## 🧩 ****Objective****

To develop a **Python-based AI Integration Framework** that:

Connects to **multiple AI providers**.

Executes a **prompt or task** across all of them.

Returns results in a **standardized format** for comparison or reporting.

## ⚙️ ****Supported AI Tools****

| **AI Tool** | **Provider** | **Library Used** | **Example Model** |
| --- | --- | --- | --- |
| OpenAI | ChatGPT, GPT-4 | openai | gpt-4o-mini |
| Anthropic | Claude | anthropic | claude-3-sonnet |
| Google AI | Gemini | google-generativeai | gemini-1.5-flash |
| Hugging Face | Transformers API | requests | google/flan-t5-xl |

Python Code: Multi-AI Integration Framework

multi\_ai\_framework.py

A Python framework compatible with multiple AI tools.

Developed for testing, comparison, and report generation across AI platforms.

"""

import os

import requests

# Import optional SDKs (install only what you need)

import openai

import anthropic

import google.generativeai as genai

# -----------------------------

# Configuration (Set API keys)

# -----------------------------

OPENAI\_API\_KEY = os.getenv("OPENAI\_API\_KEY")

ANTHROPIC\_API\_KEY = os.getenv("ANTHROPIC\_API\_KEY")

GOOGLE\_API\_KEY = os.getenv("GOOGLE\_API\_KEY")

HUGGINGFACE\_API\_KEY = os.getenv("HUGGINGFACE\_API\_KEY")

openai.api\_key = OPENAI\_API\_KEY

genai.configure(api\_key=GOOGLE\_API\_KEY)

# -----------------------------

# AI Interface Functions

# -----------------------------

def query\_openai(prompt: str, model="gpt-4o-mini"):

"""Query OpenAI GPT models"""

try:

response = openai.ChatCompletion.create(

model=model,

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

temperature=0.5

)

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

except Exception as e:

return f"OpenAI Error: {e}"

def query\_anthropic(prompt: str, model="claude-3-sonnet"):

"""Query Anthropic Claude models"""

try:

client = anthropic.Anthropic(api\_key=ANTHROPIC\_API\_KEY)

response = client.messages.create(

model=model,

max\_tokens=512,

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

)

return response.content[0].text

except Exception as e:

return f"Anthropic Error: {e}"

def query\_gemini(prompt: str, model="gemini-1.5-flash"):

"""Query Google Gemini models"""

try:

model = genai.GenerativeModel(model)

response = model.generate\_content(prompt)

return response.text

except Exception as e:

return f"Google Gemini Error: {e}"

def query\_huggingface(prompt: str, model="google/flan-t5-xl"):

"""Query Hugging Face models via API"""

try:

API\_URL = f"https://api-inference.huggingface.co/models/{model}"

headers = {"Authorization": f"Bearer {HUGGINGFACE\_API\_KEY}"}

payload = {"inputs": prompt}

response = requests.post(API\_URL, headers=headers, json=payload)

data = response.json()

if isinstance(data, list) and "generated\_text" in data[0]:

return data[0]["generated\_text"]

return str(data)

except Exception as e:

return f"Hugging Face Error: {e}"

# -----------------------------

# Unified Comparison Function

# -----------------------------

def compare\_ai\_responses(prompt: str):

"""Generate and compare responses from all configured AI tools"""

print("\n🔹 Input Prompt:", prompt)

print("-" \* 70)

results = {}

# Run across multiple AI tools

results["OpenAI GPT"] = query\_openai(prompt)

results["Anthropic Claude"] = query\_anthropic(prompt)

results["Google Gemini"] = query\_gemini(prompt)

results["Hugging Face"] = query\_huggingface(prompt)

# Display Results

for tool, output in results.items():

print(f"\n🧩 {tool} Response:\n{output}\n{'-'\*70}")

return results

# -----------------------------

# Example Execution

# -----------------------------

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

test\_prompt = "Explain blockchain technology in simple terms suitable for school students."

compare\_ai\_responses(test\_prompt)

Sample Output

🔹 Input Prompt: Explain blockchain technology in simple terms suitable for school students.

----------------------------------------------------------------------

🧩 OpenAI GPT Response:

Blockchain is like a digital notebook that everyone can see but no one can erase...

🧩 Anthropic Claude Response:

Imagine a shared online diary where every page is locked once written...

🧩 Google Gemini Response:

Blockchain is a secure digital record that helps people trust online transactions...

🧩 Hugging Face Response:

A blockchain is a chain of data blocks linked together that ensures security...

## 🧱 ****Advantages of Multi-AI Code Development****

✅ **Cross-Model Benchmarking:** Compare reasoning, creativity, and factual accuracy.

✅ **Unified Testing Environment:** Use one framework to test prompt performance.

✅ **Supports Report Generation:** Ideal for prompt engineering research projects.

✅ **Reusability:** Easily integrated into GUI apps or data pipelines.

## 🔒

## ****Security & API Key Management****

Always store API keys as **environment variables** (never hardcode).

Use .env files or a secure key vault.

Example for Linux/macOS: