package com.medicare.entities;

import java.util.Collection;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import org.springframework.security.core.GrantedAuthority;

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

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

@Table(name = "users")

public class User implements UserDetails{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long userId;

@NotBlank(message = "username cannot be null.")

private String username;

@NotBlank(message = "password cannot be null.")

@Size(min = 6, message = "enter minimum six character password")

private String password;

@NotBlank(message = "first name cannot be null.")

private String firstName;

@NotBlank(message = "last name cannot be null")

private String lastName;

@NotBlank(message = "contact number cannot be null")

private String contactNumber;

private boolean enabled = true;

@OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, mappedBy = "user")

@JsonIgnore

private Set<UserRole> userRoles = new HashSet<>();

public User() {

super();

}

public User(Long userId, String username, String password, String firstName, String lastName, String contactNumber,

boolean enabled, Set<UserRole> userRoles) {

super();

this.userId = userId;

this.username = username;

this.password = password;

this.firstName = firstName;

this.lastName = lastName;

this.contactNumber = contactNumber;

this.enabled = enabled;

this.userRoles = userRoles;

}

public Long getUserId() {

return userId;

}

public void setUserId(Long userId) {

this.userId = userId;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getContactNumber() {

return contactNumber;

}

public void setContactNumber(String contactNumber) {

this.contactNumber = contactNumber;

}

public boolean isEnabled() {

return enabled;

}

public void setEnabled(boolean enabled) {

this.enabled = enabled;

}

public Set<UserRole> getUserRoles() {

return userRoles;

}

public void setUserRoles(Set<UserRole> userRoles) {

this.userRoles = userRoles;

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

Set<Authority> authority = new HashSet<>();

this.userRoles.forEach(userRole -> {

authority.add(new Authority(userRole.getRole().getRoleName()));

});

return authority;

}

@Override

public boolean isAccountNonExpired() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isAccountNonLocked() {

// TODO Auto-generated method stub

return true;

}

@Override

public boolean isCredentialsNonExpired() {

// TODO Auto-generated method stub

return true;

}

}

package com.medicare.entities;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import javax.validation.constraints.NotBlank;

@Entity

@Table(name = "roles")

public class Role {

@Id

private Long roleId;

@NotBlank(message = "role name cannot be null.")

private String roleName;

@OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "role")

private Set<UserRole> userRoles = new HashSet<>();

public Role() {

super();

// TODO Auto-generated constructor stub

}

public Role(Long roleId, String roleName, Set<UserRole> userRoles) {

super();

this.roleId = roleId;

this.roleName = roleName;

this.userRoles = userRoles;

}

public Long getRoleId() {

return roleId;

}

public void setRoleId(Long roleId) {

this.roleId = roleId;

}

public String getRoleName() {

return roleName;

}

public void setRoleName(String roleName) {

this.roleName = roleName;

}

public Set<UserRole> getUserRoles() {

return userRoles;

}

public void setUserRoles(Set<UserRole> userRoles) {

this.userRoles = userRoles;

}

}

package com.medicare.entities;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

@Entity

public class UserOrder {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long oid;

private String username;

private String firstName;

private String lastName;

private String address;

private String district;

private int pinCode;

private String state;

private String contact;

private String date;

private String status;

private Double paidAmount;

private String paymentMode;

@ManyToMany(cascade = CascadeType.ALL)

private Set<ProductQuantity> products = new HashSet<>();

public UserOrder() {

}

public UserOrder(Long oid, String username, String firstName, String lastName, String address, String district,

int pinCode, String state, String contact, String date, String status, Double paidAmount, String paymentMode,

Set<ProductQuantity> products) {

super();

this.oid = oid;

this.username = username;

this.firstName = firstName;

this.lastName = lastName;

this.address = address;

this.district = district;

this.pinCode = pinCode;

this.state = state;

this.contact = contact;

this.date = date;

this.status = status;

this.paidAmount = paidAmount;

this.paymentMode = paymentMode;

this.products = products;

}

public Long getOid() {

return oid;

}

public void setOid(Long oid) {

this.oid = oid;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getDistrict() {

return district;

}

public void setDistrict(String district) {

this.district = district;

}

public int getPinCode() {

return pinCode;

}

public void setPinCode(int pinCode) {

this.pinCode = pinCode;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public String getContact() {

return contact;

