**Back end API source code**

**package com.foodBox;**

**import java.util.Collections;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**import org.springframework.boot.builder.SpringApplicationBuilder;**

**import org.springframework.context.annotation.Bean;**

**import org.springframework.security.web.firewall.HttpFirewall;**

**import org.springframework.security.web.firewall.StrictHttpFirewall;**

**import springfox.documentation.builders.RequestHandlerSelectors;**

**import springfox.documentation.service.ApiInfo;**

**import springfox.documentation.spi.DocumentationType;**

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

**@SpringBootApplication**

**public class FoodBoxWebApiApplication {**

**public static void main(String[] args) {**

**SpringApplication.run(FoodBoxWebApiApplication.class, args);**

**System.out.println("FoodBox web app works!");**

**}**

**@Bean**

**public HttpFirewall allowUrlSemicolonHttpFirewall() {**

**StrictHttpFirewall firewall = new StrictHttpFirewall();**

**firewall.setAllowUrlEncodedDoubleSlash(true);;**

**return firewall;**

**}**

**protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {**

**return application.sources(FoodBoxWebApiApplication.class);**

**}**

**@Bean**

**public Docket swaggerConfiguration() {**

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

**.select()**

**.apis(RequestHandlerSelectors.basePackage("com.foodbox"))**

**.build()**

**.apiInfo(apiDetails());**

**}**

**private ApiInfo apiDetails() {**

**return new ApiInfo(**

**"FoodBox Web API",**

**"Spring Boot REST API for foodbox.com",**

**"1.0",**

**"Free to use",**

**new springfox.documentation.service.Contact("Kalpana Rathod", "https://github.com/Kalp1992/", "kalpanarathod6@gmail.com"),**

