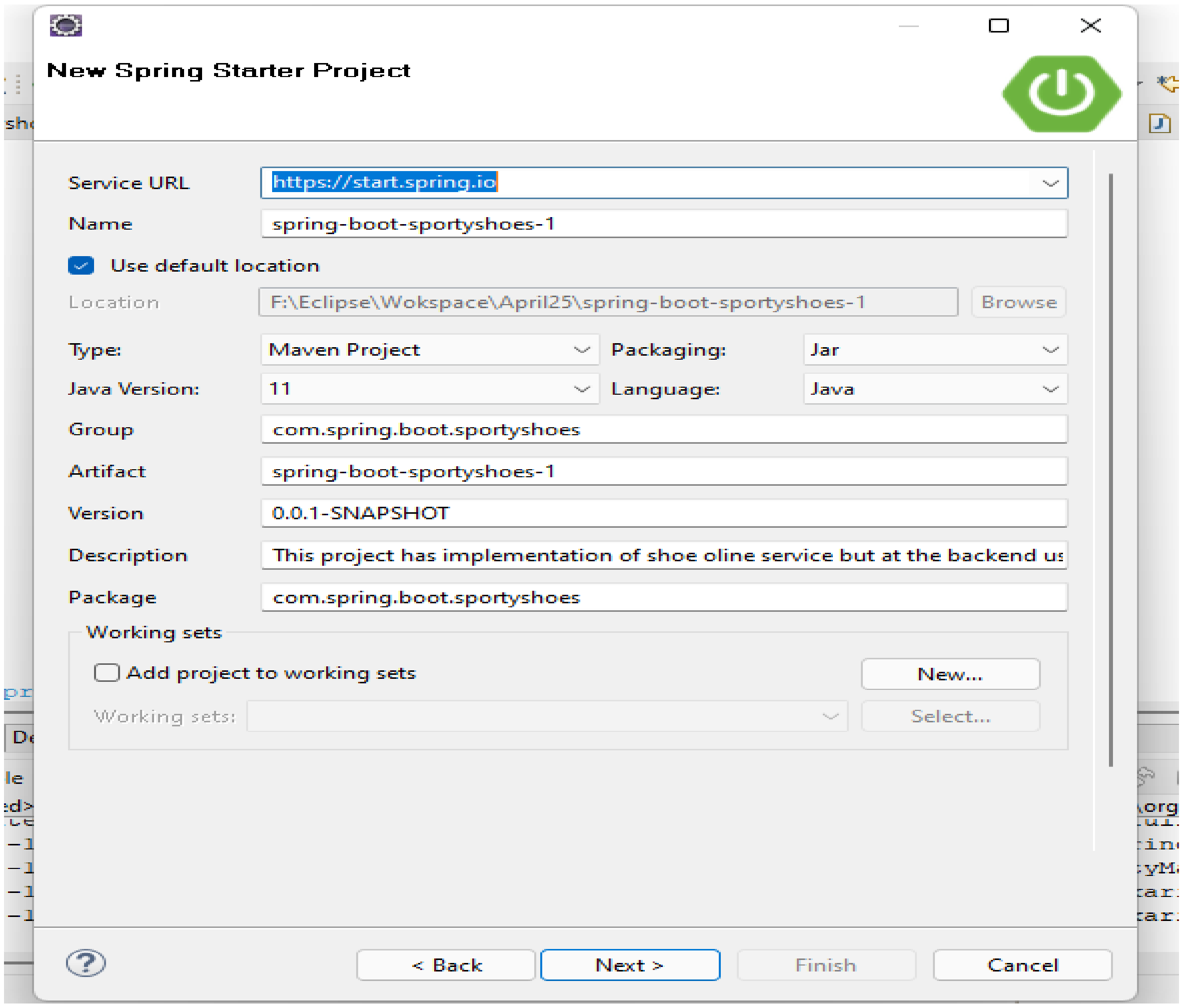
Project –

Implement Frameworks the DevOps way (Phase-4)

