國 立 臺 北 商 業 大 學

資 訊 管 理 系

1 1 2 資 訊 系 統 專 案 設 計

系統手冊

組　　別：第 112402 組

題　　目：詐騙一指通(LINE BOT)

指導老師：唐震老師

組　　長：N1096413 李卉君

組　　員：N1096416 張峻華　N1096421 吳祖霖

N1096443 楊哲維　N1096452 黃子芸

N1096455 高登

中 華 民 國 1 1 2 年 3 月 1 1 日

目錄

表目錄

圖目錄

第1章　前言

1-1 背景介紹

在這個資訊爆炸的時代，人人都有上網都需求，而上網往往都會有一些有心人士在網上散布一些假消息，設立假網站，誘騙一些對於個資的保護比較沒有概念的使用者，來達到詐取金錢或者是控制言論的目的，而我們出發點就是著重在身邊一些年紀稍長的長輩，抑或者是對於個資外洩相對不在乎者，讓他們開始注重關於個資保護的議題，在看到陌生人所傳來的訊息或者是網址時，能先想到我們的機器人可以為他們做第一層的把關，降低個資外洩且受騙的風險，減少傷財的可能性。

1-2 動機

就如上述所提，因資訊爆炸，且科技越來越進步，人們的上網需求逐漸擴大，日常生活中不管是工作、娛樂都需要用到網路，而網路詐騙一直是熱門的議題，畢竟有些不肖人士喜歡投機取巧來騙取他人的財物，導致相對沒有個資保護意識之使用者容易上當受騙，不管是誰，都不希望辛苦賺來的錢被平白無故地奪走；而我們能創造出讓使用者不易上當受騙，且透過文章更新的推播，讓個資保護意識興起，當知名度出來以後，找尋機會，透過廠商的連動來使功能更加完善。

1-3 系統目的與目標

知名網路安全公司趨勢科技有推出了一個防詐騙機器人，我們打算做的功能，相同的部分有:檢查 Email 是否外洩、檢查 Line ID 是否詐騙 ID、檢查網址是否為詐騙網站、不定時的更新詐騙資訊的文章供使用者查閱；不同的部分有:假訊息以及謠言的辨別，因我們沒有相當的知名度，所以無法透過大量的使用者來回傳訊息，藉此來判斷訊息的真假。

1-4 預期成果

功能方面:Line bot 防詐騙機器人最主要的功能就是協助使用者，去辨別和過濾來自網路上不明來源的訊息，提醒使用者可能有詐騙風險的疑慮，也能檢查自己的個人資料是否外洩，例如檢查 Email 是否外洩；還能檢查陌生的 Line ID 是否為詐騙的 ID；也可以上傳文章放在機器人的記事本裡，讓使用者不只可以直接詢問，也可以透過文章來觀看。

效益方面: 減少使用者在網路上不管是閱覽文章或者是可疑的網站可能帶來的風險和損失，增加個人的防護意識與防範的能力，使得個人資料不會輕易的外洩出去，也可以幫助在遏制詐騙犯罪身上，維護社會正義。

創新方面:可以透過跟其他平台的協作與整合，形成一個反詐騙的網絡，例如結合 165 反詐騙網站的資料，讓 Line bot 辨別能力更加的強大。

第2章　營運計畫

2-1 可行性分析

可行性研究，是管理學、投資學、市場學及工程學中常見的術語和前期準備過程，簡單而言即是計畫及考慮某項目建議的可行與否。大型項目開始之前，當然必須要有一個理性的及詳實的可行性研究，以確定工程或投資計劃是否可行，一般包括環評、安全評價、社會影響評價、地址影響評價等多方面。本研究針對「詐騙一指通」這個專案進行可行性研究，主要有四個要素：技術可行性、財務可行性、市場可行性(或市場契合度) 和營運可行性。如下敘述:

一、技術可行性

本專案製作為學校專題項目，故無與相關企業結合資源，能提供給使用者的功能非常有限，因此技術可行性偏低。

二、財務可行性

本專案在製作時並沒有產生任何費用，也沒有向使用者收取任何費用，故無收益及成本，整個專題製作無財務可行性。

三、市場可行性

雖然現在市場已有類似的功能的網站或軟體，但是數目並非常廣泛，因此市場可行性偏高。

四、營運可行性

本專案製作為學校專題項目，故暫時並無上市給外在消費者使用，只提供給指導老師及專題負責人員使用並測試。

2-2 商業模式－Business model

商業模式是指為實現各方價值最大化，把能使企業運行的內外各要素整合起來，形成一個完整的、高效率的、具有獨特核心競爭力的運行系統，並通過最好的實現形式來滿足客戶需求、實現各方價值（各方包括客戶、員工、合作夥伴、股東等利益相關者），同時使系統達成持續贏利目標的整體解決方案。簡單來說，商業模式，描述與規範了一個企業創造價值、傳遞價值以及獲取價值的核心邏輯和運行機制。本研究針對「詐騙一指通」以 商業模式圖作為本研究主要的描述與定義，結果整理如下表。

<table>
<tr><th>關鍵合作夥伴</th><th>關鍵活動</th><th>價值主張</th><th>顧客關係</th><th>目標客群:</th></tr>
<tr><td rowspan="3">·指導老師
·助教
·團隊/組員</td><td>·市場分析
·品牌定位
·產品設計</td><td rowspan="3">·查詢可疑網址
·辨識EMAIL
·辨識LINEID</td><td>·一般顧客</td><td rowspan="3">·詐騙受害者
·線上交易者
·年邁老人
·經常網購</td></tr>
<tr><td>關鍵資源</td><td>通路</td></tr>
<tr><td>·系統平台
·IT技術</td><td>·LINE</td></tr>
<tr><td>成本結構</td><td colspan="2" rowspan="2">·使用LINE Business ID免費開設帳號，所以0成本。</td><td>收益流</td><td rowspan="2">沒有向使用者收費，故無收益流。</td></tr>
<tr><td colspan="2"></td></tr>
</table>

▲圖 2-2-1 商業模式圖

2-3 市場分析－STP

STP 為三個英文單字的組合，分別是市場區隔 S (Segmentation)、目標市場 T (Targeting)、定位 P (Positioning)，也就是將廣大市場區隔開來，再從中找到目標市場，最後在目標市場中找到自己的定位。隨著時代的變遷，詐騙手法越來越多元，因此也造成許多受害者。為了不讓更多人成為受害者，近年來市場上有許多公家機關或是企業行號開始設計一些防詐騙的系統，根據詐騙手法、受害者的性別、年齡等等做出合適的反詐騙軟體。本研究針對「詐騙一指通」以 STP 分析作為本研究主要的描述與定義。結果整理如下。

1. S (Segmentation) 市場區隔：了解各個市場的不同

現在防詐騙系統除了有網頁、APP 之外還有現在比較流行的 LINE BOT，而有關防詐騙的 LINE BOT 又有分自動回覆及真人服務，提供給使用者的功能也不同。

- 地理：台灣地區，國家規模 23,500,000 人，密度為都市。
- 人口：鎖定台灣地區 20-60 歲族群
- 性別：男女不分
- 心理：針對詐騙手法經常上當
- 行為變數：經常在網路購買東西，或是在線上交易填寫個人資料

2. T (Targeting) 目標市場：找到適合傳達訊息的市場

「詐騙一指通」的出發點是因為近年來網頁商城平台會有詐騙集團偽裝店家或是客服人員,傳送詐騙連結或者訊息給消費者,因此可以推測其目標客群鎖定為：

- 20-40 歲女性以及年長者
- 頻繁網購且線上交易

- 忠誠度可能較高

3. P (Positioning)品牌定位：在目標市場中站穩腳步

現在人手一機已是常態又加上網路科技的發達，常常都會收到一些詐騙訊息，本研究「詐騙一指通」利用 LINE BOT 製作一個查詢可疑網址或辨識 EMAIL 及　的機器人。幫助使用者 能夠避免在類似購物商城官方的詐騙連結,減少自身被騙的風險。

