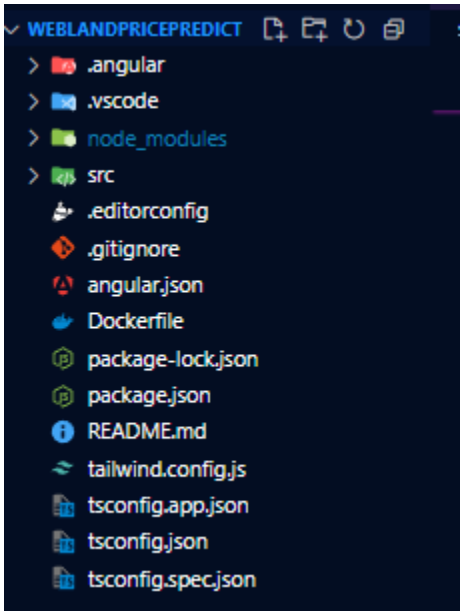
คู่มือสำหรับนักพัฒนา

1. ส่วนของแอพ Angular

ภายใน WEBLANDPRICEPREDICT จะประกอบไปด้วยไฟล์ที่จะใช้ในเว็บแอปพลิเคชัน โดยใช้ Angular สามารถโคลนได้จาก https://github.com/LPPACEGroup/WebLandPricePredict ซึ่งจะประกอบด้วยไฟล์และโฟลเดอร์ดังนี้ (ยกเว้นบางโฟลเดอร์เช่น .angular node_modules)



1.1. โฟลเดอร์ .angular

เป็นโฟลเดอร์ที่ใชเก็บข้อมูล Angular CLI เช่น cache หรือ config

1.2. โฟลเดอร์ .vscode

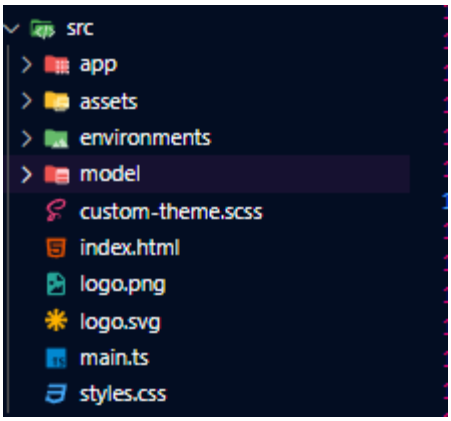
เป็นโฟลเดอร์ ที่เก็บไฟล์ setting สำหรับโปรเจ็คไม่ว่าจะเป็น การรัย คำสั่ง และ ฟอร์แมตโค้ด

1.3. โฟลเดอร์ node_modules

เก็บ dependencies ทั้งหมดที่ถูกติดตั้งจาก package.json

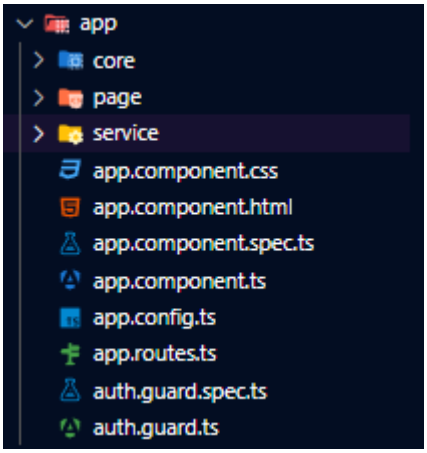
1.4. โฟลเดอร์ src

เป็นโฟล์เดอร์หลักสำหรับเก็บ code ของ แอปพลิเคชัน

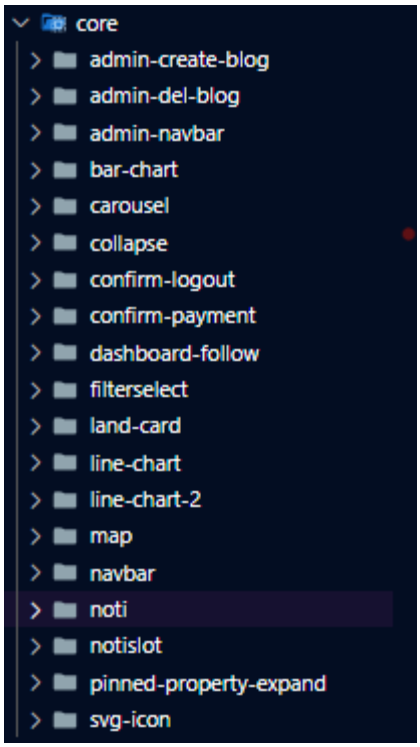


1.4.1. โฟลเดอร์ app

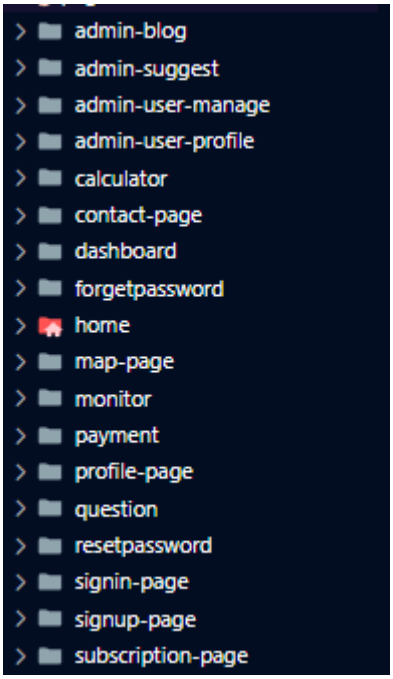
ทำหน้าที่เก็บ component หลักของ แอปพลิเคชัน



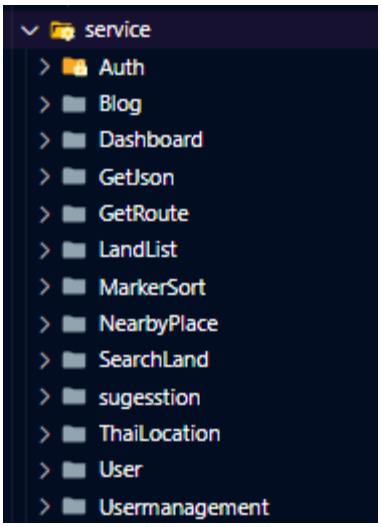
1). โฟลเดอร์ core เก็บ component ที่ใช้ในpage



2). โฟลเดอร์ page เก็บ component ของหน้าทั้งหมด



3). โฟลเดอร์ Service เก็บ service ที่ใช้ในการติดต่อกับ back end และ logic



4). app.components.x เป็นส่วนของไฟล์พื้นฐานของ component ใน angular .css สำหรับstyle .html สำรหับเป็น template .ts เป็น logic ของ component .spec.ts เป็นไฟล์สำหรับทดสอบ component ส่วนประกอบนี้จะมีทุกครั้งที่สร้าง component ขึ้นมา

5). app.config.ts ใช้ตั้งค่าแอปพลิเคชั่น

6).app.routes.ts ใช้กำหนด route

7). auth.guard.ts ใช้ protectและควบคุม route

8) auth.guard.spec.ts ใช้ทดสอบ auth.guard.ts

1.4.2. โฟลเดอร์ assets

เก็บ รูปภาพ icon ข้อมูล และ layer map ที่ใช้

1.4.3. โฟลเดอร์ environments

เก็บ environmental สำหรับ product และ development

1.4.4. โฟลเดอร์ model

เก็บโครงสร้างข้อมูลที่ใช้ไว้

1.4.5. ไฟล์ custom-theme.scss

ใช้สำหรับปรับ style Angular Material

1.4.6. ไฟล์ index.html

ไฟล์ html หลักของ เว็บ

1.4.7. ไฟล์ logo.png และ logo.svg

ไฟล์ logo ของเว็บ

1.4.8. ไฟล์ main.ts

ใช้สำหรับตั้งค่าเริ่มต้นของแอปพลิเคชัน

1.4.9. ไฟล์ styles.css

ไฟล์ style หลักของแอปพลิเคชัน ครอบคลุมทั้งโปรเจค

1.5. ไฟล์ .editorconfig

กำหนดรูปแบบการจัดโค้ด เช่น การเว้นวรรค tab หรือ encoding

1.6. ไฟล์ gitignore

ระบุไฟล์หรือ โฟลเดอร์ ที่ไม่ต้องการ track ด้วย git

1.7. ไฟล์ angular.json

Config ไฟล์ของ Angular สำหรับการ build และการพัฒนา

1.8. ไฟล์ Dockerfile

ไฟล์สำหรับสร้าง image ของ แอปพลิเคชัน

1.9. ไฟล์ package-lock.json

ระบุ version ที่แน่นอนของ dependencies

1.10. ไฟล์ package.json

รายชื่อ dependencies และ script ของ project

1.11. ไฟล์ README.md

ไฟล์สำหรับอธิบายโปรเจ็ค

1.12. ไฟล์ tailwind.config.js

ไฟล์ config สำหรบ Tailwind CSS

1.13. ไฟล์ tsconfig.app.json

ไฟล์สำหรับใช้ตั้งค่า TypeScript แอปพลิเคชัน

1.14. ไฟล์ tsconfig.json

ไฟล์หลักสำหรับใช้ตั้งค่า TypeScript

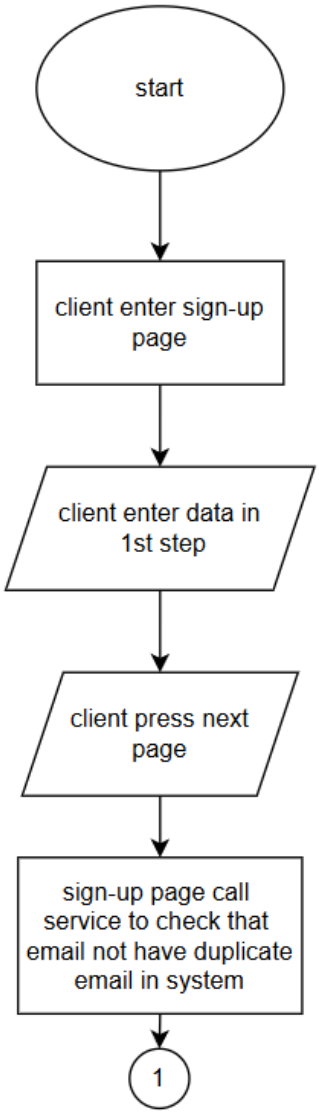
1.15. ไฟล์ tsconfig.spec.json

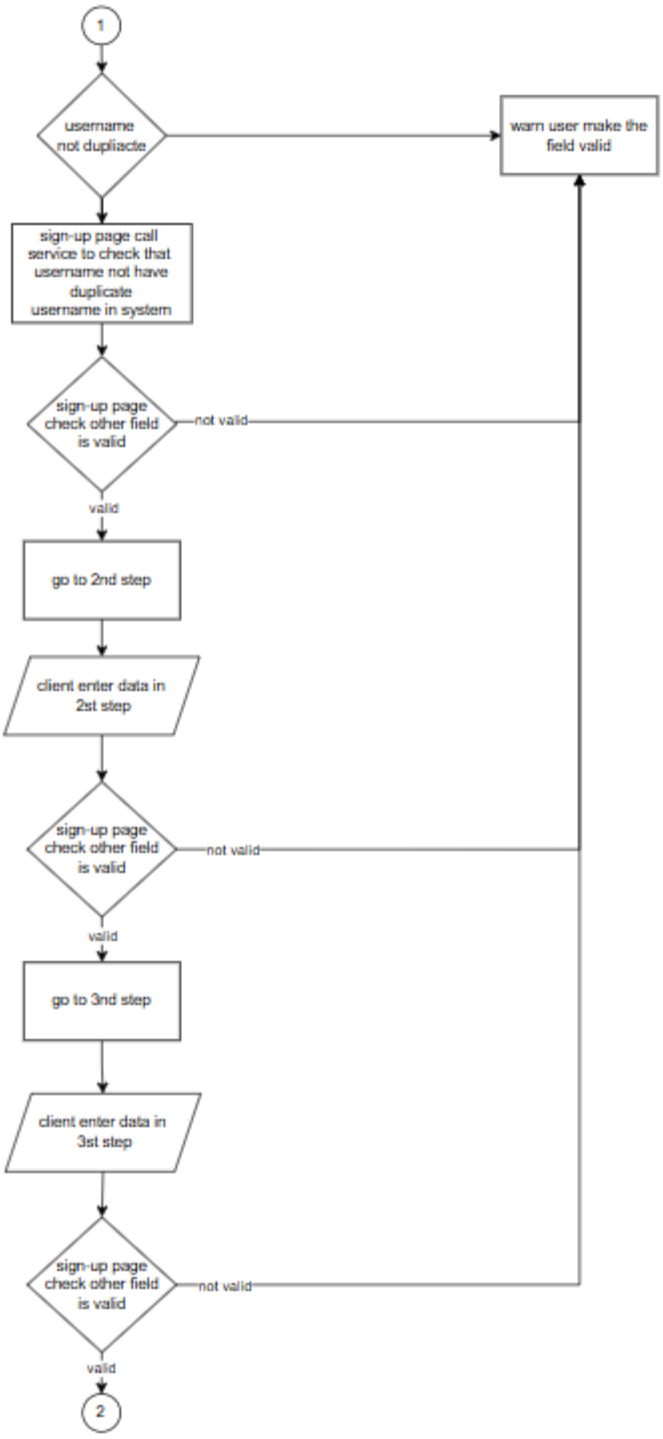
ไฟล์หลักสำหรับใช้ตั้งค่า TypeScript unit testing

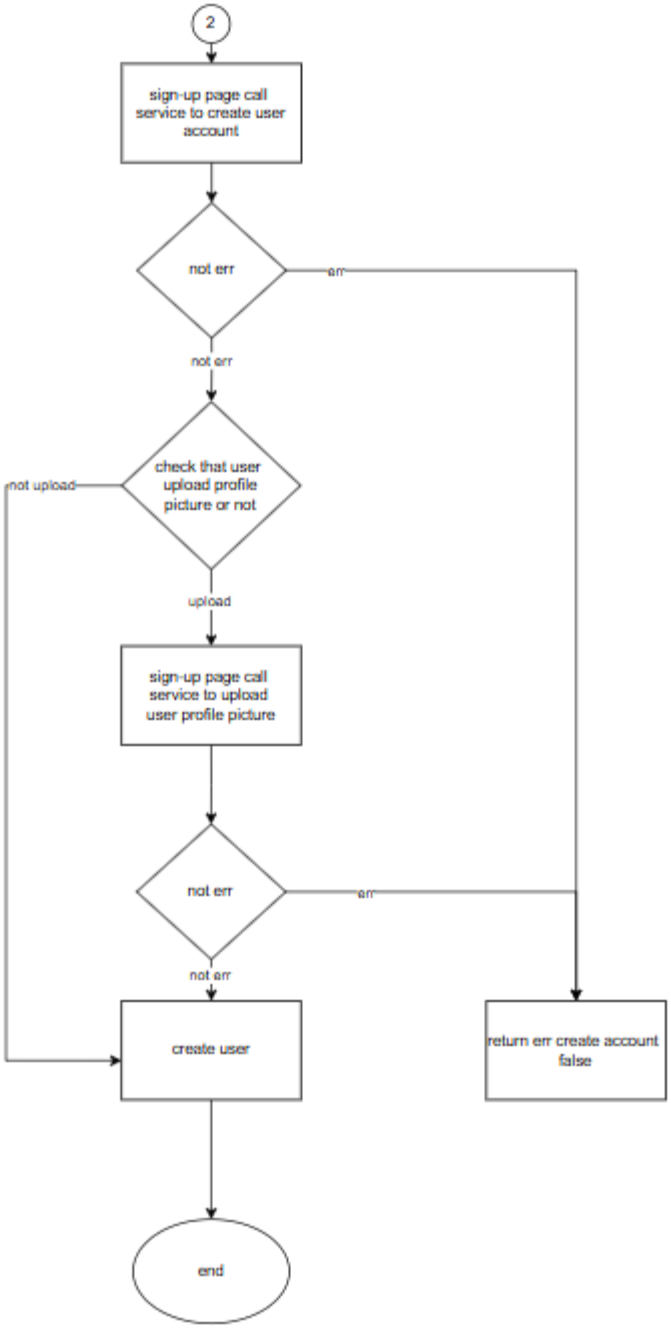
1.16. ส่วนของการทำงานของ page และ component

จะแสดงการทำงานของ page ทั้งหมดและ component ที่ไม่ได้ขึ้นกับ page

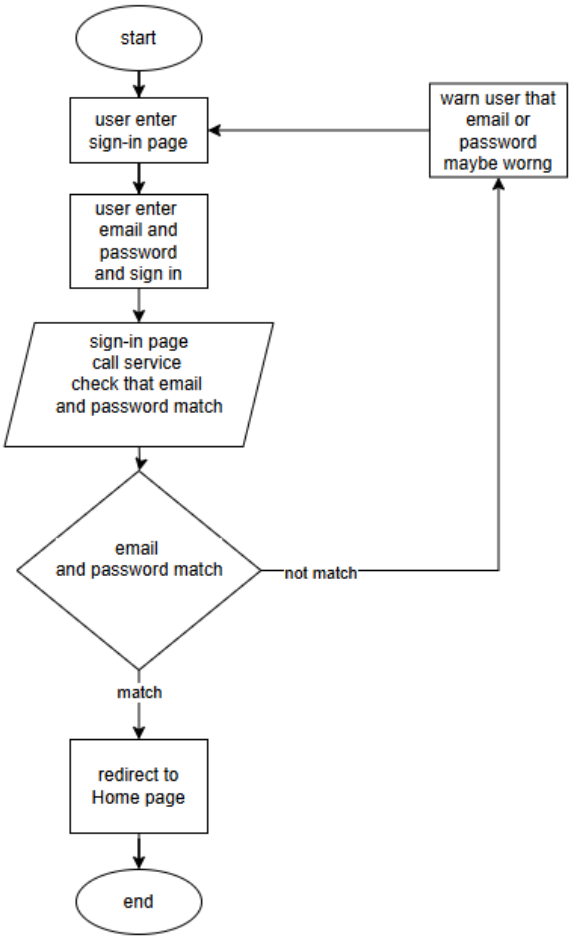
1). Sign Up Page: หน้า sign up user จะใส่ data ซึ่งจะต้องให้แต่ละ step สำเร็จจะไป step ต่อไปได้และจะทำการสร้างบัญชีเมื่อสำเร็จstep สุดท้าย


start
client enter sign-up page
client enter data in 1st step
client press next page
sign-up page call service to check that email not have duplicate email in system
1

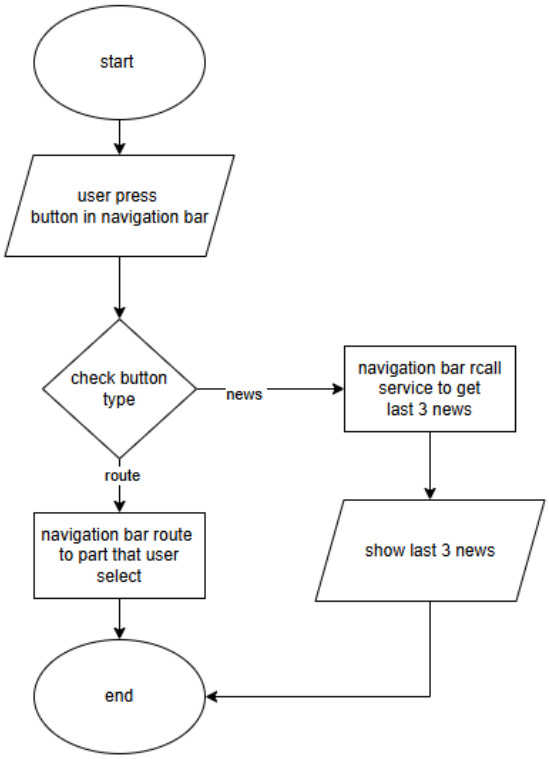

1
username not dupliacte
warn user make the field valid
sign-up page call service to check that username not have duplicate username in system
sign-up page check other field is valid
not valid
valid
go to 2nd step
client enter data in 2st step
sign-up page check other field is valid
not valid
valid
go to 3nd step
client enter data in 3st step
sign-up page check other field is valid
not valid
valid
2


2
sign-up page call service to create user account
not err
err
not err
check that user upload profile picture or not
not upload
upload
sign-up page call service to upload user profile picture
not err
err
not err
create user
return err create account false
end

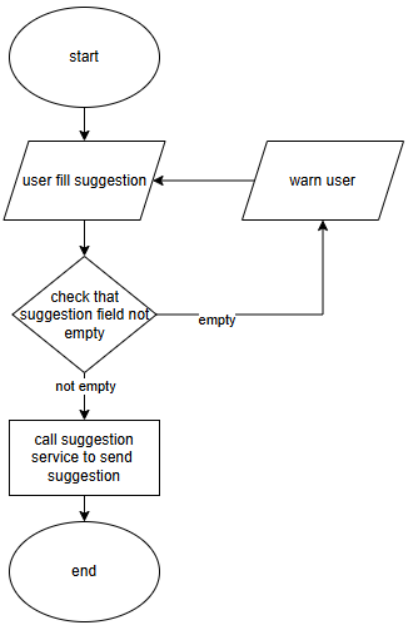
2). Sign In Page : user ทำการกรอก email password แล้วเรียก service ไปเช็คถ้าไม่match email หรือ password จะทำการเตือน



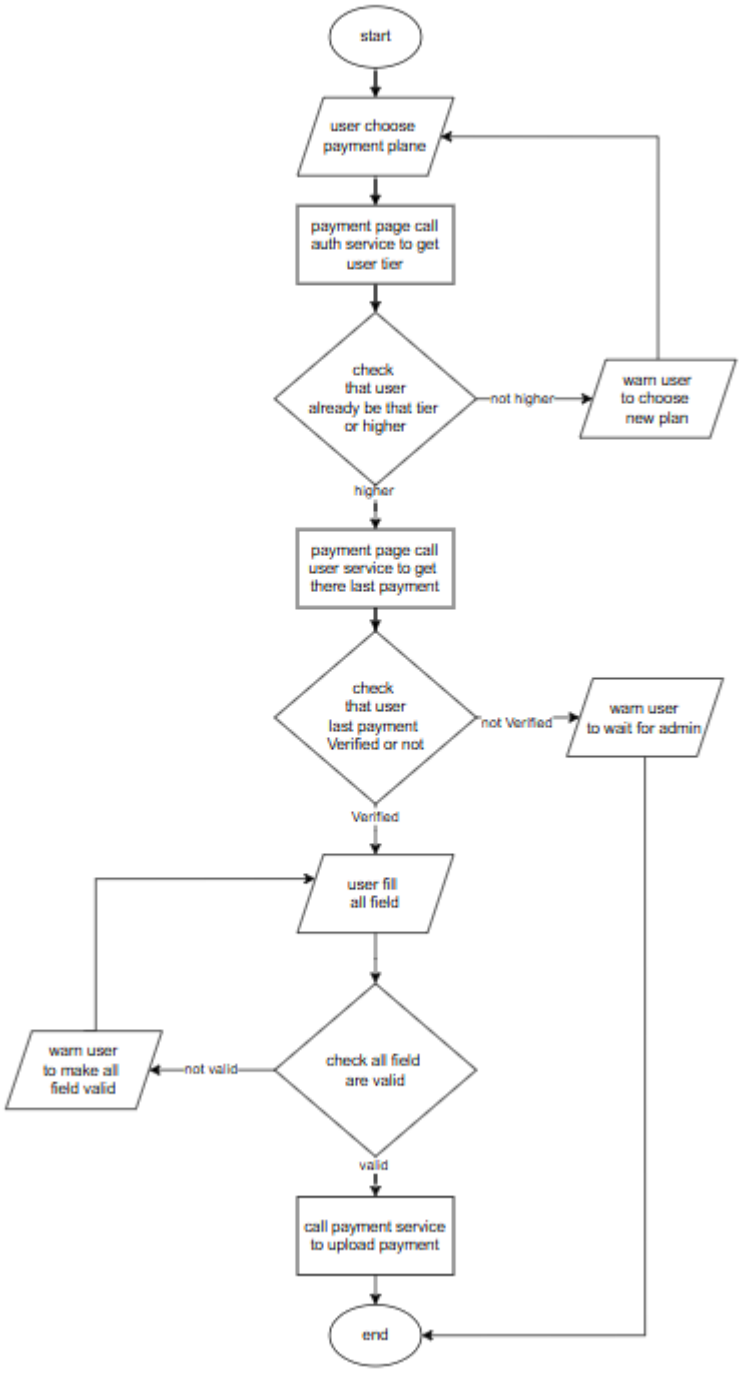
3). Navigation Bar Component: user กดที่page หรือ component ที่ต้องการจะไปถ้าเป็น pageจะพาไป route นั้น ถ้าเป็น component ถ้ากดที่รูปกระดิ่งจะโชว์ 3 ข่าวล่าสุดแทน



