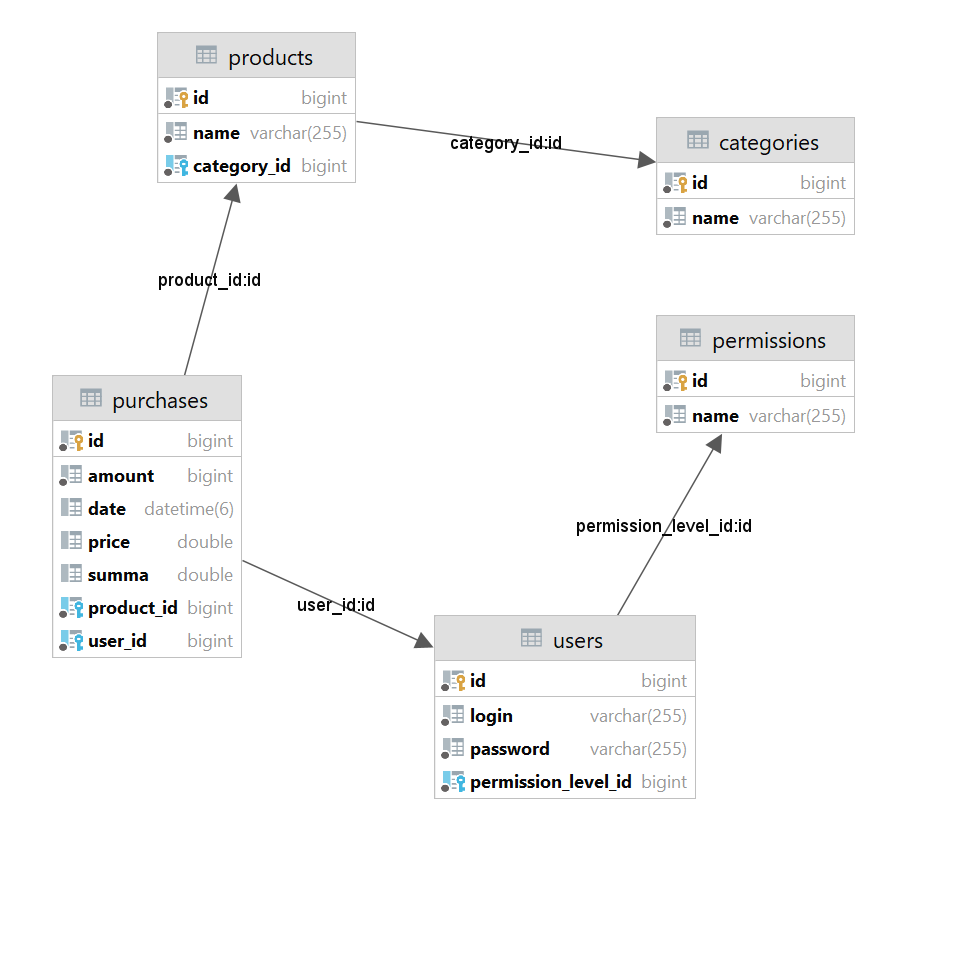
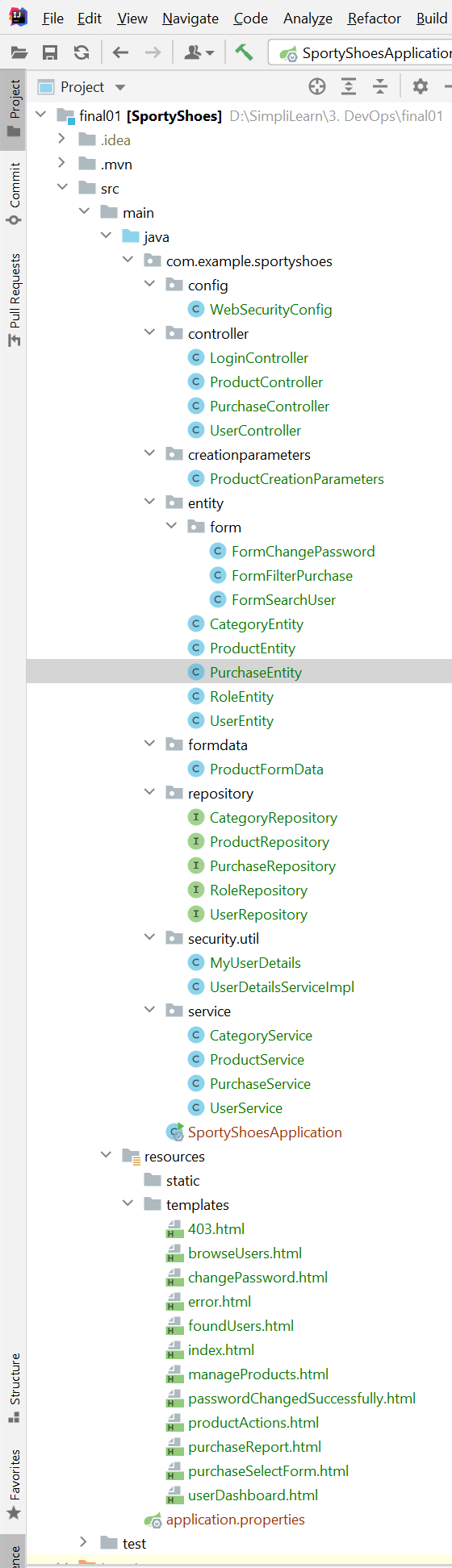
SQL Server side.