**"API License",**

**"https://github.com/Kalp1992/",**

**Collections.emptyList()**

**);**

**}**

**}**

**ADMIN CONTROLLER**

**package com.foodBox.controller.admin;**

**import java.util.HashMap;**

**import javax.annotation.security.DeclareRoles;**

**import javax.annotation.security.RolesAllowed;**

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

**import org.springframework.http.ResponseEntity;**

**import org.springframework.security.access.annotation.Secured;**

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

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

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

**import org.springframework.web.bind.annotation.RequestMapping;**

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

**import com.foodBox.RequestPojo.ApiResponse;**

**import com.foodBox.model.Products;**

**import com.foodBox.services.productService.ProductServiceDao;**

**//@CrossOrigin(origins = "http://localhost:4200")**

**@RestController**

**@RequestMapping("api/product/")**

**//@DeclareRoles({"ADMIN", "CUSTOMER"})**

**public class AddProductController {**

**@Autowired**

**ProductServiceDao productservice;**

**@RequestMapping("add")**

**//@RolesAllowed("ADMIN")**

**//@PreAuthorize("hasRole('ADMIN')")**

**public ResponseEntity<?> addProduct(@RequestBody HashMap<String,String> addRequest) {**

**try {**

**//TODO validation has to add for client request**

**Products product = productservice.addProduct(addRequest);**

**return ResponseEntity.ok(product);**

**}catch(Exception e ) {**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**}**

**package com.foodBox.controller.admin;**

**import java.util.HashMap;**

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

**import org.springframework.http.ResponseEntity;**

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

**import org.springframework.web.bind.annotation.RequestMapping;**

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

**import com.foodBox.RequestPojo.ApiResponse;**

**import com.foodBox.model.Admin;**

**import com.foodBox.services.adminService.AdminService;**

**@RestController**

**@RequestMapping("api/signup/")**

**public class AdminSignUpController {**

**@Autowired**

**AdminService adminservice;**

**@RequestMapping("admin")**

**public ResponseEntity<?> adminSignUp(@RequestBody HashMap<String,String> signupRequest) {**

**try {**

**//TODO validation has to add for client request**

**Admin admin = adminservice.signUpAdmin(signupRequest);**

**return ResponseEntity.ok(admin);**

**}catch(Exception e ) {**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**}**

**package com.foodBox.controller.admin;**

**import java.util.HashMap;**

**import java.util.List;**

**import javax.annotation.security.DeclareRoles;**

**import javax.annotation.security.RolesAllowed;**

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

**import org.springframework.http.ResponseEntity;**

**import org.springframework.security.access.annotation.Secured;**

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

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

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

**import org.springframework.web.bind.annotation.RequestMapping;**

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

**import com.foodBox.JWTConfiguration.ShoppingConfiguration;**

**import com.foodBox.RequestPojo.ApiResponse;**

**import com.foodBox.model.Products;**

**import com.foodBox.services.productService.ProductServiceDao;**

**//@CrossOrigin(origins = "http://localhost:4200")**

**@RestController**

**@RequestMapping("api/product")**

**//@DeclareRoles({"ADMIN", "CUSTOMER"})**

**public class DeleteProductbyIdController {**

**@Autowired**

**ProductServiceDao productService;**

**@RequestMapping("deleteProduct")**

**//@RolesAllowed("ADMIN")**

**//@PreAuthorize("hasRole('ADMIN')")**

**public ResponseEntity<?> deleteProductbyId(@RequestBody HashMap<String,String> removeProductRequest) {**

**try {**

**String keys[] = {"productId"};**

**if(ShoppingConfiguration.validationWithHashMap(keys, removeProductRequest)) {**

**}**

**List<Products> obj = productService.removeProductByProductId(Long.parseLong(removeProductRequest.get("productId")));**

**return ResponseEntity.ok(obj);**

**}catch(Exception e) {**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}}**

**USER CONTROLLER**

**package com.foodBox.controller.user;**

**import java.util.HashMap;**

**import java.util.List;**

**import javax.annotation.security.DeclareRoles;**

**import javax.annotation.security.RolesAllowed;**

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

**import org.springframework.http.ResponseEntity;**

**import org.springframework.security.access.annotation.Secured;**

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

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

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

**import org.springframework.web.bind.annotation.RequestMapping;**

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

**import com.foodBox.JWTConfiguration.ShoppingConfiguration;**

**import com.foodBox.RequestPojo.ApiResponse;**

**import com.foodBox.model.AddtoCart;**

**import com.foodBox.services.cartService.CartService;**

**//@CrossOrigin**

**@RestController**

**@RequestMapping("api/addtocart/")**

**//@DeclareRoles({"ADMIN", "CUSTOMER"})**

**public class AddtoCartController {**

**@Autowired**

**CartService cartService;**

**@RequestMapping("addProduct")**

**//@RolesAllowed("CUSTOMER")**

**//@PreAuthorize("hasRole('CUSTOMER')")**

**public ResponseEntity<?> addCartwithProduct(@RequestBody HashMap<String,String> addCartRequest) {**

**try {**

**String keys[] = {"productId","userId","qty","price"};**

**if(ShoppingConfiguration.validationWithHashMap(keys, addCartRequest)) {**

**}**

**long productId = Long.parseLong(addCartRequest.get("productId"));**

**long userId = Long.parseLong(addCartRequest.get("userId"));**

**int qty = Integer.parseInt(addCartRequest.get("qty"));**

**double price = Double.parseDouble(addCartRequest.get("price"));**

**List<AddtoCart> obj = cartService.addCartbyUserIdAndProductId(productId,userId,qty,price);**

**return ResponseEntity.ok(obj);**

**} catch (Exception e) {**

**e.printStackTrace();**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**@RequestMapping("updateQtyForCart")**

**//@RolesAllowed("CUSTOMER")**

**//@PreAuthorize("hasRole('CUSTOMER')")**

**public ResponseEntity<?> updateQtyForCart(@RequestBody HashMap<String,String> addCartRequest) {**

**try {**

**String keys[] = {"cartId","userId","qty","price"};**

**if(ShoppingConfiguration.validationWithHashMap(keys, addCartRequest)) {**

**}**

**long cartId = Long.parseLong(addCartRequest.get("cartId"));**

**long userId = Long.parseLong(addCartRequest.get("userId"));**

**int qty = Integer.parseInt(addCartRequest.get("qty"));**

**double price = Double.parseDouble(addCartRequest.get("price"));**

**cartService.updateQtyByCartId(cartId, qty, price);**

**List<AddtoCart> obj = cartService.getCartByUserId(userId);**

**return ResponseEntity.ok(obj);**

**} catch (Exception e) {**

**e.printStackTrace();**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**@RequestMapping("removeProductFromCart")**

**//@RolesAllowed("CUSTOMER")**

**//@PreAuthorize("hasRole('CUSTOMER')")**

**public ResponseEntity<?> removeCartwithProductId(@RequestBody HashMap<String,String> removeCartRequest) {**

**try {**

**String keys[] = {"userId","cartId"};**

**if(ShoppingConfiguration.validationWithHashMap(keys, removeCartRequest)) {**

**}**

**List<AddtoCart> obj = cartService.removeCartByUserId(Long.parseLong(removeCartRequest.get("cartId")), Long.parseLong(removeCartRequest.get("userId")));**