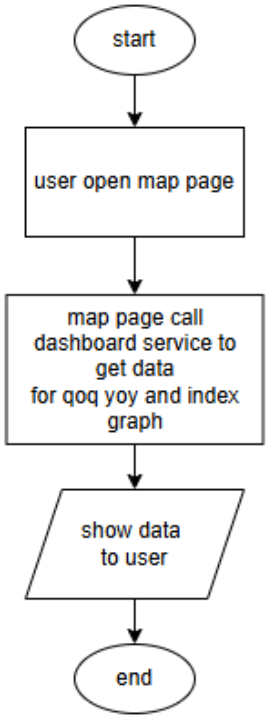
4). Home page ส่วน Suggestion Component: จะอยู่ในหน้า Home สามารกรอกและส่งข้อเสนอแนะให้ admin ได้



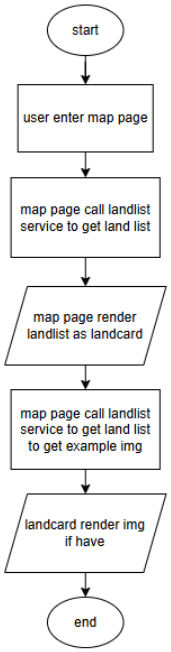
5).Payment Page : user เลือก plan แล้วเช็คว่า plan ที่เลือกสูงกว่า tier ที่เป็นอยู่แล้วไหมถ้าไม่เตือน user แล้วมาเช็คต่อว่า payment ล่าสุด admin verified หรือยังถ้าไม่ก็เตือน user หลังจาก user กรอกข้อมูลก็ดูว่า filed ทั้งหมดถูกต้องไหมถ้าไม่ก็เตือนให้ user แก้ไขถ้าถูกหมดจึงทำการ upload payment



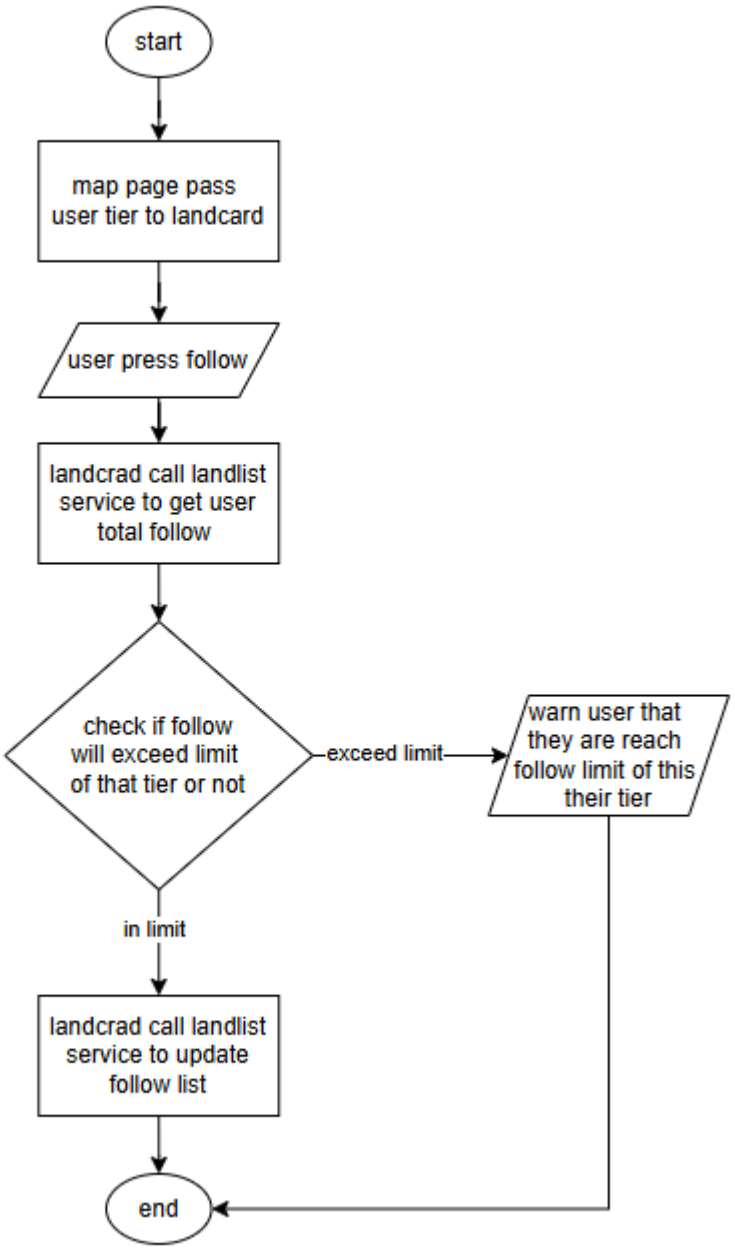
6).Map page ส่วนของ qoq yoy และ index : ทำการเรียก dashboard service เพื่อนำ goodsell data มาโชว์



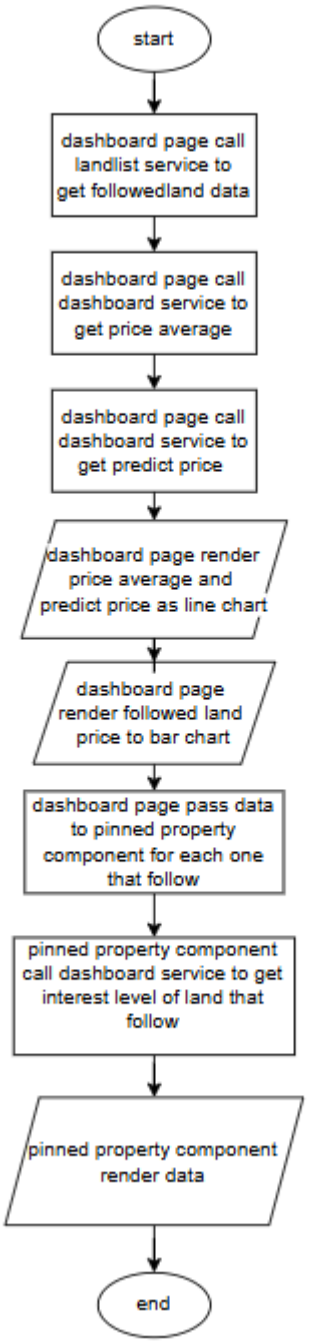
7). Map Page ส่วนของ landlist : ทำการเรียก landlist service เพื่อนำข้อมูลทั้งหมดมาแสดงใน landcard แล้ว landcard ก็เรียก landlist service เพื่อนำรูปมาแสดงต่อ



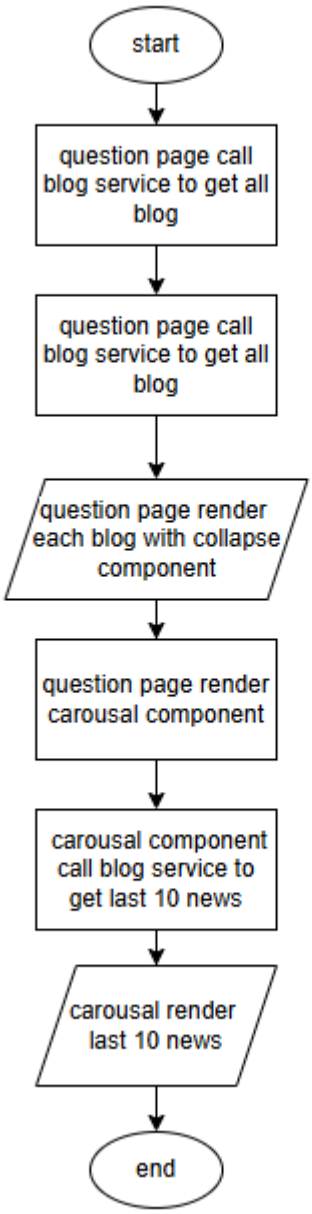
8). Land Card Componet ส่วนของการ follow : map page จะ pass tier ของ user มาเมื่อกด follow landcard จะเรียก landlist service เพื่อดู total follow ว่าเกินกับที่จะติดตามไหมถ้าไม่ก็ call landlist service อีกทีเพื่ออัพเดทการติดตาม ถ้าเกินก็จบการทำงาน



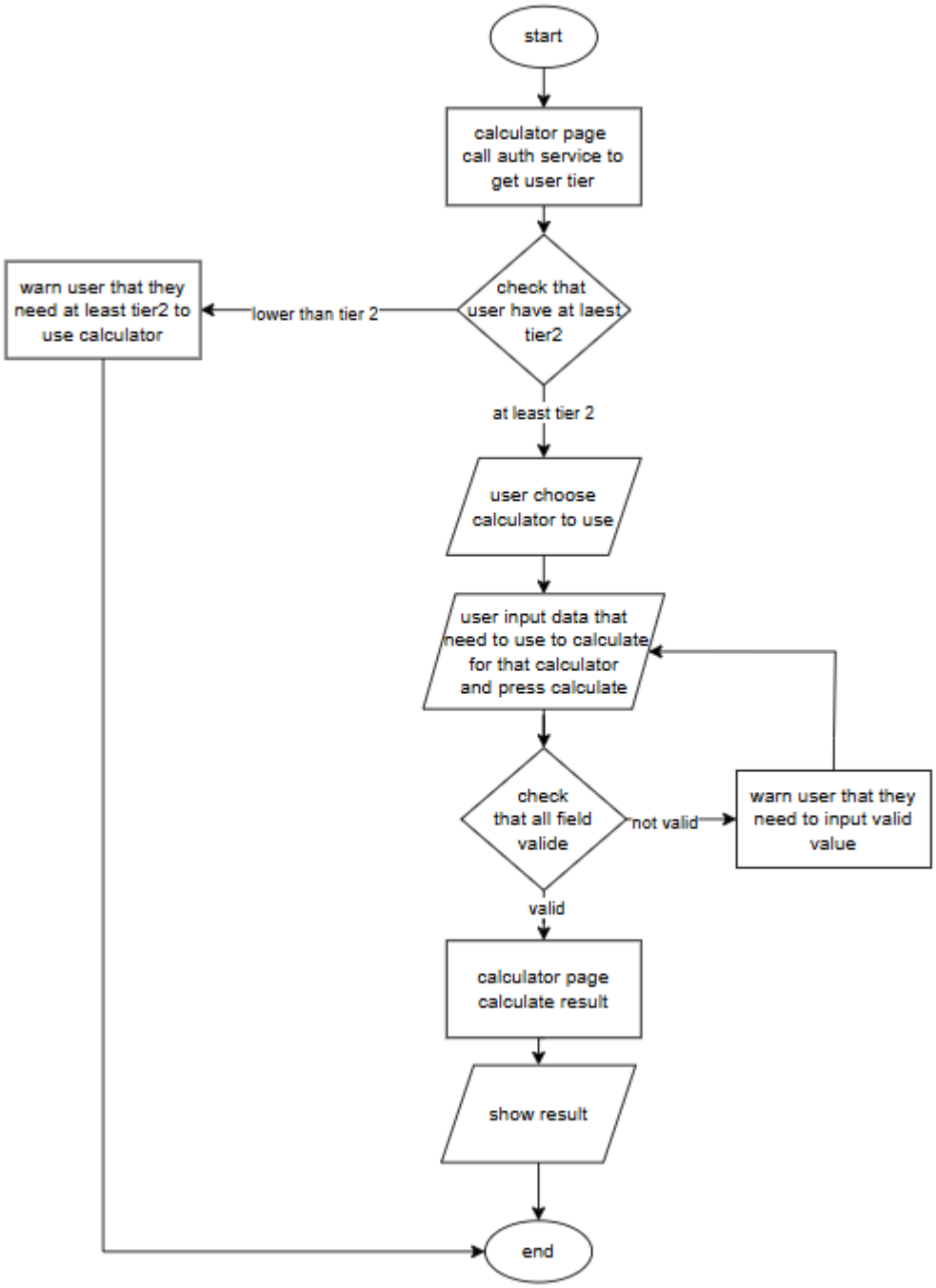
9). Dashboard Page : หน้า dashboard page จะเรียก landlist service เพื่อทำการนำราคา มาเปรียบเทียบกันด้วย bar chartแล้วจะส่งไปที่ pinned property component อีกทีเพื่อจะหาค่าความน่าสนใจโดยเรียก landlist service เพื่อหาราคาของพื้นที่ที่ติดตาม แล้วนำข้อมูลมา render ใน pinned property และในหน้า dashboard ยังมีการเรียก dashboard service เพื่อรับค่า price average และค่า predict price เพื่อนนำมาแสดงใน linechart



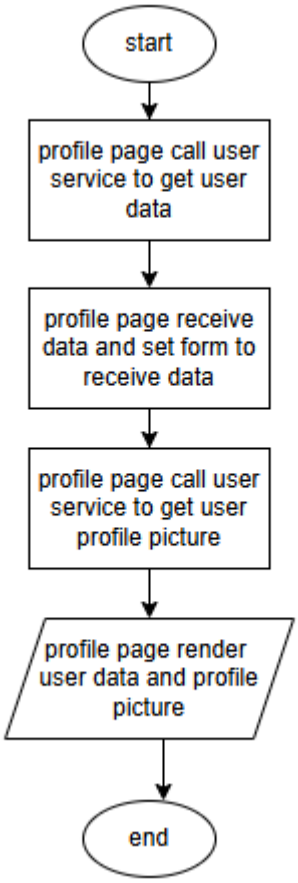
10). Question Page(Blog Page) : question page จะเรียก blog service เพื่อนำ data มา render และจะ render carousalซึ่ง carousal จะเรียก blog service เพื่อนำข่าวล่าสุด 10 ข่าวมานำเสนอใน carousal



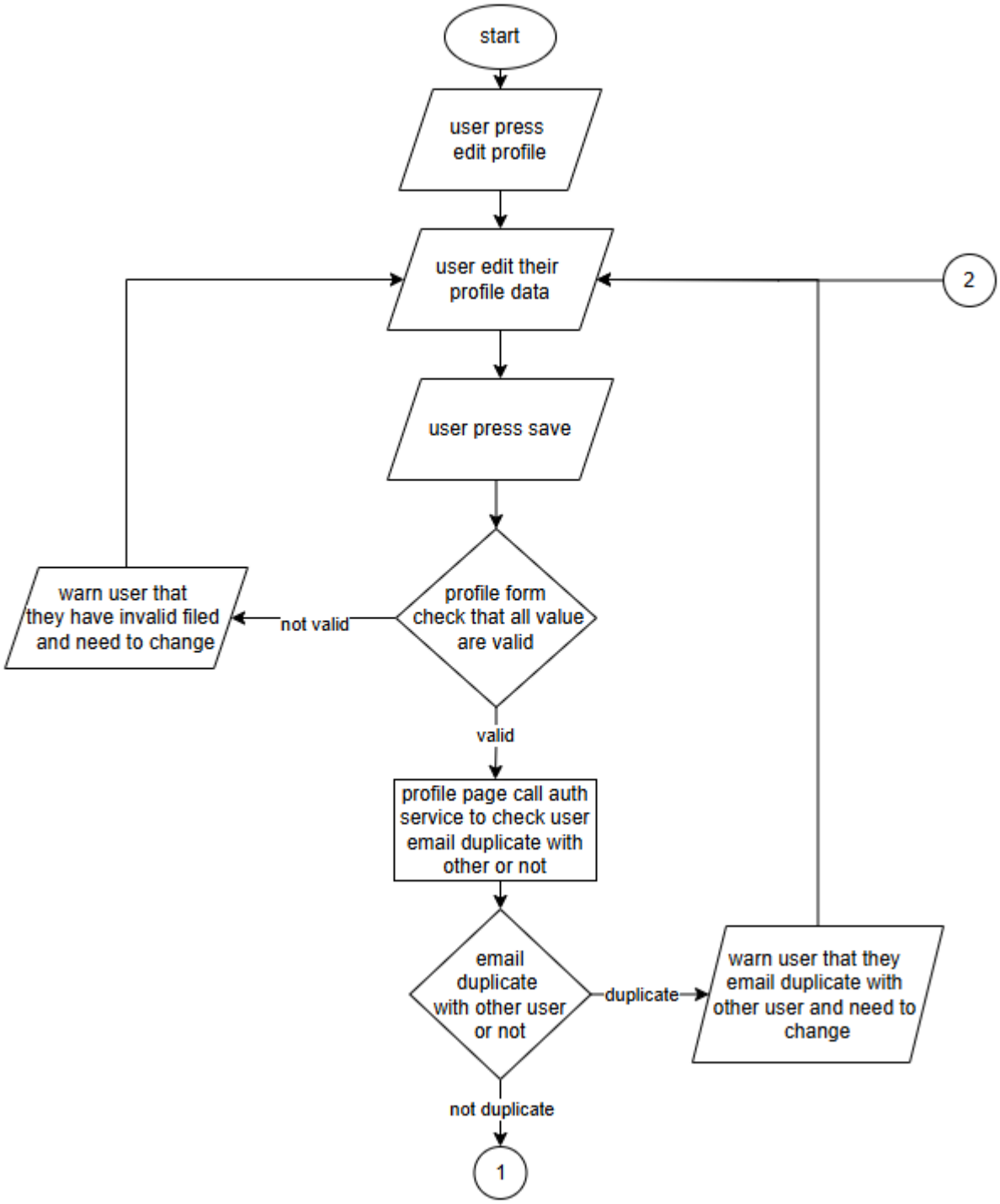
11).Calculator Page : calculator page เรียก auth service เพื่อเช็ค user tier แล้วจึงทำการให้ user เลือก calculator ที่อยากใช้จากนั้นกรอกข้อมูลเพื่อนำไปคำนวณ หลังจากกดคำนวณถ้ากรอกผิดจะเตือน userให้แก้ ถ้าถูกแล้วจึงทำการคำนวณผลและแสดงออกมา

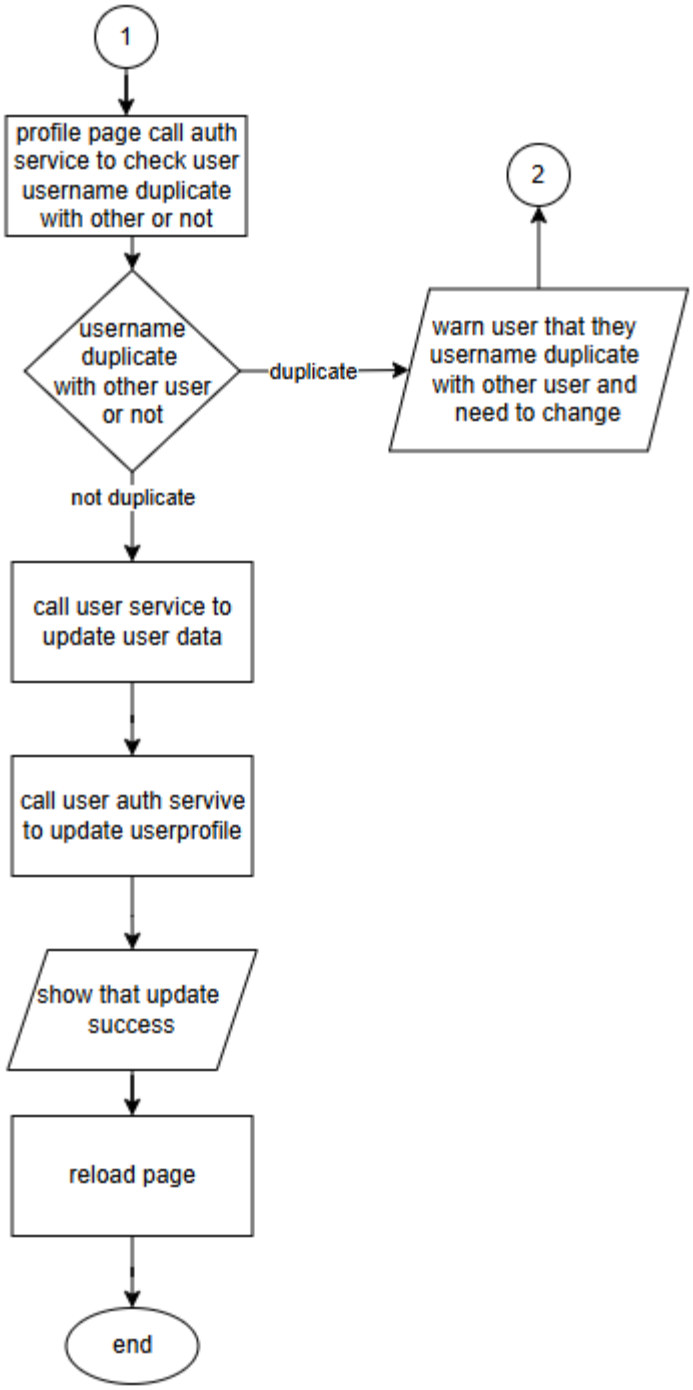


12).Profile page ส่วนของการเรียกดู :เรียก user service เพื่อรับ user data กับ user profile picture จากนั้นจึงแสดงออกมา

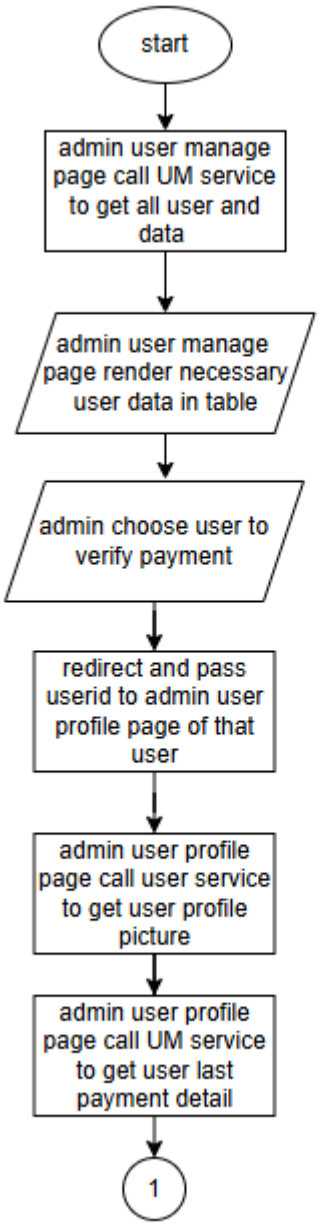


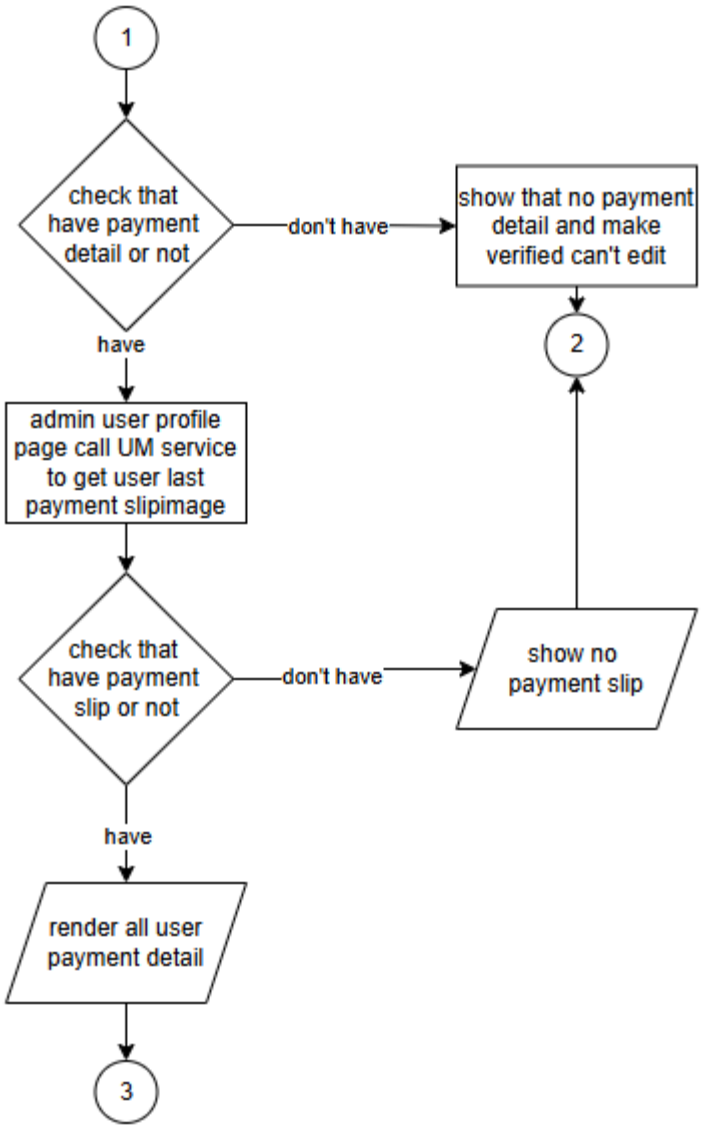
13).Profile page ส่วนของการแก้ไข Profile : เมื่อ user กดแก้ไข profile จะ enable form ให้ user กรอกแก้ไขข้อมูลหลังจากนั้นเมื่อกดบันทึกจะทำการเช็คว่า ถูกรูปแบบของ form ไหมถ้าไม่ให้แก้ จากนั้นจึงเรียก auth service เพื่อเช็ค email และ username ว่าซ้ำกับคนอื่นไหม ถ้าซ้ำก็เตือนให้แก้เมื่อสำเร็จจะบอกให้ userรู้แล้วทำการ reload หน้า

