REST API PROJECT WITH SPRING JPA AND POSTMAN REPORT

on

“ Beauty Product Store Application”

Submitted by

EBEON1221521231	**VENNILA RAMESH**
EBEON1221457628	**ESHWARI S V**
EBEON1021447885	**SREEDEVI PALLENI**

Under the Guidance of

Master Trainer Indrakka Mali
Edubridge

Introduction

As ecommerce is growing much faster than retail, many people opt for online shopping and some of the young generation started to set up an e-commerce business. This project is motivated to make contribution to the consumers by providing them a convenient way to shop online with simple steps, capable of for easy browsing.

In addition, the purpose of this project is to motivate both sellers and buyers to use this application for purchasing their beauty and skincare products.

With exponential increase in business, it becomes a tedious task to maintain records of all products made available to different customers. Manual working of the system would not be beneficial for either the organization or the working individual. So, a database management system in the form of a API needs to be developed so as to perform all the manual tasks of beauty product store database through means of computers.

1.1 Problem Statement

The objective is to develop a database management system such that:

- The system maintains details of all products such as Name, ID, Price, Username,Password and Role.With store ID as foreign key referencing the store table.
- The system maintains details of stores provided with its Name, address, ID.
- The Restful CRUD API maintains details of both store and product details.
- The apis are used to create, retrieve, update and delete a store, and then tested using postman.
- The apis are used to create, retrieve, update and delete a product, and then tested using postman.
- The system maintains all the records in the store(s).

Back End Design

2.1 Database Design

Database design refers to the process of organization of data. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database accordingly. The four main types of databases are text databases, desktop database programs, relational database management systems (RDBMS) and NoSQL and object-oriented databases.

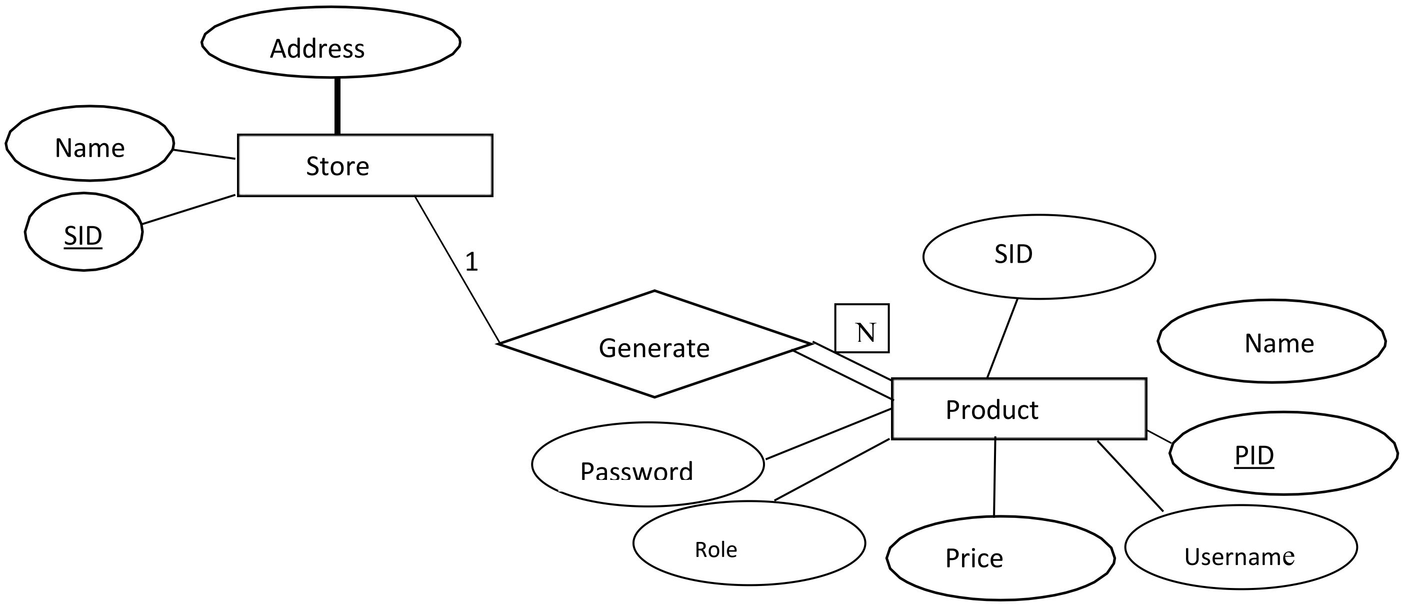
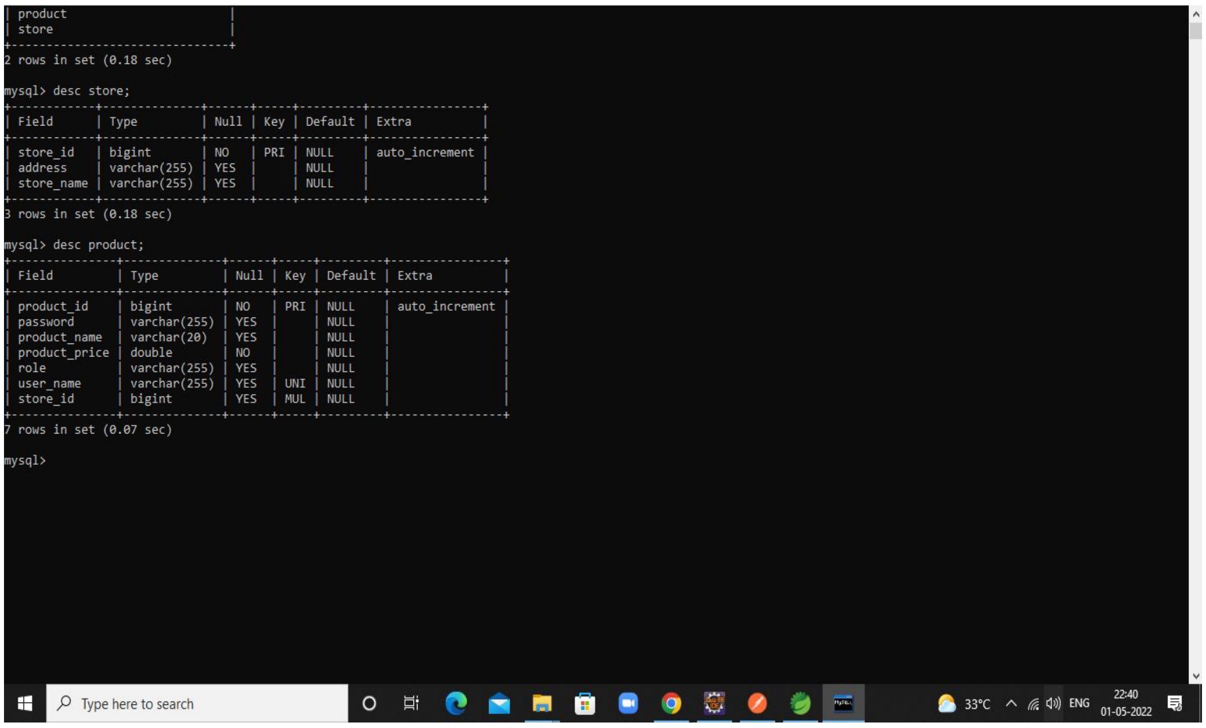


Figure 1.1 – ER Diagram

It can be observed from Figure 1 that the main entities in the system are products available in the Store, and the Store itself. It shows two kinds of relationships between the parent table(Store) and the child table(Products).

A) One-to-Many:Where one store can have many products.
B) Many-to-One:Where many products can be added to one store.

Tables in the Database



```
| product                        |
| store                          |
+--------------------------------+
2 rows in set (0.18 sec)

mysql> desc store;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| store_id   | bigint       | NO   | PRI | NULL    | auto_increment |
| address    | varchar(255) | YES  |     | NULL    |                |
| store_name | varchar(255) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
3 rows in set (0.18 sec)

mysql> desc product;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| product_id    | bigint       | NO   | PRI | NULL    | auto_increment |
| password      | varchar(255) | YES  |     | NULL    |                |
| product_name  | varchar(20)  | YES  |     | NULL    |                |
| product_price | double       | NO   |     | NULL    |                |
| role          | varchar(255) | YES  |     | NULL    |                |
| user_name     | varchar(255) | YES  | UNI | NULL    |                |
| store_id      | bigint       | YES  | MUL | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
7 rows in set (0.07 sec)

mysql>
```

2.2 Configuring MySQL Database

Spring Boot auto-configures a DataSource if spring-data-jpa is in the classpath by reading the database configuration from application.properties file.
Open application.properties file and add the following properties to it.

```
## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)

server.port = 8888

spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver

spring.datasource.url = jdbc:mysql://localhost:3306/beautyproductstore

 spring.datasource.username = root spring.datasource.password = root123

spring.jpa.show-sql = true spring.jpa.generate-ddl= true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto=create
```

2.3 Database Connectivity

You will need to create a database named beautyproductstore in MySQL, and change the spring.datasource.username & spring.datasource.password properties as per your MySQL installation.
In the above properties file, the last two properties are for hibernate. Spring Boot uses Hibernate as the default JPA implementation.
The property spring.jpa.hibernate.ddl-auto is used for database initialization. We've used the value “update” for this property.
It does two things -

- When you define a entity model, a table will automatically be created in the database and the fields of the entity model will be mapped to the corresponding columns in the table.
- Any change to the entity model will also trigger an update to the table. For example, If you change the name or type of a field, or add another field to the model, then all these

changes will be reflected in the mapped table as well.

