

# ecommerce

## Index of Files

ignore .....	<a href="#">1</a>
EcommerceApplication.java .....	<a href="#">2</a>
component/	
AuthFilter.java .....	<a href="#">3</a>
JwtUtils.java .....	<a href="#">5</a>
configuration/	
ApplicationConfig.java .....	<a href="#">6</a>
SecurityConfig.java .....	<a href="#">7</a>
controller/	
AuthController.java .....	<a href="#">8</a>
PingController.java .....	<a href="#">9</a>
ProductController.java .....	<a href="#">10</a>
UserController.java .....	<a href="#">11</a>
model/	
entity/	
CategoryEntity.java .....	<a href="#">12</a>
ProductEntity.java .....	<a href="#">13</a>
UserEntity.java .....	<a href="#">14</a>
request/	
LoginRequest.java .....	<a href="#">15</a>
RegisterRequest.java .....	<a href="#">16</a>
response/	
BaseResponse.java .....	<a href="#">17</a>
PagingInfo.java .....	<a href="#">18</a>
Pong.java .....	<a href="#">19</a>
repository/	
ProductRepository.java .....	<a href="#">20</a>
UserRepository.java .....	<a href="#">21</a>
service/	
AuthService.java .....	<a href="#">22</a>
CustomUserDetailsService.java .....	<a href="#">23</a>
ProductService.java .....	<a href="#">24</a>
UserService.java .....	<a href="#">25</a>

```
# .pdfignore
```

```
# Location: ..
```

```
...
```

```
1: *.iml
2: .kotlin
3: .gradle
4: **/build/
5: xcuserdata
6: !src/**/build/
7: local.properties
8: .idea
9: .DS_Store
10: captures
11: .externalNativeBuild
12: .cxx
13: *.xcodeproj/*
14: !*.xcodeproj/project.pbxproj
15: !*.xcodeproj/xcshareddata/
16: !*.xcodeproj/project.xcworkspace/
17: !*.xcworkspace/contents.xcworkspacedata
18: **/xcshareddata/WorkspaceSettings.xcsettings
19: .venv
20: .idea
21: code-to-pdf.py
22: code_to_pdf.py
23:
24: .git/
```

```
# EcommerceApplication.java
```

```
# Location: ..
```

```
...
```

```
1: package com.kiatkoding.ecommerce;
2:
3: import org.springframework.boot.SpringApplication;
4: import org.springframework.boot.autoconfigure.SpringBootApplication;
5:
6: @SpringBootApplication
7: public class EcommerceApplication {
8:
9:     public static void main(String[] args) {
10:         SpringApplication.run(EcommerceApplication.class, args);
11:     }
12:
13: }
```

## # AuthFilter.java

# Location: component/

...

```
1: package com.kiatkoding.ecommerce.component;
2:
3: import com.kiatkoding.ecommerce.service.CustomUserDetailsService;
4: import io.jsonwebtoken.Claims;
5: import jakarta.servlet.FilterChain;
6: import jakarta.servlet.ServletException;
7: import jakarta.servlet.http.HttpServletRequest;
8: import jakarta.servlet.http.HttpServletResponse;
9: import lombok.RequiredArgsConstructor;
10: import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
11: import org.springframework.security.core.context.SecurityContextHolder;
12: import org.springframework.security.core.userdetails.UserDetails;
13: import org.springframework.security.web.authentication.WebAuthenticationDetails;
14: import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
15: import org.springframework.stereotype.Component;
16: import org.springframework.web.filter.OncePerRequestFilter;
17:
18: import java.io.IOException;
19:
20: @Component
21: @RequiredArgsConstructor
22: public class AuthFilter extends OncePerRequestFilter {
23:     private final JwtUtils jwtUtils;
24:     private final CustomUserDetailsService userDetailsService;
25:
26:     @Override
27:     protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {
28:         if (request.getServletPath().contains("/api/v1/auth")) {
29:             filterChain.doFilter(request, response);
30:             return;
31:         }
32:
33:         final String authHeader = request.getHeader("Authorization");
34:         final String token;
35:         final String phoneNumber;
36:
37:         if (authHeader == null || !authHeader.startsWith("Bearer ")) {
38:             filterChain.doFilter(request, response);
39:             return;
40:         }
41:
42:         token = authHeader.split(" ")[1].trim();
43:         phoneNumber = jwtUtils.extractClaim(token, Claims::getSubject);
44:     }
}
```

```

45:         if (phoneNumber != null && SecurityContextHolder.getContext().getAuthentication() == null) {
46:             UserDetails userDetails = userDetailsService.loadUserByUsername(phoneNumber);
47:
48:             if (jwtUtils.isValidToken(token, userDetails)) {
49:                 UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(
50:                     userDetails, null, userDetails.getAuthorities()
51:                 );
52:
53:                 WebAuthenticationDetails webAuthenticationDetails = new WebAuthenticationDetailsSource().buildDetails(request);
54:                 authenticationToken.setDetails(webAuthenticationDetails);
55:                 SecurityContextHolder.getContext().setAuthentication(authenticationToken);
56:             }
57:         }
58:
59:         filterChain.doFilter(request, response);
60:     }
61: }

