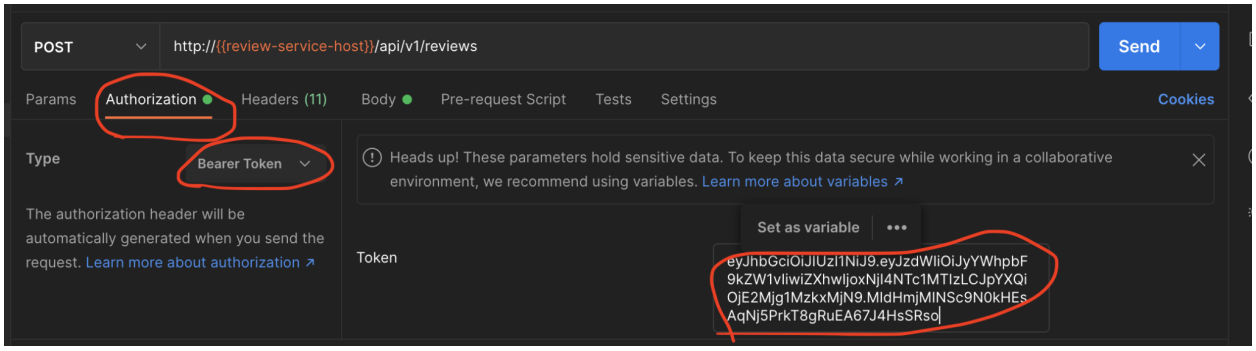
Login:

Run login API as shown in picture below



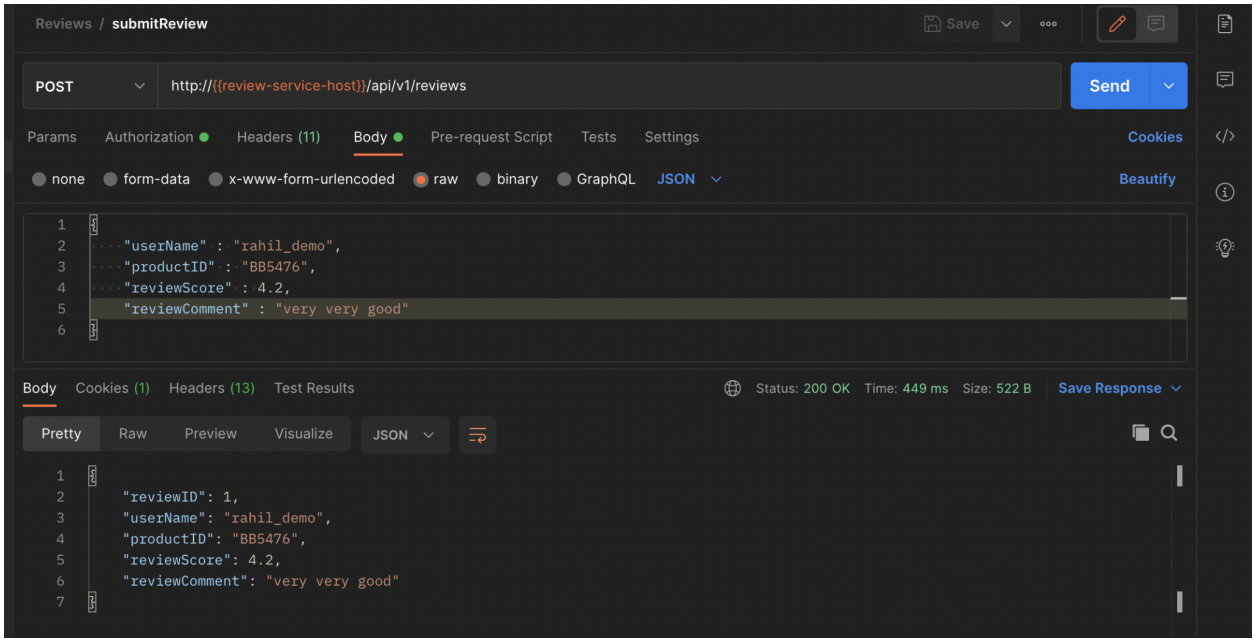
Copy the token from response

And put into submitReview API like below



SubmitReview :

