HTX AI Engineering Take Home Test

Important Notes

- Please follow all the instructions and requirements provided in this document.
- You can make assumptions to any information not specified.
- You can use any AI coding assistant or tool to do this test but you are expected to check the correctness of the solution and be familiar with the code.
- Use Git to manage and track changes of your source code.
- Please complete the test and submit the required deliverables by email within 3 calendar days upon receiving the test. If you are unable to complete within 3 days, you can inform us by email that you require up to 2 more days.
- If you have any query, reach out to us via email:
  - Jing Shen: tai_jing_shen@htx.gov.sg
  - Nicholas Lim: nicholas_lim@htx.gov.sg
  - Vincent Goh: vincent_goh@htx.gov.sg

Background

As an AI engineer in the xDigital AI Products Team, you will be responsible for developing AI/ML capabilities for software products. You need to have a good understanding of LLM, prompt engineering, and agentic AI. The ability to learn new AI/ML technologies is also important in this role.

Overview

This assessment is designed to evaluate both the technical competency and practical experience of AI Engineer candidates. You will build an AI-powered solution that extracts structured information from PDF documents, integrates external tools, and orchestrates an agentic system to analyze the document.

The test is divided into three parts of increasing complexity. Each part builds upon the previous one. While you are not expected to complete every part fully, we encourage you to attempt as much as possible and demonstrate your problem-solving skills, design decisions, and technical implementation.

You may use any LLM provider, such as Gemini, which offers free access of their API. You may create your solution using Jupyter notebooks.

Part	Description	Score Weightage	Expected Difficulty
Part 1	Document Extraction & Prompt Engineering	40%	Easy
Part 2	Tool Calling & Reasoning	40%	Moderate
Part 3	Agentic AI with LangGraph	20%	Moderate - Hard

Note: Our primary goal is to observe how you approach the problem, apply your knowledge of AI/LLM, and balance practicality with technical soundness. Do your best — even partial solutions can demonstrate strong engineering thinking.

Part 1: Document Extraction & Prompt Engineering with LangChain (40%)

Objective: Design effective prompt engineering strategies to extract structured information from unstructured documents. Link to data source.

Task:

1. **Parsing**
    - Load the document linked above in the data source using one or more libraries of your choice (e.g. Mark-it-down, Docling, PdfPlumber, PyMuPDF, LangChain loaders or any others).
    - Justify your choice of parsing approach(es).

2. **Prompt Engineering for Extraction**
    - Use LLM prompting to extract the following fields:
        - `Amount of Corporate Income Tax in 2024` (float)| (page 5)
        - `YOY percentage difference of Corp Income Tax in 2024`(float)| (page 5)
        - `Total amount of top ups in 2024` (float) | (page 20)
        - `List of taxes mentioned in section "Operating Revenue"` list(string) | (page 5-6)

- `Latest Actual Fiscal Position in billions` (float) | (page 8)

Part 2: Tool Calling & Reasoning Integration (40%)

Objective: Extend your Part 1 solution by integrating external tools and LLM reasoning. This task evaluates your ability to combine structured data extraction with intelligent decision-making using prompt engineering and function calling.

Task:

1. **Data Extraction + Datetime Tool via local MCP**
    - **If you are not able to implement an local MCP, you may define as a function using the appropriate tool decorator.**
    - Normalize submission dates extracted in Part 1 into ISO format (YYYY-MM-DD). Your final output for this step should be a list of these normalized dates.

    Information to Extract & Normalize:

    - The document's distribution date (page 1).
    - The date relating to estate duty (page 36).

2. **Reasoning over Normalised Date**
    - Using the normalized dates from the previous step, write a prompt for an LLM to reason and classify each date against the date of **`2024-01-01`**.

      The LLM should categorize each date into one of three states:

      - `Expired`: The date has already passed.
      - `Upcoming`: The date is in the future.
      - `Ongoing`: The date refers to a period that is currently active.

Sample output format:

[{ "original_text": "Distributed on Budget Day: 16 February 2024", "normalized_date": "2024-02-16", "status": "Upcoming" }....]

Part 3: Multi-Agent Supervisor (20%)

Objective: This part evaluates the ability to design and implement a multi-agent system using a supervisor pattern. Using the same data source, you will orchestrate a team of specialised agents to answer a complex query.

Task

Using a framework like LangGraph, implement a **multi-agent supervisor** that can answer the following user query by delegating sub-tasks to two specialized agents:

"What are the key government revenue streams, and how will the Budget for the Future Energy Fund be supported?"

You need to create and implement the following agents:

1. **Revenue Agent:**
    - **Role:** Specializes in identifying and extracting information on government revenue.
2. **Expenditure Agent:**
    - **Role:** Specializes in finding and analyzing information on government spending, including specific funds and sums to the correct figure.

Your submission must include:

- Various queries to demonstrate the combination of agents working together collaboratively to answer the query well.
- The code for your multi-agent system, including the supervisor and the two agents.
- Any assumptions to solve this problem must be provided.
- A clear trace of the supervisor's decision-making process, showing how it routes the query to the correct agents and synthesizes their responses to provide a final, comprehensive answer.

Part 4: Submit Deliverables

- Push your source code to a remote code repository like GitHub and make it publicly accessible. There should be a README file in your source code to document how to configure and run your application. Any documentation on system design, API, and justification of the use of code libraries and dependencies should be included in the README file as well.

- Submit your deliverables by emailing the link of your code repository and all necessary information (such as API keys) to the following:
  - Jing Shen: tai_jing_shen@htx.gov.sg
  - Nicholas Lim: nicholas_lim@htx.gov.sg
  - Vincent Goh: vincent_goh@htx.gov.sg