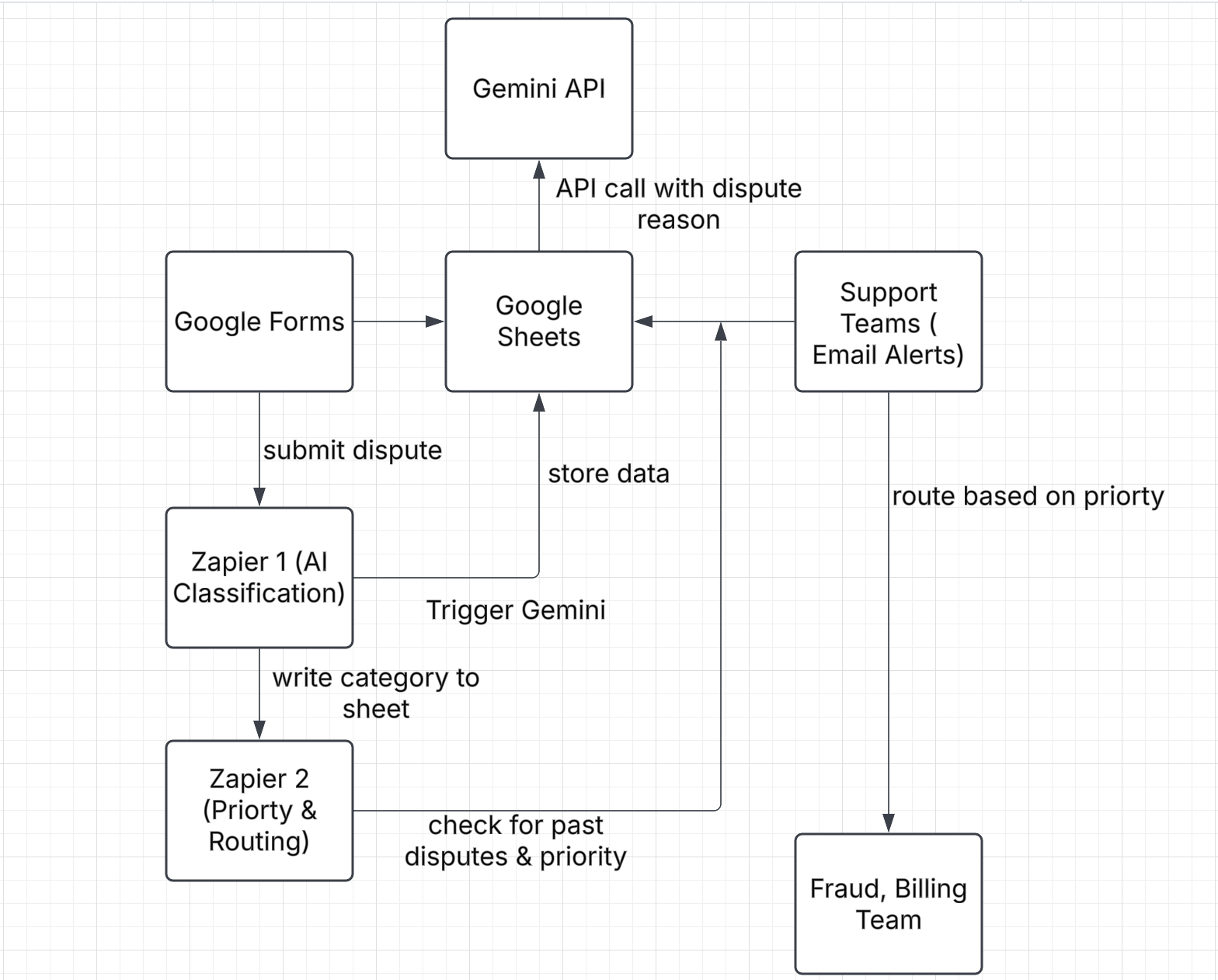
**Task 1: Rapidly Automating a Banking Process ( Workflow and API Integration)**

**System Architecture Diagram:**

****

Initially I started exploring how I can implement this process using normal tech stack and writing code from scratch and then I used the resources that Zeta has provided in the implementation process. I explored different tools like Glide, Zapier and Thunkable and for this task we just have to provide the functionality and design the workflow not much concentration on Design part and Glide has added some limitations as we can make API calls, connecting to the external DBs/ Excel sheets in the free version.

