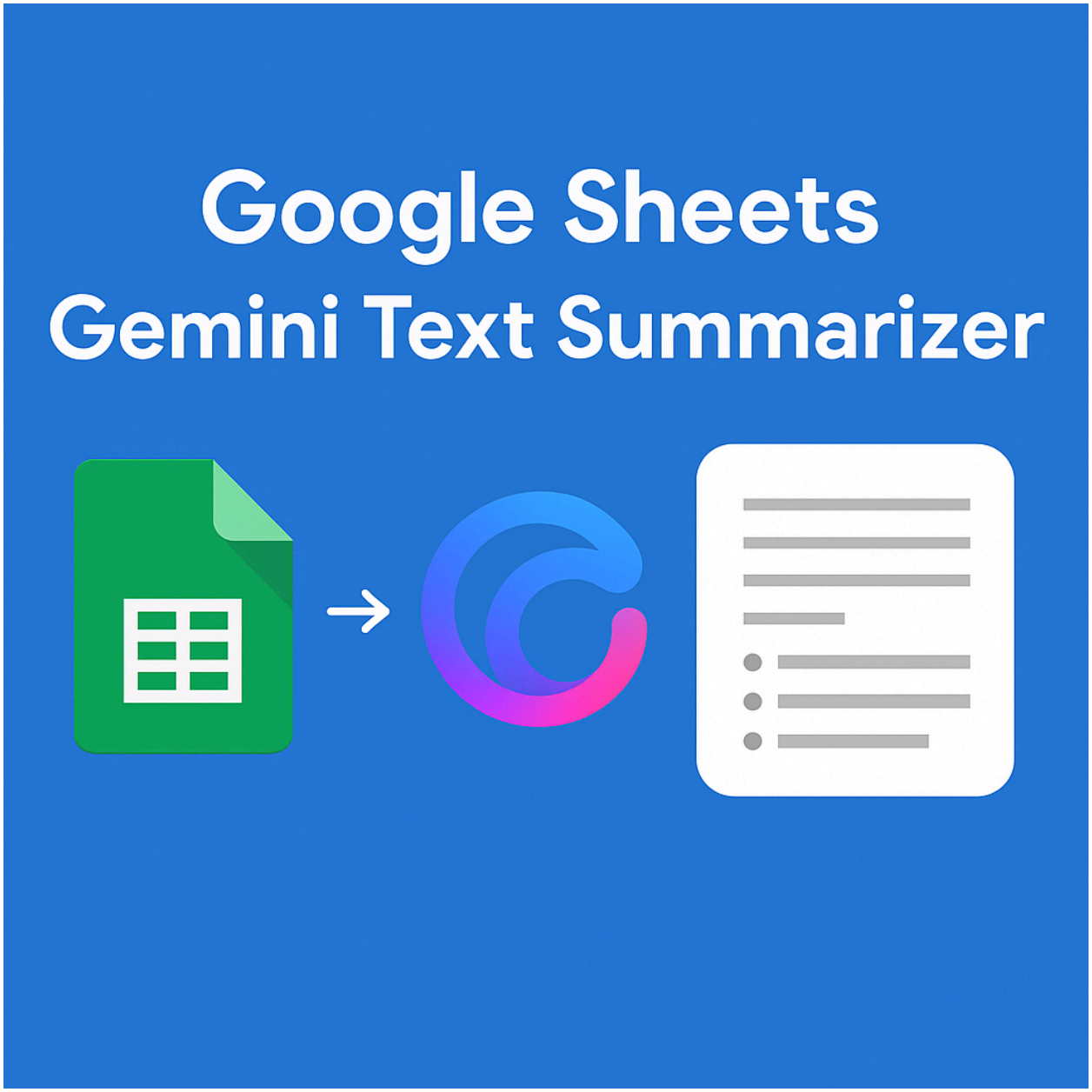
🧠 Build an AI-Powered Text Summarizer in Google Sheets Using Gemini



Github Repo
https://github.com/lsvekis/Build-an-AI-Powered-Text-Summarizer-in-Google-Sheets-Using-Gemini

Get your Gemini API Key
https://aistudio.google.com/app/api-keys

🗓️ Overview

You’ll learn how to connect **Google Sheets** with **Gemini AI**, create custom menu options, build a **sidebar UI**, and summarize text automatically — all powered by **Google Apps Script** and the **Gemini REST API**.

At the end, you’ll have a working app that looks like this:

```
🧠 AI Tools
 ├─ Summarize Selection → next column
 ├─ Summarize Column…
 ├─ Open Sidebar
 └─ Insert Sample Data
```

🧩 Step 1: Create a New Google Sheet and Open Apps Script

1. Open **Google Sheets**.
2. Click **Extensions → Apps Script**.
   This opens the **Apps Script editor** (linked or *bound* to your Sheet).
3. Delete the default `myFunction()`.

We’ll build everything from scratch step-by-step.

🧱 Step 2: Add a Custom Menu to Your Sheet

We'll start by adding a simple custom menu that appears whenever the Sheet is opened.

