PHASE – III

PROJECT

MAKE AN E-COMMERCE WEBSITE FOR SPORTY SHOES.

DESCRIPTION :

PROJECT OBJECTIVE :

As a Full Stack Developer, complete the features of the application by planning the development and pushing the source code to the GitHub repository.

BACKGROUND OF THE PROBLEM STATEMENT :

Sporty Shoes is a company that manufactures and sells sports shoes. They have a walk-in store, and now, they wish to launch their e-commerce portal sportyshoes.com.

You’re asked to develop a prototype of the application. It will be then presented to the relevant stakeholders for budget approval. Your manager has set up a meeting where you’re asked to do the following:

● Presenting the specification document which has the product’s capabilities, appearance, and user interactions  
● Setting up Git and GitHub account to store and track your enhancements of the prototype   
● Explaining the Java concepts used in the project   
● Discussing the generic features of the product:  
● There will be an admin to manage the website. An administrator login will be required to access the admin page.

**The admin should be able to change his password if he wants, he should be able to:**

● Manage the products in the store including categorizing them  
● Browse the list of users who have signed up and be able to search users  
● See purchase reports filtered by date and category

CONCEPTS USED :

🡪H2 Database

🡪Exceptions

🡪Interfaces

🡪Access Specifiers

🡪Encapsulation

SOURCE CODE :

SportyShoeApplication.java

package org.simplilearn.workshop;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SportyShoeApplication {

public static void main(String[] args) {

SpringApplication.run(SportyShoeApplication.class, args);

}

}

SwaggerConfig.java

package org.simplilearn.workshop.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.RequestHandlerSelectors;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration

@EnableSwagger2

public class SwaggerConfig {

@Bean

public Docket superHeroApiDoc() {

return new Docket(DocumentationType.SWAGGER\_2).select()

.apis(RequestHandlerSelectors.basePackage("org.simplilearn.workshop")).build();

}

}

//localhost:8080/swagger-ui

CRUDController.java

package org.simplilearn.workshop.controller;

import org.simplilearn.workshop.exceptionHandler.BusinessException;

import org.simplilearn.workshop.model.PurchaseReport;

import org.simplilearn.workshop.model.Shoe;

import org.simplilearn.workshop.service.SportyShoesService;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.util.LinkedMultiValueMap;

import org.springframework.util.MultiValueMap;

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

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;

@RestController

public class CRUDController {

@Autowired

private SportyShoesService service;

private MultiValueMap<String, String> errorMap;

@PostMapping("/admin/shoe")

public ResponseEntity<Shoe> createShoe(@RequestBody Shoe shoe) {

try {

return new ResponseEntity<>(service.createShoe(shoe), HttpStatus.OK);

} catch (BusinessException e) {

errorMap = new LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

return new ResponseEntity<>(null, errorMap, HttpStatus.BAD\_REQUEST);

}

}

@GetMapping("/admin/shoe/{id}")

public ResponseEntity<Shoe> getShoeById(@PathVariable int id) {

try {

return new ResponseEntity<>(service.getShoeById(id), HttpStatus.OK);

} catch (BusinessException e) {

errorMap = new LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

return new ResponseEntity<>(null, errorMap, HttpStatus.NOT\_FOUND);

}

}

@PutMapping("/admin/shoe")

public ResponseEntity<Shoe> updateShoe(@RequestBody Shoe shoe) {

return new ResponseEntity<>(service.updateShoe(shoe), HttpStatus.OK);

}

@DeleteMapping("/admin/shoe/{id}")

public ResponseEntity<String> deleteShoeById(@PathVariable int id) {

try {

service.deleteShoeById(id);

return new ResponseEntity<>("Succesfully deleted shoe with id: " + id, HttpStatus.OK);

} catch (BusinessException e) {

errorMap = new LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

return new ResponseEntity<>(e.getMessage(), errorMap, HttpStatus.BAD\_REQUEST);

}

}

@PostMapping("/admin/purchaseReport")

public ResponseEntity<PurchaseReport> createPurchaseReport(@RequestBody PurchaseReport pr) {

try {

return new ResponseEntity<>(service.createPurchaseReport(pr), HttpStatus.OK);

} catch (BusinessException e) {

errorMap = new LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

return new ResponseEntity<>(null, errorMap, HttpStatus.BAD\_REQUEST);

}

}

@GetMapping("/admin/purchaseReport/id/{id}")

public ResponseEntity<PurchaseReport> getPurchaseReportById(@PathVariable int id) {

try {

return new ResponseEntity<>(service.getPurchaseReportById(id), HttpStatus.OK);

} catch (BusinessException e) {

errorMap = new LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

return new ResponseEntity<>(null, errorMap, HttpStatus.NOT\_FOUND);

}

}

@PutMapping("/admin/purchaseReport")

public ResponseEntity<PurchaseReport> updatePurchaseReport(@RequestBody PurchaseReport pr) {

return new ResponseEntity<>(service.updatePurchaseReport(pr), HttpStatus.OK);

}

@DeleteMapping("/admin/purchaseReport/{id}")

public ResponseEntity<String> deletePurchaseReportById(@PathVariable int id) {

try {

service.deletePurchaseReportById(id);

return new ResponseEntity<>("Succesfully deleted Purchase Report with id: " + id, HttpStatus.OK);

} catch (BusinessException e) {

errorMap = new LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

return new ResponseEntity<>(e.getMessage(), errorMap, HttpStatus.BAD\_REQUEST);

}

}

}

SearchController.java

package org.simplilearn.workshop.controller;

import java.util.Date;

import java.util.List;

import org.simplilearn.workshop.model.PurchaseReport;

import org.simplilearn.workshop.model.Shoe;

import org.simplilearn.workshop.service.SportyShoesService;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

@RestController

public class SearchController {

@Autowired

private SportyShoesService service;

@GetMapping("/admin/shoe/all")

public ResponseEntity<List<Shoe>> getAllShoes(){

return new ResponseEntity<List<Shoe>>(service.getAllShoes(), HttpStatus.OK);

}

@GetMapping("/admin/purchaseReport/category/{category}")

public ResponseEntity<List<PurchaseReport>> getAllPurchaseReportsByCategory(@PathVariable String category){

return new ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReportsByCategory(category), HttpStatus.OK);

}

@GetMapping("/admin/purchaseReport/date/{dateInMs}")

public ResponseEntity<List<PurchaseReport>> getAllPurchaseReportsByDop(@PathVariable Long dateInMs){

Date dop = new Date(dateInMs);

return new ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReportsByDOP(dop), HttpStatus.OK);

