Alpha Catch 多變量時間序列分析平台

莊子毅、林博緯、林欣蓁、吳倩、與 賴柏伸(指導教授)

致理科技大學 財務金融系

摘要 (Abstract)

本研究旨在利用生成式人工智慧（Generative AI）協作開發一套全自動化之多變量時間序列分析平台。系統功能涵蓋從自動化資料收集、預處理，至建構關鍵階段轉換之機率預測模型。開發流程中透過「氛圍編碼（Vibe Coding）」模式，成功構建具備高度擴展性與模組化特質的通用分析架構。

本研究之實作階段雖以財金數列作為主要驗證對象，但核心開發重點聚焦於系統架構之穩健性與使用者體驗（User Experience, UX）。透過解構技術複雜度，本平台保留了極佳的擴展空間，使其能彈性應用於跨領域之時間序列視覺化、深度分析與預測模型建立，為非工程背景之領域專家提供直覺且強大之決策支援工具。

壹、研究背景與動機 (Background and Motivation)

一、時間序列分析的多樣化應用

多變量時間序列分析（Multivariate Time Series Analysis, MTSA）在現代商業與工業環境中扮演著關鍵角色。其應用範疇橫跨多個領域：

1. 金融投資領域:投資者需透過對複雜財經數據建立模型, 以精確捕捉市場訊號與關鍵轉折點。
2. 工業智慧化維護:動力機械透過監控輸出訊號進行預防性維護(Predictive Maintenance), 以降低非預期停機導致的系統風險。
3. 營運管理決策:管理者藉由數據收集機制識別特定異常狀態, 從而採取即時且有效的應變處置。事實上, 人類的日常生活與商業行為皆與「時間」維度緊密結合。分析隨時間變化的動態數據, 已成為現代決策中識別風險、發掘機會不可或缺的一環。

事實上, 人類的日常生活與商業行為皆與「時間」維度緊密結合。分析隨時間變化的動態數據, 已成為現代決策中識別風險、發掘機會不可或缺的一環。

二、數據洪流下的技術悖論

隨著資訊技術的演進與大數據理論的興盛, 企業記錄了海量的非結構化資訊。伴隨計算能力的飛躍式提升與智慧分析演算法的迭代, 理論上我們能獲得更多決策輔助。然而, 儘管擁有強大的運算資源與演算法支持, 一般民眾或非技術背景的決策者, 在摘取時間序列分析的實務成果時, 依然面臨極高的「技術門檻」。

三、人工智慧驅動的領域專家賦能

在傳統的資訊開發架構下, 實現數據分析系統往往需要高度依賴工程專家進行開發, 並由數據科學家進行精密的規劃。這種模式導致了「技術執行」與「領域專業(Domain Expertise)」之間的斷層。隨著生成式人工智慧(Generative AI)技術的成熟, 非工程背景的專業人員得以透過 AI 協作, 有效駕馭複雜的資訊技術。這不僅降低了開發成本, 更讓領域專家能將注意力集中在「數據智慧」的挖掘, 而非受限於程式邏輯的編寫。

四、本研究之目標

基於上述動機，本研究旨在透過大語言模型（LLM）與人工智慧技術的協助，建立一個以「自動化識別與標定特定時間區段資訊」為核心的智慧分析平台。本研究期望能優化人機協作流程，讓領域專家能更專注於問題解決，使雜亂無章的數據轉化為具備商業價值的智慧資產。

貳、相關研究 **(Related Works)**

一、財金指標之技術本質

在金融科技（FinTech）與量化交易的架構中，財金數列指標（Financial Time Series Indicators）不應僅被視為視覺化的圖表工具，其本質為訊號處理（**Signal Processing**）與特徵工程（**Feature Engineering**）的核心運算過程。

透過數學演算法，指標將原始且具備高雜訊的市場數據（包含開盤價、收盤價、最高價、最低價及成交量，即 OHLCV）進行轉換。此轉換過程旨在達成三大工程目的：提取市場規律、量化潛在風險以及預測價格趨勢。

二、指標之功能性分類

依據數學特性與解決問題之不同，本研究將常用指標劃分為以下四大類別：

1. 趨勢跟隨指標 **(Trend Following Indicators)**

- 功能與特性：主要用於過濾短期市場雜訊，以識別價格運行的主要方向。此類指標多具有「滯後性（Lagging）」，即訊號通常發生於行情啟動之後，但其對於趨勢確認的準確度較高。
- 代表指標：
    - 移動平均線 **(MA)**：為基礎的平滑工具，分為權重相等的簡單移動平均 (***SMA***) 與強調近期數據權重的指數移動平均 (***EMA***)。
    - 平滑異同移動平均線 **(MACD)**：結合趨勢追蹤與動能概念，常用於判斷趨勢延續或反轉的關鍵點位。

2. 震盪與動能指標 (Oscillators / Momentum Indicators)

- 功能與特性: 旨在測量價格變化的速度與幅度, 用以判斷市場是否處於「超買(Overbought)」或「超賣(Oversold)」狀態。此類指標具有「領先性(Leading)」, 能在價格反轉前發出警示, 但在強勢趨勢中易產生鈍化(假訊號)。
- 代表指標:
    - 相對強弱指標 **(RSI)**: 衡量特定期間內買盤與賣盤的力道比率, 數值區間為 0 至 100。
    - 隨機指標 **(KD)**: 比較收盤價與特定期間內價格範圍的相對位置。

3. 波動率指標 (Volatility Indicators)

- 功能與特性: 用於衡量市場的不確定性與價格變動劇烈程度, 為選擇權定價模型與風險控管系統之重要參數。
- 代表指標:
    - 布林通道 **(Bollinger Bands)**: 利用統計學標準差概念構建通道。通道收縮代表波動率低(盤整), 通道擴張則代表波動率放大(變盤)。
    - 平均真實波幅 **(ATR)**: 計算價格波動的平均幅度, 常作為演算法交易中設定止損點(Stop Loss)之依據。

4. 成交量指標 (Volume Indicators)

- 功能與特性: 基於「量是價的先行指標」之理論, 利用成交量數據確認價格趨勢的真實性與強度。
- 代表指標:
    - 能量潮 **(OBV)**: 將成交量量化為買方或賣方的累積力道。

