eCommerce On Cloud

(using Spring Boot and AWS)

A PROJECT REPORT

Submitted in partial fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

by

SURUCHI SINHA

(Univ. Roll No.: 1900290140040)

Under the Supervision

Of

Ms. Vidushi

Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

KIET Group of Institutions,

Delhi-NCR, Ghaziabad

Uttar Pradesh (201206)

DR. A.P. J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW (JUNE ,2022)

DECLARATION

I hereby declare that the work presented in this report entitled "eCommerce On Cloud (using Spring Boot and AWS)", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Suruchi Sinha

Roll.No.:1900290140040

Branch: MCA

(Candidate Signature)

TRAINING CERTFICATE

Capgemini Technology Services India Limited
Capgemini Knowledge Park SEZ, IT3/IT4
Airoli Knowledge Park, Thane - Belapur Road
Airoli, Navi Mumbai - 400 708, Maharashtra, India
Tel.: +91 22 7144 4283 | Fax: +91 22 7141 2121
E: cgcompanysecretary.in@capgemini.com
www.capgemini.com/in-en

May 12, 2022 **Capgemini/HR/AG**

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Suruchi Sinha** has successfully completed internship remotely on **JEE with DevOps & Cloud(AWS)** from **2/4/2022** to **5/6/2022.**

Suruchi Sinha interned under the guidance of **Anjulata Tembhare.** The performance was found to be satisfactory.

We wish **Suruchi Sinha** all the best for all future endeavors.

Yours Truly,
For **Capgemini Technology Services India Limited**

Arunkumar Gopalakrishnan
Senior Director – Human Resource

CERTIFICATE

Certified that **Suruchi Sinha** **(**enrollment no 1900290140040) has carried out the project work presented in this thesis entitled **"eCommerce On Cloud (using Spring Boot and AWS)"** for the award of **Master of Computer Application** from Dr. APJ Abdul Kalam Technical University, Lucknow under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not formthe basis for the award of any other degree to the candidate or to anybody else from this or any other University.

Ms. Vidushi **External Examiner**

(Internal Examiner)

Assistant Professor

Dept. of Computer Applications

KIET Group of Institutions,

Delhi-NCR, Ghaziabad

Date:

ABSTRACT

eCommerce on Cloud is an application for ecommerce that is designed using APIs and various micro services. It deals with product management for their database maintaining and generating report corresponding to the data is done on the basis of as per requirement is given.

So, we can say that it helps the management of product and give exact database management of compony according to rules and regulation. It also helps in maintaining product data and also display how many products are present in the ecommerce and also gives the details of those products.

This application is designed to manage ecommerce products for companies and organizations and also handle the sale and purchase of their products. The eCommerce on Cloud includes different modules and features for adding, editing, viewing, and deleting items.

Usually, the management of products are done on local system. But, here apart from local machine, it is also deployed on AWS based Cloud by creating the same database with the help of PostgreSQL database on RDS (one of the major services provided by AWS).

For Deployment purpose, we need to create a PostgreSQL database on RDS by creating username and password. Thereafter, we need to open pgAdmin 4 and login with the same credentials created before to connect the pgAdmin 4 with the RDS. Then we can create database followed by creating the table and giving the required records in the table.
We can then apply this link in the **application.properties** file of the SpringBoot Application to it connect with AWS.

Now, we are all done and is good to go with the application by running it on the browser using the localhost.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Ms. Vidushi** for her guidance, help and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Ajay Kumar Shrivastava**, Professorand Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot in many critical conditions.

Finally, my sincere thanks to my family members and all those who have directly and indirectly provided me moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Suruchi Sinha

Enrollment no-1900290140040

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. API – Application Programming Interface
2. CRUD – Create, Read, Update, Delete
3. IDE – Integrated Development Environment
4. OS – Operating System
5. HTTP – Hyper Text Transfer Protocol
6. URL – Uniform Resource Locator
7. RAM – Random Access Memory
8. MySQL – My Structured Query Language
9. POM – Project Object Model
10. JPA – Java Persistence API
11. K8s – Kubernetes
12. AWS – Amazon Web Services
13. RDS – Relational Database Service
14. IT – Information Technology
15. REST – Representational State Transfer
16. SOAP – Simple Object Access Protocol
17. JSON – JavaScript Object Notation

CHAPTER 1
INTRODUCTION

1.1 PROJECT DESCRIPION

In today's changing life style computer has become the most essential part of life. Most of the works being performed by the humans is now done by the computer The computer is being used in each and every field now a days.

I am developing software for ecommerce that deals with product management for their database maintaining and generating report corresponding to the data is done on the basis of as per requirement is given.

So, we can say that it helps the management of product and give exact database management of company according to rules and regulation. It also helps in maintaining product data and also display how many products are present in the ecommerce and also gives the details of those products.

This software also gives or stores each and every information about orders.
This company uses a huge data base so for security of database we give the facility of backup and also recovery as per when company need it takes backup on hard disk.

This application is designed to manage ecommerce products for companies and organizations and also handle the sale and purchase of their products. The eCommerce on Cloud includes different modules and features for adding, editing, viewing, and deleting items.

Usually, the management of products are done on local system. But, here apart from local machine, it is also deployed on AWS based Cloud by creating the same database with the help of PostgreSQL database on RDS (one of the major services provided by AWS).

1.2 PROJECT SCOPE

The purpose of this project is to develop an API using various microservices. The objective of this process is as follows:
eCommerce on Cloud is designed for the companies to easy maintain of their records and easy deal with the product**.**

Usually, the manual eCommerce run with pen and paper is not only labor-intensive but also time-consuming. This approach lacks a proper data organization structure, which can give rise to many risks associated with data mismanagement. This eCommerce project is a more efficient and improved approach to product management. It is much more secure and reliable than the manual method.

This software also gives or stores each and every information about orders. This company uses a huge data base so for security of database we give the facility of backup and also recovery as per when company need it takes backup on hard disk. It includes the purchasing of product by customer, the customer can keep track of the records of the product.

There are 3 actors in the project which are listed below:

1. Admin
2. Customer
3. Supplier

There are 5 Microservices in the project which are listed below:

1. Product module
2. Supplied Product module

3. Order Module
4. Report Module
5. Delivery Module

1.3 SOFTWARE/TOOLS

While developing the eCommerce On Cloud application I have used various software.

1.3.1 ECLIPSE

Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. It is the second-most-popular IDE for Java development.

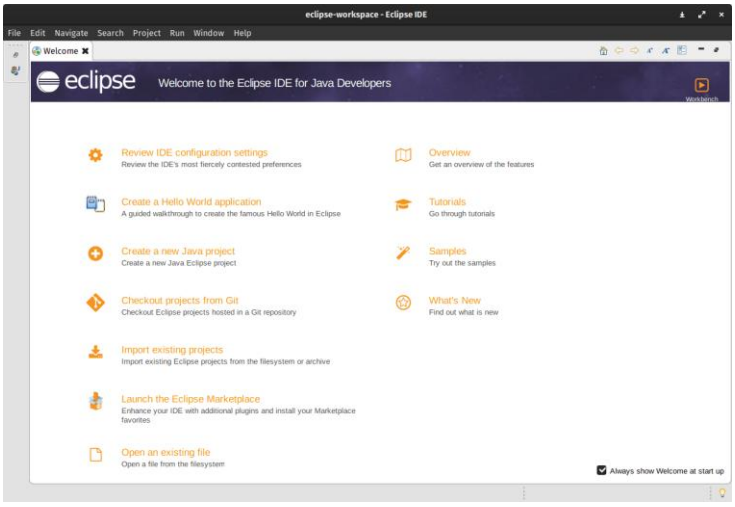


Figure 1.1(Eclipse IDE)

1.3.2 GRADLE

Gradle is a build automation tool known for its flexibility to build software. A build automation tool is used to automate the creation of applications. The building process includes compiling, linking, and packaging the code. The process becomes more consistent with the help of build automation tools.

Figure 1.2(Gradle)

Creating a simple addition application using Gradle

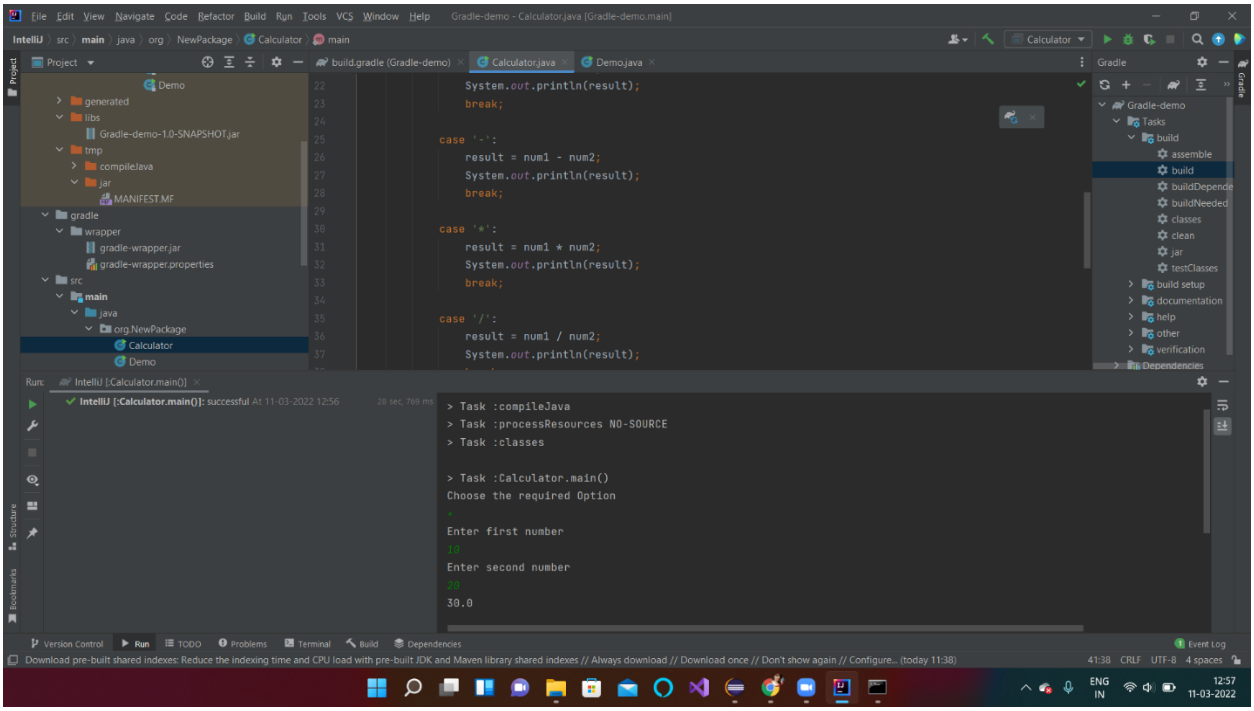


Figure 1.3(Application using Gradle)

Now accessing this on command prompt using path of the file

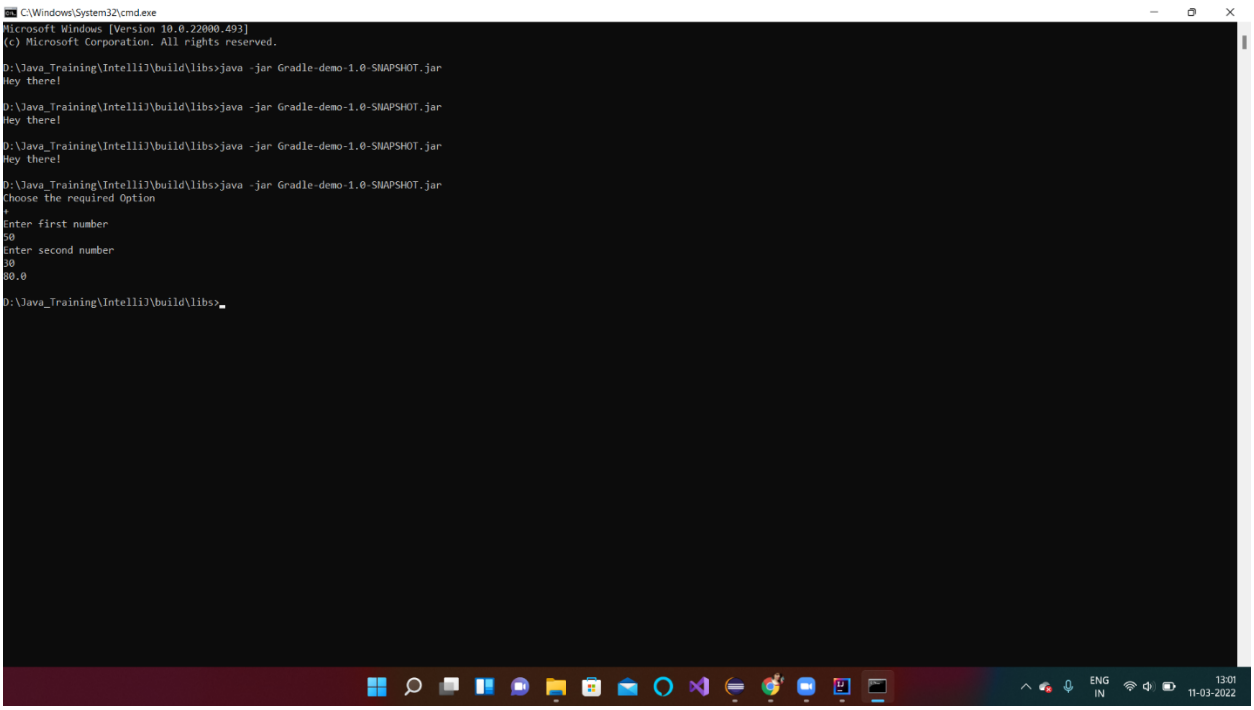


Figure 1.4(Accessing Application on Cmd)

1.3.3 POSTGRESQL

PostgreSQL is a powerful, opensource object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

Figure 1.5 (PostgreSQL)

Working on PostgreSQL

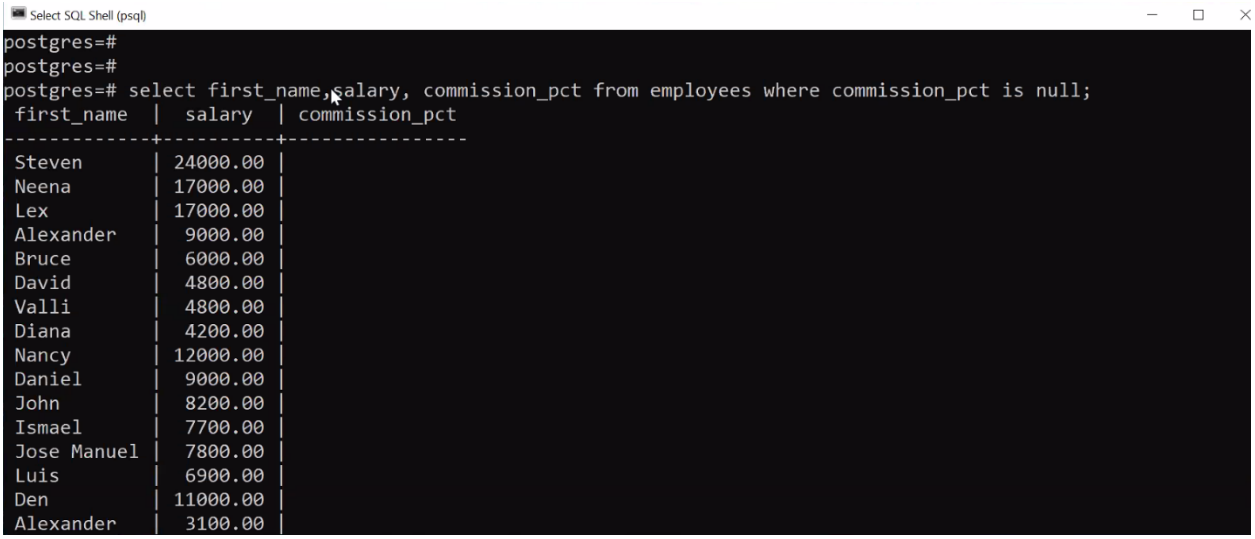


Figure 1.6(Working on PostgreSQL)

1.3.4 DOCKER

Docker is an opensource containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

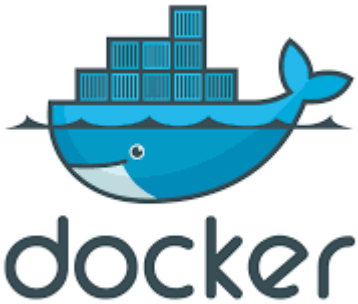


Figure 1.7(Docker)

1.3.5 KUBERNETES

Kubernetes is a portable, extensible, opensource platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

Figure 1.8(Kubernetes)

1.3.6 POSTMAN

Postman is an application used for **API testing**. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated.

Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

API Tools - A comprehensive set of tools that help accelerate the
API Lifecycle - from design, testing, documentation, and mocking to discovery.

CRUD OPERATIONS

CRUD Meaning: CRUD is an acronym that comes from the world of computer programming and refers to the four functions that are considered necessary to implement a persistent storage application: **create, read, update** and **delete**.

Persistent storage refers to any data storage device that retains power after the device is powered off, such as a hard disk or a solid-state drive.

GET: This method retrieves the information identified by the requested URL. The method used for read operations (the R in CRUD).

