System Requirements Specification

Index

For

Laptop Store App

Version 1.0

TABLE OF CONTENTS

LAPTOP STORE APP

System Requirements Specification

You need to consume APIs exposed by Backend application in Angular to make application work as FULLSTACK

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

The **Laptop Store App** is implemented using Spring Boot with a MySQL database. The application aims to provide a comprehensive platform for managing and organizing laptop related specifications.

Following is the requirement specifications:

	Laptop Store App
Modules	
1	Laptop
Event Module Functionalities	
1	Create a Laptop
2	Update the existing Laptop details
3	Get the Laptop by Id
4	Get all Laptops
5	Delete a Laptop
6	Search for Laptop by Laptop name
7	Search for Laptop by price
8	Search for Laptop by brand

2 Assumptions, Dependencies, Risks / Constraints

2.1 LAPTOP CONSTRAINTS

- When fetching a Laptop by ID, if the laptop ID does not exist, the operation should throw a custom exception.
- When updating a Laptop, if the laptop ID does not exist, the operation should throw a custom exception.
- When removing a Laptop, if the laptop ID does not exist, the operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in dto classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 Business Validations

- Name is not null, min 3 and max 20 characters.
- Price should not be null, should be a non-negative value and should not exceed 9999.
- Brand is not null.
- Storage is not null.
- RAM is not null.
- Processor is not null.

4 REST ENDPOINTS

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 TRAINCONTROLLER

URL Exposed		Purpose
1. /laptops		Fetches all the laptops
Http Method	GET	
Parameter	-	
Return	List<Laptops>	
2. /laptops		Add a new laptop details
Http Method	POST	
Parameter 1	Laptop	
Return	Laptop	
3. /laptops/{id}		Delete laptop with given laptop id
Http Method	DELETE	
Parameter 1	Long (id)	
Return	-	
4. /laptops/{id}		Fetches the laptop with the given id
Http Method	GET	
Parameter 1	Long (id)	
Return	Laptop	
5. /laptops/{id}		Updates existing Laptop info
Http Method	PUT	
Parameter 1	Long (id)	
Parameter 2	Laptop	
Return	Laptop	
6. /laptops/search?name={name}		Search the laptop with the given name
Http Method	GET	
Parameter 1	String (name)	
Return	List<Laptops>	
7. /laptops/search?price={price}		Search the laptop with the given price
Http Method	GET	
Parameter 1	Double (price)	
Return	List<Laptops>	

8. /laptops/search?brand={brand}		Search the laptop with the given brand
Http Method	GET	
Parameter 1	String (brand)	
Return	List<Laptops>	

5 TEMPLATE CODE STRUCTURE

5.1 PACKAGE: COM.LAPTOPSTORE

Resources

LaptopApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.LAPTOPSTORE.REPOSITORY

Resources

Class/Interface	Description	Status
LaptopRepository (interface)	• Repository interface exposing CRUD functionality for Laptop Entity. • You can go ahead and add any custom methods as per requirements.	Partially implemented.

5.3 PACKAGE: COM.LAPTOPSTORE.SERVICE

Resources

Class/Interface	Description	Status

LaptopService (interface)	• Interface to expose method signatures for laptop related functionality. • Do not modify, add or delete any method.	Already implemented.

5.4 PACKAGE: COM.LAPTOPSTORE.SERVICE.IMPL

Class/Interface	Description	Status
LaptopServiceImpl (class)	• Implements LaptopService. • Contains template method implementation. • Need to provide implementation for laptop related functionalities. • Do not modify, add or delete any method signature	To be implemented.

5.5 PACKAGE: COM.LAPTOPSTORE.CONTROLLER

Resources

Class/Interface	Description	Status
LaptopController (Class)	• Controller class to expose all rest-endpoints for laptop related activities. • May also contain local exception handler methods	To be implemented

5.6 PACKAGE: COM.LAPTOPSTORE.DTO

Resources

Class/Interface	Description	Status
LaptopDTO (Class)	Use appropriate annotations from the Java Bean Validation API for validating attributes of this class.	Partially implemented.

5.7 PACKAGE: COM.LAPTOPSTORE.ENTITY

Resources

Class/Interface	Description	Status
Laptop (Class)	• This class is partially implemented. • Annotate this class with proper annotation to declare it as an entity class with **laptopId** as primary key. • Map this class with a **laptop table**. • Generate the **laptopId** using the IDENTITY strategy	Partially implemented.

5.8 PACKAGE: COM.LAPTOPSTORE.EXCEPTION

Resources

Class/Interface	Description	Status
ResourceNotFoundException (Class)	• Custom Exception to be thrown when trying to fetch or delete the laptop info which does not exist. • Need to create Exception Handler for same wherever needed (local or global)	Already implemented.
ExceptionHandlerController (Class)	• RestControllerAdvice Class for defining global exception handlers. • Contains Exception Handler for **InvalidDataException** class. • Use this as a reference for creating exception handler for other custom exception classes.	Already implemented.

6 CONSIDERATIONS

A. There is no roles in this application
B. You can perform the following possible action

Laptop

1 EXECUTION STEPS TO FOLLOW FOR BACKEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **cd into your backend project folder**
4. **To build your project use command:**

   mvn clean package -Dmaven.test.skip
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**

   java -jar <your application jar file name>
6. **This editor Auto Saves the code.**
7. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
8. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
9. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**
10. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
11. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**

11. To login to mysql instance: Open new terminal and use following command:

    a. sudo systemctl enable mysql

    b. sudo systemctl start mysql

    NOTE: After typing the second sql command (sudo systemctl start mysql), you may encounter a warning message like :

    > System has not been booted with systemd as init system (PID 1). Can't operate. Failed to connect to bus: Host is down

    >> Please note that this warning is expected and can be disregarded. Proceed to the next step.

    c. mysql -u root -p

    The last command will ask for password which is 'pass@word1'

12. Mandatory: Before final submission run the following command:

    mvn test

13. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.