User Story 1: Setup Mock Backend

As a developer, I want to use json-server to simulate backend data so that I can build and test the admin dashboard without needing a real backend.

✅ Acceptance Criteria

A db.json file exists with resource collections (e.g., users, products)

Running json-server starts a local server at http://localhost:3000 (or another port)

Data can be fetched via GET requests in the browser or Postman

Project package.json contains a script like "json-server": "json-server --watch db.json"

🔧 Sub-Issues:

Create db.json with sample data (users, posts, products, etc.)

Install and run json-server (npx json-server --watch db.json)

Setup scripts in package.json to start the server

User Story 2: Build the Dashboard UI Layout

As an admin, I want a dashboard layout with a sidebar and top navigation bar so that I can easily navigate between different data sections.

✅ Acceptance Criteria

A top navbar is present with basic user info

The UI has a sidebar with navigation links (e.g., Users, Products)

A top navbar is present with basic branding or user info

Content area changes based on selected route using react-router-dom

Route structure is working (e.g., /users, /products)

🔧 Sub-Issues:

Create a dashboard layout (header, sidebar, content area)

Set up routing using react-router-dom

Add placeholder pages (e.g., /users, /products, /orders)

User Story 3: View Data Lists

As an admin, I want to view lists of data (e.g., users, products) so that I can review them easily.

✅ Acceptance Criteria

Data is fetched from json-server on component mount

Data is displayed in a table format (MUI, Bootstrap, or plain HTML)

Each row shows relevant fields like ID, name, etc.

If there’s an error or empty data, a message is displayed

Sub-Issues:

Fetch data from json-server (GET /users, etc.)

Display data in tables (using plain HTML and MUI)

Handle loading and error states

**User Story 4:** Edit Data

As an admin, I want to edit data entries from the list view so that I can correct any mistakes.

**✅ Acceptance Criteria**

* Each table row has an “Edit” button
* Clicking Edit opens a form pre-filled with existing data
* On submit, data is updated via PUT /resource/:id
* User sees success or error feedback

🔧 Sub-Issues:

Add an "Edit" button for each row

Show a form pre-filled with existing data

Update data using PUT /resource/:id to json-server

Show success/failure messages

**User Story 5:** Delete Data

As an admin, I want to delete data records that are invalid or outdated.

### ✅ Acceptance Criteria

* Each row has a “Delete” button
* Clicking Delete opens a confirmation dialog or alert
* On confirmation, a DELETE /resource/:id request is sent
* The deleted item is removed from the UI without refreshing the page

Sub-Issues:

Add a "Delete" button next to each item

Confirm before deleting (modal or alert)

Call DELETE /resource/:id

Remove item from UI after successful deletion

**User Story 6:** Add New Records

As an admin, I want to add new data records so that I can manually insert missing information.

### ✅ Acceptance Criteria

* There’s a “New” button or Add icon above the table
* Clicking it shows a blank form
* On form submit, a POST /resource request is sent
* New item appears in the list after successful creation

🔧 Sub-Issues:

Add a "New" button

Show a blank form to enter new details

Send POST request to json-server

Refresh the list after adding

**User Story 9:** Testing the Admin Panel

As a developer, I want to test dashboard features so that I can ensure reliability.

🔧 Sub-Issues:

Unit test data fetch functions

Test form validations and submission logic

Manual testing of create/edit/delete actions

Feature: Simulate backend with json-server

As a developer,

I want to use json-server to simulate backend data,

So that I can build and test the admin dashboard without needing a real backend.

Background:

Given a db.json file exists in the project root

And the file contains resource collections such as "users" and "products"

Scenario: Start json-server and verify it's running locally

When I run the command "npm run json-server"

Then a local server should start at "http://localhost:3000"

And I should see the available routes listed in the terminal

Scenario: Fetch data via GET requests

Given the json-server is running

When I send a GET request to "http://localhost:3000/users"

Then I should receive a JSON response containing a list of users

When I send a GET request to "http://localhost:3000/products"

Then I should receive a JSON response containing a list of products

Scenario: Validate package.json contains a script for json-server

Given the project has a package.json file

Then it should contain a script named "json-server"

And the script should run the command "json-server --watch db.json"

🔀 **Routing:**

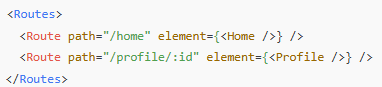
Routing is the **process of defining** the various URLs (or paths) in your application and mapping them to specific components or views.

**Example route settings:**

1. /home → HomePage component.
2. /profile/:id → ProfilePage component.

It happens in the background and determines **what to show** when a user visits a certain path.

**Example (React Router):**



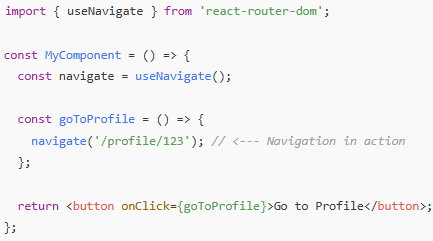
🚗 **Navigation:**

Navigation is the **action of moving** from one route to another.

This can be done programmatically or by user interaction (like clicking a link or button).

It’s the **trigger** that changes the route.

**Example (React Router):**



### 🧠 In Simple Terms:

* **Routing** = Defining **what** to show on each URL
* **Navigation** = **Going** to those URLs (changing the route)

🚗 **useNavigate Hook:**

In React (with **React Router v6+**), the useNavigate hook is used for **programmatic navigation** — that is, navigating the user to another route **without clicking a link**.

**Important:**

useNavigate only works inside **components rendered within <Router>** (like <BrowserRouter>).

It cannot be used in plain JavaScript files or outside the component tree.



`react-router-dom` returns a function that lets you **navigate to another route** from within any component logic (e.g., after form submission or login).

Example:



🛠️ Optional Parameters:



replace: true → Replaces current entry in the history stack (like window.location.replace()), so the user can't go "back" to the previous page.

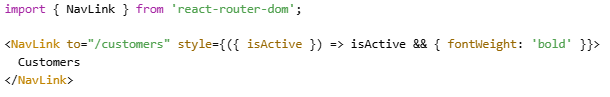
🚗 **NavLink component:**

* **Used in JSX (HTML-like)**
* Used for **navigation menus, sidebars, tabs**
* Can highlight active route (like tab selected)

📌 When to use:

* When rendering clickable links in the UI

**Example:**

****