Meal Order

team members
洪世彬、林俊佑、吳庭維、張家誠、戴靖婷

Introduction

Target Audience

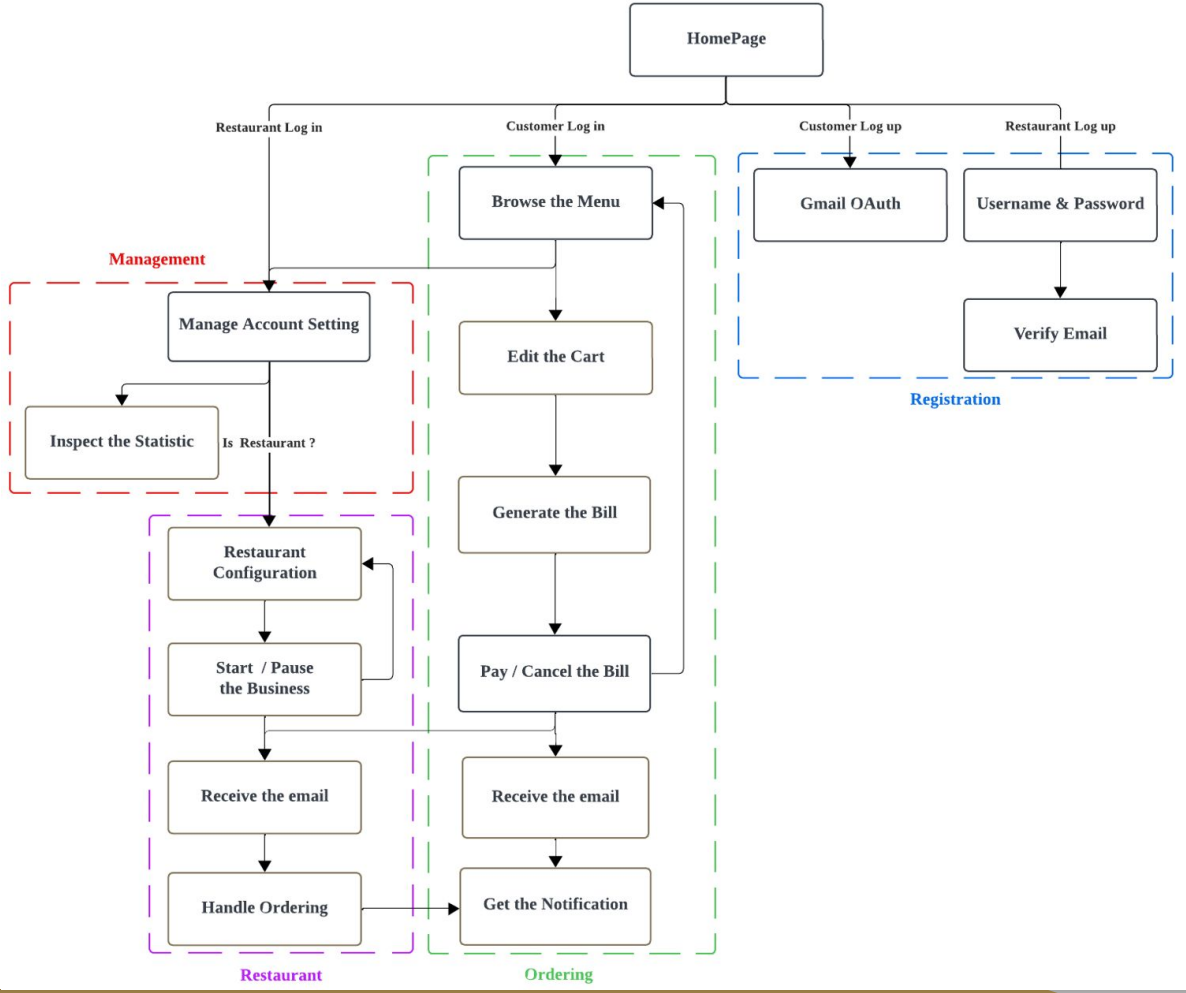
- 煩惱午晚餐吃什麼的員工
  - 條件過濾:
    可以篩選飲食禁忌及過敏原，以避免誤食
  - 消費總額計算:
    可以查詢消費歷史紀錄和總額，讓大家可以規劃月內的飲食及消費
  - 最懂你的智慧助理:
    引入大型語言模型(LLM)，可以根據喜好讓系統自動提供合適的菜單

Target Audience

- 想在平台拓展業務的餐廳業者
  - 彈性收費:
    平台收費採訂閱制，可隨時取消
  - 自由度高的菜單:
    可以提供多種標籤給消費者過濾、設定特殊節日菜單等
  - 統計數據:
    後台提供各項營收數據，方便做出變換菜色的決策

Design Choices

- 為了讓開發方向不要在初期就太分散，我們做了以下的限制
  - 一律先登入才能點餐
  - 先不考慮現實的金流問題，以內建「電子錢包」支付
  - 先只支援自取 (餐點好了由店家通知消費者)
  - 先考慮一個 restaurant 帳號最多開一間餐廳
  - 多數消費者不願意為一個點餐平台多記一個帳號
    ⇒ 統一以 gmail 登入 (=驗證外包)
  - 一開始就分成兩種帳號: customer & restaurant，採用不同的登入方式
  - restaurant 帳號採訂閱制，開始營業後每月先自動付款，錢包金額不足自動停業，也可以手動停業、重新開業


