

# Full Stack Development Base on Reference Data Model

นำเสนอด้วย นายสิทธิพงศ์ จักรสุทธิรงค์  
อาจารย์ที่ปรึกษา รองศาสตราจารย์ ดร.วรา วรรavitay

# เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

01

ศึกษา Full Stack  
Cloud Native  
(Docker,  
Postgres, PHP,  
React)

02

เรียนรู้ Data Model  
จาก Data Model  
Resource Book

03

ออกแบบ Database  
ด้วย Visual Paradigm

04

สร้าง CRUD App  
จัดการ Person ด้วย  
Database จาก  
หนังสือ

# เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

05

สร้าง CRUD App จัดการ Person ด้วย Database จาก Visual Paradigm

06

สร้าง CRUD App จัดการ Person ด้วย MVC (React เป็น View)

07

สร้าง CRUD App จัดการ Geo Data (Laravel, Artisan)

08

ใช้ Pandas ETL(extract transform load) นำ Dataset Geo จากการปักครองเข้าสู่ app

# เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

09

พัฒนา Dynamic  
Dropdown (Country,  
Province, District)

10

ปรับปรุง Laravel App  
(Response Time: 7s  
→ 1s)

11

สร้าง CRUD App  
จัดการ Person-  
Organization ด้วย  
Slim PHP

12

พัฒนา CRUD App  
เพิ่ม Role,  
Relationship,  
Communication  
Event

# เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

13

สร้าง CRUD App  
เดิม เพิ่ม JWT  
Authentication  
ด้วย FastAPI

14

ปรับ UX/UI ตามคำ  
แนะนำจาก  
ผู้ช่วยศาสตราจารย์  
ภาณุศรี ณ รังษี  
ครุสาขาออกแบบ

15

ปรับปรุง UI ตามข้อเสนอ  
แนะนำผู้ใช้ Gen Z

16

สร้าง Role-Based CRUD  
App (6 บทบาท)

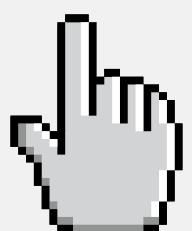
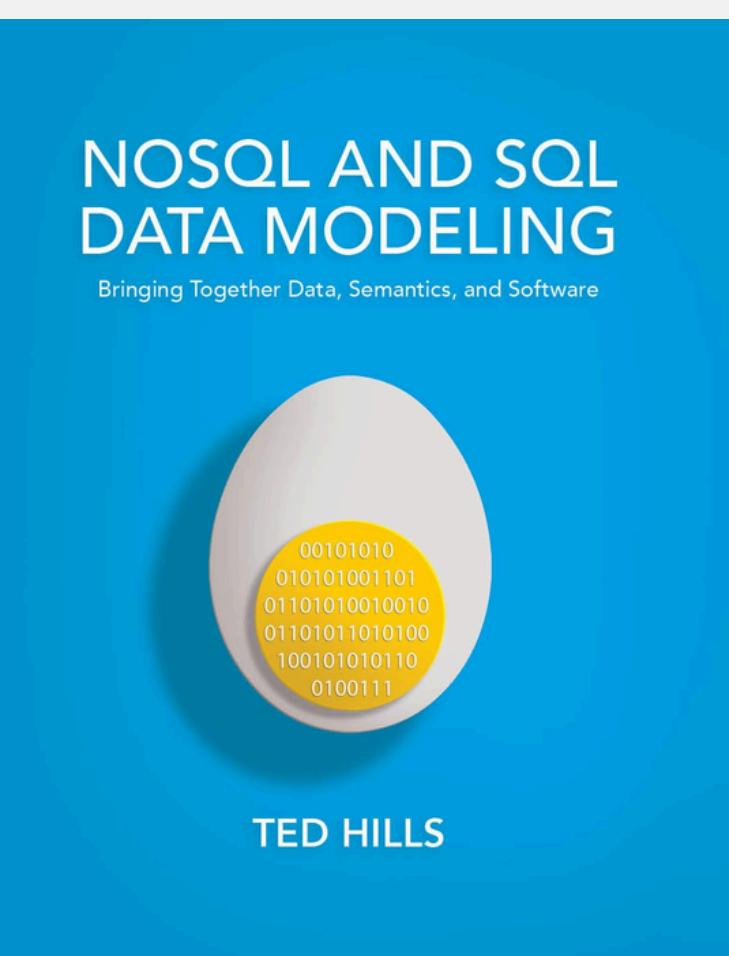
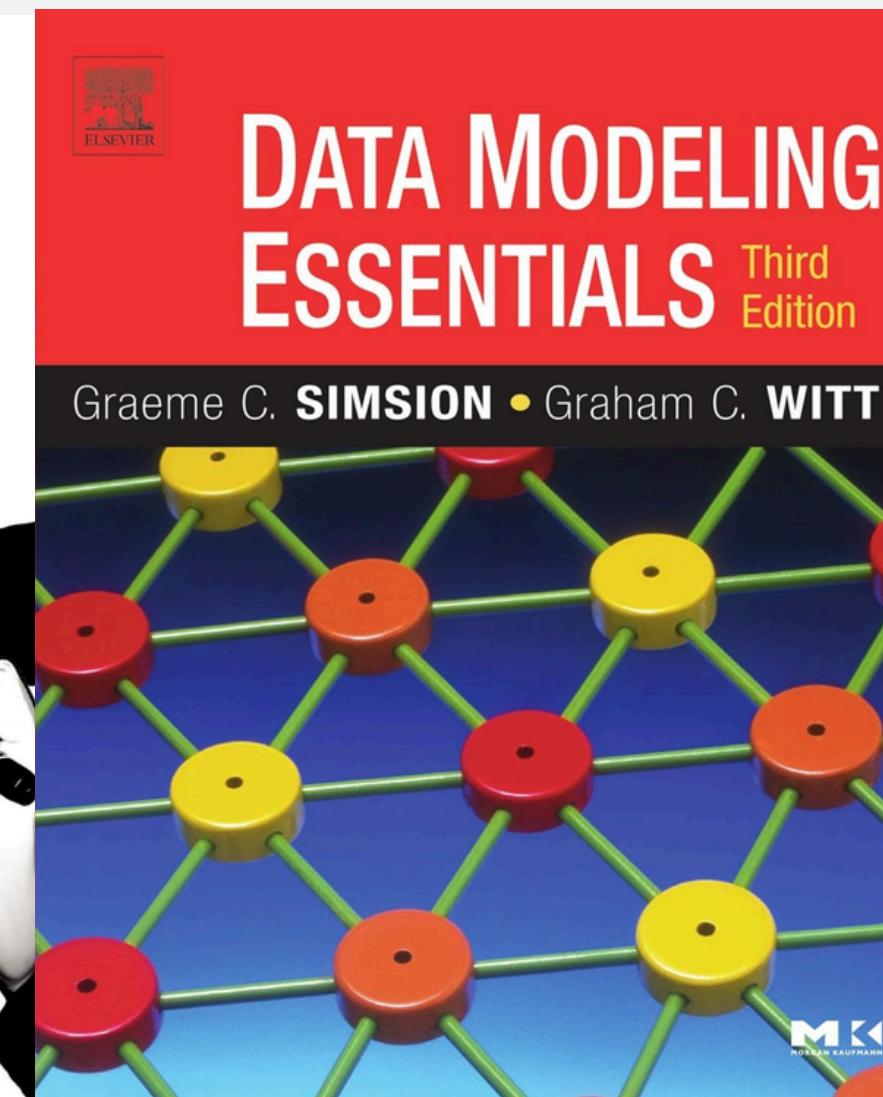
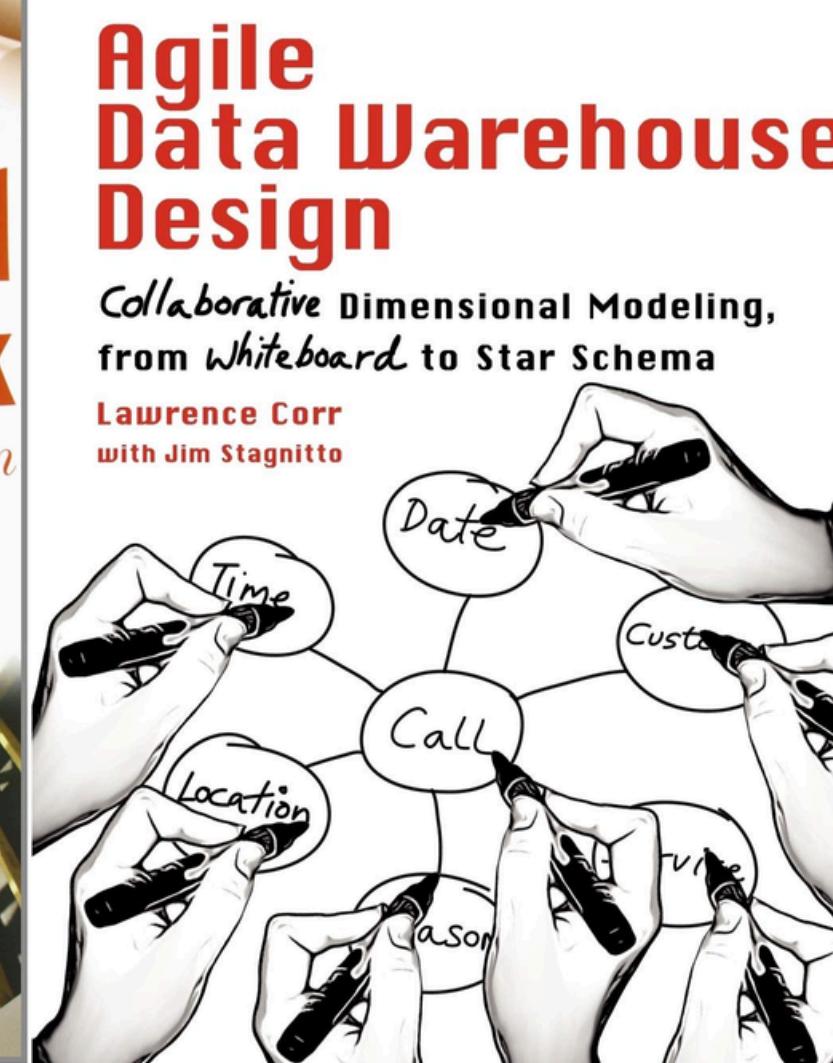
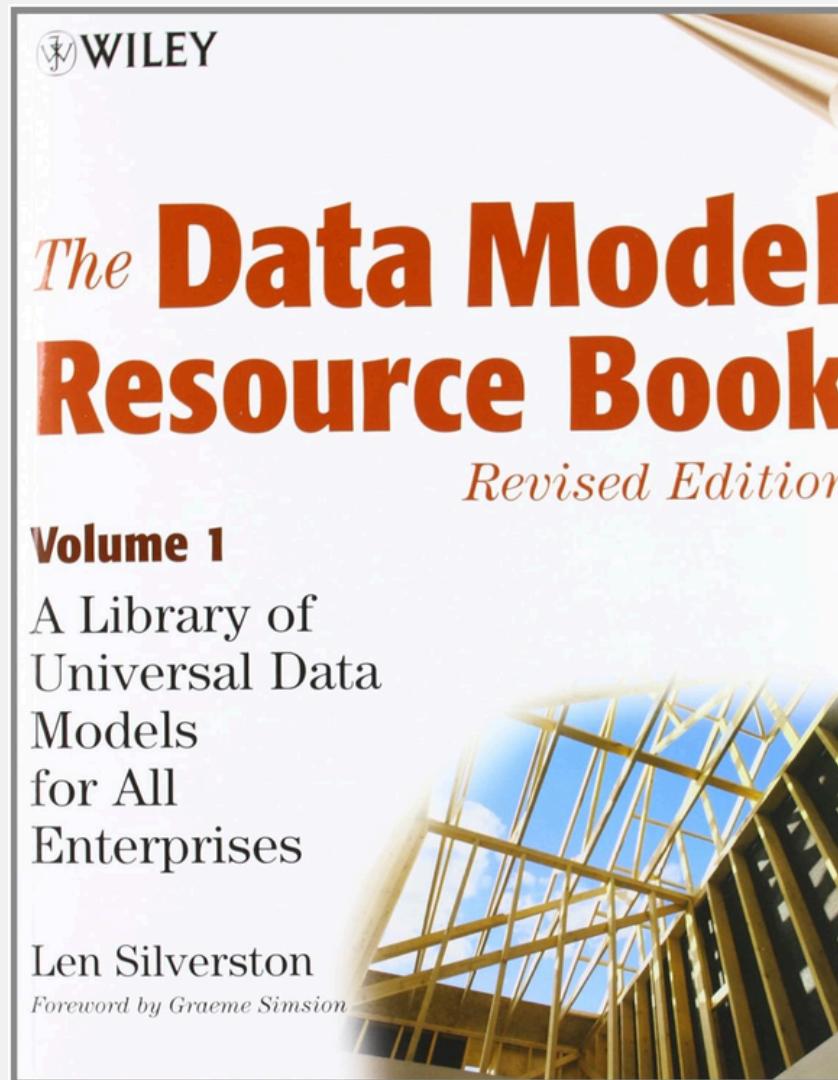
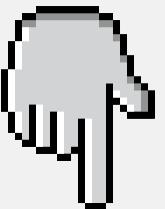
# เส้นทางการเรียนรู้ ก่อนเริ่มงานจริง

---

17

ออกแบบ Diagram ประกอบซอฟต์แวร์ (Activity, ERD, Sequence,  
State, Use Case)

