Assignment: Build a Simple News Portal Using JSON-Server (Frontend + Mock Backend)

1. Overview

In this assignment, you will develop a **basic CRUD News Portal** using **JSON-Server** as a mock backend API. The application must allow users to:

- Log in (simulated)
- Create news posts
- View all news posts
- View details of a specific post including comments
- Add comments to news posts
- Edit and delete news posts

The backend data will be stored in a `db.json` file, and JSON-Server will expose REST endpoints for `/users` and `/news`.

---

2. System Concept

The project simulates a simple news publishing portal:

- **Users** represent authors or commenters.
- Each **News item** is created by an author (`author_id`).
- Each news item contains a list of **comments**, where each comment has a `user_id`.

Since JSON-Server does not provide authentication, a **simulated login page** will be used.

---

3. Requirements

3.1 Login Page (Simulated Authentication)

- Provide a login screen where the user selects their name from a dropdown list.
- Fetch users from `/users` endpoint and populate the dropdown.
- On login:
    - Store the logged-in user in `localStorage`.
    - Redirect to the **News List** page.
- All “create”, “edit”, and “comment” actions must use the logged-in user’s ID.

3.2 News List Page

- Display all news items fetched from `/news`.
- For each news item, show:
    - Title
    - Author name (resolve using `/users`)
- Buttons/links for:
    - **View Details**
    - **Edit** (only if logged-in user is the author)
    - **Delete** (only if logged-in user is the author)

- A **“Create News”** button must be present.
- Display “Logged in as: ”.

3.3 Create News Page

- A form with:
  - News title (text)
  - News body/content (textarea)
- On submit:
  - POST request to `/news`
  - Use the logged-in user’s ID as `author_id`
  - Initialize `comments` as an empty array
- After successful creation → redirect to News List page.

3.4 News Detail Page

- Title
- Body
- Author name
- List of comments (with commenter names)

Add Comment Section

- A textbox to enter comment text.
  When submitted:
  - Append a new comment object to that news item's `comments` array.
  - Use the logged-in user’s ID as `user_id`.
  - Use PATCH request to update `/news/:id`.

Comment Object Structure

```
{
  "id": number,
  "text": "comment text",
  "user_id": number,
  "timestamp": "ISO timestamp"
}
```

3.5 Edit News Page

- Pre-fill the form with existing title and body.
- Only visible if logged-in user is the author.
- Submit changes using PATCH `/news/:id`.

3.6 Delete News

- Only the author can delete the item.
- Use DELETE `/news/:id`.
- Redirect back to News List page.

3.7 JSON-Server Requirements

- Use JSON-Server to serve:
    - `/users`
    - `/news`
- Run with: json-server --watch db.json --port 3000

---

4. REST API Endpoints to Use

Users

Method	Endpoint	Description
GET	`/users`	List all users
GET	`/users/:id`	Get a single user

News

Method	Endpoint	Description
GET	`/news`	List all news
GET	`/news/:id`	Get full news including comments
POST	`/news`	Create news item
PATCH	`/news/:id`	Edit news or comments array
DELETE	`/news/:id`	Delete news item

All comments remain stored **inside** the associated news item (nested structure). A sample database is attached with seeded data.

---

5. Validation Rules

- News title cannot be empty.
- News body must be at least 20 characters.
- Comment text cannot be empty.
- User cannot edit or delete news written by another user.

---

6. Bonus (Optional)

- Search bar to filter news by title.
- Pagination.
- Show number of comments per news.