# ECOMMERCE APPLICATION

# CONTENTS

# 1 PROBLEM STATEMENT

"E Commerce Application " is a Single Page Application (SPA) that empowers users to list all products with details like name, description and price. It allows users to edit and delete any product along with giving the option to search any product also.

# 2 PROPOSED E COMMERCE APPLICATION WIREFRAME

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

## 2.1 WELCOME PAGE

## 2.2 SCREEN SHOTS:

**\*\*\*Add Product\*\*\***

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Add a Product

Name: `Phone`   Description: `Description for Phone`   Price: `10000`   [Add Product]

## Products List

Filter by Name: [        ]

- **Name:** Mousee
  **Description:** Description for Mouse
  **Price:** 19.99 [Edit] | [Delete]
- **Name:** Keyboard
  **Description:** Description for Keyboard
  **Price:** 29.99 [Edit] | [Delete]
- **Name:** Carpet
  **Description:** Description for Carpet
  **Price:** 39.99 [Edit] | [Delete]

**\*\*\*Added Product\*\*\***

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Add a Product

Name: [        ]   Description: [        ]   Price: [        ]   [Add Product]

## Products List

Filter by Name: [        ]

- **Name:** Mousee
  **Description:** Description for Mouse
  **Price:** 19.99 [Edit] | [Delete]
- **Name:** Keyboard
  **Description:** Description for Keyboard
  **Price:** 29.99 [Edit] | [Delete]
- **Name:** Carpet
  **Description:** Description for Carpet
  **Price:** 39.99 [Edit] | [Delete]
- **Name:** Phone
  **Description:** Description for Phone
  **Price:** 10000 [Edit] | [Delete]

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Edit Product

Name: [Mousee] Description: [Description for Mouse] Price: [19.99] [Update Product]

## Products List

Filter by Name: [ ]

- **Name:** Mousee
  **Description:** Description for Mouse
  **Price:** 19.99 [Edit] [Delete]
- **Name:** Keyboard
  **Description:** Description for Keyboard
  **Price:** 29.99 [Edit] [Delete]
- **Name:** Carpet
  **Description:** Description for Carpet
  **Price:** 39.99 [Edit] [Delete]
- **Name:** Phone
  **Description:** Description for Phone
  **Price:** 10000 [Edit] [Delete]

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Edit Product

Name: [DELL Mouse] Description: [Description for Mouse] Price: [29.99] [Update Product]

## Products List

Filter by Name: [ ]

- **Name:** Mousee
  **Description:** Description for Mouse
  **Price:** 19.99 [Edit] [Delete]
- **Name:** Keyboard
  **Description:** Description for Keyboard
  **Price:** 29.99 [Edit] [Delete]
- **Name:** Carpet
  **Description:** Description for Carpet
  **Price:** 39.99 [Edit] [Delete]
- **Name:** Phone
  **Description:** Description for Phone
  **Price:** 10000 [Edit] [Delete]

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Add a Product

Name:[            ] Description:[            ] Price:[            ] Add Product

## Products List

Filter by Name:[            ]

- **Name:** DELL Mouse
  **Description:** Description for Mouse
  **Price:** 29.99 Edit | Delete
- **Name:** Keyboard
  **Description:** Description for Keyboard
  **Price:** 29.99 Edit | Delete
- **Name:** Carpet
  **Description:** Description for Carpet
  **Price:** 39.99 Edit | Delete
- **Name:** Phone
  **Description:** Description for Phone
  **Price:** 10000 Edit | Delete

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Add a Product

Name:[            ] Description:[            ] Price:[            ] Add Product

## Products List

Filter by Name:[            ]

- **Name:** DELL Mouse
  **Description:** Description for Mouse
  **Price:** 29.99 Edit | Delete
- **Name:** Keyboard
  **Description:** Description for Keyboard
  **Price:** 29.99 Edit | Delete
- **Name:** Phone
  **Description:** Description for Phone
  **Price:** 10000 Edit | Delete

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Add a Product

Name: [            ] Description: [            ] Price: [            ] [Add Product]