Database diagram:



Project structure:



Thymeleaf templates:

403.html:

<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Access Denied</title>  
</head>  
<body>  
 <h2>Sorry, you don't have permission!</h2>  
</body>  
</html>

browseUsers.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<a href="/">Home</a><br>  
  
<div align="left">  
 <h2> Search users by part of name</h2>  
 <form method="post" th:action="@{/manageUsers/search}" th:object="${formSearchUser}">  
 <input th:field="\*{name}">  
 <button type="submit">Search</button>  
  
 </form>  
</div>  
  
  
<h2> List of users</h2>  
<table>  
 <tr>  
 <th>ID</th>  
 <th>Login</th>  
 <th>Role</th>  
 </tr>  
  
 <tr th:if="${#lists.isEmpty(userList)}">  
 <td colspan="7">No Records found. Add some...!</td>  
 </tr>  
  
 <tr th:each="user : ${userList}">  
 <td th:text="${user.id}"></td>  
 <td th:text="${user.login}"></td>  
 <td th:text="${user.permissionLevel.name}"></td>  
 </tr>  
  
</table>  
</body>  
</html>

changePassword.jsp:

<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>  
<!DOCTYPE html>  
<html>  
<head>  
 <title>FlyAway</title>  
</head>  
<body>  
<h1> List of compatible flights</h1>  
  
<c:forEach var="res" items="${sRes}" >  
 <form method=post action="bookFlight.jsp">  
 Departure from  
 ${res.departure} at  
 ${res.departureDate} ------> arrival to  
 ${res.destination} at  
 ${res.arrivalDate} operated by  
 ${res.airline} availaible  
 ${res.amount} tickets  
  
 <input type="hidden" name="flightID" value=${res.id}>  
 <input type="hidden" name="amount" value=${res.amount}>  
 <input type="hidden" name="departure" value=${res.departure}>  
 <input type="hidden" name="destination" value=${res.destination}>  
 <input type="hidden" name="departureDate" value=${res.departureDate}>  
 <input type="hidden" name="arrivalDate" value=${res.arrivalDate}>  
 <input type="hidden" name="airline" value=${res.airline}>  
 <input type="hidden" name="needed" value=${paramValues.number[0]}>  
  
 <button type="submit">Book</button>  
 </form> <br>  
</c:forEach>  
  
</body>  
</html> <!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<a href="/">Home</a><br>  
  
<div align="left">  
 <h2> Search users by part of name</h2>  
 <form method="post" th:action="@{/manageUsers/search}" th:object="${formSearchUser}">  
 <input th:field="\*{name}">  
 <button type="submit">Search</button>  
  
 </form>  
</div>  
  
  
<h2> List of users</h2>  
<table>  
 <tr>  
 <th>ID</th>  
 <th>Login</th>  
 <th>Role</th>  
 </tr>  
  
 <tr th:if="${#lists.isEmpty(userList)}">  
 <td colspan="7">No Records found. Add some...!</td>  
 </tr>  
  
 <tr th:each="user : ${userList}">  
 <td th:text="${user.id}"></td>  
 <td th:text="${user.login}"></td>  
 <td th:text="${user.permissionLevel.name}"></td>  
 </tr>  
  
</table>  
</body>  
</html>

error.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<a href="/">Home</a><br>  
<h1> Something wrong!!! </h1>  
</body>  
</html>

foundUsers.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<h2> Found users:</h2>  
<table>  
 <tr>  
 <th>ID</th>  
 <th>Login</th>  
 <th>Role</th>  
 </tr>  
  
 <tr th:if="${#lists.isEmpty(foundUsers)}">  
 <td colspan="7">No Records found. Add some...!</td>  
 </tr>  
  
 <tr th:each="user : ${foundUsers}">  
 <td th:text="${user.id}"></td>  
 <td th:text="${user.login}"></td>  
 <td th:text="${user.permissionLevel.name}"></td>  
 </tr>  
  
</table>  
</body>  
</html>

index.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity5">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<div sec:authorize="isAuthenticated()">  
 Welcome <b><span sec:authentication="name">Username</span></b>  
 &nbsp;  
 <i><span sec:authentication="principal.authorities">Roles</span></i>  
</div>  
  
<form th:action="@{/logout}" method="post">  
 <input type="submit" value="Logout" />  
</form>  
  
<br>  
  
<div sec:authorize="hasAnyAuthority('Administrator')">  
 <a href="/manageProducts/main">Manage the products in the store</a> <br>  
 <a href="/manageUsers/main">Browse the list of users who have signed up</a><br>  
 <a href="/managePurchases/makeReport">See purchase reports</a><br>  
 <a href="/manageUsers/changePassword">Change password</a><br>  
</div>  
  
