a **modular Python-based agent implementation** that handles:

1. Email reading
2. Intent detection
3. Routing
4. Backend system interaction via **APIs**
5. Response generation and email reply

**🧠 Modular Agent Architecture (API-based)**

**📦 Module Breakdown:**

plaintext

CopyEdit

solution\_center\_agent/

├── config.py # API keys, endpoints, credentials

├── email\_reader.py # Read incoming emails

├── intent\_classifier.py # Detect user intent and extract details

├── router.py # Decide which system to call

├── systems/

│ ├── concur\_api.py # API handler for Concur

│ ├── corp\_card\_api.py # API handler for Corporate Card

│ └── invoice\_api.py # API handler for Invoice Processing

├── response\_generator.py # Generate email replies

├── email\_sender.py # Send response emails

└── main.py # Orchestrates the entire flow

**🧪 Step-by-Step Sample Implementation (Core Logic)**

**✅ 1. email\_reader.py (IMAP-based)**

python

CopyEdit

import imaplib, email

from email.header import decode\_header

def fetch\_latest\_email():

mail = imaplib.IMAP4\_SSL("imap.example.com")

mail.login("solution@company.com", "yourpassword")

mail.select("inbox")

result, data = mail.search(None, "UNSEEN")

if not data[0]:

return None

latest\_email\_id = data[0].split()[-1]

result, msg\_data = mail.fetch(latest\_email\_id, "(RFC822)")

raw\_email = msg\_data[0][1]

msg = email.message\_from\_bytes(raw\_email)

subject = decode\_header(msg["Subject"])[0][0]

body = ""

if msg.is\_multipart():

for part in msg.walk():

if part.get\_content\_type() == "text/plain":

body = part.get\_payload(decode=True).decode()

else:

body = msg.get\_payload(decode=True).decode()

return {"subject": subject, "body": body, "from": msg["From"]}

**✅ 2. intent\_classifier.py (OpenAI-based)**

python

CopyEdit

from openai import OpenAI

import os

import openai

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

def detect\_intent(email\_text):

prompt = f"""

Analyze the following email and return:

- intent (like 'get\_card\_balance', 'create\_travel\_request')

- system (concur, corporate\_card, invoice)

- parameters (user info, travel city, etc.)

Email:

{email\_text}

"""

response = openai.ChatCompletion.create(

model="gpt-4",

messages=[{"role": "user", "content": prompt}]

)

return response.choices[0].message['content']

**✅ 3. router.py**

python

CopyEdit

from systems import concur\_api, corp\_card\_api, invoice\_api

def handle\_request(parsed\_intent):

system = parsed\_intent['system']

intent = parsed\_intent['intent']

params = parsed\_intent['parameters']

if system == "concur":

return concur\_api.handle\_concur(intent, params)

elif system == "corporate\_card":

return corp\_card\_api.handle\_card(intent, params)

elif system == "invoice":

return invoice\_api.handle\_invoice(intent, params)

else:

return "Unknown system"

**✅ 4. systems/concur\_api.py (example stub)**

python

CopyEdit

def handle\_concur(intent, params):

if intent == "check\_travel\_eligibility":

# Example API call

# response = requests.get("https://api.concur.com/v1/eligibility", headers=..., params=...)

return "Yes, you are eligible for travel."

elif intent == "create\_travel\_request":

return f"Travel request to {params['destination']} has been created."

return "Concur intent not supported."

*(Repeat similar for corp\_card\_api.py and invoice\_api.py)*

**✅ 5. response\_generator.py**

python

CopyEdit

def generate\_response(user\_email, result):

return f"Dear user,\n\nHere is the response to your query:\n\n{result}\n\nRegards,\nSolution Center"

**✅ 6. email\_sender.py**

python

CopyEdit

import smtplib

from email.mime.text import MIMEText

def send\_email(to, subject, body):

msg = MIMEText(body)

msg["Subject"] = subject

msg["From"] = "solution@company.com"

msg["To"] = to

with smtplib.SMTP("smtp.example.com", 587) as server:

server.starttls()

server.login("solution@company.com", "yourpassword")

server.send\_message(msg)

**✅ 7. main.py**

python

CopyEdit

from email\_reader import fetch\_latest\_email

from intent\_classifier import detect\_intent

from router import handle\_request

from response\_generator import generate\_response

from email\_sender import send\_email

def main():

email = fetch\_latest\_email()

if not email:

print("No new emails.")

return

parsed = detect\_intent(email['body'])

parsed\_dict = eval(parsed) # you can use `json.loads()` with proper format

result = handle\_request(parsed\_dict)

reply = generate\_response(email["from"], result)

send\_email(email["from"], "Re: " + email["subject"], reply)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**✅ Summary**

| **Module** | **Purpose** |
| --- | --- |
| email\_reader.py | Reads unread emails |
| intent\_classifier.py | Uses GPT-4 to detect query type |
| router.py | Chooses which backend system to contact |
| systems/ | Contains API clients for each system |
| response\_generator.py | Formats natural reply |
| email\_sender.py | Sends response to user |
| main.py | Orchestrates the entire flow |