- 成交量加權平均價 **(VWAP)**：機構投資人常用指標，用於評估成交價格優劣及市場平均持有成本。

三、**FinTech** 視角下的量化應用邏輯

與傳統技術分析不同，在金融科技領域中，指標的應用側重於自動化與數據標準化：

1. 特徵工程 **(Feature Engineering)**

在建構機器學習模型（如 LSTM, XGBoost）時，原始股價通常因不具備「平穩性（Non-stationary）」而不適合直接作為輸入。系統透過計算 RSI、MACD 或乖離率（Bias），將價格轉換為具有上下界限或圍繞零軸波動的數值特徵，提升 AI 模型學習市場模式的效率。

2. 演算法交易邏輯 **(Algorithmic Logic)**

量化交易系統將主觀的圖表型態轉化為明確的布林邏輯（Boolean Logic）。

- 傳統敘述：「黃金交叉時買進」。
- 程式邏輯：`if (SMA_short > SMA_long) and (prev_SMA_short < prev_SMA_long): Execute_Buy()`

3. 數據標準化 **(Normalization)**

為解決不同標的價格區間差異（如高價股與低價股）導致的模型偏差，本研究利用指標（如 Z-Score 或百分比變化的 MA）進行數據標準化，使單一模型能同時適用於多種資產類別。

四、自編碼器 **(Autoencoder)**

自編碼器（Autoencoder）是一種無監督學習的神經網路模型，用來學習資料的有效壓縮表示。

自編碼器分為編碼器（**Encoder**）和解碼器（**Decoder**）兩部分。編碼器將高維輸入資料壓縮成低維潛在表示，解碼器則從此表示重構出接近原始輸入的輸出，透過最小化重構誤差來訓練。

自編碼器的主要應用包括：

1. 降維與特徵提取：壓縮高維資料，類似PCA但非線性，適用於影像壓縮或特徵學習。
2. 去噪：訓練時加入噪聲，讓模型學會移除干擾，重構乾淨資料。
3. 異常偵測：正常資料重構精準，異常資料誤差大，用於故障檢測。
4. 生成模型：變分自編碼器（VAE）延伸，可產生新樣本，如 Deepfake 或新圖像。

五、深度學習**: Dense** 全連接層

全連接層（**Dense Layer / Fully Connected Layer**）是最基礎且最重要的組件之一。它的核心邏輯是：層中的每一個神經元，都與前一層的所有神經元相連。

在全連接層中，每個神經元會接收來自上一層的所有輸入訊號，並透過「加權求和」與「激活函數」後輸出結果。全連接層使用的參數包括：

- 權重 **(Weights, *w*):** 代表輸入訊號的重要性。
- 偏置 **(Bias, *b*):** 用來調整模型的偏移量。

- 激活函數 **(Activation Function):** 如 **ReLU** 或 **Sigmoid**, 賦予模型處理非線性問題的能力。

全連接層通常位於神經網絡的末端, 其角色演變如下:

1. 特徵組合: 在卷積層（**CNN**）提取出局部特徵（如線條、眼睛、輪子）後, 全連接層負責將這些零碎的特徵「拼湊」起來, 形成全局概念。
2. 分類器: 它將高維度的特徵映射到樣本標籤空間。例如, 在手寫數字辨識中, 最後一層通常是具有 10 個神經元的全連接層, 分別對應數字 0-9 的機率。
3. 降維或升維: 通過調整該層神經元的數量, 可以改變資料的維度。

優點

- 擬合能力強: 理論上可以擬合任何複雜的非線性函數。
- 結構簡單: 容易理解與實作。
- 全局感知: 能考慮到輸入特徵之間的全局關係。

缺點

- 參數過多: 每個連接都有權重, 容易導致模型體積巨大, 計算量高。
- 容易過擬合 **(Overfitting)**: 因為參數多, 如果數據不夠, 模型容易死記硬背訓練集。
- 丟失空間結構: 在處理圖像時, 必須先將圖像「拉直」(Flatten), 會破壞像素間的空間位置訊息。