}

public void setContact(String contact) {

this.contact = contact;

}

public Set<ProductQuantity> getProducts() {

return products;

}

public void setProducts(Set<ProductQuantity> products) {

this.products = products;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public Double getPaidAmount() {

return paidAmount;

}

public void setPaidAmount(Double paidAmount) {

this.paidAmount = paidAmount;

}

public String getPaymentMode() {

return paymentMode;

}

public void setPaymentMode(String paymentMode) {

this.paymentMode = paymentMode;

}

}

package com.medicare.services;

import java.util.Set;

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.stereotype.Service;

import com.medicare.entities.User;

import com.medicare.entities.UserRole;

import com.medicare.repo.RoleRepo;

import com.medicare.repo.UserRepo;

@Service

public class UserService {

@Autowired

private UserRepo userRepo;

@Autowired

private RoleRepo roleRepo;

@Autowired

private BCryptPasswordEncoder passwordEncoder;

//register a new user

public User createUser(User user, Set<UserRole> userRole){

User newUser = this.userRepo.findByUsername(user.getUsername());

//if user exists or not

try {

if(newUser!=null) {

throw new Exception("Username already exists!");

}else {

//create new user

//saving roles

for(UserRole uR : userRole) {

this.roleRepo.save(uR.getRole());

}

//setting userRole in user

user.getUserRoles().addAll(userRole);

//encoding password

user.setPassword(this.passwordEncoder.encode(user.getPassword()));

newUser = this.userRepo.save(user);

}

} catch (Exception e) {

System.out.println("User is already created!");

System.out.println(e);

}

return newUser;

}

public User getByUsername(String username) {

User user = this.userRepo.findByUsername(username);

return user;

}

}

package com.medicare.entities;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Lob;

import javax.persistence.OneToOne;

import com.fasterxml.jackson.annotation.JsonBackReference;

@Entity

public class ProductImage {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long imgId;

private String name;

private String type;

@Lob

@Column(name = "imagedata")

private byte[] imageData;

@OneToOne(mappedBy = "productImage")

@JsonBackReference

private Product product;

public ProductImage() {

super();

}

public ProductImage(Long imgId, String name, String type, byte[] imageData, Product product) {

super();

this.imgId = imgId;

this.name = name;

this.type = type;

this.imageData = imageData;

this.product = product;

}

public Long getImgId() {

return imgId;

}

public void setImgId(Long imgId) {

this.imgId = imgId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public byte[] getImageData() {

return imageData;

}

public void setImageData(byte[] imageData) {

this.imageData = imageData;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

}

package com.medicare.entities;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotNull;

import com.fasterxml.jackson.annotation.JsonManagedReference;

@Entity

@Table(name="products")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long pid;

@NotBlank(message = "name cannot be blank")

private String name;

@NotBlank(message = "brand cannot be blank")

private String brand;

@NotBlank(message = "category cannot be blank")

private String category;

@NotBlank(message = "description cannot be blank")

private String description;

@NotBlank(message = "salt cannot be blank")

private String salt;

@NotNull(message = "available cannot be null")

private int totalAvailable;

@NotNull(message = "price cannot be null")

private Double price;

@NotNull(message = "isAvailable cannot be null")

private boolean isAvailable;

@OneToOne(cascade = CascadeType.ALL)

@JsonManagedReference

private ProductImage productImage;

public Product() {

super();

}

public Product(Long pid, String name, String brand, String category,String description, String salt, int totalAvailable, Double price,

boolean isAvailable, ProductImage productImage) {

super();

this.pid = pid;

this.name = name;

this.brand = brand;

this.category = category;

this.description = description;

this.salt = salt;

this.totalAvailable = totalAvailable;

this.price = price;

this.isAvailable = isAvailable;

this.productImage = productImage;

}

public Long getPid() {

return pid;

}

public void setPid(Long pid) {

this.pid = pid;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = brand;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String getSalt() {

return salt;

}

public void setSalt(String salt) {

this.salt = salt;

}

public int getTotalAvailable() {

return totalAvailable;

}

public void setTotalAvailable(int totalAvailable) {

this.totalAvailable = totalAvailable;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

public boolean isAvailable() {

return isAvailable;

}

public void setAvailable(boolean isAvailable) {

this.isAvailable = isAvailable;

}

public ProductImage getProductImage() {

return productImage;