```

## # JwtUtils.java

# Location: component/

...

```
1: package com.kiatkoding.ecommerce.component;
2:
3: import com.kiatkoding.ecommerce.model.entity.UserEntity;
4: import io.jsonwebtoken.Claims;
5: import io.jsonwebtoken.Jwts;
6: import io.jsonwebtoken.io.Decoders;
7: import io.jsonwebtoken.security.Keys;
8: import org.springframework.beans.factory.annotation.Value;
9: import org.springframework.security.core.userdetails.UserDetails;
10: import org.springframework.stereotype.Component;
11:
12: import javax.crypto.SecretKey;
13: import java.util.Date;
14: import java.util.function.Function;
15:
16: @Component
17: public class JwtUtils {
18:
19:     @Value("${jwt.secret}")
20:     private String secret;
21:
22:     private SecretKey secretKey() {
23:         byte[] bytes = Decoders.BASE64URL.decode(secret);
24:         return Keys.hmacShaKeyFor(bytes);
25:     }
26:
27:     private Boolean isTokenExpired(String token) {
28:         Date expiration = extractClaim(token, Claims::getExpiration);
29:         return expiration.before(new Date());
30:     }
31:
32:     private Claims extractAllClaims(String token) {
33:         return Jwts.parser()
34:             .verifyWith(secretKey())
35:             .build()
36:             .parseSignedClaims(token)
37:             .getPayload();
38:     }
39:
40:     public <T> T extractClaim(String token, Function<Claims, T> resolver) {
41:         final Claims claims = extractAllClaims(token);
42:         return resolver.apply(claims);
43:     }
44:
45:     public Boolean isValidToken(String token, UserDetails userDetails) {
46:         final String phoneNumber = extractClaim(token, Claims::getSubject);
47:         return phoneNumber.equals(userDetails.getUsername()) && !isTokenExpired(token);
48:     }
49:
50:     public String generateToken(UserEntity userEntity) {
51:         final long expiration = 1000L * 60 * 60 * 60 * 24;
52:         return Jwts
53:             .builder()
54:             .subject(userEntity.phoneNumber)
55:             .issuedAt(new Date(System.currentTimeMillis()))
56:             .expiration(new Date(System.currentTimeMillis() + expiration))
57:             .signWith(secretKey())
58:             .compact();
59:     }
60: }
```

