Build a Transcription Bot for Google Meet using Recall.ai

You're building a notetaking app for Google Meet. You need transcripts that capture who's speaking, what they're saying and when. Google Meet's captions won't cut it, since they often miss details and lack speaker names. There are two ways that work. One's messy and requires a team of five developers to maintain forever. The other's clean and requires a few API calls. Let's break it down.

Option 1: Build a Bot to Capture Captions

You can code a bot to join a Google Meet and automatically grab its captions. This would be a good case for smaller use cases, but you will encounter challenges when trying to scale the bot. Google often makes changes to its applications which can break your code. Captions can incorrectly transcribe words with accents or background noise. You'll possibly deal with browser logins, errors, and Google's terms of service, which might lead to account restrictions. This bot is a proof-of-concept, not a long-term solution.

Option 2: Use Recall.ai for Transcriptions

Now for the cleaner way. You can use Recall.ai's API to send a bot to your Google Meet calls and pull transcriptions. It will capture who's speaking, what they're saying, and the exact timestamp. You can set this up with a few API calls. No need for a custom bot, saving you the maintenance headaches.

We made a demo app to show you how to use Recall.ai's API to deploy a bot, join a Google Meet, and get real-time transcriptions.

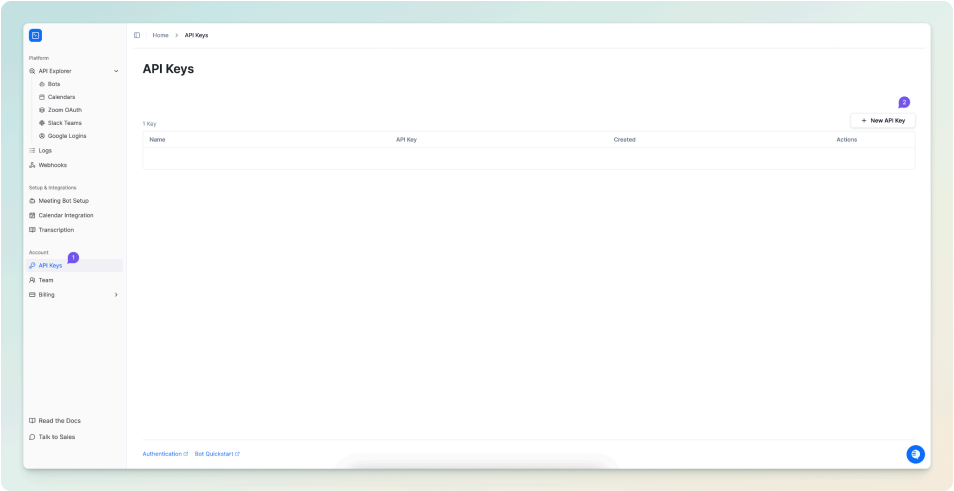
Let's walk through the steps to get it running.

Creating the Recall.ai Transcription Bot Project

Sign up at Recall.ai and create an API key

Once you are in.

1. Select **API Keys**.
2. Create a **New API Key**.



Set up a project with a **frontend** and **backend** folder

```
mkdir recall-bot-demo
cd recall-bot-demo
mkdir frontend
mkdir backend
```

Let’s build the **frontend** using **React**

```
cd frontend
npx create-react-app .
npm install socket.io-client axios
```

Below is a basic React **App.js** component to do the following

- Allow users to input a Google Meet URL and deploy the bot.
- Display real-time transcripts with speaker labels and timestamps.

Copy the code below and **paste** it into your **frontend/App.js** file

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import io from 'socket.io-client';
import './App.css';

const socket = io('http://localhost:3001');

function App() {
  const [meetingUrl, setMeetingUrl] = useState('');
  const [transcripts, setTranscripts] = useState([]);
  const [status, setStatus] = useState('');

  useEffect(() => {
    socket.on('transcript', (transcript) => {
      setTranscripts((prev) => [...prev, transcript]);
    });
    return () => socket.off('transcript');
  }, []);

  const handleDeployBot = async () => {
    if (!meetingUrl) {
      setStatus('Please enter a valid Google Meet URL');
      return;
    }
    setStatus('Deploying bot...');
    try {
      const response = await axios.post('http://localhost:3001/deploy-bot', {
        meeting_url: meetingUrl,
      });
      setStatus(`Bot deployed with ID: ${response.data.bot_id}`);
    } catch (error) {
      setStatus('Error deploying bot');
      console.error(error);
    }
  };
```

