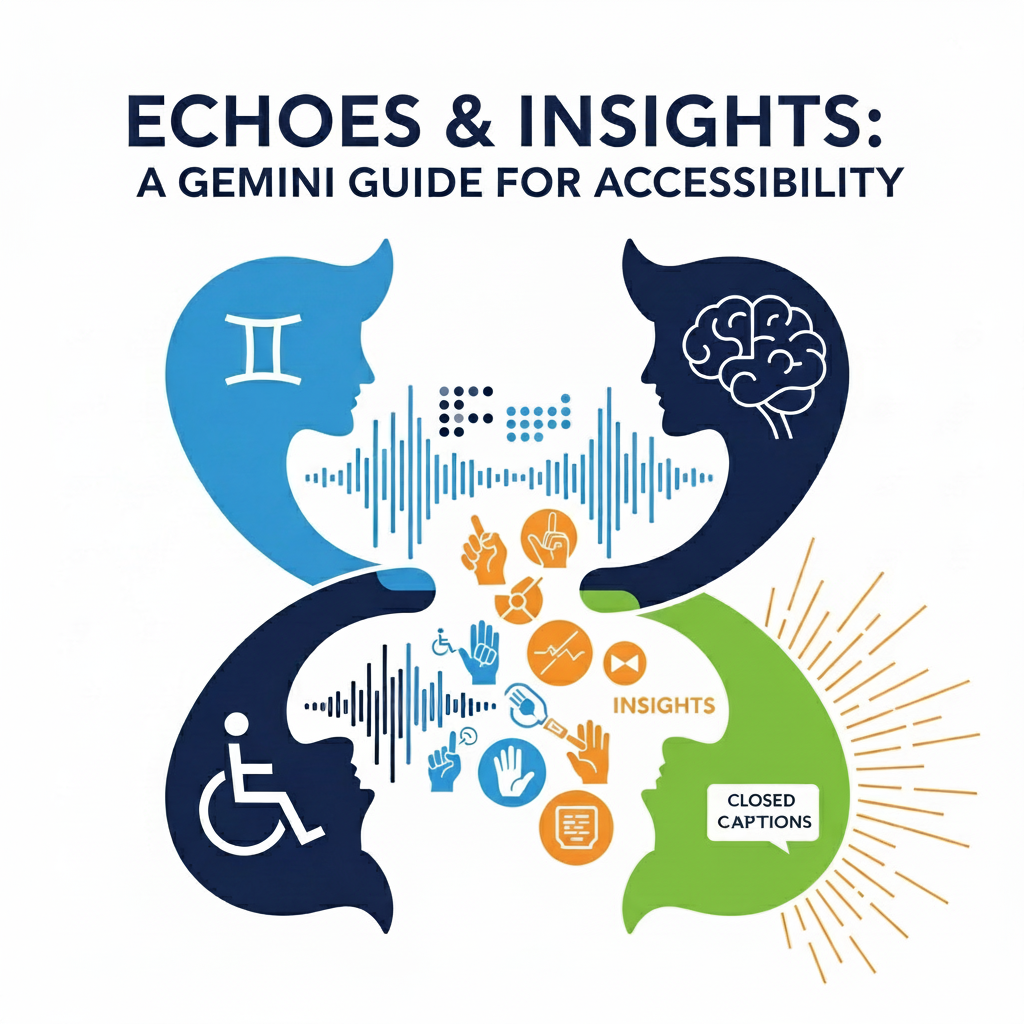
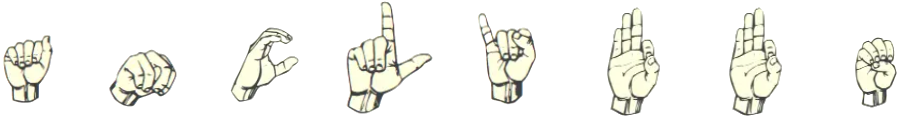
## horizontal line



Echoes & Insights: A Gemini Guide for Accessibility

⠠⠑⠉⠓⠕⠑⠎ ⠯ ⠠⠊⠝⠎⠊⠛⠓⠞⠎⠒ ⠠⠁ ⠠⠛⠑⠍⠊⠝⠊ ⠠⠛⠥⠊⠙⠑ ⠋⠕⠗ ⠠⠁⠉⠉⠑⠎⠎⠊⠃⠊⠇⠊⠞⠽

09.26.2025 (**DRAFT)**

─  
Stephen Ancliffe   
⠠⠎⠞⠑⠏⠓⠑⠝⠀⠠⠁⠝⠉⠇⠊⠋⠋⠑  
 

# Table of Contents

**Introduction: A Guide to Accessibility with Gemini**

* [**Project Overview:**](#_au51mny0sx6) A brief summary of the **Echoes & Insights** initiative.
* [**Our Goals:**](#_3at9u9s4e0vp) The core objectives of the project, including its **Zero-Cost Accessibility** policy.
* [**Specifications:**](#_4p7xi5bvhxdr) Detailed project requirements, including the Zero-Cost Accessibility policy and functions of the Closed Captioning and Braille Generation tools.
* [**Warning on Copyright and Gemini-Generated Content:**](#_fq22otqiu7p1)Important warning about the dangers and legal risks of using copyrighted material with Gemini-generated content.

**Part I: Closed Captioning**

* [**Using Gemini for Closed Captioning:**](#_g5z8j48s3gtx) A step-by-step guide for generating subtitles.
* [**Troubleshooting & Limitations:**](#_4sg6laa2hpi9) Common issues and current barriers in the process.

**Part II: Braille Generation & Analysis**

* [**Establishing a Baseline for Braille Accuracy:**](#_2i2nhq1asfil)Baseline reference for braille.
* [**Converting Text to Braille Unicode with Python:**](#_y9zcx1mvsg4q)This section provides a guide on using a custom Python script to convert text into Braille Unicode.
* [**Converting Braille Unicode to text with Python:**](#_xrw3ualvsyen)This section provides a guide on using a custom Python script to Braille Unicode into text.
* [**Voice Input and Braille Unicode Generation:**](#_x4oy2ldlldb4) How to use voice input to generate braille unicode.
* [**Google Translate rough mockup with Braille support:**](#_53vz263h37n) A rough mockup of what a fronted UI would look like. It’s nonfunctional at the moment.
* [**Braille to Text: Mobile App Concept:**](#_pd3x8r4f065p) The proposed workflow for a community-driven mobile app.
* [**Dockerized Braille Converter Web Application:**](#_659d5qlz29vs)(Inprogress)A project for a Dockerized web application that provides offline Braille Grade 1 conversion. The project includes a backend server written in Python with Flask and a simple HTML/CSS/JS frontend.

**Part III: Sign language**

* [**SignGemma:**](#_q6xqnc553til) SignGemma is a new AI model from Google DeepMind. It translates American Sign Language (ASL) to English text in real-time by analyzing visual cues like hand movements and facial expressions. The model is being developed with community input and is designed to supplement, not replace, human interpreters.