## # ApplicationConfig.java

# Location: configuration/

...

```
1: package com.kiatkoding.ecommerce.configuration;
2:
3: import com.kiatkoding.ecommerce.service.CustomUserDetailsService;
4: import lombok.RequiredArgsConstructor;
5: import org.springframework.context.annotation.Bean;
6: import org.springframework.context.annotation.Configuration;
7: import org.springframework.security.authentication.AuthenticationManager;
8: import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
9: import org.springframework.security.config.annotation.web.builders.HttpSecurity;
10: import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
11:
12: @Configuration
13: @RequiredArgsConstructor
14: public class ApplicationConfig {
15:
16:     private final CustomUserDetailsService detailsService;
17:
18:     @Bean
19:     public BCryptPasswordEncoder bCryptPasswordEncoder() {
20:         return new BCryptPasswordEncoder();
21:     }
22:
23:     @Bean
24:     public AuthenticationManager authenticationManager(
25:         HttpSecurity httpSecurity,
26:         BCryptPasswordEncoder bCryptPasswordEncoder
27:     ) throws Exception {
28:         AuthenticationManagerBuilder builder = httpSecurity.getSharedObject(AuthenticationManagerBuilder.class);
29:         builder.userDetailsService(detailsService)
30:             .passwordEncoder(bCryptPasswordEncoder);
31:         return builder.build();
32:     }
33:
34: }
```

## # SecurityConfig.java

# Location: configuration/

...

```
1: package com.kiatkoding.ecommerce.configuration;
2:
3: import com.kiatkoding.ecommerce.component.AuthFilter;
4: import lombok.RequiredArgsConstructor;
5: import org.springframework.context.annotation.Bean;
6: import org.springframework.context.annotation.Configuration;
7: import org.springframework.security.config.annotation.web.builders.HttpSecurity;
8: import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
9: import org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;
10: import org.springframework.security.config.http.SessionCreationPolicy;
11: import org.springframework.security.web.SecurityFilterChain;
12: import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
13:
14: @Configuration
15: @EnableWebSecurity
16: @RequiredArgsConstructor
17: public class SecurityConfig {
18:
19:     private static final String[] ALLOWED = {
20:         "/api/v1/auth/login",
21:         "/api/v1/auth/register"
22:     };
23:
24:     private final AuthFilter authFilter;
25:
26:     @Bean
27:     public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {
28:         httpSecurity.csrf(AbstractHttpConfigurer::disable)
29:             .authorizeHttpRequests(registry ->
30:                 registry.requestMatchers(ALLOWED)
31:                     .permitAll()
32:                     .anyRequest()
33:                     .authenticated()
34:             )
35:             .sessionManagement(managementConfigurer ->
36:                 managementConfigurer
37:                     .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
38:             )
39:             .addFilterBefore(authFilter, UsernamePasswordAuthenticationFilter.class);
40:
41:         return httpSecurity.build();
42:     }
43:
44: }
```



## # AuthController.java

# Location: controller/

...

```
1: package com.kiatkoding.ecommerce.controller;
2:
3: import com.kiatkoding.ecommerce.model.request.LoginRequest;
4: import com.kiatkoding.ecommerce.model.request.RegisterRequest;
5: import com.kiatkoding.ecommerce.model.response.BaseResponse;
6: import com.kiatkoding.ecommerce.service.AuthService;
7: import com.kiatkoding.ecommerce.service.UserService;
8: import lombok.AllArgsConstructor;
9: import org.springframework.web.bind.annotation.PostMapping;
10: import org.springframework.web.bind.annotation.RequestBody;
11: import org.springframework.web.bind.annotation.RequestMapping;
12: import org.springframework.web.bind.annotation.RestController;
13:
14: @RestController
15: @RequestMapping("/api/v1/auth")
16: @AllArgsConstructor
17: public class AuthController {
18:
19:     private final AuthService authService;
20:
21:     @PostMapping("/register")
22:     public BaseResponse postRegister(
23:         @RequestBody RegisterRequest registerRequest
24:     ) {
25:         Boolean result = authService.register(registerRequest);
26:         return new BaseResponse(result, "Success", result);
27:     }
28:
29:     @PostMapping("/login")
30:     public BaseResponse postLogin(
31:         @RequestBody LoginRequest loginRequest
32:     ) {
33:         String token = authService.login(loginRequest);
34:         return new BaseResponse(true, "Success", token);
35:     }
36: }
```