}

public void setProductImage(ProductImage productImage) {

this.productImage = productImage;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

}

**package** com.medicare.entities;

**public** **class** JwtResponse {

String token;

**public** JwtResponse() {

}

**public** JwtResponse(String token) {

**super**();

**this**.token = token;

}

**public** String getToken() {

**return** token;

}

**public** **void** setToken(String token) {

**this**.token = token;

}

}

**package** com.medicare.entities;

**public** **class** JwtRequest {

String username;

String password;

**public** JwtRequest() {

}

**public** JwtRequest(String username, String password) {

**super**();

**this**.username = username;

**this**.password = password;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

package com.medicare.entities;

import java.util.HashSet;

import java.util.Set;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotEmpty;

import javax.validation.constraints.NotNull;

public class CartOrder {

@NotBlank

private String username;

@NotBlank

private String firstName;

@NotBlank

private String lastName;

@NotBlank

private String address;

@NotBlank

private String district;

@NotNull

private int pinCode;

@NotBlank

private String state;

@NotBlank

private String contact;

@NotNull

private Double paidAmount;

@NotBlank

private String paymentMode;

@NotEmpty

private Set<CartItem> cartItem = new HashSet<>();

public CartOrder() {

}

public CartOrder(String username, String firstName, String lastName, String address, String district, int pinCode,

String state, String contact, Double paidAmount, String paymentMode, Set<CartItem> cartItem) {

super();

this.username = username;

this.firstName = firstName;

this.lastName = lastName;

this.address = address;

this.district = district;

this.pinCode = pinCode;

this.state = state;

this.contact = contact;

this.paidAmount = paidAmount;

this.paymentMode = paymentMode;

this.cartItem = cartItem;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getDistrict() {

return district;

}

public void setDistrict(String district) {

this.district = district;

}

public int getPinCode() {

return pinCode;

}

public void setPinCode(int pinCode) {

this.pinCode = pinCode;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public String getContact() {

return contact;

}

public void setContact(String contact) {

this.contact = contact;

}

public Set<CartItem> getCartItem() {

return cartItem;

}

public void setCartItem(Set<CartItem> cartItem) {

this.cartItem = cartItem;

}

public Double getPaidAmount() {

return paidAmount;

}

public void setPaidAmount(Double paidAmount) {

this.paidAmount = paidAmount;

}

public String getPaymentMode() {

return paymentMode;

}

public void setPaymentMode(String paymentMode) {

this.paymentMode = paymentMode;

}

}

**package** com.medicare.entities;

**public** **class** CartItem {

**private** Long pid;

**private** **int** quantity;

**public** CartItem() {

}

**public** CartItem(Long pid, **int** quantity) {

**super**();

**this**.pid = pid;

**this**.quantity = quantity;

}

**public** Long getPid() {

**return** pid;

}

**public** **void** setPid(Long pid) {

**this**.pid = pid;

}

**public** **int** getQuantity() {

**return** quantity;

}

**public** **void** setQuantity(**int** quantity) {

**this**.quantity = quantity;

}

}

**package** com.medicare.entities;

**import** org.springframework.security.core.GrantedAuthority;

**public** **class** Authority **implements** GrantedAuthority{

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** String authority;

**public** Authority(String authority) {

**super**();

**this**.authority = authority;

}

@Override

**public** String getAuthority() {

**return** **this**.authority;

}

}

package com.medicare.services;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.medicare.entities.ProductQuantity;

import com.medicare.entities.UserOrder;

import com.medicare.repo.OrderRepo;

import com.medicare.repo.ProductQuantityRepo;

@Service

public class UserOrderService {

@Autowired

private OrderRepo orderRepo;

@Autowired

private ProductQuantityRepo productQuantityRepo;

public UserOrder saveOrder(UserOrder userOrder) {

UserOrder orderSaved = this.orderRepo.save(userOrder);

return orderSaved;

}

public void saveProductQuantity(ProductQuantity productQuantity) {

this.productQuantityRepo.save(productQuantity);

}

public List<UserOrder> getAll(){

return this.orderRepo.findAll();

}

public List<UserOrder> getUserOrders(String username){

List<UserOrder> orders = this.orderRepo.findByUsername(username);

return orders;

}

public UserOrder getOrderById(Long oid) {

UserOrder order = this.orderRepo.findById(oid).get();

return order;

