1. Buatlah 1 struktur json array object yang berisi informasi, terserah anda.
   [
   { "id": 1, "name": "Alice", "job": "Engineer" },
   { "id": 2, "name": "Bob", "job": "Designer" },
   { "id": 3, "name": "Charlie", "job": "Writer" },
   { "id": 4, "name": "Diana", "job": "Doctor" }
   ]
   penjelasan :
   - id: ID unik untuk setiap orang
   - name: Nama orang tersebut
   - job: Pekerjaan atau profesi orang tersebut

2. Tampilkan 1 label dan buatlah 1 tombol yang bisa merubah value label. Label adalah value dari json yang sudah anda bikin di soal no 1.

```
<h3>Soal 1 & 2</h3>
<h2>Name: <span id="nameLabel">-</span></h2>
<h2>Job: <span id="jobLabel">-</span></h2>
<br><br>
<button class="btn btn-primary" onclick="changeRandom()">Ubah</button>
<button class="btn btn-primary" onclick="resetLabel()">Reset</button>

<script>
  const people = [
    { id: 1, name: "Alice", job: "Engineer" },
    { id: 2, name: "Bob", job: "Designer" },
    { id: 3, name: "Charlie", job: "Writer" },
    { id: 4, name: "Diana", job: "Doctor" }
  ];
  const nameLabel = document.getElementById('nameLabel');
  const jobLabel = document.getElementById('jobLabel');
  nameLabel.textContent = people[0].name;
  jobLabel.textContent = people[0].job;

  function changeRandom() {
    const randomIndex = Math.floor(Math.random() * people.length);
    const selected = people[randomIndex];
    document.getElementById('nameLabel').textContent = selected.name;
    document.getElementById('jobLabel').textContent = selected.job;
  }

  function resetLabel() {
    nameLabel.textContent = people[0].name;
    jobLabel.textContent = people[0].job;
  }
</script>
```

penjelasan :

Label dan digunakan untuk menampilkan

nama dan pekerjaan.

Tombol "Ubah" akan menampilkan data secara acak dari array people.

Tombol "Reset" akan mengembalikan nilai ke data pertama (yaitu Alice).

Semua data diambil dari array JSON yang sudah dibuat sebelumnya.

3. Buatlah 1 fitur yang berupa http request ke url : http://jsonplaceholder.typicode.com/posts dan cetak responsenya pada console log
4. Buatlah 1 html dengan mencetak hasil response dari soal no 3 ke dalam bentuk table. Tampilkan maksimal 10 data.
5. Buatlah 1 function untuk menghapus salah satu data pada soal no 4.
6. Hapuslah salah satu key dari object pada json response soal no 4.