## # PingController.java

# Location: controller/

...

```
1: package com.kiatkoding.ecommerce.controller;
2:
3: import com.kiatkoding.ecommerce.model.response.BaseResponse;
4: import com.kiatkoding.ecommerce.model.response.Pong;
5: import lombok.AllArgsConstructor;
6: import org.springframework.jdbc.core.JdbcTemplate;
7: import org.springframework.web.bind.annotation.GetMapping;
8: import org.springframework.web.bind.annotation.RequestMapping;
9: import org.springframework.web.bind.annotation.RestController;
10:
11: @RestController
12: @RequestMapping("/api/v1/ping")
13: @AllArgsConstructor
14: public class PingController {
15:
16:     private final JdbcTemplate jdbcTemplate;
17:
18:     @GetMapping
19:     public BaseResponse ping() {
20:         BaseResponse response = new BaseResponse();
21:
22:         try {
23:
24:             jdbcTemplate.execute("SELECT 1");
25:             Pong pong = new Pong();
26:
27:             pong.setMessage("Success");
28:             pong.setDatabaseConnection(true);
29:
30:             response.setStatus(true);
31:             response.setMessage("Database connection success");
32:             response.setData(pong);
33:
34:         } catch (Exception e) {
35:             Pong pong = new Pong();
36:
37:             pong.setMessage("Failed");
38:             pong.setErrorMessage(e.getLocalizedMessage());
39:             pong.setDatabaseConnection(false);
40:
41:             response.setStatus(false);
42:             response.setMessage("Database connection failed");
43:             response.setData(pong);
44:         }
45:
46:
47:         return response;
48:     }
49: }
```

## # ProductController.java

# Location: controller/

...

```
1: package com.kiatkoding.ecommerce.controller;
2:
3:
4: import com.kiatkoding.ecommerce.service.ProductService;
5: import com.kiatkoding.ecommerce.model.entity.ProductEntity;
6: import com.kiatkoding.ecommerce.model.response.BaseResponse;
7: import com.kiatkoding.ecommerce.model.response.PagingInfo;
8: import lombok.AllArgsConstructor;
9: import org.springframework.web.bind.annotation.GetMapping;
10: import org.springframework.web.bind.annotation.RequestMapping;
11: import org.springframework.web.bind.annotation.RequestParam;
12: import org.springframework.web.bind.annotation.RestController;
13:
14: @RestController
15: @RequestMapping("/api/v1/products")
16: @AllArgsConstructor
17: public class ProductController {
18:
19:     private final ProductService productService;
20:
21:     @GetMapping
22:     public BaseResponse getProducts(
23:         @RequestParam(defaultValue = "1") Integer page,
24:         @RequestParam(defaultValue = "10") Integer size,
25:         @RequestParam(required = false) String q,
26:         @RequestParam(required = false) Integer categoryId
27:     ) {
28:
29:         String key = (q == null ? "NQ" : "Q") + (categoryId == null ? "NC" : "C");
30:
31:         PagingInfo<ProductEntity> products = switch (key) {
32:             case "NQC" -> productService.getProducts(page, size, categoryId);
33:             case "QNC" -> productService.getProducts(page, size, q);
34:             case "QC" -> productService.getProducts(page, size, q, categoryId);
35:             default -> productService.getProducts(page, size);
36:         };
37:
38:         BaseResponse response = new BaseResponse();
39:         response.setStatus(true);
40:         response.setMessage("Success");
41:         response.setData(products);
42:         return response;
43:     }
44:
45: }
```

