ecommerce

Index of Files

.pdfignore

Location: ..

...

```
*.iml
.kotlin
.gradle
**/build/
xcuserdata
!src/**/build/
local.properties
.idea
.DS_Store
captures
.externalNativeBuild
.cxx
*.xcodeproj/*
!*.xcodeproj/project.pbxproj
!*.xcodeproj/xcshareddata/
!*.xcodeproj/project.xcworkspace/
!*.xcworkspace/contents.xcworkspacedata
**/xcshareddata/WorkspaceSettings.xcsettings
.venv
.idea
code-to-pdf.py
code_to_pdf.py

.git/
```

EcommerceApplication.java

Location: ..

...

```java
package com.kiatkoding.ecommerce;

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

@SpringBootApplication
public class EcommerceApplication {

	public static void main(String[] args) {
		SpringApplication.run(EcommerceApplication.class, args);
	}

}
```

AuthFilter.java

Location: component/

...

```java
package com.kiatkoding.ecommerce.component;

import com.kiatkoding.ecommerce.service.CustomUserDetailsService;
import io.jsonwebtoken.Claims;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;
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.WebAuthenticationDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component
@RequiredArgsConstructor
public class AuthFilter extends OncePerRequestFilter {
    private final JwtUtils jwtUtils;
    private final CustomUserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {
        if (request.getServletPath().contains("/api/v1/auth")) {
            filterChain.doFilter(request, response);
            return;
        }

        final String authHeader = request.getHeader("Authorization");
        final String token;
        final String phoneNumber;

        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            filterChain.doFilter(request, response);
            return;
        }

        token = authHeader.split(" ")[1].trim();
        phoneNumber = jwtUtils.extractClaim(token, Claims::getSubject);

```

```
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/

...

```java
package com.kiatkoding.ecommerce.component;

import com.kiatkoding.ecommerce.model.entity.UserEntity;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.io.Decoders;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import java.util.Date;
import java.util.function.Function;

@Component
public class JwtUtils {

    @Value("${jwt.secret}")
    private String secret;

    private SecretKey secretKey() {
        byte[] bytes = Decoders.BASE64URL.decode(secret);
        return Keys.hmacShaKeyFor(bytes);
    }

    private Boolean isTokenExpired(String token) {
        Date expiration = extractClaim(token, Claims::getExpiration);
        return expiration.before(new Date());
    }

    private Claims extractAllClaims(String token) {
        return Jwts.parser()
                .verifyWith(secretKey())
                .build()
                .parseSignedClaims(token)
                .getPayload();
    }

    public <T> T extractClaim(String token, Function<Claims, T> resolver) {
        final Claims claims = extractAllClaims(token);
        return resolver.apply(claims);
    }

    public Boolean isValidToken(String token, UserDetails userDetails) {
        final String phoneNumber = extractClaim(token, Claims::getSubject);
        return phoneNumber.equals(userDetails.getUsername()) && !isTokenExpired(token);
    }

    public String generateToken(UserEntity userEntity) {
        final long expiration = 1000L * 60 * 60 * 60 * 24;
        return Jwts
                .builder()
                .subject(userEntity.phoneNumber)
                .issuedAt(new Date(System.currentTimeMillis()))
                .expiration(new Date(System.currentTimeMillis() + expiration))
                .signWith(secretKey())
                .compact();
    }
}
```

ApplicationConfig.java

Location: configuration/

...

```java
package com.kiatkoding.ecommerce.configuration;

import com.kiatkoding.ecommerce.service.CustomUserDetailsService;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
@RequiredArgsConstructor
public class ApplicationConfig {

    private final CustomUserDetailsService detailsService;

    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public AuthenticationManager authenticationManager(
            HttpSecurity httpSecurity,
            BCryptPasswordEncoder bCryptPasswordEncoder
    ) throws Exception {
        AuthenticationManagerBuilder builder = httpSecurity.getSharedObject(AuthenticationManagerBuilder.class);
        builder.userDetailsService(detailsService)
                .passwordEncoder(bCryptPasswordEncoder);
        return builder.build();
    }

}
```

SecurityConfig.java

Location: configuration/

...

```java
package com.kiatkoding.ecommerce.configuration;

import com.kiatkoding.ecommerce.component.AuthFilter;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
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.configurers.AbstractHttpConfigurer;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private static final String[] ALLOWED = {
            "/api/v1/auth/login",
            "/api/v1/auth/register"
    };

    private final AuthFilter authFilter;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(registry ->
                        registry.requestMatchers(ALLOWED)
                                .permitAll()
                                .anyRequest()
                                .authenticated()
                )
                .sessionManagement(managementConfigurer ->
                        managementConfigurer
                                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                )
                .addFilterBefore(authFilter, UsernamePasswordAuthenticationFilter.class);

        return httpSecurity.build();
    }

}
```

AuthController.java

Location: controller/

...