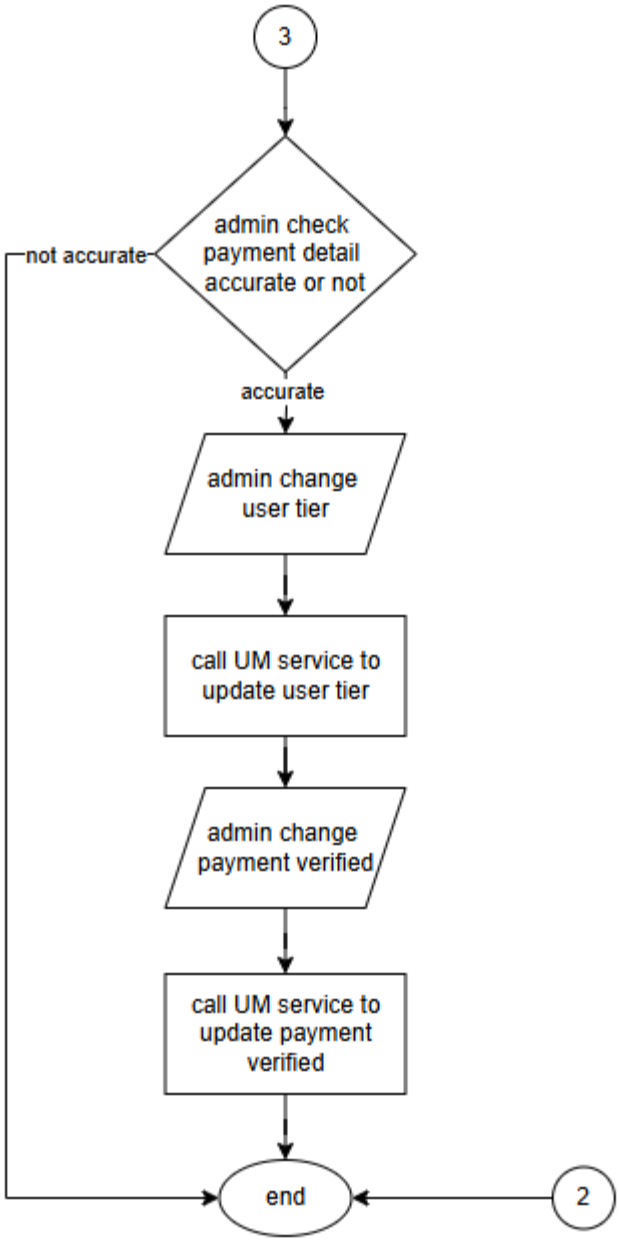

start
user press edit profile
user edit their profile data
2
user press save
profile form check that all value are valid
not valid
warn user that they have invalid filed and need to change
valid
profile page call auth service to check user email duplicate with other or not
email duplicate with other user or not
duplicate
warn user that they email duplicate with other user and need to change
not duplicate
1


1
profile page call auth service to check user username duplicate with other or not
username duplicate with other user or not
duplicate
warn user that they username duplicate with other user and need to change
2
not duplicate
call user service to update user data
call user auth servive to update userprofile
show that update success
reload page
end

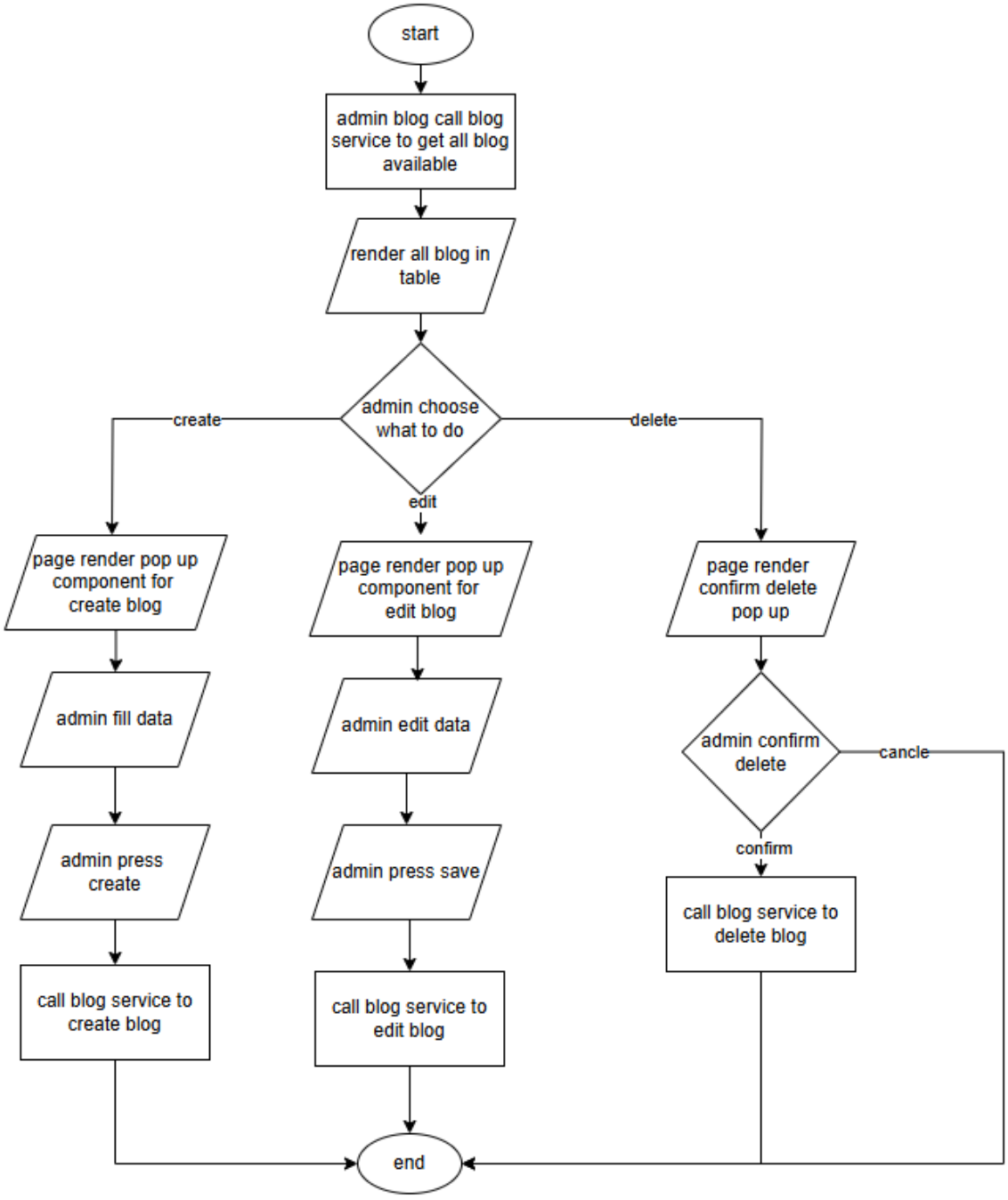
14). Admin User manage Page : ทำการเรียก UM service เพื่อรับ data ของทุก user แล้วนำข้อมูลที่จำเป็นมาแสดงในตารางสำหรับข้อมูลเต็มต้องกดเข้าไปดูเพื่อยืนยันการชำระเงิน แล้วนำ userid ที่ถูก pass มาจากตารางนำมาเรียก UM service เพื่อรับ payment detail ดับ slip หากไม่มีก็จะจบการดำเนินการเพียงเท่านี้ หาก admin ก็จะทำการเช็ค slip ว่าตรงไหม แล้วจึงทำการเปลี่ยน tier และ verified payment แล้วเรียก UM service มา อัพเดท



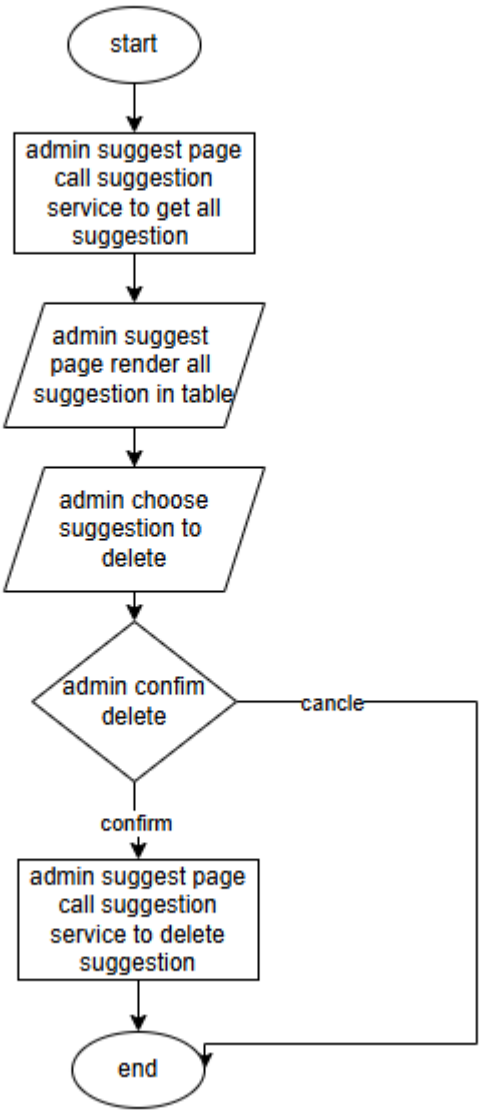

1
check that have payment detail or not
don't have
show that no payment detail and make verified can't edit
2
have
admin user profile page call UM service to get user last payment slipimage
check that have payment slip or not
don't have
show no payment slip
have
render all user payment detail
3


3
admin check payment detail accurate or not
not accurate
accurate
admin change user tier
call UM service to update user tier
admin change payment verified
call UM service to update payment verified
end
2

15). Admin Blog Page ส่วนการสร้าง Blog: หน้านี้เมื่อเข้ามาจะเรียก blog service โดย admin จะสามารถสร้าง แก้ไข และ ลบ blog ได้ผ่าน blog service

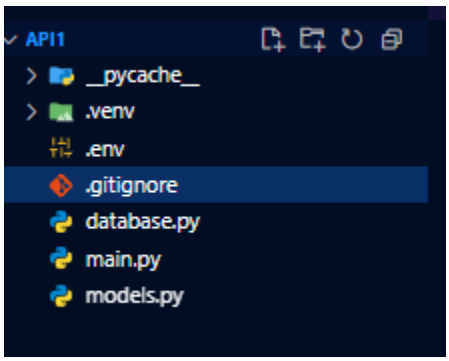


16).Admin Suggest Page : เมื่อเข้ามาจะเรียก Suggestion service เพื่อดูข้อเสนอแนะทั้งหมด โดยadmin สารถเลือกลบข้อเสนอแนะได้โดยจะลบผ่าน suggestion service



2. ส่วนหลังบ้านของ fastapi

ในโฟลเดอร์API1 จะประกอบด้วยไฟล์ที่ใช้ในส่วนของหลังบ้านสามารถโคลนได้จาก https://github.com/LPPACEGroup/Api1 ซึ่งจะประกอบด้วยไฟล์และโฟลเดอร์ดังนี้ (ยกเว้นบางโฟลเดอร์ เช่น __pycache__)



2.1. โฟลเดอร์ .venv

โฟลเดอร์ใช้เก็บ dependencies ของ python แยกจากระบบหลัก

2.2. ไฟล์ .env

เก็บ environment variables ที่สำคัญ

2.3. ไฟล์ .gitignore

ระบุไฟล์หรือ โฟลเดอร์ ที่ไม่ต้องการ track ด้วย git

2.4. ไฟล์ database.py

จัดการเชื่อมต่อกับฐานข้อมูล

2.5. ไฟล์ models.py

ไฟล์ที่เก็บโครงสร้างของฐานข้อมูล

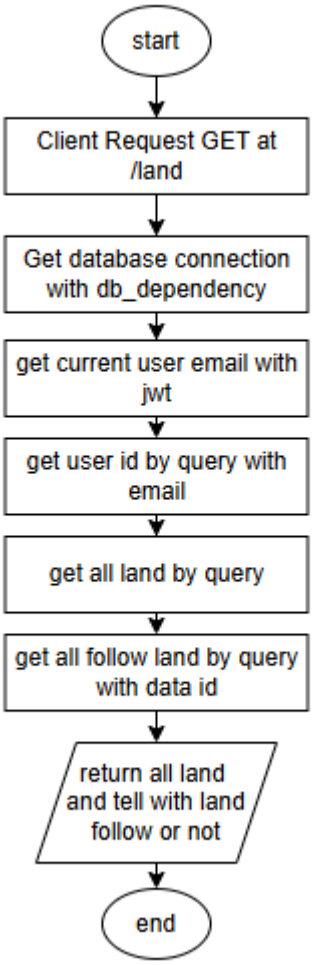
2.6. ไฟล์ main.py

ไฟล์ที่รวม api ไว้

โดยจะมี route ดังนี้

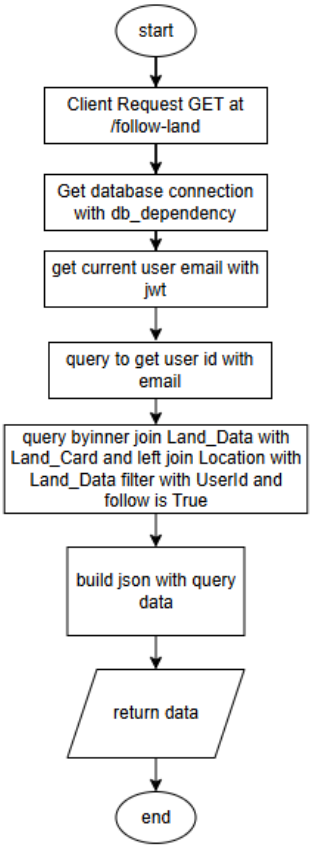
2.6.1. land

GET /land เรียก land ทั้งหมดพร้อมสถานะการติดตามของ user

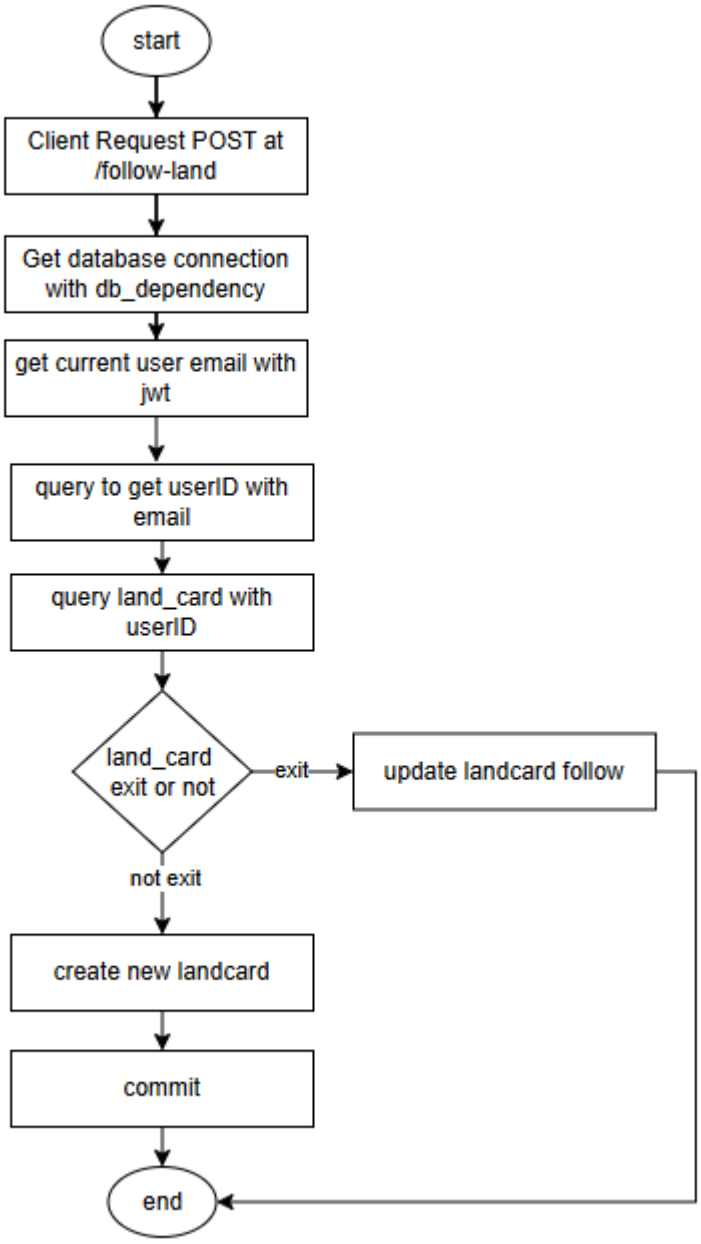


2.6.2. follow-land

GET /follow-landเรียกข้อมูลของที่ดินที่ user ติดตามไว้มั้งหมด

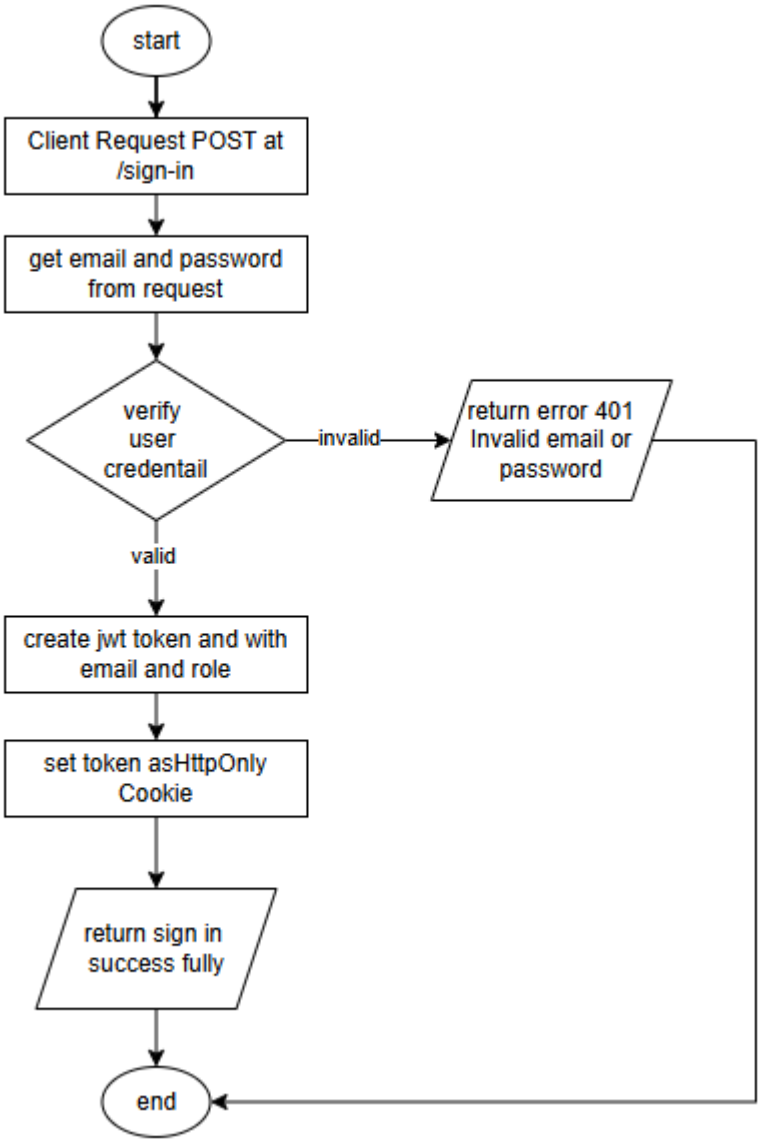


POST /follow-land เมื่อ userกด follow land จาก front end จะทำการเช็คว่ามีข้อมูลใน Land_Card ไหม ถ้ามีก็อัพเดทสถานะ follow ถ้าไม่มีก็สร้างขึ้นใหม่


start
Client Request POST at /follow-land
Get database connection with db_dependency
get current user email with jwt
query to get userID with email
query land_card with userID
land_card exit or not
exit
update landcard follow
not exit
create new landcard
commit
end

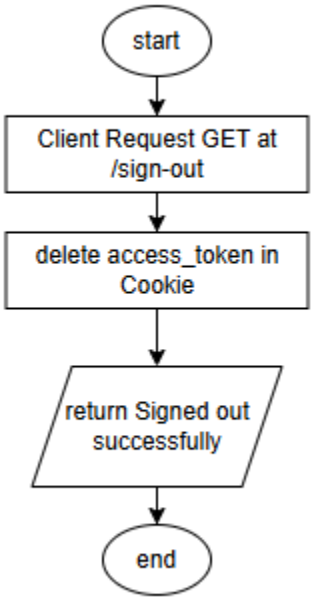
2.6.3. auth/sign-in

POST /sing-in ใช้ในการเข้าสู่ระบบ



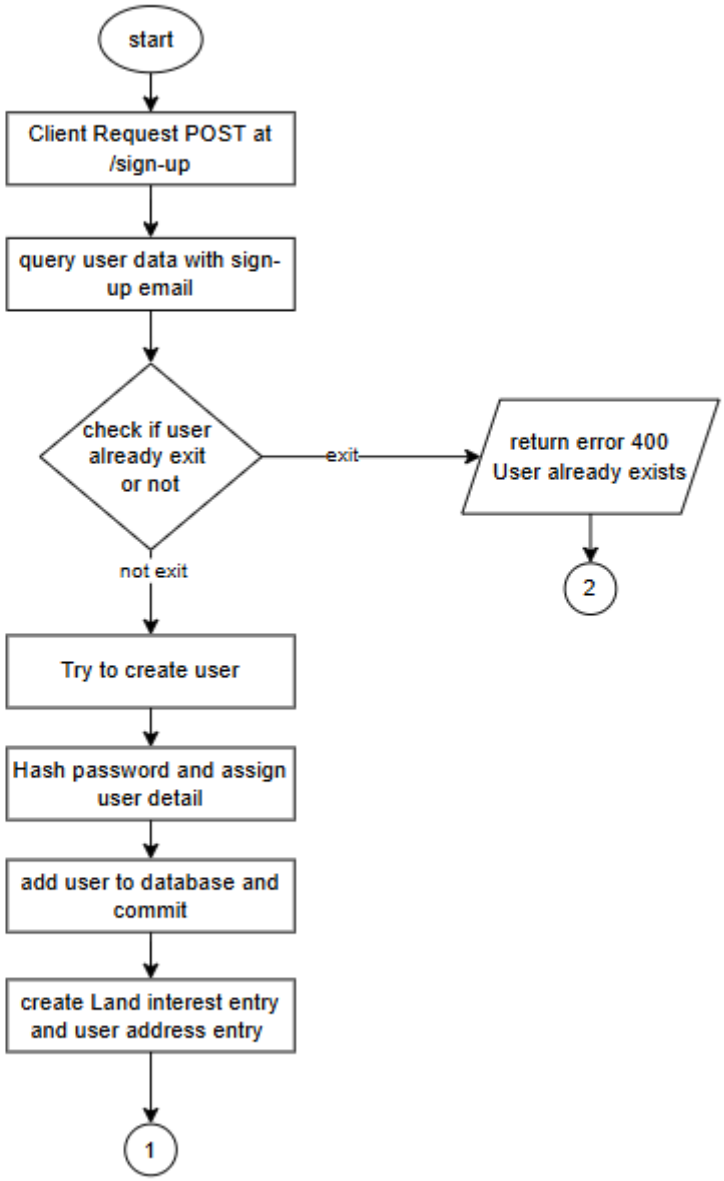
2.6.4. auth/sign-out

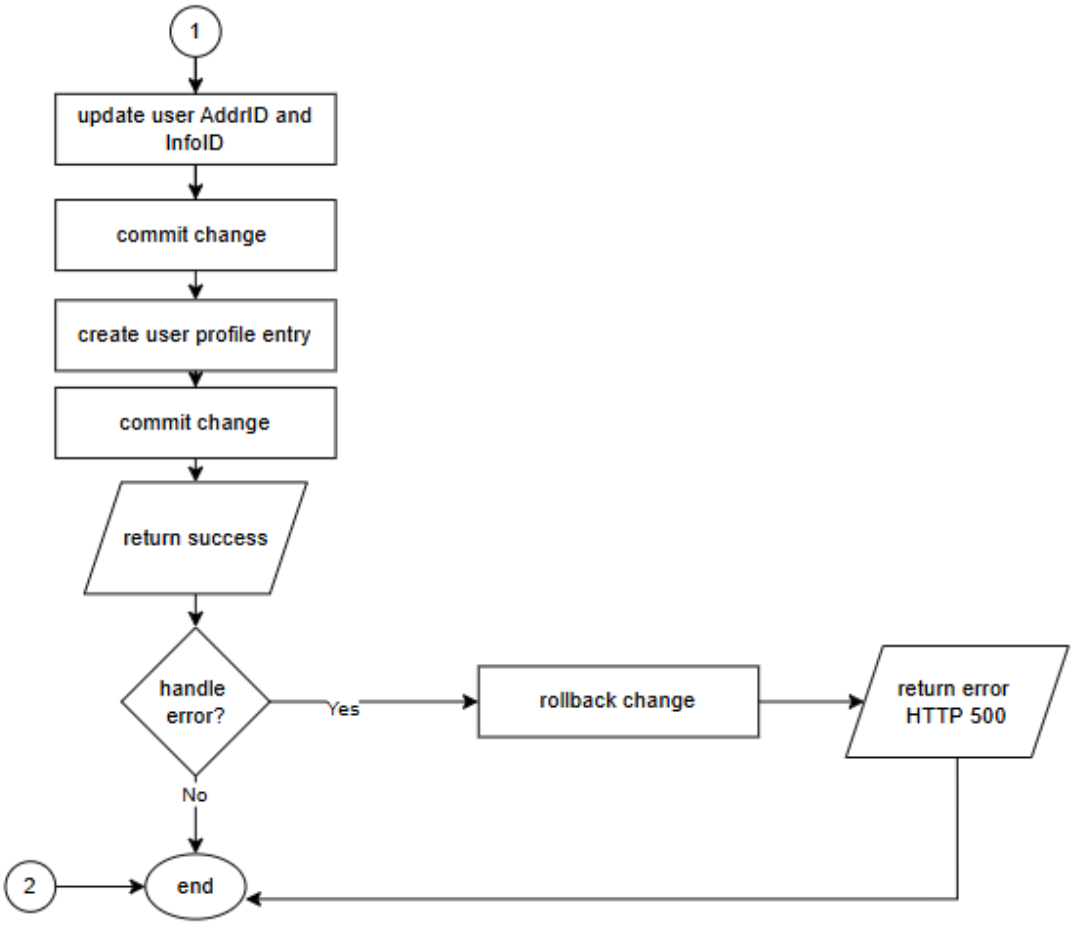
GET /sign-out ใช้ในการ ออกจากระบบของ user