## # UserController.java

# Location: controller/

...

```
1: package com.kiatkoding.ecommerce.controller;
2:
3: import com.kiatkoding.ecommerce.model.entity.UserEntity;
4: import com.kiatkoding.ecommerce.model.request.RegisterRequest;
5: import com.kiatkoding.ecommerce.model.response.BaseResponse;
6: import com.kiatkoding.ecommerce.service.UserService;
7: import lombok.AllArgsConstructor;
8: import org.springframework.web.bind.annotation.*;
9:
10: @RestController
11: @RequestMapping("/api/v1/user")
12: @AllArgsConstructor
13: public class UserController {
14:
15:     private final UserService userService;
16:
17:     @GetMapping
18:     public BaseResponse getUser() {
19:         UserEntity userEntity = userService.userEntity();
20:         return new BaseResponse(true, "Success", userEntity);
21:     }
22: }
```

## # CategoryEntity.java

# Location: model/entity/

...

```
1: package com.kiatkoding.ecommerce.model.entity;
2:
3:
4: import jakarta.persistence.*;
5: import lombok.AllArgsConstructor;
6: import lombok.Data;
7: import lombok.NoArgsConstructor;
8:
9: @Table(name = "category")
10: @Entity
11: @Data
12: @NoArgsConstructor
13: @AllArgsConstructor
14: public class CategoryEntity {
15:
16:     @Id
17:     @GeneratedValue(strategy = GenerationType.IDENTITY)
18:     public Integer id;
19:
20:     @Column(nullable = false)
21:     public String name;
22: }
```

## # ProductEntity.java

# Location: model/entity/

...

```
1: package com.kiatkoding.ecommerce.model.entity;
2:
3: import jakarta.persistence.*;
4: import lombok.AllArgsConstructor;
5: import lombok.Data;
6: import lombok.NoArgsConstructor;
7:
8: @Table(name = "product")
9: @Entity
10: @Data
11: @NoArgsConstructor
12: @AllArgsConstructor
13: public class ProductEntity {
14:
15:     @Id
16:     @GeneratedValue(strategy = GenerationType.IDENTITY)
17:     public Integer id;
18:
19:     @Column(nullable = false)
20:     public String name;
21:
22:     @Column(nullable = false)
23:     public String image;
24:
25:     @Column(nullable = false)
26:     public Double price;
27:
28:     @Column(nullable = true)
29:     public Double rating;
30:
31:     @Column(nullable = false)
32:     public String description;
33:
34:     @ManyToOne(fetch = FetchType.EAGER)
35:     @JoinColumn(name = "category_id", nullable = false)
36:     public CategoryEntity category;
37: }
```

## # UserEntity.java

# Location: model/entity/

...

```
1: package com.kiatkoding.ecommerce.model.entity;
2:
3: import jakarta.persistence.*;
4: import lombok.AllArgsConstructor;
5: import lombok.Data;
6: import lombok.NoArgsConstructor;
7:
8: @Table(name = "user", schema = "public")
9: @Entity
10: @Data
11: @NoArgsConstructor
12: @AllArgsConstructor
13: public class UserEntity {
14:
15:     @Id
16:     @GeneratedValue(strategy = GenerationType.IDENTITY)
17:     public Integer id;
18:
19:     @Column(nullable = false)
20:     public String name;
21:
22:     @Column(nullable = false, name = "phone_number", unique = true)
23:     public String phoneNumber;
24:
25:     @Column(nullable = false)
26:     public String password;
27: }
```

## # LoginRequest.java

# Location: model/request/

...

```
1: package com.kiatkoding.ecommerce.model.request;
2:
3:
4: import com.fasterxml.jackson.databind.PropertyNamingStrategies;
5: import com.fasterxml.jackson.databind.annotation.JsonNaming;
6: import jakarta.validation.constraints.NotBlank;
7: import lombok.Data;
8:
9: @Data
10: @JsonNaming(PropertyNamingStrategies.SnakeCaseStrategy.class)
11: public class LoginRequest {
12:
13:     @NotBlank(message = "Phone number is required!")
14:     private String phoneNumber;
15:
16:     @NotBlank(message = "Password is required!")
17:     private String password;
18: }
```