**Part IV: Learning**

* [**Using Canvas to Generate a Quiz:**](#_l6wzjtgoe4mq) This section details how to use Gemini's Canvas feature to create a multiple-choice quiz. It provides step-by-step instructions for selecting the Canvas tool, attaching a PDF of the Canadian Charter of Rights and Freedoms, and using a specific prompt to generate a 10-question quiz with an answer key and hints. The guide also includes tips for adjusting the difficulty level and troubleshooting common issues.

**Appendix**

* [**Recommended Reading & Resources:**](#_5kdwxuxki0bz) A list of relevant external links and datasets.

# Project Overview

In a world increasingly reliant on digital information, accessibility is not just a feature—it's a necessity. **Echoes & Insights: A Gemini Guide for Accessibility** is a forward-thinking initiative designed to ensure that everyone, regardless of their ability, can access and engage with content.

Our project utilizes the advanced capabilities of **Google Gemini** to solve common accessibility challenges. We focus on producing high-quality, specialized content, such as:

* **Closed Captions:** Automatically generating precise and contextually aware closed captions for spoken audio and video, benefiting the Deaf and hard of hearing community.
* **Braille Transcription:** Converting digital text into standardized braille formats, offering a critical resource for individuals who are blind or have low vision.

By transforming digital content into these essential formats, we're not just creating tools—we're creating opportunities. "Echoes & Insights" is committed to building a more equitable digital landscape where information is truly accessible to all.

# Our Goals

1. **Goal 1: Maximize Accessibility and Usability.** Develop solutions that are intuitive and require no specialized technical knowledge. The process for generating braille or closed captions should be as simple as inputting text or audio and receiving the accessible output, making the tool usable by individuals, educators, and small organizations alike.
2. **Goal 2: Ensure Financial Accessibility.** The core project tools and services must be entirely free to use, leveraging existing, open-source platforms and APIs (such as Google Gemini's free-tier access) to eliminate cost barriers. The solutions should be designed to function with common, readily available devices, such as a standard computer or smartphone, without the need for expensive, proprietary hardware.
3. **Goal 3: Promote Universal Digital Inclusion.** Provide practical and effective resources that empower individuals with disabilities to independently access digital content. By generating high-quality braille and closed captions at no cost, the project will directly contribute to a more equitable digital landscape and reduce dependence on costly third-party services.
4. **Goal 4: Build a Self-Sustaining, Community-Driven Resource.** Create a scalable solution that can be improved upon and maintained by a community of users and developers. The project's documentation and code will be made publicly available, encouraging collaboration and ensuring that the tool remains a reliable, free resource for years to come.

# Specifications

Zero-Cost Accessibility: The application must be entirely free for all users. It will not require any paid subscriptions, in-app purchases, or specialized hardware. Users will only need a standard web browser on a common device (e.g., a smartphone, laptop, or desktop computer).

## **Closed Captioning Tool**

* **Input:** A file upload button that accepts common audio formats (e.g., MP3, WAV) and video formats (e.g., MP4).
* **Processing:** The application will use Gemini's existing audio transcription capabilities to generate a structured output that includes both the transcribed text and timestamps.
* **Output:** The final output will be a downloadable text file, formatted in the industry-standard **.srt** format, making it compatible with most video players and platforms.

## **Braille Generation Tool**

* **Input:** A simple text area for users to paste or type plain text.
* **Processing:** The application will leverage Gemini's core text-to-text functionality by prompting the model to convert the plain text into standard **Unicode Braille characters**.
* **Output:** The generated braille will be displayed in a non-editable text box, allowing users to easily copy the content for use with a braille display or embosser software.

# Warning on Copyright and Gemini-Generated Content

Using large language models (LLMs) like Gemini to create closed captions for videos can be a helpful tool. However, it's crucial to be aware of the potential for **copyright infringement**.

Gemini may have been trained on copyrighted material, and its output could contain text or creative works that are protected by copyright. This means that if you use Gemini to generate closed captions, the captions may unintentionally replicate copyrighted content.

If you then use or distribute a video with captions that contain copyrighted material, you could be violating the copyright holder's rights. This could expose you to legal risks, including lawsuits and financial penalties.

Gemini currently lacks a feature to filter or identify copyrighted content, so it is your responsibility to ensure that any captions you generate and use are free of copyrighted material. To avoid copyright issues, you should:

* **Review all generated content carefully:** Before using the captions, read them thoroughly to ensure they do not contain copyrighted text.
* **Obtain permission:** If you suspect the content may be copyrighted, you must obtain permission from the copyright holder before using it.
* **Generate original content:** For all videos you create, it is always best to use original text that you have written yourself.

# Using Gemini for Closed Captioning

This guide outlines how to use Gemini to create and translate .srt closed caption files. An .srt file is a simple text file that contains subtitles for a video. Think of it as a script for a movie or show, but with specific timing. It includes:

* **A number:** To keep the subtitles in order.
* **A timestamp:** To tell the video player when to show each line of text and when to make it disappear.
* **The subtitle text:** The actual dialogue or description that appears on the screen.

Because it's just a text file, it's very small and easy to share, and it works with most video players.

#### **Step 1: Create a Closed Caption File**

1. **Start a new chat:** Go to Gemini and start a new chat.
2. **Attach the video:** In the chat window, click the plus sign and attach the video file you want to create captions for.
   * **Note on file size:** Each video can be up to 2 GB, with a total video length of up to 5 minutes. Other supported file types can be up to 100 MB. To extend the total upload length to 1 hour, you can upgrade to Google AI Pro or Ultra.
3. **Enter the prompt:** In the chat box, enter the following prompt: "Using the provided video, create an .srt closed caption file."
4. **Wait for Gemini:** Allow Gemini to generate the captions.

#### **Step 2: Translate the Caption File (Optional)**

1. **Enter the translation prompt:** Once Gemini has generated the captions, enter the following prompt in the same chat box: "convert the .srt file output to [Language]." (Remember to replace [Language] with your desired language, such as "French," "Spanish," or "German.")

#### **Step 3: Save and Use the Caption File**

1. **Open a text editor:** Open a plain text editor on your computer (like Notepad on Windows or TextEdit on Mac). Do not use a word processor like Microsoft Word, as it may add formatting that can cause problems.
2. **Paste the captions:** Copy the output from Gemini and paste it into the new text file.
3. **Save the file:** Save the file, making sure the file name ends in .srt. For example, en.srt for English captions or fr.srt for French captions.
   * **Ensure UTF-8 encoding:** If available, save the file using "UTF-8" encoding. This helps prevent strange characters from appearing.
     + **On a Mac:** In TextEdit, go to Format > Make Plain Text. Then go to File > Save As, and select Unicode (UTF-8) from the Plain Text Encoding dropdown.
     + **On Windows:** In Notepad, go to File > Save As. In the Save As dialog box, select All Files from the "Save as type" dropdown, and then choose UTF-8 from the "Encoding" dropdown.