Soal no 4-6

```
<!DOCTYPE html>
<html><head>
  <title>SoalUtama</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css
">
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">SoalUtama</a>
  </div>
</nav>

<script>
  function handleLogout() {
    localStorage.clear();
    location.reload(); // refresh the page to reflect logout
  }

  document.addEventListener("DOMContentLoaded", () => {
    const username = localStorage.getItem("username");
    const loginBtn = document.getElementById("loginBtn");
    const logoutBtn = document.getElementById("logoutBtn");
    const navbarWelcome = document.getElementById("navbarWelcome");

    if (username) {
      loginBtn.classList.add("d-none");
      logoutBtn.classList.remove("d-none");
      navbarWelcome.classList.remove("d-none");
      navbarWelcome.textContent = `Halo, ${username}`;
    } else {
      loginBtn.classList.remove("d-none");
      logoutBtn.classList.add("d-none");
      navbarWelcome.classList.add("d-none");
```

```
    }
  });
</script>

  <div class="container mt-4">
    <h3>Soal 3-6</h3>

<button onclick="fetchPosts()" class="btn btn-success mb-2">Ambil
Data</button>
<button onclick="hapusKeyBody()" class="btn btn-warning mb-2">Hapus Key
'body'</button>

<div class="mb-3">
  <label for="limitSelect" class="form-label">Jumlah item per
halaman:</label>
  <select id="limitSelect" class="form-select w-auto d-inline"
onchange="updateLimit()">
    <option value="5">5</option>
    <option value="10" selected="">10</option>
    <option value="20">20</option>
  </select>
</div>

<table class="table table-bordered">
  <thead>
    <tr><th>ID</th><th>Title</th><th>body</th><th>Aksi</th></tr>
  </thead>
  <tbody id="postTableBody"></tbody>
</table>

<nav>
  <ul id="pagination" class="pagination"></ul>
</nav>
<script>
  let dataGlobal = [];
  let currentPage = 1;
  let itemsPerPage = 10;
  let hasBody = true;

  async function fetchPosts() {
    const res = await fetch('https://jsonplaceholder.typicode.com/posts');
    let data = await res.json();
    console.log('Response:', data);
    dataGlobal = data;
    hasBody = true;
    currentPage = 1;
    renderTable();
  }
```

```
  function renderTable() {
    const start = (currentPage - 1) * itemsPerPage;
    const end = start + itemsPerPage;
    const dataToShow = dataGlobal.slice(start, end);

    const thead = document.querySelector('thead tr');
    thead.innerHTML = `<th>ID</th><th>Title</th>` +
      (hasBody ? `<th>Body</th>` : ``) +
      `<th>Aksi</th>`;

    const tbody = document.getElementById('postTableBody');
    tbody.innerHTML = '';

    dataToShow.forEach((d, i) => {
      tbody.innerHTML += `<tr>
        <td>${d.id}</td>
        <td>${d.title}</td>` +
        (hasBody ? `<td>${d.body}</td>` : ``) +
        `<td><button onclick="hapusData(${start + i})" class="btn btn-danger
btn-sm">Hapus</button></td>
      </tr>`;
    });

    renderPagination();
  }

  function renderPagination() {
    const totalPages = Math.ceil(dataGlobal.length / itemsPerPage);
    const pagination = document.getElementById('pagination');
    pagination.innerHTML = '';

    for (let i = 1; i <= totalPages; i++) {
      pagination.innerHTML += `
        <li class="page-item ${i === currentPage ? 'active' : ''}">
          <button class="page-link" onclick="changePage(${i})">${i}</button>
        </li>`;
    }
  }

  function changePage(page) {
    currentPage = page;
    renderTable();
  }

  function updateLimit() {
    itemsPerPage = parseInt(document.getElementById("limitSelect").value);
    currentPage = 1;
    renderTable();
  }
```

```
  function hapusData(index) {
    dataGlobal.splice(index, 1);
    if (currentPage > Math.ceil(dataGlobal.length / itemsPerPage)) {
      currentPage--;
    }
    renderTable();
  }

  function hapusKeyBody() {
    dataGlobal = dataGlobal.map(({ body, ...rest }) => rest);
    console.log('after removing body:', dataGlobal);
    hasBody = false;
    renderTable();
  }
</script>

  </div>


</body></html>
```

Penjelasan soal nomer 3 :

```
async function fetchPosts() {
    const res = await fetch('https://jsonplaceholder.typicode.com/posts');
    let data = await res.json();
    console.log('Response:', data);
    dataGlobal = data;
    hasBody = true;
    currentPage = 1;
    renderTable();
  }
```

- Fungsi fetchPosts() membuat **HTTP GET request** menggunakan fetch ke URL https://jsonplaceholder.typicode.com/posts.
- Response diubah ke format JSON dengan await res.json().
- Data hasil fetch disimpan ke variabel global dataGlobal.
- Data juga dicetak ke console.log untuk melakukan check
- Setelah data diambil, tabel ditampilkan melalui renderTable().

Penjelasan soal no.4 :
html

```
<table class="table table-bordered">
  <thead>
    <tr><th>ID</th><th>Title</th><th>body</th><th>Aksi</th></tr>
  </thead>
  <tbody id="postTableBody"></tbody>
</table>
```

