// GitHub config — REPLACE with your actual trial info

    const GITHUB\_USERNAME = "viralbalaji";

    const REPO\_NAME = "Product\_Sales";

    const REPO\_BRANCH = "main";

    const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

NOW, CREATE A NEW PROGRAM FOR SELLED PRODUCTS'S FETCHING SECTION.(NOT CHALLAN TOTAL)

- THE PRODUCTS CAN FETCH BY INDIVIDUALY, DATE, BUYER NAME, CHALLAN NUMBER, VEHICLE.

NOW, THE PROGRAM WILL WORK LIKE THIS:

- -FETCH BY DATE INTERVAL, IT MANDATORY.(DEFAULT IS CURRENT DD-MM-YYYY).

-SHOW ONLY PRODUCTS LIKE:

|"Product" | |"BAGS"|

|<b>TOTAL BAGS</b>|

--FETCH BY BUYER IN SELECTED DATE INTERVAL, :

-SHOW OPTIONS FOR AVAILABLE BUYERS.

-SHOW ONLY PRODUCTS LIKE:

<b>BUYER NAME</b>

|"Product" | |"BAGS"|

|<b>TOTAL BAGS</b>|

--FETCH BY VEHICLE BY BUYER(IF)SELECTED IN SELECTED DATE INTERVAL, :

-SHOW OPTIONS FOR AVAILABLE VEHICLE.

-SHOW ONLY PRODUCTS LIKE:

<b>VEHICLE </b> <b>HOW MANY TIME THAT VEHICLE IS USED</b>

|"Product" | |"BAGS"|

|<b>TOTAL BAGS</b>|

Now, this is JSON format which is uploaded TO github in my private repository,

{

"buyerName": "Sevak",

"challanNumber": "235",

"date": "6/6/2025",

"products": [

"500 ml TT"

],

"bags": [

"8"

],

"quantities": [

"4000"

],

"rates": [

"3.20"

],

"amounts": [

"12,800"

],

"subtotalText": " SUB TOTAL :",

"subtotalValue": "12,800.0",

"totalBags": "8",

"discountPercent": "2%",

"discountValue": "256.0",

"totalQuantity": "4,000",

"total": "12,544"

}

{

"buyerName": "Mahadev Marketing",

"challanNumber": "236",

"date": "2/6/2025",

"products": [

"250 ml W",

"300 ml W",

"300 ml BT",

"400 ml W",

"400 ml BT",

"500 ml W",

"500 ml BT",

"750 ml W",

"750 ml BT",

"1000 ml W"

],

"bags": [

"8",

"6",

"6",

"3",

"2",

"10",

"10",

"2",

"3",

"2"

],

"quantities": [

"4000",

"3000",

"3000",

"1500",

"1000",

"5000",

"5000",

"1000",

"1500",

"1000"

],

"rates": [

"2.15",

"2.90",

"2.90",

"3.00",

"3.00",

"3.05",

"3.05",

"4.15",

"4.15",

"4.70"

],

"amounts": [

"8,600",

"8,700",

"8,700",

"4,500",

"3,000",

"15,250",

"15,250",

"4,150",

"6,225",

"4,700"

],

"subtotalText": " SUB TOTAL :",

"subtotalValue": "79,075.0",

"totalBags": "52",

"discountPercent": "1.50 ",

"discountValue": "1,186.1",

"totalQuantity": "26,000",

"total": "77,889"

}

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>Selled Products Fetch</title>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/dayjs/1.11.7/dayjs.min.js"></script>

  <style>

    body { font-family: Arial; padding: 20px; background: #f9f9f9; }

    table { width: 100%; border-collapse: collapse; margin-top: 10px; }

    th, td { padding: 8px 12px; border: 1px solid #ccc; }

    th { background: #eee; }

    select, input[type="date"] { margin: 5px; padding: 5px; }

    b { display: block; margin-top: 10px; }

  </style>

</head>

<body>

  <h2>Fetch Selled Products</h2>

  <label>From: <input type="date" id="fromDate"></label>

  <label>To: <input type="date" id="toDate"></label>

  <button onclick="loadJSON()">Load</button>

  <br><br>

  <label>Buyer: <select id="buyerSelect" onchange="filterByBuyer()"></select></label>

  <label>Vehicle: <select id="vehicleSelect" onchange="filterByVehicle()"></select></label>

  <div id="result"></div>

  <script>

    const GITHUB\_USERNAME = "viralbalaji";

    const REPO\_NAME = "Product\_Sales";

    const REPO\_BRANCH = "main";

    const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

    let allData = [];

    function getTodayDateStr() {

      return dayjs().format("YYYY-MM-DD");

    }

    document.getElementById("fromDate").value = getTodayDateStr();

    document.getElementById("toDate").value = getTodayDateStr();

    async function loadJSON() {

      const from = dayjs(document.getElementById('fromDate').value);

      const to = dayjs(document.getElementById('toDate').value);

    const url = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/extracted\_data?ref=${REPO\_BRANCH}`;

      console.log("Fetching from:", url);

      try {

        const response = await fetch(url, {

          headers: { Authorization: `token ${GITHUB\_TOKEN}` }

        });

        const files = await response.json();

        if (!Array.isArray(files)) {

          console.error("Failed to fetch files:", files);

          alert("⚠️ GitHub access failed. Check token or repo.");

          return;

        }

        const jsonFiles = files.filter(file => file.name.endsWith('.json'));

        allData = [];

        for (let file of jsonFiles) {

          const res = await fetch(file.download\_url);

          const data = await res.json();

          let parts = data.date.split('/');

          if (parts.length !== 3) continue;

          let fileDate = dayjs(`${parts[2]}-${parts[1].padStart(2, '0')}-${parts[0].padStart(2, '0')}`);

          if (fileDate.isAfter(from.subtract(1, 'day')) && fileDate.isBefore(to.add(1, 'day'))) {

            allData.push(data);

          }

        }

        if (allData.length === 0) {

          document.getElementById('result').innerHTML = `<b>❌ No records found for selected date interval.</b>`;

          return;

        }

        populateBuyers();

        showProducts(allData);

      } catch (err) {

        console.error("Error fetching JSON files:", err);

        alert("⚠️ Something went wrong while fetching data.");

      }

    }

    function populateBuyers() {

      const buyerSelect = document.getElementById('buyerSelect');

      buyerSelect.innerHTML = '<option value="">-- All Buyers --</option>';

      const buyers = [...new Set(allData.map(d => d.buyerName))];

      buyers.forEach(b => {

        const opt = document.createElement('option');

        opt.value = b;

        opt.textContent = b;

        buyerSelect.appendChild(opt);

      });

    }

    function filterByBuyer() {

      const buyer = document.getElementById('buyerSelect').value;

      const filtered = buyer ? allData.filter(d => d.buyerName === buyer) : allData;

      const vehicleSelect = document.getElementById('vehicleSelect');

      vehicleSelect.innerHTML = '<option value="">-- All Vehicles --</option>';

      const vehicles = [...new Set(filtered.map(d => d.vehicle).filter(Boolean))];

      vehicles.forEach(v => {

        const opt = document.createElement('option');

        opt.value = v;

        opt.textContent = v;

        vehicleSelect.appendChild(opt);

      });

      showProducts(filtered, buyer);

    }

    function filterByVehicle() {

      const buyer = document.getElementById('buyerSelect').value;

      const vehicle = document.getElementById('vehicleSelect').value;

      const filtered = allData.filter(d =>

        (!buyer || d.buyerName === buyer) &&

        (!vehicle || d.vehicle === vehicle)

      );

      showProducts(filtered, buyer, vehicle);

    }

    function showProducts(data, buyer = '', vehicle = '') {

      let productMap = {};

      let totalBags = 0;

      data.forEach(entry => {

        entry.products.forEach((product, i) => {

          const bags = parseInt((entry.bags[i] || "0").replace(/,/g, '')) || 0;

          productMap[product] = (productMap[product] || 0) + bags;

          totalBags += bags;

        });

      });

      let html = '';

      if (vehicle) {

        html += `<b>VEHICLE: ${vehicle}</b> <b>USED: ${data.length} TIMES</b>`;

      } else if (buyer) {

        html += `<b>BUYER NAME: ${buyer}</b>`;

      }

      html += '<table><tr><th>Product</th><th>Bags</th></tr>';

      for (let product in productMap) {

        html += `<tr><td>${product}</td><td>${productMap[product]}</td></tr>`;

      }

      html += `<tr><td><b>TOTAL BAGS</b></td><td><b>${totalBags}</b></td></tr></table>`;

      document.getElementById('result').innerHTML = html;

    }

  </script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Fetch Sold Products by Date - Private Repo Fix</title>

<style>

  body { font-family: Arial, sans-serif; margin: 20px; background: #fafafa; }

  label { margin-right: 10px; }

  button { padding: 6px 12px; cursor: pointer; }

  table { border-collapse: collapse; width: 100%; margin-top: 20px; background: #fff; }

  th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

  th { background: #eee; }

  .total-row { font-weight: bold; }

  #result { margin-top: 20px; }

</style>

<script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>

</head>

<body>

<h2>Fetch Sold Products by Date Interval (Private Repo)</h2>

<label for="fromDate">From Date:</label>

<input type="date" id="fromDate" />

<label for="toDate">To Date:</label>

<input type="date" id="toDate" />

<button id="fetchBtn">Fetch Products</button>

<div id="result"></div>

<script>

  const GITHUB\_USERNAME = "viralbalaji";

  const REPO\_NAME = "Product\_Sales";

  const REPO\_BRANCH = "main";

  const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

  const FOLDER\_PATH = "extracted\_data";

  document.getElementById('fetchBtn').addEventListener('click', fetchProductData);