```java
package com.kiatkoding.ecommerce.controller;

import com.kiatkoding.ecommerce.model.request.LoginRequest;
import com.kiatkoding.ecommerce.model.request.RegisterRequest;
import com.kiatkoding.ecommerce.model.response.BaseResponse;
import com.kiatkoding.ecommerce.service.AuthService;
import com.kiatkoding.ecommerce.service.UserService;
import lombok.AllArgsConstructor;
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;

@RestController
@RequestMapping("/api/v1/auth")
@AllArgsConstructor
public class AuthController {

    private final AuthService authService;

    @PostMapping("/register")
    public BaseResponse postRegister(
            @RequestBody RegisterRequest registerRequest
    ) {
        Boolean result = authService.register(registerRequest);
        return new BaseResponse(result, "Success", result);
    }

    @PostMapping("/login")
    public BaseResponse postLogin(
            @RequestBody LoginRequest loginRequest
    ) {
        String token = authService.login(loginRequest);
        return new BaseResponse(true, "Success", token);
    }
}
```

PingController.java

Location: controller/

...

```java
package com.kiatkoding.ecommerce.controller;

import com.kiatkoding.ecommerce.model.response.BaseResponse;
import com.kiatkoding.ecommerce.model.response.Pong;
import lombok.AllArgsConstructor;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/v1/ping")
@AllArgsConstructor
public class PingController {

    private final JdbcTemplate jdbcTemplate;

    @GetMapping
    public BaseResponse ping() {
        BaseResponse response = new BaseResponse();

        try {

            jdbcTemplate.execute("SELECT 1");
            Pong pong = new Pong();

            pong.setMessage("Success");
            pong.setDatabaseConnection(true);

            response.setStatus(true);
            response.setMessage("Database connection success");
            response.setData(pong);

        } catch (Exception e) {
            Pong pong = new Pong();

            pong.setMessage("Failed");
            pong.setErrorMessage(e.getLocalizedMessage());
            pong.setDatabaseConnection(false);

            response.setStatus(false);
            response.setMessage("Database connection failed");
            response.setData(pong);
        }


        return response;
    }
}
```

ProductController.java

Location: controller/

...

```java
package com.kiatkoding.ecommerce.controller;


import com.kiatkoding.ecommerce.service.ProductService;
import com.kiatkoding.ecommerce.model.entity.ProductEntity;
import com.kiatkoding.ecommerce.model.response.BaseResponse;
import com.kiatkoding.ecommerce.model.response.PagingInfo;
import lombok.AllArgsConstructor;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/v1/products")
@AllArgsConstructor
public class ProductController {

    private final ProductService productService;

    @GetMapping
    public BaseResponse getProducts(
            @RequestParam(defaultValue = "1") Integer page,
            @RequestParam(defaultValue = "10") Integer size,
            @RequestParam(required = false) String q,
            @RequestParam(required = false) Integer categoryId
    ) {

        String key = (q == null ? "NQ" : "Q") + (categoryId == null ? "NC" : "C");

        PagingInfo<ProductEntity> products = switch (key) {
            case "NQC" -> productService.getProducts(page, size, categoryId);
            case "QNC" -> productService.getProducts(page, size, q);
            case "QC" -> productService.getProducts(page, size, q, categoryId);
            default -> productService.getProducts(page, size);
        };

        BaseResponse response = new BaseResponse();
        response.setStatus(true);
        response.setMessage("Success");
        response.setData(products);
        return response;
    }

}
```

UserController.java

Location: controller/

...

```java
package com.kiatkoding.ecommerce.controller;

import com.kiatkoding.ecommerce.model.entity.UserEntity;
import com.kiatkoding.ecommerce.model.request.RegisterRequest;
import com.kiatkoding.ecommerce.model.response.BaseResponse;
import com.kiatkoding.ecommerce.service.UserService;
import lombok.AllArgsConstructor;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/v1/user")
@AllArgsConstructor
public class UserController {

    private final UserService userService;

    @GetMapping
    public BaseResponse getUser() {
        UserEntity userEntity = userService.userEntity();
        return new BaseResponse(true, "Success", userEntity);
    }
}
```

CategoryEntity.java

Location: model/entity/

...

```java
package com.kiatkoding.ecommerce.model.entity;


import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Table(name = "category")
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class CategoryEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Integer id;

    @Column(nullable = false)
    public String name;
}
```

ProductEntity.java

Location: model/entity/

...

```java
package com.kiatkoding.ecommerce.model.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Table(name = "product")
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ProductEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Integer id;

    @Column(nullable = false)
    public String name;

    @Column(nullable = false)
    public String image;

    @Column(nullable = false)
    public Double price;

    @Column(nullable = true)
    public Double rating;

    @Column(nullable = false)
    public String description;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "category_id", nullable = false)
    public CategoryEntity category;
}
```

UserEntity.java

Location: model/entity/

...

