

Ah! Perfect — now I understand clearly: you want a fully offline workflow, where: You run the HTML file in VS Code locally Upload ( images or pdf ) to Google Drive via Apps Script backend Automatically generate Drive URLs and save them to Google Sheet Display the generated URL + Copy button in the frontend No browser publishing, no GitHub, no mobile app tell mi stating anuvars create new file and new aheet and not only image pdf also same proses i vill give at one time one one onely image or pdf any one you can janaret url same proses uplodig ->compresing->dive and sheet insid uplodig ->url

ChatGPT said:

Perfect! I can guide you step by step to create a **full workflow** where you can upload **either a single image or PDF** from a **local HTML page in VS Code**, compress (for images), upload to **Google Drive**, automatically record the file in a **Google Sheet**, and get a **direct URL with copy button**.

We will use **Google Apps Script as backend**, **Google Drive folder** for storage, and **Google Sheet** for logging.

---

## Step 1 – Create Google Sheet

1. Go to Google Sheets.
  2. Click + **Blank**.
  3. Create folder in drive URL janareter.
  4. Rename the spreadsheet to URLs.
  5. Copy the **spreadsheet ID** from the URL  
(<https://docs.google.com/spreadsheets/d/1fRV8SDTxZMANWmFZwntzJ0dAQfjF4bxwSJP-Nyls5BU/edit?gid=456033930#gid=456033930> ).
  6. Copy the **folder ID** from the URL  
([https://drive.google.com/drive/folders/1oiulwAcR51t\\_Ky1yH84B9oqZ3R0x9S8o](https://drive.google.com/drive/folders/1oiulwAcR51t_Ky1yH84B9oqZ3R0x9S8o) ).
- 

## Step 2 – Create a Google Drive folder

1. Go to Google Drive → New → Folder → name it URL janareter.
  2. Open the folder → copy the **folder ID** from URL:  
([https://drive.google.com/drive/folders/1oiulwAcR51t\\_Ky1yH84B9oqZ3R0x9S8o](https://drive.google.com/drive/folders/1oiulwAcR51t_Ky1yH84B9oqZ3R0x9S8o) )
- 

## Step 3 – Google Apps Script Backend

1. Go to [script.google.com](https://script.google.com) → New Project → name it FileUploader.
2. Sheet → Extensions → Apps Script → Code.gs → remo code Past this code

### 3. Create Code.gs and paste:

```
var FOLDER_ID = '1oiulwAcR51t_Ky1yH84B9oqZ3R0x9S8o'; // your folder
var SPREADSHEET_ID = '1fRV8SDTxZMANWmFZwntzJOdAQfjF4bxwSJP-Nyls5BU'; // your sheet
var SHEET_NAME = 'URLs'; // your tab

/**
 * Upload file (image/pdf) to Drive + save URL in Sheet
 */
function uploadFile(dataUrl, filename, mimeType) {
  try {
    var blob = Utilities.newBlob(Utilities.base64Decode(dataUrl.split(',')[1]), mimeType,
    filename);

    // Upload to Drive
    var folder = DriveApp.getFolderById(FOLDER_ID);
    var file = folder.createFile(blob);
    file.setSharing(DriveApp.Access.ANYONE_WITH_LINK, DriveApp.Permission.VIEW);

    // Direct image/pdf URL
    var url = 'https://drive.google.com/uc?export=view&id=' + file.getId();

    // Log in Sheet
    var ss = SpreadsheetApp.openById(SPREADSHEET_ID);
    var sheet = ss.getSheetByName(SHEET_NAME);
    if (!sheet) sheet = ss.insertSheet(SHEET_NAME);
    sheet.appendRow([new Date(), filename, mimeType, file.getId(), url]);

    return url;

  } catch (e) {
    throw e.toString();
  }
}

/**
 * POST endpoint for HTML frontend
 */
function doPost(e) {
  const dataUrl = e.parameter.dataUrl;
  const filename = e.parameter.filename;
  const mimeType = e.parameter.mimeType;
  const url = uploadFile(dataUrl, filename, mimeType);
  return ContentService.createTextOutput(url);
}

function doGet() {
  return HtmlService.createHtmlOutput("Use POST method from HTML frontend");
}
```

- Replace PASTE\_YOUR\_FOLDER\_ID\_HERE → your folder ID

- Replace PASTE\_YOUR\_SHEET\_ID\_HERE → your spreadsheet ID
- 

## Step 4 – Deploy Web App

1. Deploy → New Deployment → Web App
  2. Execute as: **Me**
  3. Who has access: **Anyone, even anonymous**
  4. Copy the Web App URL (we will use in VS Code HTML)
- 

## Step 5 – Create HTML Frontend (VS Code)

Create index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Upload Image or PDF to Drive</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        padding: 20px;
        max-width: 600px;
        margin: auto;
        background: #f4f6f9;
      }
      h2 {
        color: #333;
      }
      .card {
        background: #fff;
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
      }
      input,
```

```
button {
padding: 8px;
margin-top: 8px;
}

#upload {
background: #007bff;
color: white;
border: none;
border-radius: 5px;
cursor: pointer;
}

#upload:hover {
background: #0056b3;
}

#status {
margin-top: 15px;
padding: 10px;
background: #e9ecef;
border-radius: 6px;
}

