**Product Management Assignment Documentation**  
~ Sanika Savalajkar  
  
Folder Structure

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

Output Screenshots  
  
1. Swagger Login

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

2. Swagger Output

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

3. Swagger Post Method Output

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

4. Swagger Put Method Output

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

5. Swagger Get Method Output

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

6. Swagger Delete Method Output

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

Project Implementation (Code)  
1. (com.example.demo package)  
ProductManagementApplication.java

|  |
| --- |
| **package** com.example.demo;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  **public** **class** ProductManagementApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(ProductManagementApplication.**class**, args);  System.***out***.println("Product Management App is Running........");  }  } |

2. Model package  
Product.java

|  |
| --- |
| **package** com.example.demo.Model;  **import** jakarta.persistence.Entity;  **import** jakarta.persistence.Id;  @Entity  **public** **class** Product {  @Id  **private** **long** pid;  **private** String pname;  **private** **double** price;  **private** String review;  **public** Product() {  **super**();  // **TODO** Auto-generated constructor stub  }  **public** Product(**long** pid, String pname, **double** price, String review) {  **super**();  **this**.pid = pid;  **this**.pname = pname;  **this**.price = price;  **this**.review = review;  }  **public** **long** getPid() {  **return** pid;  }  **public** **void** setPid(**long** pid) {  **this**.pid = pid;  }  **public** String getPname() {  **return** pname;  }  **public** **void** setPname(String pname) {  **this**.pname = pname;  }  **public** **double** getPrice() {  **return** price;  }  **public** **void** setPrice(**double** price) {  **this**.price = price;  }  **public** String getReview() {  **return** review;  }  **public** **void** setReview(String review) {  **this**.review = review;  }  @Override  **public** String toString() {  **return** "Product [pid=" + pid + ", pname=" + pname + ", price=" + price + ", review=" + review + "]";  }  } |

3. Repository package  
ProductRepository.java Interface

|  |
| --- |
| **package** com.example.demo.Repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** com.example.demo.Model.Product;  **public** **interface** ProductRepository **extends** JpaRepository<Product, Long> {  } |

4. Service Package  
ProductService.java Interface

|  |
| --- |
| **package** com.example.demo.Service;  **import** java.util.List;  **import** com.example.demo.Model.Product;  **public** **interface** ProductService {  Product addProduct(Product product);  List<Product> getAllProducts();  Product getProductById(Long id);  Product updateProduct(Long id, Product product);  **void** deleteProduct(Long id);  } |

ProductServiceImpl.java

|  |
| --- |
| **package** com.example.demo.Service;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** com.example.demo.Exception.ProductNotFoundException;  **import** com.example.demo.Model.Product;  **import** com.example.demo.Repository.ProductRepository;  @Service  **public** **class** ProductServiceImpl **implements** ProductService {  @Autowired  **private** ProductRepository repo;  **public** Product addProduct(Product product) {  **return** repo.save(product);  }  **public** List<Product> getAllProducts() {  **return** repo.findAll();  }  **public** Product getProductById(Long id) {  **return** repo.findById(id)  .orElseThrow(() -> **new** ProductNotFoundException("Product not found with id " + id));  }  **public** Product updateProduct(Long id, Product updated) {  Product existing = getProductById(id);  existing.setPname(updated.getPname());  existing.setPrice(updated.getPrice());  existing.setReview(updated.getReview());  **return** repo.save(existing);  }  **public** **void** deleteProduct(Long id) {  Product product = getProductById(id);  repo.delete(product);  }  } |

5. Controller Package  
ProductController.java

|  |
| --- |
| **package** com.example.demo.Controller;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.web.bind.annotation.DeleteMapping;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.PathVariable;  **import** org.springframework.web.bind.annotation.PostMapping;  **import** org.springframework.web.bind.annotation.PutMapping;  **import** org.springframework.web.bind.annotation.RequestBody;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** com.example.demo.Model.Product;  **import** com.example.demo.Service.ProductService;  @RestController  @RequestMapping("/api/products")  **public** **class** ProductController {  @Autowired  **private** ProductService service;  @PostMapping  **public** Product addProduct(@RequestBody Product product) {  **return** service.addProduct(product);  }  @GetMapping  **public** List<Product> getAllProducts() {  **return** service.getAllProducts();  }  @GetMapping("/{id}")  **public** Product getProductById(@PathVariable Long id) {  **return** service.getProductById(id);  }  @PutMapping("/{id}")  **public** Product updateProduct(@PathVariable Long id, @RequestBody Product product) {  **return** service.updateProduct(id, product);  }  @DeleteMapping("/{id}")  **public** **void** deleteProduct(@PathVariable Long id) {  service.deleteProduct(id);  }  } |