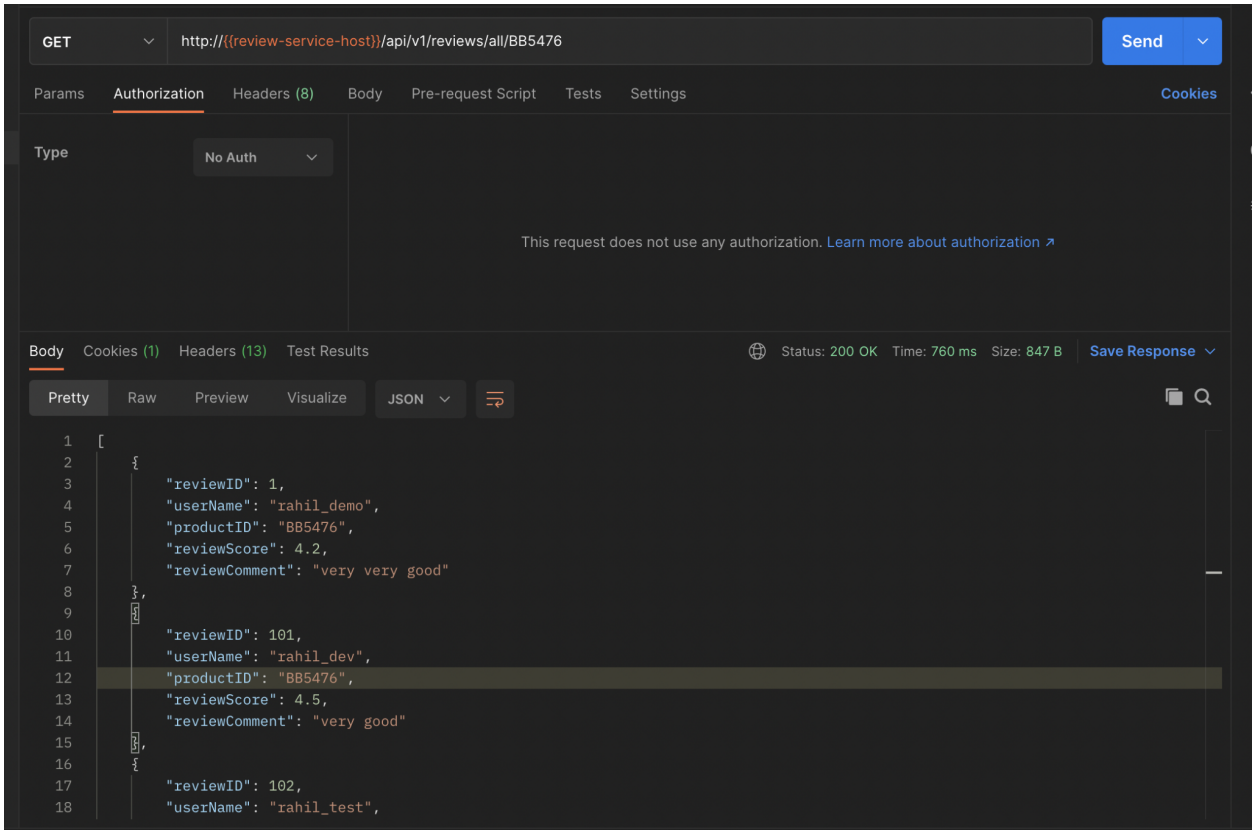
Run submitReview API as per below



FetchAllReviews :

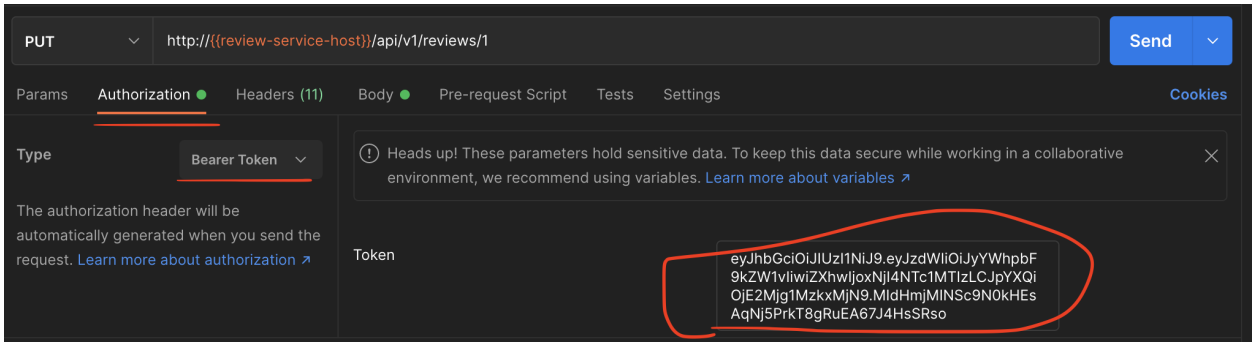
As this is GET API no token is needed for this as this is open for all users