So I decided to use the traditional Google forms and google sheets where I am taking dispute input from the user using google forms and the form by default stores in the google sheets which is acting like a storage piece in this case and I used form for user input and sheets for storage purposes. I used the zapier free version for automating the workflow purpose of making API calls and updating the dispute sheet based on the API response.

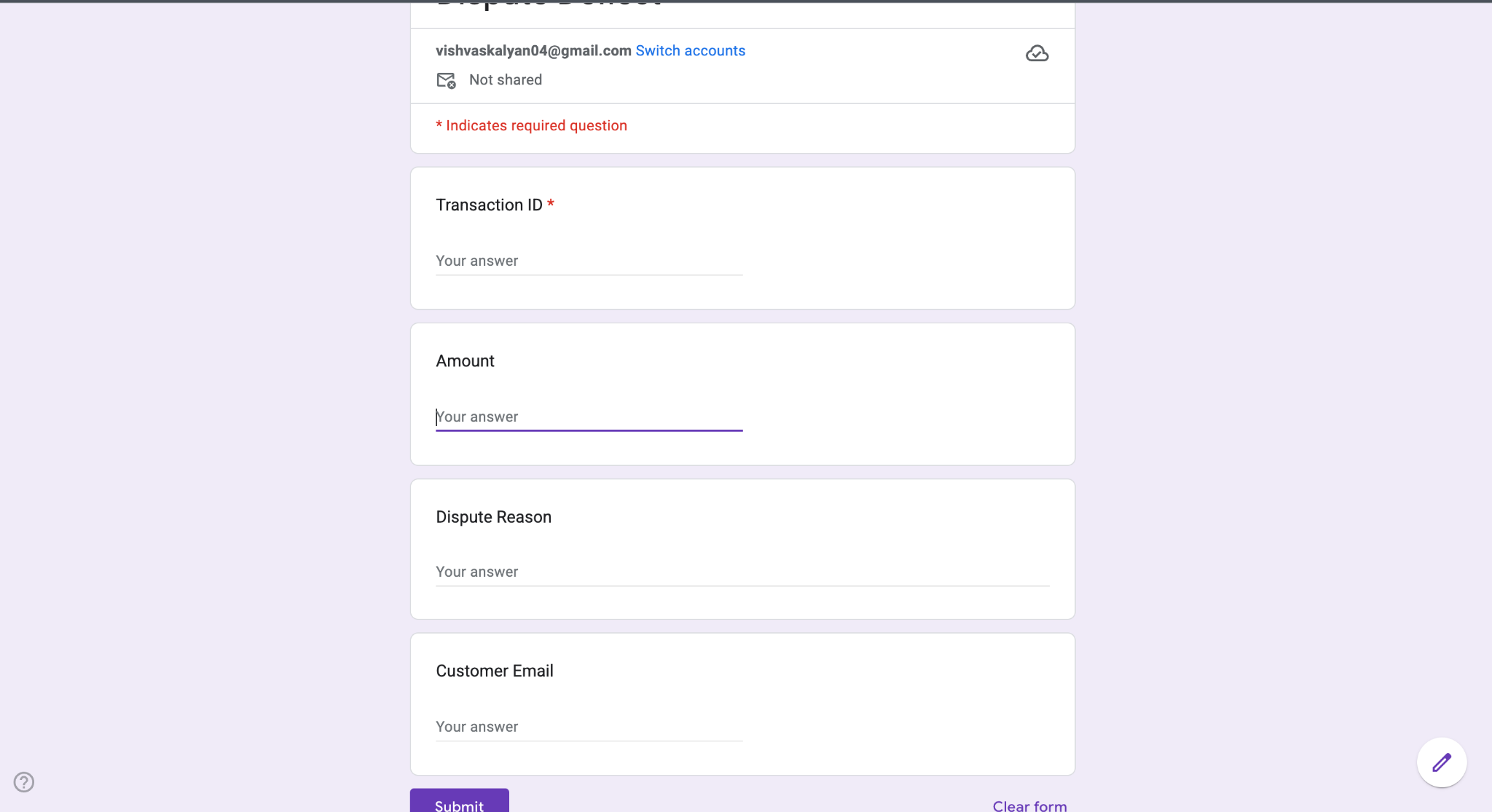
Let us deep dive into each of the step by step solution:

**Key Components:**

**User Interface (Dispute Submission)**

Google Form: Customers submit disputes by providing Transaction ID, Dispute Amount, Dispute Reason, and Email.

