Case Study 1:

E-Commerce Product Listing

(Fake Store API) API: https://fakestoreapi.com/products

Scenario: An online shop wants to display a catalog of products (title, price, image). Users should also be able to select a category (electronics, clothing, jewelry, etc.), and the products should update automatically.

```
import React, { useState, useEffect } from "react";


const ProductListing = () => {
  const [products, setProducts] = useState([]);
  const [category, setCategory] = useState("all");
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // Fetch products based on category
  useEffect(() => {
    const fetchProducts = async () => {
      try {
        setLoading(true);
        setError(null);

        let url =
          category === "all"
            ? "https://fakestoreapi.com/products"
            : `https://fakestoreapi.com/products/category/${category}`;

        const res = await fetch(url);
        if (!res.ok) throw new Error("Failed to fetch");
        const data = await res.json();
```

```
      setProducts(data);
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  };

  fetchProducts();
}, [category]);

return (
  <div style={{ padding: "20px" }}>
    <h2> E-Commerce Store</h2>

    {/* Category Selector */}
    <select
      value={category}
      onChange={(e) => setCategory(e.target.value)}
      style={{ padding: "8px", marginBottom: "20px" }}
    >
      <option value="all">All</option>
      <option value="electronics">Electronics</option>
      <option value="jewelery">Jewelry</option>
      <option value="men's clothing">Men's Clothing</option>
      <option value="women's clothing">Women's Clothing</option>
    </select>
    {error && (
      <div>
        <p style={{ color: "red" }}>{error}</p>
        <button onClick={() => setCategory(category)}>Retry</button>
```

```
 </div>
)}

{/* Loading State */}
{loading && <p>Loading...</p>}

{/* Products */}
<div
 style={{
  display: "grid",
  gridTemplateColumns: "repeat(auto-fill, minmax(200px, 1fr))",
  gap: "20px",
 }}
>
 {!loading &&
  !error &&
  products.map((product) => (
   <div
    key={product.id}
    style={{
     border: "1px solid #ccc",
     padding: "10px",
     borderRadius: "8px",
     textAlign: "center",
    }}
   >
    <img
     src={product.image}
     alt={product.title}
     style={{ width: "100px", height: "100px", objectFit: "contain" }}
    />
```

```
            <h4>{product.title}</h4>
            <p> {product.price}</p>
          </div>
        ))}
      </div>
    </div>
  );
};

export default ProductListing;
```

Case Study 2: Shopping Cart with Product Details (Fake Store API)

API: https://fakestoreapi.com/products/:id

Scenario:

When a user clicks on a product in the catalog, they should be able to see detailed information (title, price, description, rating).

```
import React, { useState, useEffect } from "react";

const ProductApp = () => {
  const [products, setProducts] = useState([]);
  const [selectedId, setSelectedId] = useState(null); // which product user clicked
  const [productDetails, setProductDetails] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);

  // Fetch all products once on load
  useEffect(() => {
    const fetchProducts = async () => {
```

```
    try {
      const res = await fetch("https://fakestoreapi.com/products");
      const data = await res.json();
      setProducts(data);
    } catch (err) {
      console.error(err);
    }
  };
  fetchProducts();
}, []);

// Fetch product details whenever selectedId changes
useEffect(() => {
  if (!selectedId) return; // no product clicked yet

  const fetchDetails = async () => {
    try {
      setLoading(true);
      setError(null);
      const res = await fetch(`https://fakestoreapi.com/products/${selectedId}`);
      if (!res.ok) throw new Error("Failed to fetch details");
      const data = await res.json();
      setProductDetails(data);
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  };

  fetchDetails();
```

```
}, [selectedId]);

return (
  <div style={{ padding: "20px" }}>
    <h2> Fake Store</h2>

    {/* Product List */}
    <div
      style={{
        display: "grid",
        gridTemplateColumns: "repeat(auto-fill, minmax(200px, 1fr))",
        gap: "20px",
        marginBottom: "30px",
      }}
    >
      {products.map((p) => (
        <div
          key={p.id}
          style={{
            border: "1px solid #ccc",
            padding: "10px",
            borderRadius: "8px",
            cursor: "pointer",
          }}
          onClick={() => setSelectedId(p.id)}
        >
          <img
            src={p.image}
            alt={p.title}
            style={{ width: "100px", height: "100px", objectFit: "contain" }}
          />
```

```
      <h4>{p.title}</h4>
      <p> {p.price}</p>
    </div>
  ))}
</div>

{/* Product Details Section */}
{selectedId && (
  <div style={{ marginTop: "20px" }}>
    <h3> Product Details</h3>

    {loading && <p>Loading...</p>}
    {error && <p style={{ color: "red" }}> {error}</p>}

    {productDetails && (
      <div
        style={{
          border: "1px solid #999",
          padding: "20px",
          borderRadius: "8px",
          maxWidth: "500px",
        }}
      >
        <img
          src={productDetails.image}
          alt={productDetails.title}
          style={{ width: "200px", height: "200px", objectFit: "contain" }}
        />
        <h3>{productDetails.title}</h3>
        <p>
          <strong> Price:</strong> {productDetails.price}
```

```
            </p>
            <p>{productDetails.description}</p>
            <p>
              Rating: {productDetails.rating?.rate} (
             {productDetails.rating?.count} reviews)
            </p>
          </div>
        )}
      </div>
    )}
  </div>
 );
};

export default ProductApp;
```

