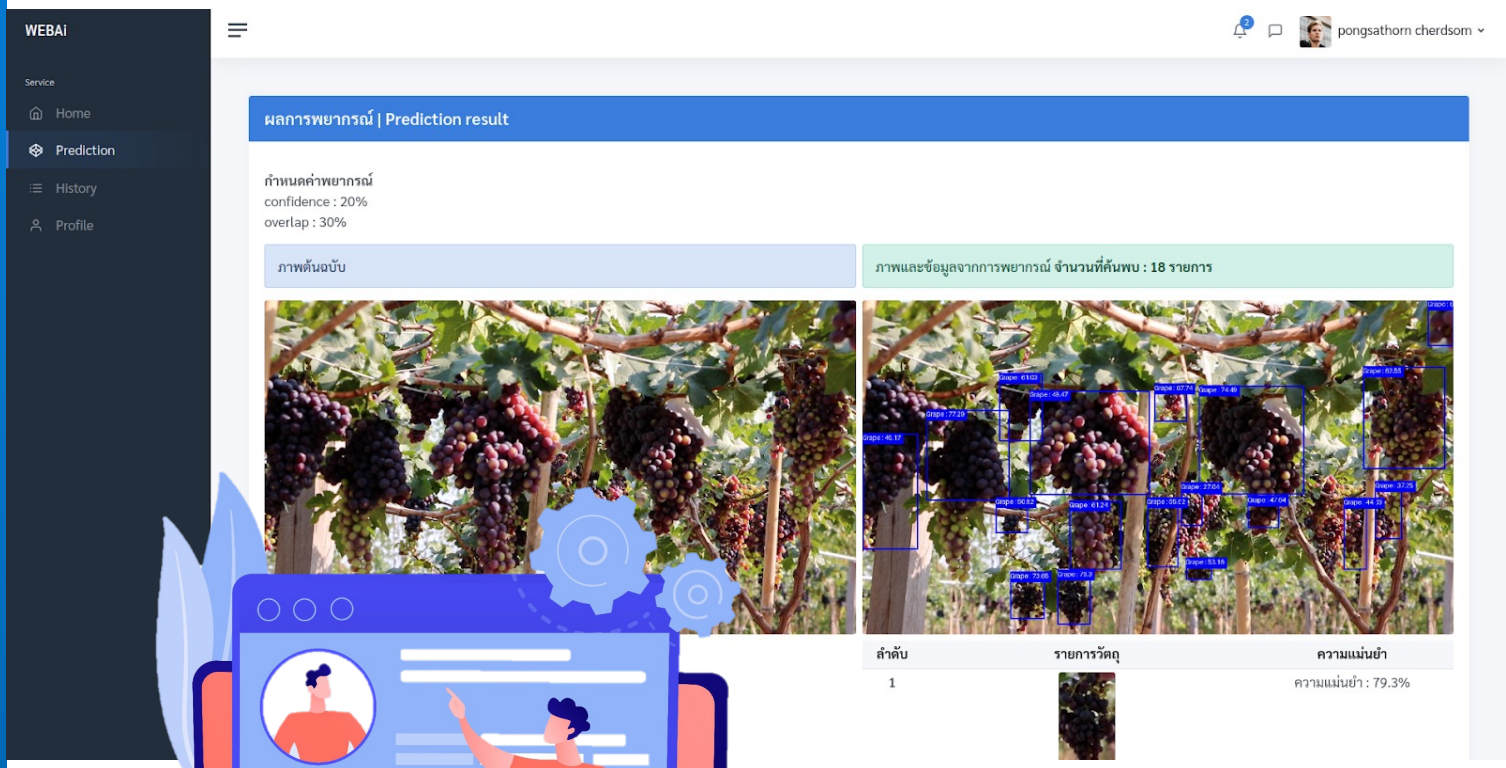


ฉบับ WORKSHOP

พัฒนาเว็บแอพพลิเคชั่นด้วยจังโก้ (Django) ระบบตรวจจับและตรวจนับองุ่น

Web Development with Django (Workshop)

การพัฒนาเว็บแอพพลิเคชั่นด้วย Django ฉบับ Workshop กับการพัฒนาระบบเพื่อเชื่อมโยง AI. ที่พัฒนาโดย Roboflow เพื่อให้บริการสำหรับการตรวจจับและตรวจนับองุ่น และนำไปติดตั้งเพื่อใช้งานจริง (Deploy)

PONGSATHORN CHEDSOM
Digitally signed by PONGSATHORN CHEDSOM
Date: 2024.01.31 07:51:22 +07'00'

*ฉบับพื้นฐาน | **Basic***
“ลองเรียนรู้ ไม่รักก็เปลี่ยนใหม่”

เขียนโดย
พงษ์ศธร เชิดสม
me@pongthorn.in.th

บทนำก่อนเริ่มอ่าน

เอกสารฉบับนี้ประกอบการบรรยายในรูปแบบการทำ Workshop
ในหัวข้อ Web development and AI with Python (Django + Roboflow)

ชื่อเรื่อง
การพัฒนาเว็บแอพพลิเคชั่นด้วยจังโก้ (Django) ฉบับ WORKSHOP
ระบบตรวจจับและตรวจนับองุ่น
Web Development with Django (Workshop)

เนื้อหา
- การติดตั้งโปรแกรมและเตรียมเครื่องมือ
- การจัดเตรียมโครงสร้างระบบ
- การเขียนระบบสำหรับการจัดการภายใน
- การเขียนระบบสำหรับการเชื่อมโยงข้อมูลผ่าน API กับ AI Model
- การนำระบบไปติดตั้งเพื่อใช้งานจริง

เนื้อหาประกอบการอธิบายอาจจะตกหล่น ไม่สมบูรณ์
หากผู้อ่านมีข้อสงสัย ข้อเสนอแนะ ข้อปรับปรุง
โปรดติดต่อนายพงษ์ธร เชิดสม (ผู้เขียน)
อีเมล me@pongthorn.in.th, pongche@kku.ac.th

พงษ์ศธร เชิดสม
เผยแพร่วันที่ 24 มกราคม 2567
อัพเดตวันที่ 31 มกราคม 2567 07.30 น.

สารบัญเนื้อหา

1. ติดตั้งโปรแกรมและส่วนเสริม..... 4
ติดตั้งโปรแกรม..... 4
ติดตั้งส่วนเสริมของ Visual Studio Code..... 6
2. ติดตั้ง Django..... 7
ขั้นตอนการติดตั้ง..... 7
ตรวจสอบเวอร์ชั่น Django (Check version Django)..... 8
3. ภาพรวมและการทำงานของระบบ..... 8
4. สร้าง Project Django | จุดเริ่มต้นของการเรียนรู้..... 10
ขั้นตอนการสร้าง project django..... 10
ทดสอบ Run server project..... 10
ทดสอบ Run server (ขั้นสูง)..... 11
5. สร้าง APP..... 12
ขั้นตอนการสร้าง app..... 12
ลงทะเบียน app เพื่อให้ django รู้จัก..... 12
6. สร้างหน้า Homepage ของหน้าระบบ (การกำหนด path แบบ static)..... 13
ทดสอบ Run Server..... 14
7. ทดสอบสร้างหน้าเว็บเพิ่มอีก 1 หน้า (การกำหนด path แบบ static)..... 14
ทดสอบ Run Server..... 16
8. กรณีส่งค่าข้อมูลผ่านทาง URL ไปแสดงผล (การกำหนด path แบบ parameter)..... 16
ทดสอบ Run Server..... 17
9. การตั้งค่าเพื่อนำไฟล์ Template html มาใช้งาน..... 18
การตั้งค่า django..... 18
10. การตั้งค่า Static file..... 19
11. การเตรียม File และ Folder Template..... 20
Download file Template..... 20
การเตรียมใช้งาน..... 21
การปรับแต่งและการจัดเตรียมหน้า..... 21
12. การนำไฟล์ Html มาใช้งานกับหน้าเว็บไซต์..... 24
13. การติดตั้งและการเชื่อมต่อฐานข้อมูล..... 27
การติดตั้ง..... 27
กรณีเกิดข้อผิดพลาด (Error)..... 28
การสร้างฐานข้อมูลใน Phpmyadmin..... 28
การตั้งค่า Settings.py..... 29
การเชื่อมต่อฐานข้อมูล..... 30
14. การตั้งค่า Alert message สำหรับการแจ้งเตือน..... 31
การตั้งค่า..... 31

การเรียกใช้งาน....32
15. สร้างหน้าเพจสมัครสมาชิก Register....34
16. สร้างฟังก์ชั่นสำหรับบันทึกผลการสมัครสมาชิก....37
การเช็คเงื่อนไขในการสมัครสมาชิก....37
การเพิ่มข้อมูลผู้ใช้งาน....39
17. สร้างหน้าลงชื่อเข้าใช้งาน Login....43
การตรวจสอบข้อมูลและการ Login....44
สรุปฟังก์ชั่นการตรวจสอบการ login....45
18. สร้างฟังก์ชั่นสำหรับการลงชื่อออกจากระบบ Logout....46
19. การจัดเตรียมหน้าสำหรับสมาชิก (ระบบหลังบ้านสำหรับสมาชิก)....47
การจัดเตรียมส่วนของ Header....47
การจัดเตรียมเมนู Menu....50
การจัดเตรียม Footer....53
20. การแสดงแถบสีฟ้าเมื่อมีการเรียกใช้งานเมนูนั้นๆ (Active menu)....54
21. ตรวจสอบการเรียกใช้งานฟังก์ชั่น สำหรับสมาชิกของระบบ....57
22. การสร้างหน้าเพจ Profile สำหรับแสดงข้อมูลผู้ใช้งาน และการแก้ไขข้อมูล....59
23. สร้างฟังก์ชั่น Update user ข้อมูลผู้ใช้งาน....62
เงื่อนไขทำงาน....62
คำสั่งการอัพเดตข้อมูล....63
อัพเดต URL ของเมนู Profile....64
24. การสร้างฐานข้อมูล....66
25. การสร้างหน้าเพจสำหรับ Prediction....69
26. สร้างฟังก์ชั่นเพื่อส่งรูปภาพไปประมวลผล ผ่านทาง API....72
27. การสร้างหน้าสำหรับแสดงผลการ Prediction (รายการที่เลือก)....78
28. การสร้างหน้าสำหรับแสดงรายการ Prediction....86
อัพเดต URL ของเมนู History....88
29. การสร้างฟังก์เพื่อลบผลการพยากรณ์ Prediction....91
30. การสร้างหน้ารายการผู้ใช้งาน (สำหรับผู้ดูแลระบบ)....95
31. การสร้างฟังก์ชั่นเพื่อลบบัญชีผู้ใช้งาน (สำหรับผู้ดูแลระบบ)....102
32. การสร้างฟังก์ชั่นเพื่ออัพเดตข้อมูลผู้ใช้งาน (สำหรับผู้ดูแลระบบ)....105
33. การสร้างฟังก์ชั่นเพื่อสร้างบัญชีผู้ใช้งาน (สำหรับผู้ดูแลระบบ)....110
34. การสร้างหน้าสำหรับแสดงรายการ Prediction ทั้งระบบ (สำหรับผู้ดูแลระบบ)....114
35. การสร้างฟังก์ชั่นเพื่อลบรายการ Prediction (สำหรับผู้ดูแลระบบ)....121
36. การสร้างหน้าสำหรับแสดงผลการ Prediction (สำหรับผู้ดูแลระบบ)....124
37. การเตรียมโปรเจคก่อนนำไปติดตั้ง....131
ส่งออกรายการ Libraries (Export lib.)....131
ปรับแต่งค่าฐานข้อมูลสำหรับใช้งานจริง....132

ปรับแต่งไฟล์ Settings.py ในส่วนอื่นๆ..... 133
รวมไฟล์ทั้งหมดของโปรเจคให้อยู่ในไฟล์ .zip..... 133
38. การติดตั้งระบบเพื่อใช้งานจริง (Deploy)..... 133
สร้าง Droplets (หรือสร้างเครื่อง VM)..... 134
การอัพโหลดไฟล์โปรเจคและการแตกไฟล์..... 140
จัดเตรียมและตั้งค่า Ubuntu Server 22.04..... 143
เริ่มติดตั้งโปรเจค webai..... 144
การตั้งค่าให้เรียกใช้งานผ่าน Domain..... 148

1. ติดตั้งโปรแกรมและส่วนเสริม

ติดตั้งโปรแกรม

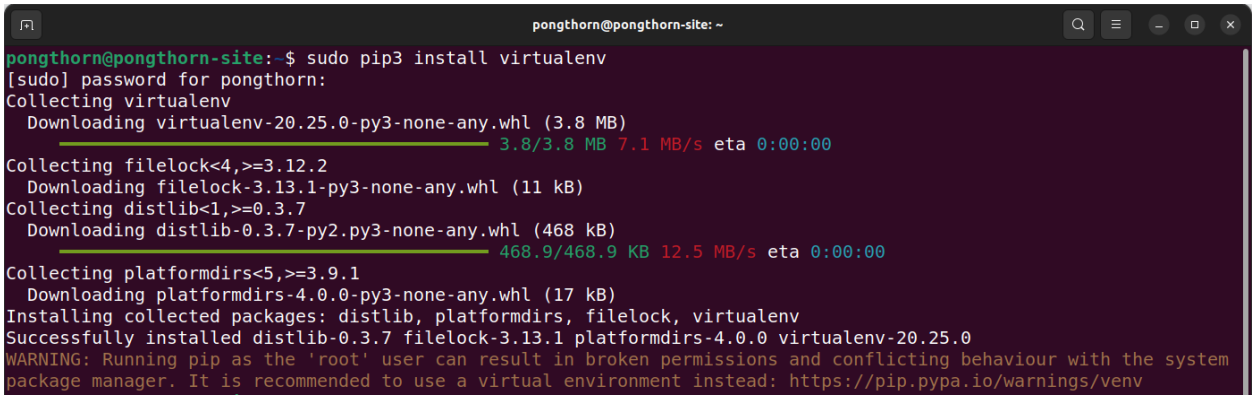
- Xampp สำหรับใช้งาน Mysql และใช้ phpmyadmin ในการบริหารจัดการ database เพิ่มเติม ในระหว่างการพัฒนาระบบ
    - สำหรับ Ubuntu
    - สำหรับ windows
- Visual Studio Code

  ดาวน์โหลดและติดตั้งได้ที่ https://code.visualstudio.com/
- Python
    - สำหรับ Ubuntu

      คำสั่งติดตั้ง : sudo apt install python3
- Python pip
    - สำหรับ Ubuntu

      คำสั่งติดตั้ง : sudo apt install python3-pip
- Virtualenv

  คำสั่งติดตั้ง : pip3 install virtualenv



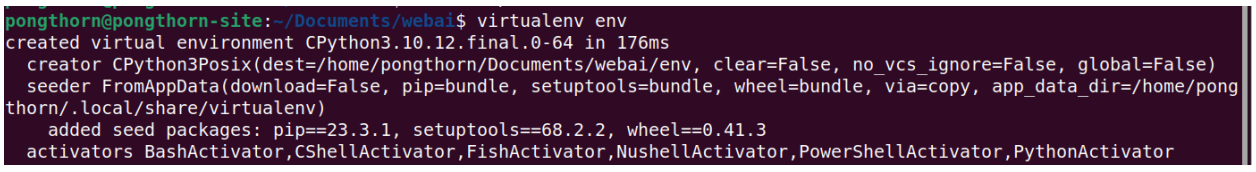
- **สำหรับ Ubuntu**

  คำสั่งสร้าง Virtual Environments

  virtualenv env หรือ

  python3 -m virtualenv env

  เมื่อสร้างเสร็จสมบูรณ์จะมี folder ที่เป็นชื่อเดียวกับ emv ที่สร้างขึ้น



คำสั่ง Activate Virtual Environments

source env/bin/activate

เมื่อสั่ง activate แล้ว จะแสดง (env) หน้าคอมมานไลน์ (Command line) บรรทัดปัจจุบัน

```
pongthorn@pongthorn-site:~/Documents/webai$ source env/bin/activate
(env) pongthorn@pongthorn-site:~/Documents/webai$
```

คำสั่ง Deactivate Virtual Environments

deactivate

คำสั่ง deactivate เมื่อ deactivate แล้ว (env) หน้าคอมมานไลน์ (Command line) บรรทัดปัจจุบันจะหายไป

```
(env) pongthorn@pongthorn-site:~/Documents/webai$ deactivate
pongthorn@pongthorn-site:~/Documents/webai$
```

- **สำหรับ Windows**

  คำสั่งสร้าง Virtual Environments

  virtualenv <ชื่อโปรเจค>

  คำสั่ง Activate Virtual Environments

  cd folder env

  activate โดยการ Scripts\activate

  คำสั่ง Deactivate Virtual Environments

  cd folder env

  deactivate โดยการ Scripts\deactivate.bat

คำแนะนำ

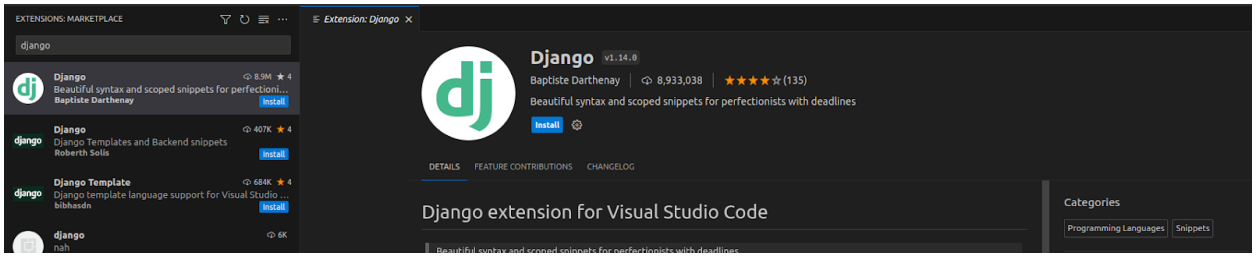
- ใช้ keyboard : Tab ช่วยเติมคำสั่งให้
- การใช้งาน env ต้องสังเกต (env name) ทุกครั้ง

ติดตั้งส่วนเสริมของ Visual Studio Code

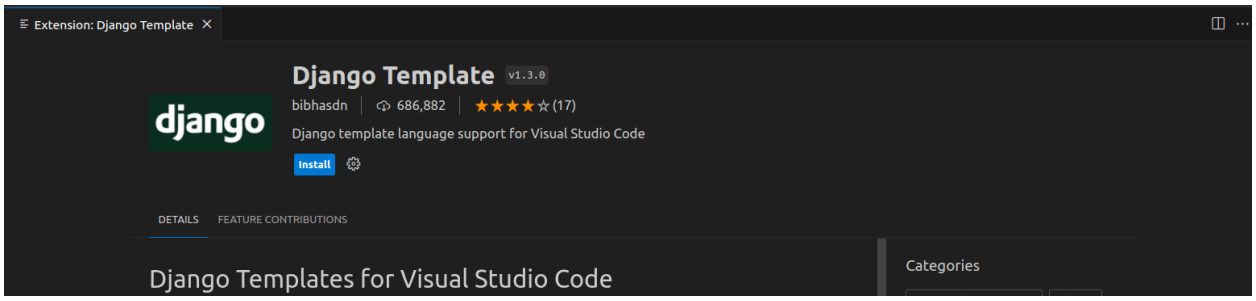
ประกอบไปด้วย

- Python

- Django



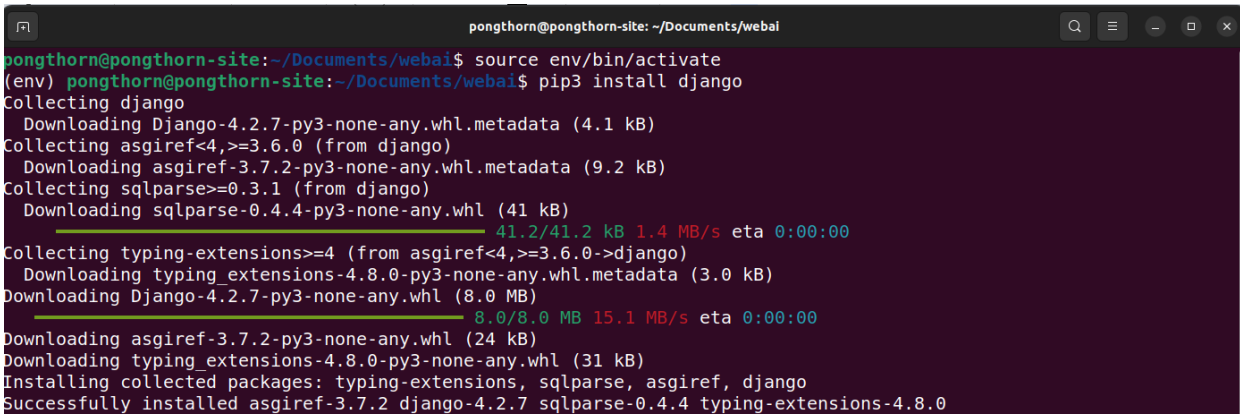
- Django Template



2. ติดตั้ง Django

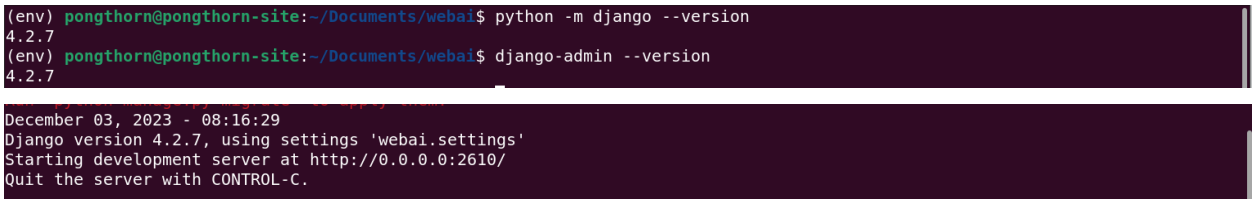
ขั้นตอนการติดตั้ง

1. Activate env ตามวิธีด้านบน
2. ใช้คำสั่งในการติดตั้ง
   - pip3 install django จะได้เป็นเวอร์ชั่นล่าสุด **(แนะนำ)** หรือ
   - กรณีที่ต้องการเวอร์ชั่น ให้ระบุเป็น pip3 install django==3.2
   - กรณีไม่สามารถติดตั้งผ่าน pip ไม่ได้ ใช้คำสั่ง git clone https://github.com/django/django.git



ตรวจสอบเวอร์ชั่น Django (Check version Django)

1. python -m django --version
2. django-admin --version
3. python manage.py runserver ช่วงที่ runserver
4. pip3 list ที่ activate env แล้ว



3. ภาพรวมและการทำงานของระบบ

ชื่อระบบ : ระบบตรวจจับและตรวจนับองุ่น

การทำงานภาพรวมของระบบ

ระบบสำหรับการส่งภาพไร่องุ่นเพื่อนำไปตรวจหาองุ่นในภาพโดยใช้ model ในการตรวจหาองุ่นจาก **Roboflow** โดยใช้รูปแบบ detection model สามารถศึกษาการเตรียมข้อมูลและขั้นตอนการพัฒนาแบบจำลอง (model) ได้ที่คู่มือ **สร้าง AI Model จาก Roboflow (A Guide to Creating Models with Roboflow)** จาก URL : https://pongthorn.in.th/handbook-aimodel-roboflow/ ซึ่งให้ศึกษาในบท **การทำ Object Detection** โดยฝั่งของระบบจะพัฒนาส่วนของการจัดการผู้ใช้งาน จัดการการพยากรณ์ข้อมูล ระบบข้อมูลพื้นฐานเช่น การลงชื่อเข้าใช้งาน การลงชื่อออกจากระบบ การแก้ไขข้อมูล การเพิ่มข้อมูล การลบข้อมูล โดยใช้ฐานข้อมูล Mysql

*** การเลือกใช้ฐานข้อมูล mysql ผู้เขียนเห็นว่าการเลือกฐานข้อมูลนี้เนื่องจากสามารถนำไปประยุกต์ใช้งานในรูปแบบอื่นๆ ได้ และการเรียนในระดับมหาวิทยาลัยของนักศึกษาสาขาคอมพิวเตอร์โดยพื้นฐานมีการเรียนรู้ฐานข้อมูลนี้อยู่แล้ว*

ผู้ใช้งานระบบ

- ผู้ดูแลระบบ

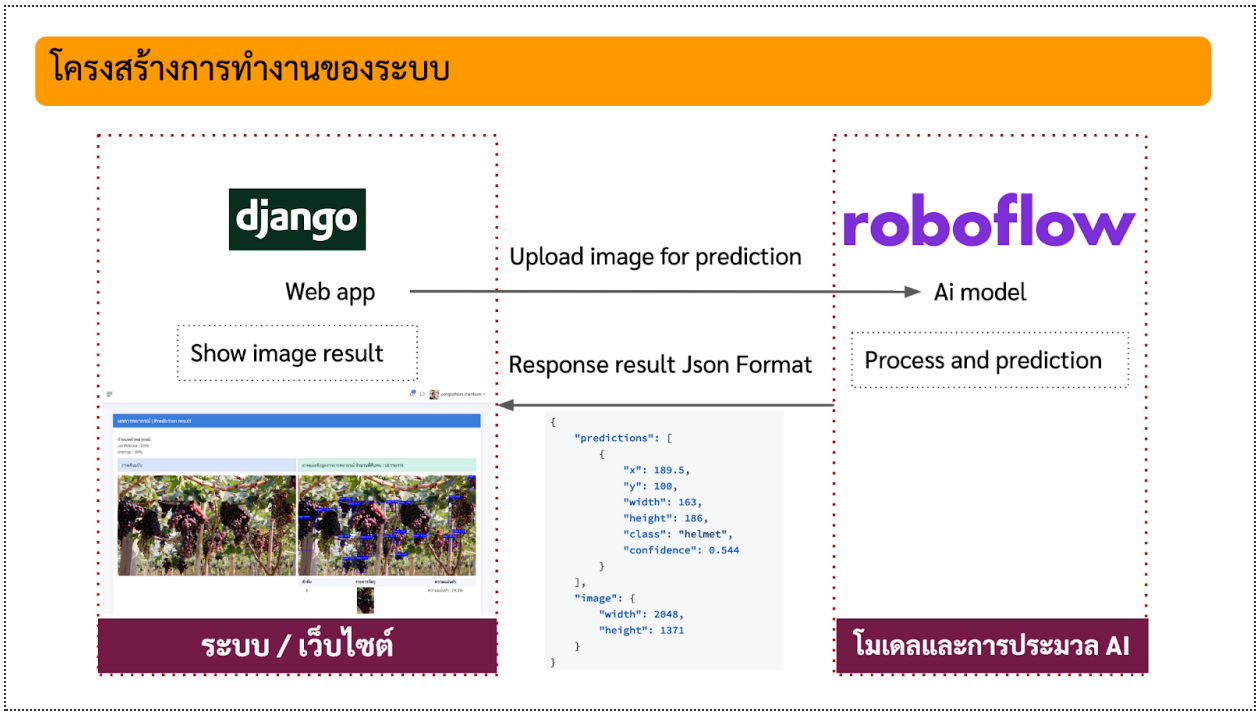
- สมาชิก

การทำงานของระบบ (แบ่งตามสมาชิก)

- ผู้ดูแลระบบ
    - รายการสมาชิก
        - แสดงรายการสมาชิก
        - เพิ่มสมาชิก
        - แก้ไขข้อมูลสมาชิก
        - ลบสมาชิก
    - รายการพยากรณ์ข้อมูล
        - ดูรายละเอียดและผลการพยากรณ์
        - ลบข้อมูลการพยากรณ์
- สมาชิก
    - พยากรณ์ข้อมูล (ส่งรูปภาพองุ่นไปตรวจหาองุ่น)
        - กำหนดข้อมูลอื่นๆ เช่น รูปแบบการพยากรณ์ ความแม่นยำ
    - รายการพยากรณ์ (เฉพาะของตัวเอง)
        - ดูรายละเอียดและผลการพยากรณ์
        - ลบข้อมูลการพยากรณ์
    - แก้ไขข้อมูลและรหัสผ่านผู้ใช้งาน

เครื่องมือ

- Os : Ubuntu (for dev in docs)
- Code editer: VS code (Visual Studio Code)
- Database : Mysql
- Framwork Backend : Django (Python)
- Framwork Frontend : Bootstrap (Html + Css)
- Roboflow : software for create ai model



4. สร้าง Project Django | จุดเริ่มต้นของการเรียนรู้

ขั้นตอนการสร้าง project django

1. Activate env ตามวิธีด้านบน
2. django-admin startproject <project-name>
   โปรเจคนี้ใช้คำสั่ง
   django-admin startproject webai

```
(env) pongthorn@pongthorn-site:~/Documents/webai$ django-admin startproject webai
(env) pongthorn@pongthorn-site:~/Documents/webai$
```

3. เมื่อสร้างเสร็จสมบูรณ์จะมี folder project เพิ่มขึ้นมา โดยใช้คำสั่ง ls จะได้ชื่อเดียวกับ project

```
(env) pongthorn@pongthorn-site:~/Documents/webai$ ls
env  webai
```

ทดสอบ Run server project

1. ให้เข้าไปใน folder project webai ถ้าตามตัวอย่างจะต้อง cd webai อีกชั้น ซึ่งจะเป็น (/webai/webai)

```
(env) pongthorn@pongthorn-site:~/Documents/webai$ cd webai/
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$
```

2. ทดสอบดูไฟล์ในโปรเจค โดยใช้คำสั่ง ls จะแสดงดังภาพ (ที่มีไฟล์ เช่น manage.py)

```
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ ls
manage.py  webai
```

3. คำสั่งรัน python3 manage.py runserver

```
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth,
 contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 03, 2023 - 08:12:44
Django version 4.2.7, using settings 'webai.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

รันได้เสร็จสมบูรณ์
เมื่อต้องการออกจากการ runserver ให้ใช้ Ctrl + C เพื่อจบการรันคำสั่ง

ทดสอบ Run server (ขั้นสูง)

- กรณีที่ต้องการเปลี่ยน port ในการรัน ให้ใช้คำสั่ง python3 manage.py runserver 2610

```
^C(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ python3 manage.py runserver 2610
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth,
 contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 03, 2023 - 08:14:01
Django version 4.2.7, using settings 'webai.settings'
Starting development server at http://127.0.0.1:2610/
Quit the server with CONTROL-C.
```

- กรณีที่ต้องการให้เครื่องอื่นๆ ภายในเครือข่ายสามารถเข้าถึงการรันโปรเจคได้ สามารถกำหนดไอพีแอสเดรสได้ โดยระบุเป็นเบอร์ 0.0.0.0:2610 และตามด้วยเบอร์ port → python3 manage.py runserver 0.0.0.0:2610

```
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ python3 manage.py runserver 0.0.0.0:2610
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth,
 contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 03, 2023 - 08:16:29
Django version 4.2.7, using settings 'webai.settings'
Starting development server at http://0.0.0.0:2610/
Quit the server with CONTROL-C.
```

5. สร้าง APP

ขั้นตอนการสร้าง app

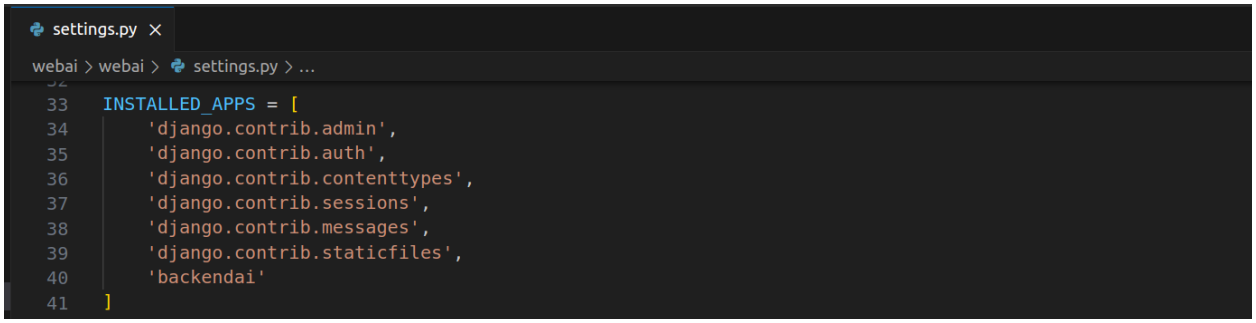
1. เข้าไปที่ตำแหน่ง root project (ที่มีไฟล์ manage.py)
2. python3 manage.py startapp <app-name>
   ในโปรเจคนี้ใช้คำสั่ง
   python3 manage.py startapp backendai

```
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ python3 manage.py startapp backendai
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$
```

ลงทะเบียน app เพื่อให้ django รู้จัก

ต้องทำทุก ๆ ครั้งที่มีการสร้าง app ใหม่

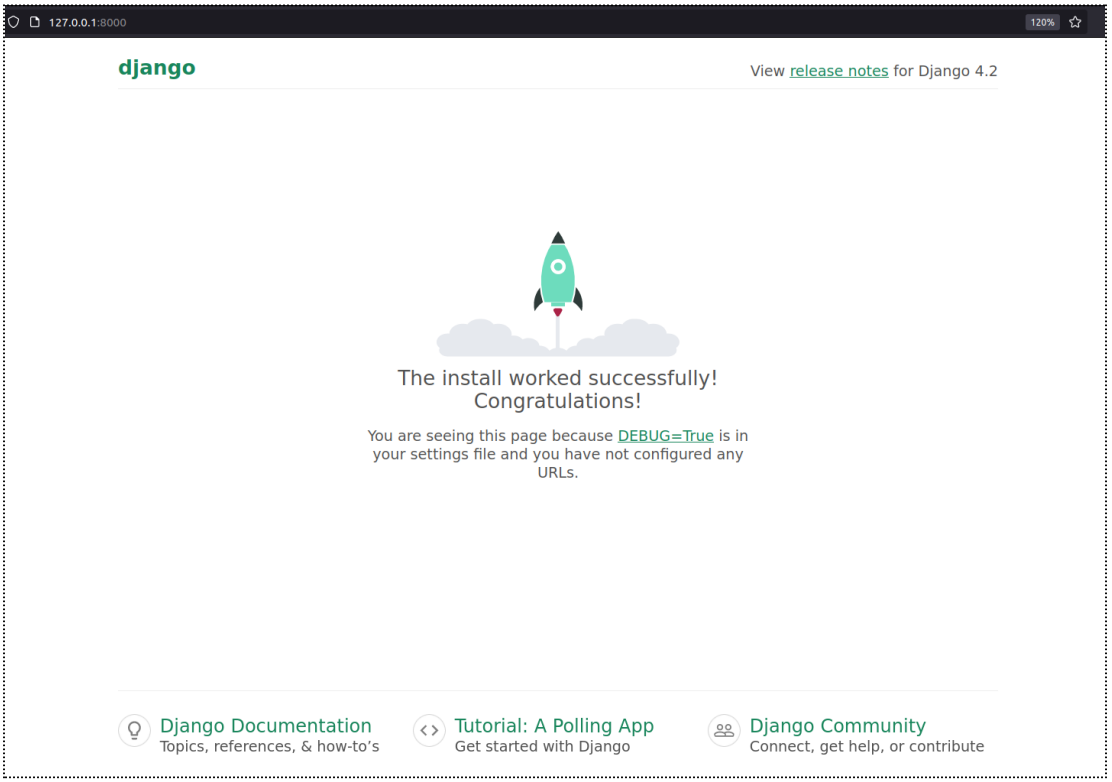
1. ไปที่ไฟล์ settings.py อยู่ใน folder project เท่านั้น ไม่มีใน folder app
2. ในส่วนของ INSTALLED_APPS ให้เพิ่มชื่อ app ดังภาพ ชื่อ app ที่สร้างขึ้น คือ backendai



ทดสอบ Run server project

python3 manage.py runserver

http://127.0.0.1:8000/



6. สร้างหน้า Homepage ของหน้าระบบ (การกำหนด path แบบ static)

ซึ่งจะทำแบบวิธี Function views โดยสร้าง url จาก urls.py ของโปรเจค (ตัวอย่างคือ webai)

File : urls.py (อยู่ใน webai)

เริ่มจากการ import views ที่อยู่ใน app (backendai) มาใช้งาน โดยใช้คำสั่ง from backendai import views หมายถึงให้ import file views.py ที่อยู่ใน folder backendai มาใช้งาน และในส่วนของ urlpatterns ให้กำหนดเป็น

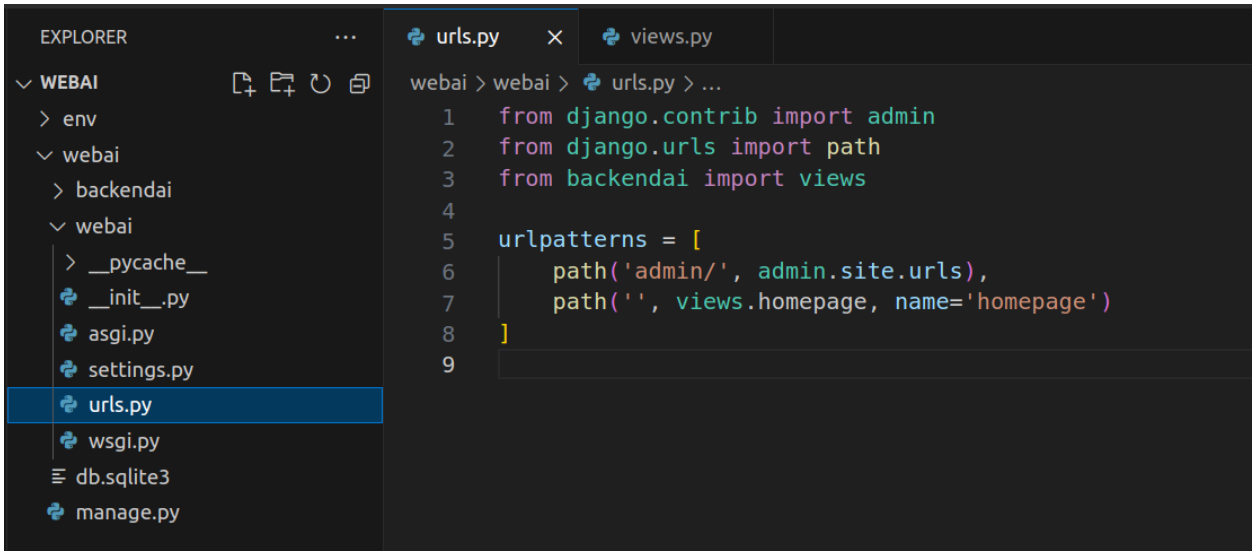
```python
path('', views.homepage, name='homepage')
```

ซึ่งประกอบด้วย 3 ส่วน ได้แก่

'' หมายถึง url ที่ไม่มี parameter ต่อท้าย หากหนดแบบนี้ถ้ามีการเรียกใช้งานจะเรียก url http://127.0.0.1:8000/ โดยไม่มีค่าตามหลัง /

views.homepage หมายถึง เรียกใช้งาน function homepage จาก views ของ backendai ที่มีการ import เข้ามา

name หมายถึง การกำหนดชื่อให้กับ url หรือ routing เส้นนี้ จะนำไปใช้งานกับหน้า html หรือในฟังก์ชั่นอื่นๆ

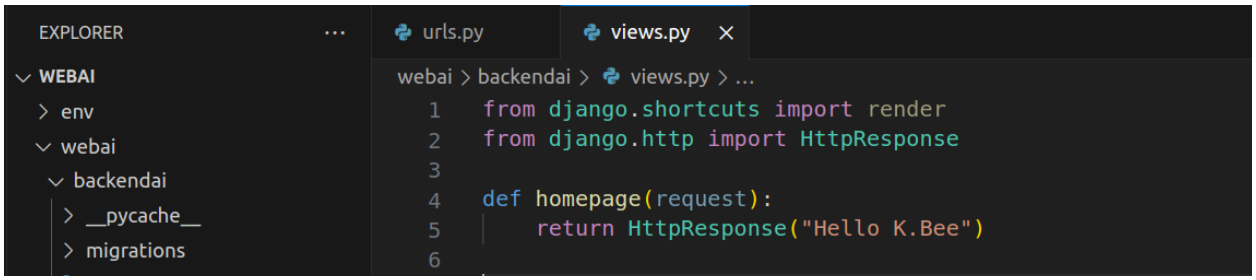


File : views.py (อยู่ใน backendai)

ให้เพิ่มคำสั่ง from django.http import HttpResponse เพื่อใช้งานคำสั่ง HttpResponse ในการส่งค่ากลับไปแสดงที่หน้าเว็บไซต์

สร้าง function ชื่อว่า homepage ที่ส่งผ่านค่า request เข้าไปในฟังก์ชั่นด้วย และส่งค่ากลับไปทางหน้าเว็บ Hello K.Bee

```python
from django.http import HttpResponse

def homepage(request):
    return HttpResponse("Hello K.Bee")
```



ทดสอบ Run Server

python3 manage.py runserver
เรียกใช้งาน http://127.0.0.1:8000/

7. ทดสอบสร้างหน้าเว็บเพิ่มอีก 1 หน้า (การกำหนด path แบบ static)

File : urls.py (อยู่ใน webai)

เพิ่มเส้นทาง

url โดยกำหนด path เป็น 'user/'

views เรียกใช้งาน function homepageuser

name กำหนดชื่อเป็น pageuser (สามารถกำหนดให้แตกต่างจากชื่อของ function ได้) การเรียกใช้งานจะเป็น http://127.0.0.1:8000/user/

```python
path('user/', views.homepageuser, name='pageuser'),
```

urls.py views.py

webai > webai > urls.py > ...