Code:

```
function onOpen() {
  SpreadsheetApp.getUi()
    .createMenu('🧠 AI Tools')
    .addItem('Open Sidebar', 'showSidebar')
    .addToUi();
}

function showSidebar() {
  const html = HtmlService.createHtmlOutput('<p>Hello from your first
sidebar!</p>')
    .setTitle('Gemini Summarizer');
  SpreadsheetApp.getUi().showSidebar(html);
}
```

Explanation:

- `onOpen()` runs automatically each time the spreadsheet is opened.
- It creates a new menu named 🧠 **AI Tools**.
- When you click **Open Sidebar**, it runs `showSidebar()`, which creates a simple HTML sidebar.

Try It:

1. Click **Save**.
2. Go back to the Sheet and **reload**.
3. You'll see a new menu called 🧠 **AI Tools → Open Sidebar**.
4. Click it — a small sidebar appears. 🎉

⚙️ Step 3: Create a Separate Sidebar File

Instead of hardcoding HTML inside Apps Script, we'll create a reusable HTML file.

1. In the Apps Script editor → **File → New → HTML file**
2. Name it: **Sidebar**
3. Paste this inside:

```
<!DOCTYPE html>
<html>
  <body>
    <h2>🧠 Gemini Summarizer</h2>
    <p>Paste text below to summarize:</p>
    <textarea id="text"
style="width:100%;height:100px"></textarea><br>
    <button onclick="summarize()">Summarize</button>
    <pre id="output"></pre>

    <script>
      function summarize() {
        const text = document.getElementById('text').value;
        document.getElementById('output').innerText = '⏳
Summarizing...';

        google.script.run
          .withSuccessHandler(result => {
            document.getElementById('output').innerText = result;
          })
          .summarizeTextFromSidebar(text);
      }
    </script>
  </body>
</html>
```

Explanation:

- This file creates a simple interface.
- The button calls `summarizeTextFromSidebar()` (we'll define that next).
- `google.script.run` lets HTML talk to your Apps Script functions.

🔐 Step 4: Get a Gemini API Key

You need an API key to connect to Gemini.

1. Visit https://aistudio.google.com/app/apikey
2. Sign in with your Google account.
3. Click **Create API key**.
4. Copy it — we'll paste it in our script.

🌐 Step 5: Connect to Gemini via REST API

We'll now use Apps Script's `UrlFetchApp.fetch()` to call Gemini's API.

Code:

```
const GEMINI_API_KEY = 'YOUR_API_KEY_HERE'; // Replace this

function summarizeWithGemini(text) {
  if (!text) return "⚠️ Please provide text to summarize.";

  const url =
`https://generativelanguage.googleapis.com/v1/models/gemini-2.5-flash:
generateContent?key=${GEMINI_API_KEY}`;

  const payload = {
    contents: [
      { role: "user", parts: [{ text: `Summarize this text in 3–5
bullet points:\n${text}` }] }
    ]
  };

  const options = {
    method: "post",
    contentType: "application/json",
    payload: JSON.stringify(payload)
  };
```

```
  try {
    const response = UrlFetchApp.fetch(url, options);
    const data = JSON.parse(response.getContentText());
    const summary = data?.candidates?.[0]?.content?.parts?.[0]?.text;
    return summary || "⚠️ No summary returned.";
  } catch (e) {
    return "❌ Error: " + e.message;
  }
}
```

Explanation:

- `UrlFetchApp.fetch()` makes an HTTP request.
- `payload` contains the message we send to Gemini.
- The response is JSON — we extract the generated text.

Test It:

Add this small helper:

```
function testGemini() {
  const text = "Artificial intelligence is changing the world by
automating tasks and improving decision-making.";
  const result = summarizeWithGemini(text);
  Logger.log(result);
}
```

Run **testGemini() → View → Logs** — you should see a summary.

🧩 Step 6: Connect the Sidebar to the Function

Add this function to `Code.gs`:

```
function summarizeTextFromSidebar(text) {
  return summarizeWithGemini(text);
}
```

Explanation:

- This is what our HTML sidebar calls via `google.script.run`.
- It simply forwards the text to `summarizeWithGemini()` and returns the result.

Now, when you open the sidebar, paste some text, and click **Summarize**, you'll get an AI-generated summary!

🧱 Step 7: Create Menu Options for Sheets Integration

Let's add tools that summarize selected cells.

Code:

```
function summarizeSelectionToRight() {
  const range = SpreadsheetApp.getActiveRange();
  const values = range.getValues();

  const output = values.map(row => {
    const text = row.join(" ");
    return [summarizeWithGemini(text)];
  });

  const sheet = range.getSheet();
  const startRow = range.getRow();
  const outCol = range.getColumn() + range.getNumColumns();
  sheet.getRange(startRow, outCol, output.length,
1).setValues(output);

  SpreadsheetApp.getActive().toast("✅ Summaries added to next
column");
}
```

Explanation:

- Takes whatever cells are selected.

- Summarizes each row.
- Writes the summaries into the next column.

Add it to your menu:

```
function onOpen() {
  SpreadsheetApp.getUi()
    .createMenu('🧠 AI Tools')
    .addItem('Summarize Selection → next column',
'summarizeSelectionToRight')
    .addItem('Open Sidebar', 'showSidebar')
    .addToUi();
}
```

Now you can **highlight cells**, click **Summarize Selection**, and get automatic summaries in the next column.

