

# BCT Internship Projects

Landon Zheng

# What did I do? (Sequentially)

01

## Data Migration

Moving API Data to an AWS S3 Bucket with Python

02

## Flowise

What is Flowise, and my experience

03

## Project 1

AI-Powered Employee Leave Database & Uploader

04

## Project 2

AI-Powered Invoice Processor & Uploader

+++



# Data Migration

Moving API Data to an AWS S3  
Bucket with Python

+++

# Migration Process



Retrieving ID Token

Obtain GPS Data

Upload Data to S3  
Bucket

# Retrieving ID Token

```
Intern > ⚡ tokenRetrieve.py > ⚡ get_id_token
1  import requests
2  import re
3
4 def get_id_token(api_key):
5     post_login_url = f"{{REDACTED}}/token"
6
7     headers = {'x-api-key': api_key}
8     data = {
9         "username": "{{REDACTED}}",
10        "password": "{{REDACTED}}"
11    }
12
13    response = requests.post(post_login_url, headers=headers, json=data, verify=True)
14
15    if response.status_code == 200:
16        json_response = response.json()
17        location = json_response.get("location", "")
18        print(location)
19        code_match = re.search(r'code=(\w+)', location)
20
21        if code_match:
22            code_value = code_match.group(1)
23            print(f"Code: {code_value}")
24        else:
25            print("Code not found in the location URL.")
26            return None
27
28        token_url = f"{{REDACTED}}/token"
29        body = {
30            "grant_type": "authorization_code",
31            "client_id": "{{REDACTED}}",
32            "code": code_value,
33            "redirect_uri": "{{REDACTED}}"
34        }
35
36        token_response = requests.post(token_url, data=body)
37        print("Token", token_response)
38
39        if token_response.status_code == 200:
40            json_response = token_response.json()
41            id_token = json_response.get("id_token", "")
42            return id_token
43        else:
44            print("Error:", token_response.json())
45            return None
46
47        print("Error:", response.text)
48        return None
```

- Rest Post API
- Authentication: Username & Password and API Key
- Using `requests.post()` for posting & response json
- Two-Step Process: Retrieve the code, then use the code for the actual ID Token.

# Obtaining GPS Data

```
intern > ♫ gpsData_to_s3.py > ⌂ get_id_token
...
63 def get_gps(id_token, start_id, end_id):
64
65     temp_url = "https://api.example.com/gps?<span style="background-color: #007bff; color: white; padding: 2px 5px; border-radius: 3px;">id={id}</span>"  

66     headers = {'Authorization': f'Bearer {id_token}'}
67     results = []
68
69     temp_folder = "temp" #"/Users/landonzheng/Documents/Intern/temp"
70     if not os.path.exists(temp_folder):
71         os.makedirs(temp_folder)
72
73     for id in range(start_id, end_id + 1):
74         url = temp_url.replace('{:id}', f'{id}')
75         response = requests.get(url, headers=headers, verify=False)
76
77         print(response.json())
78         print(f"ID: {id}, HTTP Status Code: {response.status_code}")
79
80         if response.status_code == 200:
81             json_reponse = response.json()
82             results.append(json_reponse)
83
84             file_name = f"{id}-gps.json"
85             file_path = os.path.join(temp_folder, file_name)
86             with open(file_path, "w") as outfile:
87                 json.dump(json_reponse, outfile)
88
89             upload_success(upload_success(file_path: str, bucket_name))
90             if upload_succ:
91                 os.remove(file_path)
92                 print(f"Local File '{file_path}' removed after successful upload.")
93             else:
94                 print(f"Local file '{file_path}' retained due to upload failure.")
95
96         else:
97             print(f"Failed to get data for ID: {id}, Status Code: {response.status_code}")
98
99     return results
```

- Rest Get API
- Use the ID Token retrieved as a Bearer Token for this API
- Multiple parts of GPS data, so retrieve each point with its unique ID
- Store all in a temporary file for upload later

# Uploading the Data

Intern > s3Upload.py > upload\_file

```
1 import logging
2 import boto3
3 from botocore.exceptions import ClientError
4 import os
5
6 def upload_file(file_path):
7
8     bucket_name = "████████████████████████████████"
9     access_key = "████████████████████████████████"
10    secret_key = "████████████████████████████████"
11
12    s3 = boto3.client(service_name="s3",
13                        aws_access_key_id=access_key,
14                        aws_secret_access_key=secret_key
15    )
16    response = s3.upload_file(file_path, bucket_name, file_path)
17    print(f"upload_log_to_aws response: {response}")
18    if response is None:
19        print("Success")
20    else:
21        print("Error")
```

- S3 Bucket: Database in AWS
- Using Boto3 Library for S3 Upload:  
<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-uploading-files.html>
- boto3.client().upload\_file() method

# Combining All Steps

```

Intern > $ gpsData_to_s3.py > get_id_token
1 import requests
2 import json
3 import boto3
4 from botocore.exceptions import NoCredentialsError, PartialCredentialsError
5 import re
6 import os
7
8 bucket_name = "XXXXXXXXXX"
9 access_key = "XXXXXXXXXX"
10 secret_key = "XXXXXXXXXX"
11 api_key = "XXXXXXXXXX"
12
13 s3 = boto3.client(service_name='s3',
14     aws_access_key_id=access_key,
15     aws_secret_access_key=secret_key
16 )
17
18 def get_id_token(api_key):
19     post_login_url = "XXXXXXXXXX"
20     headers = {'x-api-key': api_key}
21     data = {
22         "username": "XXXXXXXXXX",
23         "password": "XXXXXXXXXX"
24     }
25
26     response = requests.post(post_login_url, headers=headers, json=data, verify=True)
27
28 if response.status_code == 200:
29     json_response = response.json()
30     location = json_response.get("location", "")
31     #print(location)
32     code_match = re.search(r'code=(\w+)', location)
33     if code_match:
34         code_value = code_match.group(1)
35         print("Code:", code_value)
36     else:
37         print("Code not found in the location URL.")
38         return None
39
40 token_url = "XXXXXXXXXX"
41 body = {
42     "grant_type": "password",
43     "client_id": "XXXXXXXXXX",
44     "code": code_value,
45     "redirect_uri": "XXXXXXXXXX"
46 }
47
48 token_response = requests.post(token_url, data=body,
49     headers={"Content-Type": "application/x-www-form-urlencoded"})
50 #print("token", token_response)
51
52 if token_response.status_code == 200:

```

```

53     json_response = token_response.json()
54     id_token = json_response.get("id_token", "")
55     return id_token
56 else:
57     print("Error getting token:", token_response.json())
58     return None
59 else:
60     print("Login Error:", response.text)
61     return None
62
63 def get_gps(id_token, start_id, end_id):
64     temp_url = "XXXXXXXXXX":id={id}"
65     headers = {"Authorization": f'Bearer {id_token}'}
66     results = []
67
68     temp_folder = "temp" #"/Users/landonzheng/Documents/Intern/temp"
69     if not os.path.exists(temp_folder):
70         os.makedirs(temp_folder)
71
72     for id in range(start_id, end_id + 1):
73         url = temp_url.replace('{id:{id}', f'{id}')
74         response = requests.get(url, headers=headers, verify=False)
75
76         #print(response.json())
77         print(f"ID: {id}, HTTP Status Code: {response.status_code}")
78
79         if response.status_code == 200:
80             json_response = response.json()
81             results.append(json_response)
82
83             file_name = f'{id}-gps.json'
84             file_path = os.path.join(temp_folder, file_name)
85             with open(file_path, "w") as outfile:
86                 json.dump(json_response, outfile)
87
88             upload_success = upload_to_s3(file_path, bucket_name)
89             if upload_success:
90                 os.remove(file_path)
91                 print(f"Local File '{file_path}' removed after successful upload.")
92             else:
93                 print(f"Local file '{file_path}' retained due to upload failure.")
94
95             else:
96                 print(f"Failed to get data for ID: {id}, Status Code: {response.status_code}")
97
98     return results
99
100
101     if object_name is None:
102         object_name = os.path.basename(file_path)

```

```

104     try:
105         s3.upload_file(file_path, bucket_name, object_name)
106         print(f"File '{file_path}' uploaded to '{bucket_name}/{object_name}'")
107         return True
108     except FileNotFoundError:
109         print(f"The file '{file_path}' was not found.")
110         return False
111     except NoCredentialsError:
112         print("Credentials not available.")
113         return False
114     except PartialCredentialsError:
115         print("Incomplete credentials provided.")
116         return False
117     except Exception as e:
118         print(f"An error occurred: {e}")
119         return False
120
121     id_token = get_id_token(api_key)
122
123     if id_token:
124         start_id = 1
125         end_id = 10
126         data = get_gps(id_token, start_id, end_id)
127     else:
128         print("Failed to obtain id_token.")
129

```

- All 3 steps in one class
- Changed some methods in order to function all together
- By running this python script, GPS data will be automatically uploaded to the S3 Bucket



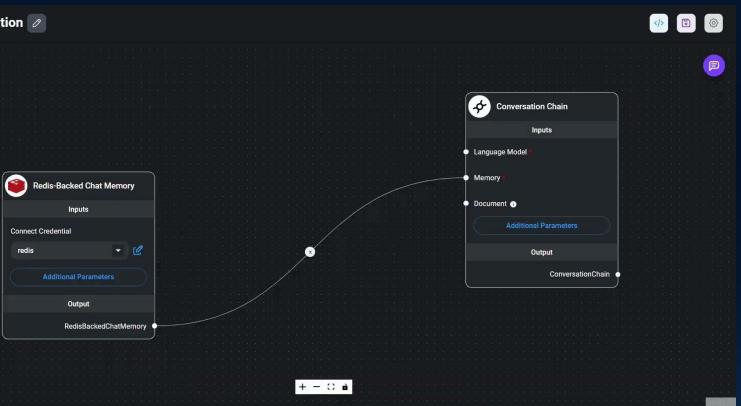
# Flowise

What is Flowise, and my  
experience

+++

# What is Flowise?

- Traditionally, connecting LLMs (Chat GPT) to serve personal projects is difficult
- Low-code tool that enables LLMs to be connected to more services through interactive chatbots
- Turns LangChain's complex code into simple drag-and-drops
- Able to build personalized AI assistants- a need for the future
- Download with npm or Docker, used on Google through local host



Credit: [Flowiseai.com](https://flowiseai.com)

# Learning Flowise

I learned Flowise with a 10-Part [Tutorial](#), exploring most of Flowise's tools:

## **C**hains & LLMs

A chain is a basic interactive chatbot. They are connected to a LLM (Large Language Model) such as Chat GPT and a prompt template for how to behave. There are many types of chains.

## **R**AG & Vector Storage

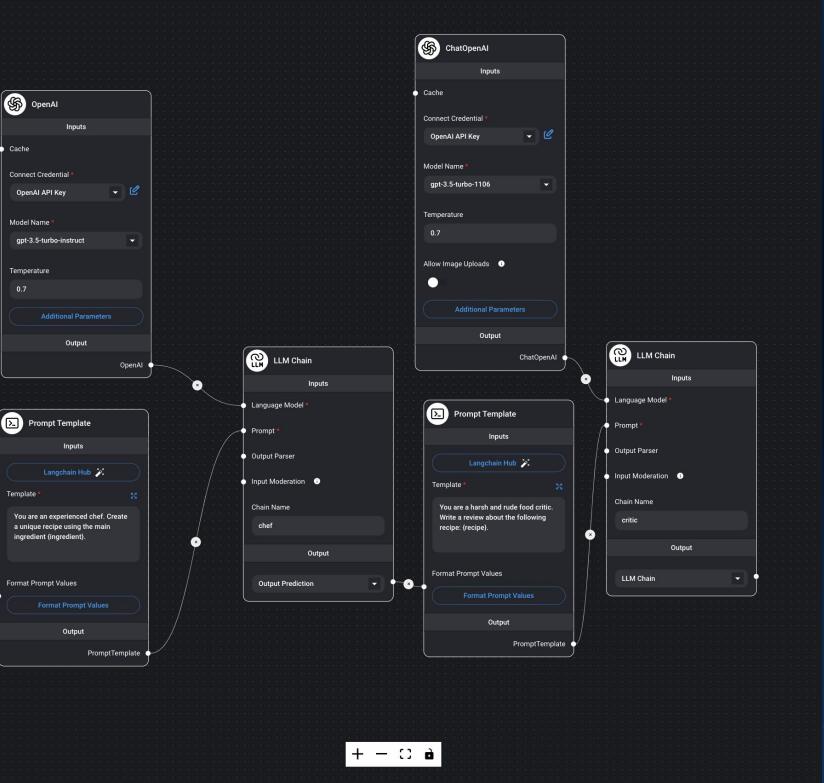
RAG (Retrieval Augmented Generation) is a useful tool in AI, allowing them to obtain information off external sources. These sources are stored in vector storages, an easy way for LLMs to comprehend the info.

## **A**gents

Agents are by far the most powerful chatbot in Flowise. They are equipped with memory for previous conversations, a LLM chat model, and multiple customizable tools. The agent consciously picks which tool it needs to complete a task.