}

@GetMapping("/admin/purchaseReport/all")

public ResponseEntity<List<PurchaseReport>> getAllPurchaseReport(){

return new ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReports(), HttpStatus.OK);

}

}

BusinessException.java

**package** org.simplilearn.workshop.exceptionHandler;

**public** **class** BusinessException **extends** Exception{

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1008128726286682480L;

**public** BusinessException() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** BusinessException(String message) {

**super**(message);

// **TODO** Auto-generated constructor stub

}

}

PurchaseReport.java

package org.simplilearn.workshop.model;

import java.util.Date;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

@Entity

@Table

public class PurchaseReport {

public PurchaseReport(int id, String purchasedBy, String category, Date dop, String orderList) {

super();

this.id = id;

this.purchasedBy = purchasedBy;

this.category = category;

this.dop = dop;

this.orderList = orderList;

}

@Id

@GeneratedValue

private int id;

private String purchasedBy; // This can be extended to utilize one to one relation with User Table [Future Implemetations]

private String category;

@Temporal(TemporalType.DATE)

private Date dop;

String orderList;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getPurchasedBy() {

return purchasedBy;

}

public void setPurchasedBy(String purchasedBy) {

this.purchasedBy = purchasedBy;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public Date getDop() {

return dop;

}

public void setDop(Date dop) {

this.dop = dop;

}

@Override

public String toString() {

return String.format("PurchaseReport [id=%s, purchasedBy=%s, category=%s, dop=%s]", id, purchasedBy, category,

dop);

}

public PurchaseReport() {

super();

}

}

Shoe.java

package org.simplilearn.workshop.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table

public class Shoe {

public Shoe(int id, String name, String category, double price) {

super();

this.id = id;

this.name = name;

this.category = category;

this.price = price;

}

@Id

@GeneratedValue

private int id;

private String name;

private String category;

private double price;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

@Override

public String toString() {

return String.format("Shoe [id=%s, name=%s, category=%s, price=%s]", id, name, category, price);

}

public Shoe() {

super();

}

}

SportyShoeService.java

package org.simplilearn.workshop.service;

import java.util.Date;

import java.util.List;

import org.simplilearn.workshop.exceptionHandler.BusinessException;

import org.simplilearn.workshop.model.PurchaseReport;

import org.simplilearn.workshop.model.Shoe;

public interface SportyShoesService {

public Shoe createShoe(Shoe shoe) throws BusinessException;

public Shoe getShoeById(int id) throws BusinessException;

public Shoe updateShoe(Shoe shoe);

public void deleteShoeById(int id) throws BusinessException;

public List<Shoe> getAllShoes();

public PurchaseReport createPurchaseReport(PurchaseReport pr) throws BusinessException;

public PurchaseReport getPurchaseReportById(int id) throws BusinessException;

public PurchaseReport updatePurchaseReport(PurchaseReport pr);

public void deletePurchaseReportById(int id) throws BusinessException;

public List<PurchaseReport> getAllPurchaseReports();

public List<PurchaseReport> getAllPurchaseReportsByCategory(String category);

public List<PurchaseReport> getAllPurchaseReportsByDOP(Date dop);

}

SportyShoeServiceImpl.java

package org.simplilearn.workshop.service;

import java.util.Date;

import java.util.List;

import java.util.NoSuchElementException;

import javax.annotation.PostConstruct;

import org.simplilearn.workshop.exceptionHandler.BusinessException;

import org.simplilearn.workshop.model.PurchaseReport;

import org.simplilearn.workshop.model.Shoe;

import org.simplilearn.workshop.repository.PurchaseReportRepository;

import org.simplilearn.workshop.repository.ShoeRepository;

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

import org.springframework.dao.EmptyResultDataAccessException;

import org.springframework.stereotype.Service;

import lombok.NoArgsConstructor;

@Service

@NoArgsConstructor

public class SportyShoesServiceImpl implements SportyShoesService{

@Autowired

private ShoeRepository shoesRepo;

@Autowired

private PurchaseReportRepository prRepo;

@PostConstruct

public void init() {

Shoe s1 = new Shoe(1,"Shoe Name 1","Basketball",1000.24);

Shoe s2 = new Shoe(2,"Shoe Name 2","Cricket",1100.24);

Shoe s3 = new Shoe(3,"Shoe Name 3","Running",900.24);

Shoe s4 = new Shoe(4,"Shoe Name 4","Football",1900.24);

shoesRepo.save(s1);

shoesRepo.save(s2);

shoesRepo.save(s3);

shoesRepo.save(s4);

Date d = new Date(0);

PurchaseReport pr1 = new PurchaseReport(5,"user\_1","Running",d,"adidas\_runner:5,nike\_airmax:10");

PurchaseReport pr2 = new PurchaseReport(6,"user\_2","Cricket",d,"adidas\_cricket:5,nike\_cricket:10");

PurchaseReport pr3 = new PurchaseReport(7,"user\_3","Basketball",d,"adidas\_basketball:5,nike\_basketball:10");

PurchaseReport pr4 = new PurchaseReport(8,"user\_4","Football",d,"adidas\_football:5,nike\_football:10");

prRepo.save(pr1);

prRepo.save(pr2);

prRepo.save(pr3);

prRepo.save(pr4);

}

public Shoe createShoe(Shoe shoe) throws BusinessException {

int id = shoe.getId();

Shoe oldShoe = null;

try {

oldShoe = shoesRepo.findById(id).get();

}catch(NoSuchElementException e) {

}

if(oldShoe!=null) throw new BusinessException("Shoe already exists with id: "+id);

return shoesRepo.save(shoe);

}

public Shoe getShoeById(int id) throws BusinessException {

Shoe shoe = null;

try {

if(id<=0) throw new BusinessException("Shoe Id can not be negative or zero");

shoe = shoesRepo.findById(id).get();

}catch(NoSuchElementException e) {

throw new BusinessException("Shoe not found with Id: "+id);

}

return shoe;

}

public Shoe updateShoe(Shoe shoe) {

return shoesRepo.save(shoe);

}

public void deleteShoeById(int id) throws BusinessException {

try {

shoesRepo.deleteById(id);

}catch(IllegalArgumentException e) {

throw new BusinessException("Invalid id: "+id);

}catch(EmptyResultDataAccessException e) {

throw new BusinessException("SHoe does not exist with id: "+id);

}

}

public List<Shoe> getAllShoes() {

return shoesRepo.findAll();

}

public PurchaseReport createPurchaseReport(PurchaseReport pr) throws BusinessException {

int id = pr.getId();