```python
from django.contrib import admin
from django.urls import path
from backendai import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.homepage, name='homepage'),
    path('user/', views.homepageuser, name='pageuser')
]

```

File : views.py (อยู่ใน backendai)

สร้าง function ชื่อว่า homepageuser ที่ส่งผ่านค่า request เข้าไปในฟังก์ชั่นด้วย และส่งค่ากลับไปทางหน้าเว็บ <h1>Your call page => pageuser</h1> ที่สามารถใช้ tag html ร่วมด้วย

Python

```python
def homepageuser(request):
      return HttpResponse("<h1>Your call page => pageuser</h1>")
```

urls.py views.py

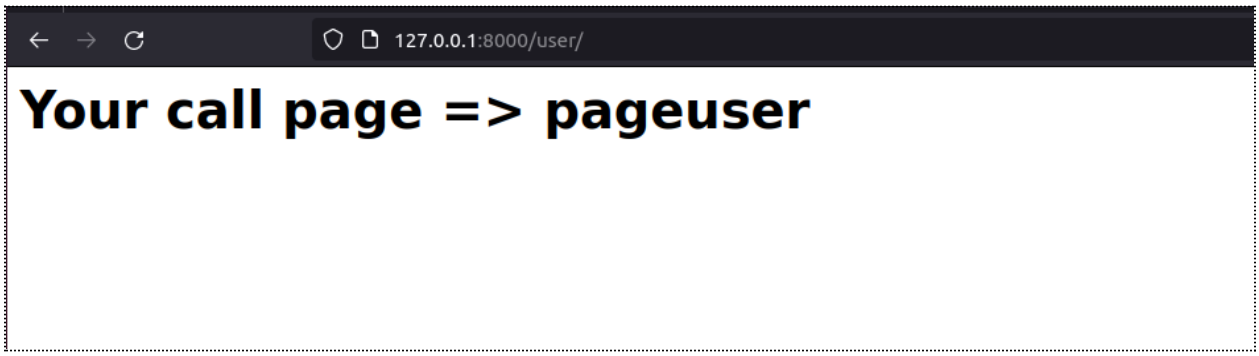
webai > backendai > views.py > ...

```python
from django.shortcuts import render
from django.http import HttpResponse

def homepage(request):
    return HttpResponse("Hello K.Bee")

def homepageuser(request):
    return HttpResponse("<h1>Your call page => pageuser</h1>")
```

ทดสอบ Run Server

python3 manage.py runserver

เรียกใช้งาน http://127.0.0.1:8000/user/



8. กรณีส่งค่าข้อมูลผ่านทาง URL ไปแสดงผล (การกำหนด path แบบ parameter)

File : urls.py (อยู่ใน webai)

เพิ่มเส้นทาง

url โดยกำหนด path เป็น 'user/<str:username>/' str หมายถึง type ของตัวแปร และ username คือชื่อของตัวแปร ที่จะนำไปเรียกใช้ใน function ฝั่งของ views

views เรียกใช้งาน function pageuserinfo

name กำหนดชื่อเป็น pageuserinfo

การเรียกใช้งานจะเป็น http://127.0.0.1:8000/user/<parameters>/ เช่น http://127.0.0.1:8000/user/pongthorn/

```python
path('user/<str:username>/, views.pageuserinfo, name='pageuserinfo')
```

ศึกษาเรื่อง URL เพิ่มเติมได้ที่ https://docs.djangoproject.com/en/4.2/topics/http/urls/

File : views.py (อยู่ใน backendai)

สร้าง function ชื่อว่า pageuserinfo ที่ส่งผ่านค่า request และ username (เป็นชื่อเดียวกันที่กำหนดใน url) ที่มาจาก url เข้าไปในฟังก์ชั่นด้วย และส่งค่า กลับไปทางหน้าเว็บ ด้วยคำสั่ง f"Hello K.{username}"

```python
def pageuserinfo(request,username):
    return HttpResponse(f"Hello K.{username}")
```

f-Strings สามารถเรียกใช้งานจาก python version ตั้งแต่ 3.6 ขึ้นไป โดยสามารถผ่านค่า variable เข้าไปโดยใช้เครื่องหมาย { }

urls.py views.py

webai > backendai > views.py > ...

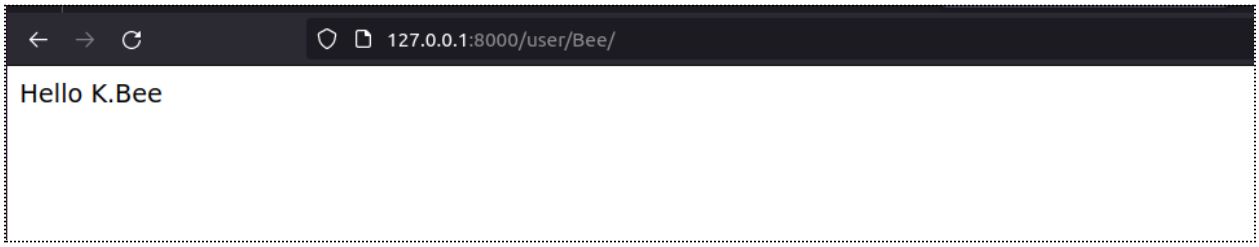
```python
from django.shortcuts import render
from django.http import HttpResponse

def homepage(request):
    return HttpResponse("Hello K.Bee")

def homepageuser(request):
    return HttpResponse("<h1>Your call page => pageuser</h1>")

def pageuserinfo(request,username):
    return HttpResponse(f"Hello K.{username}")

```

ทดสอบ Run Server

python3 manage.py runserver

เรียกใช้งาน http://127.0.0.1:8000/user/pongthorn/

เรียกใช้งาน http://127.0.0.1:8000/user/Bee/

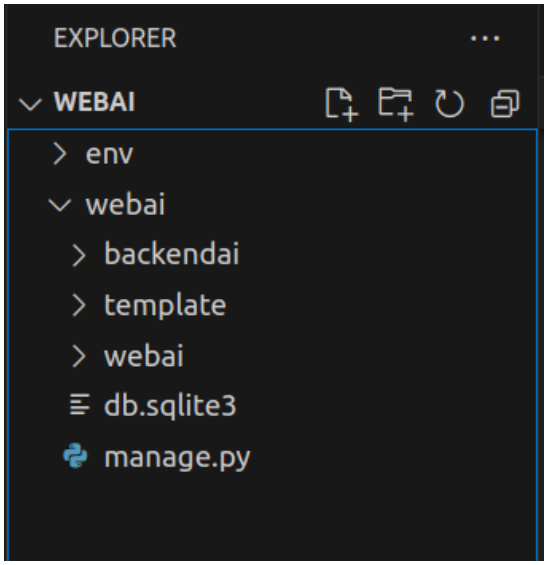


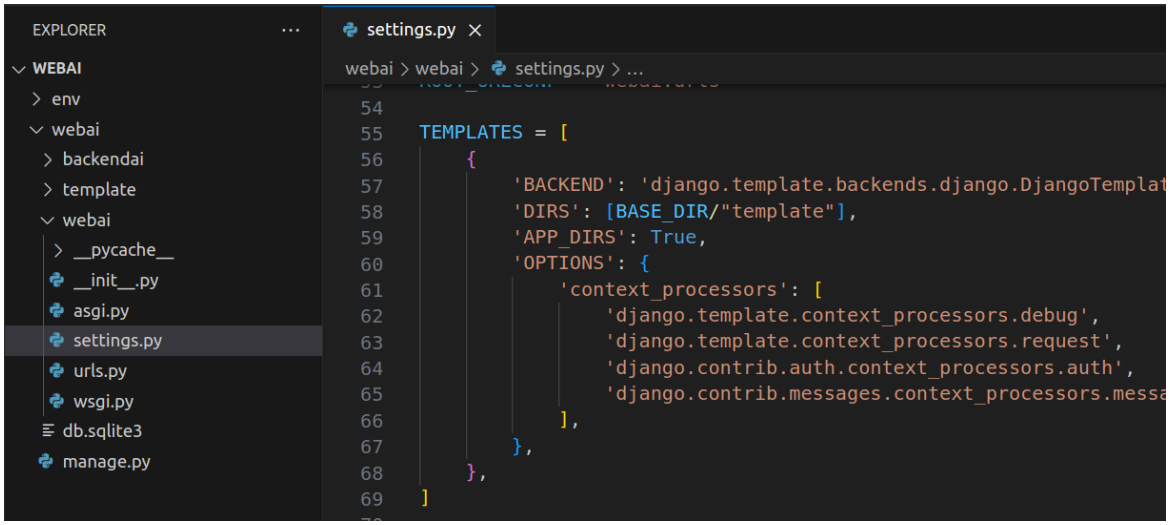
9. การตั้งค่าเพื่อนำไฟล์ Template html มาใช้งาน

การตั้งค่า django

1. สร้าง folder ชื่อว่า template ในการเก็บข้อมูลไฟล์ Html ทั้งหมดของระบบ ในตำแหน่งเดียวกับ folder project และ app folder
2. กำหนด root path ของ template ไปที่ไฟล์ settings.py ที่อยู่ใน folder project
3. ในส่วนของ TEMPLATES →'DIRS': [BASE_DIR/"template"],
    - BASE_DIR คือตัวแปรที่กำหนดเป็น root path ของ project ซึ่งถูกกำหนดใน settings.py บรรทัดที่ 15-16
    - template คือชื่อ folder ที่สร้างขึ้น สามารถใช้ชื่ออื่นได้

ตำแหน่งของ folder จะอยู่ในดังภาพ

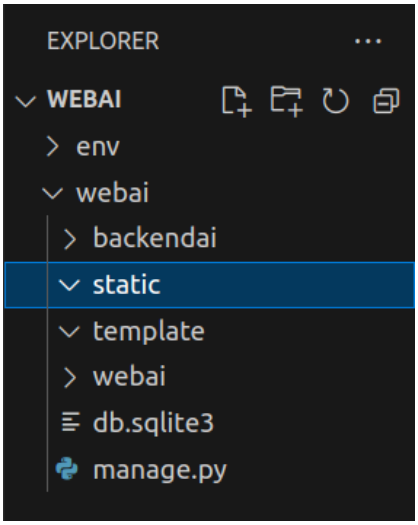




10. การตั้งค่า Static file

1. สร้าง folder ที่ชื่อว่า static ใช้สำหรับเก็บไฟล์ เช่น css js image หรือสามารถเก็บไฟล์ที่ทำการ upload ขึ้นมาบนระบบหรือเว็บไซต์ได้ ในระดับเดียวกับ template หรือ folder project และ app folder (สามารถสร้างชื่ออื่นได้)
2. ระบุตำแหน่งของ path static folder ให้กับ django ได้รู้จัก ไปที่ settings.py ที่อยู่ใน folder project
3. ในส่วนของ STATIC_URL ให้กำหนดเป็นตำแหน่งของ folder static ที่ได้สร้างไว้ ซึ่ง django ได้กำหนดไว้เป็นค่าตั้งต้นให้แล้ว ระบุเป็น 'static/' และเพิ่ม STATICFILES_DIRS = [BASE_DIR / 'static/'] ซึ่ง static คือ folder ที่ได้สร้างไว้

ตำแหน่งของ folder จะอยู่ในดังภาพ



settings.py

webai > webai > settings.py > ...

```
116  # Static files (CSS, JavaScript, Images)
117  # https://docs.djangoproject.com/en/4.2/howto/static-files/
118
119  STATIC_URL = 'static/'
120  STATICFILES_DIRS = [BASE_DIR/'static']
121
```

ศึกษาเพิ่มเติมที่ https://docs.djangoproject.com/en/4.2/howto/static-files/

11. การเตรียม File และ Folder Template

Download file Template

Download file ที่เตรียมไว้ให้ได้ที่ https://kku.world/templateadmin หรือ https://drive.google.com/file/d/1haTafYrwteTNTjXVT2YM5W0NI30Tl9gN ซึ่งเป็น template

adminkit เวอร์ชั่นฟรีจากเว็บไซต์ https://adminkit.io/ เพื่อให้เกิดความสะดวกในการเรียนรู้และการใช้งาน

เมื่อดาวน์โหลดไฟล์แล้ว นำมาแตกไฟล์ จะได้ไฟล์ 1 file และ 4 folder ดังภาพ

Folder

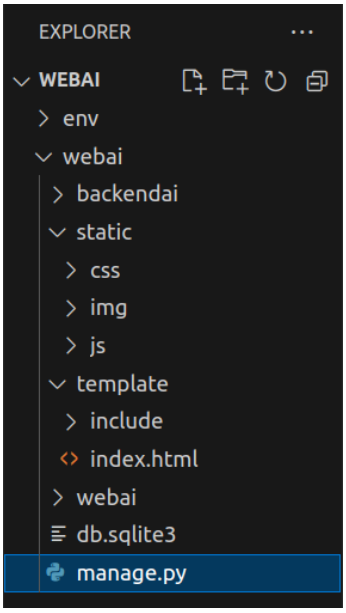
- css
- img
- include (ซึ่งจะเป็นส่วนประกอบของหน้าเว็บ) ได้แก่
  - menu.html
  - footer.html
  - header.html
- js

File

- index.html
- login.html
- register.html

การเตรียมใช้งาน

- คัดลอก folder 3 folder (css, img, js) ไปไว้ที่ folder static ที่สร้างไว้ในขั้น ตอนก่อนหน้า
- คัดลอก 1 folder (include) และ file index.html, login.html, register.html ไปไว้ที่ folder template ที่สร้างไว้ในขั้นตอนก่อนหน้า



การปรับแต่งและการจัดเตรียมหน้า

index.html , header.html

- แก้ไขชื่อไฟล์ index.html จากที่เตรียมไว้ในขั้นตอนก่อนหน้าเป็นไฟล์ base.html ซึ่งจะเป็นไฟล์ตั้งต้นในการใช้งานในทุกๆ หน้าเว็บของระบบ
- ให้ทำการเรียกใช้ {% load static %} บรรทัดที่ 1 ของไฟล์ base.html เพื่อให้สามารถเรียกใช้งานไฟล์ต่างๆใน folder static ได้
- การเรียกใช้งานไฟล์ในรูปแบบของ django ในบรรทัดที่ 12,15,45
    - บรรทัดที่ 12
      เดิม <link rel="shortcut icon" href="img/icons/icon-48x48.png" />
      ใหม่ <link rel="shortcut icon" href="{% static 'img/icons/icon-48x48.png' %}" />
    - บรรทัดที่ 15
      เดิม <link href="css/app.css" rel="stylesheet">
      ใหม่ <link href="{% static 'css/app.css' %}" rel="stylesheet">
    - บรรทัดที่ 45
      เดิม <script src="js/app.js"></script>
      ใหม่ <script src="{% static 'js/app.js' %}"></script>

โดยที่การเรียกใช้ {% static '' %} จะเป็นการเรียกไฟล์ที่อยู่ใน folder static ที่สร้างไว้เป็น root folder เช่นการเรียกใช้งานไฟล์ app.js ที่อยู่ใน folder js ก็จะถูกเรียกเป็น js/app.js

- หน้า header.html บรรทัดที่ 101 ให้กำหนดเป็น {% static 'img/avatars/avatar.jpg' %}" /> แทนที่ img/avatars/avatar.jpg เดิม เพื่อให้สามารถ load รูปภาพได้อย่างถูกต้อง และเพิ่มบรรทัดที่ 1 ให้มีการเรียกใช้งาน {% load static %}
- เชื่อมโยงไฟล์ส่วนประกอบได้แก่ ไฟล์ menu.html, menu.html, footer.html
    - ไฟล์ menu.htm จะเก็บข้อมูลของเมนูของหน้าเว็บ

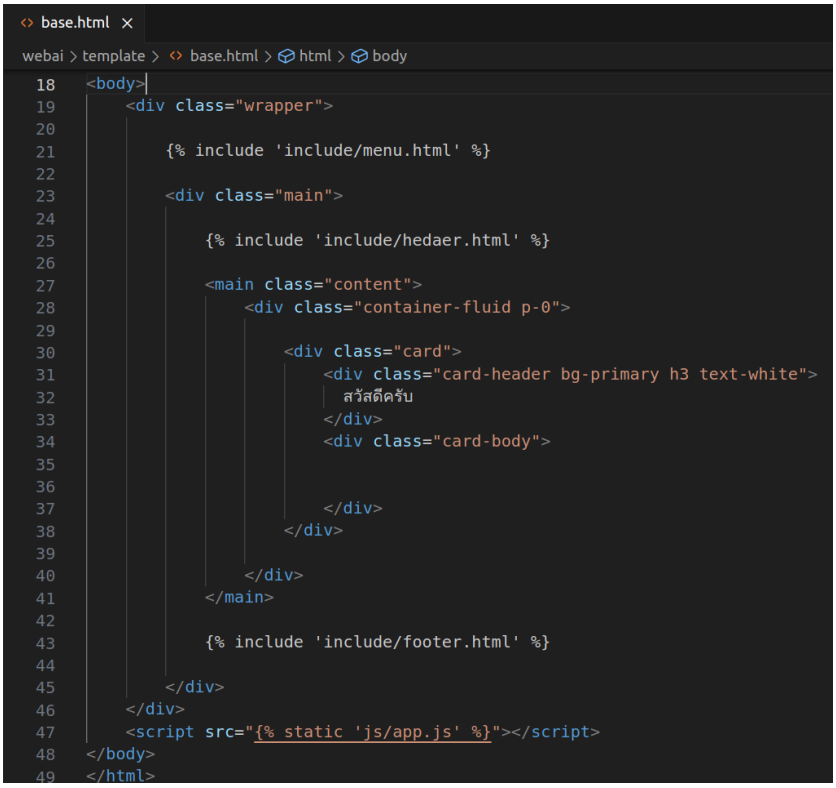
- ไฟล์ header.html จะเก็บข้อมูลของส่วนหัวที่แสดงรูปภาพและชื่อผู้ใช้งาน
- ไฟล์ footer.html จะเก็บข้อมูลส่วนท้ายของเว็บไซต์

โดยใช้คำสั่ง include {% include '' %} ในบรรทัดที่ระบุข้อความ “include ==> ชื่อไฟล์”

{% include 'include/menu.html' %}

{% include 'include/header.html' %}

{% include 'include/footer.html' %}



- สร้าง block content สำหรับการระบุตำแหน่งจะให้แสดงข้อมูลที่เปลี่ยนแปลงในหน้า .html อื่นๆ ที่จะถูกสืบทอดหน้า html นี้จากเพจอื่นๆ โดยใช้คำสั่ง {% block blockname %} {% endblock %}

ซึ่งในตัวอย่างจะกำหนดชื่อเป็น content ซึ่งจะอยู่ในตำแหน่งในไฟล์ base.html ดังภาพ และสามารถมี block ได้มากกว่า 1 block โดยระบุชื่อไม่ให้ซ้ำกัน เช่น block ที่ชื่อว่า content {% block titlecontent %} {% endblock %} ในตำแหน่งข้อความ สวัสดีครับ ซึ่งจะใช้สำหรับแสดงเป็นชื่อของหน้าเพจนั้นๆ และ {% block content %} {% endblock %} ในตำแหน่งภายใน ซึ่งจะแสดงเป็นเนื้อหาของหน้าเพจนั้นๆ ดังภาพ

ซึ่งไฟล์ base.html จะเป็นหน้าหลัก หรือ parent ให้กับเพจหรือหน้าอื่นๆ เรียกใช้งาน ที่มีการรวมส่วนประกอบทุกส่วน (menu, header, footer) เข้าไว้ใน base.html แล้ว

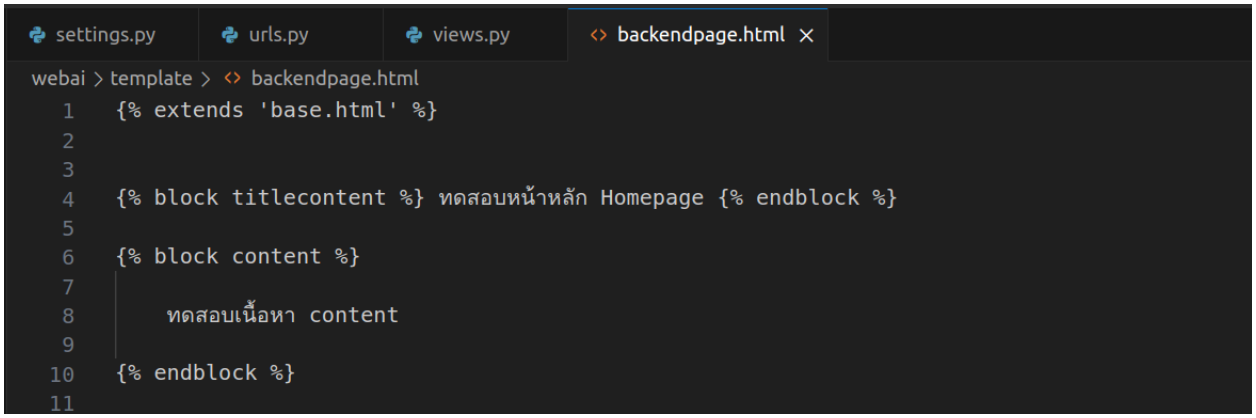
base.html × homepage.html

webai > template > base.html > html > body > div.wrapper > div.main

```
18  <body>
19      <div class="wrapper">
20
21          {% include 'include/menu.html' %}
22
23          <div class="main">
24
25              {% include 'include/hedaer.html' %}
26
27              <main class="content">
28                  <div class="container-fluid p-0">
29
30                      <div class="card">
31                          <div class="card-header bg-primary h3 text-white">
32                              {% block titlecontent %} {% endblock %}
33                          </div>
34                          <div class="card-body">
35                              {% block content %} {% endblock %}
36                          </div>
37                      </div>
38
39                  </div>
40              </main>
41
42              {% include 'include/footer.html' %}
43
44          </div>
45      </div>
46      <script src="{% static 'js/app.js' %}"></script>
47  </body>
48  </html>
```

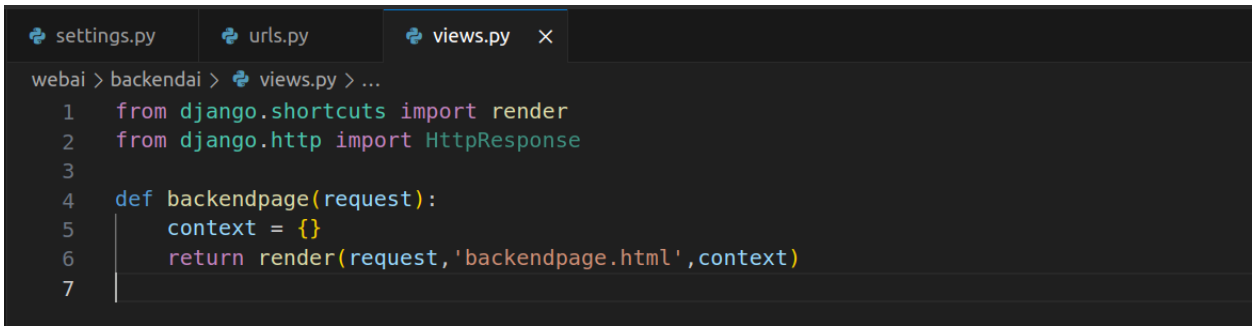
12. การนำไฟล์ Html มาใช้งานกับหน้าเว็บไซต์

- สร้างไฟล์ backendpage.html ใน folder template จากนั้นให้ระบุไฟล์ที่เป็น ตัวสืบทอด parent ของหน้าที่เตรียมไว้คือไฟล์ base.html โดยใช้คำสั่ง {% extends 'base.html' %} ในบรรทัดแรกของไฟล์
  โดยที่ extends คือชื่อคำสั่ง และ base.html คือชื่อไฟล์ที่เตรียมไว้เป็น parent ของแต่ละหน้า ซึ่งการใช้งานคำสั่ง extends และระบุชื่อไฟล์ จะถูกเรียกไฟล์ใน folder template กรณีตัวอย่างที่ไฟล์ base.html อยู่ใน folder mainpage การเรียกจะต้องเรียก path ที่ถูกต้องคือให้ระบุชื่อ folder ก่อน คือ mainpage/base.html {% extends 'base.html' %}
- กำหนดส่วนที่จะใช้แสดงผลที่ถูกสืบทอดมากจากหน้า parent ที่เตรียมไว้ ซึ่งได้สร้างไว้ 2 block ได้แก่ titlcontent และ content ดังภาพ สามารถสลับตำแหน่งของ block ได้



Python
```
{% extends 'base.html' %}
{% block titlecontent %} ทดสอบหน้าหลัก Homepage {% endblock %}

{% block content %}
    ทดสอบเนื้อหา content
{% endblock %}
```

- สร้าง function สำหรับหน้าหลักของเว็บไซต์ ชื่อว่า backendpage และเรียกใช้คำสั่ง render (เรียกว่า function) ที่ django เตรียมไว้ให้สำหรับ render html file



Python
```
def backendpage(request):
    context = {}
    return render(request,'backendpage.html',context)
```

โดยที่ render(request,'backendpage.html',context)
render คือคำสั่งในการ render file html (เรียกว่าฟังก์ชั่น)
ผ่านค่า 3 ได้แก่

request คือตัวแปรที่ต้องส่งไปด้วย (จะมีตัวอย่างการนำตัวแปร request ไปใช้งานในส่วนอื่นๆ)

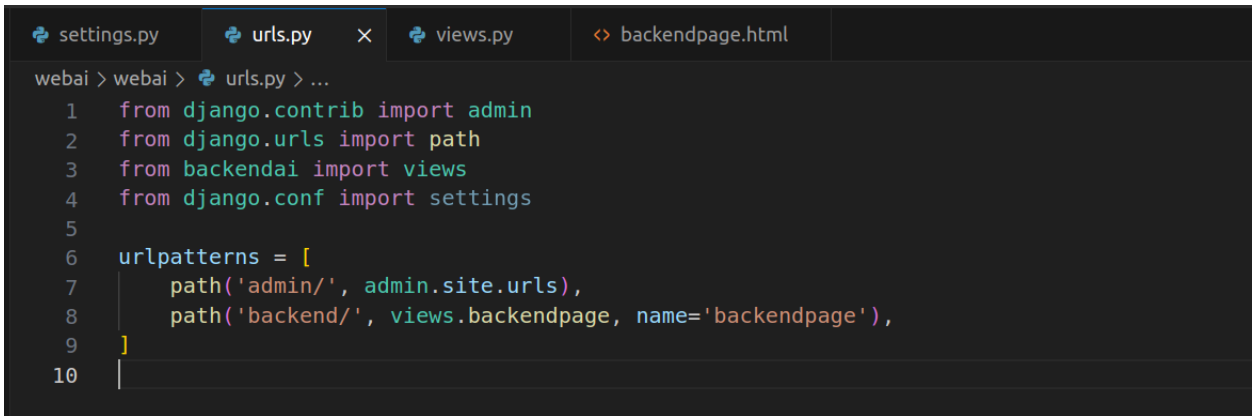
homepage.html คือ ชื่อไฟล์ html ที่อยู่ใน folder template ที่สร้างไว้ในขั้นตอนก่อนหน้า

context คือตัวแปรสำหรับส่งข้อมูลไปแสดงที่หน้า html ซึ่งอยู่ในรูปแบบของตัวแปรที่เป็น Dictionary ที่ประกอบไปด้วย key และ value
และตัวแปร context หรือค่าที่จะส่งไปแสดงผลในฟังก์ชั่น render สามารถว่างได้ ไม่จำเป็นต้องส่งค่าก็ได้

File : urls.py (อยู่ใน webai)

สร้าง routing urls.py

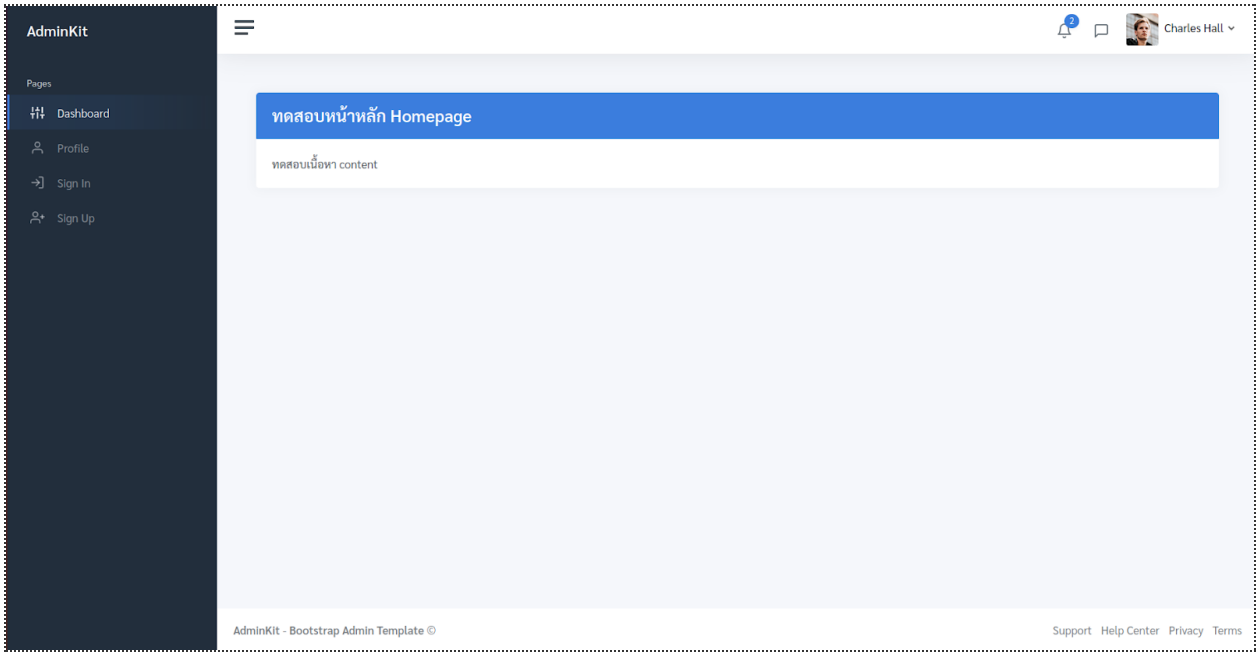
path('backend/', views.homepage, name='homepage')



ทดสอบ Runserver

python3 manage.py runserver

URL : http://127.0.0.1:8000/backend/



13. การติดตั้งและการเชื่อมต่อฐานข้อมูล

การติดตั้ง

- pip3 install mysqlclient

```
(env) pongthorn@pongthorn-site:~/Documents/webai$ pip3 install mysqlclient
Collecting mysqlclient
  Using cached mysqlclient-2.2.0.tar.gz (89 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: mysqlclient
  Building wheel for mysqlclient (pyproject.toml) ... done
  Created wheel for mysqlclient: filename=mysqlclient-2.2.0-cp310-cp310-linux_x86_64.whl size=123670 sha256=a804d1052fca15e3635b4f8a95e003fc00
f87bb609e751627b6ad13cd6175007
  Stored in directory: /home/pongthorn/.cache/pip/wheels/a4/f8/fd/0399687c0abd03c10c975ed56c692fcd3d0fb80440b5a661f1
Successfully built mysqlclient
Installing collected packages: mysqlclient
Successfully installed mysqlclient-2.2.0
```

*** อย่าลืมเรียกใช้งาน env ก่อนการติดตั้ง

```
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ pip3 install mysqlclient
Collecting mysqlclient
  Using cached mysqlclient-2.2.0.tar.gz (89 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... error
  error: subprocess-exited-with-error

  × Getting requirements to build wheel did not run successfully.
  │ exit code: 1
  ╰─> [24 lines of output]
      /bin/sh: 1: pkg-config: not found
      /bin/sh: 1: pkg-config: not found
      Trying pkg-config --exists mysqlclient
      Command 'pkg-config --exists mysqlclient' returned non-zero exit status 127.
      Trying pkg-config --exists mariadb
      Command 'pkg-config --exists mariadb' returned non-zero exit status 127.
      Traceback (most recent call last):
        File "/home/pongthorn/Documents/webai/env/lib/python3.10/site-packages/pip/_vendor/pyproject_hooks/_in_process/_in_proc
ess.py", line 353, in <module>
          main()
        File "/home/pongthorn/Documents/webai/env/lib/python3.10/site-packages/pip/_vendor/pyproject_hooks/_in_process/_in_proc
ess.py", line 335, in main
          json_out['return_val'] = hook(**hook_input['kwargs'])
        File "/home/pongthorn/Documents/webai/env/lib/python3.10/site-packages/pip/_vendor/pyproject_hooks/_in_process/_in_proc
ess.py", line 118, in get_requires_for_build_wheel
          return hook(config_settings)
        File "/tmp/pip-build-env-o29v_xd6/overlay/lib/python3.10/site-packages/setuptools/build_meta.py", line 325, in get_requ
ires_for_build_wheel
          return self._get_build_requires(config_settings, requirements=['wheel'])
        File "/tmp/pip-build-env-o29v_xd6/overlay/lib/python3.10/site-packages/setuptools/build_meta.py", line 295, in _get_bui
ld_requires
          self.run_setup()
        File "/tmp/pip-build-env-o29v_xd6/overlay/lib/python3.10/site-packages/setuptools/build_meta.py", line 311, in run_setu
p
          exec(code, locals())
        File "<string>", line 154, in <module>
        File "<string>", line 48, in get_config_posix
        File "<string>", line 27, in find_package_name
      Exception: Can not find valid pkg-config name.
      Specify MYSQLCLIENT_CFLAGS and MYSQLCLIENT_LDFLAGS env vars manually
      [end of output]
```

กรณีเกิดข้อผิดพลาด (Error)
ให้ติดตั้ง **sudo apt-get install python3-dev default-libmysqlclient-dev build-essential pkg-config** ก่อน ค่อยติดตั้ง **pip3 install mysqlclient** อีกครั้ง

```
(env) pongthorn@pongthorn-site:~/Documents/webai$ sudo apt-get install python3-dev default-libmysqlclient-dev build-essential pkg-config
[sudo] password for pongthorn:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
build-essential set to manually installed.
default-libmysqlclient-dev is already the newest version (1.0.8).
python3-dev is already the newest version (3.10.6-1~22.04).
python3-dev set to manually installed.
The following NEW packages will be installed:
  pkg-config
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 48.2 kB of archives.
After this operation, 134 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://th.archive.ubuntu.com/ubuntu jammy/main amd64 pkg-config amd64 0.29.2-1ubuntu3 [48.2 kB]
Fetched 48.2 kB in 0s (316 kB/s)
Selecting previously unselected package pkg-config.
(Reading database ... 208390 files and directories currently installed.)
Preparing to unpack .../pkg-config_0.29.2-1ubuntu3_amd64.deb ...
Unpacking pkg-config (0.29.2-1ubuntu3) ...
Setting up pkg-config (0.29.2-1ubuntu3) ...
Processing triggers for man-db (2.10.2-1) ...
```

การสร้างฐานข้อมูลใน Phpmyadmin

- เปิด phpmyadmin โดยเข้าผ่านเว็บบราวเซอร์ 127.0.0.1/phpmyadmin
- คลิกที่เมนู Databases
- กำหนดชื่อฐานข้อมูล และประเภทฐานข้อมูล
    - ชื่อฐานข้อมูล webai

- ประเภท utf8_general_ci
- สร้างฐานข้อมูลคลิกที่ปุ่ม Create

การตั้งค่า Settings.py

ไปที่ไฟล์ settings.py อยู่ใน folder project ในส่วนของ DATABASES

```
77  DATABASES = {
78      "default": {
79          "ENGINE": "django.db.backends.mysql",
80          "NAME": "webai",
81          "USER": "root",
82          "PASSWORD": "",
83          "HOST": "127.0.0.1",
84          "PORT": "3306",
85      }
86  }
```

โดยที่

ENGINE ให้ระบุเป็น django.db.backends.mysql ในฐานข้อมูลแบบ mysql
NAME คือ ชื่อฐานข้อมูล
USER คือ ชื่อผู้ใช้งานของฐานข้อมูล
PASSWORD คือ รหัสผ่านของฐานข้อมูล
HOST คือหมายเลข ip address หรือ domain name ของเครื่อง database
PORT เป็นหมายเลข port ของ Database โดยปกติหมายเลขมาตรฐานตั้งต้นคือ 3306

```python
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "webai",
        "USER": "root",
        "PASSWORD": "",
        "HOST": "127.0.0.1",
        "PORT": "3306",
    }
}
```

กรณีของการจำลองฐานข้อมูลด้วยโปรแกรม xampp **USER** ค่าตั้งต้นเป็น root **PASSWORD** ค่าตั้งต้นเป็นไม่มีรหัสผ่าน (ระบุดังตัวอย่าง)

การเชื่อมต่อฐานข้อมูล

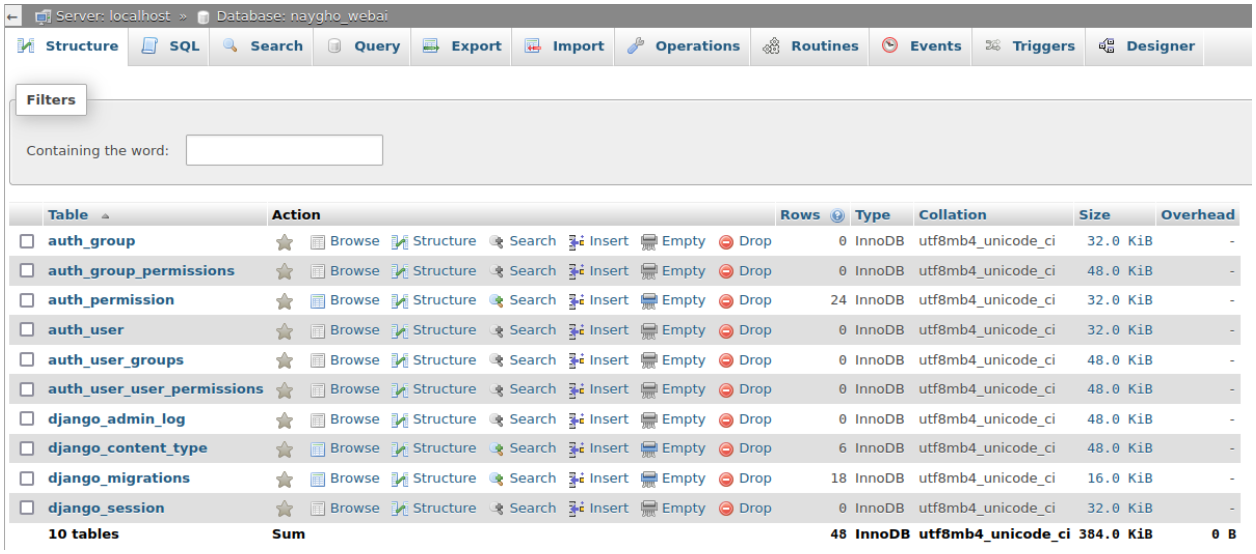
คำสั่ง python3 manage.py migrate

ในกรณีที่ยังไม่ได้มีการสร้างฐานข้อมูล models.py และสร้างฐานข้อมูลครั้งแรกจากการเชื่อมต่อและตั้งค่าฐานข้อมูล ให้ใช้คำสั่งเพียงแค่ migrate ยังไม่ต้องใช้การ makemigrations เนื่องจากยังไม่มีฐานข้อมูลที่สร้างขึ้นเองจาก models.py

```
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ python3 manage.py migrate
System check identified some issues:

WARNINGS:
?: (mysql.W002) MariaDB Strict Mode is not set for database connection 'default'
        HINT: MariaDB's Strict Mode fixes many data integrity problems in MariaDB, such as data truncation upon insertion, by escalating warni
ngs into errors. It is strongly recommended you activate it. See: https://docs.djangoproject.com/en/4.2/ref/databases/#mysql-sql-mode
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

จะพบกับตารางทั้งหมด 10 ตารางที่เกิดจากการสร้างให้โดย django และในส่วนของการ register และการ login จะใช้ตารางที่ชื่อว่า **auth_user** ในการจัดเก็บข้อมูลการและการตรวจสอบข้อมูล ซึ่งประกอบไปด้วยคอลัมน์

- id
- username (required)
- password (required)
- email
- is_superuser
- is_staff
- last_login
- is_active
- first_name
- last_name
- date_joined



14. การตั้งค่า Alert message สำหรับการแจ้งเตือน

การตั้งค่า

ไปที่หน้า settings.py และเรียกใช้งาน

from django.contrib.messages import constants as message_constants

โดยกำหนดชื่อเล่นให้เป็น message_constants ซึ่งได้กำหนดรูปแบบของ alert messages เป็น 5 รูปแบบได้แก่ DEBUG, INFO, SUCCESS, WARNING, ERROR โดยสามารถกำหนดให้เรียกใช้งานในแต่ละรูปแบบและกำหนดสีของการแจ้งเตือนได้แตกต่างกัน ซึ่งในกรณีได้กำหนดเพื่อให้นำมาใช้กับ css ของ bootstrap

```python
from django.contrib.messages import constants as message_constants

MESSAGE_TAGS = {
    message_constants.DEBUG: 'secondary',
    message_constants.INFO: 'info',
    message_constants.SUCCESS: 'success',
    message_constants.WARNING: 'warning',
    message_constants.ERROR: 'danger',
}
```

โดยที่ เมื่อมีการเรียกใช้งาน

- DEBUG เป็นคลาสของ secondary จะแสดงเป็นสีเทา
- INFO เป็นคลาสของ info จะแสดงเป็นสีฟ้า
- SUCCESS เป็นคลาสของ success จะแสดงเป็นสีเขียว

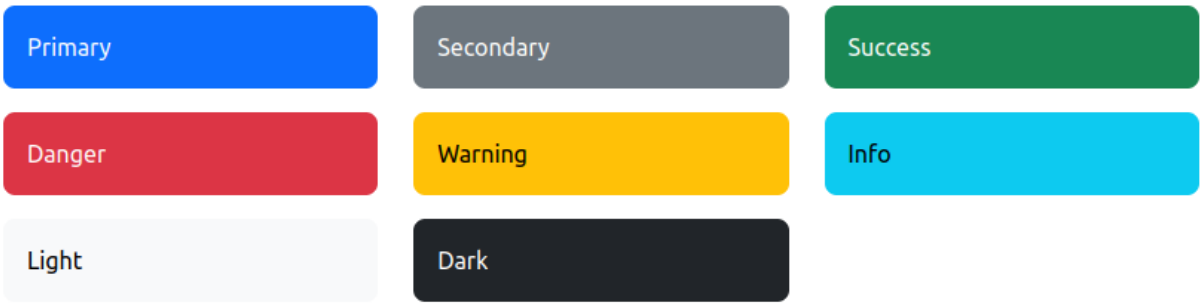
- WARNING เป็นคลาสของ warning จะแสดงเป็นสีเหลือง
- ERROR เป็นคลาสของ secondary จะแสดงเป็นสีเทา

```
from pathlib import Path
from django.contrib.messages import constants as message_constants

MESSAGE_TAGS = {
    message_constants.DEBUG: 'secondary',
    message_constants.INFO: 'info',
    message_constants.SUCCESS: 'success',
    message_constants.WARNING: 'warning',
    message_constants.ERROR: 'danger',
}
```

ตัวอย่างคลาสสีใน bootstrap

Theme colors

We use a subset of all colors to create a smaller color palette for generating color schemes, also available as Sass variables and a Sass map in Bootstrap's `scss/_variables.scss` file.