# Book About Data Model



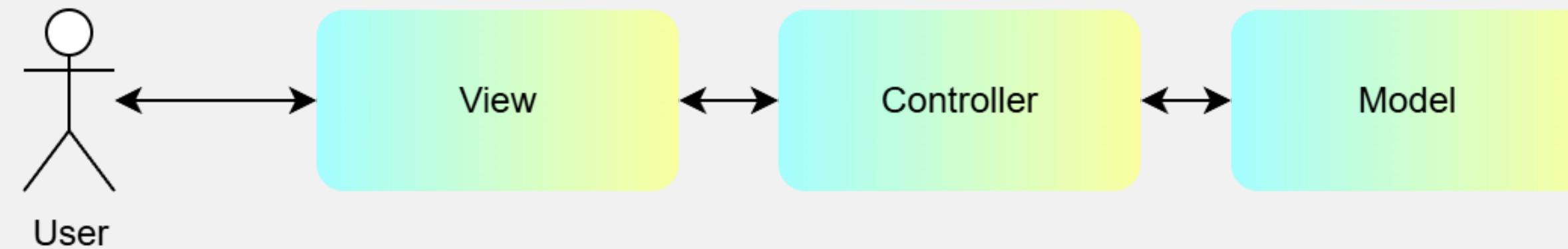
# Data Model

- Data คือข้อมูล
- Model คือรูปปั้น แบบจำลอง
- Data Model คือการสร้างแบบจำลองของข้อมูลในระบบ
- เมื่อ Developer ในทีมเห็นโครงสร้างชัดเจนก็จะเข้าใจข้อมูลในระบบมากขึ้น

# Evolution of Web

- Web 1.0 เว็บ static เน้นอ่านข้อมูลอย่างเดียว เหมือนหนังสือออนไลน์
- Web 2.0 เว็บ dynamic ผู้ใช้โต้ตอบและสร้างเนื้อหา เช่น social media
- ใช้ AJAX ทำให้หน้าเว็บอัปเดตแบบเรียลไทม์ เช่น Facebook, YouTube
- เริ่มจาก SSR สร้างหน้าเว็บที่เซิร์ฟเวอร์ ส่ง HTML ทั้งหน้า
- เปลี่ยนเป็น CSR ประมวลผลฝั่งผู้ใช้ ส่งเฉพาะ JSON

# MVC



- แบ่งงานเป็น Model, View, Controller
- Model จัดการข้อมูล
- View แสดงผลหน้าตาให้ผู้ใช้
- Controller ควบคุมการโต้ตอบ

# Library

- Library เป็นชุดโค้ดสำเร็จรูป เรียกใช้เฉพาะส่วนที่ต้องการ
- ไม่บังคับโครงสร้าง ผู้พัฒนาควบคุม flow การทำงานเอง
- ช่วยลดเวลาพัฒนา เช่น พังก์ชันจัดการ DOM หรือคำนวณ
- ต่างจาก Framework ที่กำหนดโครงสร้างและ flow การทำงาน
- Ex. jQuery สำหรับจัดการ DOM หรือ Lodash สำหรับพังก์ชัน utility

# Framework

- Framework เป็นโครงสร้างกำหนดวิธีพัฒนาแอป มีกฎและ flow การทำงาน
- ควบคุม flow เช่น MVC ใน Django จัดการ logic UI และ data
- Library เป็นชุดโค้ดเรียกใช้ได้ตามต้องการ ไม่กำหนดโครงสร้าง
- Framework บังคับโครงสร้าง (inversion of control) แต่ Library ให้อิสระ
- ตัวอย่าง Framework เช่น Django React ส่วน Library เช่น jQuery Lodash

# Frontend และ Backend

- Frontend เป็นส่วนที่ผู้ใช้เห็นและโต้ตอบ เช่น UI ปุ่ม กล่องข้อความ Animation
- Backend เป็นส่วนเชื่อม Frontend กับ Database จัดการ logic และข้อมูล
- Frontend เปรียบเหมือนหน้าร้านอาหารที่ลูกค้านั่งกินข้าว
- Backend เปรียบเหมือนเด็กเสิร์ฟที่ประสานงานระหว่างลูกค้าและครัว
- ร่วมกันสร้างประสบการณ์แบบที่สมบูรณ์และตอบสนองผู้ใช้
- ในเรื่อง MVC Backend คือ Controller, Frontend คือ view

# Application-Specific Framework

- เฟรมเวิร์กสำหรับงานเฉพาะ
- เช่น CMS (WordPress) สำหรับเว็บเนื้อหา
- Google Sites สำหรับสร้างเว็บง่ายๆ
- Shopify สำหรับร้านออนไลน์
- ปรับให้เหมาะสมกับเป้าหมายแอป

# Frontend Framework

- เพرمิเตอร์กสำหรับสร้าง UI และการโต้ตอบ
- จัดการหน้าเว็บแบบ dynamic
- เช่น React, Vue.js, Angular
- ทำให้หน้าเว็บตอบสนองและใช้งานง่าย

# Backend Framework

- เพرمิเติร์กสำหรับจัดการ logic ฝั่งเซิร์ฟเวอร์
- ช่วยพัฒนาการทำงานของฐานข้อมูลและ API
- เช่น Node.js, Django, Ruby on Rails
- ทำให้ระบบหลังบ้านเร็วและปลอดภัย

# Framework Stack (Tech Stack)

- ชุดเทคโนโลยีที่ใช้พัฒนาแอปหรือเว็บ
- รวม frontend, backend และฐานข้อมูล
- เช่น React + Node.js + MongoDB
- เลือก stack ให้เหมาะสมกับเป้าหมายโครงการ

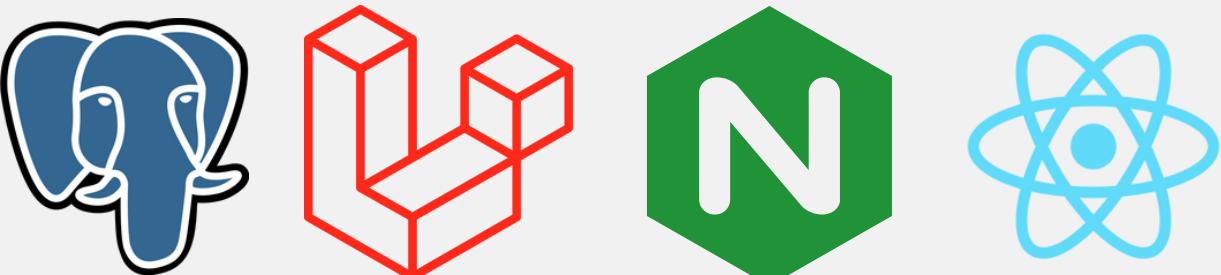
# Tech Stack Timeline



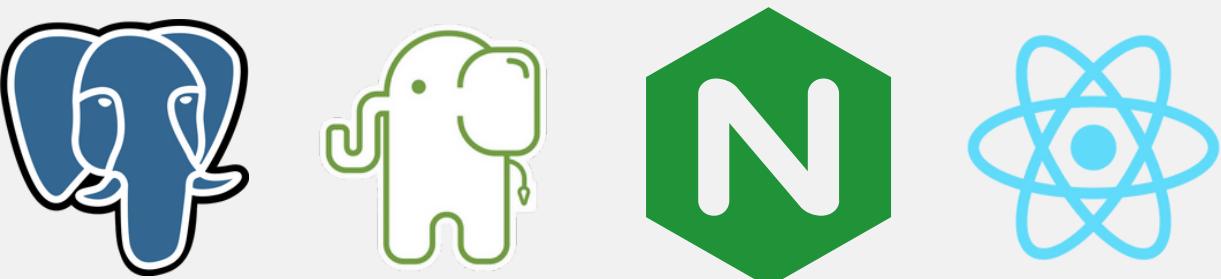
Postgres + Native PHP + React JS + Apache



Postgres + Laravel PHP + React TS + Nginx

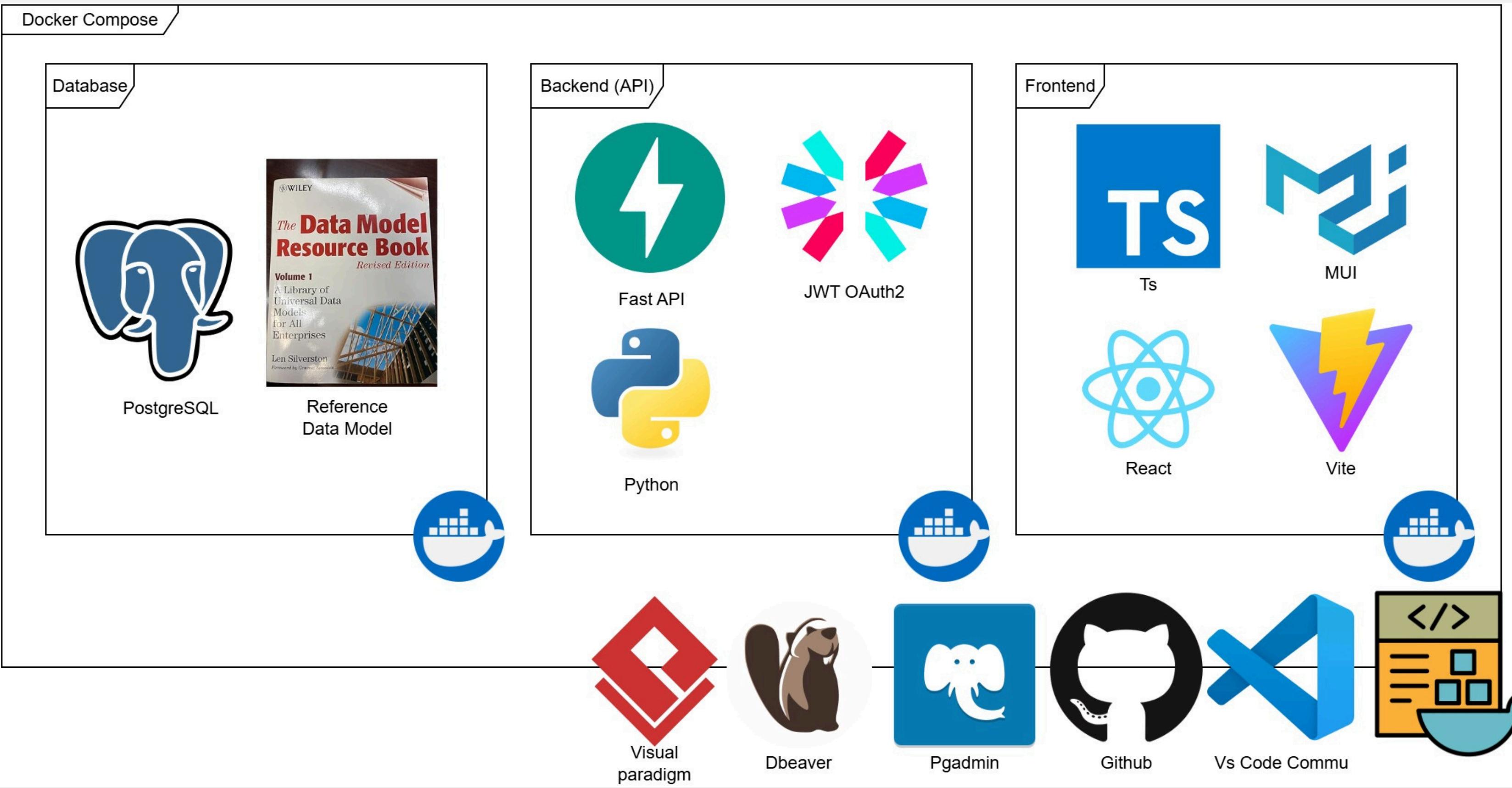


Postgres + Slim PHP + React TS + Nginx



Postgres + FastAPI Python + React TS + Uvicorn



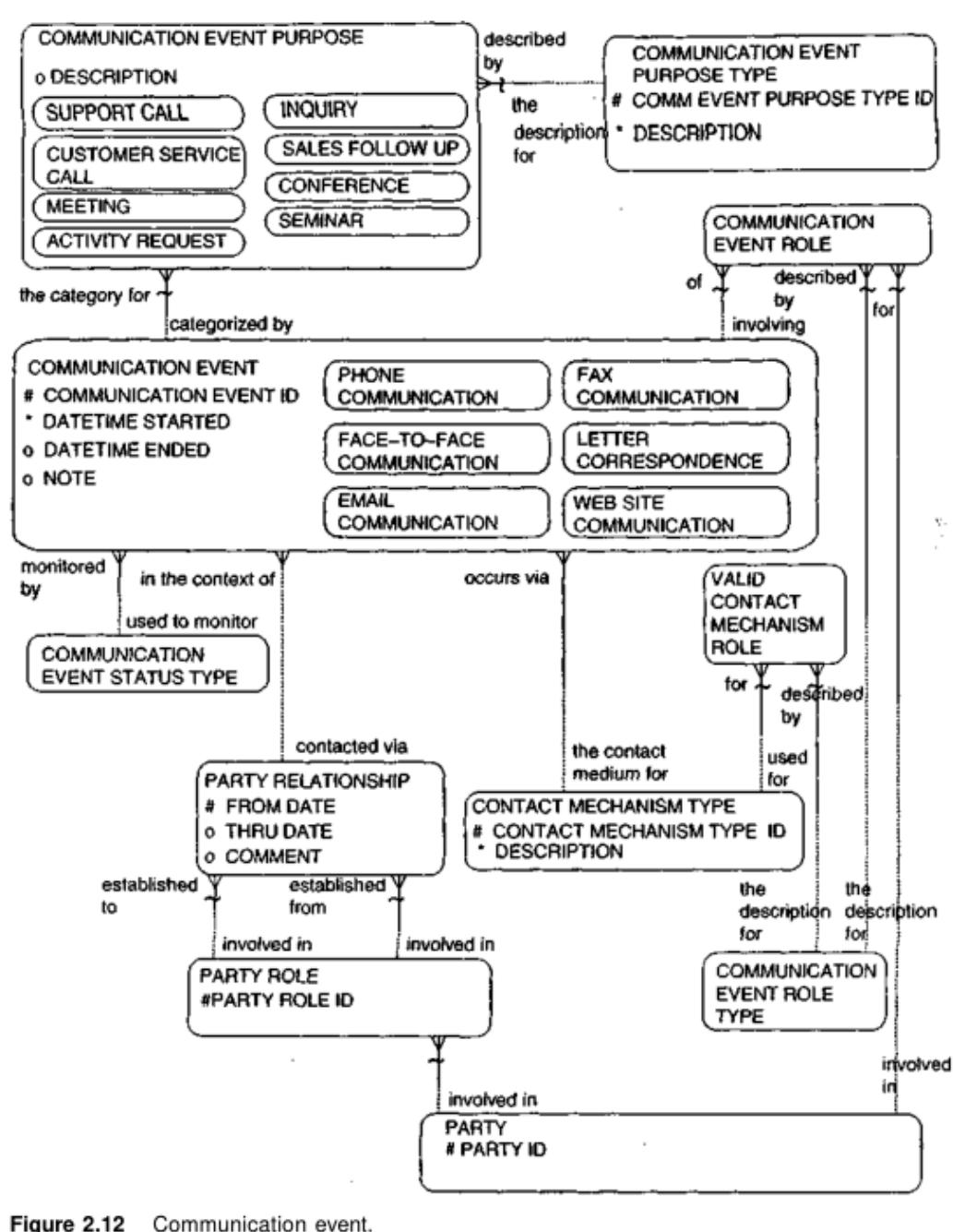


# แนวคิดของโครงงาน

- ใช้ Party Model จากหนังสือสร้างแอป
- CRUD Base Layer เชื่อมบุคคลและองค์กร ข้อมูลประเภทของสิ่งต่างๆ
- CRUD Information Layer เชื่อมรายละเอียด บุคคล/องค์กร
- CRUD Communication Layer การสื่อสารระหว่าง บุคคล/องค์กร
- แอป Cloud Native พร้อม Authen ด้วย JWT และ OAuth 2.0