2-4 競爭力分析 SWOT-TOWS 或五力分析

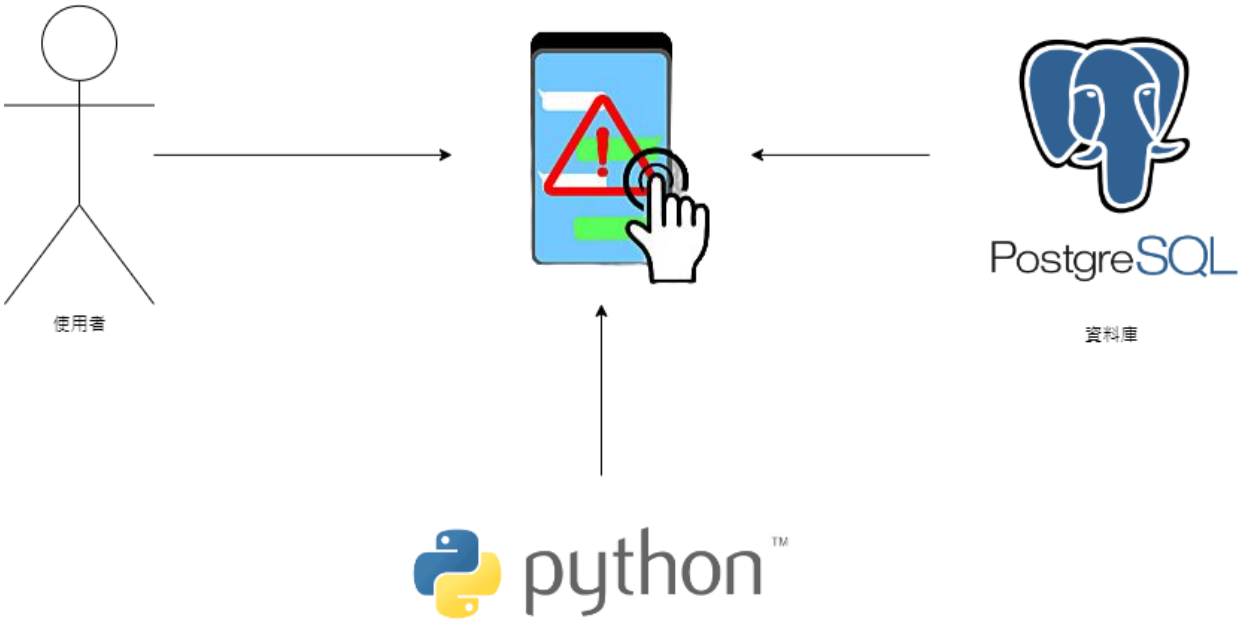
強弱危機分析（英語：SWOT Analysis），又稱優劣分析法、SWOT 分析法或道斯矩陣，是一種企業競爭態勢分析方法，是市場行銷的基礎分析方法之一，透過評價自身的優勢（Strengths）、劣勢（Weaknesses）、外部競爭上的機會（Opportunities）和威脅（Threats），用以在制定發展戰略前對自身進行深入全面的分析以及競爭優勢的定位。而此方法是 Albert Humphrey 所提。本研究針對「詐騙一指通」以 SWOT 分析作為本研究主要的描述與定義，其次分就討論內容加以整理，並分別歸類來作區隔，在內部組織方面為優勢與劣勢；在外部環境方面則歸類為機會、威脅，結果整理如表 2-4-1 。

▼表 2-4-1 詐騙一指通 SWOT 分析表

	優 勢	劣 勢
內部組織	1. 方便性:由於時代的進步，人手一機已是常態，而最常使用的聊天軟體非 LINE 莫屬。所以我們利用 LINE BOT 製作一個查詢可疑網址或辨識 EMAIL 及陌生 LINE ID 的機器人。讓使用著可以透過複製、貼上這兩個簡單的動作即可驗查。 2. 為警政署增加有關詐騙的資訊:在 LINE BOT 的選單中，我們有設計一格按鍵，點選後是可直接進入到內政部警政署 165 全民防騙官網，目的是為了讓使用者能檢舉通報，而對於警政署來說是增加詐騙案例，是往後要防範的重要參考資料。	1. 因沒有相關資源協助，故可提供給使用者的功能非常有限，例如:如果能夠與 Whoscall 合作，我們將能再提供電話號碼查詢服務等等… 2. 萬一遇到資料庫沒有的資料訊息，無法提供真人一對一的訊息查證服務，最後只能回復使用者查無訊息等回覆。
外部環境	**機 會**	**威 脅**
	1. 目前暫無學生做出類似驗證詐騙訊息等功能軟體或網站。 2. 雖然現在市場已有類似的功能的網站或軟體，但是數目不是倒非常廣泛。	1. 現在社會的詐騙手段與日俱新，在未來不一定能跟上新的詐騙手段。 2. 現階段類似功能的網站或是軟體，大部分都有配合的合作廠商，能提供的服務項目也比我們多很多，資料也比我們精準，對於使用者來說不見的會使用我們的 LINE BOT。

第 3 章　系統規格

3-1　系統架構



▲圖 3-1-1　系統架構

3-2　系統軟、硬體及需求平台

▼表 3-2-1　軟、硬體及需求平台清單

軟體需求	Line
硬體需求	個人電腦，手機
技術平台	Linux, Windows, macOS

3-3　開發標準與使用工具

▼表 3-3-1　開發標準與使用工具清單

作業系統	Windows 10
資料庫	PostgreSQL
開發環境	Visual Studio Code , Git-Fork
程式語言	Python

第 4 章　　專案時程與組織分工

4-1 專案時程：甘特圖或 PERT／CPM 圖

▲圖 4-1-1　甘特圖

4-2 專案組織與分工。

▼表 4-2-1 專案組織與分工

項目/組員		N1096413 李卉君	N1096416 張峻華	N1096421 吳祖霖	N1096443 楊哲維	N1096452 黃子芸	N1096455 高登
後端開發	資料庫建置		●				○
	資料庫連接		○				●
	伺服器架設						●
	訊息回應邏輯	○	○				●
	資料上傳		●				○
前端設計	自我介紹	●		○		○	
	功能介紹	●		○		●	
	回覆使用者訊息		○				●
	文章發布	○			●		
	165 專線連結		●				○
文件製作	第 1 章 前言	○	●				
	第 2 章 營運計畫	●				●	○
	第 3 章 系統規格	○	○				●
	第 4 章 專題時程與組織分工			○	●		○
	第 5 章 需求模型		●	○			○
	第 6 章 程序模型或設計模型				●		
	第 7 章 資料模型或實作模型		●	○			○
	第 8 章 資料庫設計		○				●
	第 9 章 程式		○				●
	第 10 章 測試模型		○				●
	第 11 章 操作手冊		○				●
	第 12 章 使用手冊		○				●
	第 13 章 感想			●	○		
	第 14 章 參考資料	●					○
美術設計	UI/ UX	●				○	
	APP 介面設計	●				○	
	色彩設計	●				○	
	Logo 設計	●				○	
	素材設計	●				○	
報告	簡報製作	●				●	

第 5 章　需求模型

5-1 使用者需求

▼表 5-1-1 功能性需求

項目	事件名稱	觸發器	來源	活動	回應	目的地
1.	點擊反詐騙 LINE BOT	LINE BOT	民眾	點擊 LINE BOT	跳出反詐騙 LINE BOT 自我介紹	LINE BOT 主畫面
2.	點擊 功能介紹	功能介紹	民眾	點擊 功能介紹	跳出 自動回應 訊息	LINE BOT 主畫面
3.	點擊 檢舉圖示	檢舉圖示	民眾	點擊 檢舉圖示	連結警政署 165 官網	警政署 165 官網
4.	點擊 分享好友	分享好友圖示	民眾	點擊 分享好友	跳出分享 LINE 好友	LINE 分享好友畫面
5.	觀看 反詐騙文章	首頁貼文	民眾	點擊 貼文	跳出 反詐騙相關 文章	貼文頁面
6.	查詢可疑 網址、信箱、帳號	小鍵盤	民眾	點擊 小鍵盤	跳出 小鍵盤	打字介面
7.	登錄自我介紹	自動回應訊息 建立	企業	登錄 自我介紹	跳出模擬 LINE 介面	模擬 LINE BOT 主畫面
8.	登錄功能介紹	自動回應訊息 建立	企業	登錄 功能介紹	跳出模擬 LINE 介面	模擬 LINE BOT 主畫面
9.	登錄反詐騙 文章	建立新貼文	企業	登錄 反詐騙 文章	建立貼文 頁面	貼文頁面