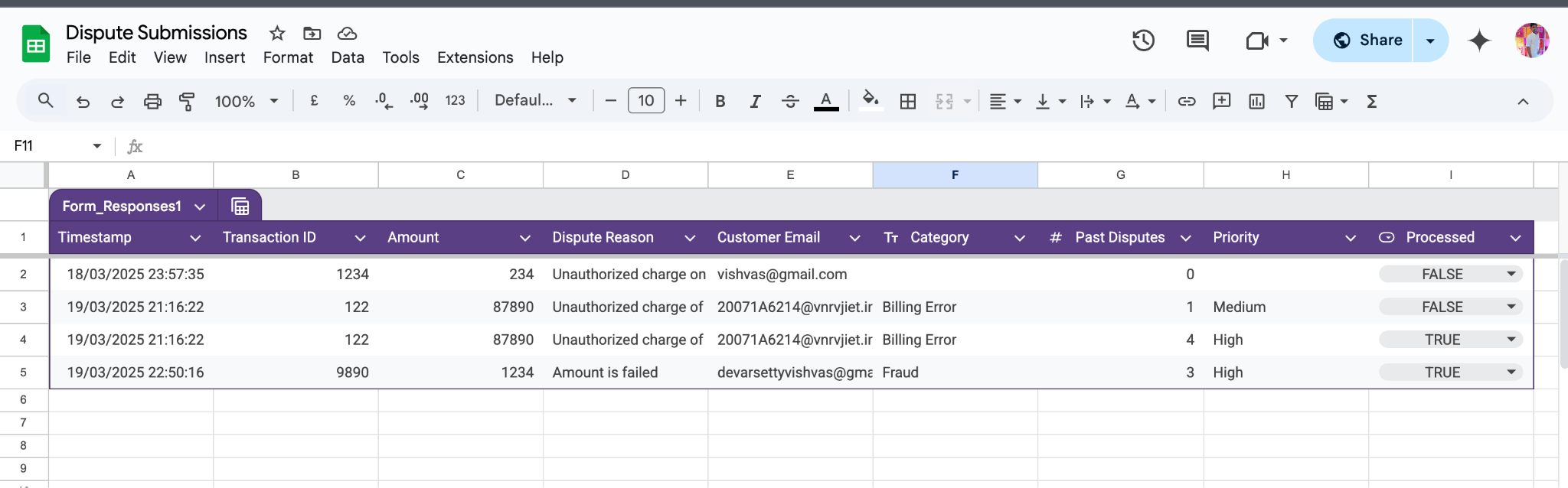
[Dispute Submission Form](https://docs.google.com/forms/d/e/1FAIpQLSf4-37J7YSN136p-X5YwutEB9rQpC2d0Wpj7JhxaKVq9xyAhw/viewform)



**Centralised Database**

Google Sheet: Automatically stores dispute submissions.

[Dispute Submissions](https://docs.google.com/spreadsheets/d/13O1NbVVGVVj19GmU8NpKjQGYbD4DvSgW6n-P6jQsllo/edit?usp=sharing)



I added extra columns of Category which is auto populated with responses generated by the AI ( Fraud, Billing Error etc.. ) and also Past disputes is used to track the count of the disputes that a user faced so far and accordingly it identifies priority. Processed is a status flag for workflow tracking.