**return ResponseEntity.ok(obj);**

**}catch(Exception e) {**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**@RequestMapping("getCartsByUserId")**

**//@RolesAllowed("CUSTOMER")**

**//@PreAuthorize("hasRole('CUSTOMER')")**

**public ResponseEntity<?> getCartsByUserId(@RequestBody HashMap<String,String> getCartRequest) {**

**try {**

**String keys[] = {"userId"};**

**if(ShoppingConfiguration.validationWithHashMap(keys, getCartRequest)) {**

**}**

**List<AddtoCart> obj = cartService.getCartByUserId(Long.parseLong(getCartRequest.get("userId")));**

**return ResponseEntity.ok(obj);**

**}catch(Exception e) {**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**}**

**package com.foodBox.controller.user;**

**import java.util.HashMap;**

**import javax.annotation.security.DeclareRoles;**

**import javax.annotation.security.RolesAllowed;**

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

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

**import org.springframework.beans.factory.annotation.Qualifier;**

**import org.springframework.boot.configurationprocessor.json.JSONObject;**

**import org.springframework.http.HttpStatus;**

**import org.springframework.http.ResponseEntity;**

**import org.springframework.security.authentication.AuthenticationManager;**

**import org.springframework.security.authentication.BadCredentialsException;**

**import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;**

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

**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.RequestMapping;**

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

**import com.foodBox.JWTConfiguration.JwtTokenUtil;**

**import com.foodBox.RequestPojo.ApiResponse;**

**import com.foodBox.RequestPojo.LoginRequest;**

**import com.foodBox.model.User;**

**//@CrossOrigin**

**@RestController**

**@RequestMapping("api")**

**//@DeclareRoles({"ADMIN", "CUSTOMER"})**

**public class LoginController {**

**@Autowired AuthenticationManager authManager;**

**@Autowired JwtTokenUtil jwtUtil;**

**@RequestMapping("/login/user")**

**//@RolesAllowed({"ADMIN","CUSTOMER"})**

**public ResponseEntity<?> login(@RequestBody LoginRequest request) {**

**try {**

**Authentication authentication = authManager.authenticate(**

**new UsernamePasswordAuthenticationToken(**

**request.getEmail(), request.getPassword())**

**);**

**User user = (User) authentication.getPrincipal();**

**String accessToken = jwtUtil.generateAccessToken(user);**

**ApiResponse response = new ApiResponse(user.getEmail(), accessToken);**

**return ResponseEntity.ok().body(response);**

**} catch (BadCredentialsException ex) {**

**return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();**

**}**

**}**

**}**

**package com.foodBox.controller.user;**

**import java.util.ArrayList;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Random;**

**import javax.annotation.security.DeclareRoles;**

**import javax.annotation.security.RolesAllowed;**

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

**import org.springframework.http.ResponseEntity;**

**import org.springframework.security.access.annotation.Secured;**

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

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

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

**import org.springframework.web.bind.annotation.RequestMapping;**

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

**import com.foodBox.JWTConfiguration.ShoppingConfiguration;**

**import com.foodBox.RequestPojo.ApiResponse;**

**import com.foodBox.model.AddtoCart;**

**import com.foodBox.model.CheckoutCart;**

**import com.foodBox.services.cartService.CartService;**

**import com.foodBox.services.productService.ProductServices;**

**//@CrossOrigin**

**@RestController**

**@RequestMapping("api/order/")**

**//@DeclareRoles({"ADMIN", "CUSTOMER"})**

**public class OrderController {**

**@Autowired**

**CartService cartService;**

**ProductServices proService;**

**@RequestMapping("checkout\_order")**

**//@RolesAllowed("CUSTOMER")**

**//@PreAuthorize("hasRole('CUSTOMER')")**

**public ResponseEntity<?> checkout\_order(@RequestBody HashMap<String,String> addCartRequest) {**

**try {**

**String keys[] = {"userId","total\_price","pay\_type","deliveryAddress"};**

**if(ShoppingConfiguration.validationWithHashMap(keys, addCartRequest)) {**

**}**

**long user\_Id = Long.parseLong(addCartRequest.get("userId"));**

**double total\_amt = Double.parseDouble(addCartRequest.get("total\_price"));**

**if(cartService.checkTotalAmountAgainstCart(total\_amt,user\_Id)) {**

**List<AddtoCart> cartItems = cartService.getCartByUserId(user\_Id);**

**List<CheckoutCart> tmp = new ArrayList<CheckoutCart>();**

**for(AddtoCart addCart : cartItems) {**

**String orderId = ""+getOrderId();**

**CheckoutCart cart = new CheckoutCart();**

**cart.setPayment\_type(addCartRequest.get("pay\_type"));**

**cart.setPrice(total\_amt);**

**cart.setUser\_id(user\_Id);**

**cart.setOrder\_id(orderId);**

**cart.setProduct(addCart.getProduct());**

**cart.setQty(addCart.getQty());**

**cart.setDelivery\_address(addCartRequest.get("deliveryAddress"));**

**tmp.add(cart);**

**}**

**cartService.saveProductsForCheckout(tmp);**

**return ResponseEntity.ok(new ApiResponse("Order placed successfully", ""));**

**}else {**

**throw new Exception("Total amount is mismatch");**

**}**

**}catch(Exception e) {**

**e.printStackTrace();**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**public int getOrderId() {**

**Random r = new Random( System.currentTimeMillis() );**

**return 10000 + r.nextInt(20000);**

**}**

**@RequestMapping("getOrdersByUserId")**

**//@RolesAllowed("CUSTOMER")**

**//@PreAuthorize("hasRole('CUSTOMER')")**

**public ResponseEntity<?> getOrdersByUserId(@RequestBody HashMap<String,String> ordersRequest) {**

**try {**

**String keys[] = {"userId"};**

**return ResponseEntity.ok(new ApiResponse("Order placed successfully", ""));**

**}catch(Exception e) {**

**e.printStackTrace();**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**}**

**package com.foodBox.controller.user;**

**import java.util.HashMap;**

**import java.util.List;**

**import javax.annotation.security.DeclareRoles;**

**import javax.annotation.security.RolesAllowed;**

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

**import org.springframework.security.access.annotation.Secured;**

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

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

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

**import org.springframework.web.bind.annotation.RequestMapping;**

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

**import com.foodBox.model.Category;**

**import com.foodBox.model.Products;**

**import com.foodBox.services.productService.ProductServices;**

**//@CrossOrigin**

**@RestController**

**@RequestMapping("api/product/")**

**//@DeclareRoles({"ADMIN", "CUSTOMER"})**

**public class ProductController {**

**@Autowired**

**ProductServices productServices;**

**@RequestMapping("getAll")**

**//@RolesAllowed({"CUSTOMER","ADMIN"})**

**//@PreAuthorize("hasRole({'CUSTOMER','ADMIN'})")**

**public List<Products> getAllPRoducts(){**

**return productServices.getAllProducts();**

**}**

**@RequestMapping("getAllCategory")**

**//@RolesAllowed({"CUSTOMER","ADMIN"})**

**//@PreAuthorize("hasRole({'CUSTOMER','ADMIN'})")**

**public List<Category> getAllCategory(){**

**return productServices.getAllCategory();**

**}**

**@RequestMapping("getProductsByCategory")**

**//@RolesAllowed({"CUSTOMER","ADMIN"})**

**//@PreAuthorize("hasRole({'CUSTOMER','ADMIN'})")**

**public List<Products> getProductsByCategory(@RequestBody HashMap<String,String> request){**

**String category\_id = request.get("cat\_id");**

**return productServices.getProductsByCategory(category\_id);**

**}**

**}**

**package com.foodBox.controller.user;**

**import java.util.HashMap;**

**import javax.annotation.security.RolesAllowed;**

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

**import org.springframework.http.ResponseEntity;**

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

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

**import org.springframework.web.bind.annotation.RequestMapping;**

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

**import com.foodBox.RequestPojo.ApiResponse;**

**import com.foodBox.model.User;**

**import com.foodBox.services.userService.UserService;**

**//@CrossOrigin**

**@RestController**

**@RequestMapping("api/signup/")**

**public class SignUpController {**

**@Autowired**

**UserService userservice;**

**@RequestMapping("user")**

**//@RolesAllowed({"ADMIN","CUSTOMER"})**

**public ResponseEntity<?> userSignUp(@RequestBody HashMap<String,String> signupRequest) {**

**try {**

**//TODO validation has to add for client request**

**User user = userservice.signUpUser(signupRequest);**

**return ResponseEntity.ok(user);**

**}catch(Exception e ) {**

**return ResponseEntity.badRequest().body(new ApiResponse(e.getMessage(), ""));**

**}**

**}**

**}**

**JWT CONFIGURATION**

**package com.foodBox.JWTConfiguration;**

**import java.util.Arrays;**

**import java.util.Collections;**

**import java.util.List;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

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

**import org.springframework.context.annotation.Bean;**

**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.core.userdetails.UsernameNotFoundException;**

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

**import org.springframework.security.crypto.factory.PasswordEncoderFactories;**

**import org.springframework.security.crypto.password.PasswordEncoder;**

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

**import org.springframework.security.web.header.writers.StaticHeadersWriter;**

**import org.springframework.web.cors.CorsConfiguration;**

**import org.springframework.web.cors.CorsConfigurationSource;**

**import org.springframework.web.cors.UrlBasedCorsConfigurationSource;**

**import org.springframework.web.filter.CorsFilter;**

**import com.foodBox.Repository.UserRepository;**

**@EnableWebSecurity**

**@EnableGlobalMethodSecurity(**

**prePostEnabled = true, securedEnabled = true, jsr250Enabled = true**

**)**

**public class ApplicationSecurity extends WebSecurityConfigurerAdapter {**

**@Autowired**

**private UserRepository userRepo;**

**@Autowired**

**CorsConfigurationSource webMvcConfig;**

**@Autowired**

**private JwtTokenFilter jwtTokenFilter;**

**@Override**

**protected void configure(HttpSecurity http) throws Exception {**

**// http.cors().configurationSource(new CorsConfigurationSource() {**

**//**

**// @Override**

**// public CorsConfiguration getCorsConfiguration(HttpServletRequest request) {**

**// CorsConfiguration config = new CorsConfiguration();**

**// config.setAllowedHeaders(Collections.singletonList("\*"));**

**// config.setAllowedMethods(Collections.singletonList("\*"));**

**// config.addAllowedOrigin("\*");**

**// config.setAllowCredentials(true);**

**// return config;**

**// }**

**// });**

**http.csrf().disable();**

**http.cors();**

**// http.headers().frameOptions().disable().and().csrf().disable().headers()**

**// // the headers you want here. This solved all my CORS problems!**

**// .addHeaderWriter(new StaticHeadersWriter("Access-Control-Allow-Origin", "\*"))**

**// .addHeaderWriter(new StaticHeadersWriter("Access-Control-Allow-Methods", "POST, GET"))**

**// .addHeaderWriter(new StaticHeadersWriter("Access-Control-Max-Age", "3600"))**

**// .addHeaderWriter(new StaticHeadersWriter("Access-Control-Allow-Credentials", "true"))**

**// .addHeaderWriter(new StaticHeadersWriter("Access-Control-Allow-Headers", "Origin,Accept,X-Requested-With,Content-Type,Access-Control-Request-Method,Access-Control-Request-Headers,Authorization"));**

