專題題目: Monify
組名: 詹松霖

隊長: 詹松霖 110703065 資訊三
組員: 陳歸中 110703004 資訊三
林丰筑 110703023 資訊三
朱虹華 110703026 資訊三
簡敦佑 110703009 資訊三
吳俊暉 110701028 應數三

分工表:

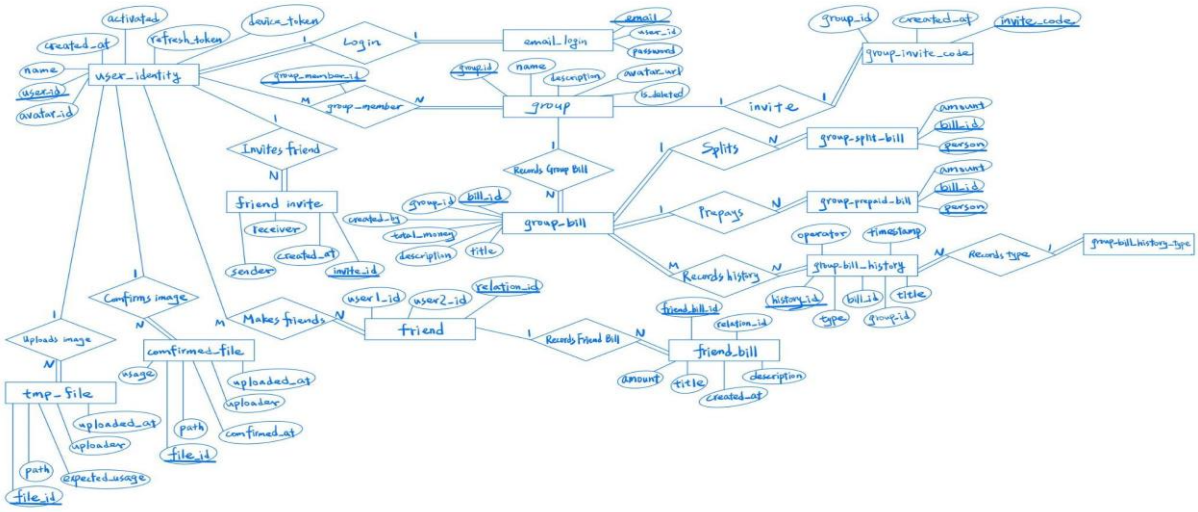
姓名	分工內容	貢獻百分比
陳歸中	前端，負責功能: bill modification, summerize；整合頁面	20%
林丰筑	前端，負責功能: authorization、username、group、multiselect	20%
朱虹華	前端，負責功能: bill insertion, history	15%
簡敦佑	寫 api 與 api 測試，ER model	17%
詹松霖	devops，專案建構，寫 api 與 api 測試	21%
吳俊輝	QA, integration test	7%

需求分析:
一群人出去玩時，常常有幫大家墊錢的時候，但往往在玩完之後要分錢時會一團混亂，甚至是總金額錯誤的問題。因此我們想製作一個 app 可以在墊錢的時候可以做紀錄，並在最後給予用戶一個建議的收支關係。

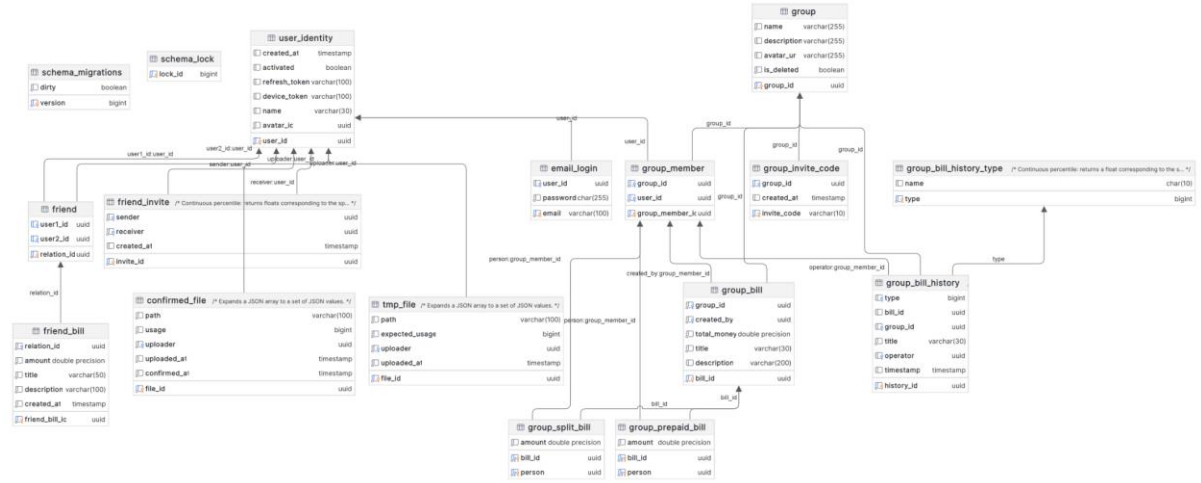
系統功能:
1. 登入/ 註冊
2. 創建群組/ 加入群組
3. 記帳/ 查帳/ 刪除
   項目：品項、金額、支付者、分帳者、日期
4. 紀錄群組總帳務/ 個人帳務
   個人帳務會顯示最少次數的轉帳結果

