
台北一日遊網站開發專案：

未來的七週，我們將運用前後端分離的架構，根據 RESTful API 的設計，整合金流服務，完成一個旅遊電商網站。在這之中，除了運用第一階段學過的所有技術之外，還得熟悉 Git、GitHub 版本管理工具，並且透過 Amazon EC2 將網站上線，學會基本雲端服務維運技巧。

那就，開始嘍！

盤點所有的資源：

開始本次的開發之前，請先確認以下資源你可以正常的取得，在未來的引導中，我們不會再用文字說明所有細節，**你應該自主遵循 Figma 設計稿和 API 規格文件的說明來進行開發：**

Figma 設計稿

https://www.figma.com/file/Wi5NkVUusfOXocObmGKhWQ

API 規格文件

https://app.swaggerhub.com/apis-docs/padax/travel-to-taipei/1.1.0

學習程式開發工作流程：使用 Git、GitHub 完成以下動作

在第二階段的訓練中，你應該利用 Git 版本管理工具以及 GitHub 雲端版控服務，來管理一個持續開發的專案，**請在週二以前，立刻完成以下動作：**

GitHub Repository 建立、同步與協作設定：

1. Local Machine

下載我們提供的基礎專案 (點此下載)，將專案資料夾解壓縮，得到一個本機專案。

2. Repository

到你的 GitHub 帳戶中創建一個 Repository，按照網頁上建立新專案的命令指示，將本機專案推送到 GitHub Repository。

3. Collaborator

請至 GitHub Repository 專案設定中 Collaborators 管理，邀請 Chao-Wei Peng (cwpeng) 加入成為你的專案協力者。

4. Review Rule

請至 GitHub Repository 專案設定中 Branches 管理工具，為你的 main 分支新增保護規則：Require a pull request before merging 並且選擇 Require approvals 至少 1 次。(建立規則時，輸入要套用的分支名稱，並且將相關的規則選項打勾)

你的電腦，本機端操作：

5. Branch & Checkout

開啟電腦中的本機專案，利用 Git 工具做版本管理，請立刻新增並 checkout 到 develop 分支上，進行未來的程式開發。

6. Add & Commit

請打開專案中的 app.py 程式檔案，把最下方的埠號改成 3000。接著將這個更新，透過 git add 指令以及 git commit 指令，在本機專案的 develop 分支，建立一個新的版本。

7. Push

將本機專案中建立好的新版本，透過 git push 指令推送到 GitHub Repository 的 develop 分支。在 GitHub Repository 的網站介面中，你會在 develop 分支看到剛剛推上來的最新程式碼狀況。

GitHub 網站操作：

8. Pull Request

在 GitHub Repository 網站介面中，建立一個新的 Pull Request，準備將你自己的 develop 分支合併到你自己的 main 分支。此時，邀請 Chao-Wei Peng (cwpeng) 擔任你的 Code Reviewer，在訊息中填入你的名字，靜待 review 的結果。

9. Review & Merge

如果 review 通過，你可以在網站上的 Pull Request 管理介面，點擊 Merge Pull Request 完成將 develop 分支合併到 main 分支的動作。(概念上 develop 分支是開發過程使用，main 分支則保存經過檢驗的程式碼版本)

10. Fix and Re-Request Review

如果 review 沒有通過，回到本機專案，修正問題後 push 新的修正版本到 develop 分支，並到 Pull Request 的網頁介面要求 Chao-Wei Peng (cwpeng) 重新 review 程式，持續這個過程，直到通過為止。在修正的過程中，你不需要建立新的 Pull Request。

未來，要持續地透過這樣的程序，在同一個專案上，完成每週的工作任務，通過檢查。