Js

```
  function renderTable() {
```

```
    const start = (currentPage - 1) * itemsPerPage;
    const end = start + itemsPerPage;
    const dataToShow = dataGlobal.slice(start, end);

    const thead = document.querySelector('thead tr');
    thead.innerHTML = `<th>ID</th><th>Title</th>` +
      (hasBody ? `<th>Body</th>` : ``) +
      `<th>Aksi</th>`;

    const tbody = document.getElementById('postTableBody');
    tbody.innerHTML = '';

    dataToShow.forEach((d, i) => {
      tbody.innerHTML += `<tr>
        <td>${d.id}</td>
        <td>${d.title}</td>` +
        (hasBody ? `<td>${d.body}</td>` : ``) +
        `<td><button onclick="hapusData(${start + i})" class="btn btn-danger
btn-sm">Hapus</button></td>
      </tr>`;
    });

    renderPagination();
  }
```

- HTML sudah menyiapkan struktur tabel (<thead>, <tbody>).
- JavaScript dalam fungsi renderTable() akan menampilkan 10 data pertama (itemsPerPage = 10).
- Mengambil data dari dataGlobal dan ditampilkan baris per baris menggunakan forEach.
- Bagian <td> hanya menampilkan field id, title, dan body, sesuai struktur data dari API.
- Terdapat juga tombol "Hapus" per baris untuk soal no 5.

Penjelasan soal no.5

```
function hapusData(index) {
    dataGlobal.splice(index, 1);
    if (currentPage > Math.ceil(dataGlobal.length / itemsPerPage)) {
      currentPage--;
    }
    renderTable();
  }
```

- Fungsi hapusData(index) akan menghapus item dari dataGlobal berdasarkan index.
- splice(index, 1) akan menghapus **1 item pada posisi tertentu**.
- Fungsi ini dipanggil oleh tombol “Hapus” pada setiap baris tabel:

<button onclick="hapusData(${start + i})">Hapus</button>

- Setelah menghapus, tabel akan diperbarui dengan renderTable().

Penjelasan soal no 6

```
function hapusKeyBody() {
  dataGlobal = dataGlobal.map(({ body, ...rest }) => rest);
  console.log('after removing body:', dataGlobal);
  hasBody = false;
  renderTable();
}
```

Penjelasan:

- Fungsi hapusKeyBody() akan menghapus properti body dari setiap objek pada array dataGlobal.
- Gunakan **destructuring dengan sisa properti (...rest)** untuk membuat array baru tanpa properti body.
- hasBody = false digunakan untuk mengontrol apakah kolom body ditampilkan di tabel atau tidak.
- Setelah itu tabel diperbarui lagi dengan renderTable().