🧪 Step 8: Add Sample Data

Add this convenience function:

```
function insertSampleData() {
  const data = [
    ["Original Text", "Summary"],
    ["Artificial intelligence (AI) is transforming industries...",
""],
    ["Photosynthesis is how plants convert light into energy.", ""],
    ["Cloud computing allows access to scalable resources online.",
""]
  ];

  const sheet =
SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.clearContents();
  sheet.getRange(1, 1, data.length, 2).setValues(data);
}
```

Now your menu can include:

```
.addItem('Insert Sample Data', 'insertSampleData')
```

🧠 Step 9: Handling Errors Gracefully

Add this fallback for better user messages:

```
function safeSummarize(text) {
  try {
    return summarizeWithGemini(text);
  } catch (e) {
    return "❌ Error: " + e.message;
  }
}
```

Replace `summarizeWithGemini()` calls in your sheet functions with `safeSummarize()`.

🚀 Step 10: Test the Whole Project

✅ Click 🧠 **AI Tools → Insert Sample Data**
✅ Select column A
✅ Click **Summarize Selection → next column**
✅ Watch column B fill with AI-generated summaries!

🧩 Step 11: (Optional) Custom Function for Formulas

Add this:

```
/**
 * =AI_SUMMARY(A2)
 */
function AI_SUMMARY(text) {
  return summarizeWithGemini(text);
}
```

Now in any cell, you can type:

```
=AI_SUMMARY(A2)
```

and it will generate a live AI summary.

🧭 Step 12: Extend the Project (Next Steps)

You can now try these enhancements:

- Add a **"Rewrite"** or **"Explain"** button in the sidebar.
- Add a **drop-down** to choose summary length (short, medium, long).
- Use the **Gemini-Pro** model for more detail.
- Store the API key securely in **Script Properties**.
- Add caching with `CacheService` to reduce API calls.

🏁 Final Thoughts

By following these steps, you learned:

- How to connect Google Sheets and Apps Script.
- How to call Gemini API via REST.
- How to build UI components (menus + sidebar).
- How to manipulate sheet data dynamically.

You now have a **working AI-powered spreadsheet tool** 🎉
and a reusable framework for any AI feature (translation, rewriting, summarization, etc.).

Awesome — your `Code.gs` is in great shape. To make this a **complete, ready-to-ship project**, here's the remaining "additional content" you need:

---

1) `Sidebar.html` (drop-in UI)

Create a new HTML file named **`Sidebar`** (exactly this name) and paste:

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
    <meta charset="utf-8">
    <style>
      :root { --pad:16px }
      body{font:14px/1.5 system-ui,-apple-system,Segoe
UI,Roboto,Arial,sans-serif;padding:var(--pad)}

textarea{width:100%;height:140px;font-family:ui-monospace,Menlo,Consol
as,monospace}
      button{background:#1a73e8;color:#fff;border:0;padding:8px
12px;border-radius:6px;cursor:pointer}
      button:disabled{opacity:.6;cursor:not-allowed}
      label{font-size:12px;color:#444}
      input[type=number]{width:90px}
      .row{display:flex;gap:8px;align-items:center;margin:10px 0}
      .hint{color:#666;font-size:12px;margin:6px 0 12px}

pre{background:#f1f3f4;padding:10px;border-radius:6px;white-space:pre-
wrap}
    </style>
  </head>
  <body>
    <h2>🧠 Gemini Summarizer</h2>
    <p class="hint">Paste any text below or close this panel and use
the menu to summarize cells.</p>

    <textarea id="text" placeholder="Paste text to
summarize…"></textarea>

    <div class="row">
      <label>Max tokens</label>
      <input id="maxTokens" type="number" min="64" max="2048"
value="512">
```

```
      <label>Temperature</label>
      <input id="temp" type="number" step="0.1" min="0" max="2"
value="0.3">
      <label>Style</label>
      <select id="style">
        <option selected>3–5 concise bullet points</option>
        <option>1 short paragraph</option>
        <option>Headings + bullets</option>
        <option>Executive summary (2–3 bullets)</option>
      </select>
    </div>

    <button id="go" onclick="summarize()">Summarize</button>

    <h3>Summary</h3>
    <pre id="out">Waiting…</pre>

    <script>
      function summarize(){
        const btn = document.getElementById('go');
        const text = document.getElementById('text').value || '';
        const maxTokens =
Number(document.getElementById('maxTokens').value || 512);
        const temp = Number(document.getElementById('temp').value ||
0.3);
        const style = document.getElementById('style').value || '3–5
concise bullet points';

        if (!text.trim()){
          document.getElementById('out').textContent = 'Please paste
some text.';
          return;
        }
        btn.disabled = true;
        document.getElementById('out').textContent = '⏳
Summarizing…';

        google.script.run
          .withSuccessHandler(r => {
            document.getElementById('out').textContent = r ||
'(empty)';
            btn.disabled = false;
          })
          .withFailureHandler(e => {
            document.getElementById('out').textContent = 'Error: ' +
e.message;
            btn.disabled = false;
          })
```

```
          .summarizeTextFromSidebar(JSON.stringify({ text, maxTokens,
temp, style }));
      }
    </script>
  </body>
</html>
```

Wire it to your backend

Your current `summarizeTextFromSidebar(text)` expects a plain string. To support the extra controls, replace the function at the bottom of **`Code.gs`** with this small wrapper:

```
// ==== SERVER ENTRY FOR SIDEBAR ====
function summarizeTextFromSidebar(payloadJson) {
  try {
    const { text, maxTokens, temp, style } = JSON.parse(payloadJson ||
'{}');
    return summarizeWithGemini(text || '', {
      temperature: (typeof temp === 'number' ? temp :
DEFAULT_TEMPERATURE),
      maxOutputTokens: (typeof maxTokens === 'number' ? maxTokens :
DEFAULT_MAX_TOKENS),
      style: style || DEFAULT_SUMMARY_STYLE
    });
  } catch (e) {
    return '❌ Error parsing sidebar payload: ' + e.message;
  }
}
```

---

2) `appsscript.json` (manifest with correct scopes)

In the Apps Script editor, open **Project Settings** ⟶ enable **“Show appsscript.json”**, then set:

```
{
  "timeZone": "America/Toronto",
  "exceptionLogging": "STACKDRIVER",
  "runtimeVersion": "V8",
  "oauthScopes": [
    "https://www.googleapis.com/auth/spreadsheets",
    "https://www.googleapis.com/auth/script.container.ui",
```

```
    "https://www.googleapis.com/auth/script.external_request"
  ]
}
```

Save, then **Run → debugOpenSidebar()** once to authorize. Reload the Sheet so the 🧠 **AI Tools** menu appears.

---

3) (Optional but recommended) Store API key in Script Properties

This keeps your key out of source code:

1. **Project Settings → Script properties → Add row**
   Key: `GEMINI_API_KEY`
   Value: your API key

2. Your `getApiKey()` already reads properties first. Once set, you can blank out the constant in code if you want:

```
const GEMINI_API_KEY = '';
```

Or leave it as a fallback — up to you.

---

4) UX polish (sheet helpers you already have)

- **Insert Sample Data**: loads A/B with curated examples.
- **Summarize Selection → next column**: works across multi-row selections.
- **Summarize Column…**: prompt `A -> B` (or just `A`) to batch process a whole column.
- Small throttle (`REQUEST_COOLDOWN_MS`) to avoid quota bursts.

---

5) QA checklist (do these once)

- Save all files.
- Manifest scopes match above.
- **Run → debugOpenSidebar()** → approve permissions.
- Back in the sheet: reload → 🧠 **AI Tools** menu appears.
- Try **Insert Sample Data**, then **Summarize Column…** (`A -> B`).
- Open **Sidebar**, paste text, adjust **Max tokens / Temperature / Style**, **Summarize**.

---

6) Troubleshooting (fast answers)

- **"Specified permissions are not sufficient … Ui.showSidebar"**
  You're likely in a *standalone* script. Reopen the editor from the Sheet via **Extensions → Apps Script** (bound project). Ensure manifest includes `script.container.ui`, then re-authorize (Run any UI function once).
- **"You do not have permission to call UrlFetchApp.fetch … external_request"**
  Run a function that triggers fetch (e.g., `testGeminiOK()`) and approve the OAuth consent. Make sure `script.external_request` is in `appsscript.json`.
- **HTTP 404**
  Use discovered models (you've confirmed: **`gemini-2.5-flash`**, **`gemini-2.5-pro`**, etc.). The client already tries both `v1` and `v1beta` and falls back through candidates.
- **No text / MAX_TOKENS**
  You already bumped `DEFAULT_MAX_TOKENS` to **512** and the extractor reports truncation. Increase to 768/1024 *or* shorten inputs.
- **Safety block**
  The extractor prints the block reason. Reword input or (carefully) relax safety (commented template in your payload).

---

7) Nice extensions (copy-paste ready)

A) Add “Rewrite” and “Explain” menu actions

```
function rewriteSelectionToRight() {
  runTransformOverSelection('Rewrite the following text to be clearer
and more concise. Return only the final rewrite:\n\n{{TEXT}}');
}

function explainSelectionToRight() {
  runTransformOverSelection('Explain the following text simply for a
general audience. Return only the explanation:\n\n{{TEXT}}');
}

function runTransformOverSelection(instruction) {
  const range = SpreadsheetApp.getActiveRange();
  if (!range) {
    SpreadsheetApp.getUi().alert('Please select one or more cells
first.');
    return;
  }
  const sheet = range.getSheet();
  const values = range.getValues();
  const startRow = range.getRow();
  const outCol = range.getColumn() + range.getNumColumns();

  const out = [];
  for (let r = 0; r < values.length; r++) {
    const rowText = values[r].map(v => (v == null ? '' :
String(v))).join(' ').trim();
    if (!rowText) { out.push(['']); continue; }
    const prompt = instruction.replace('{{TEXT}}', rowText);
    const result = callGeminiREST(prompt, { temperature: 0.3,
maxOutputTokens: 512 });
    out.push([result]);
    Utilities.sleep(REQUEST_COOLDOWN_MS);
  }
  sheet.getRange(startRow, outCol, out.length, 1).setValues(out);
  SpreadsheetApp.getActive().toast('Done.');
}
```

Add these to your menu:

```
.createMenu('🧠 AI Tools')
```

```
  .addItem('Summarize Selection → next column',
'summarizeSelectionToRight')
  .addItem('Rewrite Selection → next column',
'rewriteSelectionToRight')
  .addItem('Explain Selection → next column',
'explainSelectionToRight')
  .addItem('Summarize Column…', 'summarizeColumnPrompt')
  .addItem('Open Sidebar', 'showSidebar')
  .addSeparator()
  .addItem('Insert Sample Data', 'insertSampleData')
  .addToUi();
```

B) Single-cell custom functions

Already included:

```
// =AI_SUMMARY(A2, "1 short paragraph")
function AI_SUMMARY(text, style) {
  return summarizeWithGemini(text, { style: style ||
DEFAULT_SUMMARY_STYLE });
}
```

Add variants:

```
// =AI_REWRITE(A2)
function AI_REWRITE(text) {
  const t = (text == null ? '' : String(text)).trim();
  if (!t) return '';
  return callGeminiREST('Rewrite the following to be clearer and more
concise:\n\n' + t,
                        { temperature: 0.3, maxOutputTokens: 512 });
}

// =AI_EXPLAIN(A2)
function AI_EXPLAIN(text) {
  const t = (text == null ? '' : String(text)).trim();
  if (!t) return '';
  return callGeminiREST('Explain the following simply for a general
audience:\n\n' + t,
                        { temperature: 0.3, maxOutputTokens: 512 });
}
```

---

8) Final “build & ship” steps

1. **Authorize** (run `debugOpenSidebar()` and/or `testGeminiOK()` once).
2. **Reload** the Sheet → use the 🧠 **AI Tools** menu.
3. (Optional) **Share** the sheet with editors/viewers; the script runs under each user’s authorization the first time they invoke it.
4. If you turn this into an **Add-on**, keep your REST calls and UI — it ports cleanly.

Full Code

```
/**
* Google Sheets Gemini Text Summarizer (REST API version)
* Uses UrlFetchApp to call Gemini via HTTPS with an API key.
* Author: Laurence Svekis, GDE for Google Workspace
*
* Menu actions:
*  - Summarize Selection → next column
*  - Summarize Column…
*  - Open Sidebar
*  - Insert Sample Data
*
* NOTE: Consider moving your API key to PropertiesService for security.
*/

// ==== CONFIG ====
// (You can also store the key in PropertiesService and read it via
getApiKey() below.)
const GEMINI_API_KEY = 'AIz***DQ';

// Prefer stable 2.5/2.0 text models (present in both v1 and v1beta for your
key)
const GEMINI_MODEL_CANDIDATES = [
 'gemini-2.5-flash',
 'gemini-2.5-pro',
 'gemini-2.0-flash',
 'gemini-2.0-flash-001',
 'gemini-2.0-flash-lite',
 'gemini-2.0-flash-lite-001'
];

// Default model we start with (the client will auto-fall back if needed)
let GEMINI_MODEL = 'gemini-2.5-flash';
```

```
const DEFAULT_TEMPERATURE    = 0.3;    // a bit more deterministic
const DEFAULT_MAX_TOKENS     = 512;    // more room so we don't hit MAX_TOKENS

const DEFAULT_SUMMARY_STYLE = '3–5 concise bullet points';
const REQUEST_COOLDOWN_MS    = 150; // small delay to avoid rate spikes

// Try both API bases; your key exposes models on both.
const GEMINI_BASES = [
 'https://generativelanguage.googleapis.com/v1',
 'https://generativelanguage.googleapis.com/v1beta'
];

// ==== MENU / UI ====
function onOpen() {
 SpreadsheetApp.getUi()
   .createMenu('🧠 AI Tools')
   .addItem('Summarize Selection → next column', 'summarizeSelectionToRight')
   .addItem('Summarize Column…', 'summarizeColumnPrompt')
   .addItem('Open Sidebar', 'showSidebar')  // ← this name must match exactly
   .addSeparator()
   .addItem('Insert Sample Data', 'insertSampleData')
   .addToUi();
}


function showSidebar() {
 try {
   const html = HtmlService.createHtmlOutputFromFile('Sidebar')
     .setTitle('Gemini Summarizer');
   SpreadsheetApp.getUi().showSidebar(html);
 } catch (e) {
   // If Sidebar.html is missing or misnamed, you'll see this alert
   SpreadsheetApp.getUi().alert('Failed to open sidebar: ' + e.message);
   throw e;
 }
}
```

```
// Opens a super simple test sidebar to confirm UI works at all
function debugOpenSidebar() {
 const html = HtmlService.createHtmlOutput('<div
style="padding:16px;font-family:sans-serif">Hello from test sidebar
✅</div>')
   .setTitle('Test Sidebar');
 SpreadsheetApp.getUi().showSidebar(html);
}

// ==== PUBLIC HELPERS FOR QUICK TESTS ====
function testGeminiOK() {
 const txt = 'Artificial intelligence is transforming industries by improving
decision-making and enabling automation.';
 const out = summarizeWithGemini(txt, { maxOutputTokens: 512, temperature: 0.3
});
 Logger.log(out);
}


// ==== CORE: REST CLIENT ====

// If present, read API key from PropertiesService; else fall back to
constant.
function getApiKey() {
 const key =
PropertiesService.getScriptProperties().getProperty('GEMINI_API_KEY');
 return (key && key.trim()) ? key.trim() : GEMINI_API_KEY;
}