2.6.5. auth/sign-up

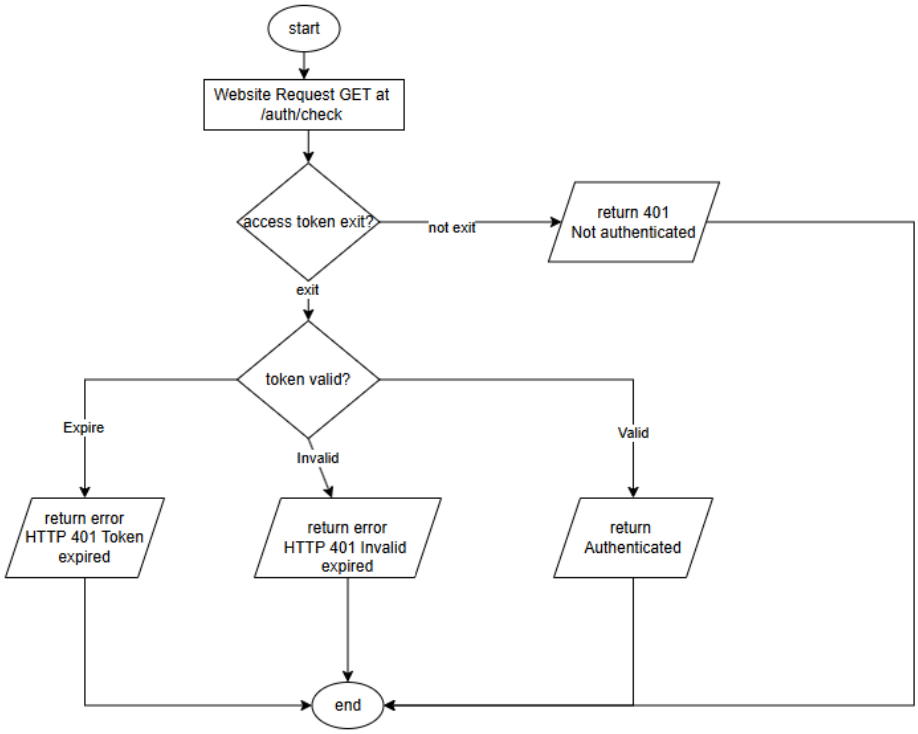
GET /sign-up ใช้สมัครสมาชิกจะเช็คว่ามี user ที่จะสมัครอยู่แล้วไหมถ้ามีอยู่แล้วจะทำการ return error ถ้าไม่มีจะดำเนินการต่อทำการนำข้อมูลที่ได้จาก request form commit ขึ้น database ตอนสุดท้านจะมี handle error ถ้ามีข้อผิดพลาดจะทำการ rollback


start
Client Request POST at /sign-up
query user data with sign-up email
check if user already exit or not
exit
return error 400 User already exists
2
not exit
Try to create user
Hash password and assign user detail
add user to database and commit
create Land interest entry and user address entry
1


1
update user AddrID and InfoID
commit change
create user profile entry
commit change
return success
handle error?
Yes
rollback change
return error HTTP 500
No
2
end

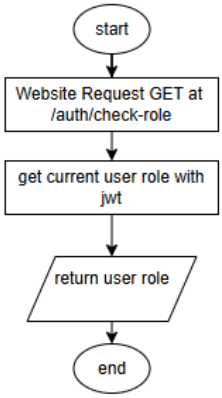
2.6.6. auth/check

GET /auth/check ทำการเช็คว่า user authenticated ไหม โดยเช็คว่า มี token ไหมแล้ว token ที่มีถูกต้องไหม



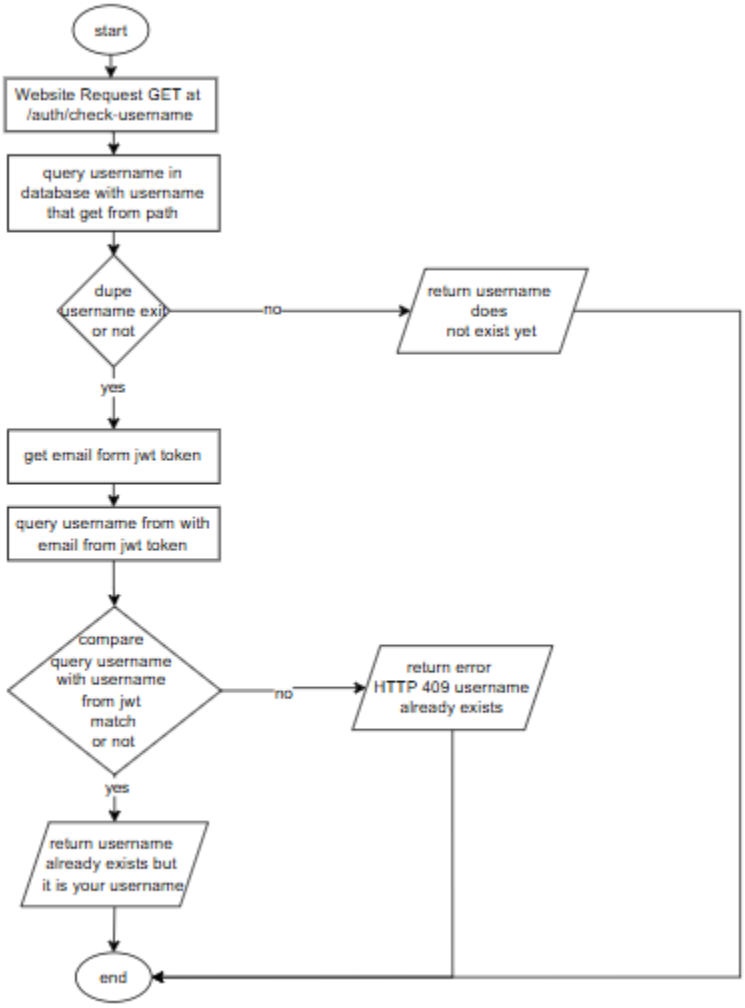
2.6.7. auth/check-role

GET / auth/check-role รับ user role มาจาก jwt token



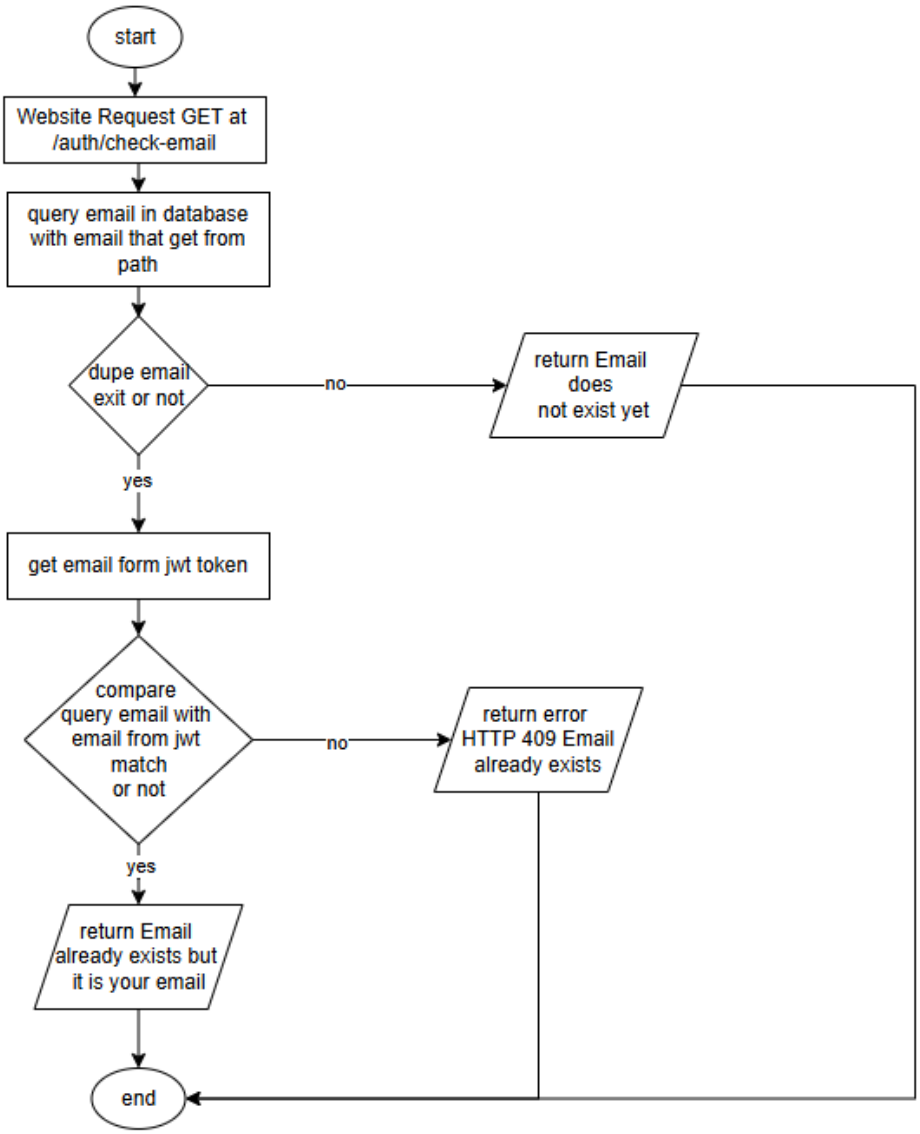
2.6.8. auth/check-username

GET /auth/check-username เช็คว่า username ที่ส่งมาอยู่ในระบบอยู่แล้วไหม แล้วที่ส่งมาใช่ของ user เองไหม ถ้าไม่ก็แปลว่าซ้ำและ return error



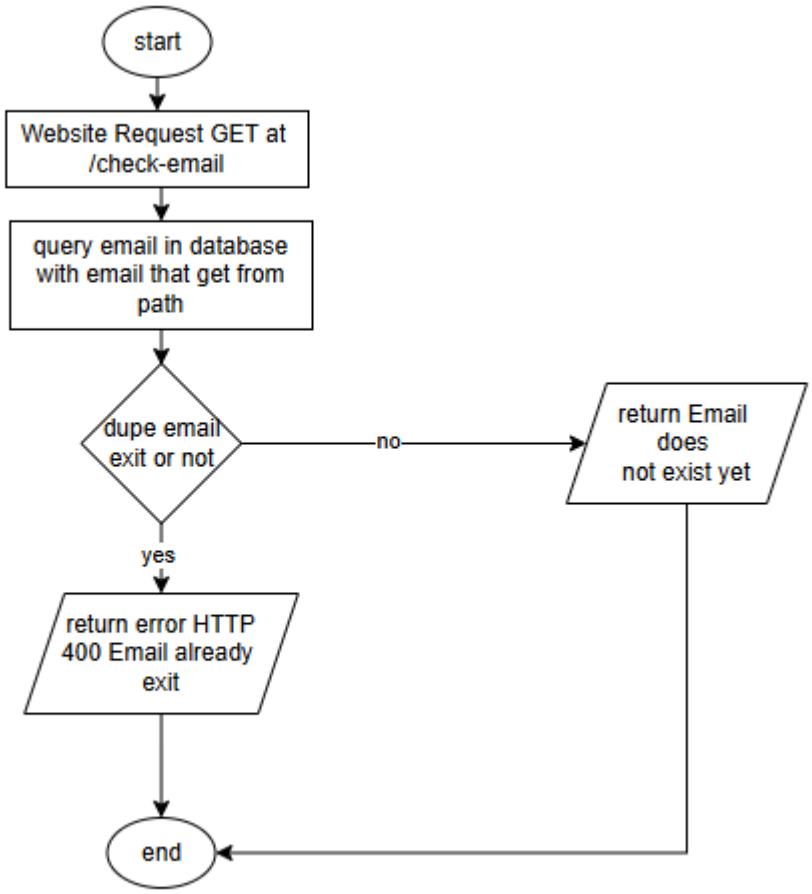
2.6.9. auth/check-email

GET /auth/check-email เช็คว่า email ที่ส่งมาอยู่ในระบบอยู่แล้วไหม แล้วที่ส่งมาใช่ของ user เองไหม ถ้าไม่ก็แปลว่าซ้ำและ return error



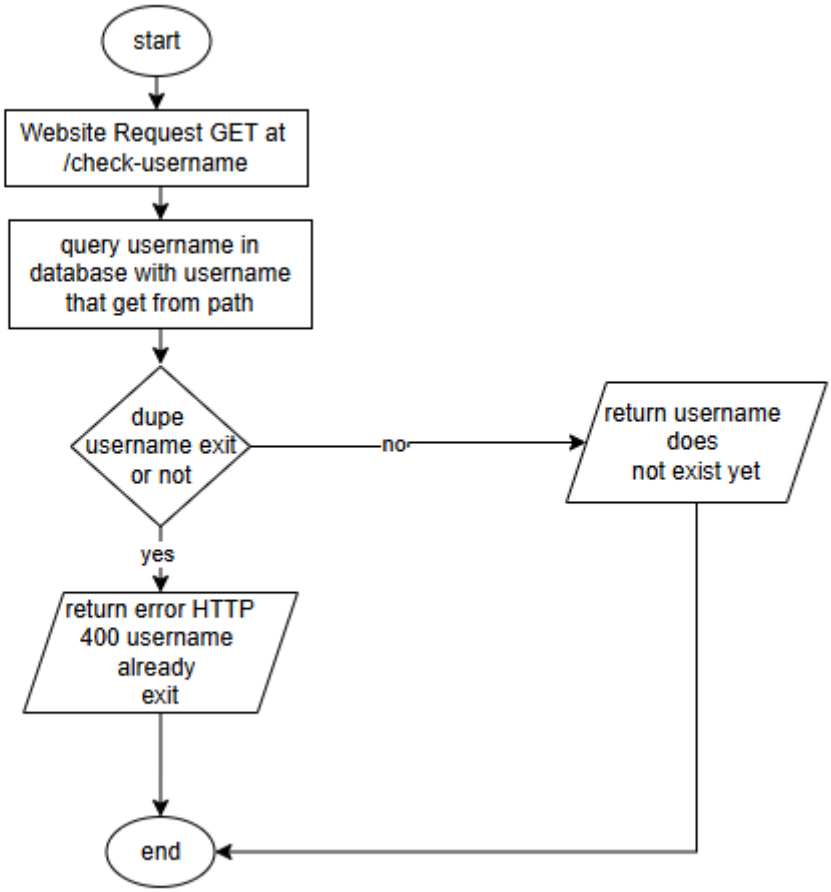
2.6.10. check-email

GET /check-email เช็คว่า email ที่ส่งมาอยู่ในระบบอยู่แล้วไหมถ้ามีอยู่แล้วจะ return error



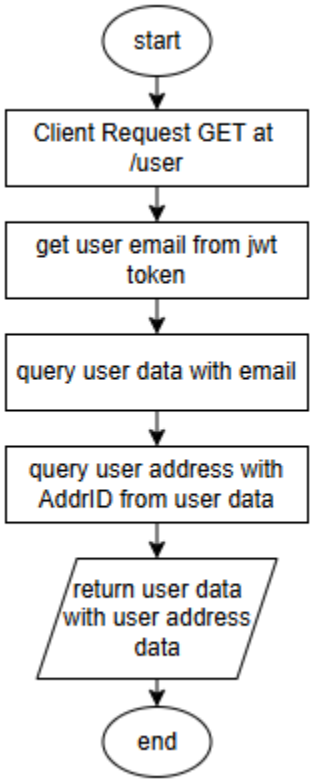
2.6.11. check-username

GET /check- username เช็คว่า username ที่ส่งมาอยู่ในระบบอยู่แล้วไหมถ้ามีอยู่แล้วจะ return error

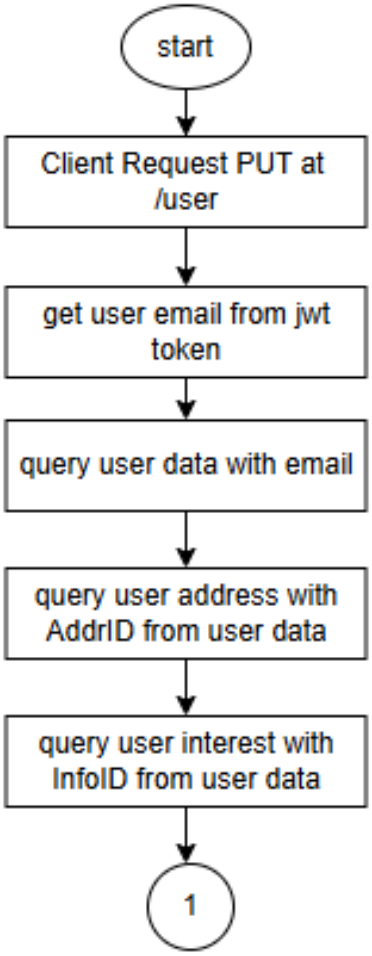


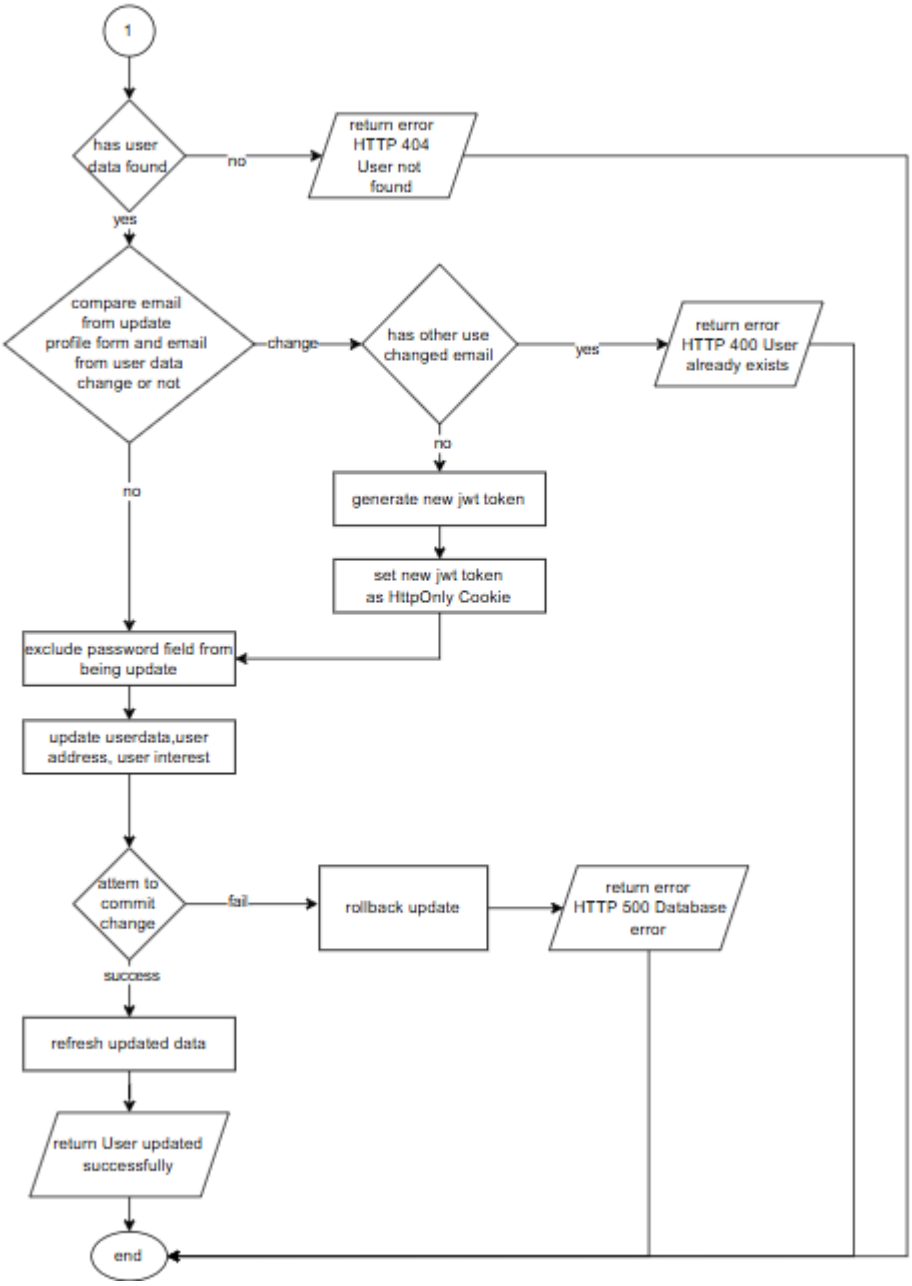
2.6.12. user

GET /user หา email ของ user จาก jwt token แล้วจึงนำมาเรียก user data จาก database แล้วนำ AddrID จาก userdata มาหา user address ต่อ



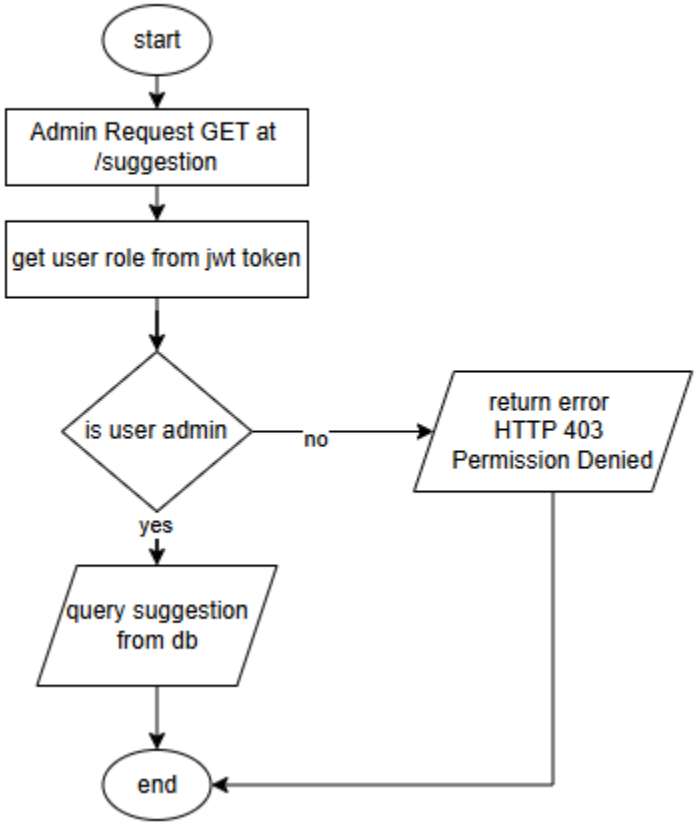
PUT /user ทำการรับข้อมูลที่ userอัพเดทมาแล้วดูว่า email เปลี่ยนไปไหมถ้าเปลี่ยนแล้วไม่ซ้ำก็สร้าง jwt token ใหม่ แล้วจึงทำการ อัพเดท data ถ้ามี error ก็ rollbackการอัพเดทถ้าไม่มีก็อัพเดทหลังจากนั้นก็ refresh


start
Client Request PUT at /user
get user email from jwt token
query user data with email
query user address with AddrID from user data
query user interest with InfoID from user data
1