Using update for spring.jpa.hibernate.ddl-auto property is fine for development. But, For production, You should keep the value of this property to “validate”, and use a database migration tool like Flyway for managing changes in the database schema.

Store Entity

Store_ID: Primary Key with Auto Increment.

Store_Name: The name of the Store. (NOT NULL field)

Store_Address: The address of the Store . (NOT NULL field)

Product Entity

Product_ID: Primary Key with Auto Increment.

Product_Name: The name of the Store. (NOT NULL field)

Product_Price: The address of the Store . (NOT NULL field)

Store_id: The Store_ID of the Store model referenced as foreign key in the Product model.(NOT NULL field)

Username: The name of the user(Unique field).

Password: The password of the user. (NOT NULL field)

Role: The role of the user . (NOT NULL field)

Code Design

3.1 Client Side Processing

Client side programming includes any coding or computation or effects or animation or any sort of interaction your website performs with the user via **browser**.

Postman is an API client that makes it easy for developers to create, share, test and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses. The result - more efficient and less tedious work.

3.2 Association Using Hibernate

Hibernate is one of the popular implementations of JPA.
@ManyToOne - Association for Many products in One store.
@JoinColumn(name = "storeId") - Joins the storeID from store table,in product table.
@JsonIgnore - To avoid infinite display of records.
Hibernate understands the mappings that we add between objects and tables. It ensures that data is stored/retrieved from the database based on the mappings.
Hibernate also provides additional features on top of JPA.
.

3.3 Exception Handling

@ControllerAdvice - It allows to handle exceptions across whole application in one global handling component.
@ResponseStatus - To mark a method or an exception class,with a status code and reason that should be a returned.
@ExceptionHandler - Handle the specific exceptions and sending the custom responses to the client.

3.4 Security

@Configuration - indicates that a class declares one or more @Bean methods and may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime.

@EnableWebSecurity - is annotated at class level with @Configuration annotation to enable web securities in our application defined by WebSecurityConfigurer implementations.

Code Implementation

Store

```
package com.example.demo.entitiy;

import java.util.ArrayList;

import java.util.List;


import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.validation.constraints.NotEmpty;



@Entity
public class Store {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long storeId;

	@NotEmpty(message="Name should never be empty")
	private String storeName;
	//@Column(unique = true)
	private String address;
	 @OneToMany(cascade = CascadeType.ALL)
	 @JoinColumn(name="storeId")
	private List<Product> productlist=new ArrayList<Product>();


	public Store() {
		super();
	}
	public Store(Long storeId, String storeName, String address, List<Product> productlist) {
		super();
		this.storeId = storeId;
		this.storeName = storeName;
		this.address = address;
		this.productlist = productlist;
	}
	public Long getStoreId() {
		return storeId;
	}
	public void setStoreId(Long storeId) {
		this.storeId = storeId;
		}
		public String getStoreName() {
			return storeName;
		}
		public void setStoreName(String storeName) {
			this.storeName = storeName;
		}
		public String getAddress() {
			return address;
		}
		public void setAddress(String address) {
			this.address = address;
		}
		public List<Product> getProductlist() {
			return productlist;
		}
		public void setProductlist(List<Product> productlist) {
			this.productlist = productlist;
		}
		@Override
		public String toString() {
			return "Store [storeId=" + storeId + ", storeName=" + storeName + ", address=" +
address + ", productlist="
					+ productlist + "]";
		}

	}
```

Product

```
package com.example.demo.entitiy;




import javax.persistence.Column;
import javax.persistence.Entity;

import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.Length;
import com.fasterxml.jackson.annotation.JsonIgnore;
@Entity
public class Product {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long productId;

        @NotBlank(message="productname should never be empty")
        @Length(min=2,max=20,message="size should be in range")
        private String productName;

        @NotNull(message="productPrice should never be empty")
        private double productPrice;
         @JsonIgnore
         @ManyToOne
         @JoinColumn(name = "storeId")
         private Store store;
        private String password;
   @Column(unique = true)
        private String username;
   private String role;
        public Product() {
                super();
        }
        public Product(Long productId,
                        @NotBlank(message = "productname should never be empty")
@Length(min = 5, max = 20, message = "size should be in range") String productName,
                        @NotNull(message = "productPrice should never be empty") double
productPrice, Store store,
                        String strPassword, String strusername, String role) {
                super();
                this.productId = productId;
                this.productName = productName;
                this.productPrice = productPrice;
                this.store = store;
                this.password = password;
                this.username = username;
                this.role = role;
        }
        public Long getProductId() {
                return productId;
        }
        public void setProductId(Long productId) {
                this.productId = productId;
        }
        public String getProductName() {
                return productName;
        }
        public void setProductName(String productName) {
                this.productName = productName;
        }
        public double getProductPrice() {
                return productPrice;
```

```
	}
	public void setProductPrice(double productPrice) {
		this.productPrice = productPrice;
	}
	public Store getStore() {
		return store;
	}
	public void setStore(Store store) {
		this.store = store;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getRole() {
		return role;
	}
	public void setRole(String role) {
		this.role = role;
	}
	@Override
	public String toString() {
		return "Product [productId=" + productId + ", productName=" + productName + ", 
productPrice=" + productPrice
				+ ", store=" + store + ", password=" + password + ", username=" + 
username + ", role="
				+ role + "]";
	}
```

```
}
```

ErrorMessage

```
package com.example.demo.entitiy;

import org.springframework.http.HttpStatus;

public class ErrorMessage {
	private HttpStatus status;
	private String messagee;
	public ErrorMessage() {
		super();
	}
	public ErrorMessage(HttpStatus status, String messagee) {
		super();
		this.status = status;
		this.messagee = messagee;
	}

	public HttpStatus getStatus() {
		return status;
	}
	public void setStatus(HttpStatus status) {
		this.status = status;
	}
	public String getMessagee() {
		return messagee;
	}
	public void setMessage(String messagee) {
		this.messagee = messagee;
	}
	@Override
	public String toString() {
		return "ErrorMessage [status=" + status + ", message=" + messagee + "]";
	}

}
```
ProductNotFoundException

```
package com.example.demo.error;

public class ProductNotFoundException extends Exception {
	public  ProductNotFoundException(String string) {
		super(string);
	}
}
```

ResponseEntityExceptionHandler

```
package com.example.demo.error;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.context.request.WebRequest;
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import com.example.demo.entitiy.ErrorMessage;

@ControllerAdvice
@ResponseStatus
public class RestResponseEntityHandler extends ResponseEntityExceptionHandler {
	@ExceptionHandler(StoreNotFoundException.class)
	publicResponseEntity<ErrorMessage>
storeNotFoundException(StoreNotFoundException exception,WebRequest request) {
		ErrorMessagemessagee=new
ErrorMessage(HttpStatus.NOT_FOUND,exception.getMessage());
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(messagee);
    }
          @ExceptionHandler(ProductNotFoundException.class)
          publicResponseEntity<ErrorMessage>
productNotFoundException(ProductNotFoundException exception,WebRequest request) {
ErrorMessage messagee=new ErrorMessage(HttpStatus.NOT_FOUND,exception.getMessage());
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(messagee);
    }

}
```

StoreNotFoundException

```
package com.example.demo.error;

public class StoreNotFoundException extends Exception{
          public   StoreNotFoundException(String string) {
                    super(string);
          }
}
```

```
package com.example.demo.error;

public class StoreNotFoundException extends Exception{
          public   StoreNotFoundException(String string) {
                    super(string);
          }
}
```

ProductRepository

```
package com.example.demo.repository;
import java.util.List;
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.demo.entitiy.Product;

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {

	List<Product> findAll();

	Product save(Product product);

	void deleteById(Long productId);

	Optional<Product> findById(Long productId);



	Product findByproductName(String productname);

	Product findByproductPrice(String price);


	Product findByUsername(String userName);
}
```

StoreRepository

```
package com.example.demo.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.example.demo.entitiy.Store;

@Repository
```

```
public interface StoreRepository extends JpaRepository<Store, Long> {

	Store findBystoreName(String storeName);

	Store findByAddress(String address);

}
```

SecurityConfig

```
package com.example.demo.securityconfig;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapt
er;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.factory.PasswordEncoderFactories;
import org.springframework.security.crypto.password.PasswordEncoder;

import com.example.demo.service.StoreService;
@Configuration
@EnableWebSecurity
public class SecurityConfig  extends WebSecurityConfigurerAdapter{
	@Autowired
	private UserDetailsService     userDetailsService;

		@Bean
	 AuthenticationProvider authenticationProvider() {
```

```
                    DaoAuthenticationProvider provider=new DaoAuthenticationProvider();
                     provider.setUserDetailsService(userDetailsService);
                    provider.setPasswordEncoder(new BCryptPasswordEncoder());
                    return provider;
          }
          protected void configure(HttpSecurity http)throws Exception{

                    http

                    .csrf().disable();

          }
          }
```

CustomeUserDetails

```
package com.example.demo.service;

import java.util.Collection;
import java.util.Collections;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import com.example.demo.entitiy.Product;


public class CustomeUserDetails implements UserDetails {
Product product;
public CustomeUserDetails(Product product) {
          super();
          this.product=product;
}

          @Override
```

```
public Collection<? extends GrantedAuthority> getAuthorities() {
        //String  str=product.getStore().toString();
        return Collections.singleton(new SimpleGrantedAuthority(product.getRole()) );
}

@Override
public String getPassword() {

        return product.getPassword();
        }

@Override
public String getUsername() {

        return product.getUsername();

}

@Override
public boolean isAccountNonExpired() {

        return true;
}

@Override
public boolean isAccountNonLocked() {

        return true;
}

@Override
public boolean isCredentialsNonExpired() {

        return true;
}
```

```
	@Override
	public boolean isEnabled() {

		return true;
	}

}
```