# เลือก Party Model จากหนังสือมาต่อยอด

60 Chapter 2



**Figure 2.12** Communication event.

30 Chapter 2

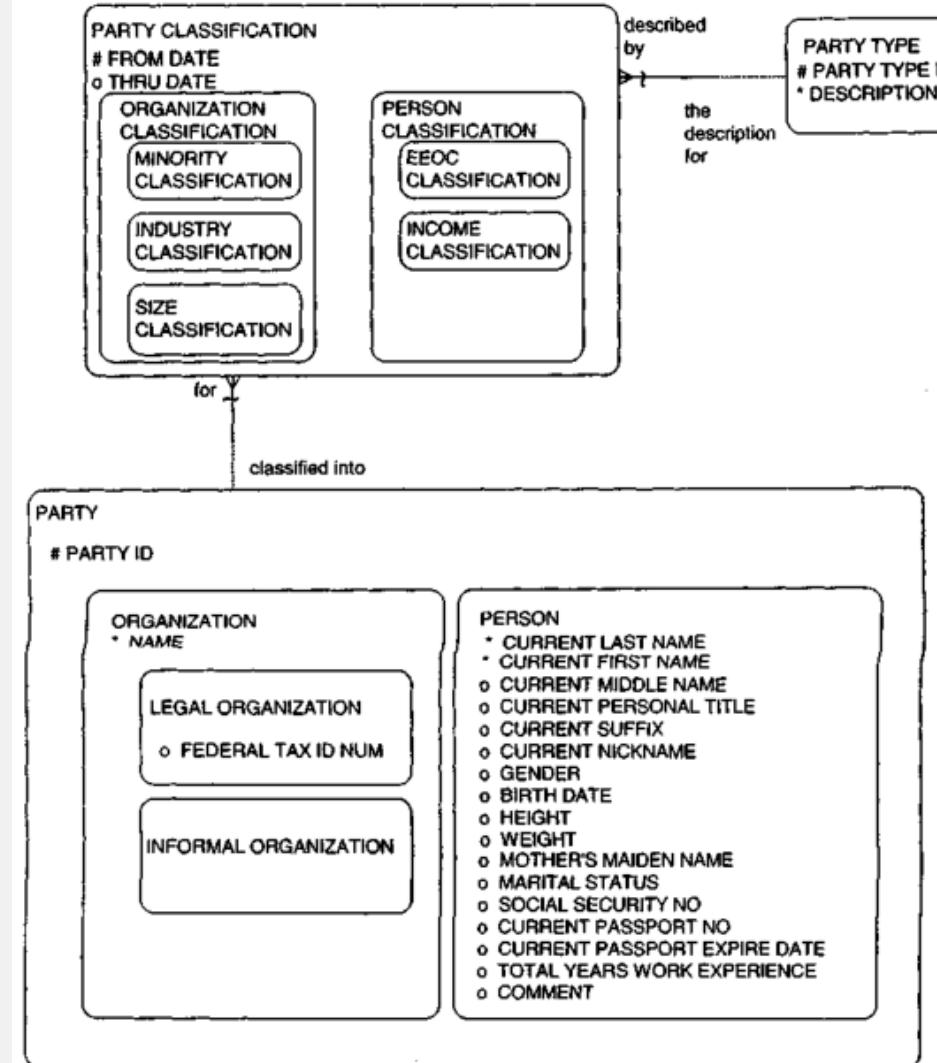
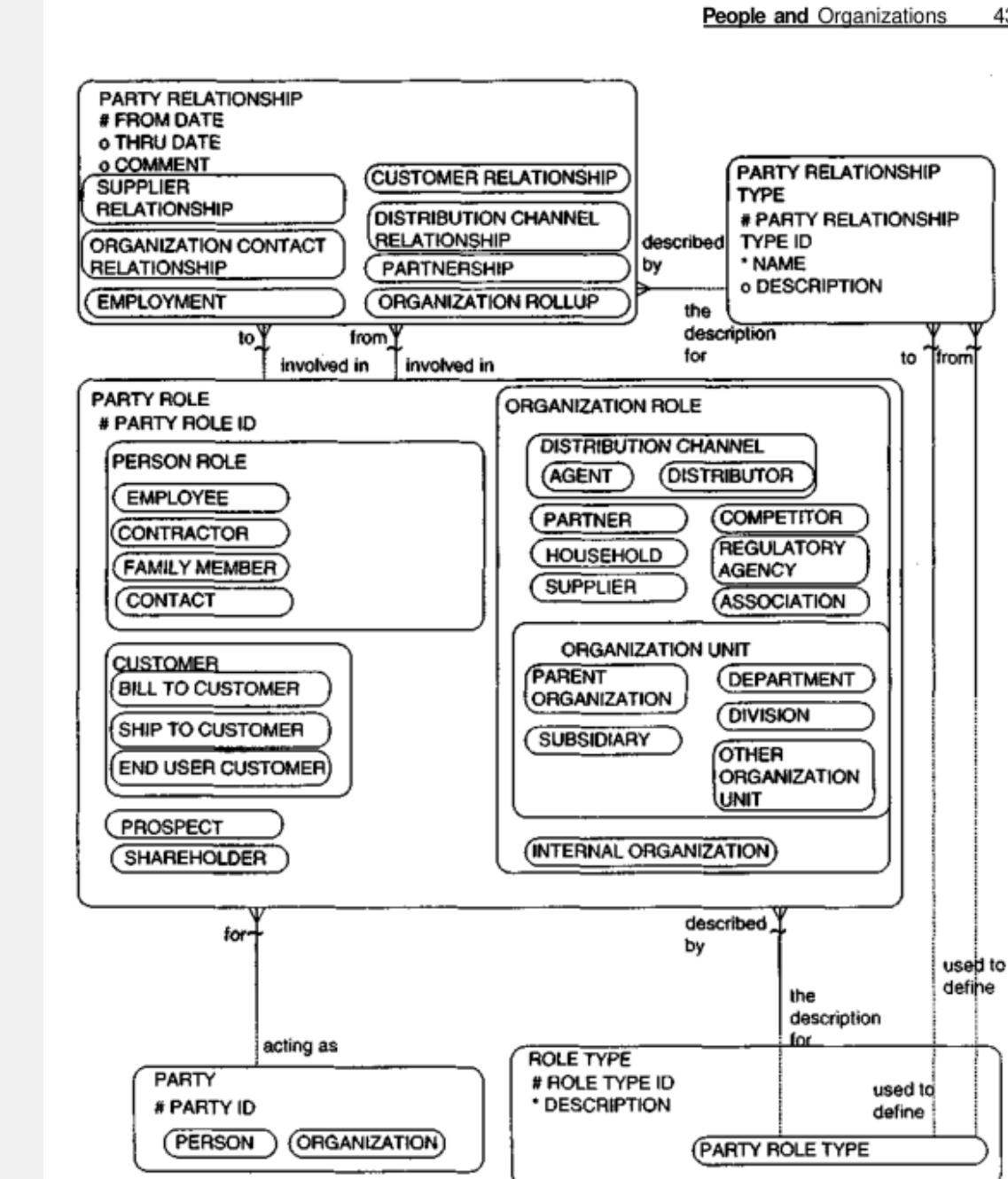


Figure 2.3 Par

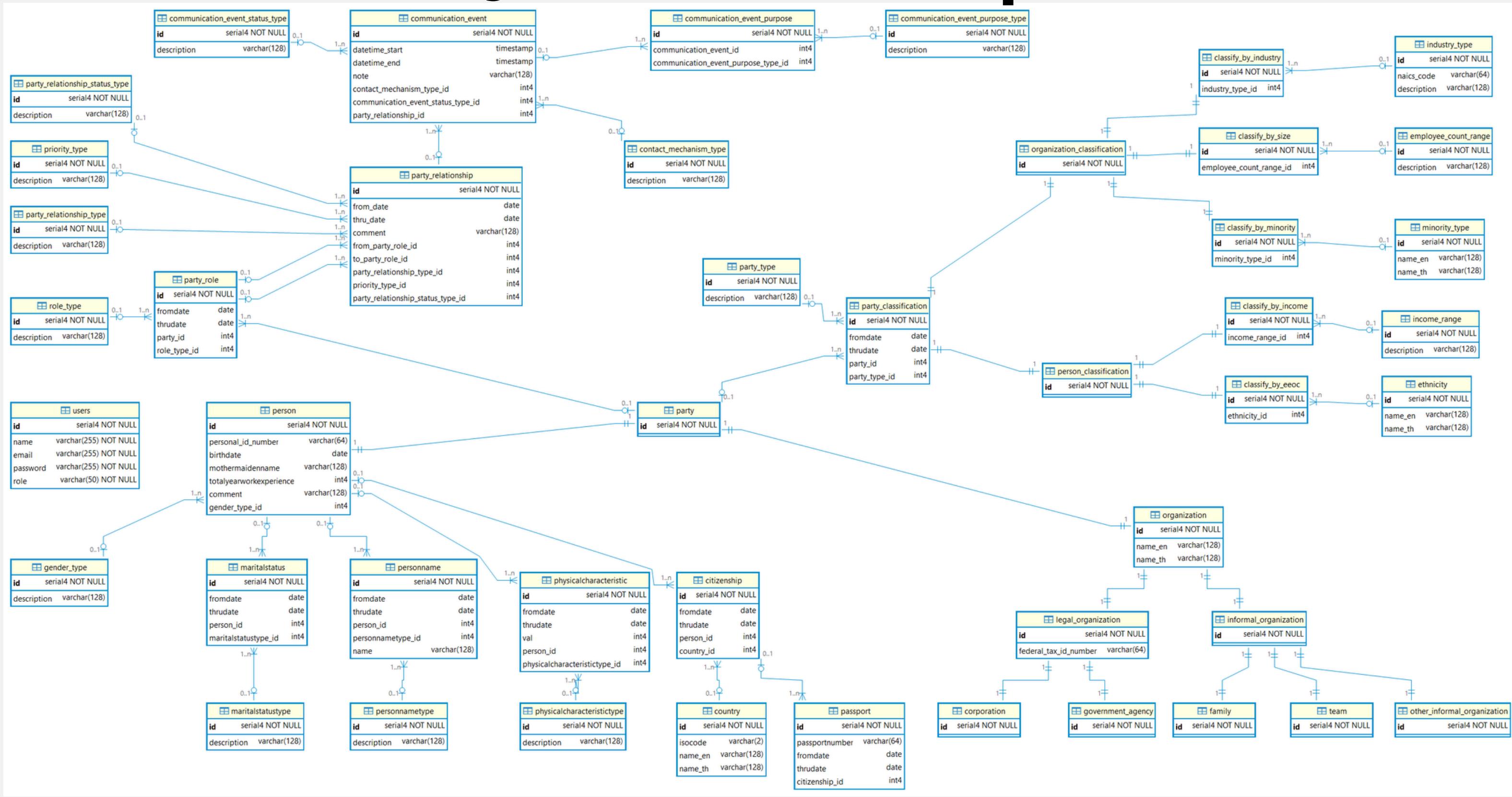
Therefore, Figure 2.3 shows a superentity named PARTY that has as its two subtypes PERSON and ORGANIZATION. This PARTY entity will enable storage of some of the common characteristics and relationships that people and organizations share.