POST: Method used to instruct the server to accept the entity enclosed in the request as a new resource. The create operations typically mapped to this HTTP method.

PUT: This method requests the server to store the enclosed entity under the requested URL. As per the HTTP specification, the server can create the resource if the entity does not exist. It is up to the web service designer to decide whether this behavior should be implemented or resource creation should only be handled by POST requests.

DELETE: The last operation not yet mapped is for the deletion of resources. The HTTP specification defines a DELETE method.

Figure 1.9 (Postman)

1.4 HARDWARE

- **RAM** - Minimum 8 GB RAM is required
- **Processor -** Intel i3/i5
- **Operating System –** Window10/CentOS 7

1.5 GAINT CHART

Gantt Chart

When creating a project schedule, the planner begins with a set of tasks (the work breakdown structure). If automated tools are used, the work breakdown is input as a task network. Effort, duration and start dates are input are each task network. As a consequence of this input, a timeline chart also called a Gantt chart is generated. A timeline chart is developed for entire project.

Gantt chart for project:

Figure 1.10 Gantt chart

CHAPTER 2

LITERATURE REVIEW

[1] By using JAVA Programming approach relies on software for distributed batch and stream processing, as well as distributed storage. This chapter focuses on an oft-ignored angle of assuredness: performance assuredness. A significant pain point today is the inability to support reconfiguration operations, such as changing of the shard key in a shared storage/database system, or scaling up (or down) of the number of virtual machines (VMs) being used in a stream or batch processing system.

[2] By using Software Design approach discuss new techniques to support such reconfiguration operations in an online manner, whereby the system does not need to be shut down and the user/client-perceived behavior is indistinguishable regardless of whether a reconfiguration is occurring in the background, that is, the performance continues to be assured in spite of ongoing background reconfiguration. Next, we describe how to scale-out and scale-in (increase or decrease) the number of machines/VMs in cloud computing frameworks like distributed stream processing and distributed graph processing systems, again while offering assured performance to the customer in spite ofthe reconfigurations occurring in the background.

[3] By Using Fundamentals of Java Programming approach is the ability to support SLAs/SLOs (service-level agreements/objectives) such as deadlines. We present a new real-time scheduler that supports priorities and hard deadlines for Hadoop jobs. We implemented our reconfiguration systems as patches to several popular and open-source cloud computing systems, including MongoDB and Cassandra (storage), Storm (stream processing), LF Graph (graph processing), and Hadoop (batch processing).

[4] By Using Java in two semesters by approach has become the industry standard for rapid application deployment, scalable server support, mobile and distributed services, and it provides access to (theoretically) infinite resources. Unfortunately, researchers are still trying to converge

towards cross-provider cloud computing frameworks to enable compatibility and seamless resource transition between cloud providers. Moreover,users are restricted to using the provider-specific pre-configured options of resources and services, irrespective of their current needs. At the same time, cloud services are provided as a direct service from the providers to the clients. This creates a segregated cloud market clientele, and non-negotiable pricing

[5] By using Learning Java with Games In this paper, we propose Java, a generic architecture for cloud composition and negotiated service delivery for cloud users. Jugo acts as a match-maker for service specifications from the users with the currently available assets from the cloud providers. The engagement of a middle-man as an opaque cloud service provider will create a better opportunity for cloud users to find cheaper deals, price-matching, and flexible resource specifications, with increased revenue and higher resource utilization for the cloud service providers.

[6] By using Computing and Software Science approach Many enterprises in industries start using Cloud Computing for their IT infrastructure services. This adoption of Cloud Computing is a part of the enterprise transformation which is the migration from a legacy IT environment to Cloud Computing. On the other hand, one of major targets is an industry solution which provides a critical business service to their end customers. This paper proposesIndustry Cloud which is the enhanced design of Cloud Computing for industry solutions. It efficiently supports industry solutions for enterprise business requirements. The paper describes Industry Cloud with a requirement analysis of industry solutions, those adopted functions, and three use case scenarios in the electronics and retail industry. The contribution of the paper is the analysis of industry wide requirements,the definition of Industry Cloud with a common function among industry solutions and the usage with usecase scenarios.

[7] By using Springboot approach the complexity of Cloud infrastructures is increasing every year, requiring new concepts and tools to face off topics such as process configuration and reconfiguration, automatic scaling, elastic computing and healthiness control. This paper presents a Smart Cloud solution based on a Knowledge Base, KB, with the aim of modeling cloud

resources, Service Level Agreements and their evolution, while enabling the reasoning on cloud structures and implementing strategies of efficient smart cloud management and intelligence.

[8] By using CRUD approach, the solution proposed is composed of Smart Cloud Engine, SCE, the Knowledge Base, KB, and the Supervisor and Monitoring module for data acquisition. It can be easily integrated with any cloud configuration manager, cloud orchestra or, and monitoring tool, since the connections with these tools are performed by using REST calls.

[9] By using AWS approach, in addition to the huge number of dedicated servers deployed in data centers, there are billions of underutilized Personal Computers (PCs), usually used only for a few hours per day, owned by individuals and organizations worldwide. The vast untapped compute and storage capacities of the underutilized PCs can be consolidated as alternative cloud fabrics to provision broad cloud services, primarily infrastructure as a service

[10] By using SWAGGER approach to as "no data center" approach, complements the data center-based cloud provision model. In this paper, we present our opportunistic Cloud Computing system, called cu Cloud, that runs on scavenged resources of underutilized PCs within an organization/community. Our system demonstrates that the "no data center" solution indeed works. Besides proving our concept, model, and philosophy, our experimental results are highly encouraging.

[11] By using KUBERNETES approach Whatever one public cloud, private cloud or a mixed cloud, the users lack of effective security quantifiable evaluation methods to grasp the security situation of its own information infrastructure on the whole. This paper provides a quantifiable security evaluation system for different clouds that can be accessed by consistent API. The evaluation system includes security scanning engine, security recovery engine, security quantifiable evaluation model, visual display module and etc.

[12] By using DOCKER approach, The model composes of a set of evaluation elements corresponding different fields, such as computing, storage, network, maintenance, application security and etc. Each element is assigned a three tuple on vulnerabilities, score and repair method.

The system adopts “One vote vetoed” mechanism for one field to count its score and adds up the summary as the total score, and to create one security view.

[13] By using PostgreSQL database approach We implement the quantifiable evaluation for different cloud users based on our AWS platform. It shows the dynamic security scanning score for one or multiple clouds with visual graphs and guided usersto modify configuration, improve operation and repair vulnerabilities, so as to improve the security of their cloud resource.

[14] By using MySQL database approach, we move applications to the cloud is not only a technical decision but also a business-oriented decision, in which both business and technical factors (e.g., transformation effort multi-tenancy and auto-scaling enablement, scalability and extensibility) should be considered. However, existing approaches and tools do not support a consumable business-oriented cloud transformation decision to select more suitable transformation solution with the right cloud delivery model, services type, affordable transformation effort and etc.

[15] By using Gradle approach we introduce a practical three-step approach and a tool, CTA (Cloud Transformation Advisor) to enable decision makers to identify the most suitable cloud transformation solution to satisfy their business goals based on a well-structured cloud transformation knowledge base

CHAPTER 3
FEASBILITY STUDY

3.1 INTRODUCTION

Feasibility of the system in an important aspect, which is to be considered. The system needsto satisfy the law of economic, which states that the maximum output should be yielded in minimum available resources.

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study separate areas that a feasibility study examines, described below.

3.1.1 Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to putStar Trek's transporters in their building currently, this project is not technically feasible.

3.1.2. Economic Feasibility

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

3.1.3. Legal Feasibility

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility studymight reveal the organization's ideal location isn't zoned for that type of business.

That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

3.1.4. Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether and how well the organization's needs can be met by completing the project. Operational feasibility

studies also examine how a project plan satisfies the requirements identified
in therequirements analysis phase of system development.

i. Scheduling Feasibility

This assessment is the most important for project success; after all, a project will fail if notcompleted on time. In scheduling feasibility, an organization estimates how much time theproject will take to complete.

When these areas have all been examined, the feasibility analysis helps identify anyconstraints the proposed project may face, including:

- Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
- Internal Corporate Constraints: Financial, Marketing, Export, etc.
- External Constraints: Logistics, Environment, Laws, and Regulations, etc.

3.2 TECHNOLOGY DESCRIPTION

3.2.1 SPRING BOOT

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run.

You can get started with minimum configurations without the need for an entire Spring configuration setup.

Spring Boot automatically configures your application based on the dependencies you have added to the project by using @EnableAutoConfiguration annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class contains **@**SpringBootApplication annotation and the main method.

Spring Boot automatically scans all the components included in the project by using **@**ComponentScan annotation.

3.2.2 DOCKER

Docker is an opensource containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers simplify delivery of distributed applications, and have become increasingly popular as organizations shift to cloud-native development and hybrid multi-cloud environments.

Developers can create containers without Docker, but the platform makes it easier, simpler, and safer to build, deploy and manage containers. Docker is essentially a toolkit that enables developers to build, deploy, run, update, and stop containers using simple commands and work-saving automation through a single API.

Docker also refers to Docker, Inc. (link resides outside IBM), the company that sells the commercial version of Docker, and to the Docker open source project (link resides outside IBM), to which Docker, Inc. and many other organizations and individuals contribute.

Notable Features of Spring Boot

1. **Autoconfiguration:** Developers can automatically configure their Spring application. However, Spring Boot is also capable of changing the configuration based on the dependencies you list. For example, when you list “MySQL” as a dependency, it will configure your Spring application with the “MySQL connector” included. And if you want to add a custom configuration, you can create a class that overrides the default configuration for your “MySQL connector”.

2. **Standalone:** There's no need to deploy your application to a web server. You simply enter the run command to start the application.

3. **Opinionated:** On the official page, we find that Spring Boot decides for you which defaults to use for the configuration. Also, it decides which packages to install for the dependencies you require. For example, if you include the Spring Boot starter "pom" for "JPA", it will autoconfigure an in-memory database, a hibernate entity manager, and a simple data source. This is an example of an opinionated default configuration that you can override. While some developers might feel this is too opinionated, Spring Boot's opinionated setup helps developers to get started quickly on their projects.

3.2.3 KUBERNETES

Kubernetes is a portable, extensible, opensource platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available. Kubernetes, often abbreviated as "K8s", orchestrates containerized applications to run on a cluster of hosts. The K8s system automates the deployment and management of cloud native applications using on-premises infrastructure or public cloud platforms. It distributes application workloads across a Kubernetes cluster and automates dynamic container networking needs. Kubernetes also allocates storage and persistent volumes to running containers, provides automatic scaling, and works continuously to maintain the desired state of applications, providing resiliency.

Kubernetes has many features that help orchestrate containers across multiple hosts, automate the management of K8s clusters, and maximize resource usage through better utilization of infrastructure. Important features include:

- **Auto-scaling**. Automatically scale containerized applications and their resources up or down based on usage
- **Lifecycle management**. Automate deployments and updates with the ability to:
    - Rollback to previous versions
    - Pause and continue a deployment
- **Declarative model**. Declare the desired state, and K8s works in the background to maintain that state and recover from any failures
- **Resilience and self-healing**. Auto placement, auto restart, auto replication and auto scaling provide application self-healing
- **Persistent storage**. Ability to mount and add storage dynamically
- **Load balancing**. Kubernetes supports a variety of internal and external load balancing options to address diverse needs
- **DevSecOps support**. DevSecOps is an advanced approach to security that simplifies and automates container operations across clouds, integrates security throughout the container lifecycle, and enables teams to deliver secure, high-quality software more quickly. Combining DevSecOps practices and Kubernetes improves developer productivity.

3.2.4 AMAZON WEB SERVICES

In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses in the form of web services -- now commonly known as cloud computing. One of the key benefits of cloud computing is the opportunity to replace up-front capital infrastructure expenses with low variable costs that scale with your business. With the Cloud, businesses no longer need to plan for and procure servers and other IT infrastructure weeks or months in advance. Instead, they can instantly spin up hundreds or thousands of servers in minutes and deliver results faster.

Today, Amazon Web Services provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world. With data centre locations in the U.S., Europe, Brazil,

Singapore, Japan, and Australia, customers across all industries are taking advantage of the following benefits:

Low Cost:

AWS offers low, pay-as-you-go pricing with no up-front expenses or long-term commitments. We are able to build and manage a global infrastructure at scale, and pass the cost saving benefits onto you in the form of lower prices. With the efficiencies of our scale and expertise, we have been able to lower our prices on 15 different occasions over the past four years.

Agility and Instant Elasticity:

AWS provides a massive global cloud infrastructure that allows you to quickly innovate, experiment and iterate. Instead of waiting weeks or months for hardware, you can instantly deploy new applications, instantly scale up as your workload grows, and instantly scale down based on demand. Whether you need one virtual server or thousands, whether you need them for a few hours or 24/7, you still only pay for what you use.

Open and Flexible:

AWS is a language and operating system agnostic platform. You choose the development platform or programming model that makes the most sense for your business. You can choose which services you use, one or several, and choose how you use them. This flexibility allows you to focus on innovation, not infrastructure.

Secure:

AWS is a secure, durable technology platform with industry-recognized certifications and audits: PCI DSS Level 1, ISO 27001, FISMA Moderate, FedRAMP, HIPAA, and SOC 1 (formerly referred to as SAS 70 and/or SSAE 16) and SOC 2 audit reports. Our services and data centers have multiple layers of operational and physical security to ensure the integrity and safety of your data.

3.3 TECHNOLOGY USED

3.3.1 SPRING BOOT

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run.

You can get started with minimum configurations without the need for an entire Spring configuration setup.

Spring Boot automatically configures your application based on the dependencies you have added to the project by using @EnableAutoConfiguration annotation. For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class Contains **@**SpringBootApplication annotation and the main method.

Spring Boot automatically scans all the components included in the project by using **@**ComponentScan annotation.

3.3.2 DOCKER

Docker is an opensource containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers simplify delivery of distributed applications, and have

become increasingly popular as organizations shift to cloud-native development and hybrid multi-cloud environments.

Developers can create containers without Docker, but the platform makes it easier, simpler, and safer to build, deploy and manage containers. Docker is essentially a toolkit that enables developers to build, deploy, run, update, and stop containers using simple commands and work-saving automation through a single API.

3.3.3 KUBERNETES

Kubernetes is a portable, extensible, opensource platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

Kubernetes, often abbreviated as “K8s”, orchestrates containerized applications to run on a cluster of hosts. The K8s system automates the deployment and management of cloud native applications using on-premises infrastructure or public cloud platforms. It distributes application workloads across a Kubernetes cluster and automates dynamic container networking needs. Kubernetes also allocates storage and persistent volumes to running containers, provides automatic scaling, and works continuously to maintain the desired state of applications, providing resiliency.

3.3.4 SWAGGER

Swagger is a set of opensource tools for writing REST-based APIs. It simplifies the process of writing APIs by notches, specifying the standards & providing the tools required to write beautiful, safe, performant & scalable APIs.

3.3.5 AMAZON WEB SERVICES

In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses in the form of web services -- now commonly known as cloud computing. One of the key benefits of cloud computing is the opportunity to replace up-front capital infrastructure expenses with low variable costs that scale with your business. With the Cloud, businesses no longer need to plan for and procure servers and other IT infrastructure weeks or months in advance. Instead, they can instantly spin up hundreds or thousands of servers in minutes and deliver results faster.

Today, Amazon Web Services provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world.