```
 return (
  <div className="App">
   <h1>Recall.ai Google Meet Transcription Demo</h1>
   <div>
    <input
     type="text"
     placeholder="Enter Google Meet URL"
     value={meetingUrl}
     onChange={(e) => setMeetingUrl(e.target.value)}
    />
    <button onClick={handleDeployBot}>Deploy Bot</button>
   </div>
   <p>{status}</p>
   <h2>Live Transcript</h2>
   <div className="transcript-container">
    {transcripts.map((t, index) => (
     <div key={index} className="transcript">
      <span>[{t.timestamp.toFixed(2)}s] {t.speaker}: </span>
      <span>{t.text}</span>
     </div>
    ))}
   </div>
  </div>
 );
}
export default App;
```

Let's build the **backend**

```
cd ../backend
touch index.js
npm init -y
```

npm init -y creates **package.json**

Install dependencies

```
npm install express axios dotenv socket.io
```

- express: For creating the server.
- axios: For making API requests to Recall.ai.
- dotenv: For managing environment variables.
- socket.io: For real-time communication with the frontend.

Set up the backend environment variables (**.env**)

- Create a **.env** file in /backend

```
echo -e "RECALL_API_KEY=YOUR_TOKEN_HERE\nWEBHOOK_URL=NGROK_URL\nPORT=BACKEND_PORT" > .env
```

- Your **backend/.env** file should look like this:

```
RECALL_API_KEY=YOUR_TOKEN_HERE
WEBHOOK_URL=NGROK_URL
PORT=BACKEND_PORT
```

- Replace **YOUR_TOKEN_HERE** with your actual Recall.ai API key
- Replace **NGROK_URL** with your ngrok server URL (e.g., https://1234-56-789-01-234.ngrok-free.app – more below)
- Replace **BACKEND_PORT** (e.g,. 3001 – more below)

Create the server (**backend/index.js**)

Below is a basic server setup to do the following

- Handle a POST request to send a bot to a Google Meet call.
- Receive webhooks from Recall.ai with transcription data.
- Broadcast transcripts to the frontend via Socket.IO.

Copy the code below and **paste** it into your **backend/index.js** file

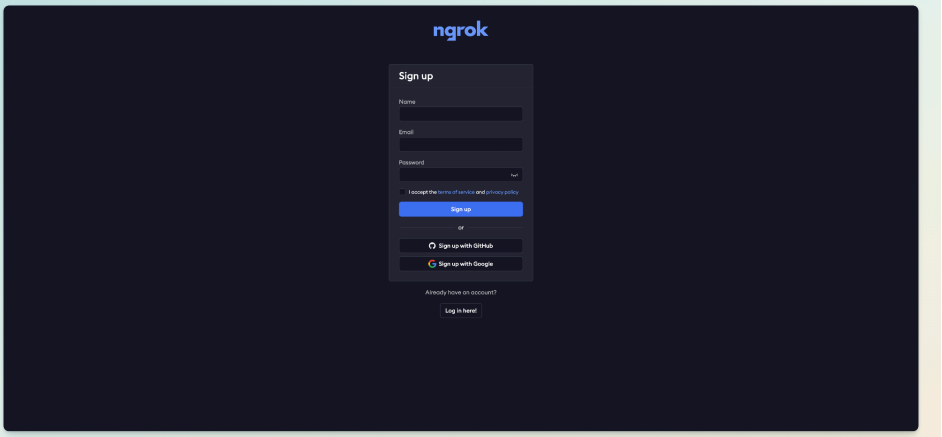
```
require("dotenv").config();
```

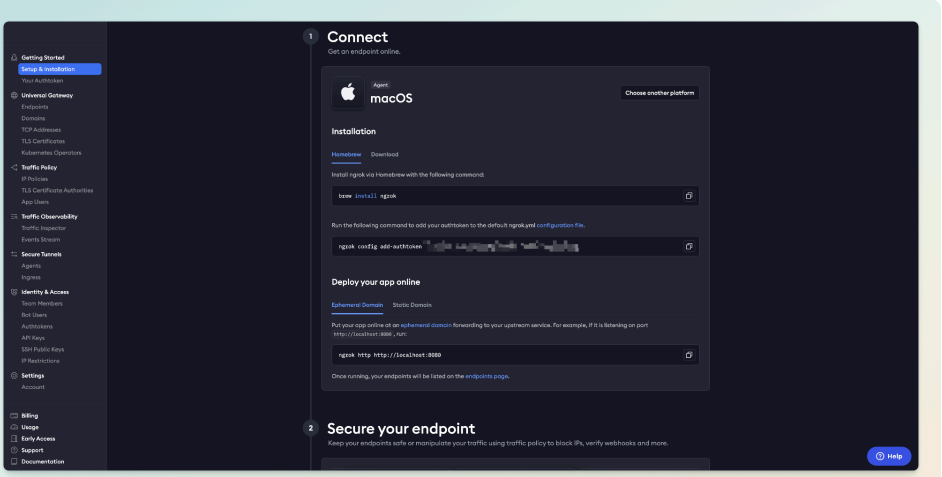
```
const express = require("express");
const axios = require("axios");
const { Server } = require("socket.io");

