```
package com.api.sportyShoes;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration;

@SpringBootApplication(exclude = { SecurityAutoConfiguration.class })
public class SportyShoes {

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

	}

}
```

==========

```
package com.api.sportyShoes.config;


import org.springframework.context.annotation.Configuration;
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.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http
		.csrf().disable()
    .authorizeRequests()
	.anyRequest()
	.authenticated()
	.and()
    .httpBasic();

	}

}
```

==========

```
package com.api.sportyShoes.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("com.api.sportyShoes")).build();
	}
}
```

==========

```
package com.api.sportyShoes.controller;

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;

import com.api.sportyShoes.exceptionHandler.BusinessException;
import com.api.sportyShoes.model.PurchaseReport;
import com.api.sportyShoes.model.Shoe;
import com.api.sportyShoes.service.SportyShoesService;

@RestController
public class CRUDController {

	@Autowired
	private SportyShoesService service;

	private MultiValueMap<String, String> errorMap;

	/**
	 * Shoe post request controller
	 *
	 * @param shoe
	 * @return ResponseEntity<Shoe> with newly created Shoe
	 */
	@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);
		}
	}
```

```
	/**
	 * Shoe get request controller
	 *
	 * @param id
	 * @return ResponseEntity<Shoe> with the given id
	 */
	@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);
		}
	}

	/**
	 * Shoe put(update) request controller
	 *
	 * @param shoe
	 * @return ResponseEntity<Shoe> with updated shoe
	 */
	@PutMapping("/admin/shoe")
	public ResponseEntity<Shoe> updateShoe(@RequestBody Shoe shoe) {
		return new ResponseEntity<>(service.updateShoe(shoe), HttpStatus.OK);
	}

	/**
	 * Shoe delete request controller
	 *
	 * @param id
	 * @return ResponseEntity<String> containing the status of delete operation
	 */
	@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);
		}
	}

	/**
	 * Purchase Report post request controller
	 *
	 * @param pr - Purchase Report
	 * @return ResponseEntity<PurchaseReport> with newly created Purchase Report
	 */
	@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);
		}
	}

	/**
	 * Purchase Report get request controller
	 *
	 * @param id
	 * @return ResponseEntity<PurchaseReport> with given id
	 */
	@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);
		}
	}

	/**
	 * Purchase Report put(update) request controller
	 *
	 * @param pr
	 * @return ResponseEntity<PurchaseReport> containing updated Purchase Report
	 */
	@PutMapping("/admin/purchaseReport")
	public ResponseEntity<PurchaseReport> updatePurchaseReport(@RequestBody
PurchaseReport pr) {
		return new ResponseEntity<>(service.updatePurchaseReport(pr), HttpStatus.OK);
	}

	/**
	 * Purchase Report delete request controller
	 *
	 * @param id
	 * @return ResponseEntity<String> containing the status of delete request.
	 */
	@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);
```

```
		}
	}
}
```

==========

```
package com.api.sportyShoes.controller;

import java.util.Date;
import java.util.List;

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;

import com.api.sportyShoes.model.PurchaseReport;
import com.api.sportyShoes.model.Shoe;
import com.api.sportyShoes.service.SportyShoesService;

@RestController
public class SearchController {

	@Autowired
	private SportyShoesService service;

	/**
	 * Shoe search controller
	 * @return all shoe list
	 */

	@GetMapping("/admin/shoe/all")
	public ResponseEntity<List<Shoe>> getAllShoes(){
		return new ResponseEntity<List<Shoe>>(service.getAllShoes(), HttpStatus.OK);
	}


	/**
	 * Purchase Report Search Controller
	 * @param category
	 * @return purchase reports filtered by the category
	 */
	@GetMapping("/admin/purchaseReport/category/{category}")
	public ResponseEntity<List<PurchaseReport>>
getAllPurchaseReportsByCategory(@PathVariable String category){
		return new
ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReportsByCategory(category),
HttpStatus.OK);
	}

	/**
	 * Purchase Report Search Controller
	 * @param dateInMs
```

```
	 * @return purchase reports filtered by date of purchase(in millisecond time)
	 */
	@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);
	}

	/**
	 * Purchase Report Search Controller
	 * @return all purchase reports
	 */
	@GetMapping("/admin/purchaseReport/all")
	public ResponseEntity<List<PurchaseReport>> getAllPurchaseReport(){
		return new ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReports(),
HttpStatus.OK);
	}
}
```

==========

```
package com.api.sportyShoes.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
	}

}
```

==========

```
package com.api.sportyShoes.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;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Entity
@Table
@Setter
@Getter
@NoArgsConstructor
@ToString
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;

/**
 * This can be used for storing orderlist as <Qty, Shoe>
 * Here implementation is made simple by using shoeId instead
 * of shoe in string format.
 */
//	@ManyToMany(cascade = CascadeType.ALL)
//	Map<Integer,Shoe> orderList = new HashMap<Integer,Shoe>();
//                                                                  OR
//	Map<Integer,Integer> orderList = new HashMap<Integer,Integer>();

	String orderList;

}
```

```
================================================================================
package com.api.sportyShoes.model;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Entity
@Table
@Getter
@Setter
@NoArgsConstructor
@ToString
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;


}
```

==========

```
package com.api.sportyShoes.repository;

import java.util.Date;
import java.util.List;

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

import com.api.sportyShoes.model.PurchaseReport;

@Repository
public interface PurchaseReportRepository extends JpaRepository<PurchaseReport, Integer>{
	public List<PurchaseReport> findByDop(Date dop);
	public List<PurchaseReport> findByCategory(String category);

}
```

==========

```
package com.api.sportyShoes.repository;


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

import com.api.sportyShoes.model.Shoe;

@Repository
public interface ShoesRepository extends JpaRepository<Shoe, Integer>{

}
```

==========

```
package com.api.sportyShoes.service;

import java.util.Date;
import java.util.List;

import com.api.sportyShoes.exceptionHandler.BusinessException;
import com.api.sportyShoes.model.PurchaseReport;
import com.api.sportyShoes.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);

}
```

==========

```
package com.api.sportyShoes.service.impl;
```

```
import java.util.Date;
import java.util.List;
import java.util.NoSuchElementException;

import javax.annotation.PostConstruct;

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

import com.api.sportyShoes.exceptionHandler.BusinessException;
import com.api.sportyShoes.model.PurchaseReport;
import com.api.sportyShoes.model.Shoe;
import com.api.sportyShoes.repository.PurchaseReportRepository;
import com.api.sportyShoes.repository.ShoesRepository;
import com.api.sportyShoes.service.SportyShoesService;

import lombok.NoArgsConstructor;

@Service
@NoArgsConstructor
public class SportyShoesServiceImpl implements SportyShoesService{

	@Autowired
	private ShoesRepository 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);
	}

}
```