# **ZeroCode AI/ML Internship – Technical Brief**

### **Project: Multimodal Identity Extractor & Visual Theme Config Generator**

## **1. Executive Summary**

This document outlines the solution approach for the ZeroCode AI/ML Internship assignment. The objective is to build a multimodal pipeline that extracts structured metadata from logos, PDFs, theme images, and text-based personas using OCR, vision models, and large language models (LLMs) through LangChain agents.

The system takes raw inputs like images, design PDFs, and persona descriptions and generates a **structured identity vector** or **UI configuration**. It leverages tools like OpenCV, CLIP/BLIP embeddings, prompt engineering, and LangChain-based agent workflows securely wrapped using LLM API keys.

## **2. Task 1: Multimodal Identity Extractor**

### **Inputs (assumed):**

* Logo image (e.g., futuristic tech-style logo)
* Design PDF (with fonts, colors, themes)
* Customer persona snippet:  
   "Tech-driven entrepreneur targeting innovation-first users with a clean modern aesthetic."

### **Steps:**

1. **Preprocessing:**
   * Convert PDFs to image format using pdf2image.
   * Denoise logos with OpenCV and resize to standard dimensions.
2. **Text Extraction (OCR):**
   * Use pytesseract or unstructured for extracting stylized text from PDFs and images.
3. **Visual Semantics (Embeddings):**
   * Use **CLIP**, **BLIP**, or **DINO** to generate image embeddings that encode style, layout, and theme.
4. **LangChain Agent:**
   * Consolidates OCR text, visual embeddings, and persona using a multi-tool agent.
   * Wraps LLM (like OpenAI/Gemini) using LLMChain and an API key via environment variables.
5. **Output – Identity Vector (JSON):**

json

CopyEdit

{

"brand\_keywords": ["modern", "AI-first", "clean"],

"dominant\_font": "Inter",

"primary\_colors": ["#1F1F1F", "#00FFCC"],

"style\_attributes": ["flat design", "minimalist", "high contrast"],

"persona\_theme": "tech-forward, innovation-focused"

}

## **3. Task 2: Visual Theme Interpreter & Config Generator**

### **Inputs (assumed):**

* Theme Image: neon purple futuristic UI screenshot
* Prompt: "Interactive futuristic dashboard with glowing neon accents"

### **Steps:**

1. **Preprocessing and Color Extraction:**
   * Extract dominant colors using OpenCV (k-means clustering).
   * Identify layout patterns (grid, modular, card-based).
2. **Vision + Prompt Interpretation:**
   * Use **CLIP/DINO** embeddings + text prompt with a LangChain agent.
   * Map abstract style words like "cyberpunk" or "neon" to actual color/mood libraries.
3. **LangChain + LLM Usage:**
   * Use LangChain’s LLMChain with secure API key injection to handle prompting and reasoning.
4. **Output – UI Theme Config (JSON):**

json

CopyEdit

{

"theme": "neon\_dashboard",

"background": "#0D0D0D",

"accent\_colors": ["#9F00FF", "#00FFC6"],

"typography": "Orbitron",

"layout": "grid-based",

"button\_style": {

"type": "glow",

"padding": "10px",

"rounded": true

},

"mood": "futuristic, immersive"

}

## **4. LangChain Agent Workflow & API Key Integration**

### **Agent Objective:**

To unify insights from image embeddings, OCR-extracted text, and human-written prompts to generate structured outputs.

### **Workflow Diagram (conceptual):**

pgsql

CopyEdit

[Image Input] → [Embedding Tool (CLIP)] →

\

[PDF Input] → [OCR Tool (Unstructured)] → → [LLM Prompt Wrapper] → [Identity/Config Output]

/

[Persona/Prompt] → [LLMChain Agent] →

### **LLM Integration via API Key:**

python

CopyEdit

import os

from langchain.chat\_models import ChatOpenAI

from langchain.chains import LLMChain

os.environ["OPENAI\_API\_KEY"] = "sk-..." # Keep it secure in .env file

llm = ChatOpenAI(model="gpt-4", temperature=0.7)

chain = LLMChain(llm=llm, prompt=your\_prompt\_template)

## **5. Key Challenges and Resolutions**

| **Challenge** | **Solution** |
| --- | --- |
| **1. Abstract visual concepts (e.g., “cyberpunk”)** | Used CLIP to semantically match keywords to image embeddings |
| **2. Noisy PDF design elements** | Filter using OpenCV and segment by layout zones |
| **3. API security** | .env file for API keys + python-dotenv |
| **4. Font detection from images** | Fallback to design document or infer based on OCR context |
| **5. Prompt ambiguity** | Crafted example-based prompts and structured output templates |

## **6. Conclusion**

This approach effectively solves the ZeroCode Internship tasks by merging powerful vision-language models with structured LLM workflows. By combining image understanding (CLIP/BLIP), OCR extraction, and persona-aware reasoning through LangChain, the system produces reliable, structured outputs.

The secure integration of LLMs using API keys and prompt templates ensures flexibility and scalability. The architecture can be extended to automate UI generation, brand audits, or personalized interfaces in real-time systems.