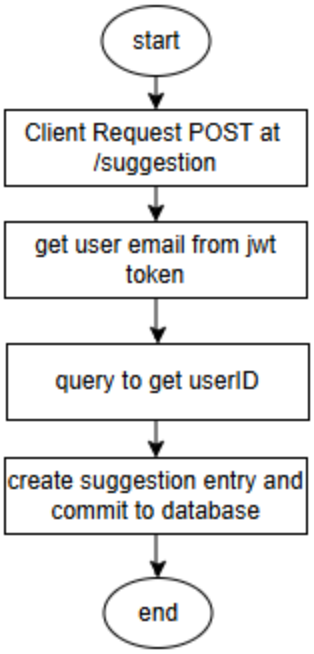

1
has user data found
no
return error HTTP 404 User not found
yes
compare email from update profile form and email from user data change or not
change
has other use changed email
yes
return error HTTP 400 User already exists
no
generate new jwt token
set new jwt token as HttpOnly Cookie
no
exclude password field from being update
update userdata,user address, user interest
attem to commit change
fail
rollback update
return error HTTP 500 Database error
success
refresh updated data
return User updated successfully
end

2.6.13. suggestion

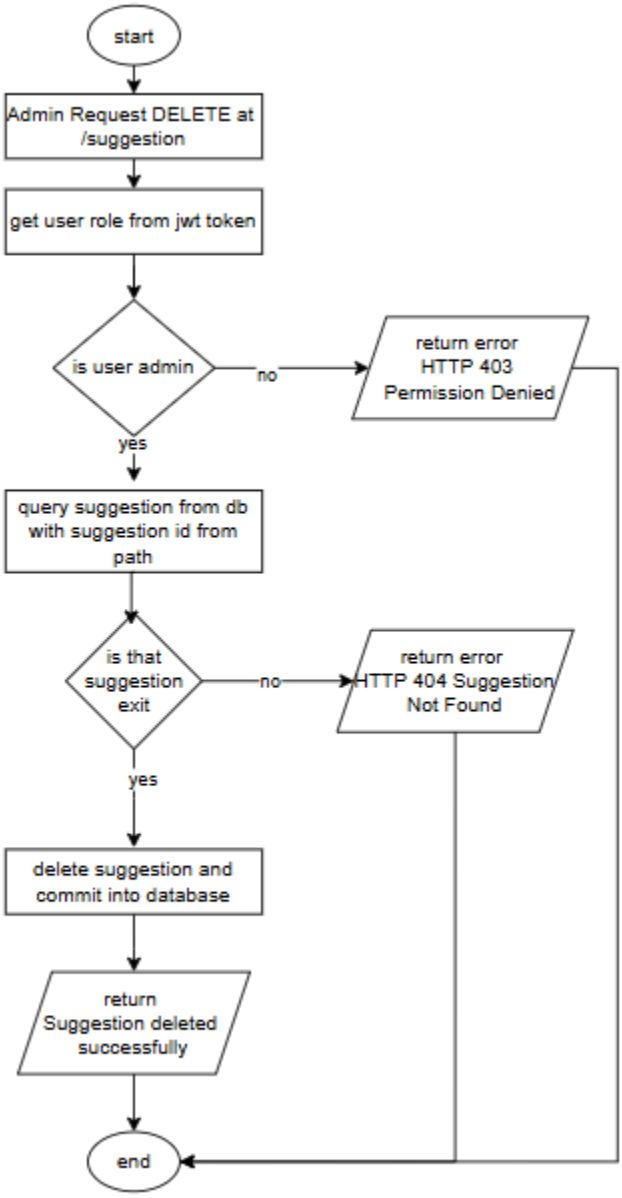
GET /suggestion ทำการเช็ค roleว่าใช่ Admin ไหม ถ้าใช้ดึงค่า suggestion ทั้งหมดไปส่ง



POST /suggestion user ส่งข้อเสนอแนะเข้าสู่ระบบ

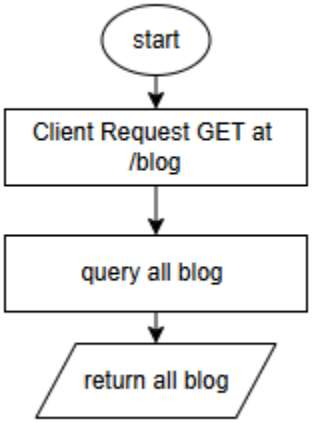


DELETE /suggestion adminส่ง request มาพร้อมกับ suggestion id เช็คว่าใช้ adminจริงไหมจึงจะดำเนินการต่อไปได้หลังจากดึงข้อเสนอแนะด้วย suggestion id ใน database แล้วเช็คว่า ข้อเสนอแนะนั้นมีอยู่จริงไหมจากนั้นจึงทำการลบ

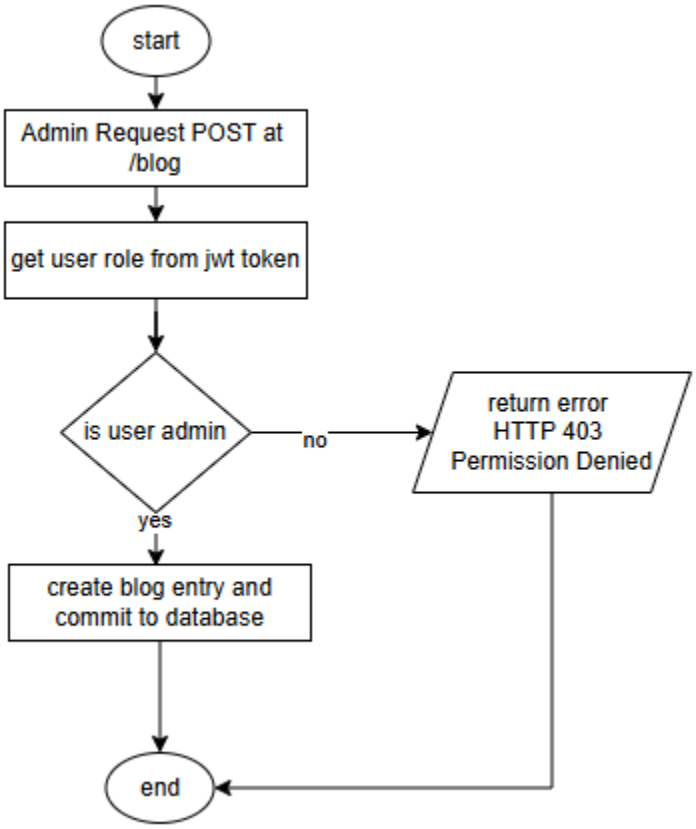

start
Admin Request DELETE at /suggestion
get user role from jwt token
is user admin
no
return error HTTP 403 Permission Denied
yes
query suggestion from db with suggestion id from path
is that suggestion exit
no
return error HTTP 404 Suggestion Not Found
yes
delete suggestion and commit into database
return Suggestion deleted successfully
end

2.6.14. blog

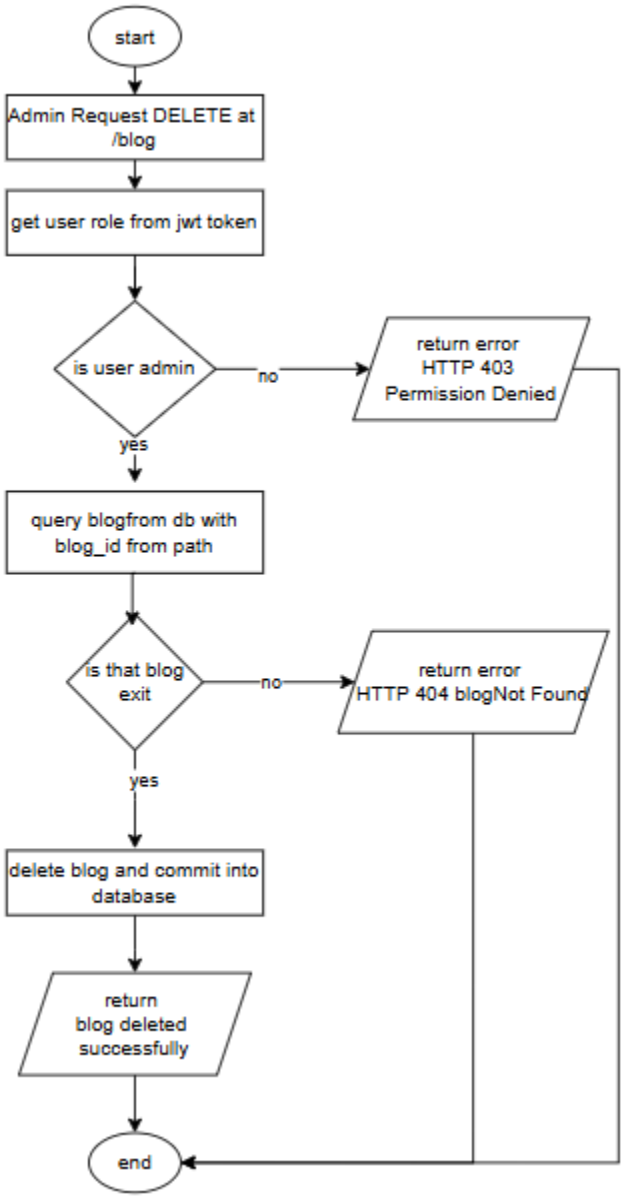
GET /blog user ทำการร้องขอดึงข้อมูล blog ทั้งหมดจาก database มาโชว์



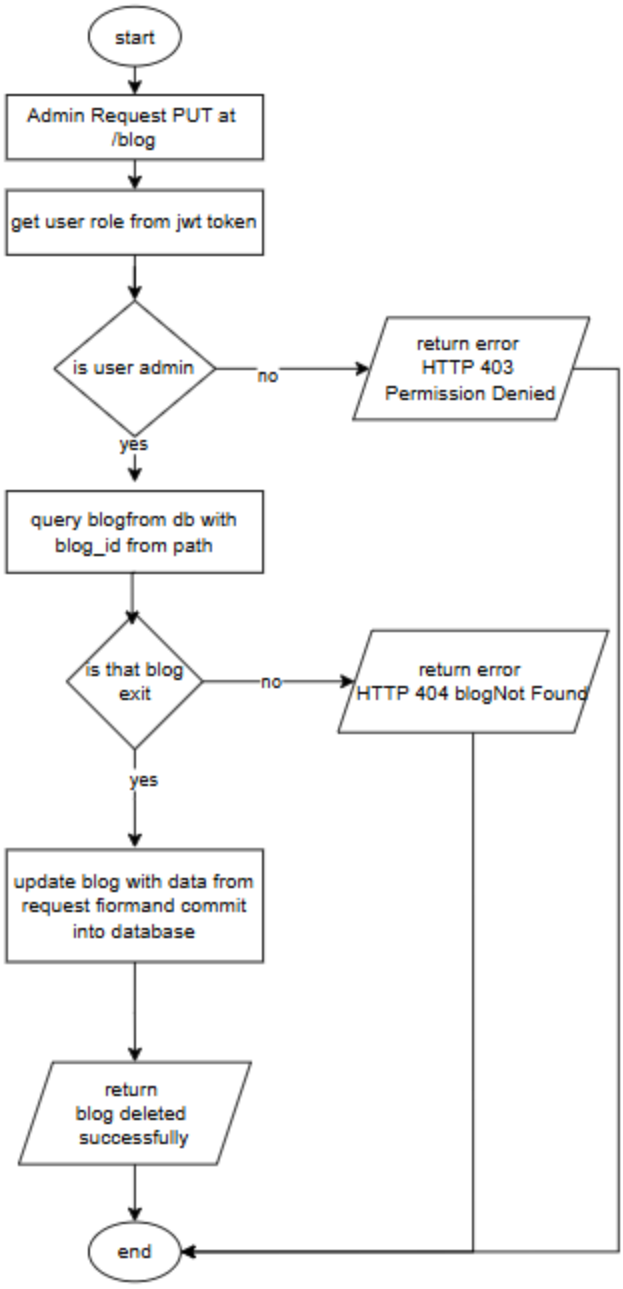
POST /blog admin ทำการร้องของ request มาเพื่อสร้างblog โดยจะตรวจสอบว่าใช่ admin จริงไหมจึงดำเนินการต่อ โดยจะนำข้อมูลที่ได้จาก request form มาสร้าง



DELETE /blog adminส่ง request มาพร้อมกับ blog_id เช็คว่าใช้ adminจริงไหมจึงจะดำเนินการต่อไปได้หลังจากดึง blogด้วย blog_id ใน database แล้วเช็คว่า blogนั้นมีอยู่จริงไหมจากนั้นจึงทำการลบ

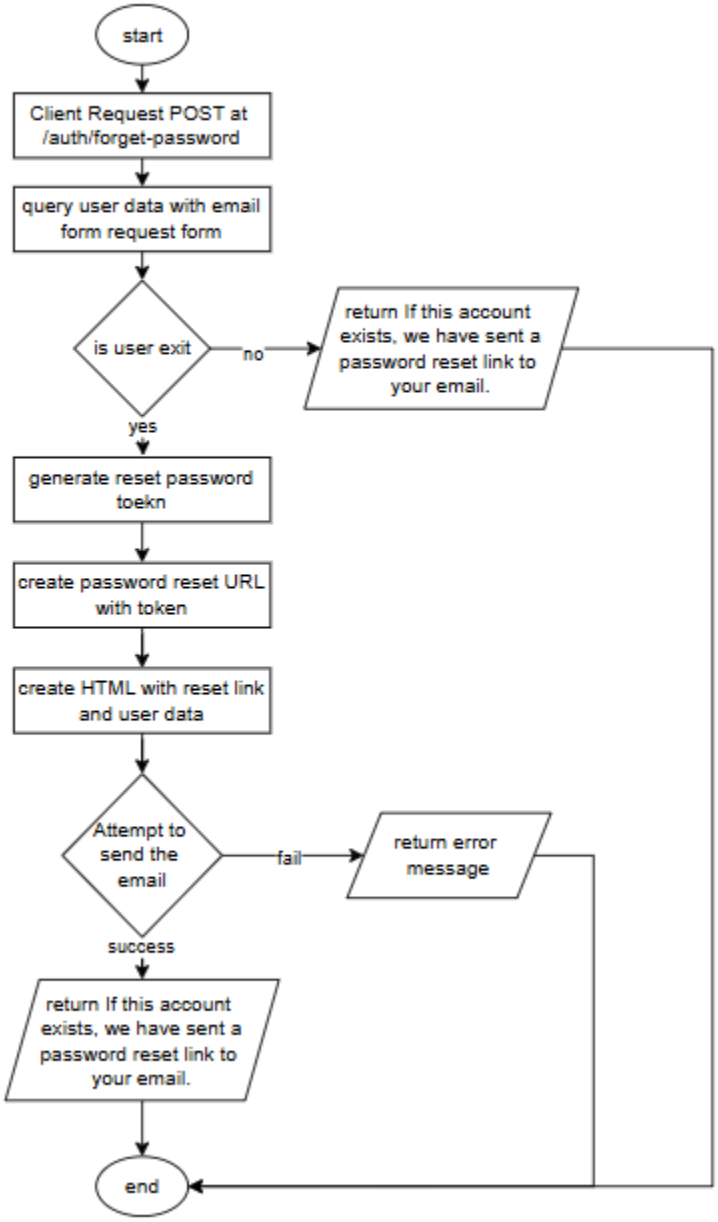


PUT /blog adminส่ง request มาพร้อมกับ blog_id เช็คว่าใช้ adminจริงไหมจึงจะดำเนินการต่อไปได้หลังจากดึง blogด้วย blog_id ใน database แล้วเช็คว่า blogนั้นมีอยู่จริงไหมจากนั้นจึงอัพเดทด้วยค่าที่ได้จาก request form


start
Admin Request PUT at /blog
get user role from jwt token
is user admin
no
return error HTTP 403 Permission Denied
yes
query blogfrom db with blog_id from path
is that blog exit
no
return error HTTP 404 blogNot Found
yes
update blog with data from request fiormand commit into database
return blog deleted successfully
end

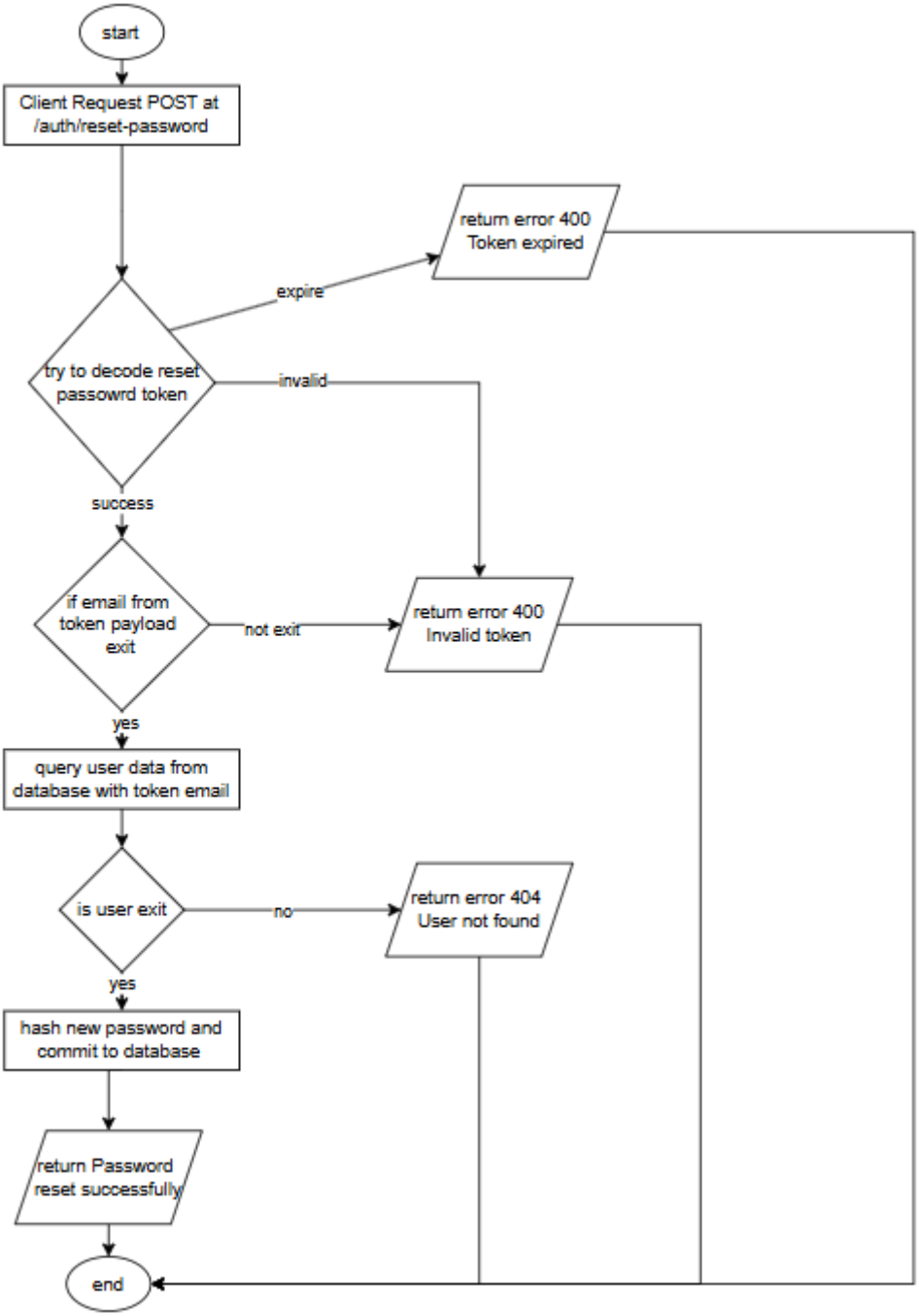
2.6.15. auth/forget-password

POST /auth/forget-password ทำการส่ง reset password URL ที่สร้างจาก tokenให้ userผ่าน email



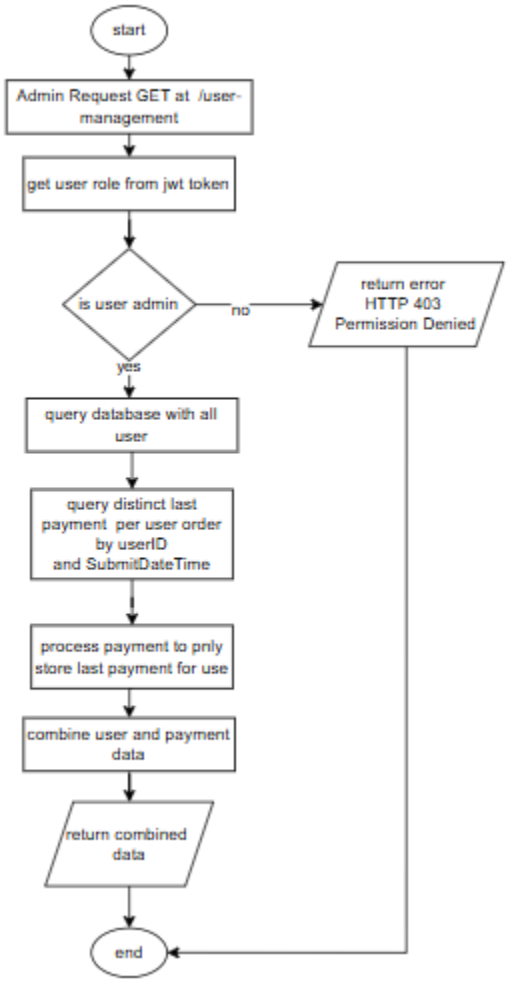
2.6.16. auth/reset-password

POST /auth/reset-password ทำการ decode reset password token ถ้า invalid ก็ return error หรือถ้าไม่มี userคนนั้นอยู่ในระบบจาก email ก็ return user not found ถ้าเจอก็ hash รหัสผ่านใหม่ update ใน database และ commit change

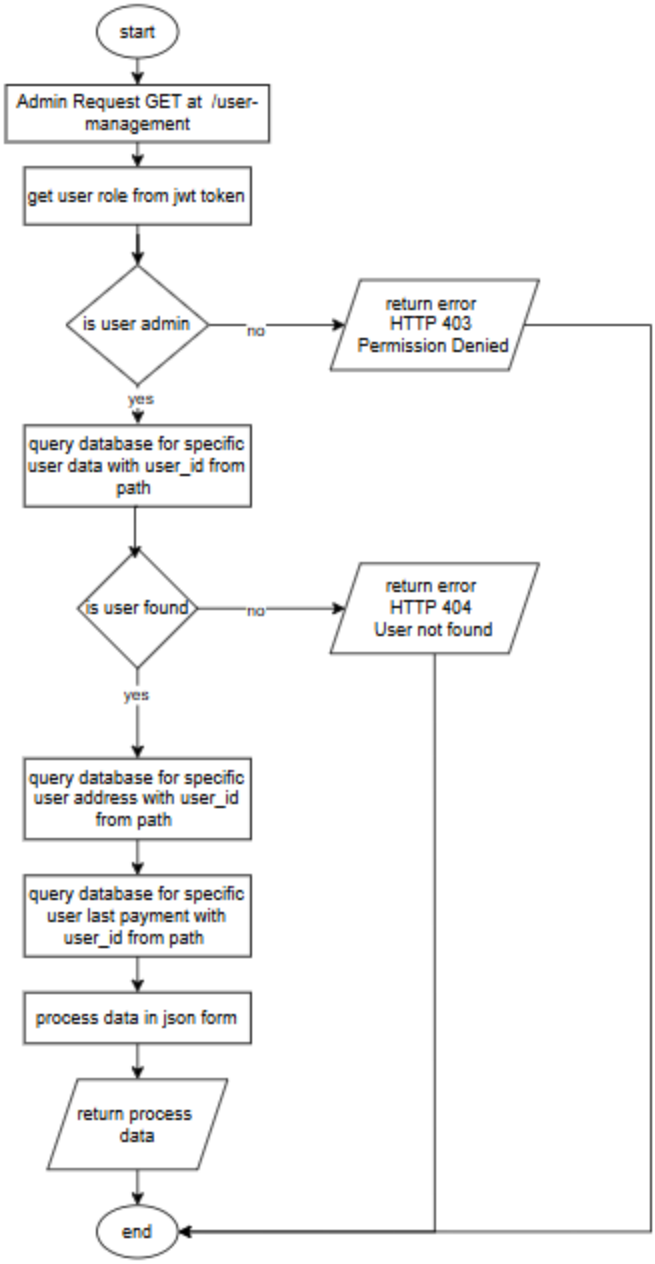

start
Client Request POST at /auth/reset-password
return error 400 Token expired
expire
try to decode reset passowrd token
invalid
success
if email from token payload exit
not exit
return error 400 Invalid token
yes
query user data from database with token email
is user exit
no
return error 404 User not found
yes
hash new password and commit to database
return Password reset successfully
end