Case Study 3: User Management Dashboard (JSONPlaceholder API)

API: https://jsonplaceholder.typicode.com/users

Scenario: An admin dashboard displays a list of all registered users (name, email, phone). Clicking on a user should show their profile details.

```
import React, { useState, useEffect } from "react";

const UserDashboard = () => {
  const [users, setUsers] = useState([]);
  const [selectedUserId, setSelectedUserId] = useState(null);
  const [userDetails, setUserDetails] = useState(null);
  const [loadingUsers, setLoadingUsers] = useState(true);
  const [loadingDetails, setLoadingDetails] = useState(false);
  const [error, setError] = useState(null);
```

```
// Fetch all users on mount
useEffect(() => {
  const fetchUsers = async () => {
    try {
      setLoadingUsers(true);
      const res = await fetch("https://jsonplaceholder.typicode.com/users");
      if (!res.ok) throw new Error("Failed to fetch users");
      const data = await res.json();
      setUsers(data);
    } catch (err) {
      setError(" Could not load users.");
    } finally {
      setLoadingUsers(false);
    }
  };
  fetchUsers();
}, []);

// Fetch single user details when selectedUserId changes
useEffect(() => {
  if (!selectedUserId) return;

  const fetchUserDetails = async () => {
    try {
      setLoadingDetails(true);
      setError(null);
      const res = await fetch(
        `https://jsonplaceholder.typicode.com/users/${selectedUserId}`
      );
      if (!res.ok) throw new Error("Failed to fetch user details");
      const data = await res.json();
```

```
      setUserDetails(data);
    } catch (err) {
      setError(" Could not load user details.");
    } finally {
      setLoadingDetails(false);
    }
  };

  fetchUserDetails();
}, [selectedUserId]);

return (
  <div style={{ display: "flex", gap: "30px", padding: "20px" }}>
    {/* Left: User List */}
    <div style={{ flex: 1 }}>
      <h2> User Management Dashboard</h2>

      {loadingUsers && <p>Loading users...</p>}
      {error && <p style={{ color: "red" }}>{error}</p>}

      <ul style={{ listStyle: "none", padding: 0 }}>
        {users.map((u) => (
          <li
            key={u.id}
            onClick={() => setSelectedUserId(u.id)}
            style={{
              border: "1px solid #ccc",
              padding: "10px",
              marginBottom: "10px",
              cursor: "pointer",
              borderRadius: "6px",
```

```
          background:
            u.id === selectedUserId ? "#f0f8ff" : "transparent",
        }}
      >
        <strong>{u.name}</strong> <br />
        {u.email} <br />
        {u.phone}
      </li>
    ))}
  </ul>
</div>

{/* Right: User Details */}
<div style={{ flex: 1 }}>
  <h2> User Profile</h2>

  {!selectedUserId && <p>Select a user to view details</p>}
  {loadingDetails && <p>Loading user details...</p>}
  {userDetails && !loadingDetails && (
    <div
      style={{
        border: "1px solid #ccc",
        padding: "20px",
        borderRadius: "8px",
        background: "#fafafa",
      }}
    >
      <h3>{userDetails.name}</h3>
      <p>
        <strong>Username:</strong> {userDetails.username}
      </p>
```

```
          <p>
            <strong>Email:</strong> {userDetails.email}
          </p>
          <p>
            <strong>Phone:</strong> {userDetails.phone}
          </p>
          <p>
            <strong>Website:</strong> {userDetails.website}
          </p>
          <p>
            <strong>Company:</strong> {userDetails.company?.name}
          </p>
          <p>
            <strong>Address:</strong>{" "}
            {`${userDetails.address?.street}, ${userDetails.address?.city}`}
          </p>
        </div>
      )}
    </div>
  </div>
 );
};

export default UserDashboard;
```

Case Study 4: Posts with Comments (JSONPlaceholder API)

API:

- **Posts →** https://jsonplaceholder.typicode.com/posts
- **Comments →** https://jsonplaceholder.typicode.com/posts/1/comments

Scenario: A blog app shows a list of posts. When a user clicks a post, its comments should load below it

```
import React, { useState, useEffect } from "react";


const BlogApp = () => {
  const [posts, setPosts] = useState([]);
  const [selectedPostId, setSelectedPostId] = useState(null);
  const [comments, setComments] = useState([]);
  const [loadingPosts, setLoadingPosts] = useState(true);
  const [loadingComments, setLoadingComments] = useState(false);
  const [error, setError] = useState(null);
  useEffect(() => {
    const fetchPosts = async () => {
      try {
        setLoadingPosts(true);
        const res = await fetch("https://jsonplaceholder.typicode.com/posts");
        if (!res.ok) throw new Error("Failed to fetch posts");
        const data = await res.json();
        setPosts(data.slice(0, 10));
      } catch (err) {
        setError(" Could not load posts.");
      } finally {
        setLoadingPosts(false);
      }
    };
```

```
  fetchPosts();
}, []);

// Fetch comments when post is selected
useEffect(() => {
  if (!selectedPostId) return;

  const fetchComments = async () => {
    try {
      setLoadingComments(true);
      setError(null);
      const res = await fetch(
        `https://jsonplaceholder.typicode.com/posts/${selectedPostId}/comments`
      );
      if (!res.ok) throw new Error("Failed to fetch comments");
      const data = await res.json();
      setComments(data);
    } catch (err) {
      setError(" Could not load comments.");
    } finally {
      setLoadingComments(false);
    }
  };

  fetchComments();
}, [selectedPostId]);

return (
  <div style={{ display: "flex", gap: "30px", padding: "20px" }}>
    {/* Left: Posts List */}
    <div style={{ flex: 1 }}>
```

```
 <h2> Blog Posts</h2>

 {loadingPosts && <p>Loading posts...</p>}
 {error && <p style={{ color: "red" }}>{error}</p>}

 <ul style={{ listStyle: "none", padding: 0 }}>
  {posts.map((post) => (
   <li
    key={post.id}
    onClick={() => setSelectedPostId(post.id)}
    style={{
     border: "1px solid #ccc",
     padding: "10px",
     marginBottom: "10px",
     cursor: "pointer",
     borderRadius: "6px",
     background:
      post.id === selectedPostId ? "#f0f8ff" : "transparent",
    }}
   >
    <strong>{post.title}</strong>
    <p>{post.body.substring(0, 60)}...</p>
   </li>
  ))}
 </ul>
</div>

{/* Right: Comments */}
<div style={{ flex: 1 }}>
 <h2>💬 Comments</h2>
```

```
      {!selectedPostId && <p>Select a post to view comments</p>}
      {loadingComments && <p>Loading comments...</p>}
      {comments.length > 0 && !loadingComments && (
        <ul style={{ listStyle: "none", padding: 0 }}>
          {comments.map((c) => (
            <li
              key={c.id}
              style={{
                border: "1px solid #ddd",
                padding: "10px",
                marginBottom: "10px",
                borderRadius: "6px",
              }}
            >
              <strong>{c.name}</strong> ({c.email})
              <p>{c.body}</p>
            </li>
          ))}
        </ul>
      )}
    </div>
  </div>
 );
};

export default BlogApp;
```

Case Study 5: Todo Tracker (JSONPlaceholder API)

API: https://jsonplaceholder.typicode.com/todos

Scenario: A productivity app shows a list of todos, with completed and pending tasks.

```
import React, { useState, useEffect } from "react";


const TodoTracker = () => {
  const [todos, setTodos] = useState([]);
  const [filter, setFilter] = useState("all");
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  useEffect(() => {
    const fetchTodos = async () => {
      try {
        setLoading(true);
        const res = await fetch("https://jsonplaceholder.typicode.com/todos");
        if (!res.ok) throw new Error("Failed to fetch todos");
        const data = await res.json();
        setTodos(data.slice(0, 20));
      } catch (err) {
        setError(" Could not load todos.");
      } finally {
        setLoading(false);
      }
    };

    fetchTodos();
  }, []);

  const filteredTodos = todos.filter((todo) => {
```

```
  if (filter === "completed") return todo.completed;
  if (filter === "pending") return !todo.completed;
  return true;
});

const completedCount = todos.filter((t) => t.completed).length;
const pendingCount = todos.filter((t) => !t.completed).length;

return (
  <div style={{ padding: "20px", maxWidth: "600px", margin: "auto" }}>
    <h2>Todo Tracker</h2>

    {loading && <p>Loading todos...</p>}
    {error && <p style={{ color: "red" }}>{error}</p>}
    <p>
      Completed: <strong>{completedCount}</strong> | Pending:{" "}
      <strong>{pendingCount}</strong>
    </p>
    <div style={{ marginBottom: "20px" }}>
      <button
        onClick={() => setFilter("all")}
        style={{ marginRight: "10px" }}
      >
        Show All
      </button>
      <button
        onClick={() => setFilter("completed")}
        style={{ marginRight: "10px" }}
      >
        Show Completed Only
      </button>
```

```
      <button onClick={() => setFilter("pending")}>
        Show Pending Only
      </button>
    </div>
    <ul style={{ listStyle: "none", padding: 0 }}>
      {filteredTodos.map((todo) => (
        <li
          key={todo.id}
          style={{
            border: "1px solid #ccc",
            padding: "10px",
            marginBottom: "8px",
            borderRadius: "6px",
            background: todo.completed ? "#d4edda" : "#f8d7da",
          }}
        >
          <strong>{todo.title}</strong>
          <span style={{ float: "right" }}>
            {todo.completed ? " Done" : " Pending"}
          </span>
        </li>
      ))}
    </ul>
  </div>
 );
};

export default TodoTracker;
```