114-2 系統分析與設計 作業二

國立臺灣大學 資訊管理學系

繳交方式：請至 NTU COOL 作業區上傳作業。
繳交格式：請使用 PDF 文件，並在文件內嵌入相關連結及圖片。
截止時間：*2026 年 4 月 20 日（一）14 點 20 分*。

第一部分：ListenAI 從過世到復活

> 工程師小梁：「明明在我電腦上可以正常運作，爲什麼一上伺服器就跑不動？」
> 工程師訓佑：「我知道了，我去教訓一下那台伺服器就好了。」
> 你：「先等等，你們有聽過 Container 嗎？」

你是一間新創公司的 CEO。在上個月的產品發表會結束後，公司成功獲得頂級加速器 IM Combinator 的種子輪投資，共計新台幣 1,000 萬元。

有了這筆啓動資金，你們決定開發一款全新的產品：ListenAI，一個專注於社群聆聽（Social Listening）的 SaaS（Software as a Service）平台。經過數週密集開發，工程師小梁與訓佑已經完成了最小可行產品（MVP, Minimal Viable Product）然而，當產品實際部署到伺服器上時，卻出現了各種意想不到的問題。

本次作業將帶領你一步步將 ListenAI 從一個僅能在本地運行的 MVP，轉變爲一個能穩定部署於雲端環境的完整產品。

任務 1：基本設置（16 分）

在開始開發之前，請先確保 ListenAI 能夠在你的本地環境中正常執行。

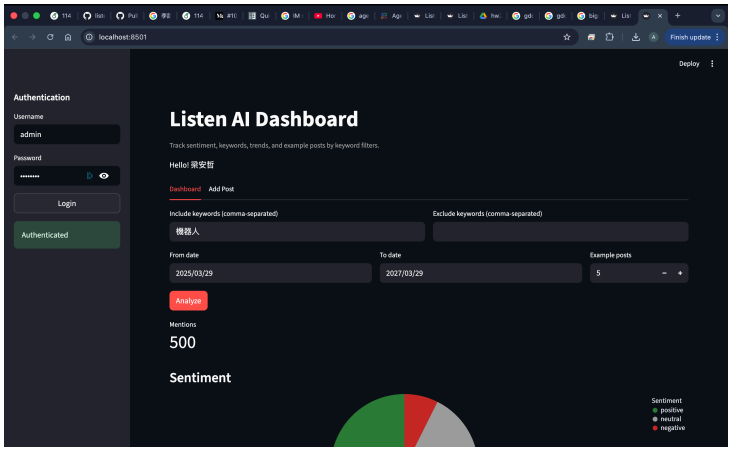

圖 1: ListenAI 在瀏覽器中成功執行的畫面

1. 請先 Fork 助教提供的 GitHub repository。

2. **提供你 Fork 後的 GitHub repository 連結**。
3. 依照 README 中的指示，在本地環境中執行 ListenAI，成功執行的畫面應如圖 1 所示。
4. 依照 `data` 資料夾中的說明，自 Google Drive 下載社群媒體測試資料。
5. 將畫面中的「Hello 梁安哲」修改爲你自己的名字。
6. **提供完成上述修改後的執行畫面截圖，並確認關鍵字「機器人」的檢索結果爲 500 筆**。

任務 2：將 ListenAI 容器化（40 分）

在完成任務 1 之後，你會發現 ListenAI 目前由四個不同的模組組成：

模組	功能	程式語言
Frontend	前端介面，負責使用者登入與畫面渲染	Python
Gateway	驗證請求是否來自已登入的使用者	Node.js (JavaScript)
NLP	自然語言處理（例如情感分析）	Python
Stat	詞頻統計與資料庫檢索	Go

由於各模組使用不同的程式語言與相依套件，直接在同一環境中管理將會變得困難。因此，本任務的目標是將各個模組分別容器化，建立對應的 Container Image，並透過 Docker Compose 將整個系統整合並啓動。

1. 將 `frontend` 模組容器化，並 **提供對應的 Dockerfile**。
2. 將 `gateway` 模組容器化，並 **提供對應的 Dockerfile**。
3. 將 `nlp` 模組容器化，並 **提供對應的 Dockerfile**。
4. 將 `stat` 模組容器化，並 **提供對應的 Dockerfile**。
5. 使用 Docker Compose 同時啓動所有服務，並確保各服務之間的連線設定正確。
6. **提供對應的 `docker-compose.yml` 檔案**。
7. **提供執行指令 `docker ps --all --filter label=com.docker.compose.project` 的結果截圖**。

