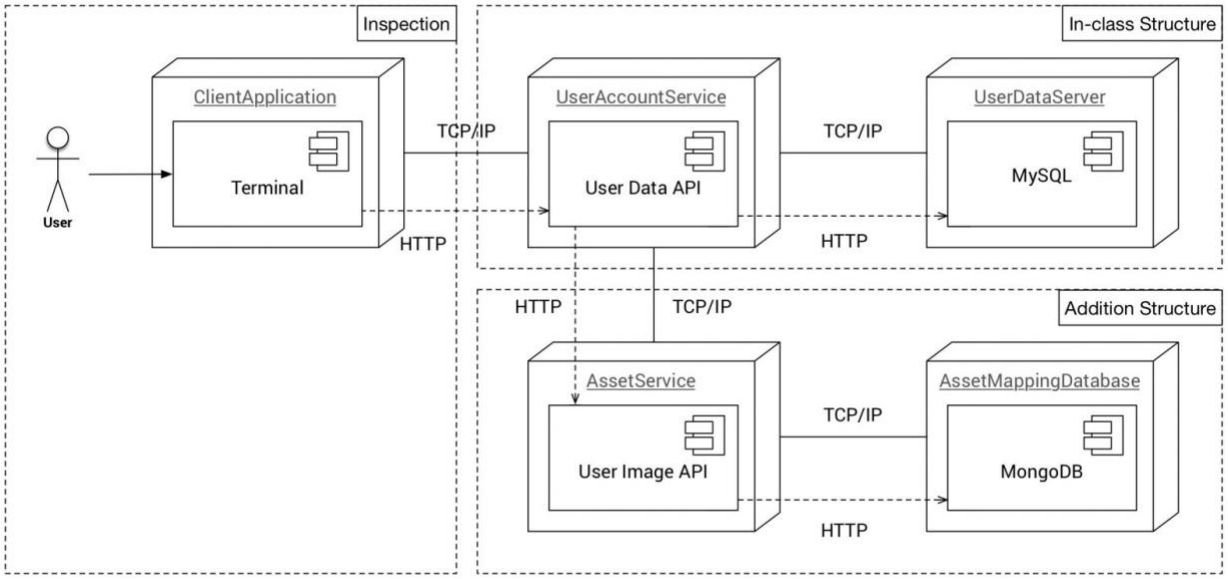
ให้ใช้โคดเริ่มต้นโดย download จาก

https://drive.google.com/drive/folders/1za5CMzed8WdS5NmnBd1Aeikj6aQ2bYHY?usp=share_link



Containers ด้านบน เป็นโครงสร้างสำหรับโปรแกรมค้นหาข้อมูลผู้ใช้งาน โดยมีคำอธิบายการใช้งานระบบดังนี้

1. ผู้ใช้เริ่มต้นใช้งานจาก *ClientApplication* ภายในส่วน *Inspection* ด้วยการส่งชื่อ *username* ที่ต้องการค้นหาให้กับ *UserAccountService* ผ่านการร้องขอข้อมูลแบบ GET ของโปรโตคอล HTTP ซึ่งอยู่ใน *In-class Structure*
2. *In-class Structure* ประกอบด้วย 2 ส่วน ประกอบด้วย *UserAcountService* จะรับข้อมูล *username* เพื่อค้นหาหมายเลขโทรศัพท์ของ *username* ดังที่ระบุ จากฐานข้อมูล MySQL ภายใน *UserDataServer* โดย *UserDataServer* นั้นได้เตรียมข้อมูลตั้งต้นไว้ภายในโฟลเดอร์ user_data แล้วเรียบร้อย
3. นอกจากนี้ *UserAcountService* จะทำการส่ง *username* ไปยัง *AssetService* เพื่อนำไปค้นหาข้อมูลรูปภาพที่ตรงกันกับ *username* ที่ต้องการ โดยข้อมูลที่อยู่ใน profile_asset.json ที่จะเก็บใน MongoDB แทน MySQL ข้างต้น

สิ่งที่นิสิตต้องทำ

1. เติมคำสั่งต่างๆ ใน Dockerfile ของ
    1.1) user_account_service ใช้ base image ของ node เป็น 17-alpine
    1.2) user_data โดย based image ของ mysql เป็น 8.0
    1.3) asset_service ใช้ base image ของ node เป็น 17-alpine
    1.4) asset_mapping_seed ใช้ base image mongo เป็น 5.0

โดย Dockerfile ของ asset_mapping ได้ทำไว้ให้แล้ว

2. Start ระบบทั้งหมดโดยใช้คำสั่ง docker-compose
3. เข้าไปที่ docker container ของ user_data โดยใช้คำสั่ง

   docker exec -it [container name] /bin/bash

   จะเข้าไปใน container ของ MySQL DB จากนั้น login เข้า MySQL โดยใช้คำสั่ง

   mysql -u root -p

   ให้ใส่ password เป็น 123 [หมายเหตุ ดู user name และ password ของ DB จาก user_data/Dockerile

   (** แต่เน้นเป็น bad practice เราไม่ควรใส่ MySQL environment variables ใน Dockerfile **)

   เมื่อเข้าไปใน MySQL DB แล้ว ลองดูว่ามีฐานข้อมูลอะไรบ้างให้ลองใช้คำสั่ง

   mysql> show databases;