2.6.17. user-management

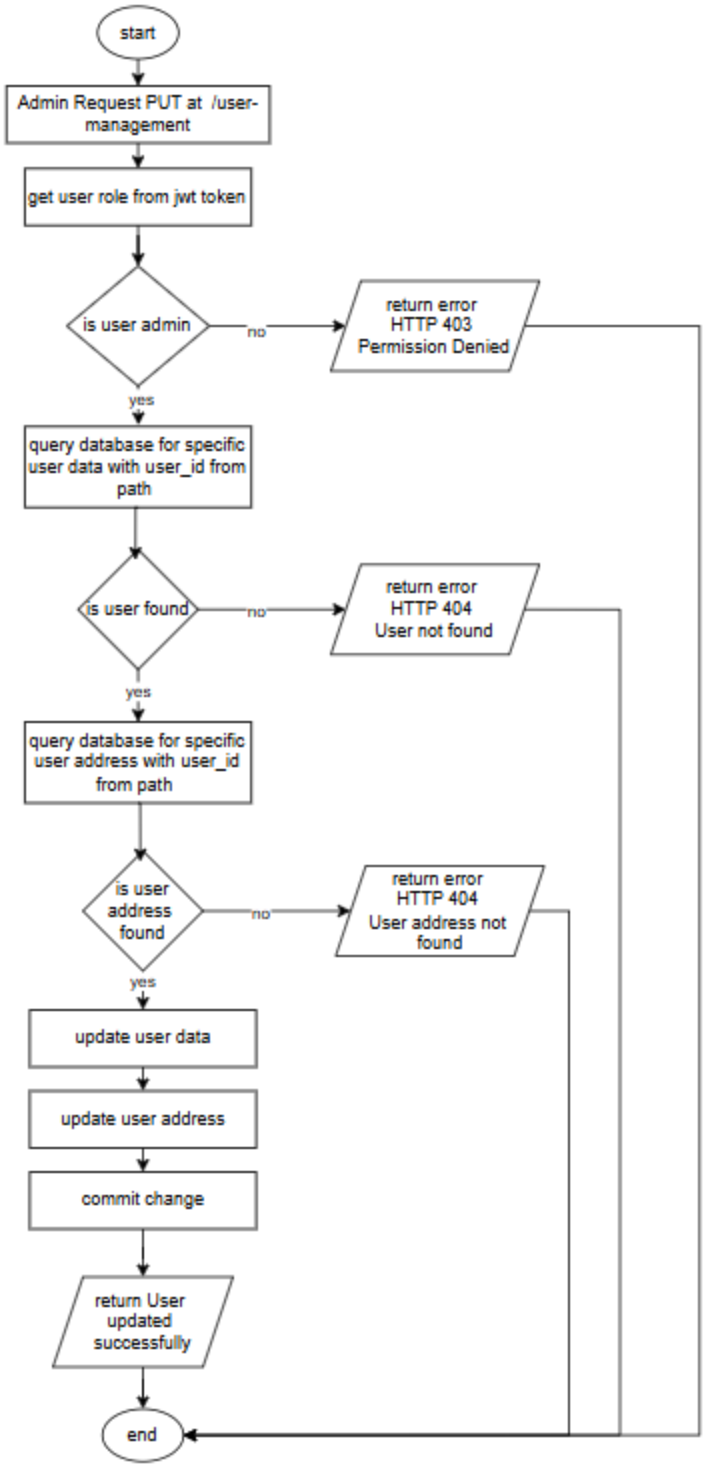
GET /user-management Admin ร้องขอรายการชำระเงินของ user โดยจะเอามาแค่การชำระเงินครั้งล่าสุดของแต่ละ user เท่านั้นหลังจากนั้นจะเอามารวมกับข้อมูลของ แต่userแล้ว return กลับไป



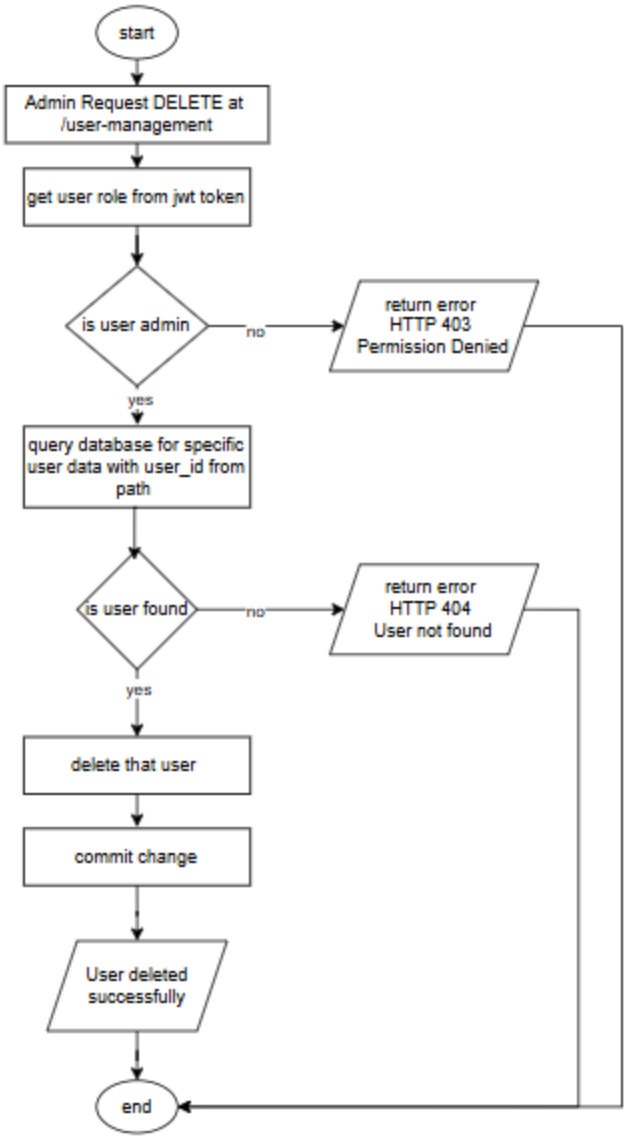
GET /user-management/{user_id} เช็คว่า role ใช่ admin ไหมจึงร้องขอมูลแบบละเอียดของ userด้วย user_id โดยจะนำมาจัดใน json แล้ว return กลับไป



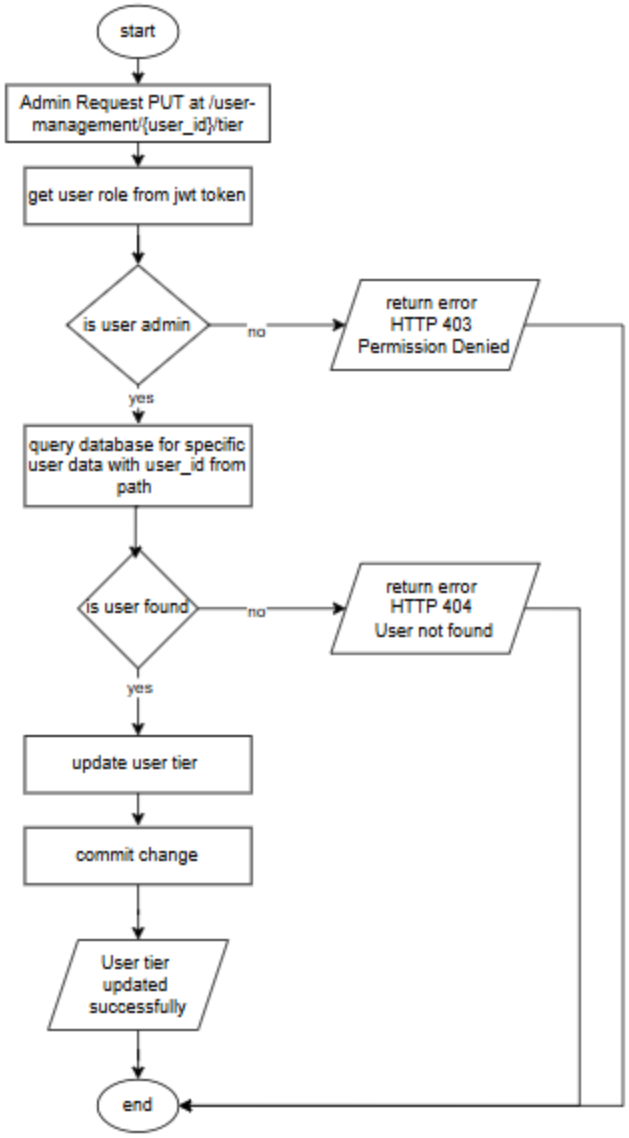
PUT /user-management เช็คว่าใช่ admin เป็นคน request ไหมจากนั้นจึงทำการร้องขอ user data กับ user address ไปยัง database ถ้าไม่เจอ user data หรือ user address ทำการ return error 404 กลับไปถ้าเจอก็ทำการ update data ที่มาจาก request form



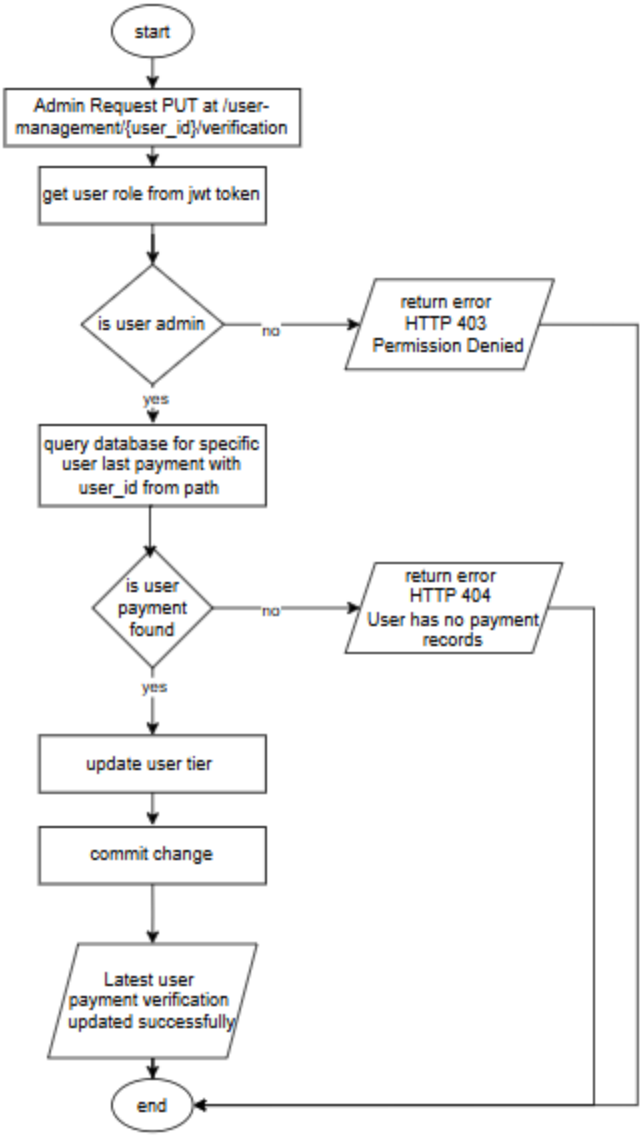
DELETE /user-management เช็คว่า request มาจาก admin ไหมจากนั้นเช็คต่อว่า user นั้นมีอยู่ไหมถ้าไม่มีก็ return error ถ้ามีก็ทำการลบ



PUT /user-management/{user_id}/tier เช็คว่า request มาจาก admin ไหมจากนั้นเช็คต่อว่า user นั้นมีอยู่ไหมถ้าไม่มีก็ return error ถ้ามีก็ทำการupdate tier ของ user ที่ได้มาจาก user_id

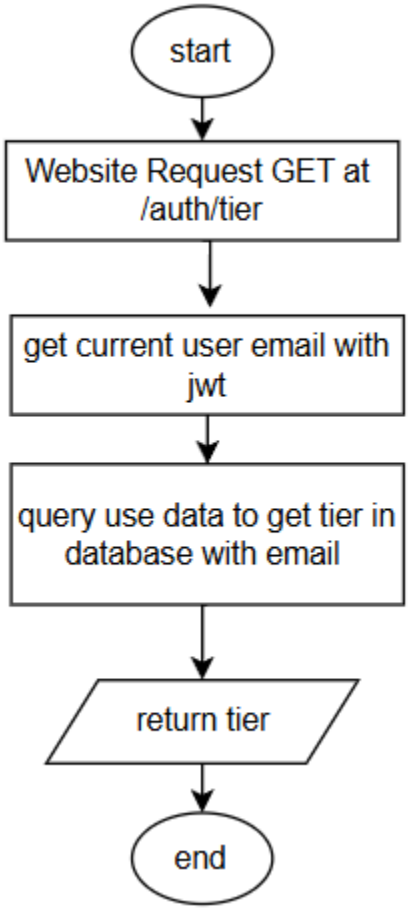


PUT /user-management/{user_id}/verification เช็คว่า request มาจาก admin ไหมจากนั้นเช็คต่อว่า user payment นั้นมีอยู่ไหมถ้าไม่มีก็ return error ถ้ามีก็ทำการupdate verification ของ last user payment ที่ได้มาจาก user_id


start
Admin Request PUT at /user-management/{user_id}/verification
get user role from jwt token
is user admin
no
return error
HTTP 403
Permission Denied
yes
query database for specific user last payment with user_id from path
is user payment found
no
return error
HTTP 404
User has no payment records
yes
update user tier
commit change
Latest user payment verification updated successfully
end

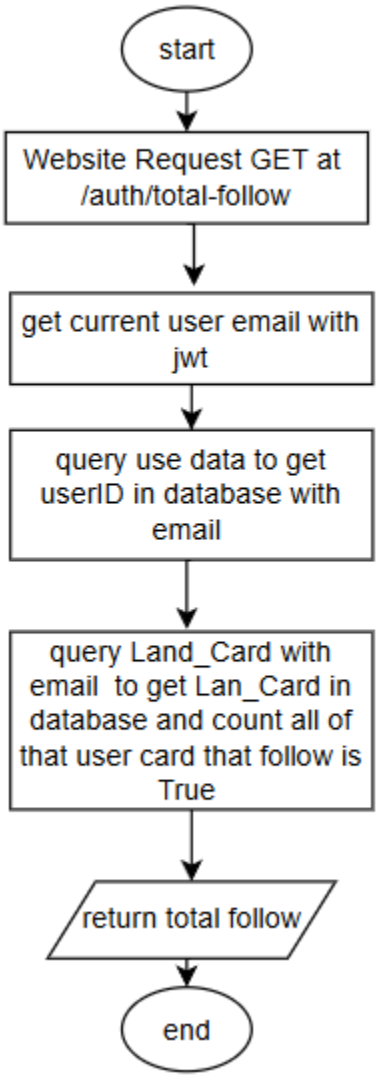
2.6.18. auth/tier

GET /auth/tier website จะทำการร้องขอ tier ของ user ที่ใช้งานโดยจะได้มาจากการนำ email ที่ได้จาก jwt token มาquery หา user dataใน database แล้วนำ tier มา return กลับไป



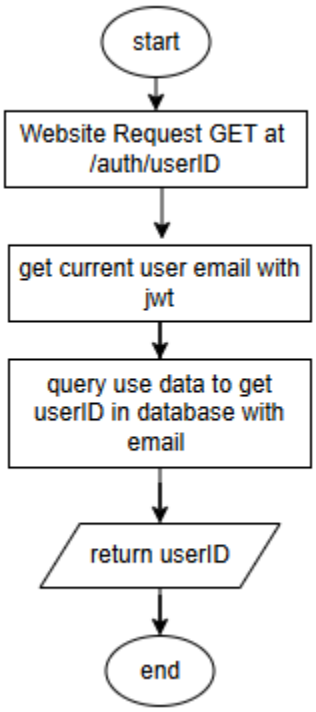
2.6.19. total-follow

GET /total-follow website จะทำการร้องขอ total follow ของ user ที่ใช้งานโดยจะได้มาจากการนำ email ที่ได้จาก jwt token มาquery หา user dataใน database แล้วนำ userID ไปหาจำนวน Land_Card ที่ follow ไว้ทั้งหมด



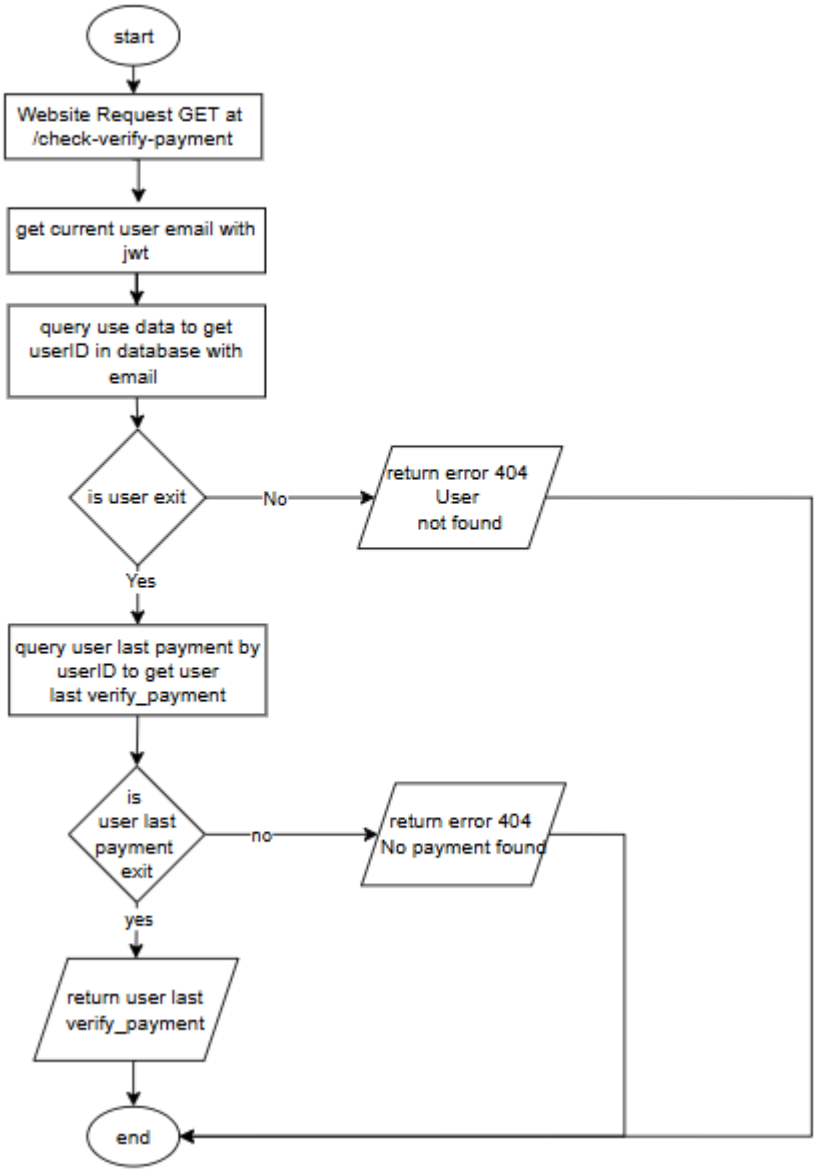
2.6.20. userID

GET /userID website จะทำการร้องขอ userID ของ user ที่ใช้งานโดยจะได้มาจากการนำ email ที่ได้จาก jwt token มาquery หา user dataใน database แล้วนำ userID มา return



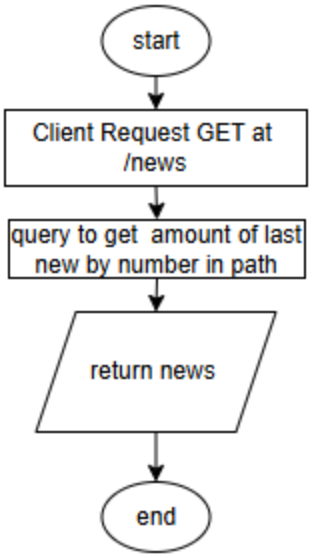
2.6.21. check-verify-payment

GET /check-verify-payment website จะทำการร้องขอ last verify payment ของ user ที่ใช้งานโดยจะได้มาจากการนำ email ที่ได้จาก jwt token มาquery หา user dataใน database แล้วนำ userID มาหา user last verify_payment แล้ว return กลับไปถ้า ถ้าไม่มีจะ return No payment found



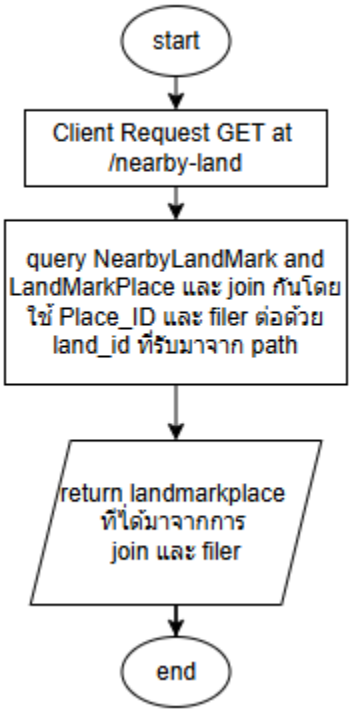
2.6.22. news

GET /news เมื่อมี request มาจาก user จำทำการ queryข่าวตาม {number} ที่ระบุมาแล้ว return ข่าวที่ query ได้กลับไป



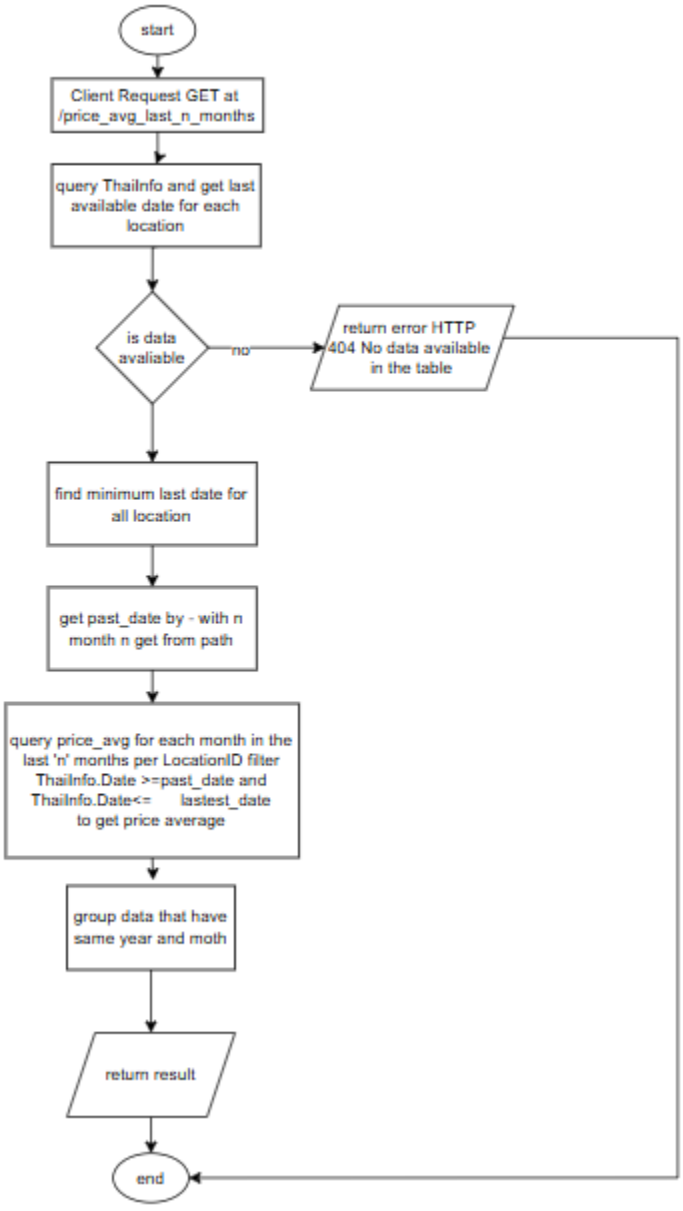
2.6.23. nearby-land

GET /nearby-land เมื่อมี request เข้ามาจะทำการ query NearbyLandMark กับ LandMarkPlace นำมาjoin กัน แล้ว filter ด้วย {land_id}จากนั้นก็ return LandMarkPlace ที่ได้กลับไป



2.6.24. price_avg_last_n_months

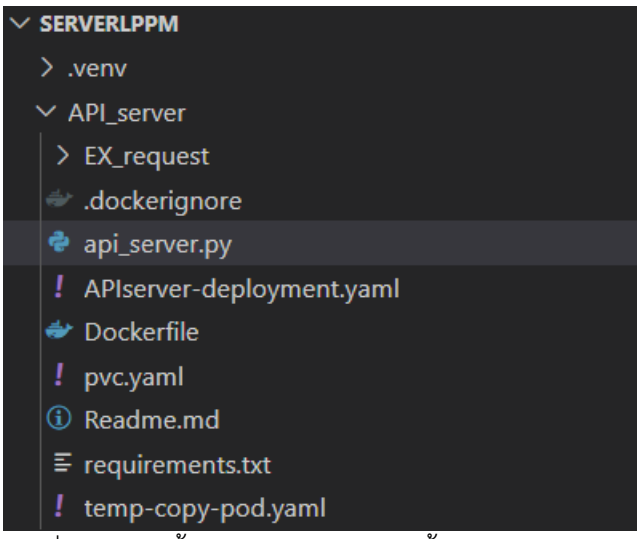
GET /price_avg_last_n_months query ThailInfo มาเพื่อหาวันสุดท้ายที่มีอยู่ในแต่ละ LocationID ถ้าไม่มีเลยจะทำการ return error กลับไปจากนั้นจึงหาวันสุดท้ายที่มีร่วมกันในทุก LocationID จากนั้น ก็คำนวณหา past_date โดยลบด้วยจำนวนเดือนที่อยากได้ซึ่งได้มาจาก {n}แล้วก็ query โดยให้วันอยู่ในช่วงระหว่างpast_date กับ last_date จากนั้นก็จัดกลุ่ม data โดยอิงตามปีและเดือนแล้วก็ return ค่ากลับไป