ProductService

```
package com.example.demo.service;

import java.util.List;

import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

import com.example.demo.entitiy.Product;
import com.example.demo.entitiy.Store;
import com.example.demo.error.ProductNotFoundException;
import com.example.demo.error.StoreNotFoundException;

public interface ProductService {

	List<Product> getproducts();

	Product saveProducts(Product product, Long storeId) throws StoreNotFoundException;

	void deleteProductsById(Long productId);

	Product updateProductsById (Long storeId ,Product product,Long productId) throws
ProductNotFoundException;

	Product getProductsById(Long productId) throws ProductNotFoundException;
	Product getProductByName(String productName) throws ProductNotFoundException;

	Product getProductByPrice(String price) throws ProductNotFoundException;

	//UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;

}
```

StoreService

```
package com.example.demo.service;

import java.util.List;

import com.example.demo.entitiy.Store;
import com.example.demo.error.StoreNotFoundException;

public interface StoreService {

	public List<Store> getStores() throws StoreNotFoundException;

	public Store saveStores(Store store) throws StoreNotFoundException;

	public void deleteStoresById(Long storeId);

	public Store updateStoresById(Long storeId, Store store) throws StoreNotFoundException;

	public Store getStoresById(Long storeId) throws StoreNotFoundException;

	public Store getStoresByAddress(String address) throws StoreNotFoundException;

	public Store getStoresByName(String storeName) throws StoreNotFoundException;
}
```

ProductServiceImplementation

```
package com.example.demo.service;

import java.util.List;
import java.util.Objects;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import com.example.demo.entitiy.Product;
import com.example.demo.entitiy.Store;
import com.example.demo.error.ProductNotFoundException;
import com.example.demo.error.StoreNotFoundException;
import com.example.demo.repository.ProductRepository;
import com.example.demo.repository.StoreRepository;

@Service
public class ProductServiceImpl implements ProductService,UserDetailsService{
@Autowired
ProductRepository productRepository;
@Autowired
StoreRepository storeRepository;
//@Autowired
//PasswordEncoder encoder;
@Override
```

```
public UserDetails loadUserByUsername(String userName) throws
UsernameNotFoundException {
Product e = productRepository.findByUsername(userName);
if(e==null)
        throw new UsernameNotFoundException("User not found");
        return new CustomeUserDetails(e);
}
@Override
public List<Product> getproducts() {

                return productRepository.findAll() ;
        }

        @Override
        public Product saveProducts(Product product,Long storeId) throws
StoreNotFoundException {

                //return productRepository.save(product);
                if(!storeRepository.existsById(storeId)) {
                        throw new StoreNotFoundException("store not found");
                }
                else {
                        Store s =storeRepository.findById(storeId).get();
                        product.setStore(s);
                        s.getProductlist().add(product);
                        return productRepository.save(product);

                }




        }

        @Override
        public void deleteProductsById(Long productId) {
                productRepository.deleteById(productId);
```

```
	}

	@Override
	public Product updateProductsById(Long storeId,Product product,Long productId)
throws ProductNotFoundException {
		Optional<Product> p=productRepository.findById(productId);
		if(p.isPresent()) {
			Store s=storeRepository.findById(storeId).get();
			product.setStore(s);
			Product pDB=productRepository.findById(productId).get();
		if(Objects.nonNull(product.getProductName())
&& !"".equalsIgnoreCase(product.getProductName())) {
		pDB.setProductName(product.getProductName());
		}
		if(Objects.nonNull(product.getProductPrice()) ) {
			pDB.setProductPrice(product.getProductPrice());
		}
		if(Objects.nonNull(product.getStore()) ) {
			pDB.setStore(product.getStore());
		}
		if(Objects.nonNull(product.getUsername())
&& !"".equalsIgnoreCase(product.getUsername())) {
			pDB.setUsername(product.getUsername());
			}
		if(Objects.nonNull(product.getPassword())
&& !"".equalsIgnoreCase(product.getPassword())) {
			pDB.setPassword(product.getPassword());
			}
		if(Objects.nonNull(product.getRole())
&& !"".equalsIgnoreCase(product.getRole())) {
			pDB.setRole(product.getRole());
			}
		return productRepository.save(pDB);
		}
		else throw new ProductNotFoundException("Product Id Does Not Exist");
```

```
                }

        @Override
        public Product getProductsById(Long productId) throws ProductNotFoundException {
                //return productRepository.findById(productId).get();
                Optional<Product> pid=(productRepository.findById(productId));
                if(!pid.isPresent()) {
                        throw new ProductNotFoundException("Product Id does not exist");
                }
                else return pid.get();

        }

        @Override
        public Product getProductByPrice(String price) throws ProductNotFoundException {
                Optional<Product>
pprice=Optional.ofNullable(productRepository.findByproductPrice(price));
                if(!pprice.isPresent()) {
                        throw new ProductNotFoundException("Product price does not exist");
                }
                else return pprice.get();

                //return productRepository.findByproductPrice(price);
        }

        @Override
        public Product getProductByName(String productName) throws
ProductNotFoundException {
                Optional<Product>
pname=Optional.ofNullable(productRepository.findByproductName(productName));

                //return productRepository.findByproductName(productName);
                if(!pname.isPresent()) {
                        throw new ProductNotFoundException("Product name does not exist");
                }
                else return pname.get();
```