  
<div sec:authorize="hasAnyAuthority('Administrator', 'User')">  
 <a href="/userDashboard">User dashboard</a><br>  
</div>  
  
</body>  
</html>

manageProducts.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<a href="/">Home</a><br>  
  
<div align="left">  
 <h2> Adding product</h2>  
 <form method="post" th:action="@{/manageProducts/addProduct}" th:object="${productFormData}">  
 <table width="50%" border="0">  
 <tr valign="top">  
 <td width="40%">Name of product</td>  
 <td><input th:field="\*{name}"></td>  
 <td th:if="${#fields.hasErrors('name')}"  
 th:text="${#strings.listJoin(#fields.errors('name'), ', ')}"></td>  
  
 </tr>  
 <tr valign="top">  
 <td width="40%">Category of product</td>  
 <td>  
 <select th:field="\*{category}">  
 <option value="">Nothing selected</option>  
 <option th:each="category :${categoryList}"  
 th:value="${category.id}"  
 th:text="${category.name}">  
 </select>  
 </td>  
 <td th:if="${#fields.hasErrors('category')}"  
 th:text="${#strings.listJoin(#fields.errors('category'), ', ')}"></td>  
 </tr>  
 <tr valign="top">  
 <td colspan=2 width="100%">  
 <button type="submit">Add product</button>  
 </td>  
 </tr>  
 </table>  
 </form>  
</div>  
  
  
<h2> List of product</h2>  
<table>  
 <tr>  
 <th>ID</th>  
 <th>Name</th>  
 <th>Category</th>  
 <th>Actions</th>  
 </tr>  
  
 <tr th:if="${#lists.isEmpty(productList)}">  
 <td colspan="7">No Records found. Add some...!</td>  
 </tr>  
  
 <tr th:each="product : ${productList}">  
 <td th:text="${product.id}"></td>  
 <td th:text="${product.name}"></td>  
 <td th:text="${product.category.name}"></td>  
 <td><a th:href="'/manageProducts/actions' + ${product.id}">✏</a></td>  
 </tr>  
</table>  
  
  
  
</body>  
</html>

passwordChangedSuccessfully.html:

<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Sporty shoes</title>  
</head>  
<body>  
  
<a href="/">Home</a><br>  
<h2>Password has been changed successfully!</h2>  
  
</body>  
</html>

productActions.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<div align="left">  
 <h2> Product actions</h2>  
 <form method="post" th:action= "@{/manageProducts/update}" th:object="${productFormData}">  
 <input type="hidden" th:field="\*{id}">  
 <table width="50%" border="0">  
 <tr valign="top">  
 <td width="40%">Name of product</td>  
 <td><input th:field="\*{name}"></td>  
 </tr>  
 <tr valign="top">  
 <td width="40%">Category of product</td>  
 <td>  
 <select th:field="\*{category}">  
 <option th:each="category :${categoryList}"  
 th:value="${category.id}"  
 th:text="${category.name}">  
 </select>  
 </td>  
 </tr>  
 <tr valign="top">  
 <td colspan=2 width="100%">  
 <button type="submit">Update product</button>  
 <button type="submit" th:formaction="'/manageProducts/delete'+${productFormData.id}">Delete product</button>  
 </td>  
 </tr>  
 </table>  
 </form>  
</div>  
  
</body>  
</html>

purchaseReport.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
  
<h2> Purchase report</h2>  
  
<th:block th:if="${formFilterPurchase.from != null || formFilterPurchase.to != null }">  
 <p>  
 Purchase  
 <th:block th:if="${formFilterPurchase.from != null}">  
 from <span th:text="${#temporals.format(formFilterPurchase.from, 'dd-MM-yyyy HH:mm')}" > </span>  
 </th:block>  
  
 <th:block th:if="${formFilterPurchase.to != null}">  
 to <span th:text="${#temporals.format(formFilterPurchase.to, 'dd-MM-yyyy HH:mm')}"> </span>  
 </th:block>  
 </p>  
</th:block>  
  
  
<th:block th:if="${formFilterPurchase.category != null}">  
 <p> Category: <span th:text="${formFilterPurchase.category.name}"> </span> </p>  
</th:block>  
  
  
<table>  
 <tr>  
 <th>ID</th>  
 <th>User</th>  
 <th>Product</th>  
 <th>Category</th>  
 <th>Date of purchase</th>  
 <th>Price</th>  
 <th>Amount</th>  
 <th>Sum</th>  
 </tr>  
  
 <tr th:if="${#lists.isEmpty(purchaseList)}">  
 <td colspan="7">No Records found. Add some...!</td>  
 </tr>  
  
 <tr th:each="purchase : ${purchaseList}">  
 <td th:text="${purchase.id}"></td>  
 <td th:text="${purchase.user.login}"></td>  
 <td th:text="${purchase.product.name}"></td>  
 <td th:text="${purchase.product.category.name}"></td>  
 <td th:text="${#temporals.format(purchase.date, 'dd-MM-yyyy HH:mm')}"></td>  
 <td th:text="${purchase.price}"></td>  
 <td th:text="${purchase.amount}"></td>  
 <td th:text="${purchase.summa}"></td>  
  