7. Buatlah function hashing dari string berikut menggunakan sha256: tanggalhariini+namadepananda+pria+ifabula. Contoh : 01112018kenpriaifabula. Hasil hashing akan di cetak pada console log

```
function sha256(message) {
    const ROTR = (x, n) => (x >>> n) | (x << (32 - n));
    const MAX_UINT32 = 2 ** 32;

    const isPrime = n => {
        for (let i = 2, sqrt = Math.sqrt(n); i <= sqrt; i++) {
            if (n % i === 0) return false;
        }
        return true;
    };

    const fracToUint32 = n => ((n - Math.floor(n)) * MAX_UINT32) | 0;
    const H = [], K = [];
    for (let i = 2, primesFound = 0; primesFound < 64; i++) {
        if (isPrime(i)) {
            if (primesFound < 8) H.push(fracToUint32(Math.sqrt(i)));
            K.push(fracToUint32(Math.cbrt(i)));
            primesFound++;
        }
    }
    const msgBytes = [];
    for (let i = 0; i < message.length; i++) {
        const code = message.charCodeAt(i);
        if (code > 255) throw new Error("Only ASCII supported");
        msgBytes.push(code);
    }
```

```
    msgBytes.push(0x80);
    while ((msgBytes.length % 64) !== 56) msgBytes.push(0x00);
    const bitLen = message.length * 8;
    const lenHi = Math.floor(bitLen / MAX_UINT32);
    const lenLo = bitLen >>> 0;


    for (let i = 3; i >= 0; i--) msgBytes.push((lenHi >>> (i * 8)) & 0xff);
    for (let i = 3; i >= 0; i--) msgBytes.push((lenLo >>> (i * 8)) & 0xff);

    const words = new Uint32Array(msgBytes.length / 4);
    for (let i = 0; i < words.length; i++) {
        words[i] =
            (msgBytes[i * 4] << 24) |
            (msgBytes[i * 4 + 1] << 16) |
            (msgBytes[i * 4 + 2] << 8) |
            (msgBytes[i * 4 + 3]);
    }

    const w = new Uint32Array(64);
    const hash = H.slice();

    for (let i = 0; i < words.length; i += 16) {
        for (let t = 0; t < 16; t++) w[t] = words[i + t];
        for (let t = 16; t < 64; t++) {
            const s0 = ROTR(w[t - 15], 7) ^ ROTR(w[t - 15], 18) ^ (w[t - 15]
>>> 3);
            const s1 = ROTR(w[t - 2], 17) ^ ROTR(w[t - 2], 19) ^ (w[t - 2]
>>> 10);
            w[t] = (w[t - 16] + s0 + w[t - 7] + s1) >>> 0;
        }

        let [a, b, c, d, e, f, g, h] = hash;

        for (let t = 0; t < 64; t++) {
            const S1 = ROTR(e, 6) ^ ROTR(e, 11) ^ ROTR(e, 25);
            const ch = (e & f) ^ (~e & g);
            const temp1 = (h + S1 + ch + K[t] + w[t]) >>> 0;
            const S0 = ROTR(a, 2) ^ ROTR(a, 13) ^ ROTR(a, 22);
            const maj = (a & b) ^ (a & c) ^ (b & c);
            const temp2 = (S0 + maj) >>> 0;

            [a, b, c, d, e, f, g, h] = [
                (temp1 + temp2) >>> 0,
                a, b, c,
                (d + temp1) >>> 0,
                e, f, g
            ];
        }
```

```
        hash[0] = (hash[0] + a) >>> 0;
        hash[1] = (hash[1] + b) >>> 0;
        hash[2] = (hash[2] + c) >>> 0;
        hash[3] = (hash[3] + d) >>> 0;
        hash[4] = (hash[4] + e) >>> 0;
        hash[5] = (hash[5] + f) >>> 0;
        hash[6] = (hash[6] + g) >>> 0;
        hash[7] = (hash[7] + h) >>> 0;
    }

    return hash.map(x => x.toString(16).padStart(8, '0')).join('');
}

async function generateHash(inputString) {
    const buffer = new TextEncoder().encode(inputString);
    const digest = await crypto.subtle.digest("SHA-256", buffer);
    const hashArray = Array.from(new Uint8Array(digest));
    const hashHex = hashArray.map(b => b.toString(16).padStart(2,
'0')).join('');
    console.log("Hash:", hashHex);
}
const firstName = "rudhi";
const today = new Date();
const dd = String(today.getDate()).padStart(2, '0');
const mm = String(today.getMonth() + 1).padStart(2, '0');
const yyyy = today.getFullYear();
const inputString = `${dd}${mm}${yyyy}${firstName}priaifabula`;
generateHash(inputString);
console.log(`hashManual: ${sha256(inputString)}`);
```

Penjelasan :
saya membuat dua fungsi untuk melakukan hashing menggunakan algoritma **SHA-256**:

1. sha256(message) – implementasi manual algoritma SHA-256 menggunakan JavaScript.
2. generateHash(inputString) – implementasi hashing menggunakan **Web Crypto API** bawaan browser atau Node.js modern.

Implentasi secara manual (function sha256)

- ROTR(x, n): operasi bitwise rotasi ke kanan.
- H[]: 8 nilai awal dari hash SHA-256 (akar kuadrat dari 8 bilangan prima pertama).
- K[]: 64 konstanta (akar kubik dari bilangan prima).
- fracToUint32(n): mengubah angka pecahan menjadi bilangan 32-bit.
- Mengubah input menjadi array byte (msgBytes).
- Menambahkan padding (mengikuti aturan SHA-256).
- Menambahkan panjang bit dari pesan asli.

- Dipecah jadi 16 word (masing-masing 32-bit), lalu diperluas ke 64 word.
- Loop sebanyak 64 kali untuk melakukan komputasi logika SHA-256.
- Mengupdate 8 variabel hash utama setiap blok (a sampai h).
- Hasil akhir adalah gabungan semua hash (dalam bentuk heksadesimal).