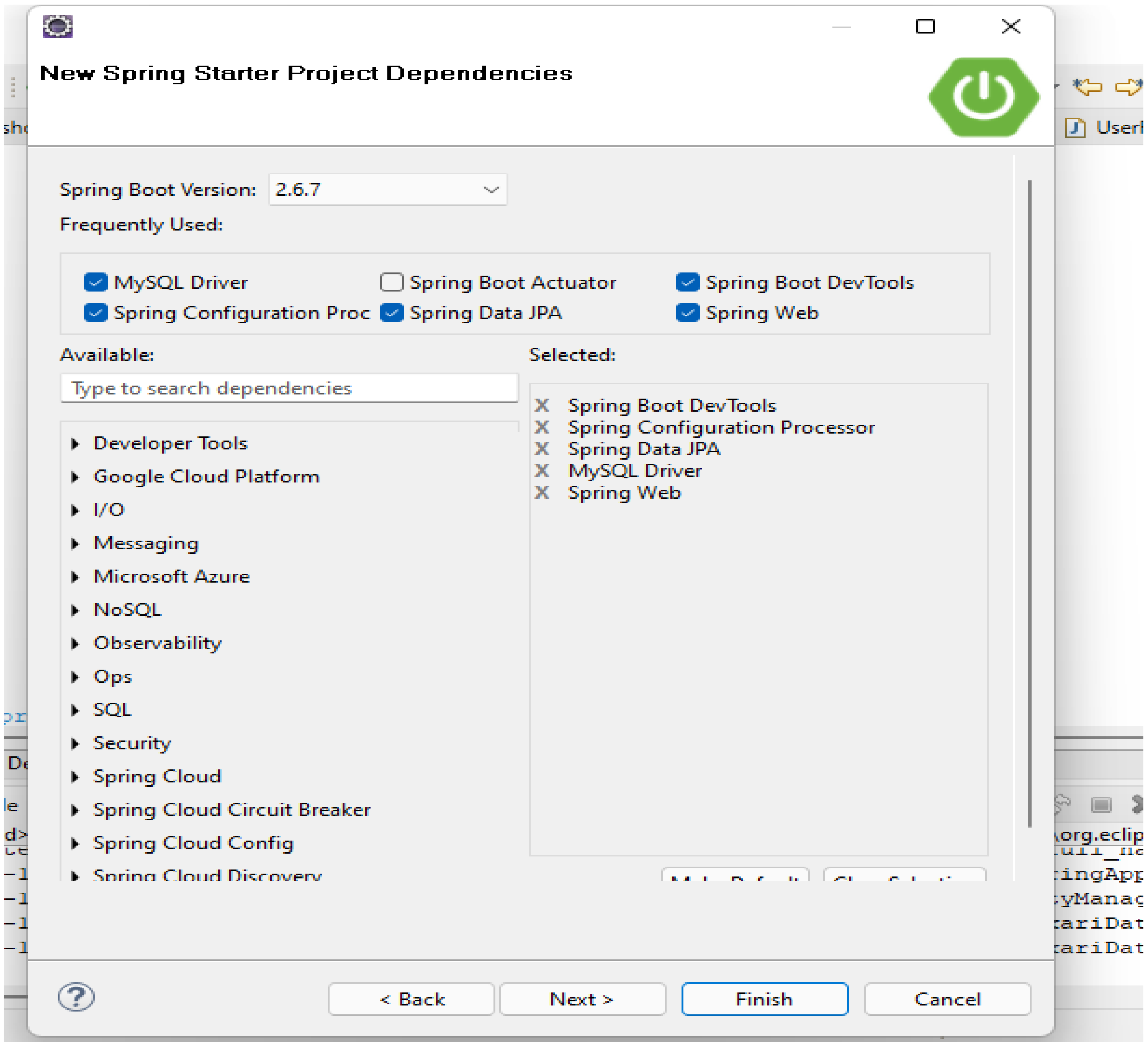
Source Code SportyShoes

Create SpringBoot project using spring Starter project-



Service URL: https://start.spring.io
Name: spring-boot-sportyshoes-1
☑ Use default location
Location: F:\Eclipse\Wokspace\April25\spring-boot-sportyshoes-1
Type: Maven Project
Packaging: Jar
Java Version: 11
Language: Java
Group: com.spring.boot.sportyshoes
Artifact: spring-boot-sportyshoes-1
Version: 0.0.1-SNAPSHOT
Description: This project has implementation of shoe oline service but at the backend us
Package: com.spring.boot.sportyshoes
Working sets
☐ Add project to working sets
Working sets:

And after spring.io open enter the dependencies and spring project will we created.



New Spring Starter Project Dependencies

Spring Boot Version: 2.6.7

Frequently Used:

[✓] MySQL Driver [ ] Spring Boot Actuator [✓] Spring Boot DevTools
[✓] Spring Configuration Proc [✓] Spring Data JPA [✓] Spring Web

Available:

Developer Tools, Google Cloud Platform, I/O, Messaging, Microsoft Azure, NoSQL, Observability, Ops, SQL, Security, Spring Cloud, Spring Cloud Circuit Breaker, Spring Cloud Config, Spring Cloud Discovery

Selected:

- Spring Boot DevTools
- Spring Configuration Processor
- Spring Data JPA
- MySQL Driver
- Spring Web

After the finish button click, spring boot project is created.

Now setup the pom.xml in the spring boot project- **spring-boot-sportyshoes .**

Pom.xml file-

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
```

```xml
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.7</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.spring.boot.sportyshoes</groupId>
    <artifactId>spring-boot-sportyshoes</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spring-boot-sportyshoes</name>
    <description>This project has implementation of shoe oline service but at the
backend using swagger and postman for displaying output.</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>

    <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-boot-starter</artifactId>
            <version>3.0.0</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Add dependency –

springfox-boot-starter – for Swagger use add this.

spring-boot-starter-data-jpa – for jpa repository use and for basic crud operation.

spring-boot-starter-web—for making web project

mysql-connector-java—for use of mysql database

spring-boot-configuration-processor –basic tools for configuration

spring-boot-devtools –tool for devtools

→Now add **application.properties** in the **src/main/resources**

```
spring.mvc.pathmatch.matching-strategy=ANT_PATH_MATCHER

#database configuration: mysql
spring.datasource.url=jdbc:mysql://localhost:3306/fun
spring.datasource.username=root
spring.datasource.password=Chemistry22@
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#Hibernate Configuration
```

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
```

Database used- fun

For activate swagger use the AntMatcher in application.properties.

Driver class for SportyShoes Code-

Path-src/main/java

Package-com.spring.boot.sportyshoes

Class-SportyshoesApplication.java

```
package com.spring.boot.sportyshoes;

import java.util.Collections;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@EnableSwagger2
@SpringBootApplication
public class SportyshoesApplication {

    public static void main(String[] args) {
        SpringApplication.run(SportyshoesApplication.class, args);
        System.out.println("Welcome to SportyShoes section!!");
    }

     @Bean
        public Docket swaggerConfiguration() {
            return new Docket(DocumentationType.SWAGGER_2)
                    .select()
                    .apis(RequestHandlerSelectors.basePackage("com.spring.boot.sp
ortyshoes"))
```

```
                    .build()
                    .apiInfo(apiDetails());
        }

        private ApiInfo apiDetails() {
            return new ApiInfo(
                    "Sportyshoes API",
                    "Spring Boot REST API for sportyshoes.com",
                    "1.0",
                    "Free to use",
                    new springfox.documentation.service.Contact("Rahul Rana",
"https://github.com/rahulpersie66/", "rahulpersie66@gmail.com"),
                    "API License",
                    "https://github.com/rahulpersie66/",
                    Collections.emptyList()

        );
      }

}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.entities

Class- AdminTable.java

Table create in DB-Admin_Table

```
package com.spring.boot.sportyshoes.entities;

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

@Entity
@Table(name="Admin_Table")
public class AdminTable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int adminId;
    private String adminUsername;
    private String password;
    public int getAdminId() {
        return adminId;
    }
    public void setAdminId(int adminId) {
        this.adminId = adminId;
    }
    public String getAdminUsername() {
        return adminUsername;
    }
    public void setAdminUsername(String adminUsername) {
        this.adminUsername = adminUsername;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public AdminTable(int adminId, String adminUsername, String password) {
        super();
        this.adminId = adminId;
        this.adminUsername = adminUsername;
        this.password = password;
    }
    public AdminTable(String adminUsername, String password) {
        super();
        this.adminUsername = adminUsername;
        this.password = password;
    }
    public AdminTable() {
        super();
    }


}
```