  // Set default date inputs to today

  const todayStr = new Date().toISOString().slice(0,10);

  document.getElementById('fromDate').value = todayStr;

  document.getElementById('toDate').value = todayStr;

  async function fetchProductData() {

    const fromInput = document.getElementById('fromDate').value;

    const toInput = document.getElementById('toDate').value;

    if (!fromInput || !toInput) {

      alert("Please select both From and To dates.");

      return;

    }

    const from = dayjs(fromInput);

    const to = dayjs(toInput);

    if (to.isBefore(from)) {

      alert("To Date must be same or after From Date.");

      return;

    }

    const headers = {

      Authorization: `token ${GITHUB\_TOKEN}`,

      Accept: "application/vnd.github.v3+json"

    };

    try {

      // Step 1: List files in folder

      const listUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}?ref=${REPO\_BRANCH}`;

      const listResp = await fetch(listUrl, { headers });

      if (!listResp.ok) throw new Error(`GitHub API error listing files: ${listResp.status}`);

      const files = await listResp.json();

      const jsonFiles = files.filter(f => f.name.endsWith('.json'));

      if (jsonFiles.length === 0) {

        document.getElementById('result').innerHTML = "<b>No JSON files found in repository folder.</b>";

        return;

      }

      const allData = [];

      // Step 2: For each file, get content via git blobs API

      for (const file of jsonFiles) {

        try {

          // Fetch blob info

          const blobUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/git/blobs/${file.sha}`;

          const blobResp = await fetch(blobUrl, { headers });

          if (!blobResp.ok) {

            console.warn(`Failed to get blob for ${file.name}: ${blobResp.status}`);

            continue;

          }

          const blobData = await blobResp.json();

          // Blob content is base64 encoded

          const contentStr = atob(blobData.content.replace(/\n/g, ''));

          let data;

          try {

            data = JSON.parse(contentStr);

          } catch (jsonErr) {

            console.warn(`JSON parse error in ${file.name}:`, jsonErr);

            continue;

          }

          if (!data.date) continue;

          const parts = data.date.split('/');

          if (parts.length !== 3) continue;

          const fileDate = dayjs(`${parts[2]}-${parts[1].padStart(2,'0')}-${parts[0].padStart(2,'0')}`);

          if (fileDate.isBefore(from) || fileDate.isAfter(to)) continue;

          allData.push(data);

        } catch (fileErr) {

          console.error(`Error fetching/parsing file ${file.name}:`, fileErr);

        }

      }

      if (allData.length === 0) {

        document.getElementById('result').innerHTML = `<b>❌ No records found for selected date interval.</b>`;

        return;

      }

      showProducts(allData);

    } catch (err) {

      console.error("Error fetching files list:", err);

      document.getElementById('result').innerHTML = `<b style="color:red;">Failed to fetch data. Check console for errors.</b>`;

    }

  }

  function showProducts(allData) {

    const productMap = {};

    allData.forEach(record => {

      if (!Array.isArray(record.products) || !Array.isArray(record.bags)) return;

      record.products.forEach((prod, i) => {

        const bags = parseInt(record.bags[i] || '0', 10);

        if (!prod) return;

        productMap[prod] = (productMap[prod] || 0) + bags;

      });

    });

    let html = `<table>

      <thead><tr><th>Product</th><th>Bags</th></tr></thead>

      <tbody>`;

    Object.entries(productMap).forEach(([prod, bags]) => {

      html += `<tr><td>${prod}</td><td>${bags}</td></tr>`;

    });

    const totalBags = Object.values(productMap).reduce((a,b) => a+b, 0);

    html += `<tr class="total-row"><td>Total Bags</td><td>${totalBags}</td></tr>`;

    html += `</tbody></table>`;

    document.getElementById('result').innerHTML = html;

  }

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Fetch Sold Products with Buyer & Vehicle Filter</title>

<style>

  body { font-family: Arial, sans-serif; margin: 20px; background: #fafafa; }

  label { margin-right: 10px; }

  select, input[type=date] { margin-right: 20px; padding: 5px; }

  button { padding: 6px 12px; cursor: pointer; }

  table { border-collapse: collapse; width: 100%; margin-top: 20px; background: #fff; }

  th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

  th { background: #eee; }

  .total-row { font-weight: bold; }

  #result { margin-top: 20px; }

</style>

<script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>

</head>

<body>

<h2>Fetch Sold Products by Date Interval with Buyer & Vehicle Filter</h2>

<div>

  <label for="fromDate">From Date:</label>

  <input type="date" id="fromDate" />

  <label for="toDate">To Date:</label>

  <input type="date" id="toDate" />

  <button id="fetchBtn">Fetch Products</button>

</div>

<div style="margin-top: 15px;">

  <label for="buyerSelect">Buyer:</label>

  <select id="buyerSelect" disabled>

    <option value="">-- Select Buyer --</option>

  </select>

  <label for="vehicleSelect">Vehicle:</label>

  <select id="vehicleSelect" disabled>

    <option value="">-- Select Vehicle --</option>

  </select>

</div>

<div id="result"></div>

<script>

  const GITHUB\_USERNAME = "viralbalaji";

  const REPO\_NAME = "Product\_Sales";

  const REPO\_BRANCH = "main";

  const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

  const FOLDER\_PATH = "extracted\_data";

  let allData = [];

  document.getElementById('fetchBtn').addEventListener('click', fetchProductData);

  document.getElementById('buyerSelect').addEventListener('change', onBuyerChange);

  document.getElementById('vehicleSelect').addEventListener('change', onVehicleChange);

  // Set default date inputs to today

  const todayStr = new Date().toISOString().slice(0,10);

  document.getElementById('fromDate').value = todayStr;

  document.getElementById('toDate').value = todayStr;

  async function fetchProductData() {

    allData = [];

    clearFilters();

    document.getElementById('result').innerHTML = "Loading...";

    const fromInput = document.getElementById('fromDate').value;

    const toInput = document.getElementById('toDate').value;

    if (!fromInput || !toInput) {

      alert("Please select both From and To dates.");

      return;

    }

    const from = dayjs(fromInput);

    const to = dayjs(toInput);

    if (to.isBefore(from)) {

      alert("To Date must be same or after From Date.");

      return;

    }

    const headers = {

      Authorization: `token ${GITHUB\_TOKEN}`,

      Accept: "application/vnd.github.v3+json"

    };

    try {

      // List files

      const listUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}?ref=${REPO\_BRANCH}`;

      const listResp = await fetch(listUrl, { headers });

      if (!listResp.ok) throw new Error(`GitHub API error listing files: ${listResp.status}`);

      const files = await listResp.json();

      const jsonFiles = files.filter(f => f.name.endsWith('.json'));

      if (jsonFiles.length === 0) {

        document.getElementById('result').innerHTML = "<b>No JSON files found in repository folder.</b>";

        return;

      }

      for (const file of jsonFiles) {

        try {

          const blobUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/git/blobs/${file.sha}`;

          const blobResp = await fetch(blobUrl, { headers });

          if (!blobResp.ok) {

            console.warn(`Failed to get blob for ${file.name}: ${blobResp.status}`);

            continue;

          }

          const blobData = await blobResp.json();

          const contentStr = atob(blobData.content.replace(/\n/g, ''));

          let data;

          try {

            data = JSON.parse(contentStr);

          } catch {

            continue;

          }

          if (!data.date) continue;

          const parts = data.date.split('/');

          if (parts.length !== 3) continue;

          const fileDate = dayjs(`${parts[2]}-${parts[1].padStart(2,'0')}-${parts[0].padStart(2,'0')}`);

          if (fileDate.isBefore(from) || fileDate.isAfter(to)) continue;

          allData.push(data);

        } catch (fileErr) {

          console.error(`Error fetching/parsing file ${file.name}:`, fileErr);

        }

      }

      if (allData.length === 0) {

        document.getElementById('result').innerHTML = `<b>❌ No records found for selected date interval.</b>`;

        return;

      }

      populateBuyerOptions();

      showProducts(allData);

    } catch (err) {

      console.error("Error fetching files list:", err);

      document.getElementById('result').innerHTML = `<b style="color:red;">Failed to fetch data. Check console for errors.</b>`;

    }

  }

  function clearFilters() {

    const buyerSelect = document.getElementById('buyerSelect');

    const vehicleSelect = document.getElementById('vehicleSelect');

    buyerSelect.innerHTML = '<option value="">-- Select Buyer --</option>';

    buyerSelect.disabled = true;

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    vehicleSelect.disabled = true;

  }

  function populateBuyerOptions() {

    const buyerSelect = document.getElementById('buyerSelect');

    const buyersSet = new Set();

    allData.forEach(rec => {

      if (rec.buyerName) buyersSet.add(rec.buyerName.trim());

    });

    const buyers = Array.from(buyersSet).sort();

    if (buyers.length === 0) {

      buyerSelect.disabled = true;

      return;

    }

    buyers.forEach(b => {

      const opt = document.createElement('option');

      opt.value = b;

      opt.textContent = b;

      buyerSelect.appendChild(opt);

    });

    buyerSelect.disabled = false;

  }

  function onBuyerChange() {

    const buyer = this.value;

    const vehicleSelect = document.getElementById('vehicleSelect');

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    vehicleSelect.disabled = true;

    if (!buyer) {

      // No buyer selected, show all

      showProducts(allData);

      return;

    }

    // Filter allData for selected buyer

    const buyerData = allData.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

    if (buyerData.length === 0) {

      document.getElementById('result').innerHTML = `<b>No records found for buyer "${buyer}".</b>`;

      return;