CHAPTER 4

BACKEND DESIGN

4.1 DATA DICTIONARY

Microservice Name	Database Name	Table Name
Products	**products**	**product_details**
Supplied Products	**suppliedproducts**	**supplied_product_details**
Order	**orders**	**order_details**
Report	**reports**	**report_details**
Delivery	**delivery**	**delivery_details**

Table 4.1

4.2 ER Diagram

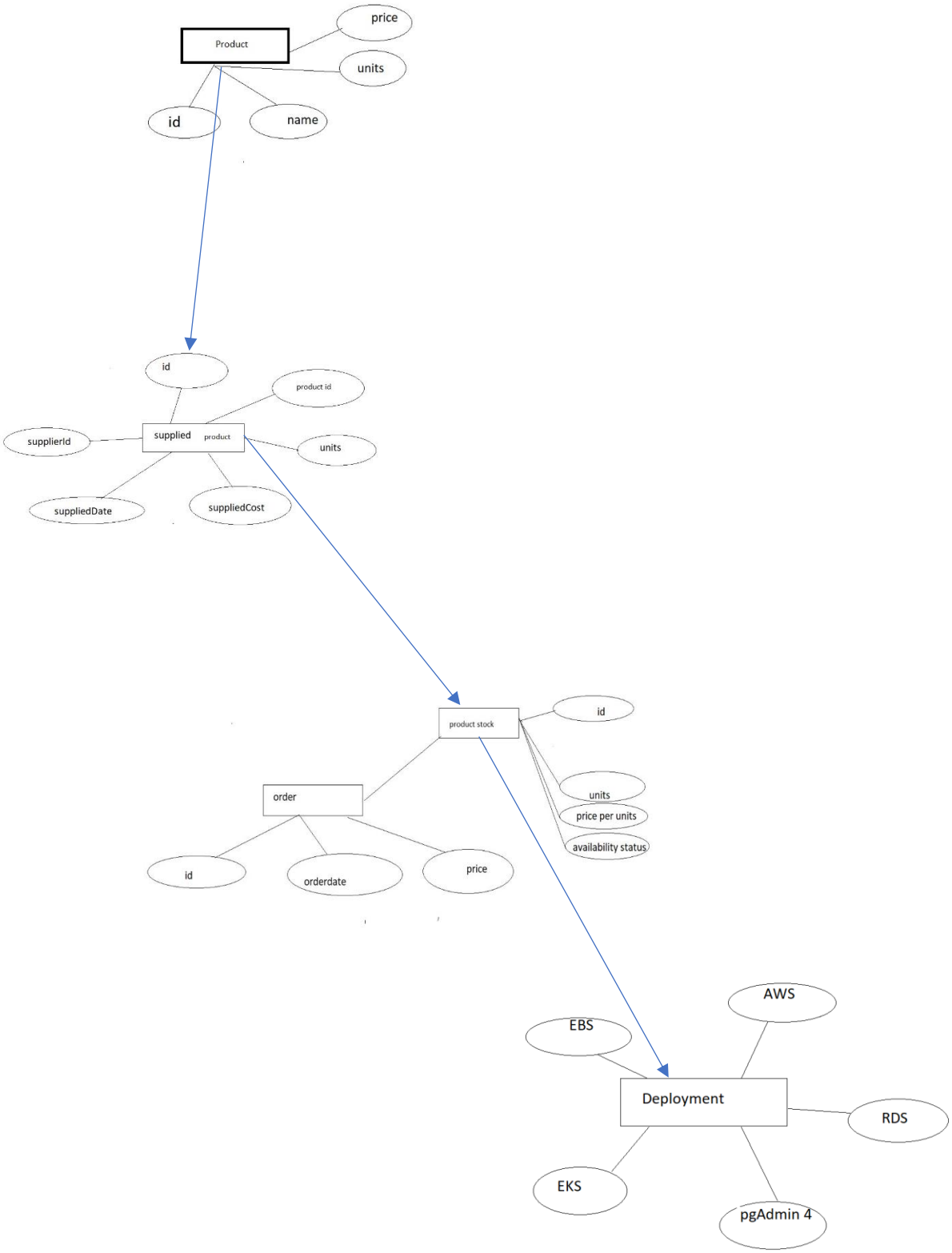


Fig 4.2

4.3 DATABASE DESIGN

i. PRODUCT

product_details

Id	Int
Product_name	varchar(20)
Product_id	int
Product_rating	int

Table 4.2

supplied_product_details

Id	Int
stockid	Int
supplierid	Int
units	Int
suppliedcost	Double
supplieddate	Date(yyyy/mm/dd)

Table 4.3

order_details

Id	Int
Orderdate	Date(yyyy/mm/dd)
price	double

Table 4.4

delivery_details

Id	Int
Orderid	Int
Deliverydate	Date(yyyy/mm/dd)
deliverystatus	Varchar(20)

Table 4.5

report_details

Id	Int
Startperiod	Date(yyyy/mm/dd)
Endperiod	Date(yyyy/mm/dd)
Createddate	Date(yyyy/mm/dd)
Totalsaleprice	Double
ordercount	Int

Table 4.6

order_stock_details

Id	Int
Stockid	Int
Units	Int
Priceperunit	Int
Availabilitystatus	Varchar(20)

Table 4.7

CHAPTER 5

CODE

build.gradle :

```
plugins {
	id 'org.springframework.boot' version '2.6.6'
	id 'io.spring.dependency-management' version '1.0.11.RELEASE'
	id 'java'
}

group = 'com.ecommerce'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
	mavenCentral()
}

dependencies {
	implementation 'org.springframework.boot:spring-boot-starter-data-rest'
	implementation 'org.springframework.boot:spring-boot-starter-web'
	implementation 'org.springframework.boot:spring-boot-starter-web-services'
	runtimeOnly 'com.h2database:h2'
	runtimeOnly 'org.postgresql:postgresql'
	testImplementation 'org.springframework.boot:spring-boot-starter-test'
	//	https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa
	implementation group: 'org.springframework.boot', name: 'spring-boot-starter-data-jpa', version: '2.6.4'

}

tasks.named('test') {
	useJUnitPlatform()
}
```

Package: com.ecommerce.controller

File: ProductController.java

```
package com.ecommerce.controller;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;


import com.ecommerce.model.Products;

import com.ecommerce.repository.ProductRepository;
```

```
@RestController
public class ProductController {

    @Autowired
    private ProductRepository repository;

    //ProductController(ProductRepository repository)
    //{
    //    this.repository = repository;
    //}

    @GetMapping("/products")
    List<Products> getAll(){

                return repository.findAll();
    }
```

```
  @GetMapping("/product/{id}")
  Optional<Products> getProduct(@PathVariable Integer id){
      if((repository.findById(id)).isEmpty())
          return Optional.empty();

          else {
              return repository.findById(id);
          }
}

 @GetMapping("/")
      public String health() {
          return "Hello & Welcome to Springboot !!!";
      }

  @PostMapping("/products")
  String addProduct(@RequestBody Products pro) {
      if((repository.findById(pro.getProduct_id())).isEmpty()) {
```

```
            repository.save(pro);
            return "Product Added";
        }
        else {
            return "Given product id already exists";
        }
}
@PostMapping("/delete/{id}")
String deleteProduct(@PathVariable Integer id)
{
    if((repository.findById(id)).isEmpty())
                return "No Product present with this id";
    else {
            repository.deleteById(id);
            return "Product id "+id +" deleted successfully";
    }
}
```

```
    @PutMapping("/products/{id}")
    String updateProduct(@RequestBody Products pro, @PathVariable Integer id) {
        return repository.findById(id).map(
                product -> {
                        product.setProduct_name(pro.getProduct_name());
                //product.setProduct_id(pro.getProduct_id());
                //product.setProduct_rating(pro.getProduct_rating());
                         repository.save(product);
                         return "Product name updated";
                }).orElseGet(() -> {
                        repository.save(pro);
                        return "New Product added";
                });
    }

}
```

Package: com.ecommerce.main

File: EcommerceProjectApplication.java

```
package com.ecommerce.main;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication
@ComponentScan("com.ecommerce.*")
@EnableJpaRepositories("com.ecommerce.repository")
@EntityScan("com.ecommerce.model")
public class EcommerceProjectApplication {

	public static void main(String[] args) {
		SpringApplication.run(EcommerceProjectApplication.class, args);
	}

}
```

Package: com.ecommerce.model

File: Products.java

```
package com.ecommerce.model;

import java.util.Objects;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="PRODUCTS")
public class Products {
	private @Id Integer product_id;
	private String product_name;
	private int product_rating;
	public Products() {

	}
	public Products(Integer product_id, String product_name, int product_rating)
{
		super();
		this.product_id = product_id;
		this.product_name = product_name;
		this.product_rating = product_rating;
	}
```

```
	public Integer getProduct_id() {
		return product_id;
	}
	public void setProduct_id(Integer product_id) {
		this.product_id = product_id;
	}
	public String getProduct_name() {
		return product_name;
	}
	public void setProduct_name(String product_name) {
		this.product_name = product_name;
	}
	public int getProduct_rating() {
		return product_rating;
	}
	public void setProduct_rating(int product_rating) {
		this.product_rating = product_rating;
	}
	@Override
	public String toString() {
		return "Products [product_id=" + product_id + ", product_name=" +
product_name + ", product_rating="
				+ product_rating + "]";
	}
	@Override
	public int hashCode() {
		return Objects.hash(product_id, product_name, product_rating);
```

```
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Products other = (Products) obj;
		return Objects.equals(product_id, other.product_id) &&
Objects.equals(product_name, other.product_name)
				&& product_rating == other.product_rating;
	}

}
```

Package: com.ecommerce.repository

File: ProductRepository.java

```
package com.ecommerce.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.ecommerce.model.Products;
public interface ProductRepository extends JpaRepository<Products, Integer> {

}
```

Package: com.ecommerce.main

File: EcommerceProjectApplicationTests.java

```
package com.ecommerce.main;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class EcommerceProjectApplicationTests {

	@Test
	void contextLoads() {
	}

}
```

application.properties :

spring.datasource.url=jdbc:postgresql://ecommerce-db.ck6kwplo7mh0.ap-south-1.rds.amazonaws.com:5432/ecommerce

spring.datasource.username=username
spring.datasource.password=password

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL81Dialect

CHAPTER 6

REPORT

REPORT/OUTPUT:

- Creating Database on Local Machine using PostgreSQL

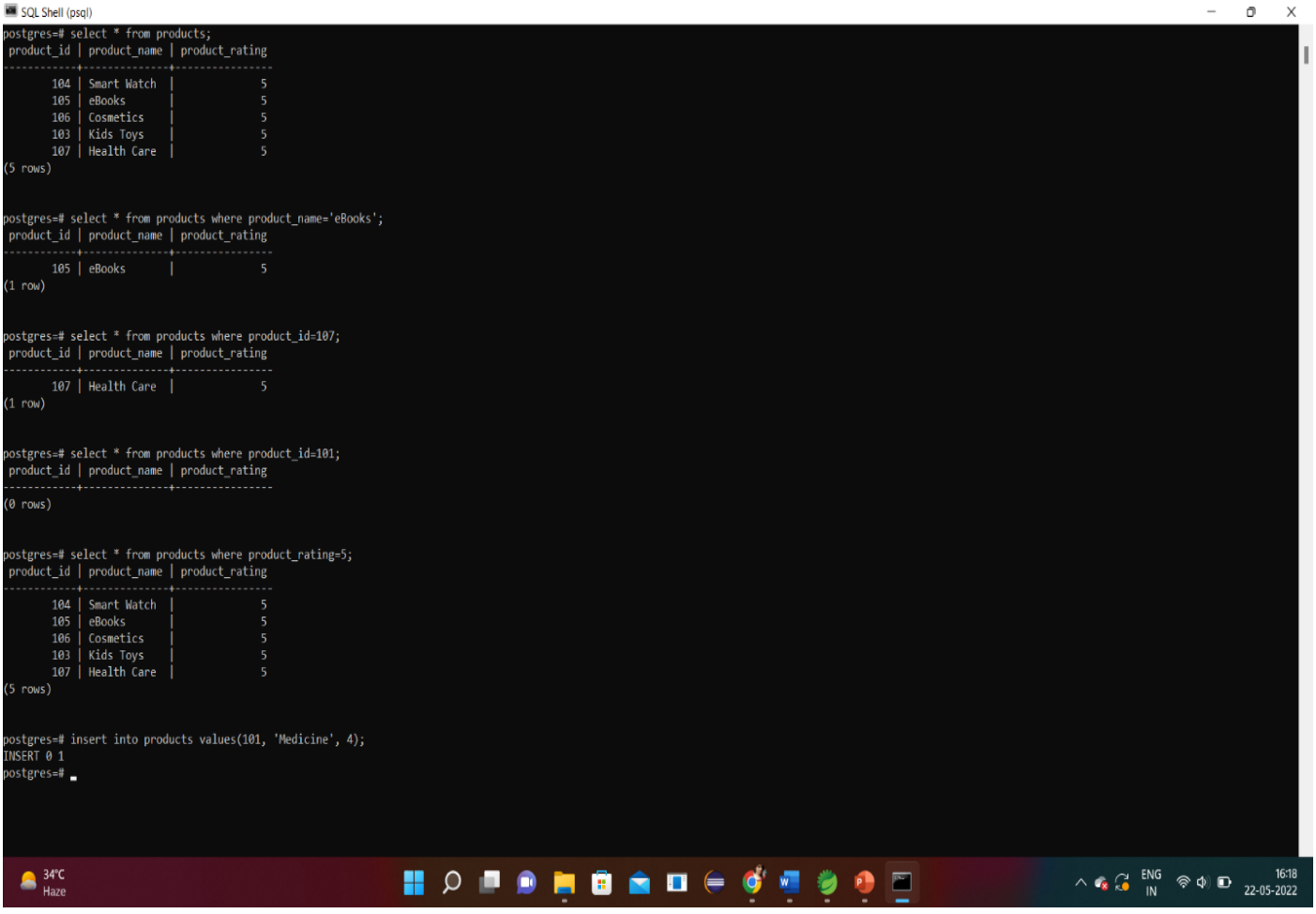
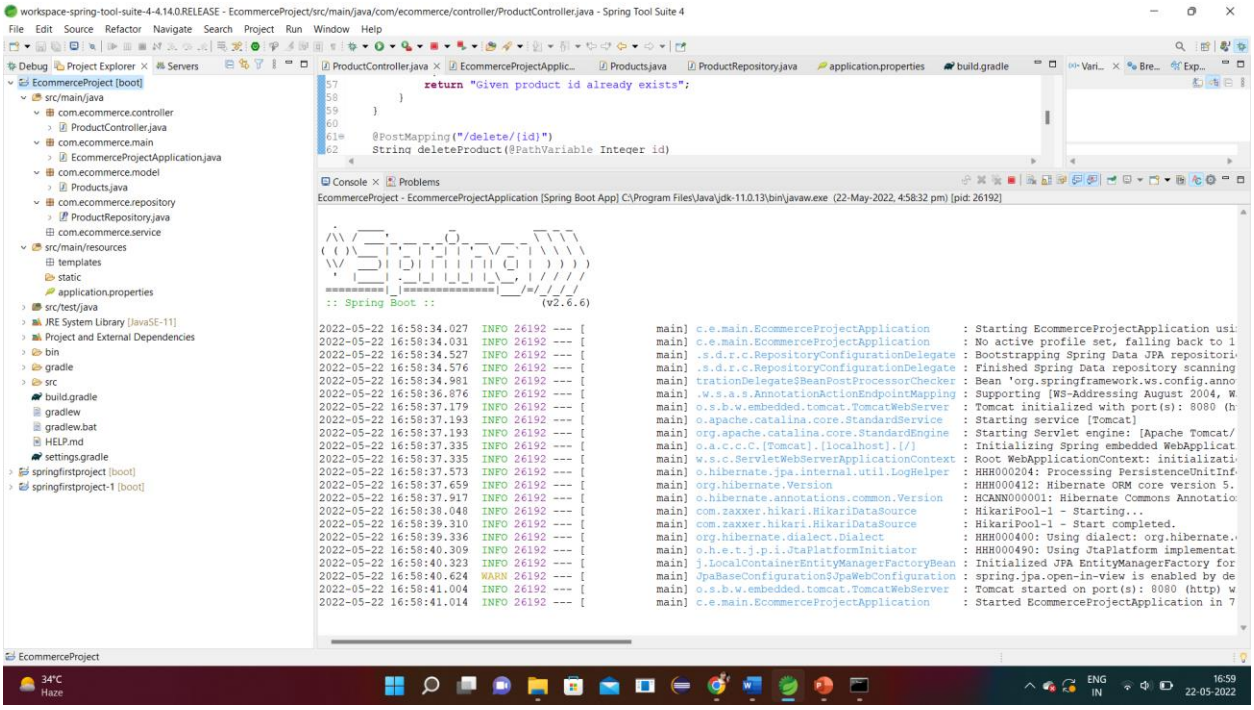


Fig 6.1 PostgreSQL

- Creating and Starting the Product Application using microservices on Spring Boot