 </tr>  
</table>  
  
  
</body>  
</html>

purchaseSelectForm.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<a href="/">Home</a><br>  
  
<div align="left">  
 <h2> Specify purchase report parameters</h2>  
 <form method="post" th:action="@{/managePurchases/show}" th:object="${formFilterPurchase}">  
 <table width="50%" border="0">  
 <tr valign="top">  
 <td width="40%">Purchases from date</td>  
 <td><input type=datetime-local th:field="\*{from}"></td>  
 </tr>  
  
 <tr valign="top">  
 <td width="40%">Purchases to date</td>  
 <td><input type=datetime-local th:field="\*{to}"></td>  
 </tr>  
 <tr valign="top">  
 <td width="40%">Category of product</td>  
 <td>  
 <select th:field="\*{category}">  
 <option value="">Nothing selected</option>  
 <option th:each="category :${categoryList}"  
 th:value="${category.id}"  
 th:text="${category.name}">  
 </select>  
 </td>  
 </tr>  
 <tr valign="top">  
 <td colspan=2 width="100%">  
 <button type="submit">Make report</button>  
 </td>  
 </tr>  
 </table>  
 </form>  
</div>

userDashboard.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Sporty shoes</title>  
</head>  
<body>  
  
<a href="/">Home</a><br>  
  
<h1>  
 <left>  
 <span style='color:red'> User tools!!</span>  
 </left>  
</h1>  
</body>  
</html>

Pom.xml:

*<?*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>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.7.1</version>  
 <relativePath/> *<!-- lookup parent from repository -->* </parent>  
 <groupId>com.example</groupId>  
 <artifactId>SportyShoes</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>SportyShoes</name>  
 <description>Demo project for Spring Boot</description>  
 <properties>  
 <java.version>1.8</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.thymeleaf.extras</groupId>  
 <artifactId>thymeleaf-extras-springsecurity5</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-ehcache</artifactId>  
 <version>5.6.8.Final</version>  
 </dependency>  
  
 *<!--Generate metamodel classes ClassName\_ -->* <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-jpamodelgen</artifactId>  
 <version>4.3.8.Final</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.validation</groupId>  
 <artifactId>validation-api</artifactId>  
 <version>2.0.1.Final</version>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <configuration>  
 <excludes>  
 <exclude>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </exclude>  
 </excludes>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

application.properties:

spring.jpa.hibernate.ddl-auto=update  
spring.datasource.url=jdbc:mysql://localhost:3306/sporty\_shoes?useSSL=false  
spring.datasource.username=root  
spring.datasource.password=vestaeagle  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
server.port=8081  
  
*# Show all queries*spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true  
logging.level.org.hibernate.type=trace  
spring.jpa.properties.hibernate.connection.isolation=2  
  
  
*#L2 Cache  
#1. enable second level cache*spring.jpa.properties.hibernate.cache.use\_second\_level\_cache=true  
*#2. specify the caching framework - EhCache*spring.jpa.properties.hibernate.cache.region.factory\_class=org.hibernate.cache.ehcache.EhCacheRegionFactory  
*#3. Only cache what I tell to cache.*spring.jpa.properties.javax.persistence.sharedCache.mode=ENABLE\_SELECTIVE

Application:

ackage com.example.sportyshoes;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SportyShoesApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SportyShoesApplication.class, args);  
 }  
  
}

Repositories.

UserRepository:

package com.example.sportyshoes.repository;  
  
import com.example.sportyshoes.entity.\*;  
import org.springframework.data.domain.Sort;  
import org.springframework.data.jpa.domain.Specification;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;  
import org.springframework.stereotype.Repository;  
  
import javax.persistence.criteria.CriteriaBuilder;  
import javax.persistence.criteria.CriteriaQuery;  
import javax.persistence.criteria.Predicate;  
import javax.persistence.criteria.Root;  
import java.util.ArrayList;  
import java.util.List;  
  
@Repository  
public interface UserRepository extends JpaRepository<UserEntity,Long>, JpaSpecificationExecutor<UserEntity> {  
 UserEntity findUserEntityByLogin(String name);  
  
 *// @Query("SELECT u from users u where u.login like :name")  
 // List<UserEntity> findUsersWithPartOfName(@Param("name") String name, Sort sort);* default List<UserEntity> findUsersWithPartOfName(String name, Sort sort) {  
 return findAll(new Specification<UserEntity>() {  
 @Override  
 public Predicate toPredicate(Root<UserEntity> root, CriteriaQuery<?> query, CriteriaBuilder criteriaBuilder) {  
  
 List<Predicate> list=new ArrayList<>();  
  
 if (name!=null){  
 return  
 criteriaBuilder.and(  
 criteriaBuilder.like(  
 root.get(UserEntity\_.*login*),name));  
  
