社群媒體與對話機器人系統設計

期中專案成果報告

GPU 購物小幫手

專案成員

資工四 107590012 陳志榮

資工四 107590037 應耀德

目錄

一、專案主題

本專案設計一套電商購物推播系統，讓顧客(Customer)能夠追蹤指定商品，進行下訂單的動作，並且在商家補貨後，會第一時間收到 Line bot 推播的補貨通知，讓顧客不錯過購買時機，同時也能查看市場統計報表，查詢商品的熱賣程度，以及品牌熱門度。管理員(Admin)可以管理商品、管理訂單，並且能查看統計報表，並查看商品的銷售報表、營業額報表。

本專案以 GPU 庫存的追蹤購買為例子，未來可擴充成不同商品類型的 Line bot，亦可以經營成電商平台的形式，定期的推播新貨通知以及優惠商品。

二、系統功能

2.1 顧客(Customer)：

1. 查詢商品(GPU)：
    - 查詢商品資訊：名稱、品牌、價格、庫存數量、圖片
    - 新增 / 移除至我的最愛功能
    - 購買商品功能
2. 我的最愛：
    - 查看最愛商品資訊
    - 最愛商品補貨時，可以接受 Line bot 推播補貨通知
3. 訂單紀錄：
    - 查看訂單資訊(店家未接受、送貨中、已完成)
    - 查看歷史訂單紀錄
4. 市場統計：
    - 查看熱銷商品排行
    - 查看熱銷品牌排行
    - 商品庫存變化圖表
    - 商品價格變化圖表

2.2 管理員(Admin)：

1. 查詢商品(GPU)：
    - 查詢商品資訊：名稱、品牌、價格、庫存數量、圖片
    - 新增 / 移除至我的最愛功能
    - 購買商品功能
2. 我的最愛：
    - 查看最愛商品資訊
    - 最愛商品補貨時，可以接受 Line bot 推播補貨通知
3. 管理訂單：
    - 編輯訂單狀態(接受訂單、送貨中、已完成)
4. 管理商品(GPU)：
    - 新增 / 刪除商品資訊
    - 上 / 下架商品
    - 編輯商品資訊：名稱、品牌、價格、庫存數量、圖片
5. 統計報表
    - 查看單一商品、全部商品的銷售報表
    - 查看指定日期內的營業額報表