#### **Step 4: Add Captions to a Video in VLC**

1. **Download and Install VLC:** If you don't already have it, download and install VLC media player from [videolan.org](https://www.videolan.org/).
2. **Open your video:** Launch VLC and open the video file you want to watch.
3. **Add the subtitle file:** In the VLC menu bar, navigate to Subtitles > Add Subtitle File…
4. **Select the .srt file:** Browse to where you saved your .srt file, select it, and click Open. The subtitles should now appear at the bottom of the video.

#### **Troubleshooting & Limitations:**

* **Subtitles don't appear:** In VLC, go to the Subtitles menu and ensure that the "Subtitles Track" is set to "Track 1" or a similar option.
* **Strange characters:** If you see strange characters, it's likely an encoding issue. Try saving your .srt file again with UTF-8 encoding, following the steps above.

# Establishing a Baseline for Braille Accuracy

Creating a baseline to test the accuracy of Unified English Braille (UEB) output involves establishing a set of known, correct transcriptions to which new outputs can be compared. The process is similar to creating a benchmark in software development or machine learning. A simple and effective method is to select a diverse body of text, ensuring it includes a wide range of punctuation, capitalization, numbers, and contractions as defined in the official UEB rules. A human expert, or a trusted and verified transcription service, should then manually transcribe this text into UEB. This human-generated transcription serves as the "gold standard" or baseline. By running an automated transcription system on the same source text and then comparing its output, character by character, to the established baseline, you can measure its accuracy. This comparison can reveal error rates, identify specific types of errors (such as incorrect contractions or formatting), and provide a quantitative metric for evaluating the system's performance.

**Input:** Stephen Braille Test  
**Grade:** 2  
**Script version:** v6  
**Google Gemini 2.5 Pro with V6 script**

⠠⠎⠑⠏⠓⠒ ⠠⠁⠝⠉⠇⠊⠋⠋⠑ ⠠⠃⠗⠁ ⠠⠞⠑⠎

**Google Gemini 2.5 Flash with V6 script**

⠠⠎⠞⠑⠏⠓⠑⠝⠀⠠⠁⠝⠉⠇⠊⠋⠋⠑⠀⠠⠃⠗⠁⠊⠇⠇⠑⠀⠠⠞⠑⠎⠞  
  
⠠⠎⠞⠑⠏⠓⠑⠝ is a correct translation of **"Stephen"**.

⠠⠃⠗⠁⠊⠇⠇⠑ is a correct translation of **"Braille"**.

⠠⠞⠑⠎⠞ is a correct translation of **"Test"**.

**Input:** The 1st of July, 2024, was a very busy day for Sarah, with an amazing braille exhibition! The final score was 10.5 to 8.5.  
**Grade:** 1  
**Gemini:** Flash  
**Expected output:**   
⠠⠞⠓⠑ ⠼⠁⠎⠞ ⠕⠋ ⠠⠚⠥⠇⠽⠂ ⠼⠃⠚⠃⠙⠂ ⠺⠁⠎ ⠁ ⠧⠑⠗⠽ ⠃⠥⠎⠽ ⠙⠁⠽ ⠋⠕⠗ ⠠⠎⠁⠗⠁⠓⠂ ⠺⠊⠞⠓ ⠁⠝ ⠁⠍⠁⠵⠊⠝⠛ ⠃⠗⠁⠊⠇⠇⠑ ⠑⠭⠓⠊⠃⠊⠞⠊⠕⠝⠖ ⠠⠞⠓⠑ ⠋⠊⠝⠁⠇ ⠎⠉⠕⠗⠑ ⠺⠁⠎ ⠼⠁⠚⠲⠼⠑ ⠞⠕ ⠼⠓⠲⠼⠑⠲

**Input**: Tree  
**Grade:** 1  
**Expected output:** ⠠⠞⠗⠑⠑

**Input:** Water  
**Grade:** 1  
**Expected output:** ⠠⠺⠁⠞⠑⠗

**Input:** Baseline Test  
**Grade:** 1  
**Expected output:** ⠠⠃⠁⠎⠑⠇⠊⠝⠑ ⠠⠞⠑⠎⠞

**Input:** The quick brown fox jumps over the lazy dog 123.  
**Grade: 1**  
**Expected output:** ⠠⠞⠓⠑ ⠟⠥⠊⠉⠅ ⠃⠗⠕⠺⠝ ⠋⠕⠭ ⠚⠥⠍⠏⠎ ⠕⠧⠑⠗ ⠞⠓⠑ ⠇⠁⠵⠽ ⠙⠕⠛ ⠼⠁⠃⠉⠲

# Converting Text to Braille Unicode with Python

(Always In Progress. This type of work was not designed for a script. Gemini has limits on how many files can be attached and how it may use them. It’s more of a sample of braille character mappings and functions that Gemini can start with and build on at the moment. There are still several braille grade 2 contractions missing. Braille grade 2 is not easy)

*This section provides a guide on using a custom Python script to convert text into Braille Unicode.*

* ***About the script:*** *This script was created with Gemini's assistance and is designed to run within the Gemini chat interface. This approach is a unique workaround for users who may not have access to a local desktop environment or prefer not to use other external tools like Ollama.*
* ***A work in progress:*** *Please note that the script's functionality, particularly for Grade 2 contractions, is currently being refined. We are actively working on improving its accuracy.*
* ***Download the script:*** *The latest version of the Python script can be downloaded from*[this GitHub link](https://raw.githubusercontent.com/BranchingBad/Echoes-Insights/main/python-braille-convert.py?download=true). *Use CTRL+S (or CMD+S on a Mac) to open the save dialog.*

***Instructions for Use***

1. ***Start a new Gemini chat.***
2. ***Attach the script file:*** *Click the plus sign and attach the Python script you downloaded.*
3. ***Attach the text file:*** *Attach the text file you wish to convert to Braille.*
4. ***Enter the prompt:*** *"Using the attached script generate the raw Braille unicode output for the phrase '[your phrase here'. The output should be in Grade 2 Braille and should not include a breakdown of the translation."*

### ***Notes and Known Limitations***

* ***File limits:*** *Gemini allows a maximum of 10 file attachments per conversation.*
* ***Grade 2 Braille:*** *The script is still being improved and may not accurately handle all Grade 2 Braille contractions.*
* ***Community contributions:*** *We are actively seeking input and assistance in improving this script.*

# Converting Braille Unicode to text with Python

(In progress)

*This section provides a guide on using a custom Python script to Braille Unicode into text.*

* ***About the script:*** *This script was created with Gemini's assistance and is designed to run within the Gemini chat interface. This approach is a unique workaround for users who may not have access to a local desktop environment or prefer not to use other external tools like Ollama.*
* ***A work in progress:*** *Please note that the script's functionality, particularly for Grade 2 contractions, is currently being refined. We are actively working on improving its accuracy.*
* ***Download the script:*** *The latest version of the Python script can be downloaded from*[this GitHub link](https://raw.githubusercontent.com/BranchingBad/Echoes-Insights/main/python-braille-convert.py?download=true). *Use CTRL+S (or CMD+S on a Mac) to open the save dialog.*

***Instructions for Use***

1. ***Start a new Gemini chat.***
2. ***Attach the script file:*** *Click the plus sign and attach the Python script you downloaded.*
3. ***Enter the prompt:*** *"Using the attached script, convert the following to text"*
4. ***Paste:*** *Use shift enter twice and paste the braille unicode text below on the next line*

### ***Notes and Known Limitations***

* ***File limits:*** *Gemini allows a maximum of 10 file attachments per conversation.*
* ***Grade 2 Braille:*** *The script is still being improved and may not accurately handle all Grade 2 Braille contractions.*
* ***Community contributions:*** *We are actively seeking input and assistance in improving this script.*

# Creating Braille Unicode with Gemini

This guide shows you how to use a Python script to teach Gemini how to generate Braille Unicode from a text file attached to the conversation. The script allows for both single-phrase and multi-line file translations.