參、系統設計與規劃 **(System Design)**

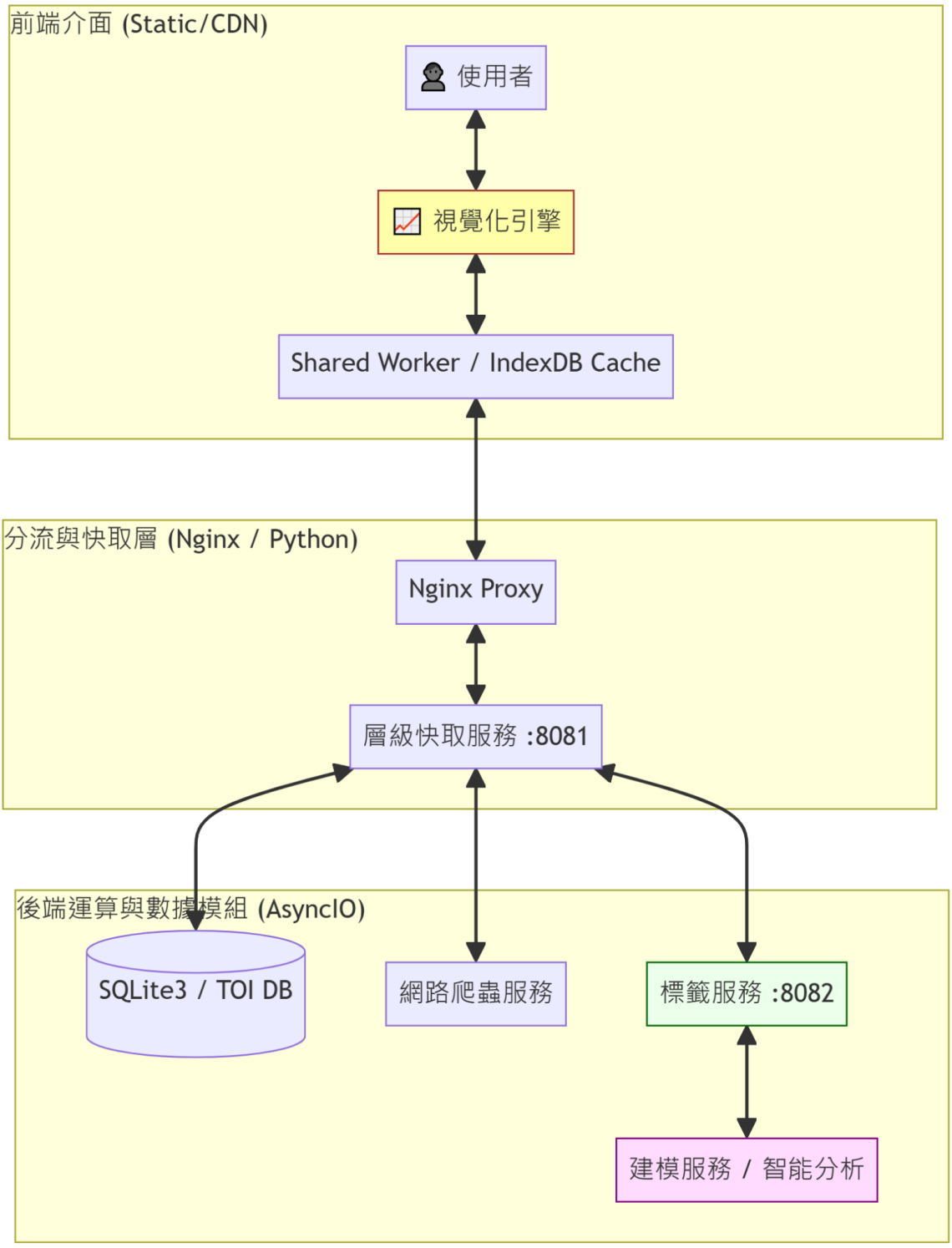
一、系統架構概要

採用多伺服器架構, 核心設計原則為前端高可用性 (High Availability) 與後端非同步 (Asynchronous) 運算。為符合資訊系統 CIA 三原則, 架構設計重點在於確保使用者端的最高可用性, 同時透過多重快取機制平衡數據的完整性。

1. 系統架構圖

如 (圖3.1.1) 所示, 從使用者到後端資料的取得與建立, 考慮到使用者體驗與效能的要求, 系統規劃了三層的架構。

[圖3.1.1] 系統架構圖


前端介面 (Static/CDN)
使用者
視覺化引擎
Shared Worker / IndexDB Cache
分流與快取層 (Nginx / Python)
Nginx Proxy
層級快取服務 :8081
後端運算與數據模組 (AsyncIO)
SQLite3 / TOI DB
網路爬蟲服務
標籤服務 :8082
建模服務 / 智能分析

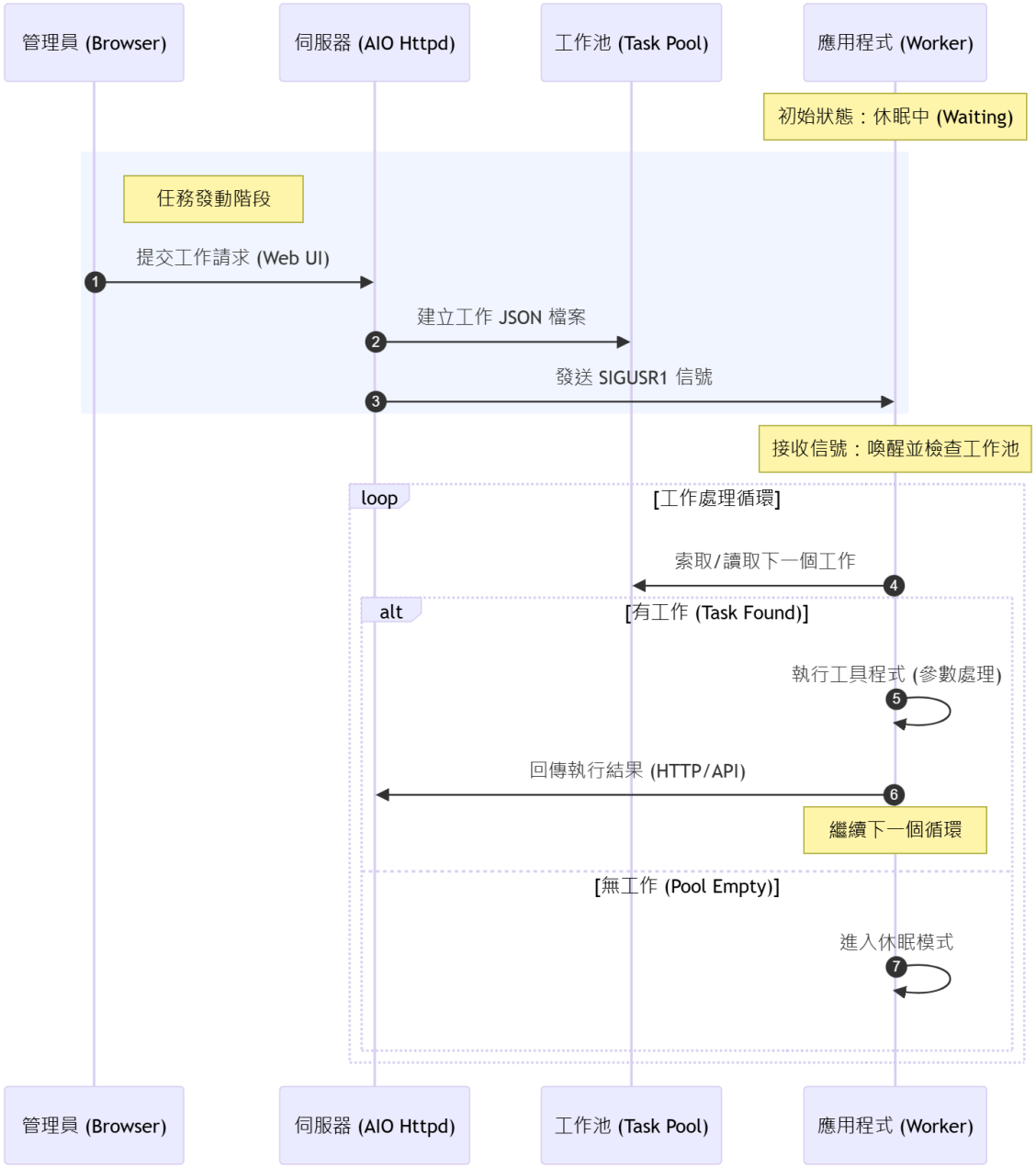
2. 服務實體類型定義

系統將服務劃分為三種主要實體類型：

1. 純客戶端運行 (Client-side Interactive)：
    - 對應模組：視覺化引擎服務。
    - 技術特性：採用 JavaScript, CSS 與 DOM 技術在使用者裝置端即時生成視覺效果。
    - 關鍵效益：運算壓力去中心化，可透過 CDN 達到全時服務不中斷 (24/7)。
2. 伺服器端網頁服務 (Server-side Hub)：
    - 對應模組：層級快取服務、標籤服務。
    - 技術特性：管理者介面與應用程式間的資料交換樞紐，處理同步請求並管理任務調度。
3. 純應用程式工具 (Background Workers)：
    - 對應模組：網路爬蟲服務、建模服務。
    - 技術特性：透過 SIGUSR1 信號驅動工作池 (Task Pool)，負責高負載的採集與訓練任務。