 }  
 return null;  
 }  
 }, sort);  
 }  
  
  
}

RoleRepository:

package com.example.sportyshoes.repository;  
  
import com.example.sportyshoes.entity.RoleEntity;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface RoleRepository extends JpaRepository<RoleEntity,Long> {  
}

PurchaseRepository:

package com.example.sportyshoes.repository;  
  
import com.example.sportyshoes.entity.\*;  
import org.springframework.data.jpa.domain.Specification;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;  
import org.springframework.stereotype.Repository;  
  
import javax.persistence.criteria.\*;  
import java.time.LocalDateTime;  
import java.util.ArrayList;  
import java.util.List;  
  
@Repository  
public interface PurchaseRepository extends JpaRepository<PurchaseEntity,Long>, JpaSpecificationExecutor<PurchaseEntity> {  
  
  
 *//Initially i did JPQL Querry, but next time i think that i need Criteria API  
 //@Query("SELECT p from purchases p where p.date between :from and :to and p.product.category=:category")  
 //List<PurchaseEntity> getPurchaseReportByDateIntervalAndCategory(@Param("from") LocalDateTime from,@Param("to") LocalDateTime to,@Param("category") CategoryEntity category);* default List<PurchaseEntity> getPurchaseReportByDateIntervalAndCategory(LocalDateTime from, LocalDateTime to, CategoryEntity category) {  
 return findAll(new Specification<PurchaseEntity>() {  
 @Override  
 public Predicate toPredicate(Root<PurchaseEntity> root, CriteriaQuery<?> query, CriteriaBuilder criteriaBuilder) {  
  
 List<Predicate> list=new ArrayList<>();  
  
 if (from!=null){  
 list.add(  
 criteriaBuilder.and(  
 criteriaBuilder.greaterThan(  
 root.get(PurchaseEntity\_.*date*),from)));  
 }  
 if (to!=null) {  
 list.add(  
 criteriaBuilder.and(  
 criteriaBuilder.lessThan(  
 root.get(PurchaseEntity\_.*date*),to)));  
 }  
  
 if (category!=null) {  
 list.add(  
 criteriaBuilder.and(  
 criteriaBuilder.equal(  
 root.get( PurchaseEntity\_.*product*).get(ProductEntity\_.*category*),category))  
 );  
 }  
  
 Predicate[] p=new Predicate[list.size()];  
  
 return criteriaBuilder.and(list.toArray(p));  
 }  
 });  
 }  
  
  
}

ProductRepository:

package com.example.sportyshoes.repository;  
  
import com.example.sportyshoes.entity.ProductEntity;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface ProductRepository extends JpaRepository<ProductEntity,Long> {  
}

CategoryRepository:

package com.example.sportyshoes.repository;  
  
  
import com.example.sportyshoes.entity.CategoryEntity;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface CategoryRepository extends JpaRepository<CategoryEntity,Long> {  
}

Util classes.

ProductFormData:

package com.example.sportyshoes.formdata;  
  
import com.example.sportyshoes.creationparameters.ProductCreationParameters;  
import com.example.sportyshoes.entity.CategoryEntity;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import javax.validation.constraints.NotNull;  
import javax.validation.constraints.Size;  
  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class ProductFormData {  
  
 Long id;  
  
 @NotNull  
 @Size(min = 1, max = 400)  
 private String name;  
  
 @NotNull  
 private CategoryEntity category;  
  
 public ProductCreationParameters toParameters() {  
 return new ProductCreationParameters(name, category, id);  
 }  
  
}

FormSearchUser:

package com.example.sportyshoes.entity.form;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
  
public class FormSearchUser {  
 private String name;  
}

FormFilterPurchase:

package com.example.sportyshoes.entity.form;  
  
  
import com.example.sportyshoes.entity.CategoryEntity;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
import org.springframework.format.annotation.DateTimeFormat;  
  
import java.sql.Timestamp;  
import java.time.LocalDateTime;  
  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
  
public class FormFilterPurchase {  
  
  
 @DateTimeFormat(pattern = "yyyy-MM-dd'T'HH:mm")  
 private LocalDateTime from;  
  
 @DateTimeFormat(pattern = "yyyy-MM-dd'T'HH:mm")  
 private LocalDateTime to;  
  
 private CategoryEntity category;  
  
}

FormChangePassword:

package com.example.sportyshoes.entity.form;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import javax.validation.constraints.NotBlank;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class FormChangePassword {  
  
 @NotBlank  
 String oldPassword;  
  
 @NotBlank  
 String newPassword;  
  
 @NotBlank  
 String confirmNewPassword;  
}

ProductCreationParameters:

package com.example.sportyshoes.creationparameters;  
  
import com.example.sportyshoes.entity.CategoryEntity;  
import lombok.Getter;  
import org.springframework.util.Assert;  
  
@Getter  
  
public class ProductCreationParameters {  
  
 Long id;  
  
 private String name;  
 private CategoryEntity category;  
  
 public ProductCreationParameters(String name, CategoryEntity category, Long id) {  
  
 Assert.*notNull*(name, "Name should not be null");  
 Assert.*notNull*(category, "Category should not be null");  
  
 this.id = id;  
 this.name = name;  
 this.category = category;  
 }  
}

Entities.