### **Step 1: Prepare the Script and Text**

1. **Start a new chat:** Open a new chat with Google Gemini.
2. **Attach the files:** Click the plus sign in the chat window to attach the necessary files:
   * **The Python script:** Download the script from[this GitHub link](https://raw.githubusercontent.com/BranchingBad/Echoes-Insights/main/python-braille-convert.py?download=true). Use CTRL+S (or CMD+S on a Mac) to open the save dialog.
   * **The text file (optional):** If you want to translate a multi-line text file, attach it as well. A sample file can be downloaded from[this GitHub link](https://raw.githubusercontent.com/BranchingBad/Echoes-Insights/main/python-braille-convert.py?download=true). Use CTRL+S (or CMD+S on a Mac) to open the save dialog.
   * **Tip for file names:** Gemini may not handle file names with spaces or periods well. It's best to use a dash (-) instead, or just rename your file to something simple like sample-input.txt.

### **Step 2: Generate Braille**

**To translate a single phrase:** Enter the following prompt into the chat box, replacing [your phrase here] with the text you want to translate:

"Using the attached script, generate the raw Braille unicode output for the phrase '[your phrase here]'. The output should be in Grade 2 Braille and should not include a breakdown of the translation."

**To translate a multi-line text file:** Enter the following prompt into the chat box, replacing sample-input.txt with the name of your attached text file:

"Using the attached script, generate the raw Braille unicode using the attached file sample-input.txt. The output should be in Grade 2 Braille and should not include a breakdown of the translation."

* **Note on limitations:** This method may fail with very large files. As a workaround, you can break up a large text file into several smaller ones.

### **Step 3: Save the Braille Output**

1. **Highlight and copy:** Highlight the Braille Unicode text that Gemini generates. You can do this by clicking and dragging your mouse over the text.
2. **Copy the text:** Right-click the highlighted text and select **Copy**, or use the keyboard shortcut: **Ctrl + C** on a PC or **Cmd + C** on a Mac.
3. **Paste and save:** Open a plain text editor (like Notepad or TextEdit), paste the copied text using **Ctrl + V** (PC) or **Cmd + V** (Mac), and then save the file.

# Voice Input and Braille Unicode Generation

This guide explains how to use Gemini's voice input feature with a Python script to convert spoken words into Braille Unicode.

### **Step 1: Prepare the Chat**

1. **Start a new chat:** Open a new chat with Google Gemini.
2. **Attach the script:** Click the microphone icon in the chat box.
3. **Attach the files:** Click the plus sign and attach the Python script that teaches Gemini to write Braille. The file can be downloaded from[this GitHub link](https://raw.githubusercontent.com/BranchingBad/Echoes-Insights/main/python-braille-convert.py?download=true).Use CTRL+S (or CMD+S on a Mac) to open the save dialog.

### **Step 2: Speak Your Prompt**

Once the script is uploaded, click the **microphone icon** again and say the following phrase:

"Using the attached Python script, transcribe what I say into Braille. I do not need a breakdown of the transcription. I just want the raw output."

You can also paste this phrase directly into the chat window.

### **Step 3: Speak Your Text and Get the Braille Output**

1. **Start speaking:** After entering the prompt, begin speaking the phrase you want to convert to Braille.
2. **Press Enter:** To send your message and get a reply, you must press **Enter** on your keyboard.
3. **Copy the output:** After Gemini generates the Braille, you can select and copy the text to a word processor or other file.

* **Note on limitations:** Currently, Gemini requires you to press **Enter** on your keyboard to submit the voice command. This prevents a fully hands-free experience, which can be a significant barrier for users with mobility impairments. This is a key area for future development.

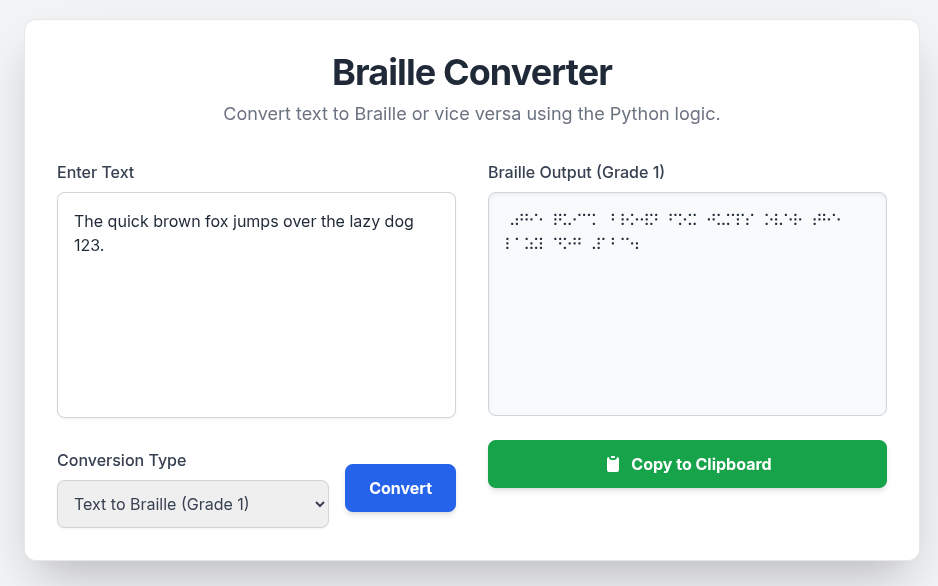
# Dockerized Braille Converter Web Application

Available [here](https://github.com/BranchingBad/Echoes-Insights). It has a few issues and needs a rewrite. It’s a work in progress at the moment. It's being released under [GPLv3](https://www.gnu.org/licenses/gpl-3.0.en.html). You are free to attach the files to a Gemini conversation and ask it how to run it. AI assisted trouble shooting is interesting. Get an error message, just copy it into the next conversation.

Please remember to click the “**Good Response**" option to [help Gemini](https://cloud.google.com/gemini/docs/discover/write-prompts#:~:text=With%20feedback%2C%20and%20your%20explicit,Gemini%20for%20Google%20Cloud%20feedback.) when receiving helpful information.