系統功能:

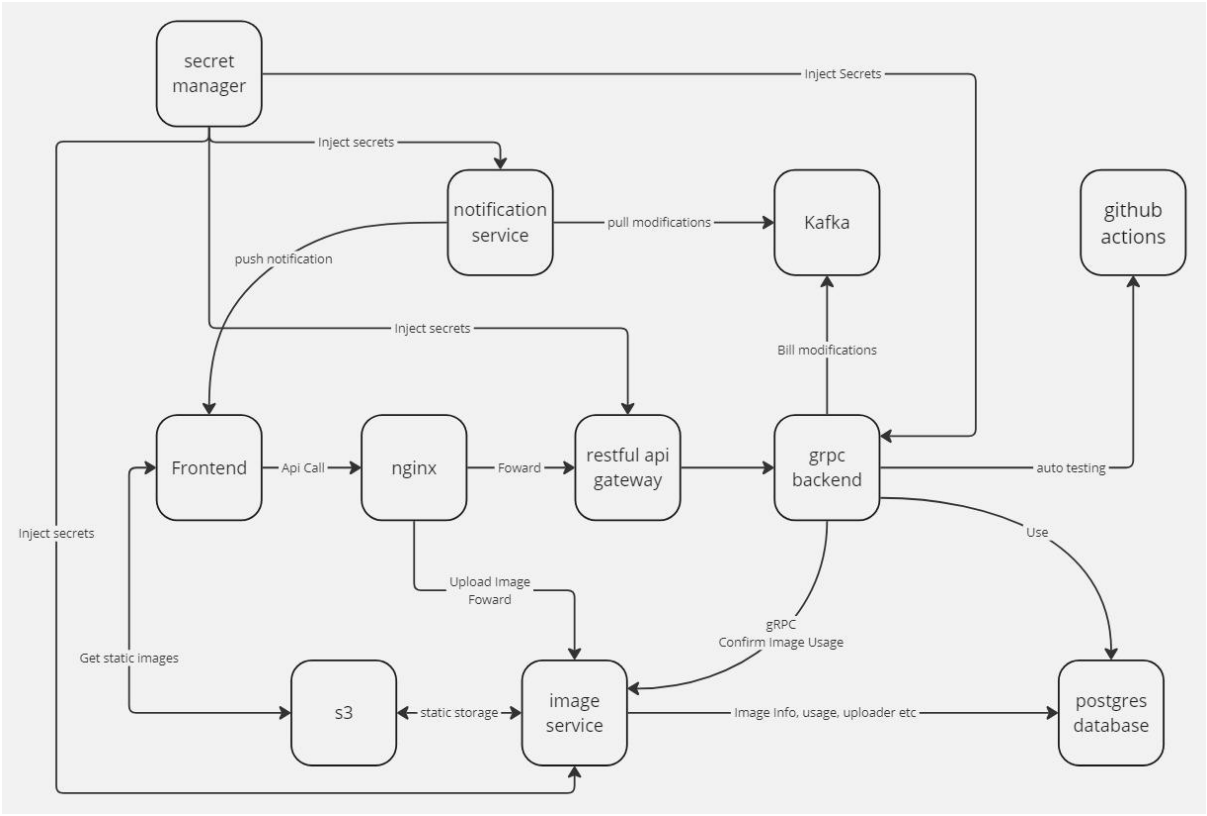
ER Model:



Relational Schema:



System Diagram:



Language：後端的部分選擇 Golang，原因是簡單好用，效能又好。前端選擇使用 React Naitive。

Restful & gRPC backend： 我們的 api 是使用 protobuf 定義，並生成 restful 與 grpc 的 api gateway，一般來說 client 可直接使用 grpc，但我們的 frontend 似乎不支援 http 2.0，所以得先轉成 restful 的格式。

Database： 使用 postgres，因為最熟悉

Image service：負責處裡使用者上傳的圖片，由於 grpc 不適合上傳圖片，所以我們另外架了一個專門處裡 multipart form file 的 service，會將使用者上傳的檔案上傳至 s3。

Kafka：我們預計當使用者在群組進行操作時，會需要通知使用者，通知的方式我們打算做兩種，一種是結合 line 的 mini app，另一種是 firebase 直接廣播到手機上。由於之後打算做各種 integration，所以我們採用 message queue 的方式與 event driven 的設計，使整個系統的耦合度更低。

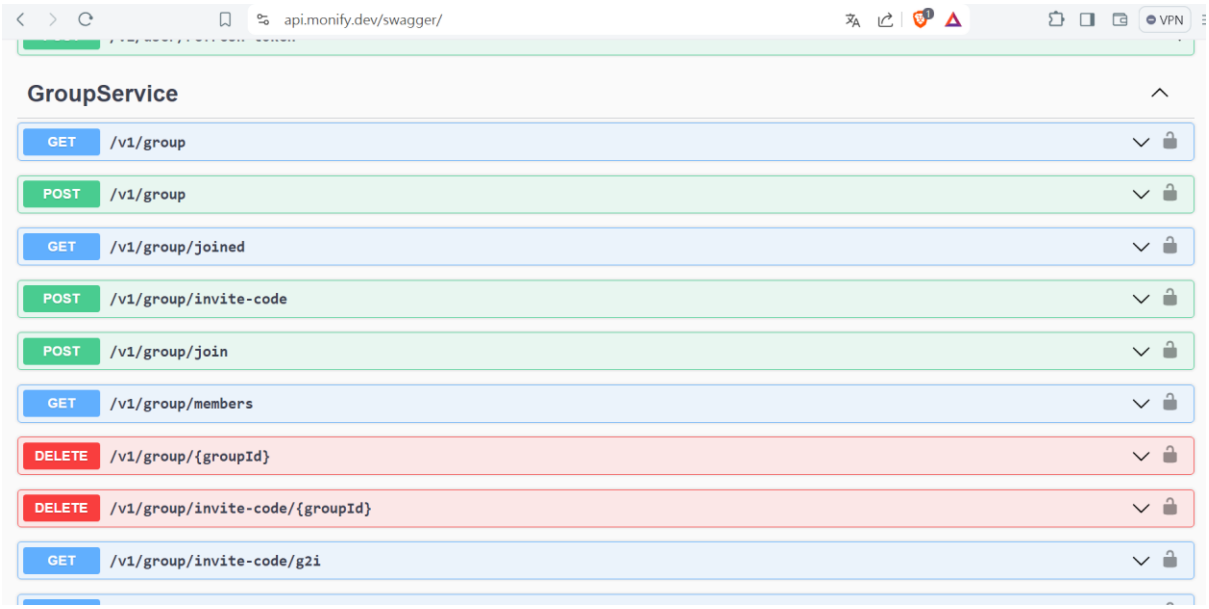
SecretManager：開發的時候管理.env 檔案實在非常麻煩，所以我們使用 secret manager 來管理所有的 api key，database url 等等。 在 service 啟動時，會跟 secret manager 拿取這些資訊。

GithubActions：我們引入了自動化測試，來確保每次的修正不會改壞以前的東西，並測試更新的新功能。

Nginx：負責將 request 導向不同 service，包含 swagger。

Swagger： protobuf 還可以自動生成 swagger 檔案，十分方便與前端對接。

Notification：由於需要 ios 與 android 的 package id 等等，所以尚未實作，預計結合 line mini app 與 firebase messenging。