**People and Organizations** 43

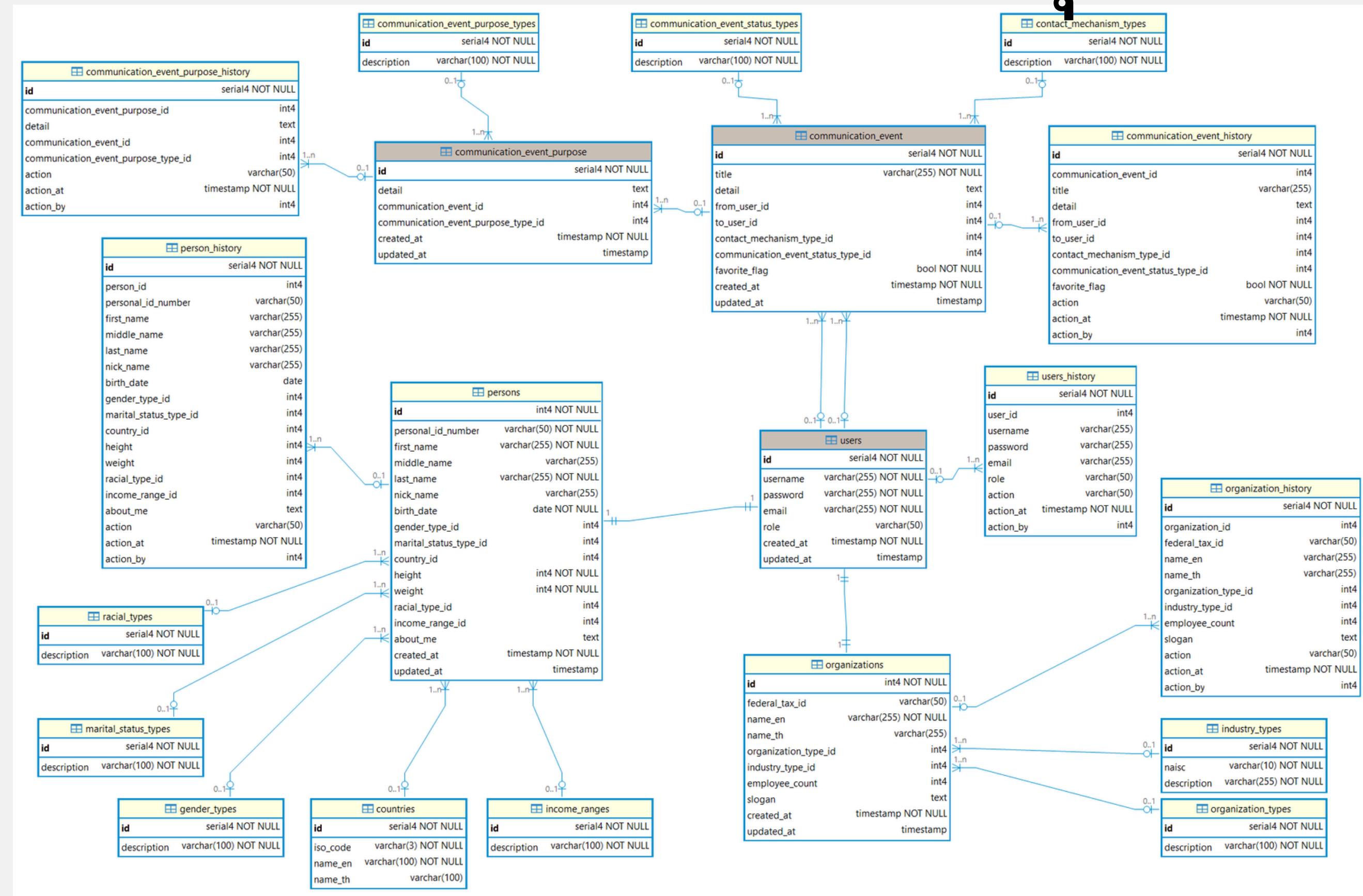


**Figure 2.6a** Common party relationships.

# นำ Party Model มา implement



# Database ឧប្បជ្ជកម្ម



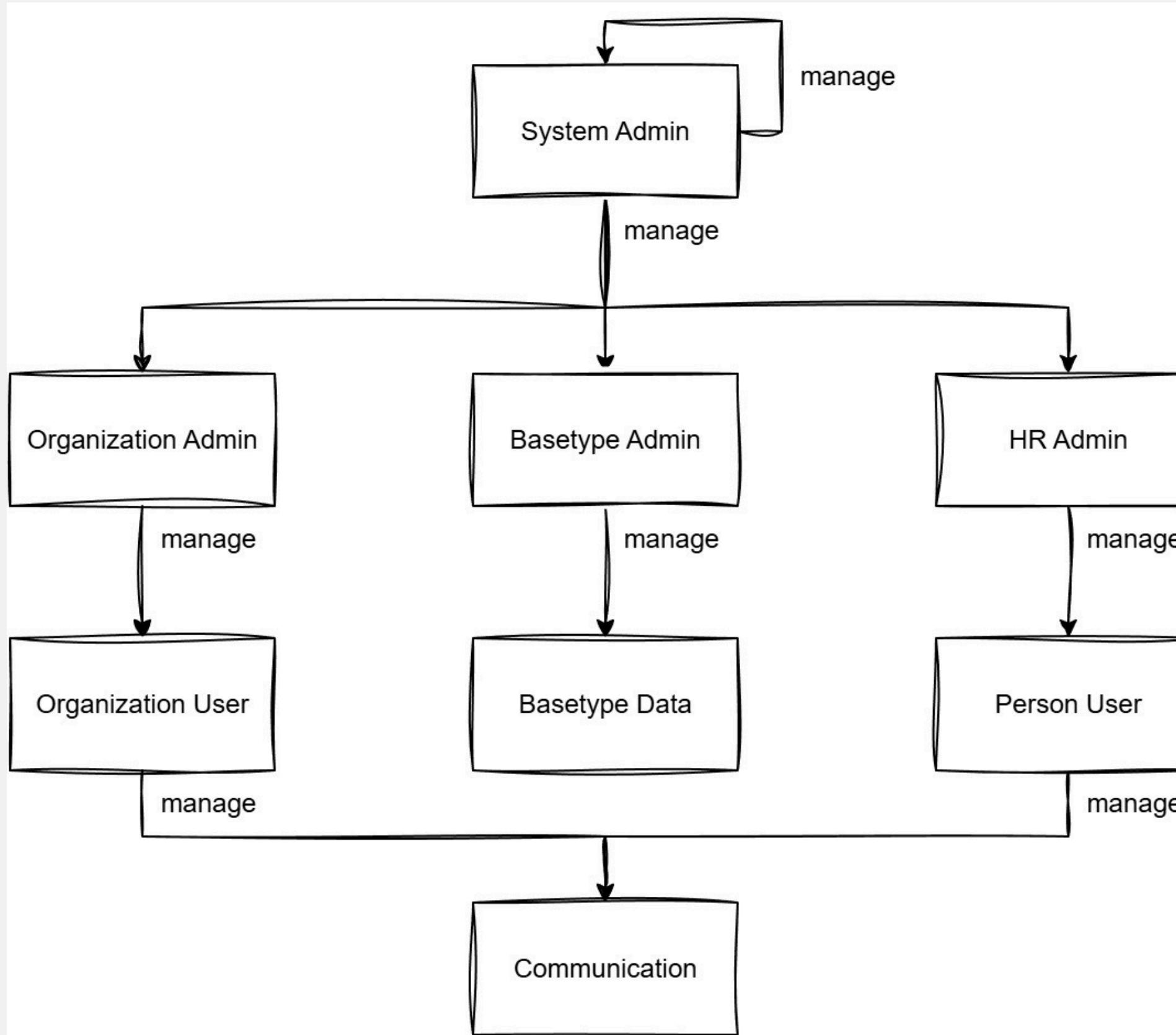
# การปรับปรุง Database

- ตัด subtype table ที่ไม่จำเป็นออกให้โครงสร้างกระชับ
- เปลี่ยนการเก็บข้อมูลที่แปรตามเวลา จาก junction table (fromdate-thru date) เป็น history table
- เพิ่มการเก็บประวัติว่าข้อมูลถูกทำอะไร เมื่อไหร่ โดยคร
- ลบ layer role และ relation เพื่อลดความซับซ้อน
- ย้ายการ classification จาก subtype หลายชั้นมาเก็บใน person และ organization table

**จากทฤษฎีสู่การลงมือทำ**

# ระบบและฟังก์ชันหลักของแอป

- ระบบ login รักษาความปลอดภัยผู้ใช้
- System Admin สร้างและจัดการ account admin ทุกประเภท
- Organization Admin เพิ่มข้อมูล organization เข้าสู่ระบบ
- HR Admin เพิ่มข้อมูล person เข้าสู่ระบบ
- Organization และ Person User บันทึกการสื่อสารผ่านแอป
- มีฟีเจอร์ star และ filter ขาเข้า-ขาออก คล้าย Gmail



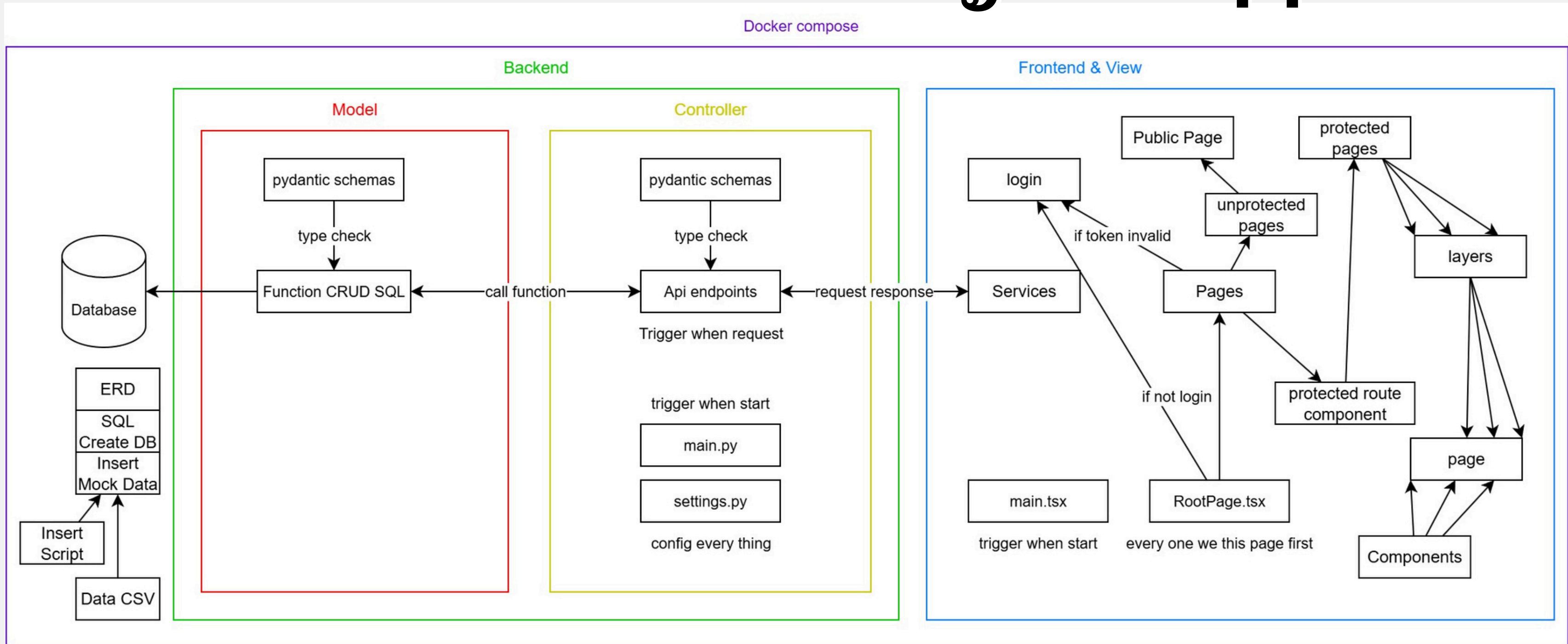
Organization Chart แสดงสิทธิ์ของแต่ละ Role ในแอปพลิเคชัน

# นำแนวคิดมาทำ Usecase Diagram



- มีทั้งหมด 6 role ในแอป
- System Admin จัดการ account Admin
- Basetype Admin จัดการข้อมูลพื้นฐาน เช่น country, gender\_type
- Organization Admin ดูแล user ประเภท organization
- HR Admin ดูแล user ประเภท person
- Organization User ใช้อเปบบันทึกการสื่อสาร
- Person User ใช้อเปบบันทึกการสื่อสาร

# สวนประกอบที่สำคัญของ App



# **Container Base**

The screenshot shows the Docker Desktop interface. The left sidebar has links for Ask Gordon (BETA), Containers (selected), Images, Volumes, Builds, Models (BETA), MCP Toolkit (BETA), Docker Hub, and Docker Scout. The Extensions section is collapsed. The main area is titled "Containers" with a "Give feedback" link. It says "View all your running containers and applications. [Learn more](#)". Below are two status sections: "Container CPU usage" (No containers are running) and "Container memory usage" (No containers are running). A "Show charts" link is next to the memory usage section. A search bar and a "Only show running containers" toggle are at the top of the container list. The container list table has columns: Name, Container ID, Image, Port(s), CPU (%), Memory usage..., Memory (%), and Actions. Four containers are listed:

	Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Actions
<input type="checkbox"/>	party_fullstack_docker5	-	-	-	N/A	N/A	N/A	<span style="color: blue;">[ ]</span> <span style="color: blue;">[ ... ]</span> <span style="color: red;">[ ]</span>
<input type="checkbox"/>	frontend-1	50fd23ffea6f	<a href="#">party_fullst</a>	<a href="#">5173:5173</a>	N/A	N/A	N/A	<span style="color: blue;">[ ]</span> <span style="color: blue;">[ ... ]</span> <span style="color: red;">[ ]</span>
<input type="checkbox"/>	db-1	65856ab5d5ad	<a href="#">postgres:1f</a>	<a href="#">5432:5432</a>	N/A	N/A	N/A	<span style="color: blue;">[ ]</span> <span style="color: blue;">[ ... ]</span> <span style="color: red;">[ ]</span>
<input type="checkbox"/>	backend-1	8aea297ad936	<a href="#">party_fullst</a>	<a href="#">8080:8080</a>	N/A	N/A	N/A	<span style="color: blue;">[ ]</span> <span style="color: blue;">[ ... ]</span> <span style="color: red;">[ ]</span>

At the bottom, it says "Showing 4 items". The footer shows "Engine running", system stats (RAM 1.71 GB, CPU 28.64%, Disk: 50.48 GB used / limit 1006.85 GB), a Terminal link, and a "New version available" notification.