Package- com.spring.boot.sportyshoes.entities

Class- UserRegisted.java

Table create in DB- User_Registered

```java
package com.spring.boot.sportyshoes.entities;

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

@Entity
@Table(name="UserRegistered")
public class UserRegistered {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String username;
    private String fullName;
    private String password;
    private String address;
    private String cardType;
    private int cardNumber;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getFullName() {
        return fullName;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getCardType() {
        return cardType;
    }
    public void setCardType(String cardType) {
        this.cardType = cardType;
    }
    public int getCardNumber() {
        return cardNumber;
    }
    public void setCardNumber(int cardNumber) {
        this.cardNumber = cardNumber;
    }
    public UserRegistered() {
        super();
    }
    public UserRegistered(int id, String username, String fullName, String
password, String address, String cardType,
            int cardNumber) {
        super();
        this.id = id;
        this.username = username;
        this.fullName = fullName;
        this.password = password;
        this.address = address;
        this.cardType = cardType;
        this.cardNumber = cardNumber;
    }
    public UserRegistered(String username, String fullName, String password) {
        super();
        this.username = username;
        this.fullName = fullName;
        this.password = password;
    }





}
```

Package- com.spring.boot.sportyshoes.entities

Class- Product.java

Table create in DB- Product_sporty_shoes

```java
package com.spring.boot.sportyshoes.entities;

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

@Entity
@Table(name="Products_Sporty_Shoes")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;

    private String productName;
    private String shoeType;
    private int size;
    private String vendorName;
    private int quantity;
    private double mrp;
    public int getProductID() {
        return productID;
    }
    public void setProductID(int productID) {
        this.productID = productID;
    }
    public String getProductName() {
        return productName;
    }
    public void setProductName(String productName) {
        this.productName = productName;
    }
    public String getShoeType() {
        return shoeType;
    }
    public void setShoeType(String shoeType) {
        this.shoeType = shoeType;
    }
    public int getSize() {
        return size;
    }
    public void setSize(int size) {
        this.size = size;
    }
    public String getVendorName() {
        return vendorName;
    }
    public void setVendorName(String vendorName) {
        this.vendorName = vendorName;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public double getMrp() {
        return mrp;
    }
    public void setMrp(double mrp) {
        this.mrp = mrp;
    }

    public Product() {
        super();
    }
    public Product( String productName, String shoeType, int size, String
vendorName, int quantity,
            double mrp) {
        super();
        this.productName = productName;
        this.shoeType = shoeType;
        this.size = size;
        this.vendorName = vendorName;
        this.quantity = quantity;
        this.mrp = mrp;
    }
    public Product(int productID, String productName, String shoeType, int size,
String vendorName, int quantity,
            double mrp) {
        super();
        this.productID = productID;
        this.productName = productName;
```

```java
        this.shoeType = shoeType;
        this.size = size;
        this.vendorName = vendorName;
        this.quantity = quantity;
        this.mrp = mrp;
    }




}
```

Package- com.spring.boot.sportyshoes.entities

Class- OrderedBooked.java

Table create in DB- Ordered_Booked

```java
package com.spring.boot.sportyshoes.entities;



import java.time.LocalDate;

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

@Entity
@Table(name="Ordered_Booked")
public class OrderedBooked {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int orderId;
    private LocalDate orderDate;
    private String prodName;
    private int prodId;
    private String vendorName;
    private double pricePerPiece;
    private int quantityOrder;
    private double totalAmount;
    private String customerName;
    private String address;
    private long cardNumber;
    private String cardType;
    private String username;
    private int userId;

    public int getUserId() {
        return userId;
    }
    public void setUserId(int userId) {
        this.userId = userId;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public int getOrderId() {
        return orderId;
    }
    public void setOrderId(int orderId) {
        this.orderId = orderId;
    }
    public LocalDate getOrderDate() {
        return orderDate;
    }
    public void setOrderDate(LocalDate orderDate) {
        this.orderDate = orderDate;
    }
    public String getProdName() {
        return prodName;
    }
    public void setProdName(String prodName) {
        this.prodName = prodName;
    }
    public int getProdId() {
        return prodId;
    }
    public void setProdId(int prodId) {
        this.prodId = prodId;
    }
    public String getVendorName() {
        return vendorName;
    }
    public void setVendorName(String vendorName) {
        this.vendorName = vendorName;
    }
    public double getPricePerPiece() {
        return pricePerPiece;
    }
    public void setPricePerPiece(double pricePerPiece) {
        this.pricePerPiece = pricePerPiece;
    }
    public int getQuantityOrder() {
        return quantityOrder;
    }
    public void setQuantityOrder(int quantityOrder) {
        this.quantityOrder = quantityOrder;
    }
    public double getTotalAmount() {
        return totalAmount;
    }
    public void setTotalAmount(double totalAmount) {
        this.totalAmount = totalAmount;
    }

    public String getCustomerName() {
        return customerName;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public long getCardNumber() {
        return cardNumber;
    }
    public void setCardNumber(long cardNumber) {
        this.cardNumber = cardNumber;
    }
    public String getCardType() {
        return cardType;
    }
    public void setCardType(String cardType) {
        this.cardType = cardType;
    }
    public OrderedBooked() {
        super();
    }


    public OrderedBooked(LocalDate orderDate, String prodName, int prodId, String
vendorName, double pricePerPiece,
            int quantityOrder, double totalAmount, String customerName, String
address, long cardNumber,
            String cardType,String username,int userId) {
        super();
        this.orderDate = orderDate;
        this.prodName = prodName;
        this.prodId = prodId;
        this.vendorName = vendorName;
        this.pricePerPiece = pricePerPiece;
        this.quantityOrder = quantityOrder;
        this.totalAmount = totalAmount;
        this.customerName = customerName;
        this.address = address;
        this.cardNumber = cardNumber;
        this.cardType = cardType;
        this.username=username;
        this.userId=userId;
    }
    public OrderedBooked(int orderId, LocalDate orderDate, String prodName, int
prodId, String vendorName,
            double pricePerPiece, int quantityOrder, double totalAmount, String
customerName, String address,
            long cardNumber, String cardType,String username,int userId) {
        super();
        this.orderId = orderId;
        this.orderDate = orderDate;
        this.prodName = prodName;
        this.prodId = prodId;
        this.vendorName = vendorName;
        this.pricePerPiece = pricePerPiece;
        this.quantityOrder = quantityOrder;
        this.totalAmount = totalAmount;
        this.customerName = customerName;
        this.address = address;
        this.cardNumber = cardNumber;
        this.cardType = cardType;
        this.username=username;
        this.userId=userId;

    }







}
```

