**Task – 01 to create one table on HTML page and fetch JSON data from API and retrieve in table**

**Step 1: Fetch Data from the API**

1. Use the provided API endpoint `[https://randomuser.me/api/?results=${rowCount}](https://randomuser.me/api/?results=$%7browCount%7d)` to get the data.
2. Make an HTTP GET request to this URL to retrieve the JSON data.

**Step 2: Understand the JSON Data**

1. Examine the JSON response from the API to understand its structure.
2. Identify the fields you want to display in your table. For example, you might want to include fields like name, email, location, etc.

**Step 3: Create the HTML Structure**

1. Set up a basic HTML file.
2. Add a <table> element where the data will be displayed.

**Step 4: Write JavaScript to Fetch and Display Data**

1. Use JavaScript (or a library like jQuery) to fetch the JSON data from the API.
2. Parse the JSON response to extract the required fields.
3. Dynamically create table rows (<tr>) and cells (<td>) for each entry in the JSON data and insert them into the table.

**Step 5: Test**

**\*\* Additional Requirements version-2.0**

**Step 6: fetch records using Nodejs GET API request**

**Step 7:** **Implement pagination (**explain with example**)**

* + First by default if we run code 10 records displayed on page
  + If user fetch 100 Records then then split page into Fetch-Records/total records

i.e 100/1000 = 10 pages

* + 10 pages numbers(buttons) shows at bottom

Format that shows buttons i.e previous 1,2,3,…9 next

* + UI requirements matches to sir’s requirements
  + Implemented pagination for the user table view to navigate through multiple pages of data.
  + Provided UI controls to navigate between pages and adjust the number of records per page.

**\*\* Additional Requirements version- 3.0**

**Step 8**: **HTML Page Creation for Form and Table View**:

* Created an HTML page for displaying a table of user data with pagination.

**Step 9:** **Form Implementation**:

* Implemented a form with various fields for user data entry.
* Added proper styling for the form to ensure a good user experience.

**Step 10:** **API Integration**:

* Integrated the form with a POST API endpoint to submit new user data.
* Integrated the table view with a GET API endpoint to fetch and display user data.

**Step 11:** **Local Data Handling**:

* Adjusted the implementation to use local storage for user data, rather than relying on a public API.

**Step 12:** **Navigation**:

* Ensured smooth navigation between the form and the table view pages, automatically redirecting to the table view after form submission.