**Workflow Automation & AI Integration:**

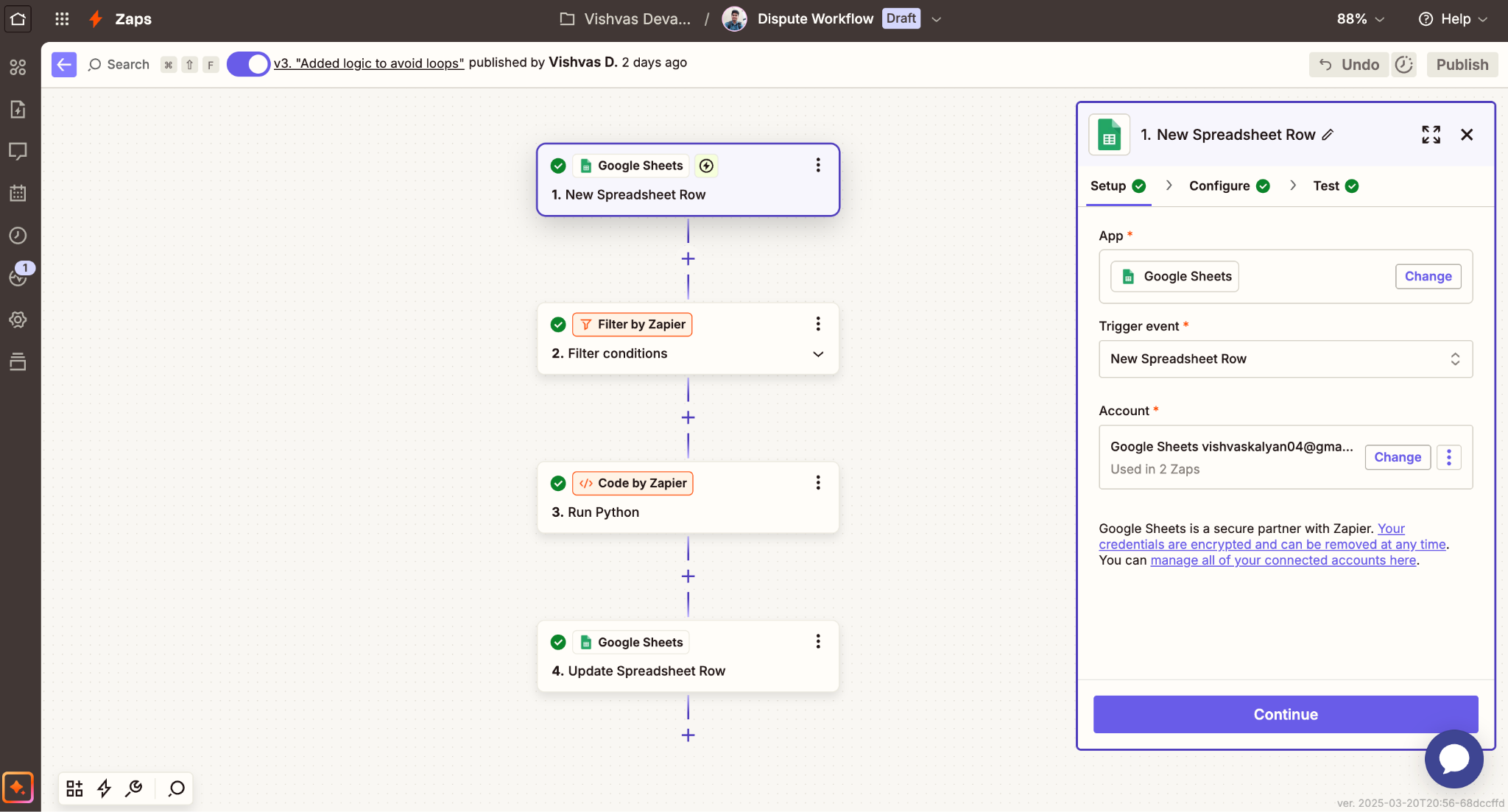
**Zapier**: Orchestrates two automated workflows ("Zaps") to handle:

AI Classification of disputes.

Priority-Based Routing to resolution teams.

