System Requirements Specification

Index

For

Ecommerce Admin App

Version 1.0

TABLE OF CONTENTS

ECOMMERCE ADMIN APP
System Requirements Specification

1 PROJECT ABSTRACT

"**Ecommerce Admin App**" is a full-stack e-commerce application designed to provide a seamless handling of product inventory for ecommerce platform. It leverages the MERN stack, with MongoDB as the database, Express.js for the API endpoints, React.Js for the frontend, and Node.js as the runtime environment.

Following is the requirement specifications:

	Ecommerce Admin App
Modules	
1	Product
Product Module Functionalities	
1	Get all Products
2	Create a Product
3	Search Product by name
4	Update the Product details
5	Delete a Product

2 ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

2.1 Product CONSTRAINTS

- When creating a Product, if any of the required field is missing, the operation should throw a custom exception.
- When updating a Product, if the Product ID does not exist, the operation should throw a custom exception.
- When deleting a Product, if the Product 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 Rest Endpoint methods and Exception Handlers must return data in Json format with status code.

3 REST ENDPOINTS

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

3.1 PRODUCT CONTROLLER

URL Exposed		Purpose
1. /api/product		Fetches all Products
Http Method	GET	
Parameter	-	
Return	Products	
2. /api/product		Creates a new product
Http Method	POST	
Parameter	Product	
Return	Product	
3. /api/product/search		Searches a products by name of product
Http Method	GET	
Parameter	Product name	
Return	Products	
4. /api/product/[productID]		Update a product by its ID
Http Method	PUT	
Parameter 1	ProductID	
Parameter 2	Product	
Return	Product	
5. /api/product/[productID]		Delete a product by its ID
Http Method	DELETE	
Parameter	ProductID	
Return	Product	

4 TEMPLATE CODE STRUCTURE

4.1 Product code structure

1) MODULES/PRODUCT: Controller

Resources

ProductController (Class)	This is the controller class for the product module.	To be implemented

2) MODULES/PRODUCT: dao

Resources

File	Description	Status
models/product model	Models for product	Already implemented
schemas/product schema	Schemas for product	Already implemented

3) MODULES/PRODUCT: routes

Resources

File	Description	Status
Product routes	Routes for product	Already implemented.

4) MODULES/PRODUCT: service

Resources

Class	Description	Status
ProductService	● Defines ProductService	Already implemented.

5) MODULES/PRODUCT: service/impl

Resources

Class	Description	Status

ProductServiceImpl	● Implements ProductService.	To be implemented.

