<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Excel Multi-Extract & GitHub Upload</title>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    #fileList {

      display: flex;

      gap: 15px;

      margin-top: 15px;

      overflow-x: auto;

      padding-bottom: 10px;

      border-bottom: 2px solid #ddd;

    }

    .file-item {

      display: flex;

      align-items: center;

      gap: 8px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 6px 12px;

      background: #f9f9f9;

      white-space: nowrap;

      position: relative;

    }

    .file-item img {

      cursor: default;

      user-select: none;

    }

    .file-name {

      max-width: 260px;

      overflow: hidden;

      text-overflow: ellipsis;

      font-size: 17px;

      white-space: nowrap;

      font-weight: 620;

    }

      .remove-btn {

      cursor: pointer;

      color: white;

      background: red;

      border: none;

      border-radius: 50%;

      width: 20px;

      height: 20px;

      font-weight: bold;

      line-height: 18px;

      text-align: center;

      user-select: none;

    }

    #extractAllBtn, #uploadAllBtn {

      margin-top: 15px;

      padding: 10px 25px;

      font-size: 1rem;

      cursor: pointer;

    }

    pre {

      background: #efefef;

      padding: 10px;

      max-height: 300px;

      overflow-y: auto;

      margin-top: 20px;

      border-radius: 6px;

      white-space: pre-wrap;

      word-break: break-word;

    }

    .result-block {

      margin-top: 15px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 10px;

      background: #fafafa;

    }

    .status {

      margin-top: 6px;

      font-size: 0.9rem;

      font-weight: 600;

    }

  </style>

</head>

<body>

  <h2>Upload Multiple Excel Files</h2>

  <input type="file" id="fileInput" multiple accept=".xls,.xlsx" />

  <div id="fileList"></div>

  <button id="extractAllBtn" disabled>Extract All Files</button>

  <button id="uploadAllBtn" disabled>Upload All Extracted Data to GitHub</button>

  <h3>Extraction Results:</h3>

  <div id="results"></div>

  <script>

    // 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";

    // Excel logo url & logo size

    const excelLogoUrl = "https://cdn-icons-png.flaticon.com/512/732/732220.png";

    const logoWidth = 24;

    const logoHeight = 24;

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

    const fileListDiv = document.getElementById('fileList');

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

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

    const resultsDiv = document.getElementById('results');

    // Array to keep track of files

    let filesArray = [];

    // Store extracted data per file name

    let extractedDataMap = new Map();

    // Utility: Convert Excel serial date to ISO yyyy-mm-dd

    function excelDateToJSDate(serial) {

      if (typeof serial === "number") {

        const utc\_days = Math.floor(serial - 25569);

        const utc\_value = utc\_days \* 86400;

        const date\_info = new Date(utc\_value \* 1000);

        return date\_info.toISOString().split("T")[0];

      }

      return serial;

    }

    // Utility: check truly empty values

    function isTrulyEmpty(value) {

      return (

        value === undefined ||

        value === null ||

        value === "" ||

        (typeof value === "string" && value.trim() === "")

      );

    }

    // Render file list with logo, filename, remove btn

    function renderFileList() {

      fileListDiv.innerHTML = '';

      if (filesArray.length === 0) {

        extractAllBtn.disabled = true;

        uploadAllBtn.disabled = true;

        resultsDiv.innerHTML = '';

        extractedDataMap.clear();

        return;

      }

      extractAllBtn.disabled = false;

      uploadAllBtn.disabled = true; // only enable after extraction

      filesArray.forEach((file, index) => {

        const fileItem = document.createElement('div');

        fileItem.className = 'file-item';

        const img = document.createElement('img');

        img.src = excelLogoUrl;

        img.width = logoWidth;

        img.height = logoHeight;

        img.alt = "Excel Logo";

        const fileNameSpan = document.createElement('span');

        fileNameSpan.className = 'file-name';

        fileNameSpan.title = file.name; // show full on hover

        fileNameSpan.textContent = file.name;

        const removeBtn = document.createElement('button');

        removeBtn.className = 'remove-btn';

        removeBtn.title = "Remove this file";

        removeBtn.textContent = "×";

        removeBtn.onclick = () => {

          filesArray.splice(index, 1);

          extractedDataMap.delete(file.name);

          renderFileList();

          renderExtractionResults();

        };

        fileItem.appendChild(img);

        fileItem.appendChild(fileNameSpan);

        fileItem.appendChild(removeBtn);

        fileListDiv.appendChild(fileItem);

      });

    }

    // Extract data from one file

    function extractDataFromFile(file) {

      return new Promise((resolve, reject) => {

        const reader = new FileReader();

        reader.onload = function(e) {

          try {

            const data = new Uint8Array(e.target.result);

            const workbook = XLSX.read(data, {type: "array", cellDates: true});

            const sheetName = workbook.SheetNames[0];

            const sheet = workbook.Sheets[sheetName];

            const jsonArray = XLSX.utils.sheet\_to\_json(sheet, {header:1, raw:false});

            const getCell = (r, c) => (jsonArray[r] && jsonArray[r][c]) || "";

            const buyerName = getCell(1,1);

            const challanNumber = getCell(1,6);

            const vehicle = getCell(2,3);

            const rawDate = getCell(2,5);

            const date = (rawDate instanceof Date) ? rawDate.toISOString().split("T")[0] : excelDateToJSDate(rawDate);

            function extractColRange(colIndex) {

              let result = [];

              for(let i=4; i<=21; i++) {

                const val = getCell(i,colIndex);

                if(!isTrulyEmpty(val) && val !== "0" && val !== 0) result.push(val);

              }

              return result.length > 0 ? result : undefined;

            }

            const products = extractColRange(1);

            const bags = extractColRange(3);

            const quantities = extractColRange(4);

            const rates = extractColRange(5);

            const amounts = extractColRange(6);

            const subtotalText = getCell(22,4);

            const subtotalValue = getCell(22,6);

            const totalBags = getCell(23,1);

            const discountPercent = getCell(23,5);

            const discountValue = getCell(23,6);

            const totalQty = getCell(24,1);

            const total = getCell(24,5);

            let extracted = {};

            if(!isTrulyEmpty(buyerName)) extracted.buyerName = buyerName;

            if(!isTrulyEmpty(challanNumber)) extracted.challanNumber = challanNumber;

            if(!isTrulyEmpty(vehicle) && vehicle !== "0") extracted.vehicle = vehicle;

            if(!isTrulyEmpty(date)) extracted.date = date;

            if(products) extracted.products = products;

            if(bags) extracted.bags = bags;

            if(quantities) extracted.quantities = quantities;

            if(rates) extracted.rates = rates;

            if(amounts) extracted.amounts = amounts;

            if(!isTrulyEmpty(subtotalText)) extracted.subtotalText = subtotalText;

            if(!isTrulyEmpty(subtotalValue) && subtotalValue !== "0") extracted.subtotalValue = subtotalValue;

            if(!isTrulyEmpty(totalBags)) extracted.totalBags = totalBags;

            if(!isTrulyEmpty(discountPercent) && discountPercent !== "0") extracted.discountPercent = discountPercent;

            if(!isTrulyEmpty(discountValue) && discountValue !== "0") extracted.discountValue = discountValue;

            if(!isTrulyEmpty(totalQty)) extracted.totalQuantity = totalQty;

            if(!isTrulyEmpty(total) && total !== "0") extracted.total = total;

            resolve({fileName: file.name, data: extracted});

          } catch(err) {

            reject(`Error processing file "${file.name}": ${err.message}`);

          }

        };

        reader.onerror = () => reject(`Failed to read file "${file.name}"`);

        reader.readAsArrayBuffer(file);

      });

    }

    // Render extracted results

    function renderExtractionResults() {

      resultsDiv.innerHTML = '';

      if (extractedDataMap.size === 0) {

        uploadAllBtn.disabled = true;

        return;

      }

      uploadAllBtn.disabled = false;

      extractedDataMap.forEach((data, filename) => {

        const block = document.createElement('div');

        block.className = 'result-block';

       const titleContainer = document.createElement('div');

titleContainer.style.display = 'flex';

titleContainer.style.alignItems = 'center';

titleContainer.style.gap = '10px';

titleContainer.style.marginBottom = '8px';

const logoImg = document.createElement('img');

logoImg.src = excelLogoUrl;

logoImg.width = logoWidth;

logoImg.height = logoHeight;

logoImg.alt = "Excel Logo";

const title = document.createElement('span');

title.textContent = filename;

title.style.fontSize = '17px';

title.style.fontWeight = '584';

title.style.maxWidth = '350px';

title.style.overflow = 'hidden';

title.style.textOverflow = 'ellipsis';

title.style.whiteSpace = 'nowrap';

titleContainer.appendChild(logoImg);

titleContainer.appendChild(title);

block.appendChild(titleContainer);

        const pre = document.createElement('pre');

        pre.textContent = JSON.stringify(data, null, 2);

        block.appendChild(pre);

        resultsDiv.appendChild(block);

      });

    }

    // Upload JSON file to GitHub repo (creates or updates)

    async function uploadFileToGitHub(filename, contentJson) {

      const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

      // First, get the SHA if file exists (to update)

      let sha;

      try {

        const resp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`, {

          headers: {

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

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

          },

        });

        if (resp.status === 200) {

          const json = await resp.json();

          sha = json.sha;

        }

      } catch(e) {

        // file may not exist, ignore

      }

      // Prepare commit message and body

      const body = {

        message: `Add/update extracted data file: ${filename}`,

        content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))), // base64 encode

        branch: REPO\_BRANCH,

      };

      if (sha) body.sha = sha;

      const putResp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`, {

        method: "PUT",

        headers: {

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

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

          "Content-Type": "application/json",

        },

        body: JSON.stringify(body),

      });

      if (!putResp.ok) {

        const errJson = await putResp.json();

        throw new Error(`GitHub API error: ${putResp.status} ${putResp.statusText} - ${errJson.message}`);

      }

      return await putResp.json();

    }

    // Event: file input change

    fileInput.addEventListener('change', function(e) {

      const newFiles = Array.from(e.target.files);

      newFiles.forEach(f => {

        if (!filesArray.some(existing => existing.name === f.name)) {

          filesArray.push(f);

        }

      });

      renderFileList();

      extractedDataMap.clear();

      resultsDiv.innerHTML = '';

      extractAllBtn.disabled = filesArray.length === 0;

      uploadAllBtn.disabled = true;

      // Clear input to allow re-upload same file if removed

      fileInput.value = "";

    });

    // Extract all files button

    extractAllBtn.addEventListener('click', async () => {

      resultsDiv.innerHTML = 'Extracting... Please wait.';

      extractAllBtn.disabled = true;

      uploadAllBtn.disabled = true;

      extractedDataMap.clear();

      try {

        for (const file of filesArray) {

          const result = await extractDataFromFile(file);

          extractedDataMap.set(result.fileName, result.data);

        }

        renderExtractionResults();

      } catch (err) {

        resultsDiv.innerHTML = `<span style="color:red;">Error: ${err}</span>`;

      } finally {

        extractAllBtn.disabled = false;