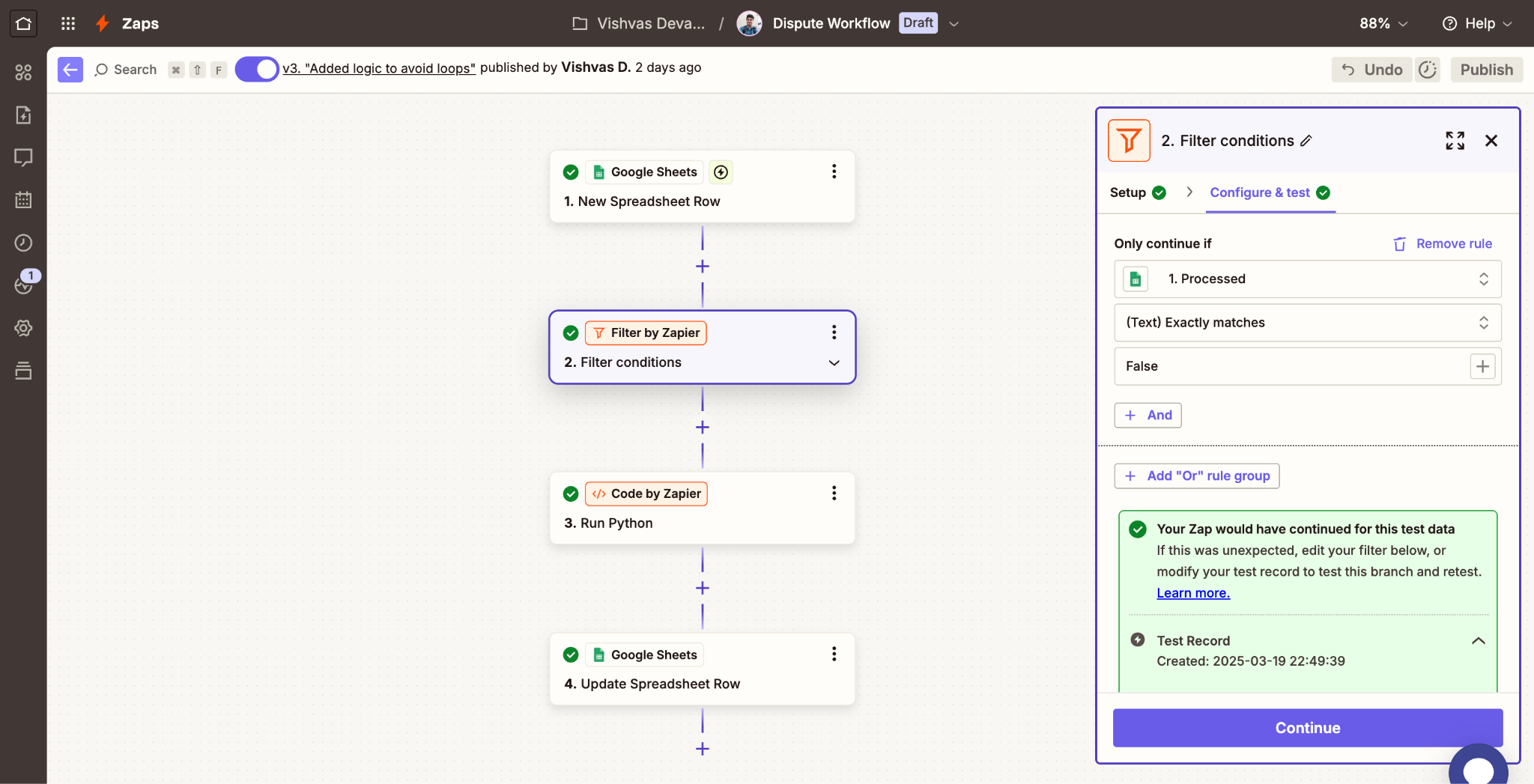
Step 1: AI-Driven Dispute Classification:

<https://zapier.com/editor/286933614/draft/286933614/setup>



**Trigger** : New row added to Google Sheets (i.e., new dispute submission).

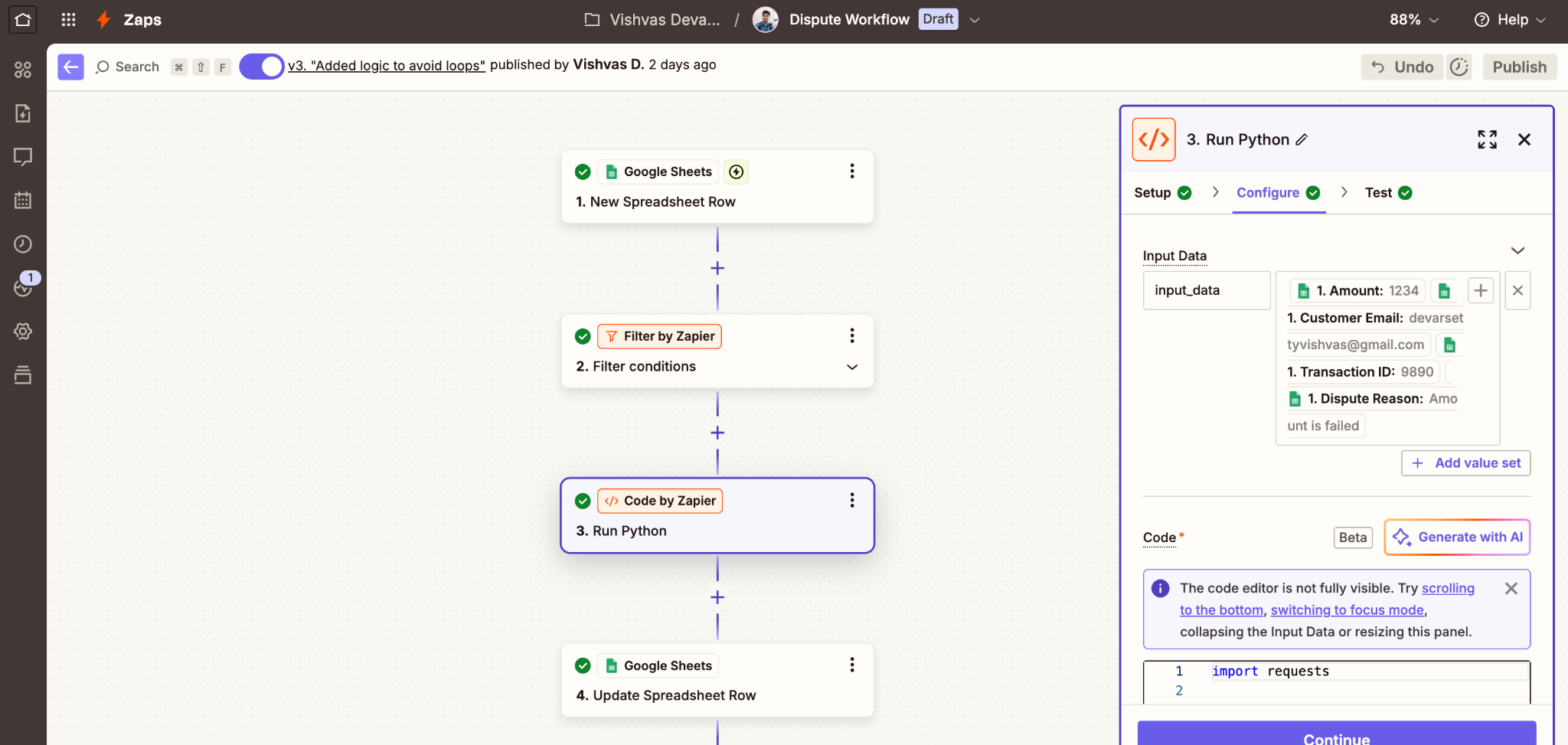
**Filter**: Checks if Processed = FALSE to ensure only new disputes proceed.



**Action**:

**Code by Zapie**r executes a Python script to call the Gemini API.

**AI Prompt**: Classifies disputes into one category: Fraud, Billing Error, Service Complaint, or Technical Error.



Code Snippet:

import requests

API\_KEY = "AIzaSyAMnaZ7-bD6COWD5oe5Fwtmwvl-9TVOVB0"

dispute\_reason = input\_data.get('Dispute\_Reason')

url = f"https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-pro-latest:generateContent?key={API\_KEY}"

prompt = {

"contents": [{

"role": "user",

"parts": [{

"text": f"Classify this dispute into exactly one category: [Fraud, Billing Error, Service Complaint, Technical Error]. Only respond with the category name. Response format: {{category}} Dispute: {dispute\_reason}"

}]

}]

}

response = requests.post(url, json=prompt)

if response.status\_code != 200:

return {"error": f"API Error: {response.text}"}

try:

category = response.json()['candidates'][0]['content']['parts'][0]['text']

return {'category': category.strip()}

except KeyError:

return {"error": "Unexpected response format"}

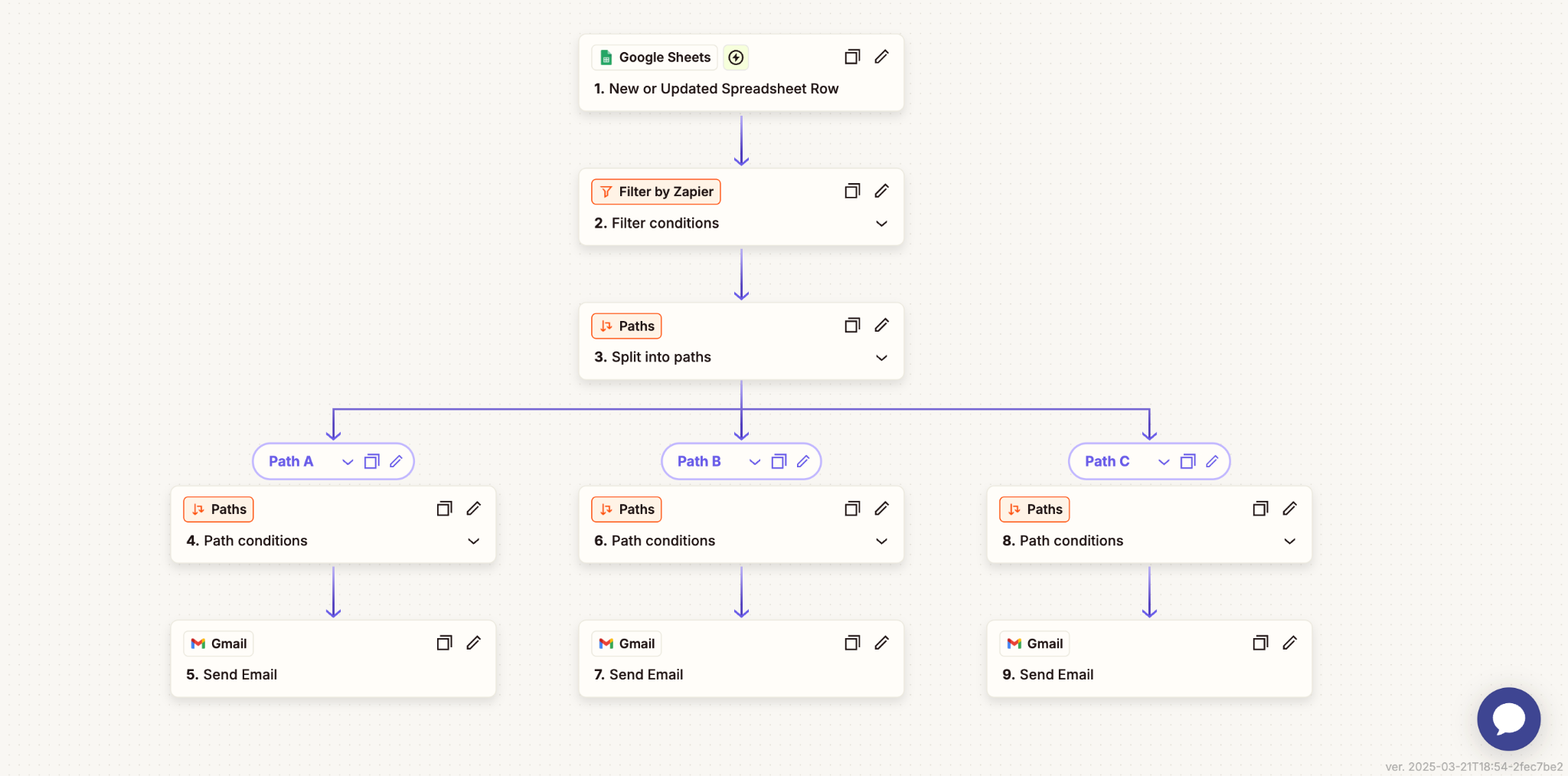
This is the code snippet of the API call and the last step it updates the database with respective categories.

