FUNDAMENTALS AND PRACTICES OF SOFTWARE DEVELOPMENT

REVIEW REPORT

BACKEND API BUILDING

FOR INVENTORY MANAGEMENT FOR RETAILERS

GURUPRAKASH K	3122235001049
CHIRANJEEEV PRASANNAA	3122235001037
DEEPALAKSHMI V	3122235001039
HARISH KANNA	3122235001302

PROBLEM STATEMENT

Develop a BackEnd API Fetcher for an E Commerce system for the admin part using custom API created by SpringBoots RestAPI Framework and other Java Web Utility classes

OVERVIEW

Through this project we have tried to create a custom backend API to extend it to the admin site of ECommerce site.
The API we have used is built on JAVA SPRINGBOOTS with the aid of high-end annotations and RestAPI building tools.
With the coming of this, the developers can focus more on the implementation logic rather than the infrastructure of the hosting site much.
The admin part of the system has been covered by our model where the admin can maintain the inventory for the E commerce system by adding, removing products and keep track of number of items available. We have also added a recommendation system that prioritises the addition of most bought items to inventory based on consumer trends.
We have used SQL to maintain our database and the H2 Framework have been used to derive data from the same
We have created our project grounding strongly on the MVC Model of application building which focuses and split and succeed strategy.

TECHNOLOGIES USED

WEB BUILDING

Spring is a comprehensive framework for building Java-based enterprise applications.Spring's flexibility and modular design allow developers to choose only the components they need, whether for web applications, REST APIs, or even batch processing.

React is a popular open-source JavaScript library developed by Facebook for building user interfaces, particularly for single-page applications (SPAs)

HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies for building web pages. HTML provides the structure and content of a webpage, while CSS controls the presentation and styling, making the page visually appealing and enhancing the user experience.

CLASSES AND OBJECTS

Java is known for its portability across platforms, as applications written in Java can run on any device that has a Java Virtual Machine (JVM). This capability is often summarized by the phrase "write once, run anywhere" (WORA)

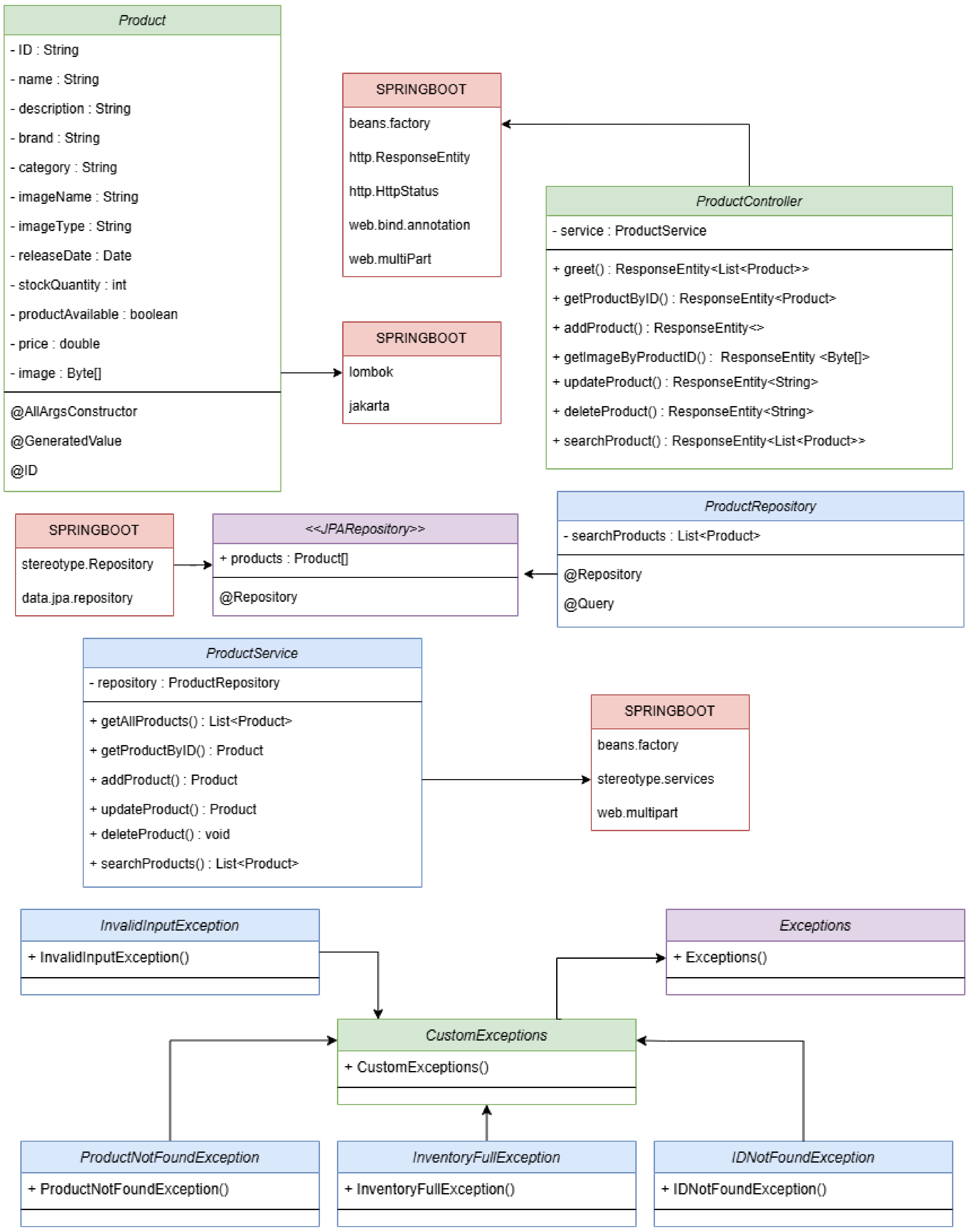
Project Lombok is a Java library that aims to reduce boilerplate code in Java applications. It achieves this by providing a set of annotations that automatically generate common methods, constructors, getters, setters, and more at compile time, allowing developers to write cleaner and more concise code