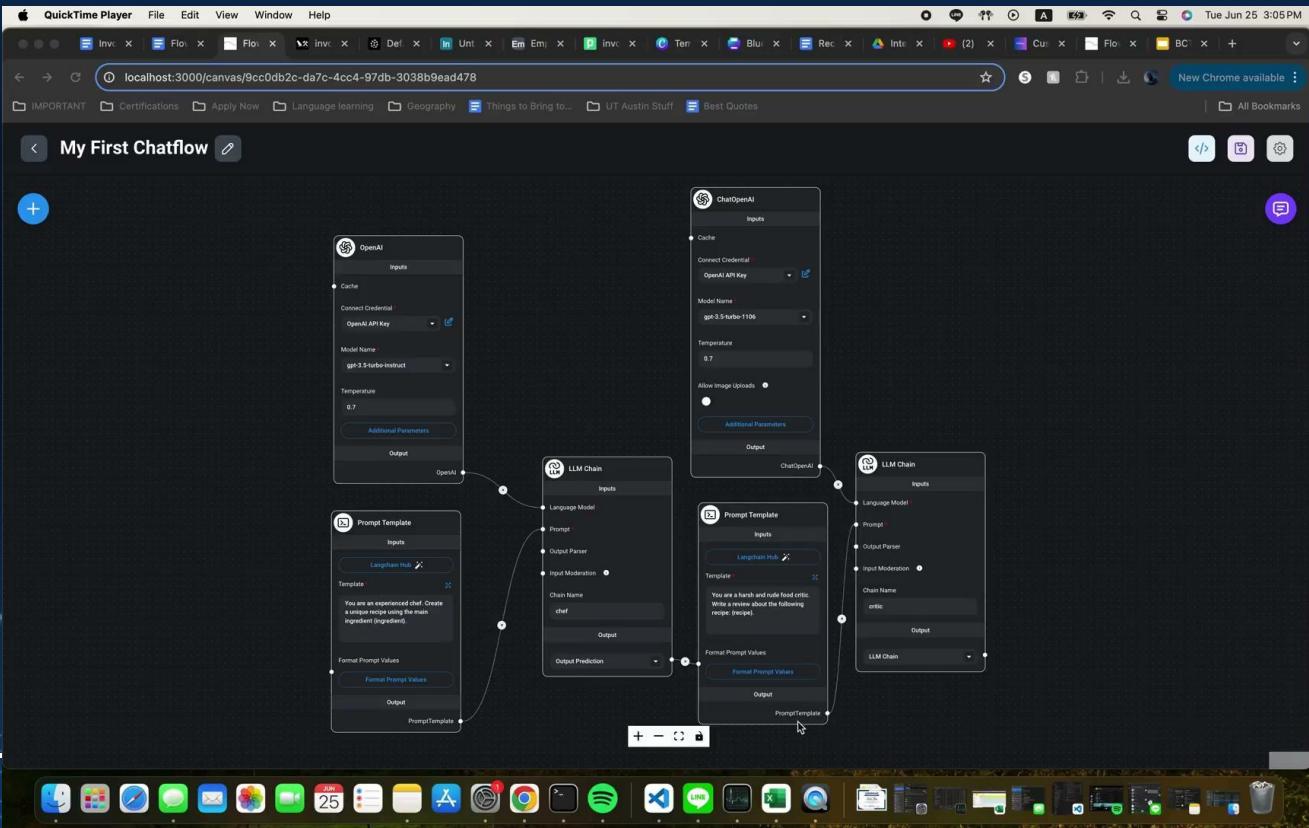


# Chains & LLMs

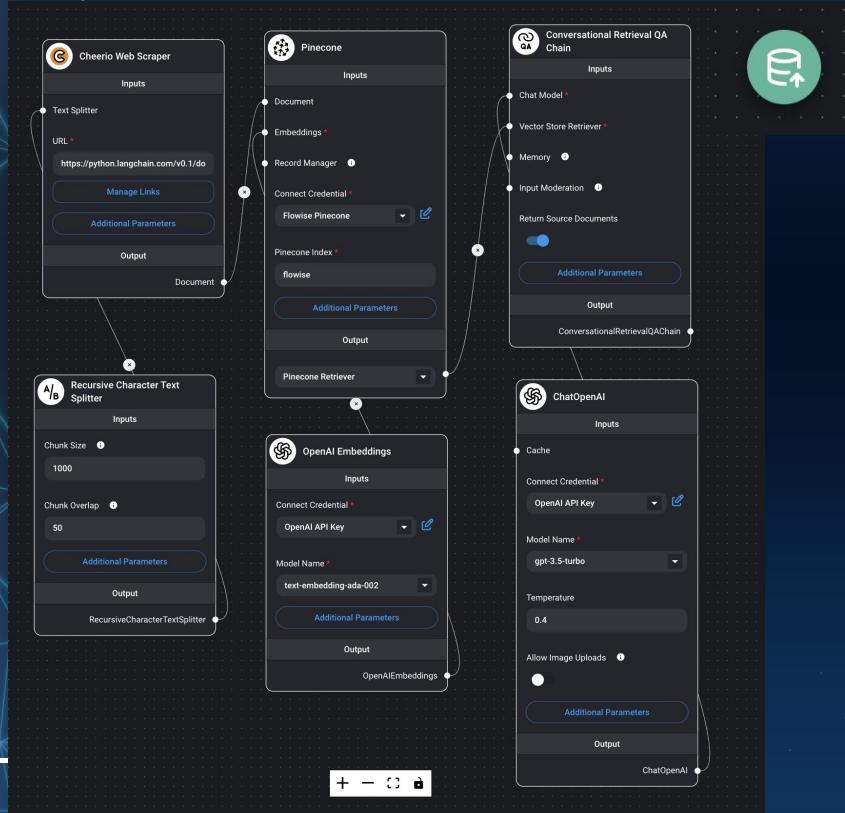


- This Chatflow has 2 LLM Chains- basic chains that connect Chat GPT to an interactive chatbot
- 1st LLM Chain (chef) has a prompt template, telling it to create a unique recipe when an ingredient is typed in
- That recipe is then passed to another prompt LLM chain with a prompt template to write a harsh review about the recipe
- So, when an ingredient is typed in, a criticism of the recipe created by the 1st LLM Chain is outputted