非功能性需求：

1. 穩定的系統

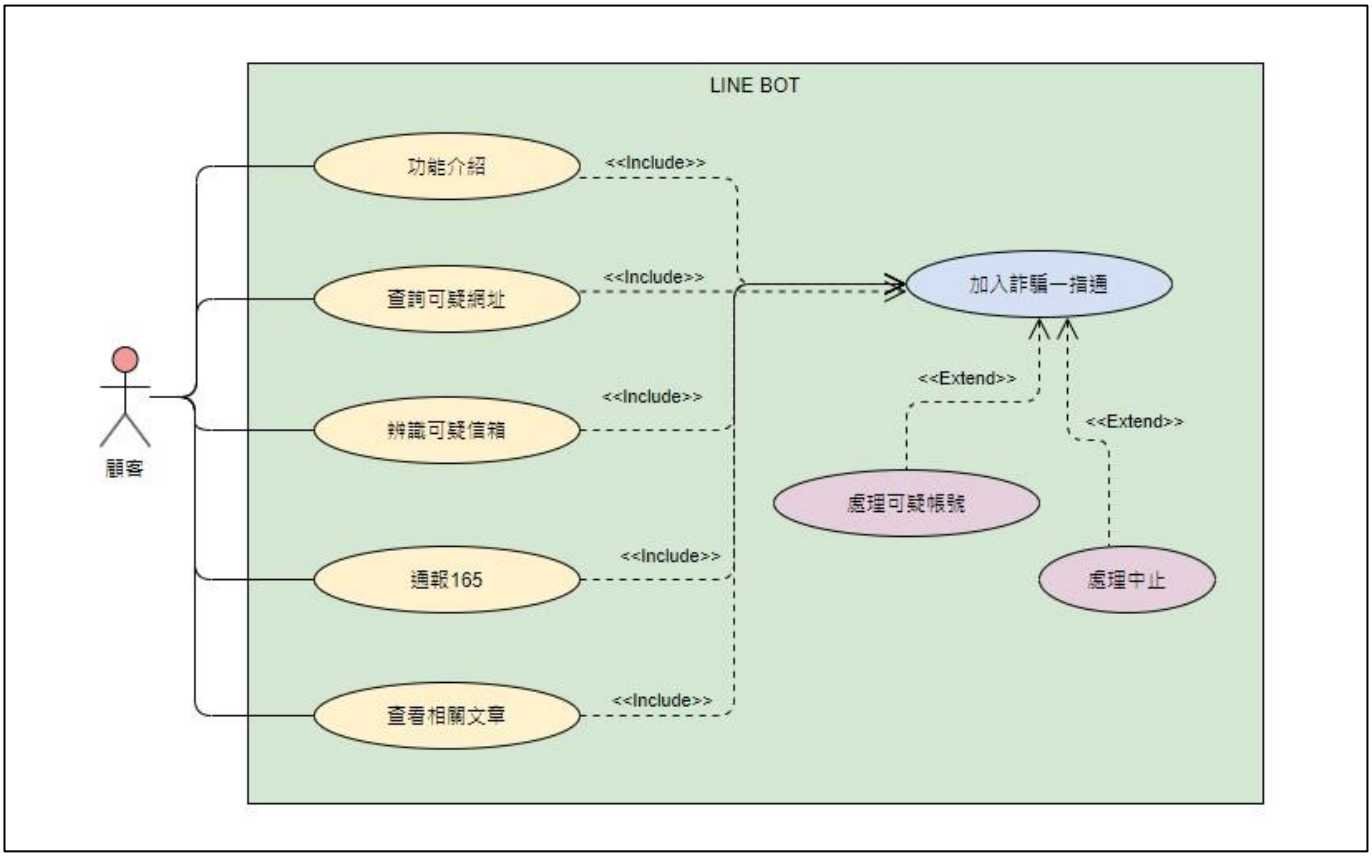
2. 易上手的操作、圖型界面介面

3. 可簡易擴充性功能

4. 系統管理者方便管理

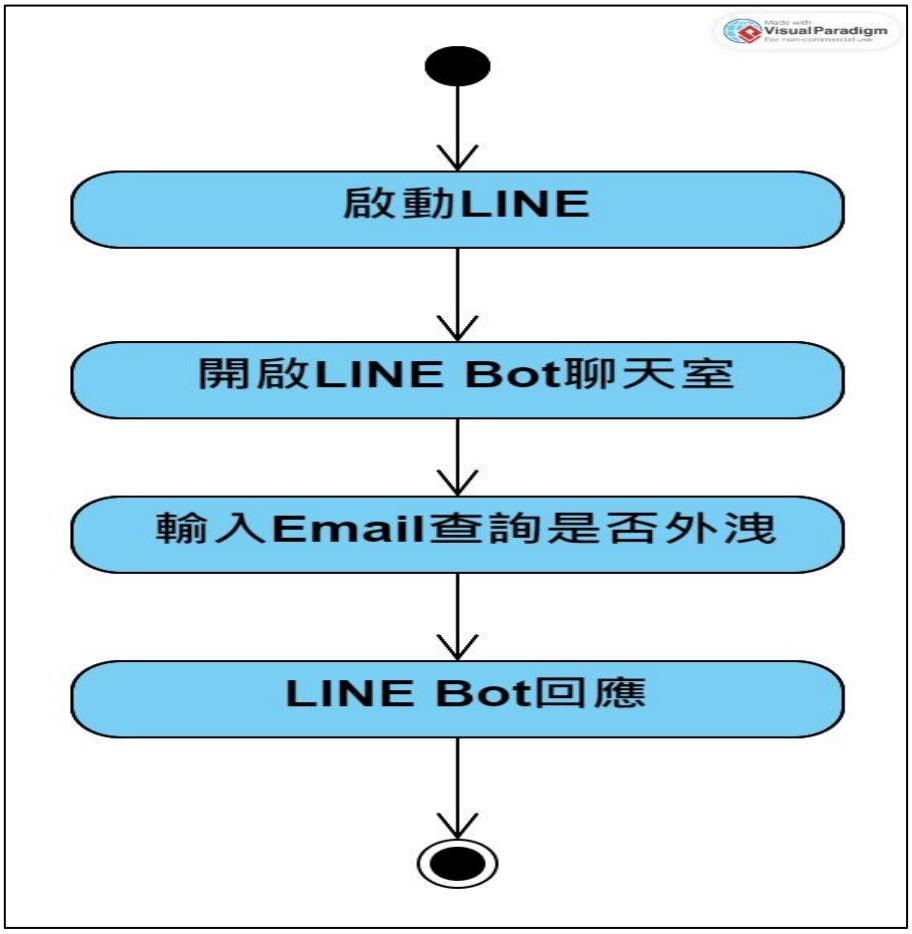
5. 簡單明瞭的規則提示

5-2 使用個案圖



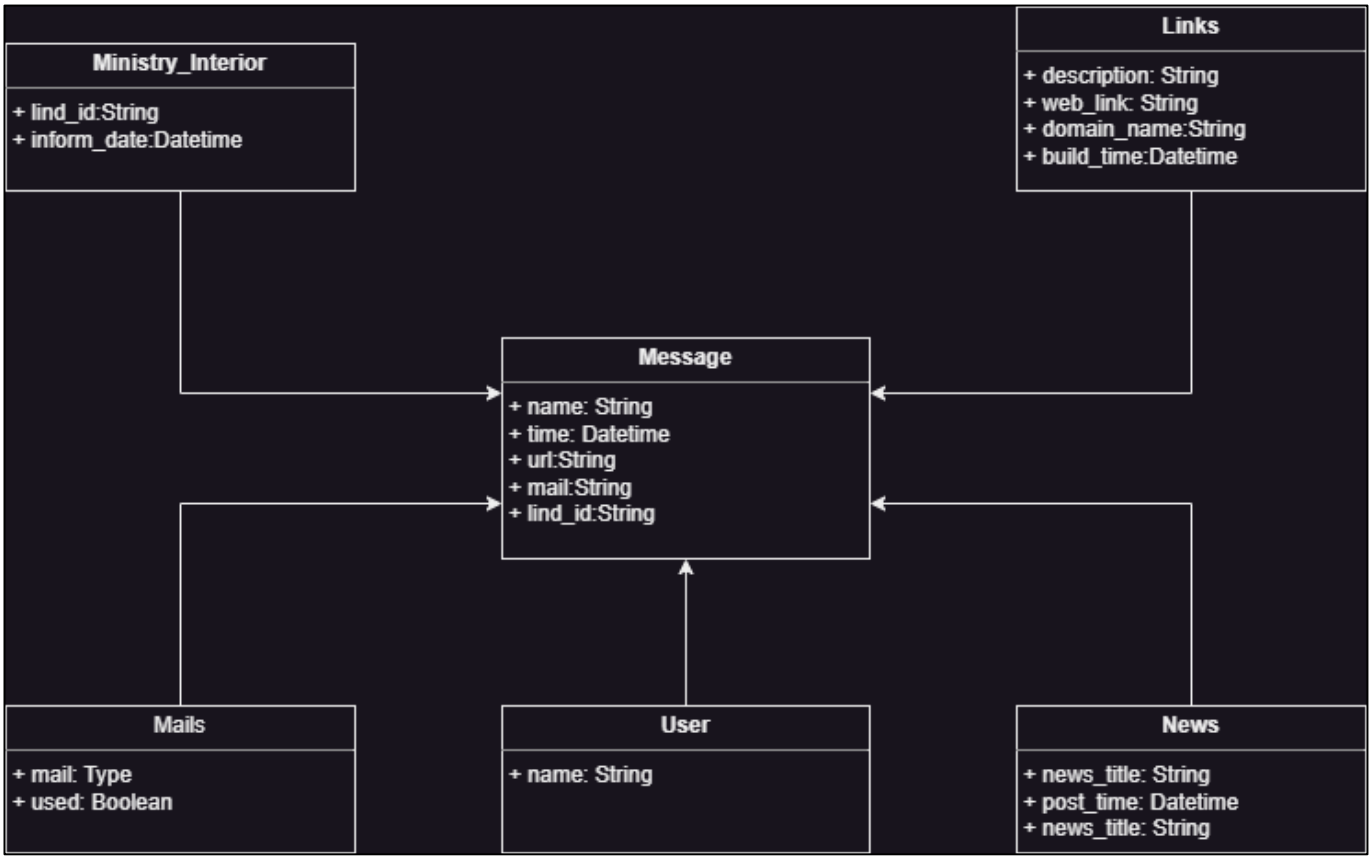
▲圖 5-2-1 使用個案圖

5-3 使用個案描述



▲圖 5-3-1 活動圖

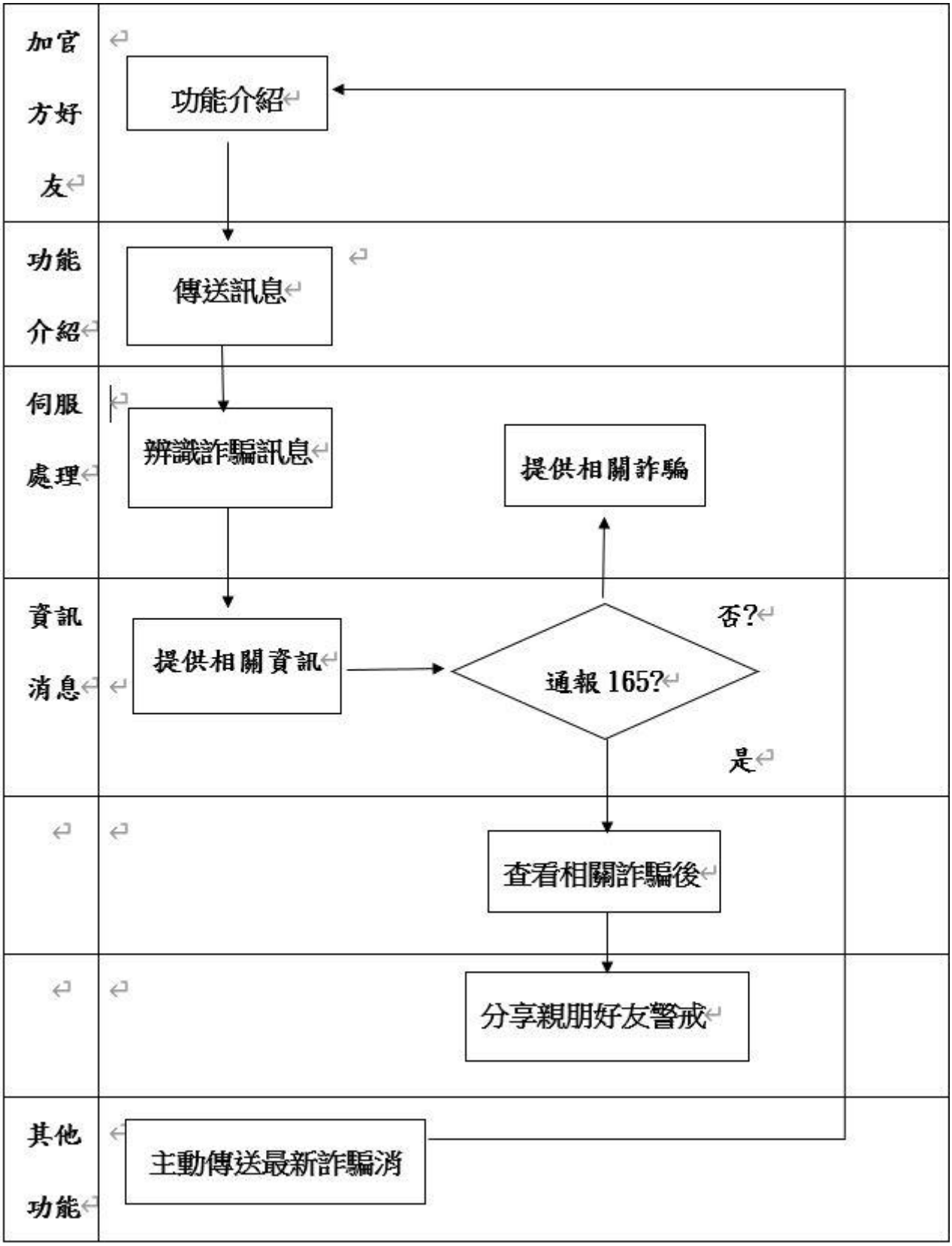
5-4 分析類別圖



▲圖 5-4-1 分析類別圖

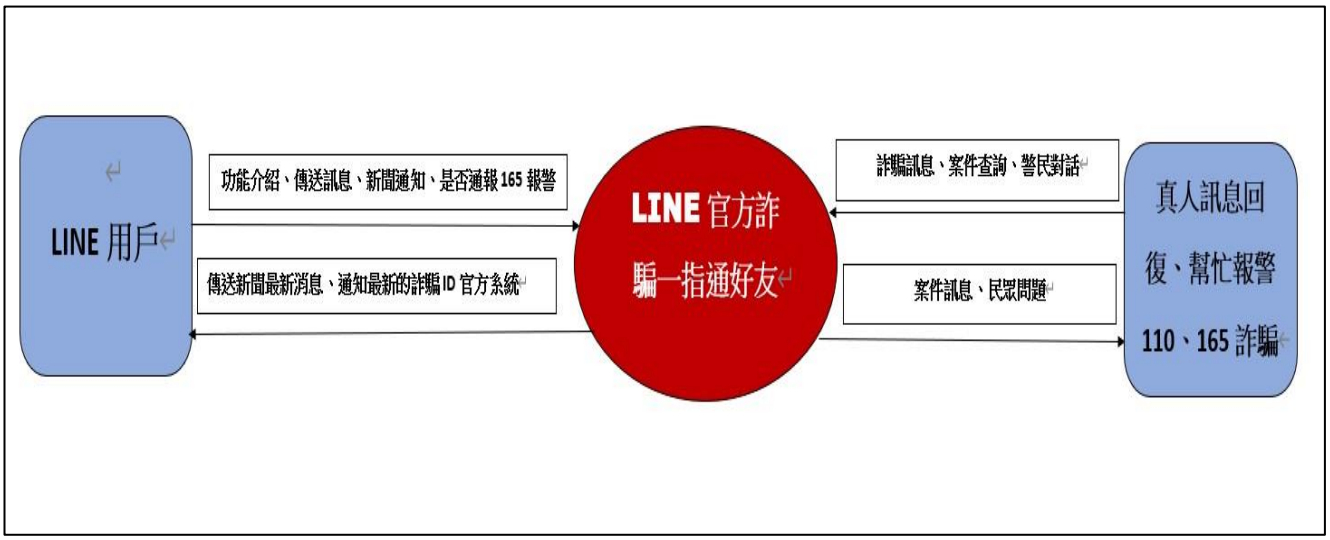
第 6 章　設計模型

6-1 循序圖或通訊圖



▲圖 6-1-1 循序圖

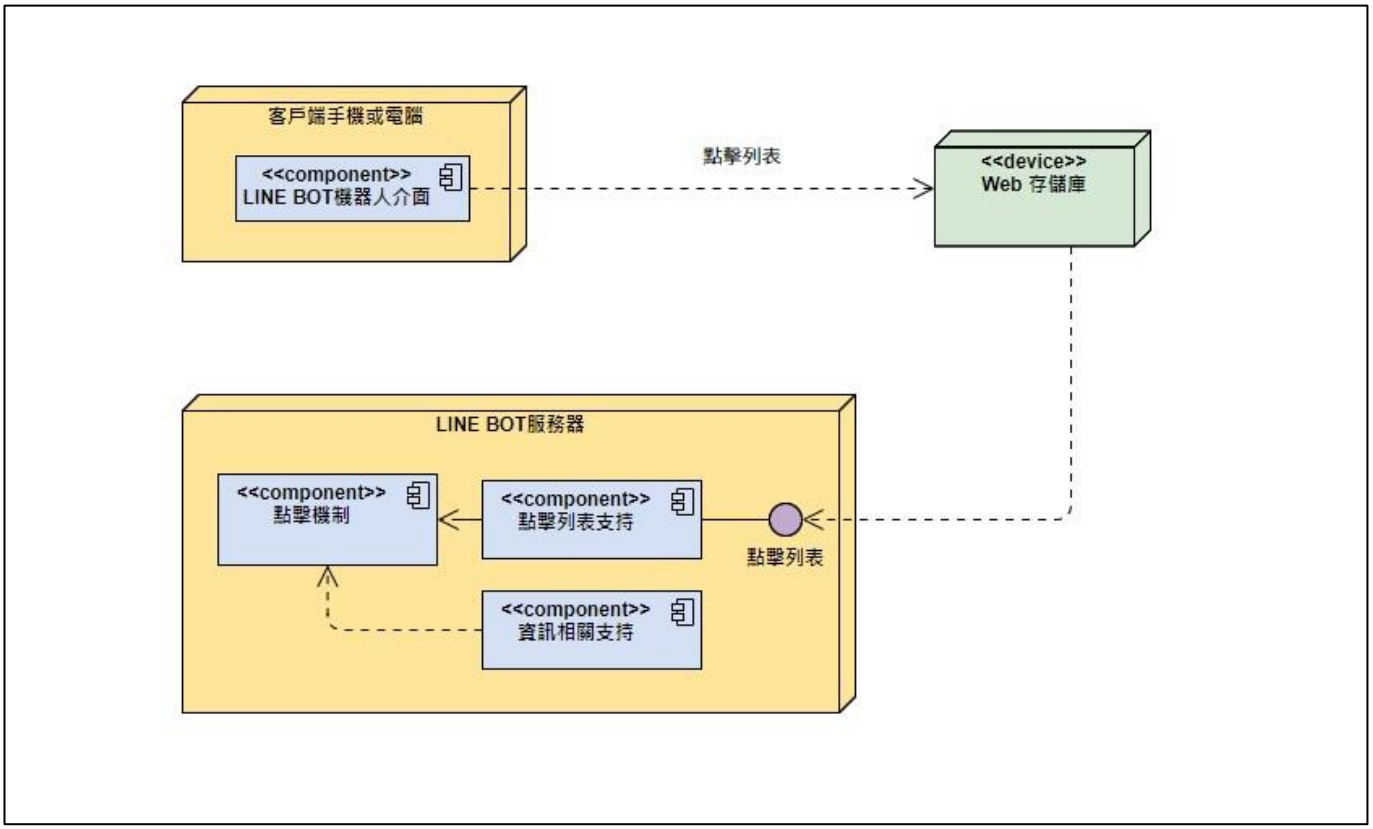
6-2 設計類別圖



▲圖 6-2-1 設計類別圖

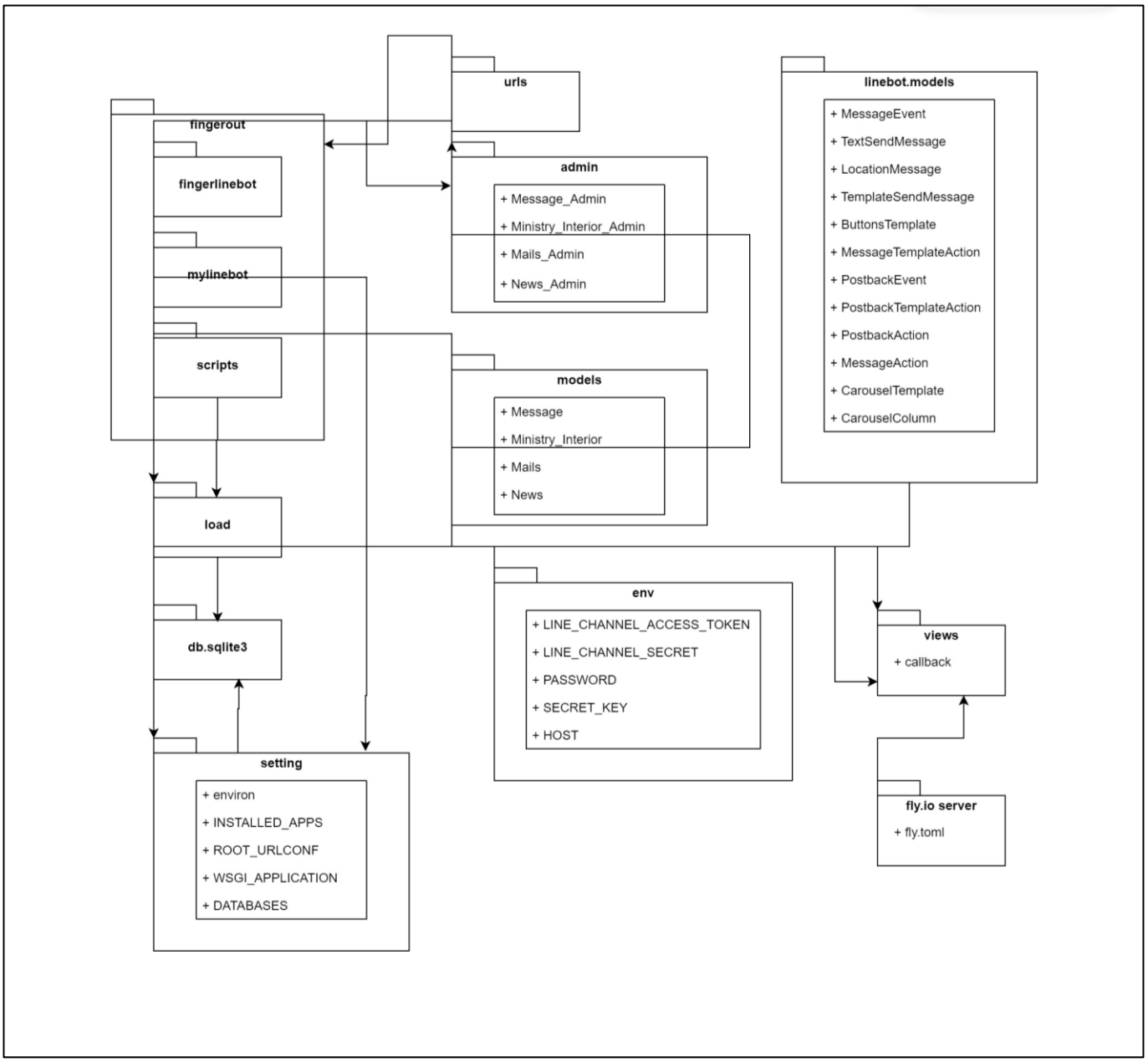
第 7 章　實作模型

7-1 部屬圖



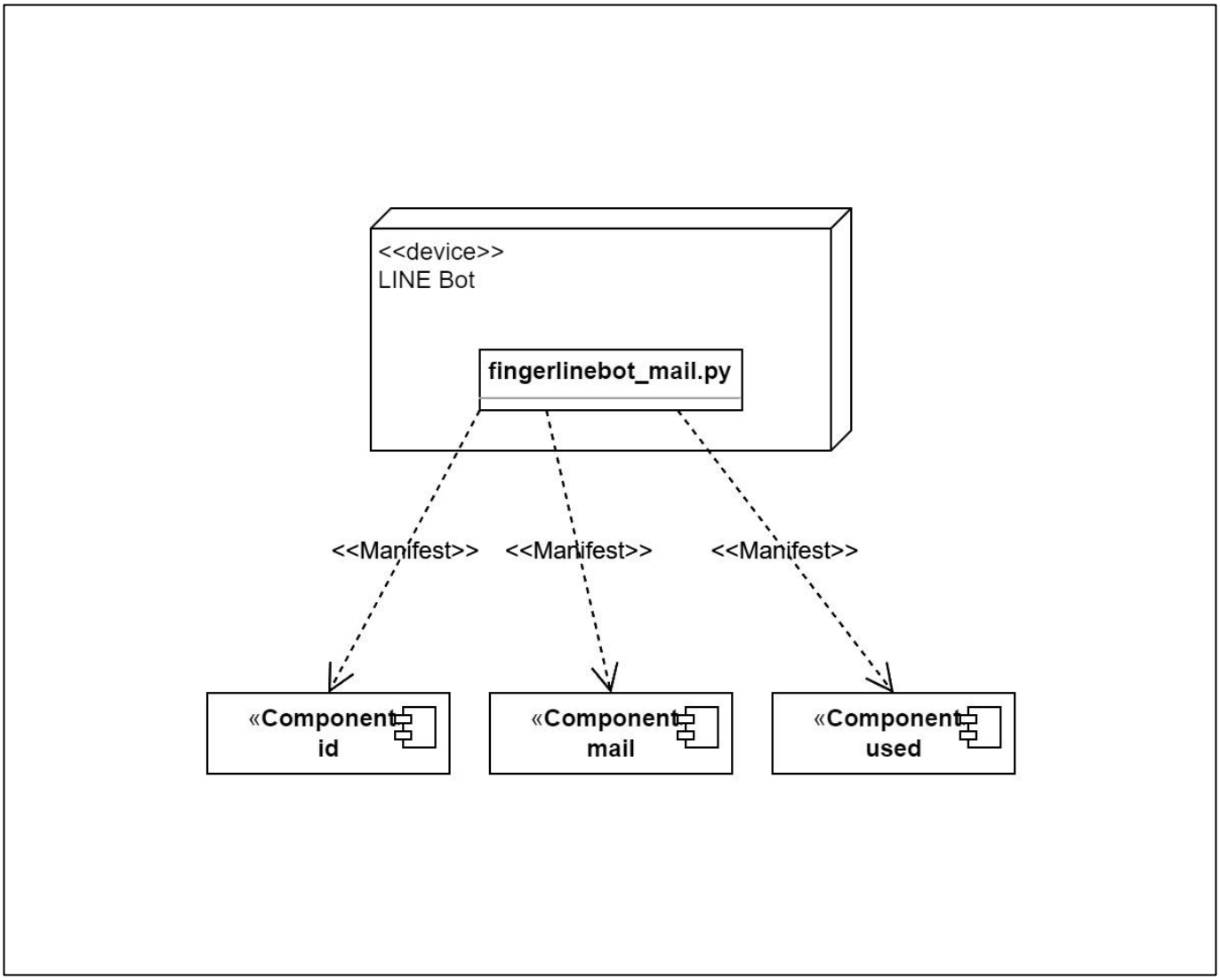
▲圖 7-1-1 部屬圖

7-2 套件圖



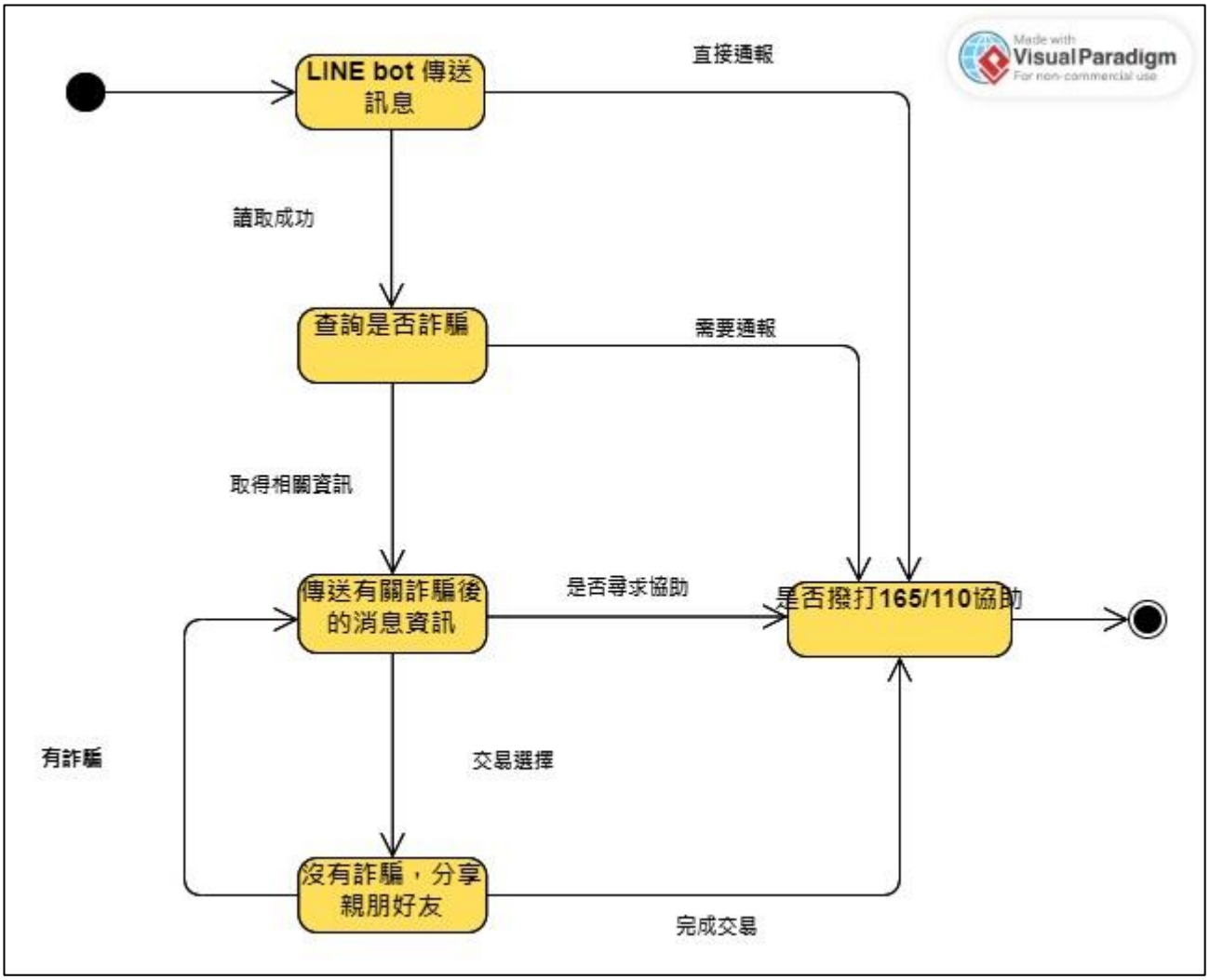
▲圖 7-2-1 套件圖

7-3 元件圖



▲圖 7-3-1 元件圖

7-4 狀態圖

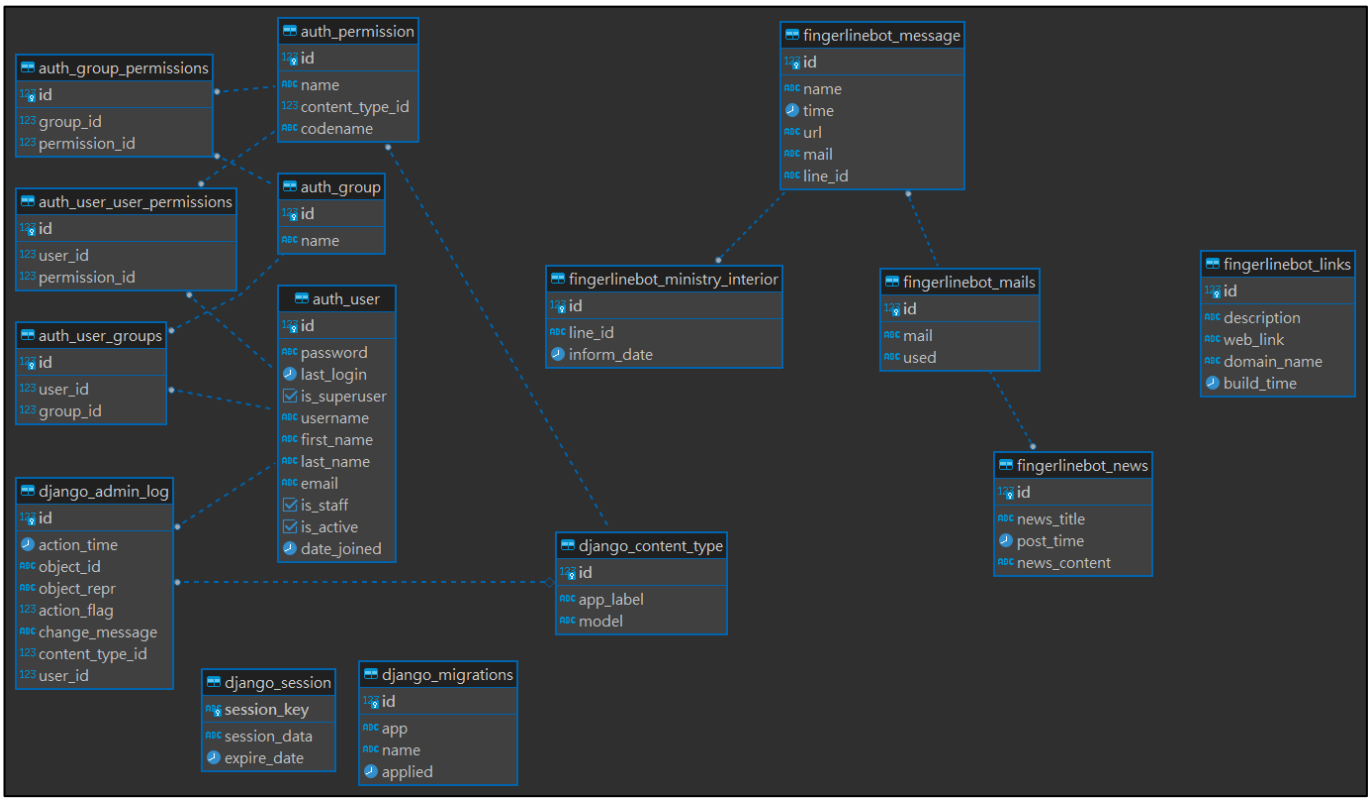


▲圖 7-4-1 狀態圖

第 8 章　資料庫設計

8-1 資料庫關聯表

▼表8-1-1　資料庫關聯表圖



8-2 表格及其 Meta data

▼表8-2-1　Message 資料表

名稱	類型	是否為空	欄位說明
id	bigserial	否	主要鍵值（PK）
name	varchar(50)	否	名稱
time	timestamptz	否	時間
url	varchar(200)	否	詐騙連結
mail	varchar(50)	否	使用者信箱
line_id	varchar(50)	否	詐騙 Lind ID

▼表8-2-2　Ministry_Interior 資料表

名稱	類型	是否為空	欄位說明
id	bigserial	否	主要鍵值（PK）
line_id	varchar(50)	否	詐騙 Lind ID