    }

    // Populate vehicle options used by this buyer

    const vehicleSet = new Map(); // vehicle -> count

    buyerData.forEach(rec => {

      const vehicle = (rec.vehicle || "").trim();

      if (vehicle) {

        vehicleSet.set(vehicle, (vehicleSet.get(vehicle) || 0) + 1);

      }

    });

    if (vehicleSet.size > 0) {

      vehicleSelect.disabled = false;

      vehicleSet.forEach((count, veh) => {

        const opt = document.createElement('option');

        opt.value = veh;

        opt.textContent = `${veh} (${count} times)`;

        vehicleSelect.appendChild(opt);

      });

    }

    // Show products for buyer (no vehicle filter yet)

    showProducts(buyerData);

  }

  function onVehicleChange() {

    const buyer = document.getElementById('buyerSelect').value;

    const vehicle = this.value;

    if (!buyer) {

      document.getElementById('result').innerHTML = `<b>Please select a Buyer first.</b>`;

      return;

    }

    let filtered = allData.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

    if (vehicle) {

      filtered = filtered.filter(rec => (rec.vehicle || "").trim() === vehicle);

    }

    if (filtered.length === 0) {

      document.getElementById('result').innerHTML = `<b>No records found for Buyer "${buyer}" and Vehicle "${vehicle}".</b>`;

      return;

    }

    showProducts(filtered);

  }

  // Show aggregated products and total bags

  function showProducts(dataArray) {

    const productMap = {};

    dataArray.forEach(record => {

      if (!Array.isArray(record.products) || !Array.isArray(record.bags)) return;

      record.products.forEach((prod, i) => {

        const bags = parseInt(record.bags[i] || '0', 10);

        if (!prod) return;

        productMap[prod] = (productMap[prod] || 0) + bags;

      });

    });

    if (Object.keys(productMap).length === 0) {

      document.getElementById('result').innerHTML = `<b>No products found.</b>`;

      return;

    }

    let html = `<table>

      <thead><tr><th>Product</th><th>Bags</th></tr></thead>

      <tbody>`;

    Object.entries(productMap).forEach(([prod, bags]) => {

      html += `<tr><td>${prod}</td><td>${bags}</td></tr>`;

    });

    const totalBags = Object.values(productMap).reduce((a,b) => a+b, 0);

    html += `<tr class="total-row"><td>Total Bags</td><td>${totalBags}</td></tr>`;

    html += `</tbody></table>`;

    document.getElementById('result').innerHTML = html;

  }

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Fetch Sold Products with Buyer & Vehicle Filter (Detailed Vehicle Usage)</title>

<style>

  body { font-family: Arial, sans-serif; margin: 20px; background: #fafafa; }

  label { margin-right: 10px; }

  select, input[type=date] { margin-right: 20px; padding: 5px; }

  button { padding: 6px 12px; cursor: pointer; }

  table { border-collapse: collapse; width: 100%; margin-top: 20px; background: #fff; }

  th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

  th { background: #eee; }

  .total-row { font-weight: bold; }

  #result { margin-top: 20px; }

  #vehicleUsageInfo { margin-top: 10px; font-style: normal; background: #e8f0fe; padding: 10px; border-radius: 5px; }

  #vehicleUsageInfo h3, #vehicleUsageInfo p { margin: 5px 0; }

  #vehicleUsageTable { margin-top: 10px; width: 100%; border-collapse: collapse; }

  #vehicleUsageTable th, #vehicleUsageTable td { border: 1px solid #bbb; padding: 6px; text-align: left; }

  #vehicleUsageTable th { background-color: #d0d9ff; }

</style>

<script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>

</head>

<body>

<h2>Fetch Sold Products by Date Interval with Buyer & Vehicle Filter</h2>

<div>

  <label for="fromDate">From Date:</label>

  <input type="date" id="fromDate" />

  <label for="toDate">To Date:</label>

  <input type="date" id="toDate" />

  <button id="fetchBtn">Fetch Products</button>

</div>

<div style="margin-top: 15px;">

  <label for="buyerSelect">Buyer:</label>

  <select id="buyerSelect" disabled>

    <option value="">-- Select Buyer --</option>

  </select>

  <label for="vehicleSelect">Vehicle:</label>

  <select id="vehicleSelect" disabled>

    <option value="">-- Select Vehicle --</option>

  </select>

</div>

<div id="vehicleUsageInfo"></div>

<div id="result"></div>

<script>

  const GITHUB\_USERNAME = "viralbalaji";

  const REPO\_NAME = "Product\_Sales";

  const REPO\_BRANCH = "main";

  const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

  const FOLDER\_PATH = "extracted\_data";

  let allData = [];

  document.getElementById('fetchBtn').addEventListener('click', fetchProductData);

  document.getElementById('buyerSelect').addEventListener('change', onBuyerChange);

  document.getElementById('vehicleSelect').addEventListener('change', onVehicleChange);

  // Set default date inputs to today

  const todayStr = new Date().toISOString().slice(0,10);

  document.getElementById('fromDate').value = todayStr;

  document.getElementById('toDate').value = todayStr;

  async function fetchProductData() {

    allData = [];

    clearFilters();

    clearVehicleUsageInfo();

    document.getElementById('result').innerHTML = "Loading...";

    const fromInput = document.getElementById('fromDate').value;

    const toInput = document.getElementById('toDate').value;

    if (!fromInput || !toInput) {

      alert("Please select both From and To dates.");

      return;

    }

    const from = dayjs(fromInput);

    const to = dayjs(toInput);

    if (to.isBefore(from)) {

      alert("To Date must be same or after From Date.");

      return;

    }

    const headers = {

      Authorization: `token ${GITHUB\_TOKEN}`,

      Accept: "application/vnd.github.v3+json"

    };

    try {

      const listUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}?ref=${REPO\_BRANCH}`;

      const listResp = await fetch(listUrl, { headers });

      if (!listResp.ok) throw new Error(`GitHub API error listing files: ${listResp.status}`);

      const files = await listResp.json();

      const jsonFiles = files.filter(f => f.name.endsWith('.json'));

      if (jsonFiles.length === 0) {

        document.getElementById('result').innerHTML = "<b>No JSON files found in repository folder.</b>";

        return;

      }

      for (const file of jsonFiles) {

        try {

          const blobUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/git/blobs/${file.sha}`;

          const blobResp = await fetch(blobUrl, { headers });

          if (!blobResp.ok) {

            console.warn(`Failed to get blob for ${file.name}: ${blobResp.status}`);

            continue;

          }

          const blobData = await blobResp.json();

          const contentStr = atob(blobData.content.replace(/\n/g, ''));

          let data;

          try {

            data = JSON.parse(contentStr);

          } catch {

            continue;

          }

          if (!data.date) continue;

          const parts = data.date.split('/');

          if (parts.length !== 3) continue;

          const fileDate = dayjs(`${parts[2]}-${parts[1].padStart(2,'0')}-${parts[0].padStart(2,'0')}`);

          if (fileDate.isBefore(from) || fileDate.isAfter(to)) continue;

          allData.push(data);

        } catch (fileErr) {

          console.error(`Error fetching/parsing file ${file.name}:`, fileErr);

        }

      }

      if (allData.length === 0) {

        document.getElementById('result').innerHTML = `<b>❌ No records found for selected date interval.</b>`;

        return;

      }

      populateBuyerOptions();

      populateVehicleOptions();

      showProducts(allData);

    } catch (err) {

      console.error("Error fetching files list:", err);

      document.getElementById('result').innerHTML = `<b style="color:red;">Failed to fetch data. Check console for errors.</b>`;

    }

  }

  function clearFilters() {

    const buyerSelect = document.getElementById('buyerSelect');

    const vehicleSelect = document.getElementById('vehicleSelect');

    buyerSelect.innerHTML = '<option value="">-- Select Buyer --</option>';

    buyerSelect.disabled = true;

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    vehicleSelect.disabled = true;

  }

  function clearVehicleUsageInfo() {

    document.getElementById('vehicleUsageInfo').innerHTML = "";

  }

  // Populate buyer dropdown based on allData

  function populateBuyerOptions() {

    const buyerSelect = document.getElementById('buyerSelect');

    const buyersSet = new Set();

    allData.forEach(rec => {

      if (rec.buyerName) buyersSet.add(rec.buyerName.trim());

    });

    const buyers = Array.from(buyersSet).sort();

    if (buyers.length === 0) {

      buyerSelect.disabled = true;

      return;

    }

    buyers.forEach(b => {

      const opt = document.createElement('option');

      opt.value = b;

      opt.textContent = b;

      buyerSelect.appendChild(opt);

    });

    buyerSelect.disabled = false;

  }

  // Populate vehicle dropdown based on allData (just vehicle names only, no counts)

  function populateVehicleOptions() {

    const vehicleSelect = document.getElementById('vehicleSelect');

    const vehicleSet = new Set();

    allData.forEach(rec => {

      const vehicle = (rec.vehicle || "").trim();

      if (!vehicle) return;

      vehicleSet.add(vehicle);

    });

    const vehicles = Array.from(vehicleSet).sort();

    if (vehicles.length === 0) {

      vehicleSelect.disabled = true;

      return;

    }

    vehicles.forEach(veh => {

      const opt = document.createElement('option');

      opt.value = veh;

      opt.textContent = veh;  // only name here

      vehicleSelect.appendChild(opt);

    });

    vehicleSelect.disabled = false;

  }

  // Buyer dropdown change handler

  function onBuyerChange() {

    clearVehicleUsageInfo();

    const buyer = this.value;

    const vehicleSelect = document.getElementById('vehicleSelect');

    if (!buyer) {

      // No buyer selected, show all data

      populateVehicleOptions(); // repopulate all vehicles for allData

      showProducts(allData);

      return;