PLATFORM OF BUILDING AND DATABASE MANAGEMENT

SQL (Structured Query Language) is a standardized programming language used for managing and manipulating relational databases. SQL allows users to perform tasks such as querying data, updating records, inserting new data, and deleting data.

Visual Studio Code (VS Code) is a popular open-source code editor developed by Microsoft. It is widely used by developers due to its flexibility, extensive feature set, and robust support for various programming languages and frameworks

FINAL CLASS DIAGRAM



CORE CONCEPTS

INVERSION OF CONTROL (IOC)

- Transfer of object creation properties to an eternal entity

HOW IT HELPS ?

- MAKES THE CODE MODULAR
- INCREASED EASY MAINTENANCE
- CHANGE IN OBJECT CHARACTER CAN JUST BE BROUGHT BY ENTITY CHANGE WITHOUT ALTERING THE CODE
- MAKES THE CLASSES MORE INDEPENDENT

DEPENDENCY INJECTION

- The process of injecting another class constructor into another class so that separate object creation and allocation is reduced

HOW IT HELPS ?

- LOOSE COUPLING
- EASE IN TESTING
- EASE IN MAINTENANCE : CLEAN CODING
- CONFIGURATION FLEXIBILITY : CONFIGURATION OUTSIDE THE APPLICATION IA ALSO POSSIBLE
- HELPS IN ANNOTATIONS (OUTSIDE CODE)

MODEL VIEW CONTROLLER MODEL

- The practice of separate class creation where the work is distributed as Modelling, Viewing and Controlling

EXAMPLE

- PRODUCT : MODELLING
- PRODUCT SERVICE AND UI : VIEWING
- PRODUCT CONTROL : CONTROLLING

PROJECT LOMBOK

- Creation of setters, getters and some other general methods during compile time reducing the need to manually write

HOW IT HELPS ?

- MODULARITY
- REDUCED CODE
- HANDLES ALL TYPE OF CONSTRUCTORS IE, NO ARG CONSTRUCTOR AND ALL ARGUMENT CONSTRUCTOR
- SNEAKY THROWS INSTEAD OF CUSTOM WRITTEN EXCEPTION FOR CERTAIN DEFINED CASES
- DYNAMIC OBJECT CREATION DURING WEBSITE HOSTING BECOMES EASY IRRESPECTIVE OF USER NAVIGATION

CORE CONCEPTS

SPRING WEB

- framework inside SpringBoots that aids in creation of websites by offering basic services like RestAPI and Dependency Injection

ANNOTATIONS TO KNOW

- @CONTROLLER : HANDLES HTTP REQUESTS
- @RESTCONTROLLER : HELPS IN RETURNING DIRECT JSON
- @RESTMAPPING : MAPS HTTP REQUESTS TO HANDLER METHODS
- @EXCEPTIONHANDLER : HANDLES EXCEPTIONS
- @GETMAPPING, @POSTMAPPING, @PUTMAPPING, @DELETEMAPPING : SPECIALISED METHODS TO PERFORM THE FUNCTION THE NAME SUGGESTS

H2 DATABASE

- Open source Relation Database Management System written in Java. Can run in-memory or stand-alone File application.