# Chains & LLMs Video Demo

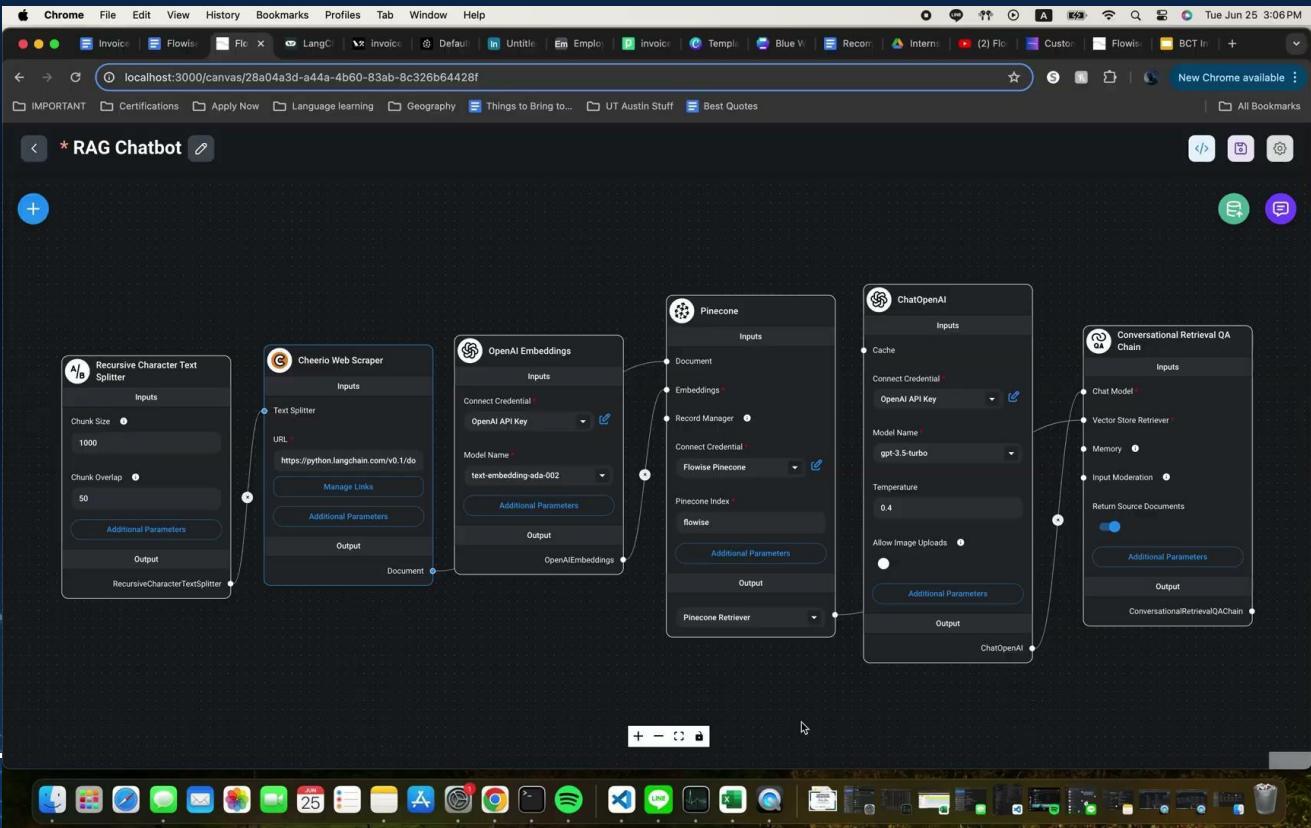


# RAG & Vector Storage

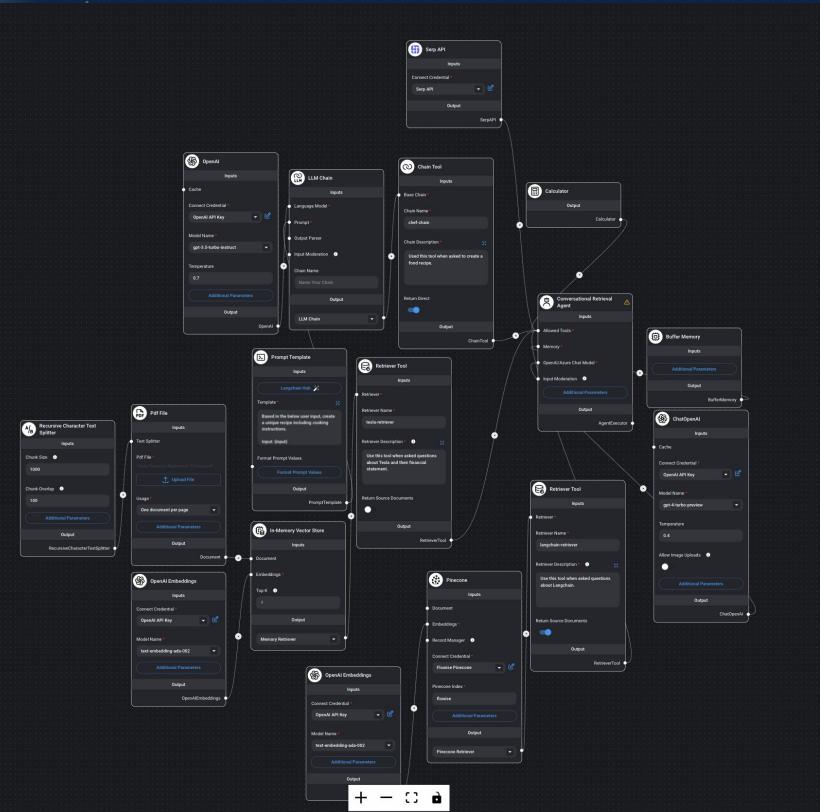


- A Conversational Retrieval QA Chain is used- a special chain that can retrieve data from vector stores
- Pinecone is a very common vector store; embeddings are used to translate the vectors into info
- Pinecone grabs information from the Cheerio Web Scraper, which scrapes a page about LCEL
- Text Splitters are used to chunk the text and make it easier to process
- MUST UPSERT INFO (Green Button on Top Right Corner) to Pinecone Vector Storage before testing AI Chatbot