## # RegisterRequest.java

# Location: model/request/

...

```
1: package com.kiatkoding.ecommerce.model.request;
2:
3: import com.fasterxml.jackson.databind.PropertyNamingStrategies;
4: import com.fasterxml.jackson.databind.annotation.JsonNaming;
5: import jakarta.validation.constraints.NotBlank;
6: import lombok.Data;
7:
8: @Data
9: @JsonNaming(PropertyNamingStrategies.SnakeCaseStrategy.class)
10: public class RegisterRequest {
11:
12:     @NotBlank(message = "Phone number is required!")
13:     private String phoneNumber;
14:
15:     @NotBlank(message = "Name is required!")
16:     private String name;
17:
18:     @NotBlank(message = "Password is required!")
19:     private String password;
20: }
```

## # BaseResponse.java

# Location: model/response/

...

```
1: package com.kiatkoding.ecommerce.model.response;
2:
3: import lombok.*;
4:
5: @Data
6: @AllArgsConstructor
7: @NoArgsConstructor
8: public class BaseResponse {
9:
10:     public Boolean status;
11:     public String message;
12:     public Object data;
13: }
```

## # PagingInfo.java

# Location: model/response/

...

```
1: package com.kiatkoding.ecommerce.model.response;
2:
3: import lombok.Data;
4: import org.springframework.data.domain.Page;
5:
6: import java.util.List;
7:
8: @Data
9: public class PagingInfo <T> {
10:     public Integer currentPage;
11:     public Integer nextPage;
12:     public Integer prevPage;
13:     public Integer totalPage;
14:     public long totalItem;
15:     public List<T> content;
16:
17:     public static <T> PagingInfo<T> convertFromPage(Page<T> page) {
18:         PagingInfo<T> pagingInfo = new PagingInfo<>();
19:
20:         pagingInfo.totalPage = page.getTotalPages();
21:         pagingInfo.totalItem = page.getTotalElements();
22:         pagingInfo.currentPage = page.getNumber() + 1;
23:
24:         if (page.hasNext()) {
25:             pagingInfo.nextPage = (page.getNumber() + 1) + 1;
26:         }
27:
28:         if (page.hasPrevious()) {
29:             if (pagingInfo.currentPage > pagingInfo.totalPage) {
30:                 pagingInfo.prevPage = pagingInfo.totalPage;
31:             } else {
32:                 pagingInfo.prevPage = pagingInfo.currentPage - 1;
33:             }
34:         }
35:
36:         pagingInfo.content = page.getContent();
37:         return pagingInfo;
38:     }
39: }
```

# Pong.java

# Location: model/response/

...

```
1: package com.kiatkoding.ecommerce.model.response;
2:
3: import lombok.Data;
4:
5: @Data
6: public class Pong {
7:
8:     public String message;
9:     public String errorMessage;
10:    public Boolean databaseConnection;
11: }
```

## # ProductRepository.java

# Location: repository/

...

```
1: package com.kiatkoding.ecommerce.repository;
2:
3: import com.kiatkoding.ecommerce.model.entity.ProductEntity;
4: import org.springframework.data.domain.Page;
5: import org.springframework.data.domain.Pageable;
6: import org.springframework.data.jpa.repository.JpaRepository;
7: import org.springframework.data.jpa.repository.Query;
8: import org.springframework.stereotype.Repository;
9:
10: @Repository
11: public interface ProductRepository extends JpaRepository<ProductEntity, Integer> {
12:
13:
14:     Page<ProductEntity> findAll(Pageable pageable);
15:
16:     // naming method
17:     Page<ProductEntity> findByNameContainsIgnoreCaseAndDescriptionContainsIgnoreCase(String name, String description);
18:
19:     // query method
20:     @Query("
21:         SELECT product FROM ProductEntity product WHERE
22:         LOWER(product.name) LIKE LOWER(CONCAT('%', :query, '%')) OR
23:         LOWER(product.description) LIKE LOWER(CONCAT('%', :query, '%'))
24:     ")
25:     Page<ProductEntity> filter(String query, Pageable pageable);
26:
27:     @Query("
28:         SELECT product FROM ProductEntity product WHERE
29:         (LOWER(product.name) LIKE LOWER(CONCAT('%', :query, '%')) OR
30:         LOWER(product.description) LIKE LOWER(CONCAT('%', :query, '%'))) AND
31:         product.category.id = :categoryId
32:     ")
33:     Page<ProductEntity> filter(String query, Integer categoryId, Pageable pageable);
34:
35:     @Query("
36:         SELECT product FROM ProductEntity product WHERE
37:         product.category.id = :categoryId
38:     ")
39:     Page<ProductEntity> filter(Integer categoryId, Pageable pageable);
40: }
```