6.2 Spring Boot Application

Check the running Application on the Browser

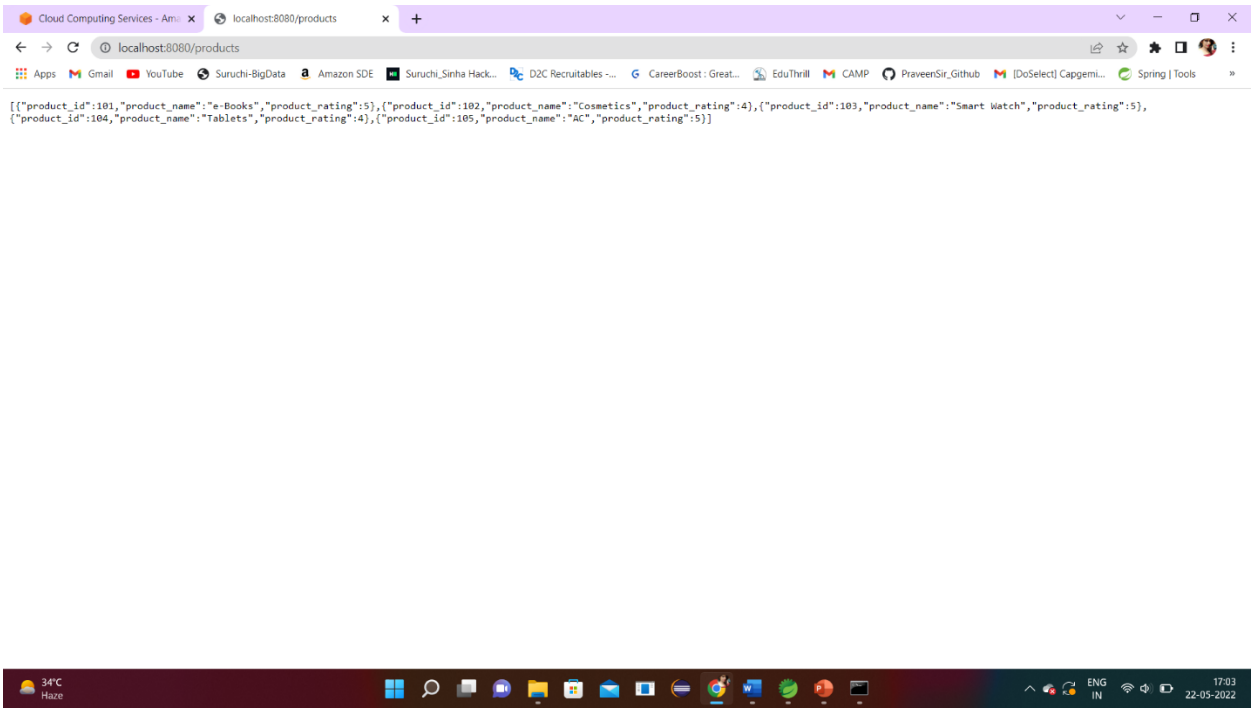


Fig 6.3 Running Application on Browser

- Check the proper functioning of CRUD Operations on Postman

R in CRUD -> Read

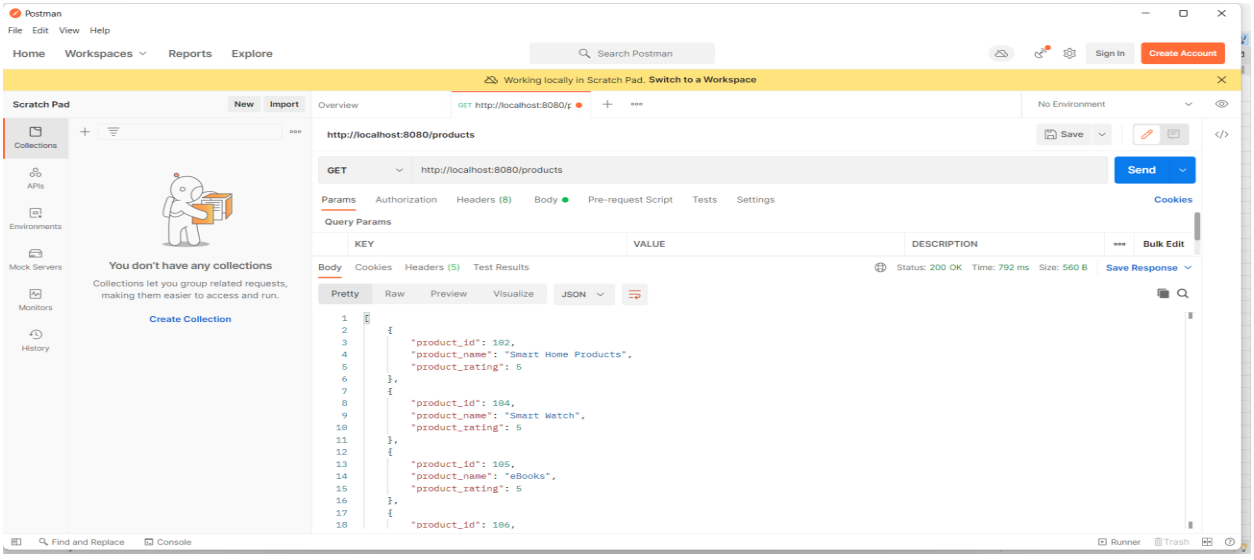


Fig 6.4 Postman (Read Operation)

C in CRUD -> Create

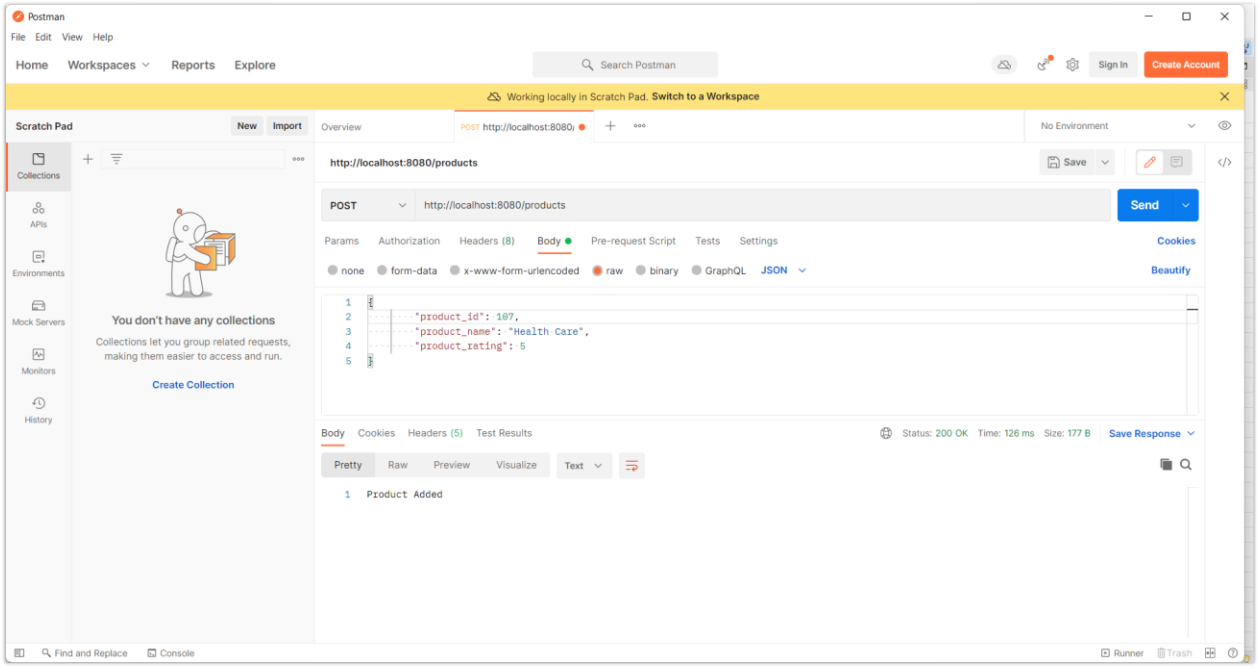


Fig 6.5 Postman (Create Operation)

U in CRUD -> Update

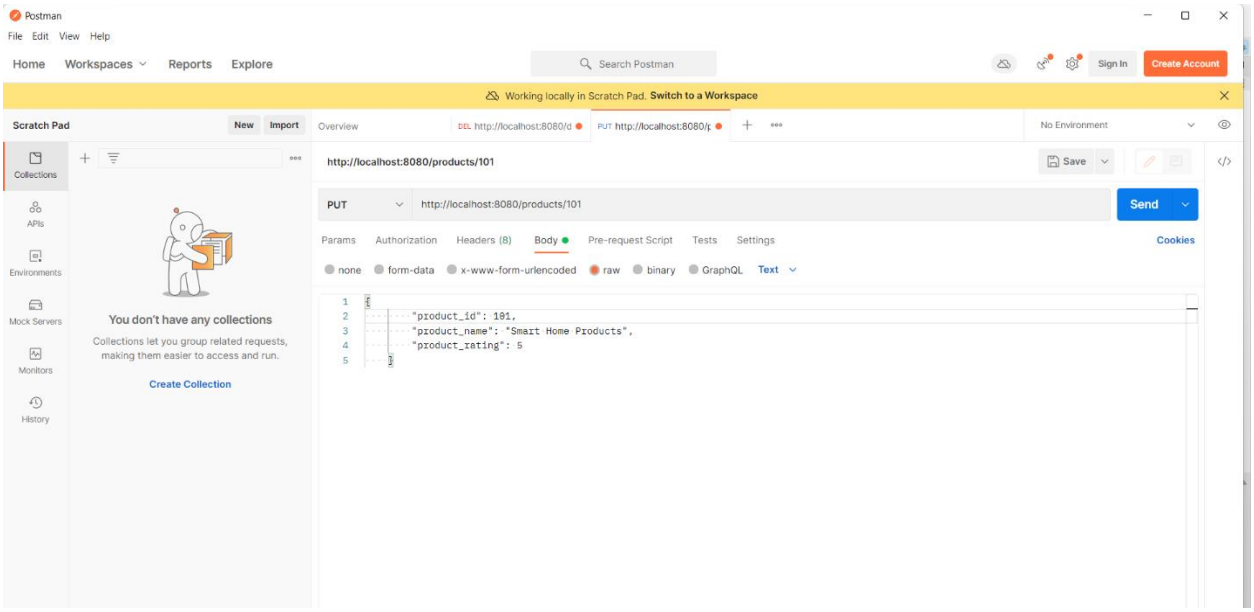


Fig 6.6 Postman (Update Operation)

U in CRUD -> Delete

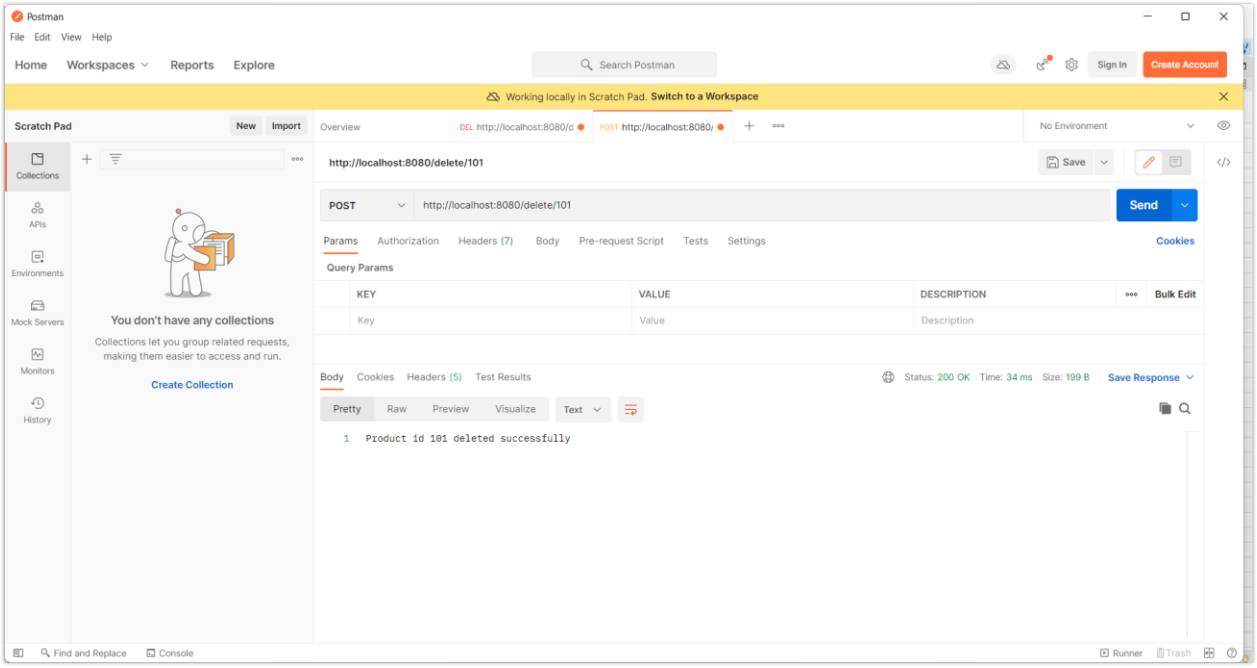


Fig 6.7 Postman (Delete Operation)

Display particular product using product id

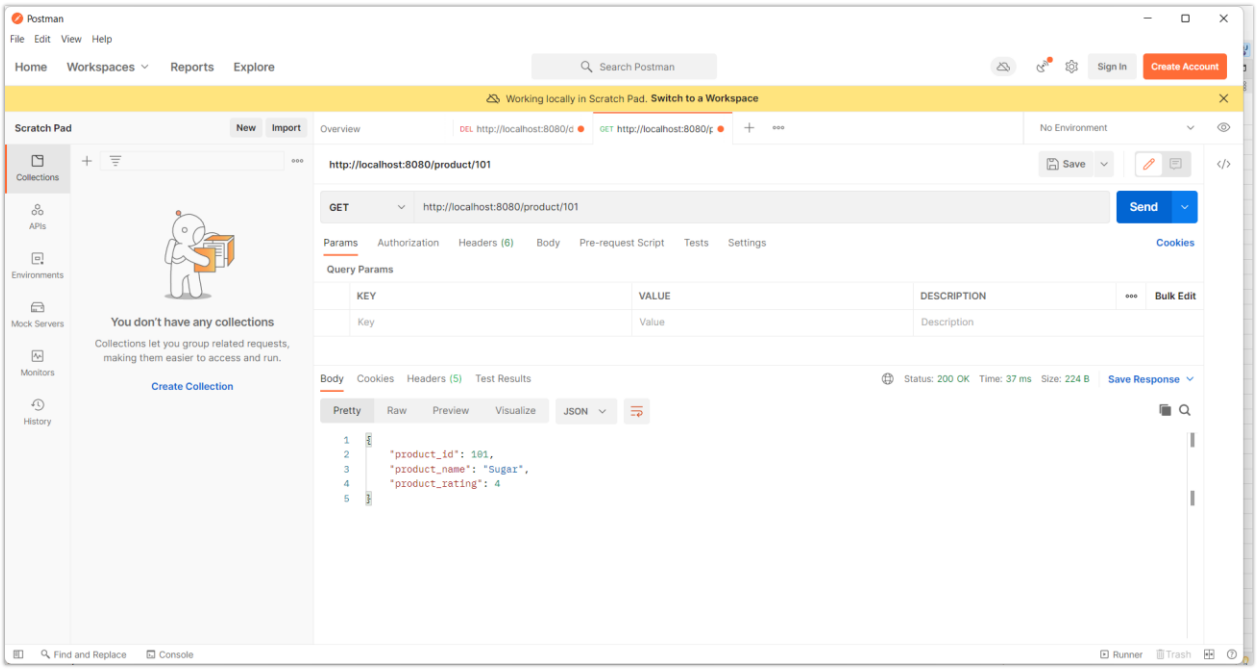


Fig 6.8 Postman (Displaying particular value)

Deployment on AWS:

Creating Database on RDS:

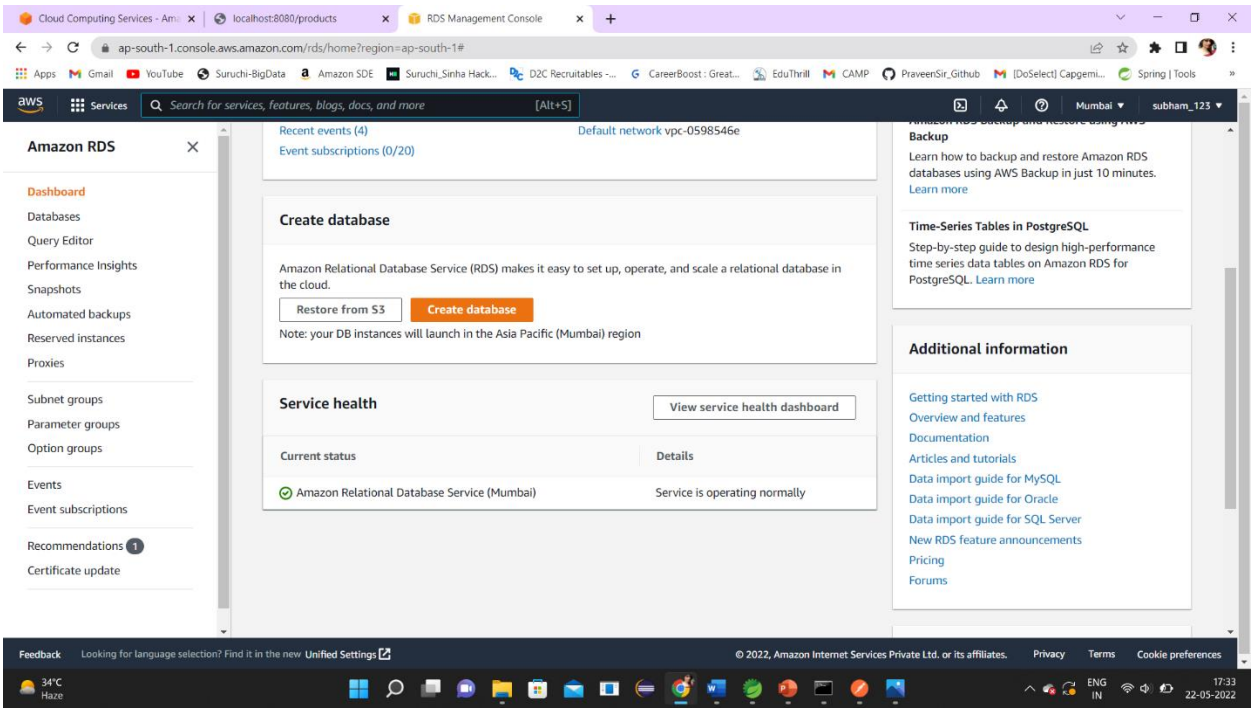


Fig 6.9 AWS RDS

Connecting RDS to pgAdmin 4

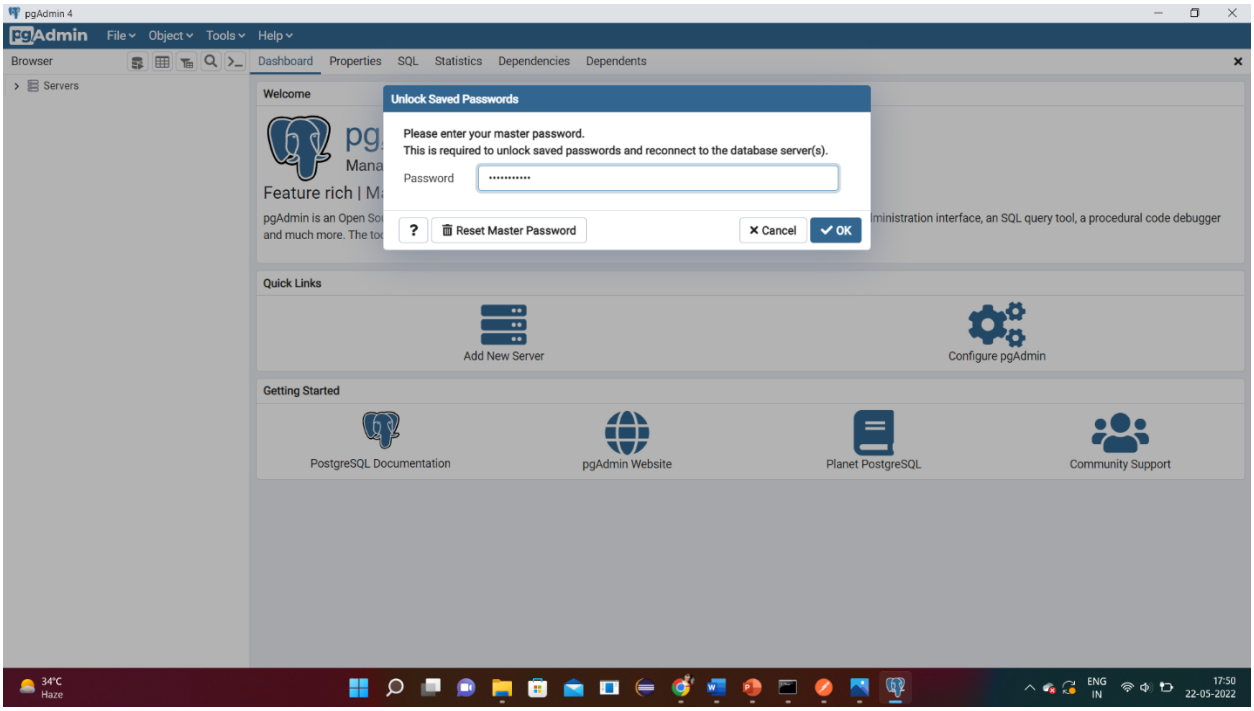


Fig 6.10 pgAdmin Login

pgAdmin Dashboard

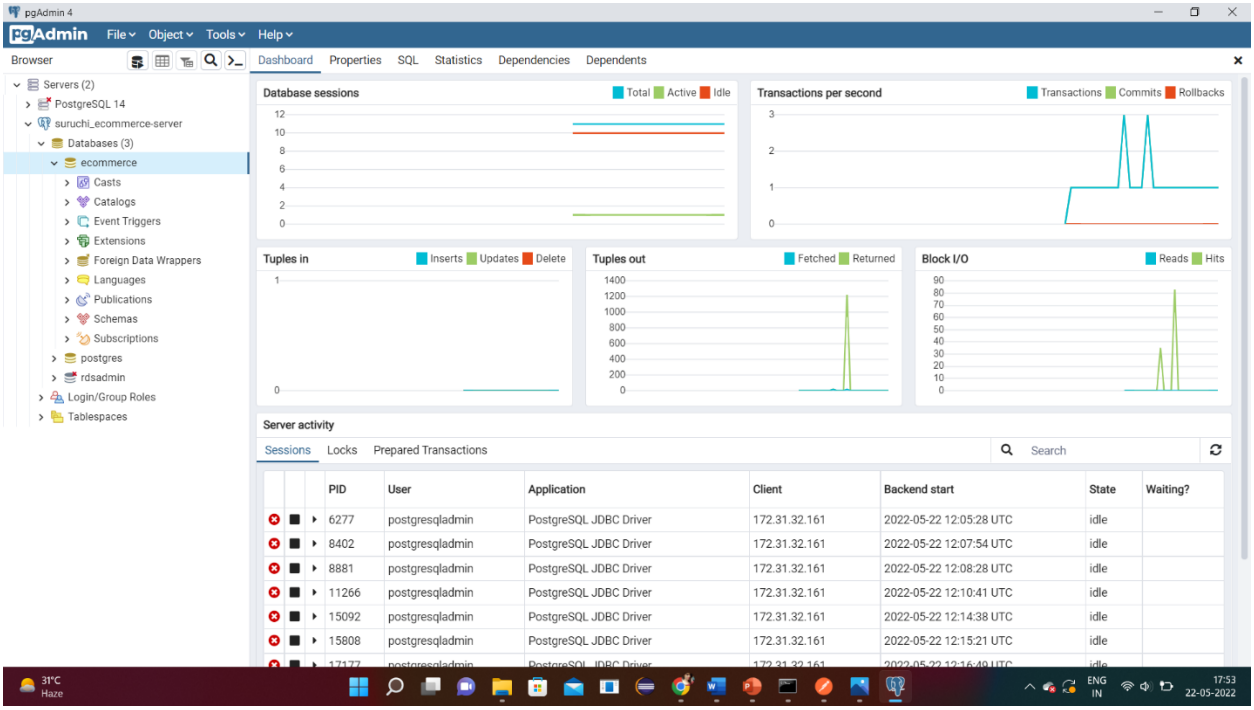


Fig 6.11 pgAdmin Dashboard

pgAdmin Active State

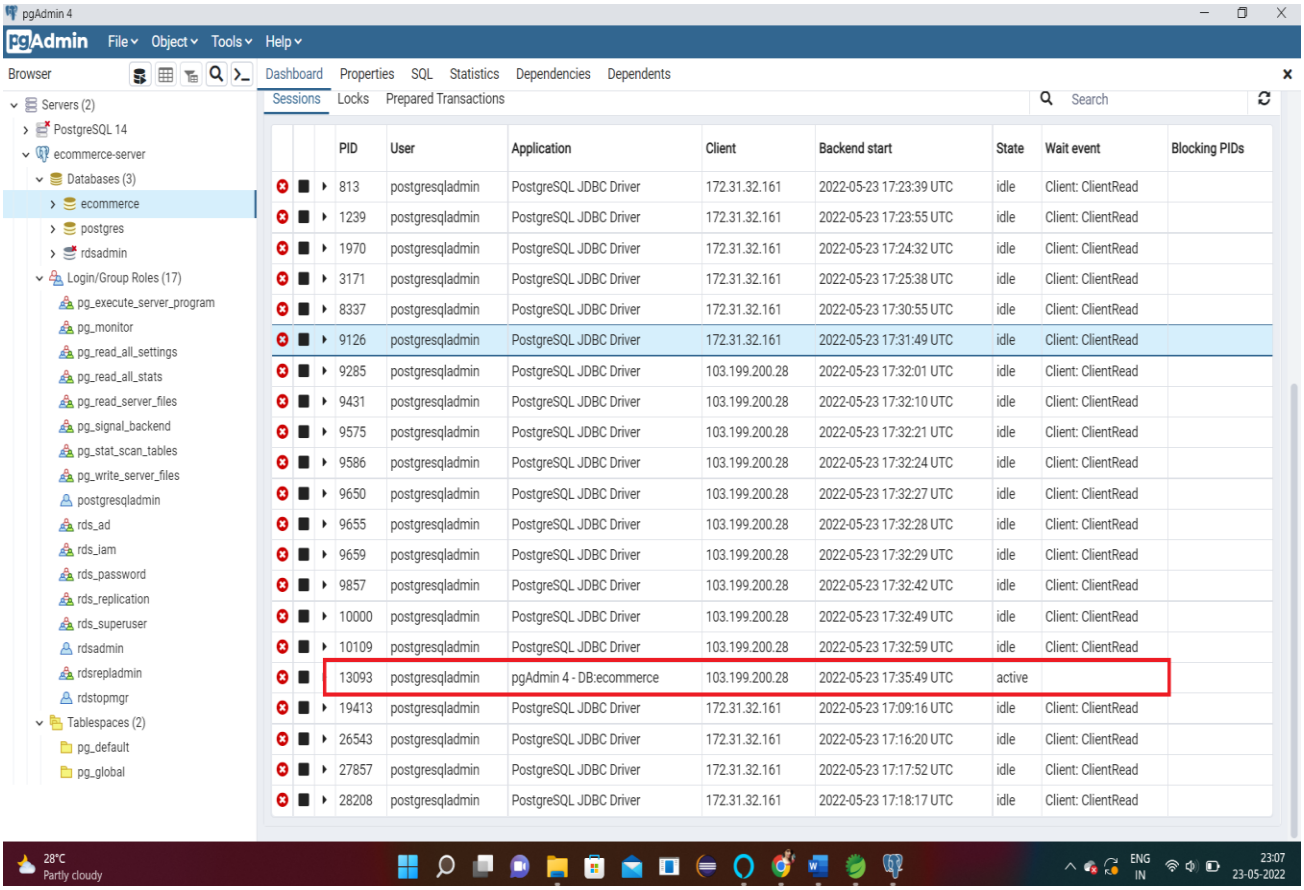


Fig 6.12 pgAdmin Active Status

Databases created on pgAdmin

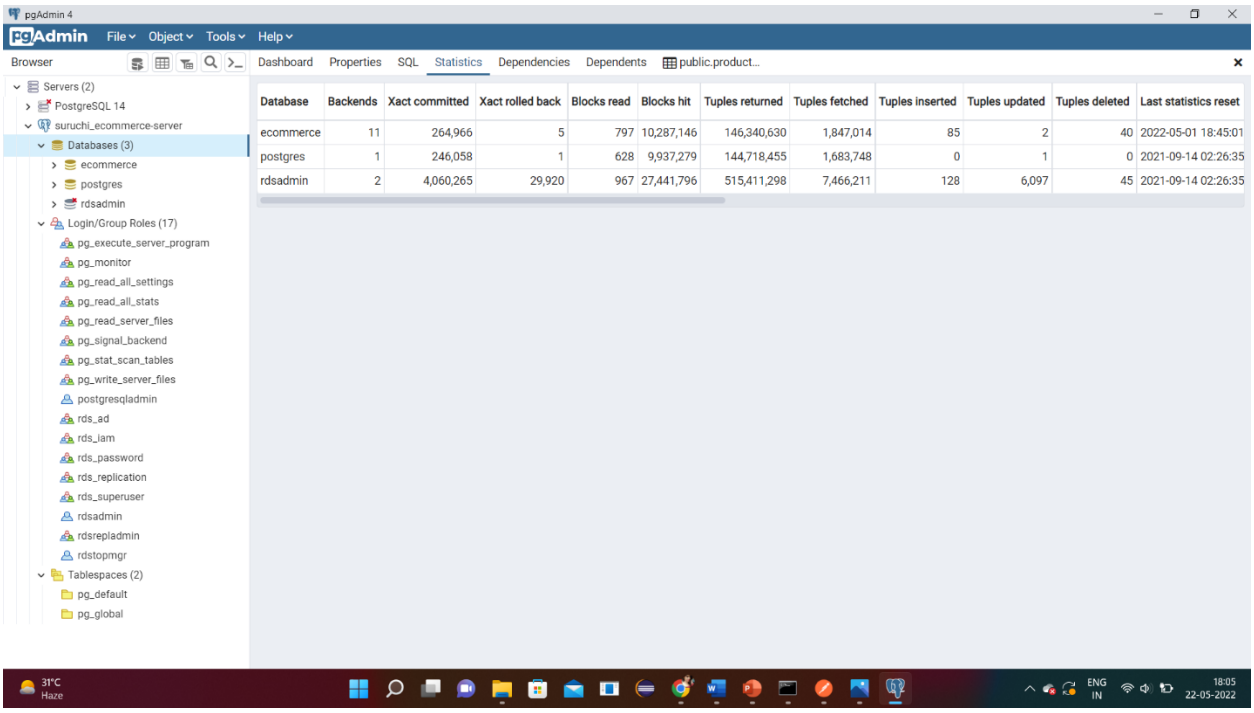


Fig 6.13 pgAdmin Databases

Table created on pgAdmin using PostgreSQL database to connect with RDS

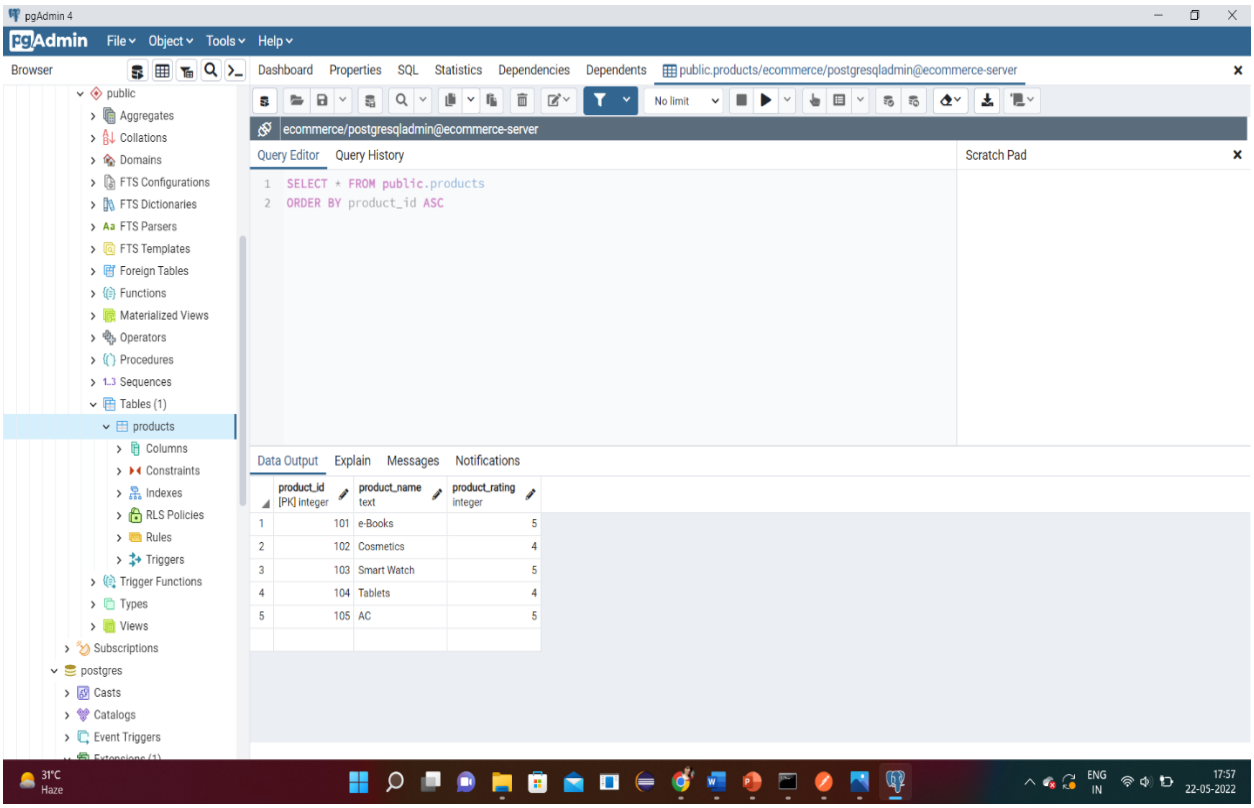