ดูข้อมูลเพิ่มเติมได้ที่ https://getbootstrap.com/docs/5.3/customize/color/

การเรียกใช้งาน

ให้ import ด้านบนของไฟล์ Views.py โดยใช้คำสั่ง

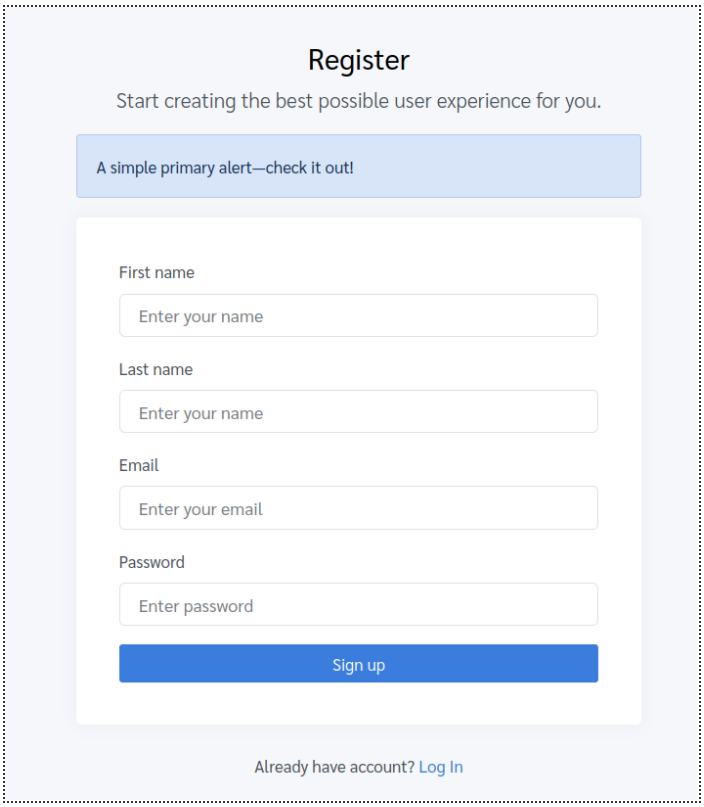
from django.contrib import messages

ตัวอย่างกรณีต้องการเป็น alert เป็นประเภท SUCCESS ให้กำหนดเป็น
messages.success(request “ข้อความแสดงการแจ้งเตือน”)

การกำหนดส่วนของการแสดงผลใน html

```python
{% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }} fs-6">
                {{ message }}
            </div>
        {% endfor %}
{% endif %}
```

ตัวอย่างและตำแหน่งการแสดงผล alert message ดังตัวอย่างแสดงในไฟล์ register.html (ในบรรทัดที่ 33-39)



15. สร้างหน้าเพจสมัครสมาชิก Register

File : urls.py (อยู่ใน webai)

โดยเพิ่มเส้นทางหนึ่งเส้นทาง register/ โดยเรียกใช้งานฟังก์ชั่น registerpage และกำหนดให้ url name คือ registerpage

```python
path('register/', views.registerpage, name='registerpage')
```

File : views.py (อยู่ใน backendai)

view สำหรับการแสดงผล ให้แยกระบุคำว่า page เพื่อแยกระหว่าง page และ process สำหรับการประมวลผล โดยสร้างชื่อฟังก์ชั้นคือ registerpage

```python
def registerpage(request):
    context = {}
    return render(request,'register.html',context)
```

Template : register.html

ในการสร้างไฟล์ html ให้ใช้ไฟล์ที่ได้จากการดาวน์โหลด ซึ่งใช้ไฟล์ที่ชื่อว่า register.html ในการทำงานของหน้านี้

สามารถปรับแต่งหน้าเว็บได้ตามความต้องการและกำหนดในส่วนของ Form

1. action = การส่งผลจากฟอร์มไปที่ฟังก์ชั่นไหน

2. method = POST โดยกำหนดให้ action ไปที่หน้า registercreate เพื่อประมวลผลและบันทึกข้อมูล กำหนดให้เป็น แต่ละ input จะประกอบไปด้วยพารามิเตอร์ต่างๆ เช่น name คือชื่อของ input ที่จะนำไปใช้งาน และ type คือ รูปแบบหรือประเภทของ input นั้นๆ เช่น text สำหรับข้อความ email สำหรับช่อง email password สำหรับรหัสผ่าน ซึ่งมีควมแตกต่างจาก text คือ เมื่อพิมพ์เข้าไปในช่องแล้ว จะแสดงขอความเป็นจุดวงกลม

ในฟอร์ม Form ประกอบไปด้วย input ทั้งหมด 4 input และแต่ละ input บังคับให้กรอกข้อมูลทุกช่องโดยระบุเป็น required ในทุกช่อง input

- First name
    - name : firstname
    - type : text
- Last name
    - name : lastname
    - type : text
- Email
    - name : email
    - type : email
- Password
    - name : password
    - type : password

และทำการเพิ่มคำสั่ง {% csrf_token %} ของ Django สำหรับการป้องกันการโจมตีแบบ Cross-site Request Forgery

ปุ่มสำหรับ Log In ระบุลิงก์ไปที่ url homepage เนื่องจากระบบออกแบบให้หน้า login เป็นหน้าหลักของเว็บไซต์

เพิ่มการแสดงผล alert message

Python

```
{% if messages %}
      {% for message in messages %}
            <div class="alert alert-{{ message.tags }} fs-6">
                  {{ message }}
            </div>
      {% endfor %}
{% endif %}
```

Python

```
<form  action="{% url 'registercreate' %}"  method="POST">
{% csrf_token %}
<div class="mb-3">
      <label class="form-label">First name</label>
      <input class="form-control form-control-lg" type="text" name="firstname"
      placeholder="Enter your name" required/>
</div>
<div class="mb-3">
      <label class="form-label">Last name</label>
      <input class="form-control form-control-lg" type="text" name="lastname"
      placeholder="Enter your name" required/>
</div>

<div class="mb-3">
      <label class="form-label">Email</label>
      <input class="form-control form-control-lg" type="email" name="email"
      placeholder="Enter your email" required/>
</div>
<div class="mb-3">
      <label class="form-label">Password</label>
      <input class="form-control form-control-lg" type="password"
      name="password" placeholder="Enter password" required/>
</div>

<div class="d-grid gap-2 mt-3">
      <button type="submit" class="btn btn-primary">Sign up</button>
</div>

</form>
```

16. สร้างฟังก์ชั่นสำหรับบันทึกผลการสมัครสมาชิก

File : urls.py (อยู่ใน webai)

เส้นทาง Url ของการสมัครสมาชิก ได้ทำการเพิ่ม /action เข้ามาเพื่อจัดกลุ่มระบุว่าเป็นการประมวลผลหรือการดำเนินการ และ /create เป็นการระบุว่าเป็นส่วนฟังก์ชั่นของการสร้างผู้ใช้งาน

```python
path('register/action/create/', views.registercreate, name='registercreate')
```

File : views.py (อยู่ใน backendai)

เรียกใช้งานฟังก์ชั่นสำหรับการ redirect ต่อจาก render ให้เพิ่มเป็น

```
from django.shortcuts import render, redirect
```

เรียกใช้งาน model User ให้เพิ่มต่อการจาก import ด้านบนสุดของ views.py

```
from django.contrib.auth.models import User
```

การเช็คเงื่อนไขในการสมัครสมาชิก

1. เช็ค method ที่ส่งมาจากหน้า from register เป็น POST หรือไม่ หากไม่ใช่ให้ redirect กลับไปที่หน้า registerpage ซึ่งชื่อ registerpage คือ name ที่ระบุใน urls.py ที่ระบุในแต่ละเส้น (ไม่ใช่ชื่อฟังก์ชั่น ในบางกรณีอาจจะกำหนดให้ชื่อฟังก์ชั่นและ name เป็นชื่อเดียวกันก็ได้) พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
2. เช็คตัวแปรที่รับค่าจาก from ไม่ให้มีค่าใดเป็นค่าว่าง โดยการใช้คำสั่ง is not None และใช้เงื่อไขแบบ and โดยทุกๆตัวแปรต้องไม่เป็นค่าว่าง หากเป็นค่าว่างให้ redirect กลับไปที่หน้า registerpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
3. ตรวจสอบชื่อผู้ใช้งาน ไม่ให้ซ้ำกับข้อมูลเดิมในฐานข้อมูล User โดยใช้การ query ข้อมูลออกมาตรวจสอบ User.objects.filter(username=emailinput).count() โดยที่
   User = ชื่อฐานข้อมูล
   filter() = คือ Where ใน mysql ใช้ในการกำหนดเงื่อนไขในการค้นหาข้อมูล
   count() = คือ การนับจำนวนข้อมูลที่ออกมาจากการค้นหา ค่าจะส่งออกมาเป็นจำนวนเต็ม
   emailinput = คือชื่อตัวแปร
   หากส่งค่ากลับมาเป็น 0 หมายถึง ไม่มีข้อมูลที่ส่งเข้าไปตรวจสอบ
   หากไม่ใช่ 0 หมายถึง มีข้อมูลที่ส่งเข้าไปตรวจสอบ ให้ redirect กลับไปที่หน้า registerpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “ชื่อผู้ใช้งานซ้ำกับ ข้อมูลในระบบ โปรดสมัครใหม่อีกครั้ง”

Python

```python
def registercreate(request):
    if request.method == "POST":
        firstnameinput = request.POST.get('firstname',None)
        lastnameinput = request.POST.get('lastname',None)
        emailinput = request.POST.get('email',None)
        passwordinput = request.POST.get('password',None)

        if firstnameinput is not None and lastnameinput is not None and
emailinput is not None and passwordinput is not None:
            checkuser = User.objects.filter(username=emailinput).count()
            if checkuser == 0:
         ### ส่วนการเพิ่มข้อมูลผู้ใช้งาน ###
            else:
                messages.error(request,"ชื่อผู้ใช้งานซ้ำกับข้อมูลในระบบ โปรดสมัครใหม่อีกครั้ง
")
                return redirect(registerpage)
        else:
            messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
            return redirect(registerpage)
    else:
        messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
        return redirect(registerpage)
```


```python
def registercreate(request):
    if request.method == "POST":
        firstnameinput = request.POST.get('firstname',None)
        lastnameinput = request.POST.get('lastname',None)
        emailinput = request.POST.get('email',None)
        passwordinput = request.POST.get('password',None)

        if firstnameinput is not None and lastnameinput is not None and emailinput is not None and
        passwordinput is not None:
            checkuser = User.objects.filter(username=emailinput).count()
            if checkuser == 0:

                pass
            else:
                messages.error(request,"ชื่อผู้ใช้งานซ้ำกับข้อมูลในระบบ โปรดสมัครใหม่อีกครั้ง")
                return redirect(registerpage)
        else:
            messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
            return redirect(registerpage)
    else:
        messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
        return redirect(registerpage)
```

การเพิ่มข้อมูลผู้ใช้งาน

โดยมีเงื่อนไขการเพิ่มข้อมูลผู้ใช้งาน

1. กรณีที่ ถ้าหากสามารถเพิ่มข้อมูลผู้ใช้งานได้ให้ redirect กลับไปที่หน้า homepage เพื่อทำการลงชื่อเข้าใช้งาน พร้อมกับระบุข้อข้อความ alert message รูปแบบ success “สมัครสมาชิกเสร็จสมบูรณ์ สามารถลงชื่อเข้าใช้งานได้ทันที” หากไม่สามารถเพิ่มข้อมูลได้โดยจาก error ใดๆ ให้ redirect ไปที่หน้า registerpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง”

```
Python
createuser = User.objects.create_user(username=emailinput,email=emailinput,password=passwordinput)
createuser.first_name = firstnameinput
createuser.last_name = lastnameinput
createuser.save()
if createuser:
        messages.success(request,"สมัครสมาชิกเสร็จสมบูรณ์ สามารถลงชื่อเข้าใช้งานได้ทันที")
        return redirect(homepage)
else:
        messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
        return redirect(registerpage)
```

สรุปฟังก์ชั่นการสมัครสมาชิก

```
Python
def registercreate(request):
    if request.method == "POST":
        firstnameinput = request.POST.get('firstname',None)
        lastnameinput = request.POST.get('lastname',None)
        emailinput = request.POST.get('email',None)
        passwordinput = request.POST.get('password',None)

        if firstnameinput is not None and lastnameinput is not None and
emailinput is not None and passwordinput is not None:
            checkuser = User.objects.filter(username=emailinput).count()
            if checkuser == 0:
                createuser =
User.objects.create_user(username=emailinput,email=emailinput,password=password
input)
                createuser.first_name = firstnameinput
                createuser.last_name = lastnameinput
                createuser.save()
                if createuser:
```

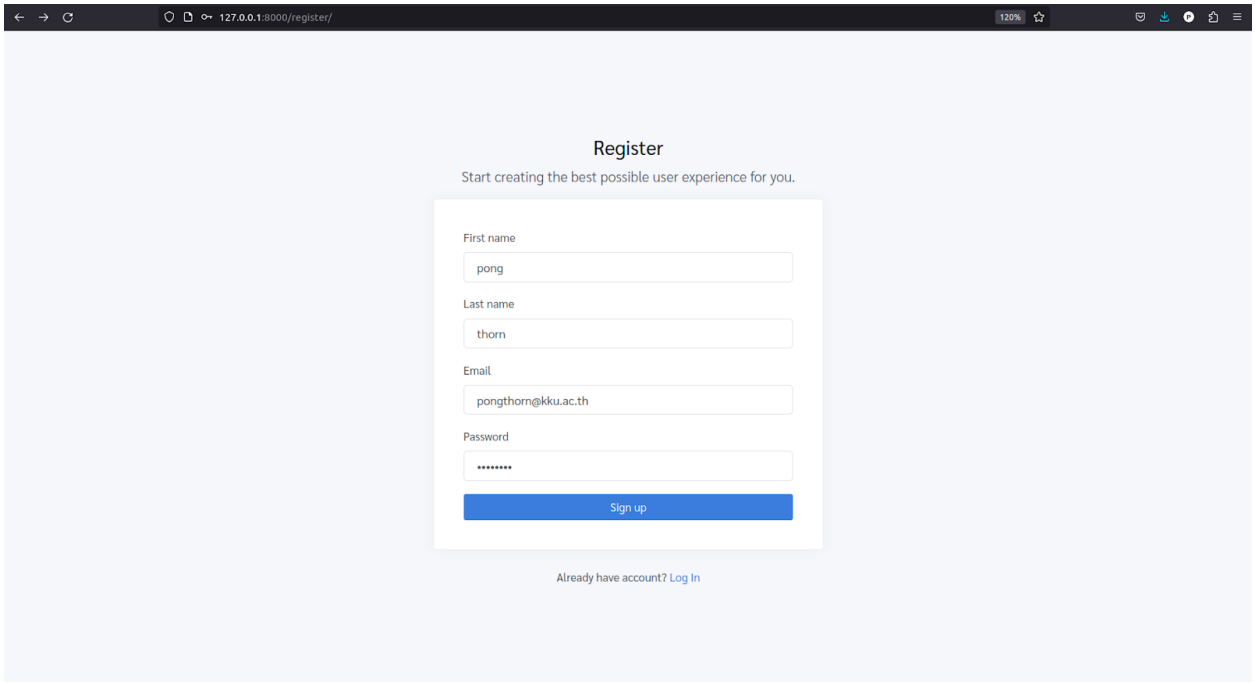
```
                    messages.success(request,"สมัครสมาชิกเสร็จสมบูรณ์ สามารถลงชื่อเข้าใช้
งานได้ทันที")
                    return redirect(homepage)
                else:
                    messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
                    return redirect(registerpage)
            else:
                messages.error(request,"ชื่อผู้ใช้งานซ้ำกับข้อมูลในระบบ โปรดสมัครใหม่อีกครั้ง
")
                return redirect(registerpage)
        else:
            messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
            return redirect(registerpage)
    else:
        messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
        return redirect(registerpage)
```

```python
def registercreate(request):
    if request.method == "POST":
        firstnameinput = request.POST.get('firstname',None)
        lastnameinput = request.POST.get('lastname',None)
        emailinput = request.POST.get('email',None)
        passwordinput = request.POST.get('password',None)

        if firstnameinput is not None and lastnameinput is not None and emailinput is not None and passwordinput is not None:
            checkuser = User.objects.filter(username=emailinput).count()
            if checkuser == 0:
                createuser = User.objects.create_user(username=emailinput,email=emailinput,password=passwordinput)
                createuser.first_name = firstnameinput
                createuser.last_name = lastnameinput
                createuser.save()
                if createuser:
                    messages.success(request,"สมัครสมาชิกเสร็จสมบูรณ์ สามารถลงชื่อเข้าใช้งานได้ทันที")
                    return redirect(homepage)
                else:
                    messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
                    return redirect(registerpage)
            else:
                messages.error(request,"ชื่อผู้ใช้งานซ้ำกับข้อมูลในระบบ โปรดสมัครใหม่อีกครั้ง")
                return redirect(registerpage)
        else:
            messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
            return redirect(registerpage)
    else:
        messages.error(request,"เกิดข้อผิดพลาด โปรดสมัครใหม่อีกครั้ง")
        return redirect(registerpage)
```

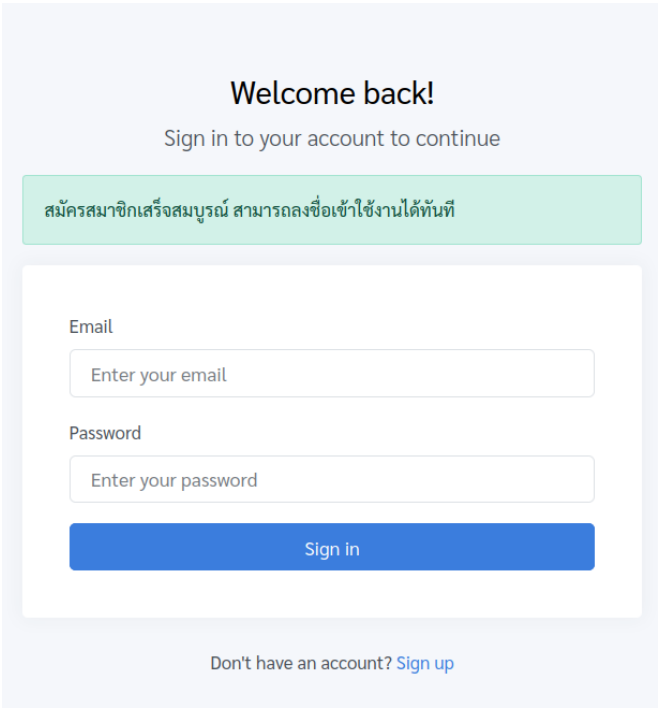
ทดสอบการ Runserver

python3 manage.py runserver

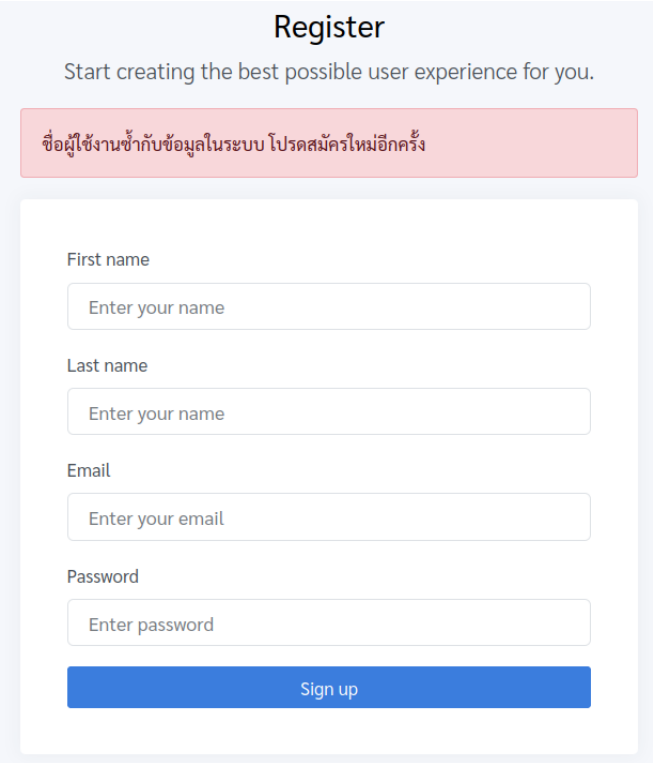
http://127.0.0.1:8000/register/



สมัครสมาชิก เสร็จสมบูรณ์



สมัครสมาชิก ไม่สมบูรณ์ ผู้ใช้งานซ้ำในระบบ



หลังจากการสมัครสมาชิก จะแสดงข้อมูลเพิ่มเข้ามาดังภาพ

17. สร้างหน้าลงชื่อเข้าใช้งาน Login

File : urls.py (อยู่ใน webai)

สร้างเส้นทางสำหรับเรียกใช้งานหน้าลงชื่อเข้าใช้งาน

```python
path('loginaction/', views.loginaction, name='loginaction')
```

Template : login.html

ไฟล์ที่จะใช้ในการแสดงผลคือไฟล์ login.html โดยได้ทำการกำหนด โดยฟังก์ชั่นที่จะเป็นหน้าเพจสำหรับหน้า login คือฟังกชั่น homepage เพราะระบบใน workshop นี้ออกแบบให้หน้าหลักของระบบเป็นหน้าสำหรับการ login

- action ของ form (การส่งผลจากฟอร์มไปที่ฟังก์ชั่นไหน) ไปที่ url loginaction

- input มีทั้งหมด 2 input
    - Email / Username
        - type : email
        - name : email
    - Password
        - type : password
        - name : password
- เพิ่มคำสั่ง {% csrf_token %} ของ Django สำหรับการป้องกันการโจมตีแบบ Cross-site Request Forgery
- เพิ่ม required ทั้ง 2 input
- ปุ่มสำหรับ Sign up ระบุลิงก์ไปที่ url registerpage

เพิ่มการแสดงผล alert message ในไฟล์ login.html

```Python
{% if messages %}
	{% for message in messages %}
		<div class="alert alert-{{ message.tags }} fs-6">
			{{ message }}
		</div>
	{% endfor %}
{% endif %}
```

ฟอร์ม html สำหรับการ login

```Python
<form action="#{% url 'loginaction' %}" method="POST">
{% csrf_token %}
	<div class="mb-3">
		<label class="form-label">Email / Username</label>
		<input class="form-control form-control-lg" type="email"
		name="email" placeholder="Enter your email" required/>
	</div>
	<div class="mb-3">
		<label class="form-label">Password</label>
		<input class="form-control form-control-lg" type="password"
		name="password" placeholder="Enter your password" required/>
	</div>
	<div class="d-grid gap-2 mt-3">
		<button type="submit" class="btn btn-primary">Sign in</button>
	</div>
</form>
```

File : views.py (อยู่ใน backendai)

การตรวจสอบข้อมูลและการ Login

เรียกใช้งานฟังก์ชั่น authenticate และ login โดยทำการ import ไว้บรรทัด

```
from django.contrib.auth import authenticate, login
```

สร้างฟังก์ชั่น loginaction สำหรับการตรวจสอบข้อมูล
โดยมีเงื่อนไขสำหรับการตรวจสอบ

1. เช็ค method ที่ส่งมาจากหน้า from login เป็น POST หรือไม่ หากไม่ใช่ให้ redirect กลับไปที่หน้า homepage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
2. เช็คตัวแปรที่รับค่าจาก from ไม่ให้มีค่าใดเป็นค่าว่าง โดยการใช้คำสั่ง is not None และใช้เงื่อไขแบบ and โดยทุกๆตัวแปรต้องไม่เป็นค่าว่าง หากเป็นค่าว่างให้ redirect กลับไปที่หน้า homepage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
3. ตรวจสอบข้อมูลผู้ใช้งานและรหัสผ่านจากฟอร์มหน้า login authenticate(username=emailinput,password=passwordinput) โดยที่ ใช้ฟังก์ชั่น authenticate ที่ django เตรียมไว้ให้ ซึ่งผ่านค่าเข้าไปสองค่าคือ username และ password

   กรณีที่ username และหรือ password ไม่ถูกต้อง (ส่งค่า None กลับมา) และให้ redirect กลับไปที่หน้า homepage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “ชื่อผู้ใช้งานหรือรหัสผ่านไม่ถูกต้อง โปรดลองใหม่อีกครั้ง”

   กรณีที่ถูกทั้งสองค่า จะส่งค่า data ของผู้ใช้งานที่ถูกต้องกลับมาให้ดำเนินการ login เข้าสู่ระบบ โดยใช้ฟังก์ชั่น login() และผ่านค่าตัวแปรที่รับผลจากการตรวจสอบชื่อผู้ใช้งาน และตัวแปร request ที่เป็นค่าตั้งต้นไปด้วย login(request,usercheck)
   จากนั้น redirect ไปที่หน้า backendpage หน้าของการจัดการสำหรับผู้ใช้งาน พร้อมกับระบุข้อข้อความ alert message รูปแบบ success “ชื่อผู้ใช้งานหรือรหัสผ่านไม่ถูกต้อง โปรดลองใหม่อีกครั้ง”

สรุปฟังก์ชั่นการตรวจสอบการ Login

Python

```python
def loginaction(request):
    if request.method == "POST":
        emailinput = request.POST.get('email',None)
        passwordinput = request.POST.get('password',None)
```

```
        if emailinput is not None and passwordinput is not None:
            usercheck =
authenticate(username=emailinput,password=passwordinput)
            if usercheck:
                login(request,usercheck)
                messages.success(request,"ลงชื่อเข้าใช้งานเสร็จสมบูรณ์")
                return redirect(backendpage)
            else:
                messages.error(request,"ชื่อผู้ใช้งานหรือรหัสผ่านไม่ถูกต้อง โปรดลองใหม่อีกครั้ง
")
                return redirect(homepage)
        else:
            messages.error(request,"เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง")
            return redirect(homepage)
    else:
        messages.error(request,"เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง")
        return redirect(homepage)
```

18. สร้างฟังก์ชั่นสำหรับการลงชื่อออกจากระบบ Logout

File : urls.py (อยู่ใน webai)

สร้างเส้นทางสำหรับการตรวจสอบและประมวลผลการ logoutaction/ โดยเรียกใช้งานฟังก์ชั่น logoutaction และกำหนดให้ url name เป็น logoutaction

Python

```
path('logoutaction/', views.logoutaction, name='logoutaction'),
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่น logoutaction สำหรับการ logout ออกจากระบบผู้ใช้งาน โดยเรียกใช้ฟังก์ชั่น logout ที่ django เตรียมไว้ให้ เพิ่มไว้ที่บรรทัดเดียวกับ from django.contrib.auth import authenticate, login, logout

ในฟังก์ชั่น logoutaction มีการเรียกใช้งาน built in function คือ logout(request) และมีการผ่านค่าตัวแปร request เข้าไปในฟังก์ชั่น

ให้ redirect ไปที่หน้า homepage พร้อมกับระบุข้อข้อความ alert message รูปแบบ success “ลงชื่อออกจากระบบเสร็จสมบูรณ์”

```python
def logoutaction(request):
    logout(request)
    messages.success(request,"ลงชื่อออกจากระบบเสร็จสมบูรณ์")
    return redirect(homepage)
```

เมื่อมีการเรียกใช้งานฟังก์ชั่นสำหรับการ logout ในหน้า html ก็ให้เรียกใช้งานผ่าน {% url 'logoutaction' %}

19. การจัดเตรียมหน้าสำหรับสมาชิก (ระบบหลังบ้านสำหรับสมาชิก)

ในส่วนของการจัดการหน้าเพจสำหรับสมาชิกที่ผ่านการ login มาแล้ว ต่อจากเรื่อง “การนำไฟล์ html มาใช้งานกับหน้าเว็บไซต์” ที่ผ่านมา ซึ่งจะมีการปรับแต่งหน้าเพจเพื่อนำไปใช้งานต่อในส่วนอื่นๆ จากการทำงาน ซึ่งจะเป็นส่วนของการปรับแต่งแก้ไขหน้าเพจ html ที่มีการเตรียมไว้ให้แล้วเบื้องต้น แบ่งการจัดการออกเป็น 3 ส่วน 1 การจัดเตรียมส่วนของ Header 2 การจัดเตรียมเมนู Menu 3 การจัดเตรียม footer

การจัดเตรียมส่วนของ Header

- การจัดการ tag title

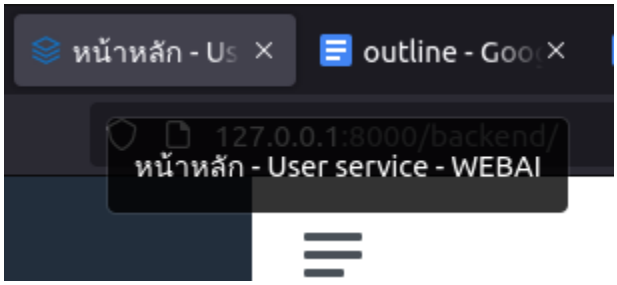
ในการแสดงผล tag title ที่แสดงชื่อของหน้าเพจนั้นๆ บน bar ของ Web Browser ซึ่งต้องทำการกำหนดที่ ไฟล์ base.html ซึ่งเป็นไฟล์ที่เตรียมไว้สำหรับเป็นเทมเพลตให้เพจอื่นๆ เรียกใช้งาน ได้กำหนดข้อความ และ block สำหรับการเปลี่ยนชื่อเพจของแต่ละเพจ

```python
<title> {% block titlebar %} {% endblock %} - User service - WEBAI</title>
```

- กำหนดชื่อ title ของหน้าหลักที่กำลังเรียกใช้งาน 127.0.0.1:8000/backend

ไปที่ไฟล์ backendpage.html โดยให้เพิ่ม block titlebar สำหรับกำหนดชื่อเพจแต่ละเพจ และทุกๆครั้งของการสร้างเพจใหม่ จะมีโครงสร้างเหมือนกับไฟล์ backendpage.html สามารถคัดลอกโครงสร้างนี้ไปใช้งานได้เลย

```python
Filename : backendpage.html

{% extends 'base.html' %}

{% block titlebar %} หน้าหลัก {% endblock %}  <=== เพิ่มบรรทัดนี้
{% block titlecontent %} ทดสอบหน้าหลัก Homepage {% endblock %}

{% block content %}
    ทดสอบเนื้อหา content  ++++
{% endblock %}
```



จะแสดงเป็น หน้าหลัก - User service - WEBAI ซึ่งข้อความ หน้าหลัก มาจาก block titlebar ที่มาจากไฟล์ backendpage.html และส่วนของ - User service - WEBAI มาจากไฟล์ base.html ที่เป็นไฟล์เทมเพลตให้เรียกใช้งาน ที่ได้กำหนดในส่วนของการ “การจัดการ tag title” ที่ผ่านมา

- แก้ไขชื่อของผู้ใช้งานจากโดยให้แสดงชื่อผู้ใช้งานที่ลงชื่อเข้าใช้งานที่ login เข้ามา

ในบรรทัดที่ 101 - 103 เดิมข้อความจะแสดงเป็น Charles Hall ให้เปลี่ยนเป็น {{ request.user.first_name }} {{ request.user.last_name }} และทำการลบ alt="Charles Hall" ออก

ซึ่ง {{ request.user.first_name }} โดยที่ first_name คือชื่อคอลัมน์ที่อยู่ในตาราง User ที่เก็บชื่อจริงของผู้ใช้งาน

และ {{ request.user.last_name }} โดยที่ last_name คือชื่อคอลัมน์ที่อยู่ในตาราง User ที่เก็บนามสกุลของผู้ใช้งาน

กรณีที่อยากจะให้แสดงข้อมูลส่วนอื่นๆ ให้ดูคอลัมน์อ้างอิงในฐานข้อมูล เช่น .username หรือ .email คือ อีเมลของผู้ใช้งาน

ในตัวอย่างนี้จะไม่เปลี่ยนรูปของผู้ใช้งาน สามารถประยุกต์ใช้งานต่อไปได้


```Python
<a class="nav-link dropdown-toggle d-none d-sm-inline-block" href="#"
data-bs-toggle="dropdown">
                    <img src="{% static 'img/avatars/avatar.jpg' %}"
class="avatar img-fluid rounded me-1" alt="Charles Hall" /> <span
class="text-dark">{{ request.user.first_name }} {{ request.user.last_name }}
</span>
</a>
```

- ลิงก์ไปที่หน้า profile และ ลบเมนู setting

  ไปที่ไฟล์ header.html ที่อยู่ใน folder include เพื่อทำการแก้ไข ลิงก์ไปที่หน้า profile และลบเมนู setting ออก โดยจะอยู่ส่วนท้ายของไฟล์

  โดยลบส่วนของ
 Settings และ ออกทั้งสองบรรทัด (บนและล่าง)

  และแก้ไข href บรรทัดของ Profile ที่อยู่ใน tag a ให้เป็น {% url 'backendpage' %} ไว้ก่อน และเมื่อมีการสร้างหน้า profile แล้วจึงกลับมาแก้ไขให้ถูกต้องอีกครั้ง

- สร้าง pop up (Modal) สำหรับการ logout

  ไปที่ไฟล์ header.html ที่อยู่ใน folder include เพื่อทำการแก้ไข เพิ่ม data-bs-toggle="modal" data-bs-target="#popuplogout" สำหรับการเรียกใช้งาน modal ของ bootstrap และเพิ่มส่วนของการแสดงผล modal ในส่วนของ tag a สำหรับการเรียกใช้งานการ logout ให้เรียกใช้งาน url name : logoutaction ที่ได้สร้างไว้ในขั้นตอนก่อนหน้า {% url 'logoutaction' %}

```Python
<div class="dropdown-menu dropdown-menu-end">
```

```
                    <a class="dropdown-item" href="{% url 'backendpage' %}"><i
class="align-middle me-1" data-feather="user"></i> Profile</a>
                    <a class="dropdown-item" href="#" data-bs-toggle="modal"
data-bs-target="#popuplogout"><i class="align-middle me-1"
data-feather="log-out"></i> Log out</a>
                </div>
                <div class="modal fade" id="popuplogout"
data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"
aria-labelledby="staticBackdropLabel" aria-hidden="true">
                    <div class="modal-dialog">
                        <div class="modal-content">
                            <div class="modal-header">
                            <h1 class="modal-title fs-5"
id="staticBackdropLabel">Logout</h1>
                            <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                            </div>
                            <div class="modal-body">
                                <h1 class="text-center">ยืนยันการ Logout</h1>
                            </div>
                            <div class="modal-footer">
                            <a href="{% url 'logoutaction' %}" class="btn
btn-primary">Confirm</a>
                            <button type="button" class="btn btn-secondary"
data-bs-dismiss="modal">Close</button>
                            </div>
                        </div>
   </div>
</div>
```

การจัดเตรียมเมนู Menu

- แก้ไขข้อความด้านบนของเมนู

ไปที่ไฟล์ menu.html ที่อยู่ใน folder include แก้ไขข้อความ AdminKit ให้เป็นชื่อระบบที่ต้องการ ในตัวอย่างจะแก้ไขเป็น WEBAi

และแก้ไข href เดิม ให้เป็น {% url 'backendpage' %} เมื่อคลิกที่ข้อความแล้ว ให้วิ่งกลับไปที่หน้าหลักของสมาชิก

Python

```
<a class="sidebar-brand" href="{% url 'backendpage' %}">
      <span class="align-middle">WEBAi</span>
</a>
```

- สร้างเมนูใหม่ประกอบด้วย

โดยใช้วิธีการเปลี่ยนข้อความจากเมนูเดิม แก้ไขลิงก์ซึ่งปัจจุบันสามารถกำหนดได้แค่หน้า Home (หมายถึงหน้า home ของสมาชิกที่ผ่านการ login มาแล้ว) เท่านั้น คือ backendpage เนื่องจากหน้าเพจอื่นๆยังไม่ได้มีการสร้าง และไอคอนใหม่ตามที่ต้องการ สามารถดูรายการไอคอนได้จาก https://feathericons.com ในส่วนของ data-feather="home" จาก home เป็นชื่อไอคอนตามหน้าเว็บ

- เมนู Home

```
Python
<li class="sidebar-item active">
                <a class="sidebar-link" href="{% url 'backendpage' %}">
                    <i class="align-middle" data-feather="home"></i> <span
class="align-middle">Home</span>
                </a>
</li>
```

- เมนู Prediction