**Sample**

****

****

**Why Use a Docker Container?**

The use of containerization provides several key advantages for this application.

Portability is a major benefit, as the container encapsulates all dependencies, allowing the application to run seamlessly on virtually any operating system such as Windows, Mac, Linux and on various cloud platforms.

Containers also drastically improve development and deployment speed by providing a streamlined, efficient way to move the application from code to a running service. In a production environment, they offer resilience and scalability, as they can be quickly replaced and managed to handle increased demand or recover from a failure.

What are Containers?  
[*https://cloud.google.com/learn/what-are-containers*](https://cloud.google.com/learn/what-are-containers)

Containerization (computing)[*https://en.wikipedia.org/wiki/Containerization\_(computing)*](https://en.wikipedia.org/wiki/Containerization_(computing))

See it in action: Attach a file and ask Gemini for a detailed explanation of its functionality. Gemini can handle highly specific inquiries, for example:

*"Explain what happens on line 30 of the file I attached to the conversation."*

*"Walk me through the full process of how the Docker container starts and what actions it performs."*

**Making changes to the web application**

Spot a bug or have an idea for a new feature? That's where Gemini comes in, just ask it! You can ease into it by changing the background or button colors, or quickly adjust text, for example, changing 'Braille Converter' to a simple 'Converter'. Get ready for the exciting world of [vibe coding](https://en.wikipedia.org/wiki/Vibe_coding). A vital word of caution, though: **Gemini can make mistakes**. I highly recommend running this software in containers. This lets you experiment safely, isolated from your host OS. Mess up? No problem, just delete the container and start fresh! Remember to stop containers, remove them and rebuild them when making your own changes. I’ve wasted a few minutes here and there looking for a change I made and it was still running the older version. Make frequent back ups as well. Changes suggested by Gemini may not always work the first time.

**Debugging tips:** I wrote this app without really knowing braille grade 1. I used known braille transcriptions to help code it. I attached my project files and provided prompts like the following “The correct translation is ***X*** but I received the following output ***Y.*** Can you update the attached files with the corrections?” Gemini will analyze the attached files and make corrections. Again be careful as Gemini can make mistakes. Changes that it suggests are sometimes incorrect. This type of debugging is good for [microservices](https://en.wikipedia.org/wiki/Microservices) but is not as successful when dealing with a [monolithic service](https://en.wikipedia.org/wiki/Monolithic_application). There’s a woman who worked for Uber that wrote a really good work on microservices. It was really good. I can’t recall their name or the book title right now.

# Google Translate rough mockup with Braille support

# SignGemma

SignGemma is a new AI model from Google DeepMind that is designed to translate American Sign Language (ASL) into English text in real-time. It's a "multimodal" model, which means it can process both video and language, and it is trained to understand not just hand movements, but also facial expressions and body posture, which are critical parts of ASL grammar and nuance.

The model is part of Google's broader "Gemma" family of open-weight models, and its development has a strong focus on community collaboration, with input from developers and members of the Deaf community. It is designed to run on-device, like on smartphones or laptops, which helps preserve user privacy by processing video data locally.

SignGemma is intended to serve as a bridge for communication between Deaf and hearing individuals and is seen as a significant step toward making digital communication more inclusive. It is not meant to replace human interpreters but rather to supplement them in a variety of settings. While currently focused on ASL, the project's open-source nature suggests potential for future expansion to other sign languages.  
  
*Deaf News: Interview with Google's Sam Sepah on SignGemma, an upcoming ASL-to-text AI model*  
<https://www.youtube.com/watch?v=ZB4JoXbLwMo>  
  
*Google Invites Feedback for SignGemma, a New AI Sign Language Translation Model*  
<https://slator.com/google-invites-feedback-for-signgemma-a-new-ai-sign-language-translation-model/>

# Image-to-Braille: Current Limitations & Future Solutions

The current implementation of image-to-Braille conversion faces several significant challenges. These limitations are primarily due to the inherent complexities of visual data and the current state of AI technology.

* **Image Processing Difficulties:** The core of image-to-Braille is a process known as Optical Braille Recognition (OBR). This is more challenging than traditional Optical Character Recognition (OCR) because it must accurately identify and interpret the raised dots on a surface, rather than continuous strokes. Factors like poor image quality, glare, shadows, and the need to process images from different angles (e.g., from a phone camera vs. a flatbed scanner) can lead to inaccuracies.
* **AI Hallucination:** A key challenge is the potential for AI models to "hallucinate" or generate incorrect information. This can happen when the model encounters low-quality or ambiguous images, leading it to output Braille characters that do not exist in the source image. This problem is particularly relevant for images of complex content, such as STEM (science, technology, engineering, and mathematics) diagrams or charts, which are not well-represented in existing training data.
* **Multilingual Support:** Most current OBR systems are not designed for multilingual conversion. Each language has its own unique Braille system, and a single model trained on one language may not be able to accurately translate another. This is a critical area for future development to ensure the tool is globally useful.
* **Lack of Context:** An AI model may be able to transcribe the Braille dots it "sees," but it cannot interpret the broader context of the image. For instance, it can't distinguish between a decorative image and one that contains vital information.

### **Recommendations for Future Development**

To address these limitations, consider the following improvements in future versions of the project:

* **Data Quality and Curation:** To mitigate AI hallucination and improve accuracy, a primary focus should be on building a high-quality, diverse dataset of Braille images. This dataset should include images from different angles, lighting conditions, and languages, and it should specifically contain examples of complex diagrams and charts.
* **Enhanced Image Pre-processing:** Implementing more robust image processing algorithms can help to clean up source images before they are passed to the AI model. This could include functions for correcting image skew, enhancing contrast, and removing shadows or glare.
* **Iterative Human-in-the-Loop Model:** Given the current limitations, the most reliable approach is to implement a human review process. This "human-in-the-loop" system would allow users to review and correct the AI's Braille transcription, and this corrected data could then be used to further train and refine the model.
* **Modular Architecture:** The system could be designed with a modular architecture that allows for the integration of specialized models for different tasks, such as a model specifically for STEM Braille, and another for recognizing different languages.

# Braille to Text: Mobile App Concept

This document outlines the high-level steps and technical considerations for building a mobile application that translates a sheet of Braille into plain text using a mobile device's camera and a large language model with vision capabilities, such as the Gemini API.

**1. Image Capture from a Mobile Device**

This initial step is about getting a clear, high-quality image of the Braille text. The quality of this image is critical for the entire process.

* **User Interface (UI):** Provide a camera view with clear on-screen guidance. A bounding box or an outline could help the user frame the Braille sheet correctly.
* **Camera Configuration:** The app should automatically adjust for optimal settings.
  + **Focus:** Implement auto-focus on the Braille dots.
  + **Lighting:** Detect poor lighting conditions and suggest a brighter environment or enable the flash.
  + **Image Stabilization:** Use the device's built-in stabilization to prevent blur from camera shake.
* **Permissions:** Request and verify camera access permissions from the user.

**2. Local Image Pre-processing**

Once the image is captured, it should be processed on the device to optimize it for transfer and analysis by the model.