      }

    });

   // Upload all extracted data button

uploadAllBtn.addEventListener('click', async () => {

  if (extractedDataMap.size === 0) {

    alert("No extracted data to upload. Please extract first.");

    return;

  }

  uploadAllBtn.disabled = true;

  extractAllBtn.disabled = true;

  resultsDiv.innerHTML = '';

  const uploadStatusContainer = document.createElement('div');

  uploadStatusContainer.style.marginTop = "15px";

  uploadStatusContainer.style.display = "flex";

  uploadStatusContainer.style.flexDirection = "column";

  uploadStatusContainer.style.gap = "10px";

  resultsDiv.appendChild(uploadStatusContainer);

  const statusElements = new Map();

  for (const [filename] of extractedDataMap.entries()) {

    const statusBlock = document.createElement('div');

    statusBlock.className = 'file-item';

    const logo = document.createElement('img');

    logo.src = excelLogoUrl;

    logo.width = logoWidth;

    logo.height = logoHeight;

    logo.alt = "Excel Logo";

    const name = document.createElement('span');

    name.className = 'file-name';

    name.textContent = filename;

    const spinner = document.createElement('div');

    spinner.className = 'spinner';

    spinner.style.width = "18px";

    spinner.style.height = "18px";

    spinner.style.border = "3px solid #000";

    spinner.style.borderTop = "3px solid transparent";

    spinner.style.borderRadius = "50%";

    spinner.style.animation = "spin 0.8s linear infinite";

    statusBlock.appendChild(logo);

    statusBlock.appendChild(name);

    statusBlock.appendChild(spinner);

    uploadStatusContainer.appendChild(statusBlock);

    statusElements.set(filename, { block: statusBlock, spinner });

  }

  let uploadErrors = [];

  let uploadedCount = 0;

  let updatedCount = 0;

  // Modified uploadFileToGitHub to return info about commit message & whether it was update or new file

  async function uploadFileToGitHubWithInfo(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  let sha;

  let isUpdate = false;

  try {

    const resp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`, {

      headers: {

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

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

      },

    });

    if (resp.status === 200) {

      const json = await resp.json();

      sha = json.sha;

      isUpdate = true;

    }

  } catch (e) {

    // ignore file not found etc.

  }

  const commitMessage = isUpdate

    ? `UPDATED: extracted data file - ${filename}`

    : `UPLOADED: extracted data file - ${filename}`;

  const body = {

    message: commitMessage,

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))), // base64 encode

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;

  const putResp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`, {

    method: "PUT",

    headers: {

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

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

      "Content-Type": "application/json",

    },

    body: JSON.stringify(body),

  });

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} ${putResp.statusText} - ${errJson.message}`);

  }

  const jsonResp = await putResp.json();

  return { commitMessage, isUpdate, jsonResp };

}

  for (const [filename, data] of extractedDataMap.entries()) {

    try {

      const { commitMessage, isUpdate } = await uploadFileToGitHubWithInfo(filename, data);

     const status = statusElements.get(filename);

status.block.removeChild(status.spinner);

// Add colored tick only

const tick = document.createElement('span');

tick.textContent = "✅";

tick.style.fontWeight = "bold";

tick.style.fontSize = "17px";

tick.style.marginLeft = "8px";

tick.style.userSelect = "none";

tick.style.color = isUpdate ? "black" : "green";

status.block.appendChild(tick);

      if (isUpdate) updatedCount++;

      else uploadedCount++;

    } catch (e) {

      const status = statusElements.get(filename);

      status.block.removeChild(status.spinner);

      const cross = document.createElement('span');

      cross.textContent = "❌";

      cross.style.fontSize = "18px";

      status.block.appendChild(cross);

      uploadErrors.push(`Failed to upload ${filename}: ${e.message}`);

    }

  }

  if (uploadErrors.length > 0) {

    const errMsg = document.createElement('div');

    errMsg.style.color = "red";

    errMsg.style.marginTop = "10px";

    errMsg.style.whiteSpace = "pre-line";

    errMsg.textContent = "Some errors occurred:\n" + uploadErrors.join("\n");

    resultsDiv.appendChild(errMsg);

  }

  // Bottom summary message

  const summaryMsg = document.createElement('div');

  summaryMsg.style.marginTop = "10px";

  summaryMsg.style.fontWeight = "bold";

  if (uploadErrors.length === 0) {

    if (uploadedCount > 0 && updatedCount === 0) {

      summaryMsg.textContent = `✅ All uploads processed. (${uploadedCount} file${uploadedCount > 1 ? "s" : ""})`;

      summaryMsg.style.color = "green"

      summaryMsg.style.fontWeight = "bold"

    } else if (updatedCount > 0 && uploadedCount === 0) {

      summaryMsg.textContent = `✅ All updates processed. (${updatedCount} file${updatedCount > 1 ? "s" : ""})`;

      summaryMsg.style.color = "black"

      summaryMsg.style.fontWeight = "bold"

    } else {

      summaryMsg.textContent = `✅ All uploads and updates processed. (${uploadedCount} upload${uploadedCount !== 1 ? "s" : ""}, ${updatedCount} update${updatedCount !== 1 ? "s" : ""})`;

    }

  } else {

    summaryMsg.textContent = `Process completed with errors.`;

    summaryMsg.style.color = "red";

  }

  resultsDiv.appendChild(summaryMsg);

  uploadAllBtn.disabled = false;

  extractAllBtn.disabled = false;

});

    // Initial render

    renderFileList();

  </script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Excel Multi-Extract & GitHub Upload</title>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    #fileList {

      display: flex;

      gap: 15px;

      margin-top: 15px;

      overflow-x: auto;

      padding-bottom: 10px;

      border-bottom: 2px solid #ddd;

    }

    .file-item {

      display: flex;

      align-items: center;

      gap: 8px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 6px 12px;

      background: #f9f9f9;

      white-space: nowrap;

      position: relative;

    }

    .file-item img {

      cursor: default;

      user-select: none;

    }

    .file-name {

      max-width: 305px;

      overflow: hidden;

      text-overflow: ellipsis;

      font-size: 17px;

      white-space: nowrap;

      font-weight: 620;

    }

      .remove-btn {

      cursor: pointer;

      color: white;

      background: red;

      border: none;

      border-radius: 50%;

      width: 20px;

      height: 20px;

      font-weight: bold;

      line-height: 18px;

      text-align: center;

      user-select: none;

    }

    #extractAllBtn, #uploadAllBtn {

      margin-top: 15px;

      padding: 10px 25px;

      font-size: 1rem;

      cursor: pointer;

    }

    pre {

      background: #efefef;

      padding: 10px;

      max-height: 300px;

      overflow-y: auto;

      margin-top: 20px;

      border-radius: 6px;

      white-space: pre-wrap;

      word-break: break-word;

    }

    .result-block {

      margin-top: 15px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 10px;

      background: #fafafa;

    }

    .status {

      margin-top: 6px;

      font-size: 0.9rem;

      font-weight: 600;

    }

  </style>

</head>

<body>

  <h2>Upload Multiple Excel Files</h2>

  <input type="file" id="fileInput" multiple accept=".xls,.xlsx" />

  <div id="fileList"></div>

  <button id="extractAllBtn" disabled>Extract All Files</button>

  <button id="uploadAllBtn" disabled>Upload All Extracted Data to GitHub</button>

  <h3>Extraction Results:</h3>

  <div id="results"></div>

  <script>

    // 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";

    // Excel logo url & logo size

    const excelLogoUrl = "https://cdn-icons-png.flaticon.com/512/732/732220.png";

    const logoWidth = 24;

    const logoHeight = 24;

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

    const fileListDiv = document.getElementById('fileList');

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

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

    const resultsDiv = document.getElementById('results');

    // Array to keep track of files

    let filesArray = [];

    // Store extracted data per file name

    let extractedDataMap = new Map();

    // Utility: Convert Excel serial date to ISO yyyy-mm-dd

    function excelDateToJSDate(serial) {

      if (typeof serial === "number") {

        const utc\_days = Math.floor(serial - 25569);

        const utc\_value = utc\_days \* 86400;

        const date\_info = new Date(utc\_value \* 1000);

        return date\_info.toISOString().split("T")[0];

      }

      return serial;

    }

    // Utility: check truly empty values

    function isTrulyEmpty(value) {

      return (

        value === undefined ||

        value === null ||

        value === "" ||

        (typeof value === "string" && value.trim() === "")

      );

    }

    // Render file list with logo, filename, remove btn

    function renderFileList() {

      fileListDiv.innerHTML = '';

      if (filesArray.length === 0) {

        extractAllBtn.disabled = true;

        uploadAllBtn.disabled = true;

        resultsDiv.innerHTML = '';

        extractedDataMap.clear();

        return;

      }

      extractAllBtn.disabled = false;

      uploadAllBtn.disabled = true; // only enable after extraction

      filesArray.forEach((file, index) => {

        const fileItem = document.createElement('div');

        fileItem.className = 'file-item';

        const img = document.createElement('img');

        img.src = excelLogoUrl;

        img.width = logoWidth;

        img.height = logoHeight;

        img.alt = "Excel Logo";

        const fileNameSpan = document.createElement('span');

        fileNameSpan.className = 'file-name';

        fileNameSpan.title = file.name; // show full on hover

        fileNameSpan.textContent = file.name;

        const removeBtn = document.createElement('button');

        removeBtn.className = 'remove-btn';

        removeBtn.title = "Remove this file";

        removeBtn.textContent = "×";

        removeBtn.onclick = () => {

          filesArray.splice(index, 1);

          extractedDataMap.delete(file.name);

          renderFileList();

          renderExtractionResults();

        };

        fileItem.appendChild(img);

        fileItem.appendChild(fileNameSpan);

        fileItem.appendChild(removeBtn);

        fileListDiv.appendChild(fileItem);

      });

    }

    // Extract data from one file

    function extractDataFromFile(file) {

      return new Promise((resolve, reject) => {

        const reader = new FileReader();

        reader.onload = function(e) {

          try {

            const data = new Uint8Array(e.target.result);

            const workbook = XLSX.read(data, {type: "array", cellDates: true});

            const sheetName = workbook.SheetNames[0];

            const sheet = workbook.Sheets[sheetName];

            const jsonArray = XLSX.utils.sheet\_to\_json(sheet, {header:1, raw:false});

            const getCell = (r, c) => (jsonArray[r] && jsonArray[r][c]) || "";

            const buyerName = getCell(1,1);

            const challanNumber = getCell(1,6);

            const vehicle = getCell(2,3);

            const rawDate = getCell(2,5);

            const date = (rawDate instanceof Date) ? rawDate.toISOString().split("T")[0] : excelDateToJSDate(rawDate);

            function extractColRange(colIndex) {

              let result = [];

              for(let i=4; i<=21; i++) {

                const val = getCell(i,colIndex);

                if(!isTrulyEmpty(val) && val !== "0" && val !== 0) result.push(val);

              }

              return result.length > 0 ? result : undefined;