}

public void deleteOrder(Long oid) {

this.orderRepo.deleteById(oid);

}

}

package com.medicare.services;

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;

import com.medicare.entities.User;

import com.medicare.repo.UserRepo;

@Service

public class UserDetailService implements UserDetailsService{

@Autowired

private UserRepo userRepo;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user = this.userRepo.findByUsername(username);

if(user == null) {

System.out.println("User not found!");

throw new UsernameNotFoundException("User does not exist!");

}

return user;

}

}

package com.medicare.services;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.medicare.entities.Product;

import com.medicare.repo.ProductRepo;

@Service

public class ProductService {

@Autowired

private ProductRepo productRepo;

// add product

public Product addProduct(Product product) {

return this.productRepo.save(product);

}

//find product by id

public Product findProduct(Long pid) {

return this.productRepo.findById(pid).get();

}

//find all products

public List<Product> findAllProducts(){

return this.productRepo.findAll();

}

//find product by name or salt

public List<Product> findByNameOrSalt(String name, String salt){

List<Product> products = this.productRepo.findByNameContainingIgnoreCaseOrSaltContainingIgnoreCase(name, salt);

return products;

}

//find product by category

public List<Product> findProductByCategory(String category){

List<Product> products = this.productRepo.findByCategory(category);

return products;

}

//delete product by id

public void deleteProductById(Long pid) {

this.productRepo.deleteById(pid);

}

//find available products

public List<Product> findTrueProduct(String name){

List<Product> products = this.productRepo.findByNameAndIsAvailableTrue(name);

return products;

}

}

package com.medicare.repo;

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

import org.springframework.stereotype.Repository;

import com.medicare.entities.User;

@Repository

public interface UserRepo extends JpaRepository<User, Long>{

public User findByUsername(String username);

}

package com.medicare.repo;

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

import org.springframework.stereotype.Repository;

import com.medicare.entities.Role;

@Repository

public interface RoleRepo extends JpaRepository<Role, Long>{

}

package com.medicare.repo;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.medicare.entities.Product;

@Repository

public interface ProductRepo extends JpaRepository<Product, Long>{

public List<Product> findByNameContainingIgnoreCaseOrSaltContainingIgnoreCase(String name, String salt);

public List<Product> findByCategory(String category);

public List<Product> findByNameAndIsAvailableTrue(String name);

}

package com.medicare.repo;

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

import org.springframework.stereotype.Repository;

import com.medicare.entities.ProductQuantity;

@Repository

public interface ProductQuantityRepo extends JpaRepository<ProductQuantity, Long>{

}

package com.medicare.entities;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

@Entity

public class ProductQuantity {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long pqid;

@OneToOne

private Product product;

private int quantity;

public ProductQuantity() {

}

public ProductQuantity(Long pqid, Product product, int quantity) {

super();

this.pqid = pqid;

this.product = product;

this.quantity = quantity;

}

public Long getPqid() {

return pqid;

}

public void setPqid(Long pqid) {

this.pqid = pqid;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

}

package com.medicare.repo;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.medicare.entities.UserOrder;

@Repository

public interface OrderRepo extends JpaRepository<UserOrder, Long>{

public List<UserOrder> findByUsername(String username);

}

package com.medicare.entities;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToOne;

@Entity

public class UserRole {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long userRoleId;

@ManyToOne(fetch = FetchType.EAGER)

private User user;

@ManyToOne

private Role role;

public UserRole() {

super();

}

public UserRole(Long userRoleId, User user, Role role) {

super();

this.userRoleId = userRoleId;

this.user = user;

this.role = role;

}

public Long getUserRoleId() {

return userRoleId;

}

public void setUserRoleId(Long userRoleId) {

this.userRoleId = userRoleId;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Role getRole() {

return role;

}

public void setRole(Role role) {

this.role = role;

}

}

package com.medicare.controller;

import java.net.URI;

import java.util.HashSet;

import java.util.Set;

import javax.annotation.PostConstruct;

import javax.validation.Valid;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.medicare.entities.Role;

import com.medicare.entities.User;

import com.medicare.entities.UserRole;

import com.medicare.services.UserService;

@RestController

@CrossOrigin(origins = "\*")

public class UserController {

@Autowired

private UserService userService;