# RAG & Vector Storages Video Demo

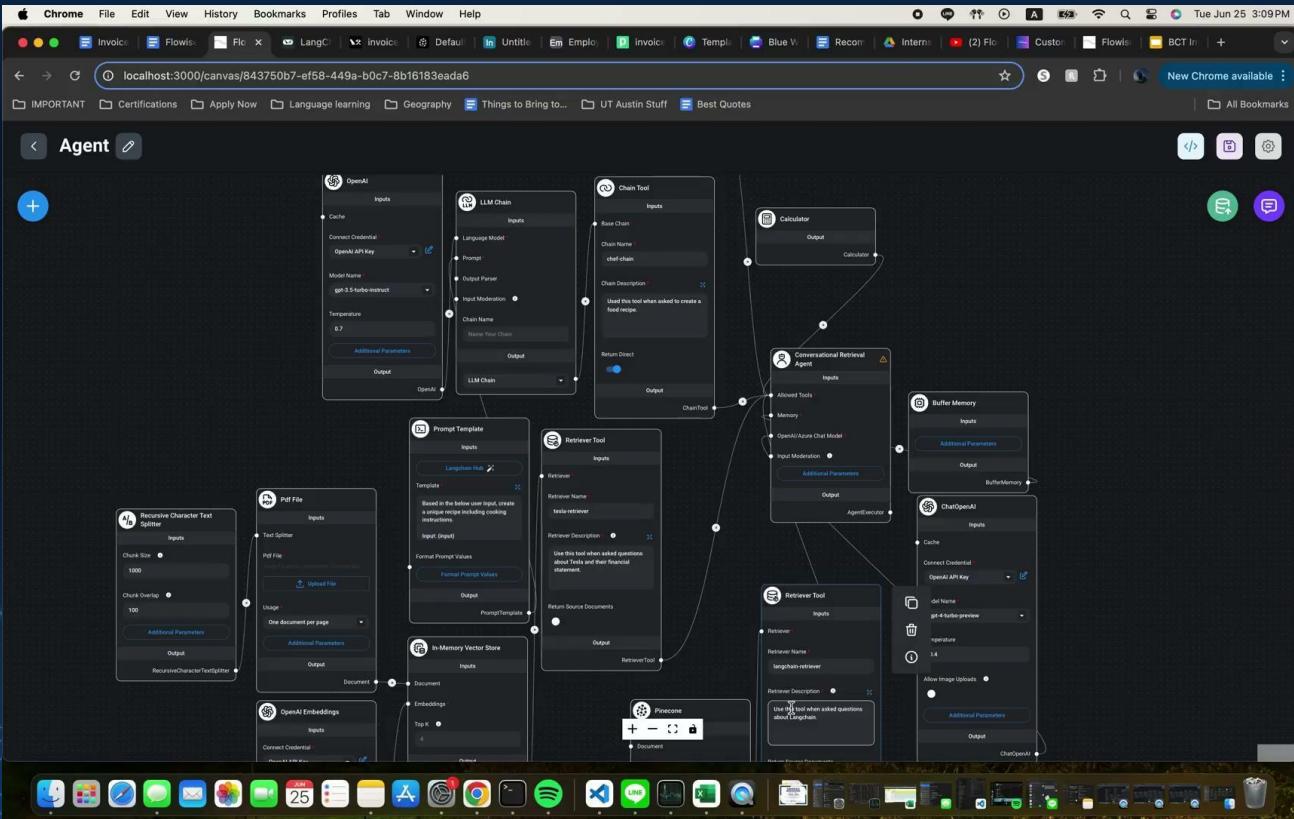


# Agents



- An agent can be connected to many tools, and picks which one to use.
- A Conversational Retrieval Agent has 2 more capabilities- RAG and Function calling. Function calling is used to call custom tools (seen later)
- This agent is connected to the tools: Serp API (searches Google), Chef Chain Tool (creates a unique recipe off an ingredient). Retriever Tool 1 (Info about Tesla Financial Statement), Retriever Tool 2 (Info about LCEL), and a Calculator (Math computation).
- These tools help the agent in specific questions.

# Agents Video Demo





# Project 1

AI-Powered Employee Leave Database  
& Uploader

# Problem to Solve

## Current situation

Right now, BCT's Employee Leave system is an old website that's glitchy and hard-to-use. An improvement to the leave system is needed.

1

### RAG

The AI has to understand BCT's Leave Policy



2

### Retrieve

The AI has to retrieve info from the database



3

### Upload

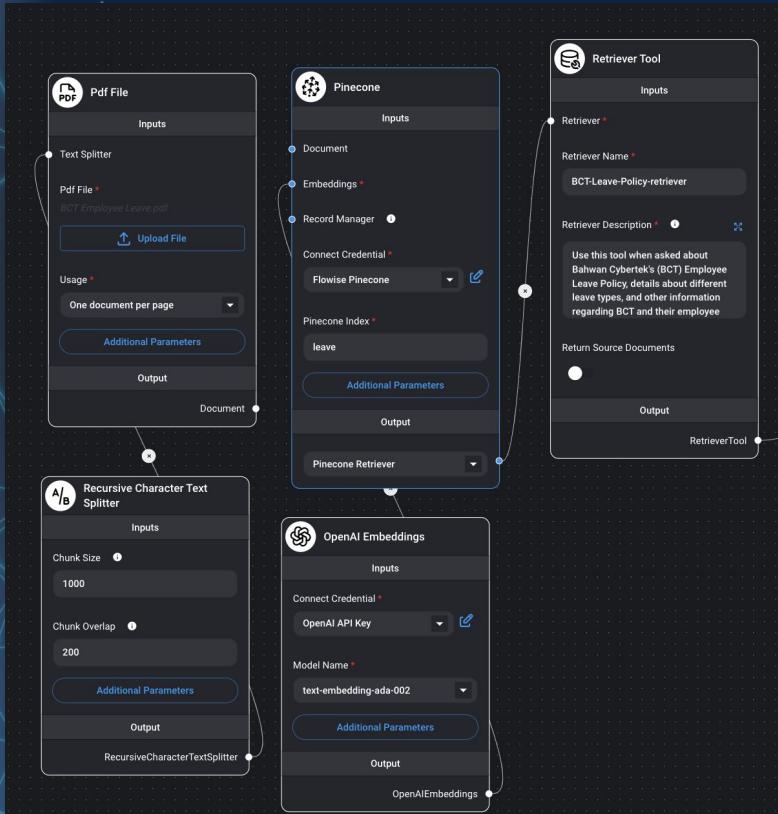
The AI has to upload leave info into the database



## Solutions Needed

I concluded I need to use a Conversational Retrieval Agent with Chat GPT 4-Turbo-Preview for RAG & Custom Tools. Separate steps are fine.

# Using RAG For BCT's Policy Sheet



- With BCT's Policy Sheet, I used a simple RAG procedure
- Retriever Tool for the Agent, specified should be used when asked about BCT Leave Policy
- Pinecone as Vector Storage
- Uploaded the PDF of the Policy Sheet as the Document, split into chunks with a text splitter
- Upserted the PDF into Pinecone (and now can disconnect PDF File to Pinecone)

# Retrieving Data with Custom Tool

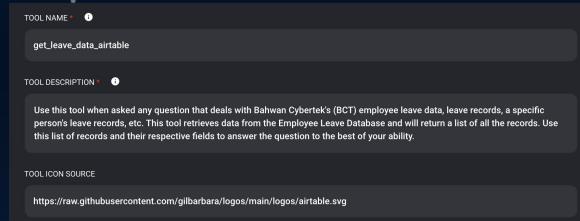
Javascript Function:

```
const fetch = require('node-fetch');

const token = 'API_KEY';
const baseId = 'BASE_ID';
const tableId = 'TABLE_ID';
const url = `https://api.airtable.com/v0/${baseId}/${tableId}`;
const fields = ['id', 'name', 'leave_type', 'start_date', 'duration'];

const options = {
  method: 'GET',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  }
};

try {
  const response = await fetch(url, options);
  const text = await response.json();
  return text.records.map(record => ({
    id: record.fields.id,
    name: record.fields.name,
    leave_type: record.fields.leave_type,
    start_date: record.fields.start_date,
    duration: record.fields.duration
  }));
} catch (error) {
  console.error(error);
  return 'Error';
}
```



- For the Database, I chose Airtable because of its free API integration
- Custom Tools use OpenAI's Function Calling, so a Conversational Retrieval Agent or OpenAI Function Agent is needed
- The JS Function sends an Airtable Get Request to fetch the entire table of leave records so the LLM can interpret them and answer the user's question about the existing leave records
- Need a Bearer API Key, Base ID, and Table ID