DAO interface for spring boot project-

Path- src/main/java

Package- com.spring.boot.sportyshoes.repositories

Interface- AdminDao.java

```
package com.spring.boot.sportyshoes.repositories;

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

import com.spring.boot.sportyshoes.entities.AdminTable;

@Repository
public interface AdminDao extends JpaRepository<AdminTable, Integer> {
//default functions like add-->save, delete,findall,find byid
}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.repositories

Interface- UserDao.java

```
package com.spring.boot.sportyshoes.repositories;

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

import com.spring.boot.sportyshoes.entities.UserRegistered;
```

```
@Repository
public interface UserDao extends JpaRepository<UserRegistered, Integer> {
//all by default methods are define here like add-->save, delete, findall,
findone
}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.repositories

Interface- ProductDao.java

```
package com.spring.boot.sportyshoes.repositories;

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

import com.spring.boot.sportyshoes.entities.Product;
@Repository
public interface ProductDao extends JpaRepository<Product, Integer> {
//all basic are added alread-- like add, findall, find one, delete
}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.repositories

Interface- OrderDao.java

```
package com.spring.boot.sportyshoes.repositories;

import java.util.List;

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

import com.spring.boot.sportyshoes.entities.OrderedBooked;

@Repository
public interface OrderDao extends JpaRepository<OrderedBooked, Integer>{
//by deafult all method -- add-save,delete, findAll,findone are there

    //sorted by orderId in orederdTable
    List<OrderedBooked> findByOrderByOrderIdAsc();

    //for descending list mention DESC
    List<OrderedBooked> findByOrderByOrderIdDesc();

    //for ascending orderdate
    List<OrderedBooked> findByOrderByOrderDate();

    //for descending orderDate
    List<OrderedBooked> findByOrderByOrderDateDesc();
}
```

Interface services will be here to work after Dao.

Path- src/main/java

Package- com.spring.boot.sportyshoes.services

Interface- AdminService.java

```
package com.spring.boot.sportyshoes.services;

import com.spring.boot.sportyshoes.entities.AdminTable;

public interface AdminService {

    //sign in the admin and search if is admin
    public Boolean signInAdmin(String adminName,String password);

    //if admin name is correct
    public Boolean checkAdminName(String adminName,String password);

    //add admin in the table-- AdminTable
    public AdminTable addAdmin(String adminName,String password);

    //to change the password of admin using - adminname
    public Boolean changePassword(String adminName, String password);
}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.services

Interface- UserService.java

```java
package com.spring.boot.sportyshoes.services;

import java.util.List;

import com.spring.boot.sportyshoes.entities.UserRegistered;

public interface UserService {

    //add the signUp details to UserRegistered table
    public boolean addUserSignUp(String username,String name, String password);

    //if userName available or not
    public boolean userNameAvailable(String username);

    //Check if username and password are correct and then provide him Signed In
Screen and fetch fullname of user
    public String signIn(String username, String password);

    //return account details from the user
    public List<String> accountDetails(String username);

    //Update the account Details of the user using username
    public Boolean accountDetailsUpdate(String username,String address,String
password,String card_type,int card_number);

    //list of all user signed up -- required by admin controller
    public List<String> allSignedUpUser();

    //search for the user by username is avialble in the databse of sportyshoes
    public Boolean searchUser(String username);

    //details of the user by userid
    public UserRegistered detailsUser(int userId);
}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.services

Interface- ProductService.java

```java
package com.spring.boot.sportyshoes.services;

import java.util.List;
import com.spring.boot.sportyshoes.entities.Product;

public interface ProductService {

    //to show all products in the product table
    public List<Product> getAllProducts();

    //to add the product
    public boolean addProduct(Product product);

    //to get One Product by productId-->use findById(ProductId).get()
    public Product getOneProduct(int productID);

    //to view all products name from table product
    public List<String> viewAllProducts();

    //to show Product details using productId
    public Product showProductDetails(int productId);

    //to update ProductName by using productId
    public Boolean updateProductName(int productId, String productName);

    //to update MSRP by using productId
    public Boolean updateProductMsrp(int productId, double msrp);

    //to update Quantity in stock by using productId
    public Boolean updateProductQuantityInStock(int productId, int quantity);

    //to update VendorName by using productId
    public Boolean updateProductVendorName(int productId, String vendorName);

    //to delete the product using productId
    public Boolean deleteProduct(int productId);
}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.services

Interface- OrderService.java

```java
package com.spring.boot.sportyshoes.services;
```

```java
import java.util.List;

import com.spring.boot.sportyshoes.entities.OrderedBooked;

public interface OrderService {

    //to add the Order details or Order Booking in table-- Order_Booked
    public Boolean addOrder(int productId,int quantity,String username);

    //to display the purchase History of the user
    public List<OrderedBooked> purchaseHistory(String username);

    //order by clause in asc on table orderedBooked
    public List<OrderedBooked> purcahseHistoryOrderByOrderId();

    //order by clause in desc on table orderedBooked
    public List<OrderedBooked> purcahseHistoryOrderByOrderIdDesc();

    //order by clause in asc on table orderedBooked--orderDate
    public List<OrderedBooked> purcahseHistoryOrderByOrderDate();

    //order by clause in desc on table orderedBooked--orderDate
    public List<OrderedBooked> purcahseHistoryOrderByOrderDateDesc();

    //list of order history by productId
    public List<OrderedBooked> orderHistoryByProdId(int productId);

    //list of order history by userId
    public List<OrderedBooked> orderHistoryByUserId(int userId);
}
```

After the service interfaces, the services are implemented using the another class serviceImplemnetation.