PurchaseReport oldPr = null;

try {

oldPr = prRepo.findById(id).get();

}catch(NoSuchElementException e) {

}

if(oldPr!=null) throw new BusinessException("Purchase report already exists with id: "+id);

return prRepo.save(pr);

}

public PurchaseReport getPurchaseReportById(int id) throws BusinessException {

PurchaseReport pr = null;

try {

if(id<=0) throw new BusinessException("Purchase Report Id can not be negative or zero");

pr = prRepo.findById(id).get();

}catch(NoSuchElementException e) {

throw new BusinessException("Purchase Report not found with Id: "+id);

}

return pr;

}

public PurchaseReport updatePurchaseReport(PurchaseReport pr) {

return prRepo.save(pr);

}

public void deletePurchaseReportById(int id) throws BusinessException {

try {

prRepo.deleteById(id);

}catch(IllegalArgumentException e) {

throw new BusinessException("Invalid id: "+id);

}catch(EmptyResultDataAccessException e) {

throw new BusinessException("Puchase Report does not exist with Id: "+id);

}

}

public List<PurchaseReport> getAllPurchaseReports() {

return prRepo.findAll();

}

public List<PurchaseReport> getAllPurchaseReportsByCategory(String category) {

return prRepo.findByCategory(category);

}

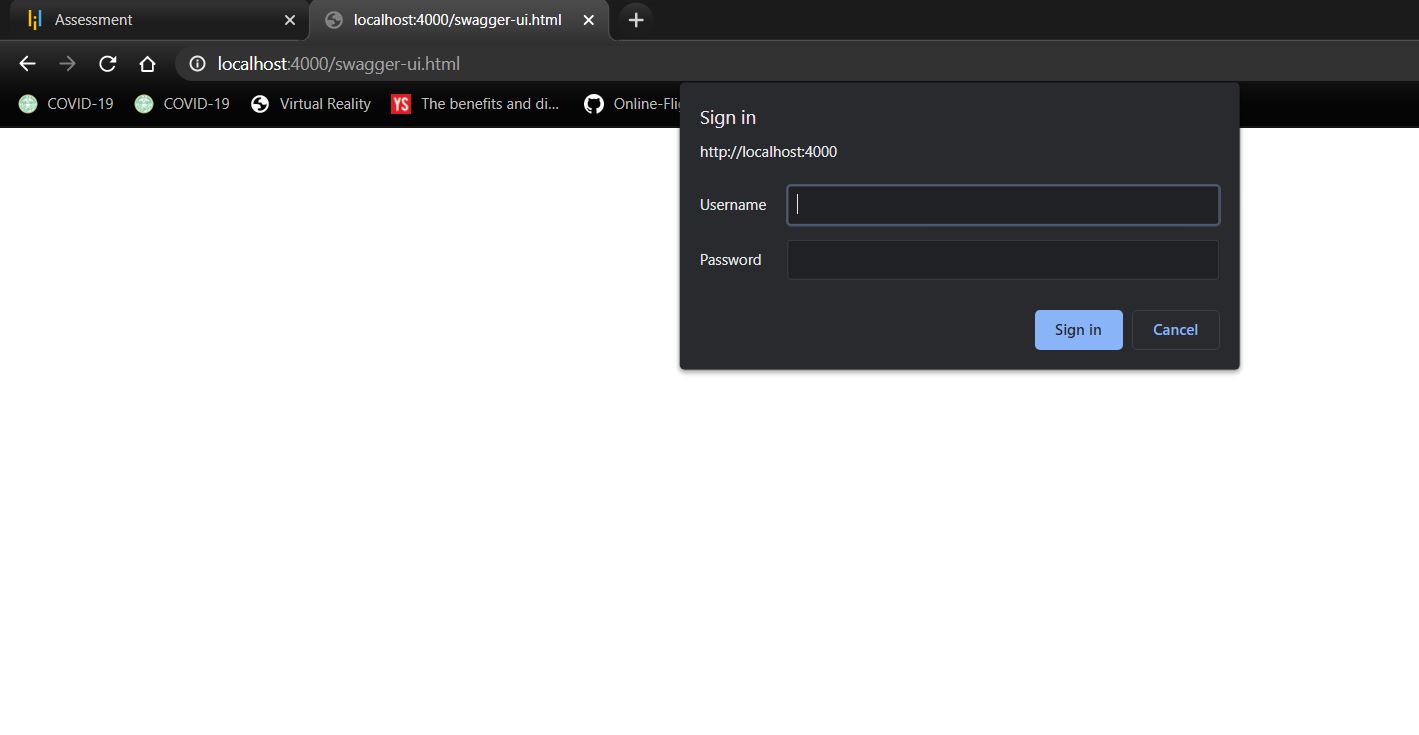
public List<PurchaseReport> getAllPurchaseReportsByDOP(Date dop) {