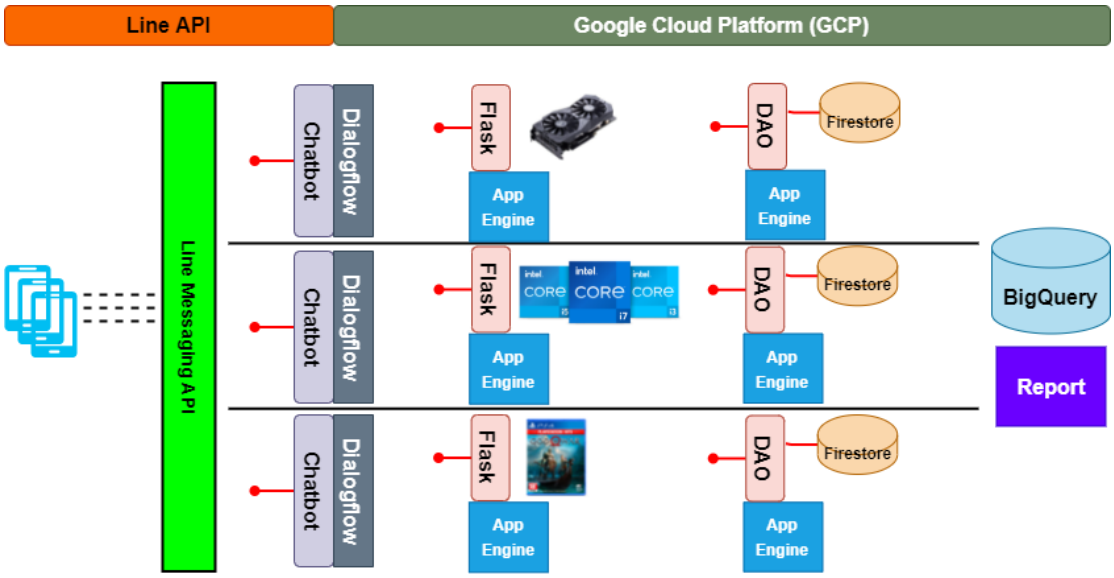
三、系統架構

3.1 開發框架

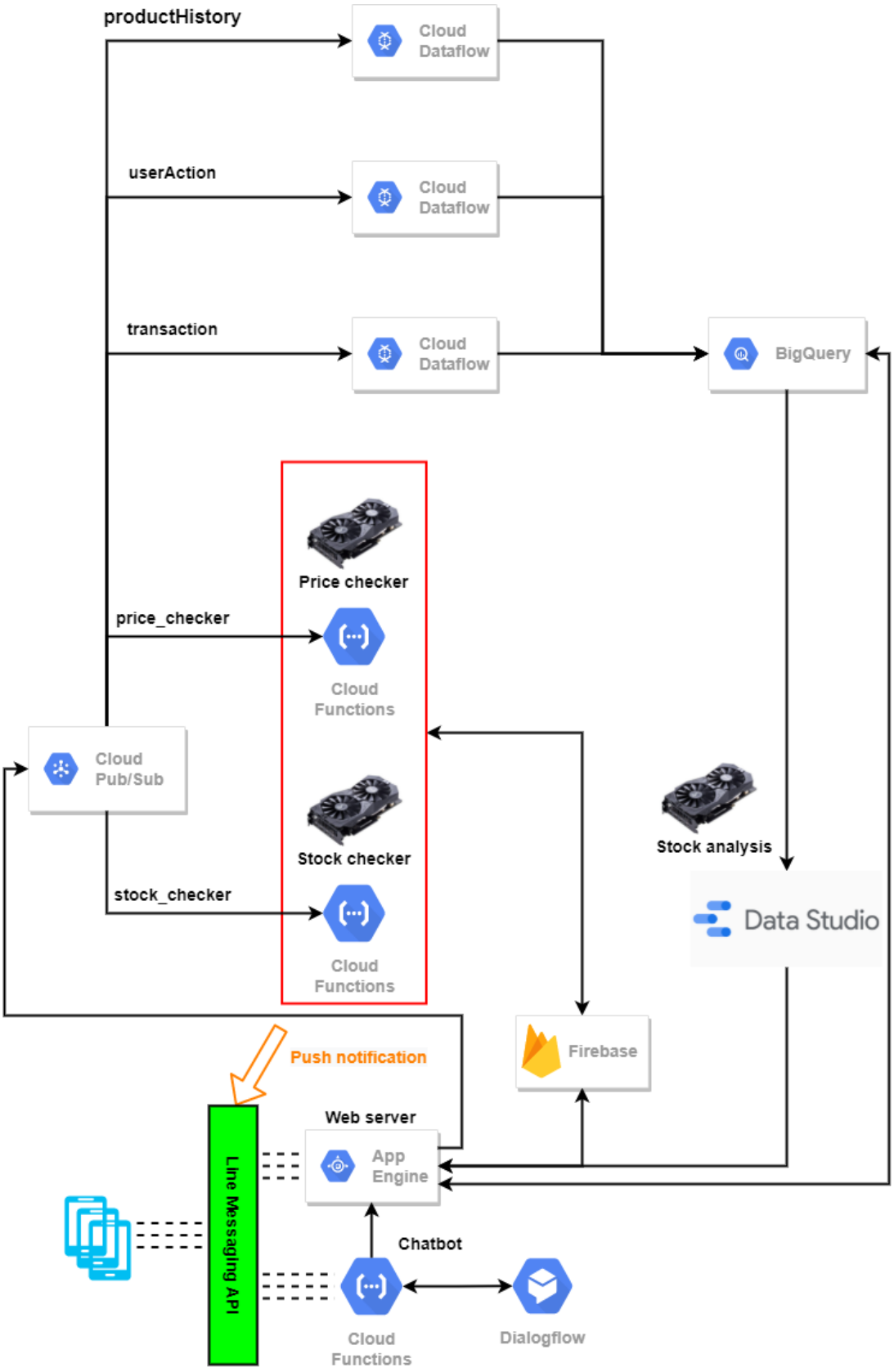
- Google Cloud Platform
- Line Messaging API
- Dialogflow
- Python Flask
- Data Studio