# Uploading Data with Custom Tool

## Javascript Function:

```
const fetch = require('node-fetch');
const baseId = 'BASE_ID';
const tableId = 'TABLE_ID';
const token = 'API_KEY';

const body = {
  "records": [
    {
      "fields": {
        "id": $id,
        "name": $name,
        "leave_type": $leave_type,
        "start_date": $start_date,
        "duration": $duration,
      }
    }
  ]
};

const options = {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(body)
};

const url = `https://api.airtable.com/v0/${baseId}/${tableId}`

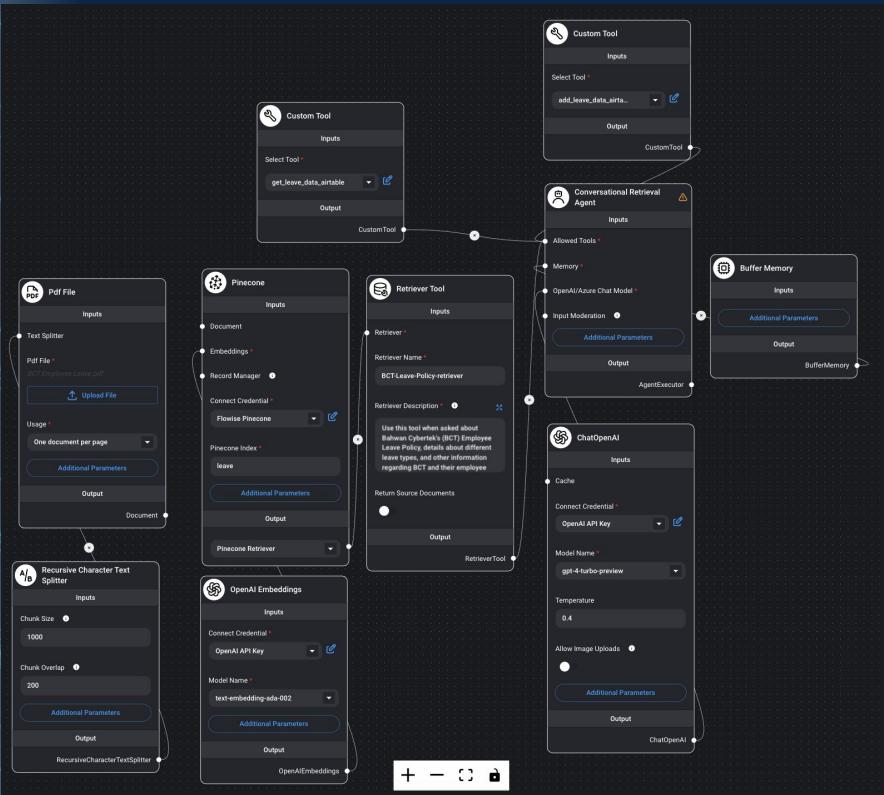
try {
  const response = await fetch(url, options);
  const text = await response.text();
  return text;
} catch (error) {
  console.error(error);
  return 'Error';
}
```

The screenshot shows the configuration interface for a custom tool named 'add\_leave\_data\_airtable'. It includes a tool description, a tool icon source URL (<https://raw.githubusercontent.com/gilbarbara/logos/main/logos/airtable.svg>), and an input schema table.

Property	Type	Description	Required
id	number	The Employee's ID	✓
name	string	The Employee's name. Correctly Capitalized	✓
leave_type	string	The type of Leave only from these options	✓
start_date	date	The date that the leave begins	✓

- The Airtable has 5 different fields needed to create a leave record: Employee ID, Employee Name, Leave Type, Start Date, and Duration
- Creating a custom tool to upload the records with a custom JS function is the most efficient way to upload a record
- The Javascript function sends a Post Request with the required fields in order to create a leave record

# Final Chatflow



- Combining all the above parts, the final chatflow has a conversational retrieval agent with 2 Custom Tools to get and upload data from Airtable, and a Retrieval Tool to understand BCT Leave Policy Data
- The agent will ask for confirmation before uploading to make sure details are correct
- Possible change: Agent asks for user and password first before granting access

# Getting Leave Records Video Demo

The screenshot shows a Mac desktop environment with a dark blue background. In the center, a browser window is open to `localhost:3000/tools`. The window title is "Edit Tool". The tool configuration includes:

- TOOL NAME:** `get_data_airtable`
- TOOL DESCRIPTION:** A detailed description explaining the tool's purpose: "Use this tool when asked any question that deals with Bahwan Cybertek's (BCT) employee leave data, leave records, a specific person's leave records, etc. This tool retrieves data from the Employee Leave Database and will return a list of the records. Use this list of records and their respective fields to answer the question to the best of your ability."
- TOOL ICON SOURCE:** <https://raw.githubusercontent.com/gilbarbara/logos/main/logos/airtable.svg>
- INPUT SCHEMA:** A table with columns: Property, Type, Description, and Required. It currently displays "No rows". Buttons for "Delete" and "Save" are at the bottom.

To the right of the browser window, a sidebar shows recent files and locations. Below the browser, a secondary window titled "Employee Leave Dat..." is visible, displaying a list of leave records in a table format. The table includes columns for ID, Name, and Type. Some entries are highlighted with blue borders. At the bottom of the screen, the Mac OS X Dock is visible with various application icons.

Property	Type	Description	Required
No rows			

Leave Record Data (Visible in Secondary Window):

ID	Name	Type
11	Renna	Casual Leave
12	Landon Zheng	Paid Time Off
13	Dylan	Leisure Leave
14	user	Casual Leave
28	Landon Zheng	Casual Leave

# Uploading Leave Records Video Demo

The screenshot shows a video player window displaying a demonstration of an Employee Leave App. The app interface includes sections for "Employee Leave App", "Pinecone", "OpenAI Embeddings", and a "Retriever Tool". A modal window titled "create leave:" shows the creation of a leave record with ID 28, name Landon Zheng, type casual, start date 6/26, and duration 8 days. In the background, a PDF titled "BCT Employee Leave.pdf" is open, detailing leave policies:

No of Days	Conditions
12 days	<ul style="list-style-type: none"><li>Associate need to submit medical certificate in case of more than 3 consecutive days.</li><li>Carry forward is not allowed.</li></ul>
6 days	<ul style="list-style-type: none"><li>Carry forward or encashment of leave is not allowed.</li></ul>
10 days, and gain one day increase by every full-service year	<ul style="list-style-type: none"><li>PTO can be availed after completing 6 months.</li><li>Carry forward of up to 45 days is allowed.</li><li>Encashment on a month's salary is allowed before end of the year.</li><li>Minimum of 30 days need to be retained during any encashment of leave.</li><li>Balance after leave balance has reached 45 days.</li></ul>
Bereavement Leave	<ul style="list-style-type: none"><li>3 days for sibling/grandparents of spouse</li><li>6 days for grandparents/children/ parents of spouse</li><li>8 days for</li><li>Bereavement leave can be applied for spouse/parents/children/siblings/grandparents of spouse/grandparents</li></ul>
Leisure Leave	<ul style="list-style-type: none"><li>Maximum 6 days in a year</li><li>Leisure leave can be claimed only when an</li></ul>

The video player toolbar at the top includes controls for play, volume, and search. The Mac OS X dock at the bottom contains icons for various applications like Finder, Mail, Safari, and Spotify.