    }

    // Filter data by buyer

    const buyerData = allData.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

    if (buyerData.length === 0) {

      document.getElementById('result').innerHTML = `<b>No records found for buyer "${buyer}".</b>`;

      vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

      vehicleSelect.disabled = true;

      return;

    }

    // Populate vehicle dropdown only with vehicles used by this buyer

    const vehicleMap = new Map();

    buyerData.forEach(rec => {

      const vehicle = (rec.vehicle || "").trim();

      if (!vehicle) return;

      vehicleMap.set(vehicle, (vehicleMap.get(vehicle) || 0) + 1);

    });

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    if (vehicleMap.size > 0) {

      vehicleSelect.disabled = false;

      vehicleMap.forEach((count, veh) => {

        const timesText = count > 1 ? 'times' : 'time';

        const opt = document.createElement('option');

        opt.value = veh;

        opt.textContent = veh;  // still only name here as per new requirement

        vehicleSelect.appendChild(opt);

      });

    } else {

      vehicleSelect.disabled = true;

    }

    showProducts(buyerData);

  }

  // Vehicle dropdown change handler

  function onVehicleChange() {

    clearVehicleUsageInfo();

    const vehicle = this.value;

    const buyer = document.getElementById('buyerSelect').value;

    if (!vehicle) {

      // no vehicle selected, show filtered by buyer or all

      if (buyer) {

        const buyerData = allData.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

        showProducts(buyerData);

      } else {

        showProducts(allData);

      }

      return;

    }

    // Filter by vehicle (with or without buyer)

    let filtered = allData.filter(rec => (rec.vehicle || "").trim() === vehicle);

    if (buyer) {

      filtered = filtered.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

    }

    if (filtered.length === 0) {

      document.getElementById('result').innerHTML = `<b>No records found for Vehicle "${vehicle}"${buyer ? ' and Buyer "' + buyer + '"' : ''}.</b>`;

      return;

    }

    // Detailed vehicle usage info:

    // We want:

    // - Vehicle - "(NAME)"

    // - X TIMES USED (Y UNIQUE DAYS USED)

    // - Table of buyers, date, use time count for that buyer on that date

    // Count total times used

    const totalTimesUsed = filtered.length;

    // Find unique dates in dd-mm-yyyy format

    const uniqueDatesSet = new Set(filtered.map(r => {

      if (!r.date) return "";

      const parts = r.date.split('/');

      if (parts.length !== 3) return "";

      return `${parts[0].padStart(2,'0')}-${parts[1].padStart(2,'0')}-${parts[2]}`;

    }).filter(d => d !== ""));

    const uniqueDaysUsed = uniqueDatesSet.size;

    // Group usage by buyer & date with counts

    // structure: { buyerName: { date: count } }

    const usageMap = {};

    filtered.forEach(r => {

      const buyerName = r.buyerName ? r.buyerName.trim() : "Unknown";

      if (!usageMap[buyerName]) usageMap[buyerName] = {};

      const parts = r.date ? r.date.split('/') : [];

      if (parts.length !== 3) return;

      const dateStr = `${parts[0].padStart(2,'0')}-${parts[1].padStart(2,'0')}-${parts[2]}`;

      usageMap[buyerName][dateStr] = (usageMap[buyerName][dateStr] || 0) + 1;

    });

    // Build HTML summary

    const timesText = totalTimesUsed > 1 ? 'TIMES' : 'TIME';

    const daysText = uniqueDaysUsed > 1 ? 'DAYS' : 'DAY';

    let html = `<h3>Vehicle - "(${vehicle})"</h3>`;

    html += `<p><b>${totalTimesUsed} ${timesText} USED (${uniqueDaysUsed} UNIQUE ${daysText} USED)</b></p>`;

    html += `<table id="vehicleUsageTable"><thead><tr><th>BUYER(s)</th><th>DATE</th><th>USE TIME</th></tr></thead><tbody>`;

    // For better ordering, sort buyers by name, then dates asc

    const sortedBuyers = Object.keys(usageMap).sort((a,b) => a.localeCompare(b));

    sortedBuyers.forEach(buyerName => {

      const dates = Object.keys(usageMap[buyerName]).sort((a,b) => {

        // parse dates dd-mm-yyyy to Date for sorting

        const [d1,m1,y1] = a.split('-').map(x=>parseInt(x));

        const [d2,m2,y2] = b.split('-').map(x=>parseInt(x));

        return new Date(y1,m1-1,d1) - new Date(y2,m2-1,d2);

      });

      dates.forEach(dateStr => {

        const count = usageMap[buyerName][dateStr];

        const countText = count > 1 ? count : count;

        html += `<tr><td>${buyerName}</td><td>${dateStr}</td><td>${countText}</td></tr>`;

      });

    });

    html += `</tbody></table>`;

    document.getElementById('vehicleUsageInfo').innerHTML = html;

    // Show aggregated products for filtered data

    showProducts(filtered);

  }

  // Show aggregated products and total bags

  function showProducts(dataArray) {

    const productMap = {};

    dataArray.forEach(record => {

      if (!Array.isArray(record.products) || !Array.isArray(record.bags)) return;

      record.products.forEach((prod, i) => {

        const bags = parseInt(record.bags[i] || '0', 10);

        if (!prod) return;

        productMap[prod] = (productMap[prod] || 0) + bags;

      });

    });

    if (Object.keys(productMap).length === 0) {

      document.getElementById('result').innerHTML = `<b>No products found.</b>`;

      return;

    }

    let html = `<table>

      <thead><tr><th>Product</th><th>Bags</th></tr></thead>

      <tbody>`;

    Object.entries(productMap).forEach(([prod, bags]) => {

      html += `<tr><td>${prod}</td><td>${bags}</td></tr>`;

    });

    const totalBags = Object.values(productMap).reduce((a,b) => a+b, 0);

    html += `<tr class="total-row"><td>Total Bags</td><td>${totalBags}</td></tr>`;

    html += `</tbody></table>`;

    document.getElementById('result').innerHTML = html;

  }

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Fetch Sold Products with Buyer & Vehicle Filter (Detailed Vehicle Usage)</title>

<style>

  body { font-family: Arial, sans-serif; margin: 20px; background: #fafafa; }

  label { margin-right: 10px; }

  select, input[type=date] { margin-right: 20px; padding: 5px; }

  button { padding: 6px 12px; cursor: pointer; }

  table { border-collapse: collapse; width: 100%; margin-top: 20px; background: #fff; }

  th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

  th { background: #eee; }

  .total-row { font-weight: bold; }

  #result { margin-top: 20px; }

  #vehicleUsageInfo { margin-top: 10px; font-style: normal; background: #e8f0fe; padding: 10px; border-radius: 5px; }

  #vehicleUsageInfo h3, #vehicleUsageInfo p { margin: 5px 0; }

  #vehicleUsageTable { margin-top: 10px; width: 100%; border-collapse: collapse; }

  #vehicleUsageTable th, #vehicleUsageTable td { border: 1px solid #bbb; padding: 6px; text-align: left; }

  #vehicleUsageTable th { background-color: #d0d9ff; }

</style>

<script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>

</head>

<body>

<h2>Fetch Sold Products by Date Interval with Buyer & Vehicle Filter</h2>

<div>

  <label for="fromDate">From Date:</label>

  <input type="date" id="fromDate" />

  <label for="toDate">To Date:</label>

  <input type="date" id="toDate" />

  <button id="fetchBtn">Fetch Products</button>

</div>

<div style="margin-top: 15px;">

  <label for="buyerSelect">Buyer:</label>

  <select id="buyerSelect" disabled>

    <option value="">-- Select Buyer --</option>

  </select>

  <label for="vehicleSelect">Vehicle:</label>

  <select id="vehicleSelect" disabled>

    <option value="">-- Select Vehicle --</option>

  </select>

</div>

<div id="vehicleUsageInfo"></div>

<div id="result"></div>

<script>

  const GITHUB\_USERNAME = "viralbalaji";

  const REPO\_NAME = "Product\_Sales";

  const REPO\_BRANCH = "main";

  const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

  const FOLDER\_PATH = "extracted\_data";

  let allData = [];

  document.getElementById('fetchBtn').addEventListener('click', fetchProductData);

  document.getElementById('buyerSelect').addEventListener('change', onBuyerChange);

  document.getElementById('vehicleSelect').addEventListener('change', onVehicleChange);

  // Set default date inputs to today

  const todayStr = new Date().toISOString().slice(0,10);

  document.getElementById('fromDate').value = todayStr;

  document.getElementById('toDate').value = todayStr;

  async function fetchProductData() {

    allData = [];

    clearFilters();

    clearVehicleUsageInfo();

    document.getElementById('result').innerHTML = "Loading...";

    const fromInput = document.getElementById('fromDate').value;

    const toInput = document.getElementById('toDate').value;

    if (!fromInput || !toInput) {

      alert("Please select both From and To dates.");

      return;

    }

    const from = dayjs(fromInput);

    const to = dayjs(toInput);

    if (to.isBefore(from)) {

      alert("To Date must be same or after From Date.");

      return;

    }

    const headers = {

      Authorization: `token ${GITHUB\_TOKEN}`,

      Accept: "application/vnd.github.v3+json"

    };

    try {

      const listUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}?ref=${REPO\_BRANCH}`;

      const listResp = await fetch(listUrl, { headers });

      if (!listResp.ok) throw new Error(`GitHub API error listing files: ${listResp.status}`);

      const files = await listResp.json();

      const jsonFiles = files.filter(f => f.name.endsWith('.json'));

      if (jsonFiles.length === 0) {

        document.getElementById('result').innerHTML = "<b>No JSON files found in repository folder.</b>";

        return;