//init admin user

@PostConstruct

public void createAdmin(){

User admin = new User();

admin.setUsername("admin@medicare.com");

admin.setPassword("admin12345");

admin.setFirstName("Twarit");

admin.setLastName("Soni");

admin.setContactNumber("6265989908");

Role role = new Role();

role.setRoleId(101L);

role.setRoleName("ADMIN");

UserRole ur = new UserRole();

ur.setUser(admin);

ur.setRole(role);

Set<UserRole> userRole = new HashSet<>();

userRole.add(ur);

User adminCreated = this.userService.createUser(admin, userRole);

System.out.println("Admin username: "+adminCreated.getUsername());

}

//create new user

@PostMapping("/user/signup")

public ResponseEntity<?> createNewUser(@Valid @RequestBody User user){

Role role = new Role();

role.setRoleId(102L);

role.setRoleName("USER");

UserRole ur = new UserRole();

ur.setUser(user);

ur.setRole(role);

Set<UserRole> userRole = new HashSet<>();

userRole.add(ur);

if(this.userService.getByUsername(user.getUsername())!=null) {

System.out.println("Username already exists!");

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).build();

}else {

User newUser = this.userService.createUser(user, userRole);

URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(newUser.getUserId()).toUri();

return ResponseEntity.created(location).build();

}

}

}

package com.medicare.controller;

import java.io.IOException;

import java.util.List;

import javax.validation.Valid;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.access.prepost.PreAuthorize;

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

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.RequestParam;

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

import org.springframework.web.multipart.MultipartFile;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.JsonMappingException;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.medicare.config.ImageUtil;

import com.medicare.entities.Product;

import com.medicare.entities.ProductImage;

import com.medicare.services.ProductService;

@RestController

@CrossOrigin(origins = "\*")

public class ProductController {

@Autowired

private ProductService productService;

@Autowired

private ObjectMapper objectMapper;

//add new product

@PreAuthorize("hasAuthority('ADMIN')")

@PostMapping("/add/product")

public ResponseEntity<?> addNewProduct(@RequestParam("product") String product,

@RequestParam("image") MultipartFile file) throws IOException{

ProductImage img = new ProductImage();

img.setName(file.getOriginalFilename());

img.setType(file.getContentType());

img.setImageData(ImageUtil.compressImage(file.getBytes()));

Product p = null;

try {

p = objectMapper.readValue(product,Product.class);

p.setProductImage(img);

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (JsonProcessingException e) {

e.printStackTrace();

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body("Invalid Request");

}

Product saveProduct = this.productService.addProduct(p);

return ResponseEntity.ok(saveProduct);

}

//update existing product

@PreAuthorize("hasAuthority('ADMIN')")

@PutMapping("/update/product/{id}")

public ResponseEntity<?> updateProduct(@PathVariable("id") Long id,@Valid @RequestBody Product product){

Product updateProduct = this.productService.findProduct(id);

updateProduct.setName(product.getName());

updateProduct.setBrand(product.getBrand());

updateProduct.setCategory(product.getCategory());

updateProduct.setDescription(product.getDescription());

updateProduct.setSalt(product.getSalt());

updateProduct.setTotalAvailable(product.getTotalAvailable());

updateProduct.setPrice(product.getPrice());

this.productService.addProduct(updateProduct);

return ResponseEntity.status(HttpStatus.CREATED).build();

}

//find product by id

@GetMapping("get-product/{id}")

public ResponseEntity<?> getProductById(@PathVariable("id") Long id){

Product product = this.productService.findProduct(id);

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

return ResponseEntity.ok(product);

}

//find all products

@GetMapping("/get/all-products")

public ResponseEntity<?> getAllProducts(){

List<Product> allProducts = this.productService.findAllProducts();

allProducts.forEach(product -> {

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

});

if(allProducts.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(allProducts);

}

}

@GetMapping(value = {"/get/products/{name}"})

public ResponseEntity<?> getProductByName(@PathVariable("name") String name,@PathVariable("name") String salt){

List<Product> products = this.productService.findByNameOrSalt(name, salt);

products.forEach(product -> {

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

});

if(products.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(products);

}

}

@GetMapping("/get/products-by-category/{category}")

public ResponseEntity<?> getProductsByCategory(@PathVariable("category") String category){