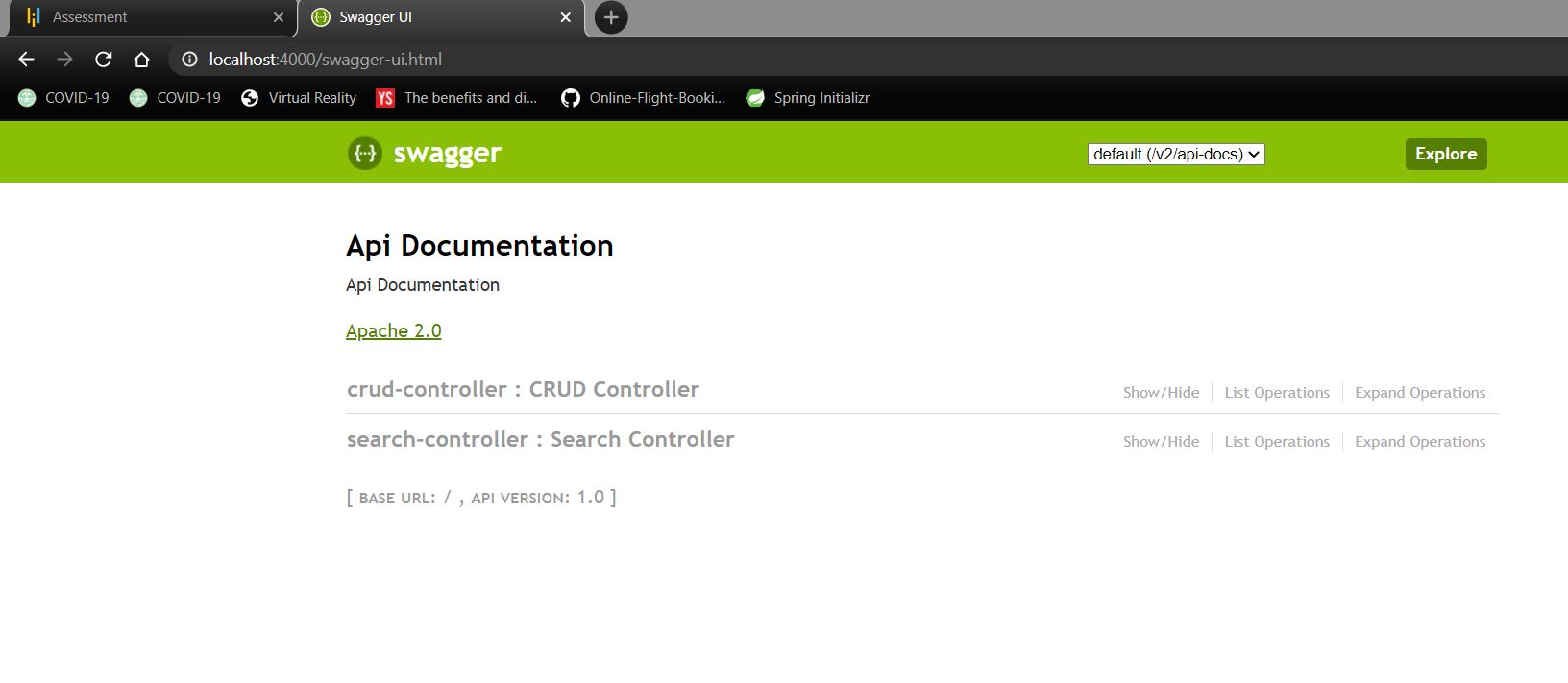
return prRepo.findByDop(dop);