任務 3：驗證容器化技術（32 分）

在完成 ListenAI 的容器化後，我們可以透過一系列實驗，驗證容器化對應用程式在不同面向上的影響。

1. 本系統共使用三種主要程式語言（Python、Node.js、Go）。**請分別回報各服務所對應的 Container Image 大小，並提供相關截圖。請分析是否存在明顯的大小差異，並說明其原因。**
2. 在 Dockerfile 中，通常包含兩個主要步驟：安裝相依套件，以及複製應用程式程式碼。以 `frontend` 爲例，調換這兩個步驟的順序，在程式碼有所修改的情況下，比較容器的 build 時間差異。**請提供 build 時間的比較表格與截圖，並解釋原因。**

3. 以 `frontend` 爲例，比較使用 `python-slim` 與 `python` 作爲 base image 時，在 build 時間與 Image 大小上的差異。**請提供相關截圖，並說明造成差異的原因。**

4. ListenAI 使用 SQLite 作爲資料庫。然而，資料庫檔案不應直接包含於 Container Image 中，而應透過 Volume 進行儲存。請修改程式碼與 `docker-compose.yml` 設定，並錄製完整操作過程，內容需包含以下步驟：

   (a) 啓動 ListenAI
   (b) 確認系統中無法檢索到「我好喜歡資管系」
   (c) 在「Add Post」頁面新增「我好喜歡資管系」
   (d) 使用指令停止所有 containers
   (e) 使用指令刪除所有 containers
   (f) 重新執行 Docker Compose
   (g) 確認系統可以成功檢索到「我好喜歡資管系」，以驗證資料已正確儲存

   請提供修改後的 `docker-compose.yml` 檔案，以及操作影片的 YouTube 連結。

第二部分：把書評變成現金流——ListenAI 的決策升級

工程師小梁：「我們現在可以分析書評情緒了！這本書好評率 92% !」
工程師訓佑：「那很好啊，所以我們要進貨幾本？」
工程師小梁：「呃……這部分我還沒寫。」

四葉草書局老闆：「我昨天因爲你們的系統多進了 200 本，結果一本都沒賣出去。」
你：「老闆你電話連線不穩我就先掛掉了喔，我們下禮拜再好好談談。」

在 ListenAI 上線後不久，其強大的資料探勘與分析能力迅速吸引了眾多零售業者的關注，並被應用於分析與預測各類商品在消費者之間的受歡迎程度。其中，一家連鎖書店「四葉草書局」看中了 ListenAI 在書店經營上的潛力，決定與你的公司簽訂一項客製化合作合約。

本次合作的核心目標爲：將「四葉草書局」所經營的書評平台「四葉草書評」整合進 ListenAI 系統。透過自動化分析讀者評論，辨識出受讀者喜愛的書籍，並進一步提供決策依據，協助實體門市人員進行備貨與訂單規劃，確保熱門書籍能夠即時上架，滿足市場需求。

任務 1：分析 BPMN 流程圖（12 分）

爲了更清楚地描述此合作流程，你的任務是使用 BPMN（Business Process Model and Notation）繪製 ListenAI 的整體流程，協助開發團隊快速理解系統運作與各角色之間的互動關係。

1. 請使用五條泳道（Swimlanes），分別代表：使用者、四葉草書評、四葉草書店的資料分析師、四葉草書店的零售人員，以及出版社。

2. 請運用 BPMN 的基本元素（事件、任務、閘道），清楚呈現系統的主要流程，並包含可能的例外處理流程（例如：資料探勘結果誤判時的處理機制）。

第三部分：ListenAI 不只沒壞，還變強了

本部分爲加分題，即使未完成，仍可獲得本作業滿分。

目前 ListenAI 是在數週內快速開發完成的原型系統，整體仍有許多可優化之處。以下爲兩項優化任務：

任務 1：優化情感分析的準確度（10 分）

在 NLP 模組中，訓佑實作了一個基於字典（lexicon-based）的情感分析演算法。此方法具有運算速度快與可解釋性高的優點，但在面對複雜語境時表現有限。請完成以下優化步驟：