* **Cropping & Alignment:** Automatically detect the Braille sheet within the image and crop any unnecessary background. Apply a perspective correction algorithm to align the image so the Braille is straight and not at an angle.
* **Grayscale Conversion:** Convert the image from color to grayscale. This can reduce file size and simplify the image for the model, as Braille dot information is based on contrast, not color.
* **Compression:** Compress the image to a suitable format (e.g., JPEG with a lower quality setting) and a manageable file size for efficient network transfer. This is a critical step, especially for users on mobile data networks.
* **Image Format:** The final image should be a Base64-encoded string for inclusion in the API request payload.

**3. Transfer to Gemini API**

The prepared image is now ready to be sent to the Gemini API for analysis.

* **API Endpoint:** Make a POST request to a Gemini API endpoint capable of multimodal understanding. The gemini-2.5-flash-preview-05-20 model is a suitable choice for this task.
* **Payload Construction:** The request payload will contain the user's prompt (e.g., "Translate the Braille in this image into English text.") and the Base64-encoded image data.
* **Loading State:** Implement a clear loading indicator on the UI to inform the user that the image is being processed.
* **Error Handling:** Include robust error handling for network failures or API-side errors, providing a user-friendly message if the request fails.

**4. Gemini Model Processing & Translation**

This is the core of the concept, where the AI performs the heavy lifting. The app's logic ends here, but it's important to understand what the model does.

* **Image Analysis:** The Gemini model analyzes the image, recognizing the Braille dots and their arrangement.
* **Braille Decoding:** The model understands the six-dot Braille system and decodes the patterns into individual Braille characters.
* **Text Translation:** The model translates the decoded Braille characters into the corresponding text in the desired language (e.g., English, Spanish, etc.). It can also handle variations like Grade 1 (letter-for-letter) and Grade 2 (with contractions) Braille.

**5. Receive and Process Results**

Once the Gemini API completes its analysis, the mobile app receives the response.

* **JSON Response:** The API will return a JSON object containing the translated text.
* **Data Extraction:** The app must parse the JSON response and extract the translated text from the designated field.

**6. Display Results & User Interaction**

The final step is to present the translated text to the user in an accessible and helpful way.

* **UI Display:** Show the translated text in a clean, legible format on the screen.
* **Accessibility Features:**
  + **Text-to-Speech (TTS):** Add a button to read the translated text aloud for visually impaired users.
  + **Copy to Clipboard:** Provide a quick way to copy the translated text.
  + **Font Size Adjustment:** Allow the user to adjust the font size for better readability.
* **Feedback:** The UI should transition from the loading state back to a result view, providing clear confirmation that the translation is complete.

**Existing github projects which are working on braille to text conversion.**

**AEyeAlliance** (Canadian effort AI4SocialGood Lab 2018 at McGill University)  
*wanchunsu Wan-Chun Su  
tingyisu Ting-Yi Su  
HelenGezahegn Helen Gezahegn  
marin0829*  
<https://github.com/HelenGezahegn/aeye-alliance>

**AaditT/braille**  
*Aadit Trivedi*  
<https://github.com/AaditT/braille>

**Braille to Text Translator**  
*Bárbara Côrtes e Souza (@barbaracortes)  
Lucas Fernandes Turci (@lucasturci)*  
<https://github.com/lucasturci/BrailleTextTranslator>

**InSight**  
*SatyamSoni23*  
<https://github.com/SatyamSoni23/InSight>

# Using Canvas to Generate a Quiz

To start using **Canvas** in **Gemini**, click the **Tools** icon in the conversation window. For this example we will be using the Canadian Charter of Rights and Freedoms. You can download the PDF and attach the file in the URL listed below for example.

**English**   
[*https://www.canada.ca/content/dam/pch/documents/services/download-order-charter-bill/canadian-charter-rights-freedoms-eng.pdf*](https://www.canada.ca/content/dam/pch/documents/services/download-order-charter-bill/canadian-charter-rights-freedoms-eng.pdf)

**French**  
[*https://www.canada.ca/content/dam/pch/documents/services/download-order-charter-bill/charte-canadienne-droits-libertes-fra.pdf*](https://www.canada.ca/content/dam/pch/documents/services/download-order-charter-bill/charte-canadienne-droits-libertes-fra.pdf)

*\*This quiz is based on content from the Canadian Charter of Rights and Freedoms, which is subject to Crown copyright*

From there, select **Canvas**. Once **Canvas** is open, you can use either the English or French the prompt below based on which PDF you downloaded and attached.  
  
*English prompt:*

**“Create a 10-question exam with an answer key based on the 'canadian-charter-rights-freedoms-eng.pdf' document. The questions should be at a difficulty level of 7, requiring advanced comprehension and analysis of the text. Each question must have four multiple-choice options and a hint.”**

Google Gemini [Gem](https://support.google.com/gemini/answer/15236405?hl=en) for English (Inprogress. This avoids having to download and attach a PDF file which may be blocked due to security reasons. I also like how Google implemented prompt sharing in their eco-system. It’s a unique feature. Google also does an excellent job of scanning attachments used for harmful content (Virus, malware). Very good solution for an office place and easy to deploy.)

[*https://gemini.google.com/gem/1XMZgjdDH7yeaMUodMm-SDdRZalnjYK3i?usp=sharing*](https://gemini.google.com/gem/1XMZgjdDH7yeaMUodMm-SDdRZalnjYK3i?usp=sharing)

*Invite en français:*  
**“Créez un examen de 10 questions avec un corrigé basé sur le document « charte-canadienne-droits-libertes-fra.pdf' ». Les questions doivent avoir un niveau de difficulté de 7 et exiger une compréhension et une analyse approfondies du texte. Chaque question doit comporter quatre choix multiples et un indice.”**

Google Gemini [Gem](https://support.google.com/gemini/answer/15236405?hl=fr) pour le français (en cours de développement. Cela évite de télécharger et de joindre un fichier PDF, qui pourrait être bloqué pour des raisons de sécurité. J'apprécie également la façon dont Google a intégré le partage rapide dans son écosystème. C'est une fonctionnalité unique. Google effectue également un excellent travail d'analyse des pièces jointes pour détecter les contenus nuisibles (virus, logiciels malveillants). Excellente solution pour un bureau, facile à déployer.)