CategoryEntity:

package com.example.sportyshoes.entity;  
  
  
import lombok.\*;  
  
import javax.persistence.\*;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@ToString  
  
@Cacheable  
@Entity(name = "categories")  
public class CategoryEntity {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String name;  
  
 public CategoryEntity(Long id) {  
 this.id = id;  
 }  
  
}

ProductEntity:

package com.example.sportyshoes.entity;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import javax.persistence.\*;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
  
@Entity(name = "products")  
public class ProductEntity {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String name;  
  
 @OneToOne  
 @JoinColumn(nullable = false)  
 private CategoryEntity category;  
  
 public ProductEntity(String name, CategoryEntity category) {  
 this.name = name;  
 this.category = category;  
 }  
}

PurchaseEntity:

package com.example.sportyshoes.entity;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import javax.persistence.\*;  
import java.time.LocalDateTime;  
  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
  
@Entity(name = "purchases")  
public class PurchaseEntity {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @OneToOne  
 @JoinColumn(nullable = false)  
 private UserEntity user;  
  
 @OneToOne  
 @JoinColumn(nullable = false)  
 private ProductEntity product;  
  
 private Double price;  
  
 @Column(nullable = false)  
 private Long amount;  
  
 private Double summa;  
  
 private LocalDateTime date;  
}

RoleEntity:

package com.example.sportyshoes.entity;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import javax.persistence.\*;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
  
@Cacheable  
@Entity(name = "permissions")  
public class RoleEntity {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String name;  
  
}

UserEntity:

package com.example.sportyshoes.entity;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import javax.persistence.\*;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
  
@Entity(name = "users")  
public class UserEntity {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String login;  
 @Column(nullable = false)  
 private String password;  
  
  
 @OneToOne  
 @JoinColumn(nullable = false)  
 private RoleEntity permissionLevel;  
  
 public UserEntity(String login, String password) {  
 this.login = login;  
 this.password = password;  
 }  
}

Services.

UserService:

package com.example.sportyshoes.service;  
  
import com.example.sportyshoes.entity.RoleEntity;  
import com.example.sportyshoes.entity.UserEntity;  
import com.example.sportyshoes.entity.form.FormChangePassword;  
import com.example.sportyshoes.repository.UserRepository;  
import com.example.sportyshoes.security.util.MyUserDetails;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Sort;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.stereotype.Service;  
  
import java.security.Principal;  
import java.util.List;  
  
@Service  
public class UserService {  
 @Autowired  
 UserRepository userRepository;  
 @Autowired  
 BCryptPasswordEncoder bCryptPasswordEncoder;  
  
  
  
 public boolean changePassword(FormChangePassword formChangePassword, Principal principal) {  
 MyUserDetails myUserDetails =((MyUserDetails) ((UsernamePasswordAuthenticationToken) principal).getPrincipal());  
  
  
 if (bCryptPasswordEncoder.matches(formChangePassword.getOldPassword(),myUserDetails.getPassword())  
 && formChangePassword.getNewPassword().equals(formChangePassword.getConfirmNewPassword())) {  
  
 UserEntity user=userRepository.findUserEntityByLogin(myUserDetails.getUsername());  
 user.setPassword(bCryptPasswordEncoder.encode(formChangePassword.getNewPassword()));  
 userRepository.save(user);  
  
 return true;  
 }  
 return false;  
  
 }  
  
 public boolean isUserAuthorised(String login, String password){  
  
 UserEntity userEntity=userRepository.findUserEntityByLogin(login);  
  
 if (userEntity==null) return false;  
  
 if (userEntity.getPassword().equals(password))  
 return true;  
 else  
 return false;  
  
 }  
  
 public RoleEntity getUserRole(String login){  
  
 UserEntity userEntity=userRepository.findUserEntityByLogin(login);  
  
 if (userEntity==null)  
 return null;  
 else  
 return userEntity.getPermissionLevel();  
  
 }  
  
 public List<UserEntity> getUserList() {  
 return userRepository.findAll();  
 }  
  
 public List<UserEntity> findUsersWithPartOfName(String name){  
 return userRepository.findUsersWithPartOfName("%"+name+"%",Sort.*by*("login"));  
 }  
}

PurchaseService:

package com.example.sportyshoes.service;  
  
  
import com.example.sportyshoes.entity.CategoryEntity;  
import com.example.sportyshoes.entity.PurchaseEntity;  
import com.example.sportyshoes.repository.PurchaseRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Sort;  
import org.springframework.stereotype.Service;  
  
import java.time.LocalDateTime;  
import java.util.List;  
  
@Service  
public class PurchaseService {  
  
 @Autowired  
 PurchaseRepository purchaseRepository;  
  
 public List<PurchaseEntity> getPurchaseReportByDateIntervalAndCategory(LocalDateTime from, LocalDateTime to, CategoryEntity category){  
 return purchaseRepository.getPurchaseReportByDateIntervalAndCategory(from, to, category);  
 }  
  