- SERVER MODE: THE DATABASE RUNS AS A STANDALONE SERVER, ALLOWING MULTIPLE CLIENTS TO CONNECT VIA TCP OR WEB CONNECTIONS.
- SQL COMPATIBILITY: H2 USES STANDARD SQL SYNTAX, MAKING IT COMPATIBLE WITH OTHER DATABASES LIKE MYSQL AND POSTGRESQL.

WHY IS IT BETTER ?

Focus on Implementation Logic: With a Spring Boot-based setup, the heavy lifting of setting up an application's infrastructure is taken care of by Spring's annotations and configurations. This enables developers to focus on the core business logic, like implementing specific functionalities, custom business rules, and optimized workflows, without worrying about lower-level infrastructure concerns.

Quick Prototyping and Iteration: Thanks to Spring Boot's streamlined development experience, it becomes easier to prototype new features or iterate on existing ones. The embedded server allows the API to be started quickly, enabling rapid testing and feedback loops during development.

Enhanced Stability and Reliability: The project leverages Spring's reliable libraries and standardized development practices, reducing the likelihood of runtime errors and ensuring a more stable application. With consistent dependency injection, unified exception handling, and built-in support for database transactions, the API is both reliable and resilient to failure.

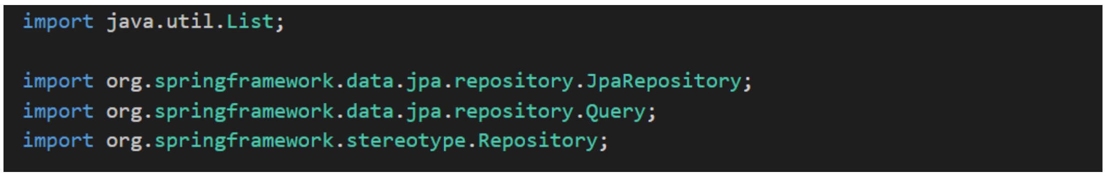
FEATURES OF OOPS

ABSTRACTION

In Java, abstraction is a fundamental concept in object-oriented programming (OOP) that focuses on exposing only essential features while hiding unnecessary details from users. It simplifies complex systems by breaking them down into more manageable parts and encourages focusing on what an object does rather than how it does it.

HOW HAS IT BEEN USED ?

Abstract classes allow subclasses to inherit common behavior while requiring them to implement specific details for abstract methods.
Interfaces enable decoupling, meaning classes can interact with each other through agreed-upon contracts without needing to know specific implementations. Abstraction allows for scalable systems where new functionality can be added without altering existing code. Spring Boot provides starters like spring-boot-starter-web, spring-boot-starter-data-jpa, and spring-boot-starter-security, which abstract away the configuration details of many dependencies. They have been used in ProductControl class which controls the mapping of function with the database.



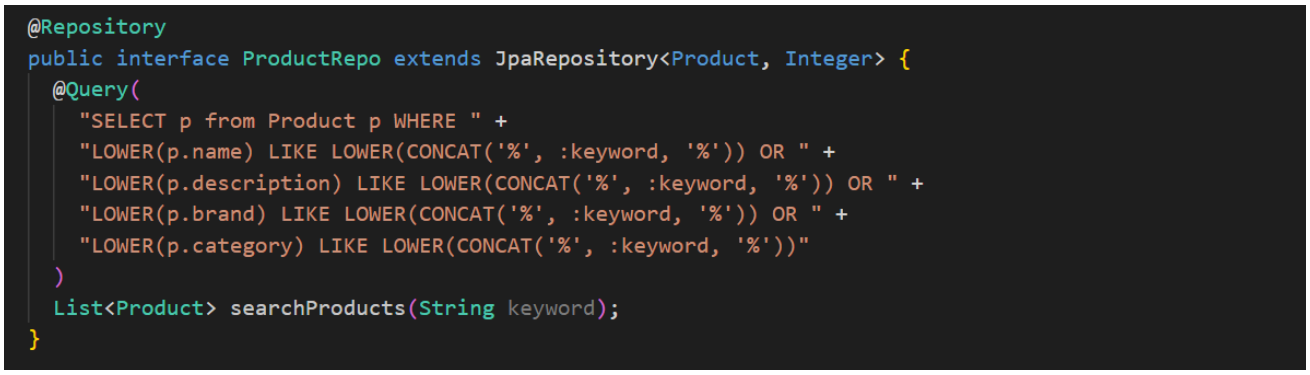
```java
import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
```

POLYMORPHISM

In Spring Boot, polymorphism is an essential concept in object-oriented programming that enables objects of different types to be treated as instances of the same type. It allows methods to perform differently based on the object they're acting upon.