Path- src/main/java

Package- com.spring.boot.sportyshoes.servicesImpl

Class- AdminServiceImpl.java

```java
package com.spring.boot.sportyshoes.servicesImpls;

import java.util.LinkedHashSet;
import java.util.Set;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.spring.boot.sportyshoes.entities.AdminTable;
import com.spring.boot.sportyshoes.repositories.AdminDao;
import com.spring.boot.sportyshoes.services.AdminService;

@Service
public class AdminServiceImpl implements AdminService {

    @Autowired
    private AdminDao adminDao;



    //Check if Admin is valid or not
    @Override
    public Boolean signInAdmin(String adminName, String password) {
        boolean flag=false;
        Set<AdminTable> allAdmins=new
LinkedHashSet<AdminTable>(adminDao.findAll());
        for(AdminTable admin:allAdmins)
            if(admin.getAdminUsername().equals(adminName) &&
admin.getPassword().equals(password))
                flag=true;
        return flag;
    }


    //check adminName is Right
    @Override
    public Boolean checkAdminName(String adminName, String password) {
        boolean flag=false;
        Set<AdminTable> allAdmins=new
LinkedHashSet<AdminTable>(adminDao.findAll());
        for(AdminTable admin:allAdmins)
            if(admin.getAdminUsername().equals(adminName))
                flag=true;
        return flag;
    }


    //add the admin details in admin Table
    @Override
    public AdminTable addAdmin(String adminName, String password) {
        AdminTable admin=new AdminTable(adminName,password);
        adminDao.save(admin);
        return admin;
    }

    //to change the password using adminName
    @Override
    public Boolean changePassword(String adminName, String password) {
        boolean flag=false;
        Set<AdminTable> allAdmins=new
LinkedHashSet<AdminTable>(adminDao.findAll());

        for(AdminTable admin:allAdmins)
            if(admin.getAdminUsername().equals(adminName))
            {
                AdminTable newAdmin=new
AdminTable(admin.getAdminId(),admin.getAdminUsername(),password);
                adminDao.save(newAdmin);
                flag=true;
            }

        return flag;
    }





}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.servicesImpl

Class- UserServiceImpl.java

```
package com.spring.boot.sportyshoes.servicesImpls;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;
```

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

import com.spring.boot.sportyshoes.entities.UserRegistered;
import com.spring.boot.sportyshoes.repositories.UserDao;
import com.spring.boot.sportyshoes.services.UserService;

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserDao userDao;

    //add the SignUp user details in the table UserRegistered in the Database
    @Override
    public boolean addUserSignUp(String username, String name, String password) {
        Boolean flag=false;
            UserRegistered user=new UserRegistered(username, name, password);
            userDao.save(user);
            flag=true;
        return flag;
    }

    //check if userName available or NOT
    @Override
    public boolean userNameAvailable(String username) {
        Boolean flag=false;
        Set<UserRegistered> usernames=new
LinkedHashSet<UserRegistered>(userDao.findAll());
        for(UserRegistered user:usernames)
            if(user.getUsername().equals(username))
                flag=true;
        return flag;
    }


    //Check username and password coorect if correct then signed in
    @Override
    public String signIn(String username, String password) {
        String fullName="null";
        Set<UserRegistered> users=new
LinkedHashSet<UserRegistered>(userDao.findAll());
        for(UserRegistered user:users)
            if(user.getUsername().equals(username) &&
user.getPassword().equals(password))
```

```
                fullName=user.getFullName();
        return fullName;
    }


    //return Account details
    @Override
    public List<String> accountDetails(String username) {
        List<String> accDetails=new ArrayList<>();
        List<UserRegistered> lists=new
ArrayList<UserRegistered>(userDao.findAll());
        for(UserRegistered user:lists)
            if(user.getUsername().equals(username))
            {
                accDetails.add("Address: "+user.getAddress());
                accDetails.add("Card Type: "+user.getCardType());
                accDetails.add("Card Number: "+user.getCardNumber());

            }
        return accDetails;
    }


    //update the account details using username
    @Override
    public Boolean accountDetailsUpdate(String username, String address, String
password, String card_type,
            int card_number) {
        boolean flag=false;
        int userId;String fullName;
        Set<UserRegistered> listUser=new
LinkedHashSet<UserRegistered>(userDao.findAll());
        for(UserRegistered userOld:listUser)
            if(userOld.getUsername().equals(username))
                {userId=userOld.getId();
                fullName=userOld.getFullName();
                UserRegistered userUpdate=new
UserRegistered(userId,username,fullName,password,address,card_type,card_number);
                userDao.save(userUpdate);
                }
        flag=true;
        return flag;
    }
    //list of all signed up user required by admin controller
    @Override
    public List<String> allSignedUpUser() {
        List<String> allUser=new ArrayList<String>();
        Set<UserRegistered> regUser=new
LinkedHashSet<UserRegistered>(userDao.findAll());
        for(UserRegistered user:regUser)
                allUser.add("User name: "+user.getUsername() +" -- "+"Full Name:
"+user.getFullName() );
        return allUser;
    }

    //search if user is found in the UserRegistered table
    @Override
    public Boolean searchUser(String username) {
        boolean flag=false;
        Set<UserRegistered> regUser=new
LinkedHashSet<UserRegistered>(userDao.findAll());
        for(UserRegistered user:regUser)
            if(user.getUsername().equals(username))
                flag=true;
        return flag;
    }


    //all deatils of user by userId
    @Override
    public UserRegistered detailsUser(int userId) {
        UserRegistered user=userDao.findById(userId).get();
        return user;
    }




}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.servicesImpl

Class- ProductServiceImpl.java

```
package com.spring.boot.sportyshoes.servicesImpls;
```

```java
import java.util.ArrayList;
import java.util.List;

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

import com.spring.boot.sportyshoes.entities.Product;
import com.spring.boot.sportyshoes.repositories.ProductDao;
import com.spring.boot.sportyshoes.services.ProductService;

@Service
public class ProductServiceImpl implements ProductService {

    @Autowired
    private ProductDao productDao;

    //to get all the products inthe Product table
    @Override
    public List<Product> getAllProducts() {
        return productDao.findAll();
    }

    // to add the product in the Product Table
    @Override
    public boolean addProduct(Product product) {
        Boolean flag= false;
        Product pro=new
Product(product.getProductName(),product.getShoeType(),product.getSize(),product.
getVendorName(),product.getQuantity(),product.getMrp());
        productDao.save(pro);
        flag=true;
        return flag;
    }


    //to get One Product from Product table
    @Override
    public Product getOneProduct(int productId) {
        return productDao.findById(productId).get();
    }

    //view all the products names from product table
    @Override
    public List<String> viewAllProducts() {
        List<Product> allProducts=new ArrayList<Product>(productDao.findAll());
        List<String> allProductsName=new ArrayList<String>();
        for(Product product:allProducts)
            allProductsName.add(product.getProductName());
        return allProductsName;
    }