  
}

ProductService:

package com.example.sportyshoes.service;  
  
import com.example.sportyshoes.creationparameters.ProductCreationParameters;  
import com.example.sportyshoes.entity.ProductEntity;  
import com.example.sportyshoes.formdata.ProductFormData;  
import com.example.sportyshoes.repository.ProductRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class ProductService {  
 @Autowired  
 ProductRepository productRepository;  
  
 public List<ProductEntity> getProductList(){  
 return productRepository.findAll();  
 }  
  
 public ProductEntity getProductById(Long id){  
 return productRepository.findById(id).get();  
 }  
  
 public void addProduct(ProductCreationParameters creationParameters){  
 productRepository.save(new ProductEntity(creationParameters.getName(), creationParameters.getCategory()));  
 }  
  
 public void deleteProductById(Long id){  
 productRepository.deleteById(id);  
 }  
  
 public void updateProduct(ProductCreationParameters creationParameters){  
 productRepository.save(new ProductEntity(  
 creationParameters.getId(),  
 creationParameters.getName(),  
 creationParameters.getCategory()));  
 }  
  
 public Object getProductFormDataById(Long id) {  
 ProductEntity productEntity=productRepository.findById(id).get();  
 return new ProductFormData(productEntity.getId(),productEntity.getName(),productEntity.getCategory());  
 }  
}

CategoryService:

package com.example.sportyshoes.service;  
  
import com.example.sportyshoes.entity.CategoryEntity;  
import com.example.sportyshoes.repository.CategoryRepository;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CategoryService {  
  
 @Autowired  
 CategoryRepository categoryRepository;  
  
 public List<CategoryEntity> getCategoryList(){  
 return categoryRepository.findAll();  
 }  
  
  
}

Controllers.

UserController:

package com.example.sportyshoes.controller;  
  
import java.security.Principal;  
import com.example.sportyshoes.entity.UserEntity;  
import com.example.sportyshoes.entity.form.FormChangePassword;  
import com.example.sportyshoes.entity.form.FormSearchUser;  
import com.example.sportyshoes.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import java.util.List;  
  
@RequestMapping("/manageUsers")  
@Controller  
public class UserController {  
  
 @Autowired  
 UserService userService;  
  
  
 @GetMapping("/main")  
 public String getUsersPage(Model model){  
 model.addAttribute("formSearchUser",new FormSearchUser());  
 return "browseUsers";  
 }  
  
 @GetMapping("/changePassword")  
 public String getUsersChangePasswordPage(Model model){  
 model.addAttribute("formChangePassword",new FormChangePassword());  
 return "changePassword";  
 }  
  
 @PostMapping("/changePassword")  
 public String getSuccessChangePasswordPage(@ModelAttribute FormChangePassword formChangePassword, Principal principal){  
 if (userService.changePassword(formChangePassword,principal))  
 return "passwordChangedSuccessfully";  
 else return "error";  
 }  
  
  
 @PostMapping("/search")  
 public String getUsersPage(@ModelAttribute FormSearchUser formSearchUser, Model model){  
 List<UserEntity> foundUsers=userService.findUsersWithPartOfName(formSearchUser.getName());  
 model.addAttribute("foundUsers",foundUsers);  
 return "foundUsers";  
 }  
  
 @ModelAttribute("userList")  
 public List<UserEntity> getUserList(){  
 return userService.getUserList();  
 }  
  
  
}

PurchaseController:

package com.example.sportyshoes.controller;  
  
  
import com.example.sportyshoes.entity.CategoryEntity;  
import com.example.sportyshoes.entity.form.FormFilterPurchase;  
import com.example.sportyshoes.entity.form.FormSearchUser;  
import com.example.sportyshoes.service.CategoryService;  
import com.example.sportyshoes.service.PurchaseService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import java.util.List;  
  
@Controller  
@RequestMapping("/managePurchases")  
public class PurchaseController {  
  
 @Autowired  
 PurchaseService purchaseService;  
 @Autowired  
 CategoryService categoryService;  
  
  
 @GetMapping("/makeReport")  
 public String getPurchaseSelectFormPage(Model model){  
 model.addAttribute("formFilterPurchase",new FormFilterPurchase());  
 return "purchaseSelectForm";  
 }  
  
  
 @PostMapping("/show")  
 public String getPurchaseR(@ModelAttribute("formFilterPurchase") FormFilterPurchase formFilterPurchase, Model model){  
 model.addAttribute("formFilterPurchase",formFilterPurchase);  
 model.addAttribute("purchaseList",purchaseService.getPurchaseReportByDateIntervalAndCategory(formFilterPurchase.getFrom(),formFilterPurchase.getTo(),formFilterPurchase.getCategory()));  
 return "purchaseReport";  
 }  
  
 @ModelAttribute("categoryList")  
 public List<CategoryEntity> getCategoryList(){  
 return categoryService.getCategoryList();  
 }  
  