04

# Project 2

AI-Powered Invoice Processor &  
Uploader

+++

# Problem to Solve

## Current situation

BCT, as any company, has many expenses. Some come in invoices, and the traditional method is to manually scan and upload them. This can be improved with AI chatbots.

1

### Understand

The AI must understand invoice image uploads



2

### Summarize

The AI has to summarize the needed fields



3

### Upload

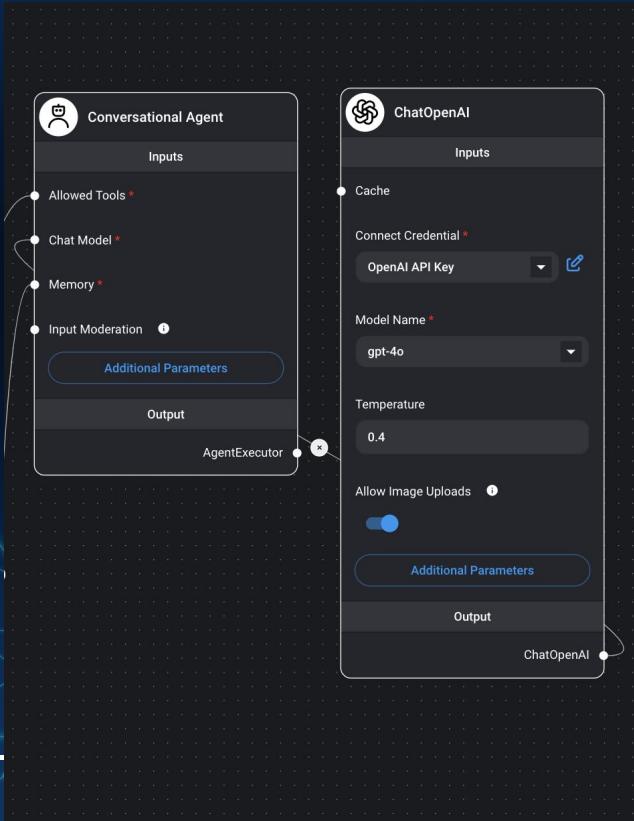
The AI has to upload invoice info into the database



## Solutions Needed

I concluded I need to use a Conversational Agent with Chat GPT 4o for Image Understanding. Also, I wanted the invoice processing and uploading to happen ALL AT ONCE, rather than at separate steps like Employee Leave.

# Understanding Image Uploads



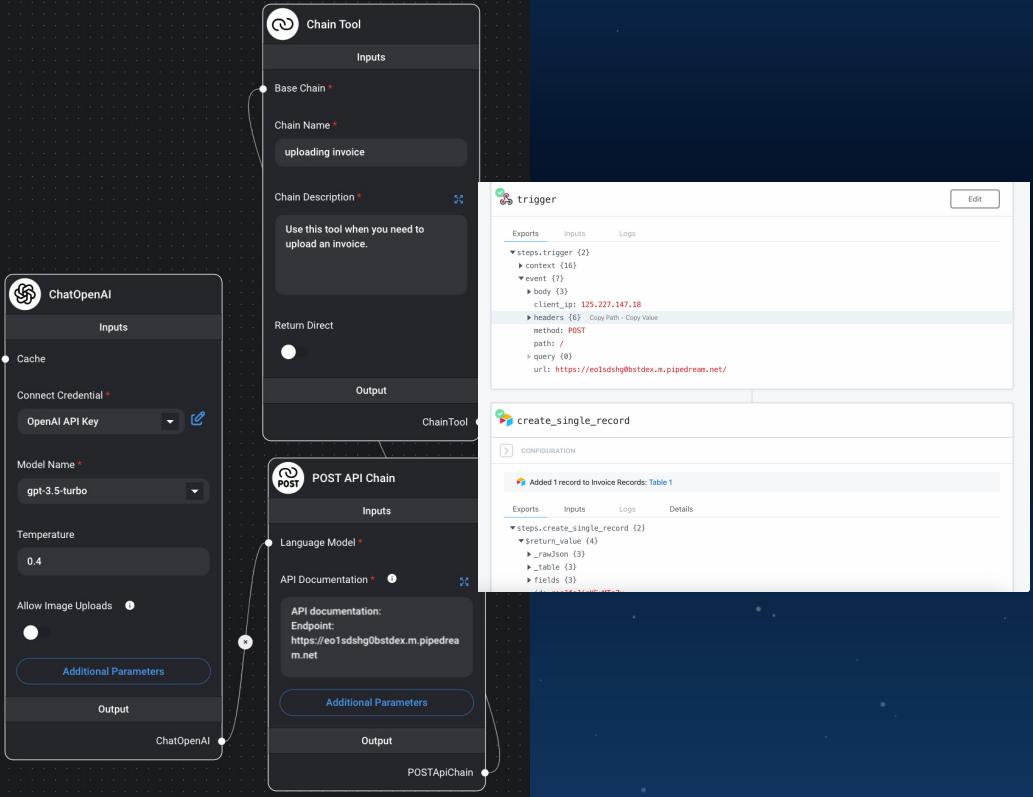
- OpenAI recently released Vision Capabilities, which means that some models can process images. GPT-4o is one of them, so I used this model.
- Since Flowise is still in development, only Conversational Agent & React Agent are compatible with Image Uploads. I picked Conversational Agent
- The Conversational Agent becomes a problem pretty soon.

# Summarizing Invoice Images



- This is an example invoice. The required fields to obtain are the Name of the place purchased, the Price (總計), and the Buyer ID (買方). This example would be Miramar, 600, and 43942656
- Full Invoice Documentation can be seen [here](#)
- Vision Capabilities automatically allow Chat GPT to get these fields (although sometimes inaccurate), just needed to specify in the prompt template of the agent under “Additional Parameters”