const cors = require("cors");
const app = express();
const server = require("http").createServer(app);
const io = new Server(server, {
  cors: { origin: "http://localhost:3000" },
});

app.use(express.json());
app.use(cors());

const recall = axios.create({
  baseURL: "https://us-west-2.recall.ai/api/v1",
  headers: {
    Authorization: `Token ${process.env.RECALL_API_KEY}`,
    "Content-Type": "application/json",
  },
});

app.post("/deploy-bot", async (req, res) => {
  const { meeting_url } = req.body;
  if (!meeting_url) {
    return res.status(400).json({ error: "Meeting URL is required" });
  }
  try {
    const response = await recall.post("/bot", {
      meeting_url,
      bot_name: "Bot",
      recording_config: {
        transcript: { provider: { meeting_captions: {} } },
        realtime_endpoints: [
          {
            type: "webhook",
            url: `${process.env.WEBHOOK_URL}/webhook/transcription`,
            events: ["transcript.data"],
```

```
        },
       ],
     },
    });
    res.json({ bot_id: response.data.id });
  } catch (error) {
    res.status(500).json({ error: "Failed to deploy bot" });
  }
});

app.post("/webhook/transcription", (req, res) => {
  const transcriptData = req.body.data?.data || {};
  if (!transcriptData.words || !Array.isArray(transcriptData.words)) {
    return res.status(200).json({});
  }
  const transcript = {
    speaker: transcriptData.participant?.name || "Unknown",
    text: transcriptData.words.map((w) => w.text).join(" "),
    timestamp: transcriptData.words[0].start_timestamp?.relative || 0,
  };
  io.emit("transcript", transcript);
  res.status(200).json({});
});

server.listen(process.env.PORT || 3001, () => {
  console.log("Server running on port 3001");
});
```

Sign up to get a **ngrok** account

- Go to ngrok and create an account
- Install ngrok following the instructions





Configure ngrok to expose the **backend**