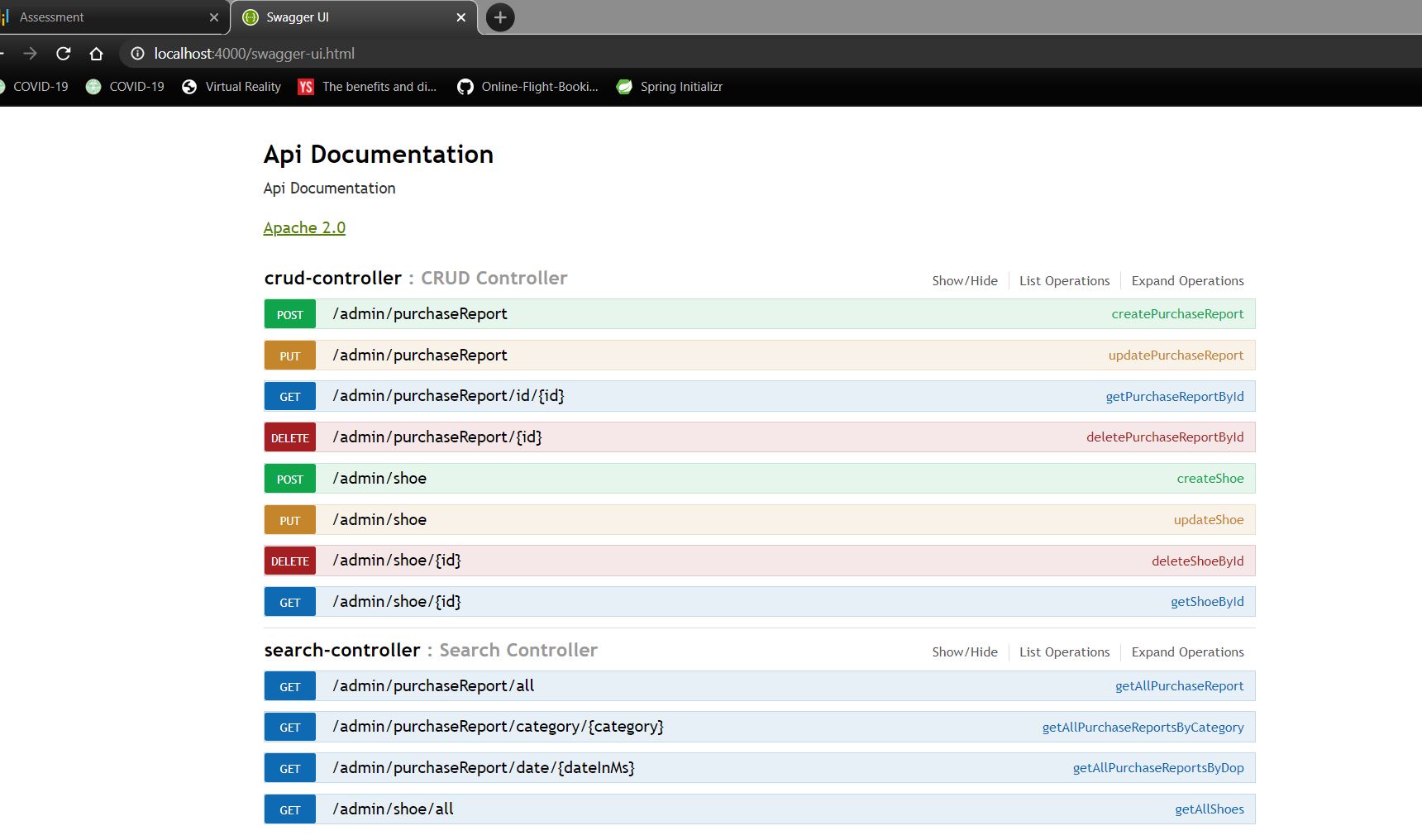
}

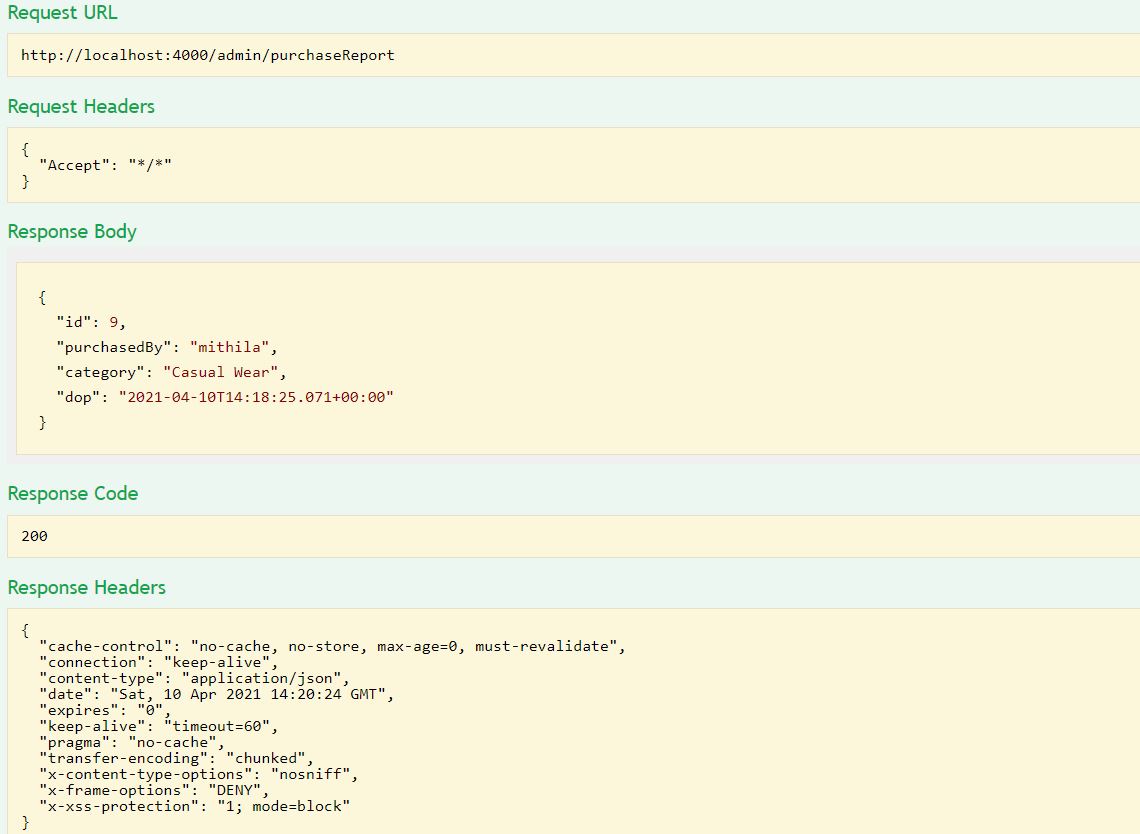
}

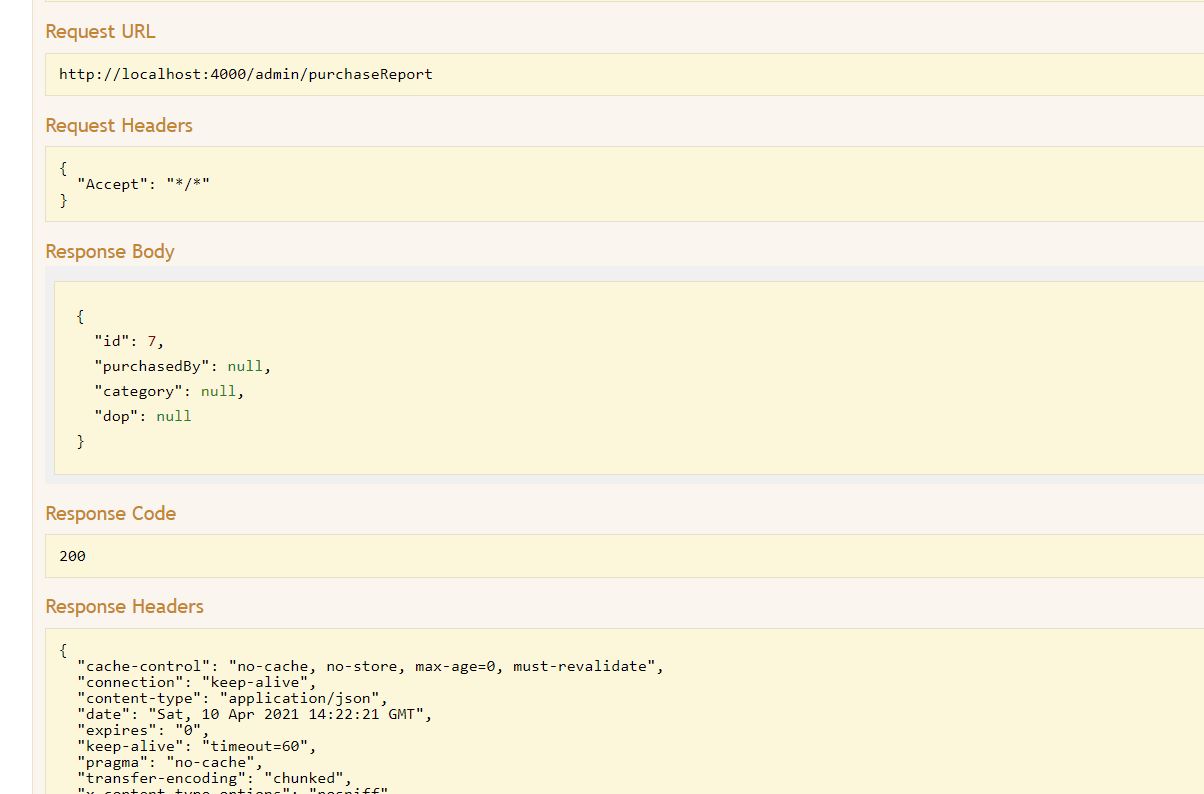
SCREENSHOTS :

