System Requirements Specification

Index

For

Property Listing Platform

Version 1.0

TABLE OF CONTENTS

<div align="center">

PROPERTY LISTING PLATFORM
System Requirements Specification

</div>

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

<div align="center">

BACKEND-SPRING BOOT RESTFUL APPLICATION

</div>

.

1 PROJECT ABSTRACT

.

The **Property Listing Platform** is a FullStack Application with a backend implemented using Spring Boot with a MySQL database and a frontend developed using Angular. It serves as a comprehensive platform for managing and listing properties for sale.

Following is the requirement specifications:

		Property Listing Platform
Modules		
	1	Property
Property Module Functionalities		
	1	Create a Property
	2	Update the existing Property details
	3	Get the Property by Id
	4	Get all Properties
	5	Delete an Property
	6	Search for Property by Name
	7	Search for Property by Price
	8	Search for Property by Category

2 Assumptions, Dependencies, Risks / Constraints

2.1 PROPERTY CONSTRAINTS

- When fetching a Property by ID, if the property ID does not exist, the operation should throw a custom exception.
- When updating a property, if the property ID does not exist, the operation should throw a custom exception.
- When removing a property, if the property 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.
- The number of rooms should not be null and should be a min 5 and max 200.
- Price should not be null, should be a non-negative value and should not exceed 9999.
- Category is not null, and select only from APARTMENTS, VILLAS, COTTAGES and HOUSES.

4 REST ENDPOINTS

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

4.1 PROPERTYCONTROLLER

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

8. /properties/search?category={category}		Fetches the property with the given category
Http Method	GET	
Parameter 1	String (category)	
Return	Property	

5 TEMPLATE CODE STRUCTURE

5.1 PACKAGE: COM.YAKSHA.PROPERTYLIST

Resources

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

5.2 PACKAGE: COM.YAKSHA.PROPERTYLIST.REPOSITORY

Resources

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

5.3 PACKAGE: COM.YAKSHA.PROPERTYLIST.SERVICE

Resources

Class/Interface	Description	Status
PropertyService (interface)	● Interface to expose method signatures for property related functionality. ● Do not modify, add or delete any method.	Already implemented.

5.4 PACKAGE: COM.YAKSHA.PROPERTYLIST.SERVICE.IMPL

Resources

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

5.5 PACKAGE: COM.YAKSHA.PROPERTYLIST.CONTROLLER

Resources

Class/Interface	Description	Status
PropertyController (Class)	1. Controller class to expose all rest-endpoints for property related activities. 2. May also contain local exception handler methods	To be implemented

5.6 PACKAGE: COM.YAKSHA.PROPERTYLIST.DTO

Resources

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

5.7 PACKAGE: COM.YAKSHA.PROPERTYLIST.ENTITY

Resources

Class/Interface	Description	Status
Property (Class)	- This class is partially implemented. - Annotate this class with proper annotation to declare it as an entity class with propertyId as primary key. - Map this class with a property table. - Generate the propertyId using the IDENTITY strategy	Partially implemented.
PropertyCategory (Enum)	This enum is already implemented.	Already implemented.

5.8 PACKAGE: COM.YAKSHA.PROPERTYLIST.EXCEPTION

Resources

Class/Interface	Description	Status
ErrorResponse (Class)	● Object of this class is supposed to be returned in case of exception through exception handlers	Already implemented.
ResourceNotFoundException (Class)	● Custom Exception to be thrown when trying to fetch or delete the user info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Partially implemented.
RestExceptionHandler (Class)	● Custom Exception to be thrown when trying to fetch or delete the user info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Partially implemented.