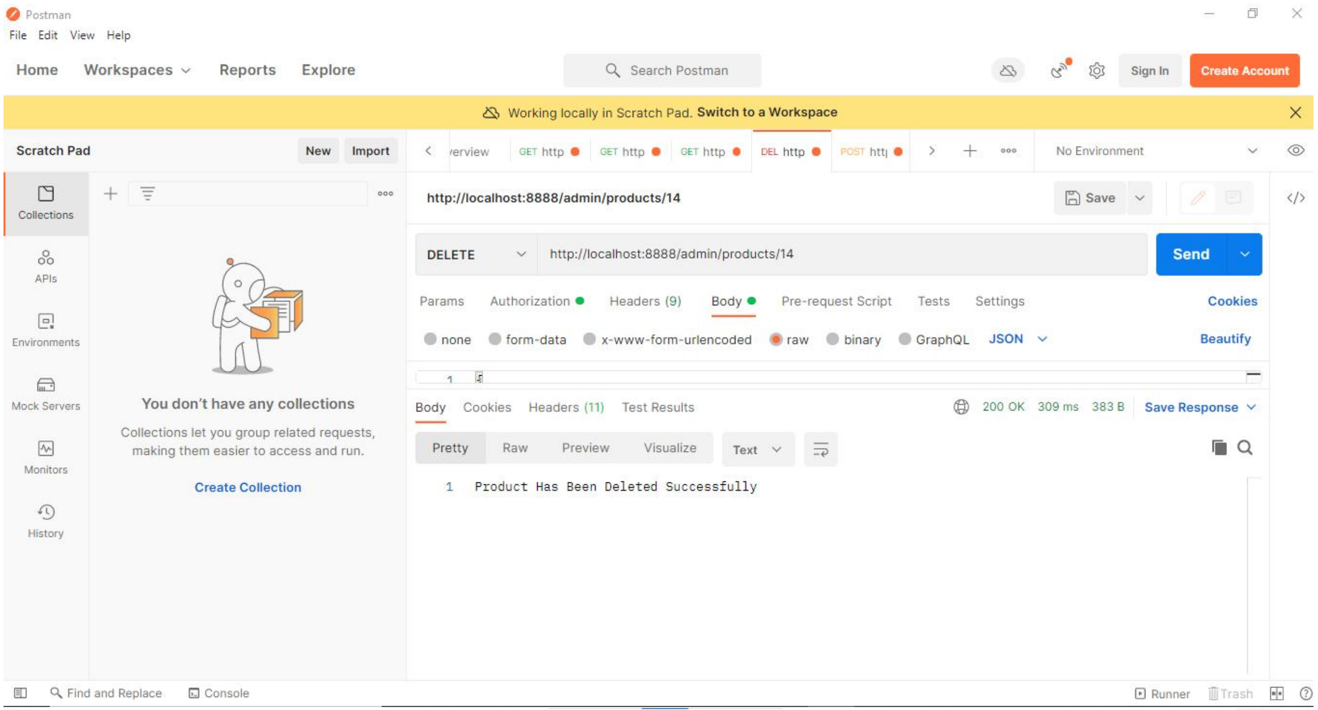
```
	}
}
```

StoreServiceImplementation

```
package com.example.demo.service;

import java.util.List;
import java.util.Objects;
import java.util.Optional;

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

import com.example.demo.entitiy.Store;
import com.example.demo.error.StoreNotFoundException;
import com.example.demo.repository.StoreRepository;
@Service
public class StoreServiceImpl implements StoreService {
@Autowired
StoreRepository storeRepository;

@Override
public List<Store> getStores() throws StoreNotFoundException {

	return storeRepository.findAll();
//Optional<List<Store>> sl=Optional.ofNullable((storeRepository.findAll()));

	//return storeRepository.save(store);
}

@Override
public Store saveStores(Store store) throws StoreNotFoundException {
	Optional<Store> ss=Optional.ofNullable(storeRepository.save(store));

	return storeRepository.save(store);
```

```
}

@Override
public void deleteStoresById(Long storeId) {
        storeRepository.deleteById(storeId);

}

@Override
public Store updateStoresById(Long storeId, Store store) throws StoreNotFoundException {
        Optional<Store> s=storeRepository.findById(storeId);
        Store sDB=storeRepository.findById(storeId).get();
        if(Objects.nonNull(store.getStoreName())
&& !"".equalsIgnoreCase(store.getStoreName())) {
        sDB.setStoreName(store.getStoreName());
        }
        if(Objects.nonNull(store.getAddress()) && !"".equalsIgnoreCase(store.getAddress())) {
                sDB.setAddress(store.getAddress());
        }
        if(s.isPresent()) {
                return s.get();
        }
        else
                throw new StoreNotFoundException("Updation is not Posiible, Entered valued
already available in store");



        //return storeRepository.save(sDB);
}

@Override
public Store getStoresById(Long storeId) throws StoreNotFoundException {
        //return storeRepository.findById(storeId).get();
        Optional<Store> sid=storeRepository.findById(storeId);
        if(!sid.isPresent()) {
                throw new StoreNotFoundException("Store id does not exist");
        }
        else return sid.get();
}

@Override
public Store getStoresByAddress(String address) throws StoreNotFoundException {
        //return storeRepository.findByaddress(address);
        Optional<Store> sadd=Optional.ofNullable((storeRepository.findByAddress(address)));
        if(!sadd.isPresent()) {
                throw new StoreNotFoundException("Store Address does not exist");
        }
                else return sadd.get();

}

@Override
public Store getStoresByName(String storeName) throws StoreNotFoundException {
        Optional<Store>
sname=Optional.ofNullable(storeRepository.findBystoreName(storeName));
        if(!sname.isPresent()) {
                throw new StoreNotFoundException("Store name does not exist");
        }
        else return sname.get();
}



}
```

Results

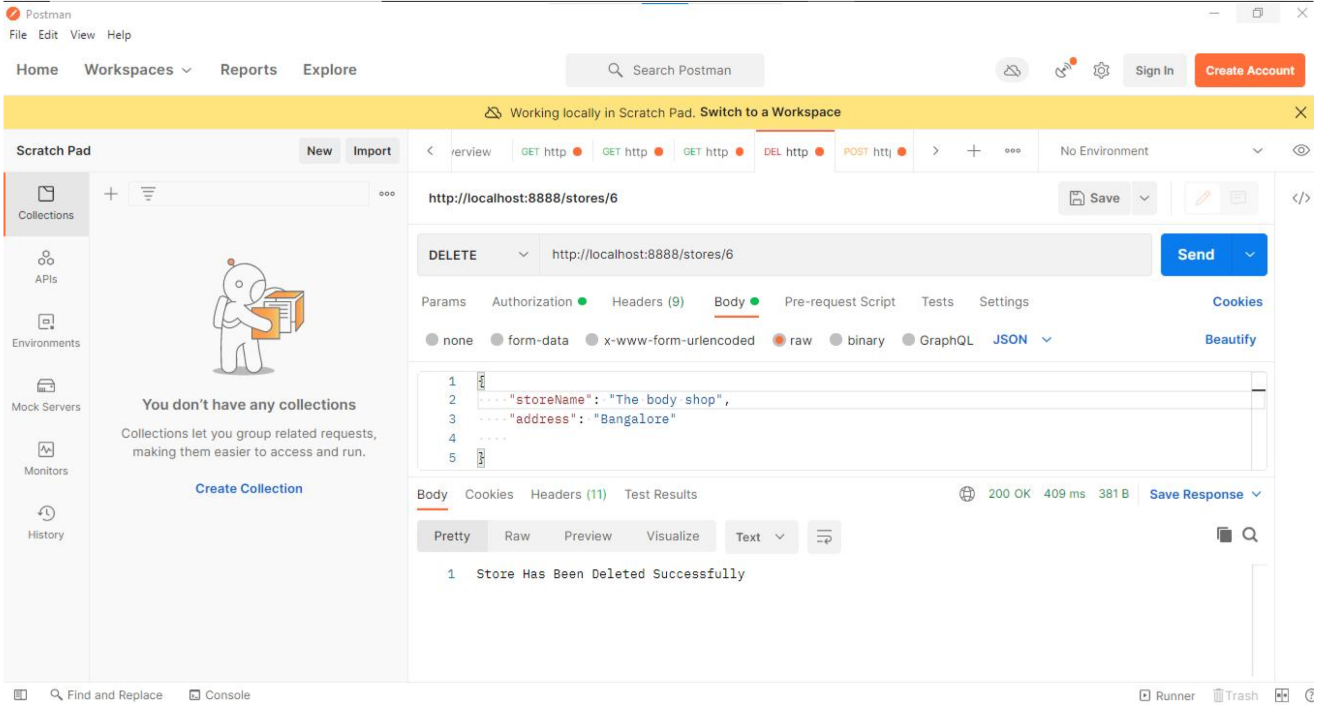

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize Text
Find and Replace
Console
Runner Trash

http://localhost:8888/admin/products/14

DELETE http://localhost:8888/admin/products/14

200 OK 309 ms 383 B

```
Product Has Been Deleted Successfully
```


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize Text
Find and Replace
Console
Runner Trash

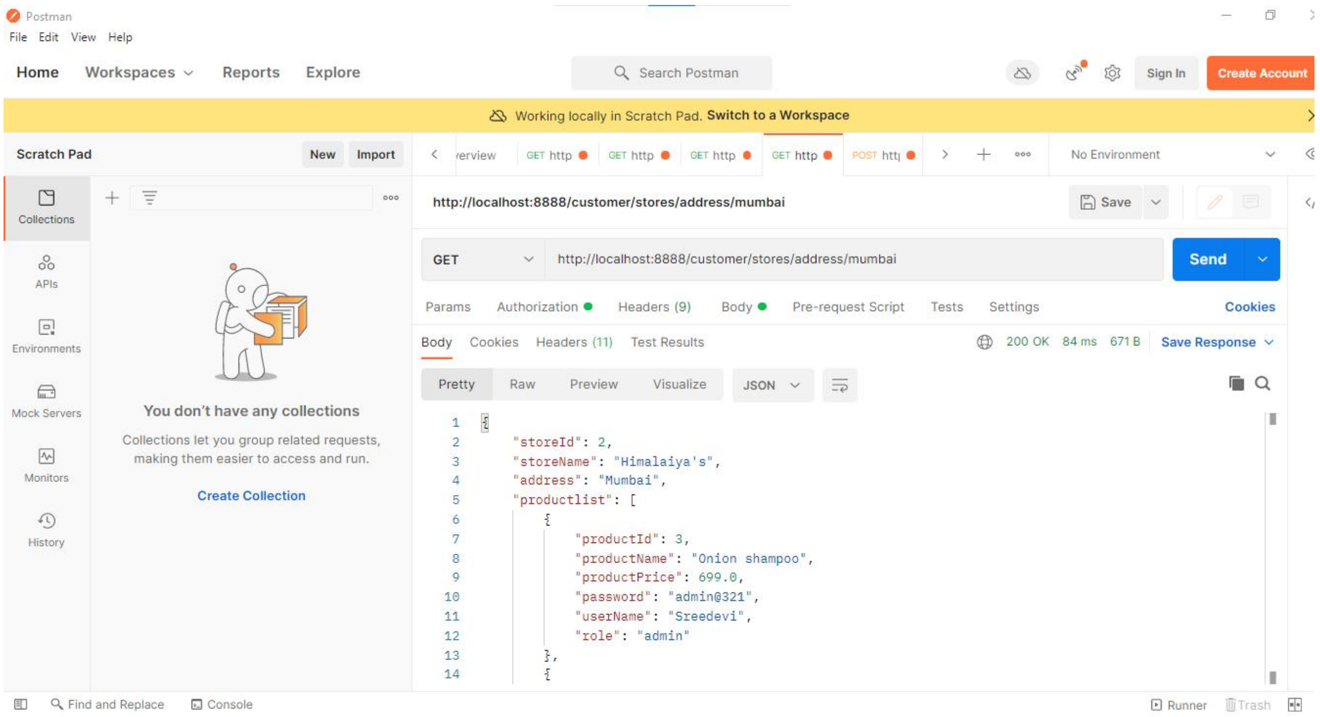
http://localhost:8888/stores/6

DELETE http://localhost:8888/stores/6

```
{
    "storeName": "The body shop",
    "address": "Bangalore"
    
}
```

200 OK 409 ms 381 B

```
Store Has Been Deleted Successfully
```


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad New Import
Collections APIs Environments Mock Servers Monitors History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
Pretty Raw Preview Visualize
Find and Replace Console Runner Trash