      }

      for (const file of jsonFiles) {

        try {

          const blobUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/git/blobs/${file.sha}`;

          const blobResp = await fetch(blobUrl, { headers });

          if (!blobResp.ok) {

            console.warn(`Failed to get blob for ${file.name}: ${blobResp.status}`);

            continue;

          }

          const blobData = await blobResp.json();

          const contentStr = atob(blobData.content.replace(/\n/g, ''));

          let data;

          try {

            data = JSON.parse(contentStr);

          } catch {

            continue;

          }

          if (!data.date) continue;

          const parts = data.date.split('/');

          if (parts.length !== 3) continue;

          const fileDate = dayjs(`${parts[2]}-${parts[1].padStart(2,'0')}-${parts[0].padStart(2,'0')}`);

          if (fileDate.isBefore(from) || fileDate.isAfter(to)) continue;

          allData.push(data);

        } catch (fileErr) {

          console.error(`Error fetching/parsing file ${file.name}:`, fileErr);

        }

      }

      if (allData.length === 0) {

        document.getElementById('result').innerHTML = `<b>❌ No records found for selected date interval.</b>`;

        return;

      }

      populateBuyerOptions();

      populateVehicleOptions();

      showProducts(allData);

    } catch (err) {

      console.error("Error fetching files list:", err);

      document.getElementById('result').innerHTML = `<b style="color:red;">Failed to fetch data. Check console for errors.</b>`;

    }

  }

  function clearFilters() {

    const buyerSelect = document.getElementById('buyerSelect');

    const vehicleSelect = document.getElementById('vehicleSelect');

    buyerSelect.innerHTML = '<option value="">-- Select Buyer --</option>';

    buyerSelect.disabled = true;

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    vehicleSelect.disabled = true;

  }

  function clearVehicleUsageInfo() {

    document.getElementById('vehicleUsageInfo').innerHTML = "";

  }

  // Populate buyer dropdown based on allData

  function populateBuyerOptions() {

    const buyerSelect = document.getElementById('buyerSelect');

    const buyersSet = new Set();

    allData.forEach(rec => {

      if (rec.buyerName) buyersSet.add(rec.buyerName.trim());

    });

    const buyers = Array.from(buyersSet).sort();

    if (buyers.length === 0) {

      buyerSelect.disabled = true;

      return;

    }

    buyers.forEach(b => {

      const opt = document.createElement('option');

      opt.value = b;

      opt.textContent = b;

      buyerSelect.appendChild(opt);

    });

    buyerSelect.disabled = false;

  }

  // Populate vehicle dropdown based on allData (just vehicle names only, no counts)

  function populateVehicleOptions() {

    const vehicleSelect = document.getElementById('vehicleSelect');

    const vehicleSet = new Set();

    allData.forEach(rec => {

      const vehicle = (rec.vehicle || "").trim();

      if (!vehicle) return;

      vehicleSet.add(vehicle);

    });

    const vehicles = Array.from(vehicleSet).sort();

    if (vehicles.length === 0) {

      vehicleSelect.disabled = true;

      return;

    }

    vehicles.forEach(veh => {

      const opt = document.createElement('option');

      opt.value = veh;

      opt.textContent = veh;  // only name here

      vehicleSelect.appendChild(opt);

    });

    vehicleSelect.disabled = false;

  }

  // Buyer dropdown change handler

  function onBuyerChange() {

    clearVehicleUsageInfo();

    const buyer = this.value;

    const vehicleSelect = document.getElementById('vehicleSelect');

    if (!buyer) {

      // No buyer selected, show all data

      populateVehicleOptions(); // repopulate all vehicles for allData

      showProducts(allData);

      return;

    }

    // Filter data by buyer

    const buyerData = allData.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

    if (buyerData.length === 0) {

      document.getElementById('result').innerHTML = `<b>No records found for buyer "${buyer}".</b>`;

      vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

      vehicleSelect.disabled = true;

      return;

    }

    // Populate vehicle dropdown only with vehicles used by this buyer

    const vehicleMap = new Map();

    buyerData.forEach(rec => {

      const vehicle = (rec.vehicle || "").trim();

      if (!vehicle) return;

      vehicleMap.set(vehicle, (vehicleMap.get(vehicle) || 0) + 1);

    });

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    if (vehicleMap.size > 0) {

      vehicleSelect.disabled = false;

      vehicleMap.forEach((count, veh) => {

        const timesText = count > 1 ? 'times' : 'time';

        const opt = document.createElement('option');

        opt.value = veh;

        opt.textContent = veh;  // still only name here as per new requirement

        vehicleSelect.appendChild(opt);

      });

    } else {

      vehicleSelect.disabled = true;

    }

    showProducts(buyerData);

  }

  // Vehicle dropdown change handler

  function onVehicleChange() {

    clearVehicleUsageInfo();

    const vehicle = this.value;

    const buyer = document.getElementById('buyerSelect').value;

    if (!vehicle) {

      // no vehicle selected, show filtered by buyer or all

      if (buyer) {

        const buyerData = allData.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

        showProducts(buyerData);

      } else {

        showProducts(allData);

      }

      return;

    }

    // Filter by vehicle (with or without buyer)

    let filtered = allData.filter(rec => (rec.vehicle || "").trim() === vehicle);

    if (buyer) {

      filtered = filtered.filter(rec => rec.buyerName && rec.buyerName.trim() === buyer);

    }

    if (filtered.length === 0) {

      document.getElementById('result').innerHTML = `<b>No records found for Vehicle "${vehicle}"${buyer ? ' and Buyer "' + buyer + '"' : ''}.</b>`;

      return;

    }

    // Count total times used

    const totalTimesUsed = filtered.length;

    // Find unique dates in dd-mm-yyyy format

    const uniqueDatesSet = new Set(filtered.map(r => {

      if (!r.date) return "";

      const parts = r.date.split('/');

      if (parts.length !== 3) return "";

      return `${parts[0].padStart(2,'0')}-${parts[1].padStart(2,'0')}-${parts[2]}`;

    }).filter(d => d !== ""));

    const uniqueDaysUsed = uniqueDatesSet.size;

    // Group usage by buyer & date with counts

    // structure: { buyerName: { date: count } }

    const usageMap = {};

    filtered.forEach(r => {

      const buyerName = r.buyerName ? r.buyerName.trim() : "Unknown";

      if (!usageMap[buyerName]) usageMap[buyerName] = {};

      const parts = r.date ? r.date.split('/') : [];

      if (parts.length !== 3) return;

      const dateStr = `${parts[0].padStart(2,'0')}-${parts[1].padStart(2,'0')}-${parts[2]}`;

      usageMap[buyerName][dateStr] = (usageMap[buyerName][dateStr] || 0) + 1;

    });

    // Build HTML summary

    const timesText = totalTimesUsed > 1 ? 'TIMES' : 'TIME';

    const daysText = uniqueDaysUsed > 1 ? 'DAYS' : 'DAY';

    let html = `<h3>Vehicle - "(${vehicle})"</h3>`;

    html += `<p><b>${totalTimesUsed} ${timesText} USED (${uniqueDaysUsed} UNIQUE ${daysText} USED)</b></p>`;

    html += `<table id="vehicleUsageTable"><thead><tr><th>BUYER(s)</th><th>DATE</th><th>USE TIME</th></tr></thead><tbody>`;

    // For better ordering, sort buyers by name, then dates asc

    const sortedBuyers = Object.keys(usageMap).sort((a,b) => a.localeCompare(b));

    sortedBuyers.forEach(buyerName => {

      const dates = Object.keys(usageMap[buyerName]).sort((a,b) => {

        // parse dates dd-mm-yyyy to Date for sorting

        const [d1,m1,y1] = a.split('-').map(x=>parseInt(x));

        const [d2,m2,y2] = b.split('-').map(x=>parseInt(x));

        return new Date(y1,m1-1,d1) - new Date(y2,m2-1,d2);

      });

      dates.forEach(dateStr => {

        const count = usageMap[buyerName][dateStr];

        const countText = count > 1 ? count : count;

        html += `<tr><td>${buyerName}</td><td>${dateStr}</td><td>${countText}</td></tr>`;

      });

    });

    html += `</tbody></table>`;

    document.getElementById('vehicleUsageInfo').innerHTML = html;

    // Show aggregated products for filtered data

    showProducts(filtered);

  }

  // Show aggregated products and total bags

  function showProducts(dataArray) {

    const productMap = {};

    dataArray.forEach(record => {

      if (!Array.isArray(record.products) || !Array.isArray(record.bags)) return;

      record.products.forEach((prod, i) => {

        const bags = parseInt(record.bags[i] || '0', 10);

        if (!prod) return;

        productMap[prod] = (productMap[prod] || 0) + bags;

      });

    });

    if (Object.keys(productMap).length === 0) {

      document.getElementById('result').innerHTML = `<b>No products found.</b>`;

      return;

    }

    let html = `<table>

      <thead><tr><th>Product</th><th>Bags</th></tr></thead>

      <tbody>`;

    Object.entries(productMap).forEach(([prod, bags]) => {

      html += `<tr><td>${prod}</td><td>${bags}</td></tr>`;

    });

    const totalBags = Object.values(productMap).reduce((a,b) => a+b, 0);

    html += `<tr class="total-row"><td>Total Bags</td><td>${totalBags}</td></tr>`;

    html += `</tbody></table>`;

    document.getElementById('result').innerHTML = html;

  }

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>Fetch Sold Products with Buyer & Vehicle Filter (Detailed Vehicle Usage)</title>