## Products List

Filter by Name: [key    ]

- **Name:** Keyboard
  **Description:** Description for Keyboard
  **Price:** 29.99 [Edit] [Delete]

localhost:8081/

# Welcome to your Product Catalog

## Product Form

## Add a Product

Name: [            ] Description: [            ] Price: [            ] [Add Product]

## Products List

Filter by Name: [charger ]

- No products found

# 3 BUSINESS-REQUIREMENT:

As an application developer, develop the E Commerce Application (Single Page App) with below guidelines:

| User Story # | User Story Name | User Story |
|---|---|---|
| US_01 | Welcome Page | As a user I should be able to visit the welcome page as default page.<br><br>## Acceptance Criteria:<br>### AppComponent:<br>1. Should display the heading **"Welcome to your Product Catalog"** inside an <h2> tag.<br>2. Should display the heading **"Product Form"** inside an <h2> tag.<br>3. Should render the **ProductForm** component for product creation and editing.<br>4. Should display the heading **"Products List"** inside an <h2> tag.<br>5. Should render the **ProductList** component to display available products.<br><br>**State & Functionality Overview:**<br>**State Variables**:<br>1. products:<br>    ● Type: array<br>    ● Purpose: Holds the list of all products.<br>    ● Initial Value: [ ] (empty array)<br>2. editProduct:<br>    ● Type: object or null<br>    ● Purpose: Stores the product currently being edited.<br>    ● Initial Value: null<br>**Fetching Products:**<br>1. Use the useEffect hook to fetch products from the backend **once when the component mounts**.<br>2. Make a GET request to: http://localhost:4000/products<br>3. On success, update the products state with the response data.<br>4. On failure, log the error to the console with a meaningful message.<br>**Add Product Functionality**:<br>1. Accepts a product object as an argument. |

2. Makes a **POST** request to:
   `http://localhost:4000/products`
3. On success, append the new product (from the response) to
   the `products`.
4. Handle any errors by logging them.

**Delete Product Functionality**:
1. Accepts a productId object as an argument.
2. Makes a **DELETE** request to:
   `http://localhost:4000/products/{id}`
3. On success, remove the corresponding product from the
   `products` array.
4. Log errors appropriately.

**Update Product Functionality**:
1. Accepts a product object with updated data as an argument.
2. Makes a **PUT** request to:
   `http://localhost:4000/products/{id}`
3. On success, update the corresponding product in the
   `products` list.
4. Also reset the `editProduct` state back to `null` after
   updating.
5. Errors should be logged to the console.

## ProductForm Component:

This component is used for both **adding new products** and **editing
existing ones**. It displays a form with input fields for product details
and triggers the appropriate handler (`addProduct` or
`updateProduct`) on form submission.

**Props:**
1. **addProduct(product)** – Function to create a new
   product.
2. **editProduct** – The product object being edited, or `null`.
3. **updateProduct(product)** – Function to update an
   existing product.

**State Management:**
1. Maintains a local state object `product` with the following
   properties:
   - `name` (string)
   - `description` (string)
   - `price` (number)
2. On component mount or whenever `editProduct`
   changes:

9

- If `editProduct` exists, populate the form fields with its values.
- If not, reset the form to empty fields.
3. Use the `useEffect` hook to handle this logic.

**Form Behavior:**
1. **Form Fields**:
    - `name` – required
    - `description`
    - `price` – required, numeric input
2. **Form Mode**:
    - Determine if it's an **add** or **edit** operation using the presence of `editProduct`.
    - Show appropriate heading in <h2> and button text respectively as:
        ➔ If editing: "Edit Product" and "Update Product"
        ➔ If adding: "Add a Product" and "Add Product"
3. **Submission Logic**:
    - On form submission:
        ➔ Prevent default behavior.
        ➔ If editing, call `updateProduct(product)`.
        ➔ If adding, call `addProduct(product)`.
    - After submission, reset the form fields to empty.
4. **Validation**:
    - Disable the submit button if required fields (`name` or `price`) are empty.