```java
package com.kiatkoding.ecommerce.model.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Table(name = "user", schema = "public")
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Integer id;

    @Column(nullable = false)
    public String name;

    @Column(nullable = false, name = "phone_number", unique = true)
    public String phoneNumber;

    @Column(nullable = false)
    public String password;
}
```

LoginRequest.java

Location: model/request/

...

```java
package com.kiatkoding.ecommerce.model.request;


import com.fasterxml.jackson.databind.PropertyNamingStrategies;
import com.fasterxml.jackson.databind.annotation.JsonNaming;
import jakarta.validation.constraints.NotBlank;
import lombok.Data;

@Data
@JsonNaming(PropertyNamingStrategies.SnakeCaseStrategy.class)
public class LoginRequest {

    @NotBlank(message = "Phone number is required!")
    private String phoneNumber;

    @NotBlank(message = "Password is required!")
    private String password;
}
```

RegisterRequest.java

Location: model/request/

...

```java
package com.kiatkoding.ecommerce.model.request;

import com.fasterxml.jackson.databind.PropertyNamingStrategies;
import com.fasterxml.jackson.databind.annotation.JsonNaming;
import jakarta.validation.constraints.NotBlank;
import lombok.Data;

@Data
@JsonNaming(PropertyNamingStrategies.SnakeCaseStrategy.class)
public class RegisterRequest {

    @NotBlank(message = "Phone number is required!")
    private String phoneNumber;

    @NotBlank(message = "Name is required!")
    private String name;

    @NotBlank(message = "Password is required!")
    private String password;
}
```

BaseResponse.java

Location: model/response/

...

```java
package com.kiatkoding.ecommerce.model.response;

import lombok.*;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class BaseResponse {

    public Boolean status;
    public String message;
    public Object data;
}
```

PagingInfo.java

Location: model/response/

...

```java
package com.kiatkoding.ecommerce.model.response;

import lombok.Data;
import org.springframework.data.domain.Page;

import java.util.List;

@Data
public class PagingInfo <T> {
    public Integer currentPage;
    public Integer nextPage;
    public Integer prevPage;
    public Integer totalPage;
    public long totalItem;
    public List<T> content;

    public static <T> PagingInfo<T> convertFromPage(Page<T> page) {
        PagingInfo<T> pagingInfo = new PagingInfo<>();

        pagingInfo.totalPage = page.getTotalPages();
        pagingInfo.totalItem = page.getTotalElements();
        pagingInfo.currentPage = page.getNumber() + 1;

        if (page.hasNext()) {
            pagingInfo.nextPage = (page.getNumber() + 1) + 1;
        }

        if (page.hasPrevious()) {
            if (pagingInfo.currentPage > pagingInfo.totalPage) {
                pagingInfo.prevPage = pagingInfo.totalPage;
            } else  {
                pagingInfo.prevPage = pagingInfo.currentPage - 1;
            }
        }

        pagingInfo.content = page.getContent();
        return pagingInfo;
    }
}
```

Pong.java

Location: model/response/

...

```java
package com.kiatkoding.ecommerce.model.response;

import lombok.Data;

@Data
public class Pong {

    public String message;
    public String errorMessage;
    public Boolean databaseConnection;
}
```

ProductRepository.java

Location: repository/

...

```java
package com.kiatkoding.ecommerce.repository;

import com.kiatkoding.ecommerce.model.entity.ProductEntity;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

@Repository
public interface ProductRepository extends JpaRepository<ProductEntity, Integer> {


    Page<ProductEntity> findAll(Pageable pageable);

    // naming method
    Page<ProductEntity> findByNameContainsIgnoreCaseAndDescriptionContainsIgnoreCase(String name, String descriptio

    // query method
    @Query("""
            SELECT product FROM ProductEntity product WHERE
            LOWER(product.name) LIKE LOWER(CONCAT('%', :query, '%')) OR
            LOWER(product.description) LIKE LOWER(CONCAT('%', :query, '%'))
            """)
    Page<ProductEntity> filter(String query, Pageable pageable);

    @Query("""
            SELECT product FROM ProductEntity product WHERE
            (LOWER(product.name) LIKE LOWER(CONCAT('%', :query, '%')) OR
            LOWER(product.description) LIKE LOWER(CONCAT('%', :query, '%'))) AND
            product.category.id = :categoryId
            """)
    Page<ProductEntity> filter(String query, Integer categoryId, Pageable pageable);

    @Query("""
            SELECT product FROM ProductEntity product WHERE
            product.category.id = :categoryId
            """)
    Page<ProductEntity> filter(Integer categoryId, Pageable pageable);
}
```

UserRepository.java

Location: repository/

...