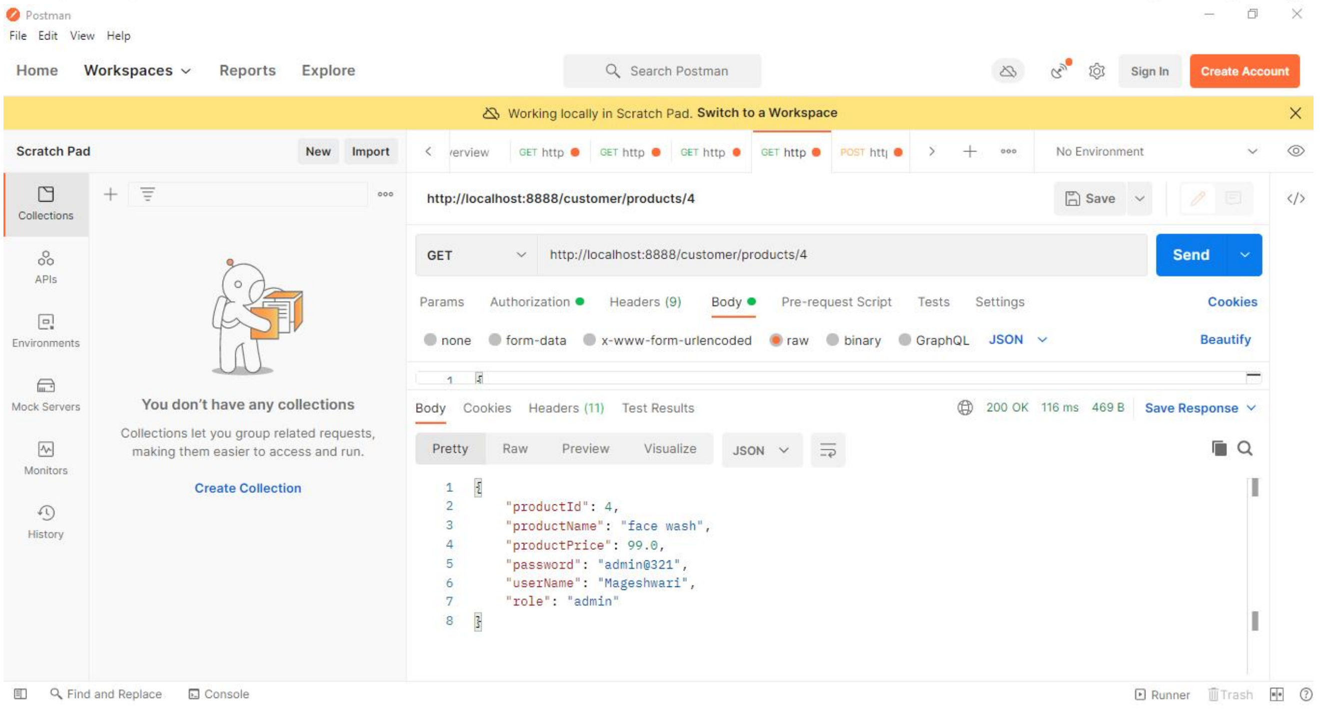
http://localhost:8888/customer/stores/address/mumbai

GET http://localhost:8888/customer/stores/address/mumbai

Body Cookies Headers (11) Test Results — 200 OK 84 ms 671 B Save Response

JSON

```
{
    "storeId": 2,
    "storeName": "Himalaiya's",
    "address": "Mumbai",
    "productlist": [
        {
            "productId": 3,
            "productName": "Onion shampoo",
            "productPrice": 699.0,
            "password": "admin@321",
            "userName": "Sreedevi",
            "role": "admin"
        },
        {
```


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad New Import
Collections APIs Environments Mock Servers Monitors History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL Beautify
Pretty Raw Preview Visualize
Find and Replace Console Runner Trash

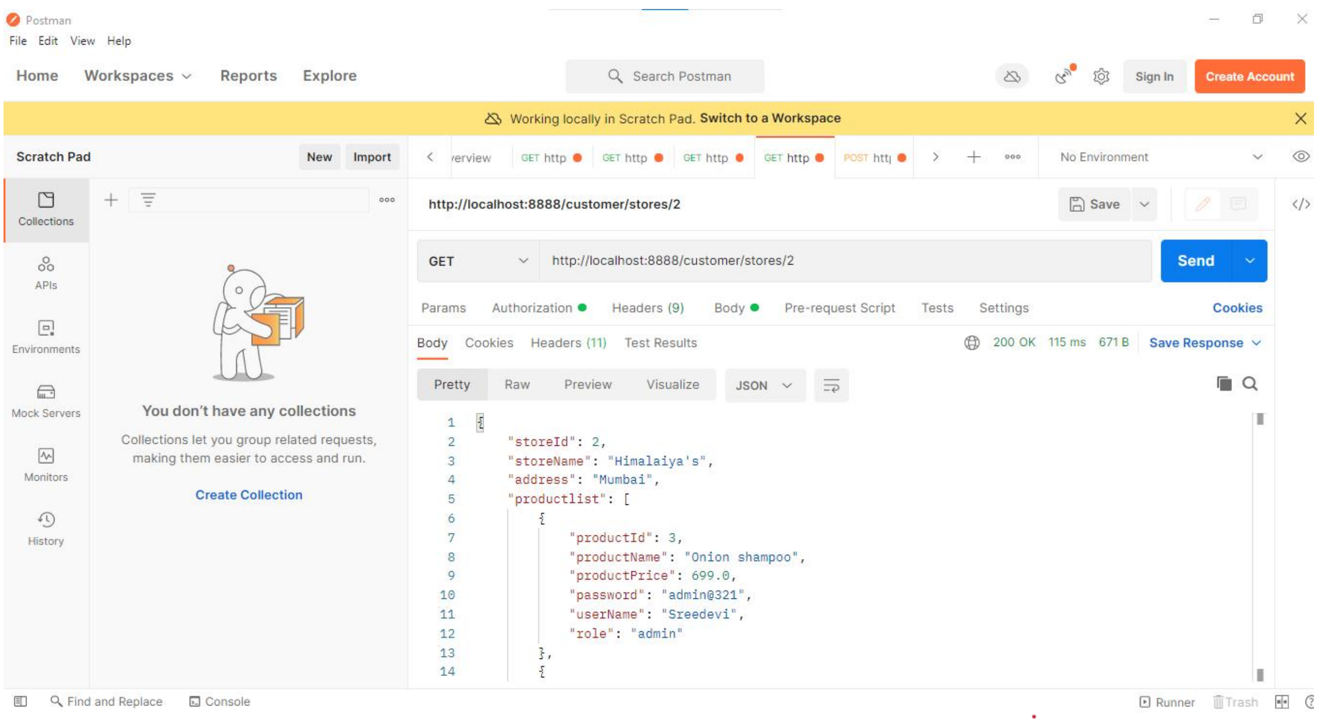
http://localhost:8888/customer/products/4

GET http://localhost:8888/customer/products/4

Body Cookies Headers (11) Test Results — 200 OK 116 ms 469 B Save Response

JSON

```
{
    "productId": 4,
    "productName": "face wash",
    "productPrice": 99.0,
    "password": "admin@321",
    "userName": "Mageshwari",
    "role": "admin"
}
```

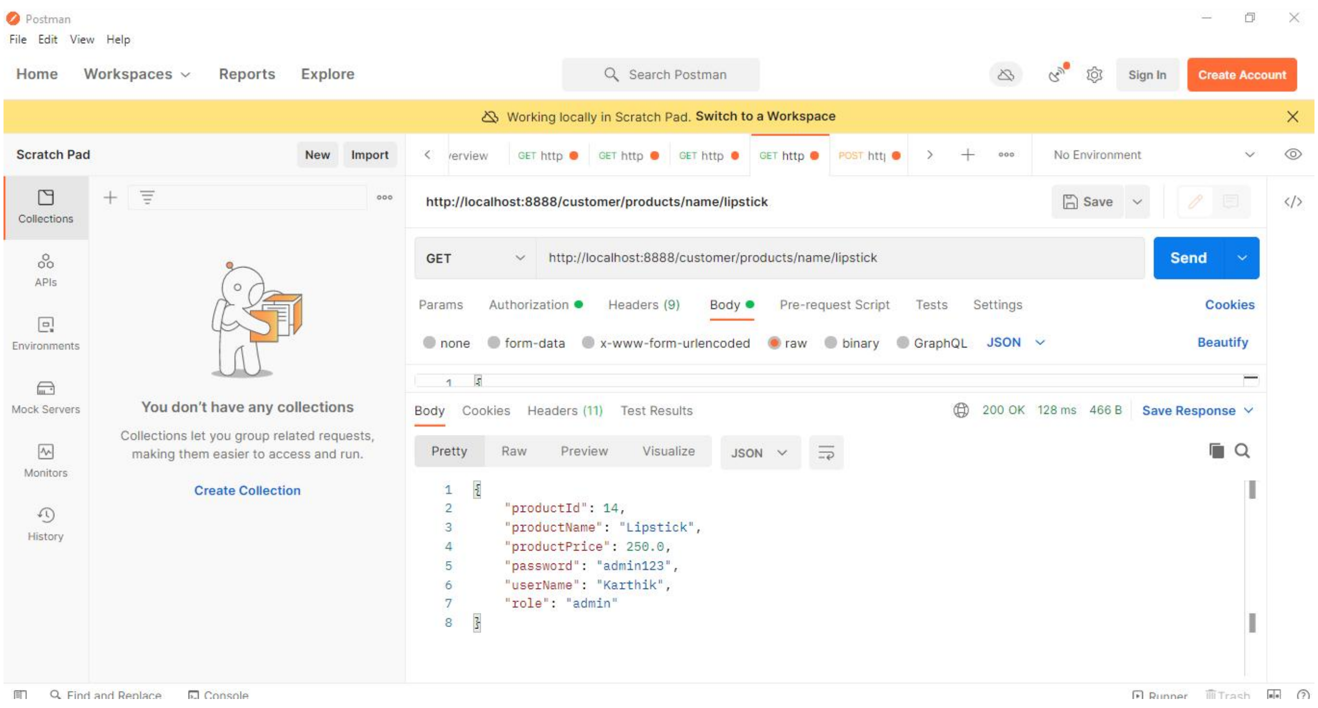

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace
Console
Runner
Trash

http://localhost:8888/customer/stores/2

GET http://localhost:8888/customer/stores/2

200 OK 115 ms 671 B

```
{
    "storeId": 2,
    "storeName": "Himalaiya's",
    "address": "Mumbai",
    "productlist": [
        {
            "productId": 3,
            "productName": "Onion shampoo",
            "productPrice": 699.0,
            "password": "admin@321",
            "userName": "Sreedevi",
            "role": "admin"
        },
        {
```