  <style>

    body { font-family: Arial, sans-serif; margin: 20px; background: #fafafa; }

    label { margin-right: 10px; }

    select, input[type=date] { margin-right: 20px; padding: 5px; }

    button { padding: 6px 12px; cursor: pointer; }

    table { border-collapse: collapse; width: 100%; margin-top: 20px; background: #fff; }

    th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

    th { background: #eee; }

    .total-row { font-weight: bold; }

    #result { margin-top: 20px; }

    #vehicleUsageInfo { display: none; margin-top: 10px; font-style: normal; background: #e8f0fe; padding: 10px; border-radius: 5px; }

    #vehicleUsageInfo.active { display: block; }

    #vehicleUsageInfo h3, #vehicleUsageInfo p { margin: 5px 0; }

    #vehicleUsageTable { margin-top: 10px; width: 100%; border-collapse: collapse; }

    #vehicleUsageTable th, #vehicleUsageTable td { border: 1px solid #bbb; padding: 6px; text-align: left; }

    #vehicleUsageTable th { background-color: #d0d9ff; }

  </style>

  <script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>

</head>

<body>

<h2>Fetch Sold Products by Date Interval with Buyer & Vehicle Filter</h2>

<div>

  <label for="fromDate">From Date:</label>

  <input type="date" id="fromDate" />

  <label for="toDate">To Date:</label>

  <input type="date" id="toDate" />

  <button id="fetchBtn">Fetch Products</button>

</div>

<div style="margin-top: 15px;">

  <label for="buyerSelect">Buyer:</label>

  <select id="buyerSelect" disabled>

    <option value="">-- Select Buyer --</option>

  </select>

  <label for="vehicleSelect">Vehicle:</label>

  <select id="vehicleSelect" disabled>

    <option value="">-- Select Vehicle --</option>

  </select>

</div>

<div id="vehicleUsageInfo"></div>

<div id="result"></div>

<script>

  const GITHUB\_USERNAME = "viralbalaji";

  const REPO\_NAME = "Product\_Sales";

  const REPO\_BRANCH = "main";

  const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

  const FOLDER\_PATH = "extracted\_data";

  let allData = [];

  document.getElementById('fetchBtn').addEventListener('click', fetchProductData);

  document.getElementById('buyerSelect').addEventListener('change', onBuyerChange);

  document.getElementById('vehicleSelect').addEventListener('change', onVehicleChange);

  const todayStr = new Date().toISOString().slice(0, 10);

  document.getElementById('fromDate').value = todayStr;

  document.getElementById('toDate').value = todayStr;

  async function fetchProductData() {

    allData = [];

    clearFilters();

    clearVehicleUsageInfo();

    document.getElementById('result').innerHTML = "Loading...";

    const from = dayjs(document.getElementById('fromDate').value);

    const to = dayjs(document.getElementById('toDate').value);

    if (!from.isValid() || !to.isValid() || to.isBefore(from)) {

      alert("Invalid date range.");

      return;

    }

    const headers = {

      Authorization: `token ${GITHUB\_TOKEN}`,

      Accept: "application/vnd.github.v3+json"

    };

    try {

      const listUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}?ref=${REPO\_BRANCH}`;

      const listResp = await fetch(listUrl, { headers });

      const files = await listResp.json();

      const jsonFiles = files.filter(f => f.name.endsWith('.json'));

      for (const file of jsonFiles) {

        const blobUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/git/blobs/${file.sha}`;

        const blobResp = await fetch(blobUrl, { headers });

        const blobData = await blobResp.json();

        const contentStr = atob(blobData.content.replace(/\n/g, ''));

        const data = JSON.parse(contentStr);

        if (!data.date) continue;

        const [dd, mm, yyyy] = data.date.split('/');

        const fileDate = dayjs(`${yyyy}-${mm}-${dd}`);

        if (fileDate.isBefore(from) || fileDate.isAfter(to)) continue;

        allData.push(data);

      }

      if (allData.length === 0) {

        document.getElementById('result').innerHTML = "<b>No data found in date range.</b>";

        return;

      }

      populateBuyerOptions();

      populateVehicleOptions(allData);

      showProducts(allData);

    } catch (err) {

      console.error(err);

      document.getElementById('result').innerHTML = "<b>Error loading data.</b>";

    }

  }

  function clearFilters() {

    const buyerSelect = document.getElementById('buyerSelect');

    const vehicleSelect = document.getElementById('vehicleSelect');

    buyerSelect.innerHTML = '<option value="">-- Select Buyer --</option>';

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    buyerSelect.disabled = true;

    vehicleSelect.disabled = true;

  }

  function clearVehicleUsageInfo() {

    const infoDiv = document.getElementById('vehicleUsageInfo');

    infoDiv.innerHTML = '';

    infoDiv.classList.remove('active');

  }

  function populateBuyerOptions() {

    const buyerSelect = document.getElementById('buyerSelect');

    const buyers = [...new Set(allData.map(r => r.buyerName?.trim()).filter(Boolean))].sort();

    buyers.forEach(b => {

      const opt = document.createElement('option');

      opt.value = b;

      opt.textContent = b;

      buyerSelect.appendChild(opt);

    });

    buyerSelect.disabled = buyers.length === 0;

  }

  function populateVehicleOptions(data) {

    const vehicleSelect = document.getElementById('vehicleSelect');

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    const vehicleCount = {};

    data.forEach(rec => {

      const v = (rec.vehicle || "").trim();

      if (v) vehicleCount[v] = (vehicleCount[v] || 0) + 1;

    });

    const sortedVehicles = Object.entries(vehicleCount)

      .sort((a, b) => b[1] - a[1]) // sort by most used

      .map(([vehicle]) => vehicle);

    sortedVehicles.forEach(v => {

      const opt = document.createElement('option');

      opt.value = v;

      opt.textContent = v;

      vehicleSelect.appendChild(opt);

    });

    vehicleSelect.disabled = sortedVehicles.length === 0;

  }

  function onBuyerChange() {

    clearVehicleUsageInfo();

    const buyer = this.value;

    const filtered = buyer

      ? allData.filter(r => r.buyerName?.trim() === buyer)

      : allData;

    populateVehicleOptions(filtered);

    showProducts(filtered);

  }

  function onVehicleChange() {

    const vehicle = this.value;

    const buyer = document.getElementById('buyerSelect').value;

    clearVehicleUsageInfo();

    let filtered = allData.filter(r => (r.vehicle || "").trim() === vehicle);

    if (buyer) filtered = filtered.filter(r => r.buyerName?.trim() === buyer);

    if (!vehicle) {

      showProducts(buyer ? allData.filter(r => r.buyerName?.trim() === buyer) : allData);

      return;

    }

    const infoDiv = document.getElementById('vehicleUsageInfo');

    infoDiv.classList.add('active');

    const totalTimesUsed = filtered.length;

    const uniqueDates = new Set(filtered.map(r => r.date)).size;

    let html = `<h3>Vehicle - "(${vehicle})"</h3>`;

    html += `<p><b>${totalTimesUsed} ${totalTimesUsed > 1 ? "TIMES" : "TIME"} USED (${uniqueDates} UNIQUE ${uniqueDates > 1 ? "DAYS" : "DAY"} USED)</b></p>`;

    html += `<table id="vehicleUsageTable"><thead><tr><th>BUYER</th><th>DATE</th><th>USE TIME</th></tr></thead><tbody>`;

    const usageMap = {};

    filtered.forEach(r => {

      const b = r.buyerName || "Unknown";

      const date = r.date || "??";

      usageMap[b] = usageMap[b] || {};

      usageMap[b][date] = (usageMap[b][date] || 0) + 1;

    });

    Object.entries(usageMap).forEach(([buyer, dates]) => {

      Object.entries(dates).forEach(([date, count]) => {

        html += `<tr><td>${buyer}</td><td>${date}</td><td>${count}</td></tr>`;

      });

    });

    html += "</tbody></table>";

    infoDiv.innerHTML = html;

    showProducts(filtered);

  }

  function showProducts(dataArray) {

    const map = {};

    dataArray.forEach(r => {

      (r.products || []).forEach((p, i) => {

        const b = parseInt(r.bags?.[i] || 0);

        if (!p) return;

        map[p] = (map[p] || 0) + b;

      });

    });

    if (Object.keys(map).length === 0) {

      document.getElementById('result').innerHTML = "<b>No products found.</b>";

      return;

    }

    let html = `<table><thead><tr><th>Product</th><th>Bags</th></tr></thead><tbody>`;

    let total = 0;

    for (const [prod, bags] of Object.entries(map)) {

      html += `<tr><td>${prod}</td><td>${bags}</td></tr>`;

      total += bags;

    }

    html += `<tr class="total-row"><td>Total Bags</td><td>${total}</td></tr>`;

    html += "</tbody></table>";

    document.getElementById('result').innerHTML = html;

  }

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>Fetch Sold Products with Buyer & Vehicle Filter (Detailed Vehicle Usage)</title>

  <style>

    body { font-family: Arial, sans-serif; margin: 20px; background: #fafafa; }

    label { margin-right: 10px; }

    select, input[type=date] { margin-right: 20px; padding: 5px; }

    button { padding: 6px 12px; cursor: pointer; }

    table { border-collapse: collapse; width: 100%; margin-top: 20px; background: #fff; }

    th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

    th { background: #eee; }

    .total-row { font-weight: bold; }

    #result { margin-top: 20px; }

    #vehicleUsageInfo { display: none; margin-top: 10px; font-style: normal; background: #e8f0fe; padding: 10px; border-radius: 5px; }

    #vehicleUsageInfo.active { display: block; }

    #vehicleUsageInfo h3, #vehicleUsageInfo p { margin: 5px 0; }

    #vehicleUsageTable { margin-top: 10px; width: 100%; border-collapse: collapse; }

    #vehicleUsageTable th, #vehicleUsageTable td { border: 1px solid #bbb; padding: 6px; text-align: left; }