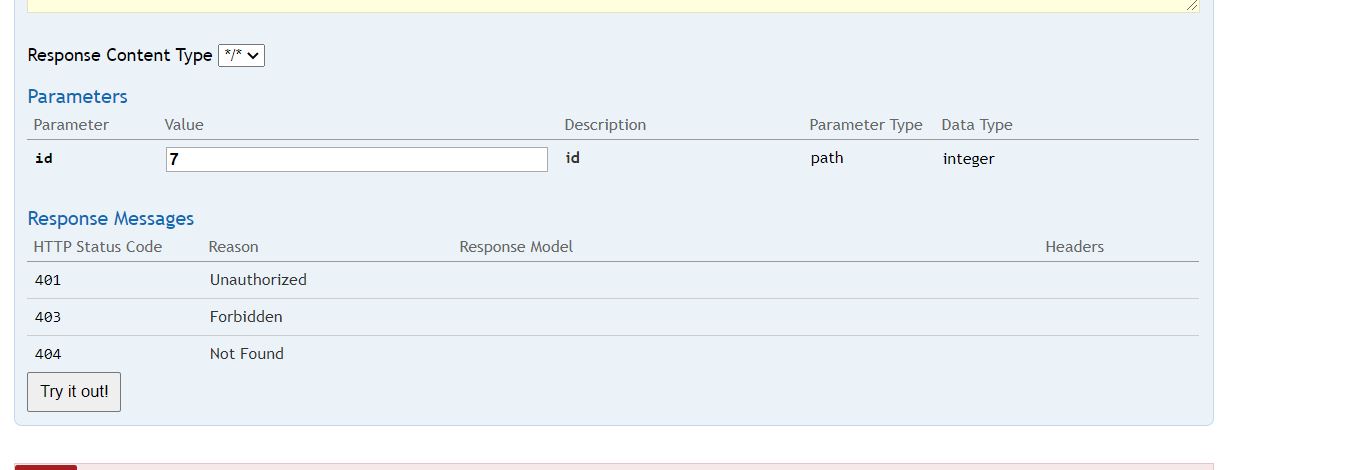


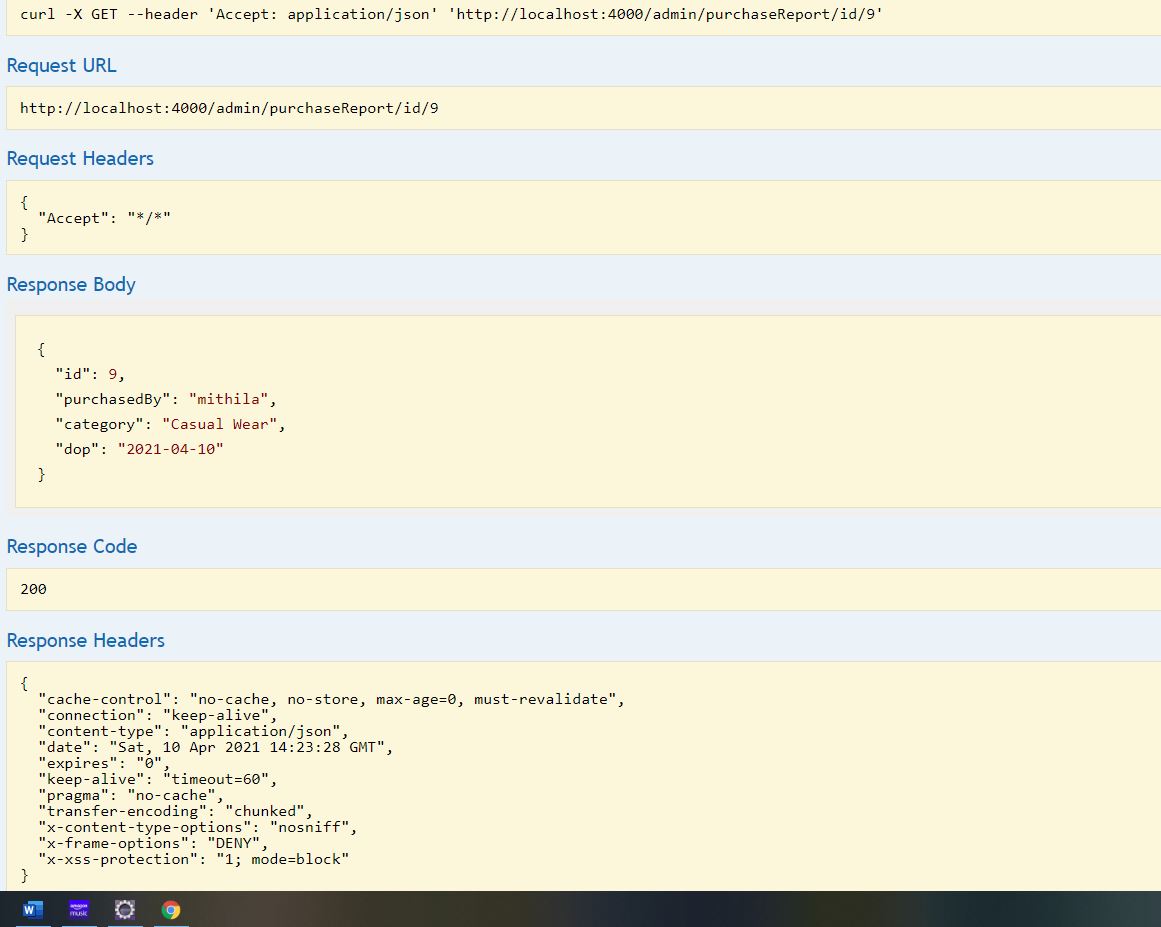






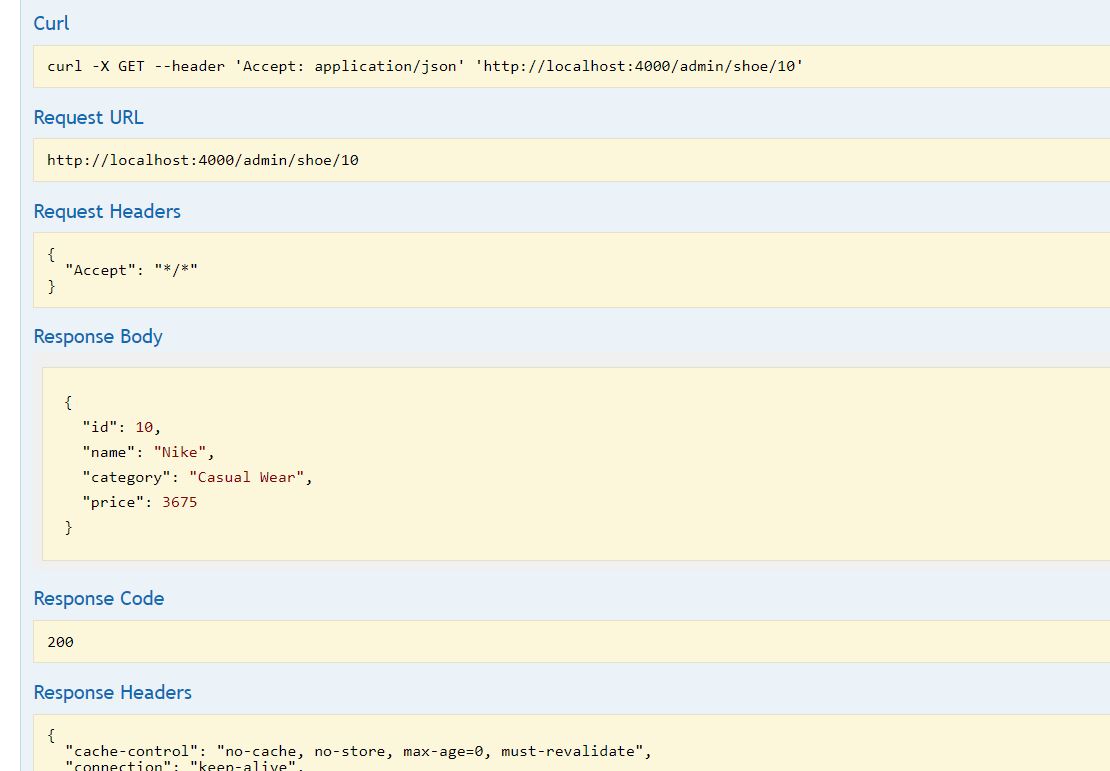