List<Product> products = this.productService.findProductByCategory(category);

products.forEach(product -> {

ProductImage img = new ProductImage();

img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

img.setImgId(product.getProductImage().getImgId());

img.setName(product.getProductImage().getName());

img.setType(product.getProductImage().getType());

product.setProductImage(img);

});

if(products.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(products);

}

}

@PreAuthorize("hasAuthority('ADMIN')")

@DeleteMapping("/delete/product/{id}")

public ResponseEntity<?> deleteProduct(@PathVariable("id") Long id){

this.productService.deleteProductById(id);

return ResponseEntity.status(HttpStatus.OK).build();

}

@PreAuthorize("hasAuthority('ADMIN')")

@PutMapping("/set-availability/product/{id}")

public ResponseEntity<?> setAvailability(@PathVariable("id") Long id, @RequestBody Product product){

Product updateProduct = this.productService.findProduct(id);

updateProduct.setAvailable(product.isAvailable());

this.productService.addProduct(updateProduct);

return ResponseEntity.status(HttpStatus.CREATED).build();

}

@GetMapping("/get/{name}")

public ResponseEntity<?> getAvailable(@PathVariable("name") String name){

List<Product> products = this.productService.findTrueProduct(name);

if(products.isEmpty()) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}else {

return ResponseEntity.ok(products);

}

}

}

package com.medicare.controller;

import java.security.Principal;

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.BadCredentialsException;

import org.springframework.security.authentication.DisabledException;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

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

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

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

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

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

import com.medicare.config.JwtUtil;

import com.medicare.entities.JwtRequest;

import com.medicare.entities.JwtResponse;

import com.medicare.entities.User;

import com.medicare.services.UserDetailService;

@RestController

@CrossOrigin(origins = "\*")

public class JwtController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private UserDetailService userDetailService;

@Autowired

private JwtUtil jwtUtil;

//generate token

@PostMapping("/generate-token")

public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest) throws Exception{

try {

authenticate(jwtRequest.getUsername(), jwtRequest.getPassword());

}catch(UsernameNotFoundException e) {

e.printStackTrace();

throw new Exception("User does not exist or invalid credentials!");

}

// validated

UserDetails userDetails = this.userDetailService.loadUserByUsername(jwtRequest.getUsername());

String token = this.jwtUtil.generateToken(userDetails);

return ResponseEntity.ok(new JwtResponse(token));

}

private void authenticate(String username, String password) throws Exception {

try {

this.authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username, password));

} catch (BadCredentialsException e) {

throw new Exception("Invalid Credentials! "+e.getMessage());

}catch(DisabledException e) {

throw new Exception("User Disabled! "+e.getMessage());

}

}

@GetMapping("/current-user")

public User getCurrentUser(Principal principal) {

return (User)this.userDetailService.loadUserByUsername(principal.getName());

}

}

package com.medicare.config;

import java.util.HashMap;

import java.util.Map;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.FieldError;

import org.springframework.web.bind.MethodArgumentNotValidException;

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

import org.springframework.web.context.request.WebRequest;

import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@ControllerAdvice

public class ValidationHandler extends ResponseEntityExceptionHandler{

@Override

protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,

HttpHeaders headers, HttpStatus status, WebRequest request) {

Map<String, String> errors = new HashMap<>();

ex.getBindingResult().getAllErrors().forEach((error) ->{

String fieldName = ((FieldError) error).getField();

String message = error.getDefaultMessage();

errors.put(fieldName, message);

});

return new ResponseEntity<Object>(errors, HttpStatus.BAD\_REQUEST);

}

}

package com.medicare.config;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

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.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.medicare.services.UserDetailService;

@SuppressWarnings("deprecation")

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter{

@Autowired

private UserDetailService userDetailService;

@Autowired

private AuthEntryPoint authEntryPoint;

@Autowired

private JwtAuthFilter jwtAuthFilter;

@Bean

public BCryptPasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Override

@Bean

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(this.userDetailService).passwordEncoder(passwordEncoder());

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf()

.disable()

.cors()

.disable()

.authorizeRequests()

.antMatchers("/generate-token").permitAll()

.antMatchers("/user/signup","/get/all-products","/get/products/\*\*","/get/products-by-category/\*\*","/get-product/\*\*").permitAll()

.antMatchers(HttpMethod.OPTIONS).permitAll()

.anyRequest().authenticated()

.and().exceptionHandling().authenticationEntryPoint(authEntryPoint)