inform_date	timestamptz	是	時間

▼表8-2-3　Mails 資料表

名稱	類型	是否為空	欄位說明
id	bigserial	否	主要鍵值（PK）
mail	varchar(50)	否	使用者信箱
used	timestamptz	否	是否被使用

▼表8-2-4　Links 資料表

名稱	類型	是否為空	欄位說明
id	bigserial	否	主要鍵值（PK）
description	varchar(50)	否	使用者信箱
web_link	varchar(500)	否	是否被使用
domain_name	varchar(50)	否	詐騙網址
build_time	timestamptz	否	收入的時間點

第 9 章　程式

9-1 元件清單及其規格描述

▼表 9-1-1　LineBot 前端程式檔案

回覆訊息	
檔案名稱	檔案描述
View.py	Linebot 訊息回覆

▼表 9-1-2　Line Bot 後端程式檔案

連接伺服器	
檔案名稱	檔案描述
url.py	Line Bot 應用程式的連結網址
urls.py	設定應用程式的網址
wsgi.py	指定接口並位於服務器和應用程序或框架之間
asgi.py	解決 WSGI 不支持當前 Web 新的協議標準

▼表 9-1-3　Line Bot 資料庫檔案

資料表製作	
檔案名稱	檔案描述
model.py	建立資料表模型
settings.py	與 Line bot、資料庫連結
資料表上傳	
admin.py	註冊資料表更新到資料庫後台網站
admin.py	建立資料模型到網站資料庫
app.py	連接資料模型到專案
資料上傳	
load.py	把 csv 檔所有資料上傳到資料庫

▼表 9-1-4　Line Bot 資料檔案

管理資料	
檔案名稱	檔案描述
NPA_165.csv	詐騙新聞所有資料
NPA_LineID.csv	LineID 所有資料
NPA_link.csv	詐騙網址所有資料

▼表 9-1-5　admin.py 程式碼

admin.py

```
from django.apps import AppConfig


class FingerlinebotConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'fingerlinebot'
```

▼表 9-1-6　apps.py 程式碼

apps.py

```
from django.contrib import admin
from fingerlinebot.models import *