**HTML Structure:**
1. Displays a heading using an `<h2>` tag:
    - **"Edit Product"** if editing.
    - **"Add a Product"** if adding.
2. Renders a `<form>` with the following labeled input fields:
    - **Name** (`<input type="text">`) – required.
    - **Description** (`<input type="text">`) – optional.
    - **Price** (`<input type="number">`) – required.
3. The form contains three input fields, each with a label. Here's how they should be structured:
    - **Name Field**:
        ➔ Use a label with the text **"Name:"**.

➔ This label should be linked to the input using the `htmlFor` attribute.
➔ The corresponding input should have the `id` set to `"name"`.
- **Description Field**:
  ➔ Use a label with the text **"Description:"**.
  ➔ Link the label to its input with `htmlFor="description"`.
  ➔ The input should have `id="description"`.
- **Price Field**:
  ➔ Use a label with the text **"Price:"**.
  ➔ Link it using `htmlFor="price"` and match the input `id` accordingly.

4. Each input uses the `value` from state and updates state on `onChange`.
5. A `<button type="submit">` is shown:
   - Text: **"Update Product"** when editing.
   - Text: **"Add Product"** when adding.
   - The button is **disabled** if required fields (`name`, `price`) are empty.

**\*\* Kindly refer to the screenshots for any clarifications. \*\***

## ProductList Component:

This component is responsible for displaying a list of products. It allows users to **filter products by name**, **edit a selected product**, or **delete a product**. It also handles dynamic filtering and rendering of the product list.

**Props:**
1. `products` – An array of product objects to be displayed.
2. `deleteProduct(productId)` – Function to delete a product by its ID.
3. `setEditProduct(product)` – Function to set a selected product for editing (used in the form).

**State Management:**
1. Maintains local state `filters` as an object with:
   - `name` (string) – used to filter the displayed products by name.

2. The list of products is filtered on each render based on the current value of `filters.name`:
    - Matching is case-insensitive.
    - Only products whose `name` includes the entered filter text are displayed.

**Filter Behavior:**
1. A single input field should be provided to filter products by name.
2. As the user types in the input, the filter state updates in real time and the product list is re-rendered accordingly.

**HTML Structure:**
1. **Filter Section**:
    - Includes a label with the text **"Filter by Name:"**.
    - This label is associated with the input using `htmlFor="name"`.
    - The input element has `id="name"` and type `"text"`.
    - The input's value is bound to `filters.name`.
    - On `onChange`, the filter state is updated accordingly.
2. **Product List Section**:
    - Uses an unordered list (`<ul>`) to display the filtered products.
    - For each product, the following is rendered inside a list item (`<li>`):
        ➔ Name: shown using a bold label "Name:"
        ➔ Description: shown using a bold label "Description:"
        ➔ Price: shown using a bold label "Price:"
        ➔ Two buttons:
            ➢ **Edit** button: triggers `setEditProduct(product)` when clicked.
            ➢ **Delete** button: triggers `deleteProduct(product.id)` when clicked.
3. **If no products match the filter criteria:**
    - Display a list item with the message: "No products found".

**\*\* Kindly refer to the screenshots for any clarifications. \*\***

## 4   Validations

- All required fields must be fulfilled with valid data.
- Price field should accept only numeric values.
- In the starting "Add Product" button should be disabled.
- Only after validating fields, "Add Product" button should be enabled.

## 5   Constraints

- You should be able to press the "TAB" key and "SHIFT + TAB" to navigate from top field to bottom field and vice-versa.
- On clicking the "Add Product" button, the particular product should be added.

## 6   Mandatory Assessment Guidelines

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) 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. **This is a web-based application, to run the application on a browser, use the**

internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.

Note: The application will not run in the local browser

8. You can follow series of command to setup React 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:8080/8081 to open project in browser -> takes 2 to 3 min

   c. npm run json-start -> To start the json server. As all data is stored in our db.json file.

   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

9. You may also run "npm run jest" while developing the solution to re-factor the code to pass the test-cases.

10. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.