            }

            const products = extractColRange(1);

            const bags = extractColRange(3);

            const quantities = extractColRange(4);

            const rates = extractColRange(5);

            const amounts = extractColRange(6);

            const subtotalText = getCell(22,4);

            const subtotalValue = getCell(22,6);

            const totalBags = getCell(23,1);

            const discountPercent = getCell(23,5);

            const discountValue = getCell(23,6);

            const totalQty = getCell(24,1);

            const total = getCell(24,5);

            let extracted = {};

            if(!isTrulyEmpty(buyerName)) extracted.buyerName = buyerName;

            if(!isTrulyEmpty(challanNumber)) extracted.challanNumber = challanNumber;

            if(!isTrulyEmpty(vehicle) && vehicle !== "0") extracted.vehicle = vehicle;

            if(!isTrulyEmpty(date)) extracted.date = date;

            if(products) extracted.products = products;

            if(bags) extracted.bags = bags;

            if(quantities) extracted.quantities = quantities;

            if(rates) extracted.rates = rates;

            if(amounts) extracted.amounts = amounts;

            if(!isTrulyEmpty(subtotalText)) extracted.subtotalText = subtotalText;

            if(!isTrulyEmpty(subtotalValue) && subtotalValue !== "0") extracted.subtotalValue = subtotalValue;

            if(!isTrulyEmpty(totalBags)) extracted.totalBags = totalBags;

            if(!isTrulyEmpty(discountPercent) && discountPercent !== "0") extracted.discountPercent = discountPercent;

            if(!isTrulyEmpty(discountValue) && discountValue !== "0") extracted.discountValue = discountValue;

            if(!isTrulyEmpty(totalQty)) extracted.totalQuantity = totalQty;

            if(!isTrulyEmpty(total) && total !== "0") extracted.total = total;

            resolve({fileName: file.name, data: extracted});

          } catch(err) {

            reject(`Error processing file "${file.name}": ${err.message}`);

          }

        };

        reader.onerror = () => reject(`Failed to read file "${file.name}"`);

        reader.readAsArrayBuffer(file);

      });

    }

    // Render extracted results

    function renderExtractionResults() {

      resultsDiv.innerHTML = '';

      if (extractedDataMap.size === 0) {

        uploadAllBtn.disabled = true;

        return;

      }

      uploadAllBtn.disabled = false;

      extractedDataMap.forEach((data, filename) => {

        const block = document.createElement('div');

        block.className = 'result-block';

       const titleContainer = document.createElement('div');

titleContainer.style.display = 'flex';

titleContainer.style.alignItems = 'center';

titleContainer.style.gap = '10px';

titleContainer.style.marginBottom = '8px';

const logoImg = document.createElement('img');

logoImg.src = excelLogoUrl;

logoImg.width = logoWidth;

logoImg.height = logoHeight;

logoImg.alt = "Excel Logo";

const title = document.createElement('span');

title.textContent = filename;

title.style.fontSize = '17px';

title.style.fontWeight = '584';

title.style.maxWidth = '350px';

title.style.overflow = 'hidden';

title.style.textOverflow = 'ellipsis';

title.style.whiteSpace = 'nowrap';

titleContainer.appendChild(logoImg);

titleContainer.appendChild(title);

block.appendChild(titleContainer);

        const pre = document.createElement('pre');

        pre.textContent = JSON.stringify(data, null, 2);

        block.appendChild(pre);

        resultsDiv.appendChild(block);

      });

    }

    // Upload JSON file to GitHub repo (creates or updates)

 async function uploadFileToGitHub(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  // First check if file exists to determine if this is an update or new upload

  let sha = null;

  let fileExists = false;

  try {

    const checkResp = await fetch(

      `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`,

      {

        headers: {

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

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

        },

      }

    );

    if (checkResp.status === 200) {

      const fileData = await checkResp.json();

      sha = fileData.sha;

      fileExists = true;

    }

  } catch (e) {

    console.error("Error checking file existence:", e);

  }

  // Set clear commit message based on whether file exists

  const commitMessage = fileExists

    ? `UPDATED: ${filename}`

    : `UPLOADED: ${filename}`;

  const body = {

    message: commitMessage,  // This shows in GitHub's last commit

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))),

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;  // Required for updates

  const putResp = await fetch(

    `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`,

    {

      method: "PUT",

      headers: {

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

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

        "Content-Type": "application/json",

      },

      body: JSON.stringify(body),

    }

  );

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} - ${errJson.message}`);

  }

  return await putResp.json();

}

    // Event: file input change

    fileInput.addEventListener('change', function(e) {

      const newFiles = Array.from(e.target.files);

      newFiles.forEach(f => {

        if (!filesArray.some(existing => existing.name === f.name)) {

          filesArray.push(f);

        }

      });

      renderFileList();

      extractedDataMap.clear();

      resultsDiv.innerHTML = '';

      extractAllBtn.disabled = filesArray.length === 0;

      uploadAllBtn.disabled = true;

      // Clear input to allow re-upload same file if removed

      fileInput.value = "";

    });

    // Extract all files button

    extractAllBtn.addEventListener('click', async () => {

      resultsDiv.innerHTML = 'Extracting... Please wait.';

      extractAllBtn.disabled = true;

      uploadAllBtn.disabled = true;

      extractedDataMap.clear();

      try {

        for (const file of filesArray) {

          const result = await extractDataFromFile(file);

          extractedDataMap.set(result.fileName, result.data);

        }

        renderExtractionResults();

      } catch (err) {

        resultsDiv.innerHTML = `<span style="color:red;">Error: ${err}</span>`;

      } finally {

        extractAllBtn.disabled = false;

      }

    });

   // Upload all extracted data button

uploadAllBtn.addEventListener('click', async () => {

  if (extractedDataMap.size === 0) {

    alert("No extracted data to upload. Please extract first.");

    return;

  }

  uploadAllBtn.disabled = true;

  extractAllBtn.disabled = true;

  resultsDiv.innerHTML = '';

  const uploadStatusContainer = document.createElement('div');

  uploadStatusContainer.style.marginTop = "15px";

  uploadStatusContainer.style.display = "flex";

  uploadStatusContainer.style.flexDirection = "column";

  uploadStatusContainer.style.gap = "10px";

  resultsDiv.appendChild(uploadStatusContainer);

  const statusElements = new Map();

  for (const [filename] of extractedDataMap.entries()) {

    const statusBlock = document.createElement('div');

    statusBlock.className = 'file-item';

    const logo = document.createElement('img');

    logo.src = excelLogoUrl;

    logo.width = logoWidth;

    logo.height = logoHeight;

    logo.alt = "Excel Logo";

    const name = document.createElement('span');

    name.className = 'file-name';

    name.textContent = filename;

    const spinner = document.createElement('div');

    spinner.className = 'spinner';

    spinner.style.width = "18px";

    spinner.style.height = "18px";

    spinner.style.border = "3px solid #000";

    spinner.style.borderTop = "3px solid transparent";

    spinner.style.borderRadius = "50%";

    spinner.style.animation = "spin 0.8s linear infinite";

    statusBlock.appendChild(logo);

    statusBlock.appendChild(name);

    statusBlock.appendChild(spinner);

    uploadStatusContainer.appendChild(statusBlock);

    statusElements.set(filename, { block: statusBlock, spinner });

  }

  let uploadErrors = [];

  let uploadedCount = 0;

  let updatedCount = 0;

  // Modified uploadFileToGitHub to return info about commit message & whether it was update or new file

  async function uploadFileToGitHubWithInfo(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  let sha;

  let isUpdate = false;

  try {

    const resp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`, {

      headers: {

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

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

      },

    });

    if (resp.status === 200) {

      const json = await resp.json();

      sha = json.sha;

      isUpdate = true;

    }

  } catch (e) {

    // ignore file not found etc.

  }

  const commitMessage = isUpdate

    ? `UPDATED: extracted data file - ${filename}`

    : `UPLOADED: extracted data file - ${filename}`;

  const body = {

    message: commitMessage,

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))), // base64 encode

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;

  const putResp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`, {

    method: "PUT",

    headers: {

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

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

      "Content-Type": "application/json",

    },

    body: JSON.stringify(body),

  });

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} ${putResp.statusText} - ${errJson.message}`);

  }

  const jsonResp = await putResp.json();

  return { commitMessage, isUpdate, jsonResp };

}

  for (const [filename, data] of extractedDataMap.entries()) {

    try {

      const { commitMessage, isUpdate } = await uploadFileToGitHubWithInfo(filename, data);

     const status = statusElements.get(filename);

status.block.removeChild(status.spinner);

// Add colored tick only

const tick = document.createElement('span');

tick.textContent = "✅";

tick.style.fontWeight = "bold";

tick.style.fontSize = "17px";

tick.style.marginLeft = "8px";

tick.style.userSelect = "none";

tick.style.color = isUpdate ? "black" : "green";

status.block.appendChild(tick);

      if (isUpdate) updatedCount++;

      else uploadedCount++;

    } catch (e) {

      const status = statusElements.get(filename);

      status.block.removeChild(status.spinner);

      const cross = document.createElement('span');

      cross.textContent = "❌";

      cross.style.fontSize = "18px";

      status.block.appendChild(cross);

      uploadErrors.push(`Failed to upload ${filename}: ${e.message}`);

    }

  }

  if (uploadErrors.length > 0) {

    const errMsg = document.createElement('div');

    errMsg.style.color = "red";

    errMsg.style.marginTop = "10px";

    errMsg.style.whiteSpace = "pre-line";

    errMsg.textContent = "Some errors occurred:\n" + uploadErrors.join("\n");

    resultsDiv.appendChild(errMsg);

  }

  // Bottom summary message

const summaryMsg = document.createElement('div');

summaryMsg.style.marginTop = "10px";

summaryMsg.style.fontWeight = "bold";

if (uploadErrors.length === 0) {

  if (uploadedCount > 0 && updatedCount === 0) {

    summaryMsg.textContent = `✅ All uploads processed. (${uploadedCount} file${uploadedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "green";

    summaryMsg.style.fontWeight = "bold";

  } else if (updatedCount > 0 && uploadedCount === 0) {

    summaryMsg.textContent = `✅ All updates processed. (${updatedCount} file${updatedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "black";

    summaryMsg.style.fontWeight = "bold";

  } else {

    // New styled message with different colors

    summaryMsg.innerHTML = `

      <span style="color: #8B4513;">✅ All uploads and updates processed.</span>

      <span style="color: green; font-weight: bold;"> (${uploadedCount} upload${uploadedCount !== 1 ? "s" : ""})</span>

      <span style="color: black; font-weight: bold;">, (${updatedCount} update${updatedCount !== 1 ? "s" : ""})</span>

    `;

  }

} else {

  summaryMsg.textContent = `Process completed with errors.`;

  summaryMsg.style.color = "red";

  }

  resultsDiv.appendChild(summaryMsg);

  uploadAllBtn.disabled = false;

  extractAllBtn.disabled = false;

});

    // Initial render

    renderFileList();

  </script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Excel Multi-Extract & GitHub Upload</title>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    #fileList {

      display: flex;

      gap: 15px;

      margin-top: 15px;

      overflow-x: auto;

      padding-bottom: 10px;

      border-bottom: 2px solid #ddd;

