

NAME	Vikash kr jha
UID	23BCS11391
CLASS	622-A

➤ Full Stack PRACTICE 7.1

- CODE

Server.js :::

```
// backend-api/server.js

const express = require('express');
const cors = require('cors');
const app = express();
const PORT = 5000;

// Enable CORS for all routes and origins
app.use(cors());

// Product Data
const products = [
    { id: 1, name: 'Laptop', price: 1200 },
    { id: 2, name: 'Mouse', price: 25 },
    { id: 3, name: 'Keyboard', price: 45 },
];

// Product List Endpoint
app.get('/api/products', (req, res) => {
    // Simulate a network delay for better demonstration of loading state
    // (optional)
    setTimeout(() => {
```

```

        res.json(products);
    }, 500);
});

// Start the server
app.listen(PORT, () => {
    console.log(`Express API is running on http://localhost:${PORT}`);
});

```

App.js ::::

```

// frontend-app/src/App.js

import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './App.css'; // We'll add the CSS next

// Component to display a single product card
const ProductCard = ({ name, price }) => (
    <div className="product-card">
        <h3>{name}</h3>
        <p>Price: ${price}</p>
        <button className="buy-button">Buy Now</button>
    </div>
);

function App() {
    const [products, setProducts] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        // Function to fetch data
        const fetchProducts = async () => {
            try {
                // 1. Use Axios to make a GET request to the Express API
                const response = await axios.get('http://localhost:5000/api/products');

                // 2. Update state with the fetched data
                setProducts(response.data);
                setError(null); // Clear any previous errors
            } catch (err) {

```

```
// 3. Handle errors (e.g., API is down, network issue)
console.error("Error fetching data:", err);
setError('Failed to fetch products. Is the backend running on port
5000?');
} finally {
    // 4. Update loading state
    setLoading(false);
}
};

fetchProducts();
}, []); // The empty dependency array ensures this runs only once on mount

// --- Rendering Logic ---

if (loading) {
    return (
        <div className="container">
            <h1 className="title">Product List</h1>
            <p className="loading-message">Loading products... ✍</p>
        </div>
    );
}

if (error) {
    return (
        <div className="container">
            <h1 className="title">Product List</h1>
            <p className="error-message">Error: {error}</p>
        </div>
    );
}

return (
    <div className="container">
        <h1 className="title">Product List</h1>
        <div className="product-list-grid">
            {/* Map over the fetched products to display them */}
            {products.map(product => (
                <ProductCard
                    key={product.id}
                    name={product.name}
                    price={product.price}
                />
            )));
        </div>
    </div>
);
```

```
        </div>
    </div>
);
}

export default App;
```

### App.css :::

```
/* frontend-app/src/App.css */

/* General Styling */
body {
    margin: 0;
    font-family: sans-serif;
    background-color: #222; /* Dark background */
    color: white;
}

.container {
    padding: 20px;
    text-align: center;
}

.title {
    font-size: 2em;
    margin-bottom: 40px;
    color: white;
}

/* Product Grid Layout */
.product-list-grid {
    display: flex;
    justify-content: center;
    gap: 30px; /* Space between cards */
    flex-wrap: wrap; /* Allows wrapping on smaller screens */
}

/* Individual Product Card */
.product-card {
    background-color: #333;
    border: 1px solid #444;
    border-radius: 8px;
    padding: 30px 20px;
```

```
width: 250px;
min-height: 180px;
display: flex;
flex-direction: column;
justify-content: space-between;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.4);
transition: transform 0.2s;
}

.product-card:hover {
  transform: translateY(-5px);
  border-color: #666;
}

h3 {
  margin-top: 0;
  font-size: 1.5em;
}

p {
  font-size: 1.2em;
  color: #aaa;
}

/* Buy Button Styling */
.buy-button {
  background-color: #007bff; /* Blue button */
  color: white;
  border: none;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
  font-size: 1em;
  font-weight: bold;
  transition: background-color 0.2s;
}

.buy-button:hover {
  background-color: #0056b3;
}

/* Message Styles */
.loading-message, .error-message {
  padding: 20px;
  font-size: 1.2em;
```

```
    color: #f8d7da; /* Light red for error */
}
.loading-message {
    color: #add8e6; /* Light blue for loading */
}
```

OUTPUT:

