# MASTER BLUEPRINT:

Bayesian Marketing Analytics Template

#### This is a modular, client-adaptable, professional-grade system built to:

* Ingest live marketing data (Meta, LinkedIn, GA4)
* Perform Bayesian inference on ad performance
* Generate automated reports with GPT-4o insight commentary
* Export beautifully designed PDF or HTML reports
* Scale across multiple clients via config files

### Folder Strśucture

Bayesian\_Ads\_Template/

│

├── bayesian\_ads.ipynb ← Main interactive notebook (template interface)

├── main.py ← CLI entry point (optional)

│

├── /modules

│ ├── data\_loader.py ← APIs: Meta, LinkedIn, GA4

│ ├── bayesian\_model.py ← Core Bayesian logic (priors, posteriors)

│ ├── visualization.py ← All plotting: distributions, comparisons

│ ├── reporting.py ← PDF & HTML generation (WeasyPrint/ReportLab)

│ └── openai\_client.py ← GPT-4o insights & narrative generator ✅ NEW

│

├── /configs

│ ├── default\_priors.yaml ← Prior settings (by platform or use case)

│ ├── client\_savvly\_meta.json ← Data structure map for a specific client

│

├── /outputs

│ ├── Savvly\_Report.pdf

│ ├── narrative\_savvly.txt ← GPT-generated narrative

│ └── /charts

│ ├── posterior\_adA.png

│ └── ...

│

├── /utils

│ ├── helpers.py ← Shared utilities

│ └── auth.py ← Token mgmt & API credential loading

│

├── requirements.txt ← Dependencies for pip install

├── .env ← OpenAI keys & secrets (never commit this!)

├── README.md ← Setup instructions, usage

└── LICENSE ← Your chosen open-source license

**New Module: openai\_client.py**

**Responsibilities:**

* Accept structured data and posterior results
* Send it to GPT-4o with an expert system prompt
* Return client-facing narrative ready for export

**Function Example:**

python

CopyEdit

def generate\_insight\_narrative(data\_summary: dict, tone: str = "Professional") -> str:

import openai

import os

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

SYSTEM\_PROMPT = f"You are an expert Bayesian marketing analyst. Your tone should be {tone}. Use plain language, and explain results clearly for business decision-makers."

messages = [

{"role": "system", "content": SYSTEM\_PROMPT},

{"role": "user", "content": f"Analyze the following Bayesian ad results:\n\n{data\_summary}"}

]

response = openai.ChatCompletion.create(

model="gpt-4o",

messages=messages,

temperature=0.7,

max\_tokens=700

)

return response.choices[0].message.content.strip()

**🔄 Data Flow Summary**

plaintext

CopyEdit

+--------------------+

| Meta / GA / LinkedIn|

+--------------------+

↓

[ modules/data\_loader.py ]

↓

[ modules/bayesian\_model.py ]

↓

+----------------------------------------+

| Posterior summaries + statistical plots|

+----------------------------------------+

↓

[ modules/openai\_client.py ]

↓

GPT-4o narrative insights (text string)

↓

[ modules/reporting.py ]

| PDF + HTML Export

**🔐 Security & API Management**

* All API keys stored in .env
* Use python-dotenv to load them:

bash

CopyEdit

pip install python-dotenv

* Example .env:

ini

CopyEdit

OPENAI\_API\_KEY=sk-...

FB\_API\_TOKEN=...

GA\_CLIENT\_ID=...

**🧪 Testing & Maintenance**

* **Each module** has a test stub for local runs (e.g., if \_\_name\_\_ == "\_\_main\_\_":)
* **Client configs** can be rotated in via CLI, UI, or Jupyter form field
* **Reports** will show:
  + Posterior means & credible intervals
  + Decision recommendations
  + GPT-generated explanation section