Fig 6.14 pgAdmin Table

Generate the jar file of the application by building the project on IntelliJ

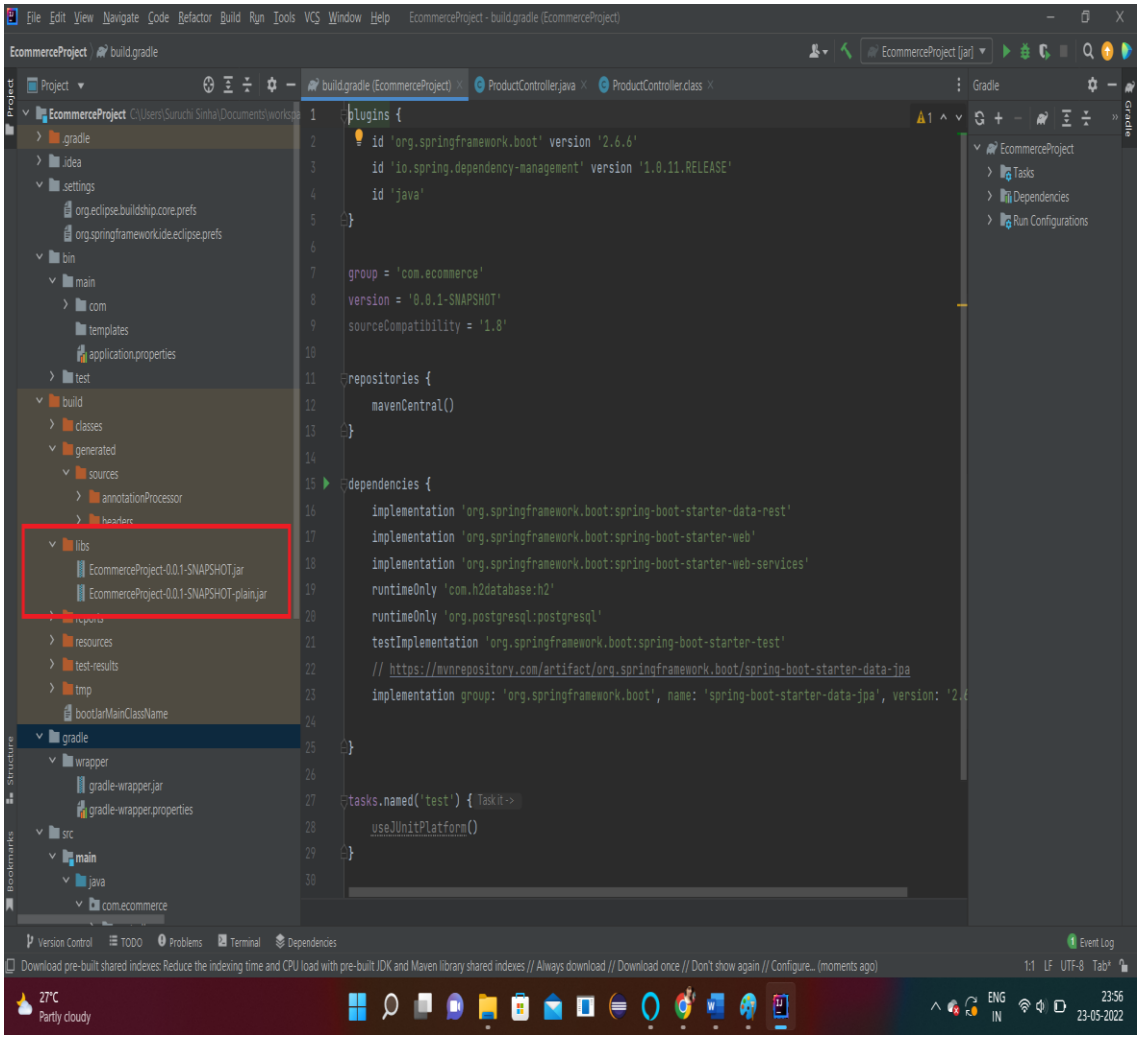


Fig 6.15 Generating jar file

Select Upload your code on EBS in AWS

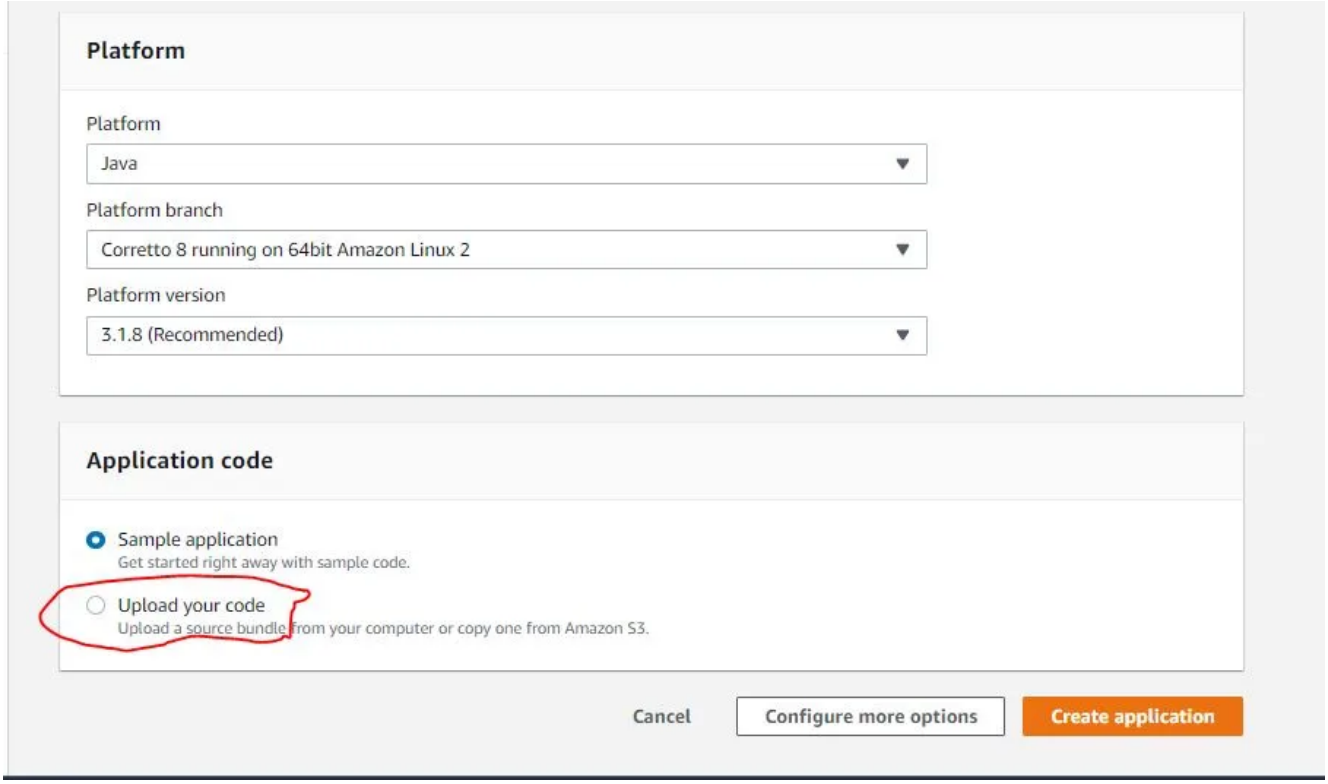


Fig 6.16 Uploading code

Choose the generated jar file to upload it on EBS

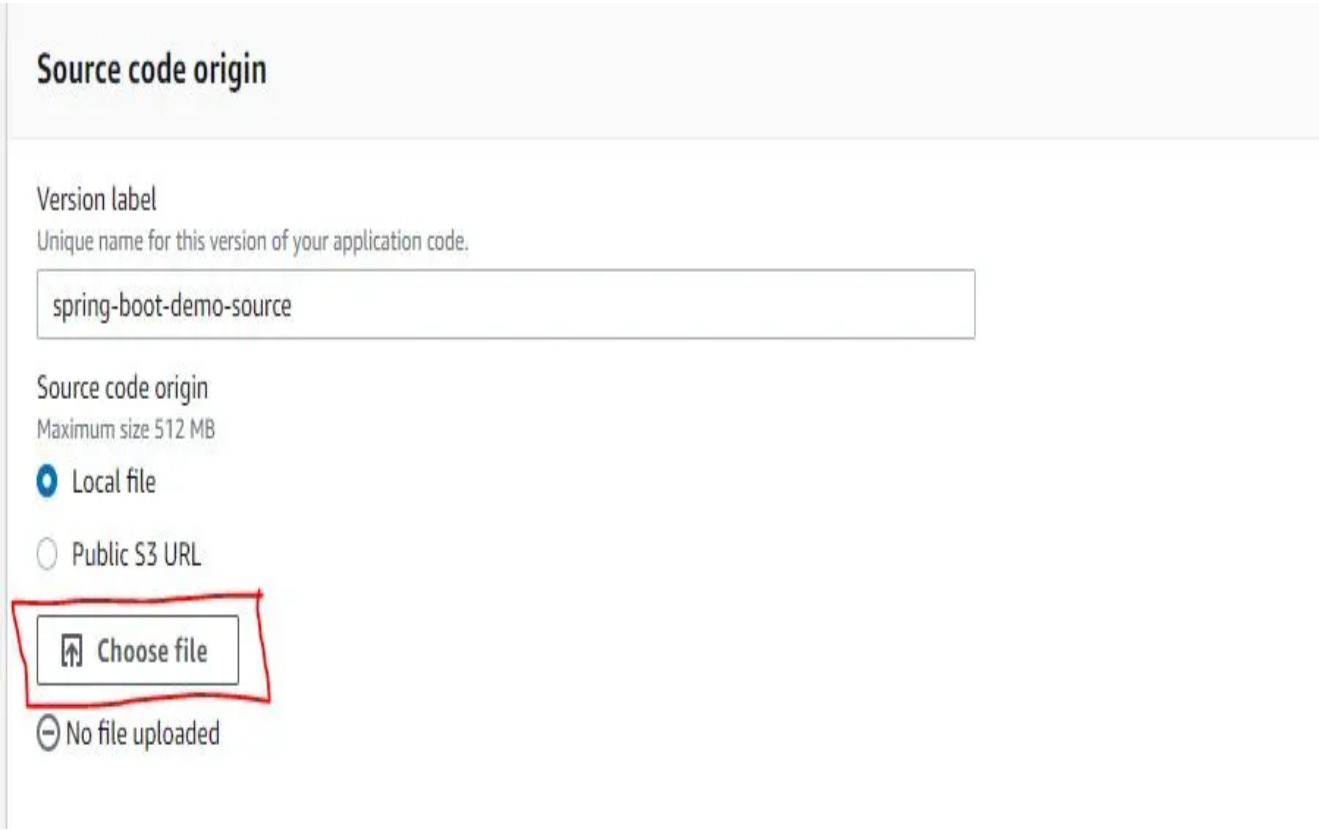


Fig 6.17 Choose file

Upload the generated and selected file

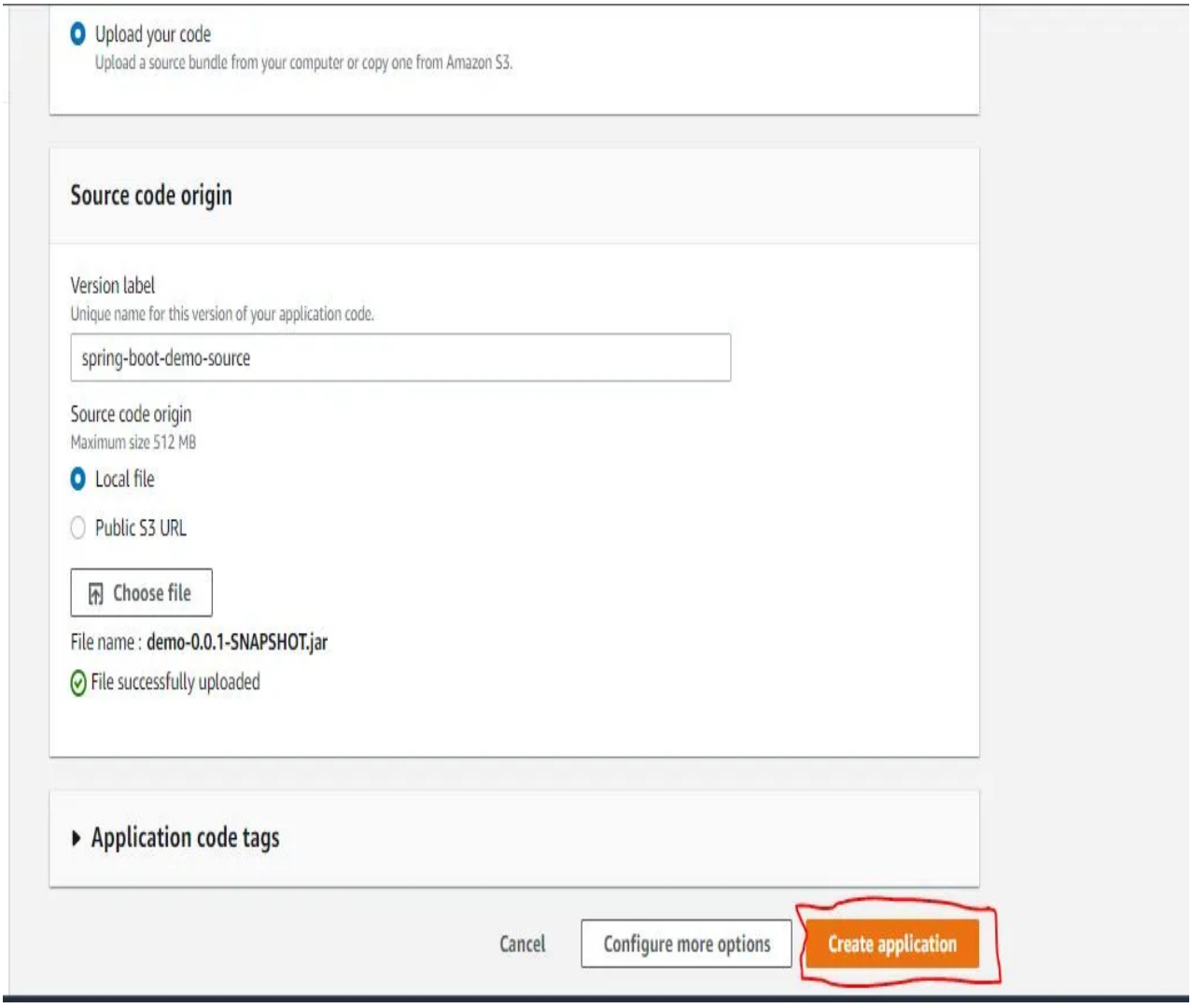


Fig 6.18 Upload file

Deployment on AWS using EBS (ELSTIC BEANSTALK)

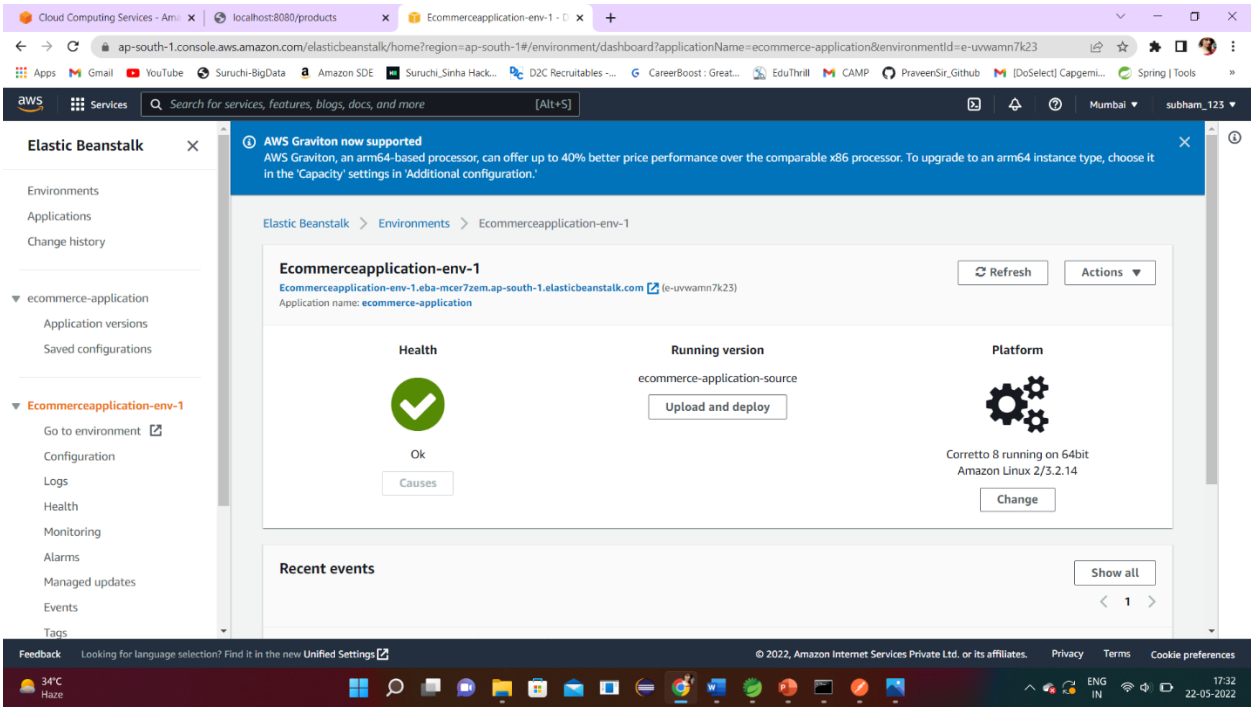


Fig 6.19 Heath-Check and Deployment

Connect Spring Boot with AWS using the generated URL in RDS

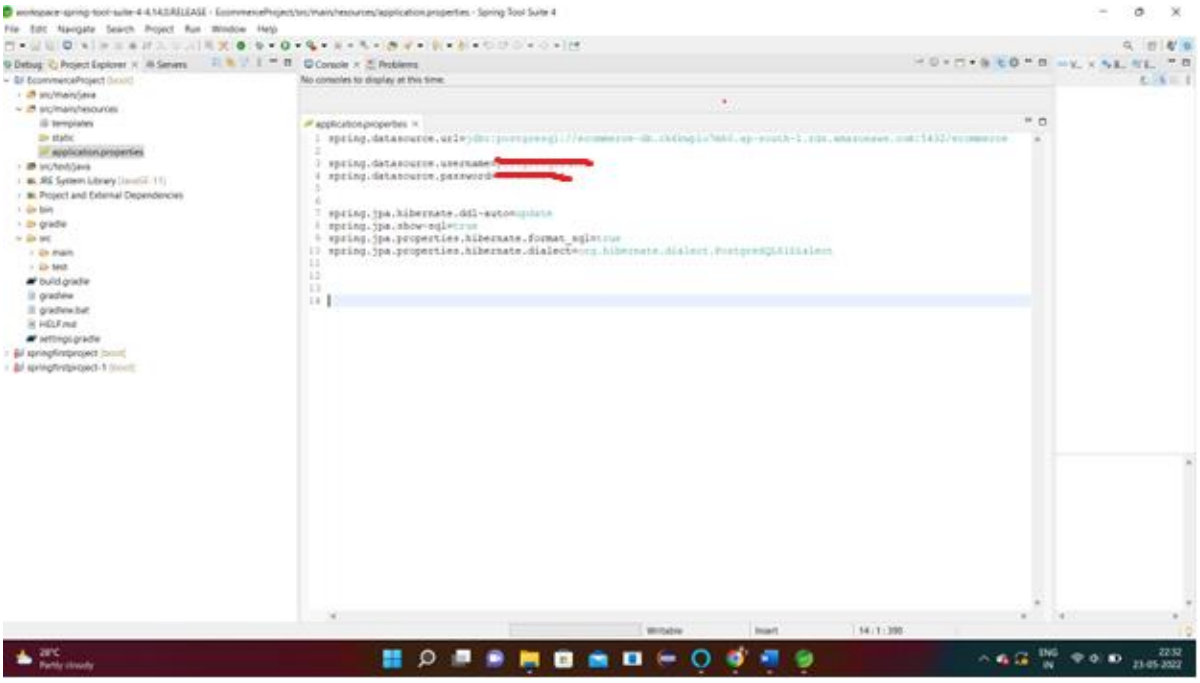


Fig 6.20 Connecting Spring Boot

Check the running application on the Browser post Deployment

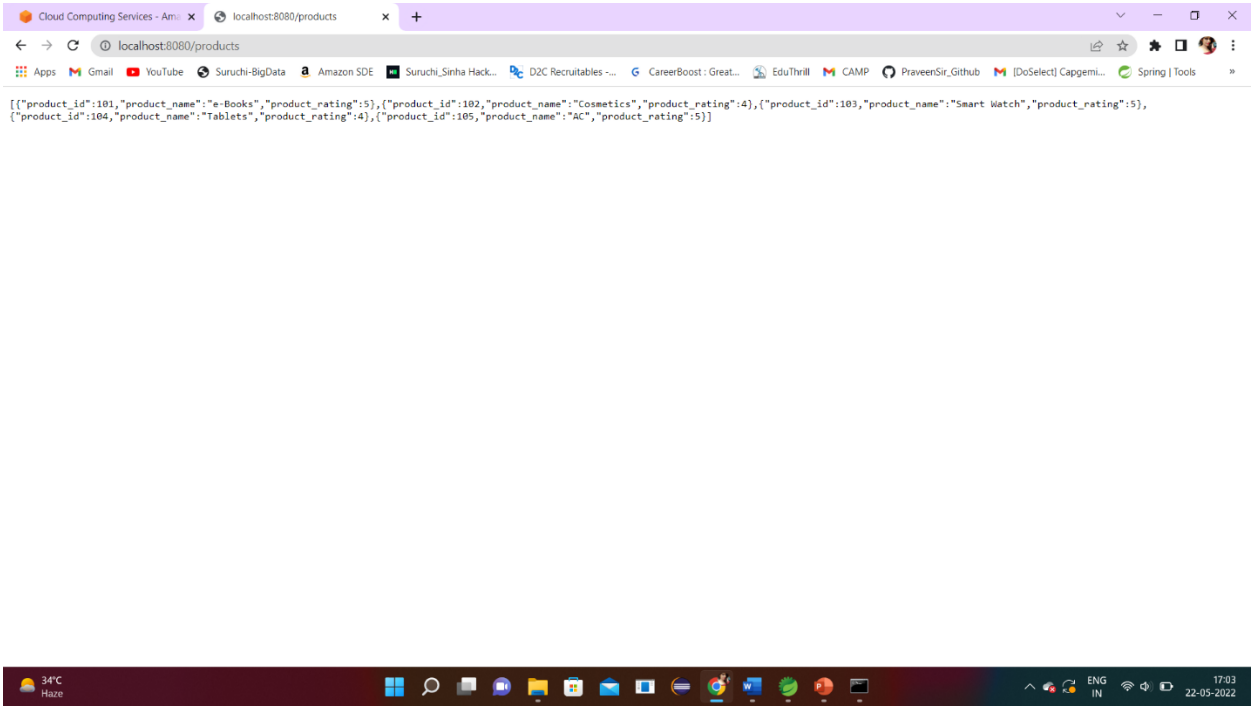


Fig 6.21 Output on the Browser post Deployment

Also check the running application on Postman

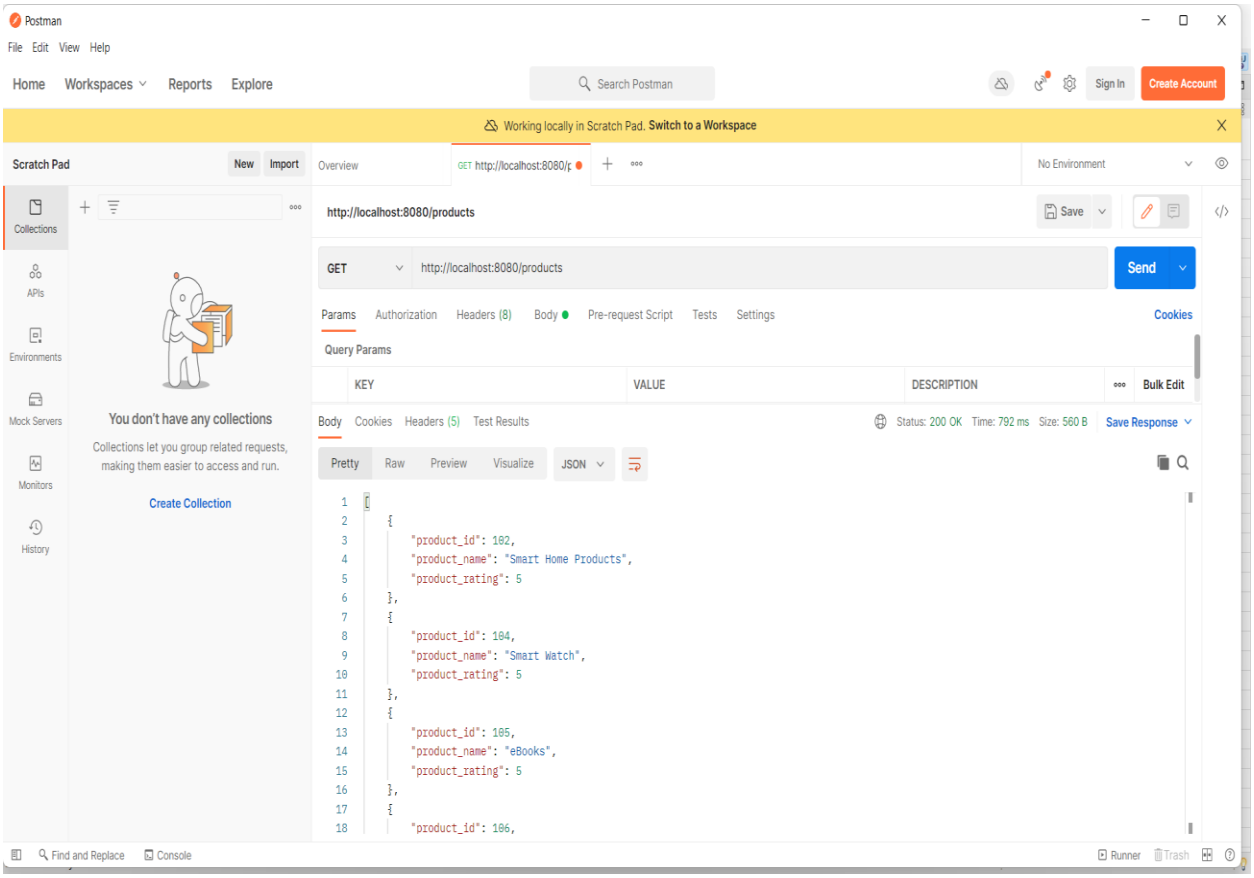


Fig 6.22 Output on the Postman post Deployment

CHAPTER 7

TESTING

```
    @Test
    public void test_add() {
        long stockId = (long) 2;
        long supplierId = (long) 3;
        String date = "04-04-2022";
        double cost = 45000.0;
        int unit = 5;
        AddSupplyStockRequest addSupply = new AddSupplyStockRequest();
        addSupply.setStockId(stockId);
        addSupply.setSupplierId(supplierId);
        addSupply.setSuppliedDate(date);
        addSupply.setSuppliedCost(cost);
        addSupply.setUnits(unit);

        SuppliedStockDetails details = mock(SuppliedStockDetails.class);
        SuppliedStock stock = mock(SuppliedStock.class);
        LocalDate localdate = LocalDate.of(2022, 04, 04);
        doReturn(stock).when(service).newSuppliedStock();
        when(dateUtil.convertToDate(date)).thenReturn(localdate);
        when(repository.save(stock)).thenReturn(stock);
        when(suppliedStoclUtil.toSuppliedStockDetails(stock)).thenReturn(details);

        SuppliedStockDetails result = service.add(addSupply);
        Assertions.assertSame(details, result);

    }
}
```

Figure 7.1

```java

    @Test
    public void test_add() {
        long stockId = (long) 2;
        long supplierId = (long) 3;
        String date = "04-04-2022";
        double cost = 45000.0;
        int unit = 5;
        AddSupplyStockRequest addSupply = new AddSupplyStockRequest();
        addSupply.setStockId(stockId);
        addSupply.setSupplierId(supplierId);
        addSupply.setSuppliedDate(date);
        addSupply.setSuppliedCost(cost);
        addSupply.setUnits(unit);

        SuppliedStockDetails details = mock(SuppliedStockDetails.class);
        SuppliedStock stock = mock(SuppliedStock.class);
        LocalDate localdate = LocalDate.of(2022, 04, 04);
        doReturn(stock).when(service).newSuppliedStock();
        when(dateUtil.convertToDate(date)).thenReturn(localdate);
        when(repository.save(stock)).thenReturn(stock);
        when(suppliedStoclUtil.toSuppliedStockDetails(stock)).thenReturn(details);

        SuppliedStockDetails result = service.add(addSupply);
        Assertions.assertSame(details, result);

    }
}
```