/**
* Calls Gemini generateContent; retries across models/base paths on 404.
* @param {string} prompt
* @param {{temperature?: number, maxOutputTokens?: number, model?: string}}
[opts]
* @returns {string}
```

```
*/
// --- helper: pull text from any parts; show safety blocks nicely
function extractGeminiText(json) {
 try {
   if (json?.error?.message) return `❌ API error: ${json.error.message}`;

   const block = json?.promptFeedback?.blockReason;
   if (block) {
     const details = (json.promptFeedback?.safetyRatings || [])
       .map(r => `${r.category}:${r.probability}`).join(', ');
     return `❌ Blocked by safety (${block}${details ? ' — '+details :
''}).`;
   }

   const cand = json?.candidates?.[0];
   if (!cand) return '';

   // If the model cut off due to max tokens, tell the user explicitly.
   if (cand.finishReason === 'MAX_TOKENS') {
     // Try to pull any partial text if present
     const parts = cand?.content?.parts || [];
     const partial = parts.map(p => p?.text || '').join('').trim();
     return partial
       ? `${partial}\n\n⚠️ Truncated (MAX_TOKENS). Increase maxOutputTokens
or shorten input.`
       : '⚠️ Truncated (MAX_TOKENS) and no visible text. Increase
maxOutputTokens or shorten input.';
   }

   const parts = cand?.content?.parts || [];
   const text = parts.map(p => p?.text || '').join('').trim();
   if (text) return text;

   const cText = cand?.text;
   if (cText && String(cText).trim()) return String(cText).trim();
```

```
    return '';
  } catch (_) {
    return '';
  }
}


function callGeminiREST(prompt, opts = {}) {
  const apiKey = getApiKey();
  if (!apiKey || apiKey === 'YOUR_API_KEY_HERE') {
    return '❌ Error: Set GEMINI_API_KEY in Code.gs or Script Properties.';
  }

  const temperature     = opts.temperature ?? DEFAULT_TEMPERATURE;
  const maxOutputTokens = opts.maxOutputTokens ?? DEFAULT_MAX_TOKENS;

  const primaryModel = (opts.model || GEMINI_MODEL);
  const discovered   = getDiscoveredModels();
  const cascadeModels = unique([primaryModel].concat(discovered.length ?
discovered : []).concat(GEMINI_MODEL_CANDIDATES));

const payload = {
  generationConfig: {
    temperature: temperature,
    maxOutputTokens: maxOutputTokens,
    // (optional tunables)
    topP: 0.95,
    topK: 40
  },
  // If you later want to try responseMimeType, put it here (top-level), not
inside generationConfig:
  // responseMimeType: "text/plain",
  contents: [
    { role: "user", parts: [{ text: prompt }] }
  ]
};
```

```
 let lastBody = '';
 let lastCode = 0;

 for (const base of GEMINI_BASES) {
   for (const model of cascadeModels) {
     const url =
`${base}/models/${encodeURIComponent(model)}:generateContent?key=${encodeURICo
mponent(apiKey)}`;
     const options = {
       method: 'post',
       contentType: 'application/json',
       payload: JSON.stringify(payload),
       muteHttpExceptions: true
     };

     try {
       const res  = UrlFetchApp.fetch(url, options);
       const code = res.getResponseCode();
       const body = res.getContentText();
       lastBody = body; lastCode = code;

       if (code >= 200 && code < 300) {
         const data = JSON.parse(body);
         const text = extractGeminiText(data);
         if (text) {
           GEMINI_MODEL = model; // remember working model
           return text;
         }
         // Log the full JSON once to help diagnose why it was empty
         Logger.log(`Empty text for ${model} @ ${base}: ${body}`);
         return '⚠️ No summary returned (see Logs for raw response).';
       }
```

```
      if (code === 404) continue; // try next model/base

      Logger.log(`Gemini HTTP ${code} @ ${url}\n${body}`);
      return `❌ Error: Gemini HTTP ${code}. See Logs for details.`;

    } catch (err) {
      Logger.log(`Gemini fetch error @ ${base}/${model}: ${err}`);
      continue;
    }
  }
 }

 Logger.log(`Gemini final failure. HTTP ${lastCode}\n${lastBody}`);
 if (lastCode === 404) {
   return '❌ Error: Model/endpoint not found (404). Run debugListModels()
and set GEMINI_MODEL.';
 }
 return `❌ Error: Gemini HTTP ${lastCode || 'unknown'}. See Logs for
details.`;
}



// ==== MODEL DISCOVERY (optional but handy) ====

/**
* Lists models visible to your API key and caches them (6 hours).
* Run this once and check the Logs; then set GEMINI_MODEL to a listed id.
*/
function debugListModels() {
 const apiKey = getApiKey();
 if (!apiKey || apiKey === 'YOUR_API_KEY_HERE') {
   Logger.log('Set GEMINI_API_KEY first.');
   return;
 }
 const urlV1b =
```

```
`https://generativelanguage.googleapis.com/v1beta/models?key=${encodeURICompon
ent(apiKey)}`;
 const urlV1  =
`https://generativelanguage.googleapis.com/v1/models?key=${encodeURIComponent(
apiKey)}`;