FRONTEND-REACTJS SPA

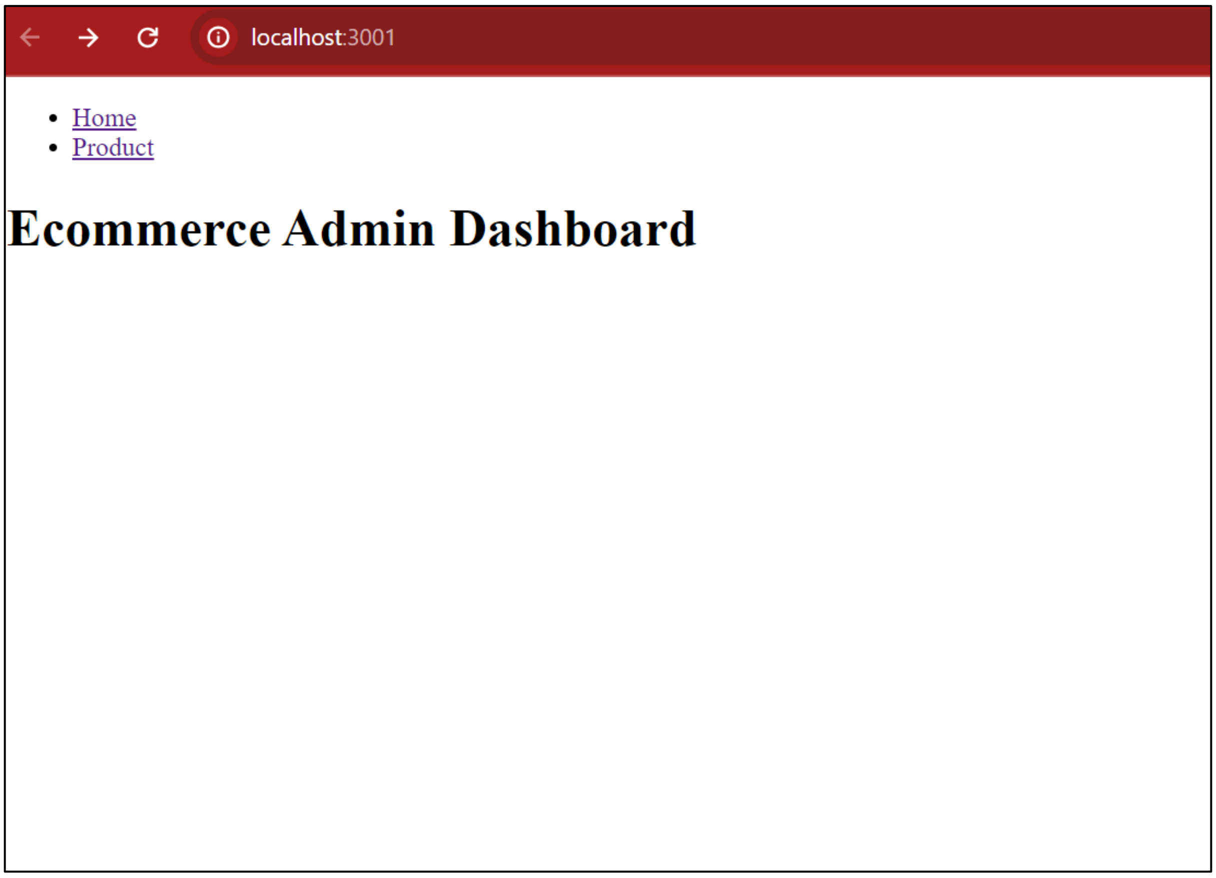
1 PROBLEM STATEMENT

"**Ecommerce Admin App**" is a SPA in ReactJs with full-stack e-commerce application designed to provide a seamless handling of products.

2 PROPOSED Ecommerce Admin App

UI needs improvisation and modification as per given use case and to make test cases passed.