    }

    .file-item {

      display: flex;

      align-items: center;

      gap: 8px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 6px 12px;

      background: #f9f9f9;

      white-space: nowrap;

      position: relative;

    }

    .file-item img {

      cursor: default;

      user-select: none;

    }

    .file-name {

      max-width: 305px;

      overflow: hidden;

      text-overflow: ellipsis;

      font-size: 17px;

      white-space: nowrap;

      font-weight: 620;

    }

      .remove-btn {

      cursor: pointer;

      color: white;

      background: red;

      border: none;

      border-radius: 50%;

      width: 20px;

      height: 20px;

      font-weight: bold;

      line-height: 18px;

      text-align: center;

      user-select: none;

    }

    #extractAllBtn, #uploadAllBtn {

      margin-top: 15px;

      padding: 10px 25px;

      font-size: 1rem;

      cursor: pointer;

    }

    pre {

      background: #efefef;

      padding: 10px;

      max-height: 300px;

      overflow-y: auto;

      margin-top: 20px;

      border-radius: 6px;

      white-space: pre-wrap;

      word-break: break-word;

    }

    .result-block {

      margin-top: 15px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 10px;

      background: #fafafa;

    }

    .status {

      margin-top: 6px;

      font-size: 0.9rem;

      font-weight: 600;

    }

  </style>

</head>

<body>

  <h2>Upload Multiple Excel Files</h2>

  <input type="file" id="fileInput" multiple accept=".xls,.xlsx" />

  <div id="fileList"></div>

  <button id="extractAllBtn" disabled>Extract All Files</button>

  <button id="uploadAllBtn" disabled>Upload All Extracted Data to GitHub</button>

  <h3>Extraction Results:</h3>

  <div id="results"></div>

  <script>

    // 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";

    // Excel logo url & logo size

    const excelLogoUrl = "https://cdn-icons-png.flaticon.com/512/732/732220.png";

    const logoWidth = 24;

    const logoHeight = 24;

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

    const fileListDiv = document.getElementById('fileList');

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

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

    const resultsDiv = document.getElementById('results');

    // Array to keep track of files

    let filesArray = [];

    // Store extracted data per file name

    let extractedDataMap = new Map();

    // Utility: Convert Excel serial date to ISO yyyy-mm-dd

    function excelDateToJSDate(serial) {

      if (typeof serial === "number") {

        const utc\_days = Math.floor(serial - 25569);

        const utc\_value = utc\_days \* 86400;

        const date\_info = new Date(utc\_value \* 1000);

        return date\_info.toISOString().split("T")[0];

      }

      return serial;

    }

    // Utility: check truly empty values

    function isTrulyEmpty(value) {

      return (

        value === undefined ||

        value === null ||

        value === "" ||

        (typeof value === "string" && value.trim() === "")

      );

    }

    // Render file list with logo, filename, remove btn

    function renderFileList() {

      fileListDiv.innerHTML = '';

      if (filesArray.length === 0) {

        extractAllBtn.disabled = true;

        uploadAllBtn.disabled = true;

        resultsDiv.innerHTML = '';

        extractedDataMap.clear();

        return;

      }

      extractAllBtn.disabled = false;

      uploadAllBtn.disabled = true; // only enable after extraction

      filesArray.forEach((file, index) => {

        const fileItem = document.createElement('div');

        fileItem.className = 'file-item';

        const img = document.createElement('img');

        img.src = excelLogoUrl;

        img.width = logoWidth;

        img.height = logoHeight;

        img.alt = "Excel Logo";

        const fileNameSpan = document.createElement('span');

        fileNameSpan.className = 'file-name';

        fileNameSpan.title = file.name; // show full on hover

        fileNameSpan.textContent = file.name;

        const removeBtn = document.createElement('button');

        removeBtn.className = 'remove-btn';

        removeBtn.title = "Remove this file";

        removeBtn.textContent = "×";

        removeBtn.onclick = () => {

          filesArray.splice(index, 1);

          extractedDataMap.delete(file.name);

          renderFileList();

          renderExtractionResults();

        };

        fileItem.appendChild(img);

        fileItem.appendChild(fileNameSpan);

        fileItem.appendChild(removeBtn);

        fileListDiv.appendChild(fileItem);

      });

    }

    // Extract data from one file

    function extractDataFromFile(file) {

      return new Promise((resolve, reject) => {

        const reader = new FileReader();

        reader.onload = function(e) {

          try {

            const data = new Uint8Array(e.target.result);

            const workbook = XLSX.read(data, {type: "array", cellDates: true});

            const sheetName = workbook.SheetNames[0];

            const sheet = workbook.Sheets[sheetName];

            const jsonArray = XLSX.utils.sheet\_to\_json(sheet, {header:1, raw:false});

            const getCell = (r, c) => (jsonArray[r] && jsonArray[r][c]) || "";

            const buyerName = getCell(1,1);

            const challanNumber = getCell(1,6);

            const vehicle = getCell(2,3);

            const rawDate = getCell(2,5);

            const date = (rawDate instanceof Date) ? rawDate.toISOString().split("T")[0] : excelDateToJSDate(rawDate);

            function extractColRange(colIndex) {

              let result = [];

              for(let i=4; i<=21; i++) {

                const val = getCell(i,colIndex);

                if(!isTrulyEmpty(val) && val !== "0" && val !== 0) result.push(val);

              }

              return result.length > 0 ? result : undefined;

            }

            const products = extractColRange(1);

            const bags = extractColRange(3);

            const quantities = extractColRange(4);

            const rates = extractColRange(5);

            const amounts = extractColRange(6);

            const subtotalText = getCell(22,4);

            const subtotalValue = getCell(22,6);

            const totalBags = getCell(23,1);

            const discountPercent = getCell(23,5);

            const discountValue = getCell(23,6);

            const totalQty = getCell(24,1);

            const total = getCell(24,5);

            let extracted = {};

            if(!isTrulyEmpty(buyerName)) extracted.buyerName = buyerName;

            if(!isTrulyEmpty(challanNumber)) extracted.challanNumber = challanNumber;

            if(!isTrulyEmpty(vehicle) && vehicle !== "0") extracted.vehicle = vehicle;

            if(!isTrulyEmpty(date)) extracted.date = date;

            if(products) extracted.products = products;

            if(bags) extracted.bags = bags;

            if(quantities) extracted.quantities = quantities;

            if(rates) extracted.rates = rates;

            if(amounts) extracted.amounts = amounts;

            if(!isTrulyEmpty(subtotalText)) extracted.subtotalText = subtotalText;

            if(!isTrulyEmpty(subtotalValue) && subtotalValue !== "0") extracted.subtotalValue = subtotalValue;

            if(!isTrulyEmpty(totalBags)) extracted.totalBags = totalBags;

            if(!isTrulyEmpty(discountPercent) && discountPercent !== "0") extracted.discountPercent = discountPercent;

            if(!isTrulyEmpty(discountValue) && discountValue !== "0") extracted.discountValue = discountValue;

            if(!isTrulyEmpty(totalQty)) extracted.totalQuantity = totalQty;

            if(!isTrulyEmpty(total) && total !== "0") extracted.total = total;

            resolve({fileName: file.name, data: extracted});

          } catch(err) {

            reject(`Error processing file "${file.name}": ${err.message}`);

          }

        };

        reader.onerror = () => reject(`Failed to read file "${file.name}"`);

        reader.readAsArrayBuffer(file);

      });

    }

    // Render extracted results

    function renderExtractionResults() {

      resultsDiv.innerHTML = '';

      if (extractedDataMap.size === 0) {

        uploadAllBtn.disabled = true;

        return;

      }

      uploadAllBtn.disabled = false;

      extractedDataMap.forEach((data, filename) => {

        const block = document.createElement('div');

        block.className = 'result-block';

       const titleContainer = document.createElement('div');

titleContainer.style.display = 'flex';

titleContainer.style.alignItems = 'center';

titleContainer.style.gap = '10px';

titleContainer.style.marginBottom = '8px';

const logoImg = document.createElement('img');

logoImg.src = excelLogoUrl;

logoImg.width = logoWidth;

logoImg.height = logoHeight;

logoImg.alt = "Excel Logo";

const title = document.createElement('span');

title.textContent = filename;

title.style.fontSize = '17px';

title.style.fontWeight = '584';

title.style.maxWidth = '350px';

title.style.overflow = 'hidden';

title.style.textOverflow = 'ellipsis';

title.style.whiteSpace = 'nowrap';

titleContainer.appendChild(logoImg);

titleContainer.appendChild(title);

block.appendChild(titleContainer);

        const pre = document.createElement('pre');

        pre.textContent = JSON.stringify(data, null, 2);

        block.appendChild(pre);

        resultsDiv.appendChild(block);

      });

    }

    // Upload JSON file to GitHub repo (creates or updates)

 async function uploadFileToGitHub(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  // First check if file exists to determine if this is an update or new upload

  let sha = null;

  let fileExists = false;

  try {

    const checkResp = await fetch(

      `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`,

      {

        headers: {

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

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

        },

      }

    );

    if (checkResp.status === 200) {

      const fileData = await checkResp.json();

      sha = fileData.sha;

      fileExists = true;

    }

  } catch (e) {

    console.error("Error checking file existence:", e);

  }

  // Set clear commit message based on whether file exists

  const commitMessage = fileExists

    ? `UPDATED: ${filename}`

    : `UPLOADED: ${filename}`;

  const body = {

    message: commitMessage,  // This shows in GitHub's last commit

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))),

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;  // Required for updates

  const putResp = await fetch(

    `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`,

    {

      method: "PUT",

      headers: {

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

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

        "Content-Type": "application/json",

      },

      body: JSON.stringify(body),

    }

  );

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} - ${errJson.message}`);

  }

  return await putResp.json();

}

    // Event: file input change

    fileInput.addEventListener('change', function(e) {

      const newFiles = Array.from(e.target.files);

      newFiles.forEach(f => {

        if (!filesArray.some(existing => existing.name === f.name)) {

          filesArray.push(f);

        }

      });

      renderFileList();

      extractedDataMap.clear();

      resultsDiv.innerHTML = '';

      extractAllBtn.disabled = filesArray.length === 0;

      uploadAllBtn.disabled = true;

      // Clear input to allow re-upload same file if removed

      fileInput.value = "";

    });

    // Extract all files button

    extractAllBtn.addEventListener('click', async () => {

      resultsDiv.innerHTML = 'Extracting... Please wait.';

      extractAllBtn.disabled = true;

      uploadAllBtn.disabled = true;

      extractedDataMap.clear();

      try {

        for (const file of filesArray) {

          const result = await extractDataFromFile(file);

          extractedDataMap.set(result.fileName, result.data);

        }

        renderExtractionResults();

      } catch (err) {

        resultsDiv.innerHTML = `<span style="color:red;">Error: ${err}</span>`;

      } finally {

        extractAllBtn.disabled = false;

      }

    });

   // Upload all extracted data button

uploadAllBtn.addEventListener('click', async () => {

  if (extractedDataMap.size === 0) {

    alert("No extracted data to upload. Please extract first.");

    return;