    //to show product details using productId
    @Override
    public Product showProductDetails(int productId) {
        List<Product> allProducts=new ArrayList<Product>(productDao.findAll());
        for(Product product:allProducts)
            if(product.getProductID()==productId)
                return product;
        return null;
    }

    //to update ProductName using ProductId
    @Override
    public Boolean updateProductName(int productId,String productName) {
        boolean flag=false;
        List<Product> allProducts=new ArrayList<Product>(productDao.findAll());
        for(Product product:allProducts)
            if(product.getProductID()==productId)
                {   Product prod=new
Product(product.getProductID(),productName,product.getShoeType(),product.getSize(
),product.getVendorName(),product.getQuantity(),product.getMrp());
                    productDao.save(prod);
                    flag=true;
                }
        return flag;
    }

    //to update MSRP using ProductId
    @Override
    public Boolean updateProductMsrp(int productId, double msrp) {
        boolean flag=false;
        List<Product> allProducts=new ArrayList<Product>(productDao.findAll());
        for(Product product:allProducts)
            if(product.getProductID()==productId)
                {   Product prod=new
Product(product.getProductID(),product.getProductName(),product.getShoeType(),pro
duct.getSize(),product.getVendorName(),product.getQuantity(),msrp);
                    productDao.save(prod);
                    flag=true;
                }
        return flag;
    }


    //to update QuantityInStock using ProductId
    @Override
    public Boolean updateProductQuantityInStock(int productId, int quantity) {
        boolean flag=false;
        List<Product> allProducts=new ArrayList<Product>(productDao.findAll());
        for(Product product:allProducts)
            if(product.getProductID()==productId)
                {   Product prod=new
Product(product.getProductID(),product.getProductName(),product.getShoeType(),pro
duct.getSize(),product.getVendorName(),quantity,product.getMrp());
                    productDao.save(prod);
                    flag=true;
                }
        return flag;
    }


    //to update VendorName using ProductId
    @Override
    public Boolean updateProductVendorName(int productId, String vendorName) {
        boolean flag=false;
        List<Product> allProducts=new ArrayList<Product>(productDao.findAll());
        for(Product product:allProducts)
            if(product.getProductID()==productId)
                {   Product prod=new
Product(product.getProductID(),product.getProductName(),product.getShoeType(),pro
duct.getSize(),vendorName,product.getQuantity(),product.getMrp());
                    productDao.save(prod);
                    flag=true;
                }
        return flag;
    }


    //delete the product using productId
    @Override
    public Boolean deleteProduct(int productId) {
        boolean flag=false;
```

```
            Product prod=productDao.getById(productId); // or can use
.getOne(prodId)
            productDao.delete(prod);
            flag=true;
        return flag;
    }




}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.servicesImpl

Class- OrderServiceImpl.java

```
package com.spring.boot.sportyshoes.servicesImpls;

import java.sql.Date;
import java.sql.Timestamp;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

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

import com.spring.boot.sportyshoes.entities.OrderedBooked;
import com.spring.boot.sportyshoes.entities.Product;
import com.spring.boot.sportyshoes.entities.UserRegistered;
import com.spring.boot.sportyshoes.repositories.OrderDao;
import com.spring.boot.sportyshoes.repositories.ProductDao;
import com.spring.boot.sportyshoes.repositories.UserDao;
import com.spring.boot.sportyshoes.services.OrderService;
@Service
public class OrderServiceImpl implements OrderService {

    @Autowired
```

```java
    private OrderDao orderDao;

    @Autowired
    private ProductDao productDao;

    @Autowired
    private UserDao userDao;

    @Override
    public Boolean addOrder(int productId, int quantity,String username) {
        LocalDate localDate=LocalDate.now();
        LocalDate Date=LocalDate.parse("2021-05-14");
        Timestamp
localDateTime=Timestamp.valueOf(Date.atTime(LocalTime.MIDNIGHT));
        Product pro=productDao.findById(productId).get();
        //search for user by uname
        UserRegistered userData=null;
        Set<UserRegistered> allUser=new
LinkedHashSet<UserRegistered>(userDao.findAll());
        for(UserRegistered user:allUser)
            if(user.getUsername().equals(username))
                userData=user;
//      UserRegistered user=userDao.findById(userId).get();
        OrderedBooked order=new
OrderedBooked(localDate,pro.getProductName(),pro.getProductID(),pro.getVendorName
(),pro.getMrp(),quantity,quantity*pro.getMrp(),userData.getFullName(),userData.ge
tAddress(),userData.getCardNumber(),userData.getCardType(),username,userData.getI
d());
        orderDao.save(order);
        //product quantity update in table -Product
        Product p=new
Product(productId,pro.getProductName(),pro.getShoeType(),pro.getSize(),pro.getVen
dorName(),pro.getQuantity()-quantity,pro.getMrp());
        productDao.save(p);
        return true;
    }



    //to dispaly the ordered booked in purchase history of user
    @Override
    public List<OrderedBooked> purchaseHistory(String username) {
        List<OrderedBooked> allPurchase=new ArrayList<OrderedBooked>();
        List<OrderedBooked> purchase=new
ArrayList<OrderedBooked>(orderDao.findAll());
```

```
        for(OrderedBooked order:purchase)
            if(username.equals(order.getUsername()))
                allPurchase.add(order);
        return allPurchase;
    }


    //Order by clause on orderId in OrderedBooked table --purchase history
    @Override
    public List<OrderedBooked> purcahseHistoryOrderByOrderId() {
        return orderDao.findByOrderByOrderIdAsc();
    }

    //Order by clause on orderId in OrderedBooked table --purchase history
    @Override
    public List<OrderedBooked> purcahseHistoryOrderByOrderIdDesc() {
        return orderDao.findByOrderByOrderIdDesc();
    }

    //Order by clause on orderId in OrderedBooked table --purchase history
    @Override
    public List<OrderedBooked> purcahseHistoryOrderByOrderDate() {
        return orderDao.findByOrderByOrderDate();
    }

    //Order by clause on orderId in OrderedBooked table --purchase history
    @Override
    public List<OrderedBooked> purcahseHistoryOrderByOrderDateDesc() {
        return orderDao.findByOrderByOrderDateDesc();
    }


    //view all order history by productId
    @Override
    public List<OrderedBooked> orderHistoryByProdId(int productId) {
        List<OrderedBooked> allPurchase=new ArrayList<OrderedBooked>();
        List<OrderedBooked> purchase=new
ArrayList<OrderedBooked>(orderDao.findAll());
        for(OrderedBooked order:purchase)
            if(productId==order.getProdId())
                allPurchase.add(order);
        return allPurchase;
    }
    //view all order history by userId
    @Override
    public List<OrderedBooked> orderHistoryByUserId(int userId) {
        List<OrderedBooked> allPurchase=new ArrayList<OrderedBooked>();
        List<OrderedBooked> purchase=new
ArrayList<OrderedBooked>(orderDao.findAll());
        for(OrderedBooked order:purchase)
            if(userId==order.getUserId())
                allPurchase.add(order);
        return allPurchase;
    }


}
```

After service implementations are completed, all these services will be called by Controllers of Admin and User via springboot swagger 2 or postman using URL.