HomePage
Restaurant Log in
Customer Log in
Customer Log up
Restaurant Log up
Browse the Menu
Gmail OAuth
Username & Password
Management
Manage Account Setting
Edit the Cart
Verify Email
Inspect the Statistic
Is Restaurant ?
Registration
Generate the Bill
Restaurant Configuration
Start / Pause the Business
Pay / Cancel the Bill
Receive the email
Receive the email
Handle Ordering
Get the Notification
Restaurant
Ordering

UI / UX

Design Thinking

- 用戶可能面臨的問題:
  - 自行購買需要預備現金或其他支付方式
  - 現場購買可能面臨排隊及等候餐點的問題
  - 自行購買受限於店家距離
  - 決定餐點會有選擇障礙

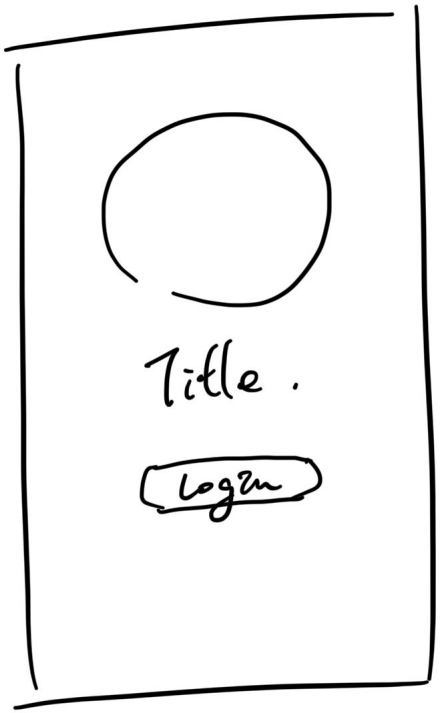
Design Thinking

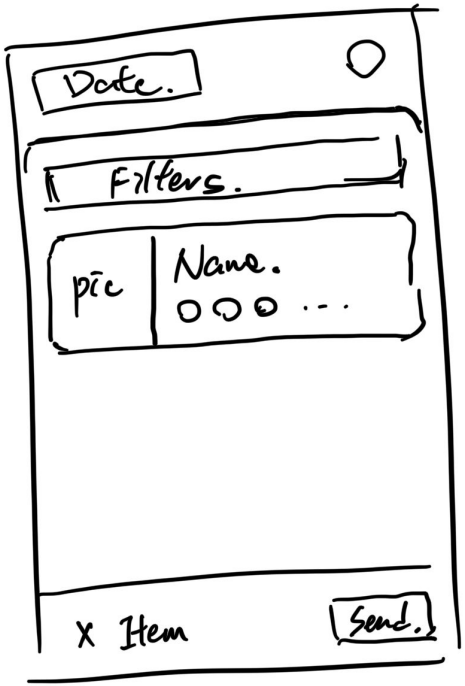
- 為了讓員工可以方便快速的取得餐點，因此開發一個訂餐系統。
- 由於使用者都是公司員工，所以可以省去費用支付的問題(採月結)，可以直接送出訂單。
- 由於餐點的數量較多，讓使用者難以決定餐點，因此導入大型語言模型來替使用者解決選擇障礙的問題。
- 多數用戶會希望知道自己訂購了什麼餐點，所以已訂購的餐點應清楚展示。
- 由於飲食偏好的差異，提供一個便捷的篩選方式能讓用戶快速找到期望的餐點。
- 餐廳管理者使用電腦來管理餐點會更方便，所以設計介面時採用桌面應用的尺寸設計。

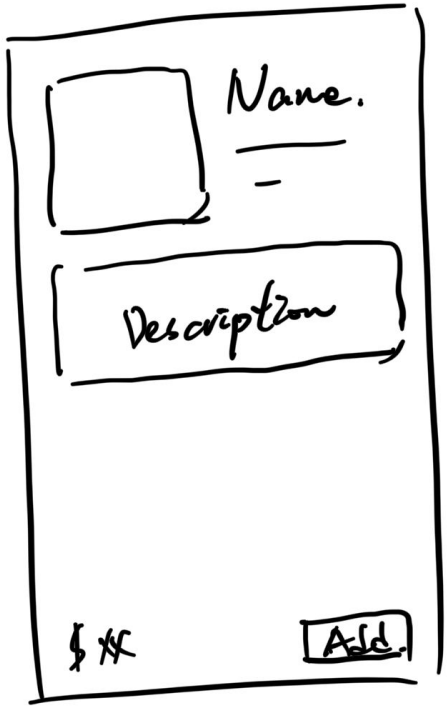
Design Thinking

- 發放優惠券
- **統計各菜色的銷售額**
- **希望可以看到菜色裡是否含有某些過敏原(ex: 海鮮)**
- 希望可以收到優惠的即時通知
- 設定自己喜愛的菜色，可以優先推薦
- **透過自然語言，在特定需求下獲取推薦菜單**
- **透過 GPS 抓取用戶地點讓用戶快速選取廠區和取餐點**
- 員工可以建議新店家
- 消費者可以針對訂餐做評論和評分
- **商家提供每個商品的剩餘供應量**
- **預定提醒通知（訂餐前、特殊節日發送提醒）**
- **…**