Figure 7.2

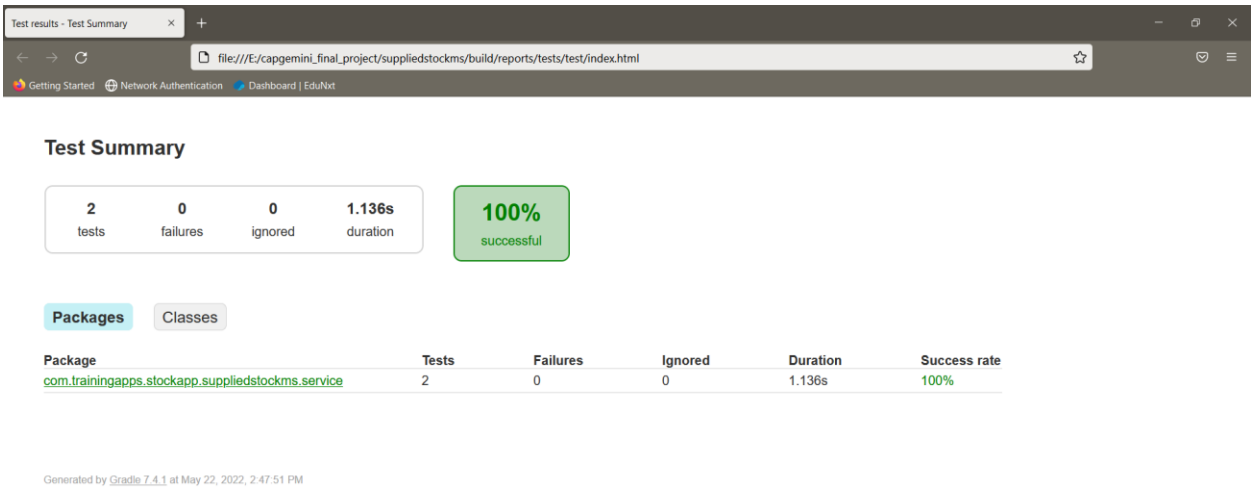


Figure 7.3

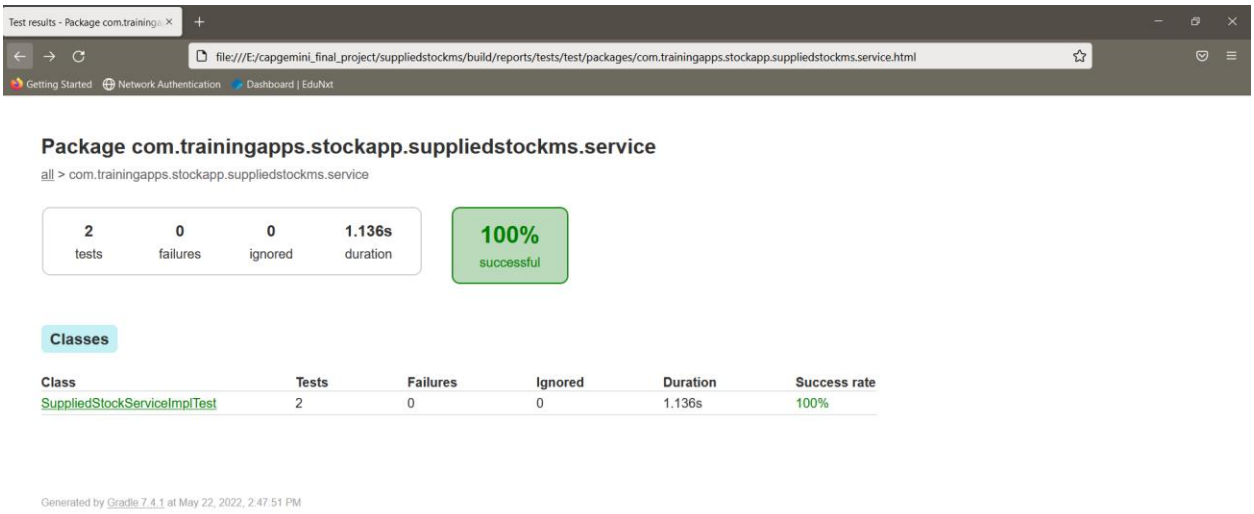


Figure 7.4

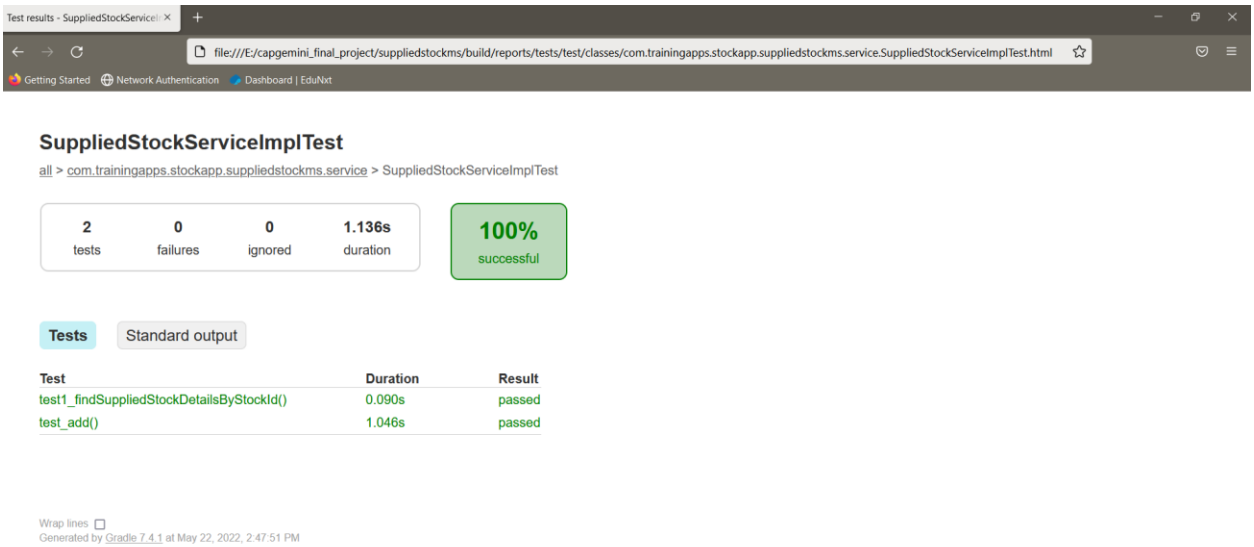


Figure 7.5

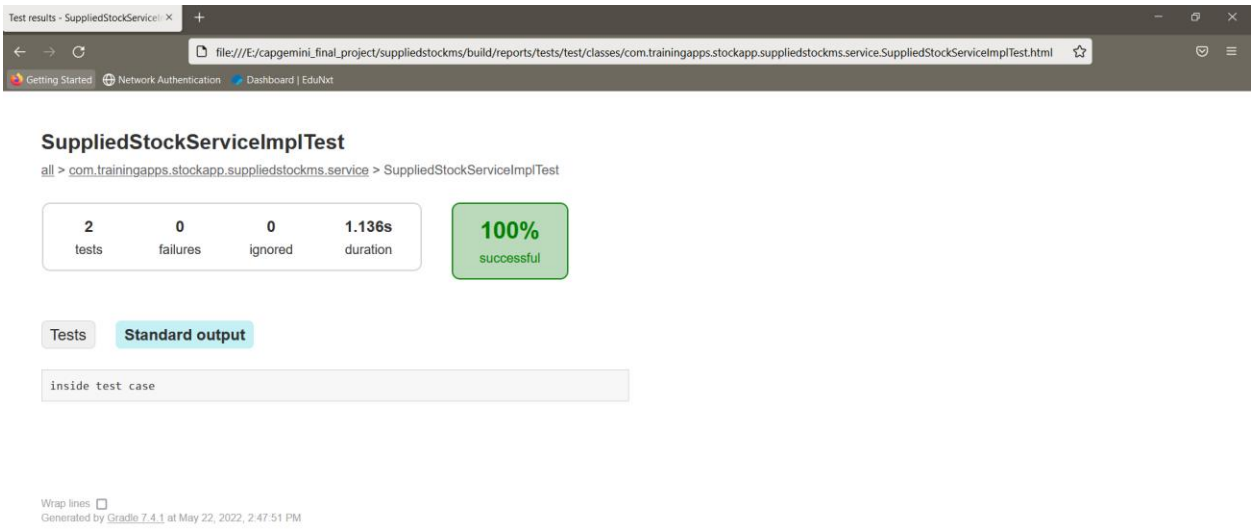


Figure 7.6

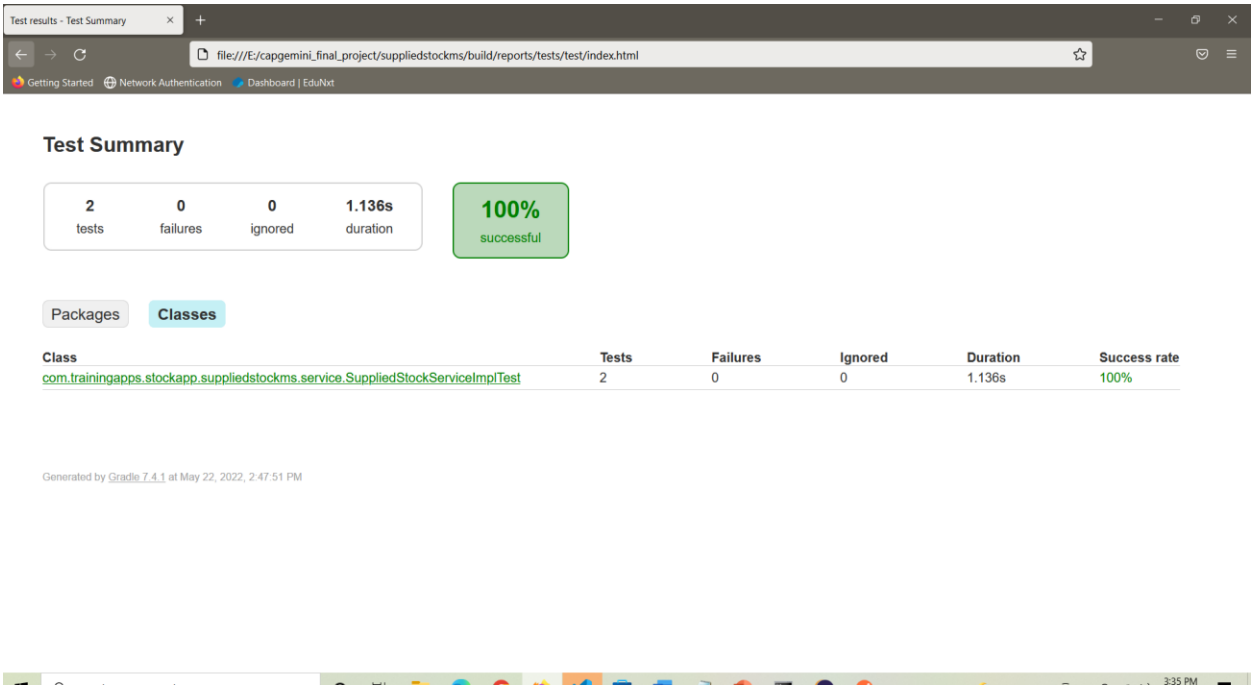


Figure 7.7

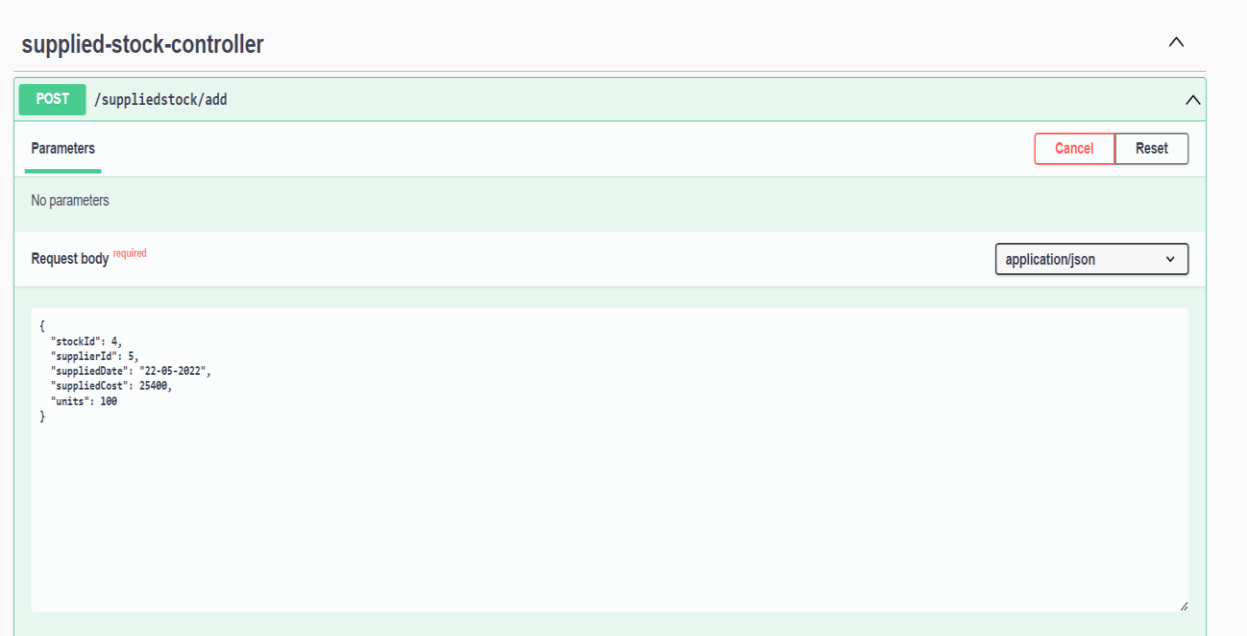


Figure 7.8

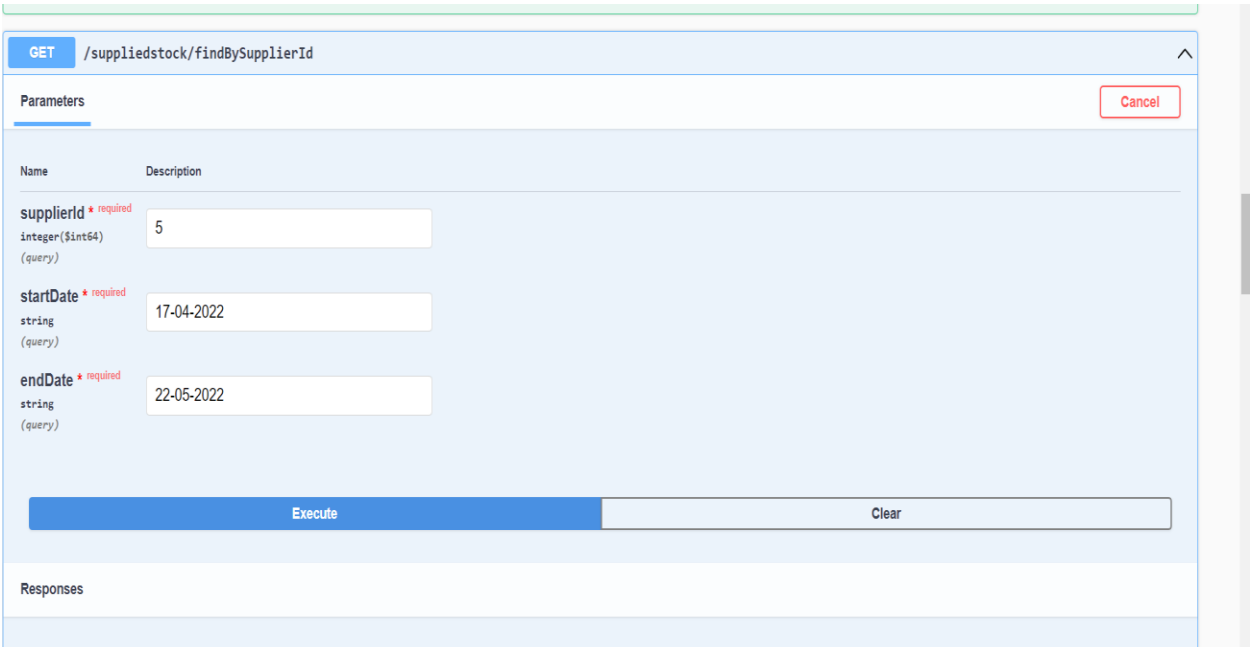


Figure 7.9

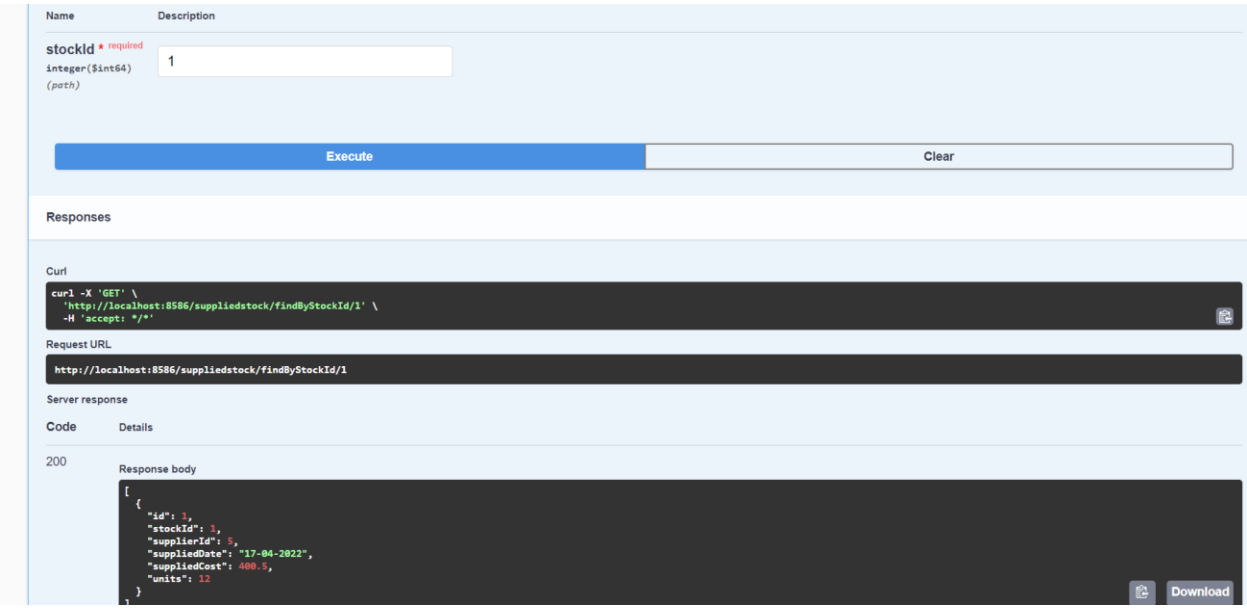

Name
Description
stockId * required
integer($int64)
(path)
1
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:8586/suppliedstock/findByStockId/1' \
-H 'accept: */*'
Request URL
http://localhost:8586/suppliedstock/findByStockId/1
Server response
Code
Details
200
Response body
[
{
"id": 1,
"stockId": 1,
"supplierId": 5,
"suppliedDate": "17-04-2022",
"suppliedCost": 400.5,
"units": 12
}
]
Download

Figure 7.10

CHAPTER 8
LIMITATION

8.1 Limitation

An API is a set of definitions and protocols for building and integrating application software. It's sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response).

In this REST API currently we're performing only CRUD operations on these microservices i.e., this microservice will provide users to perform GET, PUT, POST, DELETE operations on the database through the defined controllers. Which means a user can use this API for getting all the details of the existing products, or can add a new product and details related to the product, or update the details of the existing product, or can delete an existing product. Beside these we can also perform various operations using microservices.
Other than this some service which can be added on this API are PATCH method for making partial changes to an existing resource, HEAD method for a response identical to that of a GET request, but without the response body, LINK method to add meta information (Object head information) to an object, without touching the object's content, and many more methods.

8.2 Future Scope

REST overcomes many of the disadvantages of SOAP, such as the need for clients to know the operation semantics as a pre-requisite for its use, or the use of different ports for different types of notifications. In addition, REST can handle many resources, while SOAP needs many operations to accomplish that.

- It is usually simple to build and adapt.

- Low use of resources.
- Process instances are created explicitly.
- With the initial URI, the client does not require routing information.
- Clients can have a generic ‘listener’ interface for notifications.

While SOAP focuses on the design of distributed applications, REST does so with scalability and large-scale performance for distributed hypermedia systems.

- **Scalability**. This protocol stands out due to its scalability. Thanks to the separation between client and server, a product may be scaled by a development team without much difficulty.
- **Flexibility and portability.** With the indispensable requirement for data from one of the requests to be properly sent, it is possible to perform a migration from one server to another or carry out changes on the database at any time. Front and back can therefore be hosted on different servers, which is a significant management advantage.
- **Independence.** With the separation between client and server, the protocol makes it easy for developments across a project to take place independently. In addition, the REST API adapts at all times to the working syntax and platform. This offers the opportunity to use multiple environments while developing.

8.3 Future Enhancement

Primarily, this API provides users to perform CRUD operations or microservices on the database. So, in future other operations or microservices can be added on. Some of the enhancements which can be applied are as follows:
PATCH method for making partial changes to an existing resource
HEAD method for a response identical to that of a GET request, but without the response body

LINK method to add meta information (Object head information) to an object, without touching the object's content

UNLINK method to decouple the item, to remove a link, to delink

PURGE, COPY, HEAD, LOCK, UNLOCK, PROPFIND, etc microservices can be implemented in the existing REST API, which will eventually enhance the performance of the API and become more useful for the users.

CHAPTER 9

CONCLUSION

To conclude, this REST API is a type of data transfer that is built upon the architecture of the HTTP protocol. It allows you to easily send and retrieve data between two different services using JSON.

When using this REST API, you will get to do all the following four actions:

- Create (POST) - The create function allows users to create a new record in the database.

- Read (GET) - The read function is similar to a search function. It allows users to search and retrieve specific records in the table and read their values.

- Update (PUT) – The update function is used to update the records that already existing records in the database.

- Delete (DELETE) – The delete function is used to delete/remove any specific record from the database.

When you build an application that requires a JavaScript heavy front-end, or integration with a smartphone application, using a RESTful architecture becomes a necessity because it allows you to transfer data from the API to the client.

CHAPTER 10

REFERENCES

JAVA Programming by Rajiv Chopra - https://elib4u.ipublishcentral.com/product/java-programming-50073021

Introduction to Software Design with Java by Martin P. Robillard - https://link.springer.com/book/10.1007/978-3-030-24094-3

Fundamentals of Java Programming by Mitsunori Ogihara - https://link.springer.com/book/10.1007/978-3-319-89491-1

Java in Two Semesters by Quentin Charatan, Aaron Kans - https://link.springer.com/book/10.1007/978-3-319-99420-8

Learning Java with Games by Chong-wei Xu - https://link.springer.com/book/10.1007/978-3-319-72886-5

Computing and Software Science by Bernhard Steffen, Gerhard Woeginger - https://link.springer.com/book/10.1007/978-3-319-91908-9

Concise Guide to Software Testing by Gerard O'Regan - https://link.springer.com/book/10.1007/978-3-030-28494-7

Dependable Software Engineering. Theories, Tools, and Applications by Nan Guan, Joost-Pieter Katoen, Jun Sun - https://link.springer.com/book/10.1007/978-3-030-35540-1

Human-Centered Software Engineering by Cristian Bogdan, Kati Kuusinen, Marta Kristín Lárusdóttir, Philippe Palanque, Marco Winckler - https://link.springer.com/book/10.1007/978-3-030-05909-5

Information and Software Technologies by Robertas Damaševičius, Giedrė Vasiljevienė - https://link.springer.com/book/10.1007/978-3-030-30275-7

Model-Driven Engineering and Software Development
By Luís Ferreira Pires, Slimane Hammoudi, Bran Selic - https://link.springer.com/book/10.1007/978-3-319-94764-8

New Opportunities for Software Reuse by Rafael Capilla, Barbara Gallina, Carlos Cetina - https://link.springer.com/book/10.1007/978-3-319-90421-4

Requirements Engineering: Foundation for Software Quality by Erik Kamsties, Jennifer Horkoff, Fabiano Dalpiaz - https://link.springer.com/book/10.1007/978-3-319-77243-1

Software Analysis, Testing, and Evolution by Lei Bu, Yingfei Xiong - https://link.springer.com/book/10.1007/978-3-030-04272-1

Software Failure Investigation by Jan Eloff, Madeleine Bihina Bella - https://link.springer.com/book/10.1007/978-3-319-61334-5