HOW HAS IT BEEN USED ?

In Spring Boot, you can have multiple implementations of an interface and inject the desired implementation dynamically. This makes it easy to extend and modify functionality without changing the client code. When building REST APIs, you might have common operations for different entities. By using polymorphism, you can reduce redundancy by creating a base controller and extending it for specific entities. In Spring Data JPA, you can use repository inheritance, which allows you to define common repository methods in a base repository interface and extend it with specific ones for various entities. Here the class ProductRepo has implemented the JPARepository interface.



```java
@Repository
public interface ProductRepo extends JpaRepository<Product, Integer> {
  @Query(
    "SELECT p from Product p WHERE " +
    "LOWER(p.name) LIKE LOWER(CONCAT('%', :keyword, '%')) OR " +
    "LOWER(p.description) LIKE LOWER(CONCAT('%', :keyword, '%')) OR " +
    "LOWER(p.brand) LIKE LOWER(CONCAT('%', :keyword, '%')) OR " +
    "LOWER(p.category) LIKE LOWER(CONCAT('%', :keyword, '%'))"
  )
  List<Product> searchProducts(String keyword);
}
```

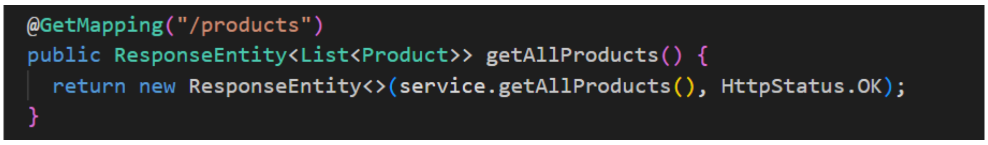
FEATURES OF OOPS

INHERITANCE

In a Spring Boot application, you can use inheritance to create a base controller class with common methods, which other controllers can extend. This helps to avoid duplicate code and make common functionalities reusable. Similar to controllers, services can also inherit from a base service class to avoid code duplication. Spring Data JPA repositories can extend a base repository interface to share common methods.

HOW HAS IT BEEN USED ?

When building RESTful APIs, some methods (such as GET, POST, DELETE) often need to perform similar functions across different resources. For example, if you have multiple resources like Product and Categories, you may have a lot of duplicate code if each controller independently handles similar tasks like fetching all items or fetching by ID. The org.springframework.stereotype.Service has been used to implement the Product services. The org.springframework.context.annotation.Bean, org.springframework.context.annotation.Configuration has been used to implement beans and configure the repositories



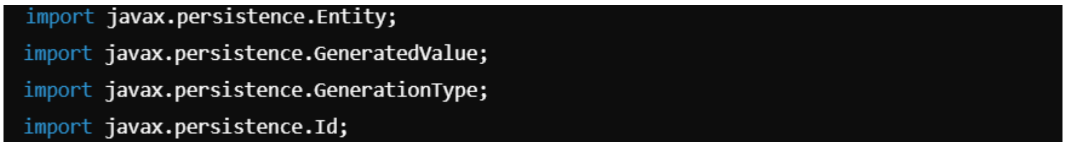
```
@GetMapping("/products")
public ResponseEntity<List<Product>> getAllProducts() {
    return new ResponseEntity<>(service.getAllProducts(), HttpStatus.OK);
}
```

ENCAPSULATION

encapsulation is an essential principle that protects data and implementation details by restricting direct access to the internal workings of objects, exposing only what is necessary. Encapsulation helps manage complexity, reduces dependency on internal logic, and promotes modularity by organizing code into distinct layers and classes with well-defined responsibilities.

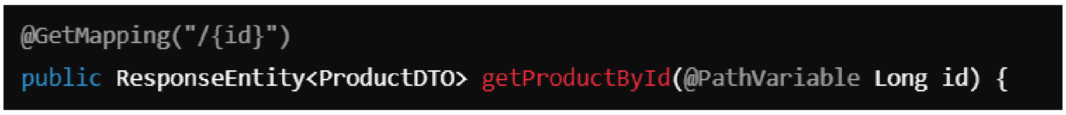
HOW HAS IT BEEN USED ?