3.2 架構圖：

3.2.1 Line bot 架構圖

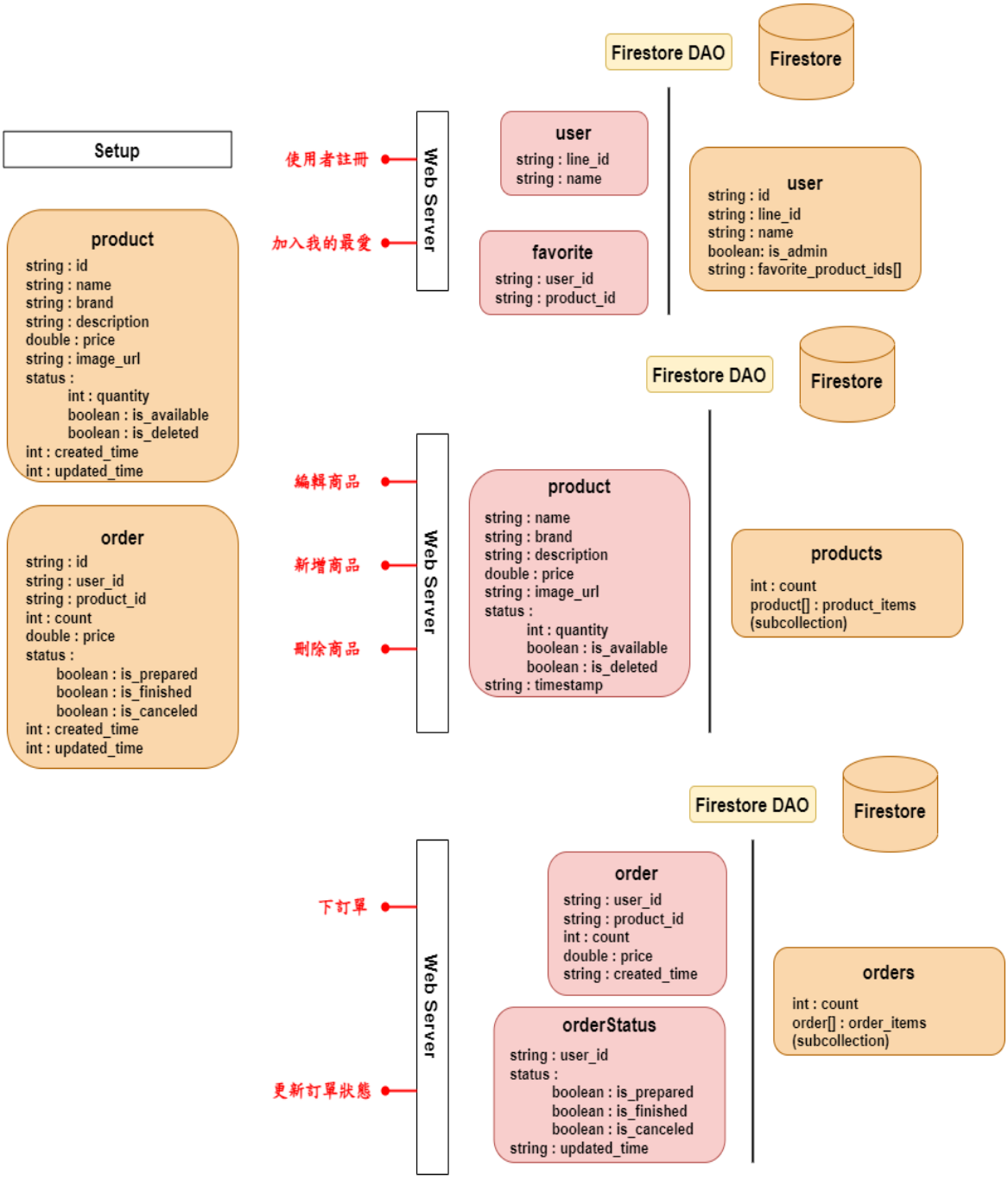


3.2.2 整體架構圖

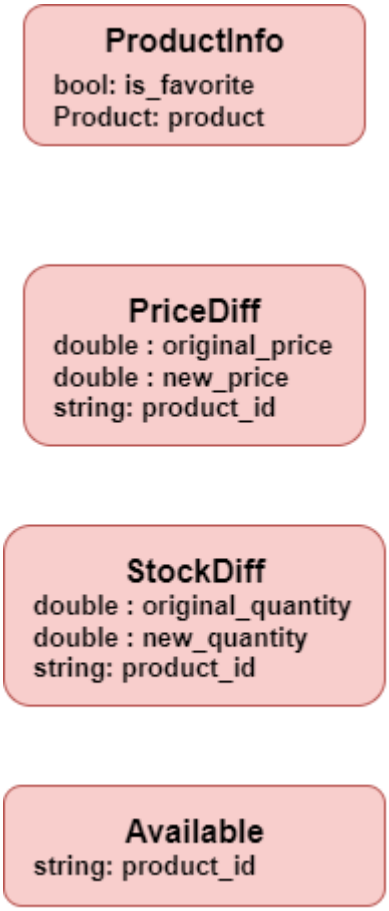

productHistory
Cloud Dataflow
userAction
Cloud Dataflow
transaction
Cloud Dataflow
BigQuery
Price checker
price_checker
Cloud Functions
Cloud Pub/Sub
Stock checker
stock_checker
Cloud Functions
Stock analysis
Data Studio
Firebase
Push notification
Web server
App Engine
Line Messaging API
Chatbot
Cloud Functions
Dialogflow