6 CONSIDERATIONS
.

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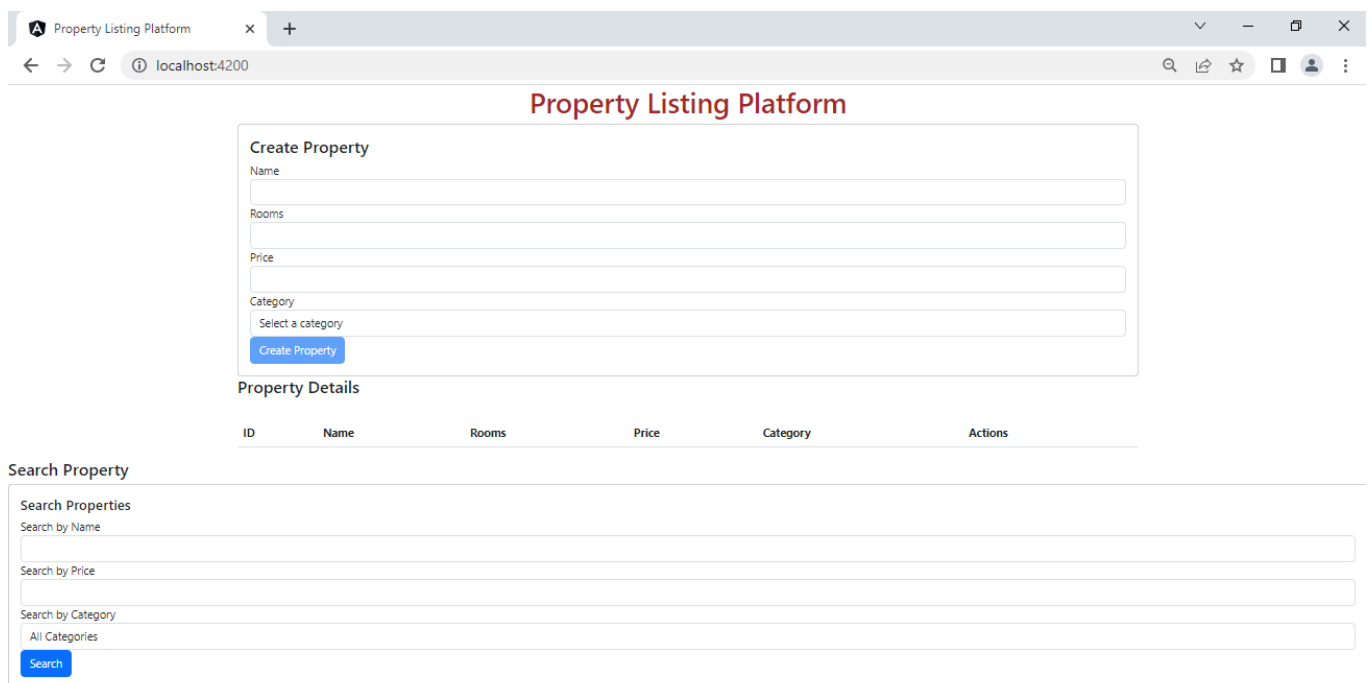B. You can perform the following possible action

Property

FRONTEND-ANGULAR SPA

1 PROBLEM STATEMENT

Property Listing Platform is SPA (Single Page Application), it allows to manage the properties with functionalities to create a new property, update an existing property, get detailed information, and search property.

2. PROPOSED PROPERTY LISTING PLATFORM WIREFRAME

2. 1 HOME PAGE

AFTER CREATING THE PROPERTIES:

Property Listing Platform

Create Property

Name

Rooms

Price

Category

Create Property

Property Details

ID	Name	Rooms	Price	Category	Actions
3	Property1	6	9999	COTTAGES	Edit Delete
4	Property2	7	8000	HOUSES	Edit Delete

Search Property

Search Properties

Search by Name

Search by Price

Search by Category

All Categories

Search

ON CLICK EDIT BUTTON:

Property Listing Platform

Create Property

Name

Property1

Rooms

6

Price

9999

Category

Cottages

Update Property

Property Details

ID	Name	Rooms	Price	Category	Actions
3	Property1	6	9999	COTTAGES	Edit Delete
4	Property2	7	8000	HOUSES	Edit Delete

Search Property

Search Properties

Search by Name

Search by Price

Search by Category

All Categories

Search

3. BUSINESS-REQUIREMENT:

As an application developer, develop the Property Listing Platform App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Home Page	As a user I should be able to visit the home page as the default page. Where I can see a form to create or update the property, a list of all properties with options to edit and delete any property and at last there should be a form to search any property on its name, price or category criteria.

US_01	Home Page	As a user I should be able to see the homepage and perform all operations:
		
		Acceptance criteria:
		
		1. As a user I should be able to furnish the following details at the time of creating the property.
		
		    1.1 Name
		
		    1.2 Rooms
		
		    1.3 Price
		
		    1.4 Category
		
		2. Create Property button should be disabled until all fields are validated.
		
		3. Update Property button should be displayed when you click on the Edit button.
		
		4. Update Property button should be disabled until all fields are validated.
		
		5. Name field min length is 3 and max length 20.
		
		6. Rooms filed min value is 5 and max value 200.
		
		7. Price field min value is 0 and max value 9999.
		
		8. All fields are mandatory. If any field is missing or if any constraint is not satisfied then must show a message.
		
		9. Property Form control names should be case sensitive and they should be like as follows:
		
		    name
		
		    rooms
		
		    price
		
		    category

		10. Search form fields are case sensitive and they should be like as follows: name price category These 3 fields are not mandatory in the search form.

7 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**

12. 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 any of the above commands you might encounter any warnings.

    >> 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'

13. Mandatory: Before final submission run the following command:
    mvn test

14. 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.

8   EXECUTION STEPS TO FOLLOW FOR FRONTEND

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

   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.

3. This is a web-based application, to run the application on a browser, use the internal browser in the environment.

4. You can follow series of command to setup Angular environment once you are in your project-name folder:

   a. npm install -> Will install all dependencies -> takes 10 to 15 min

   b. npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min

   c. npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min

5. 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.