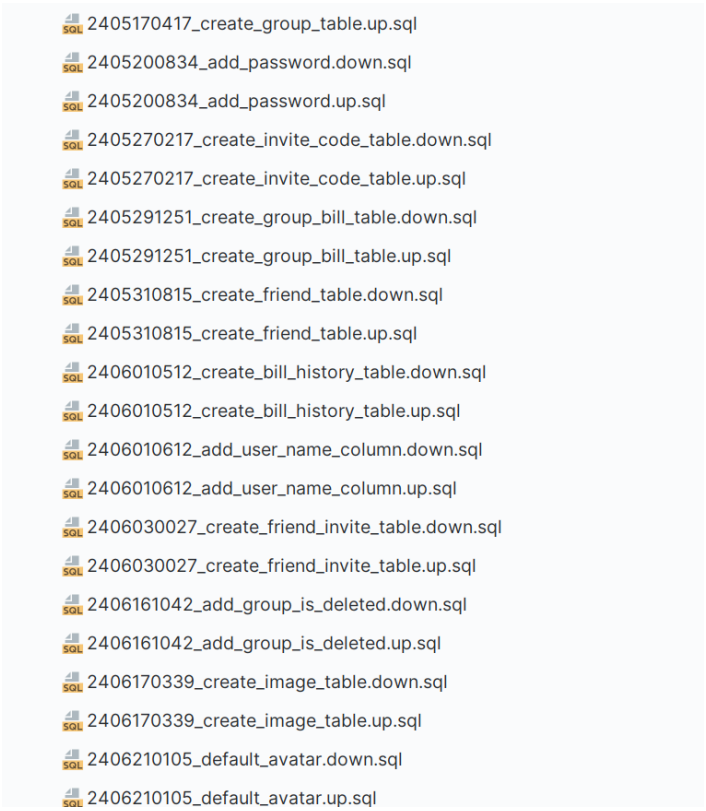


Swagger 連結： https://api.monify.dev/swagger

```go
func TestGroupInviteCode(t *testing.T) {
    client := GetTestClient(t)
    _ = client.CreateTestUser()
    group, err := client.CreateGroup(context.Background(), &monify.CreateGroupRequest{Name: "test", Description: "test123"})
    assert.NoError(t, err)
    assert.NotEmpty(t, group)
    code, err := client.GenerateInviteCode(context.Background(), &monify.GenerateInviteCodeRequest{GroupId: group.GroupId})
    assert.NoError(t, err)
    assert.NotEmpty(t, code)
    inviteCode, err := client.GetInviteCode(context.Background(), &monify.GetInviteCodeRequest{GroupId: group.GroupId})
    assert.NoError(t, err)
    assert.Equal(t, code.InviteCode, inviteCode.InviteCode)
    code2, err := client.GenerateInviteCode(context.Background(), &monify.GenerateInviteCodeRequest{GroupId: group.GroupId})
    assert.NoError(t, err)
    assert.Equal(t, code.InviteCode, code2.InviteCode)

    groupBrief, err := client.GetGroupByInviteCode(context.Background(), &monify.GetGroupByInviteCodeRequest{InviteCode: cod
    assert.NoError(t, err)
    assert.Equal(t, group.GroupId, groupBrief.GroupId)
    assert.Equal(t,  expected: "test", groupBrief.Name)
    assert.Equal(t,  expected: "test123", groupBrief.Description)
```

我們的測試框架使得撰寫測試十分容易，呼叫 GetClient 後就能將測試伺服器架起來。還能夠過 CreateTestUser 與 SetTestUser，輕鬆的創建跟切換使用者。
測試的時候會架一個 sqlite 的檔案起來，方便測試。



我們做了 database 的版本控制，輕鬆做 migration 不再頭痛

```
SUB_DIRS = protobuf
PACKAGES    ?= $(shell go list ./...)

all: $(SUB_DIRS)

$(SUB_DIRS):
    make -C $@

test:
    -mkdir build
    go test $(PACKAGES) -v -cover -failfast

test_docker:
    -docker stop monify-test-postgres
    -docker rm monify-test-postgres
    docker run --name monify-test-postgres -p 5432:5432 -e POSTGRES_PASSWORD=password -d postgres
    go test $(PACKAGES) -v -cover -failfast -tags docker

clean:
    -rm -rf build


docker_push_proxy: docker_build_proxy
    docker push registry.nccupass.com/monify_restful_proxy
```