```java
package com.kiatkoding.ecommerce.repository;

import com.kiatkoding.ecommerce.model.entity.UserEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<UserEntity, Integer> {

    Optional<UserEntity> findByPhoneNumber(String phoneNumber);
}
```

AuthService.java

Location: service/

...

```java
package com.kiatkoding.ecommerce.service;

import com.kiatkoding.ecommerce.component.JwtUtils;
import com.kiatkoding.ecommerce.model.entity.UserEntity;
import com.kiatkoding.ecommerce.model.request.LoginRequest;
import com.kiatkoding.ecommerce.model.request.RegisterRequest;
import com.kiatkoding.ecommerce.repository.UserRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Optional;


@Service
@RequiredArgsConstructor
public class AuthService {

    private final UserRepository userRepository;
    private final BCryptPasswordEncoder bCryptPasswordEncoder;
    private final AuthenticationManager authenticationManager;
    private final JwtUtils jwtUtils;

    public Boolean register(RegisterRequest registerRequest) {
        Optional<UserEntity> userEntity = userRepository.findByPhoneNumber(registerRequest.getPhoneNumber());

        if (userEntity.isPresent()) {
            // save
            return false;
        } else {
            // existing
            try {
                UserEntity newUserEntity = new UserEntity();
                newUserEntity.name = registerRequest.getName();
                newUserEntity.password = bCryptPasswordEncoder.encode(registerRequest.getPassword());
                newUserEntity.phoneNumber = registerRequest.getPhoneNumber();

                userRepository.save(newUserEntity);
                return true;
            } catch (Exception e) {
                return false;
            }
        }
    }

    public String login(LoginRequest loginRequest) {
        UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(
                loginRequest.getPhoneNumber(), loginRequest.getPassword()
        );

        authenticationManager.authenticate(authenticationToken);

        UserEntity userEntity = userRepository.findByPhoneNumber(loginRequest.getPhoneNumber())
                .orElseThrow();

        return jwtUtils.generateToken(userEntity);
    }
}
```

CustomUserDetailsService.java

Location: service/

...

```java
package com.kiatkoding.ecommerce.service;

import com.kiatkoding.ecommerce.model.entity.UserEntity;
import com.kiatkoding.ecommerce.repository.UserRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {
    private final UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return userRepository
                .findByPhoneNumber(username)
                .map(this::mapFromEntity)
                .orElseThrow();
    }

    private UserDetails mapFromEntity(UserEntity userEntity) {
        return User.builder()
                .username(userEntity.phoneNumber)
                .password(userEntity.password)
                .build();
    }
}
```

ProductService.java

Location: service/

...

```java
package com.kiatkoding.ecommerce.service;

import com.kiatkoding.ecommerce.model.entity.ProductEntity;
import com.kiatkoding.ecommerce.model.response.PagingInfo;
import com.kiatkoding.ecommerce.repository.ProductRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class ProductService {

    private final ProductRepository productRepository;

    public PagingInfo<ProductEntity> getProducts(
            Integer pageNumber,
            Integer pageSize
    ) {
        PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
        Page<ProductEntity> products = productRepository.findAll(pageRequest);

        return PagingInfo.convertFromPage(products);
    }

    public PagingInfo<ProductEntity> getProducts(
            Integer pageNumber,
            Integer pageSize,
            String query
    ) {
        PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
        Page<ProductEntity> products = productRepository
                .filter(query, pageRequest);

        return PagingInfo.convertFromPage(products);
    }

    public PagingInfo<ProductEntity> getProducts(
            Integer pageNumber,
            Integer pageSize,
            String query,
            Integer categoryId
    ) {
        PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
        Page<ProductEntity> products = productRepository
                .filter(query, categoryId, pageRequest);

        return PagingInfo.convertFromPage(products);
    }

    public PagingInfo<ProductEntity> getProducts(
            Integer pageNumber,
            Integer pageSize,
            Integer categoryId
    ) {
        PageRequest pageRequest = PageRequest.of(pageNumber-1, pageSize);
        Page<ProductEntity> products = productRepository
                .filter(categoryId, pageRequest);

        return PagingInfo.convertFromPage(products);
    }
}
```

UserService.java

Location: service/

...

```java
package com.kiatkoding.ecommerce.service;

import com.kiatkoding.ecommerce.model.entity.UserEntity;
import com.kiatkoding.ecommerce.model.request.RegisterRequest;
import com.kiatkoding.ecommerce.repository.UserRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
@RequiredArgsConstructor
public class UserService {

    private final UserRepository userRepository;

    public UserEntity userEntity() {
        UserDetails userDetails = (UserDetails) SecurityContextHolder.getContext()
                .getAuthentication()
                .getPrincipal();

        String phoneNumber = userDetails.getUsername();
        return userRepository.findByPhoneNumber(phoneNumber)
                .orElseThrow();
    }


}
```