 [urlV1b, urlV1].forEach(url => {
   try {
     const res  = UrlFetchApp.fetch(url, { muteHttpExceptions: true });
     const code = res.getResponseCode();
     const body = res.getContentText();
     Logger.log(`GET ${url} → ${code}`);
     if (code >= 200 && code < 300) {
       const data = JSON.parse(body);
       const ids = (data?.models || []).map(m =>
m?.name?.split('/').pop()).filter(Boolean);
       Logger.log('Available models: ' + ids.join(', '));
       if (ids.length) {
         CacheService.getScriptCache().put('gemini_models',
JSON.stringify(ids), 6 * 60 * 60);
       }
     } else {
       Logger.log(body);
     }
   } catch (e) {
     Logger.log(`Error listing models from ${url}: ${e}`);
   }
 });
}

function getDiscoveredModels() {
 try {
   const raw = CacheService.getScriptCache().get('gemini_models');
   if (!raw) return [];
   const ids = JSON.parse(raw);
   return Array.isArray(ids) ? ids : [];
```

```
 } catch (e) {
   return [];
 }
}

function unique(arr) {
 return Array.from(new Set(arr.filter(Boolean)));
}

// ==== SUMMARIZATION WRAPPER ====

/**
* Summarize a block of text with Gemini (REST).
* @param {string} text
* @param {{temperature?: number, maxOutputTokens?: number, style?: string,
model?: string}} [opts]
* @return {string}
*/
function summarizeWithGemini(text, opts = {}) {
 const content = (text == null ? '' : String(text)).trim();
 if (!content) return '';

 const style = opts.style ?? DEFAULT_SUMMARY_STYLE;
 const prompt = [
   `Summarize the following text clearly in ${style}.`,
   'Keep facts accurate and avoid inventing details.',
   '',
   content
 ].join('\n');

 return callGeminiREST(prompt, opts);
}

// ==== SHEETS ACTIONS ====

// Selection → next column
```

```
function summarizeSelectionToRight() {
 const range = SpreadsheetApp.getActiveRange();
 if (!range) {
   SpreadsheetApp.getUi().alert('Please select one or more cells first.');
   return;
 }

 const sheet    = range.getSheet();
 const values   = range.getValues();
 const startRow = range.getRow();
 const startCol = range.getColumn();
 const outCol   = startCol + range.getNumColumns(); // immediate right of
selection

 const out = [];
 for (let r = 0; r < values.length; r++) {
   const row = values[r];
   const cellText = row.map(x => (x == null ? '' : String(x))).join('
').trim();
   if (!cellText) {
     out.push(['']);
     continue;
   }
   const summary = summarizeWithGemini(cellText);
   out.push([summary]);
   Utilities.sleep(REQUEST_COOLDOWN_MS);
 }

 sheet.getRange(startRow, outCol, out.length, 1).setValues(out);
 SpreadsheetApp.getActive().toast(`Summarized ${out.length} row(s) → column
${colToLetter(outCol)}`);
}

// Prompt for columns, then batch summarize
function summarizeColumnPrompt() {
 const ui = SpreadsheetApp.getUi();
```

```
  const input = ui.prompt(
    'Summarize Column',
    'Enter input and output columns (e.g., A -> B or A,B). If you enter only A,
output will be the next column.',
    ui.ButtonSet.OK_CANCEL
  );
  if (input.getSelectedButton() !== ui.Button.OK) return;

  const raw = input.getResponseText().trim().toUpperCase();
  let [inCol, outCol] = raw.replace(/\s+/g, '').replace('->', ',').split(',');

  if (!/^[A-Z]+$/.test(inCol || '')) {
    ui.alert('Please provide a valid input column letter, e.g., A');
    return;
  }
  if (!outCol) {
    outCol = colToLetter(letterToCol(inCol) + 1);
  } else if (!/^[A-Z]+$/.test(outCol)) {
    ui.alert('Please provide a valid output column letter, e.g., B');
    return;
  }

  summarizeColumn(inCol, outCol);
}

/**
* Summarize every non-blank cell in input column and write to output column.
* @param {string} inColLetter e.g., "A"
* @param {string} outColLetter e.g., "B"
*/
function summarizeColumn(inColLetter, outColLetter) {
  const sheet   = SpreadsheetApp.getActiveSheet();
  const lastRow = sheet.getLastRow();
  const inCol   = letterToCol(inColLetter);
  const outCol  = letterToCol(outColLetter);
```

```
 if (lastRow < 1) {
   SpreadsheetApp.getUi().alert('No data found.');
   return;
 }

 const range  = sheet.getRange(1, inCol, lastRow, 1);
 const values = range.getValues();

 const out = [];
 for (let r = 0; r < values.length; r++) {
   const text = values[r][0];
   const trimmed = (text == null ? '' : String(text)).trim();
   if (!trimmed) {
     out.push(['']);
     continue;
   }
   const summary = summarizeWithGemini(trimmed);
   out.push([summary]);
   Utilities.sleep(REQUEST_COOLDOWN_MS);
 }

 sheet.getRange(1, outCol, out.length, 1).setValues(out);
 SpreadsheetApp.getActive().toast(`Summarized ${out.length} row(s):
${inColLetter} → ${outColLetter}`);
}