1. 紀錄並說明 ListenAI 執行環境的硬體規格。
2. 選擇一個適當規模的資料集，並以人工方式或大型語言模型（LLM, Large Language Model）進行標註。
3. 評估現有演算法的準確度（例如 accuracy、F1-score 等指標）。
4. 提出新演算法，可採用統計方法（如 TF-IDF）或深度學習方法（如 BERT-based classifier）。
5. 實作新演算法，並提供更新後的 NLP 模組程式碼。
6. 評估新演算法的準確度與計算成本。
7. 綜合評估是否適合以新演算法取代原有演算法，並說明理由。

任務 2：避免重複運算的架構設計（30 分）

在現有系統中，資料庫儲存一筆社群媒體貼文的格式如下：

欄位名稱	**資料型別**	**限制條件（Constraints）**	**說明**
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	主鍵，自動遞增
platform	TEXT	NOT NULL	平台名稱
author	TEXT	NOT NULL	作者
content	TEXT	NOT NULL	內容
created_at	TEXT	NOT NULL	建立時間

然而，在此設計下，每次執行資料探勘任務時（例如詞頻分析、情感分析與趨勢分析），系統皆需重新對原始資料進行運算，導致效能受限，難以支援更大規模的應用場景。

本任務的目標是設計並實作一套機制，以避免重複運算並提升系統效能。請完成以下步驟：

1. 紀錄並說明 ListenAI 執行環境的硬體規格。
2. 描述現有系統中，一篇文章從進入系統到觸發資料探勘需求的完整處理流程，並繪製對應的流程圖（需涵蓋各模組之間的互動）。

3. 提出改良後的資料儲存設計，並繪製更新後的系統流程圖。
4. 實作改良後的系統架構，並提供各模組對應的修改程式碼。
5. 本作業提供的資料集約爲 5,000 筆社群媒體貼文。請自行擴展資料規模至至少 1,000,000 筆，並比較不同資料規模下，系統效能的差異，以評估本次優化的效果。
6. 分析此改動可能帶來的缺點，例如儲存成本增加、資料一致性問題，或模型／演算法更新的困難性。

結語

當所有任務完成後，ListenAI 已經從一個原型系統，逐步成長爲一個初具雛形的 SaaS 產品。當然，系統仍有許多值得優化與擴展的方向，例如導入搜尋引擎（如 Elasticsearch）、部署至雲端平台，以及建立資料蒐集管線以擷取 Twitter、YouTube、Facebook、Threads 等平台的內容。然而，這些挑戰也意味著更高的技術門檻，未必是目前工程團隊可以立即承擔的。

一個好的產品，能夠幫助新創公司在競爭激烈的市場中站穩腳步。隨著產品逐步成熟，你可能會進一步募集資金、拓展客戶、擴大產品線與團隊規模。若一切順利，公司也許會持續成長，甚至成爲獨角獸企業。而你，作爲 CEO，正站在這段旅程的起點，對未來仍有無限的想像與可能性。

評分與大綱

請參考表 1 的評分標準，各任務分數與重點如下：

任務	**分數**
第一部分	
任務 1：基本設置	16
任務 2：將 ListenAI 容器化	40
任務 3：驗證容器化技術	32
第二部分	
任務 1：分析 BPMN 流程圖	12
第三部分（加分題）	
任務 1：優化情感分析的準確度	+10
任務 2：避免重複運算的架構設計	+30
總分（不含加分）	**100**

表 1: 本作業評分與大綱

最終說明

1. **提交格式**：請以 PDF 文件爲主，務必**嵌入所有必要的截圖與超連結**。
2. **程式碼呈現**：提交程式碼時，請以白底爲主，並僅提供關鍵程式片段，同時加上適當的語法標註（syntax highlighting）以提升可讀性。若題目要求提供截圖，則不限制背景顏色。
3. **文件的專業性**：請確保內容結構清晰、敘述精確，並適當使用圖表與說明輔助讀者理解。一份良好的技術文件，應該能讓他人在不額外說明的情況下也可以清楚理解你的設計思路與實作細節。這項能力在學術研究與實務工作中都一樣重要～

相信完成本次作業後，你應能更深入理解容器化技術與系統整合的核心概念，並掌握從原型系統到可部署服務的完整流程。同時，你也將體會如何將資料分析結果轉化爲實際的商業決策，並應用於真實世界的情境中。**祝大家作業順利！**