# SWE - Applied AI

## Take Home Exercise

### Overview:

This take-home has two complementary parts:

1. **Pipeline Build:** Create an end-to-end, reproducible pipeline that ingests raw data and transforms them into structured, decision-ready outputs. Think of this as the “production path” that leadership could run every day.
2. **Data Analysis & Communication**: Run one small, controlled experiment to test an assumption about the pipeline, then present the question, method, results, and recommendation in a concise deck. This section shows how you reason with metrics and translate findings into action.

### What we are trying to evaluate:

- This exercise is intentionally hybrid: we want to see how well you can blend classical machine-learning techniques with modern GenAI capabilities.
- Beyond raw accuracy, we’re gauging your engineering discipline, your ability to design , run focused experiments, and how clearly you can articulate complex findings.
- In short, we care about pragmatic end-to-end thinking: from data prep, design, technical implementation, to effective technical communication that drives product direction.

# Part1 - Pipeline Build

## Problem Statement:

The Company PrimeApple's two flagship devices -EchoPad and EchoPad Pro - generate thousands of public reviews, yet leadership lacks a clear, data-driven picture of what customers are praising or struggling with. Your mission is to build a self-contained insight pipeline that ingests these raw reviews, groups them into a small set of data-backed themes, attaches sentiment and volume metrics, and produces concise summaries. The end result should let executives see, at a glance, which issues or opportunities to tackle next.

## Required Output:

Given a dataset containing reviews of these two products, develop a script that produces a ranked list of ~5–10 themes of reviews spanning both products. For each theme provide:

- Title (≤ 8 words, plain English customers would understand)
- Short explanation (≤ 50 words)
- Volume (% of total reviews the theme covers)
- Sentiment mix (% positive / negative / neutral)
- 3 representative customer quotes (verbatim snippets)

## Technical Requirements:

| Requirement (in order of pipeline) | Details |
|---|---|
| **Semantic representation** | 1. Represent each review in a semantic vector space (your choice of model; CPU-friendly is fine).<br>2. Persist representations to enable iterative runs. |
| **Theme Discovery** | 1. Group reviews into themes using a reasonable clustering method (or) some other method.<br>2. Keep runs **deterministic** (seeded or low-temperature setting). |

| | |
|---|---|
| **Theme summaries & titles (LLM prompting)** | 1. Generate the theme title and short explanation with a prompt that is concise and consistent.<br>2. Showcase solid prompt design/context engineering:<br>  a. Think about what makes a summary good vs bad.<br>  b. How could you get the LLM to achieve this quality?<br>  c. Explain this in the README.<br>3. Make LLM hyperparameters configurable (config file or CLI flags for model, temperature, top_p, etc.). |
| **Sentiment & Exemplars** | 1. Compute per-review sentiment. Aggregate to per-theme **pos/neg/neutral**.<br>2. Come up with a strategy to pick representative quotes automatically. Explain this in the README. |
| **Storage Schema** | 1. Identify what needs to be persisted from the pipeline and the right data models. Refer to the suggested tables section. |

**Suggested tables:**

*(You don’t need to stick to these. Feel free to adapt or extend the schema to fit your design.)*

None

1. clusters – Stores the canonical view of each discovered theme: which product it belongs to, its latest summary metrics (sentiment mix and share-of-voice), and a timestamp so changes can be tracked over time.

2. quotes – Could holds the individual customer snippets that exemplify each theme, preserving their original rating and creation date to enable drill-downs or fresh sampling without re-scraping.

3. runs - Logs every execution of the pipeline, recording the timestamp, LLM model and hyper-parameters used, total tokens consumed, and any notes on data scope or config changes. This audit trail lets you compare quality-versus-cost across runs and reproduce results later.

# Part2 - Data Analysis and Communication

## Required Output:

Run **ONE** focused experiment likely to influence how we'd ship this. Examples:

***Note: Choose ONE from this list or propose your own***

1. Assuming you use k-means clustering, you might explore clustering granularity.
    a. For example, testing different values of k and comparing silhouette scores with the perceived readability of the resulting themes.
2. If you decide to try different embedding models, you could compare two of them.
    a. Ex: Measure the impact on metrics such as silhouette score or the number of negative-silhouette points (under the assumption that you are using clustering)
3. If you experiment with different prompting styles for summarising clusters, you could compare a direct summary prompt with a brief chain-of-thought prompt and assess groundedness percentages and token usage.
4. If you want to explore hyperparameter effects, you might try varying the temperature (e.g., 0.0, 0.3, 0.6, 0.9) and see how it influences diversity, and token counts.

This is the expected presentation format: **hypothesis → design → results (table/plot) → recommendation**.

# Deliverables Checklist

- Runnable artifact (CLI/notebook/dashboard)
- **README** with setup & additional commands.
- **Outputs:**
    - Instructions on how to get all the identified themes and the following for each theme:
        - Title (≤ 8 words, plain English customers would understand)
        - Short explanation (≤ 50 words)
        - Volume (% of total reviews the theme covers)
        - Sentiment mix (% positive / negative / neutral)
        - 3 representative customer quotes (verbatim snippets)
    - **Mini-presentation** for your experiment with the following slides:
        - Hypothesis
        - Design
        - Results
        - Recommendation

**What we'll provide**: reviews.csv

**Submission**: GitHub link or zip