3.3 資料模型圖

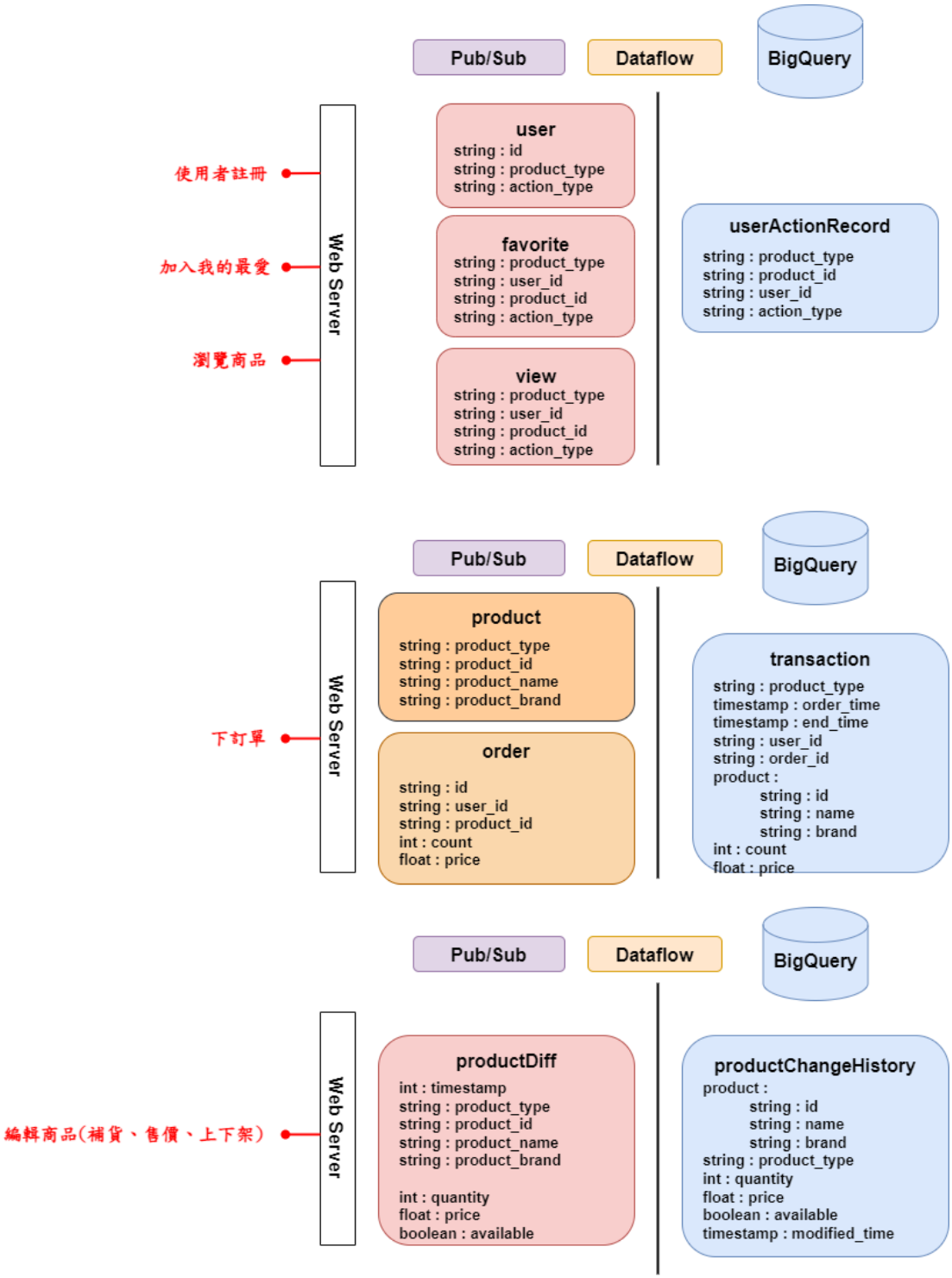
3.3.1 FireStore 資料模型圖



3.3.2 Data Transfer Object 資料模型

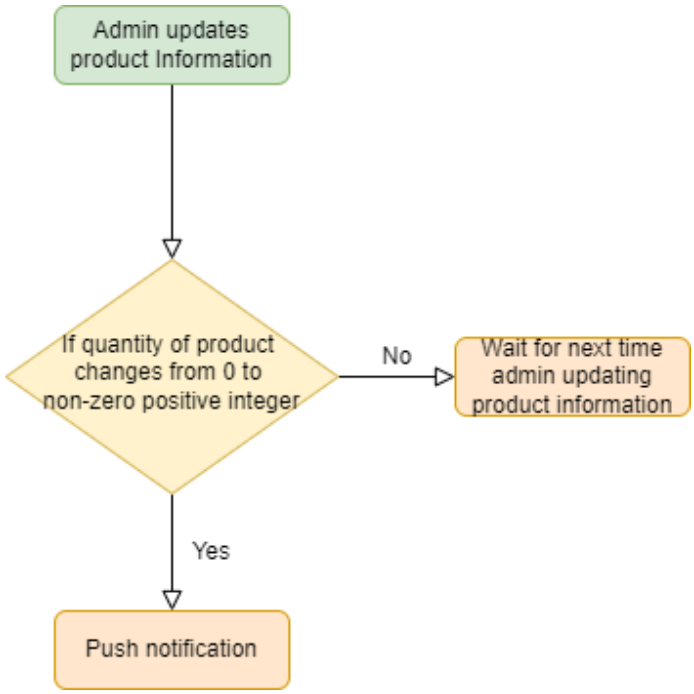


3.3.3 Big Query 資料模型圖

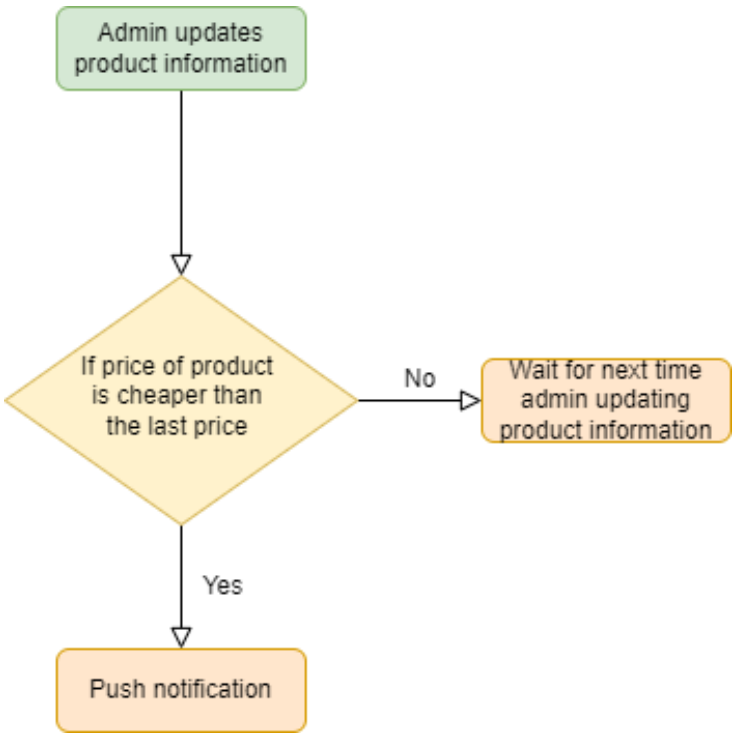


四、推播演算法設計

4.1 商品補貨推播通知



4.2 商品降價推播通知



五、測試流程及測試結果截圖

Coming Soon

六、專案學習心得

應耀德：

此次期中專題難在如何構想功能、設計架構與需考慮各個模組規劃得可行性，在實作中也曾遇到 Firestore 並不支援 **count** 功能，原本是打算直接 stream loop 去計算，但當資料量較大時，一般的 count 作法可能就不是最佳解，於是參照官方建議實作以 Distributed Counter 方式儲存 count，其概念與 Map 實作相似，先決定 bucket(比喻)個數後，建立 document id 為 index 值的 document，欄位為 count 預設值為 0，subcollection 為儲存資料用。儲存資料時，使用亂數取值，將數值對應至 document id，存入該 document 中的 subcollection 後，再把 document 欄位的 count++。