```
cd backend
ngrok http 3001
```

Your **ngrok session** should look like this

```
Session Status                online
```

```
Account                       Your Name (Plan: Free)
Version                       3.22.1
Region                        United States (State) (us-state-1)
Latency                       22ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://1234-56-789-01-234.ngrok-free.app -> http://localhost:3001
```

- Replace **NGROK_URL** with your ngrok server URL `https://1234-56-789-01-234.ngrok-free.app`

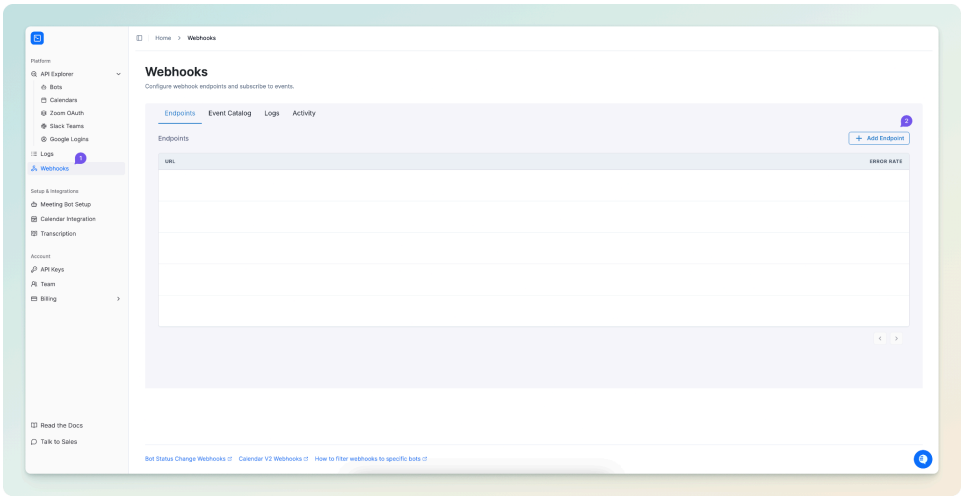
```
WEBHOOK_URL=https://1234-56-789-01-234.ngrok-free.app
```

Create a webhook in Recall.ai

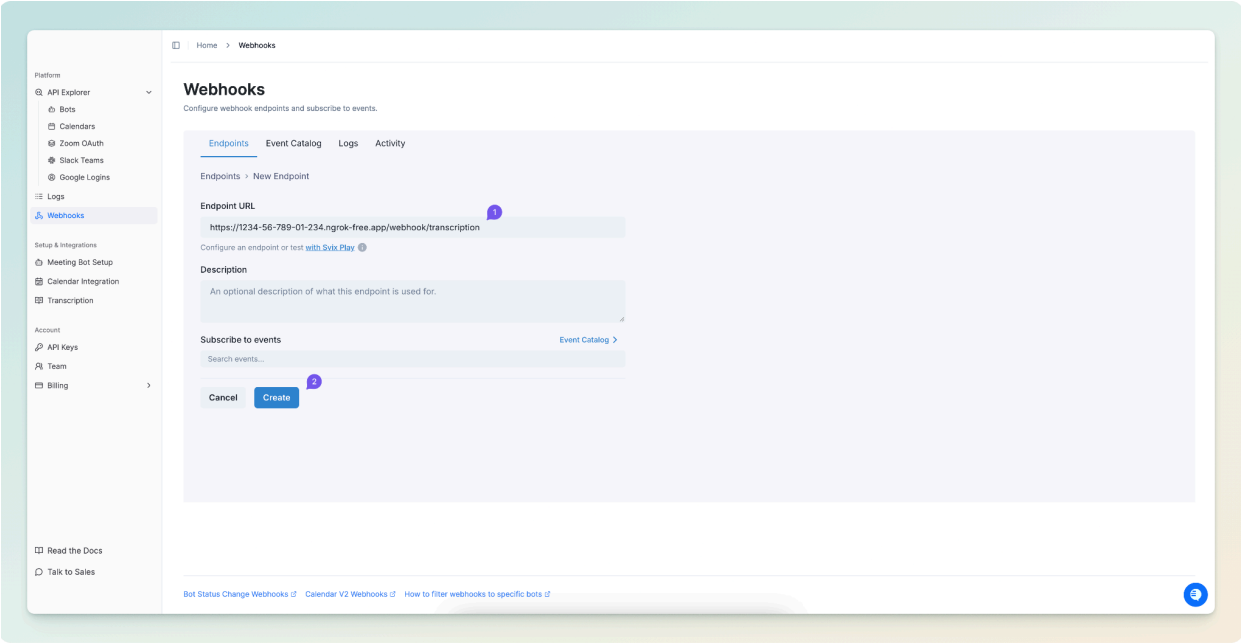
Once you are in.

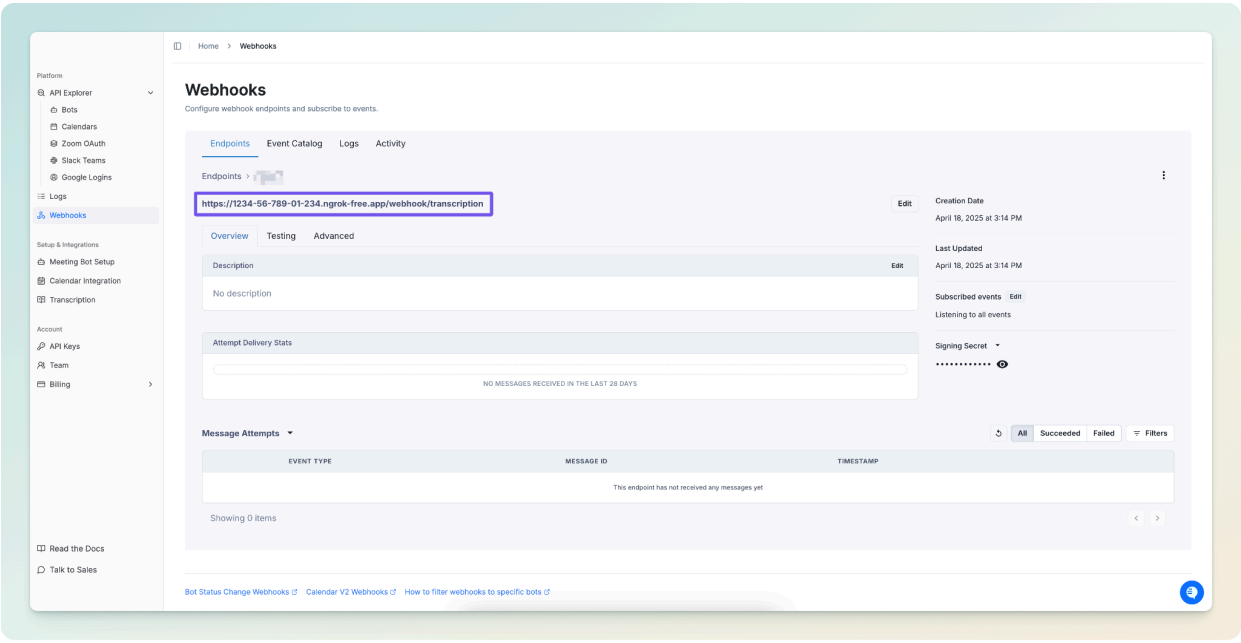
1. Select **Webhooks**.
2. **Add a new endpoint.**

This will be used to listen for transcript events.



1. Use your **WEBHOOK_URL** to **create the Webhook Endpoint URL** (e.g., https://1234-56-789-01-234.ngrok-free.app/webhook/transcription).
2. Click on **Create**.





By now, your **backend/.env** file should have the RECALL_API_KEY, WEBHOOK_URL, and PORT values like this example:

```
RECALL_API_KEY=YOUR_API_KEY
WEBHOOK_URL=https://1234-56-789-01-234.ngrok-free.app
PORT=3001
```

Launch the **backend**

```
cd backend
node index.js
```

Launch the **frontend**

```
cd ../frontend
npm start
```

Ensure **ngrok** is running

Deploying Recall.ai's transcription bot

1. Start a Google Meet, stay in the meeting, and copy its URL
2. Go to your local React server (e.g., http://localhost:3000)
3. Enter a Google Meet URL (e.g., https://meet.google.com/abc-defg-hij)
4. Click the "Deploy Bot" button
5. Let the bot into the Google Meet when it shows up
6. Talk for a little bit (*make sure to turn on your microphone*)
7. Watch live transcripts roll live in the app

Conclusion

And that's a wrap! You've just built a transcription bot for Google Meet using Recall.ai that grabs real-time transcripts with speaker tags and timestamps. It's fast, reliable, and simple to use. Now your users can focus on the meeting, not note-taking.

You can find the full code of this tutorial on Github where you just need to insert your own Recall.ai API key and other environment variables.

If you enjoyed the tutorial, you can learn more in the Recall.ai blog.

Until next time, and happy coding!

Richard Chan is a **Content Engineer** @ **Recall.ai**, building tools to make work easier.

More Resources

- GitHub Demo Repository
- Walkthrough Video
- Recall.ai Blog
- Recall.ai Documentation
- Recall.ai YouTube