## # UserRepository.java

# Location: repository/

...

```
1: package com.kiatkoding.ecommerce.repository;
2:
3: import com.kiatkoding.ecommerce.model.entity.UserEntity;
4: import org.springframework.data.jpa.repository.JpaRepository;
5: import org.springframework.stereotype.Repository;
6:
7: import java.util.Optional;
8:
9: @Repository
10: public interface UserRepository extends JpaRepository<UserEntity, Integer> {
11:
12:     Optional<UserEntity> findByPhoneNumber(String phoneNumber);
13: }
```

## # AuthService.java

# Location: service/

...

```
1: package com.kiatkoding.ecommerce.service;
2:
3: import com.kiatkoding.ecommerce.component.JwtUtils;
4: import com.kiatkoding.ecommerce.model.entity.UserEntity;
5: import com.kiatkoding.ecommerce.model.request.LoginRequest;
6: import com.kiatkoding.ecommerce.model.request.RegisterRequest;
7: import com.kiatkoding.ecommerce.repository.UserRepository;
8: import lombok.RequiredArgsConstructor;
9: import org.springframework.security.authentication.AuthenticationManager;
10: import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
11: import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
12: import org.springframework.stereotype.Service;
13:
14: import java.util.Optional;
15:
16:
17: @Service
18: @RequiredArgsConstructor
19: public class AuthService {
20:
21:     private final UserRepository userRepository;
22:     private final BCryptPasswordEncoder bCryptPasswordEncoder;
23:     private final AuthenticationManager authenticationManager;
24:     private final JwtUtils jwtUtils;
25:
26:     public Boolean register(RegisterRequest registerRequest) {
27:         Optional<UserEntity> userEntity = userRepository.findByPhoneNumber(registerRequest.getPhoneNumber());
28:
29:         if (userEntity.isPresent()) {
30:             // save
31:             return false;
32:         } else {
33:             // existing
34:             try {
35:                 UserEntity newUserEntity = new UserEntity();
36:                 newUserEntity.name = registerRequest.getName();
37:                 newUserEntity.password = bCryptPasswordEncoder.encode(registerRequest.getPassword());
38:                 newUserEntity.phoneNumber = registerRequest.getPhoneNumber();
39:
40:                 userRepository.save(newUserEntity);
41:                 return true;
42:             } catch (Exception e) {
43:                 return false;
44:             }
45:         }
46:     }
47:
48:     public String login(LoginRequest loginRequest) {
49:         UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(
50:             loginRequest.getPhoneNumber(), loginRequest.getPassword()
51:         );
52:
53:         authenticationManager.authenticate(authenticationToken);
54:
55:         UserEntity userEntity = userRepository.findByPhoneNumber(loginRequest.getPhoneNumber())
56:             .orElseThrow();
57:
58:         return jwtUtils.generateToken(userEntity);
59:     }
60: }
```