**http.authorizeRequests()**

**.antMatchers("/",**

**"/favicon.ico",**

**"/\*\*/\*.png",**

**"/\*\*/\*.gif",**

**"/\*\*/\*.svg",**

**"/\*\*/\*.jpg",**

**"/\*\*/\*.html",**

**"/\*\*/\*.css",**

**"/\*\*/\*.js")**

**.permitAll()**

**.antMatchers("/api/status/\*\*","/api/login/user","/api/signup/\*\*")**

**.permitAll()**

**// .antMatchers("/api/user/checkUsernameAvailability", "/api/user/checkEmailAvailability")**

**// .permitAll().antMatchers(HttpMethod.GET, "/api/polls/\*\*", "/api/users/\*\*")**

**.antMatchers(HttpMethod.GET, "/api/status/\*\*")**

**.permitAll()**

**.antMatchers(HttpMethod.OPTIONS).permitAll()**

**.anyRequest()**

**.authenticated();**

**http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);**

**http.exceptionHandling()**

**.authenticationEntryPoint(**

**(request, response, ex) -> {**

**response.sendError(**

**HttpServletResponse.SC\_UNAUTHORIZED,**

**ex.getMessage()**

**);**

**}**

**);**

**http.addFilterBefore(jwtTokenFilter, UsernamePasswordAuthenticationFilter.class);**

**}**

**// @Bean**

**// public CorsFilter corsFilter() {**

**// final UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();**

**// final CorsConfiguration config = new CorsConfiguration();**

**// config.setAllowedOrigins(Collections.singletonList("http://localhost:4200")); // Provide list of origins if you want multiple origins**