# Register your models here.
class Message_Admin(admin.ModelAdmin):
    list_display = ('name','time','url','mail','line_id')


class Ministry_Interior_Admin(admin.ModelAdmin):
    list_display = ('line_id','inform_date')
admin.site.register(Ministry_Interior,Ministry_Interior_Admin)

class Mails_Admin(admin.ModelAdmin):
    list_display = ('mail','used')
admin.site.register(Mails,Mails_Admin)

class News_Admin(admin.ModelAdmin):
    list_display = ('news_title','post_time','news_content')
admin.site.register(News,News_Admin)

class Links_Admin(admin.ModelAdmin):
    list_display = ('description','web_link','domain_name','build_time')
admin.site.register(Links,Links_Admin)
```

▼表 9-1-7　models.py 程式碼

models.py

```
from django.db import models


# Create your models here.
class Message(models.Model):
    name = models.CharField(max_length)
    time = models.DateTimeField(auto_now_add)
    url = models.URLField(max_length)
    mail = models.CharField(max_length)
    line_id = models.CharField(max_length)

    def __str__(self):
        return self.name,self.time,self.url,self.mail,self.line_id


class Ministry_Interior(models.Model):
    line_id = models.CharField(max_length)
    inform_date = models.DateField(auto_now)

    def __str__(self):
        return self.line_id,self.inform_date

class Mails(models.Model):
    mail = models.CharField(max_length)
    used = models.CharField(max_length)

    def __str__(self):
        return self.mail,self.used


class News(models.Model):
    news_title = models.CharField(max_length)
    post_time = models.DateTimeField(auto_now)
    news_content = models.CharField(max_length)

    def __str__(self):
        return self.news_title,self.post_time,self.news_content
```

```
class Links(models.Model):
        description = models.CharField(max_length)
        web_link = models.CharField(max_length)
        domain_name = models.CharField(max_length)
        build_time = models.DateTimeField(auto_now)

        def __str__(self):
                return
self.description,self.web_link,self.domain_name,self.build_time
```

▼表 9-1-8　urls.py 程式碼

urls.py

```
from django.urls import path
from . import views

urlpatterns = [
        path('callback', views.callback)
]
```