Implementasi dengan Web Crypto API bawaan generateHash(inputString)

- TextEncoder().encode() → ubah string ke buffer (byte array).
- crypto.subtle.digest("SHA-256", buffer) → hitung SHA-256 secara **asinkron**.
- Konversi hasil ArrayBuffer menjadi array byte.
- Ubah ke heksadesimal dan console.log() hasilnya.

8. Cobalah melakukan debuging pada file “testdebug.html” dan perbaiki file tersebut kemudian jabarkan errornya di line mana saja.
    1. **Tidak Ada Koma Antar Objek Array**
       Dua objek JSON dideklarasikan secara berurutan tanpa dipisahkan koma, menyebabkan SyntaxError saat parsing:

       {
       "attribute": "case.CountryOfContract",
       "valueAsString": 1f087fa1-4b87-485e-88f5-de581e9440b7"
       }
       {
       "attribute": "case.ClientPresent",
       "valueAsString": "fc7610f9-7c80-4182-b815-a82bff28524c"
       }
       Solusi :
       {
       "attribute": "case.CountryOfContract",
       "valueAsString": 1f087fa1-4b87-485e-88f5-de581e9440b7"
       },
       {
       "attribute": "case.ClientPresent",
       "valueAsString": "fc7610f9-7c80-4182-b815-a82bff28524c"
       }
    2. **Properti Tanpa Nilai**
       Salah satu elemen JSON tidak memiliki nilai valueAsString, menyebabkan undefined saat digunakan:

       {
       "attribute": "case.life[0].product[0].benefit[12].amount",
       "valueAsString":
       }
       Solusi :
       tambahakan nilai pada valueAsString setidaknya “” jika memang nullable
       {
           "attribute": "case.life[0].product[0].benefit[12].amount",
           "valueAsString": "de701835-42fc-40ae-b31e-b2388167a1d2"
       }
    3. **String Tidak Ditutup**
       Terdapat string yang tidak memiliki tanda kutip penutup, contohnya:

       "attribute": "case.life[0].product[0].benefit[10].amount,

Solusi : tutup dengan tanda kutip
"attribute": "case.life[0].product[0].benefit[10].amount”,

4. **Masalah pada Fungsi compareDeep2**

    a. **Salah Akses Properti**
    Baris var tmpStrb = b.attributes; seharusnya menggunakan b.attribute.

    b. **Syntax Error dalam Loop**
    Pemanggilan cleanA[].slice(...) tidak valid dan menyebabkan error.
    Seharusnya ditulis:

    parseInt(cleanA[i].slice(1, cleanA[i].length));

    c. **Pemanggilan Fungsi dengan Parameter Tidak Lengkap**
    Fungsi compare() dipanggil hanya dengan satu parameter:
    return compare(tmpStra);
    Padahal membutuhkan dua parameter: compare(tmpIgA, tmpIgB).

5. Masalah pada logika compareDeep.
    a. Fungsi hanya mengambil elemen ke-4 dari path (index ke-3).
    b. Jika struktur path berubah (misalnya kurang dari 5 segmen, atau informasi penting bukan di index ke-3), maka hasilnya salah atau tidak relevan.
    c. tmpStra[3].match(/\d/g) bisa mengembalikan null jika tidak ada angka.
    d. Pemanggilan .join("") pada null akan menyebabkan error (TypeError: Cannot read properties of null).
    e. Hanya membandingkan satu bagian dari path. Jadi jika dua objek punya path berbeda di segmen lain, itu diabaikan.
6. Masalah pada logika compareDeep2.
    a. Fungsi hanya melihat angka di dalam [], seperti [0], [2].
    b. Tidak memperhatikan nama segmen seperti user[1].data[2] vs admin[1].data[2] — padahal itu mungkin berarti berbeda secara semantik.
    c. Jika semua elemen dibandingkan dan setara, tidak ada return 0 di akhir fungsi.
    d. → Hasil undefined, yang bisa membuat .sort() tidak stabil.
    e. Jika path seperti "user.info.name", maka match(/\[\d+/g) akan mengembalikan null.
    f. Kasus seperti ini diperlakukan sebagai "lebih kecil" dari path yang mengandung [], tanpa mempertimbangkan konteks yang sebenarnya.
    g. Variabel a di-loop menimpa parameter a. Ini bisa menimbulkan bug tak disengaja, terutama di code lebih besar.