3. **ส่วนของ API ต่าง ๆ**

หลังจากที่โคลน https://github.com/LPPACEGroup/APIServerLPPM.git

จะเห็นว่ามีโฟลเดอร์ API_server ซึ่งภายในนั้นจะมีไฟล์ api_server.py เป็นไฟล์ที่ประกอบไปด้วย API ที่ใช้บน server และทำงานร่วมกันกับ frontend

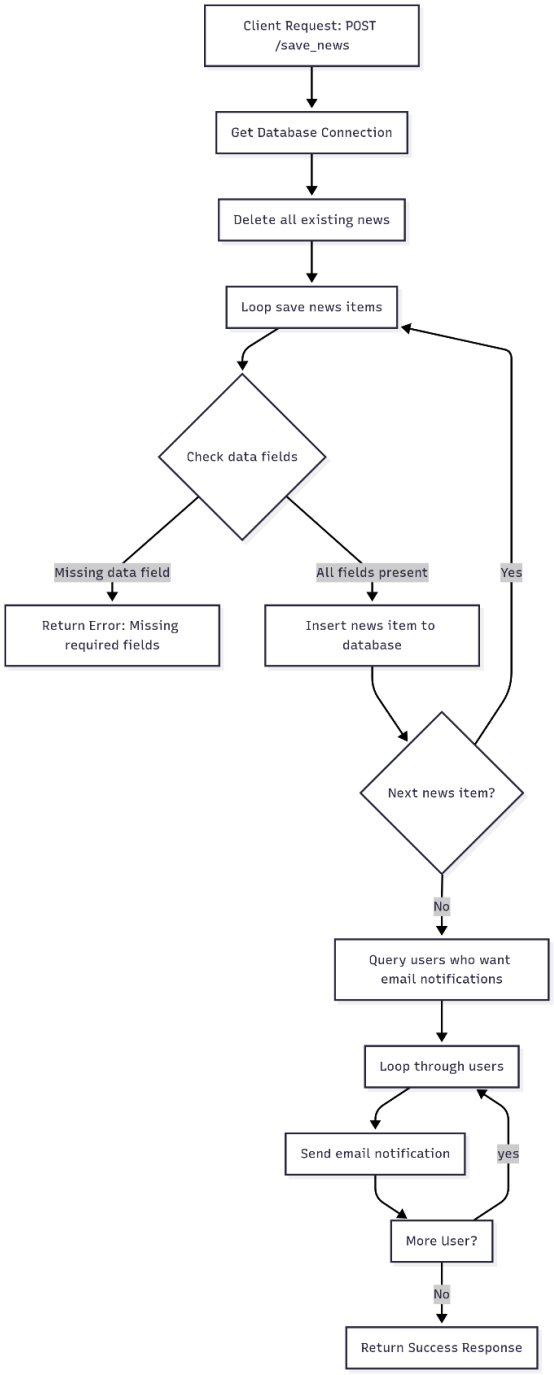


ซึ่ง API ที่อยู่ในไฟล์นี้จะประกอบไปด้วยดังนี้

- save_news :เป็น API สำหรับ Admin สำหรับการบันทึกข่าวสารใหม่และส่งการแจ้งเตือนไปหาผู้ใช้
- calculate_land_tax :เป็น API สำหรับการคำนวณภาษีของที่ดินตาม
- get_goodsale_analytics :เป็น API สำหรับการคำนวณอัตราการเปลี่ยนแปลงของราคาเฉลี่ยของปีต่อปีและไตรมาสต่อไตรมาส
- get_profile :เป็น API สำหรับส่ง URL รูปภาพโปรไฟล์ของผู้ใช้งานเพื่อนำไปแสดงบนเว็บไซต์ได้
- get_latest_slip :เป็น API สำหรับส่ง URL รูปภาพที่ผู้ใช้ได้ทำการชำระเงินมาและนำมาแสดงในส่วนเว็ปไซต์ของฝั่งแอดมิน
- get_land_image :เป็น API สำหรับส่ง URL รูปภาพที่ดินโดยจะรับค่า image_id และส่งรูปนั้นกลับไป
- get_land_images :เป็น API สำหรับส่ง URL รูปภาพที่ดินทั้งหมดของ LandDataID นั้นๆมาแสดงบนเว็บไซต์
- upload_profile :เป็น API สำหรับให้ผู้ใช้งานอัปโหลดรูปภาพโปรไฟล์ที่แสดงบนเว็บไซต์ใหม่ได้
- submit_payment :เป็น API สำหรับการชำระเงินที่จะรับรูปภาพการโอนเงินรวมถึงข้อมูลการชำระเงินเข้าสู่ฐานข้อมูล

3.1. save_news

จะรับข้อมูล JSON ที่ประกอบไปด้วยข้อมูล Topic, Content, Link, Pic, Date ของข่าวและจะทำการลบข้อมูลข่าวเดิมทั้งหมดที่มีในฐานข้อมูลออกและจะบันทึกข่าวใหม่ลงฐานข้อมูล MySQL และหลังจากนั้นจะทำการส่ง Email แจ้งเตือนผู้ใช้งานที่เปิดรับการแจ้งเตือนผ่านทางอีเมลบนเว็บไซต์และเมื่อสิ้นสุดการทำงานจะแสดงข้อความ News saved and notifications sent successfully.

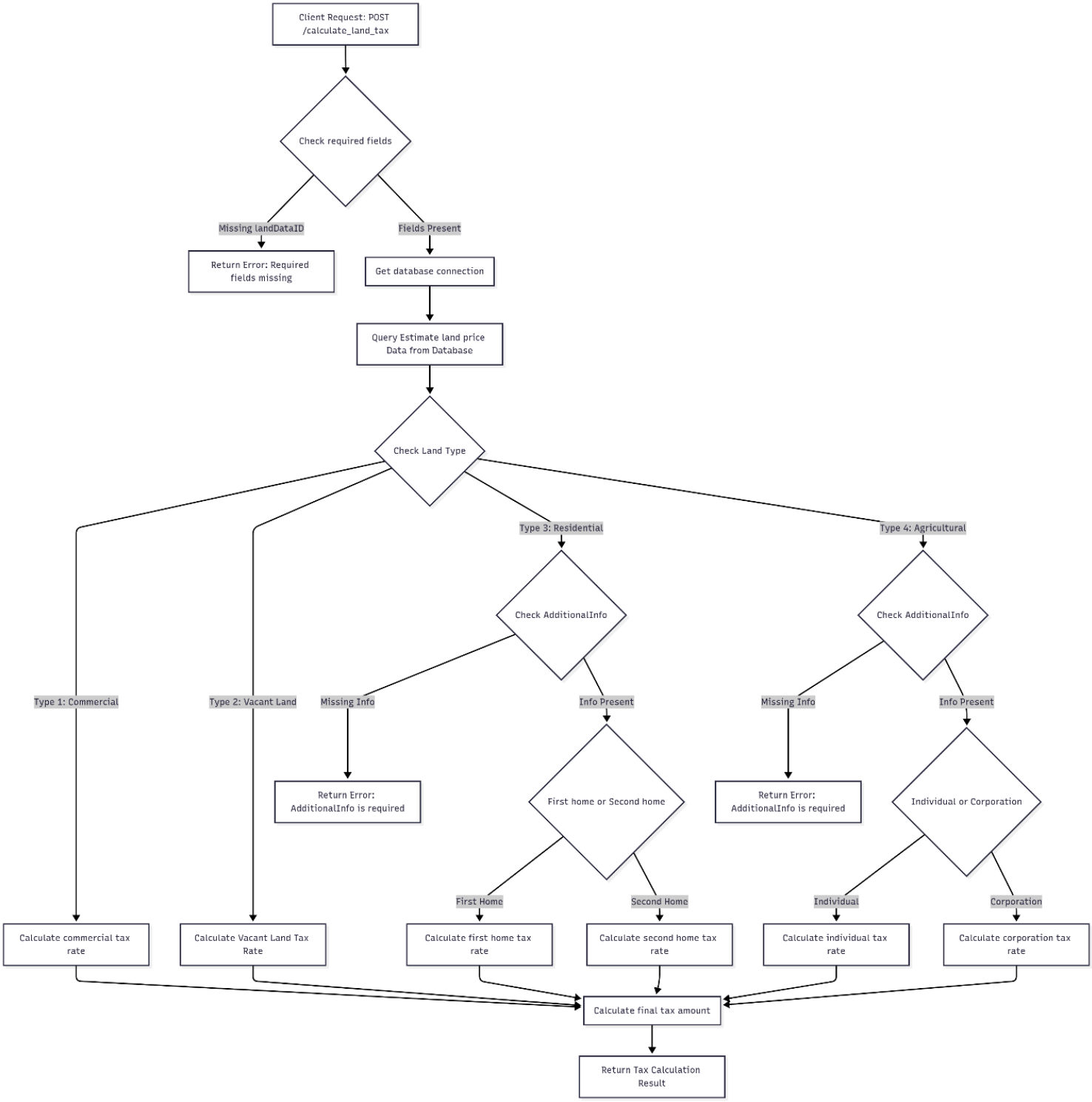


3.2. calculate_land_tax

จะรับข้อมูลจากหน้าเว็บไซต์ตามที่ผู้ใช้ได้ทำการส่งมา LandType และ AdditionalInfo ซึ่งระบบจะทำการนำข้อมูลราคาประเมินของที่ดินที่ผู้ใช้ต้องการคำนวณนั้นมาคิดภาษีโดยเงื่อนไขในการคำนวนภาษีนั้นจะเลือกจาก LandType ทั้ง 4 ประเภทคือ

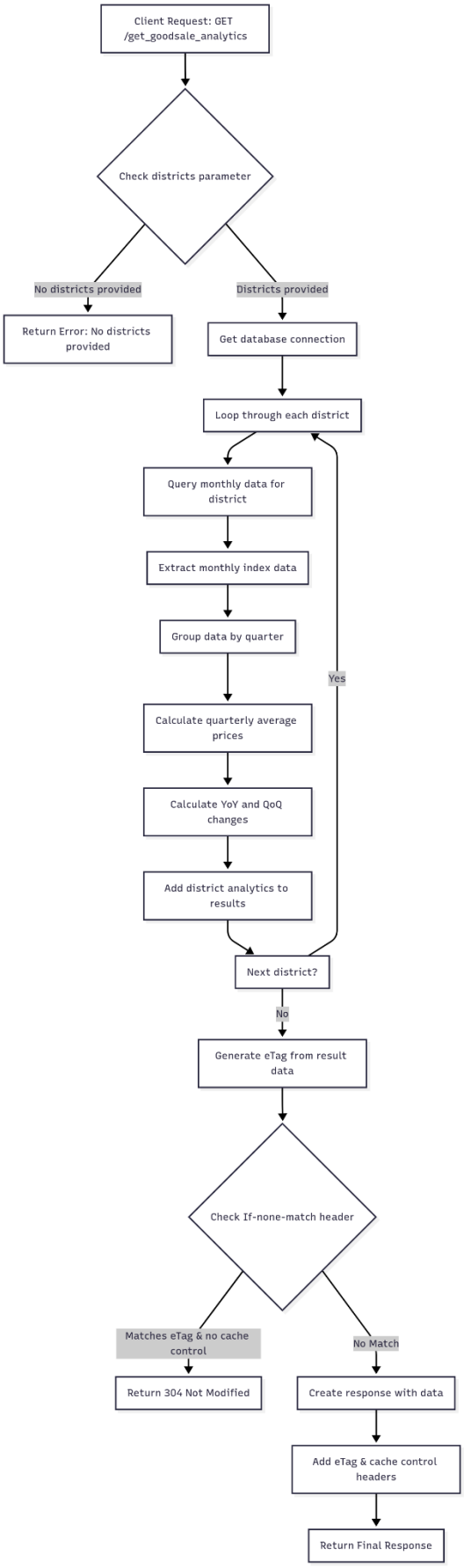
- เท่ากับ 1 เชิงพานิชย์
- เท่ากับ 2 ที่ดินรกล้าง
- เท่ากับ 3 ที่อยู่อาศัย ซึ่งจะต้องใช้ข้อมูล AdditionalInfo แยกอีกว่า
    - เท่ากับ 1 บ้านหลังหลักหลังแรก
    - เท่ากับ 2 บ้านหลังที่สองขึ้นไป
- เท่ากับ 4 ที่ดินการเกษตร ซึ่งจะต้องใช้ข้อมูล AdditionalInfo แยกอีกว่า
    - เท่ากับ 1 บุคคลธรรมดา
    - เท่ากับ 2 นิติบุคคล

หลังจากนั้นจะนำอัตราการเสียภาษีของที่ดินประเภทต่างๆ มาคำนวณกับราคาประเมินเพื่อให้ได้ภาษีที่ดินที่ต้องจ่าย


Client Request: POST /calculate_land_tax
Check required fields
Missing landDataID
Fields Present
Return Error: Required fields missing
Get database connection
Query Estimate land price Data from Database
Check Land Type
Type 1: Commercial
Type 2: Vacant Land
Type 3: Residential
Type 4: Agricultural
Check AdditionalInfo
Check AdditionalInfo
Missing Info
Info Present
Missing Info
Info Present
Return Error: AdditionalInfo is required
First home or Second home
Return Error: AdditionalInfo is required
Individual or Corporation
First Home
Second Home
Individual
Corporation
Calculate commercial tax rate
Calculate Vacant Land Tax Rate
Calculate first home tax rate
Calculate second home tax rate
Calculate individual tax rate
Calculate corporation tax rate
Calculate final tax amount
Return Tax Calculation Result

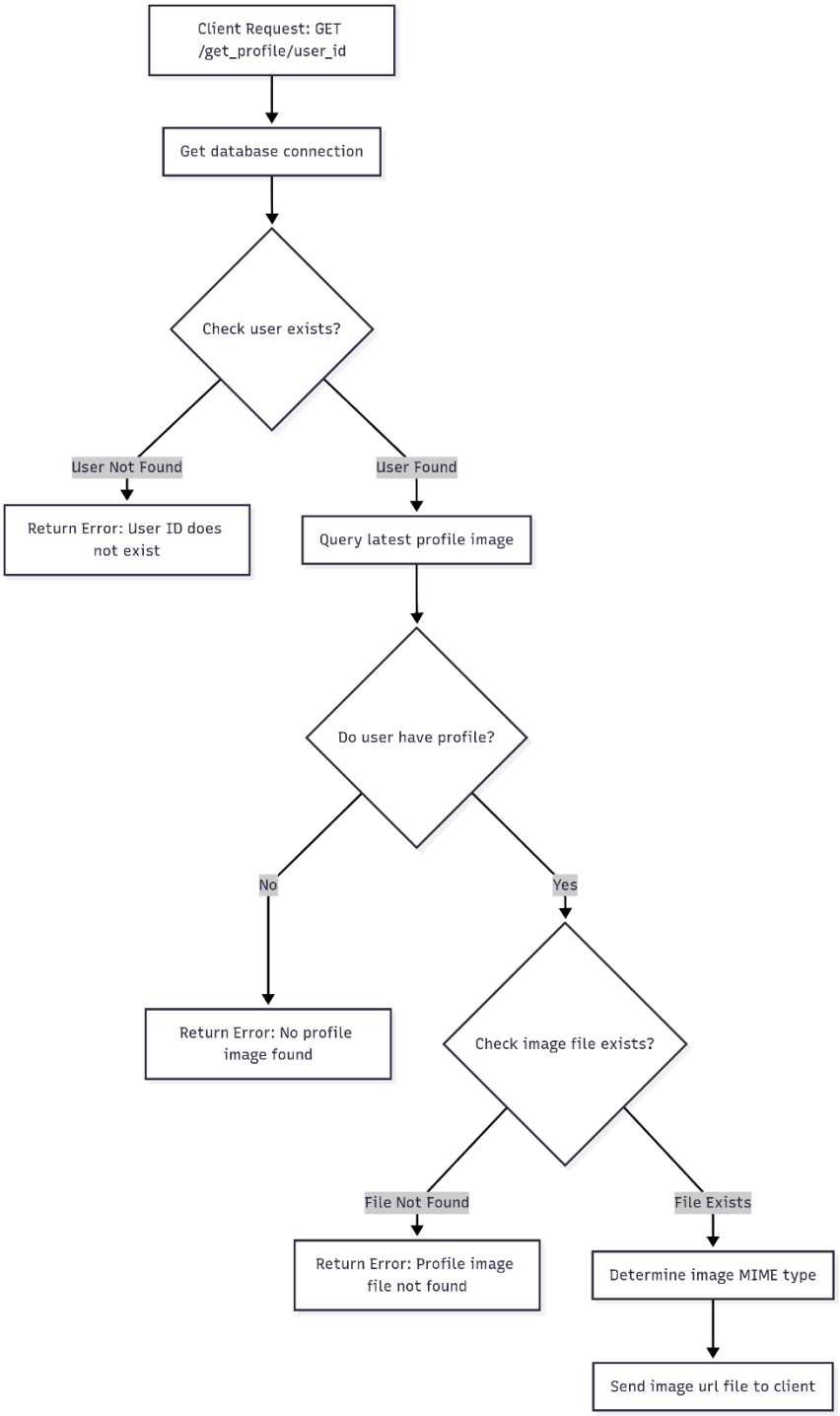
3.3. get_goodsale_analytics

จะรับชื่อ เขต ที่ต้องการคำนวณจากทาง frontend มาและทำการเรียกข้อมูลจากฐานข้อมูล MySQL โดยจะเรียกข้อมูล year(ปี) ,month(เดือน) ,price_avg(ราคาที่ดินเฉลี่ย) ,IndexLand(ดัชนีราคาที่ดิน) จากนั้นจะนำข้อมูลมาจัดกลุ่มเป็นไตรมาสต่างๆ และทำการคำนวณราคาที่ดินเฉลี่ยในแต่ละไตรมาสเลย จากนั้นจำทำการคำนวณ QoQ(Quarter over Quarter) โดยจะคำนวณจากไตรมาสปัจจุบันและไตรมาสก่อนหน้า และ YoY(Year over Year) โดยจะคำนวนไตรมาสล่าสุดของปีปัจจุบันและไตรมาสเดียวกันของปีก่อนหน้า และเพื่อไม่ให้ frontend ส่งคำขอข้อมูลมาตลอดเวลาจึงต้องกำหนด Cache Controller เพื่อให้ข้อมูลที่ถูกเรียกถูกเก็บไว้ได้ โดยถ้าไม่มีการเปลี่ยนแปลงจะไม่ส่งข้อมูลใหม่และใช้ข้อมูลเดิมทำให้ไม่ต้องโหลดข้อมูลทุกครั้งและส่งข้อมูลเป็นรูปแบบ JSON โดยมีข้อมูลดังนี้ 'year','quarter', 'avg_price', 'qoq', 'qoq_comparison', 'yoy', 'yoy_comparison', 'index', 'monthly_indices'


Client Request: GET /get_goodsale_analytics
Check districts parameter
No districts provided
Districts provided
Return Error: No districts provided
Get database connection
Loop through each district
Query monthly data for district
Extract monthly index data
Group data by quarter
Calculate quarterly average prices
Calculate YoY and QoQ changes
Add district analytics to results
Next district?
Yes
No
Generate eTag from result data
Check If-none-match header
Matches eTag & no cache control
No Match
Return 304 Not Modified
Create response with data
Add eTag & cache control headers
Return Final Response

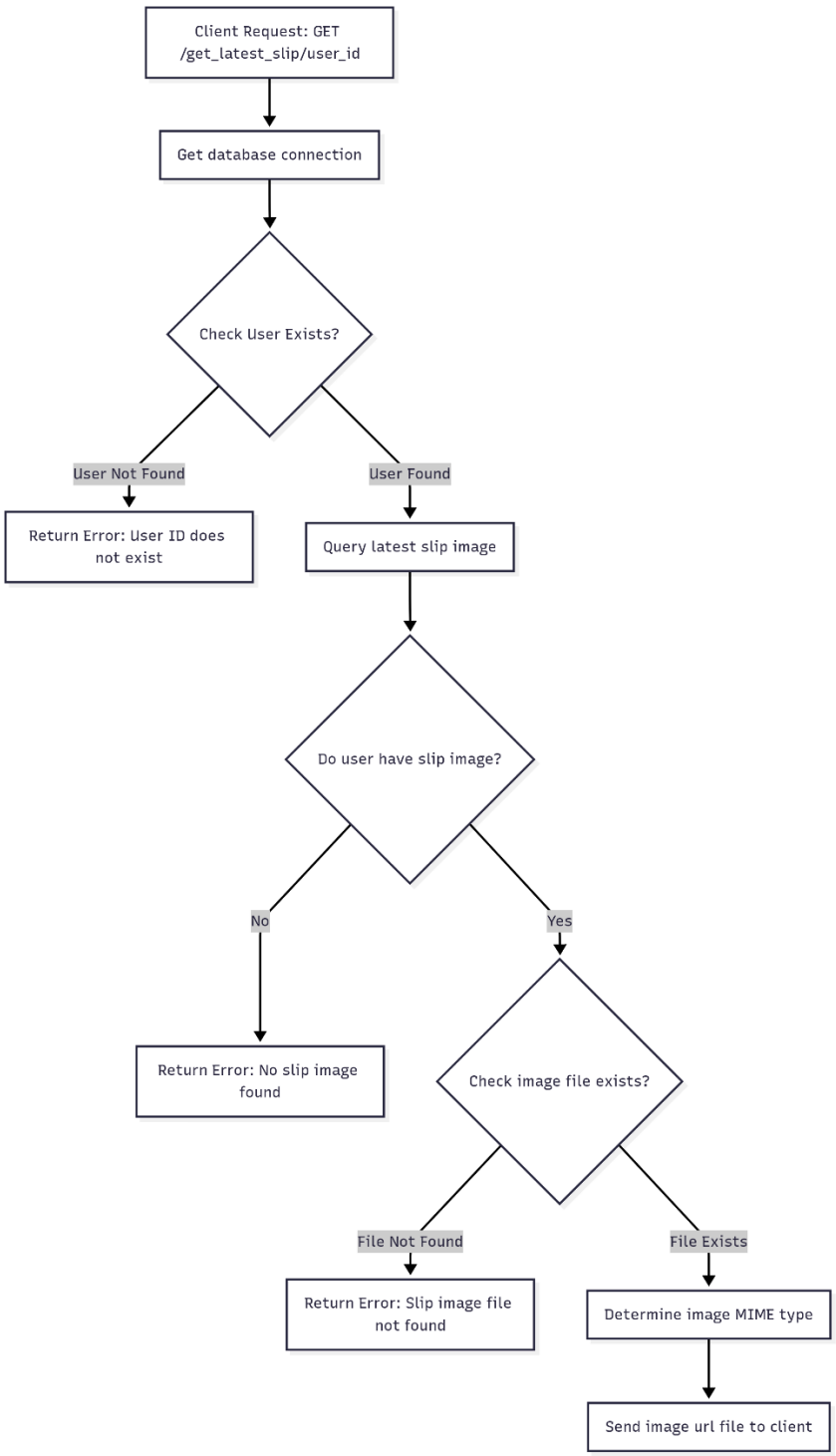
3.4. get_profile

จะรับ User_ID ของผู้ใช้งานโดยจะเรียก path รูปภาพจากฐานข้อมูลและส่งเป็นไฟล์นามสกุล jpg หรือ jpeg ออกไปให้ส่วนของ frontend นำไปแสดงในส่วนต่างๆ