#preview {
margin-top: 15px;
max-width: 100%;
border-radius: 8px;
display: none;
}

#copyBtn {
margin-top: 10px;
padding: 8px 12px;
border: none;
background: #28a745;
color: white;
border-radius: 5px;
cursor: pointer;
display: none;
}
```

```

    }
    #copyBtn:hover {
      background: #218838;
    }
  </style>
</head>
<body>
  <div class="card">
    <h2>Upload Image or PDF</h2>
    <input
      type="file"
      id="file"
      accept="image/*,application/pdf"
    /><br /><br />
    <label>
      >Target image size (KB, only for images):
      <input type="number" id="targetKB" value="50" />
    </label>
    ><br /><br />
    <button id="upload">Upload</button>

    <div id="status"></div>
    <img id="preview" alt="preview" />
    <button id="copyBtn">Copy URL</button>
  </div>

  <script>
    const WEB_APP_URL =
      "https://script.google.com/macros/s/AKfycbwA3YWME5f2bZFm_E
      9eGWrwHkzrlyU6fdTW0YJ3M1MD5D929Kk4l_vPS1JVeCBpb21c/exec";

    // Helpers
    function readImage(file) {
      return new Promise((resolve, reject) => {

```

```
const fr = new FileReader();
fr.onload = () => {
  const img = new Image();
  img.onload = () => resolve(img);
  img.onerror = reject;
  img.src = fr.result;
};
fr.onerror = reject;
fr.readAsDataURL(file);
});
}

function canvasToBlob(canvas, quality) {
  return new Promise((res) => canvas.toBlob(res,
"image/jpeg", quality));
}

function blobToDataURL(blob) {
  return new Promise((res) => {
    const fr = new FileReader();
    fr.onload = () => res(fr.result);
    fr.readAsDataURL(blob);
  });
}

async function compress(file, maxBytes) {
  const img = await readImage(file);
  let w = img.width,
  h = img.height;
  const canvas = document.createElement("canvas");
  const ctx = canvas.getContext("2d");
  let quality = 0.9;
  while (true) {
    canvas.width = w;
    canvas.height = h;
    ctx.clearRect(0, 0, w, h);
    ctx.drawImage(img, 0, 0, w, h);
    const blob = await canvasToBlob(canvas, quality);
```

```
    if (blob.size <= maxBytes || quality < 0.3) return
blob;
    quality -= 0.1;
    if (quality < 0.3) {
        w = Math.round(w * 0.9);
        h = Math.round(h * 0.9);
        quality = 0.9;
    }
}
}
```

```
document.getElementById("upload").addEventListener("click"
, async () => {
    const file = document.getElementById("file").files[0];
    if (!file) return alert("Choose a file");
```

```
    const status = document.getElementById("status");
    status.innerText = "Processing...";
```

```
    try {
        let dataUrl;
        if (file.type.startsWith("image/")) {
            const maxBytes =
                (parseInt(document.getElementById("targetKB").value)
|| 50) *
                1024;
            const blob = await compress(file, maxBytes);
            dataUrl = await blobToDataURL(blob);
        } else if (file.type === "application/pdf") {
            dataUrl = await new Promise((resolve, reject) => {
                const fr = new FileReader();
                fr.onload = () => resolve(fr.result);
                fr.onerror = reject;
                fr.readAsDataURL(file);
```

```
    });  
  } else {  
    return alert("Unsupported file type");  
  }  
}
```

```
const formData = new FormData();  
formData.append("dataUrl", dataUrl);  
formData.append("filename", file.name);  
formData.append("mimeType", file.type);
```

```
status.innerText = "Uploading...";
```

```
// FIX: add CORS-safe request  
const res = await fetch(WEB_APP_URL, {  
  method: "POST",  
  body: formData,  
  mode: "cors", // allow cross origin  
});
```

```
if (!res.ok) throw new Error("Upload failed: " +  
res.status);  
const url = await res.text();
```

```
status.innerHTML =  
' Uploaded! <a href="' +  
url +  
'" target="_blank">' +  
url +  
"</a>";  
if (file.type.startsWith("image/")) {  
  const preview = document.getElementById("preview");  
  preview.src = url;  
  preview.style.display = "block";  
}
```



```
const copyBtn = document.getElementById("copyBtn");
copyBtn.style.display = "inline-block";
copyBtn.onclick = () => {
  navigator.clipboard.writeText(url);
  alert("URL copied!");
};
} catch (e) {
  status.innerText = "Error: " + e;
}
});
</script>
</body>
</html>
```

- Replace PASTE\_YOUR\_WEB\_APP\_URL\_HERE → your deployed Web App URL.
- 

## Step 6 – Run HTML page in VS Code

1. Install **Live Server** extension.
  2. Right-click `index.html` → Open with **Live Server**.
  3. Pick **one file at a time** (image or PDF) → it compresses (images), uploads → logs in **Sheet** → gives **direct URL** with copy button.
- 

### ✅ Features this workflow provides:

- Upload **image or PDF** (single file at a time)
- Compress images to target size
- Upload to Google Drive folder
- Log metadata in Google Sheet: timestamp, filename, type, Drive ID, URL
- Get **direct URL** and **Copy button**
- Fully works from **VS Code locally**