Path- src/main/java

Package- com.spring.boot.sportyshoes.controllers

Class- AdminController.java

```
package com.spring.boot.sportyshoes.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
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.spring.boot.sportyshoes.entities.AdminTable;
import com.spring.boot.sportyshoes.entities.OrderedBooked;
import com.spring.boot.sportyshoes.entities.Product;
import com.spring.boot.sportyshoes.entities.UserRegistered;
import com.spring.boot.sportyshoes.servicesImpls.AdminServiceImpl;
import com.spring.boot.sportyshoes.servicesImpls.OrderServiceImpl;
import com.spring.boot.sportyshoes.servicesImpls.ProductServiceImpl;
import com.spring.boot.sportyshoes.servicesImpls.UserServiceImpl;
```

```
@RestController
public class AdminController {

    @Autowired
    private AdminServiceImpl adminServiceImpl;
    @Autowired
    private UserServiceImpl userServiceImpl;
    @Autowired
    private ProductServiceImpl productServiceImpl;
    @Autowired
    private OrderServiceImpl orderServiceImpl;

        //to signIn user admin -- if admin is valid Admin
        @GetMapping("/auth/signIn/{adminname}/{password}")
        public String signInAdmin(@PathVariable String adminname,@PathVariable
String password)
        {
            if(adminServiceImpl.signInAdmin(adminname, password))
                return "You are valid ADMIN- "+adminname+" !! Welcome to Admin
page.";
            else if(adminServiceImpl.checkAdminName(adminname, password))
                return "Admin your password is wrong!!";
            else
                return "Not a valid admin";
        }

        //add the admin details
        @PostMapping("/auth/addAdmin/{adminname}/{password}")
        public AdminTable addAdmin(@PathVariable String adminname,@PathVariable
String password) {
            return adminServiceImpl.addAdmin(adminname, password);
        }

        //to signout the adminFunctions
        @GetMapping("/admin/signOut")
        public String adminSignOut()
        {
            return "Admin is succesfully Signed Out.";
        }


        //to change admin password by taking adminname as arguments
        @PutMapping("/admin/changePassword/{adminName}/{password}")
        public String changePassword(@PathVariable String adminName,@PathVariable 
String password)
        {   if(adminServiceImpl.changePassword(adminName, password))
                return adminName+" , your password has been sucessfully changed 
to "+ password;
        else
            return "Issue in the network cannot change the password";
        }

        //View list of all the user that are sugned up
        @GetMapping("/admin/allSignedUpUser")
        public List<String> allSignedUpUser()
        {
            return userServiceImpl.allSignedUpUser();
        }

        //Username is user avilable or not in databse
        @GetMapping("/admin/searchUser/{username}")
        public String searchUser(@PathVariable String username)
        {   if(userServiceImpl.searchUser(username))
                return username+" found is registered on the SportyShoes!!";
            else
                return "No such user exists in the SportyShoes!!";
        }


        //View the details of user by userId from table-- UserRegistered
        @GetMapping("/admin/detailsOfUser/{userId}")
        public UserRegistered detailsOfUser(@PathVariable int userId)
        {
            return userServiceImpl.detailsUser(userId);
        }

        //List of all products name in the Product table
        @GetMapping("/admin/allProducts")
        public List<String> allProducts()
        {
            return productServiceImpl.viewAllProducts();
        }

        //Add a new product in the product table
        @PostMapping("/admin/addProduct")
        public String addTheProduct(@RequestBody Product product)
        {   if(productServiceImpl.addProduct(product))
                return "Product added successfully";
            else
                return "some error occurred";
        }

        //view details of product by ProductId
        @GetMapping("/admin/showProductDetails/{productId}")
        public Product showProductDetails(@PathVariable int productId)
        {
            return productServiceImpl.showProductDetails(productId);
        }

        //update the name of product using product id
        @PutMapping("/admin/updateProductName/{productId}/{productName}")
        public String updateProductName(@PathVariable int productId,@PathVariable
String productName)
        {   if(productServiceImpl.updateProductName(productId, productName))
                return "product name of productId: "+productId+" is updated to:
"+productName;
            else
                return "Some issue occured , product not updated.";
        }

        //update the MSRP of product using product id
        @PutMapping("/admin/updateProductMSRP/{productId}/{msrp}")
        public String updateProductMsrp(@PathVariable int productId,@PathVariable
double msrp)
        {   if(productServiceImpl.updateProductMsrp(productId, msrp))
                return "Product MSRP of productId: "+productId+" is updated to:
"+msrp;
            else
                return "Some issue occured , product not updated.";
        }

        //update the QuantityInStock of product using product id
        @PutMapping("/admin/updateProductQuantityInStock/{productId}/{quantity}")
        public String updateProductQuantityInStock(@PathVariable int
productId,@PathVariable int quantity)
        {   if(productServiceImpl.updateProductQuantityInStock(productId,
quantity))
                return "Product Quantity in Stock  of productId: "+productId+" is
updated to: "+quantity;
            else
                return "Some issue occured , product not updated.";
        }
```

```java
        //update the VenodorName of product using product id
        @PutMapping("/admin/updateProductVendorName/{productId}/{vendorName}")
        public String updateProductVendorName(@PathVariable int productId,
@PathVariable String vendorName)
        {   if(productServiceImpl.updateProductVendorName(productId, vendorName))
                return "Product Vendor name of productId: "+productId+" is
updated to: "+vendorName;
            else
                return "Some issue occured , product not updated.";
        }

        //to delete the product item using productId
        @DeleteMapping("/admin/deleteProduct/{productId}")
        public String deleteProduct(@PathVariable int productId) {
            if(productServiceImpl.deleteProduct(productId))
                return "Product item deleted succesfully!!";
            else
                return "delete of product was unsuccesful";
        }

        //View the list of all orders(history purcase) order by orderId asc
        @GetMapping("/admin/orderHistoryAsc")
        public List<OrderedBooked> purchasedHistoryOrderByOrderId()
        {
            return orderServiceImpl.purcahseHistoryOrderByOrderId();
        }

        //View the list of all orders(history purcase) order by orderId asc
        @GetMapping("/admin/orderHistoryDesc")
        public List<OrderedBooked> purchasedHistoryOrderByOrderIdDesc()
        {
            return orderServiceImpl.purcahseHistoryOrderByOrderIdDesc();
        }

        //View the list of all orders(history purcase) order by orderId asc
        @GetMapping("/admin/orderHistoryOrderDateAsc")
        public List<OrderedBooked> purchasedHistoryOrderByOrderDate()
        {
            return orderServiceImpl.purcahseHistoryOrderByOrderDate();
        }

        //View the list of all orders(history purcase) order by orderId asc
        @GetMapping("/admin/orderHistoryOrderDateDesc")
        public List<OrderedBooked> purchasedHistoryOrderByOrderDateDesc()
        {
            return orderServiceImpl.purcahseHistoryOrderByOrderDateDesc();
        }

        //view all orders using product pid
        @GetMapping("/admin/orderHistory/productId/{productId}")
        public List<OrderedBooked> orderHistoryByProdId(@PathVariable int
productId){
            return orderServiceImpl.orderHistoryByProdId(productId);
        }

        //view all orders using userId
        @GetMapping("/admin/orderHistory/userId/{userId}")
        public List<OrderedBooked> orderHistoryByUserId(@PathVariable int
userId){
            return orderServiceImpl.orderHistoryByUserId(userId);
        }


}
```