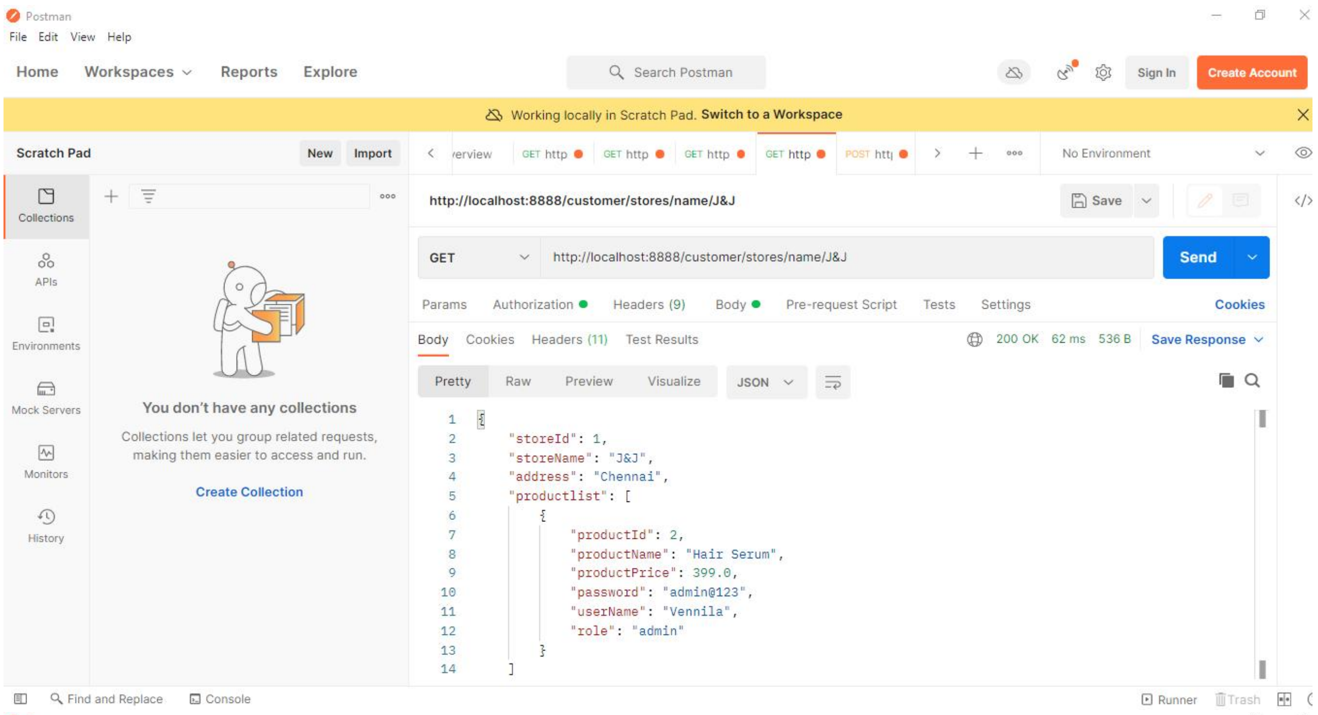

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace
Console
Runner
Trash

http://localhost:8888/customer/products/name/lipstick

GET http://localhost:8888/customer/products/name/lipstick

200 OK 128 ms 466 B

```
{
    "productId": 14,
    "productName": "Lipstick",
    "productPrice": 250.0,
    "password": "admin123",
    "userName": "Karthik",
    "role": "admin"
}
```

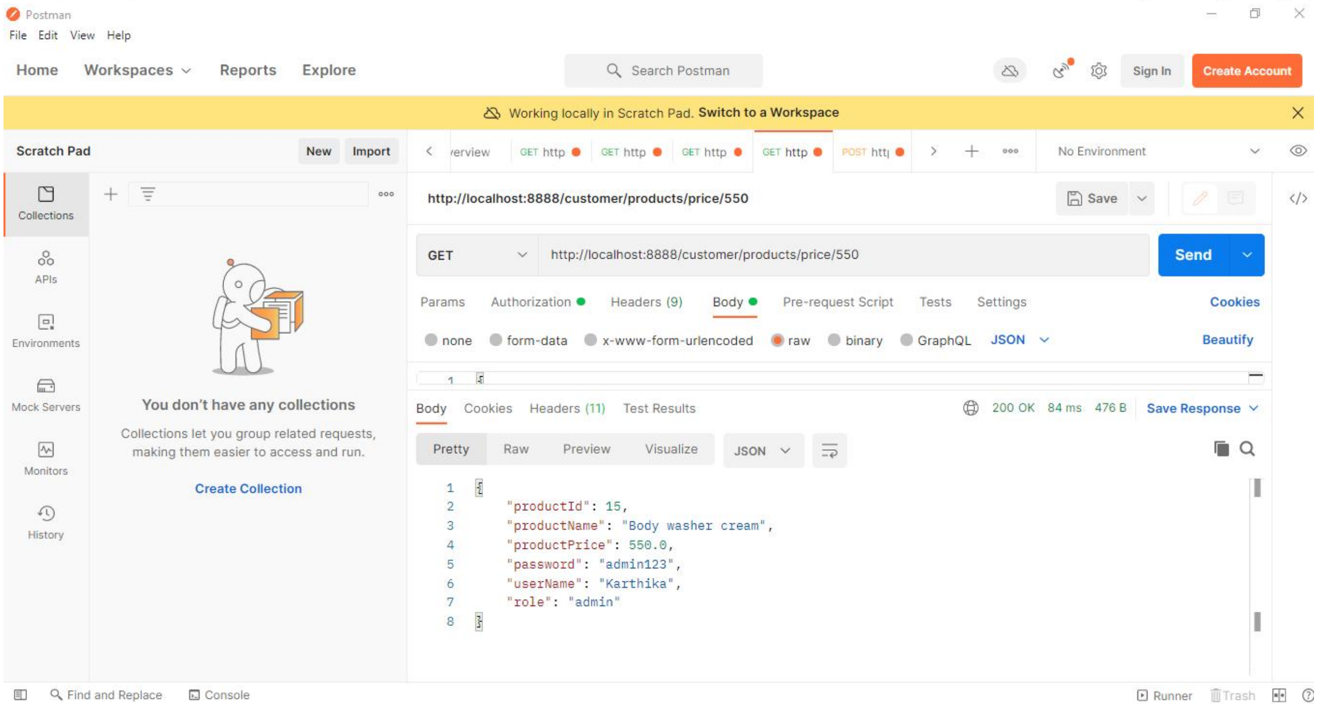

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console
Runner Trash

http://localhost:8888/customer/stores/name/J&J

GET http://localhost:8888/customer/stores/name/J&J

200 OK 62 ms 536 B

```
{
    "storeId": 1,
    "storeName": "J&J",
    "address": "Chennai",
    "productlist": [
        {
            "productId": 2,
            "productName": "Hair Serum",
            "productPrice": 399.0,
            "password": "admin@123",
            "userName": "Vennila",
            "role": "admin"
        }
    ]
```

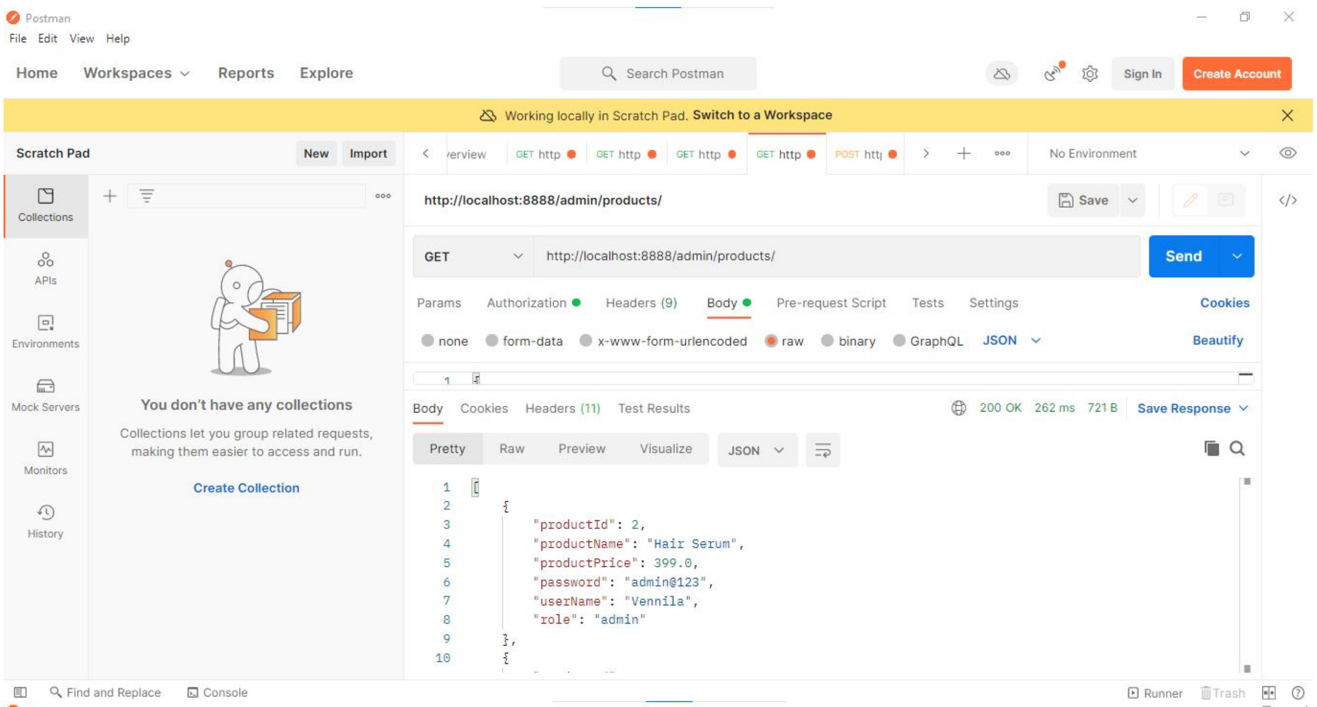

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console
Runner Trash

http://localhost:8888/customer/products/price/550

GET http://localhost:8888/customer/products/price/550

200 OK 84 ms 476 B

```
{
    "productId": 15,
    "productName": "Body washer cream",
    "productPrice": 550.0,
    "password": "admin123",
    "userName": "Karthika",
    "role": "admin"
}
```