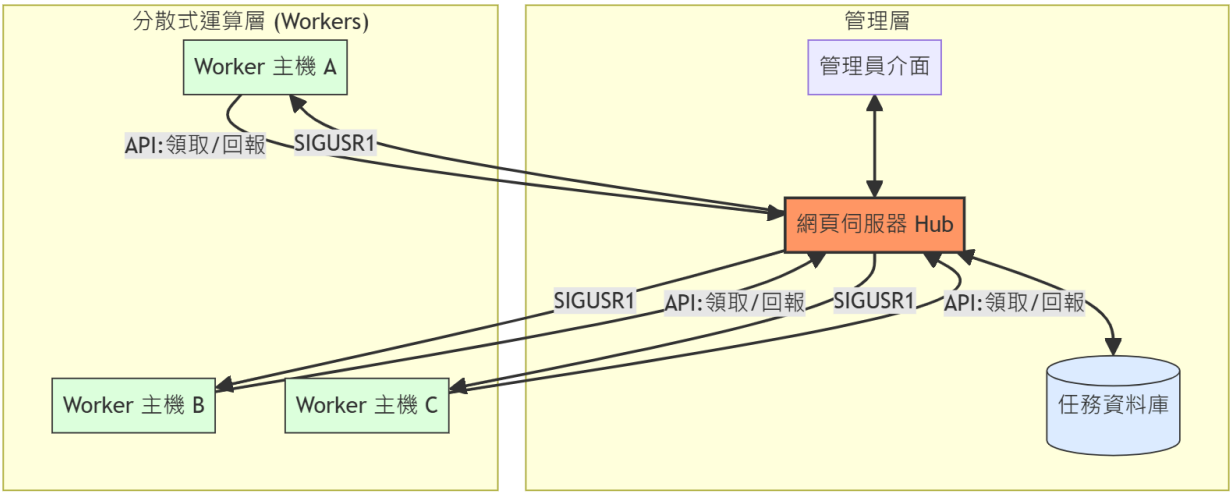
為兼顧可用性與 DevOps 的需求，伺服器端的工作皆採用 伺服器網頁服務 搭配 純應用程式工具 的方式進行。如 (圖 3.1.2) 所示，系統採用非同步的信號驅動機制。管理員透過網頁伺服器提交任務後，伺服器將任務實體化至工作池，並發送系統層級的 SIGUSR1 信號喚醒處於休眠狀態的應用程式。應用程式被喚醒後即進入處理循環，透過 Web API 索取任務細節並回報執行結果，直到工作池清空後再次進入休眠，以達成極低能耗且反應即時的任務處理。

(圖 3.1.2: 工作時序說明)


管理員 (Browser)
伺服器 (AIO Httpd)
工作池 (Task Pool)
應用程式 (Worker)
初始狀態：休眠中 (Waiting)
任務發動階段
提交工作請求 (Web UI)
1
建立工作 JSON 檔案
2
發送 SIGUSR1 信號
3
接收信號：喚醒並檢查工作池
loop
[工作處理循環]
索取/讀取下一個工作
4
alt
[有工作 (Task Found)]
執行工具程式 (參數處理)
5
回傳執行結果 (HTTP/API)
6
繼續下一個循環
[無工作 (Pool Empty)]
進入休眠模式
7
管理員 (Browser)
伺服器 (AIO Httpd)
工作池 (Task Pool)
應用程式 (Worker)

如此設計的好處是，當系統的支援漸趨完整，負載增加之時，服務將具有可延展性。如(圖3.1.3) 所示，藉由 Web API 實現的解耦合設計，網頁伺服器可演進為集中式的調度樞紐（Hub），管理分布於不同物理主機上的多個應用程式執行單元（Workers）。伺服器能透過統一的信號分發與 API 狀態監控，確保多個任務在不同的 Worker 主機間維持原子性操作與負載平衡，實現高效能的分散式運算架構。

(圖3.1.3: 多應用主機架構)



二、核心服務模組規格

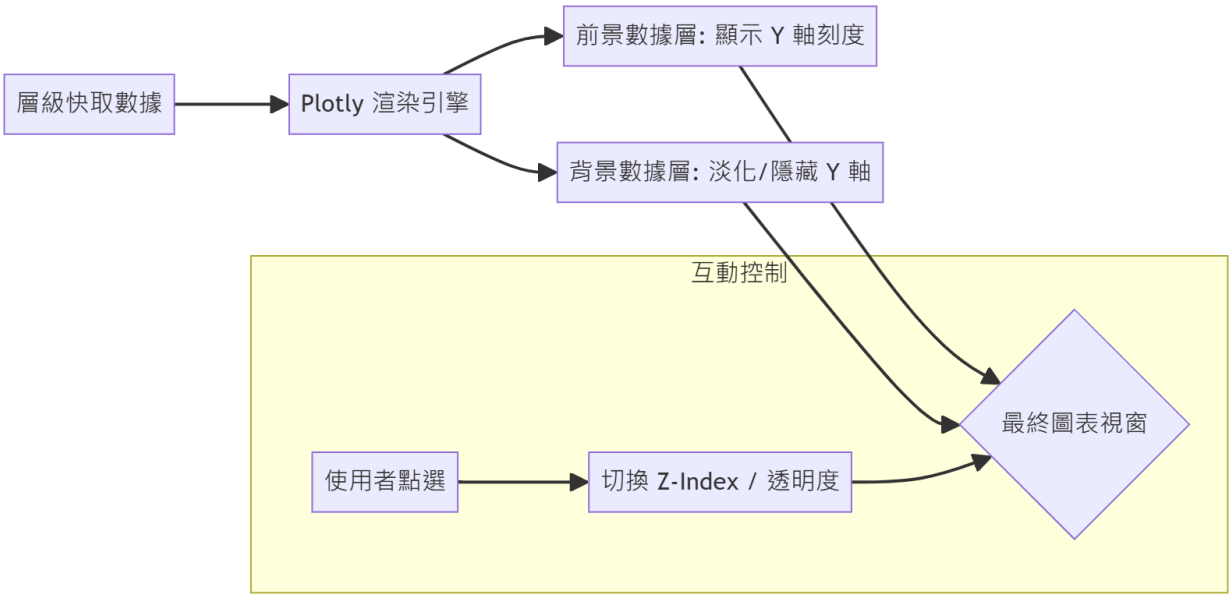
1. 視覺化服務 (靜態網站技術)