```
Python
<li class="sidebar-item">
   <a class="sidebar-link" href="pages-profile.html">
       <i class="align-middle" data-feather="codepen"></i> <span
class="align-middle">Prediction</span>
   </a>
</li>
```

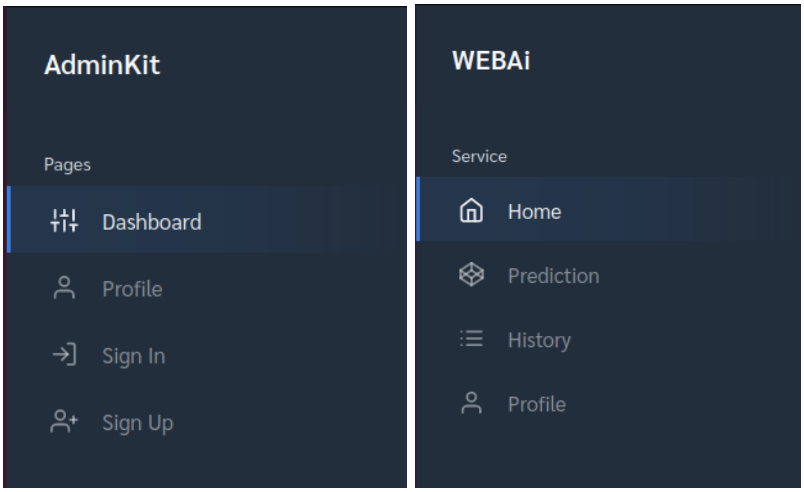
- เมนู History

```
Python
<li class="sidebar-item">
  <a class="sidebar-link" href="pages-sign-in.html">
     <i class="align-middle" data-feather="list"></i> <span
class="align-middle">History</span>
   </a>
</li>
```

- เมนู Profile

```python
<li class="sidebar-item">
   <a class="sidebar-link" href="pages-sign-up.html">
      <i class="align-middle" data-feather="user"></i> <span
class="align-middle">Profile</span>
   </a>
</li>
```

สรุป menu.html



การจัดเตรียม Footer

ไปที่ไฟล์ footer.html อยู่ใน folder include

- เปลี่ยนข้อความ (ด้านขวามือ) เช่นตัวอย่าง WEBAi by Pongthorn - 2023 Demo version ©
- เพิ่มลิงก์สำหรับข้อมูลอื่นๆ (ด้านซ้ายมือ) เพิ่มลิงก์ และการค้นหาข้อมูลอื่นๆ ดังตัวอย่าง

Python

```
<footer class="footer">
    <div class="container-fluid">
        <div class="row text-muted">
            <div class="col-6 text-start">
                <p class="mb-0">
                    <strong>WEBAi by Pongthorn</strong> - 2023 Demo version
&copy;
                </p>
            </div>
            <div class="col-6 text-end">
                <ul class="list-inline">
                    <li class="list-inline-item">
                        <a class="text-muted" href="#">คู่มือการใช้งาน</a>
                    </li>
                    <li class="list-inline-item">
                        <a class="text-muted" href="#">ปัญหาการใช้งาน</a>
                    </li>
                    <li class="list-inline-item">
                        <a class="text-muted" href="#">ความเป็นส่วนตัว</a>
                    </li>
                    <li class="list-inline-item">
```

```
                        <a class="text-muted" href="#">ทีม</a>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</footer>
```

```
urls.py  views.py  footer.html
webai > template > include > footer.html > ...
<footer class="footer">
    <div class="container-fluid">
        <div class="row text-muted">
            <div class="col-6 text-start">
                <p class="mb-0">
                    <strong>WEBAi by Pongthorn</strong> - 2023 Demo version &copy;
                </p>
            </div>
            <div class="col-6 text-end">
                <ul class="list-inline">
                    <li class="list-inline-item">
                        <a class="text-muted" href="#">คู่มือการใช้งาน</a>
                    </li>
                    <li class="list-inline-item">
                        <a class="text-muted" href="#">ปัญหาการใช้งาน</a>
                    </li>
                    <li class="list-inline-item">
                        <a class="text-muted" href="#">ความเป็นส่วนตัว</a>
                    </li>
                    <li class="list-inline-item">
                        <a class="text-muted" href="#">ทีม</a>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</footer>
```

WEBAi by Pongthorn - 2023 Demo version © คู่มือการใช้งาน ปัญหาการใช้งาน ความเป็นส่วนตัว ทีม

20. การแสดงแถบสีฟ้าเมื่อมีการเรียกใช้งานเมนูนั้นๆ (Active menu)

ในการทำ active menu (แสดงแถบสีฟ้าเมื่อมีการเรียกใช้งานเมนูนั้นๆ ดังภาพ) จะใช้ session มาช่วยในการตรวจสอบการเรียกใช้งานเมนู

โดยกำหนดใน Views.py และนำค่าไปเช็คในส่วนของเมนู menu.html เช่น

กำหนดในฟังก์ชั่น backendpage

คือฟังก์ชั่นสำหรับหน้าหลักของสมาชิกหลังจากการ login ให้สร้างตัวแปร session เก็บข้อมูล request.session['menuactive'] = "backendpage" โดยที่ request.session['menuactive'] คือการสร้างตัวแปรแบบ session ที่มีชื่อว่า menuactiveและ กำหนดค่าให้เป็น "backendpage" และให้ทำทุกๆ ฟังก์ชั่นให้กำหนดเป็นตัวแปรเดิม request.session['menuactive'] = และเปลี่ยนแปลงค่าให้เป็นของเพจนั้นๆ

เมนู และค่าในตัวแปร

- Home "backendpage"
  request.session['menuactive'] = "backendpage"
- Prediction "prediction"
  request.session['menuactive'] = "prediction"
- History "history"
  request.session['menuactive'] = "history"
- Profile "profile"
  request.session['menuactive'] = "profile"

```python
request.session['menuactive'] = "backendpage"
```

การเรียกใช้งานในไฟล์ menu.html

ให้เรียกใช้งานโดยมีการตรวจสอบว่าตรงกับเมนูนั้น ๆ หรือไม่ ดังตัวอย่าง {% if request.session.menuactive == 'backendpage' %} active {% endif %} โดยที่

request.session.menuactive หมายถึงการเรียกใช้งานตัวแปรประเภท session ชื่อตัวแปร menuactive แล้วนำมาเช็คในเงื่อนไขว่าเท่ากับ 'backendpage' หรือไม่ หากใช่ให้แสดงค่า active ในเมนูนั้นๆ

```python
{% if request.session.menuactive == 'backendpage' %} active {% endif %}
```

รวมเมนูทั้งหมด 4 เมนู

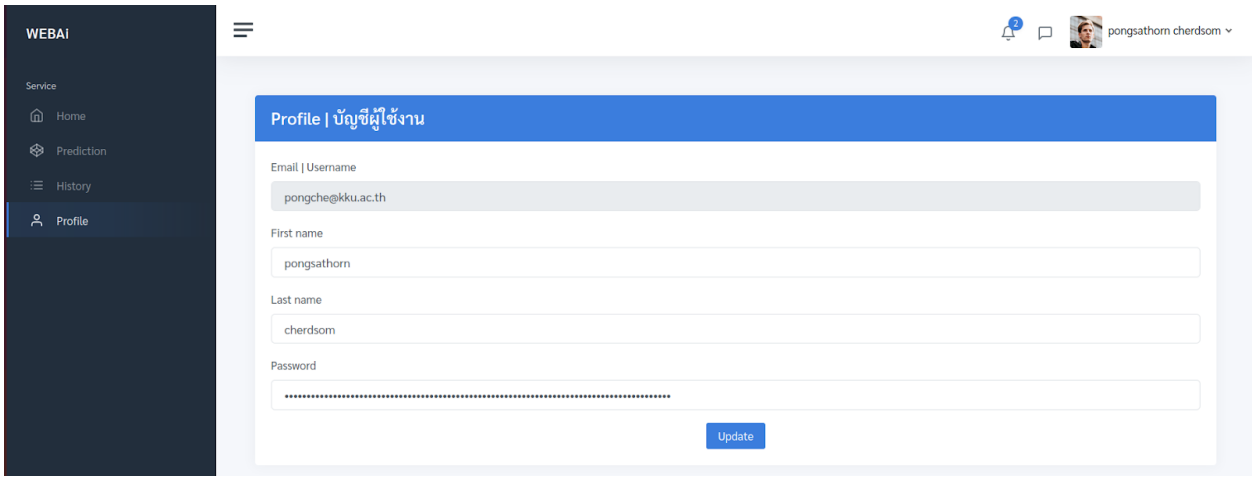
```python
<li class="sidebar-item {% if request.session.menuactive == 'backendpage' %}
active {% endif %}">
     <a class="sidebar-link" href="{% url 'backendpage' %}">
                    <i class="align-middle" data-feather="home"></i> <span
class="align-middle">Home</span>
     </a>
</li>

<li class="sidebar-item {% if request.session.menuactive == 'prediction' %}
active {% endif %}">
    <a class="sidebar-link" href="pages-profile.html">
                    <i class="align-middle" data-feather="codepen"></i> <span
class="align-middle">Prediction</span>
```

```
                </a>
            </li>

            <li class="sidebar-item {% if request.session.menuactive ==
'history' %} active {% endif %}">
                <a class="sidebar-link" href="pages-sign-in.html">
                    <i class="align-middle" data-feather="list"></i> <span
class="align-middle">History</span>
                </a>
            </li>

            <li class="sidebar-item {% if request.session.menuactive ==
'profile' %} active {% endif %}">
                <a class="sidebar-link" href="{% url 'profilepage' %}">
                    <i class="align-middle" data-feather="user"></i> <span
class="align-middle">Profile</span>
                </a>
</li>
```

webai > template > include > menu.html

```
<nav id="sidebar" class="sidebar js-sidebar">
    <div class="sidebar-content js-simplebar">
        <a class="sidebar-brand" href="{% url 'backendpage' %}">
            <span class="align-middle">WEBAi</span>
        </a>

        <ul class="sidebar-nav">
            <li class="sidebar-header">
               Service
            </li>

            <li class="sidebar-item {% if request.session.menuactive == 'backendpage' %} active {% endif %}">
                <a class="sidebar-link" href="{% url 'backendpage' %}">
                    <i class="align-middle" data-feather="home"></i> <span class="align-middle">Home</span>
                </a>
            </li>

            <li class="sidebar-item {% if request.session.menuactive == 'prediction' %} active {% endif %}">
                <a class="sidebar-link" href="pages-profile.html">
                    <i class="align-middle" data-feather="codepen"></i> <span class="align-middle">Prediction</span>
                </a>
            </li>

            <li class="sidebar-item {% if request.session.menuactive == 'history' %} active {% endif %}">
                <a class="sidebar-link" href="pages-sign-in.html">
                    <i class="align-middle" data-feather="list"></i> <span class="align-middle">History</span>
                </a>
            </li>

            <li class="sidebar-item {% if request.session.menuactive == 'profile' %} active {% endif %}">
                <a class="sidebar-link" href="{% url 'profilepage' %}">
                    <i class="align-middle" data-feather="user"></i> <span class="align-middle">Profile</span>
                </a>
            </li>

        </ul>
    </div>
</nav>
```

เมื่อเรียกใช้งานเมนู Profile ก็จะแสดงแถบสีฟ้าที่เมนู Profile



21. ตรวจสอบการเรียกใช้งานฟังก์ชั่น สำหรับสมาชิกของระบบ

การตรวจสอบสิทธิ์ต่างๆ ก่อนเรียกใช้งานฟังก์ชั่นอื่นๆ สำหรับสมาชิกที่ผ่านการ login มาแล้วเท่านั้น และแยกสำหรับ admin หรือ สำหรับ user
ซึ่งสามารถใช้ฟังก์ชั่นที่ django ได้จัดเตรียมไว้ให้ ได้แก่ staff_member_required() สำหรับเช็คว่า user ที่เข้าใช้งาน function มีการลงชื่อเข้าใช้งานและเป็น staff หรือไม่
และ ใช้ login_required() สำหรับเช็คว่ามีการลงชือเข้าใช้งานแล้วหรือไม่

Python
```python
from django.contrib.admin.views.decorators import staff_member_required
from django.contrib.auth.decorators import login_required


@staff_member_required(login_url='url loginpage or name url login')
def testfun():
      pass

@login_required(login_url='url loginpage or name url login')
def testuser():
      pass
```

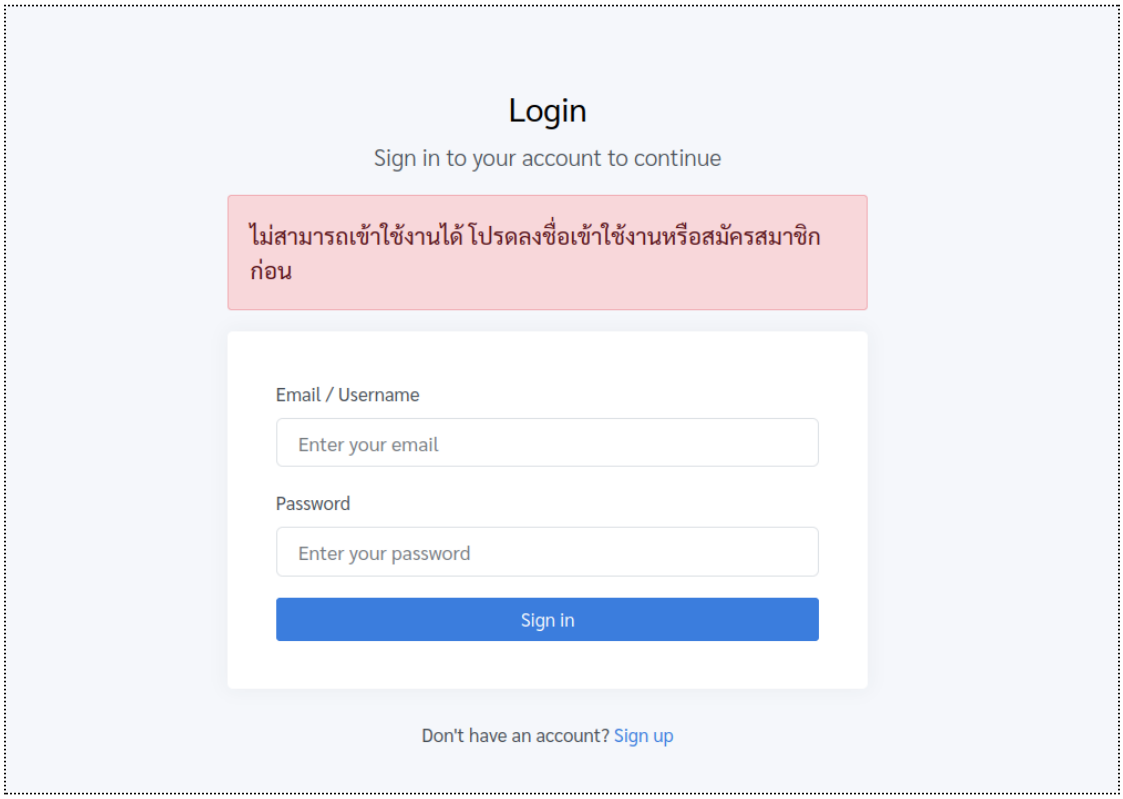
แต่ใน workshop นี้จะใช้วิธีการเช็คสถานะการลงชื่อเข้าใช้งานจากตัวแปร request.user.is_authenticated และเช็คสถานะผู้ใช้งานเป็น admin หรือ user request.user.is_staff และในแต่ละกรณีหากส่งค่าเป็น False จะให้ redirect พร้อมกับแสดงข้อความผิดพลาด (alert message) กลับไปที่หน้าของการ redirect

ทำการตรวจสอบสิทธิ์ของฟังก์ชั่น backendpage() ที่เป็นหน้าหลักของสมาชิก หลังจากการ login แล้ว ซึ่งในหน้านี้สามารถเข้าใช้งานได้ทั้ง user และ admin จึงเรียกใช้แค่ request.user.is_authenticated ในการตรวจสอบการ login ก็พอ และกรณี Flase ยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”

ซึ่งในแต่ละเพจ หรือ ฟังก์ชั่น ถ้าสำหรับสมาชิกที่มีการลงชื่อเข้าใช้งาน และหรือ เฉพาะ สิทธิ์แอดมิน ให้มีการเช็คในรูปแบบนี้ทุกๆเพจหรือฟังก์ชั่น

```python
def backendpage(request):
    if request.user.is_authenticated: <== ตรวจสอบการ login
        context = {}
        return render(request,'backendpage.html',context)
    else: <== กรณีไม่มีการ login
        messages.error(request,'ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน')
        return redirect(homepage)
```

กรณีเข้าใช้งานหน้าที่ยังไม่ได้ลงชื่อเข้าใช้งาน จะวิ่งกลับมาที่หน้าหลักของระบบ พร้อมกับแสดงข้อความดังภาพ



22. การสร้างหน้าเพจ Profile สำหรับแสดงข้อมูลผู้ใช้งาน และการแก้ไขข้อมูล

File : urls.py (อยู่ใน webai)

สร้างหน้าเพจ profile
ซึ่งการกำหนดเส้นทาง ได้กำหนดให้อยู่ภายใต้ /backend เพื่อเป็นการจัดกลุ่มของการทำงานและฟังก์ชั่น โดยในตัวอย่างนี้ฟังก์ชั่นที่ทำงานหลังจากการ login จะให้อยู่ในกลุ่มของ /backend ตามด้วย / ชื่อของเพจนั้น หรือ คำสั่งนั้นๆ

```python
path('backend/profile/', views.profilepage, name='profilepage'),
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่น profilepage สำหรับการแสดงหน้า profile ของผู้ใช้งาน โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ส่งค่าข้อมูลผู้ใช้งาน จากการ query จาก database ไปใช้งาน (ในตัวอย่างใช้วิธีการนี้) ซึ่งสามารถใช้วิธีแสดงผลจากตัวแปร {{ requests.user.<> }} ผ่านหน้า html ได้เลยเช่นกัน และใช้วิธีการนี้เนื่องจากจะให้เรียนรู้การส่งข้อมูลจากฟังก์ชั่นไปแสดงผลที่หน้า html
  โดยใช้วิธี getuser = User.objects.get(id=request.user.id) คือ getuser คือตัวแปรที่รับข้อมูลจากการ query User คือ ชื่อฐานข้อมูล get() คือการกำหนดเงื่อนไขในการค้นหาข้อมูลจากฐานข้อมูล และมีความแตกต่างจาก filter() คือ หาก get() แล้วมีข้อมูลมากกว่า 1 แถว จะเกิดการ error ขึ้นทันที
  จึงมีการเรียกใช้งาน try: except: เพื่อดัก error จะเกิดขึ้น หากคำสั่งที่อยู่ใน try ทำงานไม่ได้ จะวิ่งไปทำงานที่ except ทันที และในตัวอย่าง หาก error ตัวแปร getuser จะมีค่าเป็น None
  การส่งค่าเพื่อไปแสดงผลหน้า html จะส่งข้อมูล โดยจะแนบเข้าไปในตัวแปร context ที่สร้างไว้ในรูปแบบของ dictionary ซึ่ง context = { 'getuser' : getuser } 'getuser' ฝั่งซ้ายหมายถึงชื่อตัวแปรเรียกใช้งานจากหน้า html และ getuser ฝั่งขวาคือตัวแปรข้อมูลที่มาจากและเกิดในฟังก์ชั่น ฝั่งหลังบ้าน

Template : profile.html

สร้างไฟล์ profile.html มาใช้ในฟังก์ชั่น profilepage ซึ่งมีการส่งข้อมูลจาก views มาที่หน้าบ้านโดยตัวแปร getuser ซึ่งในหน้าเพจนี้จะมีลักษณะเหมือนกับหน้า registerpage จึงสามารถคัดลอกมาใช้งานได้ เฉพาะส่วนของ from เท่านั้น
ในฟอร์มจะมีทั้งหมด 4 input ในแต่ละ input ให้เติมคำสั่ง value="{{ getuser.first_name }}" เพื่อเป็นการนำค่าจากตัวแปรมาเติมใน input แต่ละช่อง ซึ่งตัวแปร getuser มากจากตัวแปร คั่นด้วย . ส่วน first_name คือช่อคอลัมน์ของข้อมูลในฐานข้อมูล User

- Input Email หรือ Username จะปิดไม่ให้กรอกข้อมูล ให้แสดงผลเท่านั้น (กรณีนี้ไม่อนุญาตให้ผู้ใช้งานเปลี่ยนอีเมลในระบบ) โดยใช้คำสั่ง disabled ใน tag input
- Input อื่นๆ value ให้กำหนดให้ตรงกับแต่ละ input
- และ action ไปที่ url

Python

```
<form action="{% url 'profileupdatedata' %}" method="POST">
    {% csrf_token %}
    <div class="mb-3">
        <label class="form-label">Email | Username</label>
        <input class="form-control form-control-lg" type="email" name="email"
value="{{ getuser.username }}" disabled>
    </div>
    <div class="mb-3">
        <label class="form-label">First name</label>
        <input class="form-control form-control-lg" type="text"
name="firstname" placeholder="Enter your name" required value="{{
getuser.first_name }}"/>
    </div>
    <div class="mb-3">
        <label class="form-label">Last name</label>
        <input class="form-control form-control-lg" type="text" name="lastname"
placeholder="Enter your name" required value="{{ getuser.last_name }}"/>
    </div>
    <div class="mb-3">
        <label class="form-label">Password</label>
        <input class="form-control form-control-lg" type="password"
name="password" placeholder="Enter password" required value="{{
getuser.password }}"/>
    </div>
    <div class="mt-3 text-center">
        <button type="submit" class="btn btn-primary">Update</button>
    </div>
</form>
```

23. สร้างฟังก์ชั่น Update user ข้อมูลผู้ใช้งาน

File : urls.py (อยู่ใน webai)

```
Python
path('backend/profile/updatedata/', views.profileupdatedata,
name='profileupdatedata'),
```

File : views.py (อยู่ใน backendai)

เรียกใช้งานฟังก์ชั่น make_password() โดยให้เพิ่ม

from django.contrib.auth.hashers import make_password

ไว้ด้านบนของไฟล์ views.py

สร้างฟังก์ชั่น profileupdatedata สำหรับการอัพเดตข้อมูลผู้ใช้งาน โดยลักษะการทำงานเหมือนกับฟังก์ชั่น registercreate แต่ฟังก์ชั่นนี้จะมีการเช็คสิทธิ์ในการเข้าใช้งานก่อน

เงื่อนไขทำงาน

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- เช็ค method ที่ส่งมาจากหน้า from updateuser เป็น POST หรือไม่ หากไม่ใช่ให้ redirect กลับไปที่หน้า profilepage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
- เช็คตัวแปรที่รับค่าจาก from ไม่ให้มีค่าใดเป็นค่าว่าง โดยการใช้คำสั่ง is not None และใช้เงื่อไขแบบ and โดยทุกๆตัวแปรต้องไม่เป็นค่าว่าง หากเป็นค่าว่างให้ redirect กลับไปที่หน้า profilepage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
- เช็คการเปลี่ยนแปลงรหัสผ่าน จากหน้าฟอร์ม html เทียบกับ ฐานข้อมูล (จากตัวแปร request.user.password) ว่ามีค่าตรงกันหรือไม่ ถ้าไม่ตรงแสดงว่ารหัสผ่านที่มาจากหน้าฟอร์ม html มีการเปลี่ยนแปลง (เช่น กรอกรหัสผ่านใหม่) จะมีการเข้ารหัสผ่าน (Encode) รหัสผ่านใหม่ก่อนนำไปอัพเดตในฐานข้อมูล ถ้ามีมีการเปลี่ยนแปลงจะไม่อัพเดตข้อมูลรหัสผ่าน
- เช็คการบันทึกอัพเดตข้อมูลเข้าสู่ฐานข้อมูลถ้าหากเสร็จสมบูรณ์ให้ redirect กลับไปที่หน้า profilepage พร้อมกับระบุข้อข้อความ alert message รูปแบบ success “อัพเดตข้อมูลผู้ใช้งานเสร็จสมบูรณ์” หากเกิดข้อผิดพลาด ให้ redirect กลับไปที่หน้า profilepage พร้อมกับ

ระบุข้อข้อความ alert message รูปแบบ error “ไม่สามารถอัพเดตข้อมูลผู้ใช้งานได้ โปรดลองใหม่อีกครั้ง”

คำสั่งการอัพเดตข้อมูล

- กำหนดแถวของข้อมูลที่ต้องการอัพเดต โดยใช้ User.objects.get(id=request.user.id) ค้นหาข้อมูล id ในฐานข้อมูล User โดยค้นหาจากตัวแปร request.user.id ซึ่งเป็นข้อมูลที่ได้หลังจากการ login ดังนั้น ก็จะเป็นข้อมูล id ของผู้เรียกใช้งานฟังก์ชั่นนี้เท่านั้น และมีตัวแปร updatuser มารับเพื่อดำเนินการต่อ
- อัพเดตข้อมูลคอลัมน์ first_name และ last_name โดยที่ updatuser.first_name updatuser คือตัวแปรที่รับค่ามากจากการ query บรรทัดแรก คั่นด้วย . ตามด้วยชื่อของคอลัมน์ในตาราง User ซึ่งสามารถดูได้จากฐานข้อมูล (กรณีนี้ใช้งานใน phpmyadmin ให้เปิดดูที่ตาราง auth_user)
- อัพเดตข้อมูลคอลัมน์ password จะมีการเช็คก่อนการอัพเดต และใช้ฟังก์ชั่น make_password() แปลงรหัสผ่านก่อนเก็บเข้าฐานข้อมูล
- และสุดท้าย updatuser.save() คือคำสั่งในการบันทึกข้อมูล
- เมื่อมีการอัพเดตข้อมูลรหัสผ่านของผู้ใช้งาน django จะให้ลงชื่อเข้าใช้งานระบบใหม่อีกรอบก่อนเข้าใช้งาน

```python
updatuser = User.objects.get(id=request.user.id)
updatuser.first_name = firstnameinput
updatuser.last_name = lastnameinput
if passwordinput != request.user.password:
      updatuser.password = make_password(passwordinput)
updatuser.save()
```

ขั้นตอนการเข้ารหัส (Encode) ของรหัสผ่าน โดยใช้ฟังก์ชั่น make_password()

อัพเดต URL ของเมนู Profile

ในไฟล์ menu.html และ header.html ที่อยู่ใน folder inclue
อัพเดตเมนูในส่วนของเมนู profile tag a href ให้เป็น {% url 'profilepage' %}

Python

FileName : menu.html

```
<li class="sidebar-item">
                <a class="sidebar-link" href="{% url 'profilepage' %}">
                    <i class="align-middle" data-feather="user"></i> <span
class="align-middle">Profile</span>
                </a>
</li>
```

Python

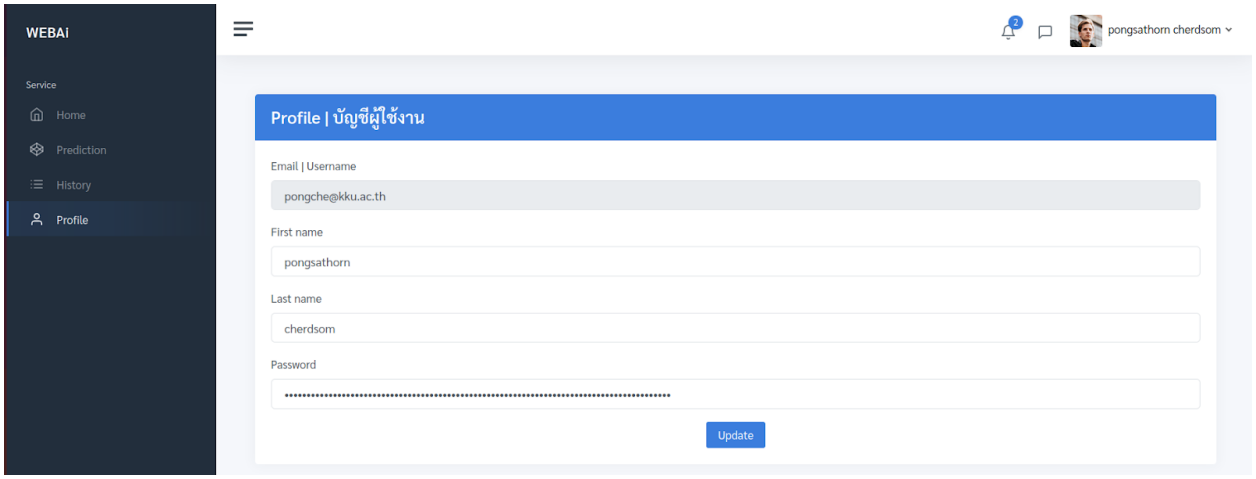
FileName : header.html

```
<a class="dropdown-item" href="{% url 'profilepage' %}"><i class="align-middle
me-1" data-feather="user"></i> Profile </a>
```

ทดสอบ Runserver

python3 manage.py runserver

http://127.0.0.1:8000/backend/profile/



สรุปฟังก์ชั่น

Python

```
def profileupdatedata(request):
    if request.user.is_authenticated:
        if request.method == "POST":
            firstnameinput = request.POST.get('firstname',None)
```

```
            lastnameinput = request.POST.get('lastname',None)
            passwordinput = request.POST.get('password',None)

            if firstnameinput is not None and lastnameinput is not None and
passwordinput is not None:

                updatuser = User.objects.get(id=request.user.id)
                updatuser.first_name = firstnameinput
                updatuser.last_name = lastnameinput
                if passwordinput != request.user.password:
                    updatuser.password = make_password(passwordinput)
                updatuser.save()

                if updatuser:
                    messages.success(request,'อัพเดตข้อมูลผู้ใช้งานเสร็จสมบูรณ์')
                    return redirect(profilepage)
                else:
                    messages.error(request,'ไม่สามารถอัพเดตข้อมูลผู้ใช้งานได้ โปรดลองใหม่
อีกครั้ง')
                    return redirect(profilepage)
            else:
                messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
                return redirect(profilepage)
        else:
            messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
            return redirect(profilepage)
    else:
        messages.error(request,'ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิก
ก่อน')
        return redirect(homepage)
```

```python
def profileupdatedata(request):
    if request.user.is_authenticated:
        if request.method == "POST":
            firstnameinput = request.POST.get('firstname',None)
            lastnameinput = request.POST.get('lastname',None)
            passwordinput = request.POST.get('password',None)

            if firstnameinput is not None and lastnameinput is not None and passwordinput is not None:

                updatuser = User.objects.get(id=request.user.id)
                updatuser.first_name = firstnameinput
                updatuser.last_name = lastnameinput
                if passwordinput != request.user.password:
                    updatuser.password = make_password(passwordinput)
                updatuser.save()

                if updatuser:
                    messages.success(request,'อัพเดตข้อมูลผู้ใช้งานเสร็จสมบูรณ์')
                    return redirect(profilepage)
                else:
                    messages.error(request,'ไม่สามารถอัพเดตข้อมูลผู้ใช้งานได้ โปรดลองใหม่อีกครั้ง')
                    return redirect(profilepage)
            else:
                messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
                return redirect(profilepage)
        else:
            messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
            return redirect(profilepage)
    else:
        messages.error(request,'ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน')
        return redirect(homepage)
```

24. การสร้างฐานข้อมูล

ในการสร้างฐานขอมูล จะสร้างในไฟล์ models.py ที่อยู่ใน folder backendai (คือ folder ของ app ที่สร้างขึ้นใหม่)

ใน workshop นี้จะสร้างฐานข้อมูลเพื่อเก็บข้อมูลการ prediction จาก model ai จากฝั่งของ roboflow api ชื่อตาราง predictprocess ประกอบไปด้วย

- indexpredict Primary Key ของตาราง
- refuser ผู้ใช้งานที่ทำรายการ
- imagefile ชื่อไฟล์รูปภาพที่นำมา พยากรณ์
- resultpredict ผลการพยากรณ์
- datetimepre เก็บวันที่ เวลา ของการทำรายการ
- uuidpredict uuid ของรายการ (เพื่อนำไปเรียกดูประวัติ)

ซึ่งในตาราจะมีการเรียกใช้งานตารางข้อมูลผู้ใช้งาน User จึงต้อง import เข้ามาให้ models.py รู้จัก from django.contrib.auth.models import User

- indexpredict = models.AutoField(primary_key=True) โดยที่ AutoField ให้มีการรันตัวเลขแบบอัตโนมัติ มีความหมายเดียวกับ Auto increment และ primary_key=True กำหนดให้เป็น primary key ของตาราง
- refuser = models.ForeignKey(User,on_delete=models.CASCADE) โดยที่ ForeignKey หมายถึงกำหนดให้เป็น Foreign Key ของตารางและ User คือตารางที่ต้องการเชื่อมโยง และ on_delete หมายถึงเมื่อข้อมูลที่ใช้เชื่อมโยงมีการลบออกจากฐานข้อมูล ตารางที่นำมาอ้างอิง

จะมีการทำงานอย่างไร models.CASCADE ให้ทำการลบข้อมูลไปด้วย models.SET_NULL หมายถึง กำหนดให้คอลัมน์เป็นค่าว่าง

- imagefile = models.CharField(max_length=150) โดยที่ CharField คือข้อมูลประเภทตัวอักษร ความหมายเดียวกับ varchar ส่วน max_length หมายถึง ขนาดจำนวนตักอักษรมากที่สุด
- typepredict = models.CharField(max_length=10) โดยที่ CharField คือข้อมูลประเภทตัวอักษร ความหมายเดียวกับ varchar ส่วน max_length หมายถึง ขนาดจำนวนตักอักษรมากที่สุด
- confidence = models.IntegerField() โดยที่ IntegerField คือข้อมูลประเภทตัวอักษร ความหมายเดียวกับ int
- overlap = models.IntegerField() โดยที่ IntegerField คือข้อมูลประเภทตัวอักษร ความหมายเดียวกับ int
- resultpredict = models.TextField() โดยที่ TextField หมายถึงข้อมูลประเภท
- datetimepre = models.DateTimeField(auto_now=True) โดยที่ DateTimeField หมายถึงข้อมูลประเภท DateTime ที่มีการกำหนด auto_now=True หมายถึงให้เพิ่มเวลาปัจจุบันเข้าไปโดยอัตโนมัติ
- uuidpredict = models.CharField(max_length=30) โดยที่ CharField คือข้อมูลประเภทตัวอักษร ที่มีขนาดไม่เกิดน max_length 30 ตัวอักษร

Python
```python
from django.db import models
from django.contrib.auth.models import User

# Create your models here.
class Predictprocess(models.Model):
    indexpredict = models.AutoField(primary_key=True)
    refuser = models.ForeignKey(User,on_delete=models.SET_NULL,null=True)
    imagefile = models.CharField(max_length=150)
    typepredict = models.CharField(max_length=10)
    confidence = models.IntegerField()
    overlap = models.IntegerField()
    resultpredict = models.TextField()
    datetimepre = models.DateTimeField(auto_now=True)
    uuidpredict = models.CharField(max_length=30)
```

สร้างตาราง โดยใช้คำสั่ง

- python3 manage.py makemigrations เป็นการสร้างข้อมูลจากไฟล์ models.py ให้เป็น schema พร้อมที่จะนำไปสร้างฐานข้อมูล

```
^C(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ python3 manage.py makemigrations
Migrations for 'backendai':
  backendai/migrations/0001_initial.py
    - Create model Predictprocess
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$
```

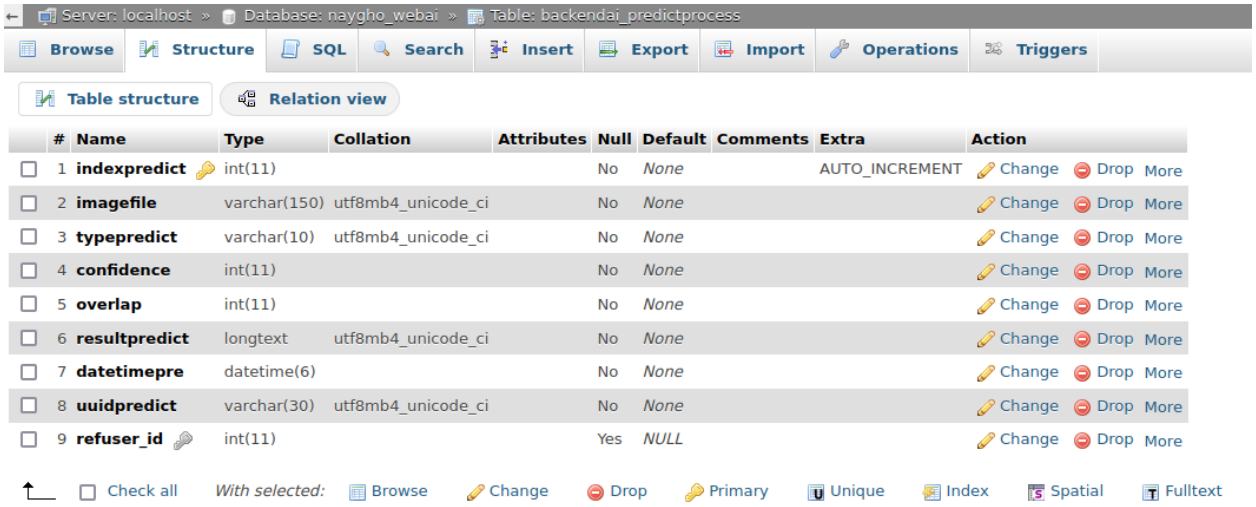
backendai > migrations > 0001_initial.py > Migration > operations

```python
# Generated by Django 4.2.7 on 2023-12-11 08:48

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='Predictprocess',
            fields=[
                ('indexpredict', models.AutoField(primary_key=True, serialize=False)),
                ('imagefile', models.CharField(max_length=150)),
                ('typepredict', models.CharField(max_length=10)),
                ('confidence', models.IntegerField()),
                ('overlap', models.IntegerField()),
                ('resultpredict', models.TextField()),
                ('datetimepre', models.DateTimeField(auto_now=True)),
                ('uuidpredict', models.CharField(max_length=30)),
                ('refuser', models.ForeignKey(null=True, on_delete=django.db.models.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]
```

- python3 manage.py migrate เป็นการนำ schema ที่แปลงไว้ไปสร้างเป็นฐานข้อมูลในฐานข้อมูลที่เชื่อมต่อไว้

```
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$ python3 manage.py migrate
System check identified some issues:

WARNINGS:
?: (mysql.W002) MariaDB Strict Mode is not set for database connection 'default'
        HINT: MariaDB's Strict Mode fixes many data integrity problems in MariaDB, such as data truncation upon i
nsertion, by escalating warnings into errors. It is strongly recommended you activate it. See: https://docs.djang
oproject.com/en/4.2/ref/databases/#mysql-sql-mode
Operations to perform:
  Apply all migrations: admin, auth, backendai, contenttypes, sessions
Running migrations:
  Applying backendai.0001_initial... OK
(env) pongthorn@pongthorn-site:~/Documents/webai/webai$
```

และจะได้ฐานข้อมูลเพิ่มขึ้นมาดังภาพ ซึ่งตารางจะขึ้นต้นด้วยชื่อของ app ที่สร้าง database (models.py อยู่ใน app นั้นๆ) ตามด้วยชื่อของฐานข้อมูลใน models.py ที่กำหนดไว้ backendai_predictprocess แต่ในขั้นตอนการเรียกใช้งาน ก็เรียกใช้งานเป็นชื่อที่สร้าง class ไว้ใน models.py เช่นเดิม Predictprocess



