目標與要求

專案最終目標在搭建一 LLM 模型並完成開發 API 伺服器，以允許使用者能夠呼叫 API 對 LLM 模型進行 Prompt 且 LLM 模型通過 Response 回覆 Prompt 結果。

- 目標:
  - 搭建 LLM 模型:。
    1. 輸入: Prompt 文字。
    2. 輸出: 對於 Prompt 的回應
  - APIs: 這些 APIs 提供客戶端通過 Http 的方式使用 LLM 模型，需使用 Django 框架開發。
- 繳交:
  1. 操作手冊: 這個手冊應詳細說明如何搭建/測試您的系統，請使用 Step By Step 的方式說明。
  2. Docker Container: 請將您的系統建置在 Docker Container 內，並製作一個 start.sh，讓助教直接呼叫 start.sh 就可以開始測試您開發的系統。
  3. Codes: 原始程式碼，含 LLM Model & Architecture、API Server。
  4. 測試報告: 報告中應說明系統的執行結果。
- 驗收時間: 2024 年 X 月
- 程式碼要求: 請依照以下要求進行程式開發，以確保助教閱讀順利。
  - 命名: 檔案名稱、類別、方法、參數需準確命名。
    1. 私有元素請以小寫方式命名(如「types」)、公有元素請以首字大寫命名(如「Type」)。
    2. 若命名由多個單詞組成，請以下底線「_」進行命名。
       - 小寫命名範例: current_position
       - 首字大小命名範例: Current_Position
  - 重用性: 開發時請保持程式碼的重用性，避免相同或相似的程式碼功能獨立於不同的區域。
    1. 命名上，請在若程式碼為重用元件，請在程式碼命名後端加上能夠辨識的重用元件名詞。
    2. 重用性高的程式碼範例:
       - GUI_Base.py
         ∟Login_GUI.py (繼承 GUI_Base.py)
         ∟…
         ∟Shio_GUI.py (繼承 GUI_Base.py)
  - 結構化:
    1. 程式碼需由資料夾進行結構化管理。
       - 正確的資料夾管理範例:
         - ∟GUI
           ∟Command_Manager.py
           ∟Config_Loader.py
           ∟Displayer.py
           ∟GUIs
           ∟Login_GUI.py
           ∟…

└ Shop_GUI.py

2. 一個程式碼檔案僅包含一個類別。

系統規格

本次目標將由兩個系統組合而成，分別為「圖像生成模型」與「圖像生成 API 伺服器」，詳細規格說明如下所示。

1. LLM **模型:**

- 深度學習框架: PyTorch
- 輸入: Prompt 文字。
- 輸出: 對於 Prompt 的回應。

2. APIs

- 伺服器框架: Django
- 需求: 本 API 伺服器將同時會有多人存取，為此 API 回覆需即時，需以非同步執行的角度開發與理解。
- API 規格:
  1. 方法: POST
  2. 傳入參數格式: JSON
  3. 回傳結果格式:
     - API 執行成功: {status: true, data: 回傳結果}
     - API 參數錯誤: {status: false, message: 錯誤參數資訊}
     - API 執行階段發生錯誤: {status: false, failed: 錯誤原因}
  4. 伺服器啟動時，載入 LLM 模型。
- API 開發項目:
  1. Prompt: 對 LLM 模型進行 Prompt。
     - 傳入參數:
       - prompt: String，使用者的 Prompt。
     - 回傳結果:
       - response: String，針對使用者 Prompt 的回覆結果。
     - 範例結果:
       輸入: {prompt: “嗨，你好嗎?”}
       輸出: {status: true, data: {response: “你好! 很高興認識你!”}}