**Google-Doc Implementation Information**

**Timing- 2 Months**

### Pricing - Only pay for what you use-

With Google Cloud’s pay-as-you-go pricing structure, you only pay for the services you use. No up-front fees. No termination charges. Pricing varies by product and usage—[view detailed price list.](https://cloud.google.com/pricing/list)

[**https://cloud.google.com/pricing/list#section-4**](https://cloud.google.com/pricing/list#section-4)

Click a product name below to view its pricing details. New Google Cloud customers get [$300 in free credits](https://cloud.google.com/free/docs/free-cloud-features) for 90 days to run, test, and deploy workloads. All customers can use [25+ products for free](https://cloud.google.com/free/docs/gcp-free-tier#free-tier), up to monthly usage limits.

**Instructions to integrate Google Docs into your application:**

**1. \*Create a Google Cloud Project:\***

- Go to the [Google Cloud Console](<https://console.cloud.google.com/>).

- Create a new project or use an existing one.

**2. \*Enable the Google Docs API:\***

- In the Google Cloud Console, navigate to "APIs & Services" > "Dashboard."

- Click on the "+ ENABLE APIS AND SERVICES" button.

- Search for "Google Docs API" and enable it for your project.

**3. \*Set up OAuth 2.0 Credentials:\***

- In the Google Cloud Console, navigate to "APIs & Services" > "Credentials."

- Click on "Create credentials" and select "OAuth client ID."

- Configure the OAuth consent screen with the necessary information, including the authorized domains and application name.

- Select "Web application" as the application type.

- Add authorized redirect URIs where users will be redirected after granting access (e.g., `http://localhost:8080/callback`).

- Click "Create" to create the OAuth client ID.

- You'll receive a client ID and client secret; keep these for later use.

**4. \*Implement OAuth 2.0 Authentication:\***

- In your application, use a library or SDK to handle OAuth 2.0 authentication. Common libraries include OAuth2.0 libraries for your specific programming language (e.g., `google-auth-library` for Node.js).

- Authenticate users and obtain an access token using the OAuth 2.0 flow. You'll need to redirect users to Google's OAuth consent screen for approval.

**5. \*Use the Google Docs API:\***

- With an authenticated user, you can now make requests to the Google Docs API to create, read, update, and manage Google Docs documents.

- Refer to the [Google Docs API documentation](<https://developers.google.com/docs/api>

) for details on the API endpoints and how to use them.

**6. \*Handle Document Manipulation:\***

- Depending on your use case, you can create new Google Docs documents, retrieve existing ones, edit content, format documents, and more using API calls.

**7. \*Manage Access and Permissions:\***

- Implement access control and document sharing as needed for your application. You can control who can view, edit, or comment on documents.

* Use [documents.create](https://developers.google.com/docs/api/reference/rest/v1/documents/create) to create a document.
* Use [documents.get](https://developers.google.com/docs/api/reference/rest/v1/documents/get) to retrieve the contents of a specified document.
* Use [documents.batchUpdate](https://developers.google.com/docs/api/reference/rest/v1/documents/batchUpdate) to atomically perform a set of updates on a specified document.
* The get and batchUpdate methods require a documentId (see below) as a parameter to specify the target document. The create method returns an instance of the created document, from which you can read the ID.

**8. \*Handle Errors and Exceptions:\***

- Be prepared to handle errors and exceptions that may occur during API interactions. Google provides error codes and descriptions to help with troubleshooting.