25. การสร้างหน้าเพจสำหรับ Prediction

File : urls.py (อยู่ใน webai)

```python
path('backend/prediction/', views.predictionpage, name='predictionpage'),
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่น predictionpage ในไฟล์ views.py สำหรับส่งข้อมูลไปประมวลผลกับ model ai ผ่านทาง api

โดยมีเงื่อนไขสำหรับการตรวจสอบ

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”

คำสั่งภายในฟังก์ชั่น

- กำหนดตัวแปร session ของ menuactive ให้เป็น prediction (ที่จะนำไปเช็คในไฟล์ menu.html เพื่อให้เมนูนั้นๆ active)
- ส่งค่าตัวแปร

Template : prediction.py (อยู่ใน webai)

สร้างไฟล์ prediction.html โดยใช้โครงสร้างเดียวกับ backendpage.html

โดยที่

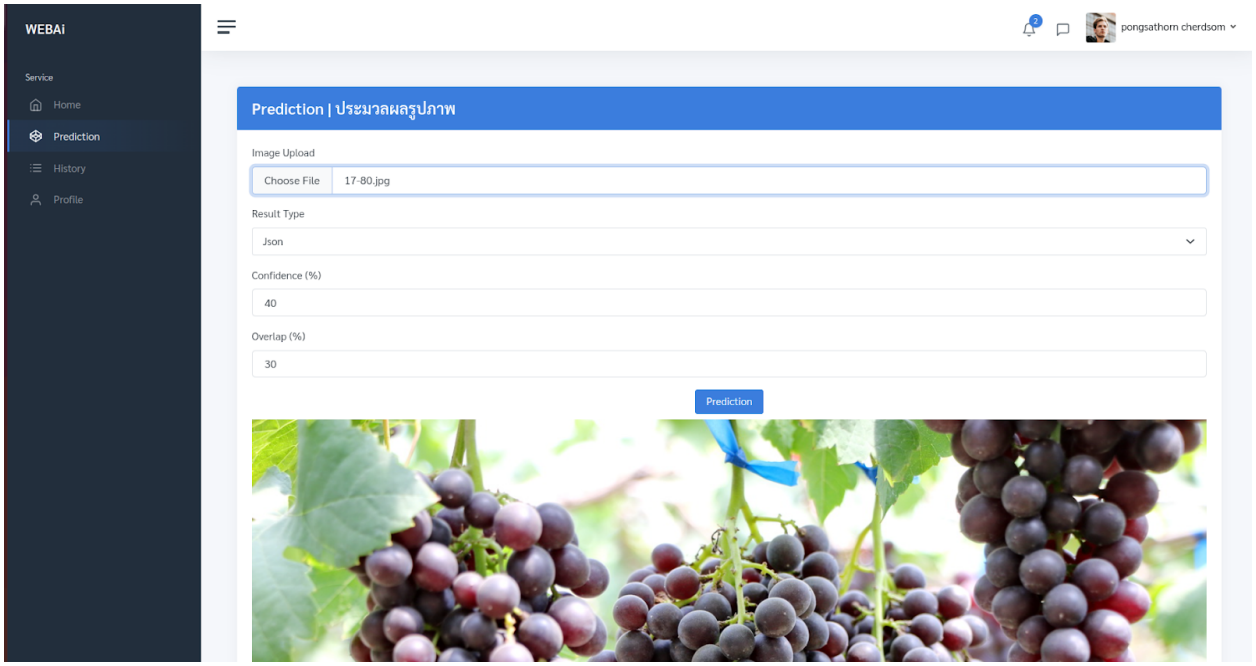
- ใน form เพิ่ม enctype="multipart/form-data" เข้าไปเพื่อทำการอัปโหลดไฟล์รูปภาพ
- ข้อมูลภายในฟอร์มประกอบด้วย (input)
    - input imagefile type : file คือสำหรับ upload file รูปภาพ ที่จำกัดให้อัพโหลดเฉพาะนามสกุลของรูปภาพ (accept="image/*") และบังคับให้กรอก (required)
    - input resulttype type select
    - input confidence type : number และได้กำหนดค่าตั้งต้นให้เป็น 40 (value="40") ค่าสูงที่สุดให้เป็น 100 (max="100") และบังคับให้กรอก (required)
    - input overlap type : number และได้กำหนดค่าตั้งต้นให้เป็น 30 (value="30") ค่าสูงที่สุดให้เป็น 100 (max="100") และบังคับให้กรอก (required)
- นำรูปภาพจาก input imagefile มาแสดงตัวอย่างของรูปภาพ โดยกำหนดใน input imagefile (รูปภาพ) ให้ id="imagefile" onchange="showimage(event)" เพื่อนำมาใช้งานกับ javascript ซึ่งนำมาแสดงผลในส่วนของ

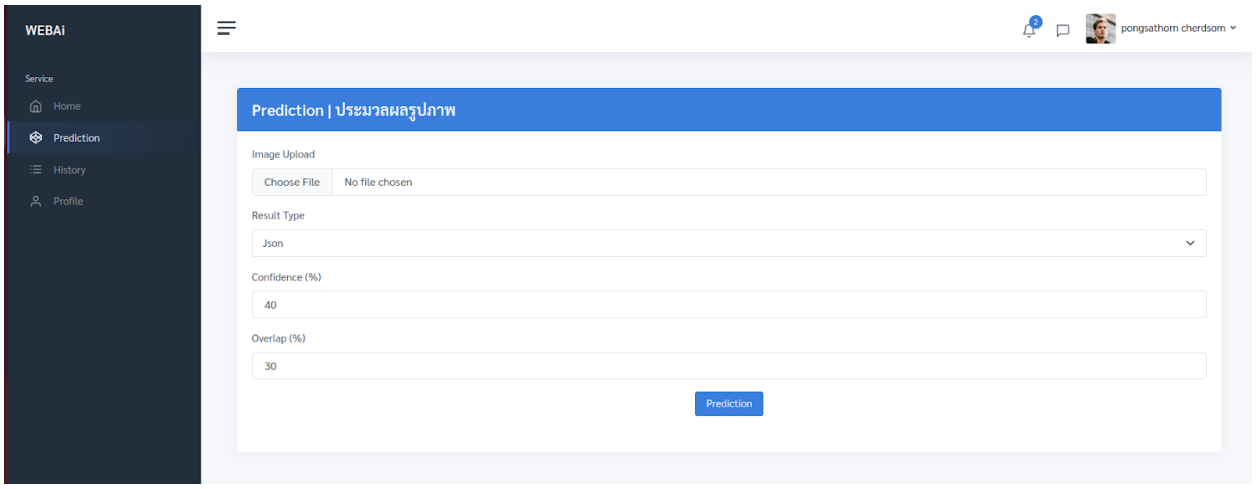
```JavaScript
<img id="output" class="img-fluid" /> <== แสดงผล

<== Js ประมวลผลเพื่อนำไปแสดงผล ==>
<script>
    var showimage = function(event) {
        var image = document.getElementById('output');
        image.src = URL.createObjectURL(event.target.files[0]);
    };
</script>
```

- รูปแบบผลลัพธ์การพยากรณ์ (prediction) มี 2 รูปแบบคือ 1 แบบ josn จะได้ผลในรูปแบบของข้อมูล json ที่เหมาะกับการนำมาจัดการข้อมูลต่อภายในระบบ (สามารถปรับแต่งได้มากกว่ารูปแบบที่ 2) และ 2 แบบ image ที่จะได้ผลลัพธ์แบบรูปภาพสำเร็จจาก api ทันที

ทดสอบ Runserver

python3 manage.py runserver
http://127.0.0.1:8000/backend/prediction/





26. สร้างฟังก์ชั่นเพื่อส่งรูปภาพไปประมวลผล ผ่านทาง API

เรียกใช้งาน database ที่อยู่ใน models โดยเรียกใช้งาน
from .models import *

ติดตั้ง library สำหรับการเรียกใช้งาน roboflow
pip install roboflow

```
Using cached roboflow-1.1.12-py3-none-any.whl (68 kB)
Using cached certifi-2023.7.22-py3-none-any.whl (158 kB)
Using cached opencv_python_headless-4.8.0.74-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (49.1 MB)
Using cached kiwisolver-1.4.5-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.6 MB)
Using cached PyYAML-6.0.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (705 kB)
Using cached matplotlib-3.8.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
Using cached supervision-0.17.1-py3-none-any.whl (77 kB)
Using cached contourpy-1.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (310 kB)
Using cached fonttools-4.46.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.6 MB)
Using cached scipy-1.11.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (36.4 MB)
Installing collected packages: scipy, PyYAML, python-magic, python-dotenv, python-dateutil, pyparsing, opencv-python-headless, ki
wisolver, idna, fonttools, cycler, contourpy, chardet, certifi, matplotlib, supervision, requests-toolbelt, roboflow
  Attempting uninstall: idna
    Found existing installation: idna 3.6
    Uninstalling idna-3.6:
      Successfully uninstalled idna-3.6
  Attempting uninstall: certifi
    Found existing installation: certifi 2023.11.17
    Uninstalling certifi-2023.11.17:
      Successfully uninstalled certifi-2023.11.17
Successfully installed PyYAML-6.0.1 certifi-2023.7.22 chardet-4.0.0 contourpy-1.2.0 cycler-0.10.0 fonttools-4.46.0 idna-2.10 kiwi
solver-1.4.5 matplotlib-3.8.2 opencv-python-headless-4.8.0.74 pyparsing-2.4.7 python-dateutil-2.8.2 python-dotenv-1.0.0 python-ma
gic-0.4.27 requests-toolbelt-1.0.0 roboflow-1.1.12 scipy-1.11.4 supervision-0.17.1
```

เรียกใช้งาน library roboflow โดยที่ทำการเพิ่ม

from roboflow import Roboflow

ไว้ด้านบนของไฟล์ views.py

เรียกใช้งาน การเก็บไฟล์เข้าระบบ FileSystemStorage โดยทำการเพิ่ม

from django.core.files.storage import FileSystemStorage

เรียกใช้งาน settings โดยทำการเพิ่ม

from django.conf import settings

เรียกใช้งาน os โดยทำการเพิ่ม

import os

เรียกใช้งาน uuid4 โดยทำการเพิ่ม

import uuid

สรุปการ import ด้านบน

```python
from .models import *
from roboflow import Roboflow
from django.core.files.storage import FileSystemStorage
from django.conf import settings
import os
import uuid
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่นชื่อ predictionprocessai สำหรับการส่งข้อมูลไปประมวลผลที่ roboflow ผ่านทาง api

เงื่อนไขการทำงาน

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- เช็ค method ที่ส่งมาจากหน้า from predictionprocessai เป็น POST หรือไม่ หากไม่ใช่ให้ redirect กลับไปที่หน้า predictionpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
- เช็คตัวแปรที่รับค่าจาก from ไม่ให้มีค่าใดเป็นค่าว่าง โดยการใช้คำสั่ง is not None และใช้เงื่อไขแบบ and โดยทุกๆตัวแปรต้องไม่เป็นค่าว่าง หากเป็นค่าว่างให้ redirect กลับไปที่หน้า predictionpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง”
- เช็คขนาดของไฟล์ และนามสกุลของไฟล์ หากไม่เข้ากับเงื่อนไข ให้ redirect กลับไปที่หน้า predictionpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “ขนาดไฟล์มีขนาดใหญ่เกินที่กำหนด และหรือ ชนิดของไฟล์ไม่ได้รับอนุญาต โปรดลองใหม่อีกครั้ง”
- เช็คการอัพโหลดรูปภาพที่สมบูรณ์ หากไม่เข้ากับเงื่อนไข ให้ redirect กลับไปที่หน้า predictionpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “ไม่สามารถอัพโหลดรูปภาพได้ โปรดลองใหม่อีกครั้ง”
- ตรวจสอบการบันทึกข้อมูล การประมวลผล หากไม่เข้ากับเงื่อนไข ให้ redirect กลับไปที่หน้า predictionpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “ไม่สามารบันทึกข้อมูลได้ โปรดลองใหม่อีกครั้ง”
- ตรวจสอบการประมวลผลภาพ หากสามารถประมวลผลได้เสร็จสมบูรณ์ ให้ redirect กลับไปที่หน้า **สำหรับแสดงผลข้อมูลการประมวลผล** พร้อมกับระบุข้อข้อความ alert message รูปแบบ success “ ประมวลผลรูปภาพเสร็จสมบูรณ์” หากไม่เข้ากับเงื่อนไข ให้ redirect กลับไปที่หน้า predictionpage พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “ไม่สามารถประมวลผลได้ โปรดลองใหม่อีกครั้ง”

การทำงานของฟังก์ชั่น

- กำหนดตัวแปร session ของ menuactive ให้เป็น prediction (ที่จะนำไปเช็คในไฟล์ menu.html เพื่อให้เมนูนั้นๆ active)
- สร้าง path เรียกไฟล์ โดยใช้ settings.BASE_DIR จากการ import settings ด้านบน และมีการกำหนด path เพิ่ม 'static/fileupload'
- อัพโหลดไฟล์รูปภาพเข้าระบบ
- บันทึกข้อมูลในฐานข้อมูล
- ส่งไฟล์ไปประมวลผล
- เมื่อประมวลผลเสร็จสมบูรณ์ redirect ไปที่ **หน้าแสดงผลการประมวลผล**

```python
def predictionprocessai(request):
    if request.user.is_authenticated:
        request.session['menuactive'] = "prediction"

        fileimage = request.FILES['imagefile']
        resulttype = request.POST.get('resulttype',None)
        confidence = request.POST.get('confidence',None)
        overlap = request.POST.get('overlap',None)

        if request.method == "POST":
            if fileimage is not None and resulttype is not None and confidence is not None and overlap is not None:
                typefile = ['.png','.gif','.jpg','.jpeg']
                name, ext = os.path.splitext(fileimage.name)
                sizelimit = 10 * 1024 * 1024
                pathfileupload = settings.BASE_DIR / 'static/fileupload'

                if ext in typefile and fileimage.size < sizelimit:
                    uuidgenprocess= uuid.uuid4()

                    fs = FileSystemStorage(location="static/fileupload/")
                    renamefile = str(uuidgenprocess)+ext
                    filename = fs.save(renamefile, fileimage)
                    if filename:
                        addimage = Predictprocess()
                        addimage.refuser = User.objects.get(id=request.user.id)
                        addimage.imagefile = renamefile
                        addimage.typepredict = resulttype
                        addimage.confidence = confidence
                        addimage.overlap = overlap
                        addimage.uuidpredict = uuidgenprocess
                        addimage.save()

                        if addimage:
                            rf = Roboflow(api_key="7[redacted]")
                            project = rf.workspace().project("g[redacted]")
                            model = project.version(1).model

                            pathfilelocal = str(pathfileupload)+'/'+renamefile
                            if resulttype =="Json":
                                predict_result = model.predict(pathfilelocal, confidence=confidence, overlap=overlap).json()
                                addimage.resultpredict = predict_result
                                addimage.save()

                            elif resulttype =="Image":
                                predict_result = model.predict(pathfilelocal, confidence=confidence, overlap=overlap).save(f"
                                {pathfileupload}/predict_{renamefile}")
                                addimage.resultpredict = f"predict_{renamefile}"
                                addimage.save()

                            else:
                                predict_result = {}
                                addimage = False

                            if addimage:
                                messages.success(request,'ประมวลผลรูปภาพเสร็จสมบูรณ์')
                                return redirect(predictionpage)
                            else:
                                messages.error(request,'ไม่สามารถประมวลผลได้ โปรดลองใหม่อีกครั้ง')
                                return redirect(predictionpage)
                        else:
                            messages.error(request,'ไม่สามารถบันทึกข้อมูลได้ โปรดลองใหม่อีกครั้ง')
                            return redirect(predictionpage)
                    else:
                        messages.error(request,'ไม่สามารถอัพโหลดรูปภาพได้ โปรดลองใหม่อีกครั้ง')
                        return redirect(predictionpage)
                else:
                    messages.error(request,'ขนาดไฟล์มีขนาดใหญ่เกินที่กำหนด และหรือ ชนิดของไฟล์ไม่ได้รับอนุญาต โปรดลองใหม่อีกครั้ง')
                    return redirect(predictionpage)
            else:
                messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
                return redirect(predictionpage)
        else:
            messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
            return redirect(predictionpage)
    else:
        messages.error(request,'ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน')
        return redirect(homepage)
```

Python

```
def predictionprocessai(request):
    if request.user.is_authenticated:
        request.session['menuactive'] = "prediction"

        fileimage = request.FILES['imagefile']
        resulttype = request.POST.get('resulttype',None)
        confidence = request.POST.get('confidence',None)
        overlap = request.POST.get('overlap',None)

        if request.method == "POST":
            if fileimage is not None and resulttype is not None and confidence
is not None and overlap is not None:
                typefile = ['.png','.gif','.jpg','.jpeg']
                name, ext = os.path.splitext(fileimage.name)
                sizelimit = 10 * 1024 * 1024
                pathfileupload = settings.BASE_DIR / 'static/fileupload'

                if ext in typefile and fileimage.size < sizelimit:
                    uuidgenprocess= uuid.uuid4()

                    fs = FileSystemStorage(location="static/fileupload/")
                    renamefile = str(uuidgenprocess)+ext
                    filename = fs.save(renamefile, fileimage)
                    if filename:
                        addimage = Predictprocess()
                        addimage.refuser = User.objects.get(id=request.user.id)
                        addimage.imagefile = renamefile
                        addimage.typepredict = resulttype
                        addimage.confidence = confidence
                        addimage.overlap = overlap
                        addimage.uuidpredict = uuidgenprocess
                        addimage.save()

                        if addimage:
                            rf = Roboflow(api_key="7L8iJqFfvyqjvYbheblF")
                            project =
rf.workspace().project("grape-detection-hfjsk")
                            model = project.version(1).model

                            pathfilelocal = str(pathfileupload)+'/'+renamefile
                            if resulttype =="Json":
                                predict_result = model.predict(pathfilelocal,
confidence=confidence, overlap=overlap).json()
                                addimage.resultpredict = predict_result
```

```
                                addimage.save()

                            elif resulttype =="Image":
                                predict_result = model.predict(pathfilelocal,
confidence=confidence,
overlap=overlap).save(f"{pathfileupload}/predict_{renamefile}")
                                addimage.resultpredict =
f"predict_{renamefile}"
                                addimage.save()

                            else:
                                predict_result = {}
                                addimage = False

                            if addimage:
                                messages.success(request,'ประมวลผลรูปภาพเสร็จสมบูรณ์
')
                                return redirect(predictionpage)
                            else:
                                messages.error(request,'ไม่สามารถประมวลผลได้ โปรด
ลองใหม่อีกครั้ง')
                                return redirect(predictionpage)
                        else:
                            messages.error(request,'ไม่สามารบันทึกข้อมูลได้ โปรดลองใหม่
อีกครั้ง')
                            return redirect(predictionpage)
                    else:
                        messages.error(request,'ไม่สามารถอัพโหลดรูปภาพได้ โปรดลองใหม่
อีกครั้ง')
                        return redirect(predictionpage)
                else:
                    messages.error(request,'ขนาดไฟล์มีขนาดใหญ่เกินที่กำหนด และหรือ ชนิด
ของไฟล์ไม่ได้รับอนุญาต โปรดลองใหม่อีกครั้ง')
                    return redirect(predictionpage)
            else:
                messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
                return redirect(predictionpage)
        else:
            messages.error(request,'เกิดข้อผิดพลาด โปรดลองใหม่อีกครั้ง')
            return redirect(predictionpage)
    else:
        messages.error(request,'ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิก
ก่อน')
        return redirect(homepage)
```

27. การสร้างหน้าสำหรับแสดงผลการ Prediction (รายการที่เลือก)

File : urls.py (อยู่ใน webai)

ซึ่งการกำหนดเส้นทาง ได้กำหนดให้อยู่ภายใต้ /backend เพื่อเป็นการจัดกลุ่มของการทำงานและฟังก์ชั่น โดยกำหนดให้เป็น backend/result/item/<str:uuidprocesss>/ ซึ่งจะอยู่ภายใน /result อีกหนึ่งชั้น (โดยที่ backend/result/ จะนำไปใช้สำหรับหน้าแสดงผลการประมวลผลทั้งหมดของผู้ใช้งาน)
และ <str:uuidprocesss> เป็นการส่งตัวแปรชื่อ uuidprocesss ชนิดตัวอักษร นำไปใช้ในฟังก์ชั่น ตัวอย่างใน **หัวข้อที่ 7 กรณีส่งค่าข้อมูลผ่านทาง URL ไปแสดงผล (การกำหนด path แบบ parameter)**

```python
path('backend/result/item/<str:uuidprocesss>/', views.predictionresultinfopage, name='predictionresultinfopage'),
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่นชื่อ predictionresultinfopage เพื่อนำมาแสดงผลของการประมวลผลที่ได้จาก roboflow ซึ่งในขั้นตอนก่อนหน้ามีรูปแบบผลลัพธ์การประมวลผล 2 แบบคือ json และ image

เรียกใช้งาน import ast เพื่อแปลงข้อมูลตัวอักษรเป็นแบบ dict() ไว้ด้านบนของไฟล์ views.py ในลำดับถัดลงมาจาก import หรือ form บรรทัดล่าสุด

เรียกใช้งาน from PIL import Image
เรียกใช้งาน import base64
เรียกใช้งาน from io import BytesIO

```python
import ast
from PIL import Image
import base64
from io import BytesIO
```

เงื่อนไขการตรวจสอบ

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”

- ตรวจสอบข้อมูล จากฐานข้อมูลตาราง Predictprocess หากพบข้อมูลให้ดำเนินการต่อไป หากไม่พบข้อมูลให้ redirect กลับไปที่หน้า predictionpage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่พบข้อมูล โปรดลองใหม่อีกครั้ง”

การทำงานของฟังก์ชั่น

- กำหนดตัวแปร session ของ menuactive ให้เป็น prediction (ที่จะนำไปเช็คในไฟล์ menu.html เพื่อให้เมนูนั้นๆ active)
- ค้นหา ข้อมูลตัวแปร uuidprocesss ที่ส่งมา จากฐานข้อมูลตาราง Predictprocess ที่เก็บผลการพยากรณ์ข้อมูล ว่ามีข้อมูลนี้ในฐานข้อมูลหรือไม่
  โดยการ query ข้อมูลแบบ get และมีการใช้งาน try: except เพื่อดักค่า error กรณีที่มีข้อมูลจากการ get มากกว่า 1 ข้อมูล
- สร้างตัวแปรเพื่อเป็นค่าตั้งต้นได้แก่
  - countitem = 0 จำนวนวัตถุที่พยากรณ์ได้
  - listitems = [] รายการวัตถุทั้งหมดที่พยากรณ์ได้
  - imagepredict = None รูปภาพผลการพยากรณ์ ที่เขียนเส้นกรอบวัตถุ ชื่อคลาส และค่าความแม่นยำ
- เช็คเงื่อนไขเงื่อนไขเดียวคือ รูปแบบของการแสดงผล (คอลัมน์ typepredict) แบบ Json หรือไม่ หากใช่จะมีการดำเนินการอย่างอื่นต่อ ซึ่งแตกต่างจากรูปแบบ Image ที่นำรูปไปแสดงผลได้เพียงอย่างเดียว
  การดำเนินการภายในเงื่อนไข typepredict == "Json" จะมีการดำเนินการ
  - แปลงข้อมูลจากข้อความ ที่เก็บอยู่ในคอลัมน์ resultpredict ให้เป็นตัวแปรแบบ dict() โดยใช้คำสั่ง ast.literal_eval(ตัวแปรข้อมูล)
  - นับจำนวนวัตถุที่สามารถพยากรณ์ได้ จากข้อมูลที่ชื่อว่า “predictions” ที่ได้จากการพยากรณ์ของ roboflows โดยใช้คำสั่ง len()
  - สร้างตัวแปรเป็น path ของระบบ สำหรับเรียกไฟล์ที่อยู่ภายในเครื่อง
  - อ่านไฟล์รูปภาพที่อยู่ภายในเครื่อง โดยใช้คำสั่ง Image.open()
  - ใช้คำสั่ง copy() รูปภาพเพื่อนำไปใช้ในส่วนของการเขียนเส้น (ตีกรอบวัตถุ) และผลที่ได้จากการพยากรณ์
  - ทำการลูปข้อมูลที่อยู่ใน “predictions” เพื่อนำมาเขียนเส้นของวัตถุ โดยใช้คำสั่ง for — in —
  - ภายในลูป จะทำการคำนวนตำแหน่งของการเขียนข้อความและเส้นกรอบ
  - กำหนดข้อความที่จะแสดง และขนาดของตัวอักษร
  - สร้างตัวแปรเพื่อเริ่มต้นการเขียนข้อความและเส้นกรอบวัตถุ
  - เขียนเส้นกรอบวัตถุสีน้ำเงิน และเขียนข้อความสีขาว พร้อมด้วยพื้นหลังสีน้ำเงิน
  - นำตำแหน่งที่พยากรณ์ได้ มาตัดรูปภาพเฉพาะส่วนที่มีการพยากรณ์ถูกต้อง นำมาสร้างเป็นรูปภาพรูปแบบ base64
  - เก็บข้อมูลการลูปลงในตัวแปร datapre ชนิด dict() เพื่อเก็บข้อมูลและนำไปแสดงผล เพื่อให้เกิดความสะดวกในการเรียกใช้งาน ซึ่งจะเก็บข้อมูลได้แก่ imagecrop_base64 คือส่วนของรูปภาพที่มีการตัดเฉพาะส่วนที่พยากรณ์ถูกต้อง,

class คือชื่อของวัตถุที่พยากรณ์ , confidence คือ ค่าความแม่นยำถูกต้องจากการพยากรณ์ (ของแต่ละลูป)
    - และนำข้อมูลที่ได้จากตัวแปร datapre มาเก็บเข้า ตัวแปร listitems ชนิด list() ที่จำเก็บข้อมูลทั้งหมดภายในลูป ซึ่งจะนำปแสดงผลในส่วนของ รายการการพยากรณ์
    - สร้างรูปภาพที่นำเส้นกรอบของวัตถุทั้งหมดที่ได้จากการพยากรณ์นำมาเขียนไว้ในภาพ (ทุกวัตถุ) ในรูปแบบ base64 หนึ่งภาพ ในตัวแปร imagepredict
- ส่งข้อมูลจากการ query ไปแสดงในหน้า resultinfo.html โดยแนบไปในตัวแปร context ซึ่งมีข้อมูล context = { 'datapredict': getinfopro , 'countitem':countitem, 'listitems' : listitems , 'imagepredict' : imagepredict } และข้อมูลอื่นๆทั้งหมดในฟังก์ชั่นแนบไปด้วย (ในตัวอย่างนี้ได้ใช้ชื่อตัวแปรซ้ายมือ และตัวแปรขวามือไม่ตรงกันเป็นตัวอย่างในการนำไปใช้งาน)

Template : resultinfo.html

- สร้างไฟล์ resultinfo.html ซึ่งมีโครงสร้างเหมือนกับ สามารถคัดลอก backendpage.html มาใช้งานได้
- เรียกใช้ {% load static %} ไว้บรรทัดล่างของ {% extends 'base.html' %} ของไฟล์ resultinfo.html เพื่อให้สามารถเรียกใช้ไฟล์รูปภาพจากระบบได้
- รูปแบบการแสดงผล โดยใช้ if elif else ในการตรวจสอบรูปแบบการแสดงผล
    - 1 แสดงรูปภาพจากการพยากรณ์
    - 2 แสดงข้อมูลจากข้อมูล json
    - 3 ไม่เข้าเงื่อนไขใดๆ แสดงข้อความผิดพลาด

```Python
{% if datapredict.typepredict == "Image" %}
      << แสดงผลสำหรับรูปแบบรูปภาพ
{% elif datapredict.typepredict == "Json" %}
      << แสดงผลสำหรับรูปแบบรูปภาพ
{% else %}
      << แสดงผลสำหรับไม่เข้าเงื่อนไขทั้งสองด้านบน (แสดงข้อความผิดพลาด)
{% endif %}
```

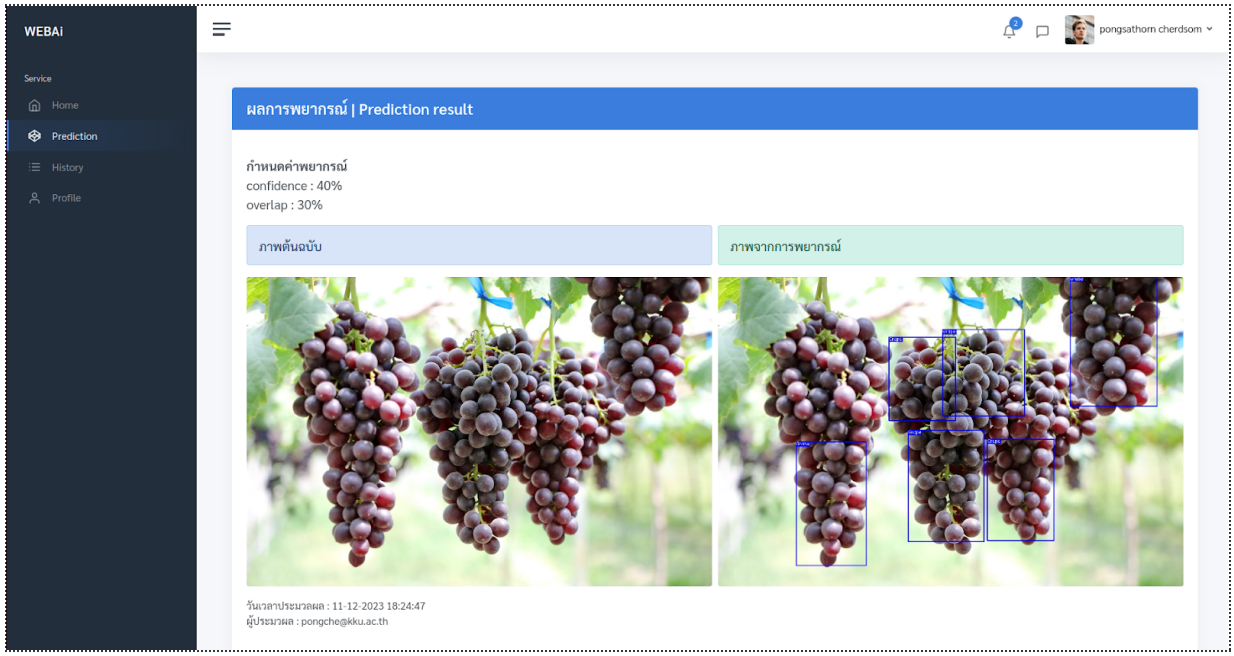
- ภายในเงื่อนไข Image
    - ทำการแสดงผลรูปภาพที่นำมาใช้ในการพยากรณ์ (ต้นฉบับ) และภาพที่ได้จากการพยากรณ์ วันที่และเวลาทำรายการ ผู้ทำรายกากร โดยแบ่งหน้าจอให้เป็น 2 คอลัมน์ ซึ่งในการแสดงผลแบบ image นี้จะไม่สามารถแสดงรายละเอียดอื่นๆได้ นอกจากรูปภาพจากการพยากรณ์
    - ในส่วนของวันที่และเวลา สามารถใช้ filter Date โดยเขียน |date:'d-m-Y H:i:s' คือรูปแบบของวันที่ที่ต้องการให้แสดงผลดังตัวอย่างจะแสดง 11-12-2023 18:24:47

ศึกษาเพิ่มเติมได้ที่
https://docs.djangoproject.com/en/5.0/ref/templates/builtins/#date

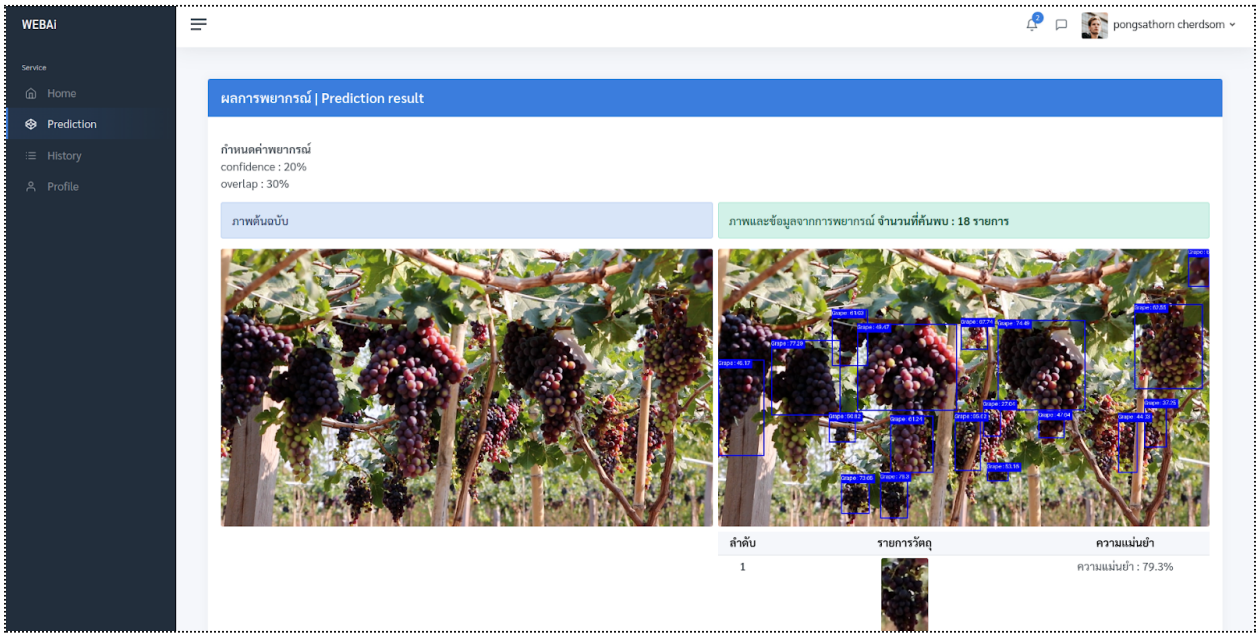
- ในส่วนของการแสดงข้อมูลผู้ใช้งาน สามารถเชื่อมต่อตารางได้ดังภาพ

Python

```
<div class="row g-2">
        <div class="col-md-6">
            <div class="alert alert-primary h4" role="alert">
                ภาพต้นฉบับ
              </div>
            <img src="{% static 'fileupload' %}/{{ datapredict.imagefile }}"
class="img-fluid rounded" />
        </div>
        <div class="col-md-6">
            <div class="alert alert-success h4" role="alert">
                ภาพจากการพยากรณ์
            </div>
            <img src="{% static 'fileupload' %}/{{ datapredict.resultpredict
}}" class="img-fluid rounded" />
        </div>
</div>

<p class="mt-3 fs-5">
        วันเวลาประมวลผล : {{ datapredict.datetimepre|date:'d-m-Y H:i:s' }} <br>
       ผู้ประมวลผล : {{ datapredict.refuser.username }}
</p>
```

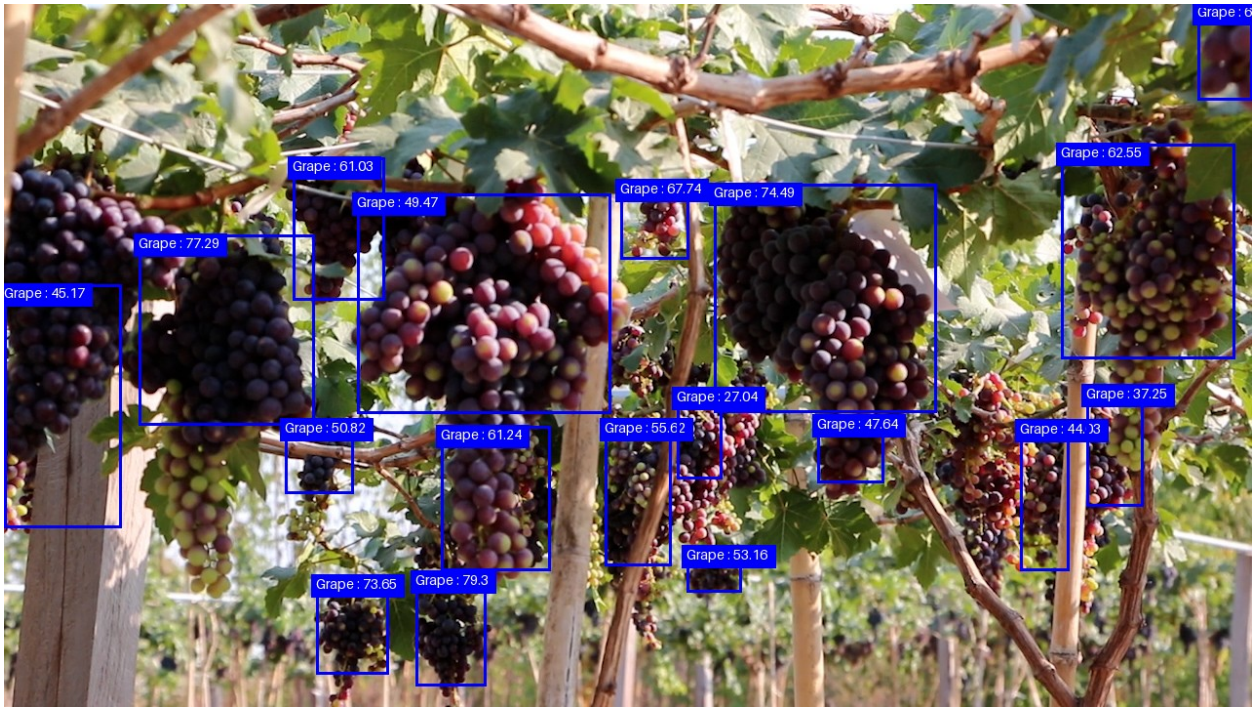


- ภายในเงื่อนไข Json
    - จะแสดงภาพต้นฉบับ จำนวนองุ่นที่สามารถตรวจพบทั้งหมด รายการองุ่นที่ตรวจสอบภาพและความแม่นยำที่ได้ แสดงภาพผลลัพธ์จาก json ที่นำมาประมวลผลเอง (วาดเส้น ระบุชื่อคลาส และค่าความแม่นยำ) รวมถึงวันเวลาประมวลผล และชื่อผู้ทำรายการ



รายการวัตถุที่ได้จากการตรวจจับ พร้อมทั้งแสดงค่าความแม่นยำ

ลำดับ	รายการวัตถุ	ความแม่นยำ
1		ความแม่นยำ : 79.3%
2		ความแม่นยำ : 77.29%
3		ความแม่นยำ : 74.49%
4		ความแม่นยำ : 73.65%
5		ความแม่นยำ : 67.74%
6		ความแม่นยำ : 62.55%

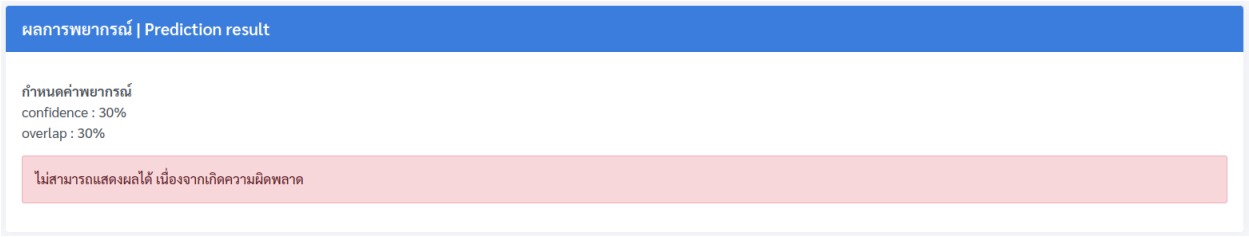


Python

```
<div class="row g-2">
        <div class="col-md-6">
            <div class="alert alert-primary h4" role="alert">
                ภาพต้นฉบับ
              </div>
            <img src="{% static 'fileupload' %}/{{ datapredict.imagefile }}"
class="img-fluid rounded" />
        </div>
        <div class="col-md-6">
            <div class="alert alert-success h4" role="alert">
                ภาพและข้อมูลจากการพยากรณ์ <span class="fw-bold"> จำนวนที่ค้นพบ : {{
countitem }} รายการ </span>
            </div>

            <img src="{{ imagepredict }}" class="img-fluid rounded" /><br>

            <table class="table table-sm text-center mt-2">
                <thead class="table-light">
                  <tr>
                    <th scope="col-2">ลำดับ</th>
                    <th scope="col-8">รายการวัตถุ</th>
                    <th scope="col-2">ความแม่นยำ</th>
                  </tr>
                </thead>
                <tbody>
                    {% for item in listitems %}
                        <tr>
                            <th scope="row">{{ forloop.counter }}</th>
                            <td>
                                <img src="{{ item.imagecrop_base64 }}"
class="img-fluid rounded" /><br>
                            </td>
                            <td class="align-top">
                                <span>ความแม่นยำ : {{ item.confidence }}%</span>
                            </td>
                      </tr>
                    {% endfor %}
                </tbody>
            </table>