  }

  uploadAllBtn.disabled = true;

  extractAllBtn.disabled = true;

  resultsDiv.innerHTML = '';

  const uploadStatusContainer = document.createElement('div');

  uploadStatusContainer.style.marginTop = "15px";

  uploadStatusContainer.style.display = "flex";

  uploadStatusContainer.style.flexDirection = "column";

  uploadStatusContainer.style.gap = "10px";

  resultsDiv.appendChild(uploadStatusContainer);

  const statusElements = new Map();

  for (const [filename] of extractedDataMap.entries()) {

    const statusBlock = document.createElement('div');

    statusBlock.className = 'file-item';

    const logo = document.createElement('img');

    logo.src = excelLogoUrl;

    logo.width = logoWidth;

    logo.height = logoHeight;

    logo.alt = "Excel Logo";

    const name = document.createElement('span');

    name.className = 'file-name';

    name.textContent = filename;

    const spinner = document.createElement('div');

    spinner.className = 'spinner';

    spinner.style.width = "18px";

    spinner.style.height = "18px";

    spinner.style.border = "3px solid #000";

    spinner.style.borderTop = "3px solid transparent";

    spinner.style.borderRadius = "50%";

    spinner.style.animation = "spin 0.8s linear infinite";

    statusBlock.appendChild(logo);

    statusBlock.appendChild(name);

    statusBlock.appendChild(spinner);

    uploadStatusContainer.appendChild(statusBlock);

    statusElements.set(filename, { block: statusBlock, spinner });

  }

  let uploadErrors = [];

  let uploadedCount = 0;

  let updatedCount = 0;

  // Modified uploadFileToGitHub to return info about commit message & whether it was update or new file

  async function uploadFileToGitHubWithInfo(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  let sha;

  let isUpdate = false;

  try {

    const resp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`, {

      headers: {

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

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

      },

    });

    if (resp.status === 200) {

      const json = await resp.json();

      sha = json.sha;

      isUpdate = true;

    }

  } catch (e) {

    // ignore file not found etc.

  }

  const commitMessage = isUpdate

    ? `UPDATED: extracted data file - ${filename}`

    : `UPLOADED: extracted data file - ${filename}`;

  const body = {

    message: commitMessage,

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))), // base64 encode

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;

  const putResp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`, {

    method: "PUT",

    headers: {

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

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

      "Content-Type": "application/json",

    },

    body: JSON.stringify(body),

  });

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} ${putResp.statusText} - ${errJson.message}`);

  }

  const jsonResp = await putResp.json();

  return { commitMessage, isUpdate, jsonResp };

}

  for (const [filename, data] of extractedDataMap.entries()) {

    try {

      const { commitMessage, isUpdate } = await uploadFileToGitHubWithInfo(filename, data);

     const status = statusElements.get(filename);

status.block.removeChild(status.spinner);

// Add colored tick only

const tick = document.createElement('span');

tick.textContent = "✅";

tick.style.fontWeight = "bold";

tick.style.fontSize = "17px";

tick.style.marginLeft = "8px";

tick.style.userSelect = "none";

tick.style.color = isUpdate ? "black" : "green";

status.block.appendChild(tick);

      if (isUpdate) updatedCount++;

      else uploadedCount++;

    } catch (e) {

      const status = statusElements.get(filename);

      status.block.removeChild(status.spinner);

      const cross = document.createElement('span');

      cross.textContent = "❌";

      cross.style.fontSize = "18px";

      status.block.appendChild(cross);

      uploadErrors.push(`Failed to upload ${filename}: ${e.message}`);

    }

  }

  if (uploadErrors.length > 0) {

    const errMsg = document.createElement('div');

    errMsg.style.color = "red";

    errMsg.style.marginTop = "10px";

    errMsg.style.whiteSpace = "pre-line";

    errMsg.textContent = "Some errors occurred:\n" + uploadErrors.join("\n");

    resultsDiv.appendChild(errMsg);

  }

  // Bottom summary message

const summaryMsg = document.createElement('div');

summaryMsg.style.marginTop = "10px";

summaryMsg.style.fontWeight = "bold";

if (uploadErrors.length === 0) {

  if (uploadedCount > 0 && updatedCount === 0) {

    summaryMsg.textContent = `✅ All uploads processed. (${uploadedCount} file${uploadedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "green";

    summaryMsg.style.fontWeight = "bold";

  } else if (updatedCount > 0 && uploadedCount === 0) {

    summaryMsg.textContent = `✅ All updates processed. (${updatedCount} file${updatedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "black";

    summaryMsg.style.fontWeight = "bold";

  } else {

    // New styled message with different colors

    summaryMsg.innerHTML = `

      <span style="color: #8B4513;">✅ All uploads and updates processed.</span>

      <span style="color: green; font-weight: bold;"> (${uploadedCount} upload${uploadedCount !== 1 ? "s" : ""})</span>

      <span style="color: black; font-weight: bold;">, (${updatedCount} update${updatedCount !== 1 ? "s" : ""})</span>

    `;

  }

} else {

  summaryMsg.textContent = `Process completed with errors.`;

  summaryMsg.style.color = "red";

  }

  resultsDiv.appendChild(summaryMsg);

// Auto-reload logic

  const countdown = document.createElement('div');

  countdown.style.marginTop = '10px';

  countdown.style.color = 'black';

  resultsDiv.appendChild(countdown);

  let secondsLeft = 0; // I WILL BE ADD THIS LATER (\*DONT TAKE THIS SERIOS\*)

  const countdownInterval = setInterval(() => {

    secondsLeft--;

    countdown.textContent = `Page Is Reloading In  ${secondsLeft} second${secondsLeft !== 1 ? 's' : ''}...`;

    if (secondsLeft <= 0) {

      clearInterval(countdownInterval);

      location.reload();

    }

  }, 1008); // I WILL BE Edit THIS LATER (\*DONT TAKE THIS SERIOS\*)

  uploadAllBtn.disabled = false;

  extractAllBtn.disabled = false;

});

    // Initial render

    renderFileList();

  </script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Excel Multi-Extract & GitHub Upload</title>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    #fileList {

      display: flex;

      gap: 15px;

      margin-top: 15px;

      overflow-x: auto;

      padding-bottom: 10px;

      border-bottom: 2px solid #ddd;

    }

    .file-item {

      display: flex;

      align-items: center;

      gap: 8px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 6px 12px;

      background: #f9f9f9;

      white-space: nowrap;

      position: relative;

    }

    .file-item img {

      cursor: default;

      user-select: none;

    }

    .file-name {

      max-width: 305px;

      overflow: hidden;

      text-overflow: ellipsis;

      font-size: 17px;

      white-space: nowrap;

      font-weight: 620;

    }

      .remove-btn {

      cursor: pointer;

      color: white;

      background: red;

      border: none;

      border-radius: 50%;

      width: 20px;

      height: 20px;

      font-weight: bold;

      line-height: 18px;

      text-align: center;

      user-select: none;

    }

    #extractAllBtn, #uploadAllBtn {

      margin-top: 15px;

      padding: 10px 25px;

      font-size: 1rem;

      cursor: pointer;

    }

    pre {

      background: #efefef;

      padding: 10px;

      max-height: 300px;

      overflow-y: auto;

      margin-top: 20px;

      border-radius: 6px;

      white-space: pre-wrap;

      word-break: break-word;

    }

    .result-block {

      margin-top: 15px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 10px;

      background: #fafafa;

    }

    .status {

      margin-top: 6px;

      font-size: 0.9rem;

      font-weight: 600;

    }

  </style>

</head>

<body>

  <h2>Upload Multiple Excel Files</h2>

  <input type="file" id="fileInput" multiple accept=".xls,.xlsx" />

  <div id="fileList"></div>

  <button id="extractAllBtn" disabled>Extract All Files</button>

  <button id="uploadAllBtn" disabled>Upload All Extracted Data to GitHub</button>

  <h3>Extraction Results:</h3>

  <div id="results"></div>

  <script>

    // 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";

    // Excel logo url & logo size

    const excelLogoUrl = "https://cdn-icons-png.flaticon.com/512/732/732220.png";

    const logoWidth = 24;

    const logoHeight = 24;

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

    const fileListDiv = document.getElementById('fileList');

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

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

    const resultsDiv = document.getElementById('results');

    // Array to keep track of files

    let filesArray = [];

    // Store extracted data per file name

    let extractedDataMap = new Map();

    // Utility: Convert Excel serial date to ISO yyyy-mm-dd

    function excelDateToJSDate(serial) {

      if (typeof serial === "number") {

        const utc\_days = Math.floor(serial - 25569);

        const utc\_value = utc\_days \* 86400;

        const date\_info = new Date(utc\_value \* 1000);

        return date\_info.toISOString().split("T")[0];

      }

      return serial;

    }

    // Utility: check truly empty values

    function isTrulyEmpty(value) {

      return (

        value === undefined ||

        value === null ||

        value === "" ||

        (typeof value === "string" && value.trim() === "")

      );

    }

    // Render file list with logo, filename, remove btn

    function renderFileList() {

      fileListDiv.innerHTML = '';

      if (filesArray.length === 0) {

        extractAllBtn.disabled = true;

        uploadAllBtn.disabled = true;

        resultsDiv.innerHTML = '';

        extractedDataMap.clear();

        return;

      }

      extractAllBtn.disabled = false;

      uploadAllBtn.disabled = true; // only enable after extraction

      filesArray.forEach((file, index) => {

        const fileItem = document.createElement('div');

        fileItem.className = 'file-item';

        const img = document.createElement('img');

        img.src = excelLogoUrl;

        img.width = logoWidth;

        img.height = logoHeight;

        img.alt = "Excel Logo";

        const fileNameSpan = document.createElement('span');

        fileNameSpan.className = 'file-name';

        fileNameSpan.title = file.name; // show full on hover

        fileNameSpan.textContent = file.name;

        const removeBtn = document.createElement('button');

        removeBtn.className = 'remove-btn';

        removeBtn.title = "Remove this file";

        removeBtn.textContent = "×";

        removeBtn.onclick = () => {

          filesArray.splice(index, 1);

          extractedDataMap.delete(file.name);

          renderFileList();

          renderExtractionResults();

        };

        fileItem.appendChild(img);

        fileItem.appendChild(fileNameSpan);

        fileItem.appendChild(removeBtn);

        fileListDiv.appendChild(fileItem);

      });

    }

    // Extract data from one file

    function extractDataFromFile(file) {

      return new Promise((resolve, reject) => {

        const reader = new FileReader();

        reader.onload = function(e) {

          try {

            const data = new Uint8Array(e.target.result);

            const workbook = XLSX.read(data, {type: "array", cellDates: true});

            const sheetName = workbook.SheetNames[0];

            const sheet = workbook.Sheets[sheetName];

            const jsonArray = XLSX.utils.sheet\_to\_json(sheet, {header:1, raw:false});

            const getCell = (r, c) => (jsonArray[r] && jsonArray[r][c]) || "";

            const buyerName = getCell(1,1);

            const challanNumber = getCell(1,6);

            const vehicle = getCell(2,3);

            const rawDate = getCell(2,5);

            const date = (rawDate instanceof Date) ? rawDate.toISOString().split("T")[0] : excelDateToJSDate(rawDate);

            function extractColRange(colIndex) {

              let result = [];

              for(let i=4; i<=21; i++) {

                const val = getCell(i,colIndex);

                if(!isTrulyEmpty(val) && val !== "0" && val !== 0) result.push(val);