- 數據獲取：優先請求 CDN 靜態 JSON 資料，失敗時轉向數據快取服務請求。
- 功能要點：
    - 純客戶端運行 (Client-side Interactive)
    - 供使用者互動檢視與標定 Time of Interest (TOI)。
    - 支援人工標籤與機器學習預測標籤的對比檢視。
    - 多對象，多指標，多區段，前景背景集合疊合檢視 (圖 3.2.1)
    - 為簡化操作流程，系統採用「中央對齊標定法」。使用者僅需平移走勢圖使目標事件對齊視窗中央線，系統即透過下式自動換算數據索引：

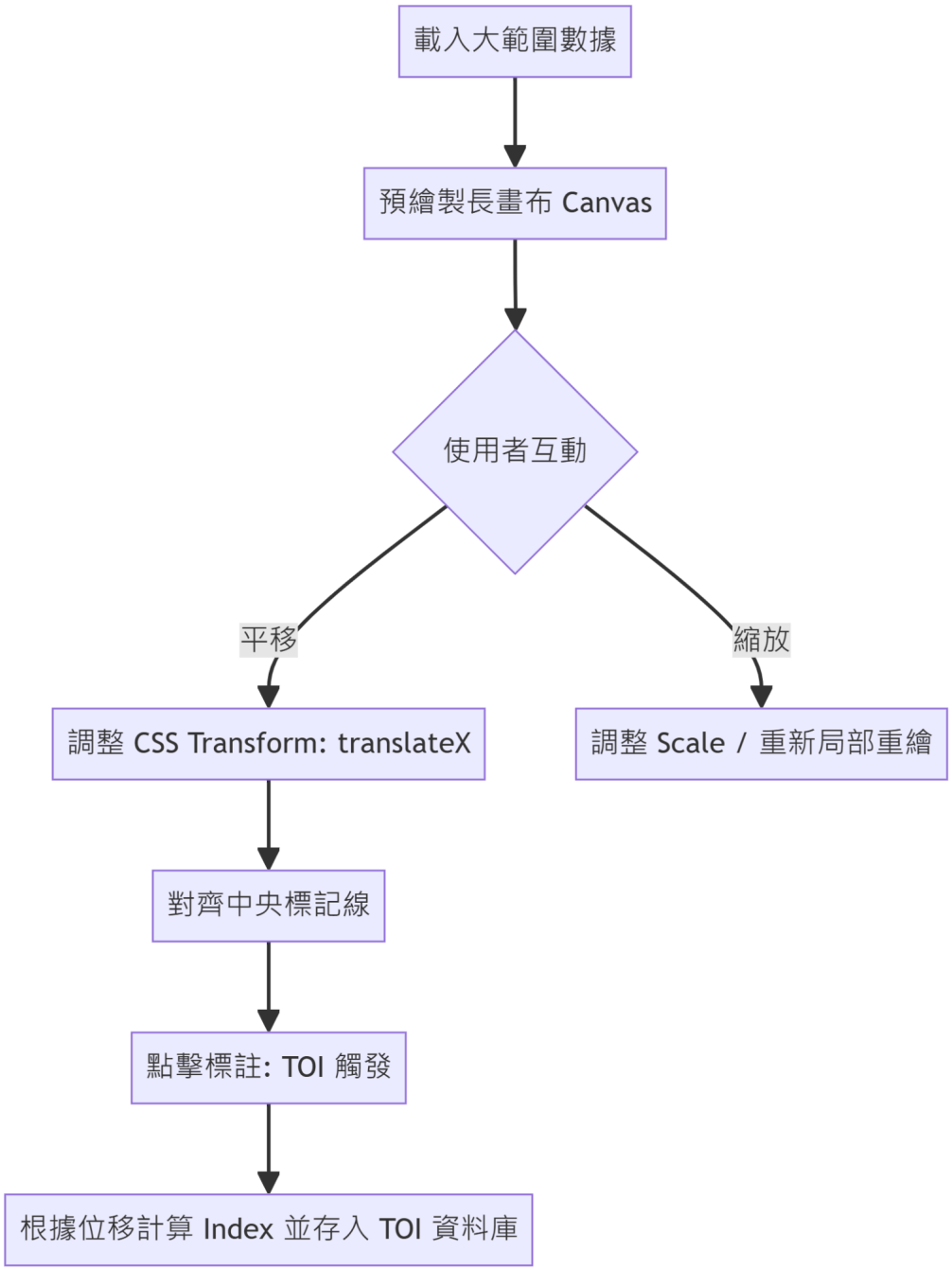
$$I_{toi} = \lfloor (X_{offset} + W_{view}/2) / P_{dist} \rfloor$$ ，其中

        - I_{toi}：數據索引 (Index)，
        - X_{offset}：畫布向左偏移像素值之絕對值，
        - W_{view}：可見視窗寬度，
        - P_{dist}：單位數據點像素寬度。
    - 單調操作，使用者僅需利用縮放與平移兩個功能就可將數據移動到關注區域內，並進行標籤的設定 (圖 3.2.2)。
    - 點擊檢視單一時間點數據資訊，釘選表格必較多組數據資訊。

(圖 3.2.1: 走勢圖工具, 異質多層次疊合方法。)



(圖 3.2.2: 走勢圖工具, 單純操作控制方法。)



2. 層級快取服務 (Leveling Cache)

- 多層快取機制:
    - **L1 (Client Side)**:利用瀏覽器 **IndexDB** 存儲已下載的年度資料段(Data Segments), 減少重複的網路請求。
    - **L2 (Server RAM)**:網頁伺服器(**AIO Httpd**)於記憶體維護近期熱門查詢的資料快取表, 提供微秒級的反應速度。
    - **L3 (Persistent Storage)**:伺服器端本地 **SQLite3** 資料庫, 作為指標數據與原始數據的永久存儲層。
- 例外處理與任務調度: 資料缺失時伺服器回覆 Syncing 標籤告知使用者, 並同步觸發後端爬蟲或計算任務。為了確保系統效率, 本服務具備「原子性請求合併 **(Atomic Request Joining)**」功能:若多個使用者同時請求同一個缺失的年度資料, 系統僅會派發一個 TaskID, 避免重複執行爬蟲任務對外部數據源造成壓力。
- 指標計算服務 **(Indicator Service)**: 負責計算資料的相關指標數據並快取之。系統採取「預計算 **(Pre-calculation)**」策略, 當原始資料更新時即觸發指標運算並存入 L3, 確保前端請求時能直接獲取完整指標, 無需重新運算。
- 數據有效性管理 **(Data TTL)**:
    - 歷史資料: 非當前年份之資料視為靜態唯讀, 可永久快取於各層。
    - 當前資料: 當前年度資料設定「生存時間 (Time-To-Live)」, 定期過期並觸發自動同步以維持資料即時性。
- 數據封裝格式:
    - 請求單位: 以年為基本單位。例如:***?2330_TW-2025*** 將傳回台積電 **2025** 年的資訊。