Solusi masalah 5 dan 6 :
membuat function splitpath yang digunakan untuk **memecah string path menjadi unit-unit logis (segmen dan indeks)**, agar bisa dibandingkan secara struktur, bukan hanya sebagai string biasa. Kemudian gunakan loop dengan length terpanjang, setelah itu compare nama terlebih dahulu dan return agar function berhenti, jika compare namenya sama baru gunakan indexnya. Dengan ini kita dapat Menentukan mana bagian nama, mana bagian indeks, dan juga Menghindari error parsing saat segmen tidak punya [].

function compareDeepNew(a, b) {

```
  const pathA = splitPath(a.attribute);
  const pathB = splitPath(b.attribute);

  const maxLen = Math.max(pathA.length, pathB.length);

  for (let i = 0; i < maxLen; i++) {
    const segA = pathA[i] || { name: "", index: -1 };
    const segB = pathB[i] || { name: "", index: -1 };

    const nameCompare = segA.name.localeCompare(segB.name);
    if (nameCompare !== 0) return nameCompare;

    if (segA.index !== segB.index) {
      return segA.index - segB.index;
    }
  }

  return 0;
}

function splitPath(path) {
  const segments = path.split(".");
  return segments.map(seg => {
    const match = seg.match(/^([^\[\]]+)(?:\[(\d+)\])?$/);
    return {
      name: match[1],
      index: match[2] !== undefined ? parseInt(match[2]) : -1
    };
  });
}
```

9. Buatlah satu halaman :
    a. 1 textbox username
    b. 1 textbox password
    c. 1 tombol login
    d. 1 tombol logout
    e. 1 label “selamat datang”

    Buatlah logic login dengan store data username dan password kedalam localStorage. Pertama2 hide tombol logout dan label selamat datang. Jika sudah melakukan login kemudian hide textbox username dan password beserta tombol login, munculkan label selamat datang plus username yang di login dan tombol logout. Ketika logout mohon di bersihkan localStoragenya.

```
<h3>Soal 9 - Login</h3>

<div id="loginSection">
  <input id="username" class="form-control mb-2" placeholder="Username">
  <input id="password" type="password" class="form-control mb-2"
placeholder="Password">
  <button onclick="login()" class="btn btn-primary">Login</button>
</div>

<div id="welcomeSection" style="display:none;">
  <label id="welcomeLabel"></label><br>
  <button onclick="logout()" class="btn btn-danger">Logout</button>
</div>

<script>
  function login() {
    const user = document.getElementById('username').value;
    const pass = document.getElementById('password').value;
    localStorage.setItem('username', user);
    localStorage.setItem('password', pass);
    window.location.reload();
  }

  function logout() {
    localStorage.clear();
    window.location.reload();
  }

  window.onload = () => {
    const user = localStorage.getItem('username');
    if (user) {
      document.getElementById('welcomeLabel').innerText = `Selamat datang,
${user}`;
      document.getElementById('loginSection').style.display = 'none';
      document.getElementById('welcomeSection').style.display = 'block';
    }
```

```
  }
</script>
```

1 textbox username

- Element: <input id="username">
- Fungsinya untuk memasukkan nama pengguna.

1 textbox password

- Element: <input id="password" type="password">
- Digunakan untuk memasukkan kata sandi pengguna.

1 tombol login

- Element: <button onclick="login()">Login</button>
- Ketika diklik:
    - Ambil nilai username dan password.
    - Simpan ke localStorage.
    - Reload halaman untuk memperbarui tampilan.

tombol logout

- Element: <button onclick="logout()">Logout</button>
- Awalnya disembunyikan.
- Setelah login, tombol ini muncul.
- Jika diklik:
    - Menghapus data localStorage.
    - Reload halaman ke tampilan awal.

label “selamat datang”

- Element: <label id="welcomeLabel"></label>
- Awalnya disembunyikan.
- Setelah login, akan menampilkan teks:
    - “Selamat datang, [username]”

Logika Login (Fungsi login())

- Ambil username dan password dari input.
- Simpan ke localStorage.
- Reload halaman agar data ditampilkan.