**Step 2**: Priority-Based Escalation

<https://zapier.com/editor/287142599/published>

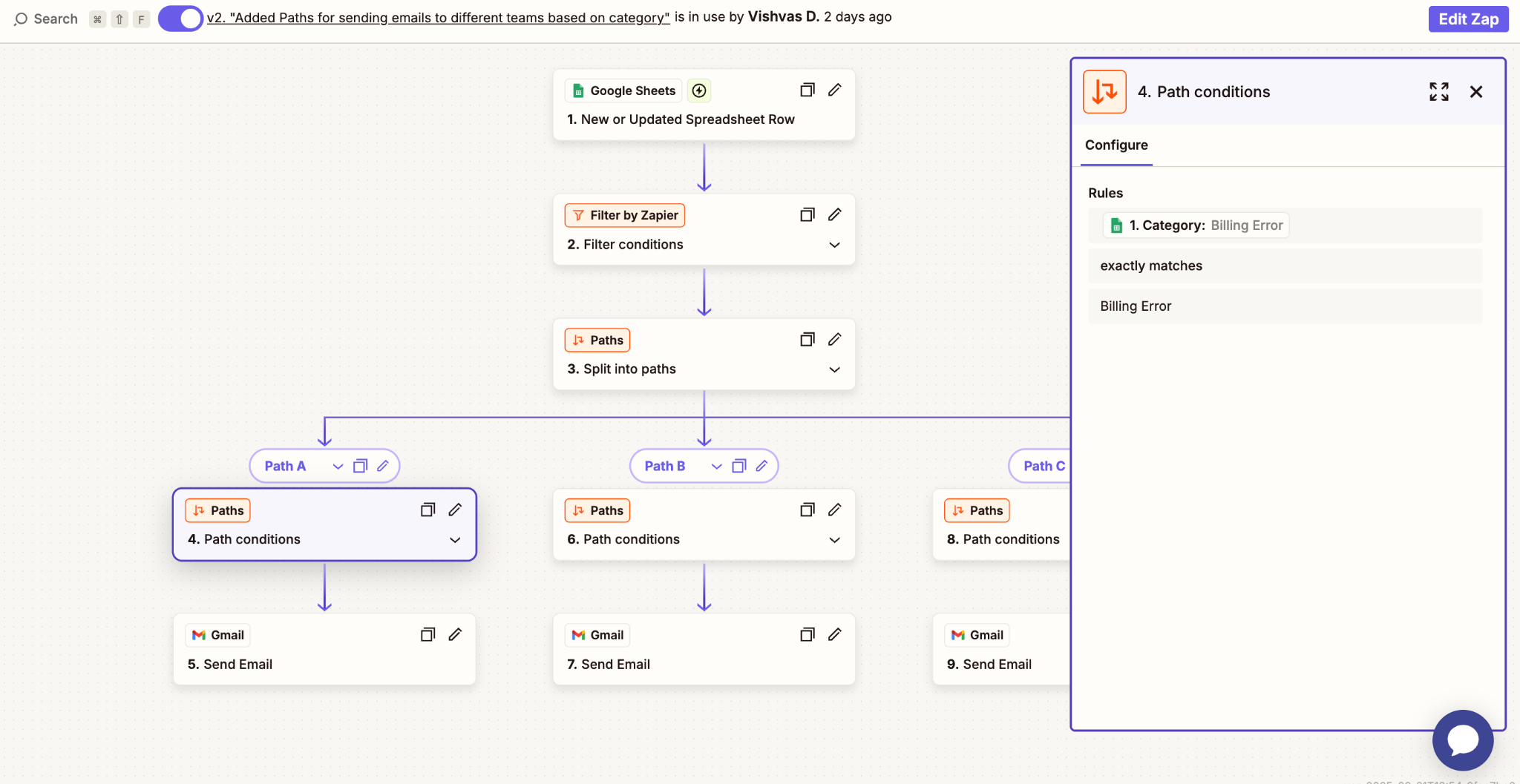
**Trigger**: Google Sheets row update (specifically when Category is populated).

**Filter**: Routes disputes marked as High Priority (determined by Past Disputes count).



**Action**:

Split Path: Sends email alerts to respective teams (e.g., Fraud Team, Billing Team) based on the dispute category.



**The reason I didn't choose to build an API where it interacts with Gemini is because Zapier is already providing the feature where we can write simple api request calls and return the response to us in JSON format and we can use it wherever we want.**

**Why Zapier?**

**Minimal Code Compliance:**

Zapier’s "Code by Zapier" step allowed Python scripting for Gemini API integration without deploying a standalone server.

**End-to-End Automation:** Kept the entire workflow (form → AI → routing) within one platform.

**Tool Trade-offs:**

**Pros**: No server costs, rapid prototyping, Zapier handles retries/error logging.

**Cons**: Zapier’s free tier has usage limits (2000 tasks/month).

**Scalability**

I want Replace Gemini with a custom ML model trained on historical dispute data.

Migrate to a microservices architecture (e.g., FastAPI + React) for enterprise-scale traffic.

I want to replace it with an attractive UI using Glide and connect it with Airtable Database which achieves more flexibility.

**I didn't use thunkable because I just want to show more on functionality rather than UI and use simple tools for this task.**