- 回覆規範: 回覆格式為包裹字典的串列(List of Dictionaries)。舉例來說:**[0, {"O":100, "C":105, "H":110, "L":95, ...}]** 表示該年度第二天的資料為 **100/105/110/95**。沒有交易資料之日期將以 0 代替, 以優化資料傳輸效能與解析成本。

任務描述實體與重試策略 (Task Schema) 本服務採用網頁伺服器搭配應用程式的框架實現。伺服器使用 **SIGUSR1** 訊號通知應用程式領取任務 JSON 描述。為應對網路不穩定等例外, JSON 結構中納入了錯誤重試機制:

```
{
  "TaskID": 1234,
  "Script": "yfinance",       // 指定執行工具 (Python/PlayWright/Shell)
  "Args": {                   // 工具參數設定
    "Target": "2330.TW",
    "Since": "2025-01-01",
    "Range": 365
  },
  "Retry": 3,                 // 最大重試次數
  "Callback": {               // 完成後回報之 API 路徑
    "URI": "/api/commit?taskID=1234",
    "Body": "{...}"           // 回報之數據內容封裝範本
  }
}
```

3. 標籤服務 (Labeling Service)

- 服務定位:本服務運行於本地主機(或雲端樞紐)的 8082 通訊埠, 作為前端「視覺化引擎」與後端「建模服務」之間的數據中繼站。其核心功能是將使用者產生的標註匯聚成結構化的學習資料。
- **TOI** 標籤管理:
  - 人工標註 **(Manual TOI)**: 接收來自前端透過「中央對齊標定法」產出的索引數據, 並記錄標註者資訊與標籤類別。

    - 規則產生 **(Rule based TOI)**：根據預定地規則，系統自動產生標定的標籤。規則套用在未來數據之上，使用訂定的規則可以產生大量的學習資料，供人工智能模型學習 TOI 發生前的數據。
    - 機器預測 **(AI Predicted)**：同步顯示由建模服務產出的模式辨識結果，供使用者進行比對與驗證。
- 衝突處理：當人工標註與機器預測重疊時，系統提供權重設定，優先採信人工標定之數據作為後續訓練的 Ground Truth。
- 管理與調度功能：
    - 模型訓練發動：管理者可透過標籤服務的管理頁面，定義訓練集範圍（如：特定年份、特定標的）並發送任務指令予建模服務。
    - 即時辨識開關：提供開關控制是否在視覺化前端顯示即時的模式辨識標籤。
- 多主機與權限擴展：
    - 貢獻者認證：具備資料貢獻者認證管理，確保 TOI 資料的來源可靠性與權限控管。
    - 分散式支援：支援多台應用主機（Workers）同時回報標籤結果，並由本服務進行統一的匯整與去重處理。

標籤數據交換格式 (Label Schema) 前端送出或後端回報標籤時，採用以下 JSON 結構：

```
{
  "Target": "2330.TW",
  "Type": "TOI",              // 標籤類型：TOI, Pattern, Event
  "Source": "Manual",         // 來源：Manual (人工) 或 AI_Model_V1 (機器)
  "Data": [
    {
      "Index": 125,           // 根據中央對齊法計算出的數據索引
      "Label": "Stage_3",     // 標註類別 (如：起漲點、盤整期)
      "Confidence": 1.0,       // 人工標註為 1.0, 機器標註則帶入模型信心分數
      "Timestamp": "2026-01-01T12:00:00Z"
    }
  ]
}
```

標籤服務與建模服務的協作邏輯如下:

- 資料收集: 視覺化服務將使用者標定的 TOI 傳送至標籤服務。
- 訓練請求: 管理者於標籤服務介面確認資料足夠後, 發動訓練任務。
- 異步建模: 標籤服務將任務寫入資料庫, 並發送 SIGUSR1 訊號通知建模服務領取任務。
- 結果更新: 建模服務完成後透過 API 回傳新標籤, 標籤服務更新資料庫並通知前端重新整理檢視。

三、智能分析模型:三階段決策框架

不限制特定學習引擎, 現階段採用 Autoencoder、Dense Classifier 與 Decision Tree 的混合架構。其核心設計理念在於利用無監督學習捕捉市場的常態模式, 並透過 Meta-Classifier 學習模型自身的偏誤, 以在極端行情下維持系統的魯棒性(Robustness)與極高精準度(High Precision)。

3.1 階段一:無監督特徵學習 (Unsupervised Feature Learning)

目標:透過 Seq2Seq Autoencoder 學習財金數據的「潛在表示」, 將高維度的時序特徵壓縮為具備語義信息的特徵向量。

- 數據準備:
  - 輸入 $\boldsymbol{X}$:採用 252 天(約一交易年)的滑動窗口, 包含 $\boldsymbol{N}$ 個參數(如 OHLC、成交量及技術指標)。
  - 預處理:進行時間序列歸一化(Normalization), 確保不同量級的指標(如股價與成交量)在同一尺度下運算。
- 網絡架構:
  - Encoder (編碼器):利用 LSTM 層處理時序連續性, 將序列 $\boldsymbol{X}_{252\times N}$ 壓縮至潛在空間向量 $\boldsymbol{z}$ (Hidden State)。
  - Decoder (解碼器):將 $\boldsymbol{z}$ 作為輸入, 透過 RepeatVector 重構出與原始輸入相同維度的序列 $\boldsymbol{X}'$。
- 損失函數:
  - 採用均方誤差(**Mean Squared Error, MSE**):

$$\boldsymbol{L}_{ae} = |X - X'|^2$$

- 重建誤差 **(Reconstruction Error)** 的價值：訓練完成後，該誤差可作為衡量「數據異質性」的指標。當誤差異常大時，代表當前行情脫離了歷史常態，即 Out-of-Distribution (OOD) 狀態。

