Example Prompt

Create a single-file, object-oriented PHP script that functions generic data grid for viewing any table in a MySQL database. The script should be self-contained, generating a full HTML page with embedded CSS and JavaScript.

The entire functionality must be encapsulated within a single PHP class called `TableView`.

Key Requirements:

1. Class Structure and Initialization:

- The `TableView` class should handle all logic.
- The constructor (`__construct`) will be the main entry point, orchestrating calls to private methods to set up the environment, process user input, and fetch data.
- A final public method, `render()`, will be called to output the complete HTML, CSS, and JavaScript.

2. Database and Configuration:

- Use **PDO** for all database interactions to ensure security against SQL injection.
- The script must dynamically load database credentials (host, user, password, database) from a configuration file located at `../config/database.txt`.
- It should connect to the database by requiring a separate class file: `require_once __DIR__ . '/../lib/database.php';`.
- It must be ableto **exclude specific tables** from being displayed (e.g., `beers`, `clients`, `form_options`).

3. Dynamic Table Handling:

- At the top of the page, display an HTML `<select>` dropdown that is automatically populated with all available (non-excluded) tables from the database. Changing the selection should reload the page to display the chosen table.
- The script must dynamically determine the **primary key** and fetch all **column names and data types** for the currently selected table using `SHOW KEYS` and `SHOW COLUMNS`.

4. Core Features & User Interface:

- **Column Selection & Reordering:**
  - Provide a "Columns" button that opens a modal window.
  - Inside the modal, list all available columns for the current table with checkboxes.
  - The user must be able to **drag and drop** the columns to reorder them.
  - The user's choice of visible columns and their order must be **saved in a browser cookie**. The cookie name should be unique to the table (e.g., `selected_columns_orders`).
    The cookie is valid for 90 days and when the cookie is read (used) the cookie is set to be valid for 90 days again. So when a table is used within 90 days, the settings will not be lost.
  - If no cookie is set, default to showing the first 5 columns.
- **Sorting:**
  - The header of each displayed column must be clickable to sort the data.
  - Clicking a header should toggle the sort order between `ASC` and `DESC`.
  - The current sort column and order must be reflected in the URL (e.g., `?sort=column_name&order=ASC`) and visually indicated with an arrow (↑ or ↓) in the header.
- **Searching:**
  - Immediately below the header row, include a row of text inputs, one for each displayed column.
  - Typing in these inputs and clicking a "Search" button should filter the results using a `LIKE '%...%'` query for each respective column. Search terms must be maintained across page loads via URL parameters.
- **Pagination:**
  - Display a fixed number of rows per page (default value is 200).
  - Automatically generate pagination links at the bottom of the page if the total number of records exceeds the limit.
- **Action Links:**
  - For a predefined list of "actionable" tables (e.g., `'orders'`), the first column containing the row number should also be a link. On hover, the number should fade out and a "tool" emoji (□□) should appear, linking to a corresponding action page (e.g., `action_orders.php?id=PRIMARY_KEY_VALUE`).

5. HTML, CSS, and JavaScript:

- The `render()` method must generate the entire HTML document structure.
- All **CSS must be embedded** in a `<style>` block in the `<head>`. The styling should be clean, modern, and user-friendly, with clear hover states and a professional look for the modal and table.
- All **JavaScript must be embedded** in a `<script>` tag before the closing `</body>` tag. It should handle the sort-by-column URL generation, the modal's open/close logic, and the logic to capture the reordered columns from SortableJS to be submitted in a hidden form field.

5. GUI specifications

Revision #1
Created 2025-07-13 20:24:48 UTC by Max
Updated 2025-07-13 20:32:25 UTC by Max