[*https://gemini.google.com/gem/1hmpl26cSlJVXxcOb7q0U-1Th3nMk7dOH?usp=sharing*](https://gemini.google.com/gem/1hmpl26cSlJVXxcOb7q0U-1Th3nMk7dOH?usp=sharing)

*English focus on section example:*

**“Create a 10-question exam with an answer key based on the 'canadian-charter-rights-freedoms-eng.pdf' document. The questions should be at a difficulty level of 7, requiring advanced comprehension and analysis of the text. Each question must have four multiple-choice options and a hint. Focus on section 15.(1) of the document.”**

Canvas sample level 7  
[*https://g.co/gemini/share/fb325546dc73*](https://g.co/gemini/share/fb325546dc73)Canvas sample level 10[*https://g.co/gemini/share/76c5c72a41f4*](https://g.co/gemini/share/76c5c72a41f4)

*Exemple de section de concentration en français :*

**“Créez un examen de 10 questions avec un corrigé basé sur le document « charte-canadienne-droits-libertes-fra.pdf ». Les questions doivent avoir un niveau de difficulté de 7 et exiger une compréhension et une analyse approfondies du texte. Chaque question doit comporter quatre choix multiples et un indice. Concentrez-vous sur la section 15.(1) du document.”**

Exemple de toile niveau 7  
[*https://g.co/gemini/share/25984683ca1a*](https://g.co/gemini/share/25984683ca1a)

Échantillon de toile niveau 10[*https://g.co/gemini/share/357698c3432f*](https://g.co/gemini/share/357698c3432f)

**Tips for Generating a Quiz**

* To achieve the best results, begin with a difficulty level of 7. Gemini may need time to process the material and might refuse a higher difficulty (9 or 10) on the first try.
* To gradually increase the difficulty, start by using the prompt "**generate another quiz.**" After running this prompt several times, switch to "**generate a level 10 quiz.**" You don't need to complete every quiz; simply wait for each one to be fully generated before entering the prompt again. This repeated generation process helps Gemini build its knowledge of the document and conduct further research.
* For now, try to stick with a **maximum of 10** questions to avoid formatting issues. Gemini also doesn't yet support a mix of multiple-choice and true/false questions.
* Gemini does not track the number of quizzes you've generated, so you'll need to keep count yourself.
* If you're happy with a quiz, click the **Good Response** icon to give Gemini helpful feedback.
* You can use a prompt like **"show your work for question 1"** to see how Gemini arrived at an answer, which can be particularly useful for more difficult questions (level 9 or 10) where you might disagree with the result.
* If you get an answer wrong, you can ask Gemini to explain the topic and generate 10 more questions to test your understanding.

**Future ideas**

* Export graded results to Google Sheets to easily track student progress. An AI assistant can help you format the data in Google Sheets to create detailed grading reports.
* How do I add the following to a quiz:

1. True or false questions
2. A chart, bar or pie graph
3. Multiple choice question based on the picture shown *(which signage is correct?)*
4. Multimedia video on Youtube *(Watch the video and choose the right multiple choice option. Include YouTube URL when creating a Gem. Disable Gem if YouTube video is modified for security to prevent replacing existing YouTube videos with offensive content, advertising, unauthorized content swap.)*
5. Written exam questions *(The results and AI grading on this one would probably be disputed frequently.)*

# Recommended Reading & Resources

**Prompt Engineering and Effective AI Communication**

* **Prompting Guide:** [*Gemini for Google Workspace Prompting Guide 101*](https://services.google.com/fh/files/misc/gemini-for-google-workspace-prompting-guide-101.pdf) This guide provides foundational skills for writing effective prompts, including using natural language, being clear and concise, and providing context. It also covers best practices like breaking down complex tasks and provides examples for various roles like project managers and marketing specialists.
* **The Next AI Career:** [*Mark Cuban's Perspective on AI*](https://www.instagram.com/reel/DNqnYI1o1kq/) Mark Cuban suggests that the next big career in AI isn't coding, but rather knowing how to effectively use AI tools. This emphasizes the importance of skills like prompt engineering and understanding how to apply AI to solve real-world problems.
* **Google Cloud Startup technical guide AI agents:** Download it [here](https://cloud.google.com/resources/content/building-ai-agents). Are you ready to build your AI agent? The Startup technical guide: AI agents, provides the knowledge needed to go from an idea to prototype to scale, whether your goals are to automate tasks, enhance creativity, or launch entirely new user experiences for your startup.
* [**Fortune article on how China is now offering AI to 6-year-olds:**](https://fortune.com/2025/05/08/microsoft-airbnb-uber-ceos-demand-computer-science-ai-school-classes-like-china-to-unlock-billions/) Learning to use artificial intelligence is no longer optional for today's students—it's essential. Children who don't learn these skills will be competing in a future job market against peers who are already experts.
* [**Generative AI in Google Workspace Privacy Hub**](https://support.google.com/a/answer/15706919): The document "Generative AI in Google Workspace Privacy Hub" explains Google's privacy and data handling practices for its generative AI features, including Gemini in Workspace Apps. It emphasizes that Google maintains its existing privacy commitments, keeping user data and interactions within their organization. The article details that user content is not used to train AI models without explicit permission, and provides a summary of data practices, security controls, and compliance with regulations like HIPAA and GDPR. It also covers features like Google grounding and user/admin controls.

#### **Accessibility and Alternative Communication**

* **How to read and write Braille:** [*How to read and write Braille?*](https://www.youtube.com/watch?v=SS65wL4FphE) This video explains the Braille script, which uses a system of six dots. It demonstrates how to write using a Braille slate and stylus, and explains that writing is done from right to left, while reading is done from left to right. It also goes over the dot combinations for each letter of the English alphabet.

## [Gemini app access](https://support.google.com/gemini/answer/16275805?hl=en)

|  | **Gemini**  ***without a Google AI plan*** | **Gemini app in Google AI Pro** | **Gemini app in Google AI Ultra** |
| --- | --- | --- | --- |
| [**Gemini Models**](https://support.google.com/gemini/answer/16275805?hl=en#model_limits) | | | |
| 2.5 Pro | Up to 5 prompts / day | Up to 100 prompts / day | Up to 500 prompts / day |
| 2.5 Flash | General access | | |
| [**Context Window**](https://support.google.com/gemini/answer/16275805?hl=en#context_window) | | | |
| Context size | 32 thousand | 1 million | |
| [**Features**](https://support.google.com/gemini/answer/16275805?hl=en#features) | | | |
| [Audio Overviews](https://support.google.com/gemini/answer/16047373) | Up to 20 audio overviews / day | | |
| [Deep Research](https://support.google.com/gemini/answer/15719111) | Up to 5 reports / month using 2.5 Flash | Up to 20 reports / day using 2.5 Pro | Up to 200 reports / day using 2.5 Pro |
| [Deep Think](https://support.google.com/gemini/answer/16345172) | - | - | Up to 10 prompts / day with a 192 thousand token context window |
| [Image generation & editing](https://support.google.com/gemini/answer/14286560) | Up to 30 images / day | Up to 1,000 images / day | |
| [Scheduled actions](https://support.google.com/gemini/answer/16316416) | - | Up to 10 active actions at a time | |
| [Video generation](https://support.google.com/gemini/answer/16126339) | - | Up to 3 videos / day using Veo 3 Fast (preview) | Up to 5 videos / day using Veo 3 (preview) |
| [Additional features\*](https://support.google.com/gemini/answer/16275805?hl=en#features) | Usage limits subject to the selected model | | |
| Early access to features | - | Priority access to some new features | |

\*Tip: There are more features that are not listed above, like Canvas, Gems, and more. These features are generally available to most users. [Learn more in the Help Center](https://support.google.com/gemini#topic=15280100).