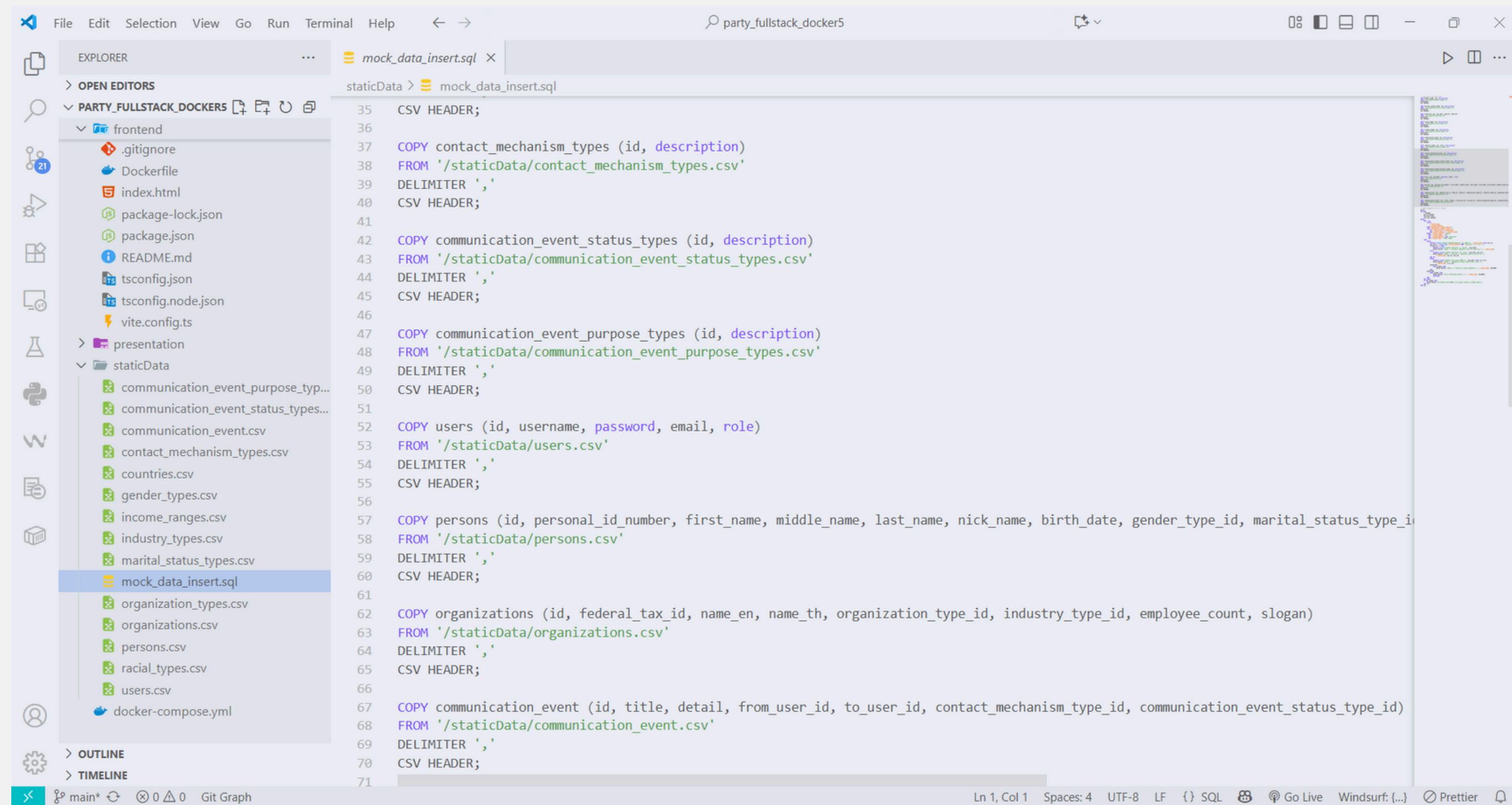
Compose และ Container ของ Project ที่ทำ

# **Database**

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** party\_fullstack\_docker5.
- Explorer:** Shows a file tree under 'PARTY\_FULLSTACK\_DOCKERS'. Key files include 'init\_db\_v1.sql', 'init\_db\_v2.sql', 'insert\_script.txt', and 'reset\_max\_seq\_id\_all.sql'.
- Editor:** The main area displays three SQL scripts:
  - communication\_event\_history:** A table definition with columns: title (VARCHAR(255)), detail (TEXT), from\_user\_id (INT), to\_user\_id (INT), contact\_mechanism\_type\_id (INT), communication\_event\_status\_type\_id (INT), favorite\_flag (BOOLEAN NOT NULL DEFAULT FALSE), action (VARCHAR(50)), action\_at (TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP), and action\_by (INT).
  - communication\_event\_purpose:** A table definition with columns: id (SERIAL PRIMARY KEY), detail (TEXT), communication\_event\_id (INT REFERENCES communication\_event(id) ON DELETE SET NULL), communication\_event\_purpose\_type\_id (INT REFERENCES communication\_event\_purpose\_types(id) ON DELETE SET NULL), created\_at (TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP), and updated\_at (TIMESTAMP).
  - communication\_event\_purpose\_history:** A table definition with columns: id (SERIAL PRIMARY KEY), communication\_event\_purpose\_id (INT REFERENCES communication\_event\_purpose(id) ON DELETE SET NULL), detail (TEXT), communication\_event\_id (INT), communication\_event\_purpose\_type\_id (INT), action (VARCHAR(50)), action\_at (TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP), and action\_by (INT).
- Bottom Bar:** Includes icons for main, Git Graph, and various status indicators like sitthipong\_tufA15 (3 weeks ago), Ln 169, Col 17, Spaces: 4, UTF-8, CRLF, SQL, Go Live, Windsurf: ..., Prettier, and a bell icon.

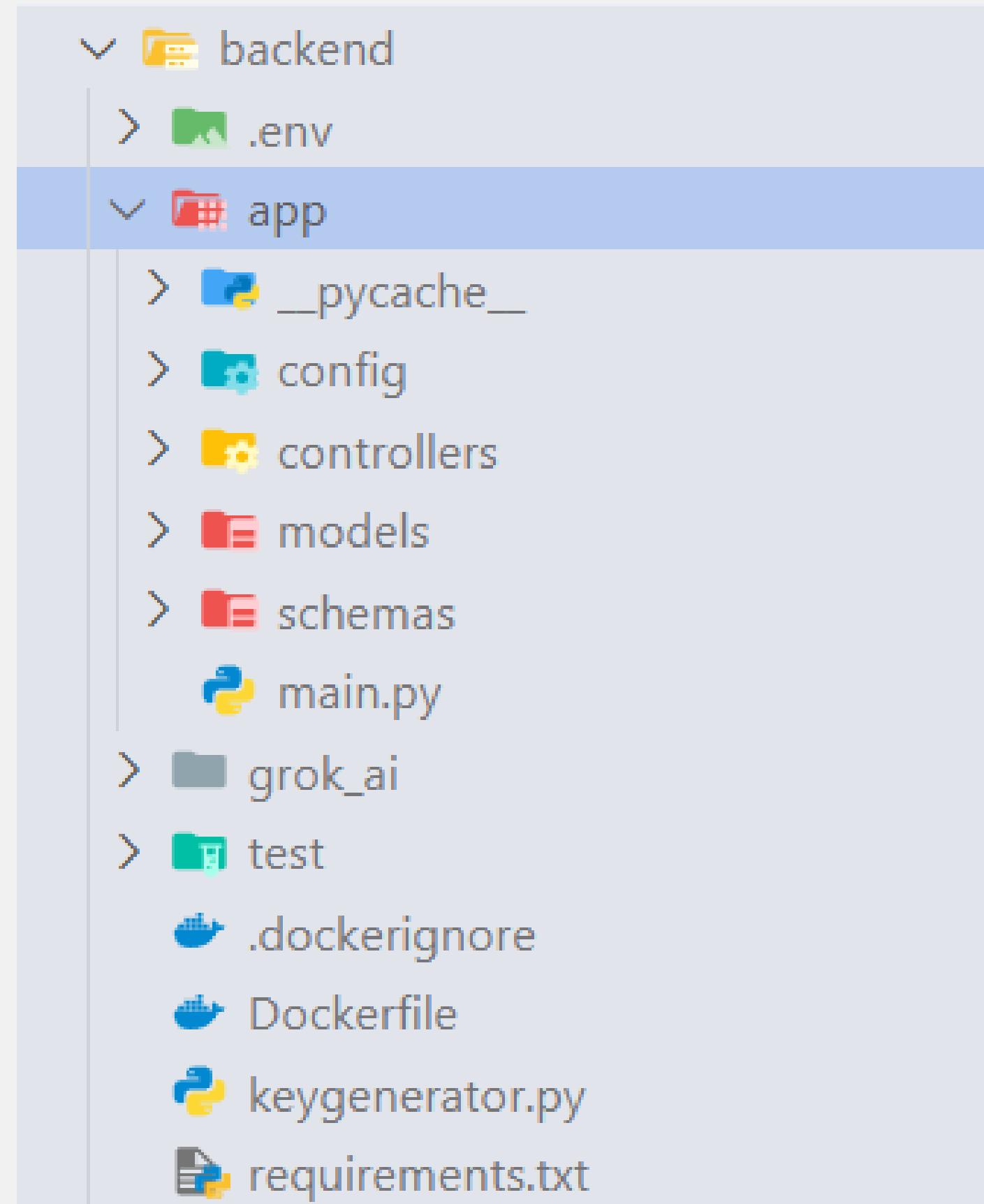
# SQL ที่ใช้ในการสร้าง Database



```
35 CSV HEADER;
36
37 COPY contact_mechanism_types (id, description)
38 FROM '/staticData/contact_mechanism_types.csv'
39 DELIMITER ','
40 CSV HEADER;
41
42 COPY communication_event_status_types (id, description)
43 FROM '/staticData/communication_event_status_types.csv'
44 DELIMITER ','
45 CSV HEADER;
46
47 COPY communication_event_purpose_types (id, description)
48 FROM '/staticData/communication_event_purpose_types.csv'
49 DELIMITER ','
50 CSV HEADER;
51
52 COPY users (id, username, password, email, role)
53 FROM '/staticData/users.csv'
54 DELIMITER ','
55 CSV HEADER;
56
57 COPY persons (id, personal_id_number, first_name, middle_name, last_name, nick_name, birth_date, gender_type_id, marital_status_type_id, organization_type_id, industry_type_id, employee_count, slogan)
58 FROM '/staticData/persons.csv'
59 DELIMITER ','
60 CSV HEADER;
61
62 COPY organizations (id, federal_tax_id, name_en, name_th, organization_type_id, industry_type_id, employee_count, slogan)
63 FROM '/staticData/organizations.csv'
64 DELIMITER ','
65 CSV HEADER;
66
67 COPY communication_event (id, title, detail, from_user_id, to_user_id, contact_mechanism_type_id, communication_event_status_type_id)
68 FROM '/staticData/communication_event.csv'
69 DELIMITER ','
70 CSV HEADER;
```

# Insert Mock Data

# Backend



- แบ่ง backend เป็น 4 ส่วนหลัก
- Schemas ตรวจสอบ datatype ของ payload
- Models จัดการการเชื่อมต่อกับ database
- Controllers ควบคุมและจัดการ request
- main.py รวมทุก API endpoint
- ไม่มี View เพราะ Frontend ทำหน้าที่เป็น View

EXPLORER

> OPEN EDITORS

PARTY\_FULLSTACK\_DOCKERS

- backend
- app
- models
- schemas

  - \_pycache\_
  - communication\_event\_history.py
  - communication\_event\_purpose...
  - communication\_event\_status\_ty...
  - communication\_event.py
  - contact\_mechanism\_types.py
  - countries.py
  - gender\_type.py
  - income\_ranges.py
  - industry\_types.py
  - marital\_status\_types.py
  - organization\_history.py
  - organization\_types.py
  - organization.py
  - person\_history.py
  - person.py
  - racial\_types.py
  - user.py
  - users\_history.py

main.py

grok\_ai

test

.dockerignore

Dockerfile

keygenerator.py

```

backend > app > schemas > user.py > UserUpdate
1  from pydantic import BaseModel, EmailStr
2  from typing import Optional
3  from datetime import datetime
4
5  # Schema สำหรับสร้างผู้ใช้
6  class UserCreate(BaseModel):
7      username: str
8      email: EmailStr
9      password: str
10     role: Optional[str] = None
11
12 # Schema สำหรับอัปเดตผู้ใช้
13 class UserUpdate(BaseModel):
14     username: Optional[str] = None
15     email: Optional[EmailStr] = None
16     password: Optional[str] = None
17     role: Optional[str] = None
18
19 # Schema สำหรับแสดงผลผู้ใช้
20 class UserOut(BaseModel):
21     id: int
22     username: str
23     email: EmailStr
24     role: Optional[str] = None
25     created_at: datetime
26     updated_at: Optional[datetime] = None
27
28     class Config:
29         from_attributes = True
30
31 # Schema สำหรับล็อกอิน
32 class UserLogin(BaseModel):
33     email: EmailStr
34     password: str

```

EXPLORER

> OPEN EDITORS

PARTY\_FULLSTACK\_DOCKERS

- backend
- app
- schemas

  - income\_ranges.py
  - industry\_types.py
  - marital\_status\_types.py
  - organization\_history.py
  - organization\_types.py
  - organization.py
  - person\_history.py
  - person.py
  - racial\_types.py
  - user.py
  - users\_history.py