.and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

http.addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

}

}

package com.medicare.config;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.stereotype.Component;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

@Component

public class JwtUtil {

private String SECRET\_KEY = "medicare";

public String extractUsername(String token) {

return extractClaim(token, Claims::getSubject);

}

public Date extractExpiration(String token) {

return extractClaim(token, Claims::getExpiration);

}

public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

final Claims claims = extractAllClaims(token);

return claimsResolver.apply(claims);

}

private Claims extractAllClaims(String token) {

return Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody();

}

private boolean isTokenExpired(String token) {

return extractExpiration(token).before(new Date());

}

public String generateToken(UserDetails userDetails) {

Map<String, Object> claims = new HashMap<>();

return createToken(claims, userDetails.getUsername());

}

private String createToken(Map<String, Object> claims, String subject) {

return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 10))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY).compact();

}

public boolean validateToken(String token, UserDetails userDetails) {

final String username = extractUsername(token);

return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));

}

}

package com.medicare.config;

import java.io.IOException;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

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

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.medicare.services.UserDetailService;

import io.jsonwebtoken.ExpiredJwtException;

@Component

public class JwtAuthFilter extends OncePerRequestFilter{

@Autowired

private UserDetailService userDetailService;

@Autowired

private JwtUtil jwtUtil;

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)

throws ServletException, IOException {

final String requestTokenHeader = request.getHeader("Authorization");

String username = null;

String jwtToken = null;

if(requestTokenHeader!=null && requestTokenHeader.startsWith("Bearer ")) {

jwtToken = requestTokenHeader.substring(7);

try {

username = this.jwtUtil.extractUsername(jwtToken);

}catch(ExpiredJwtException e) {

e.printStackTrace();

System.out.println("Token Expired!");

}catch(Exception e) {

e.printStackTrace();

}

}else {

System.out.println("Invalid token! Not start's from Bearer string!");

}

// validation successful

if(username!=null && SecurityContextHolder.getContext().getAuthentication()==null) {

final UserDetails userDetails = this.userDetailService.loadUserByUsername(username);

if(this.jwtUtil.validateToken(jwtToken, userDetails)) {

//token is valid

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(userDetails,null,userDetails.getAuthorities());

usernamePasswordAuthenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

}else {

System.out.println("Token is invalid! Please generate a new token!");

}

}else {

System.out.println("Invalid username!");

}

filterChain.doFilter(request, response);

}

}

package com.medicare.config;

import java.io.ByteArrayOutputStream;

import java.util.zip.Deflater;

import java.util.zip.Inflater;

public class ImageUtil {

public static byte[] compressImage(byte[] data) {

Deflater deflater = new Deflater();

deflater.setLevel(Deflater.BEST\_COMPRESSION);

deflater.setInput(data);

deflater.finish();

ByteArrayOutputStream outputStream = new ByteArrayOutputStream(data.length);

byte[] tmp = new byte[4\*1024];

while (!deflater.finished()) {

int size = deflater.deflate(tmp);

outputStream.write(tmp, 0, size);

}

try {

outputStream.close();

} catch (Exception ignored) {

}

return outputStream.toByteArray();

}

public static byte[] decompressImage(byte[] data) {

Inflater inflater = new Inflater();

inflater.setInput(data);

ByteArrayOutputStream outputStream = new ByteArrayOutputStream(data.length);

byte[] tmp = new byte[4\*1024];

try {

while (!inflater.finished()) {

int count = inflater.inflate(tmp);

outputStream.write(tmp, 0, count);

}

outputStream.close();

} catch (Exception ignored) {

}

return outputStream.toByteArray();

}

}

package com.medicare.config;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.web.AuthenticationEntryPoint;

import org.springframework.stereotype.Component;

@Component

public class AuthEntryPoint implements AuthenticationEntryPoint{

@Override

public void commence(HttpServletRequest request, HttpServletResponse response,

AuthenticationException authException) throws IOException, ServletException {

response.sendError(HttpServletResponse.SC\_UNAUTHORIZED, "Unauthorized");

}

}

package com.medicare;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class MedicareBackendApplication {

public static void main(String[] args) {

SpringApplication.run(MedicareBackendApplication.class, args);

}

}

spring.datasource.name=medicare

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

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

spring.datasource.username=root

spring.datasource.password=123456789

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true