**// config.setAllowedHeaders(Arrays.asList("Origin", "Content-Type", "Accept"));**

**// config.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "OPTIONS", "DELETE", "PATCH"));**

**// config.setAllowCredentials(true);**

**// source.registerCorsConfiguration("/\*\*", config);**

**// return new CorsFilter(source);**

**// }**

**//**

**@Override**

**protected void configure(AuthenticationManagerBuilder auth) throws Exception {**

**auth.userDetailsService(**

**username -> userRepo.findByEmail(username)**

**.orElseThrow(**

**() -> new UsernameNotFoundException("User " + username + " not found.")));**

**}**

**// @Bean**

**// public PasswordEncoder passwordEncoder() {**

**// return new BCryptPasswordEncoder();**

**// }**

**@Bean**

**public BCryptPasswordEncoder passwordEncoder() {**

**return new BCryptPasswordEncoder();**

**}**

**@Override**

**@Bean**

**public AuthenticationManager authenticationManagerBean() throws Exception {**

**return super.authenticationManagerBean();**

**}**

**}**

**package com.foodBox.JWTConfiguration;**

**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.util.ObjectUtils;**

**import org.springframework.web.filter.OncePerRequestFilter;**

**import com.foodBox.model.Role;**

**import com.foodBox.model.User;**

**import io.jsonwebtoken.Claims;**

**@Component**

**public class JwtTokenFilter extends OncePerRequestFilter{**

**@Autowired**

**private JwtTokenUtil jwtUtil;**

**@Override**

**protected void doFilterInternal(HttpServletRequest request,**

**HttpServletResponse response, FilterChain filterChain)**

**throws ServletException, IOException {**

**if (!hasAuthorizationBearer(request)) {**

**filterChain.doFilter(request, response);**

**return;**

**}**

**String token = getAccessToken(request);**

**if (!jwtUtil.validateAccessToken(token)) {**

**filterChain.doFilter(request, response);**

**return;**

**}**

**setAuthenticationContext(token, request);**

**filterChain.doFilter(request, response);**

**}**

**private boolean hasAuthorizationBearer(HttpServletRequest request) {**

**String header = request.getHeader("Authorization");**

**if (ObjectUtils.isEmpty(header) || !header.startsWith("Bearer")) {**

**return false;**

**}**

**return true;**

**}**

**private String getAccessToken(HttpServletRequest request) {**

**String header = request.getHeader("Authorization");**

**String token = header.split(" ")[1].trim();**

**return token;**

**}**

**private void setAuthenticationContext(String token, HttpServletRequest request) {**

**UserDetails userDetails = getUserDetails(token);**

**UsernamePasswordAuthenticationToken**

**authentication = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());**

**authentication.setDetails(**

**new WebAuthenticationDetailsSource().buildDetails(request));**

**SecurityContextHolder.getContext().setAuthentication(authentication);**

**}**

**private UserDetails getUserDetails(String token) {**

**User userDetails = new User();**

**Claims claims = jwtUtil.parseClaims(token);**

**String subject = (String) claims.get(Claims.SUBJECT);**

**String roles = (String) claims.get("roles");**

**roles = roles.replace("[", "").replace("]", "");**

**String[] roleNames = roles.split(",");**

**for (String aRoleName : roleNames) {**

**userDetails.addRole(new Role(aRoleName));**

**}**

**String[] jwtSubject = subject.split(",");**

**userDetails.setId(Long.parseLong(jwtSubject[0]));**

**userDetails.setEmail(jwtSubject[1]);**

**return userDetails;**

**}**

**}**

**package com.foodBox.JWTConfiguration;**

**import java.util.Date;**

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

**import org.springframework.beans.factory.annotation.Value;**

**import org.springframework.stereotype.Component;**

**import com.foodBox.model.User;**

**import io.jsonwebtoken.Claims;**

**import io.jsonwebtoken.ExpiredJwtException;**

**import io.jsonwebtoken.Jwts;**

**import io.jsonwebtoken.MalformedJwtException;**

**import io.jsonwebtoken.SignatureAlgorithm;**

**import io.jsonwebtoken.SignatureException;**

**import io.jsonwebtoken.UnsupportedJwtException;**

**@Component**

**public class JwtTokenUtil {**

**private static final long EXPIRE\_DURATION = 24 \* 60 \* 60 \* 1000; // 24 hour**

**private static final Logger LOGGER = LoggerFactory.getLogger(JwtTokenUtil.class);**

**@Value("${app.jwt.secret}")**

**private String SECRET\_KEY;**

**public String generateAccessToken(User user) {**

**return Jwts.builder()**

**.setSubject(String.format("%s,%s", user.getId(), user.getEmail()))**

**.setIssuer("FoodBox")**

**.claim("roles", user.getRoles().toString())**

**.setIssuedAt(new Date())**

**.setExpiration(new Date(System.currentTimeMillis() + EXPIRE\_DURATION))**

**.signWith(SignatureAlgorithm.HS512, SECRET\_KEY)**

**.compact();**

**}**

**public boolean validateAccessToken(String token) {**

**try {**

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

**return true;**

**} catch (ExpiredJwtException ex) {**

**LOGGER.error("JWT expired", ex.getMessage());**

**} catch (IllegalArgumentException ex) {**

**LOGGER.error("Token is null, empty or only whitespace", ex.getMessage());**

**} catch (MalformedJwtException ex) {**

**LOGGER.error("JWT is invalid", ex);**

**} catch (UnsupportedJwtException ex) {**

**LOGGER.error("JWT is not supported", ex);**

**} catch (SignatureException ex) {**

**LOGGER.error("Signature validation failed");**

**}**

**return false;**

**}**

**public String getSubject(String token) {**

**return parseClaims(token).getSubject();**

**}**

**public Claims parseClaims(String token) {**

**return Jwts.parser()**

**.setSigningKey(SECRET\_KEY)**

**.parseClaimsJws(token)**

**.getBody();**

**}**

**}**

**package com.foodBox.JWTConfiguration;**

**import java.util.HashMap;**

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

**@Configuration**

**public class ShoppingConfiguration {**

**public static Boolean validationWithHashMap(String keys[],HashMap<String,String> request) throws Exception{**

**Boolean status = false;**

**try {**

**for(int start = 0;start<keys.length;start++) {**

**if(request.containsKey(keys[start])) {//not exist**

**if(request.get(keys[start]).equals("")){//if empty**

**throw new Exception(keys[start]+" Should not be empty");**

**}**

**}else {**

**throw new Exception(keys[start]+" is missing");**

**}**

**}**

**}catch(Exception e) {**

**e.printStackTrace();**

**throw new Exception("Error is "+e.getMessage());**

**}**

**return status;**

**}**

**}**

**package com.foodBox.JWTConfiguration;**

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

**import org.springframework.web.servlet.config.annotation.CorsRegistry;**

**import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;**

**@Configuration**

**public class WebMvcConfig implements WebMvcConfigurer {**

**private final long MAX\_AGE\_SECS = 3600;**

**@Override**

**public void addCorsMappings(CorsRegistry registry) {**

**registry.addMapping("/\*\*")**

**.allowedOriginPatterns("\*")**

**.allowedMethods("HEAD", "OPTIONS", "GET", "POST", "PUT", "PATCH", "DELETE")**

**.allowedHeaders("\*")**

**.allowCredentials(true);**

**//.allowedMethods("HEAD", "OPTIONS", "GET", "POST")**

**// .maxAge(MAX\_AGE\_SECS);**

**}**

**}**

**MODEL**

**package com.foodBox.model;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.FetchType;**

**import javax.persistence.Id;**

**import javax.persistence.JoinColumn;**

**import javax.persistence.OneToOne;**

**import javax.persistence.Table;**

**import javax.persistence.Transient;**

**import com.fasterxml.jackson.annotation.JsonIgnore;**

**@Entity**

**@Table(name = "add\_to\_cart")**

**public class AddtoCart {**

**@Id**

**long id;**

**@JsonIgnore**

**@OneToOne(fetch=FetchType.LAZY)**

**@JoinColumn(name = "product\_id")**

**Products product;**

**int qty;**

**double price;**

**Long user\_id;**

**@Column(updatable=false, insertable=false)**

**String added\_date;**

**@Transient**

**String productName;**

**public long getId() {**

**return id;**

**}**

**public void setId(long id) {**

**this.id = id;**

**}**

**public Products getProduct() {**

**return product;**

**}**

**public void setProduct(Products product) {**

**this.product = product;**

**}**

**public int getQty() {**

**return qty;**

**}**

**public void setQty(int qty) {**

**this.qty = qty;**

**}**

**public double getPrice() {**

**return price;**

**}**

**public void setPrice(double price) {**

**this.price = price;**

**}**

**public Long getUser\_id() {**

**return user\_id;**

**}**

**public void setUser\_id(Long user\_id) {**

**this.user\_id = user\_id;**

**}**

**public String getAdded\_date() {**

**return added\_date;**

**}**

**public void setAdded\_date(String added\_date) {**

**this.added\_date = added\_date;**

**}**

**public String getProductName() {**

**return productName;**

**}**

**}**

**package com.foodBox.model;**

**import javax.persistence.Entity;**

**import javax.persistence.Id;**

**import javax.persistence.Table;**

**@Entity**

**@Table(name="admins")**

**public class Admin {**

**@Id**

**long id;**

**String name,email,**

**password, created\_at,**

**login\_token,**

**type,**

**address,**

**is\_email\_verified,**

**mobile;**

**public long getId() {**

**return id;**

**}**

**public void setId(long id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getEmail() {**

**return email;**

**}**

**public void setEmail(String email) {**

**this.email = email;**

**}**

**public String getPassword() {**

**return password;**

**}**

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

