# EE4216 Lab 2 – Reactive Web Application Design

(all checkpoints require a verification in the lab)

**Objectives:**

- Learn to create reactive UI with Vue
- Learn to integrate networking functions into Vue

Unlike server-side rendered apps that need expensive roundtrips, a reactive web quickly reacts to user interaction. The pressure on the backend servers is also decreased, making these apps scale much better under heavy loads, all leading to a smoother user experience.

Dynamic data table allows information to be passed and manipulated dynamically. In this lab, you will be guided to build a dynamic data table step by step using Vue. At start, you are given a Vue base app that contains a complete CSS and a script of the Vue root instance as follows:

```css
/* style of table */
table {
  text-align: left;
  font-family: 'Open Sans', sans-serif;
  width: 500px;
  border-collapse: collapse;
  border: 2px solid #444777;
  margin: 10px;
}

table th {
  background: #444777;
  color: #FFF;
  padding: 5px;
  min-width: 30px;
}

table td {
  padding: 5px;
  border-right: 2px solid #444777;
}

table tbody tr:nth-child(2n) {
  background: #D4D8F9;
}
```

```javascript
/* Vue root instance */
var app = new Vue({
  el: '#app',
  data: {
    columns: {
      id: 'ID',
      name: 'Full Name',
      phone: 'Phone',
    },
    rows: [
      { id: 1, name: "Test Name 1", phone: '305-917-1301'},
      { id: 2, name: "Test Name 2", phone: '210-684-8953'},
      { id: 3, name: "Test Name 3", phone: '765-338-0312'},
    ]
  }
});
```

**Task 1 - Creating a Vue Table Template**

From the script above, two sets of test data – ***columns*** and ***rows*** – are used to define the structure and content of the table shown below. You are asked to create this HTML table using Vue template. You do not need to modify the given CSS and JS in this task.

1. Mount your Vue app instance to a  container in your HTML.
2. Use `v-for` directive to iterate through the columns and rows array to generate the table header and body respectively.

| ID | Full Name | Phone |
|---|---|---|
| 1 | Test Name 1 | 305-917-1301 |
| 2 | Test Name 2 | 210-684-8953 |
| 3 | Test Name 3 | 765-338-0312 |

**Checkpoint 1:**

Demonstrate your HTML code and the static table.

**Task 2 – Adding Dynamic Content to Table**

In the previous task, the rows of the table are preset for testing purpose. Now you are asked to load the table with dynamic user data from a remote API endpoint. In this task, you only need to modify the JS script file. User data can be obtained from this endpoint: https://jsonplaceholder.typicode.com/users

1. Write a function to asynchronously fetch the user data from the above endpoint and update the table rows.
2. Hook the function up to the Vue instance such that the data is loaded into the table when it is displayed.

| ID | Full Name | Phone |
|---|---|---|
| 1 | Leanne Graham | 1-770-736-8031 x56442 |
| 2 | Ervin Howell | 010-692-6593 x09125 |
| 3 | Clementine Bauch | 1-463-123-4447 |
| 4 | Patricia Lebsack | 493-170-9623 x156 |
| 5 | Chelsey Dietrich | (254)954-1289 |
| 6 | Mrs. Dennis Schulist | 1-477-935-8478 x6430 |
| 7 | Kurtis Weissnat | 210.067.6132 |
| 8 | Nicholas Runolfsdottir V | 586.493.6943 x140 |
| 9 | Glenna Reichert | (775)976-6794 x41206 |
| 10 | Clementina DuBuque | 024-648-3804 |

The content of the table is loaded from *JsonPlaceholder*.

**Checkpoint 2:**

Demonstrate your JavaScript code and the dynamic table.

**Task 3 – Manipulating Data in Table**

You are asked to add a sorting function to the table in this task. When a column header is first clicked, the table will be sorted by that column in ascending order. Subsequent clicks will toggle the sorting order inversely.

1. Write a function to sort a 2D array (table rows) by a given key (column).
2. Attach the sort function to the Vue instance.
3. Use `v-on:click` directive to trigger the sort function.
4. Use `v-bind:class` directive to append a small arrow (up and down) to the header to indicate the current ordering. The arrow can be made with this CSS trick: https://css-tricks.com/snippets/css/css-triangle/.

| ID | Full Name ▲ | Phone |
|---|---|---|
| 5 | Chelsey Dietrich | (254)954-1289 |
| 10 | Clementina DuBuque | 024-648-3804 |
| 3 | Clementine Bauch | 1-463-123-4447 |
| 2 | Ervin Howell | 010-692-6593 x09125 |
| 9 | Glenna Reichert | (775)976-6794 x41206 |
| 7 | Kurtis Weissnat | 210.067.6132 |
| 1 | Leanne Graham | 1-770-736-8031 x56442 |
| 6 | Mrs. Dennis Schulist | 1-477-935-8478 x6430 |
| 8 | Nicholas Runolfsdottir V | 586.493.6943 x140 |
| 4 | Patricia Lebsack | 493-170-9623 x156 |

The table is sorted in ascending order by the name column.

| ID ▼ | Full Name | Phone |
|---|---|---|
| 10 | Clementina DuBuque | 024-648-3804 |
| 9 | Glenna Reichert | (775)976-6794 x41206 |
| 8 | Nicholas Runolfsdottir V | 586.493.6943 x140 |
| 7 | Kurtis Weissnat | 210.067.6132 |
| 6 | Mrs. Dennis Schulist | 1-477-935-8478 x6430 |
| 5 | Chelsey Dietrich | (254)954-1289 |
| 4 | Patricia Lebsack | 493-170-9623 x156 |
| 3 | Clementine Bauch | 1-463-123-4447 |
| 2 | Ervin Howell | 010-692-6593 x09125 |
| 1 | Leanne Graham | 1-770-736-8031 x56442 |

The table is sorted in descending order by the ID column.

**Checkpoint 3:**

Demonstrate your HTML/JavaScript code and the sorting function of the data table.

END OF THE LAB