Develop a REST API for an online shopping platform that:

- Allow customers to fetch a list of products
- Retrieve product details
- Add new products
- Update product stock
- Implements soft deletion for products
- Write a Spring Boot controller that supports these operations and ensure exception handling, request validation, and proper HTTP status codes. (16 Marks)

```
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
            <version>3.0.2</version>
        </dependency>
          <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- H2 In-Memory Database -->
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

📁 Project Structure (simplified)

```
com.example.shop
├── controller
│   └── ProductController.java
├── exception
│   ├── GlobalExceptionHandler.java
│   └── ProductNotFoundException.java
├── model
│   └── Product.java
├── repository
│   └── ProductRepository.java
├── service
│   └── ProductService.java
└── ShopApplication.java
```

✅ `Product.java` (Model)

```
package com.example.shop.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import lombok.*;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

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

    @NotNull(message = "Price is required")
    @PositiveOrZero(message = "Price must be >= 0")
    private Double price;

    @PositiveOrZero(message = "Stock must be >= 0")
    private Integer stock;

    private Boolean active = true;
}
```

✅ ProductRepository.java

```
package com.example.shop.repository;

import com.example.shop.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface ProductRepository extends JpaRepository<Product, Long> {
    List<Product> findByActiveTrue();
}
```

✅ ProductService.java

```
package com.example.shop.service;

import com.example.shop.exception.ProductNotFoundException;
import com.example.shop.model.Product;
import com.example.shop.repository.ProductRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ProductService {

    private final ProductRepository repo;

    public ProductService(ProductRepository repo) {
        this.repo = repo;
    }

    public List<Product> getAllActiveProducts() {
        return repo.findByActiveTrue();
    }

    public Product getProduct(Long id) {
        return repo.findById(id)
                .filter(Product::getActive)
                .orElseThrow(() -> new ProductNotFoundException(id));
    }

    public Product addProduct(Product product) {
        product.setId(null); // Ensure it's new
        product.setActive(true);
```

```
        return repo.save(product);
    }

    public Product updateStock(Long id, Integer stock) {
        Product product = getProduct(id);
        product.setStock(stock);
        return repo.save(product);
    }

    public void softDelete(Long id) {
        Product product = getProduct(id);
        product.setActive(false);
        repo.save(product);
    }
}
```

✅ `ProductController.java`

```
package com.example.shop.controller;

import com.example.shop.model.Product;
import com.example.shop.service.ProductService;
import jakarta.validation.Valid;
import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;

import java.util.List;

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

    private final ProductService service;

    public ProductController(ProductService service) {
        this.service = service;
    }

    @GetMapping
    public ResponseEntity<List<Product>> getAll() {
        return ResponseEntity.ok(service.getAllActiveProducts());
    }

    @GetMapping("/{id}")
    public ResponseEntity<Product> getById(@PathVariable Long id) {
        return ResponseEntity.ok(service.getProduct(id));
    }
```

```
    @PostMapping
    public ResponseEntity<Product> create(@Valid @RequestBody Product product) {
        Product saved = service.addProduct(product);
        return new ResponseEntity<>(saved, HttpStatus.CREATED);
    }

    @PatchMapping("/{id}/stock")
    public ResponseEntity<Product> updateStock(@PathVariable Long id,
                                               @RequestParam Integer stock) {
        return ResponseEntity.ok(service.updateStock(id, stock));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> softDelete(@PathVariable Long id) {
        service.softDelete(id);
        return ResponseEntity.noContent().build();
    }
}
```

✅ ProductNotFoundException.java

```
package com.example.shop.exception;

public class ProductNotFoundException extends RuntimeException {
    public ProductNotFoundException(Long id) {
        super("Product not found with id: " + id);
    }
}
```

✅ GlobalExceptionHandler.java

```
package com.example.shop.exception;

import org.springframework.http.*;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;

@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ProductNotFoundException.class)
```

```
    public ResponseEntity<Map<String, String>> handleProductNotFound(ProductNotFoundException
ex) {
        Map<String, String> body = new HashMap<>();
        body.put("error", ex.getMessage());
        return new ResponseEntity<>(body, HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<Map<String, String>> handleValidation(MethodArgumentNotValidException
ex) {
        Map<String, String> errors = new HashMap<>();
        ex.getBindingResult().getFieldErrors().forEach(err ->
            errors.put(err.getField(), err.getDefaultMessage()));
        return new ResponseEntity<>(errors, HttpStatus.BAD_REQUEST);
    }
}
```

✅ `application.properties`

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.h2.console.enabled=true
```

▶ How to Run

1. Add **Spring Web**, **Spring Data JPA**, **H2 Database**, and **Validation** dependencies.
2. Use `@SpringBootApplication` in `ShopApplication.java`
3. Run the project.
4. Use tools like **Postman** or **cURL** to test endpoints:
    - `GET /api/products`

- `GET /api/products/{id}`
- `POST /api/products` (with JSON)
- `PATCH /api/products/{id}/stock?stock=10`
- `DELETE /api/products/{id}` (soft delete)

```
{
 "name": "Wireless Mouse",
 "price": 25.99,
 "stock": 50
}
```