**this.password = password;**

**}**

**public String getCreated\_at() {**

**return created\_at;**

**}**

**public void setCreated\_at(String created\_at) {**

**this.created\_at = created\_at;**

**}**

**public String getLogin\_token() {**

**return login\_token;**

**}**

**public void setLogin\_token(String login\_token) {**

**this.login\_token = login\_token;**

**}**

**public String getType() {**

**return type;**

**}**

**public void setType(String type) {**

**this.type = type;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**public String getIs\_email\_verified() {**

**return is\_email\_verified;**

**}**

**public void setIs\_email\_verified(String is\_email\_verified) {**

**this.is\_email\_verified = is\_email\_verified;**

**}**

**public String getMobile() {**

**return mobile;**

**}**

**public void setMobile(String mobile) {**

**this.mobile = mobile;**

**}**

**}**

**package com.foodBox.model;**

**import javax.persistence.Entity;**

**import javax.persistence.Id;**

**@Entity(name="category")**

**public class Category {**

**@Id**

**long id;**

**String name;**

**public long getId() {**

**return id;**

**}**

**public void setId(long id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**}**

**package com.foodBox.model;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.FetchType;**

**import javax.persistence.GeneratedValue;**

**import javax.persistence.GenerationType;**

**import javax.persistence.Id;**

**import javax.persistence.JoinColumn;**

**import javax.persistence.OneToOne;**

**import javax.persistence.Table;**

**@Entity**

**@Table(name="checkout\_cart")**

**public class CheckoutCart {**

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**long id;**

**String order\_id,payment\_type,delivery\_address;**

**long user\_id;**

**@OneToOne(fetch=FetchType.LAZY)**

**@JoinColumn(name = "product\_id")**

**Products product;**

**//long ;**

**int qty;**

**double price;**

**@Column(updatable=false, insertable=false)**

**String order\_date;**

**public long getId() {**

**return id;**

**}**

**public void setId(long id) {**

**this.id = id;**

**}**

**public String getOrder\_id() {**

**return order\_id;**

**}**

**public void setOrder\_id(String order\_id) {**

**this.order\_id = order\_id;**

**}**

**public String getPayment\_type() {**

**return payment\_type;**

**}**

**public void setPayment\_type(String payment\_type) {**

**this.payment\_type = payment\_type;**

**}**

**public String getDelivery\_address() {**

**return delivery\_address;**

**}**

**public void setDelivery\_address(String delivery\_address) {**

**this.delivery\_address = delivery\_address;**

**}**

**public long getUser\_id() {**

**return user\_id;**

**}**

**public void setUser\_id(long user\_id) {**

**this.user\_id = user\_id;**

**}**

**public Products getProduct() {**

**return product;**

**}**

**public void setProduct(Products product) {**

**this.product = product;**

**}**

**public int getQty() {**

**return qty;**

**}**

**public void setQty(int qty) {**

**this.qty = qty;**

**}**

**public double getPrice() {**

**return price;**

**}**

**public void setPrice(double price) {**

**this.price = price;**

**}**

**public String getOrder\_date() {**

**return order\_date;**

**}**

**public void setOrder\_date(String order\_date) {**

**this.order\_date = order\_date;**

**}**

**}**

**package com.foodBox.model;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.Id;**

**import javax.persistence.Table;**

**@Entity**

**@Table(name="products")**

**public class Products {**

**@Id**

**long id;**

**String name,price,category\_id, description;**

**@Column(updatable=false, insertable=false)**

**String added\_on;**

**public long getId() {**

**return id;**

**}**

**public void setId(long id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getPrice() {**

**return price;**

**}**

**public void setPrice(String price) {**

**this.price = price;**

**}**

**public String getAdded\_on() {**

**return added\_on;**

**}**

**public void setAdded\_on(String added\_on) {**

**this.added\_on = added\_on;**

**}**

**public String getCategory\_id() {**

**return category\_id;**

**}**

**public void setCategory\_id(String category\_id) {**

**this.category\_id = category\_id;**

**}**

**public String getDescription() {**

**return description;**

**}**

**public void setDescription(String description) {**

**this.description = description;**

**}**

**}**

**package com.foodBox.model;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.GeneratedValue;**

**import javax.persistence.GenerationType;**

**import javax.persistence.Id;**

**import javax.persistence.Table;**

**@Entity**

**@Table(name="roles")**

**public class Role {**

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**private Long id;**

**@Column(nullable = false, length = 50, unique = true)**

**private String name;**

**public Role() { }**

**public Role(String name) {**

**this.name = name;**

**}**

**public Role(Long id) {**

**this.id = id;**

**}**

**public Long getId() {**

**return id;**

**}**

**public void setId(Long id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**@Override**

**public String toString() {**

**return this.name;**

**}**

**}**

**package com.foodBox.model;**

**import java.util.ArrayList;**

**import java.util.Collection;**

**import java.util.HashSet;**

**import java.util.List;**

**import java.util.Set;**

**import javax.persistence.Entity;**

**import javax.persistence.GeneratedValue;**

**import javax.persistence.GenerationType;**

**import javax.persistence.Id;**

**import javax.persistence.JoinColumn;**

**import javax.persistence.JoinTable;**

**import javax.persistence.ManyToMany;**

**import javax.persistence.Table;**

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

**import org.springframework.security.core.authority.SimpleGrantedAuthority;**

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

**@Entity**

**@Table(name="users")**

**public class User implements UserDetails{**

**@Id @GeneratedValue(strategy = GenerationType.IDENTITY)**

**long id;**

**String name,email,**

**password, created\_at,**

**login\_token,**

**type,**

**address,**

**is\_email\_verified,**

**mobile;**

**public User() {**

**super();**

**}**

**public User(long id, String name, String email, String password, String created\_at, String login\_token, String type,**

**String address, String is\_email\_verified, String mobile) {**

**super();**

**this.id = id;**

**this.name = name;**

**this.email = email;**

**this.password = password;**

**this.created\_at = created\_at;**

**this.login\_token = login\_token;**

**this.type = type;**

**this.address = address;**

**this.is\_email\_verified = is\_email\_verified;**

**this.mobile = mobile;**

**}**

**public User(String email, String password) {**

**super();**

**this.email = email;**

**this.password = password;**

**}**

**@ManyToMany**

**@JoinTable(**

**name = "users\_roles",**

**joinColumns = @JoinColumn(name = "user\_id"),**

**inverseJoinColumns = @JoinColumn(name = "role\_id")**

**)**

**private Set<Role> roles = new HashSet<>();**

**public long getId() {**

**return id;**

**}**

**public void setId(long id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getEmail() {**

**return email;**

**}**

**public void setEmail(String email) {**

**this.email = email;**

**}**

**public String getPassword() {**

**return password;**

**}**

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

**this.password = password;**

**}**

**public String getCreated\_at() {**

**return created\_at;**

**}**

**public void setCreated\_at(String created\_at) {**

**this.created\_at = created\_at;**

**}**

**public String getLogin\_token() {**

**return login\_token;**

**}**

**public void setLogin\_token(String login\_token) {**

**this.login\_token = login\_token;**

**}**

**public String getType() {**

**return type;**

**}**

**public void setType(String type) {**

**this.type = type;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**public String getIs\_email\_verified() {**

**return is\_email\_verified;**

**}**

**public void setIs\_email\_verified(String is\_email\_verified) {**

**this.is\_email\_verified = is\_email\_verified;**

**}**

**public String getMobile() {**

**return mobile;**

**}**

**public void setMobile(String mobile) {**

**this.mobile = mobile;**

**}**

**@Override**

**public Collection<? extends GrantedAuthority> getAuthorities() {**

**List<SimpleGrantedAuthority> authorities = new ArrayList<>();**

**for (Role role : roles) {**

**authorities.add(new SimpleGrantedAuthority(role.getName()));**

**}**

**return authorities;**

**}**

**public Set<Role> getRoles() {**

**return roles;**

**}**

**public void setRoles(Set<Role> roles) {**

**this.roles = roles;**

**}**

**public void addRole(Role role) {**

**this.roles.add(role);**

**}**

**@Override**

**public String getUsername() {**

**return this.email;**

**}**

**@Override**

**public boolean isAccountNonExpired() {**

**return true;**

**}**

**@Override**

**public boolean isAccountNonLocked() {**

**return true;**

**}**

**@Override**

**public boolean isCredentialsNonExpired() {**

**return true;**

**}**

**@Override**

**public boolean isEnabled() {**

**return true;**

**}**

**}**

**REPOSITORY**

**package com.foodBox.Repository;**

**import java.util.List;**

**import java.util.Optional;**

**import javax.transaction.Transactional;**

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

**import org.springframework.data.jpa.repository.Modifying;**

**import org.springframework.data.jpa.repository.Query;**

**import org.springframework.data.repository.query.Param;**

**import org.springframework.stereotype.Repository;**

**import com.foodBox.model.AddtoCart;**

**@Repository**

**public interface AddToCartRepo extends JpaRepository<AddtoCart,Long> {**

**@Query("Select sum(addCart.price) FROM AddtoCart addCart WHERE addCart.user\_id=:user\_id")**

**double getTotalAmountByUserId(@Param("user\_id")Long user\_id);**

**@Query("Select addCart FROM AddtoCart addCart WHERE addCart.user\_id=:user\_id")**

**List<AddtoCart> getCartByuserId(@Param("user\_id")Long user\_id);**

**@Query("Select addCart FROM AddtoCart addCart ")**

**Optional<AddtoCart> getCartByuserIdtest();**

**@Query("Select addCart FROM AddtoCart addCart WHERE addCart.product.id= :product\_id and addCart.user\_id=:user\_id")**

**Optional<AddtoCart> getCartByProductIdAnduserId(@Param("user\_id")Long user\_id,@Param("product\_id")Long product\_id);**