除了上述遇到的問題外，在網頁前端呈現與後端 API 溝通，也使用 partial rendering 部分的 view，例如：查詢結果等。在換頁時，我們也採用 Infinite Scrolling 的方式進行取下一段資料。比較可惜的是，專題無法在期限內實作完全，但也學到軟工方面的相關知識與在設計功能面時，所需要注意的部分等等。第一次做雲端系統整合，剛開始還很不熟悉，現在也漸漸熟悉指令部署、Remote GitHub Repo、IAM、Service Account 權限分配等。

陳志榮：

這次的期中專題成，經過期中企劃的報告，融合老師的建議回饋後，重新修改了我們的架構，一個禮拜從零開始實做，我主要是負責 website 的部分，一開始先刻好前端的版型，再利用 flask 的 route 功能讓使用者能順利跳轉到指定功能的頁面，後來進行到 firestoreDAO 的部分，讓頁面的操作能拿到正確資料，並且可以進行 CRUD，但是因為我自己的後端能力較為薄弱，開發中遇到許多瓶頸，需要反覆的察看 firestore documentation 相關的說明進行嘗試，程式碼的修正也靠組員幫助許多，但是也在過程中更加地了解整體的軟體開發框架，是真的要動手做才能真正的學進腦內，不過這一個禮拜的時間真的十分有限，想做的功能無法盡善盡美，但是第一次利用雲端架設前後端系統的整合，真的是滿載而歸、受益良多。