3.2 階段二：有監督分類 (Supervised Stage Classification)

- 目標：利用階段一提取的高效特徵向量 z，進行特定市場階段（如起漲點、盤整期）的分類辨識。
- 模型結構：
  - 輸入層：接收來自編碼器的潛在向量 z。
  - 隱藏層：採用多層全連接神經網路（**Dense Layers**），並使用 **ReLU** 激活函數以捕捉非線性關係。
  - 輸出層：採用 **Softmax** 激活函數，將結果轉換為各類別的機率分佈

$$\boldsymbol{P} = [p_1, p_2, ..., p_k]。$$

  - 訓練邏輯：標籤來源於「2.3 標籤服務」中收集的人工標註 TOI。
- 損失函數採用類別交叉熵（**Categorical Crossentropy**），優化模型對特定模式的辨識能力。

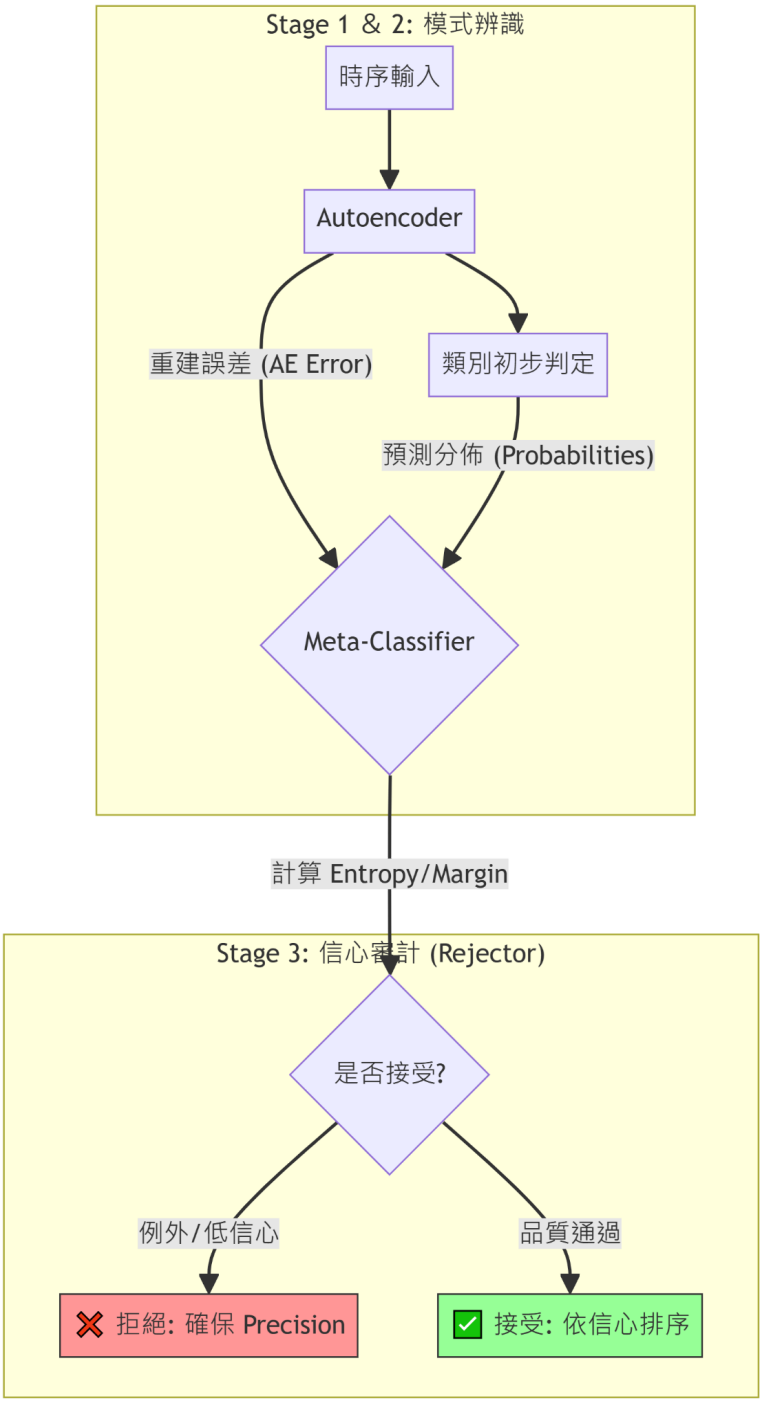
3.3 階段三：**Meta-Classifier** 品質監控 **(Confidence Scoring)**

- 目標：作為系統的「防禦門檻」，透過審計前兩階段的輸出，攔截潛在的誤判，確保輸出的極高精準度 (Precision)。
- 特徵工程 **(Meta-Features)**：從第一、二階段提取反映「不確定性」的數據作為輸入：
  - 最大機率值 **(Max Probability)**：預測類別的絕對信心分。
  - 預測熵值 **(Entropy)**：衡量類別分佈的混亂度。公式為 $H(P) = -\sum p_i \log(p_i)$

- 邊際差距 **(Margin)**：第一名與第二名機率的差值，反映分類邊界的模糊度。
- 重建誤差 **(AE Error)**：來自階段一，反映資料是否為極端例外狀況。

- 決策邏輯 **(Decision Tree)**：
  - 採用決策樹 (Decision Tree) 訓練「接受/拒絕」的規則。
  - 優勢：提供極佳的可解釋性 (Explainability)。例如管理者可查閱規則：「若 AE Error > 0.5 且 Entropy > 0.8，則攔截訊號」。
  - 系統決策矩陣如 (圖 3.3.1)
- 輸出機制：
  - **Reject (**攔截**)**：對於低信心或異常資料，系統不予輸出，以確保極高精準度。
  - **Rank (**排序**)**：對於通過審計的訊號，根據 Meta-Classifier 的信心分進行排序，產出投資優先順序名單。

(圖 3.3.1: 數據從輸入到最終決策的過濾流向)



肆、實作方法論 **(Methodology)**

研究採用「人工智慧協作驅動開發」(**AI-Augmented Development, AIAD**)流程, 結合提示工程(**Prompt Engineering**)與 迭代式原型設計(**Iterative Prototyping**), 以確保系統架構的嚴謹性與功能實作的精確度。具體實作路徑分為以下三個階段:

4.1 提示工程策略:由架構至功能之演化

為了有效引導大語言模型(**LLM**)產出符合預定系統架構的程式碼, 本研究採取「框架先行, 功能後置」的提示策略。

1. 結構定義階段:首要步驟不直接要求模型實作具體功能, 而是定義系統的「空白框架頁面」。在此階段, 重點在於確立軟體架構(**Software Architecture**)的邊界, 包含定義物件屬性、函式介面(**Function Signatures**)以及關鍵程式區塊的邏輯註解。
2. 邏輯注入階段:在確立了與預期規格一致的框架後, 再依循各功能模組的描述細節, 引導語言模型填充具體的實作代碼。此舉能有效降低模型輸出的隨機性(**Hallucination**), 確保最終產出符合既定的軟體工程規範。

4.2 迭代式前端介面開發

在前端介面的開發流程中, 本研究充分利用 ***Gemini Canvas*** 的即時協作特性, 採取「核心驅動, 元件增量」的實作方法:

1. 原型啟動:優先滿足系統最核心的功能需求(**Core Requirements**), 確保底層邏輯與資料流轉正確無誤。

2. 增量開發（**Incremental Development**）：透過描述性的指示，逐步向 人機介面 **(User Interface)** 加入互動式 UI 元件。
3. 即時驗證（**Real-time Validation**）：每一階段的 UI 更新皆伴隨即時的功能測試，確保新加入的元件不會干擾既有邏輯，達成開發與驗證的高度同步。

4.3 模組化設計與分段驗證機制

面對龐大且複雜的系統規劃，本研究採用模組化設計（**Modular Design**）作為開發的核心指導原則。

- 開發效益：透過模組化，將繁雜的業務邏輯拆解為可獨立運作的單元（Units），使得開發工作得以「分段完成、逐步驗證」。
- 系統穩定性：此種設計不僅提升了開發初期的除錯效率，更在系統後續的擴充性（Scalability）與接續開發中，扮演了維持結構一致性的關鍵角色。

藉由上述方法論，本專題成功建立了一套可重複、可預期且高效能的軟體開發流程，展現了商科學生在科技金融（FinTech）背景下，整合人工智慧工具與系統分析能力的實作素養。

伍、結論 **(Conclusion)**

5.1 研究成果與技術實作

本研究成功運用大語言模型（LLM）建構出一套具備高擴展性的「大型時間序列分析平台」。透過軟體工程中的模組化設計原則與生成式 AI 的深度協作，團隊驗證了在「低度後端源碼微調」的條件下，開發複雜大型系統的可行性。此成果不僅體現了技術的自動化，更證明了人工智慧在處理高維度時序邏輯時的優異潛力。

5.2 開發流程之典範轉移

本專題之開發流程突破了傳統開發模式的侷限，主要體現在以下兩個層次：

1. 需求精煉與視覺原型：研究初期導入敏捷開發中的使用者故事（User Story）訪談技術，由團隊成員輪流擔任「業務端（客戶）」與「技術端（工程師）」角色，確保需求擷取（Requirement Elicitation）的完整性。隨後，利用開源設計工具 Penpot 繪製詳細之介面分鏡圖（Storyboard），在程式實作前即確立了嚴謹的操作邏輯與視覺規範。
2. 人機協作下的「氛圍開發模式（**Vibe Coding**）」：進入實作階段後，團隊將 Gemini 作為核心協作夥伴，執行新興的「氛圍開發（Vibe Coding）」模式。透過結構化的規格文件與介面原型的引導，Gemini 不僅能精準轉化開發意圖，更能主動針對系統架構提出具建設性的優化建議，實現了高效、高質量的雙向協作。

5.3 研究貢獻與未來展望

本研究最重要的發現在於:透過大語言模型的輔助，系統開發的重心得以實現從「技術可行性（Technical Feasibility）」向「商業價值與使用者體驗（User Experience）」的典範轉移。

- 回歸問題本質:開發者得以將全數精力投入於商業模式考量、研究功能優化以及操作便捷性上。技術上的硬性侷限與遷就，不再是主導系統設計的決定因素。
- 未來趨勢:展望未來，資訊系統的開發將回歸「功能性」與「需求面」的本質考量。隨者 AI 協作技術的成熟，資訊系統將能以前所未有的深度貼近使用者，使技術開發真正服務於解決現實世界的商務與科學問題。

Alpha Catch 專題草稿審閱意見與建議

這份草稿在架構設計（System Design）與模型理論（3-Stage Framework）上已經非常完整。針對後續「成果部分」的填補，建議可以從以下幾個面向進行實作與撰寫：

1. 針對「肆、研究結果與分析」的撰寫建議

由於您的系統強調「極高精準度（High Precision）」，建議此章節包含以下數據：

- 模型效能數據：
    - 提供第一階段 Autoencoder 的 重構誤差分布圖 **(Reconstruction Error Distribution)**，證明模型能有效識別「常態」與「異質（OOD）」行情。
    - 展示第三階段 Meta-Classifier 介入前後的 精準度 **(Precision)** 對比。例如：攔截前精準度 65%，攔截低信心訊號後提升至 85%。
- 視覺化工具實績：
    - 放入幾張系統渲染後的 **Plotly** 走勢圖截圖（包含您提到的異質多層次疊合成果）。
    - 展示使用者透過「中央對齊標定法」產生的標籤數據分布。

2. 草稿內容的小提醒 **(**針對目前的 **PDF)**

- 圖表編號：PDF 第 9 頁提到的「圖 3.1.1」與後續圖號建議在最終版再次校對，確保與內文描述完全一致。
- 公式排版：第 14 頁的 I_{toi} 公式建議使用標準的 LaTeX 格式（如已在 Markdown 顯示則無礙，轉 PDF 時需注意）。

- 專有名詞一致性：文中提到的 "Vibe Coding" 是一個很棒的亮點，建議在「實作方法論」中可以多加一個小標題，描述具體是如何與 Gemini 進行連續性對話來修正 Code 的。

3. 接下來一個月的開發重點建議

- **L1 Cache** 壓力測試：確保 Shared Worker 在多分頁切換時，IndexDB 的資料讀取不會造成畫面卡頓。
- 標籤回饋閉環：優先完成「人工標記 -> 重新訓練 -> 機器預測更新」這個流程的端對端測試，這是系統最有價值的地方。

結語

目前的草稿已經具備優等專題的規模，接下來只需將「實驗數據」與「系統執行畫面」填入，就能成為一份非常強大的報告。如果在實作 肆、研究結果 期間有遇到數據解讀或圖表生成的困難，隨時可以找我討論！