main.py

grok\_ai

test

.dockerignore

Dockerfile

keygenerator.py

requirements.txt

database\_note

db-data

docs

frontend

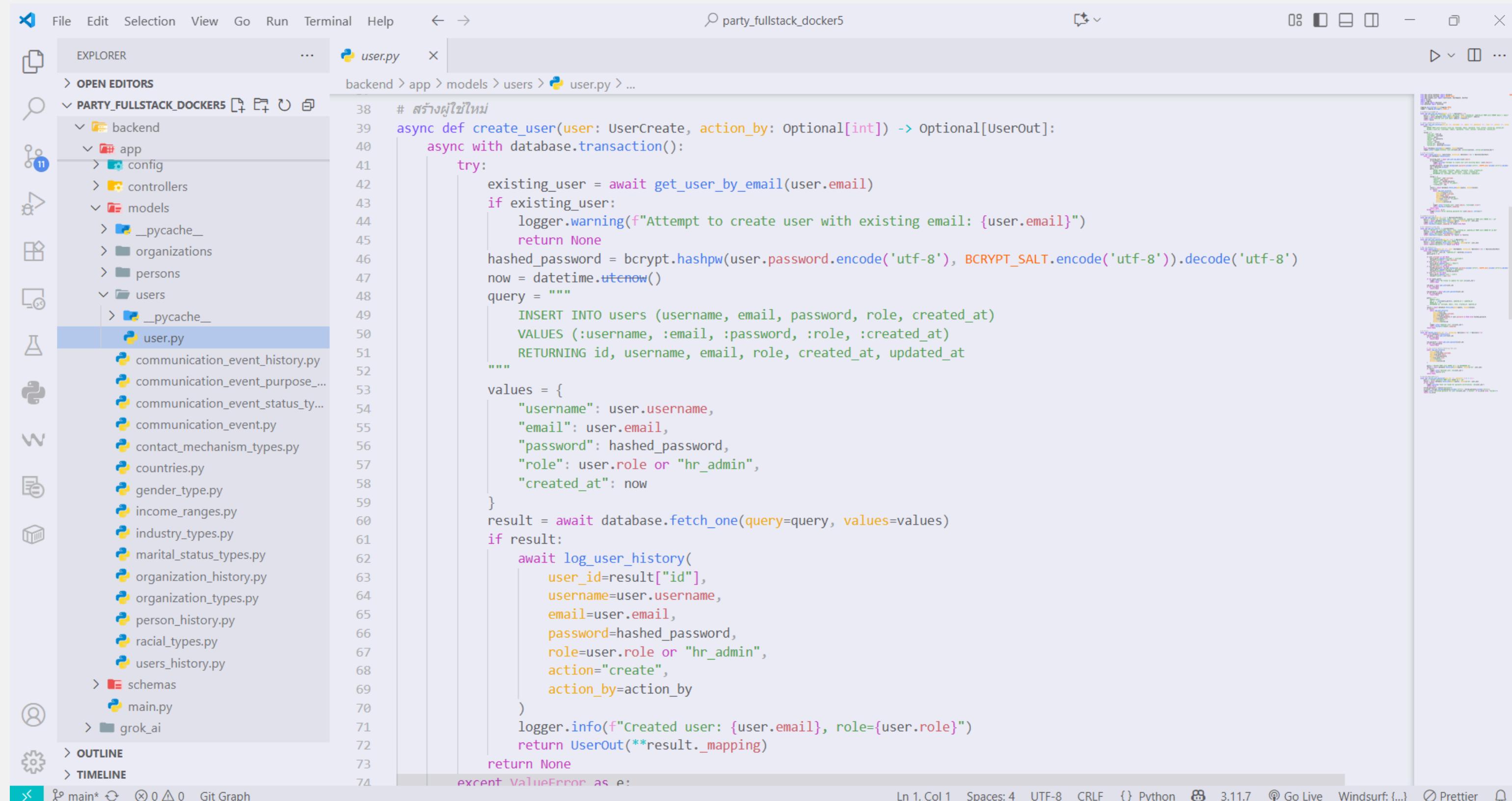
presentation

```

backend > app > schemas > users_history.py > UsersHistoryOut
1  from pydantic import BaseModel
2  from typing import Optional
3  from datetime import datetime
4
5  # Schema สำหรับแสดงผล users_history
6  class UsersHistoryOut(BaseModel):
7      id: int
8      user_id: Optional[int]
9      username: Optional[str]
10     password: Optional[str]
11     email: Optional[str]
12     role: Optional[str]
13     action: Optional[str]
14     action_at: datetime
15     action_by: Optional[int]
16
17     class Config:
18         from_attributes = True

```

## schema ของ endpoint user, user\_history



```
# สร้างผู้ใช้ใหม่
async def create_user(user: UserCreate, action_by: Optional[int]) -> Optional[UserOut]:
    async with database.transaction():
        try:
            existing_user = await get_user_by_email(user.email)
            if existing_user:
                logger.warning(f"Attempt to create user with existing email: {user.email}")
                return None
            hashed_password = bcrypt.hashpw(user.password.encode('utf-8'), BCRYPT_SALT.encode('utf-8')).decode('utf-8')
            now = datetime.utcnow()
            query = """
                INSERT INTO users (username, email, password, role, created_at)
                VALUES (:username, :email, :password, :role, :created_at)
                RETURNING id, username, email, role, created_at, updated_at
            """
            values = {
                "username": user.username,
                "email": user.email,
                "password": hashed_password,
                "role": user.role or "hr_admin",
                "created_at": now
            }
            result = await database.fetch_one(query=query, values=values)
            if result:
                await log_user_history(
                    user_id=result["id"],
                    username=user.username,
                    email=user.email,
                    password=hashed_password,
                    role=user.role or "hr_admin",
                    action="create",
                    action_by=action_by
                )
                logger.info(f"Created user: {user.email}, role={user.role}")
                return UserOut(**result._mapping)
        except ValueError as e:
            logger.error(f"ValueError: {e}")
    return None
```

## ตัวอย่าง Code Model (function CRUD SQL)

```
party_fullstack_dockers
user.py

backend > app > controllers > users > user.py > ...
22 # ตรวจสอบผู้ใช้จาก JWT token
23 async def get_current_user(token: str = Depends(oauth2_scheme)):
24     try:
25         payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
26         logger.info(f"Decoded JWT payload: {payload}")
27         user_id: str = payload.get("sub")
28         role: str = payload.get("role")
29         if user_id is None:
30             logger.error("Invalid token: missing 'sub'")
31             raise HTTPException(status_code=401, detail="Invalid token")
32         logger.info(f"Authenticated user: id={user_id}, role={role}")
33         return {"id": int(user_id), "role": role}
34     except JWTError as e:
35         logger.error(f"JWT decode failed: {str(e)}")
36         raise HTTPException(status_code=401, detail="Invalid token")
37
38 # ดึง role ของผู้ใช้ปัจจุบัน
39 @router.get("/myrole")
40 async def get_user_role(current_user: dict = Depends(get_current_user)):
41     logger.info(f"Retrieved role for user: id={current_user['id']}, role={current_user['role']}")
42     return {"role": current_user["role"]}
43
44 # สร้างผู้ใช้ใหม่ (system_admin เท่านั้น)
45 @router.post("/", response_model=UserOut)
46 async def create_user_endpoint(user: UserCreate, current_user: dict = Depends(get_current_user)):
47     if current_user["role"] != "system_admin":
48         logger.warning(f"Unauthorized attempt to create user by user: id={current_user['id']}, role={current_user['role']}")
49         raise HTTPException(status_code=403, detail="System admin access required")
50     valid_roles = ["system_admin", "hr_admin", "organization_admin", "basetype_admin"]
51     if user.role is not None and user.role not in valid_roles:
52         logger.warning(f"Invalid role provided: {user.role}")
53         raise HTTPException(status_code=422, detail=f"Role must be one of {valid_roles}")
54     result = await create_user(user, action_by=current_user["id"])
55     if not result:
56         logger.warning(f"Failed to create user: {user.email}")
57         raise HTTPException(status_code=400, detail="Email already exists")
58     logger.info(f"Created user: {user.email}, role={result.role}")
```

## ตัวอย่าง Code Controller (API endpoint)

```

● ○ ●
1  from dotenv import load_dotenv
2  from fastapi import FastAPI
3  from fastapi.middleware.cors import CORSMiddleware
4  from contextlib import asynccontextmanager
5  from app.config.database import database
6  from app.controllers.auth.auth import router as auth_router
7  from app.controllers.users.user import router as user_router
8  from app.controllers.persons.person import router as person_router
9  from app.controllers.organizations.organization import router as organization_router
10 from app.controllers.gender_type import router as gender_type_router
11 from app.controllers.marital_status_types import router as marital_status_type_router
12 from app.controllers.countries import router as country_router
13 from app.controllers.racial_types import router as racial_type_router
14 from app.controllers.income_ranges import router as income_range_router
15 from app.controllers.organization_types import router as organization_type_router
16 from app.controllers.industry_types import router as industry_type_router
17 from app.controllers.contact_mechanism_types import router as contact_mechanism_type_router
18 from app.controllers.communication_event_status_types import router as communication_event_status_type_router
19 from app.controllers.communication_event_purpose_types import router as communication_event_purpose_type_router
20 from app.controllers.users_history import router as users_history_router
21 from app.controllers.person_history import router as person_history_router
22 from app.controllers.organization_history import router as organization_history_router
23 from app.controllers.communication_event import router as communication_event_router
24 from app.controllers.communication_event_history import router as communication_event_history_router
25
26 load_dotenv()
27
28 # Define Lifespan event handler for FastAPI application
29 @asynccontextmanager
30 async def lifespan(app: FastAPI):
31     await database.connect()
32     yield
33     await database.disconnect()

```

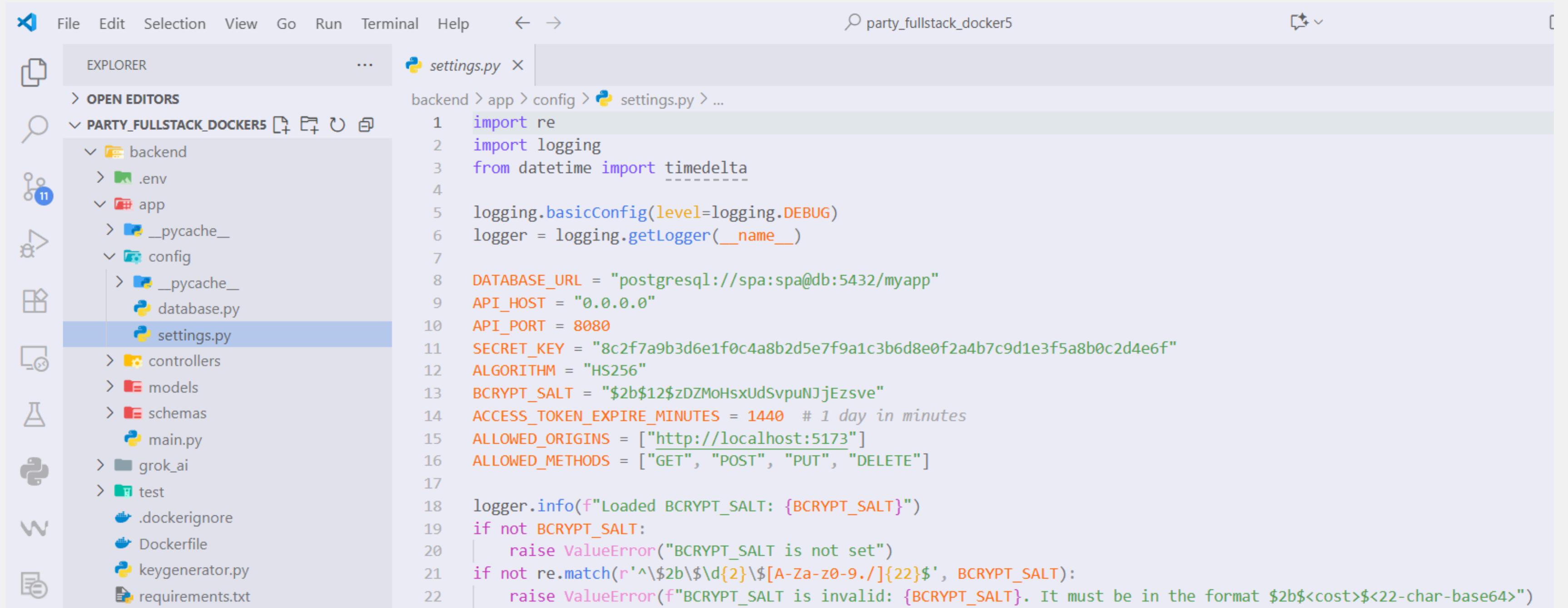
  

```

● ○ ●
1  # Initialize FastAPI app with Lifespan event handler
2  app = FastAPI(lifespan=lifespan)
3
4  # Configure CORS to allow all origins for development
5  app.add_middleware(
6      CORSMiddleware,
7      allow_origins=["*"],
8      allow_credentials=True,
9      allow_methods=["*"],
10     allow_headers=["*"],
11 )
12
13 app.include_router(auth_router)
14 app.include_router(user_router)
15 app.include_router(person_router)
16 app.include_router(organization_router)
17 app.include_router(gender_type_router)
18 app.include_router(marital_status_type_router)
19 app.include_router(country_router)
20 app.include_router(racial_type_router)
21 app.include_router(income_range_router)
22 app.include_router(organization_type_router)
23 app.include_router(industry_type_router)
24 app.include_router(contact_mechanism_type_router)
25 app.include_router(communication_event_status_type_router)
26 app.include_router(communication_event_purpose_type_router)
27 app.include_router(users_history_router)
28 app.include_router(person_history_router)
29 app.include_router(organization_history_router)
30 app.include_router(communication_event_router)
31 app.include_router(communication_event_history_router)
32
33 @app.get("/")
34 async def root():
35     return {"message": "FastAPI Backend", "github": "https://github.com/sszz-TH/party_fullstack_docker5"}

```

# main.py រាយទួរ API endpoint



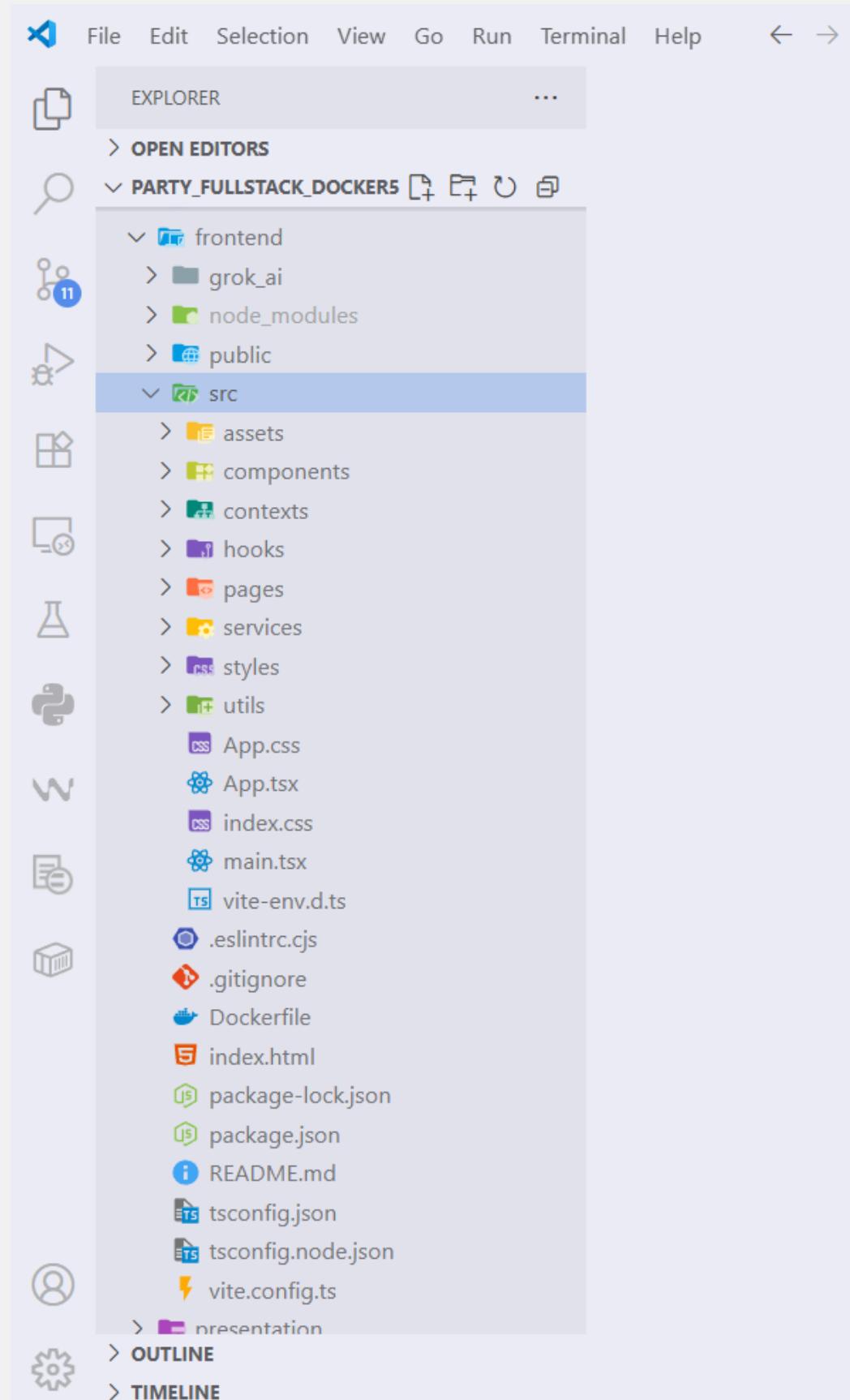
The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** party\_fullstack\_docker5
- Explorer:** Shows the project structure under "PARTY\_FULLSTACK\_DOCKERS".
  - backend: .env
  - app: \_\_pycache\_\_, config
    - \_\_pycache\_\_
    - database.py
  - settings.py (selected)
  - controllers
  - models
  - schemas
  - main.py
  - grok\_ai
  - test
    - .dockerignore
    - Dockerfile
    - keygenerator.py
  - requirements.txt
- Editor:** Displays the content of "settings.py".