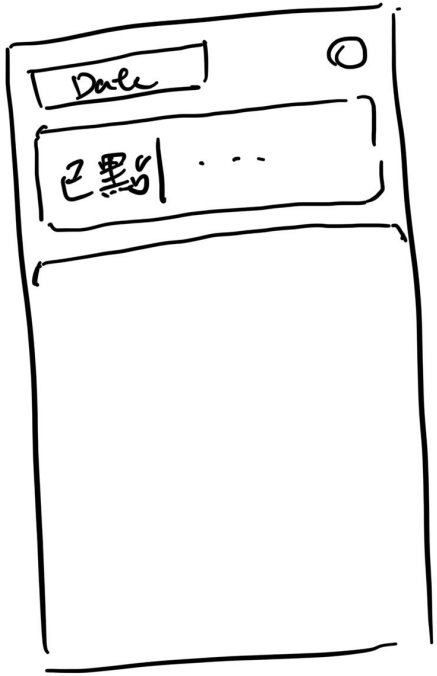
Wireframe

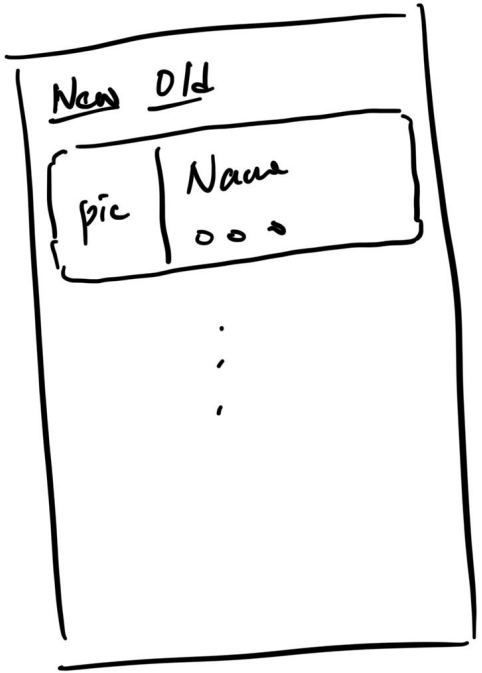

Title.
Login

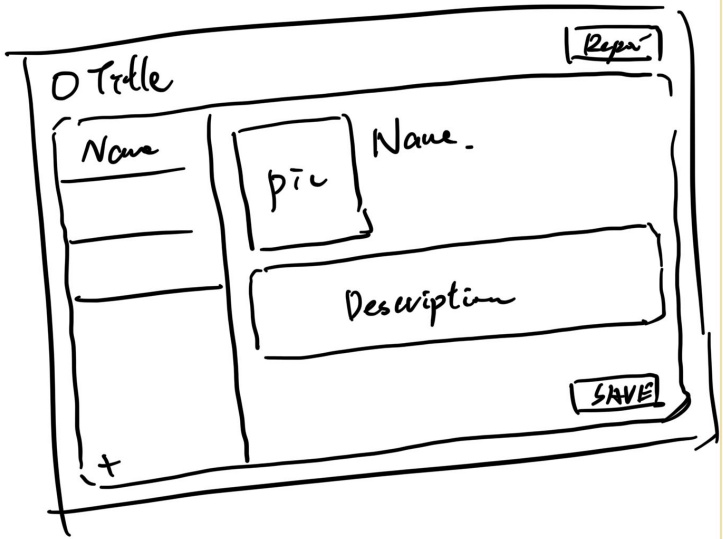

Date.
Filters.
pic
Name.
X Item
Send.


Name.
Description
$ XX
Add.

Wireframe


Date
已點 ...

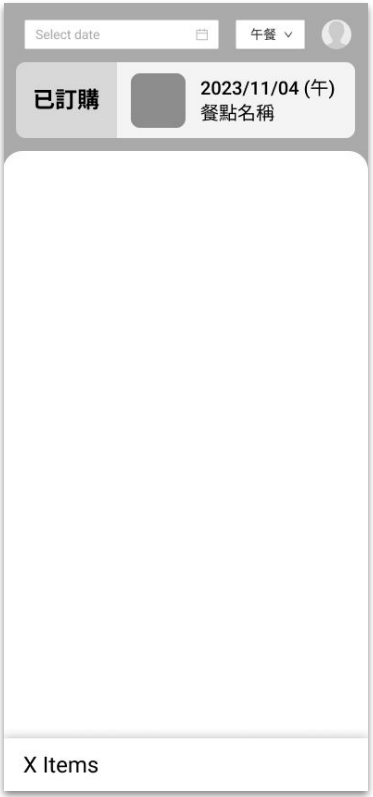

New Old
pic
Name
...