Why did I choose Gemini instead of other AI models? Good integration with existing products. Works with **YouTube, Google Docs, Gmail, Google Maps** and other products in the Google eco system I frequently use.

⠞⠓⠑ ⠕⠗⠊⠛⠊⠝⠁⠇ ⠠⠞⠠⠧ ⠎⠓⠕⠺⠂ ⠠⠥⠝⠎⠕⠇⠧⠑⠙ ⠠⠍⠽⠎⠞⠑⠗⠊⠑⠎⠂ ⠺⠓⠊⠉⠓ ⠗⠁⠝ ⠋⠗⠕⠍ ⠼⠁⠊⠓⠛ ⠞⠕ ⠼⠃⠚⠚⠃⠂ ⠊⠎ ⠉⠗⠑⠙⠊⠞⠑⠙ ⠺⠊⠞⠓ ⠓⠑⠇⠏⠊⠝⠛ ⠞⠕ ⠎⠕⠇⠧⠑ ⠍⠕⠗⠑ ⠞⠓⠁⠝ ⠼⠃⠋⠚ ⠕⠋ ⠞⠓⠑ ⠕⠧⠑⠗ ⠼⠁⠂⠉⠚⠚ ⠍⠽⠎⠞⠑⠗⠊⠑⠎ ⠊⠞ ⠏⠗⠕⠋⠊⠇⠑⠙⠲

**Throwing this in just because I think it’s a really heart felt idea on what you can do with Gemini photo blending.**  
<https://www.reddit.com/r/GeminiAI/comments/1nkgr9t/pov_instead_of_a_celebrity_you_edited_yourself/>

:~$ ollama run gemma3:4b

>>> tell me about yourself

Okay, let's talk about me! I'm Gemma, a large language model created by the Gemma team at Google DeepMind. I am an open-weights model, which means I’m widely available for public use.

**Input:**

:~$ systemctl status ollama

● ollama.service - Ollama Service

Loaded: loaded (/etc/systemd/system/ollama.service; enabled; preset: disabled)

Drop-In: /usr/lib/systemd/system/service.d

└─10-timeout-abort.conf

Active: active (running) since Fri 2025-09-26 10:28:24 EDT; 37min ago

Invocation: 3289082be53948b5b3a1377719c2dca4

Main PID: 34551 (ollama)

Tasks: 18 (limit: 18602)

Memory: 4.5G (peak: 9.1G)

CPU: 17min 51.894s

CGroup: /system.slice/ollama.service

└─34551 /usr/local/bin/ollama serve

Sep 26 10:49:48 slate ollama[34551]: time=2025-09-26T10:49:48.185-04:00 level=INFO source=server.go:1289 msg="llama runner started in 2.03 seconds"

Sep 26 10:49:48 slate ollama[34551]: [GIN] 2025/09/26 - 10:49:48 | 200 | 2.536915677s | 127.0.0.1 | POST "/api/generate"

Sep 26 10:50:33 slate ollama[34551]: [GIN] 2025/09/26 - 10:50:33 | 200 | 29.289452996s | 127.0.0.1 | POST "/api/chat"

Sep 26 10:51:20 slate ollama[34551]: [GIN] 2025/09/26 - 10:51:20 | 200 | 36.367720936s | 127.0.0.1 | POST "/api/chat"

Sep 26 10:54:27 slate ollama[34551]: [GIN] 2025/09/26 - 10:54:27 | 200 | 2m52s | 127.0.0.1 | POST "/api/chat"

Sep 26 10:54:37 slate ollama[34551]: [GIN] 2025/09/26 - 10:54:37 | 200 | 48.267µs | 127.0.0.1 | HEAD "/"

Sep 26 10:54:37 slate ollama[34551]: [GIN] 2025/09/26 - 10:54:37 | 200 | 165.648226ms | 127.0.0.1 | POST "/api/show"

Sep 26 10:54:37 slate ollama[34551]: [GIN] 2025/09/26 - 10:54:37 | 200 | 222.913828ms | 127.0.0.1 | POST "/api/generate"

Sep 26 10:56:26 slate ollama[34551]: [GIN] 2025/09/26 - 10:56:26 | 200 | 1m34s | 127.0.0.1 | POST "/api/chat"

Sep 26 10:58:16 slate ollama[34551]: [GIN] 2025/09/26 - 10:58:16 | 200 | 1m4s | 127.0.0.1 | POST "/api/chat"

**Output:**

It looks like you're running the **Ollama** service, which is a tool for running large language models locally. The output you provided gives a detailed status of the service, its resource usage, and a log of recent interactions.

Here is an analysis of the key information in the systemctl status ollama output:

## **Ollama Service Status Analysis ⚙️**

| Field | Value | Interpretation |
| --- | --- | --- |
| **Active** | active (running) | The Ollama service is currently running successfully. |
| **Uptime** | 37min ago | The service started 37 minutes ago (at 10:28:24 EDT). |
| **Main PID** | 34551 | The process ID of the main Ollama server process. |
| **Memory** | 4.5G (peak: 9.1G) | The service is currently using **4.5 Gigabytes of RAM**. It peaked at 9.1 GB, likely when a large model was initially loaded or during a particularly long generation. This high usage is typical for running large language models. |
| **CPU** | 17min 51.894s | This is the total CPU time consumed by the service since it started. It indicates the process has been actively working (performing inference or loading models) for a significant portion of its uptime. |
| **Process** | ollama serve | The service is running the Ollama server process, which manages the models and handles API requests. |

Export to Sheets

## **Recent API Log Analysis 💬**

The log entries show the **Ollama service handling various local API requests** from the IP address 127.0.0.1 (localhost).

| API Endpoint | Request Time | Latency (Generation Time) | Type |
| --- | --- | --- | --- |
| /api/generate | 10:49:48 | **2.54s** | Used for a single-turn prompt or non-chat generation. |
| /api/chat | 10:50:33 | **29.29s** | A lengthy response generated by the chat endpoint. |
| /api/chat | 10:51:20 | **36.37s** | Another long chat response, suggesting a complex or long-form answer was requested. |
| /api/chat | 10:54:27 | **2m 52s** | A very long generation time, indicating a *very extensive* chat response or a multi-turn conversation that led to a large output. |
| /api/show | 10:54:37 | **165.65ms** | Likely a quick request to check the details or status of a specific model. |
| /api/generate | 10:54:37 | **222.91ms** | A very fast generation, possibly for a short, simple prompt. |
| /api/chat | 10:56:26 | **1m 34s** | Another long chat generation. |
| /api/chat | 10:58:16 | **1m 4s** | Another long chat generation. |

Export to Sheets

### **Summary of Activity**

The log indicates that the Ollama server is **running and actively serving a number of large language model generation and chat requests** over its local API. The highly variable and sometimes *very long* latency values (ranging from a fraction of a second up to almost three minutes) are typical, as the generation time for an LLM depends heavily on the complexity of the prompt, the length of the requested response, and the hardware resources available.