Logika Logout (Fungsi logout())

- Menghapus semua data dari localStorage.
- Reload halaman ke tampilan awal.

Logika Saat Halaman Dibuka (window.onload)

- Cek apakah username sudah tersimpan.
- Jika ada:
    - Sembunyikan form login.
    - Tampilkan pesan "Selamat datang" dan tombol logout.

10. Buatlah project nodejs menggunakan express. Buat 2 API dengan 2 method yang berbeda yaitu “GET” dan “POST”. Data request maupun response boleh ditentukan anda.
11. Tambahkan header pada saat request ke API di soal no 10 :
    a. User-id : ifabula
    b. Scope: user

    Validasilah proses request ke API anda dan jika header diatas tidak cocok, balikan response :

    {
      responseCode: 401,
      responseMessage: "UNAUTHORIZED"
    }

```
const express = require('express');
const router = express.Router();



let dummyData = [
  { id: 1, name: "Alice Johnson", email: "alice.johnson@example.com", age: 25
},
  { id: 2, name: "Bob Smith", email: "bob.smith@example.com", age: 30 },
  { id: 3, name: "Charlie Brown", email: "charlie.brown@example.com", age: 28
},
  { id: 4, name: "Diana Prince", email: "diana.prince@example.com", age: 32
},
  { id: 5, name: "Edward Norton", email: "edward.norton@example.com", age: 45
},
  { id: 6, name: "Fiona Gallagher", email: "fiona.gallagher@example.com",
age: 22 },
  { id: 7, name: "George Miller", email: "george.miller@example.com", age: 35
},
  { id: 8, name: "Helen Mirren", email: "helen.mirren@example.com", age: 50
},
  { id: 9, name: "Ivan Petrov", email: "ivan.petrov@example.com", age: 40 },
  { id: 10, name: "Julia Roberts", email: "julia.roberts@example.com", age:
38 }
];


router.use((req, res, next) => {
  const userId = req.header('User-id');
  const scope = req.header('Scope');
  if (userId !== 'ifabula' || scope !== 'user') {
    return res.status(401).json({
      responseCode: 401,
      responseMessage: "UNAUTHORIZED"
    });
```

```
  }
  next();
});


router.get('/data', (req, res) => {
  res.json({
    responseCode: 200,
    responseMessage: "Success",
    data: dummyData
  });
});

router.post('/submit', (req, res) => {
  const { name, email, age } = req.body;

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

  if (!name || !email) {
    return res.status(400).json({
      responseCode: 400,
      responseMessage: "Name and Email are required"
    });
  }

  if (!emailRegex.test(email)) {
    return res.status(400).json({
      responseCode: 400,
      responseMessage: "Invalid email format"
    });
  }

  const newEntry = {
    id: dummyData.length + 1,
    name,
    email,
    age: age || null
  };

  dummyData.push(newEntry);

  res.json({
    responseCode: 201,
    responseMessage: "Data successfully submitted",
    data: newEntry
  });
});

module.exports = router;
```

Api/Get:

1. **router.get('/data', ...)**
   Menangani permintaan HTTP GET ke endpoint /data.
2. **res.json({...})**
   Mengirimkan response dalam format JSON ke client.
3. **Isi JSON Response:**
   responseCode: 200 → Menandakan permintaan berhasil.
   responseMessage: "Success" → Pesan bahwa data berhasil dikirim.
   data: dummyData → Isi data berasal dari array dummyData, yaitu daftar pengguna.

📌 **Api/Post:**

1. **router.post('/submit', ...)**
   Menangani permintaan HTTP POST ke endpoint /submit.
2. **const { name, email, age } = req.body;**
   Mengambil data yang dikirim oleh client melalui body JSON.
3. **Validasi:**
   Jika name atau email kosong → response **400** dengan pesan "Name and Email are required".
   Jika format email tidak valid (dicek dengan regex) → response **400** dengan pesan "Invalid email format".
4. **Response:**
   responseCode: 201 → Menandakan data berhasil ditambahkan.
   responseMessage: "Data successfully submitted"
   data: newEntry → Data yang baru saja dikirim oleh user.

Semua soal dapat di akses di file express.js yang di kirimkan untuk nomer 10 dan 11 ada swaggernya untuk mencobanya