milestone3

I am not sure if this is a service other than Redis. We intend to design the bot to provide a function of ordering things for people isolated at home. Many people entering Hong Kong need to isolate themselves at home for 14 days. They can use this bot to upload their location and the list of supplies they need. Here are some sample code.

```
def reply_text(token, id, txt):
    global users
    me = users[id]

    if me['save']  == False:
        if '外賣' in txt:
            queries = ConfirmTemplate(
                text=f"{me['name']}您好，請問要上傳您的所在地嗎？",
                actions=[
                    URIAction(
                        label='上傳地點',
                        uri='line://nv/location'
                    ),
                    MessageAction(label='不需要', text='不需要')
                ])
            temp_msg = TemplateSendMessage(alt_text='確定',
                                           template=queries)
            line_bot_api.reply_message(token, temp_msg)
            me['save'] = True # 開始紀錄訊息
        else:
            line_bot_api.reply_message(
                token,
                TextSendMessage(text="收到訊息了，謝謝！"))
    else:
```

```
        if txt=='不需要':
            line_bot_api.reply_message(
                token,
                TextSendMessage(text="好的，請大致描述狀況。"))
        elif me['logs']['事由'] == '':
            line_bot_api.reply_message(
                token,
                TextSendMessage(text="我記下來了，外賣員马上爲您送達！"))
            me['logs']['事由'] = txt  # 儲存事由
            dt = datetime.now().strftime('%Y/%m/%d %H:%M:%S')
            me['logs']['日期時間'] = dt
            me['save'] = False   # 紀錄完畢

            print('資料紀錄:', me['logs'])
            logs = [id, me['name'], me['logs']['日期時間'],
                        me['logs']['經緯度'], me['logs']['地址'], me['logs']['事由']]
            gs.append_row(logs)
```

```python
@handler.add(MessageEvent, message=LocationMessage)
def handle_location_message(event):
    global users

    _id = event.source.user_id
    me = users[_id]
    addr=event.message.address    # 地址
    lat=str(event.message.latitude)    # 緯度
    lon=str(event.message.longitude)   # 經度

    if addr is None:
        msg=f'收到GPS座標：({lat}, {lon})\n謝謝您！'
    else:
        msg=f'收到GPS座標：({lat}, {lon})。\n地址：{addr}\n謝謝您！'

    if me['save']:
        me['logs']['經緯度'] = f'({lat}, {lon})'
        me['logs']['地址'] = addr

        line_bot_api.reply_message(
            event.reply_token, [
                TextSendMessage(text=msg),
                TextSendMessage(text='請問您需要些什麼物資呢？')
        ])
    else:
        line_bot_api.reply_message(
            event.reply_token,
            TextSendMessage(text=msg))
```

Pack the returned information into a list format and store it in Google sheet.

```python
class GoogleSheet():
    '''
    建構式參數:
    wks_name: 試算表名稱
    ws_title: 工作表名稱, 預設None。
    oauth_file: 憑證檔名
    '''

    def __init__(self, wks_name, wks_title=None, oauth='oauth.json'):  # 請自行修改檔名
        scope = ['https://spreadsheets.google.com/feeds',
                 'https://www.googleapis.com/auth/drive']

        try:  # 嘗試讀取憑證檔
            JSON_PATH = os.path.join(os.getcwd(), 'model', oauth)
            cr = sac.from_json_keyfile_name(JSON_PATH, scope)
        except Exception as e:
            print('無法開啟憑證檔', e)
            sys.exit(1)  # 關閉程式

        try:  # 嘗試開啟Google試算表
            gc = gspread.authorize(cr)
            sh = gc.open(wks_name)
        except Exception as e:
            print('無法開啟Google試算表', e)
            sys.exit(1)

        if wks_title is None:
            # 開啟「工作表1」
            self._wks = sh.sheet1
        else:
```

```
            try:  # 嘗試開啟指定工作表
                self._wks = sh.worksheet(wks_title)
            except Exception as e:
                print('無法開啟工作表', e)
                sys.exit(1)

        # 讀取標題列
        # self.headers = self._wks.row_values(1)

    def append_row(self, data):
        self._wks.append_row(data)  # 插入新列

    def resize(self, n=1):
        self._wks.resize(n)

    def update_header(self, data, delete=True):
        if delete:
            self._wks.delete_row(1)

        self._wks.insert_row(data, 1)

    @property
    def headers(self):
        return self._wks.row_values(1)
```