- Fields are typically marked as private, and access is provided through public getters and setters.
- Additional logic, like validation, can be added within setters to control the state of the object and ensure data integrity.



```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

Controllers in Spring Boot encapsulate the API logic, handling HTTP requests and responses. They interact with service classes to process data and return results to the client



```
@GetMapping("/{id}")
public ResponseEntity<ProductDTO> getProductById(@PathVariable Long id) {
```

SOME COMMONLY USED ANNOTATIONS

1. @SpringBootApplication

- This is a combination of three annotations: @Configuration, @EnableAutoConfiguration, and @ComponentScan.
- It marks the main class of a Spring Boot application and enables auto-configuration, component scanning, and allows Spring Boot to automatically configure components based on the classpath.

2. @RestController

- Combines @Controller and @ResponseBody.
- Indicates that the class will handle HTTP requests and the return values of its methods should be written directly to the HTTP response body as JSON or XML (typically for RESTful APIs).

3. @RequestMapping

- Used to map HTTP requests to specific handler methods or classes in a controller.
- Can be used for routing requests based on HTTP methods (GET, POST, etc.) and URLs.
- Also supports customization for headers, parameters, and content types.

4.@GetMapping, @PostMapping, @PutMapping, @DeleteMapping

- Specialized versions of @RequestMapping for specific HTTP methods (GET, POST, PUT, DELETE, PATCH).
- These annotations are more intuitive and concise, particularly for RESTful endpoints.

5. @Autowired

- Used for dependency injection.
- Automatically injects a bean (object managed by the Spring IoC container) into a class field, constructor, or setter method.
- This allows for loosely coupled code by managing dependencies externally.

6. @Service

- A specialized form of @Component intended for service layer classes.
- Primarily used to indicate that the class provides business logic and may also contain transaction management.

7. @Repository

- Another specialization of @Component for data access or DAO (Data Access Object) classes.
- It encapsulates database operations and serves as an abstraction layer between the application and data sources.
- Automatically enables exception translation, converting database exceptions into Spring’s DataAccessException hierarchy.

8. @Entity

- Marks a Java class as a JPA entity that maps to a database table.
- Used in conjunction with ORM (Object-Relational Mapping) frameworks like Hibernate to map class fields to table columns.

9. @Id

- Indicates the primary key of a JPA entity.
- This field represents the unique identifier for each row in the database table.

10. @GeneratedValue

- Specifies the generation strategy for primary keys.
- Common strategies include AUTO, IDENTITY, SEQUENCE, and TABLE.

11. @Transactional

- Specifies that a method or class requires transaction management.
- When applied, Spring will automatically handle transactions for the annotated method or all methods within the annotated class.

12. @Configuration

- Indicates that a class contains bean definitions and acts as a source of configuration for the Spring container.
- Classes annotated with @Configuration can define beans using @Bean methods.

13. @Bean

- Used to define a bean in a @Configuration class.
- Methods annotated with @Bean will produce Spring beans to be managed by the Spring container.

14. @Primary

- Used to specify a default bean when multiple beans of the same type exist.
- This is useful for autowiring when there are multiple candidates but only one should be the default.

15. @Qualifier

- Used to resolve ambiguity by specifying which bean should be autowired when multiple beans of the same type are present.

16. @Value

- Allows for injection of values from properties files or environment variables directly into fields or method parameters.
- Commonly used to inject configuration values into components.

17. @PathVariable

- Used to extract dynamic URI values and map them to method parameters.
- Often used in REST APIs to get values from the URL path (e.g., /users/{id}).

18. @RequestParam

- Used to extract query parameters from a URL and map them to method parameters.
- Useful for getting parameters in a URL such as ?name=John.

19. @RequestBody

- Binds the HTTP request body to a method parameter, usually in JSON or XML format.
- Commonly used in REST APIs to receive data from clients.

SCOPES AND LIMITATIONS

SCOPE

- The system can accomodate <> users with the help of WebTools we used
- The database is created solely by us and hence it does not translate the real time user trends
- We have five categories to offer the users and each category consistes of an average of <*> items.
- The description and the specifications of the items are mentioned in consistence with the provided Class diagrams
- The system encompass minimal security features as of now

LIMITATIONS

- The system we have created applies recommendation Algorithms to cater to the user trends. But the algorithm is not very refined to capture minuscule trends of the user.
- The system does not handle certain exceptions like PaymentNotAcceptedExceptions and InputNotValidExceptions and the like.
- The user can primarily opt COD
- The search system primarily works on Name of the product and the category of the product but soon will incorporate trend based Search.