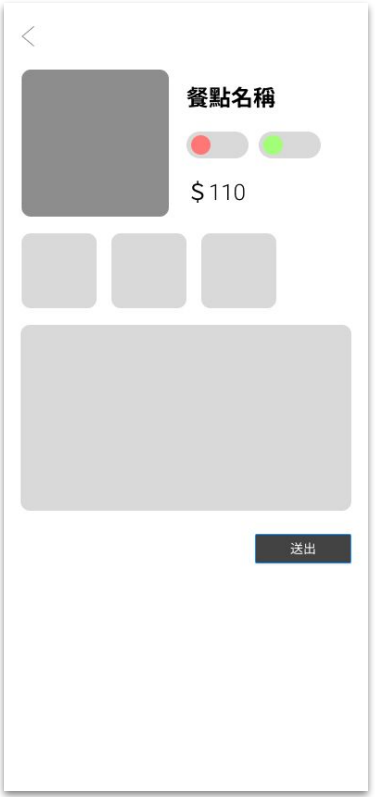

Title
Name
pic
Name.
Description
SAVE
+

Mockup


Title

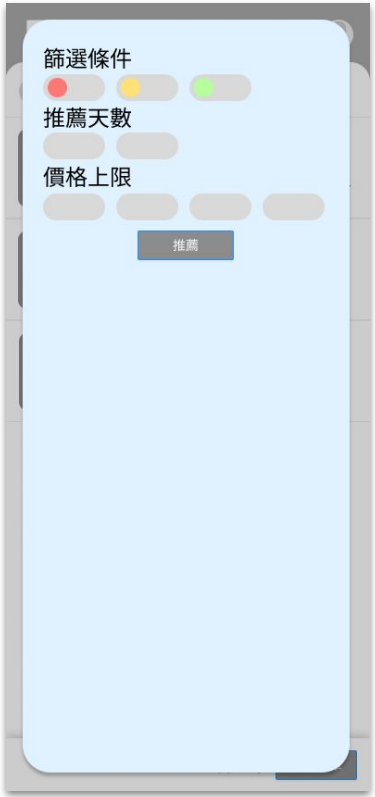

Select date
午餐
餐點名稱
$100
剩 X 份
餐點名稱
$110
餐點名稱
$90
AI
X Items
檢視訂單


Select date
午餐
已訂購
2023/11/04 (午)
餐點名稱
X Items


餐點名稱
$110
送出

Mockup


Select date
午餐
餐點名稱
$100
剩 X 份
餐點名稱
$110
餐點名稱
$90
AI
X Items
檢視訂單


篩選條件
推薦天數
價格上限
推薦


篩選條件
推薦天數
價格上限
2023/11/04
餐點名稱
$110
餐點名稱
$110
2023/11/04
餐點名稱
$110
餐點名稱
$110
取消
送出


已訂購
已取消
已完成
2023/11/04
餐點名稱
$110
餐點名稱
$110
2023/11/05
餐點名稱
$110
餐點名稱
$110
2023/11/06
餐點名稱
$110
2023/11/07
餐點名稱
$110

System Architecture

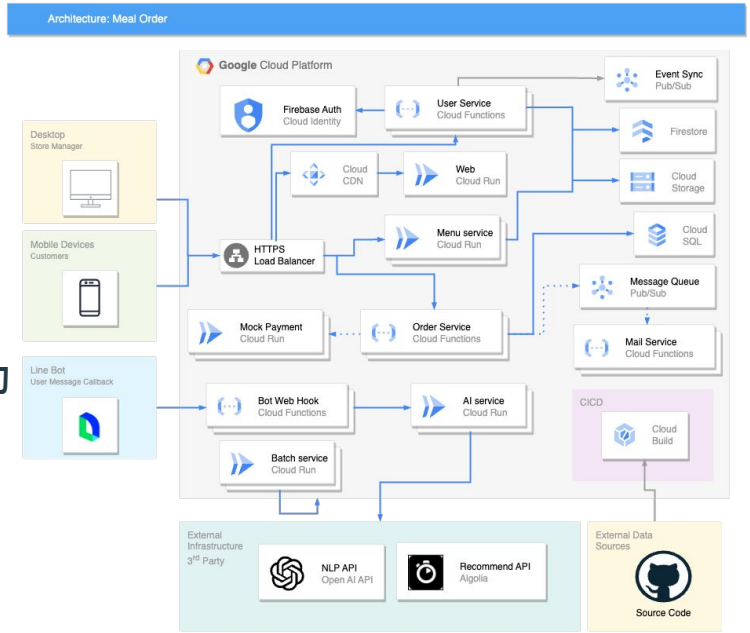
Technical Stack

前端	框架	React + Next.js
前端	語言	Typescript
前端	UI component	Ant Design
後端	資料庫	MSSQL / firestore
後端(核心服務)	框架	.Net 7.0 + minimal-api
後端(核心服務)	語言	C#
後端(次要服務)	框架	echo
後端(次要服務)	語言	golang
infra	cloud vendor	GCP
infra	Infrastructure as Code	Terraform