  
}

ProductController:

package com.example.sportyshoes.controller;  
  
import com.example.sportyshoes.entity.CategoryEntity;  
import com.example.sportyshoes.entity.ProductEntity;  
import com.example.sportyshoes.formdata.ProductFormData;  
import com.example.sportyshoes.service.CategoryService;  
import com.example.sportyshoes.service.ProductService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.validation.BindingResult;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.validation.Valid;  
import java.util.List;  
  
  
@RequestMapping("/manageProducts")  
@Controller  
public class ProductController {  
  
 @Autowired  
 CategoryService categoryService;  
 @Autowired  
 ProductService productService;  
  
  
  
 @GetMapping("/main")  
 public String getProductsPage(Model model){  
 model.addAttribute("productFormData",new ProductFormData());  
 return "manageProducts";  
 }  
  
 @GetMapping("/actions{id}")  
 public String getProductActionsPage(@PathVariable("id") Long id, Model model){  
  
 model.addAttribute("productFormData",productService.getProductFormDataById(id));  
 return "productActions";  
 }  
  
 @PostMapping("/delete{id}")  
 public String deleteProduct(@PathVariable("id") Long id){  
 productService.deleteProductById(id);  
 return "redirect:/manageProducts/main";  
 }  
  
 @PostMapping("/update")  
 public String updateProduct(@Valid @ModelAttribute("productFormData") ProductFormData productFormData, BindingResult bindingResult, Model model){  
  
 if (bindingResult.hasErrors()) {  
 return "manageProducts/main";  
 }  
 productService.updateProduct(productFormData.toParameters());  
  
 return "redirect:/manageProducts/main";  
 }  
  
 @PostMapping({"/addProduct"})  
 public String getLoginPage(@Valid @ModelAttribute("productFormData") ProductFormData productFormData, BindingResult bindingResult, Model model){  
  
  
 if (bindingResult.hasErrors()) {  
 return "manageProducts/main";  
 }  
 productService.addProduct(productFormData.toParameters());  
  
 return "redirect:/manageProducts/main";  
 }  
  
 @ModelAttribute("categoryList")  
 public List<CategoryEntity> getCategoryList(){  
 return categoryService.getCategoryList();  
 }  
  
 @ModelAttribute("productList")  
 public List<ProductEntity> getProductList(){  
 return productService.getProductList();  
 }  
  
  
}

LoginController:

package com.example.sportyshoes.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
  
  
@Controller  
public class LoginController {  
  
 @GetMapping("/")  
 public String getLoginPage(){  
 return "index";  
 }  
  
 @GetMapping("/userDashboard")  
 public String getUserDashboardPage(){  
 return "userDashboard";  
 }  
  
}

Spring security.

WebSecurityConfig:

package com.example.sportyshoes.config;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
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;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
  
@Configuration  
@EnableWebSecurity  
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  
  
 @Autowired  
 public UserDetailsService userDetailsService;  
  
 @Bean  
 public BCryptPasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public DaoAuthenticationProvider authenticationProvider() {  
 DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();  
 authProvider.setUserDetailsService(userDetailsService);  
 authProvider.setPasswordEncoder(passwordEncoder());  
 return authProvider;  
 }  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.authenticationProvider(authenticationProvider());  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests()  
 .antMatchers("/").hasAnyAuthority("User", "Administrator")  
 .antMatchers("/manageProducts/\*\*").hasAuthority("Administrator")  
 .antMatchers("/managePurchases/\*\*").hasAuthority("Administrator")  
 .antMatchers("/manageUsers/\*\*").hasAuthority("Administrator")  
 .anyRequest().authenticated()  
 .and()  
 .formLogin().permitAll()  
 .and()  
 .logout().permitAll()  
 .and()  
 .exceptionHandling().accessDeniedPage("/403")  
 ;  
 }  
}

MyUserDetails:

package com.example.sportyshoes.security.util;  
  
import com.example.sportyshoes.entity.UserEntity;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.List;  
  
public class MyUserDetails implements UserDetails {  
  
 private UserEntity user;  
  
 public MyUserDetails(UserEntity user) {  
 this.user = user;  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
 List<SimpleGrantedAuthority> authorities = new ArrayList<>();  
 authorities.add(new SimpleGrantedAuthority(user.getPermissionLevel().getName()));  
 return authorities;  
 }  
  
 @Override  
 public String getPassword() {  
 return user.getPassword();  
 }  
  
 @Override  
 public String getUsername() {  
 return user.getLogin();  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
 return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
 return true;  
 }  
  
}

UserDetailsServiceImpl:

package com.example.sportyshoes.security.util;  
  
import com.example.sportyshoes.entity.UserEntity;  
import com.example.sportyshoes.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
  
@Service  
public class UserDetailsServiceImpl implements UserDetailsService {  
  
 @Autowired  
 private UserRepository userRepository;  
  
 @Override  
 public UserDetails loadUserByUsername(String username)  
 throws UsernameNotFoundException {  
 UserEntity user = userRepository.findUserEntityByLogin(username);  
  
 if (user == null) {  
 throw new UsernameNotFoundException("Could not find user");  
 }  
 return new MyUserDetails(user);  
 }  
  
}