Run as per below screen shot

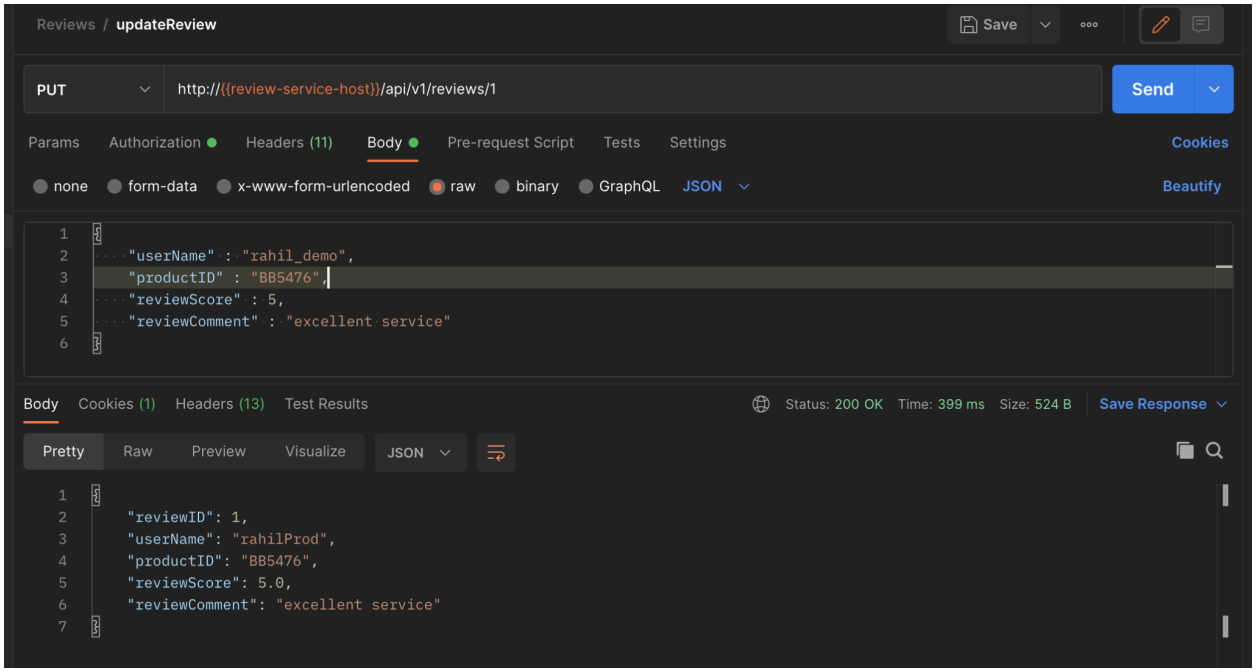


Update Review :

