## 🧠 Task 9: Service Analyzer

---

In this task, we'll practice using AI to integrate external inputs, extract structured insights, and generate multi-perspective analysis reports through a custom API-powered system.

---

### 📚 Theory

The OpenAI API gives you access to state-of-the-art language models capable of understanding and generating human-like text based on structured prompts. For this challenge, the API plays a central role in transforming raw service descriptions or names into rich, multi-perspective reports.

Unlike traditional tools that require rigid rule-based parsing or domain-specific knowledge, the OpenAI API understands context, makes inferences, and generates content in natural language—making it ideal for extracting structured insights from unstructured input.

When building the Service Analyzer console app, you'll use the API to:

- Interpret vague or loosely structured descriptions (e.g., "Our platform helps creators monetize content") and extract meaningful business, technical, and user-facing details.
- Generate markdown-formatted reports with well-organized sections, saving you time on formatting and phrasing.
- Apply reasoning to infer information not explicitly stated (e.g., deducing a product's audience or monetization model from context).
- Support both known services (by prompting the model to recall relevant public knowledge) and raw text (by guiding the model with few-shot examples).

The API also supports **few-shot learning**, allowing you to include 1–3 well-structured examples in your prompts to teach the model how to format and analyze new inputs. This drastically improves consistency and quality across outputs.

By combining natural language understanding, structured generation, and reasoning capabilities, the OpenAI API becomes your core engine for intelligent analysis—freeing you from manually parsing content or writing hard-coded logic for each scenario.

In this challenge, your goal isn't just to build an AI-powered tool—it's to **learn how to guide the model effectively** using input structure, few-shot examples, and output constraints. This skill will transfer to a wide range of real-world applications where structured content needs to be extracted from messy, ambiguous input.

---

### 🧠 AI Technique

This challenge combines several modern AI prompting techniques into a cohesive real-world task, including:

#### 🧩 Information Extraction with AI

You'll prompt the model to interpret diverse text formats (from marketing blurbs to user reviews) and distill relevant facts from noise. This includes detecting timelines, audiences, and feature lists even when they're not explicitly formatted.

#### 💻 Console-Based AI Workflow

You'll build a lightweight command-line tool. This reinforces how AI can be integrated into everyday developer workflows—making it easier to automate insights without complex infrastructure.

#### 📄 Structured Output Generation

The report must follow a **markdown-based format**, containing predefined sections (like "Tech Stack Insights" and "Business Model"). You'll guide the AI to

produce clean, readable, and sectioned responses—ensuring it behaves like a reliable summarization tool.

---

## 🛠️ Task

In this task, your goal is to create a **lightweight console application** that can accept service or product information and return a comprehensive, **markdown-formatted report** from multiple viewpoints—including business, technical, and user-focused perspectives.

This task simulates a real-world scenario where product managers, investors, or prospective users want quick, structured insights about a digital service. You'll build a system that uses AI to extract and synthesize relevant information from provided text (like an "About Us" section) or from known service names.

Your challenge is to implement a working console application that accepts either:

- A known service name (e.g. “Spotify”, “Notion”), or
- Raw service description text

... and returns a **markdown-formatted multi-section analysis report**.

---

## 🎯 Objectives

### ☑️ Build a Console App

- Accept user input via command line (either service name or text)
- Implement prompt logic to guide the AI
- Output the generated markdown report in the terminal (or to a file)

### ☑️ Analyze and Generate Report

- Use AI to process the input text or recognized service
- Ensure the output includes all required sections (see below)

---

📄 **Your report must include:**

- **Brief History**: Founding year, milestones, etc.
- **Target Audience**: Primary user segments
- **Core Features**: Top 2–4 key functionalities
- **Unique Selling Points**: Key differentiators
- **Business Model**: How the service makes money
- **Tech Stack Insights**: Any hints about technologies used
- **Perceived Strengths**: Mentioned positives or standout features
- **Perceived Weaknesses**: Cited drawbacks or limitations

---

📌 **Requirements**

- The app must include a call to **OpenAI API** using your API key
- The app must accept input from the **console** (text or known service name)
- README.md should contain **clear and detailed instructions** on how to run the application
- sample_outputs.md must include **at least two sample runs** of your application
- Output must be **clear, properly formatted**, and align with all the requirements stated in task description