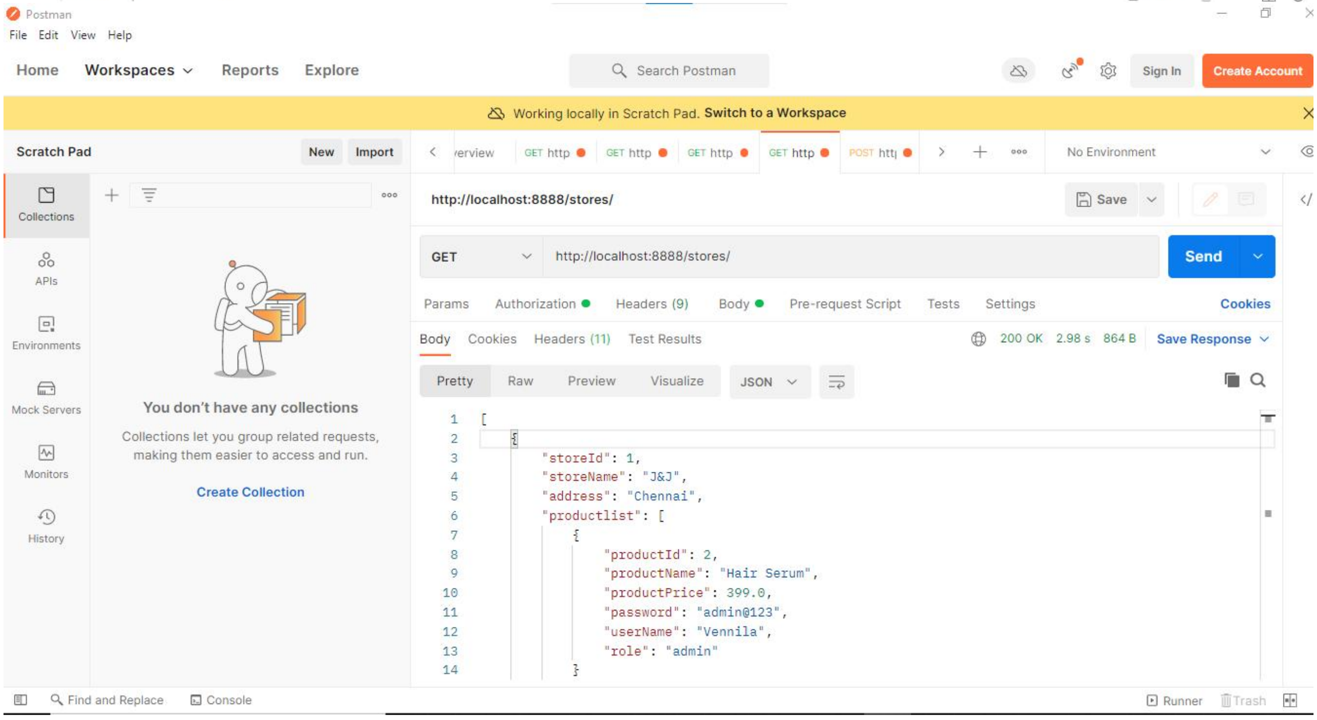

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console Runner Trash

http://localhost:8888/admin/products/

GET http://localhost:8888/admin/products/

200 OK 262 ms 721 B

```
[
    {
        "productId": 2,
        "productName": "Hair Serum",
        "productPrice": 399.0,
        "password": "admin@123",
        "userName": "Vennila",
        "role": "admin"
    },
    {
```

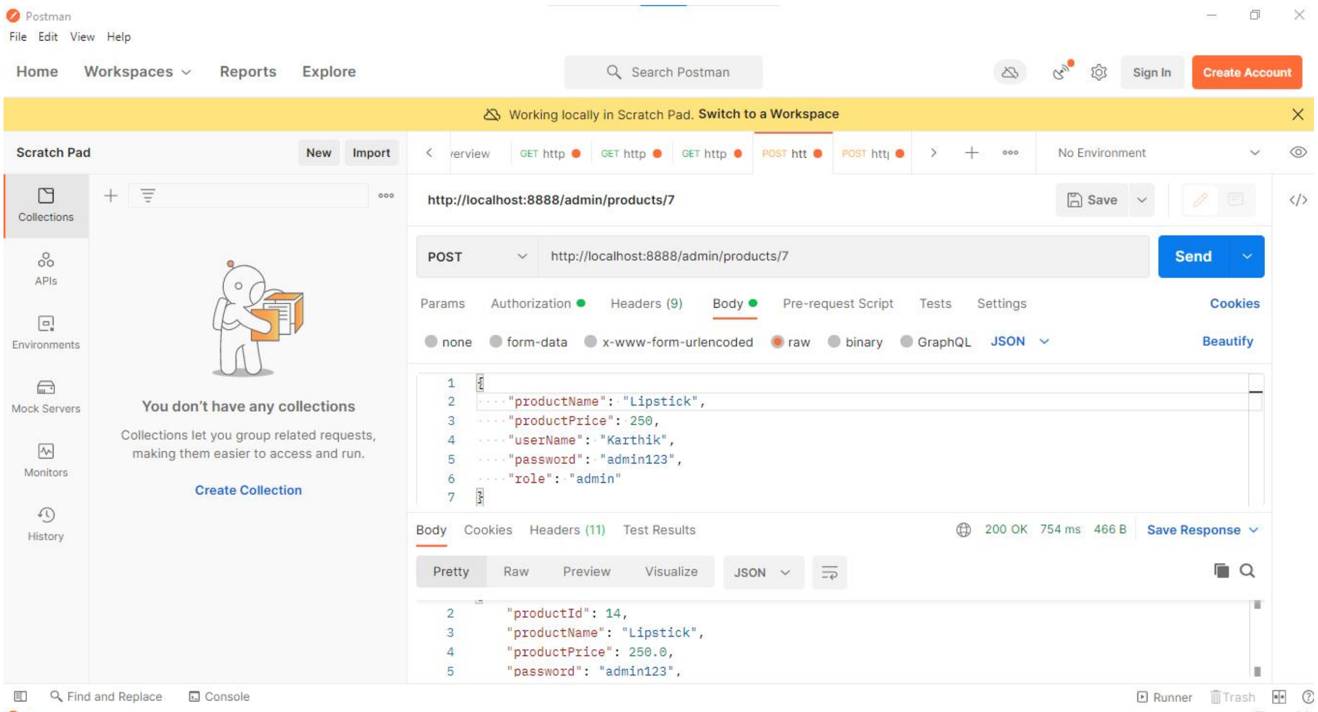

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console Runner Trash

http://localhost:8888/stores/

GET http://localhost:8888/stores/

200 OK 2.98 s 864 B

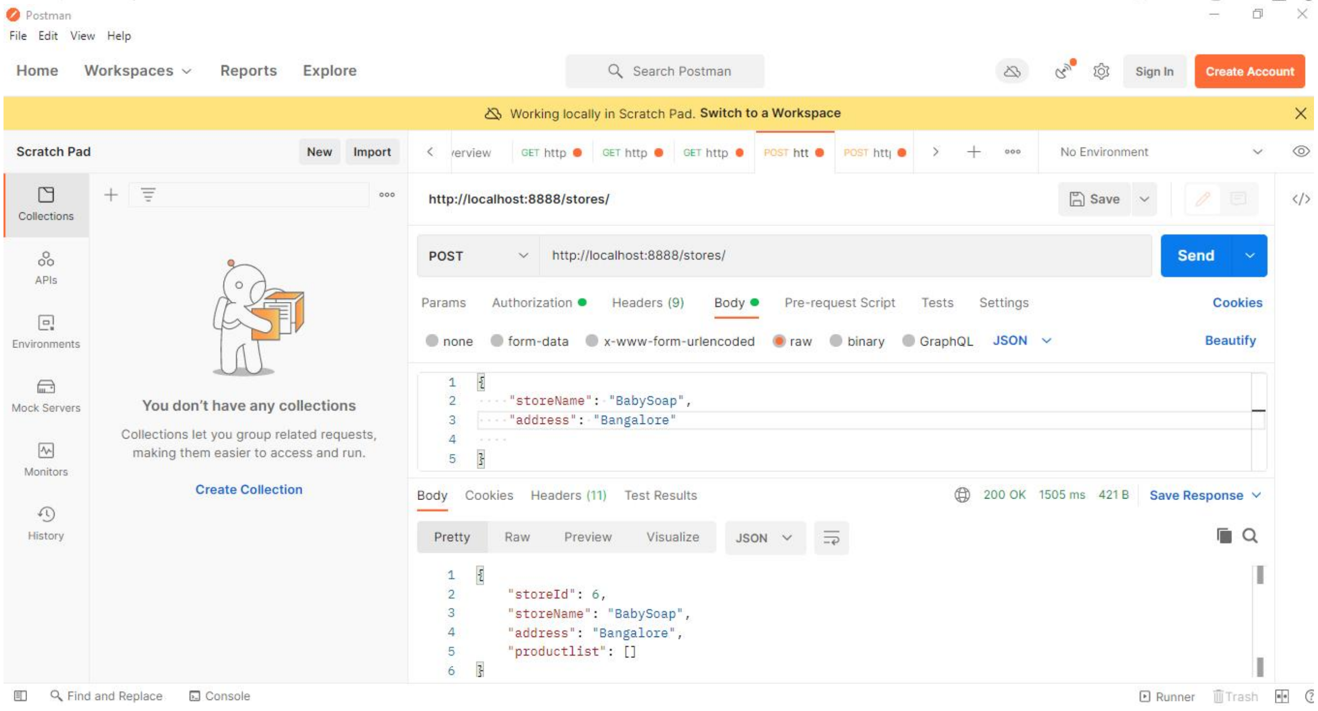
```
[
    {
        "storeId": 1,
        "storeName": "J&J",
        "address": "Chennai",
        "productlist": [
            {
                "productId": 2,
                "productName": "Hair Serum",
                "productPrice": 399.0,
                "password": "admin@123",
                "userName": "Vennila",
                "role": "admin"
            }
```