6. Exception Package  
ProductNotFoundException.java

|  |
| --- |
| **package** com.example.demo.Exception;  **public** **class** ProductNotFoundException **extends** RuntimeException {  **public** ProductNotFoundException(String message) {  **super**(message);  }  } |

GlobalExceptionHandler.java

|  |
| --- |
| **package** com.example.demo.Exception;  **import** org.springframework.http.HttpStatus;  **import** org.springframework.http.ResponseEntity;  **import** org.springframework.web.bind.annotation.ExceptionHandler;  **import** org.springframework.web.bind.annotation.RestControllerAdvice;  @RestControllerAdvice  **public** **class** GlobalExceptionHandler {  @ExceptionHandler(ProductNotFoundException.**class**)  **public** ResponseEntity<String> handleProductNotFound(ProductNotFoundException ex) {  **return** **new** ResponseEntity<>(ex.getMessage(), HttpStatus.***NOT\_FOUND***);  }  @ExceptionHandler(Exception.**class**)  **public** ResponseEntity<String> handleGeneral(Exception ex) {  **return** **new** ResponseEntity<>("Internal Server Error: " + ex.getMessage(), HttpStatus.***INTERNAL\_SERVER\_ERROR***);  }  } |

7. Configuration Package  
SecurityConfig.java

|  |
| --- |
| **package** com.example.demo.Configuration;  **import** org.springframework.context.annotation.Bean;  **import** org.springframework.context.annotation.Configuration;  **import** org.springframework.security.config.Customizer;  **import** org.springframework.security.config.annotation.web.builders.HttpSecurity;  **import** org.springframework.security.web.SecurityFilterChain;  @Configuration  **public** **class** SecurityConfig {  @Bean  **public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {  http  .csrf(csrf -> csrf.disable()) // Disable CSRF for simplicity  .authorizeHttpRequests(auth -> auth  .requestMatchers("/swagger-ui/\*\*", "/v3/api-docs/\*\*").permitAll() // Allow Swagger access without login  .anyRequest().authenticated() // All other endpoints require authentication  )  .httpBasic(Customizer.*withDefaults*()); // Enable basic authentication  **return** http.build();  }  } |

8. application.properties file

|  |
| --- |
| spring.application.name=ProductManagement  spring.datasource.url=jdbc:mysql://localhost:3306/productdb  spring.datasource.username=root  spring.datasource.password=Sanika@12345  spring.jpa.hibernate.ddl-auto=update  spring.jpa.show-sql=true  springdoc.swagger-ui.enabled=true  spring.security.user.name=admin  spring.security.user.password=admin123 |

9. pom.xml file

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>3.2.5</version>  <relativePath/>  </parent>  <groupId>com.example</groupId>  <artifactId>product-management</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>product-management</name>  <description>Spring Boot Monolithic Project for Product Management</description>  <properties>  <java.version>17</java.version>  </properties>  <dependencies>  <!-- Spring Web -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- Spring Data JPA -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <!-- MySQL Driver -->  <dependency>  <groupId>com.mysql</groupId>  <artifactId>mysql-connector-j</artifactId>  <scope>runtime</scope>  </dependency>  <!-- Spring Security -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-security</artifactId>  </dependency>  <!-- Springdoc OpenAPI Swagger -->  <dependency>  <groupId>org.springdoc</groupId>  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  <version>2.2.0</version>  </dependency>  <!-- DevTools for hot reload -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  <!-- Testing dependencies -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.springframework.security</groupId>  <artifactId>spring-security-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

10. MySQL Database

|  |
| --- |
| create database productdb;  use productdb; SELECT \* FROM productdb.product; |

|  |
| --- |
|  |