▼表 9-1-9 view.py 程式碼

view.py

```
from django.shortcuts import render

# Create your views here.'
from django.http import HttpResponse, HttpResponseBadRequest,
HttpResponseForbidden
from django.views.decorators.csrf import csrf_exempt
from django.conf import settings
from django.core import serializers
from django.forms import model_to_dict

from linebot import LineBotApi, WebhookParser
from linebot.exceptions import InvalidSignatureError, LineBotApiError

from models import (
    MessageEvent,
    TextSendMessage,
    LocationMessage,
    TemplateSendMessage,
    ButtonsTemplate,
    MessageTemplateAction,
    PostbackEvent,
    PostbackTemplateAction,
    PostbackAction,
    MessageAction,
    CarouselTemplate,
    CarouselColumn
)

from .models import *


import re
import urllib.parse
import requests
import json

line_bot_api                                                    =
LineBotApi(settings.LINE_CHANNEL_ACCESS_TOKEN)
```

```
parser = WebhookParser(settings.LINE_CHANNEL_SECRET)



#篩選資料庫資料庫裡的 News 資料


news_data = json.dumps(list(news_information))
news_result = json.loads(news_data)

#篩選資料庫資料庫裡的 Ministry_Interior 資料


lineid_data = json.dumps(list(lineid_information))
lineid_result = json.loads(lineid_data)



@csrf_exempt
def callback(request):

    if request.method == 'POST':
        signature = request.META['HTTP_X_LINE_SIGNATURE']
        body = request.body.decode('utf-8')

        try:

            events = parser.parse(body, signature)   # 傳入的事件

        except InvalidSignatureError:
            return HttpResponseForbidden()
        except LineBotApiError:
            return HttpResponseBadRequest()

        for event in events:

            if isinstance(event, MessageEvent):   # 如果有訊息事件


                #使用者的訊息

                user_msg = event.message.text.lower()
```

```
                #比照資料庫裡的 Mails 資料
                accounts = Mails.objects.filter(mail=user_msg)
                #回覆給使用者訊息
                message = []

                #功能選單
                if user_msg == '更多功能':
                    line_bot_api.reply_message(
                        event.reply_token,
                        TemplateSendMessage(
                        alt_text='Carousel template',
                        template=CarouselTemplate(
                            columns=[
                                CarouselColumn(
                                    title='詐騙新聞情報',
                                    text='顯示目前最新新聞',
                                    actions=[
                                        PostbackAction(
                                            label='新聞情報',
                                            text='新聞',
                                            data='新聞'
                                        ),
                                    ]
                                ),
                                CarouselColumn(
                                    title='詐騙 Line ID',
                                    text='顯示目前最近所有的
Line ID',
                                    actions=[
                                        PostbackAction(
```

```
                                        label=' 詐 騙  Line
ID',
                                        text='詐騙',
                                        data='詐騙'
                                    ),
                                ]
                            ),
                            CarouselColumn(
                                title='查詢帳號',
                                text=' 查 詢 目 前 帳 號 是 否 存
在並幫您查詢有被外洩的問題',
                                actions=[
                                    PostbackAction(
                                        label='查詢帳號',
                                        text='查詢帳號',
                                        data='查詢帳號'
                                    ),
                                ]
                            ),
                            CarouselColumn(
                                title='新增帳號',
                                text='新增帳號，以助於之後
能夠查詢是否有被外洩的問題',
                                actions=[
                                    PostbackAction(
                                        label='新增帳號',
                                        text='新增帳號',
```

```
                                                    data='新增帳號'
                                                ),
                                            ]
                                        )
                                    ]
                                )
                            )
                        )

                    if user_msg == '新聞':
                        message.clear()
                        title = news_result[0]['news_title']
                        content = news_result[0]['news_content']
                        message.append(TextSendMessage(text='最新新聞：'+ '\n' + title + '\n' + '內容：' + content))

                    if user_msg == '詐騙':
                        message.clear()
                        message.append(TextSendMessage(text='最近的詐騙 ID: '+ '\n' + lineid_data))

                    if user_msg == '查詢帳號' or user_msg == '新增帳號':
                        message.clear()
                        message.append(TextSendMessage(text='請輸入帳號'))

                    if accounts.exists() and '@'in user_msg:
                        message.append(TextSendMessage(text='有此帳號歐'))
```

```
                elif '@' not in user_msg:
                    pass
                else:
                    message.append(TextSendMessage(text='無此帳
號歐 '+'\n'+'正在幫您把目前輸入的帳號新增置資料庫'))

                    Mails.objects.create(mail=user_msg)

                    if accounts.exists():
                        message.append(TextSendMessage(text='已
幫您新增至資料庫，請等待人員幫您檢測帳號是否有被外洩'))
                    else:
                        message.append(TextSendMessage(text='無
法新增至資料庫，請等待人員幫您檢測'))



                for account in accounts:
                    #判斷使用者輸入的帳號是否被使用過,回復傳
入的訊息文字
                    if user_msg == account.mail and account.used ==
'是':
                        message.append(TextSendMessage(text =
account.mail + '\n' +'此帳號有外洩風險！！！'))
                    elif user_msg == account.mail and
len(account.used) == 0:
                        message.append(TextSendMessage(text =
account.mail+'\n'+'目前人員還在幫您檢測，會盡快幫您確認請放心～
'))
```

```
                        else:
                            message.append(TextSendMessage(text     =
account.mail + '\n' +'此帳號無風險，請放心～'))


                    line_bot_api.reply_message(   # 回復傳入的訊息文
字
                        event.reply_token,
                        message,
                    )
            return HttpResponse()
        else:
            return HttpResponseBadRequest()
```

▼表 9-1-10 urls.py 程式碼

urls.py
from django.urls import path from . import views urlpatterns = [path('callback', views.callback)]

▼表 9-1-11　settings.py 程式碼

settings.py

```
"""

Generated by 'django-admin startproject' using Django 4.1.7.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""

from pathlib import Path
import os
import environ

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

environ.Env.read_env(os.path.join(BASE_DIR, '.env'))
env = environ.Env()

# env variables

LINE_CHANNEL_ACCESS_TOKEN                                    =
env('LINE_CHANNEL_ACCESS_TOKEN')

LINE_CHANNEL_SECRET = env('LINE_CHANNEL_SECRET')

ROOT_DIR = (environ.Path(__file__) - 1)



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = env('SECRET_KEY')

# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True


ALLOWED_HOSTS = [
	'*',
]

# Application definition

INSTALLED_APPS = [
	'django.contrib.admin',
	'django.contrib.auth',
	'django.contrib.contenttypes',
	'django.contrib.sessions',
	'django.contrib.messages',
	'django.contrib.staticfiles',
	'django_extensions',
	'fingerlinebot.apps.Config',
]

MIDDLEWARE = [
	'django.middleware.security.SecurityMiddleware',
	'django.contrib.sessions.middleware.SessionMiddleware',
	'django.middleware.common.CommonMiddleware',
	'django.middleware.csrf.CsrfViewMiddleware',
	'django.contrib.auth.middleware.AuthenticationMiddleware',
	'django.contrib.messages.middleware.MessageMiddleware',
	'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF =.urls'

TEMPLATES = [
	{
		'BACKEND':
'django.template.backends.django.DjangoTemplates',
		'DIRS': [os.path.join(BASE_DIR,'templates')],
		'APP_DIRS': True,
		'OPTIONS': {
			'context_processors': [
				'django.template.context_processors.debug',
```

```
                    'django.template.context_processors.request',
                    'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',
                ],
            },
        },
]

WSGI_APPLICATION = 'mylinebot.wsgi.application'



# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases



DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME':,  #資料庫名稱

        'USER':,  #資料庫帳號

        'PASSWORD': env('PASSWORD'),  #資料庫密碼

        'HOST': env('HOST'),  #Server(伺服器)位址

        'PORT': '   #PostgreSQL Port 號

    }
}


# Password validation
#       https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = 'zh-hant'

TIME_ZONE = 'Asia/Taipei'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

▼表 9-1-12　urls.py 程式碼

urls.py

```
"""mylinebot URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
from fingerlinebot.views import main

urlpatterns = [
    path('', main,name='main'),
    path('admin/', admin.site.urls),
    path('fingerlinebot/', include('fingerlinebot.urls')) #包含應用程式的網址
]
```

▼表 9-1-13　wsgi.py 程式碼

wsgi.py

```
"""

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE'.settings')

application = get_wsgi_application()
```

▼表 9-1-14　load.py 程式碼

load.py

```
import csv
import os
from .models import Ministry_Interior,News,Links

def run():
    read_file = csv.reader(file)