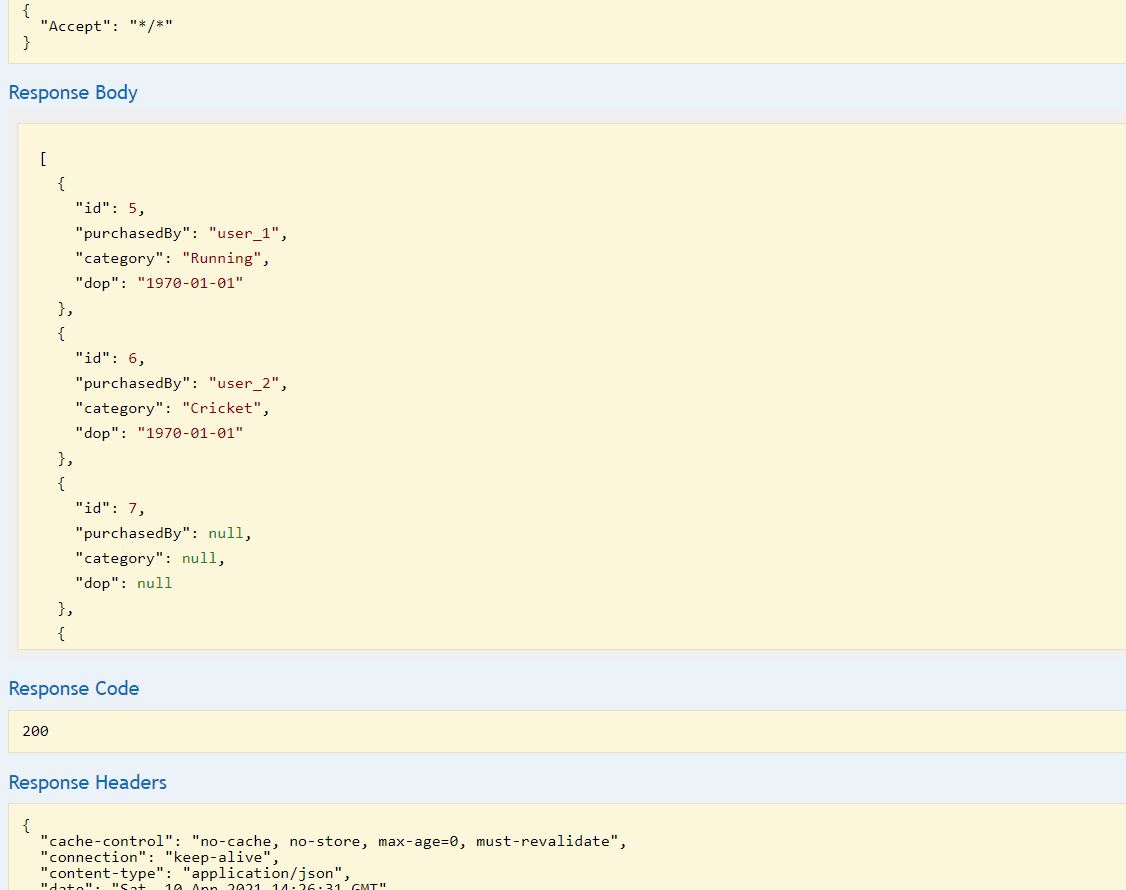


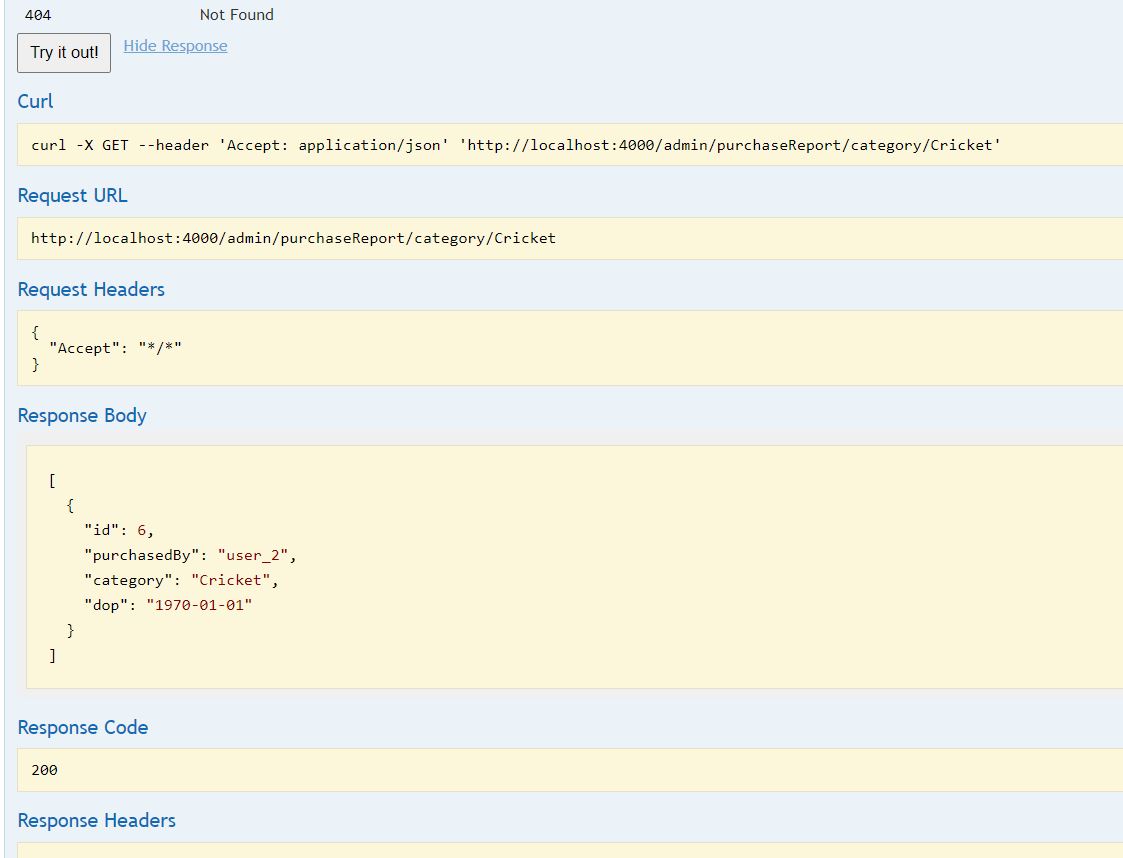


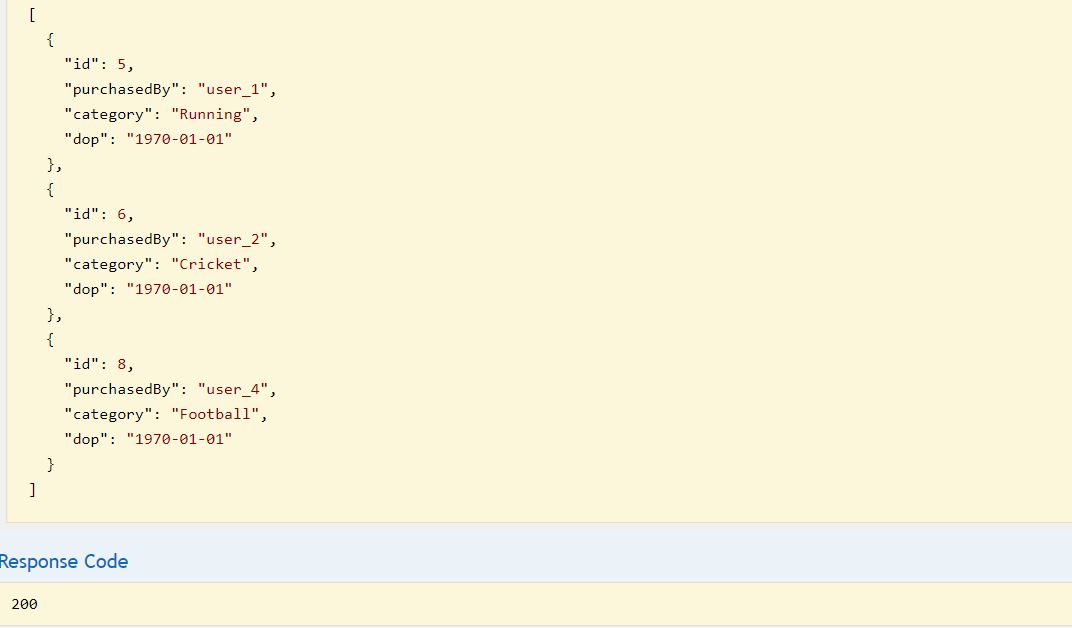