        </div>
    </div>

<p class="mt-3">
```

```
        วันเวลาประมวลผล : {{ datapredict.datetimepre|date:'d-m-Y H:i:s' }} <br>
        ผู้ประมวลผล : {{ datapredict.refuser.username }}
</p>
```

- ไม่เข้าเงื่อนไขทั้ง image และ Json เพื่อป้องกันความผิดพลาด โดยให้แสดงข้อความ “ไม่สามารถแสดงผลได้ เนื่องจากเกิดความผิดพลาด”



Python
```
<div class="alert alert-danger" role="alert">
            ไม่สามารถแสดงผลได้ เนื่องจากเกิดความผิดพลาด
 </div>
```

28. การสร้างหน้าสำหรับแสดงรายการ Prediction

สร้างหน้าแสดงรายการทั้งหมดที่ส่งข้อมูลไป Prediction ซึ่งจะอยู่ในเมนู History แสดงเฉพาะข้อมูลของผู้ใช้งานที่ทำรายการเท่านั้น
เรียกใช้งาน Paginator, EmptyPage, PageNotAnInteger เพื่อทำ pagination (การแบ่งหน้าของข้อมูล) ใส่ในไฟล์ views.py ที่ทำงาน

Python
```
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทาง กำหนดให้เป็น /result ภายใต้ /backend และเรียกใช้งานฟังก์ชั่น resultpredictionpage ในหัวข้อก่อนหน้าจะเป็น backend/result/item/ สำหรับแสดงผลของแต่ละรายการที่เลือก ส่วนหัวข้อนี้จะเป็นส่วนที่ใช้แสดงรายการทั้งหมด แล้วค่อยคลิกเลือกเพื่อแสดงผลรายละเอียดที่ได้จากการพยากรณ์

```python
path('backend/result/', views.resultpredictionpage,
name='resultpredictionpage'),
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่น resultpredictionpage สำหรับแสดงรายการพยากรณ์ทั้งหมดของผู้ใช้งาน
โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- กำหนดตัวแปร session ของ menuactive ให้เป็น history (ที่จะนำไปเช็คในไฟล์ menu.html เพื่อให้เมนูนั้นๆ active)
- query รายการข้อมูลจากตาราง Predictprocess กำหนดเงื่อนไขให้เลือกเฉพาะของผู้ใช้งานนั้น refuser_id=request.user.id โดยที่
    - refuser คือชื่อคอลัมน์ในตาราง Predictprocess
    - _id คือการเชื่อโยงไปที่คอลัมน์ id ของตาราง User จะสารถเรียกใช้งานได้ทันที เนื่องจากใน models ได้กำหนดให้เป็น models.ForeignKey(User) แล้ว
    - request.user.id คือ ข้อมูลคอลัมน์ id ของผู้ใช้งานนั้นที่ลงชื่อเข้าใช้งาน

  และจัดเรียงข้อมูล order_by('-datetimepre') - หมายถึงเรียงจากมากไปน้อย ส่วน datetimepre คือคอลัมน์ที่ใช้ในการจัดเรียง (datetimepre เก็บข้อมูลวันที่และเวลา)
- นำข้อมูลจากการ query จากด้านบนมาทำให้ pagination โดยได้กำหนด
    - ตัวแปร showperpage กำหนดจำนวนข้อมูลที่แสดงในหนึ่งหน้า
    - ตัวแปร page รับข้อมูลแบบ get จากตัวแปรที่ชื่อ page และหากไม่มีค่าให้กำหนดตั้งต้นเป็น 1
    - ตัวแปร paginator เรียกใช้งานฟังก์ชั่น Paginator() ที่ django เตรียมไว้ให้และส่งค่าข้อมูลที่ได้จากการ query และ ตัวแปร showperpage เข้าไปสองตัว Paginator(listpredict, showperpage)
    - ใช้ try except PageNotAnInteger และ EmptyPage ในการตรวจสอบความผิดพลาด
- ส่งค่า {'listpredict':listpredict} ไปแสดงผลที่ไฟล์ history.html

```python
def resultpredictionpage(request):
    if request.user.is_authenticated:
        request.session['menuactive'] = "history"
```

```
        listpredict =
Predictprocess.objects.filter(refuser__id=request.user.id).order_by('-datetimep
re')

        showperpage = 5
        page = request.GET.get('page', 1)
        paginator = Paginator(listpredict, showperpage)
        try:
            listpredict = paginator.page(page)
        except PageNotAnInteger:
            listpredict = paginator.page(1)
        except EmptyPage:
            listpredict = paginator.page(paginator.num_pages)

        context = {'listpredict':listpredict}
        return render(request,'history.html',context)
    else:
        messages.error(request,'ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิก
ก่อน')
```

อัพเดต URL ของเมนู History

ในไฟล์ menu.html ที่อยู่ใน folder inclue

อัพเดตเมนูในส่วนของเมนู history tag a href ให้เป็น {% url 'resultpredictionpage' %}

Python

```
<li class="sidebar-item {% if request.session.menuactive == 'history' %} active
{% endif %}">
                <a class="sidebar-link" href="{% url 'resultpredictionpage'
%}">
                    <i class="align-middle" data-feather="list"></i> <span
class="align-middle">History</span>
                </a>
</li>
```

Template : history.html

สร้างไฟล์ชื่อ history.html โดยใช้โครงสร้างเดียวกับ backendpage.html

- เรียกใช้งาน {% load static %} ใต้บรรทัดการทำ extends
- แสดงข้อมูลโดยใช้การลูป for in ในตาราง ซึ่งจะแสดงข้อมูล

- ลำดับ ใช้เป็นตัวเลขลำดับการแสดงผล โดยใช้ตัวแปร forloop.counter0|add:listpredict.start_index ซึ่ง add คือเพิ่มการบวกตัวเลข listpredict.start_index จำนวนที่เริ่มต้นของหน้านั้นๆ เช่น แสดงจำนวนต่อหน้า 20 ถ้าอยู่หน้าที่ 2 ต้องเริ่มจาก 40 แล้วนำไปบวกกับ forloop.counter0 คือลำดับที่ของข้อมูลในหน้านั้นๆ เริ่มจาก 0 ที่ใช้ในการนับจำนวนให้โดยอัตโนมัติ
- ชื่อรายการ ใช้ชื่อรายการเป็นวันที่ทำรายการ และจะใช้สำหรับคลิกเพื่อไปแสดงในหน้าของ predictionresultinfopage ที่แสดงผลการพยากรณ์แบบละเอียด โดยใช้ tag a กำหนดปลายทางไปที่ {% url 'predictionresultinfopage' item.uuidpredict %} หมายถึงไปที่เส้นทาง 'predictionresultinfopage' ซึ่งฟังก์ชั่นนี้ต้องมีการรับค่าหนึ่งค่า คือ uuidprocesss ไปประมวลลผล รูปแบบของการเรียกใช้งาน url ที่มีการส่งค่าไปด้วยให้กำหนดดังตัวอย่าง
- รูปแบบการพยากรณ์ รูปแบบ image หรือ json
- รูปภาพที่ใช้พยากรณ์ เป็นรูปภาพที่เก็บอยู่ในคอลัมน์ imagefile (รูปภาพต้นฉบับที่ส่งไป) กำหนดขนาดความกว้างให้เป็น 100px width="100px"
- ลบรายการ สำหรับปุ่มลบข้อมูลการพยากรณ์

Python

```
<table class="table table-striped">
    <thead>
        <tr>
        <th scope="col-1" class="text-center">#</th>
        <th scope="col-6">รายการ</th>
        <th scope="col-2">รูปแบบ</th>
        <th scope="col-2">รูปภาพ</th>
    <th scope="col-1">ลบ</th>
        </tr>
    </thead>
    <tbody>
        {% for item in listpredict %}
        <tr>
            <th class="text-center">{{
forloop.counter0|add:listpredict.start_index }}</th>
            <td class="align-top">
                <a href="{% url 'predictionresultinfopage' item.uuidpredict %}"
>รายการ {{ item.datetimepre|date:'d-m-Y H:i:s' }}</a>
            </td>
            <td class="align-top">{{ item.typepredict }}</td>
            <td class="align-top">
                <img src="{% static '/fileupload' %}/{{ item.imagefile }}"
width="100px" class="img-fluid rounded"/>
            </td>
       <td class="align-top">
```

```
      </td>
       </tr>
       {% endfor %}
    </tbody>
    </table>
```

- แสดงผลการทำ pagination

Python

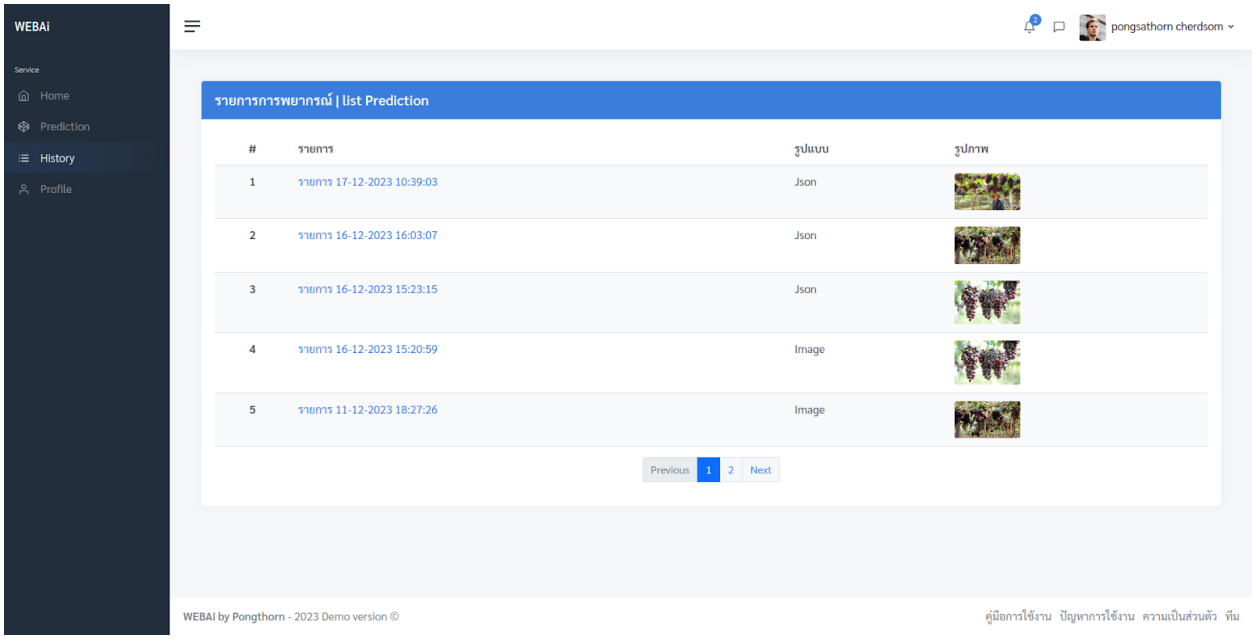
```
<div class="d-flex mt-3">
        <div class="mx-auto">
          <nav >
            <ul class="pagination">
                {% if listpredict.has_previous %}
                <li class="page-item">
                  <a href="?page={{ listpredict.previous_page_number }}"
class="page-link">Previous</a>
                </li>
                <li>
                  <a class="page-link" href="?page={{
listpredict.previous_page_number }}">
                    {{ listpredict.previous_page_number }}
                  </a>
                </li>
                {% else %}
                <li class="page-item disabled">
                  <span class="page-link">Previous</span>
                </li>
                {% endif %}

                <li class="page-item active">
                  <span class="page-link">{{ listpredict.number }}</span>
                </li>

                {% if listpredict.has_next %}
                <li class="page-item">
                  <a class="page-link" href="?page={{
listpredict.next_page_number }}">
                    {{ listpredict.next_page_number }}
                  </a>
                </li>
```

```
                <li class="page-item ">
                  <a class="page-link" href="?page={{
listpredict.next_page_number }}">Next</a>
                </li>
                {% else %}
                <li class="page-item disabled">
                  <span class="page-link">Next</span>
                </li>
                {% endif %}

              </ul>
            </nav>
          </div>
        </div>
```

ทดสอบ Runserver

python3 manage.py runserver

http://127.0.0.1:8000/backend/result/



29. การสร้างฟังก์เพื่อลบผลการพยากรณ์ Prediction

Template : history.html

ทำงานในหน้าของ รายการการพยากรณ์ ฟังก์ชั่น resultpredictionpage ไฟล์ชื่อว่า history.html ซึ่งต่อจากหัวข้อก่อนหน้า จะทำงานในส่วนของ tag <a> ปุ่มในการลบรายการอยู่ในคอลัมน์สุดท้ายของตาราง

โดยที่

- เพิ่ม ลบ โดยที่ delete{{ item.indexpredict }} คือชื่อตัวแปรของ popup (Modal) ของการยืนยันการลบข้อมูล เพื่อเป็นการแสดงข้อมูลต่างๆก่อนการยืนยันการลบ
- เพิ่มส่วนของ popup (Modal) ให้กำหนด id="delete{{ item.indexpredict }}" คือชื่อเดียวกับที่ระบุใน tag <a>
- ส่วนของปุ่ม ยืนยันการลบข้อมูล ยืนยันการลบ จะทำการเรียกใช้งาน url ที่ชื่อว่า resultdeleteitem และส่งค่า uuidpredict ไปด้วย เพื่อใช้ในการประมวลผลในการลบข้อมูล

Python

```
<td class="align-top">
                <a class="btn btn-primary" href="#" data-bs-toggle="modal"
data-bs-target="#delete{{ item.indexpredict }}" role="button">ลบ</a>


                <div class="modal fade" id="delete{{ item.indexpredict }}"
tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
                  <div class="modal-dialog">
                    <div class="modal-content">
                      <div class="modal-header">
                        <h1 class="modal-title fs-3">ยืนยันการลบข้อมูล</h1>
                        <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                      </div>
                      <div class="modal-body text-dark">
                        รายการ {{ item.datetimepre|date:'d-m-Y H:i:s' }} <br>
                        รูปแบบ {{ item.typepredict }} <br><br>
                        <img src="{% static '/fileupload' %}/{{ item.imagefile
}}" class="img-fluid rounded"/>
                        <br><br>
                        <a class="btn btn-primary" href="{% url
'resultdeleteitem' item.uuidpredict %}" role="button">ยืนยันการลบข้อมูล</a>
```

```
                        <button type="button" class="btn btn-secondary" data-bs
-dismiss="modal">ยกเลิกการลบ</button>
                    </div>
                  </div>
                </div>
              </div>
</td>
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทาง กำหนดให้เป็น backend/result/delete/ และกำหนดการรับค่า uuidprocesss โดยกำหนดชนิดของตัวแปรเป็น str คือ string ตัวอักษร และเรียกใช้งานฟังก์ชั่นที่ชื่อว่า resultdeleteitem กำหนดชื่อเรียกของเส้นทางนี้คือ resultdeleteitem (ซึ่งจะเป็นชื่อเรียกใน tag <a> ที่เรียกใช้งานผ่านหน้า html)

Python

```
path('backend/result/delete/<str:uuidprocesss>/', views.resultdeleteitem,
name='resultdeleteitem'),
```

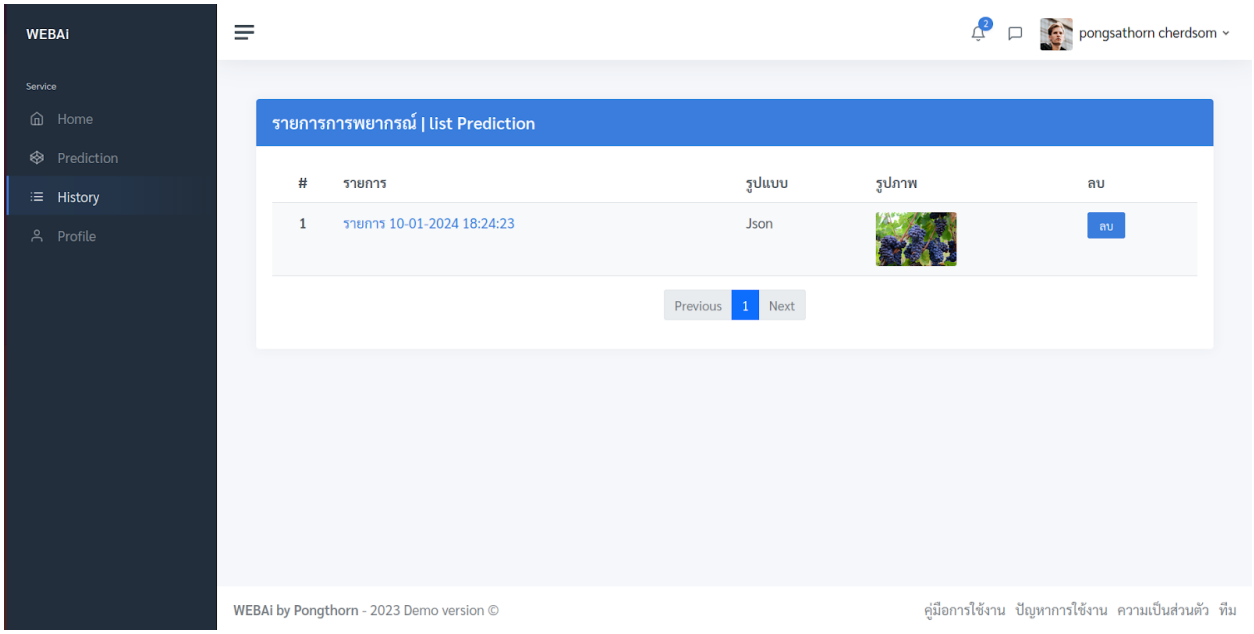
File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่นที่ชื่อว่า resultdeleteitem และรับค่าตัวแปร uuidprocesss ชื่อเดียวกับที่สร้างใน url จะได้เป็น def resultdeleteitem(request,uuidprocesss) สำหรับใช้ในการลบข้อมูลจากฐานข้อมูลและลบไฟล์ที่อัพโหลดขึ้นไปบนระบบ
โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ทำการ query เช็คข้อมูลจากตาราง Predictprocess โดยกำหนดเงื่อนไขให้ค้นหา uuidprocesss และ เป็นของผู้เจ้าของข้อมูล Predictprocess.objects.get(refuser_id=request.user.id,uuidpredict=uuidprocesss) เก็บไว้ในตัวแปร deleteitem
- เช็คเงื่อนไขการจากตัวแปร deleteitem ว่ามีข้อมูลจากการ query หรือไม่ หากมีให้ดำเนินการถัดไป หากไม่มีส่งข้อมูลกลับไปที่ฟังก์ชั่น resultpredictionpage พร้อมกับข้อความที่ “ไม่พบข้อมูล โปรดลองใหม่อีกครั้ง” และ alert message รูปแบบ error
- ทำการเช็ค if deleteitem.typepredict == "Image": ในกรณีที่ส่งข้อมูลไปประมวลผลในรูปแบบ Image ระบบที่พัฒนาจะเก็บผลการพยากรณ์ในรูปแบบของไฟล์รูปภาพจาก roboflow จึงต้องเช็คเพื่อลบรูปภาพจากการพยากรณ์ด้วย และใช้คำสั่งในการลบไฟล์คือ os.remove(f"{pathfileupload}/{deleteitem.resultpredict}")
- ลบไฟล์ต้นฉบับที่ส่งไปประมวลผลที่ roboflow โดยใช้คำสั่ง os.remove(f"{pathfileupload}/{deleteitem.imagefile}")
- จากนั้นใช้คำสั่ง deleteitem.delete() เพื่อทำการลบข้อมูลออกจากฐานข้อมูล และ redirect กลับไปที่ url name resultpredictionpage พร้อมทั้งแสดงข้อความ “ลบข้อมูลการประมวลผลรูปภาพเสร็จสมบูรณ์” และ alert message รูปแบบ success

ทดสอบ Runserver

python3 manage.py runserver
http://127.0.0.1:8000/backend/result/



30. การสร้างหน้ารายการผู้ใช้งาน (สำหรับผู้ดูแลระบบ)

Template : menu.html

ทำการเพิ่มเมนูขึ้นมาใหม่ สำหรับผู้ดูแลระบบชื่อเมนู **User** ไปที่ไฟล์ include -> menu.html

สำหรับเมนูของผู้ดูแลระบบ จะเช็คว่าผู้ใช้งานนั้นเป็นผู้ดูแลระบบหรือไม่ โดยใช้คำสั่ง {% if request.user.is_admin %} สำหรับการเช็ค โดยที่ is_admin คือคอลัมน์ที่ใช้เก็บข้อมูลที่ระบุว่า user นั้นๆ เป็นผู้ดูแลระบบหรือไม่ และเมนูของผู้ดูแลระบบทั้งหมดจะอยู่ภายในเงื่อนไขนี้ทั้งหมด

โดยเรียกใช้งาน url name backendlistuser และกำหนด request.session.menuactive เป็น userlist

Python

```
{% if request.user.is_staff %}
            <li class="sidebar-header">
                Admin
            </li>
            <li class="sidebar-item {% if request.session.menuactive ==
'userlist' %} active {% endif %}">
                <a class="sidebar-link" href="{% url 'backendlistuser' %}">
                    <i class="align-middle" data-feather="users"></i> <span
class="align-middle">User</span>
                </a>
            </li>
{% endif %}
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทาง กำหนดให้เป็น backend/listuser/ เรียกใช้งานฟังก์ชั่น backendlistuser และกำหนดชื่อเส้นทางนี้เป็น backendlistuser เพื่อให้สอดคล้องกับหน้า html ที่ได้กำหนดไว้ในขั้นตอนก่อนหน้านี้

```python
path('backend/listuser/', views.backendlistuser, name='backendlistuser'),
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่นชื่อว่า backendlistuser ชื่อเดียวกันกับที่ระบุใน url views.backendlistuser ในขั้นตอนก่อนหน้า ซึ่งในฟังก์ชั่นสำหรับผู้ดูแลระบบนี้มีการเช็คเงื่อนไขเพิ่มเติมคือ request.user.is_staff สำหรับเช็คว่าเป็นผู้ดูแลระบบหรือไม่

โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_staff ว่าผู้ใช้งานที่เรียกใช้งานฟังก์นี้เป็นผู้ดูแลระบบหรือไม่ หากไม่ใช่ ให้ redirect กลับไปที่หน้า backendpage และ alert message รูปแบบ warning พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น”
- กำหนดตัวแปร session ของ menuactive ให้เป็น userlist (ที่จะนำไปเช็คในไฟล์ menu.html เพื่อให้เมนูนั้นๆ active)
- สร้างการ query ข้อมูลผู้ใช้งานทั้งหมดในระบบโดยเก็บไว้ในตัวแปร listuser และจัดเรียงข้อมูลโดยใช้คำสั่ง order_by('-first_name') ระบุคอลัมน์ที่ต้องการใช้เป็นเงื่อนไขในการจัดเรียนคือ first_name และเครื่องหมาย - การเรียงจากมากไปน้อย
- นำข้อมูลจากการ query จากด้านบนมาทำให้ pagination โดยได้กำหนด
    - ตัวแปร showperpage กำหนดจำนวนข้อมูลที่แสดงในหนึ่งหน้า
    - ตัวแปร page รับข้อมูลแบบ get จากตัวแปรที่ชื่อ page และหากไม่มีค่าให้กำหนดตั้งต้นเป็น 1
    - ตัวแปร paginator เรียกใช้งานฟังก์ชั่น Paginator() ที่ django เตรียมไว้ให้และส่งค่าข้อมูลที่ได้จากการ query และ ตัวแปร showperpage เข้าไปสองตัว Paginator(listpredict, showperpage)
    - ใช้ try except PageNotAnInteger และ EmptyPage ในการตรวจสอบความผิดพลาด
- ส่งค่า {'listuser':listuser} ไปแสดงผลที่ไฟล์ listuser.html

Template : listuser.html

สร้างไฟล์ชื่อ listuser.html โดยใช้โครงสร้างเดียวกับ backendpage.html
โดยที่

- เรียกใช้งาน {% load static %} ใต้บรรทัดการทำ extends
- กำหนด titlebar และ titlecontent ให้เป็น
    - {% block titlebar %} รายการผู้ใช้งานระบบ {% endblock %}
    - {% block titlecontent %} รายการผู้ใช้งานระบบ | list User {% endblock %}
- สร้างตารางแสดงข้อมูล โดยใช้สักษณะตารางเดียวกับหน้า history.html
- แสดงข้อมูลโดยใช้การลูป for in ในตาราง ซึ่งจะแสดงข้อมูล
    - ลำดับ ใช้เป็นตัวเลขลำดับการแสดงผล โดยใช้ตัวแปร forloop.counter0|add:listuser.start_index ซึ่ง add คือเพิ่มการบวกตัวเลข listuser.start_index จำนวนที่เริ่มต้นของหน้านั้นๆ เช่น แสดงจำนวนต่อหน้า 20 ถ้าอยู่หน้าที่ 2 ต้องเริ่มจาก 40 แล้วนำไปบวกกับ forloop.counter0 คือลำดับที่ของข้อมูลในหน้านั้นๆ เริ่มจาก 0 ที่ใช้ในการนับจำนวนให้โดยอัตโนมัติ โดยที่ listuser คือตัวแปรที่ส่งมาจาก views.py โดยเก็บข้อมูลจากการ query ของข้อมูล user ออกมา
    - คอลัมน์ของตารางมีทั้งหมด 5 คอลัมน์ ได้แก่
        - #
        - อีเมล
        - ชื่อ-สกุล
        - ประเภท แบ่งออกเป็น 2 ประเภทได้แก่ ผู้ใช้งานทั่วไป และ ผู้ดูแลระบบ โดยใช้การเช็ค is_staff ของผู้ใช้งาน {% if item.is_staff %} ผู้ดูแลระบบ {% else %} ผู้ใช้งาน {% endif %}
        - จัดการ สำหรับใส่ปุ่มแก้ไขข้อมูลผู้ใช้งาน และปุ่มลบบัญชีผู้ใช้งาน
    - การจัดการผู้ใช้งาน
        - ปุ่มแก้ไขข้อมูลผู้ใช้งาน ใส่สีของปุ่มคือสีเหลือง btn-warning และให้มีขนาดเล็ก btn-sm โดยใช้ชื่อของปุ่มเป็น edituser{{ item.id }} การระบุ {{ item.id }} เข้าไปเพื่อให้แต่ละปุ่มมีชื่อที่แตกต่างกันและเพื่อเรียกใช้งานในแต่ละคนที่ไม่ซ้ำกัน ซึ่งจะสัมพันธ์กับชื่อที่อยู่ใน popup (modal) ที่จะแสดงแบบฟอร์มสำหรับการแก้ไขข้อมูลผู้ใช้งาน
        - popup (modal) ระบุชื่อ id="edituser{{ item.id }}" สำหรับแสดงแบบฟอร์มสำหรับการแก้ไขข้อมูลผู้ใช้งาน
        - ปุ่มลบบัญชีผู้ใช้งาน ใส่สีของปุ่มคือสีแดง btn-danger และให้มีขนาดเล็ก btn-sm โดยใช้ชื่อของปุ่มเป็น deleteuser{{ item.id }} การระบุ {{ item.id }} เข้าไปเพื่อให้แต่ละปุ่มมีชื่อที่แตกต่างกันและเพื่อเรียกใช้งานในแต่ละคนที่ไม่ซ้ำกัน ซึ่งจะสัมพันธ์กับชื่อที่อยู่ใน popup (modal) ที่จะแสดงข้อมูลของผู้ใช้งานก่อนที่จะยืนยันการลบบัญชีผู้ใช้งาน

- popup (modal) ระบุชื่อ id="deleteuser{{ item.id }}" สำหรับแสดงข้อมูลผู้ใช้งานก่อนลบดำเนินการต่อ คือการกดยืนยันการลบข้อมูลผู้ใช้งาน

Python

```
<table class="table table-striped table-sm">
      <thead>
          <tr>
          <th scope="col-1" class="text-center">#</th>
          <th scope="col-3">อีเมล</th>
          <th scope="col-4">ชื่อ-สกุล</th>
          <th scope="col-2">ประเภท</th>
          <th scope="col-2">จัดการ</th>
          </tr>
      </thead>
      <tbody>
          {% for item in listuser %}
          <tr>
              <th class="text-center">{{
forloop.counter0|add:listuser.start_index }}</th>
              <td class="align-top">
                  {{ item.email }}
              </td>
              <td class="align-top">{{ item.first_name }} {{ item.last_name
}}</td>
              <td class="align-top">
                  {% if item.is_staff %} ผู้ดูแลระบบ {% else %} ผู้ใช้งาน {% endif %}
              </td>
              <td class="align-top">
                  <a class="btn btn-warning btn-sm" href="#"
data-bs-toggle="modal" data-bs-target="#edituser{{ item.id }}" role="button">
แก้ไข</a>
                  <a class="btn btn-danger btn-sm" href="#"
data-bs-toggle="modal" data-bs-target="#deleteuser{{ item.id }}" role="button">
ลบ</a>

                  <div class="modal fade" id="deleteuser{{ item.id }}"
tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
                    <div class="modal-dialog">
                      <div class="modal-content">
                        <div class="modal-header">
                          <h1 class="modal-title fs-3">ยืนยันการลบผู้ใช้งาน</h1>
                          <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                        </div>
```

```
                        <div class="modal-body text-dark">


                        </div>
                      </div>
                    </div>
                  </div>

                  <div class="modal fade" id="edituser{{ item.id }}"
tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
                      <div class="modal-dialog">
                        <div class="modal-content">
                          <div class="modal-header">
                            <h1 class="modal-title fs-3">แก้ไขบัญชีผู้ใช้งาน</h1>
                            <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                          </div>
                          <div class="modal-body text-dark">


                          </div>
                        </div>
                      </div>
                  </div>

              </td>
          </tr>
          {% endfor %}
      </tbody>
    </table>
```

- แสดงผลการทำ pagination

Python

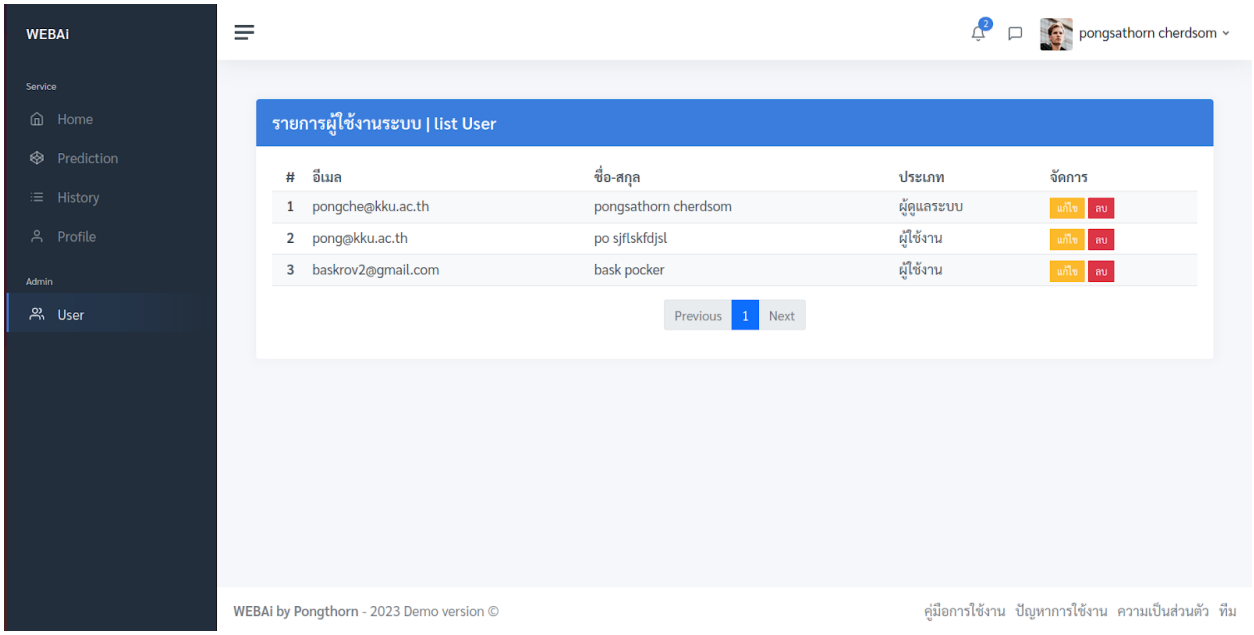
```
<div class="d-flex mt-3">
        <div class="mx-auto">
          <nav >
            <ul class="pagination">
                {% if listuser.has_previous %}
                <li class="page-item">
```

```
                  <a href="?page={{ listuser.previous_page_number }}"
class="page-link">Previous</a>
                </li>
                <li>
                  <a class="page-link" href="?page={{
listuser.previous_page_number }}">
                    {{ listuser.previous_page_number }}
                  </a>
                </li>
                {% else %}
                <li class="page-item disabled">
                  <span class="page-link">Previous</span>
                </li>
                {% endif %}

                <li class="page-item active">
                  <span class="page-link">{{ listuser.number }}</span>
                </li>

                {% if listuser.has_next %}
                <li class="page-item">
                  <a class="page-link" href="?page={{ listuser.next_page_number
}}">
                    {{ listuser.next_page_number }}
                  </a>
                </li>
                <li class="page-item ">
                  <a class="page-link" href="?page={{ listuser.next_page_number
}}">Next</a>
                </li>
                {% else %}
                <li class="page-item disabled">
                  <span class="page-link">Next</span>
                </li>
                {% endif %}

              </ul>
            </nav>
          </div>
        </div>
```

ทดสอบ Runserver

python3 manage.py runserver

http://127.0.0.1:8000/backend/listuser/



31. การสร้างฟังก์ชั่นเพื่อลบบัญชีผู้ใช้งาน (สำหรับผู้ดูแลระบบ)

Template : listuser.html

ทำงานที่ไฟล์ listuser.html ในส่วนของปุ่มการลบบัญชีผู้ใช้งาน

ปุ่ม สำหรับการเรียกใช้งาน popup (modal) โดยใช้ชื่อว่า data-bs-target="#deleteuser{{ item.id }}"

```Python
<a class="btn btn-danger btn-sm" href="#" data-bs-toggle="modal"
data-bs-target="#deleteuser{{ item.id }}" role="button">ลบ</a>
```

Popup (modal) สำหรับแสดงข้อมูลผู้ใช้งานก่อน ยืนยันการลบข้อมูล โดยกำหนดชื่อของ popup คือ id="deleteuser{{ item.id }}" และปุ่มสำหรับยืนยันการลบข้อมูลในส่วนของ href ให้ระบุเป็น {% url 'backenddeleteuser' item.username %} โดยที่ url 'deleteuser' คือชื่อของเส้นทาง url ที่เรียกใช้งาน และ item.username คือข้อมูลที่ต้องการส่งไปพร้อมกับ url เส้นนี้

```Python
<div class="modal fade" id="deleteuser{{ item.id }}" tabindex="-1"
aria-labelledby="exampleModalLabel" aria-hidden="true">
                  <div class="modal-dialog">
                    <div class="modal-content">
                      <div class="modal-header">
                        <h1 class="modal-title fs-3">ยืนยันการลบผู้ใช้งาน</h1>
                        <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                      </div>
                      <div class="modal-body text-dark">
                        ชื่อ-สกุล {{ item.first_name }} {{ item.last_name }} <br>
                        อีเมล {{ item.email }} <br><br>
                       <a class="btn btn-primary" href="{% url
'backenddeleteuser' item.username %}" role="button">ยืนยันการลบข้อมูล</a>
                        <button type="button" class="btn btn-secondary" data-bs
-dismiss="modal">ยกเลิกการลบ</button>
                      </div>
                    </div>
                  </div>
</div>
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทางสำหรับการเรียกใช้งานฟังก์ชั่นการลบข้อมูลผู้ใช้งานคือ backend/deleteuser/<str:username> และกำหนดการรับค่า username โดยกำหนดชนิดของตัวแปรเป็น str คือ string ตัวอักษร และเรียกใช้งานฟังก์ชั่นที่ชื่อว่า backenddeleteuser กำหนดชื่อเรียกของเส้นทางนี้คือ backenddeleteuser (ซึ่งจะเป็นชื่อเรียกใน tag <a> ที่เรียกใช้งานผ่านหน้า html)

```Python
path('backend/deleteuser/<str:username>', views.backenddeleteuser,
name='backenddeleteuser'),
```

File : views.py (อยู่ใน backendai)

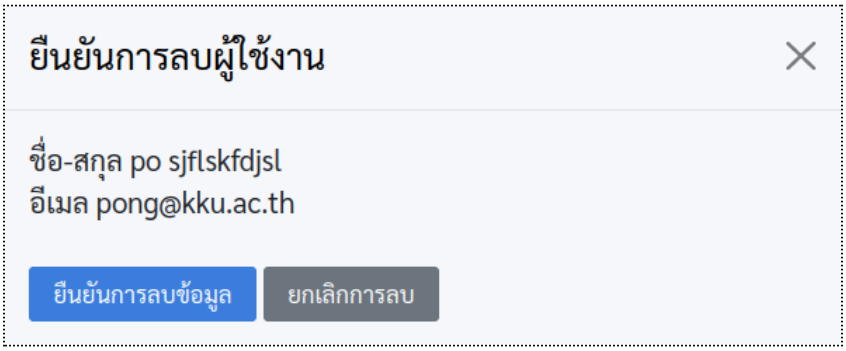
สร้างฟังก์ชั่นที่ชื่อว่า backenddeleteuser และรับค่าตัวแปร username ชื่อเดียวกับที่สร้างใน url จะได้เป็น def backenddeleteuser(request,username) สำหรับการลบข้อมูลผู้ใช้งาน

โดยที่

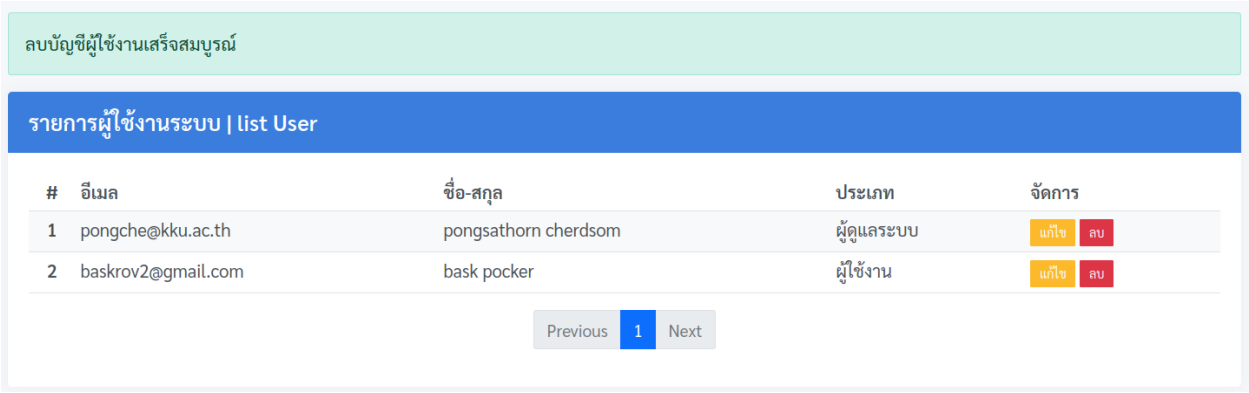
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_staff ว่าผู้ใช้งานที่เรียกใช้งานฟังก์นี้เป็นผู้ดูแลระบบหรือไม่ หากไม่ใช่ ให้ redirect กลับไปที่หน้า backendpage และ alert message รูปแบบ warning พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น”
- ตรวจสอบข้อมูลจากตัวแปร username (ข้อมูลอีเมล) ที่ถูกส่งมาจากหน้าบ้าน ตรวจสอบจากฐานข้อมูล user โดยกำหนดเงื่อนไขคือ email=username โดยใช้ try: และ except: ในการตรวจเช็ค error ที่จะเกิดขึ้นในกรณีที่ไม่มีข้อมูลที่ส่งมา หรือมีข้อมูลผู้ใช้งานมากกว่า 1 คน หาก except จะให้ getuser = None กรณีที่ทำงานได้ปกติ จะให้ getuser = User.objects.get(email=username)
- เช็คข้อมูลจากการ query ในตัวแปร getuser หากไม่มีข้อมูลให้ redirect กลับไปที่ url name backendlistuser และ alert message รูปแบบ error พร้อมกับข้อความ “เกิดข้อผิดพลาด ไม่พบข้อมูลผู้ใช้งาน” หากมีข้อมูลให้ดำเนินการลบข้อมูลโดยใช้คำสั่ง getuser.delete() ซึ่งสามารถเรียกใช้งานตัวแปร getuser ได้ต่อเนื่องจากการ query ด้านบน ซึ่งในการ User.objects.get(email=username) คือการระบุผู้ที่ต้องการลบจาก email เรียบร้อยแล้ว ไม่จำเป็นต้อง query หรือระบุใหม่อีกรอบ
- เช็คการลบข้อมูลผู้ใช้งานโดยเช็คจากตัวแปร getuser เช่นเดิม if getuser: หากลบข้อมูลได้ปกติ ให้ redirect กลับไปที่ url name backendlistuser และ alert message รูปแบบ success พร้อมกับข้อความ “ลบบัญชีผู้ใช้งานเสร็จสมบูรณ์” กรณีเกิดข้อผิดพลาดไม่สามารถลบข้อมูลได้ ให้ redirect กลับไปที่ url name backendlistuser และ alert message รูปแบบ error พร้อมกับข้อความ “เกิดข้อผิดพลาด ไม่สามารถลบผู้ใช้งานได้ โปรดลองใหม่อีกครั้ง”

ทดสอบ Runserver

python3 manage.py runserver
http://127.0.0.1:8000/backend/listuser/



เมื่อลบข้อมูลผู้ใช้งานเสร็จสมบูรณ์



32. การสร้างฟังก์ชั่นเพื่ออัพเดตข้อมูลผู้ใช้งาน (สำหรับผู้ดูแลระบบ)

Template : listuser.html

ทำงานที่ไฟล์ listuser.html ในส่วนของปุ่มการแก้ไขบัญชีผู้ใช้งาน
ปุ่ม สำหรับการเรียกใช้งาน popup (modal) โดยใช้ชื่อว่า data-bs-target="#edituser{{ item.id }}"