**@Modifying**

**@Transactional**

**@Query("DELETE FROM AddtoCart addCart WHERE addCart.id =:cart\_id and addCart.user\_id=:user\_id")**

**void deleteCartByIdAndUserId(@Param("user\_id")Long user\_id,@Param("cart\_id")Long cart\_id);**

**@Modifying**

**@Transactional**

**@Query("DELETE FROM AddtoCart addCart WHERE addCart.user\_id=:user\_id")**

**void deleteAllCartByUserId(@Param("user\_id")Long user\_id);**

**@Modifying**

**@Transactional**

**@Query("DELETE FROM AddtoCart addCart WHERE addCart.user\_id=:user\_id")**

**void deleteAllCartUserId(@Param("user\_id")Long user\_id);**

**@Modifying**

**@Transactional**

**@Query("update AddtoCart addCart set addCart.qty=:qty,addCart.price=:price WHERE addCart.id=:cart\_id")**

**void updateQtyByCartId(@Param("cart\_id")Long cart\_id,@Param("price")double price,@Param("qty")Integer qty);**

**}**

**package com.foodBox.Repository;**

**import java.util.Optional;**

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

**import org.springframework.stereotype.Repository;**

**import com.foodBox.model.Admin;**

**@Repository**

**public interface AdminRepository extends JpaRepository<Admin, Long> {**

**Optional<Admin> findByMobile(String mobile);**

**}**

**package com.foodBox.Repository;**

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

**import com.foodBox.model.Category;**

**public interface CategoryRepo extends JpaRepository<Category, Long> {**

**}**

**package com.foodBox.Repository;**

**import java.util.List;**

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

**import org.springframework.data.jpa.repository.Query;**

**import org.springframework.data.repository.query.Param;**

**import org.springframework.stereotype.Repository;**

**import com.foodBox.model.CheckoutCart;**

**@Repository**

**public interface CheckoutRepo extends JpaRepository<CheckoutCart, Long> {**

**@Query("Select checkCart FROM CheckoutCart checkCart WHERE checkCart.user\_id=:user\_id")**

**List<CheckoutCart> getByuserId(@Param("user\_id")Long user\_id);**

**}**

**package com.foodBox.Repository;**

**import java.util.List;**

**import javax.transaction.Transactional;**

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

**import org.springframework.data.jpa.repository.Modifying;**

**import org.springframework.data.jpa.repository.Query;**

**import org.springframework.data.repository.query.Param;**

**import org.springframework.stereotype.Repository;**

**import com.foodBox.model.Products;**

**@Repository**

**public interface ProductRepo extends JpaRepository<Products, Long>{**

**@Query("Select pro FROM Products pro WHERE pro.category\_id=:cat\_id")**

**List<Products> getByCategoryId(@Param("cat\_id")String cat\_id);**

**@Modifying**

**@Transactional**

**@Query("DELETE FROM Products pro WHERE pro.id =:product\_id")**

**void deleteProductById(@Param("product\_id")Long product\_id);**

**}**

**package com.foodBox.Repository;**

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

**import com.foodBox.model.Role;**

**public interface RoleRepository extends JpaRepository<Role, Long> {**

**}**

**package com.foodBox.Repository;**

**import java.util.Optional;**

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

**import org.springframework.stereotype.Repository;**

**import com.foodBox.model.User;**

**@Repository**

**public interface UserRepository extends JpaRepository<User, Long> {**

**//Optional<User> findByMobile(String mobile);**

**Optional<User> findByEmail(String email);**

**Optional<User> findById(Long id);**

**}**

**REQUEST POJO**

**package com.foodBox.RequestPojo;**

**import java.util.Arrays;**

**import java.util.HashMap;**

**import java.util.List;**

**import org.springframework.boot.configurationprocessor.json.JSONArray;**

**import org.springframework.boot.configurationprocessor.json.JSONObject;**

**public class ApiResponse {**

**private String email;**

**private String accessToken;**

**public ApiResponse() { }**

**public ApiResponse(String email, String accessToken) {**

**this.email = email;**

**this.accessToken = accessToken;**

**}**

**public String getEmail() {**

**return email;**

**}**

**public void setEmail(String email) {**

**this.email = email;**

**}**

**public String getAccessToken() {**

**return accessToken;**

**}**

**public void setAccessToken(String accessToken) {**

**this.accessToken = accessToken;**

**}**

**}**

**package** com.foodBox.RequestPojo;

**public** **class** LoginRequest {

String email;

String password;

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

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

**this**.password = password;

}

}

ADMIN SERVICE

package com.foodBox.services.adminService;

import java.util.HashMap;

import org.springframework.stereotype.Service;

import com.foodBox.model.Admin;

@Service

public interface AdminService {

Admin findByMobile(String mobile) throws Exception;

Admin getAdminDetailById(long adminId) throws Exception;

Admin signUpAdmin(HashMap<String,String> signupRequest) throws Exception;

}

package com.foodBox.services.adminService;

import java.util.HashMap;

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

import org.springframework.stereotype.Service;

import com.foodBox.Repository.AdminRepository;

import com.foodBox.model.Admin;

@Service

public class AdminServiceImpl implements AdminService{

@Autowired

AdminRepository adminRepo;

@Override

public Admin findByMobile(String mobile) throws Exception {

return adminRepo.findByMobile(mobile).orElseThrow(()->new Exception("Admin Not found.."));

}

@Override

public Admin getAdminDetailById(long adminId) throws Exception {

return adminRepo.findById(adminId).orElseThrow(()->new Exception("Admin Not found.."));

}

@Override

public Admin signUpAdmin(HashMap<String, String> signupRequest) throws Exception {

try {

if(adminRepo.findByMobile(signupRequest.get("mobile")).isPresent()) {

throw new Exception("Admin is already registered with Mobile No.");

}

Admin admin = new Admin();

admin.setName(signupRequest.get("name"));

admin.setEmail(signupRequest.get("email"));

admin.setMobile(signupRequest.get("mobile"));

admin.setPassword(signupRequest.get("password"));

adminRepo.save(admin);

return admin;

}catch(Exception e) {

throw new Exception(e.getMessage());

}

}

}

CART SERVICE

package com.foodBox.services.cartService;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import com.foodBox.Repository.AddToCartRepo;

import com.foodBox.Repository.CheckoutRepo;

import com.foodBox.model.AddtoCart;

import com.foodBox.model.CheckoutCart;

import com.foodBox.model.Products;

import com.foodBox.services.productService.ProductServices;

@Service

public class CartSerivceImpl implements CartService {

@Autowired

AddToCartRepo addCartRepo;

@Autowired

CheckoutRepo checkOutRepo;

@Autowired

ProductServices proServices;

private static final Logger logger = LoggerFactory.getLogger(CartSerivceImpl.class);

@Override

public List<AddtoCart> addCartbyUserIdAndProductId(long productId, long userId,int qty,double price) throws Exception {

try {

if(addCartRepo.getCartByProductIdAnduserId(userId, productId).isPresent()){

throw new Exception("Product is already exist.");

}

AddtoCart obj = new AddtoCart();

obj.setQty(qty);

obj.setUser\_id(userId);

Products pro = proServices.getProductsById(productId);

obj.setProduct(pro);