              }

              return result.length > 0 ? result : undefined;

            }

            const products = extractColRange(1);

            const bags = extractColRange(3);

            const quantities = extractColRange(4);

            const rates = extractColRange(5);

            const amounts = extractColRange(6);

            const subtotalText = getCell(22,4);

            const subtotalValue = getCell(22,6);

            const totalBags = getCell(23,1);

            const discountPercent = getCell(23,5);

            const discountValue = getCell(23,6);

            const totalQty = getCell(24,1);

            const total = getCell(24,5);

            let extracted = {};

            if(!isTrulyEmpty(buyerName)) extracted.buyerName = buyerName;

            if(!isTrulyEmpty(challanNumber)) extracted.challanNumber = challanNumber;

            if(!isTrulyEmpty(vehicle) && vehicle !== "0") extracted.vehicle = vehicle;

            if(!isTrulyEmpty(date)) extracted.date = date;

            if(products) extracted.products = products;

            if(bags) extracted.bags = bags;

            if(quantities) extracted.quantities = quantities;

            if(rates) extracted.rates = rates;

            if(amounts) extracted.amounts = amounts;

            if(!isTrulyEmpty(subtotalText)) extracted.subtotalText = subtotalText;

            if(!isTrulyEmpty(subtotalValue) && subtotalValue !== "0") extracted.subtotalValue = subtotalValue;

            if(!isTrulyEmpty(totalBags)) extracted.totalBags = totalBags;

            if(!isTrulyEmpty(discountPercent) && discountPercent !== "0") extracted.discountPercent = discountPercent;

            if(!isTrulyEmpty(discountValue) && discountValue !== "0") extracted.discountValue = discountValue;

            if(!isTrulyEmpty(totalQty)) extracted.totalQuantity = totalQty;

            if(!isTrulyEmpty(total) && total !== "0") extracted.total = total;

            resolve({fileName: file.name, data: extracted});

          } catch(err) {

            reject(`Error processing file "${file.name}": ${err.message}`);

          }

        };

        reader.onerror = () => reject(`Failed to read file "${file.name}"`);

        reader.readAsArrayBuffer(file);

      });

    }

    // Render extracted results

    function renderExtractionResults() {

      resultsDiv.innerHTML = '';

      if (extractedDataMap.size === 0) {

        uploadAllBtn.disabled = true;

        return;

      }

      uploadAllBtn.disabled = false;

      extractedDataMap.forEach((data, filename) => {

        const block = document.createElement('div');

        block.className = 'result-block';

       const titleContainer = document.createElement('div');

titleContainer.style.display = 'flex';

titleContainer.style.alignItems = 'center';

titleContainer.style.gap = '10px';

titleContainer.style.marginBottom = '8px';

const logoImg = document.createElement('img');

logoImg.src = excelLogoUrl;

logoImg.width = logoWidth;

logoImg.height = logoHeight;

logoImg.alt = "Excel Logo";

const title = document.createElement('span');

title.textContent = filename;

title.style.fontSize = '17px';

title.style.fontWeight = '584';

title.style.maxWidth = '350px';

title.style.overflow = 'hidden';

title.style.textOverflow = 'ellipsis';

title.style.whiteSpace = 'nowrap';

titleContainer.appendChild(logoImg);

titleContainer.appendChild(title);

block.appendChild(titleContainer);

        const pre = document.createElement('pre');

        pre.textContent = JSON.stringify(data, null, 2);

        block.appendChild(pre);

        resultsDiv.appendChild(block);

      });

    }

    // Upload JSON file to GitHub repo (creates or updates)

 async function uploadFileToGitHub(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  // First check if file exists to determine if this is an update or new upload

  let sha = null;

  let fileExists = false;

  try {

    const checkResp = await fetch(

      `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`,

      {

        headers: {

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

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

        },

      }

    );

    if (checkResp.status === 200) {

      const fileData = await checkResp.json();

      sha = fileData.sha;

      fileExists = true;

    }

  } catch (e) {

    console.error("Error checking file existence:", e);

  }

  // Set clear commit message based on whether file exists

  const commitMessage = fileExists

    ? `UPDATED: ${filename}`

    : `UPLOADED: ${filename}`;

  const body = {

    message: commitMessage,  // This shows in GitHub's last commit

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))),

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;  // Required for updates

  const putResp = await fetch(

    `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`,

    {

      method: "PUT",

      headers: {

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

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

        "Content-Type": "application/json",

      },

      body: JSON.stringify(body),

    }

  );

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} - ${errJson.message}`);

  }

  return await putResp.json();

}

    // Event: file input change

    fileInput.addEventListener('change', function(e) {

      const newFiles = Array.from(e.target.files);

      newFiles.forEach(f => {

        if (!filesArray.some(existing => existing.name === f.name)) {

          filesArray.push(f);

        }

      });

      renderFileList();

      extractedDataMap.clear();

      resultsDiv.innerHTML = '';

      extractAllBtn.disabled = filesArray.length === 0;

      uploadAllBtn.disabled = true;

      // Clear input to allow re-upload same file if removed

      fileInput.value = "";

    });

    // Extract all files button

    extractAllBtn.addEventListener('click', async () => {

      resultsDiv.innerHTML = 'Extracting... Please wait.';

      extractAllBtn.disabled = true;

      uploadAllBtn.disabled = true;

      extractedDataMap.clear();

      try {

        for (const file of filesArray) {

          const result = await extractDataFromFile(file);

          extractedDataMap.set(result.fileName, result.data);

        }

        renderExtractionResults();

      } catch (err) {

        resultsDiv.innerHTML = `<span style="color:red;">Error: ${err}</span>`;

      } finally {

        extractAllBtn.disabled = false;

      }

    });

   // Upload all extracted data button

uploadAllBtn.addEventListener('click', async () => {

  if (extractedDataMap.size === 0) {

    alert("No extracted data to upload. Please extract first.");

    return;

  }

  uploadAllBtn.disabled = true;

  extractAllBtn.disabled = true;

  resultsDiv.innerHTML = '';

  const uploadStatusContainer = document.createElement('div');

  uploadStatusContainer.style.marginTop = "15px";

  uploadStatusContainer.style.display = "flex";

  uploadStatusContainer.style.flexDirection = "column";

  uploadStatusContainer.style.gap = "10px";

  resultsDiv.appendChild(uploadStatusContainer);

  const statusElements = new Map();

  for (const [filename] of extractedDataMap.entries()) {

    const statusBlock = document.createElement('div');

    statusBlock.className = 'file-item';

    const logo = document.createElement('img');

    logo.src = excelLogoUrl;

    logo.width = logoWidth;

    logo.height = logoHeight;

    logo.alt = "Excel Logo";

    const name = document.createElement('span');

    name.className = 'file-name';

    name.textContent = filename;

    const spinner = document.createElement('div');

    spinner.className = 'spinner';

    spinner.style.width = "18px";

    spinner.style.height = "18px";

    spinner.style.border = "3px solid #000";

    spinner.style.borderTop = "3px solid transparent";

    spinner.style.borderRadius = "50%";

    spinner.style.animation = "spin 0.8s linear infinite";

    statusBlock.appendChild(logo);

    statusBlock.appendChild(name);

    statusBlock.appendChild(spinner);

    uploadStatusContainer.appendChild(statusBlock);

    statusElements.set(filename, { block: statusBlock, spinner });

  }

  let uploadErrors = [];

  let uploadedCount = 0;

  let updatedCount = 0;

  // Modified uploadFileToGitHub to return info about commit message & whether it was update or new file

  async function uploadFileToGitHubWithInfo(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  let sha;

  let isUpdate = false;

  try {

    const resp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`, {

      headers: {

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

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

      },

    });

    if (resp.status === 200) {

      const json = await resp.json();

      sha = json.sha;

      isUpdate = true;

    }

  } catch (e) {

    // ignore file not found etc.

  }

  const commitMessage = isUpdate

    ? `UPDATED: extracted data file - ${filename}`

    : `UPLOADED: extracted data file - ${filename}`;

  const body = {

    message: commitMessage,

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))), // base64 encode

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;

  const putResp = await fetch(`https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`, {

    method: "PUT",

    headers: {

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

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

      "Content-Type": "application/json",

    },

    body: JSON.stringify(body),

  });

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} ${putResp.statusText} - ${errJson.message}`);

  }

  const jsonResp = await putResp.json();

  return { commitMessage, isUpdate, jsonResp };

}

  for (const [filename, data] of extractedDataMap.entries()) {

    try {

      const { commitMessage, isUpdate } = await uploadFileToGitHubWithInfo(filename, data);

     const status = statusElements.get(filename);

status.block.removeChild(status.spinner);

// Add colored tick only

const tick = document.createElement('span');

tick.textContent = "✅";

tick.style.fontWeight = "bold";

tick.style.fontSize = "17px";

tick.style.marginLeft = "8px";

tick.style.userSelect = "none";

tick.style.color = isUpdate ? "black" : "green";

status.block.appendChild(tick);

      if (isUpdate) updatedCount++;

      else uploadedCount++;

    } catch (e) {

      const status = statusElements.get(filename);

      status.block.removeChild(status.spinner);

      const cross = document.createElement('span');

      cross.textContent = "❌";

      cross.style.fontSize = "18px";

      status.block.appendChild(cross);

      uploadErrors.push(`Failed to upload ${filename}: ${e.message}`);

    }

  }

  if (uploadErrors.length > 0) {

    const errMsg = document.createElement('div');

    errMsg.style.color = "red";

    errMsg.style.marginTop = "10px";

    errMsg.style.whiteSpace = "pre-line";

    errMsg.textContent = "Some errors occurred:\n" + uploadErrors.join("\n");

    resultsDiv.appendChild(errMsg);

  }

  // Bottom summary message

const summaryMsg = document.createElement('div');

summaryMsg.style.marginTop = "10px";

summaryMsg.style.fontWeight = "bold";

if (uploadErrors.length === 0) {

  if (uploadedCount > 0 && updatedCount === 0) {

    summaryMsg.textContent = `✅ All uploads processed. (${uploadedCount} file${uploadedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "green";

    summaryMsg.style.fontWeight = "bold";

  } else if (updatedCount > 0 && uploadedCount === 0) {

    summaryMsg.textContent = `✅ All updates processed. (${updatedCount} file${updatedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "black";

    summaryMsg.style.fontWeight = "bold";

  } else {

    // New styled message with different colors

    summaryMsg.innerHTML = `

      <span style="color: #8B4513;">✅ All uploads and updates processed.</span>

      <span style="color: green; font-weight: bold;"> (${uploadedCount} upload${uploadedCount !== 1 ? "s" : ""})</span>

      <span style="color: black; font-weight: bold;">, (${updatedCount} update${updatedCount !== 1 ? "s" : ""})</span>

    `;

  }

} else {

  summaryMsg.textContent = `Process completed with errors.`;

  summaryMsg.style.color = "red";

  }

  resultsDiv.appendChild(summaryMsg);

// Auto-reload logic

  const countdown = document.createElement('div');

  countdown.style.marginTop = '10px';

  countdown.style.color = 'black';

  resultsDiv.appendChild(countdown);

  let secondsLeft = 0; // I WILL BE ADD THIS LATER (\*DONT TAKE THIS SERIOS\*)

  const countdownInterval = setInterval(() => {

    secondsLeft--;

    countdown.textContent = `Page Is Reloading In  ${secondsLeft} second${secondsLeft !== 1 ? 's' : ''}...`;

    if (secondsLeft <= 0) {

      clearInterval(countdownInterval);

      location.reload();

    }

  }, 1008); // I WILL BE Edit THIS LATER (\*DONT TAKE THIS SERIOS\*)

  uploadAllBtn.disabled = false;

  extractAllBtn.disabled = false;

});

    // Initial render

    renderFileList();

  </script>

</body>

</html>

// 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";

**✅ WHAT YOU PROPOSE:**

You're asking:

Can we create a **separate folder for each month** (e.g., 2025-06, 2025-07, etc.) on GitHub, and store each file in the corresponding month’s folder based on the date field inside the JSON?