Micro Service

- Why not Monolith ?
  - 確實拆分每個成員的任務範疇
  - 可以體驗更新的架構設計風格

- GCP, rather than Azure ?
  - 成本低 (9000 NTD credit)
  - 強大的數據處理能力, 作為 AI 服務驅動力
  - 團隊成員有 GCP 相關專業



Micro Service

- Overview
  - User Service : 一般消費者和店家的會員服務
  - Menu Service : 店家菜單相關服務
  - Web Service : 用戶網頁與店家網頁
  - Order Service : 訂單創建(更新)、訂單支付與結算服務
  - AI Service : 利用自然語言模型推薦店家
  - Batch Service : 維持系統持續運行的腳本(含數據備份、數據搬遷、...)
  - Mail Service : 被 Payment 服務調用，寄 email 給用戶
  - Bot Service : 接收 line bot 資訊與調用 AI service
  - Mock Payment Service : 模擬支付相關服務，方便測試

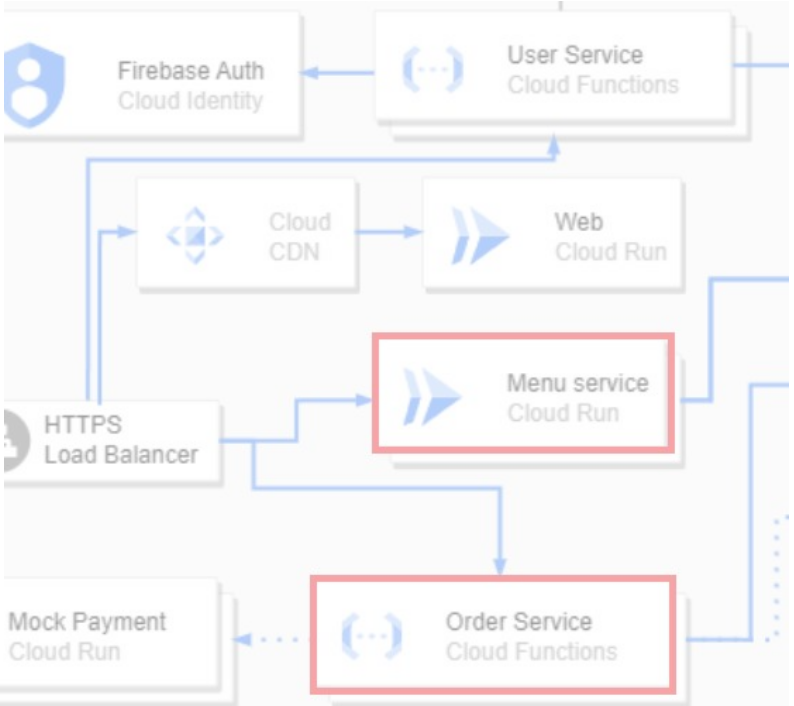
Details & Reasons (Front-end)

- Web Service : 用戶網頁與店家網頁
  - React + Next.js
  - Typescript
  - Ant Design
  - 在 cloud run 上執行
  - 配合 cloud CDN 加速
  - GCS 提供用戶上傳的靜態資源

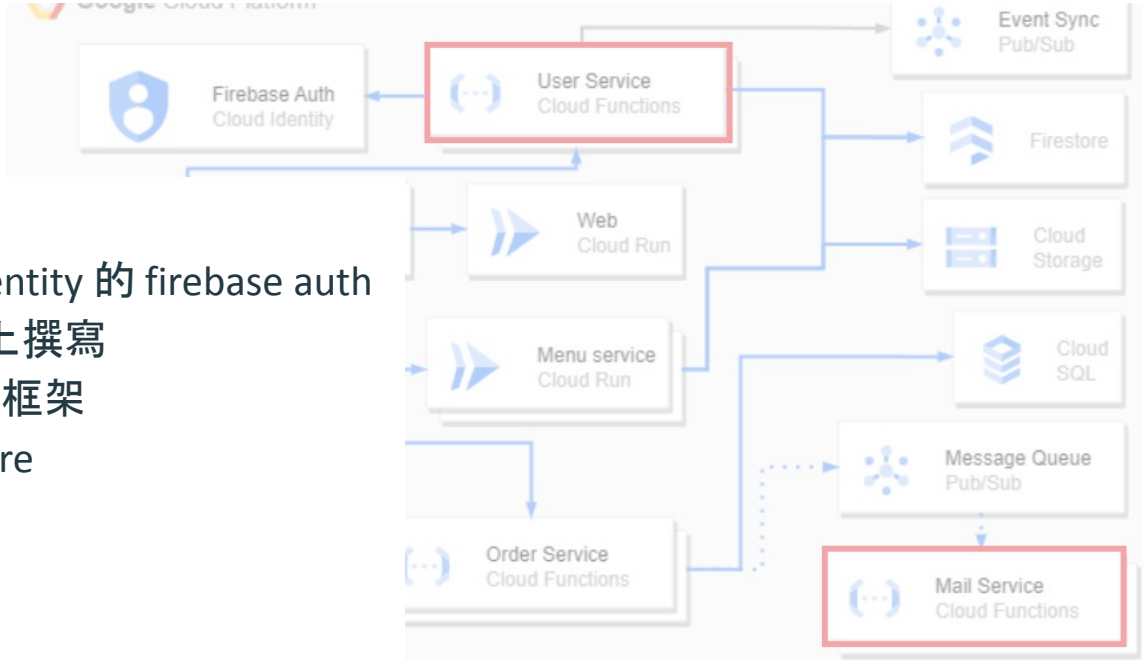
Details & Reasons (Back-end Core)