POST http://localhost:8888/admin/products/7

```
{
    "productName": "Lipstick",
    "productPrice": 250,
    "userName": "Karthik",
    "password": "admin123",
    "role": "admin"
}
```

200 OK 754 ms 466 B

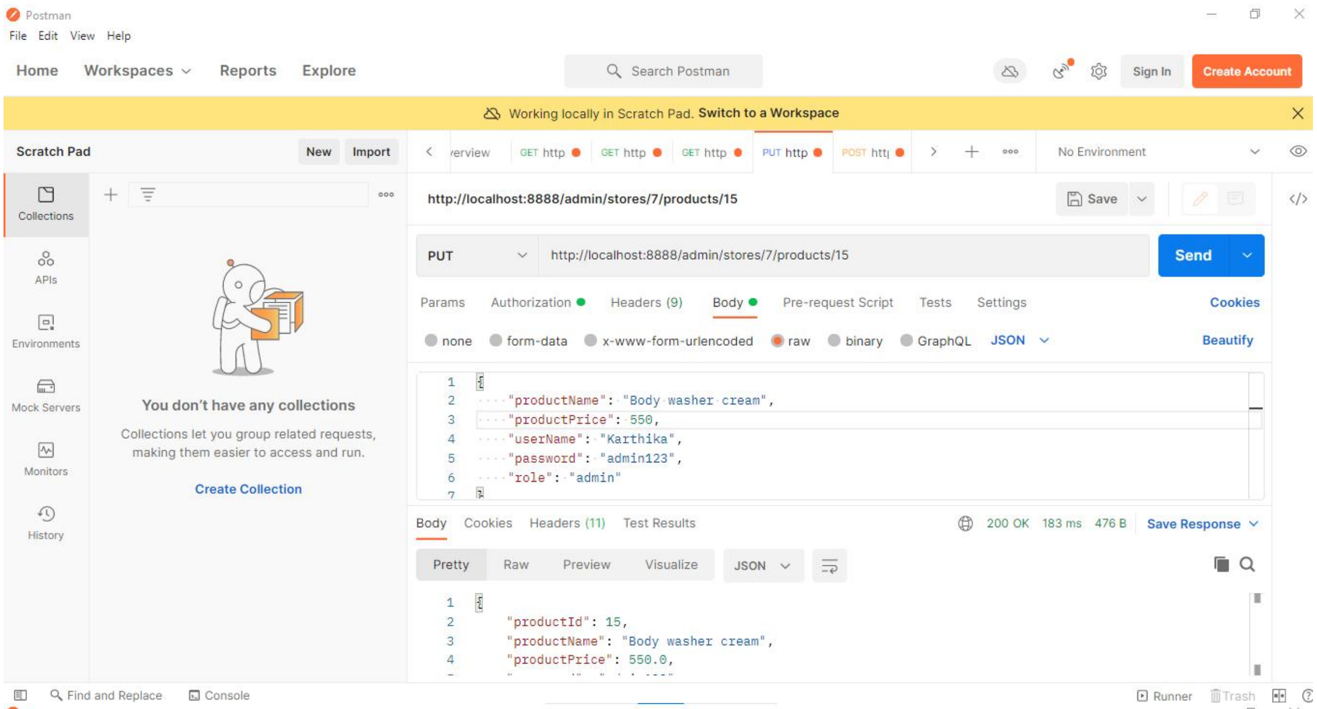
```
    "productId": 14,
    "productName": "Lipstick",
    "productPrice": 250.0,
    "password": "admin123",
```


POST http://localhost:8888/stores/

```
{
    "storeName": "BabySoap",
    "address": "Bangalore"
    
}
```

200 OK 1505 ms 421 B

```
{
    "storeId": 6,
    "storeName": "BabySoap",
    "address": "Bangalore",
    "productlist": []
}
```


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console Runner Trash

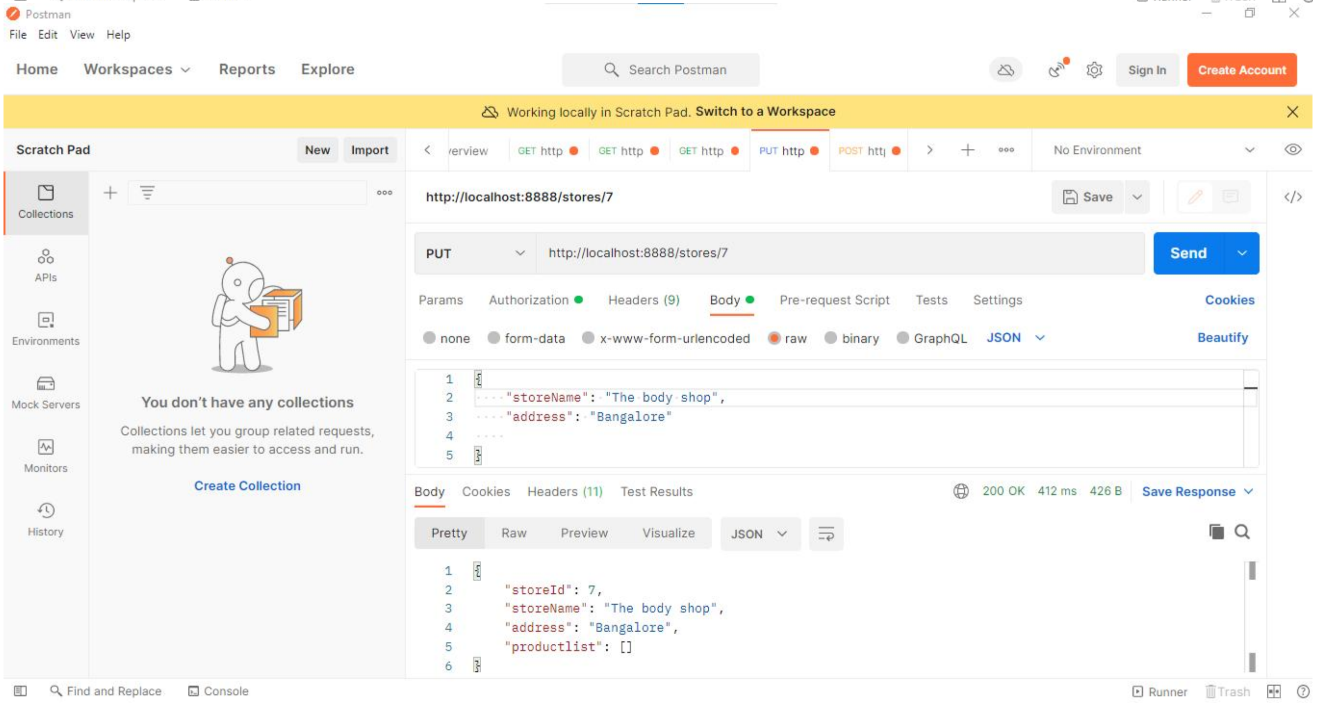
http://localhost:8888/admin/stores/7/products/15

PUT http://localhost:8888/admin/stores/7/products/15

```
{
    "productName": "Body washer cream",
    "productPrice": 550,
    "userName": "Karthika",
    "password": "admin123",
    "role": "admin"
}
```

200 OK 183 ms 476 B

```
{
    "productId": 15,
    "productName": "Body washer cream",
    "productPrice": 550.0,
```


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad New Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
No Environment
Save
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
Body Cookies Headers (11) Test Results
Save Response
Pretty Raw Preview Visualize JSON
Find and Replace Console Runner Trash

http://localhost:8888/stores/7

PUT http://localhost:8888/stores/7

```
{
    "storeName": "The body shop",
    "address": "Bangalore"
    
}
```

200 OK 412 ms 426 B

```
{
    "storeId": 7,
    "storeName": "The body shop",
    "address": "Bangalore",
    "productlist": []
}
```

Conclusion and Application

The project entitled Beauty product store application was completed successfully. The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to develop a rest api application and an added feature of security for storing items in a shop. This project helped us in gaining valuable information and practical knowledge on several topics like designing using Spring JPA, usage of responsive annotations, and management of database using mysql . The entire system is secured. Also the project helped us understanding about the development phases of a project and software development life cycle. We learned how to test different features of a project. This project has given us great satisfaction in having designed an application which can be implemented to any nearby shops or branded shops selling various kinds of products by simple modifications.

In future,it can be further enhanced with authorization and several added features meeting upto the cutting edge technology demands.