    #vehicleUsageTable th { background-color: #d0d9ff; }

  </style>

  <script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>

  <script src="https://cdn.jsdelivr.net/npm/dayjs@1/plugin/isBetween.js"></script>

</head>

<body>

<h2>Fetch Sold Products by Date Interval with Buyer & Vehicle Filter</h2>

<div>

  <label for="fromDate">From Date:</label>

  <input type="date" id="fromDate" />

  <label for="toDate">To Date:</label>

  <input type="date" id="toDate" />

  <button id="fetchBtn">Fetch Products</button>

</div>

<div style="margin-top: 15px;">

  <label for="buyerSelect">Buyer:</label>

  <select id="buyerSelect" disabled>

    <option value="">-- Select Buyer --</option>

  </select>

  <label for="vehicleSelect">Vehicle:</label>

  <select id="vehicleSelect" disabled>

    <option value="">-- Select Vehicle --</option>

  </select>

</div>

<div id="vehicleUsageInfo"></div>

<div id="result"></div>

<script>

  const GITHUB\_USERNAME = "viralbalaji";

  const REPO\_NAME = "Product\_Sales";

  const REPO\_BRANCH = "main";

  const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

  const FOLDER\_PATH = "extracted\_data";

  let allData = [];

  document.getElementById('fetchBtn').addEventListener('click', fetchProductData);

  document.getElementById('buyerSelect').addEventListener('change', onBuyerChange);

  document.getElementById('vehicleSelect').addEventListener('change', onVehicleChange);

  const todayStr = new Date().toISOString().slice(0, 10);

  document.getElementById('fromDate').value = todayStr;

  document.getElementById('toDate').value = todayStr;

   dayjs.extend(window.dayjs\_plugin\_isBetween);

 async function fetchProductData() {

    allData = [];

    clearFilters();

    clearVehicleUsageInfo();

    document.getElementById('result').innerHTML = "Loading...";

    const from = dayjs(document.getElementById('fromDate').value);

    const to = dayjs(document.getElementById('toDate').value);

    if (!from.isValid() || !to.isValid() || to.isBefore(from)) {

      alert("Invalid date range.");

      return;

    }

    const headers = {

      Authorization: `token ${GITHUB\_TOKEN}`,

      Accept: "application/vnd.github.v3+json"

    };

    try {

      // 1. First get list of monthly folders

      const foldersUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}?ref=${REPO\_BRANCH}`;

      const foldersResp = await fetch(foldersUrl, { headers });

      const folders = await foldersResp.json();

      // Filter for only folders (type === 'dir') and exclude 'other' if needed

      const monthFolders = folders.filter(f => f.type === 'dir' && /^[A-Z]{3}\_\d{4}$/.test(f.name));

      // 2. Process each monthly folder

      for (const folder of monthFolders) {

        const folderUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}/${folder.name}?ref=${REPO\_BRANCH}`;

        const folderResp = await fetch(folderUrl, { headers });

        const files = await folderResp.json();

        const jsonFiles = files.filter(f => f.name.endsWith('.json'));

        // 3. Process each file in the monthly folder

        for (const file of jsonFiles) {

          const blobUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/git/blobs/${file.sha}`;

          const blobResp = await fetch(blobUrl, { headers });

          const blobData = await blobResp.json();

          const contentStr = atob(blobData.content.replace(/\n/g, ''));

          const data = JSON.parse(contentStr);

          if (!data.date) continue;

          // Parse date (handles both 16/5/2025 and 2025-05-16 formats)

          let fileDate;

          if (data.date.includes('/')) {

            const [dd, mm, yyyy] = data.date.split('/');

            fileDate = dayjs(`${yyyy}-${mm.padStart(2, '0')}-${dd.padStart(2, '0')}`);

          } else {

            fileDate = dayjs(data.date);

          }

          if (fileDate.isBetween(from, to, null, '[]')) { // inclusive range

            // Add folder info to the data for reference

            data.sourceFolder = folder.name;

            allData.push(data);

          }

        }

      }

      if (allData.length === 0) {

        document.getElementById('result').innerHTML = "<b>No data found in date range.</b>";

        return;

      }

      populateBuyerOptions();

      populateVehicleOptions(allData);

      showProducts(allData);

    } catch (err) {

      console.error(err);

      document.getElementById('result').innerHTML = `<b>Error loading data: ${err.message}</b>`;

    }

  }

  function clearFilters() {

    const buyerSelect = document.getElementById('buyerSelect');

    const vehicleSelect = document.getElementById('vehicleSelect');

    buyerSelect.innerHTML = '<option value="">-- Select Buyer --</option>';

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    buyerSelect.disabled = true;

    vehicleSelect.disabled = true;

  }

  function clearVehicleUsageInfo() {

    const infoDiv = document.getElementById('vehicleUsageInfo');

    infoDiv.innerHTML = '';

    infoDiv.classList.remove('active');

  }

  function populateBuyerOptions() {

    const buyerSelect = document.getElementById('buyerSelect');

    const buyers = [...new Set(allData.map(r => r.buyerName?.trim()).filter(Boolean))].sort();

    buyers.forEach(b => {

      const opt = document.createElement('option');

      opt.value = b;

      opt.textContent = b;

      buyerSelect.appendChild(opt);

    });

    buyerSelect.disabled = buyers.length === 0;

  }

  function populateVehicleOptions(data) {

    const vehicleSelect = document.getElementById('vehicleSelect');

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    const vehicleCount = {};

    data.forEach(rec => {

      const v = (rec.vehicle || "").trim();

      if (v) vehicleCount[v] = (vehicleCount[v] || 0) + 1;

    });

    const sortedVehicles = Object.entries(vehicleCount)

      .sort((a, b) => b[1] - a[1]) // sort by most used

      .map(([vehicle]) => vehicle);

    sortedVehicles.forEach(v => {

      const opt = document.createElement('option');

      opt.value = v;

      opt.textContent = v;

      vehicleSelect.appendChild(opt);

    });

    vehicleSelect.disabled = sortedVehicles.length === 0;

  }

  function onBuyerChange() {

    clearVehicleUsageInfo();

    const buyer = this.value;

    const filtered = buyer

      ? allData.filter(r => r.buyerName?.trim() === buyer)

      : allData;

    populateVehicleOptions(filtered);

    showProducts(filtered);

  }

  function onVehicleChange() {

    const vehicle = this.value;

    const buyer = document.getElementById('buyerSelect').value;

    clearVehicleUsageInfo();

    let filtered = allData.filter(r => (r.vehicle || "").trim() === vehicle);

    if (buyer) filtered = filtered.filter(r => r.buyerName?.trim() === buyer);

    if (!vehicle) {

      showProducts(buyer ? allData.filter(r => r.buyerName?.trim() === buyer) : allData);

      return;

    }

    const infoDiv = document.getElementById('vehicleUsageInfo');

    infoDiv.classList.add('active');

    const totalTimesUsed = filtered.length;

    const uniqueDates = new Set(filtered.map(r => r.date)).size;

    let html = `<h3>Vehicle - "(${vehicle})"</h3>`;

    html += `<p><b>${totalTimesUsed} ${totalTimesUsed > 1 ? "TIMES" : "TIME"} USED (${uniqueDates} UNIQUE ${uniqueDates > 1 ? "DAYS" : "DAY"} USED)</b></p>`;

    html += `<table id="vehicleUsageTable"><thead><tr><th>BUYER</th><th>DATE</th><th>USE TIME</th></tr></thead><tbody>`;

    const usageMap = {};

    filtered.forEach(r => {

      const b = r.buyerName || "Unknown";

      const date = r.date || "??";

      usageMap[b] = usageMap[b] || {};

      usageMap[b][date] = (usageMap[b][date] || 0) + 1;

    });

    Object.entries(usageMap).forEach(([buyer, dates]) => {

      Object.entries(dates).forEach(([date, count]) => {

        html += `<tr><td>${buyer}</td><td>${date}</td><td>${count}</td></tr>`;

      });

    });

    html += "</tbody></table>";

    infoDiv.innerHTML = html;

    showProducts(filtered);

  }

  function showProducts(dataArray) {

    const map = {};

    dataArray.forEach(r => {

      (r.products || []).forEach((p, i) => {

        const b = parseInt(r.bags?.[i] || 0);

        if (!p) return;

        map[p] = (map[p] || 0) + b;

      });

    });

    if (Object.keys(map).length === 0) {

      document.getElementById('result').innerHTML = "<b>No products found.</b>";

      return;

    }

    let html = `<table><thead><tr><th>Product</th><th>Bags</th></tr></thead><tbody>`;

    let total = 0;

    for (const [prod, bags] of Object.entries(map)) {

      html += `<tr><td>${prod}</td><td>${bags}</td></tr>`;

      total += bags;

    }

    html += `<tr class="total-row"><td>Total Bags</td><td>${total}</td></tr>`;

    html += "</tbody></table>";

    document.getElementById('result').innerHTML = html;

  }

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>Fetch Sold Products with Buyer & Vehicle Filter (Detailed Vehicle Usage)</title>

  <style>

    body { font-family: Arial, sans-serif; margin: 20px; background: #fafafa; }

    label { margin-right: 10px; }

    select, input[type=date] { margin-right: 20px; padding: 5px; }

    button { padding: 6px 12px; cursor: pointer; }

    table { border-collapse: collapse; width: 100%; margin-top: 20px; background: #fff; }

    th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }

    th { background: #eee; }

    .total-row { font-weight: bold; }

    #result { margin-top: 20px; }

    #vehicleUsageInfo { display: none; margin-top: 10px; font-style: normal; background: #e8f0fe; padding: 10px; border-radius: 5px; }