## # CustomUserDetailsService.java

# Location: service/

...

```
1: package com.kiatkoding.ecommerce.service;
2:
3: import com.kiatkoding.ecommerce.model.entity.UserEntity;
4: import com.kiatkoding.ecommerce.repository.UserRepository;
5: import lombok.RequiredArgsConstructor;
6: import org.springframework.security.core.userdetails.User;
7: import org.springframework.security.core.userdetails.UserDetails;
8: import org.springframework.security.core.userdetails.UserDetailsService;
9: import org.springframework.security.core.userdetails.UsernameNotFoundException;
10: import org.springframework.stereotype.Service;
11:
12: @Service
13: @RequiredArgsConstructor
14: public class CustomUserDetailsService implements UserDetailsService {
15:     private final UserRepository userRepository;
16:
17:     @Override
18:     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
19:         return userRepository
20:             .findByPhoneNumber(username)
21:             .map(this::mapFromEntity)
22:             .orElseThrow();
23:     }
24:
25:     private UserDetails mapFromEntity(UserEntity userEntity) {
26:         return User.builder()
27:             .username(userEntity.phoneNumber)
28:             .password(userEntity.password)
29:             .build();
30:     }
31: }
```



## # ProductService.java

# Location: service/

...

```
1: package com.kiatkoding.ecommerce.service;
2:
3: import com.kiatkoding.ecommerce.model.entity.ProductEntity;
4: import com.kiatkoding.ecommerce.model.response.PagingInfo;
5: import com.kiatkoding.ecommerce.repository.ProductRepository;
6: import lombok.RequiredArgsConstructor;
7: import org.springframework.data.domain.Page;
8: import org.springframework.data.domain.PageRequest;
9: import org.springframework.stereotype.Service;
10:
11: @Service
12: @RequiredArgsConstructor
13: public class ProductService {
14:
15:     private final ProductRepository productRepository;
16:
17:     public PagingInfo<ProductEntity> getProducts(
18:         Integer pageNumber,
19:         Integer pageSize
20:     ) {
21:         PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
22:         Page<ProductEntity> products = productRepository.findAll(pageRequest);
23:
24:         return PagingInfo.convertFromPage(products);
25:     }
26:
27:     public PagingInfo<ProductEntity> getProducts(
28:         Integer pageNumber,
29:         Integer pageSize,
30:         String query
31:     ) {
32:         PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
33:         Page<ProductEntity> products = productRepository
34:             .filter(query, pageRequest);
35:
36:         return PagingInfo.convertFromPage(products);
37:     }
38:
39:     public PagingInfo<ProductEntity> getProducts(
40:         Integer pageNumber,
41:         Integer pageSize,
42:         String query,
43:         Integer categoryId
44:     ) {
45:         PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
46:         Page<ProductEntity> products = productRepository
47:             .filter(query, categoryId, pageRequest);
48:
49:         return PagingInfo.convertFromPage(products);
50:     }
51:
52:     public PagingInfo<ProductEntity> getProducts(
53:         Integer pageNumber,
54:         Integer pageSize,
55:         Integer categoryId
56:     ) {
57:         PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
58:         Page<ProductEntity> products = productRepository
59:             .filter(categoryId, pageRequest);
60:
61:         return PagingInfo.convertFromPage(products);
62:     }
63: }
```

## # UserService.java

# Location: service/

...

```
1: package com.kiatkoding.ecommerce.service;
2:
3: import com.kiatkoding.ecommerce.model.entity.UserEntity;
4: import com.kiatkoding.ecommerce.model.request.RegisterRequest;
5: import com.kiatkoding.ecommerce.repository.UserRepository;
6: import lombok.RequiredArgsConstructor;
7: import org.springframework.security.core.context.SecurityContextHolder;
8: import org.springframework.security.core.userdetails.UserDetails;
9: import org.springframework.stereotype.Service;
10:
11: import java.util.Optional;
12:
13: @Service
14: @RequiredArgsConstructor
15: public class UserService {
16:
17:     private final UserRepository userRepository;
18:
19:     public UserEntity userEntity() {
20:         UserDetails userDetails = (UserDetails) SecurityContextHolder.getContext()
21:             .getAuthentication()
22:             .getPrincipal();
23:
24:         String phoneNumber = userDetails.getUsername();
25:         return userRepository.findByPhoneNumber(phoneNumber)
26:             .orElseThrow();
27:     }
28:
29:
30: }
```