```
Python
<a class="btn btn-warning btn-sm" href="#" data-bs-toggle="modal"
data-bs-target="#edituser{{ item.id }}" role="button">แก้ไข</a>
```

Popup (modal) สำสหรับแสดงแบบฟอร์มข้อมูลผู้ใช้งานก่อน โดยกำหนดชื่อของ popup คือ id="edituser{{ item.id }}" โดยในส่วนของ <form> ระบุ action ไปที่ url name backendupdateuser action="{% url 'backendupdateuser' %}"

โดยที่

- ทุก input ให้ระบุ value ทุก input เพื่อให้แสดงข้อมูลของผู้ใช้งานเดิมในช่องกรอกข้อมูลนั้น ๆ ให้สอดคล้องกับชื่อของ input
- Input name = email ให้ระบุเป็น value="{{ item.email }}" และจะปิดการแก้ไขโดยระบุ disabled ในช่องนี้
- Input name = firstname ให้ระบุเป็น value="{{ item.first_name }}"
- Input name = lastname ให้ระบุเป็น value="{{ item.last_name }}"
- Input name = password ให้ระบุเป็น value="{{ item.password }}"
- และเพิ่ม input name=iduser สำหรับใช้ส่งข้อมูล iduser ไปในแบบฟอร์มเพื่อเป็นเงื่อนไขในระบุผู้ใช้งานที่ต้องการอัพเดตข้อมูลด้วย และปิดการแสดงผลให้ผู้ใช้งานเห็นโดยระบุ hidden

```
Python
<div class="modal fade" id="edituser{{ item.id }}" tabindex="-1"
aria-labelledby="exampleModalLabel" aria-hidden="true">
```

```
                    <div class="modal-dialog">
                      <div class="modal-content">
                        <div class="modal-header">
                          <h1 class="modal-title fs-3">แก้ไขบัญชีผู้ใช้งาน</h1>
                          <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                        </div>
                        <div class="modal-body text-dark">
                        <form action="{% url 'backendupdateuser' %}"
method="POST">
                        {% csrf_token %}
                         <input type="text" name="iduser" value="{{ item.id }}"
hidden required/>
                        <div class="mb-3">
                        <label class="form-label">Email</label>
                        <input class="form-control form-control-lg"
type="email" name="email" value="{{ item.email }}" disabled placeholder="Enter
your email" required/>
                        </div>
                       <div class="mb-3">
                       <label class="form-label">First name</label>
                       <input class="form-control form-control-lg" type="text"
name="firstname" value="{{ item.first_name }}" placeholder="Enter your name"
required/>
                    </div>
                    <div class="mb-3">
                    <label class="form-label">Last name</label>
                    <input class="form-control form-control-lg" type="text"
name="lastname" value="{{ item.last_name }}" placeholder="Enter your name"
required/>
                    </div>
                    <div class="mb-3">
                    <label class="form-label">Password</label>
                    <input class="form-control form-control-lg" type="password"
name="password" value="{{ item.password }}" placeholder="Enter password"
required/>
                    </div>
                    <div class="d-grid gap-2 mt-3">
                      <button type="submit" class="btn btn-primary">อัพเดตข้อมูลผู้ใช้
งาน</button>
                       <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">ยกเลิกการอัพเดต</button>
                    </div>
                </form>
```

```
            </div>
        </div>
    </div>
</div>
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทาง กำหนดให้เป็น backend/updateuser/ เรียกใช้งานฟังก์ชั่น backendupdateuser และกำหนดชื่อเส้นทางนี้เป็น backendupdateuser เพื่อให้สอดคล้องกับหน้า html ที่ได้กำหนดไว้ในขั้นตอนก่อนหน้านี้ในส่วนการ action ของ from

Python
```
path('backend/updateuser/', views.backendupdateuser, name='backendupdateuser'),
```

File : views.py (อยู่ใน backendai)

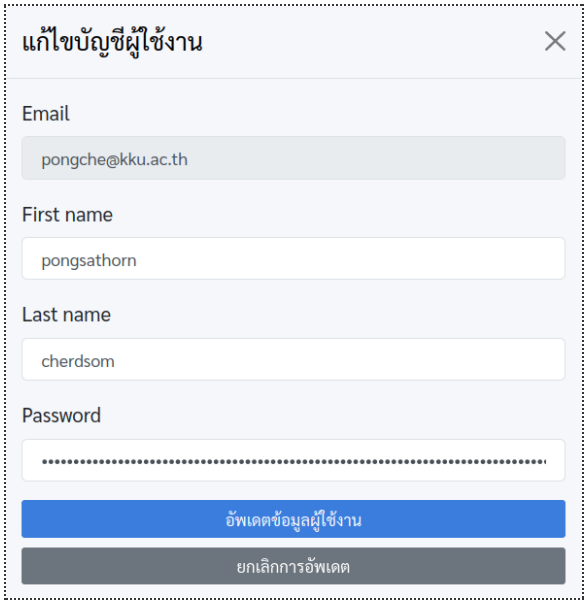
สร้างฟังก์ชั่น backendupdateuser ชื่อเดียวกันกับที่ระบุใน url views.backendupdateuser ในขั้นตอนก่อนหน้า
โดยที่

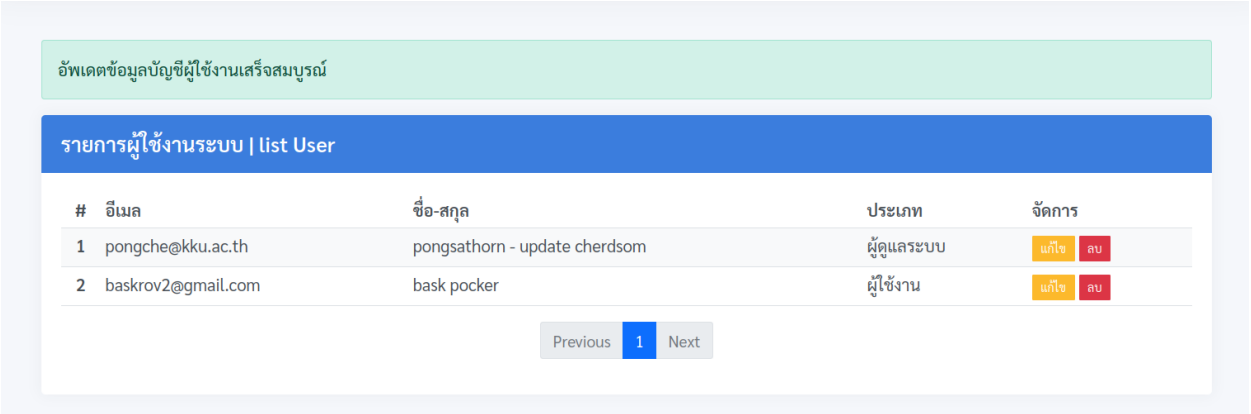
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_staff ว่าผู้ใช้งานที่เรียกใช้งานฟังก์นี้เป็นผู้ดูแลระบบหรือไม่ หากไม่ใช่ ให้ redirect กลับไปที่หน้า backendpage และ alert message รูปแบบ warning พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น”
- สร้างตัวแปรเพื่อรับข้อมูลจากแบบฟอร์มจากหน้าบ้าน กรณีที่ไม่ได้ส่งค่ามาด้วยจะกำหนดให้ค่าเป็น None
    - iduser = request.POST.get('iduser',None)
    - firstname = request.POST.get('firstname',None)
    - lastname = request.POST.get('lastname',None)
    - password = request.POST.get('password',None)
- ตรวจสอบข้อมูลจากตัวแปร iduser (ข้อมูลไอดีผู้ใช้งาน) ที่ถูกส่งมาจากหน้าบ้าน ตรวจสอบจากฐานข้อมูล user โดยกำหนดเงื่อนไขคือ id=iduser โดยใช้ try: และ except: ในการตรวจเช็ค error ที่จะเกิดขึ้นในกรณีที่ไม่มีข้อมูลที่ส่งมา หรือมีข้อมูลผู้ใช้งานมากกว่า 1 คน หาก except จะให้ getuser = None กรณีที่ทำงานได้ปกติ จะให้ getuser = User.objects.get(id=iduser)

- เช็คข้อมูลจากการ query ในตัวแปร getuser หากไม่มีข้อมูลให้ redirect กลับไปที่ url name backendlistuser และ alert message รูปแบบ error พร้อมกับข้อความ “เกิดข้อผิดพลาด ไม่พบข้อมูลผู้ใช้งาน” หากมีข้อมูลให้ดำเนินอัพเดตข้อมูลผู้ใช้งานถัดไป
- การอัพเดตข้อมูลผู้ใช้งานสามารถเรียกใช้ตัวแปร getuser ได้ทันที เนื่องจากมีการระบุข้อผู้ใช้งานที่ต้องการอัพเดตข้อมูลจากการ query ที่ผ่านมาแล้ว สามารถทำได้โดย
    - getuser.first_name = firstname
    - getuser.last_name = lastname
- เช็คการเปรียบเทียบรหัสผ่านที่ส่งมาจากหน้าบ้าน และ รหัสผ่านที่อยู่ในฐานข้อมูลเดิม หากไม่มีการเปลี่ยนแปลงรหัสผ่านจากหน้าบ้านจะไม่แก้ไขข้อมูลในฐานข้อมูล และหากมีการเปลี่ยนแปลงรหัสผ่านจากหน้าบ้านต้องนำข้อมูลนั้น มาเข้ารหัสเหมือนขั้นตอนการเข้ารหัสของรหัสผ่านจากหน้าบ้านก่อนเก็บเข้าฐานข้อมูลอีกครั้ง โดยใช้คำสั่ง if getuser.password != password: จะทำงานกรณีที่ไม่ตรงกันเท่านั้น โดยเรียกใช้งานฟังก์ชั่น make_password และผ่านค่าตัวแปร password ที่ส่งมาจากหน้าบ้านในการเข้ารหัสและบันทึกเข้าฐานข้อมูล โดยใช้คำสั่ง etuser.password = make_password(password)
- จากนั้นบันทึกการอัพเดตข้อมูลโดยใช้คำสั่ง getuser.save()
- เช็คการบันทึกข้อมูลผู้ใช้งานโดยเช็คจากตัวแปร getuser เช่นเดิม if getuser: หากลันทึกข้อมูลได้ปกติ ให้ redirect กลับไปที่ url name backendlistuser และ alert message รูปแบบ success พร้อมกับข้อความ “อัพเดตข้อมูลบัญชีผู้ใช้งานเสร็จสมบูรณ์” กรณีเกิดข้อผิดพลาดไม่สามารถลบข้อมูลได้ ให้ redirect กลับไปที่ url name backendlistuser และ alert message รูปแบบ error พร้อมกับข้อความ “เกิดข้อผิดพลาด ไม่สามารถอัพเดตข้อมูลผู้ใช้งานได้ โปรดลองใหม่อีกครั้ง”

ทดสอบ Runserver

python3 manage.py runserver
http://127.0.0.1:8000/backend/listuser/





33. การสร้างฟังก์ชั่นเพื่อสร้างบัญชีผู้ใช้งาน (สำหรับผู้ดูแลระบบ)

Template : listuser.html

ทำงานที่ไฟล์ listuser.html ทำการสร้างปุ่มสร้างบัญชีผู้ใช้งานในรูปแบบของ popup (modal) โดยสร้างปุ่มโดยใช้คำสั่งและใช้ชื่อว่า data-bs-target="#createuser" และกำหนดให้แสดงด้านขวาของหน้าจอ

Python

```
<div class="text-end">
  <button type="button" class="btn btn-primary" data-bs-toggle="modal"
      data-bs-target="#createuser">
      สร้างบัญชีผู้ใช้งาน
  </button>
```

```
</div>
```

ส่วนของการแสดง Popup (modal) สำหรับใช้แสดงแบบฟอร์มสำหรับบันทึกข้อมูลผู้ใช้งาน จะรวมข้อมูลของแบบฟอร์มสำหรับกรอกข้อมูลสมาชิกเหมือนกับหน้า register จากหน้าหลักของระบบที่ได้ทำผ่านมาแล้ว ในแบบฟอร์ม action ไปที่ฟังก์ชั่น backendcreateuser
โดยที่จะประกอบไปด้วย

- Input name = firstname และ type = text
- Input name = lastname และ type = text
- Input name = email และ type = email
- Input name = password และ type = password
- ทุกๆ input จะไม่มีการระบุค่า value ไว้ให้

Python

```
<div class="modal fade" id="createuser" tabindex="-1"
aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title fs-5" id="exampleModalLabel">สร้างบัญชีผู้ใช้งาน</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <form action="{% url 'backendcreateuser' %}" method="POST">
          {% csrf_token %}
          <div class="mb-3">
            <label class="form-label">First name</label>
            <input class="form-control form-control-lg" type="text"
name="firstname" placeholder="Enter your name" required/>
          </div>
          <div class="mb-3">
            <label class="form-label">Last name</label>
            <input class="form-control form-control-lg" type="text"
name="lastname" placeholder="Enter your name" required/>
          </div>
          <div class="mb-3">
            <label class="form-label">Email</label>
            <input class="form-control form-control-lg" type="email"
name="email" placeholder="Enter your email" required/>
          </div>
```

```
          <div class="mb-3">
            <label class="form-label">Password</label>
            <input class="form-control form-control-lg" type="password"
name="password" placeholder="Enter password" required/>
          </div>
          <div class="d-grid gap-2 mt-3">
            <button type="submit" class="btn btn-primary">Sign up</button>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทาง กำหนดให้เป็น backend/createuser/ เรียกใช้งานฟังก์ชั่น backendcreateuser และกำหนดชื่อเส้นทางนี้เป็น backendcreateuser เพื่อให้สอดคล้องกับหน้า html ที่ได้กำหนดไว้ในขั้นตอนก่อนหน้านี้ในส่วนการ action ของ from

Python
```
path('backend/createuser/', views.backendcreateuser, name='backendcreateuser')
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่น backendcreateuser ชื่อเดียวกันกับที่ระบุใน url views.backendcreateuser ในขั้นตอนก่อนหน้า
โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_staff ว่าผู้ใช้งานที่เรียกใช้งานฟังก์นี้เป็นผู้ดูแลระบบหรือไม่ หากไม่ใช่ ให้ redirect กลับไปที่หน้า backendpage และ alert message รูปแบบ warning พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น”
- สร้างตัวแปรเพื่อรับข้อมูลจากแบบฟอร์มจากหน้าบ้าน จะประกอบไปด้วย

    - email = request.POST.get('email',None)
    - firstname = request.POST.get('firstname',None)
    - lastname = request.POST.get('lastname',None)
    - password = request.POST.get('password',None)
- ทำการตรวจสอบ email ที่ส่งมาจากหน้าบ้านว่ามีอีเมลนี้ในระบบแล้วหรือไม่ โดยใช้คำสั่งลักษณะเดียวกับการสมัครสมาชิกจากหน้าบ้าน หากมีแล้วให้ redirect กลับไปที่ url name backendlistuser และ alert message รูปแบบ error พร้อมกับข้อความ “ชื่อผู้ใช้งานซ้ำกับข้อมูลในระบบ โปรดสมัครใหม่อีกครั้ง” หากไม่มีให้ดำเนินสร้างบัญชีผู้ใช้งานถัดไป
- ทำการเพิ่มข้อมูลผู้ใช้งานลักษณะเดียวกันกับหน้าสมัครสมาชิกจากหน้าบ้าน โดยใช้ฟังก์ชั่น create_user
- กรณีที่ ถ้าหากสามารถเพิ่มข้อมูลผู้ใช้งานได้ให้ redirect กลับไปที่หน้า homepage เพื่อทำการลงชื่อเข้าใช้งาน พร้อมกับระบุข้อข้อความ alert message รูปแบบ success “เพิ่มข้อมูลสมาชิกเสร็จสมบูรณ์” หากไม่สามารถเพิ่มข้อมูลได้โดยจาก error ใดๆ ให้ redirect ไปที่หน้า backendlistuser พร้อมกับระบุข้อข้อความ alert message รูปแบบ error “เกิดข้อผิดพลาด ไม่สามารถเพิ่มข้อมูลได้ โปรดลองใหม่อีกครั้ง”

Python

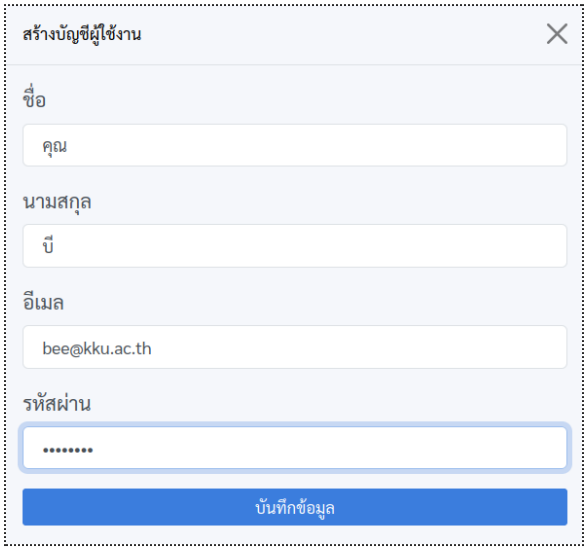
```python
def backendcreateuser(request):
    if request.user.is_authenticated:
        if request.user.is_staff:
            email = request.POST.get('email',None)
            firstname = request.POST.get('firstname',None)
            lastname = request.POST.get('lastname',None)
            password = request.POST.get('password',None)

            checkuser = User.objects.filter(username=email).count()
            if checkuser == 0:
                createuser =
User.objects.create_user(username=email,email=email,password=password)
                createuser.first_name = firstname
                createuser.last_name = lastname
                createuser.save()
                if createuser:
                    messages.success(request,"เพิ่มข้อมูลสมาชิกเสร็จสมบูรณ์")
                    return redirect(backendlistuser)
                else:
                    messages.error(request,"เกิดข้อผิดพลาด ไม่สามารถเพิ่มข้อมูลได้ โปรด
ลองใหม่อีกครั้ง")
                    return redirect(backendlistuser)
            else:
                messages.error(request,'เกิดข้อผิดพลาด ไม่พบข้อมูลผู้ใช้งาน')
                return redirect(backendlistuser)
```

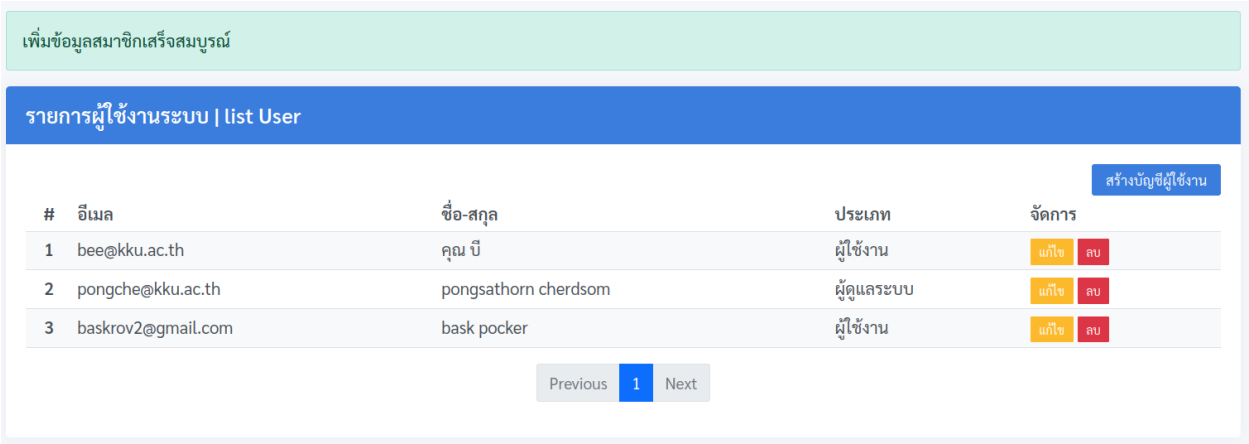
```
        else:
            messages.warning(request,'ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น')
            return redirect(backendpage)
    else:
        messages.error(request,'ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน')
        return redirect(homepage)
```

ทดสอบ Runserver

python3 manage.py runserver

http://127.0.0.1:8000/backend/listuser/





34. การสร้างหน้าสำหรับแสดงรายการ Prediction ทั้งระบบ (สำหรับผู้ดูแลระบบ)

Template : menu.html (in include folder)

จะทำการเพิ่มเมนูขึ้นมาใหม่ สำหรับผู้ดูแลระบบชื่อเมนู Prediction ไปที่ไฟล์ include -> menu.html

สำหรับเมนูของผู้ดูแลระบบ จะเช็คว่าผู้ใช้งานนั้นเป็นผู้ดูแลระบบหรือไม่ โดยใช้คำสั่ง {% if request.user.is_admin %} สำหรับการเช็ค โดยที่ is_admin คือคอลัมน์ที่ใช้เก็บข้อมูลที่ระบุว่า user นั้นๆ เป็นผู้ดูแลระบบหรือไม่ และเมนูของผู้ดูแลระบบทั้งหมดจะอยู่ภายในเงื่อนไขนี้ทั้งหมด

โดยเรียกใช้งาน url name backendlistprediction และกำหนด request.session.menuactive เป็น listprediction (รวมกับเมนูก่อนหน้า)

Python

```
{% if request.user.is_staff %}
            <li class="sidebar-header">
                Admin
            </li>
            <li class="sidebar-item {% if request.session.menuactive ==
'userlist' %} active {% endif %}">
                <a class="sidebar-link" href="{% url 'backendlistuser' %}">
                    <i class="align-middle" data-feather="users"></i> <span
class="align-middle">User</span>
                </a>
            </li>
            <li class="sidebar-item {% if request.session.menuactive ==
'listprediction' %} active {% endif %}">
                <a class="sidebar-link" href="{% url 'backendlistprediction'
%}">
                    <i class="align-middle" data-feather="list"></i> <span
class="align-middle">Prediction</span>
                </a>
            </li>
{% endif %}
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทาง กำหนดให้เป็น backend/listprediction/ เรียกใช้งานฟังก์ชั่น backendlistprediction และกำหนดชื่อเส้นทางนี้เป็น backendlistprediction เพื่อให้สอดคล้องกับหน้า html ที่ได้กำหนดไว้ในขั้นตอนก่อนหน้านี้

```
Python
path('backend/listprediction/', views.backendlistprediction,
name='backendlistprediction')
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่นชื่อว่า backendlistprediction ชื่อเดียวกันกับที่ระบุใน url views.backendlistprediction ในขั้นตอนก่อนหน้า ซึ่งในฟังก์ชั่นสำหรับผู้ดูแลระบบนี้มีการเช็คเงื่อนไขเพิ่มเติมคือ request.user.is_staff สำหรับเช็คว่าเป็นผู้ดูแลระบบหรือไม่
โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_staff ว่าผู้ใช้งานที่เรียกใช้งานฟังก์นี้เป็นผู้ดูแลระบบหรือไม่ หากไม่ใช่ ให้ redirect กลับไปที่หน้า backendpage และ alert message รูปแบบ warning พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น”
- กำหนดตัวแปร session ของ menuactive ให้เป็น listprediction (ที่จะนำไปเช็คในไฟล์ menu.html เพื่อให้เมนูนั้นๆ active)
- สร้างการ query ข้อมูลผู้ใช้งานทั้งหมดในระบบโดยเก็บไว้ในตัวแปร listprediction.html และจัดเรียงข้อมูลโดยใช้คำสั่ง order_by('-datetimepre) ระบุคอลัมน์ที่ต้องการใช้เป็นเงื่อนไขในการจัดเรียนคือ datetimepre และเครื่องหมาย - การเรียงจากมากไปน้อย ถ้าเป็นเวลาจะเรียงจากเวลาปัจจุบันไปอดีต
- นำข้อมูลจากการ query จากด้านบนมาทำให้ pagination โดยได้กำหนด
    - ตัวแปร showperpage กำหนดจำนวนข้อมูลที่แสดงในหนึ่งหน้า
    - ตัวแปร page รับข้อมูลแบบ get จากตัวแปรที่ชื่อ page และหากไม่มีค่าให้กำหนดตั้งต้นเป็น 1
    - ตัวแปร paginator เรียกใช้งานฟังก์ชั่น Paginator() ที่ django เตรียมไว้ให้และส่งค่าข้อมูลที่ได้จากการ query และ ตัวแปร showperpage เข้าไปสองตัว Paginator(listpredict, showperpage)
    - ใช้ try except PageNotAnInteger และ EmptyPage ในการตรวจสอบความผิดพลาด
- ส่งค่า {'listprediction':listprediction} ไปแสดงผลที่ไฟล์ listprediction.html

Template : listprediction.html

สร้างไฟล์ชื่อ listprediction.html โดยใช้โครงสร้างเดียวกับ backendpage.html
โดยที่

- เรียกใช้งาน {% load static %} ใต้บรรทัดการทำ extends

- กำหนด titlebar และ titlecontent ให้เป็น
    - {% block titlebar %} รายการพยากรณ์ข้อมูลทั้งหมดในระบบ {% endblock %}
    - {% block titlecontent %} รายการพยากรณ์ข้อมูลทั้งหมดในระบบ | list Prediction {% endblock %}
- สร้างตารางแสดงข้อมูล โดยใช้สักษณะตารางเดียวกับหน้า history.html
- แสดงข้อมูลโดยใช้การลูป for in ในตาราง ซึ่งจะแสดงข้อมูล
    - ลำดับ ใช้เป็นตัวเลขลำดับการแสดงผล โดยใช้ตัวแปร forloop.counter0|add:listprediction.start_index ซึ่ง add คือเพิ่มการบวกตัวเลข listprediction.start_index จำนวนที่เริ่มต้นของหน้านั้นๆ เช่น แสดงจำนวนต่อหน้า 20 ถ้าอยู่หน้าที่ 2 ต้องเริ่มจาก 40 แล้วนำไปบวกกับ forloop.counter0 คือลำดับที่ของข้อมูลในหน้านั้นๆ เริ่มจาก 0 ที่ใช้ในการนับจำนวนให้โดยอัตโนมัติ โดยที่ listprediction คือตัวแปรที่ส่งมาจาก views.py โดยเก็บข้อมูลจากการ query ของข้อมูล user ออกมา
    - คอลัมน์ของตารางมีทั้งหมด 5 คอลัมน์ ได้แก่
        - #
        - รายการ
        - ผู้ดำเนินการ
        - รูปแบบ
        - ลบข้อมูล
    - การจัดการพยากรณ์ในระบบทั้งหมด
        - ปุ่มลบข้อมูลการพยากรณ์ ใส่สีของปุ่มคือสีแดง btn-danger และให้มีขนาดเล็ก btn-sm โดยใช้ชื่อของปุ่มเป็น deletedata{{ item.indexpredict }} การระบุ {{ item.indexpredict }} เข้าไปเพื่อให้แต่ละปุ่มมีชื่อที่แตกต่างกันและเพื่อเรียกใช้งานในแต่ละคนที่ไม่ซ้ำกัน ซึ่งจะสัมพันธ์กับชื่อที่อยู่ใน popup (modal) ที่จะแสดงข้อมูลของผู้ใช้งานก่อนที่จะยืนยันการลบข้อมูลการพยากรณ์
        - popup (modal) ระบุชื่อ id="deletedata{{ item.indexpredict }}" สำหรับแสดงข้อมูลพยากรณ์ก่อนลบดำเนินการต่อ คือการกดยืนยันการลบข้อมูลการพยากรณ์

Python

```
<table class="table table-striped table-sm">
    <thead>
        <tr>
        <th scope="col-1" class="text-center">#</th>
        <th scope="col-3">รายการ</th>
        <th scope="col-4">ผู้ดำเนินการ</th>
        <th scope="col-2">ประเภท</th>
        <th scope="col-2">จัดการ</th>
```

```
        </tr>
    </thead>
    <tbody>
        {% for item in listprediction %}
        <tr>
            <th class="text-center">{{
forloop.counter0|add:listprediction.start_index }}</th>
            <td class="align-top">
                รายการ {{ item.datetimepre|date:'d-m-Y H:i:s' }}
            </td>
            <td class="align-top">{{ item.refuser.first_name }} {{
item.refuser.last_name }}</td>
            <td class="align-top">
                {{ item.typepredict }}
            </td>
            <td class="align-top">
                <a class="btn btn-danger btn-sm" href="#"
data-bs-toggle="modal" data-bs-target="#deletedata{{ item.indexpredict }}" role
="button">ลบข้อมูล</a>

                <div class="modal fade" id="deletedata{{ item.indexpredict }}"
tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
                  <div class="modal-dialog">
                    <div class="modal-content">
                      <div class="modal-header">
                        <h1 class="modal-title fs-3">ยืนยันการลบข้อมูล</h1>
                        <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                      </div>
                      <div class="modal-body text-dark">


                      </div>
                    </div>
                  </div>
                </div>
            </td>
        </tr>
        {% endfor %}
    </tbody>
  </table>
```

- แสดงผลการทำ pagination

Python

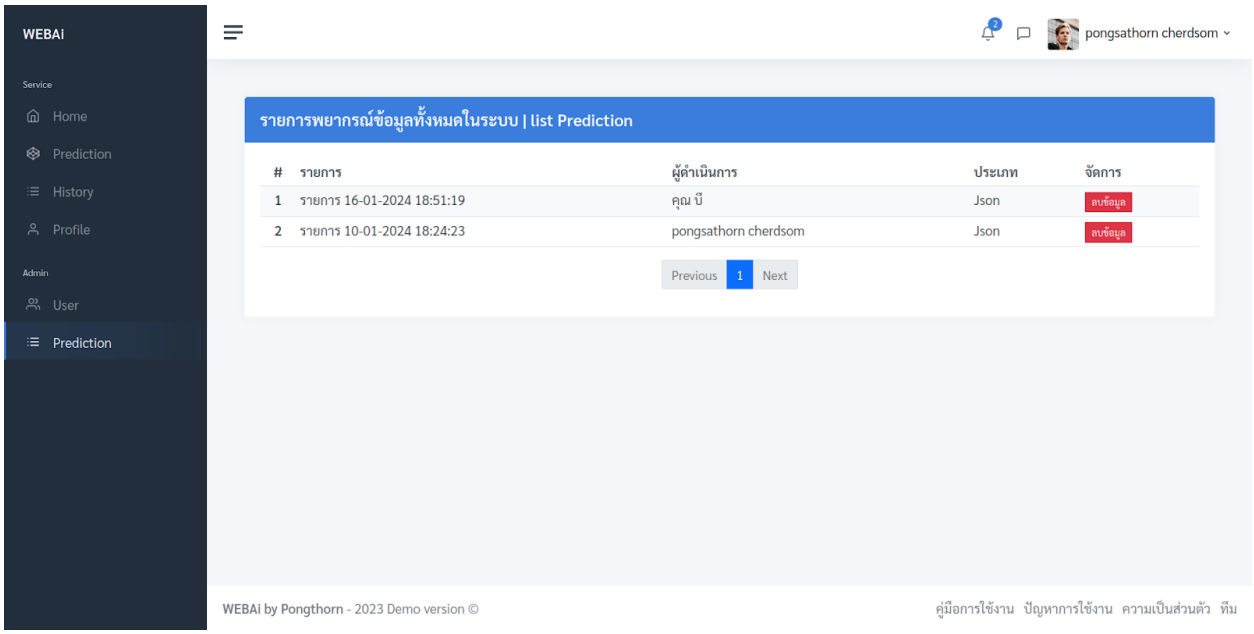
```
<div class="d-flex mt-3">
    <div class="mx-auto">
      <nav >
        <ul class="pagination">
            {% if listprediction.has_previous %}
            <li class="page-item">
              <a href="?page={{ listprediction.previous_page_number }}"
class="page-link">Previous</a>
            </li>
            <li>
              <a class="page-link" href="?page={{
listprediction.previous_page_number }}">
                {{ listprediction.previous_page_number }}
              </a>
            </li>
            {% else %}
            <li class="page-item disabled">
              <span class="page-link">Previous</span>
            </li>
            {% endif %}

            <li class="page-item active">
              <span class="page-link">{{ listprediction.number }}</span>
            </li>

            {% if listprediction.has_next %}
            <li class="page-item">
              <a class="page-link" href="?page={{
listprediction.next_page_number }}">
                {{ listprediction.next_page_number }}
              </a>
            </li>
            <li class="page-item ">
              <a class="page-link" href="?page={{
listprediction.next_page_number }}">Next</a>
            </li>
            {% else %}
            <li class="page-item disabled">
              <span class="page-link">Next</span>
            </li>
            {% endif %}

          </ul>
      </nav>
```

```
    </div>
  </div>
```

ทดสอบ Runserver

python3 manage.py runserver

http://127.0.0.1:8000/backend/listprediction/



35. การสร้างฟังก์ชั่นเพื่อลบรายการ Prediction (สำหรับผู้ดูแลระบบ)

Template : listprediction.html

ทำงานที่ไฟล์ listprediction.html ในส่วนของปุ่มการลบข้อมูลการพยากรณ์

ปุ่ม สำหรับการเรียกใช้งาน popup (modal) โดยใช้ชื่อว่า data-bs-target="#deletedata{{ item.indexpredict }}"

Python

```
<a class="btn btn-danger btn-sm" href="#" data-bs-toggle="modal"
data-bs-target="#deletedata{{ item.indexpredict }}" role="button">ลบข้อมูล</a>
```

Popup (modal) สำหรับแสดงข้อมูลการพยากรณ์ก่อน ยืนยันการลบข้อมูล โดยกำหนดชื่อของ popup คือ id="deletedata{{ item.indexpredict }}" และปุ่มสำหรับยืนยันการลบข้อมูลในส่วนของ href ให้ระบุเป็น {% url 'backenddeleteprediction' item.uuidpredict %} โดยที่ url 'deleteuser'

คือชื่อของเส้นทาง url ที่เรียกใช้งาน และ item.uuidpredict คือข้อมูลที่ต้องการส่งไปพร้อมกับ url เส้นนี้

```
Python
<div class="modal fade" id="deletedata{{ item.indexpredict }}" tabindex="-1"
aria-labelledby="exampleModalLabel" aria-hidden="true">
                  <div class="modal-dialog">
                    <div class="modal-content">
                      <div class="modal-header">
                        <h1 class="modal-title fs-3">ยืนยันการลบข้อมูล</h1>
                        <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Close"></button>
                      </div>
                      <div class="modal-body text-dark">
                        รายการ {{ item.datetimepre|date:'d-m-Y H:i:s' }} <br>
                        ผู้ดำเนินการ {{ item.refuser.first_name }} {{
item.refuser.last_name }} <br>
                        รูปแบบ {{ item.typepredict }} <br><br>
                        <img src="{% static '/fileupload' %}/{{ item.imagefile
}}" class="img-fluid rounded"/>
                        <br><br>
                        <a class="btn btn-primary" href="{% url
'backenddeleteuser' item.uuidpredict %}" role="button">ยืนยันการลบข้อมูล</a>
                        <button type="button" class="btn btn-secondary" data-bs
-dismiss="modal">ยกเลิกการลบ</button>
                      </div>
                    </div>
                  </div>
</div>
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทางสำหรับการเรียกใช้งานฟังก์ชั่นการลบข้อมูลการพยากรณ์คือ backend/deleteprediction/<str:uuidprediction> และกำหนดการรับค่า uuidprediction โดยกำหนดชนิดของตัวแปรเป็น str คือ string ตัวอักษร และเรียกใช้งานฟังก์ชั่นที่ชื่อว่า backenddeleteprediction กำหนดชื่อเรียกของเส้นทางนี้คือ backenddeleteprediction (ซึ่งจะเป็นชื่อเรียกใน tag <a> ที่เรียกใช้งานผ่านหน้า html)

```
Python
path('backend/deleteprediction/<str:uuidprediction>/',
views.backenddeleteprediction, name='backenddeleteprediction')
```

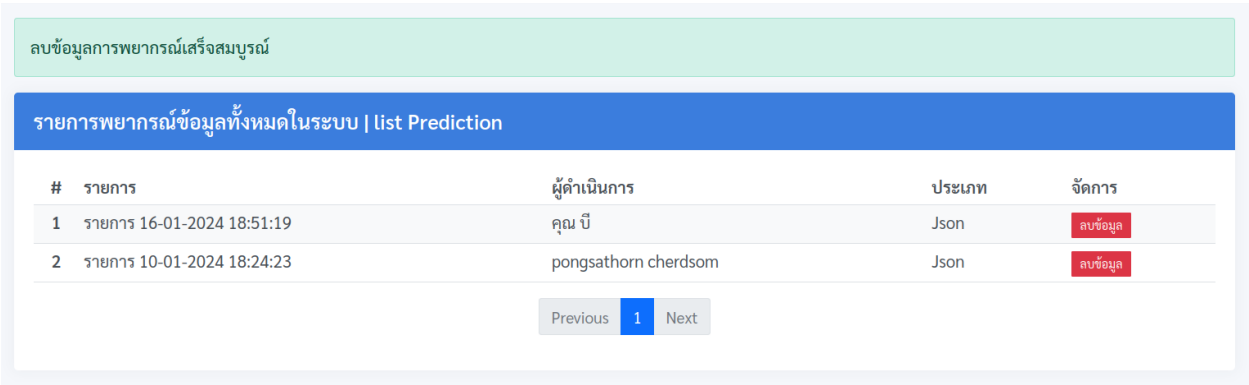
File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่นที่ชื่อว่า backenddeleteprediction และรับค่าตัวแปร uuidprediction ชื่อเดียวกับที่สร้างใน url จะได้เป็น def backenddeleteprediction (request,uuidprediction) สำหรับการลบข้อมูลการพยากรณ์ และลบไฟล์ที่อัพโหลดขึ้นไปบนระบบ
โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_staff ว่าผู้ใช้งานที่เรียกใช้งานฟังก์นี้เป็นผู้ดูแลระบบหรือไม่ หากไม่ใช่ ให้ redirect กลับไปที่หน้า backendpage และ alert message รูปแบบ warning พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น”
- เช็คเงื่อนไขการจากตัวแปร deleteitem ว่ามีข้อมูลจากการ query หรือไม่ หากมีให้ดำเนินการถัดไป หากไม่มีส่งข้อมูลกลับไปที่ฟังก์ชั่น backendlistprediction พร้อมกับข้อความที่ “ไม่พบข้อมูลการพยากรณ์ โปรดลองใหม่อีกครั้ง” และ alert message รูปแบบ error
- ทำการเช็ค if deleteitem.typepredict == "Image": ในกรณีที่ส่งข้อมูลไปประมวลผลในรูปแบบ Image ระบบที่พัฒนาจะเก็บผลการพยากรณ์ในรูปแบบของไฟล์รูปภาพจาก roboflow จึงต้องเช็คเพื่อลบรูปภาพจากการพยากรณ์ด้วย และใช้คำสั่งในการลบไฟล์คือ os.remove(f"{pathfileupload}/{deleteitem.resultpredict}")
- ลบไฟล์ต้นฉบับที่ส่งไปประมวลผลที่ roboflow โดยใช้คำสั่ง os.remove(f"{pathfileupload}/{deleteitem.imagefile}")
- จากนั้นใช้คำสั่ง deleteitem.delete() เพื่อทำการลบข้อมูลออกจากฐานข้อมูล และ redirect กลับไปที่ url name resultpredictionpage พร้อมทั้งแสดงข้อความ “ลบข้อมูลการประมวลผลรูปภาพเสร็จสมบูรณ์” และ alert message รูปแบบ success

ทดสอบ Runserver

python3 manage.py runserver
http://127.0.0.1:8000/backend/listprediction/



36. การสร้างหน้าสำหรับแสดงผลการ Prediction (สำหรับผู้ดูแลระบบ)

Template : listprediction.html

ทำงานที่ไฟล์ listprediction.html ในส่วนของตารางแสดงรายการข้อมูล คอลัมน์รายการ โดยใช้ tag <a> เพื่อสร้างลิงก์ไปยังฟังก์ชั่นสำหรับการแสดงผลการพยากรณ์ข้อมูล โดยส่งไปที่ url name backendpredictioninfo และแนบค่า item.uuidpredict ไปด้วย
โดยมีลักษณะคล้ายกับเรื่อง **การสร้างหน้าสำหรับแสดงผลการ Prediction (รายการที่เลือก)**

Python
```
<a href="{% url 'backendpredictioninfo' item.uuidpredict %}">รายการ {{
item.datetimepre|date:'d-m-Y H:i:s' }}</a>
```

File : urls.py (อยู่ใน webai)

สร้างเส้นทาง กำหนดให้เป็น backend/predictioninfo/<str:uuidprediction>/ เรียกใช้งานฟังก์ชั่น backendpredictioninfo และกำหนดชื่อเส้นทางนี้เป็น backendpredictioninfo เพื่อให้สอดคล้องกับหน้า html ที่ได้กำหนดไว้ใน tag <a>ขั้นตอนก่อนหน้านี้