我們使用 Makefile 來做 build system 的控制，包含測試，build docker image，protobuf generation 等等功能。

Deployment：

我們目前是架在 linode 的主機上，deployment 可自行選擇方式，目前是使用 docker-compose.yml，之後有打算改架 k8s。

範例 docker-compose.yml：

```
version: "3.0"
networks:
  grpc-network:
    driver: bridge

services:
  monify:
    image: registry.nccupass.com/monify
    restart: always
    environment:
      CLIENT_ID: XXXX
      CLIENT_SECRET: XXXX
```

```
      ENVIRONMENT: dev
    networks:
      - grpc-network
  swagger:
    image: swaggerapi/swagger-ui:v4.0.0
    environment:
      SWAGGER_JSON_URL: https://raw.githubusercontent.com/Monify-
Dev/monify/main/protobuf/gen/monify.swagger.json
    networks:
      - grpc-network
  restful_proxy:
    image: registry.nccupass.com/monify_restful_proxy
    restart: always
    command: /app -grpc-server-endpoint=monify:2302
    networks:
      - grpc-network
  media_service:
    image: registry.nccupass.com/media_service
    restart: always
    networks:
      - grpc-network
    environment:
      CLIENT_ID: XXXX
      CLIENT_SECRET: XXXXX
      ENVIRONMENT: dev
  nginx:
    build: ./nginx
    ports:
      - "443:443"
    restart: always
    networks:
      - grpc-network
```

nginx.conf

```
http{
        server{
                listen 443 ssl;
```

```
                server_name api.monify.dev;

                ssl_certificate
/etc/letsencrypt/live/api.monify.dev/fullchain.pem;
                ssl_certificate_key
/etc/letsencrypt/live/api.monify.dev/privkey.pem;
                include /etc/letsencrypt/options-ssl-nginx.conf;
                ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

                location /swagger/ {
                        proxy_pass http://swagger:8080/;
                }

                location /media/ {
                        proxy_pass http://media_service:8080/;
                }
                location / {
                        proxy_pass http://restful_proxy:8081;
                }
        }
}
```

Others

一些其他有趣的設計:

1. 生成群組邀請碼：
   邀請碼總共 6 碼，在創建的 10 分鐘後會過期，每一碼會從 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ 當中選一個 char。選法是前 4 位由當前當前時間/10 分鐘的餘數決定，後兩位為隨機數。總共有 36^6 的組合，以此來降低碰撞機率。生成的時候若是真的不行發生碰撞，會透過 transaction 來確保不會發生覆蓋掉已經存在且尚未過期的邀請碼，同時會 retry 3 次。
2. 身份驗證與安全：
   註冊的時候我們會將密碼進行雜湊，以確保即便資料庫外洩，使用者的密碼也不會外流。另外我們使用 jwt 來管理身份，當前端使用者登入時，會拿到一組 access token 與 refresh token。access token 可以代表本人跟後端呼叫 api，但會在一段時間後過期，這時候就得呼叫 refresh token 的 api 並傳入 refresh token，來獲得新的 access token。

連結:
前端 github ： https://github.com/Jasmine0108/moneyApp_frontend
後端 github ： https://github.com/Monify-Dev/monify
Swagger: https://api.monify.dev/swagger

心得:

陳歸中:這次的前端由於是資料庫系統的應用，不少 component 都需要接來自後端的資料，因此寫了很多邏輯與函式處裡，不同於我熟悉的版面設計，相當有挑戰，也很有趣，最後做出來很有成就感。

林丰筑:以前有寫過一點前端，但透過這次的專案，接觸到很多之前沒接觸過的前端相關內容，也是第一次接 api，覺得這次的專案蠻有趣也很有成就感。

朱虹華：第一次使用 React Native 寫前端架構，學到非常多實用的內容

簡敦佑： 在這次的專案當中第一次接觸做 app，覺得做後端的 api 還蠻有趣的，寫完蠻有成就感的，希望之後能學會怎麼做整個 app。

詹松霖： 藉這次資料庫的機會把之前想做的 app 做了一部分，並試了一些之前想嘗試的技術跟開發流程，整體還蠻順利的。

吳俊暉：後端以及使用沒有用過的語言相當有挑戰性，要讓測試跑起來都成困難，在這次的專案中貢獻很少，感謝組員不厭其煩的教導。