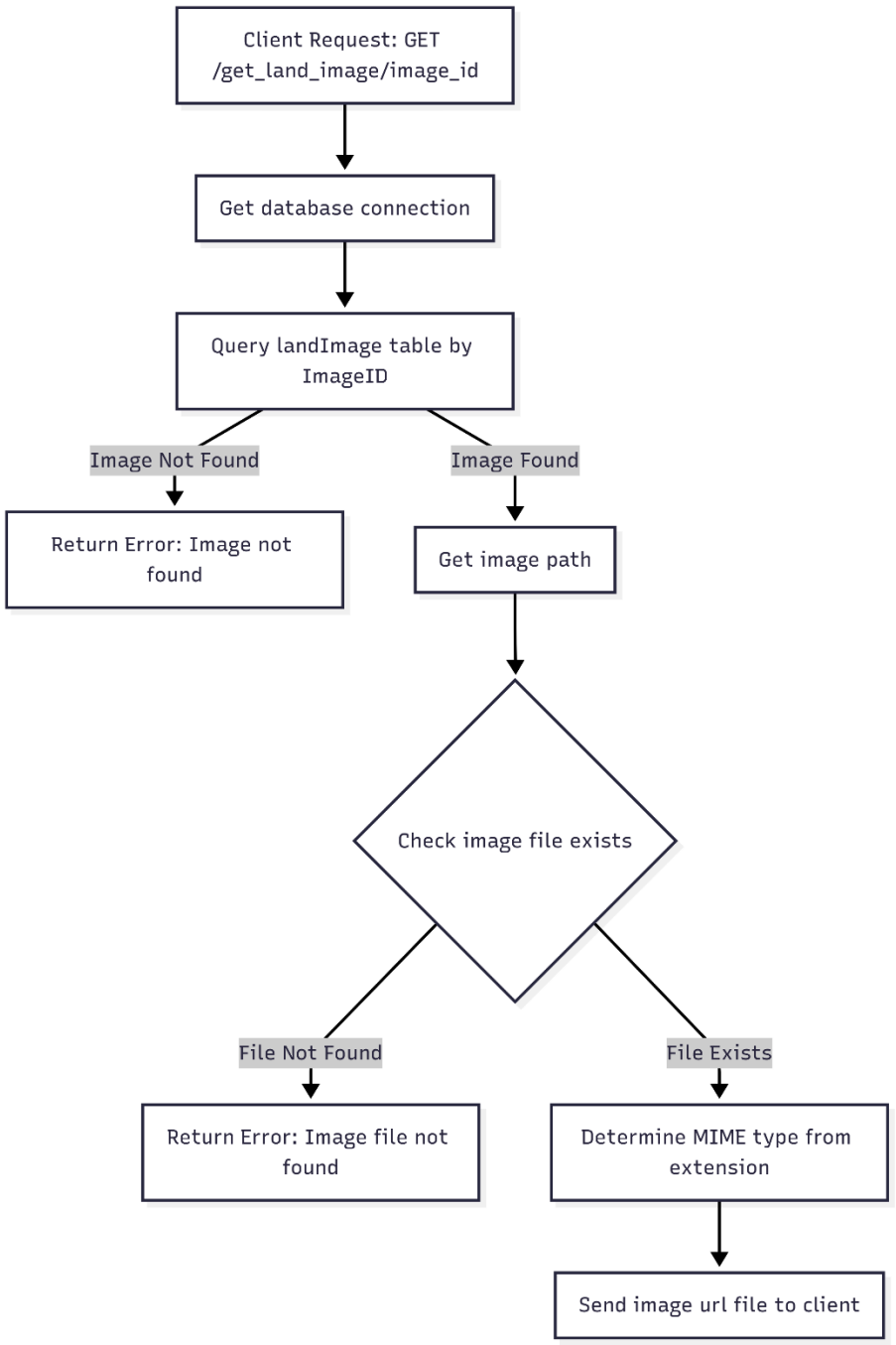
3.5. get_latest_slip

จะเป็นการรับค่า User_ID ของผู้ใช้เพื่อนำไปหา path ของรูปการณ์ชำระเงินล่าสุดจากฐานข้อมูลและส่งไฟล์ออกเป็นนามสกุล jpg หรือ jpeg เพื่อแสดงผลที่ส่วน frontend ของฝั่ง admin



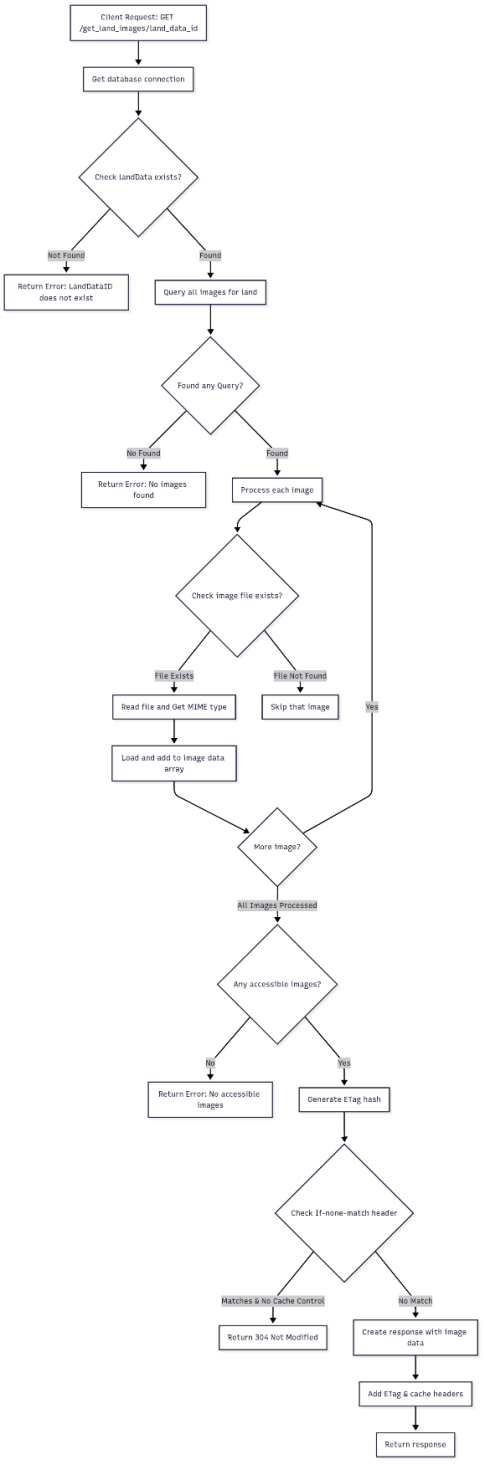
3.6. get_land_image

รับค่า image_id เพื่อหา path รูปภาพที่ดินจากฐานข้อมูลและส่งไฟล์ออกเป็นนามสกุล jpg หรือ jpeg



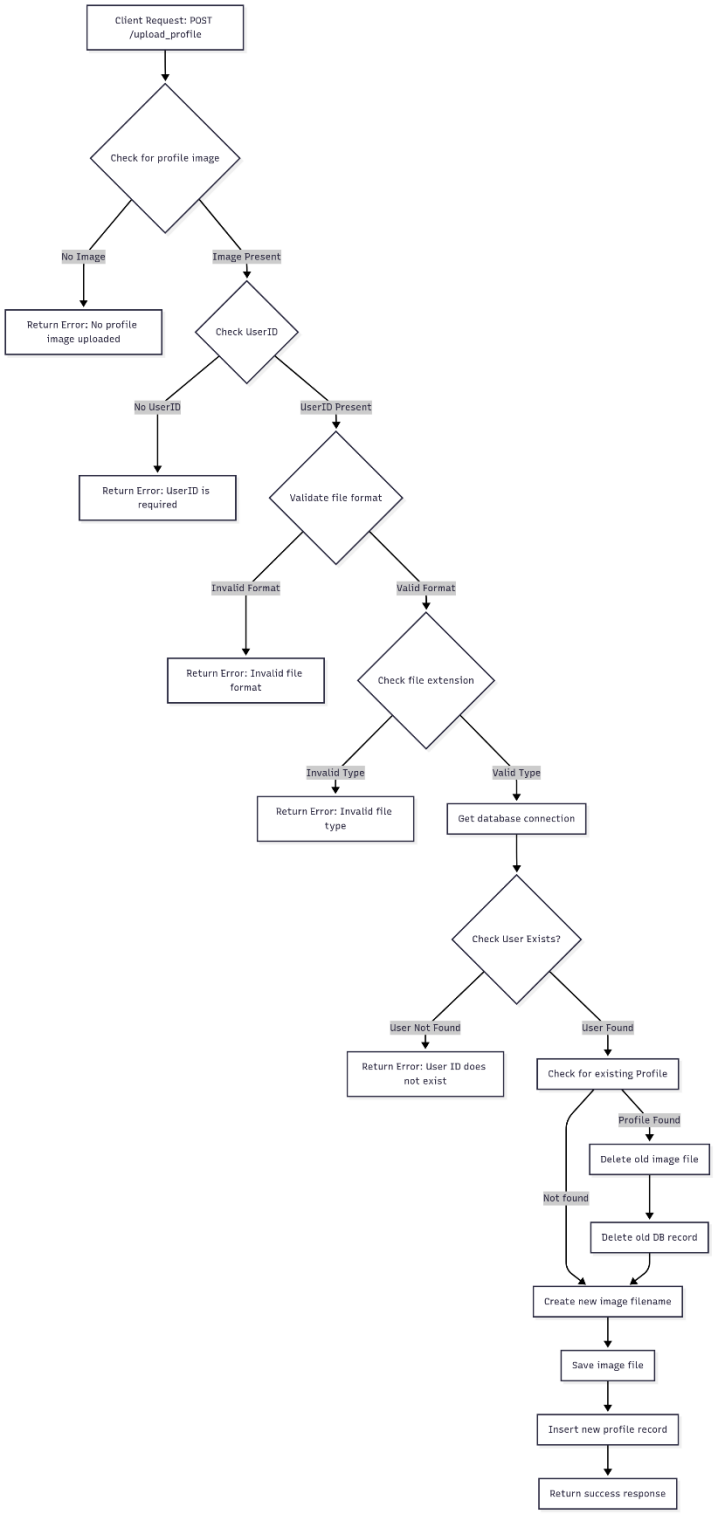
3.7. get_land_images

รับค่า LandDataID ของที่ดินมาเพื่อหา path ของรูปภาพของที่ดินนั้น ๆ ทุกรูปที่อยู่ในฐานข้อมูล MySQL และส่งไฟล์เหล่านั้นออกเป็นนามสกุล jpg หรือ jpeg ไปแสดงผลบน frontend โดยก่อนที่จะทำการส่งรูปภาพทั้งหมดนั้นจะทำการ Cache Controller เพื่อไม่ให้มีการส่งข้อมูลทุกครั้ง โดยจะถูกเก็บไว้ใน cache และถ้าหากข้อมูลรูปภาพมีการเปลี่ยนแปลงถึงจะส่งข้อมูลใหม่ได้



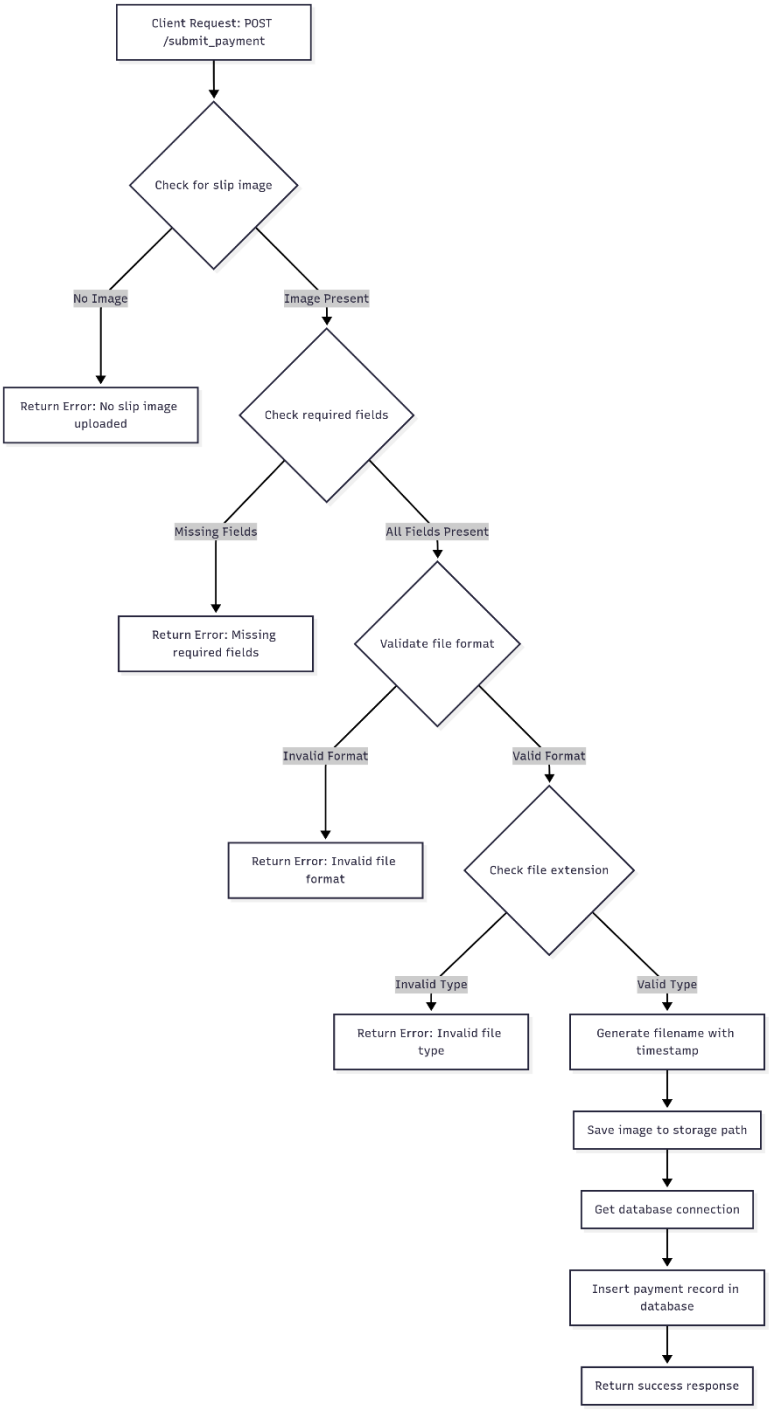
3.8. upload_profile

จะรับรูปภาพที่ถูกอัพโหลดจาก frontend และ User_ID เพื่อทำการบันทึกลงฐานข้อมูล MySQL แต่ก่อนจะบันทึกจะต้องลบข้อมูลรูปภาพเดิมออกจากฐานข้อมูลก่อนรวมถึงไฟล์ที่ถูกเก็บไว้ด้วย หลังจากนั้นถึงจะทำการบันทึกไฟล์รูปภาพและข้อมูลลงฐานข้อมูลได้



3.9. submit_payment

จะรับข้อมูลและรูปภาพการชำระเงินจาก frontend โดยจะมีข้อมูลดังนี้ slip_image(รูปภาพการชำระเงิน), UserID, BuyTier(ระดับสมาชิกที่ต้องการ), AccName(ชื่อบัญชีของผู้ใช้), PaidPrice(จำนวนเงินที่จ่าย), Detail(หมายเหตุ) โดยจะนำไฟล์รูปโหลดเก็บไว้ลงบน server และนำ path รวมถึงข้อมูลต่าง ๆ เก็บลงในฐานข้อมูล MySQL



4. ส่วนของระบบ API คำนวณความน่าสนใจของที่ดิน

API นี้จะอยู่ในโฟลเดอร์ InterestLand ชื่อไฟล์ api_estimate.py โดย API จะรับค่า LandDataID ของที่ดินเพื่อนำมาหาสถานที่สำคัญทั้งหมดของที่ดินนั้น ๆ จากนั้นนำเก็บไว้ในตัวแปร nearby_places เพื่อส่งเข้าสู่ฟังก์ชัน calculate_land_interest_score โดยในฟังก์ชันนี้จะส่งข้อมูล places_nearby preprocess_places() เพื่อทำการจัดกลุ่มประเภทของสถานที่ใกล้เคียงโดยนำสถานที่ที่ใกล้ที่สุดในแต่ละประเภท 2 ที่เท่านั้นในการคำนวณหาค่าเฉลี่ยของระยะทางและส่งข้อมูล ระยะทางของประเภทสถานที่สำคัญกลับมาที่ฟังก์ชัน calculate_land_interest_score โดยเก็บไว้ที่ processed_places และจะนำเข้าสู่ฟังก์ชัน calculate_type_score พร้อมกับค่า WEIGHT_MAPPING ซึ่งเป็นค่าความสำคัญของประเภทสถานที่ต่าง ๆ ซึ่งค่า WEIGHT_MAPPING จะมีค่าดังต่อไปนี้

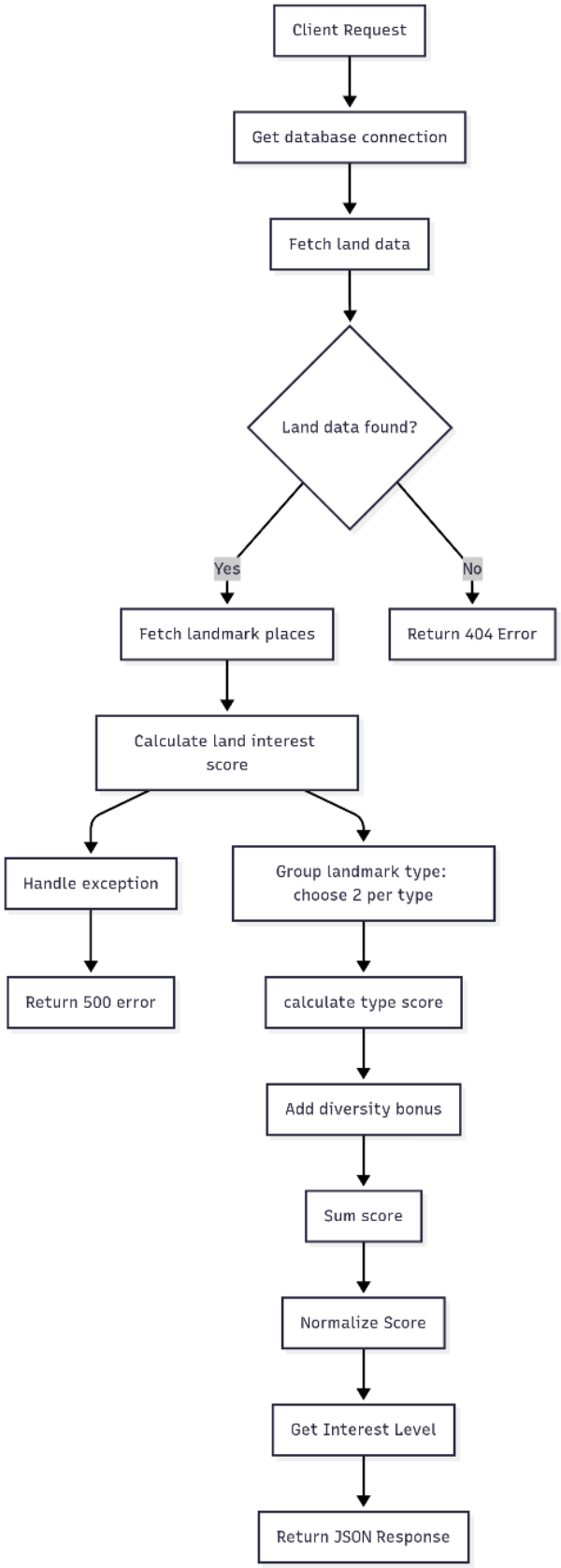
- "main_road": 8,
- "shopping": 7,
- "public_transport": 9,
- "station": 9,
- "school": 5,
- "commercial": 7,
- "park": 5,
- "tourist_attraction": 7,
- "hospital": 6,
- "other": 4,
- "historic_site": 7,
- "university": 5

โดยในฟังก์ชัน calculate_type_score จะส่งค่าคะแนนออกเป็น 0 ทันทีถ้าระยะทางไกลเกิน 5 กิโลเมตรและถ้าน้อยกว่า 100 เมตร จะใช้ค่าความสำคัญของประเภทที่ดินนั้นเลยและถ้าไม่ใช้ทั้งสองจะทำการคำนวนโดยใช้สูตร weight * math.exp(-distance / decay_factor) และส่งค่าคะแนนออกไปที่ฟังก์ชัน calculate_land_interest_score โดยเก็บไว้ที่ scores จากนั้นจะทำการบวกคะแนนความหลายหลายของประเภทที่ดินเป็นคะแนนพิเศษและหลังจากนั้นจะนำ scores มารวมกันและส่งไปที่การทำงานหลักโดยเก็บไว้ที่ raw_score และนำคะแนนที่ได้นั้นทำการ normalize ให้อยู่ในช่วง 0 ถึง 10 โดยสูตรที่ใช้จะเป็น (score / MAX_POSSIBLE_SCORE) * 10 โดย MAX_POSSIBLE_SCORE จะคิดจาก WEIGHT_MAPPING มากที่สุด

5 อันดับแรกเท่านั้น จากนั้นจะนำคะแนนที่ทำการ normalize ที่เก็บไว้ตัวแปร normalized_score นำเข้าสู่ฟังก์ชัน get_interest_level เพื่อแบ่งระดับความน่าสนใจของที่ดินนั้นโดยจะแบ่งดังนี้

- น้อยกว่า 4 ไม่น่าสนใจ
- น้อยกว่า 6 ปานกลาง
- น้อยกว่า 8 น่าสนใจ
- มากกว่าหรือเท่ากับ 8 ขึ้นไป น่าสนใจมาก

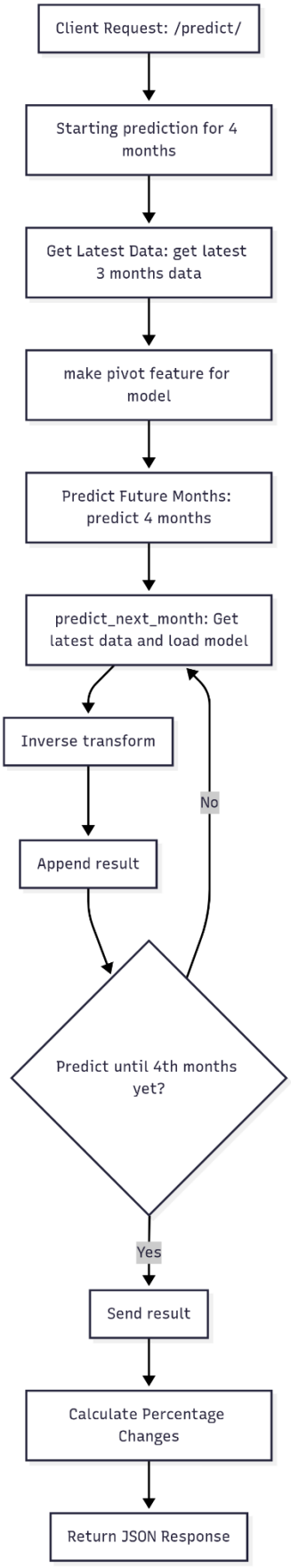
จากนั้นจะนำผลลัพธ์รวมเป็น JSON และส่งไปแสดงผลที่ frontend โดยรวมข้อมูลดังนี้ "land_id","raw_score","normalized_score","interest_level"


Client Request
Get database connection
Fetch land data
Land data found?
Yes
No
Fetch landmark places
Return 404 Error
Calculate land interest score
Handle exception
Group landmark type: choose 2 per type
Return 500 error
calculate type score
Add diversity bonus
Sum score
Normalize Score
Get Interest Level
Return JSON Response

5. ส่วนของระบบการทำนายราคาที่ดินเฉลี่ย

หลังจากที่ git clone https://github.com/LPPACEGroup/DeployML.git

ให้เข้าไปที่โฟลเดอร์ชื่อ Deploy และจะมีไฟล์ API สำหรับการรันโมเดลชื่อ model_service.py โดย API นี้จะรับค่าจำนวนเดือนที่ต้องการทำนายอีกกี่เดือนข้างหน้า โดยระบบนี้จะเริ่มจากฟังก์ชัน get_latest_data(n_months=3) เป็นฟังก์ชันที่จะเรียกข้อมูลสำหรับการทำนายจากฐานข้อมูลของทุกเขตมาทั้งหมด 3 เดือนล่าสุดแล้วและส่งค่ากลับมาที่ตัวแปร latest_data เพื่อนำเข้าสู่ฟังก์ชัน prepare_data ซึ่งทำหน้าที่นำ feature ทุกตัวแบ่งออกเป็นของแต่ละเขตอย่างชัดเจนเพื่อใช้สำหรับการเข้าสู่โมเดลหลังจากนั้นทำการนำข้อมูล NA ออกแล้วส่งค่าเก็บไว้ที่ตัวแปร prepared_data เพื่อนำเข้าสู่ฟังก์ชัน predict_future_months พร้อมกับจำนวนเดือนที่ต้องการทำนาย โดยจะทำหน้าที่แบ่งข้อมูลทีละ 3 แถวและเลื่อนไปที่ละ 1 แถวเพื่อเข้าสู่ฟังก์ชัน predict_next_month โดยจะนำข้อมูลเข้าพร้อมกับโมเดลและตัวสเกลเลอร์ที่ทำการบันทึกเก็บไว้บน server เพื่อทำการทำนาย 1 เดือนถัดไปและ inverse transform ค่าการทำนาย และส่งกลับมาที่ตัวแปร y_pred และเก็บไว้ในตัวแปร predictions และวนลูปจนทำนายครบตามเดือนที่กำหนด จากนั้นจะนำข้อมูลที่ถูกบันทึกลงใน predictions และนำผลลัพธ์การทำนายราคาที่ดินเฉลี่ยของแต่ละเขตมาคำนวณ percentage_changes และส่งกลับไปที่ frontend เป็น JSON


Client Request: /predict/
Starting prediction for 4 months
Get Latest Data: get latest 3 months data
make pivot feature for model
Predict Future Months: predict 4 months
predict_next_month: Get latest data and load model
Inverse transform
Append result
Predict until 4th months yet?
No
Yes
Send result
Calculate Percentage Changes
Return JSON Response