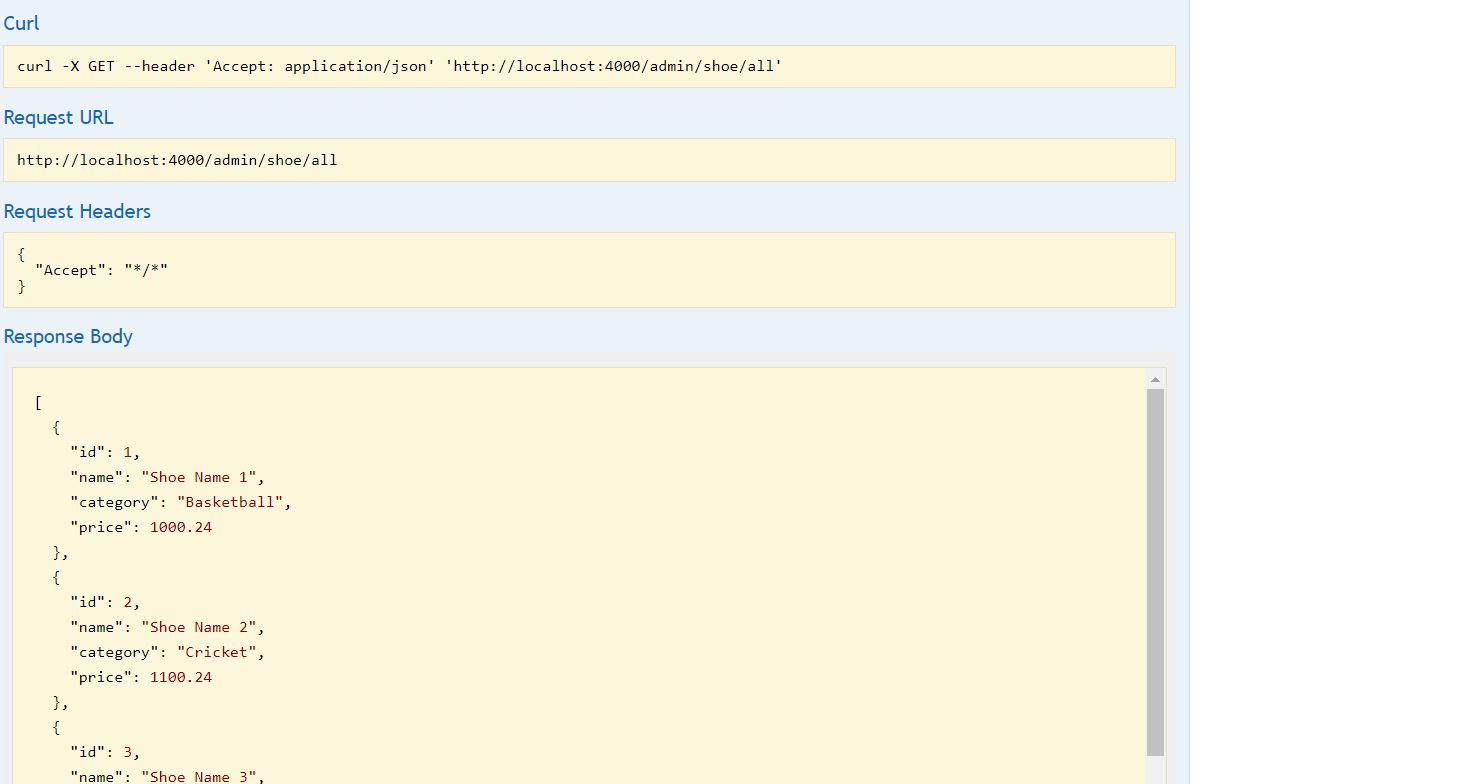












GITHUB LINK :

https://github.com/mithilagaddam/SportyShoes