# Deal-VPS Assignment

**Project Description:** Create a simple Java backend application that

implements a RESTful API for managing a collection of items. The specific requirements are as follows:

**1.** **Item Model:** Define a Java class representing an item with properties

like id, name, description, and any other relevant fields.

**2. Data Storage:** The provided Spring Boot application uses Spring Data JPA for data storage. Specifically, it interacts with MySQL.

This setup allows the application to persist and retrieve Item entities in/from a MySQL database using Spring Data JPA.

**3. API Endpoints:** Create RESTful API endpoints to perform the following

operations on items:

•    Add a new item

•    Get a single item by ID

**4. Input Validation:** Validate the input when adding to ensure that

required fields are present and any constraints are satisfied.

# Project Overview

# 1. ItemController class

# This class is a Spring MVC controller that handles HTTP requests related to items.

# POST /api/items/: Creates a new item.

# PUT /api/items/{itemId}: Updates an existing item.

# DELETE /api/items/{itemId}: Deletes an item.

# GET /api/items/: Retrieves a list of all items.

# GET /api/items/{itemId}: Retrieves a single item by ID.

# 2. Item class

# This class represents the entity Item that is persisted in the database. It has an id, name, and description.

# 3. GlobalExceptionHandler class

# This class is a global exception handler for the application. It handles two types of exceptions:

# ResourseNotFoundException: Thrown when a requested resource (item) is not found.

# MethodArgumentNotValidException: Thrown when validation of method arguments fails, typically used for input validation.

# 4. ResourseNotFoundException class

# A custom exception class for resource not found scenarios.

# 5. ApiResponse class

# A simple DTO (Data Transfer Object) representing the response format for API responses.

# 6. ItemDto class

# A DTO representing the data transfer object for Item. It is used for input/output when interacting with the API.

# 7. ItemRepository interface

# An interface that extends JpaRepository for basic CRUD operations on the Item entity.

# 8. ItemService interface and ItemServiceImpl class

# The service layer interfaces and implementations for performing business logic related to items. It includes methods for creating, updating, retrieving, listing, and deleting items.

# 9. MohitcodesApplication class

# The main class with the main method to run the Spring Boot application. It also contains a ModelMapper bean for mapping between entities and DTOs.

# Note:

# The code uses Spring Data JPA for database operations.

# The ModelMapper is used to simplify the mapping between DTOs and entities.

# Validation annotations (@Valid, @NotNull, @NotEmpty, @Size) are used for input validation.

# The code follows RESTful conventions for API design.

# Make sure to configure your database settings in the application properties file, and you may need to add additional dependencies based on your project requirements.