- Order Service
  - C#
  - .Net 7.0 + minimal-api
  - 在 cloud run 上發布
  - 資料庫為基於 cloud SQL 的 MSSQL

- Menu Service
  - C#
  - .Net 7.0 + minimal-api
  - 在 cloud run 上發布
  - 資料庫為 firestore
  - 靜態圖片存在 GCS



Details & Reasons (Back-end Minor)

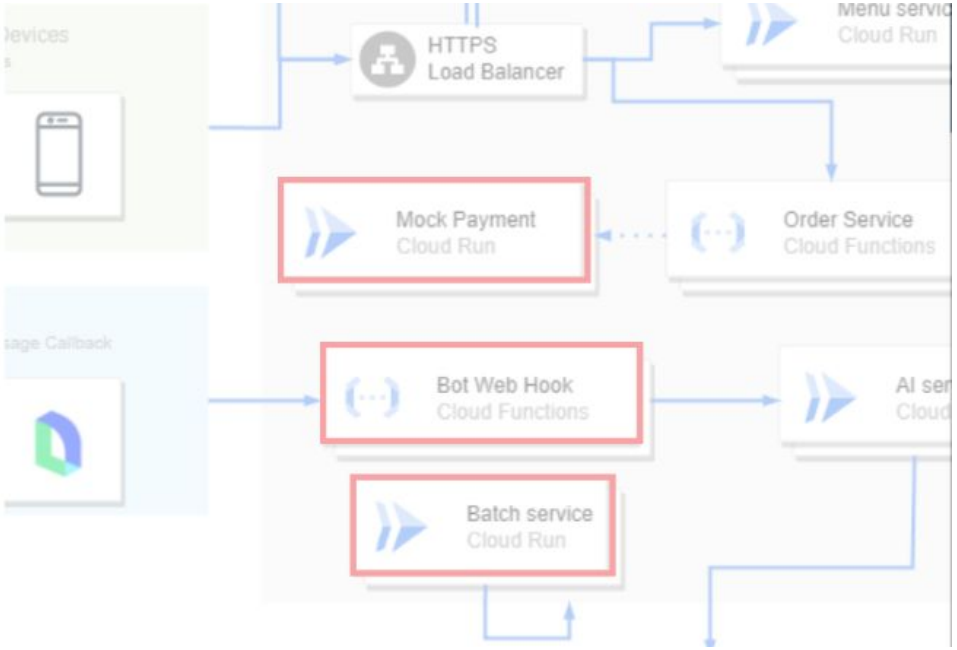


- User Service
  - 引用基於 cloud identity 的 firebase auth
  - 在 cloud function 上撰寫
  - 使用 go 配合 echo 框架
  - 資料庫使用 firestore

- Mail Service
  - 使用 go 撰寫
  - 在 cloud function 上執行
  - 利用 pub/sub 觸發

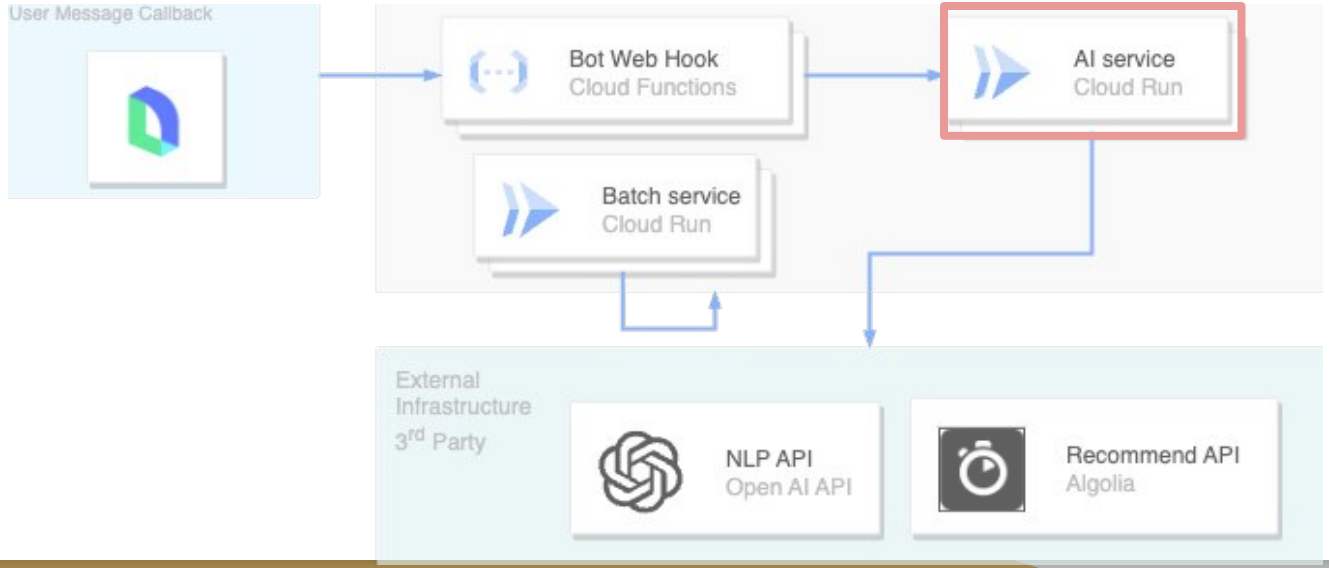
Details & Reasons (Back-end Minor)