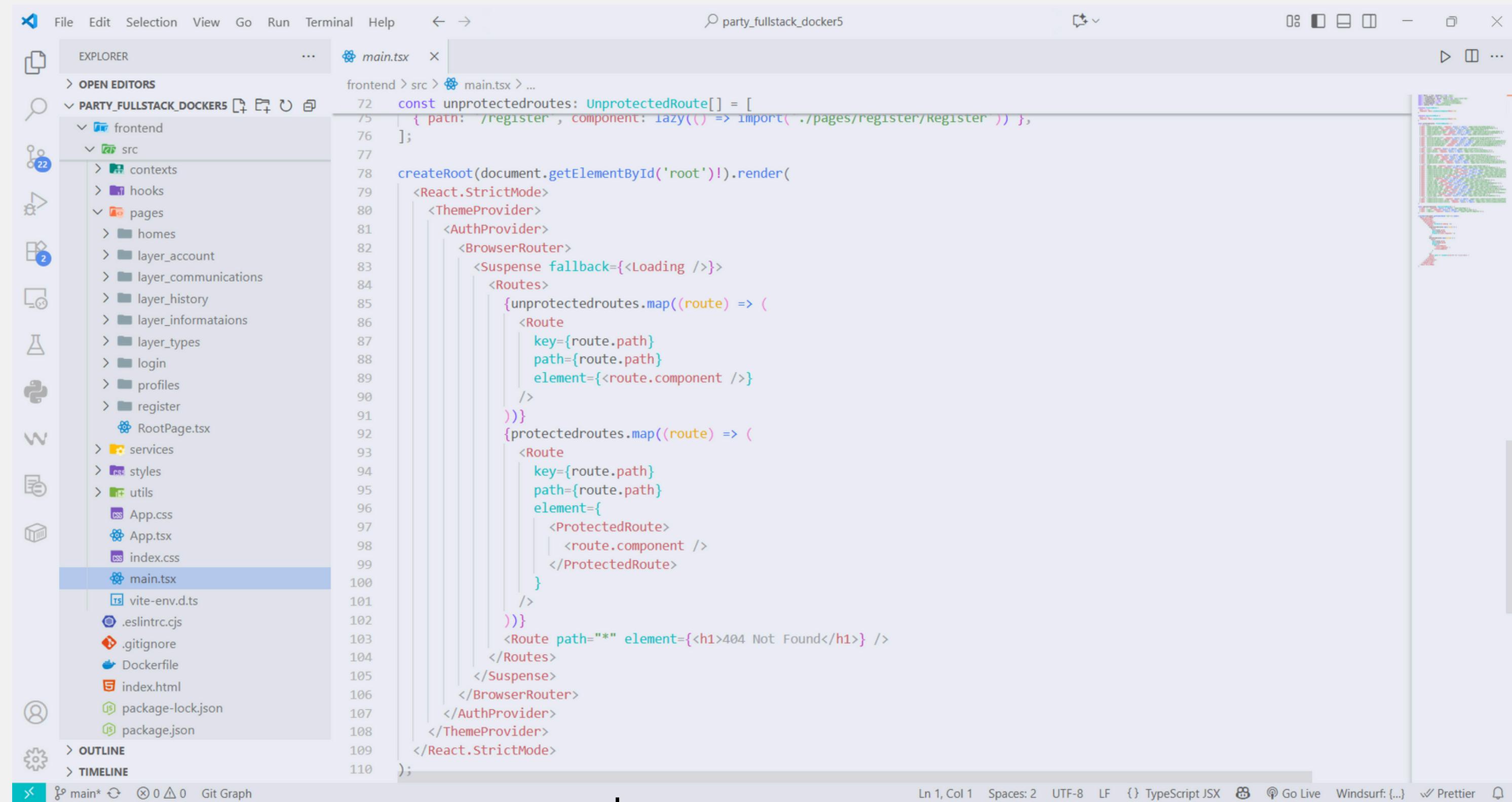
```
backend > app > config > settings.py > ...
1 import re
2 import logging
3 from datetime import timedelta
4
5 logging.basicConfig(level=logging.DEBUG)
6 logger = logging.getLogger(__name__)
7
8 DATABASE_URL = "postgresql://spa:spa@db:5432/myapp"
9 API_HOST = "0.0.0.0"
10 API_PORT = 8080
11 SECRET_KEY = "8c2f7a9b3d6e1f0c4a8b2d5e7f9a1c3b6d8e0f2a4b7c9d1e3f5a8b0c2d4e6f"
12 ALGORITHM = "HS256"
13 BCRYPT_SALT = "$2b$12$zDZMoHsxUdSvpuNJjEzsve"
14 ACCESS_TOKEN_EXPIRE_MINUTES = 1440 # 1 day in minutes
15 ALLOWED_ORIGINS = ["http://localhost:5173"]
16 ALLOWED_METHODS = ["GET", "POST", "PUT", "DELETE"]
17
18 logger.info(f"Loaded BCRYPT_SALT: {BCRYPT_SALT}")
19 if not BCRYPT_SALT:
20     raise ValueError("BCRYPT_SALT is not set")
21 if not re.match(r'^\$2b\$\\d{2}@[A-Za-z0-9./]{22}$', BCRYPT_SALT):
22     raise ValueError(f"BCRYPT_SALT is invalid: {BCRYPT_SALT}. It must be in the format $2b$<cost>$<22-char-base64>")
```

การ deploy ในอนาคต สามารถ setting ทุกอย่างได้ในที่เดียว

# Frontend



- จัดวาง folder ตามแนวทาง Born to Dev จาก YouTube
- มีบาง folder เพื่อไว้ แม้ยังไม่ได้ใช้งาน
- Components เก็บส่วนประกอบเว็บที่ใช้ช้าได้ สะดวกต่อการเรียกใช้
- Contexts เก็บ context ทำหน้าที่เหมือน global variable
- Pages เก็บทุกหน้าเว็บของโปรเจกต์
- Services เก็บไฟล์ที่ใช้ติดต่อกับ backend
- Styles เก็บ color scheme และ theme (light/dark)
- Utils เก็บฟังก์ชันทั่วไป เช่น จัดการวันที่และเวลา
- main.tsx ทำหน้าที่ router จัดการ mapping path ไปยังหน้าเว็บ
- RootPage.tsx หน้าแรกที่ผู้ใช้เห็น นำทางไป dashboard ตามผู้ใช้ที่ล็อกอิน



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** party\_fullstack\_docker5
- Editor:** The main.tsx file is open, displaying the following code:

```
const unprotectedroutes: UnprotectedRoute[] = [
  { path: '/register', component: lazy(() => import('./pages/register/Register')) },
];

createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <ThemeProvider>
      <AuthProvider>
        <BrowserRouter>
          <Suspense fallback={<Loading />}>
            <Routes>
              {unprotectedroutes.map((route) => (
                <Route
                  key={route.path}
                  path={route.path}
                  element={<route.component />}
                />
              ))}
              {protectedroutes.map((route) => (
                <Route
                  key={route.path}
                  path={route.path}
                  element={
                    <ProtectedRoute>
                      <route.component />
                    </ProtectedRoute>
                  }
                />
              ))}
            </Routes>
          </Suspense>
        </BrowserRouter>
      </AuthProvider>
    </ThemeProvider>
  </React.StrictMode>
);
```

- Explorer:** Shows the project structure under 'PARTY\_FULLSTACK\_DOCKERS'. The 'frontend' folder contains 'src' which includes 'contexts', 'hooks', 'pages' (with subfolders 'homes', 'layer\_account', 'layer\_communications', 'layer\_history', 'layer\_informataions', 'layer\_types', 'login', 'profiles', 'register', 'RootPage.tsx'), 'services', 'styles', 'utils' (with files 'App.css', 'App.tsx', 'index.css'), and 'main.tsx' (which is currently selected).
- Right Panel:** A Git Graph visualization showing the commit history of the repository.
- Bottom Status Bar:** Ln 1, Col 1, Spaces: 2, UTF-8, LF, TypeScript JSX, Go Live, Windsurf: ..., Prettier.

main.tsx เป็นไฟล์เริ่มต้นสำหรับรัน frontend container

The screenshot shows the Visual Studio Code interface with the following details:

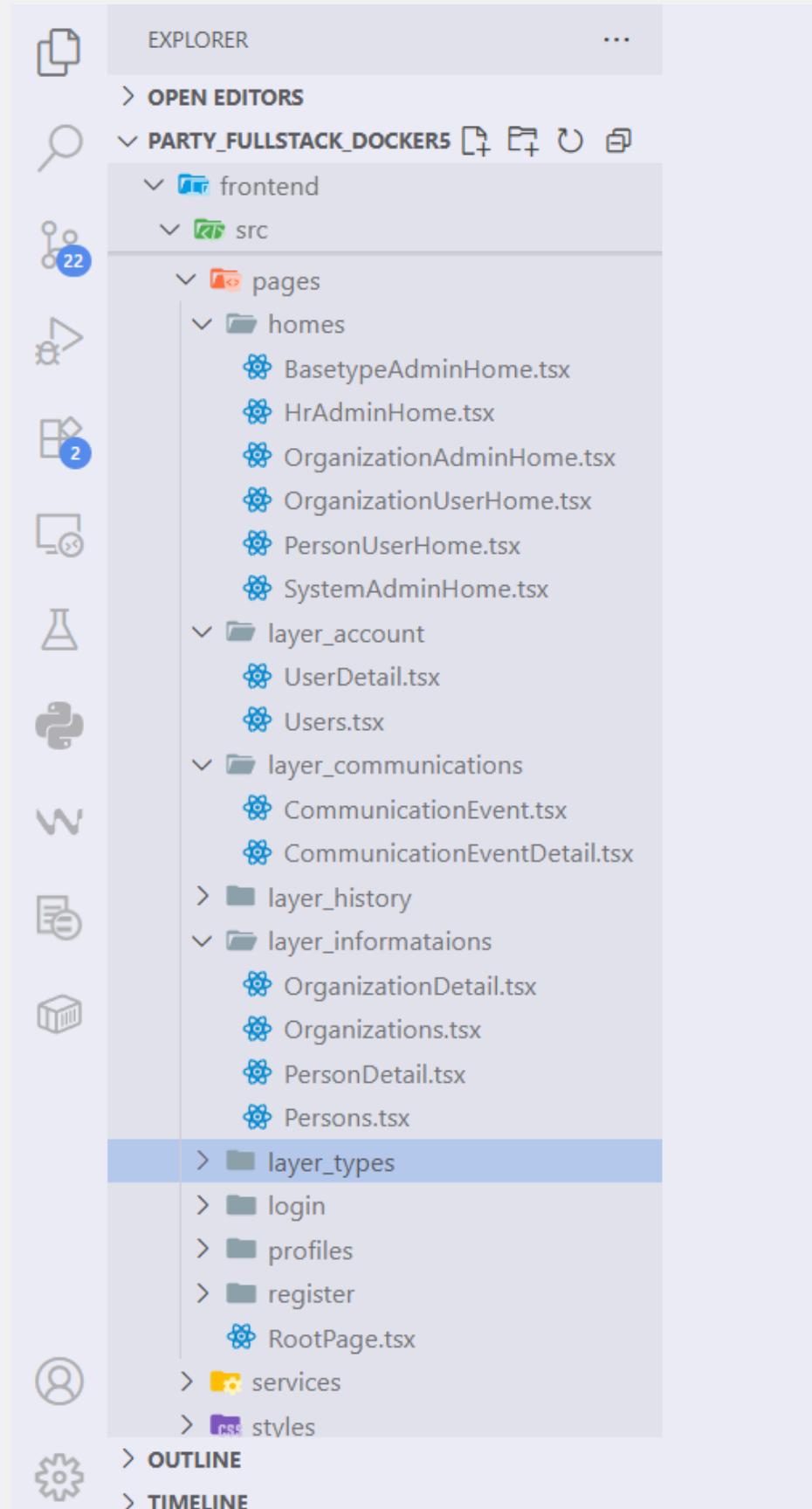
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** party\_fullstack\_docker5.
- Explorer:** Shows the project structure under "PARTY\_FULLSTACK\_DOCKERS".
  - frontend
  - src
    - contexts
    - hooks
    - pages
      - homes
      - layer\_account
      - layer\_communications
      - layer\_history
      - layer\_informataions
      - layer\_types
      - login
      - profiles
      - register
    - RootPage.tsx
    - services
    - styles
    - utils
      - App.css
      - App.tsx
      - index.css
      - main.tsx
      - vite-env.d.ts
    - .eslintrc.js
    - .gitignore
    - Dockerfile
    - index.html
    - package-lock.json
    - package.json
- Editor:** The "RootPage.tsx" file is open, showing TypeScript code for a React component.

```
7  export default function RootPage() {
8    const { setIsAuthenticated, setRole } = useContext(AuthContext);
9    const navigate = useNavigate();
10   const { isDarkMode } = useTheme();

11   useEffect(() => {
12     const token = Cookies.get('access_token');
13     const role = Cookies.get('role');

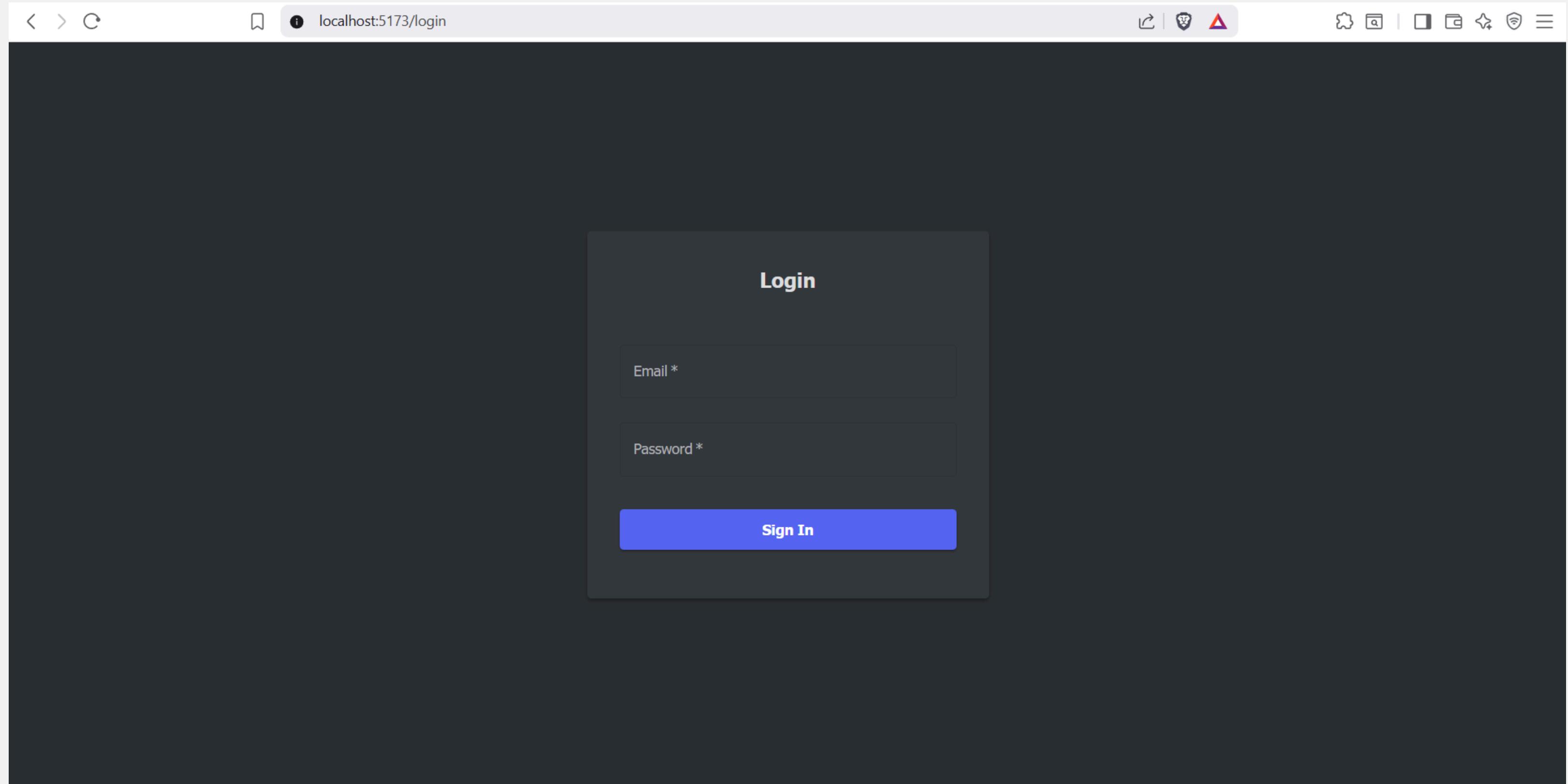
14     if (token && role) {
15       setIsAuthenticated(true);
16       setRole(role);
17       switch (role) {
18         case 'system_admin':
19           navigate('/homes/system-admin');
20           break;
21         case 'basetype_admin':
22           navigate('/homes/basetype-admin');
23           break;
24         case 'hr_admin':
25           navigate('/homes/hr-admin');
26           break;
27         case 'organization_admin':
28           navigate('/homes/organization-admin');
29           break;
30         case 'organization_user':
31           navigate('/homes/organization-user');
32           break;
33         case 'person_user':
34           navigate('/homes/person-user');
35           break;
36         default:
37           navigate('/login');
38       }
39     } else {
40       navigate('/login');
41     }
42   }
43 }
```
- Bottom Status Bar:** sitthipong\_tufA15 (3 weeks ago), Ln 19, Col 22, Spaces: 2, CRLF, {} TypeScript JSX, Go Live, Windsurf: {}, Prettier.

RootPage เป็นหน้าแรกที่ผู้ใช้พบ ทำหน้าที่ redirect ไปยังหน้าอื่นๆ ตามลำดับ



- Folder Page เก็บ layer ต่างๆ ของหน้าเว็บ
- Homes รวมหน้า Home เฉพาะสำหรับแต่ละ role
- Layer Account จัดการ CRUD สำหรับ account ในแอป
- Layer Communication จัดการ CRUD การสื่อสาร
- Layer History เก็บหน้าแสดงประวัติทั้งหมด
- Layer Information จัดการ CRUD สำหรับ organization และ person
- Layer Types จัดการ CRUD ข้อมูล base type
- Profile เก็บหน้า profile ของแต่ละ role

# ຕົວອຍ່າງໜ້າເວັບ



หน้า Login

The figure displays six browser windows showing different administrative and user dashboards for a system. Each window has a blue header bar with a globe icon and a sidebar containing 'Profile', 'Settings', 'About', 'Database', and 'Tutorial' links.

- System Admin Dashboard**: Manage all admin roles and organizational data. Includes icons for Manage User, Users Log, Persons Log, Organizations Log, and Communication Events Log.
- Basetype Admin Dashboard**: Manage data types such as gender, industry, and income ranges. Includes icons for Gender Type, Communication Event Purpose Type, Communication Event Status Type, Contact Mechanism Type, Country, Income Range, Industry Type, Marital Status Type, and Organization Type. A 'Racial Type' icon is also present.
- Organization Admin Dashboard**: Manage organization users and data. Includes icons for Organizations, Organizations Log, and Communication Events Log.
- HR Admin Dashboard**: Manage person-related data and classifications. Includes icons for Persons, Persons Log, and Communication Events Log.
- Organization User Dashboard**: Manage your organization's information and classifications. Includes a Communications icon.
- Person User Dashboard**: Manage your personal information and related data. Includes a Communications icon.

Dashboard สำหรับผู้ใช้งาน 6 บทบาท

localhost:5173/users

### All Users

Actions	ID	Username	Email	Role	Created At	Updated At
	1	sysadmin1	sysadmin1@gmail.com	system_admin	04/09/2025 15:02:00	N/A
	2	sysadmin2	sysadmin2@gmail.com	system_admin	04/09/2025 15:02:00	N/A
	3	baseadmin1	baseadmin1@gmail.com	basetype_admin	04/09/2025 15:02:00	N/A
	4	baseadmin2	baseadmin2@gmail.com	basetype_admin	04/09/2025 15:02:00	N/A
	5	hradmin1	hradmin1@gmail.com	hr_admin	04/09/2025 15:02:00	N/A
	6	hradmin2	hradmin2@gmail.com	hr_admin	04/09/2025 15:02:00	N/A
	7	orgadmin1	orgadmin1@gmail.com	organization_admin	04/09/2025 15:02:00	N/A
	8	orgadmin2	orgadmin2@gmail.com	organization_admin	04/09/2025 15:02:00	N/A

Rows per page: 10 ▾ 1–8 of 8 < >

System Admin จัดการ Admins

The screenshot shows a web application for managing persons. The URL in the address bar is `localhost:5173/persons`. The main title is "Persons" and the sub-page title is "All Persons". There is a back arrow on the left and a plus sign (+) on the right. The table has the following data:

Actions	ID	Username	Email	Personal ID	First Name	Middle Name	Last Name
	14	person1	person1@gmail.com	1234567890123	John	N/A	Doe
	15	person2	person2@gmail.com	2345678901234	Jane	Ann	Smith
	16	person3	person3@gmail.com	3456789012345	Mike	N/A	Brown
	17	person4	person4@gmail.com	4567890123456	Anna	N/A	Lee
	18	person5	person5@gmail.com	5678901234567	Somchai	N/A	Jaidee

At the bottom, there is a navigation bar with arrows, a "Rows per page: 10" dropdown, and a "1-5 of 5" indicator.

HR Admin จัดการ Persons

The screenshot shows a web browser window with the URL `localhost:5173/profiles/person-user`. The title bar is blue and displays "Person User Profile". The main content area has a dark gray background and contains a section titled "Profile". Inside this section, there is a list of user details:

- ID:** 14
- Username:** person1
- Email:** person1@gmail.com
- Role:**
- Personal ID Number:** 1234567890123
- First Name:** John
- Middle Name:** N/A
- Last Name:** Doe
- Nick Name:** JD
- Birth Date:** 15/05/2533
- Gender Type ID:** 1
- Marital Status Type ID:** 1
- Country ID:** 2
- Height:** 175 cm
- Weight:** 70 kg
- Racial Type ID:** 2
- Income Range ID:** 3
- About Me:** Software engineer with a passion for coding.
- Created At:** 04/09/2025 15:02:00
- Updated At:** N/A

At the bottom right of the profile box, there are two buttons: "Home" (blue) and "Logout" (red).

Person สามารถดู Profile ตัวเองได้

localhost:5173/communication-events

## Communication Events

### All Communication Events

Actions	ID	Title	Detail	From User ID	From User	To User ID	To User
	3	Order Confirmation	Order #1234 confirmed	10	10 - Global Trade Connecting the World (Organization)	14	14
	5	Feedback	Positive feedback on service	14	14 - John Doe Software engineer with a passion fo...	10	10
	1	Project Inquiry	Discussing new software project	9	9 - TechCorp Innovate the Future (Organization)	14	14
	2	Complaint	Issue with product delivery	14	14 - John Doe Software engineer with a passion fo...	15	15

Rows per page: 10 1-4 of 4

Person และ Organization จัดการเกี่ยวกับการสื่อสารได้

# พีเจอร์สำหรับ ผู้ใช้งานที่เป็น<sup>2</sup>

# Person และ Organization

< > ⌂

localhost:5173/communication-events/create

⟳ | 🛡️ 🔍

↶ Create Communication Event ↶

## Create New Communication Event

Title

Detail

To User

Contact Mechanism Type

Status

X

Save

Create Communication

Communication Events								
All Communication Events								
Actions	ID	Title	Detail	From User ID	From User	To User ID	To User	Timestamp
	3	Order Confirmation	Order #1234 confirmed	10	10 - Global Trade Connecting the World (Organization)	14	John Doe Software engineer with a passion for technology	14-09-2023 10:30:00
	5	Feedback	Positive feedback on service	14	14 - John Doe Software engineer with a passion for technology	10	Jane Smith Marketing specialist at TechCorp	14-09-2023 11:00:00
	1	Project Inquiry	Discussing new software project	9	9 - TechCorp Innovate the Future (Organization)	14	John Doe Software engineer with a passion for technology	14-09-2023 10:45:00
	2	Complaint	Issue with product delivery	14	14 - John Doe Software engineer with a passion for technology	15	Sarah Johnson Customer support representative	14-09-2023 11:15:00

Rows per page: 10 ▾ 1-4 of 4 < >

## Read Communication

The screenshot shows a web application window with a dark theme. At the top, the URL bar displays "localhost:5173/communication-events/3". The main content area is titled "Edit Communication Event". The form contains the following fields:

- ID: 3
- Title: Order Confirmation
- Detail: Order #1234 confirmed
- To User: 14 - John Doe Software engineer with a passion for coding. (Person)
- Contact Mechanism Type: Address
- Status: Completed

At the bottom of the form are three buttons: a red button with a white "X", a red button with a white trash icon, and a blue button with a white folder icon.

## Update and Delete Communication

# ปัญหาที่เจอ

- ออกแบบ Database ให้ต่อยอดง่ายและมีประสิทธิภาพมาก
- สร้าง Full Stack Cloud Native App ชั้บช้อน ต้องเรียนรู้แยกส่วน
- Data Model Resource Book เก่า อ่านยาก เข้าใจลำบาก
- JWT Authentication สำหรับ 6 บทบาทผู้ใช้จัดการยาก
- จัดการ hashed password ในระบบ login ชับช้อน
- ออกแบบ UX/UI ใช้งานง่ายสำหรับ Gen Z ท้าทาย
- จัดการข้อมูลวันเวลาอยุ่งยาก
- เก็บประวัติข้อมูลว่าใครทำอะไร เมื่อไหร่ เป็นเรื่องชับช้อน

# แนวทางต่อยอด

- พัฒนา Full Stack App ให้รองรับทุกภาษา
- ใช้ Reference Data Model อื่นๆ ที่ซับซ้อนและมีประสิทธิภาพมากกว่า
- UX/UI ตั้งค่าเปลี่ยน theme เว็บได้ตามต้องการ
- เพิ่ม AI/ML วิเคราะห์ Communication
- ขยายสู่ Mobile App (ใช้ React Native แบบ via bus)
- นำ App ที่ได้ไป Deploy ในระบบ Cloud Computing อื่นๆ

**Q & A**