# Uploading Invoice Details



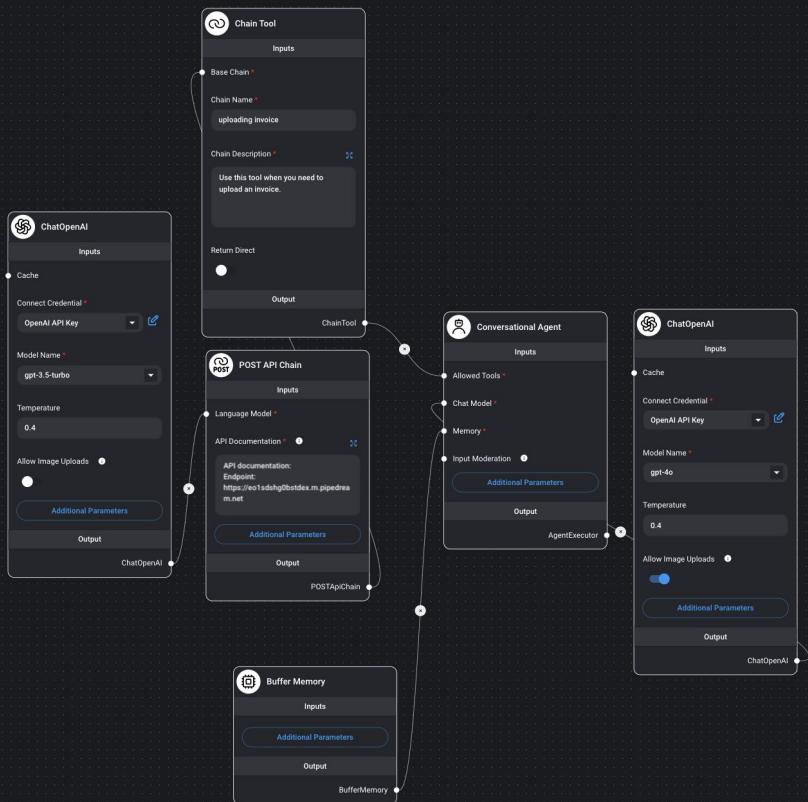
- Airtable was used as the database
- To upload the invoice, I used a chain tool that connected to a Post API Chain- executing Post Requests.
- API Documentation tells the LLM to put the name, price, and buyer fields in the body
- This chain posts to a Pipedream API, which triggers a process due to the HTML post. This then creates a record inside Airtable.
- This method costs \$29/month because Pipedream isn't free

# Why is Uploading so Complicated?

As you can see, uploading invoices is quite complicated and expensive. Why can't I just use a Custom Tool like when uploading Employee Leave Records? This is because:

- Remember how I was restricted to Conversational Agents due to image processing compatibility? Conversational Agents don't have function calling, meaning they can't call a Custom Tool or another Chatflow. This rules out Custom tools completely.
- Additionally, Posting directly to Airtable with the Post API Chain (instead of Posting onto Pipedream which posts onto Airtable) is not possible, as Flowise only configured LLMs to post requests with JSON strings. Airtable API requires JSON objects, which thus makes anything posting directly to Airtable impossible.
- Thus, the best solution I can find right now is posting on a Pipedream API to trigger a post onto Airtable.

# Final Chatflow



- Combining all the above parts, the final chatflow has a conversational agent with 1 Chain Tool, which uploads invoice details to Airtable using Pipedream as a middleman
- Keep in mind I configured this process to be automatic, so that when an invoice was uploaded the Agent would automatically process and summarize the invoice image details and then upload the details onto Airtable in one go.

# Invoice Processing Video Demo

The screenshot displays a video demo interface with two main windows. The left window shows a complex flowchart titled "[DRAFT] Invoice Processing" using various AI tools. The right window shows the resulting invoice document.

**Left Window: [DRAFT] Invoice Processing**

- POST API Chain:** An AI tool connected to the flowchart. It has "Language Model" and "API documentation Endpoint" inputs, and "Additional Parameters" and "Output" outputs.
- Chain Tool:** An AI tool connected to the flowchart. It has "Chain Name" (set to "uploading invoice"), "Chain Description" (set to "Use this tool when you need to upload an invoice."), "Return Direct" (set to "Output"), and "Output" outputs.
- Conversational Agent:** An AI tool connected to the flowchart. It has "Allowed Tools" (set to "Chat Model", "Memory", and "Input Moderation"), "Additional Parameters" and "Output" outputs.
- ChatOpenAI:** An AI tool connected to the flowchart. It has "Cache", "Connect Credential", "OpenAI API Key", "Model Name" (set to "gpt-4e"), "Temperature" (set to 0.4), "Allow Image Uploads" (set to "On"), and "Output" outputs.

**Right Window: invoice\_02.png**

**flare 電子發票證明聯**  
113 年 03-04 月  
ZM31711769  
2024-03-19 22:29:07 格式 25  
隨機碼: 5676 總計: 2,332 元  
賣方: 90277843 買方: 43942656

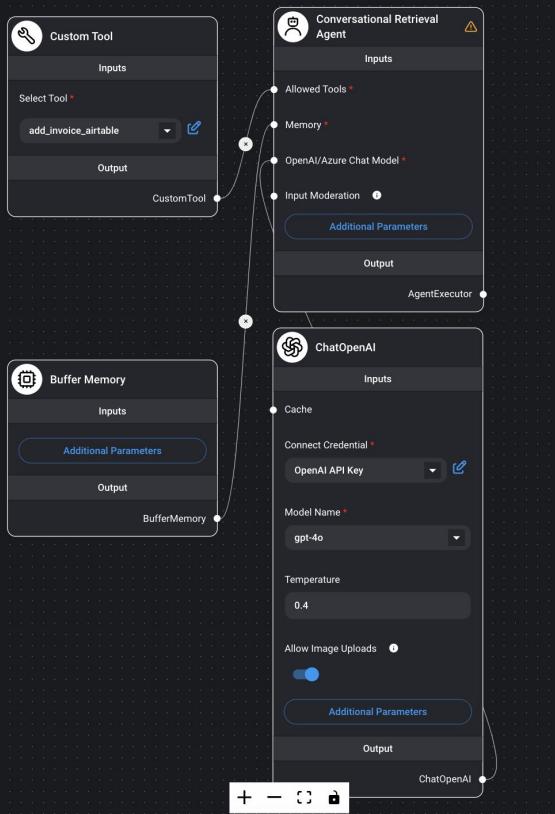
**Barcode:** A standard barcode representing the invoice number ZM31711769.

**QR Codes:** Two QR codes, one for the invoice and one for payment.

**Invoice Details:**

品名	單價*	數量	金額(未稅)	課稅別
餐費	2221	*1	2221	TX
銷售額合計:	2221			
稅額合計:	111			
發票號碼:	ZM31711769			
已付:	2332 元			尚欠: 0 元

# Potential Chatflow in the Future?



## Javascript Function for Tool:

```
const fetch = require('node-fetch');
const baseld = 'BASE_ID';
const tableid = 'TABLE_ID';
const token = 'API_KEY';

const body = {
  "records": [
    {
      "fields": {
        "name": $name,
        "price": $price,
        "buyer": $buyer
      }
    }
  ]
};

const options = {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(body)
};

const url = `https://api.airtable.com/v0/${baseld}/${tableid}`

try {
  const response = await fetch(url, options);
  const text = await response.text();
  return text;
} catch (error) {
  console.error(error);
  return 'Error';
}
```

- A Conversational Retrieval Agent would make this much easier. With function calling, I could just use a custom tool (already built) to upload directly to Airtable, just like in the Employee Leave Project.
- Conversational Retrieval Agent will be deprecated (updated) in the future, so if it has image upload compatibility, this is a much simpler and more efficient way.

+++

Thank you for  
everything, BCT. It was  
amazing!

Special Thanks to Dylan for giving me this intern chance, Jeff for mentoring me through and through, Renna for the assistance during the migration process, and Ken & Alex for making the lunches so enjoyable.

+++