    #vehicleUsageInfo.active { display: block; }

    #vehicleUsageInfo h3, #vehicleUsageInfo p { margin: 5px 0; }

    #vehicleUsageTable { margin-top: 10px; width: 100%; border-collapse: collapse; }

    #vehicleUsageTable th, #vehicleUsageTable td { border: 1px solid #bbb; padding: 6px; text-align: left; }

    #vehicleUsageTable th { background-color: #d0d9ff; }

  </style>

  <script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>

  <script src="https://cdn.jsdelivr.net/npm/dayjs@1/plugin/isBetween.js"></script>

</head>

<body>

<h2>Fetch Sold Products by Date Interval with Buyer & Vehicle Filter</h2>

<div>

  <label for="fromDate">From Date:</label>

  <input type="date" id="fromDate" />

  <label for="toDate">To Date:</label>

  <input type="date" id="toDate" />

  <button id="fetchBtn">Fetch Products</button>

</div>

<div style="margin-top: 15px;">

  <label for="buyerSelect">Buyer:</label>

  <select id="buyerSelect" disabled>

    <option value="">-- Select Buyer --</option>

  </select>

  <label for="vehicleSelect">Vehicle:</label>

  <select id="vehicleSelect" disabled>

    <option value="">-- Select Vehicle --</option>

  </select>

</div>

<div id="vehicleUsageInfo"></div>

<div id="result"></div>

<script>

  const GITHUB\_USERNAME = "viralbalaji";

  const REPO\_NAME = "Product\_Sales";

  const REPO\_BRANCH = "main";

  const GITHUB\_TOKEN = "ghp\_Wo7ydDzQrHcFg4EMyTfJtunzjm4jmC4EdQ3E";

  const FOLDER\_PATH = "extracted\_data";

  let allData = [];

  document.getElementById('fetchBtn').addEventListener('click', fetchProductData);

  document.getElementById('buyerSelect').addEventListener('change', onBuyerChange);

  document.getElementById('vehicleSelect').addEventListener('change', onVehicleChange);

  const todayStr = new Date().toISOString().slice(0, 10);

  document.getElementById('fromDate').value = todayStr;

  document.getElementById('toDate').value = todayStr;

   dayjs.extend(window.dayjs\_plugin\_isBetween);

 async function fetchProductData() {

    allData = [];

    clearFilters();

    clearVehicleUsageInfo();

    document.getElementById('result').innerHTML = "Loading...";

    const from = dayjs(document.getElementById('fromDate').value);

    const to = dayjs(document.getElementById('toDate').value);

    if (!from.isValid() || !to.isValid() || to.isBefore(from)) {

      alert("Invalid date range.");

      return;

    }

    const headers = {

      Authorization: `token ${GITHUB\_TOKEN}`,

      Accept: "application/vnd.github.v3+json"

    };

    try {

      // 1. First get list of monthly folders

      const foldersUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}?ref=${REPO\_BRANCH}`;

      const foldersResp = await fetch(foldersUrl, { headers });

      const folders = await foldersResp.json();

      // Filter for only folders (type === 'dir') and exclude 'other' if needed

      const monthFolders = folders.filter(f => f.type === 'dir' && /^[A-Z]{3}\_\d{4}$/.test(f.name));

      // 2. Process each monthly folder

      for (const folder of monthFolders) {

        const folderUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${FOLDER\_PATH}/${folder.name}?ref=${REPO\_BRANCH}`;

        const folderResp = await fetch(folderUrl, { headers });

        const files = await folderResp.json();

        const jsonFiles = files.filter(f => f.name.endsWith('.json'));

        // 3. Process each file in the monthly folder

        for (const file of jsonFiles) {

          const blobUrl = `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/git/blobs/${file.sha}`;

          const blobResp = await fetch(blobUrl, { headers });

          const blobData = await blobResp.json();

          const contentStr = atob(blobData.content.replace(/\n/g, ''));

          const data = JSON.parse(contentStr);

          if (!data.date) continue;

          // Parse date (handles both 16/5/2025 and 2025-05-16 formats)

          let fileDate;

          if (data.date.includes('/')) {

            const [dd, mm, yyyy] = data.date.split('/');

            fileDate = dayjs(`${yyyy}-${mm.padStart(2, '0')}-${dd.padStart(2, '0')}`);

          } else {

            fileDate = dayjs(data.date);

          }

          if (fileDate.isBetween(from, to, null, '[]')) { // inclusive range

            // Add folder info to the data for reference

            data.sourceFolder = folder.name;

            allData.push(data);

          }

        }

      }

      if (allData.length === 0) {

        document.getElementById('result').innerHTML = "<b>No data found in date range.</b>";

        return;

      }

      populateBuyerOptions();

      populateVehicleOptions(allData);

      showProducts(allData);

    } catch (err) {

      console.error(err);

      document.getElementById('result').innerHTML = `<b>Error loading data: ${err.message}</b>`;

    }

  }

  function clearFilters() {

    const buyerSelect = document.getElementById('buyerSelect');

    const vehicleSelect = document.getElementById('vehicleSelect');

    buyerSelect.innerHTML = '<option value="">-- Select Buyer --</option>';

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    buyerSelect.disabled = true;

    vehicleSelect.disabled = true;

  }

  function clearVehicleUsageInfo() {

    const infoDiv = document.getElementById('vehicleUsageInfo');

    infoDiv.innerHTML = '';

    infoDiv.classList.remove('active');

  }

  function populateBuyerOptions() {

    const buyerSelect = document.getElementById('buyerSelect');

    const buyers = [...new Set(allData.map(r => r.buyerName?.trim()).filter(Boolean))].sort();

    buyers.forEach(b => {

      const opt = document.createElement('option');

      opt.value = b;

      opt.textContent = b;

      buyerSelect.appendChild(opt);

    });

    buyerSelect.disabled = buyers.length === 0;

  }

  function populateVehicleOptions(data) {

    const vehicleSelect = document.getElementById('vehicleSelect');

    vehicleSelect.innerHTML = '<option value="">-- Select Vehicle --</option>';

    const vehicleCount = {};

    data.forEach(rec => {

      const v = (rec.vehicle || "").trim();

      if (v) vehicleCount[v] = (vehicleCount[v] || 0) + 1;

    });

    const sortedVehicles = Object.entries(vehicleCount)

      .sort((a, b) => b[1] - a[1]) // sort by most used

      .map(([vehicle]) => vehicle);

    sortedVehicles.forEach(v => {

      const opt = document.createElement('option');

      opt.value = v;

      opt.textContent = v;

      vehicleSelect.appendChild(opt);

    });

    vehicleSelect.disabled = sortedVehicles.length === 0;

  }

  function onBuyerChange() {

    clearVehicleUsageInfo();

    const buyer = this.value;

    const filtered = buyer

      ? allData.filter(r => r.buyerName?.trim() === buyer)

      : allData;

    populateVehicleOptions(filtered);

    showProducts(filtered);

  }

  function onVehicleChange() {

    const vehicle = this.value;

    const buyer = document.getElementById('buyerSelect').value;

    clearVehicleUsageInfo();

    let filtered = allData.filter(r => (r.vehicle || "").trim() === vehicle);

    if (buyer) filtered = filtered.filter(r => r.buyerName?.trim() === buyer);

    if (!vehicle) {

      showProducts(buyer ? allData.filter(r => r.buyerName?.trim() === buyer) : allData);

      return;

    }

    const infoDiv = document.getElementById('vehicleUsageInfo');

    infoDiv.classList.add('active');

    const totalTimesUsed = filtered.length;

    const uniqueDates = new Set(filtered.map(r => r.date)).size;

    let html = `<h3>Vehicle - "(${vehicle})"</h3>`;

    html += `<p><b>${totalTimesUsed} ${totalTimesUsed > 1 ? "TIMES" : "TIME"} USED (${uniqueDates} UNIQUE ${uniqueDates > 1 ? "DAYS" : "DAY"} USED)</b></p>`;

    html += `<table id="vehicleUsageTable"><thead><tr><th>BUYER</th><th>DATE</th><th>USE TIME</th></tr></thead><tbody>`;

    const usageMap = {};

    filtered.forEach(r => {

      const b = r.buyerName || "Unknown";

      const date = r.date || "??";

      usageMap[b] = usageMap[b] || {};

      usageMap[b][date] = (usageMap[b][date] || 0) + 1;

    });

    Object.entries(usageMap).forEach(([buyer, dates]) => {

      Object.entries(dates).forEach(([date, count]) => {

        html += `<tr><td>${buyer}</td><td>${date}</td><td>${count}</td></tr>`;

      });

    });

    html += "</tbody></table>";

    infoDiv.innerHTML = html;

    showProducts(filtered);

  }

  function showProducts(dataArray) {

    const map = {};

    dataArray.forEach(r => {

      (r.products || []).forEach((p, i) => {

        const b = parseInt(r.bags?.[i] || 0);

        if (!p) return;

        map[p] = (map[p] || 0) + b;

      });

    });

    if (Object.keys(map).length === 0) {

      document.getElementById('result').innerHTML = "<b>No products found.</b>";

      return;

    }

    let html = `<table><thead><tr><th>Product</th><th>Bags</th></tr></thead><tbody>`;

    let total = 0;

   const sorted = Object.entries(map).sort((a, b) => b[1] - a[1]); // sort by bags DESC

for (const [prod, bags] of sorted) {

  html += `<tr><td>${prod}</td><td>${bags}</td></tr>`;

  total += bags;

}

    html += `<tr class="total-row"><td>Total Bags</td><td>${total}</td></tr>`;

    html += "</tbody></table>";

    document.getElementById('result').innerHTML = html;

  }

</script>

</body>

</html>