//TODO price has to check with qty

obj.setPrice(price);

addCartRepo.save(obj);

return this.getCartByUserId(userId);

}catch(Exception e) {

e.printStackTrace();

logger.error(""+e.getMessage());

throw new Exception(e.getMessage());

}

}

@Override

public List<AddtoCart> getCartByUserId(long userId) {

return addCartRepo.getCartByuserId(userId);

}

@Override

public List<AddtoCart> removeCartByUserId(long cartId, long userId) {

addCartRepo.deleteCartByIdAndUserId(userId, cartId);

return this.getCartByUserId(userId);

}

@Override

public void updateQtyByCartId(long cartId, int qty, double price) throws Exception {

addCartRepo.updateQtyByCartId(cartId,price,qty);

}

@Override

public Boolean checkTotalAmountAgainstCart(double totalAmount,long userId) {

double total\_amount =addCartRepo.getTotalAmountByUserId(userId);

if(total\_amount == totalAmount) {

return true;

}

System.out.print("Error from request "+total\_amount +" --db-- "+ totalAmount);

return false;

}

@Override

public List<CheckoutCart> getAllCheckoutByUserId(long userId) {

return checkOutRepo.getByuserId(userId);

}

@Override

public List<CheckoutCart> saveProductsForCheckout(List<CheckoutCart> tmp) throws Exception {

try {

long user\_id = tmp.get(0).getUser\_id();

if(tmp.size() >0) {

checkOutRepo.saveAll(tmp);

this.removeAllCartByUserId(user\_id);

return this.getAllCheckoutByUserId(user\_id);

}

else {

throw new Exception("Should not be empty");

}

}catch(Exception e) {

throw new Exception("Error while checkout "+e.getMessage());

}

}

@Override

public List<AddtoCart> removeAllCartByUserId(long userId) {

addCartRepo.deleteAllCartByUserId(userId);

return null;

}

}

package com.foodBox.services.cartService;

import java.util.List;

import org.springframework.stereotype.Service;

import com.foodBox.model.AddtoCart;

import com.foodBox.model.CheckoutCart;

@Service

public interface CartService {

List<AddtoCart> addCartbyUserIdAndProductId(long productId,long userId,int qty,double price) throws Exception;

void updateQtyByCartId(long cartId,int qty,double price) throws Exception;

List<AddtoCart> getCartByUserId(long userId);

List<AddtoCart> removeCartByUserId(long cartId,long userId);

List<AddtoCart> removeAllCartByUserId(long userId);

Boolean checkTotalAmountAgainstCart(double totalAmount,long userId);

List<CheckoutCart> getAllCheckoutByUserId(long userId);

List<CheckoutCart> saveProductsForCheckout(List<CheckoutCart> tmp) throws Exception;

}

**PRODUCT SERVICE**

**package com.foodBox.services.productService;**

**import java.util.HashMap;**

**import java.util.List;**

**import org.springframework.stereotype.Service;**

**import com.foodBox.model.Products;**

**@Service**

**public interface ProductServiceDao {**

**Products addProduct(HashMap<String,String> addProductRequest) throws Exception;**

**List<Products> removeProductByProductId(long productId);**

**}**

**package com.foodBox.services.productService;**

**import java.util.HashMap;**

**import java.util.List;**

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

**import org.springframework.stereotype.Service;**

**import com.foodBox.Repository.ProductRepo;**

**import com.foodBox.model.Products;**

**@Service**

**public class ProductServiceDaoImpl implements ProductServiceDao {**

**@Autowired**

**ProductRepo productRepo;**

**@Override**

**public Products addProduct(HashMap<String, String> addProductRequest) throws Exception {**

**try {**

**Products product = new Products();**

**product.setName(addProductRequest.get("name"));**

**product.setPrice(addProductRequest.get("price"));**

**product.setCategory\_id(addProductRequest.get("category\_id"));**

**product.setDescription(addProductRequest.get("description"));**

**product.setAdded\_on(addProductRequest.get("added\_on"));**

**productRepo.save(product);**

**return product;**

**}catch(Exception e) {**

**throw new Exception(e.getMessage());**

**}**

**}**

**@Override**

**public List<Products> removeProductByProductId(long productId) {**

**productRepo.deleteProductById(productId);;**

**return productRepo.findAll();**

**}**

**}**

**package com.foodBox.services.productService;**

**import java.util.List;**

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

**import org.springframework.stereotype.Service;**

**import com.foodBox.Repository.CategoryRepo;**

**import com.foodBox.Repository.ProductRepo;**

**import com.foodBox.model.Category;**

**import com.foodBox.model.Products;**

**@Service**

**public class ProductServices {**

**@Autowired**

**ProductRepo productRepo;**

**@Autowired**

**CategoryRepo cateRepo;**

**public List<Products>getAllProducts(){**

**return productRepo.findAll();**

**}**

**public List<Products>getProductsByCategory(String product\_id){**

**return productRepo.getByCategoryId(product\_id);**

**}**

**public List<Category>getAllCategory(){**

**return cateRepo.findAll();**

**}**

**public Products getProductsById(long productId) throws Exception {**

**return productRepo.findById(productId).orElseThrow(() ->new Exception("Product is not found"));**

**}**

**}**

**USER SERVICE**

**package com.foodBox.services.userService;**

**import java.util.HashMap;**

**import org.springframework.stereotype.Service;**

**import com.foodBox.model.User;**

**@Service**

**public interface UserService {**

**User findByEmail(String mobile) throws Exception;**

**User getUserDetailById(long userId) throws Exception;**

**User signUpUser(HashMap<String,String> signupRequest) throws Exception;**

**}**

**package com.foodBox.services.userService;**

**import java.util.HashMap;**

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

**import org.springframework.context.annotation.Bean;**

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

**import org.springframework.stereotype.Service;**

**import com.foodBox.Repository.UserRepository;**

**import com.foodBox.model.User;**

**@Service**

**public class UserServiceImpl implements UserService{**

**@Autowired**

**UserRepository userRepo;**

**@Autowired**

**private BCryptPasswordEncoder encoder;**

**@Override**

**public User findByEmail(String email) throws Exception {**

**return userRepo.findByEmail(email).orElseThrow(()->new Exception("User Not found.."));**

**}**

**@Override**

**public User getUserDetailById(long userId) throws Exception {**

**return userRepo.findById(userId).orElseThrow(()->new Exception("User Not found.."));**

**}**

**@Override**

**public User signUpUser(HashMap<String, String> signupRequest) throws Exception {**

**try {**

**if(userRepo.findByEmail(signupRequest.get("email")).isPresent()) {**

**throw new Exception("User is already registered with Email Id.");**

**}**

**User user = new User();**

**user.setName(signupRequest.get("name"));**

**user.setEmail(signupRequest.get("email"));**

**user.setMobile(signupRequest.get("mobile"));**

**user.setPassword(encoder.encode(signupRequest.get("password")));**

**userRepo.save(user);**

**return user;**

**}catch(Exception e) {**

**throw new Exception(e.getMessage());**

**}**

**}**

**}**