Put token as shown below

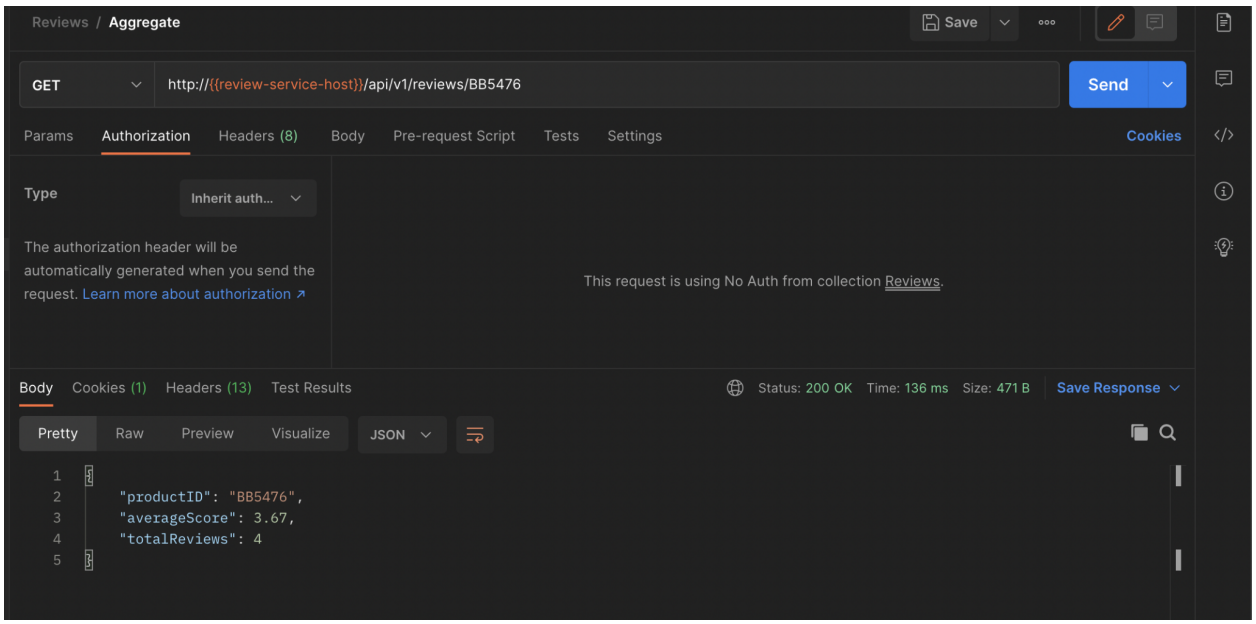


Run API as per below



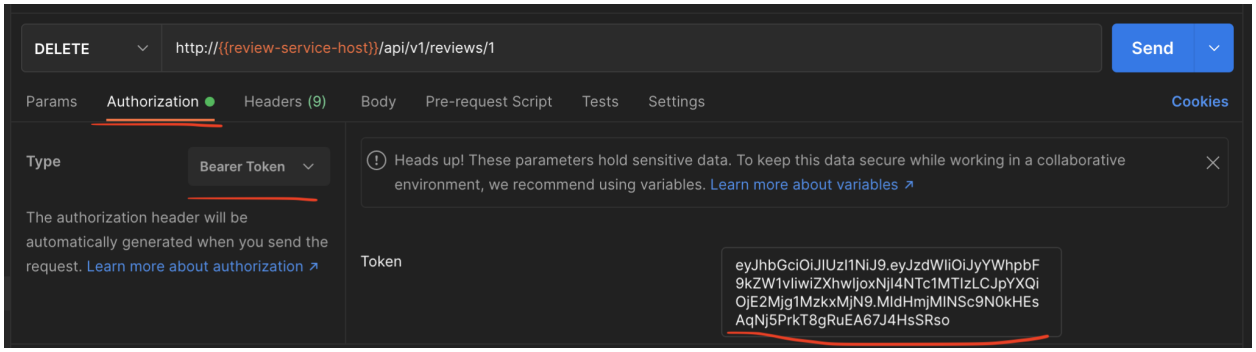
Aggregate :

Again this is GET API open for all so no token is needed , run as per below screenshot



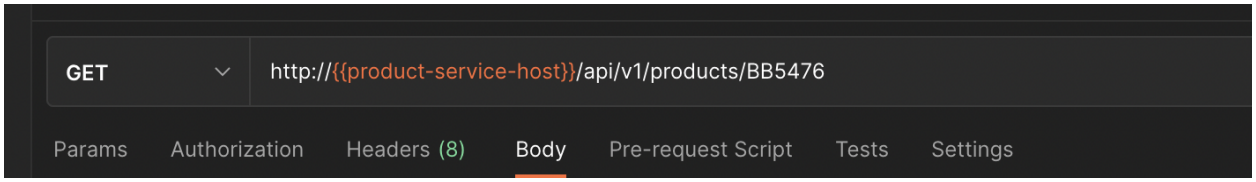
Delete :

Put token as per below and run the API



<u>ProductDetails :</u>

Run the API as per below



Docker and Containerization

I started working on it but due to time constraint i couldn’t implement so i will continue working on it and will push update in Git