Home Page

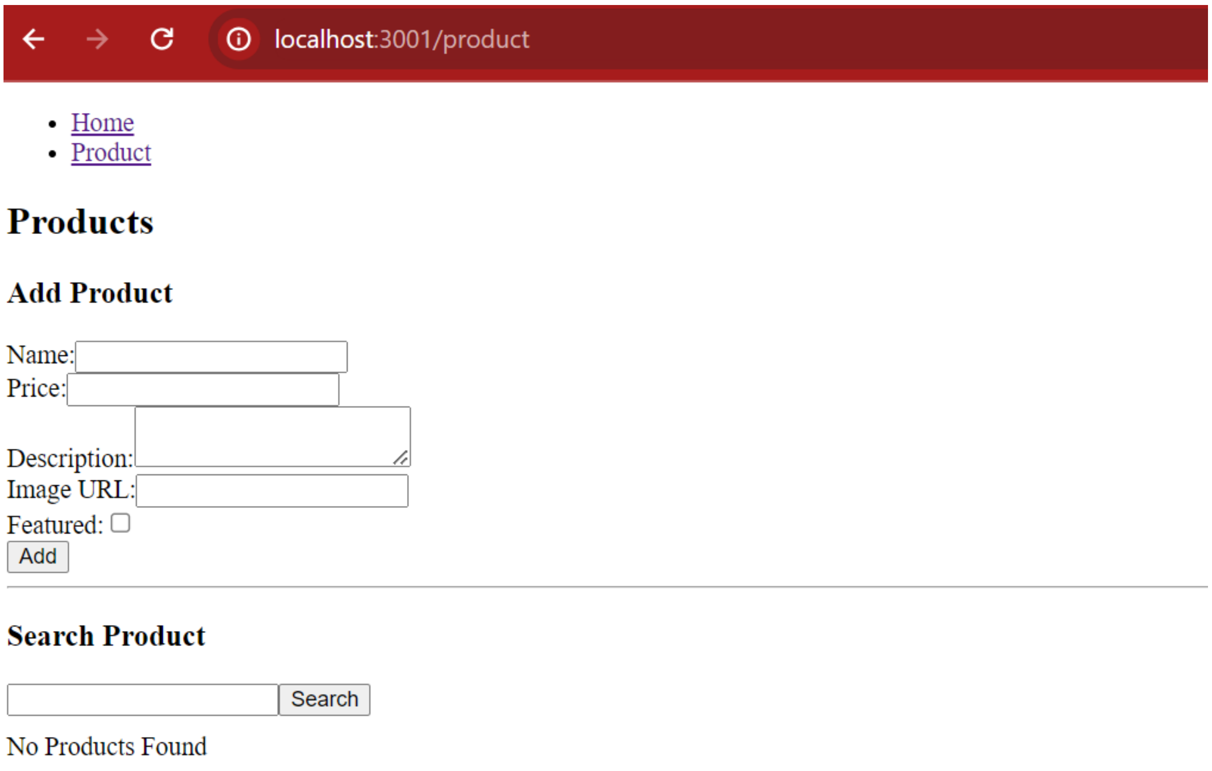


localhost:3001

- Home
- Product

Ecommerce Admin Dashboard

Product Page


localhost:3001/product

- Home
- Product

Products

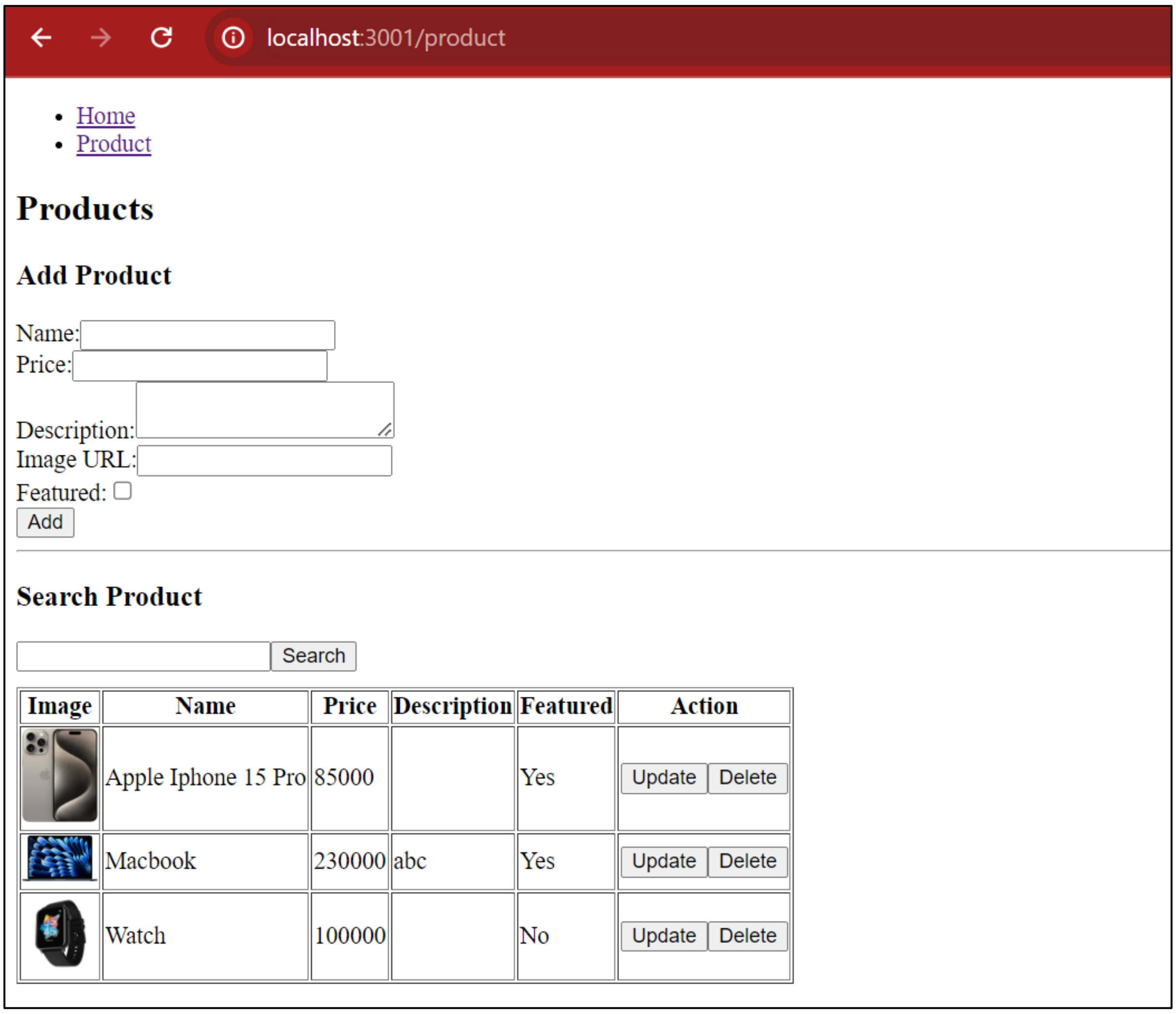
Add Product

Name:
Price:
Description:
Image URL:
Featured:
Add

Search Product

Search

No Products Found


localhost:3001/product

- Home
- Product

Products

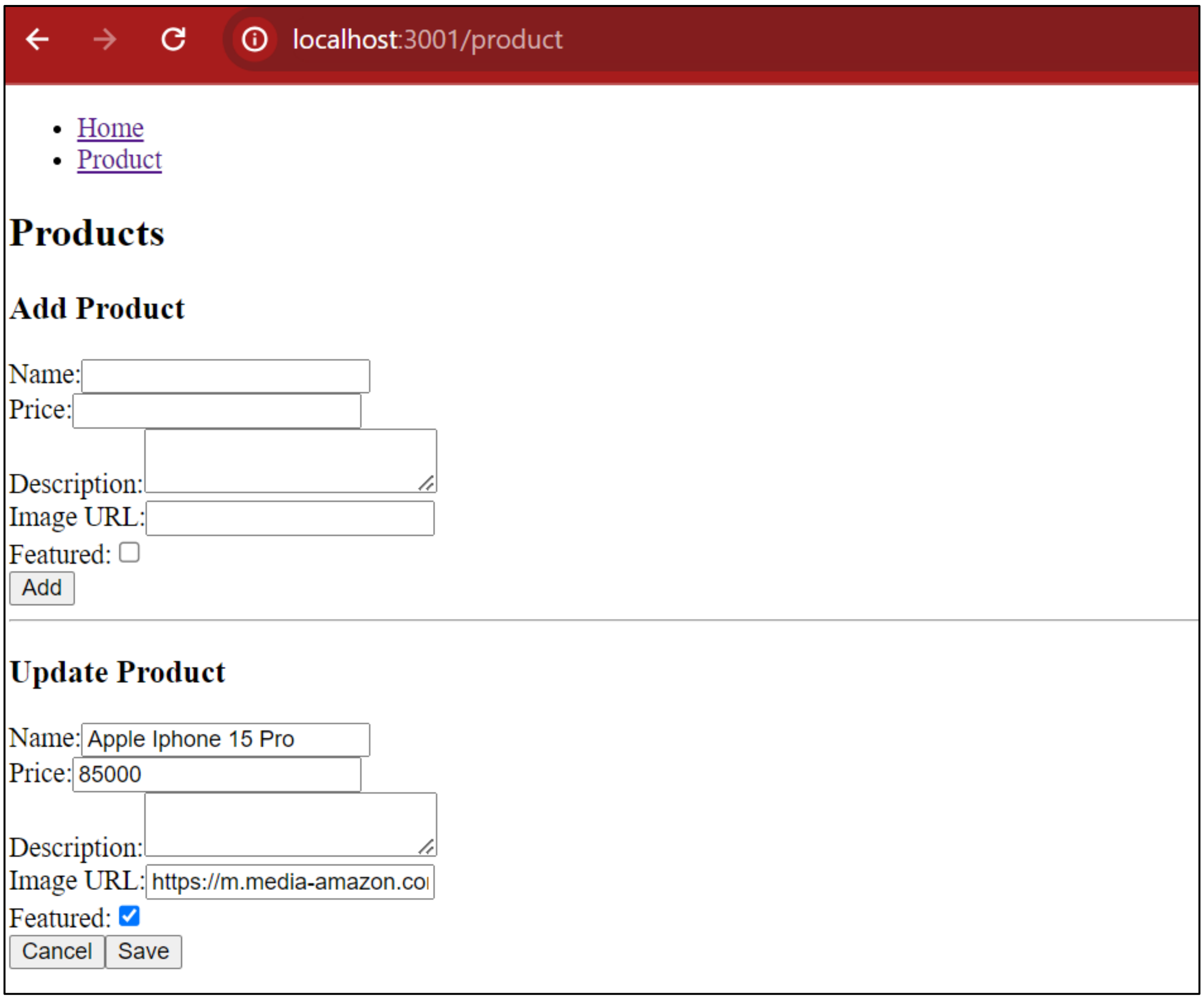
Add Product

Name:
Price:
Description:
Image URL:
Featured:
Add

Search Product

Search

Image	Name	Price	Description	Featured	Action
	Apple Iphone 15 Pro	85000		Yes	Update Delete
	Macbook	230000	abc	Yes	Update Delete
	Watch	100000		No	Update Delete



- Home
- Product

Products

Add Product

Name:
Price:
Description:
Image URL:
Featured: ☐
Add

Update Product

Name: Apple Iphone 15 Pro
Price: 85000
Description:
Image URL: https://m.media-amazon.co
Featured: ✓
Cancel Save

3 Business-Requirement:

As an application developer, develop the 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 and below links are accessible to me: 1. Home 2. Product
US_02	Navigation Links	As a user, I should able to see navigation link on every pages.

US_03	Product Page	As a user, I should able to see product page when click on product navigation link. Acceptance Criteria: 1. Be able to see Add product form with appropriate label and all fields. 2. Be able to see “No Product Found” when no product is available. 3. Be able to search product by it name. 4. Be able to see product in table with right heading when product are fetch. 5. Be able to see update product form when click on update button

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**

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

3. **This editor Auto Saves the code.**
4. **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.**
5. **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.**
6. **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.**

7. **You can follow series of command to setup express 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 run the project.**
   c. **npm run jest -> to run all test cases and see the summary of all passed and failed test cases.**
   d. **npm run test -> to run all test cases and register the result of all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
8. **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.**

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 ReactJs 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:3000 to open project in browser -> takes 2 to 3 min**
   c. **npm run jest -> To run all test cases and check the summary.**
   d. **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.**