   จะเป็นฐานข้อมูลต่อไปนี้

```
+--------------------+
| Database           |
+--------------------+
| ABCompany          |
| information_schema |
| performance_schema |
+--------------------+
```

   เราสามารถเข้าไปดูว่าในฐานข้อมูล ABCompany มีตารางอะไรบ้างโดยใช้คำสั่ง

   mysql> use ABCompany;

   เพื่อ switch ไปยังฐานข้อมูล ABCompany

   จากนั้นสามารถดูว่ามีตารางอะไรบ้างโดยใช้คำสั่ง

   mysql> show tables;

   [submission #1] screen capture ว่ามีตารางอะไรบ้างและตารางนั้นมีข้อมูลอะไรบ้าง

   และทำการ แก้ไข setting ของ mysql โดยใช้คำสั่งต่อไปนี้

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH
mysql_native_password BY '123';

ALTER USER 'users_service' IDENTIFIED WITH mysql_native_password
BY '123';

flush privileges;
```

   [submission #2] ตอบคำถามว่าทำไมถึงต้องทำ 3 คำสั่งข้างต้นใน MySQL

4. ทดสอบว่า containers ของระบบทำงานได้ถูกต้อง โดยใช้คำสั่งต่อไปนี้ที่ terminal ของ host (เครื่องเรา) **

   หมายเหตุ ถ้าเรียกครั้งแรกแล้วค่า profile_image ยังเป็น “” ให้ลองเรียกอีกครั้ง

   curl -L "http://localhost/?username=alice" | json_pp

ผลลัพธ์ของคำสั่ง curl ต้องได้ตามนี้

```
{
   "phoneNumber" : "1234567890", <=มาจาก MySQL
   "profile_image" : "https://picsum.photos/1000", <=มาจาก MongoDB
   "username" : "alice" <=มาจาก MySQL
}
```

ถ้าข้อมูลส่งออกได้ผลตามข้างต้น

[submission #3] ให้นิสิต zip folder และส่งกลับมาใน MyCourseville โดยมี Dockerfile ของ services ต่างๆ และ docker-compose.yml ที่แก้ไขให้สามารถนำมารัน docker-compose แล้วได้ผลตามข้างต้น

ใน docker-compose.yml ให้ legacy links: ให้นิสิตลองเปลี่ยนเป็นใช้ networks แทนแบบในตัวอย่าง docker-compose.yml ที่เรียนวันอังคาร ลอง Start ระบบทั้งหมดโดยใช้คำสั่ง docker-compose อีกครั้งว่าทำงานได้ไหม

[submission #4] รายงานผลการเปลี่ยนจาก links เป็น networks ว่าได้ผลแบบข้างต้นหรือไม่ อย่างไร

Ref: https://docs.docker.com/engine/network/links/

Appendix

Asset Mapping Database (ในส่วนของ mongo)

เพื่ออำนวยความสะดวก ได้จัดเตรียม *AssetMappingDatabase* พร้อมทั้งเพิ่มข้อมูลด้านในไว้ให้แล้วเรียบร้อย โดยหลังจากเรียกคำสั่ง docker-compose build หรือ docker-compose up -d แล้ว ถ้าเรียกคำสั่ง

```
$ docker exec -it [container name of asset_mapping] mongo
```

และคำสั่งของ mongo ตามตัวอย่างต่อไปนี้ จะเห็นรายการ userProfile ทั้งหมดในระบบ

```
> use user

> db.userProfile.find()

{ "_id" : ObjectId("59f03cfa6574d1b68c74caf6"), "user_id" : "5C66F17A-534E-48C2-8352-2631EBC61094"
, "uname" : "trudy", "profile_image" : "https://picsum.photos/900?random", "last_update" : 1508821332 }

{ "_id" : ObjectId("59f03cfa6574d1b68c74caf7"), "user_id" : "610982C8-226A-483B-B597-22B0768C82AF",
"uname" : "carol", "profile_image" : "https://picsum.photos/1000", "last_update" : 1508821332 }
```

{ "_id" : ObjectId("59f03cfa6574d1b68c74caf8"), "user_id" : "D4660FDE-180D-4535-8D97-B94BA226B72C", "uname" : "eve", "profile_image" : "https://picsum.photos/1000", "last_update" : 1508821332 }

{ "_id" : ObjectId("59f03cfa6574d1b68c74caf9"), "user_id" : "C916EA7C-C9C7-4564-AA38-46F0B1CFB915", "uname" : "bob", "profile_image" : "https://picsum.photos/1000", "last_update" : 1508821332 }

{ "_id" : ObjectId("59f03cfa6574d1b68c74cafa"), "user_id" : "0888D506-108D-44BF-B75B-F8406CD9C4D4", "uname" : "alice", "profile_image" : "https://picsum.photos/1000", "last_update" : 1508821332 }

{ "_id" : ObjectId("59f03cfa6574d1b68c74cafb"), "user_id" : "BCE16C91-69BD-4FB4-9066-CFE01CEBC8B2", "uname" : "frank", "profile_image" : "https://picsum.photos/g/1000?random", "last_update" : 1508821332 }

{ "_id" : ObjectId("59f03cfa6574d1b68c74cafe"), "user_id" : "A47CC1B8-0B92-48D0-B681-5BAA346008C2", "uname" : "victor", "profile_image" : "https://picsum.photos/g/1200?randome", "last_update" : 1508821332 }

…