**✅ YES. AND HERE’S WHY IT’S USEFUL:**

* Right now, you're **loading all JSON files** from a single folder (extracted\_data) — which becomes slow when the file count grows.
* By organizing files **by month**, e.g.:

pgsql

CopyEdit

extracted\_data/

├── 2025-06/

│ ├── Buyer1\_08.json

│ ├── Buyer2\_11.json

├── 2025-07/

│ ├── Buyer3\_02.json

You can:

* 🔹 Fetch only the folder for the selected date range (e.g., only 2025-06).
* 🔹 Reduce API requests and file parsing time significantly.
* 🔹 Improve UI responsiveness for long-term usage.

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Excel Multi-Extract & GitHub Upload</title>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

    }

    #fileList {

      display: flex;

      gap: 15px;

      margin-top: 15px;

      overflow-x: auto;

      padding-bottom: 10px;

      border-bottom: 2px solid #ddd;

    }

    .file-item {

      display: flex;

      align-items: center;

      gap: 8px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 6px 12px;

      background: #f9f9f9;

      white-space: nowrap;

      position: relative;

    }

    .file-item img {

      cursor: default;

      user-select: none;

    }

    .file-name {

      max-width: 305px;

      overflow: hidden;

      text-overflow: ellipsis;

      font-size: 17px;

      white-space: nowrap;

      font-weight: 620;

    }

      .remove-btn {

      cursor: pointer;

      color: white;

      background: red;

      border: none;

      border-radius: 50%;

      width: 20px;

      height: 20px;

      font-weight: bold;

      line-height: 18px;

      text-align: center;

      user-select: none;

    }

    #extractAllBtn, #uploadAllBtn {

      margin-top: 15px;

      padding: 10px 25px;

      font-size: 1rem;

      cursor: pointer;

    }

    pre {

      background: #efefef;

      padding: 10px;

      max-height: 300px;

      overflow-y: auto;

      margin-top: 20px;

      border-radius: 6px;

      white-space: pre-wrap;

      word-break: break-word;

    }

    .result-block {

      margin-top: 15px;

      border: 1px solid #ccc;

      border-radius: 6px;

      padding: 10px;

      background: #fafafa;

    }

    .status {

      margin-top: 6px;

      font-size: 0.9rem;

      font-weight: 600;

    }

  </style>

</head>

<body>

  <h2>Upload Multiple Excel Files</h2>

  <input type="file" id="fileInput" multiple accept=".xls,.xlsx" />

  <div id="fileList"></div>

  <button id="extractAllBtn" disabled>Extract All Files</button>

  <button id="uploadAllBtn" disabled>Upload All Extracted Data to GitHub</button>

  <h3>Extraction Results:</h3>

  <div id="results"></div>

  <script>

    // 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";

    // Excel logo url & logo size

    const excelLogoUrl = "https://cdn-icons-png.flaticon.com/512/732/732220.png";

    const logoWidth = 24;

    const logoHeight = 24;

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

    const fileListDiv = document.getElementById('fileList');

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

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

    const resultsDiv = document.getElementById('results');

    // Array to keep track of files

    let filesArray = [];

    // Store extracted data per file name

    let extractedDataMap = new Map();

    // Utility: Convert Excel serial date to ISO yyyy-mm-dd

    function excelDateToJSDate(serial) {

      if (typeof serial === "number") {

        const utc\_days = Math.floor(serial - 25569);

        const utc\_value = utc\_days \* 86400;

        const date\_info = new Date(utc\_value \* 1000);

        return date\_info.toISOString().split("T")[0];

      }

      return serial;

    }

    // Utility: check truly empty values

    function isTrulyEmpty(value) {

      return (

        value === undefined ||

        value === null ||

        value === "" ||

        (typeof value === "string" && value.trim() === "")

      );

    }

    // Render file list with logo, filename, remove btn

    function renderFileList() {

      fileListDiv.innerHTML = '';

      if (filesArray.length === 0) {

        extractAllBtn.disabled = true;

        uploadAllBtn.disabled = true;

        resultsDiv.innerHTML = '';

        extractedDataMap.clear();

        return;

      }

      extractAllBtn.disabled = false;

      uploadAllBtn.disabled = true; // only enable after extraction

      filesArray.forEach((file, index) => {

        const fileItem = document.createElement('div');

        fileItem.className = 'file-item';

        const img = document.createElement('img');

        img.src = excelLogoUrl;

        img.width = logoWidth;

        img.height = logoHeight;

        img.alt = "Excel Logo";

        const fileNameSpan = document.createElement('span');

        fileNameSpan.className = 'file-name';

        fileNameSpan.title = file.name; // show full on hover

        fileNameSpan.textContent = file.name;

        const removeBtn = document.createElement('button');

        removeBtn.className = 'remove-btn';

        removeBtn.title = "Remove this file";

        removeBtn.textContent = "×";

        removeBtn.onclick = () => {

          filesArray.splice(index, 1);

          extractedDataMap.delete(file.name);

          renderFileList();

          renderExtractionResults();

        };

        fileItem.appendChild(img);

        fileItem.appendChild(fileNameSpan);

        fileItem.appendChild(removeBtn);

        fileListDiv.appendChild(fileItem);

      });

    }

    // Extract data from one file

    function extractDataFromFile(file) {

      return new Promise((resolve, reject) => {

        const reader = new FileReader();

        reader.onload = function(e) {

          try {

            const data = new Uint8Array(e.target.result);

            const workbook = XLSX.read(data, {type: "array", cellDates: true});

            const sheetName = workbook.SheetNames[0];

            const sheet = workbook.Sheets[sheetName];

            const jsonArray = XLSX.utils.sheet\_to\_json(sheet, {header:1, raw:false});

            const getCell = (r, c) => (jsonArray[r] && jsonArray[r][c]) || "";

            const buyerName = getCell(1,1);

            const challanNumber = getCell(1,6);

            const vehicle = getCell(2,3);

            const rawDate = getCell(2,5);

            const date = (rawDate instanceof Date) ? rawDate.toISOString().split("T")[0] : excelDateToJSDate(rawDate);

            function extractColRange(colIndex) {

              let result = [];

              for(let i=4; i<=21; i++) {

                const val = getCell(i,colIndex);

                if(!isTrulyEmpty(val) && val !== "0" && val !== 0) result.push(val);

              }

              return result.length > 0 ? result : undefined;

            }

            const products = extractColRange(1);

            const bags = extractColRange(3);

            const quantities = extractColRange(4);

            const rates = extractColRange(5);

            const amounts = extractColRange(6);

            const subtotalText = getCell(22,4);

            const subtotalValue = getCell(22,6);

            const totalBags = getCell(23,1);

            const discountPercent = getCell(23,5);

            const discountValue = getCell(23,6);

            const totalQty = getCell(24,1);

            const total = getCell(24,5);

            let extracted = {};

            if(!isTrulyEmpty(buyerName)) extracted.buyerName = buyerName;

            if(!isTrulyEmpty(challanNumber)) extracted.challanNumber = challanNumber;

            if(!isTrulyEmpty(vehicle) && vehicle !== "0") extracted.vehicle = vehicle;

            if(!isTrulyEmpty(date)) extracted.date = date;

            if(products) extracted.products = products;

            if(bags) extracted.bags = bags;

            if(quantities) extracted.quantities = quantities;

            if(rates) extracted.rates = rates;

            if(amounts) extracted.amounts = amounts;

            if(!isTrulyEmpty(subtotalText)) extracted.subtotalText = subtotalText;

            if(!isTrulyEmpty(subtotalValue) && subtotalValue !== "0") extracted.subtotalValue = subtotalValue;

            if(!isTrulyEmpty(totalBags)) extracted.totalBags = totalBags;

            if(!isTrulyEmpty(discountPercent) && discountPercent !== "0") extracted.discountPercent = discountPercent;

            if(!isTrulyEmpty(discountValue) && discountValue !== "0") extracted.discountValue = discountValue;

            if(!isTrulyEmpty(totalQty)) extracted.totalQuantity = totalQty;

            if(!isTrulyEmpty(total) && total !== "0") extracted.total = total;

            resolve({fileName: file.name, data: extracted});

          } catch(err) {

            reject(`Error processing file "${file.name}": ${err.message}`);

          }

        };

        reader.onerror = () => reject(`Failed to read file "${file.name}"`);

        reader.readAsArrayBuffer(file);

      });

    }

    // Render extracted results

    function renderExtractionResults() {

      resultsDiv.innerHTML = '';

      if (extractedDataMap.size === 0) {

        uploadAllBtn.disabled = true;

        return;

      }

      uploadAllBtn.disabled = false;

      extractedDataMap.forEach((data, filename) => {

        const block = document.createElement('div');

        block.className = 'result-block';

       const titleContainer = document.createElement('div');

titleContainer.style.display = 'flex';

titleContainer.style.alignItems = 'center';

titleContainer.style.gap = '10px';

titleContainer.style.marginBottom = '8px';

const logoImg = document.createElement('img');

logoImg.src = excelLogoUrl;

logoImg.width = logoWidth;

logoImg.height = logoHeight;

logoImg.alt = "Excel Logo";

const title = document.createElement('span');

title.textContent = filename;

title.style.fontSize = '17px';

title.style.fontWeight = '584';

title.style.maxWidth = '350px';

title.style.overflow = 'hidden';

title.style.textOverflow = 'ellipsis';

title.style.whiteSpace = 'nowrap';

titleContainer.appendChild(logoImg);

titleContainer.appendChild(title);

block.appendChild(titleContainer);

        const pre = document.createElement('pre');

        pre.textContent = JSON.stringify(data, null, 2);

        block.appendChild(pre);

        resultsDiv.appendChild(block);

      });