// ==== SAMPLE DATA ====
function insertSampleData() {
 const data = [
   ['Original Text', 'AI Summary'],
   ['Artificial intelligence (AI) is transforming industries by automating
processes, improving decision-making, and enabling innovation in fields like
healthcare and finance.', ''],
   ['Photosynthesis is the process by which plants convert light energy into
chemical energy stored in glucose, using water and carbon dioxide.', ''],
   ['In project management, defining a clear scope and measurable objectives
```

```
helps ensure team alignment and successful delivery.', ''],
   ['Cloud computing provides scalable, on-demand access to computing
resources over the internet, reducing the need for physical infrastructure.',
''],
   ['The Industrial Revolution in the 18th century marked a major turning
point in human history, introducing machines, factories, and urbanization.',
''],
   ['Exercise not only improves physical health but also enhances mood,
cognitive function, and overall mental well-being.', ''],
   ['Renewable energy sources like solar and wind are critical in reducing
carbon emissions and combating climate change.', ''],
   ['The Great Wall of China was built to protect against invasions and stands
as one of the most iconic architectural achievements in history.', ''],
   ['Machine learning is a subset of AI where algorithms learn from data to
make predictions or decisions without explicit programming.', ''],
   ['The human brain contains billions of neurons that communicate through
electrical and chemical signals, enabling thought, memory, and emotion.', ''],
   ['Social media has reshaped communication, marketing, and global
connectivity, but it also raises concerns about privacy and misinformation.',
''],
   ['Climate change refers to long-term shifts in temperature and weather
patterns, largely driven by human activities such as fossil fuel use.', ''],
   ['The internet has revolutionized access to information, commerce, and
communication, connecting billions of people worldwide.', ''],
   ['A balanced diet includes carbohydrates, proteins, fats, vitamins, and
minerals to support overall health and body function.', ''],
   ['The moon's gravitational pull causes tides on Earth and has been a
subject of scientific and cultural fascination for centuries.', ''],
   ['Electric vehicles use battery-powered motors instead of internal
combustion engines, reducing pollution and reliance on fossil fuels.', ''],
   ['The printing press, invented by Johannes Gutenberg, made mass
communication possible and transformed education and literacy.', ''],
   ['Artificial neural networks mimic the structure of the human brain to
process complex data patterns in tasks like image recognition.', ''],
   ['Emotional intelligence involves recognizing and managing one's own
emotions as well as understanding others' feelings effectively.', ''],
```

```
    ['Space exploration expands scientific knowledge, drives innovation, and inspires humanity to reach beyond Earth's boundaries.', '']
  ];

  const sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.clearContents();
  sheet.getRange(1, 1, data.length, data[0].length).setValues(data);
  SpreadsheetApp.getActive().toast('✅ Sample data inserted in columns A–B');
}

// ==== UTILITIES ====
function colToLetter(col) {
  let temp = '';
  let letter = '';
  while (col > 0) {
    temp = (col - 1) % 26;
    letter = String.fromCharCode(temp + 65) + letter;
    col = (col - temp - 1) / 26;
  }
  return letter;
}

function letterToCol(letter) {
  let col = 0;
  for (let i = 0; i < letter.length; i++) {
    col = col * 26 + (letter.charCodeAt(i) - 64);
  }
  return col;
}

// ==== SERVER ENTRY FOR SIDEBAR ====
function summarizeTextFromSidebar(text) {
  return summarizeWithGemini(text, {
    temperature: DEFAULT_TEMPERATURE,
    maxOutputTokens: DEFAULT_MAX_TOKENS
  });
```

```
}

// ==== (Optional) Custom Function for single-cell formulas ====
// =AI_SUMMARY(A2, "1 short paragraph")
/**
* @param {string} text
* @param {string=} style
* @return {string}
* @customfunction
*/
function AI_SUMMARY(text, style) {
 return summarizeWithGemini(text, { style: style || DEFAULT_SUMMARY_STYLE });
}
```

Sidebar.html

```
<!DOCTYPE html>
<html>
 <head>
   <base target="_top">
   <meta charset="utf-8">
   <style>
     body{font:14px/1.5 system-ui, -apple-system, Segoe UI, Roboto, Arial,
sans-serif; padding:16px}
     textarea{width:100%; height:140px; font-family:ui-monospace, Menlo,
Consolas, monospace}
     button{background:#1a73e8; color:#fff; border:0; padding:8px 12px;
border-radius:6px; cursor:pointer}
     pre{background:#f1f3f4; padding:10px; border-radius:6px;
white-space:pre-wrap}
   </style>
 </head>
 <body>
   <h2>🧠 Gemini Summarizer</h2>
   <p>Paste text and click Summarize, or close this and use the menu
actions.</p>
```

```
    <textarea id="text" placeholder="Paste text…"></textarea>
    <div style="margin:8px 0">
      <button onclick="summarize()">Summarize</button>
    </div>
    <h3>Summary</h3>
    <pre id="out">Waiting…</pre>

    <script>
      function summarize(){
        const text = document.getElementById('text').value;
        document.getElementById('out').textContent = '⏳ Summarizing…';
        google.script.run
          .withSuccessHandler(t => document.getElementById('out').textContent =
t || '(empty)')
          .withFailureHandler(e => document.getElementById('out').textContent =
'Error: ' + e.message)
          .summarizeTextFromSidebar(text);
      }
    </script>
  </body>
</html>
```