- Bot Service
  - 使用 go 配合 echo 框架
  - 在 cloud function 上執行
- Mock Payment Service (測試用)
  - 使用 go 配合 echo 框架
  - 在 cloud run 上執行
- Batch Service
  - 使用 go 配合 echo 框架
  - 使用 cloud Scheduler 來觸發
  - 使用 cloud workflow 來做流程控管
  - 備份區在 GCS



Details & Reasons (Back-end Minor)

- AI Service
  - 採用 python 基於 fastAPI 框架
  - 在 Cloud Run 上撰寫
  - 盡可能串接成熟外部AI 服務

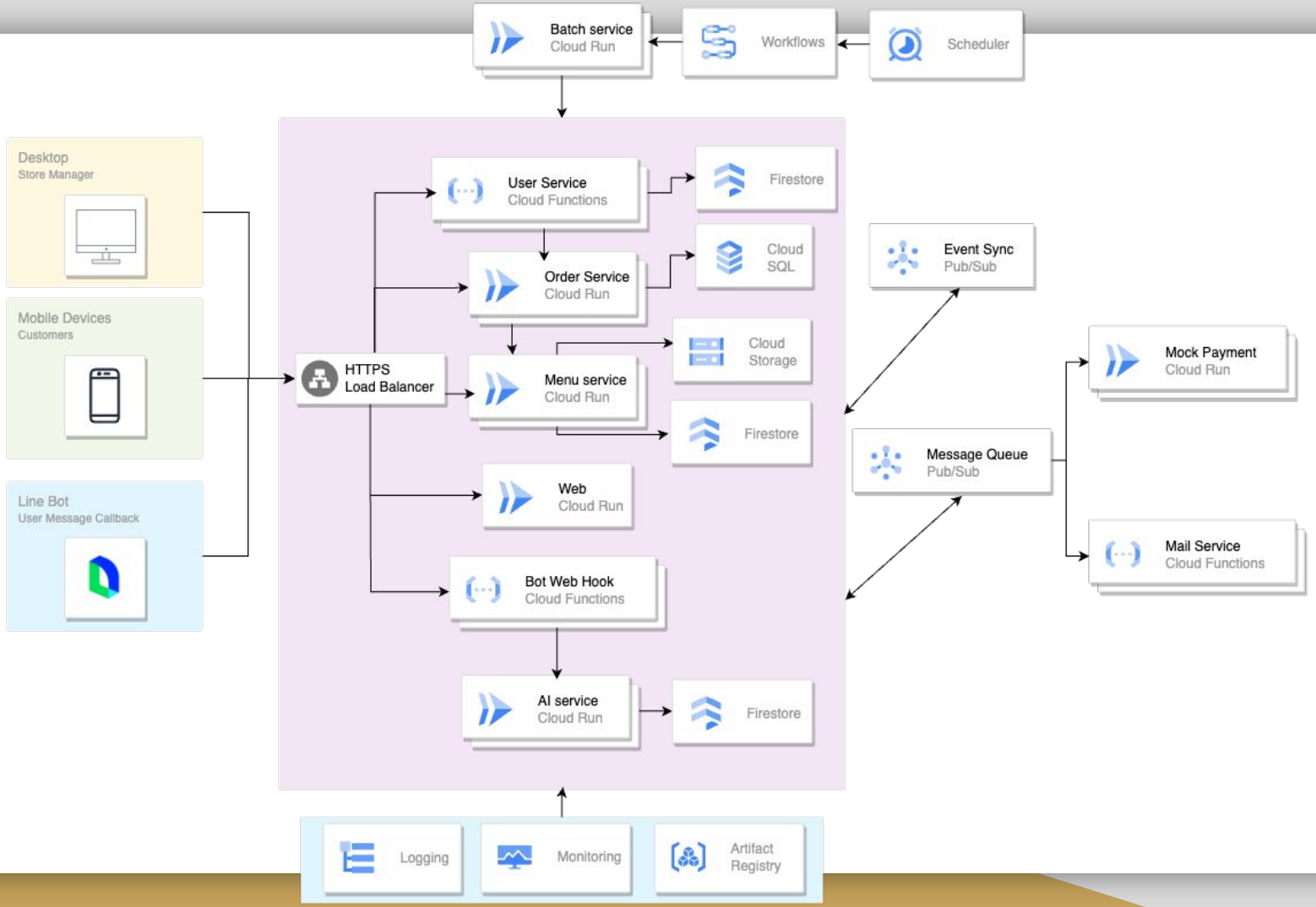


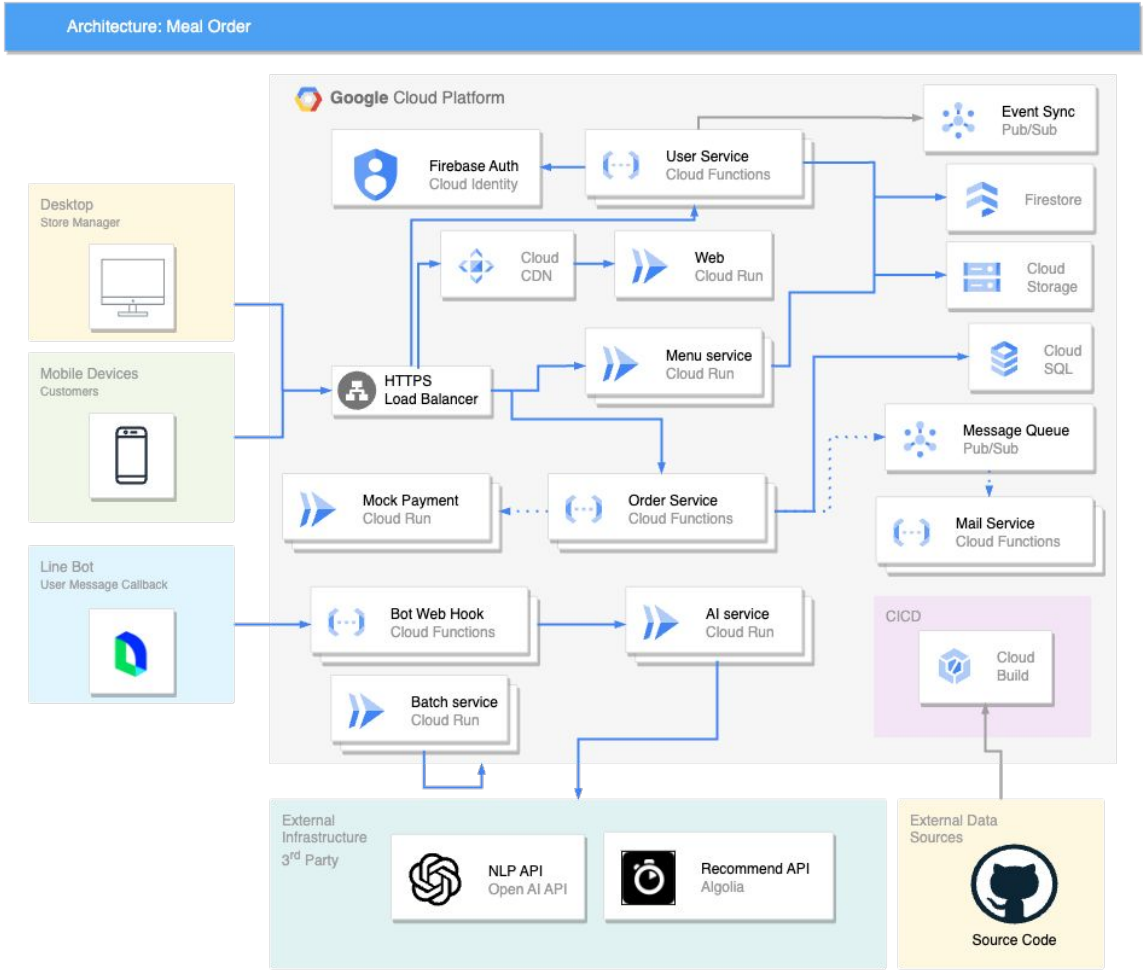
Infra

- 利用 pub/sub 做微服務的 event sync
  - 為官方推薦的最佳實踐(更簡單更便宜)
  - 其他選項各有缺點(ex: 自建 RabbitMQ, Redis, …)

- 利用 pub/sub 做 mail service 的 message queue
  - 為官方推薦的最佳實踐(更簡單更便宜)
  - 其他選項各有缺點(ex: cloud task, kafca)

- 利用 Cloud Logging 和 Cloud monitor 做 **監控**
- 利用 load balancer 的 health checks 配合 email 做 **告警**
- 利用 Artifact Registry 存放 docker Image / npm package / go module


Batch service
Cloud Run
Workflows
Scheduler
Desktop
Store Manager
Mobile Devices
Customers
Line Bot
User Message Callback
HTTPS
Load Balancer
User Service
Cloud Functions
Firestore
Order Service
Cloud Run
Cloud
SQL
Menu service
Cloud Run
Cloud
Storage
Firestore
Web
Cloud Run
Bot Web Hook
Cloud Functions
AI service
Cloud Run
Firestore
Event Sync
Pub/Sub
Message Queue
Pub/Sub
Mock Payment
Cloud Run
Mail Service
Cloud Functions
Logging
Monitoring
Artifact
Registry



food.drawio - link (diagrams.net)

SA Full Information

https://leon1234858.notion.site/mid-report-9dc7f5339af74353b17be01affe535e8

Thank You for Listening

leon123858/tsmc-meal-order: NTU cloud native class final project (github.com)