    }

    // Upload JSON file to GitHub repo (creates or updates)

 async function uploadFileToGitHub(filename, contentJson) {

  const path = `extracted\_data/${filename.replace(/\.[^/.]+$/, "")}.json`;

  // First check if file exists to determine if this is an update or new upload

  let sha = null;

  let fileExists = false;

  try {

    const checkResp = await fetch(

      `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`,

      {

        headers: {

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

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

        },

      }

    );

    if (checkResp.status === 200) {

      const fileData = await checkResp.json();

      sha = fileData.sha;

      fileExists = true;

    }

  } catch (e) {

    console.error("Error checking file existence:", e);

  }

  // Set clear commit message based on whether file exists

  const commitMessage = fileExists

    ? `UPDATED: ${filename}`

    : `UPLOADED: ${filename}`;

  const body = {

    message: commitMessage,  // This shows in GitHub's last commit

    content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))),

    branch: REPO\_BRANCH,

  };

  if (sha) body.sha = sha;  // Required for updates

  const putResp = await fetch(

    `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`,

    {

      method: "PUT",

      headers: {

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

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

        "Content-Type": "application/json",

      },

      body: JSON.stringify(body),

    }

  );

  if (!putResp.ok) {

    const errJson = await putResp.json();

    throw new Error(`GitHub API error: ${putResp.status} - ${errJson.message}`);

  }

  return await putResp.json();

}

    // Event: file input change

    fileInput.addEventListener('change', function(e) {

      const newFiles = Array.from(e.target.files);

      newFiles.forEach(f => {

        if (!filesArray.some(existing => existing.name === f.name)) {

          filesArray.push(f);

        }

      });

      renderFileList();

      extractedDataMap.clear();

      resultsDiv.innerHTML = '';

      extractAllBtn.disabled = filesArray.length === 0;

      uploadAllBtn.disabled = true;

      // Clear input to allow re-upload same file if removed

      fileInput.value = "";

    });

    // Extract all files button

    extractAllBtn.addEventListener('click', async () => {

      resultsDiv.innerHTML = 'Extracting... Please wait.';

      extractAllBtn.disabled = true;

      uploadAllBtn.disabled = true;

      extractedDataMap.clear();

      try {

        for (const file of filesArray) {

          const result = await extractDataFromFile(file);

          extractedDataMap.set(result.fileName, result.data);

        }

        renderExtractionResults();

      } catch (err) {

        resultsDiv.innerHTML = `<span style="color:red;">Error: ${err}</span>`;

      } finally {

        extractAllBtn.disabled = false;

      }

    });

   // Upload all extracted data button

uploadAllBtn.addEventListener('click', async () => {

  if (extractedDataMap.size === 0) {

    alert("No extracted data to upload. Please extract first.");

    return;

  }

  uploadAllBtn.disabled = true;

  extractAllBtn.disabled = true;

  resultsDiv.innerHTML = '';

  const uploadStatusContainer = document.createElement('div');

  uploadStatusContainer.style.marginTop = "15px";

  uploadStatusContainer.style.display = "flex";

  uploadStatusContainer.style.flexDirection = "column";

  uploadStatusContainer.style.gap = "10px";

  resultsDiv.appendChild(uploadStatusContainer);

  const statusElements = new Map();

  for (const [filename] of extractedDataMap.entries()) {

    const statusBlock = document.createElement('div');

    statusBlock.className = 'file-item';

    const logo = document.createElement('img');

    logo.src = excelLogoUrl;

    logo.width = logoWidth;

    logo.height = logoHeight;

    logo.alt = "Excel Logo";

    const name = document.createElement('span');

    name.className = 'file-name';

    name.textContent = filename;

    const spinner = document.createElement('div');

    spinner.className = 'spinner';

    spinner.style.width = "18px";

    spinner.style.height = "18px";

    spinner.style.border = "3px solid #000";

    spinner.style.borderTop = "3px solid transparent";

    spinner.style.borderRadius = "50%";

    spinner.style.animation = "spin 0.8s linear infinite";

    statusBlock.appendChild(logo);

    statusBlock.appendChild(name);

    statusBlock.appendChild(spinner);

    uploadStatusContainer.appendChild(statusBlock);

    statusElements.set(filename, { block: statusBlock, spinner });

  }

  let uploadErrors = [];

  let uploadedCount = 0;

  let updatedCount = 0;

  // Utility: Format date to MONTH\_YEAR (e.g., "MAY\_2025")

  function getMonthFolderName(dateStr) {

    if (!dateStr) return "other";

    // Handle different date formats (16/5/2025 or 2025-05-16)

    let dateObj;

    if (dateStr.includes("/")) {

      const [day, month, year] = dateStr.split("/");

      dateObj = new Date(year, month - 1, day);

    } else {

      dateObj = new Date(dateStr);

    }

    if (isNaN(dateObj.getTime())) return "other";

    const monthNames = ["JANUARY", "FEBRUARY", "MARCH", "APRIL", "MAY", "JUNE",

                       "JULY", "AUGUST", "SEPTMBER", "OCTBER", "NOVMBER", "DECMBER"];

    const month = monthNames[dateObj.getMonth()];

    const year = dateObj.getFullYear();

    return `${month}\_${year}`;

  }

  // Modified uploadFileToGitHub to return info about commit message & whether it was update or new file

 // Modified upload function with monthly folders

  async function uploadFileToGitHubWithInfo(filename, contentJson) {

    // Get month folder name from the date in JSON

    const monthFolder = getMonthFolderName(contentJson.date);

    const baseFilename = filename.replace(/\.[^/.]+$/, "");

    const path = `extracted\_data/${monthFolder}/${baseFilename}.json`;

    // First check if file exists to determine if this is an update or new upload

    let sha = null;

    let isUpdate = false;

    try {

      // First try to create the monthly folder if it doesn't exist

      try {

        await fetch(

          `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/extracted\_data/${monthFolder}/.gitkeep`,

          {

            method: "PUT",

            headers: {

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

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

              "Content-Type": "application/json",

            },

            body: JSON.stringify({

              message: `Creating folder for ${monthFolder}`,

              content: btoa(""), // empty file

              branch: REPO\_BRANCH,

            }),

          }

        );

      } catch (folderError) {

        // Folder likely already exists, which is fine

        console.log(`Folder ${monthFolder} likely exists already`);

      }

      // Now check if our specific file exists

      const checkResp = await fetch(

        `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}?ref=${REPO\_BRANCH}`,

        {

          headers: {

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

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

          },

        }

      );

      if (checkResp.status === 200) {

        const fileData = await checkResp.json();

        sha = fileData.sha;

        isUpdate = true;

      }

    } catch (e) {

      console.error("Error checking file existence:", e);

    }

    const commitMessage = isUpdate

      ? `UPDATED: ${filename} in ${monthFolder}`

      : `UPLOADED: ${filename} to ${monthFolder}`;

    const body = {

      message: commitMessage,

      content: btoa(unescape(encodeURIComponent(JSON.stringify(contentJson, null, 2)))),

      branch: REPO\_BRANCH,

    };

    if (sha) body.sha = sha;

    const putResp = await fetch(

      `https://api.github.com/repos/${GITHUB\_USERNAME}/${REPO\_NAME}/contents/${path}`,

      {

        method: "PUT",

        headers: {

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

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

          "Content-Type": "application/json",

        },

        body: JSON.stringify(body),

      }

    );

    if (!putResp.ok) {

      const errJson = await putResp.json();

      throw new Error(`GitHub API error: ${putResp.status} - ${errJson.message}`);

    }

    return { commitMessage, isUpdate, monthFolder, jsonResp: await putResp.json() };

  }

  for (const [filename, data] of extractedDataMap.entries()) {

    try {

      const { commitMessage, isUpdate } = await uploadFileToGitHubWithInfo(filename, data);

     const status = statusElements.get(filename);

status.block.removeChild(status.spinner);

// Add colored tick only

const tick = document.createElement('span');

tick.textContent = "✅";

tick.style.fontWeight = "bold";

tick.style.fontSize = "17px";

tick.style.marginLeft = "8px";

tick.style.userSelect = "none";

tick.style.color = isUpdate ? "black" : "green";

status.block.appendChild(tick);

      if (isUpdate) updatedCount++;

      else uploadedCount++;

    } catch (e) {

      const status = statusElements.get(filename);

      status.block.removeChild(status.spinner);

      const cross = document.createElement('span');

      cross.textContent = "❌";

      cross.style.fontSize = "18px";

      status.block.appendChild(cross);

      uploadErrors.push(`Failed to upload ${filename}: ${e.message}`);

    }

  }

  if (uploadErrors.length > 0) {

    const errMsg = document.createElement('div');

    errMsg.style.color = "red";

    errMsg.style.marginTop = "10px";

    errMsg.style.whiteSpace = "pre-line";

    errMsg.textContent = "Some errors occurred:\n" + uploadErrors.join("\n");

    resultsDiv.appendChild(errMsg);

  }

  // Bottom summary message

const summaryMsg = document.createElement('div');

summaryMsg.style.marginTop = "10px";

summaryMsg.style.fontWeight = "bold";

if (uploadErrors.length === 0) {

  if (uploadedCount > 0 && updatedCount === 0) {

    summaryMsg.textContent = `✅ All uploads processed. (${uploadedCount} file${uploadedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "green";

    summaryMsg.style.fontWeight = "bold";

  } else if (updatedCount > 0 && uploadedCount === 0) {

    summaryMsg.textContent = `✅ All updates processed. (${updatedCount} file${updatedCount > 1 ? "s" : ""})`;

    summaryMsg.style.color = "black";

    summaryMsg.style.fontWeight = "bold";

  } else {

    // New styled message with different colors

    summaryMsg.innerHTML = `

      <span style="color: #8B4513;">✅ All uploads and updates processed.</span>

      <span style="color: green; font-weight: bold;"> (${uploadedCount} upload${uploadedCount !== 1 ? "s" : ""})</span>

      <span style="color: black; font-weight: bold;">, (${updatedCount} update${updatedCount !== 1 ? "s" : ""})</span>

    `;

  }

} else {

  summaryMsg.textContent = `Process completed with errors.`;

  summaryMsg.style.color = "red";

  }

  resultsDiv.appendChild(summaryMsg);

// Auto-reload logic

  const countdown = document.createElement('div');

  countdown.style.marginTop = '10px';

  countdown.style.color = 'black';

  resultsDiv.appendChild(countdown);

  let secondsLeft = 80; // I WILL BE ADD THIS LATER (\*DONT TAKE THIS SERIOS\*)

  const countdownInterval = setInterval(() => {

    secondsLeft--;

    countdown.textContent = `Page Is Reloading In  ${secondsLeft} second${secondsLeft !== 1 ? 's' : ''}...`;

    if (secondsLeft <= 0) {

      clearInterval(countdownInterval);

      location.reload();

    }

  }, 1008); // I WILL BE Edit THIS LATER (\*DONT TAKE THIS SERIOS\*)

  uploadAllBtn.disabled = false;

  extractAllBtn.disabled = false;

});

    // Initial render

    renderFileList();

  </script>

</body>

</html>