    #optional
    Links.objects.all().delete()
    # Ministry_Interior.objects.all().delete()
    # News.objects.all().delete()

    count = 1

    for read in read_file:
        if count == 1:
            pass
        else:
            print(read)
            Links.objects.create(description=read[1],web_link=read[2]
,domain_name=read[3],build_time=read[4])
            #
Ministry_Interior.objects.create(line_id=read[1],inform_date=read[2])
            #
News.objects.create(news_title=read[1],post_time=read[2],news_content=read[3])
        count += 1
```

▼表 9-1-15　manage.py 程式碼

manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE' .settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

9-2 其他附屬之各種元件

▼表 9-2-1 本機設定檔案

內部設定	
檔案名稱	檔案描述
settings.py	整個專案的主設定檔
.env	存放機密內部資訊
.env.template	只顯示要放的資訊欄位
Dockerfile	更新到伺服器的基本設定
fly.toml	伺服器連接到本機檔案
manage.py	設定環境變數，指向 settings.py
requirements.txt	安裝所需的模組，設定版本

第 10 章　測試模型

10-1 測試計畫

Line bot

1. 測試 Email 查詢是否有外洩的功能

2. 測試查詢詐騙的 Line id 的功能

3. 測試在聊天室輸入「新聞」這個關鍵字會自動回覆一篇最新新聞的功能

10-2 測試個案與測試結果資料

▼表 10-2-1　Email 查詢是否有外洩的功能測試結果表

測試項目編號	1	測試項目內容	Email 外洩查詢的關鍵字是否正常
測試流程	1. 開啟 Line bot 聊天室 2. 在聊天室欄直接輸入自己的 Email，例如: n1096416@ntub.edu.tw，會先將電子郵件加入資料庫，再由人員手動幫忙查詢是否外洩。		
預期結果	電子郵件需順利加入資料庫		
最終結果	可以順利加入資料庫		

▼表 10-2-2 查詢詐騙的 Line id 的功能測試結果表

測試項目編號	2	測試項目內容	詐騙的 Line id 查詢的關鍵字是否正常
測試流程	1. 開啟 Line bot 聊天室 2. 在聊天室欄直接輸入「詐騙」這個關鍵字，就會列出近期 165 反詐騙所提供的詐騙 Line id 的名單		
預期結果	順利列出詐騙 Line id 的名單		
最終結果	可以順利列出詐騙 Line id 的名單		

▼表 10-2-3 關鍵字回覆的功能測試結果表

測試項目編號	3	測試項目內容	輸入關鍵字會回覆一篇最新新聞
測試流程	1. 開啟 Line bot 聊天室 2. 在聊天室欄直接輸入「新聞」這個關鍵字，就會列出近期 165 反詐騙所提供的反詐騙新聞		
預期結果	順利回覆反詐騙新聞		
最終結果	可以順利回覆反詐騙新聞		

第 11 章　操作手冊

1. 首先開啟 line，看到主頁右上第三個新增好友點開

▲ 圖11-1-1　　加入好友

2. 在新增好友查詢@340nedab 這一個 line id 把機器人加入好友

▲ 圖11-1-2　　加入機器人

▲ 圖11-1-3　　機器人主畫面

第 12 章　使用手冊

1. 一開始會簡單介紹此系統的功能和解說

▲ 圖12-1-1　　加入畫面

2. 輸入功能介紹，讓使用者了解基本功能

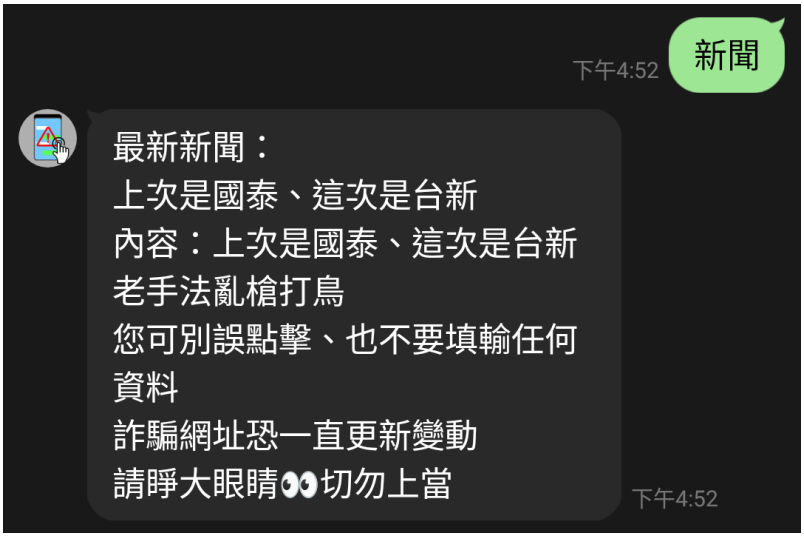
▲ 圖12-1-2　　輸入功能介紹

▲ 圖12-1-3　功能介紹選單

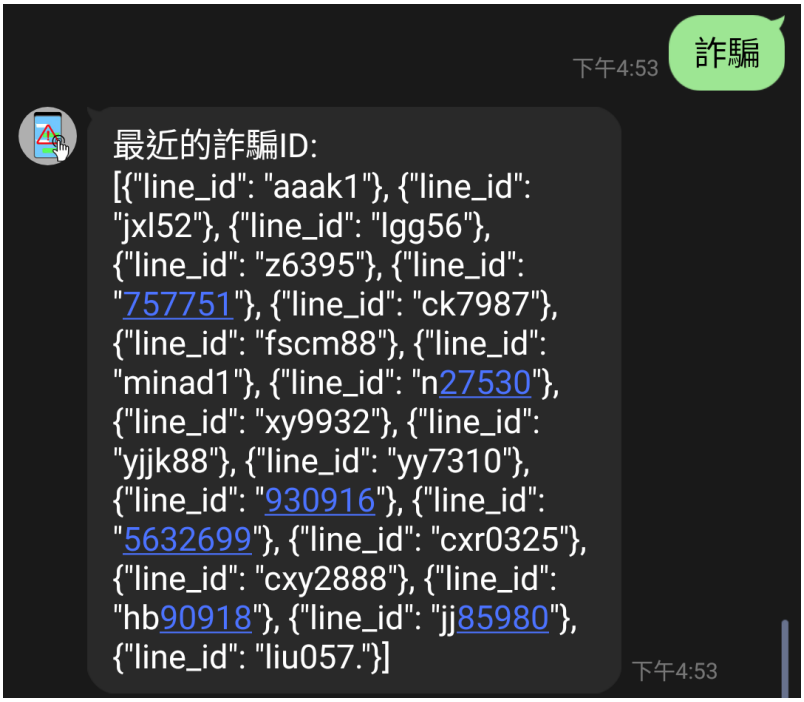
3. 輸入更多功能，裡面還有其他功能能夠使用

▲ 圖12-1-4　輸入更多功能

▲ 圖12-1-5　點選新聞情報



▲ 圖12-1-6　顯示最新詐騙新聞



▲ 圖12-1-7　點選詐騙，顯示所有詐騙 Line ID

第 13 章　感想

關於為什麼我們這次的專題選擇製作防詐騙機器人這個主題，因為近年來詐騙的手法日新月異，近期比較流行的就是可以賺大錢然後到東南亞工作，想當然天底下沒有白吃的午餐，這麼好賺的工作怎麼會留給別人做，但是還是很多人相信並前往，大多數人可能都抱持著賭一把的心態，但也有可能有的人真的缺一筆錢要來生活或是做什麼事，所以心裡一時被迷惑著，這個好在新聞還有大量報導讓很多人認清，但其實最常見的電話詐騙或是中獎、領貨、繳費詐騙等等，才是真正可怕的事情，尤其是老人家或是平常沒什麼在關注這類消息的人，收到訊息或接到電話，對方稍微認真的跟他們說個幾句，可能就會上當受騙而且不會懷疑對方是真的還假的，而這些金額可能是小則幾百元，高則幾萬元到幾十萬元以上的金額不等，再加上源頭難以查找，即使抓到了也只是來頂罪的羔羊而已，因此這種類型的詐騙會一直存在，自己能做到的就是先懷疑到底是不是真的，再來就是查證，現在有很多的反詐騙的管道可以來搜尋，165 警政署也有提供一些相關資訊，現在比起以前有更多資訊、更多管道可以確認事實的真假，也代表有著各式各樣不同的詐騙訊息以假亂真混入其中，一般人是難以分辨清楚的，所以我們提供一個讓大家可以查詢真偽的反詐騙機器人這個管道，當然詐騙也是每天都在更新，我們沒有辦法做到即時的更新，但是對於舊有的詐騙方式，我們希望讓大眾可以用簡單的方式來辨別真假。

第 14 章　參考資料

讓 Linebot 回覆特定訊息

一分鐘學會如何製作 Word 目錄

[Python+LINE Bot 教學]建構具網頁爬蟲功能的 LINE Bot 機器人

[Python+LINE Bot 教學]6 步驟快速上手 LINE Bot 機器人

參考的 Line bot 名稱:

Cofacts 真的假的 | 轉傳查證

Gogolook 美玉姨

MyGoPen 麥擱騙 真人客服

趨勢科技防詐達人

附錄

評審建議事項	修正情形

▼專題成果工作內容與貢獻度表

序號	姓名	工作內容<各限 100 字以內>	貢獻度
1	組長 李卉君	Line bot (logo、封面以及圖表選單）圖片設計製作 Word：系統手冊、第 2 章(營運計畫) PPT 簡報製作，組員分工以及協調各個事務。	16 %
2	組員 高登	Line bot： 前置作業 / 程式設計 / 資料庫建置 / 伺服器架設 Word： 系統架構 / 系統軟、硬體需求與技術平台 / 資料庫設計程式	20 %
3	組員 楊哲維	word 內容:分工表、專案時程與組織分工、實作模型 7-2 套件圖、設計模型 Line bot 機器人前置作業(新聞文章)	16 %
4	組員 吳祖霖	Line bot 機器人前置作業 word 內容:分工表、需求模型、資料模型、感想，會議記錄統整。	16 %
5	組員 黃子芸	Line bot 機器人前置作業及系統自動回覆內容 Word：第 2 章(營運計畫) PPT 簡報製作	16 %

6	組員 張峻華	Line bot: 負責測試功能及回報以及提出如果功能無法實行的解決方案。 Word: 第一章 (前言) 5-3 使用個案描述 7-3 元件圖 第十章 (測試模型) 以及 Word 的調整以及統整，並且做最後的校稿工作	16 %
			總計:100%