Path- src/main/java

Package- com.spring.boot.sportyshoes.controllers

Class- UserController.java

```
package com.spring.boot.sportyshoes.controllers;

import java.util.List;

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.spring.boot.sportyshoes.entities.OrderedBooked;
import com.spring.boot.sportyshoes.entities.Product;
import com.spring.boot.sportyshoes.entities.UserRegistered;
import com.spring.boot.sportyshoes.servicesImpls.OrderServiceImpl;
import com.spring.boot.sportyshoes.servicesImpls.ProductServiceImpl;
import com.spring.boot.sportyshoes.servicesImpls.UserServiceImpl;
```

```java
@RestController
public class UserController {

    @Autowired
    UserServiceImpl userServiceImpl;
    @Autowired
    ProductServiceImpl productServiceImpl;
    @Autowired
    OrderServiceImpl orderServiceImpl;

    //to display WELCOME menu at SportyShoes
    @GetMapping("/home")
    public String home()
    {
        return "Welcome to SportyShoes Website!!!";
    }


    //add the UserSignUp details --> tb: UserRegistered
    @PostMapping("/signUp")
    public String addUserSignUp(@RequestBody UserRegistered user)
    {   if(userServiceImpl.userNameAvailable(user.getUsername()))
            return "UserName: "+user.getUsername()+" not available!! please try
another username.";
        else
        {
        if(userServiceImpl.addUserSignUp(user.getUsername(), user.getFullName(),
user.getPassword()))
            return "Congratulation!!! "+user.getFullName()+",you have
successfully registered at SportyShoes.";
        else
            return "User :"+user.getFullName()+" not added to SportyShoes
registration.";
        }
    }


    //signIn
    @GetMapping("/signIn/{username}/{password}")
    public String signIn(@PathVariable String username,@PathVariable String
password)
    {   if(!userServiceImpl.signIn(username, password).equals("null"))
```

```java
            return userServiceImpl.signIn(username, password) +". You are
successfully signed In.";
        //here userNAmeAvail method used for check if password is wrong but
username correct
        else
        {
            if(userServiceImpl.signIn(username, password).equals("null") &&
userServiceImpl.userNameAvailable(username))
                return "Your password is INCORRECT. I think you forget your
password!!";
            else
                return "Please Register Yourself first!! Go to signUp page!!";
        }

    }


    //signedOut
    @GetMapping("/signOut")
    public String signOut() {
        return "Bie!! you r successfully Signed Out.";
    }


//  //add the product
//  @PostMapping("/addTheProduct")
//  public String addTheProduct(@RequestBody Product product)
//  {   if(productServiceImpl.addProduct(product))
//          return "Product added successfully";
//      else
//          return "some error occurred";
//  }

    //display the results of the list in the product
    @GetMapping("/signIn/showAllProducts")
    public List<Product> showAllProducts()
    {
        return productServiceImpl.getAllProducts();
    }


    //get one product by productId
    @GetMapping("/signIn/getOneProduct/{productId}")
    public Product getThatProduct(@PathVariable int productId)
    {
        return productServiceImpl.getOneProduct(productId);
    }

    //Product purchase by the productId and quantity suing method addOrder
    @PostMapping("/signIn/purchaseProduct/{productId}/{quantity}/{username}")
    public String productPurchased(@PathVariable int productId,@PathVariable int
quantity,@PathVariable String username)
    {   if(orderServiceImpl.addOrder(productId, quantity,username))
            return "ordered is succefully placed.";
        else
            return "Failed placing ordered.";
    }


    //to display ACCount Details of the customer who is signed in
    @GetMapping("/signIn/accountDetails/{username}")
    public List<String> accountDetails(@PathVariable String username)
    {
        return userServiceImpl.accountDetails(username);
    }

    //to display Purchase history of specfic user by userId
    @GetMapping("/signIn/purchaseHistory/{username}")
    public List<OrderedBooked> purchaseHistory(@PathVariable String username)
    {
        return orderServiceImpl.purchaseHistory(username);
    }

    //to edit USer account details using username in user_registered table
    @PutMapping("/signIn/editAccountDetails")
    public String editAccountDetails(@RequestBody UserRegistered user)
    {   if(userServiceImpl.accountDetailsUpdate(user.getUsername(),
user.getAddress(), user.getPassword(), user.getCardType(), user.getCardNumber()))
            return "Your account has been updated mr./mrs. "+user.getUsername()+"
.";
        else
            return "There's issue while updating...Check your network
connection.";
    }
}
```