```python
path('backend/predictioninfo/<str:uuidprediction>/',
views.backendpredictioninfo, name='backendpredictioninfo'),
```

File : views.py (อยู่ใน backendai)

สร้างฟังก์ชั่นชื่อ backendpredictioninfo เพื่อนำมาแสดงผลของการประมวลผลที่ได้จาก roboflow ซึ่งในขั้นตอนก่อนหน้ามีรูปแบบผลลัพธ์การประมวลผล 2 แบบคือ json และ image โดยมีลักษณะคล้ายกับเรื่อง **การสร้างหน้าสำหรับแสดงผลการ Prediction (รายการที่เลือก)** โดยที่

- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_authenticated หากยังไม่มีการ login ให้ redirect กลับไปที่หน้า homepage และ alert message รูปแบบ error พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ โปรดลงชื่อเข้าใช้งานหรือสมัครสมาชิกก่อน”
- ตรวจสอบการเรียกใช้งานฟังก์ชั่น โดยใช้ request.user.is_staff ว่าผู้ใช้งานที่เรียกใช้งานฟังก์นี้เป็นผู้ดูแลระบบหรือไม่ หากไม่ใช่ ให้ redirect กลับไปที่หน้า backendpage และ alert message รูปแบบ warning พร้อมกับข้อความ “ไม่สามารถเข้าใช้งานได้ สำหรับผู้ดูแลระบบเท่านั้น”
- ค้นหา ข้อมูลตัวแปร uuidprediction ที่ส่งมา จากฐานข้อมูลตาราง Predictprocess ที่เก็บผลการพยากรณ์ข้อมูล ว่ามีข้อมูลนี้ในฐานข้อมูลหรือไม่
- โดยการ query ข้อมูลแบบ get และมีการใช้งาน try: except เพื่อดักค่า error กรณีที่มีข้อมูลจากการ get มากกว่า 1 ข้อมูล
- สร้างตัวแปรเพื่อเป็นค่าตั้งต้นได้แก่
- countitem = 0 จำนวนวัตถุที่พยากรณ์ได้
- listitems = [] รายการวัตถุทั้งหมดที่พยากรณ์ได้
- imagepredict = None รูปภาพผลการพยากรณ์ ที่เขียนเส้นกรอบวัตถุ ชื่อคลาส และค่าความแม่นยำ
- เช็คเงื่อนไขเงื่อนไขเดียวคือ รูปแบบของการแสดงผล (คอลัมน์ typepredict) แบบ Json หรือไม่ หากใช่จะมีการดำเนินการอย่างอื่นต่อ ซึ่งแตกต่างจากรูปแบบ Image ที่นำรูปไปแสดงผลได้เพียงอย่างเดียว
- การดำเนินการภายในเงื่อนไข typepredict == "Json" จะมีการดำเนินการ
- แปลงข้อมูลจากข้อความ ที่เก็บอยู่ในคอลัมน์ resultpredict ให้เป็นตัวแปรแบบ dict() โดยใช้คำสั่ง ast.literal_eval(ตัวแปรข้อมูล)

- นับจำนวนวัตถุที่สามารถพยากรณ์ได้ จากข้อมูลที่ชื่อว่า “predictions” ที่ได้จากการพยากรณ์ของ roboflows โดยใช้คำสั่ง len()
- สร้างตัวแปรเป็น path ของระบบ สำหรับเรียกไฟล์ที่อยู่ภายในเครื่อง
- อ่านไฟล์รูปภาพที่อยู่ภายในเครื่อง โดยใช้คำสั่ง Image.open()
- ใช้คำสั่ง copy() รูปภาพเพื่อนำไปใช้ในส่วนของการเขียนเส้น (ตีกรอบวัตถุ) และผลที่ได้จากการพยากรณ์
- ทำการลูปข้อมูลที่อยู่ใน “predictions” เพื่อนำมาเขียนเส้นของวัตถุ โดยใช้คำสั่ง for — in —
- ภายในลูป จะทำการคำนวนตำแหน่งของการเขียนข้อความและเส้นกรอบ
- กำหนดข้อความที่จะแสดง และขนาดของตัวอักษร
- สร้างตัวแปรเพื่อเริ่มต้นการเขียนข้อความและเส้นกรอบวัตถุ
- เขียนเส้นกรอบวัตถุสีน้ำเงิน และเขียนข้อความสีขาว พร้อมด้วยพื้นหลังสีน้ำเงิน
- นำตำแหน่งที่พยากรณ์ได้ มาตัดรูปภาพเฉพาะส่วนที่มีการพยากรณ์ถูกต้อง นำมาสร้างเป็นรูปภาพรูปแบบ base64
- เก็บข้อมูลการลูปลงในตัวแปร datapre ชนิด dict() เพื่อเก็บข้อมูลและนำไปแสดงผล เพื่อให้เกิดความสะดวกในการเรียกใช้งาน ซึ่งจะเก็บข้อมูลได้แก่ imagecrop_base64 คือส่วนของรูปภาพที่มีการตัดเฉพาะส่วนที่พยากรณ์ถูกต้อง, class คือชื่อของวัตถุที่พยากรณ์ , confidence คือ ค่าความแม่นยำถูกต้องจากการพยากรณ์ (ของแต่ละลูป)
- และนำข้อมูลที่ได้จากตัวแปร datapre มาเก็บเข้า ตัวแปร listitems ชนิด list() ที่จำเก็บข้อมูลทั้งหมดภายในลูป ซึ่งจะนำปแสดงผลในส่วนของ รายการการพยากรณ์
- สร้างรูปภาพที่นำเส้นกรอบของววัตถุทั้งหมดที่ได้จากการพยากรณ์นำมาเขียนไว้ในภาพ (ทุกวัตถุ) ในรูปแบบ base64 หนึ่งภาพ ในตัวแปร imagepredict
- ส่งข้อมูลจากการ query ไปแสดงในหน้า predictioninfo.html โดยแนบไปในตัวแปร context ซึ่งมีข้อมูล context = { 'datapredict': getinfopro , 'countitem':countitem, 'listitems' : listitems , 'imagepredict' : imagepredict } และข้อมูลอื่นๆทั้งหมดในฟังก์ชั่นแนบไปด้วย (ในตัวอย่างนี้ได้ใช้ชื่อตัวแปรซ้ายมือ และตัวแปรขวามือไม่ตรงกันเป็นตัวอย่างในการนำไปใช้งาน)

Template : predictioninfo.html

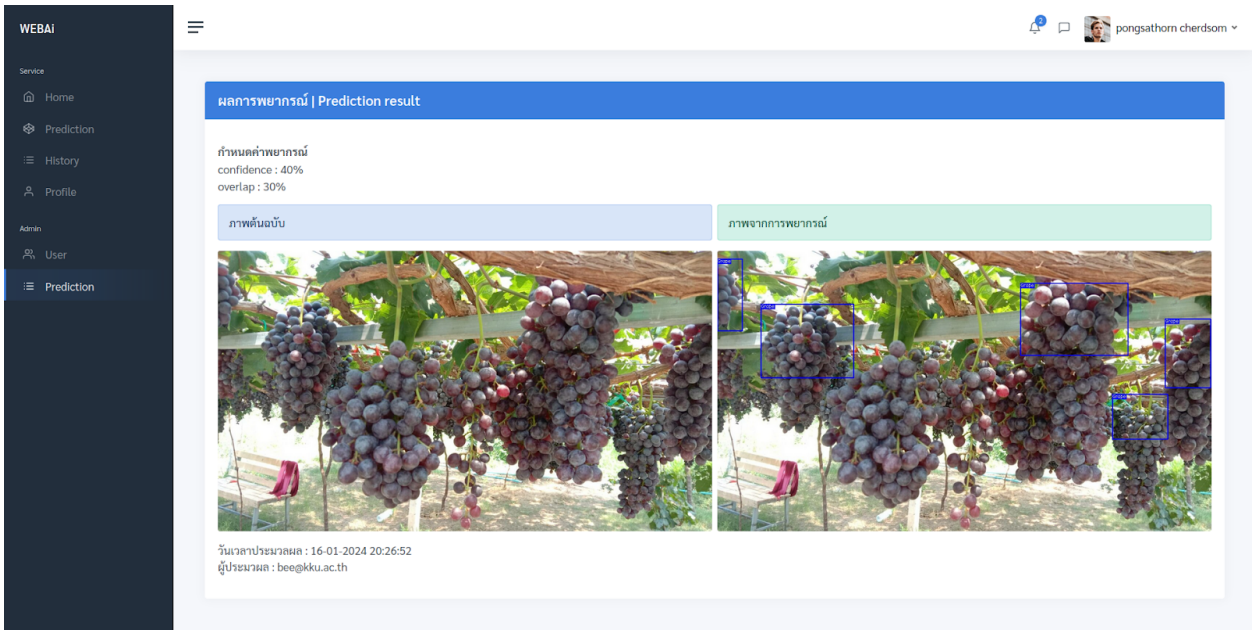
- สร้างไฟล์ predictioninfo.html ซึ่งมีโครงสร้างเหมือนกับ สามารถคัดลอก backendpage.html มาใช้งานได้
- เรียกใช้ {% load static %} ไว้บรรทัดล่างของ {% extends 'base.html' %} ของไฟล์ resultinfo.html เพื่อให้สามารถเรียกใช้ไฟล์รูปภาพจากระบบได้
- รูปแบบการแสดงผล โดยใช้ if elif else ในการตรวจสอบรูปแบบการแสดงผล
    - 1 แสดงรูปภาพจากการพยากรณ์
    - 2 แสดงข้อมูลจากข้อมูล json
    - 3 ไม่เข้าเงื่อนไขใดๆ แสดงข้อความผิดพลาด

```
Python
{% if datapredict.typepredict == "Image" %}
      << แสดงผลสำหรับรูปแบบรูปภาพ
{% elif datapredict.typepredict == "Json" %}
      << แสดงผลสำหรับรูปแบบรูปภาพ
{% else %}
      << แสดงผลสำหรับไม่เข้าเงื่อนไขทั้งสองด้านบน (แสดงข้อความผิดพลาด)
{% endif %}
```

- ภายในเงื่อนไข Image
  - ทำการแสดงผลรูปภาพที่นำมาใช้ในการพยากรณ์ (ต้นฉบับ) และภาพที่ได้จากการพยากรณ์ วันที่และเวลาทำรายการ ผู้ทำรายกากร โดยแบ่งหน้าจอให้เป็น 2 คอลัมน์ ซึ่งในการแสดงผลแบบ image นี้จะไม่สามารถแสดงรายละเอียดอื่นๆได้ นอกจากรูปภาพจากการพยากรณ์
  - ในส่วนของวันที่และเวลา สามารถใช้ filter Date โดยเขียน |date:'d-m-Y H:i:s' คือรูปแบบของวันที่ที่ต้องการให้แสดงผลดังตัวอย่างจะแสดง 11-12-2023 18:24:47 ศึกษาเพิ่มเติมได้ที่ https://docs.djangoproject.com/en/5.0/ref/templates/builtins/#date
  - ในส่วนของการแสดงข้อมูลผู้ใช้งาน สามารถเชื่อมต่อตารางได้ดังภาพ

```
Python
<div class="row g-2">
        <div class="col-md-6">
            <div class="alert alert-primary h4" role="alert">
                ภาพต้นฉบับ
              </div>
            <img src="{% static 'fileupload' %}/{{ datapredict.imagefile }}"
class="img-fluid rounded" />
        </div>
        <div class="col-md-6">
            <div class="alert alert-success h4" role="alert">
                ภาพจากการพยากรณ์
            </div>
            <img src="{% static 'fileupload' %}/{{ datapredict.resultpredict
}}" class="img-fluid rounded" />
        </div>
</div>

<p class="mt-3 fs-5">
        วันเวลาประมวลผล : {{ datapredict.datetimepre|date:'d-m-Y H:i:s' }} <br>
```

```
        ผู้ประมวผล : {{ datapredict.refuser.username }}
</p>
```

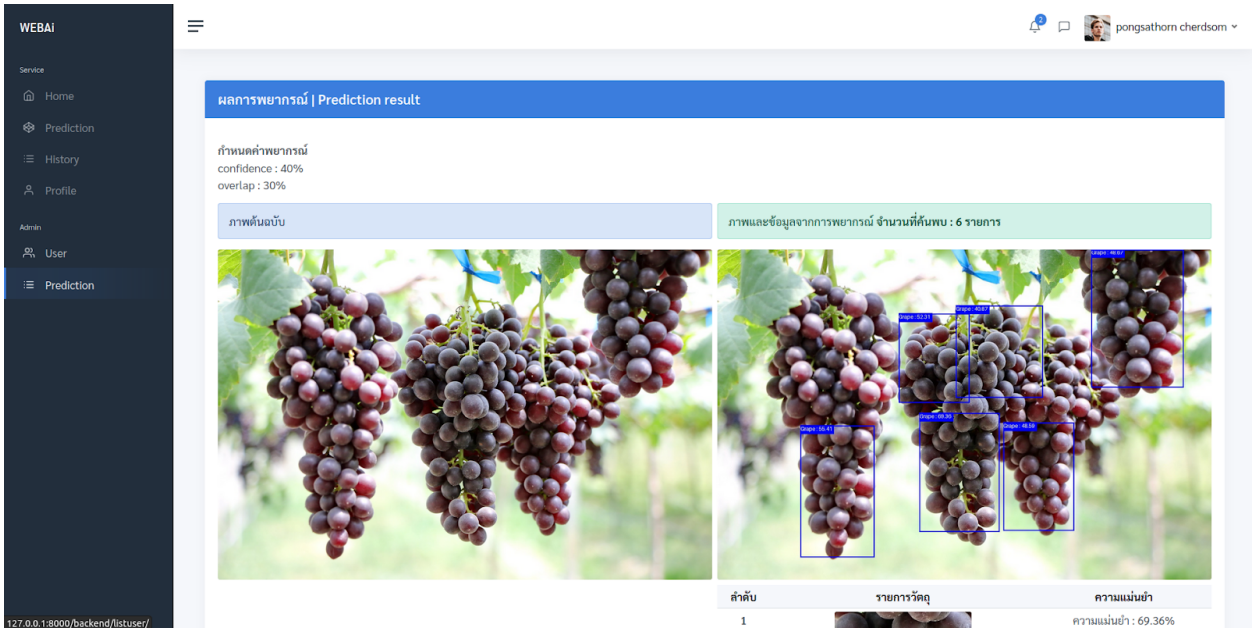


- ภายในเงื่อนไข Json
    - จะแสดงภาพต้นฉบับ จำนวนองุ่นที่สามารถตรวจพบทั้งหมด รายการองุ่นที่ตรวจสอบภาพและความแม่นยำที่ได้ แสดงภาพผลลัพธ์จาก json ที่นำมาประมวลผลเอง (วาดเส้น ระบุชื่อคลาส และค่าความแม่นยำ) รวมถึงวันเวลาประมวลผล และชื่อผู้ทำรายการ

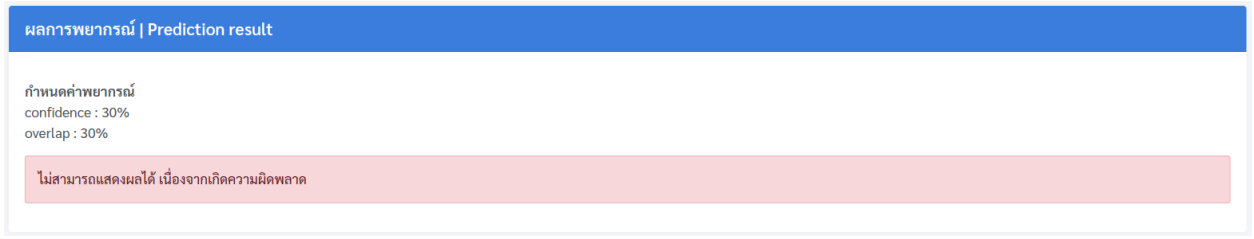
Python

```
<div class="row g-2">
        <div class="col-md-6">
            <div class="alert alert-primary h4" role="alert">
                ภาพต้นฉบับ
              </div>
            <img src="{% static 'fileupload' %}/{{ datapredict.imagefile }}"
class="img-fluid rounded" />
        </div>
        <div class="col-md-6">
            <div class="alert alert-success h4" role="alert">
                ภาพและข้อมูลจากการพยากรณ์ <span class="fw-bold"> จำนวนที่ค้นพบ : {{
countitem }} รายการ </span>
            </div>
```

```
            <img src="{{ imagepredict }}" class="img-fluid rounded" /><br>

            <table class="table table-sm text-center mt-2">
                <thead class="table-light">
                  <tr>
                    <th scope="col-2">ลำดับ</th>
                    <th scope="col-8">รายการวัตถุ</th>
                    <th scope="col-2">ความแม่นยำ</th>
                  </tr>
                </thead>
                <tbody>
                    {% for item in listitems %}
                        <tr>
                            <th scope="row">{{ forloop.counter }}</th>
                            <td>
                                <img src="{{ item.imagecrop_base64 }}"
class="img-fluid rounded" /><br>
                            </td>
                            <td class="align-top">
                                <span>ความแม่นยำ : {{ item.confidence }}%</span>
                            </td>
                      </tr>
                    {% endfor %}
                </tbody>
            </table>

        </div>
    </div>

    <p class="mt-3">
        วันเวลาประมวลผล : {{ datapredict.datetimepre|date:'d-m-Y H:i:s' }} <br>
        ผู้ประมวผล : {{ datapredict.refuser.username }}
    </p>
```



- ไม่เข้าเงื่อนไขทั้ง image และ Json เพื่อป้องกันความผิดพลาด โดยให้แสดงข้อความ “ไม่สามารถแสดงผลได้ เนื่องจากเกิดความผิดพลาด”




```python
<div class="alert alert-danger" role="alert">
            ไม่สามารถแสดงผลได้ เนื่องจากเกิดความผิดพลาด
 </div>
```

ทดสอบ Runserver

python3 manage.py runserver

http://127.0.0.1:8000/backend/listprediction/

37. การเตรียมโปรเจคก่อนนำไปติดตั้ง

การเตรียมความพร้อมของโปรเจคก่อนนำไปติดตั้งเพื่อใช้งานจริง เมื่อเสร็จสิ้นการพัฒนาระบบและพร้อมนำไปใช้งานจริงแล้ว มีสิ่งที่ต้องจัดเตรียมเพื่อให้การติดตั้งบน server จริงทำได้ง่ายและสะดวกมากยิ่งขึ้น ซึ่งมีขั้นตอนทั้งหมด 2 ขั้นตอน

1. ส่งออกรายการ Libraries ทั้งหมดที่ใช้งานการพัฒนาระบบ เพื่อนำไปติดตั้งใน server จริง หรือสามารถนำ Folder Virtual Environments ที่สร้างไว้ในขั้นตอนแรกๆ ไปใช้งานได้เช่นกัน (ในตัวอย่างนี้จะใช้วิธีแรกคือการส่งออกรายชื่อ Libraries)
2. ปรับแต่งค่าฐานข้อมูลสำหรับใช้งานจริง
3. ปรับแต่งไฟล์ settings.py ในส่วนอื่นๆ
4. รวมไฟล์ทั้งหมดของโปรเจคให้อยู่ในไฟล์ .zip

ส่งออกรายการ Libraries (Export lib.)

1. เข้าไปที่ Folder ของโปรเจค
2. เรียกใช้งาน Environments ที่ใช้พัฒนาระบบ (Activate Virtual Environments) เหมือนกับขั้นตอนในหัวข้อที่ 1
    - คำสั่งสำหรับ ubuntu

```
source env/bin/activate
```

    - คำสั่งสำหรับ windows

```
cd <folderenv> (เข้าไปที่ Folder ของ Virtual Environments ที่สร้างไว้)
Scripts\activate
```

3. ส่งออกรายการ Libraries (Export Libraries)

```
pip3 freeze > requirements.txt

Output:
requirements.txt

asgiref==3.7.2
certifi==2023.7.22
chardet==4.0.0
charset-normalizer==3.3.2
contourpy==1.2.0
cycler==0.10.0
Django==4.2.7
fonttools==4.46.0
idna==2.10
kiwisolver==1.4.5
matplotlib==3.8.2
mysqlclient==2.2.0
numpy==1.26.2
opencv-python-headless==4.8.0.74
packaging==23.2
Pillow==10.1.0
pyparsing==2.4.7
```

```
python-dateutil==2.8.2
python-dotenv==1.0.0
python-magic==0.4.27
PyYAML==6.0.1
requests==2.31.0
requests-toolbelt==1.0.0
roboflow==1.1.12
scipy==1.11.4
six==1.16.0
sqlparse==0.4.4
supervision==0.17.1
tqdm==4.66.1
typing_extensions==4.8.0
urllib3==2.1.0
```

ปรับแต่งค่าฐานข้อมูลสำหรับใช้งานจริง

ปรับแต่งค่าของฐานข้อมูลให้ตรงกับฐานข้อมูลที่จะนำมาใช้งานจริง ซึ่งจะอยู่ใน Folder webai -> settings.py ในหัวข้อ DATABASES โดยกำหนดให้ครบถ้วนและถูกต้อง ได้แก่

- ชื่อฐานข้อมูล (NAME)
- ชื่อผู้ใช้งาน (USER)
- รหัสผ่าน (PASSWORD)
- ไอพีของเครื่อง (HOST)
- พอร์ตของฐานข้อมูล โดยปกติจะเป็นพอร์ต 3306 (PORT)

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "xxxxxxx",
        "USER": "xxxxxxx",
        "PASSWORD": ".....",
        "HOST": "x.x.x.x",
        "PORT": "3306",
    }
}
```

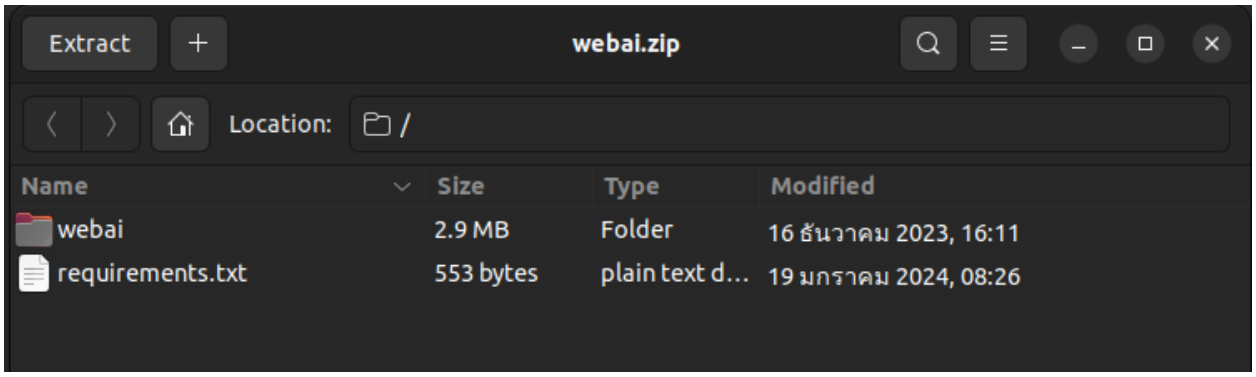
ปรับแต่งไฟล์ Settings.py ในส่วนต่าง ๆ

- การ ALLOWED HOSTS สำหรับการกำหนดเส้นทางในการเรียกใช้งานระบบ ให้กำหนดใน Folder webai -> settings.py ในหัวข้อ ALLOWED_HOSTS = ['*'] หมายถึงสามารถเรียกใช้งานได้ทุกเส้นทาง เช่น ผ่าน ip address ของเครื่อง ผ่าน Domain name ของเครื่อง หากมีการเรียกใช้งานผ่าน Domain name ให้เรียกใช้งานผ่าน Domain name เท่านั้น ALLOWED_HOSTS = ['webai.pongthorn.in.th'] เป็นต้น

- การ DEBUG ให้กำหนดค่าเป็น False หมายถึงปิดการแสดง error หรือความผิดพลาดของระบบ ในระหว่างการตั้งค่า การติดตั้งควรกำหนดให้เป็น True เพื่อให้แสดงความผิดพลาด หากทำงานบน prodction จริงให้กำหนดเป็น False เสมอ
- เพิ่ม STATIC_ROOT = [BASE_DIR/'static'] ใต้บรรทัดของ STATICFILES_DIRS เพื่อเป็นการเรียกใช้งานไฟล์ได้อย่างถูกต้อง

รวมไฟล์ทั้งหมดของโปรเจคให้อยู่ในไฟล์ .zip

ในการรวมไฟล์จะรวมไฟล์โปรเจคที่อยู่ใน Folder webai เท่านั้น จะไม่นำ Folder env มาด้วย (Virtual Environments) และไฟล์ requirements.txt และบีบอัดให้เป็นไฟล์ .zip เพื่อสะดวกในการ upload และแตกไฟล์ฝั่งของ ubuntu server จะได้ดังภาพ



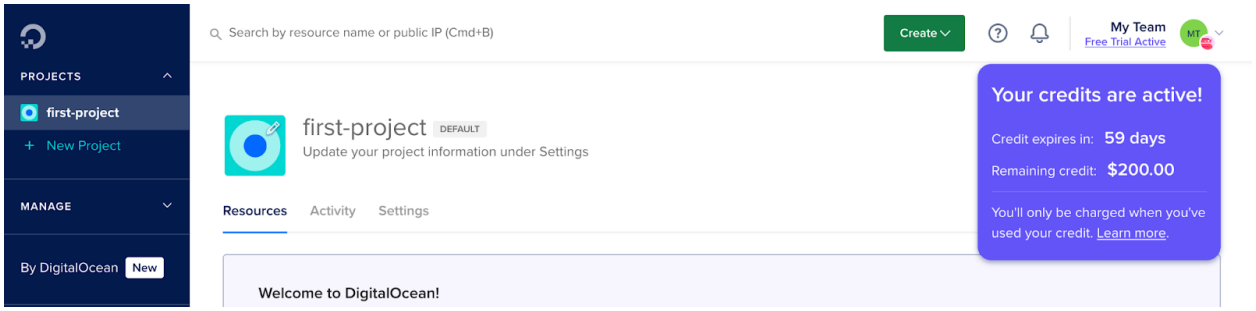
38. การติดตั้งระบบเพื่อใช้งานจริง (Deploy)

การติดตั้งเพื่อใช้งานจริงใน workshop นี้จะใช้บริการ cloud จาก Digitalocean โดยใช้ ubuntu server 22.04 การนำไปใช้งานจริงสามารถใช้บริการจากเจ้าอื่น ๆ ได้

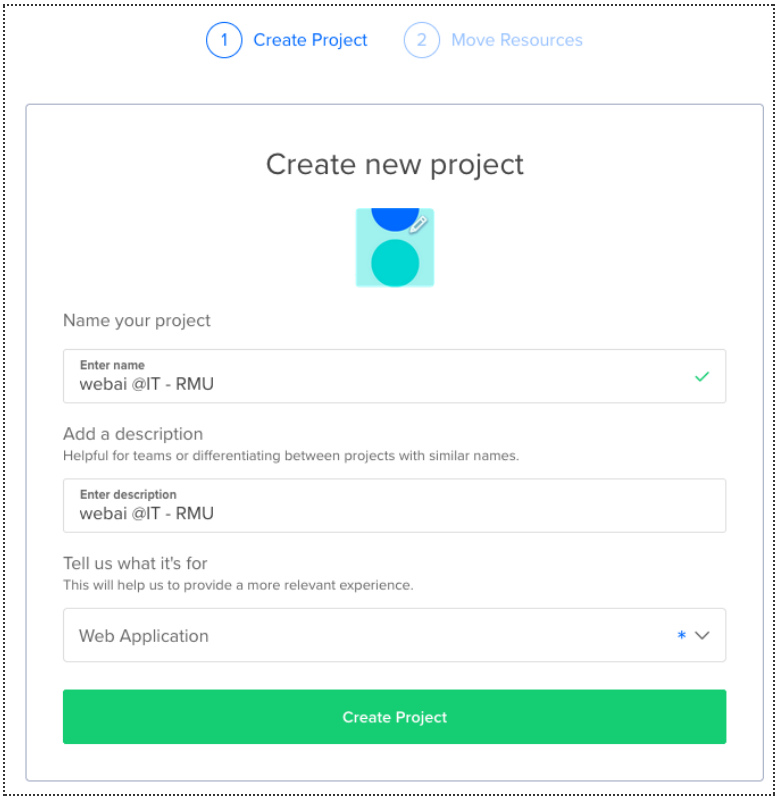
สร้าง Droplets (หรือสร้างเครื่อง VM)

หลังจากที่สมัครใช้งานบริการของ Digitalocean แล้ว

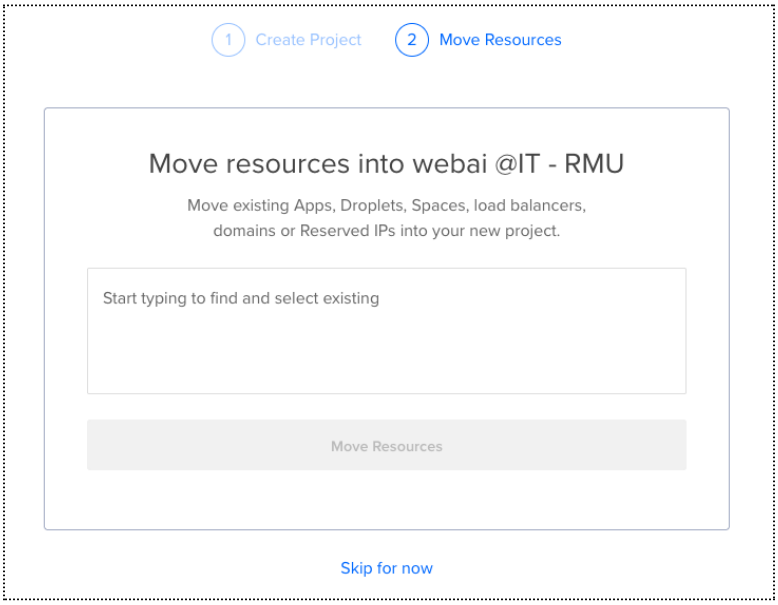
1. ให้ทำการสร้างกลุ่มให้กับ Droplets นี้ โดยคลิกที่ **+ New Project** ด้านซ้ายมือ



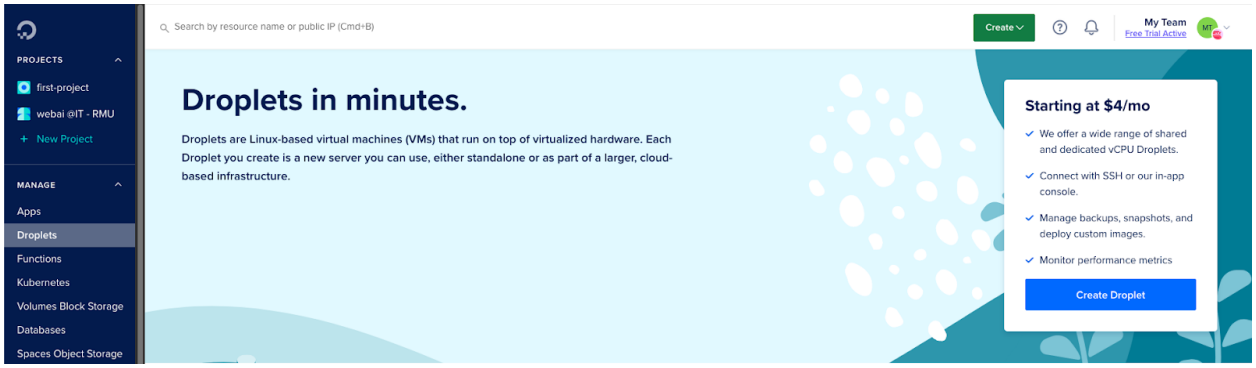
2. กำหนดชื่อโปรเจค (ช่องที่หนึ่ง) รายละเอียดของโปรเจค (ช่องที่สอง) และสำหรับวัตถุประสงค์ของการใช้งานโปรเจคนี้ (ช่องที่สาม)



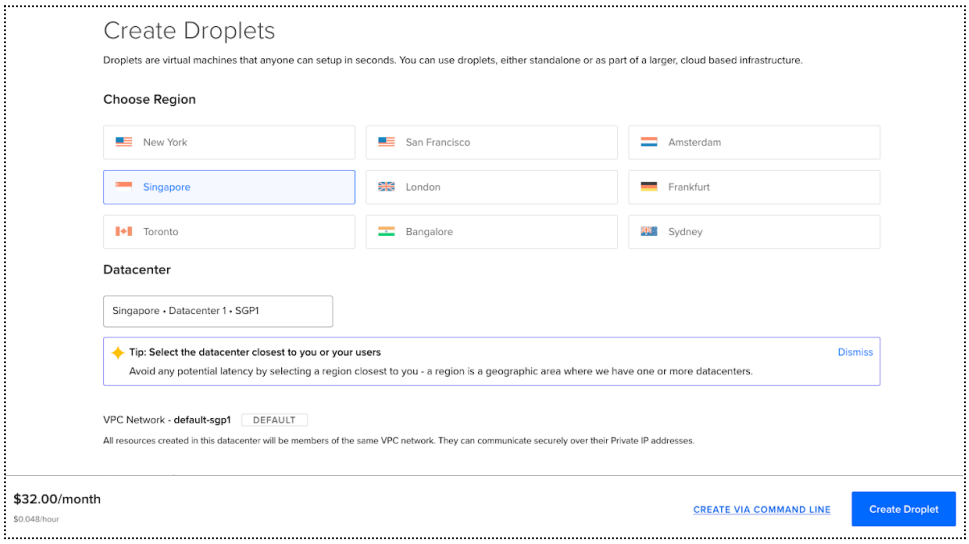
3. ส่วนของ Move Resources ให้คลิกที่ **Skop for now** ได้เลย



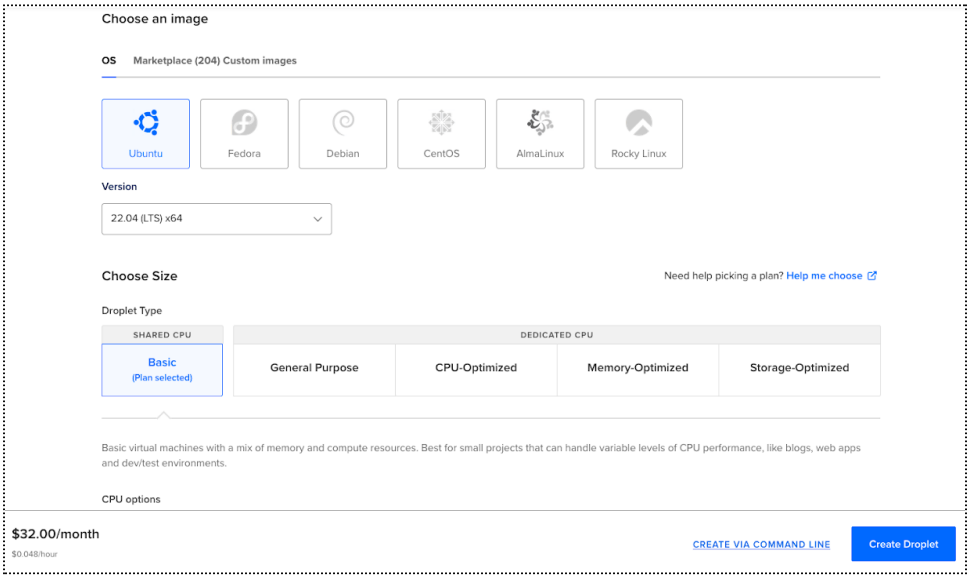
4. จากนั้นคลิกที่**เมนู Droplets** ในกลุ่มของ MANAGE ด้านขวามือ
5. คลิกที่ **Create Droplets** ด้านซ้ายมือ



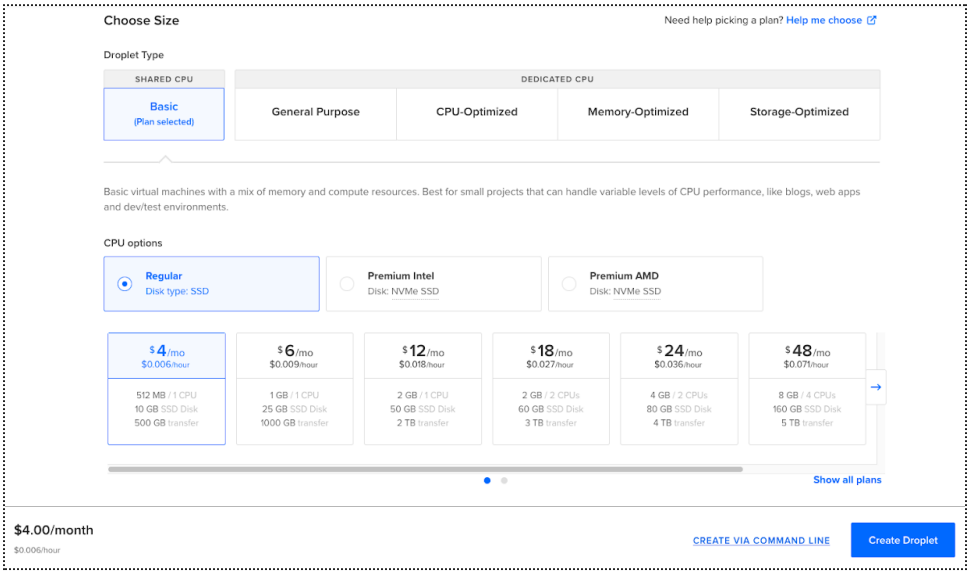
6. เลือก Region โซนของ Droplets ที่จะสร้างขึ้น ให้เลือกเป็น Singapore (เป็นประเทศที่ใกล้ประเทศไทยมากที่สุด)



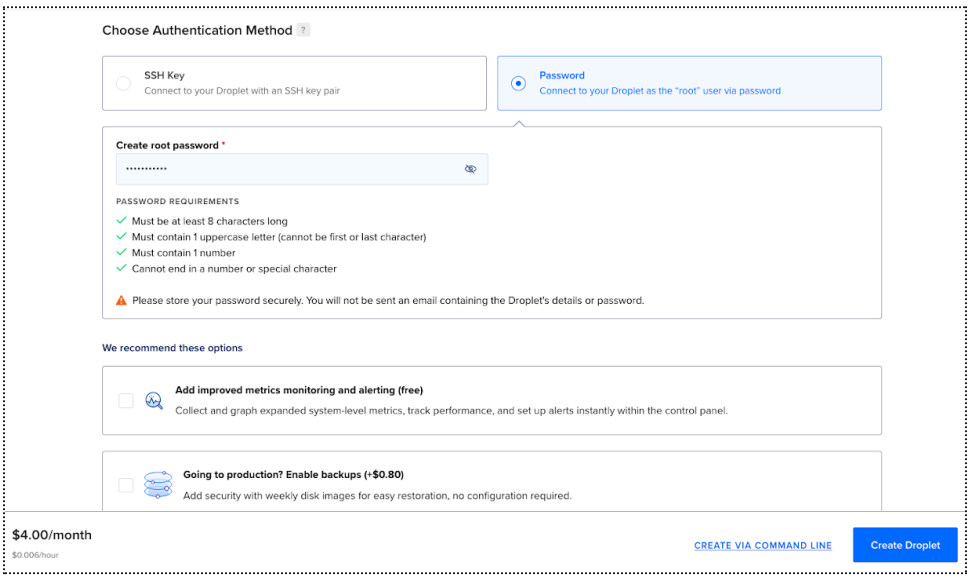
7. เลือก OS ระบบปฏิบัติการในหัวข้อ **Choose an image** กำหนดให้เป็น ubuntu Version 22.04 LTS X64



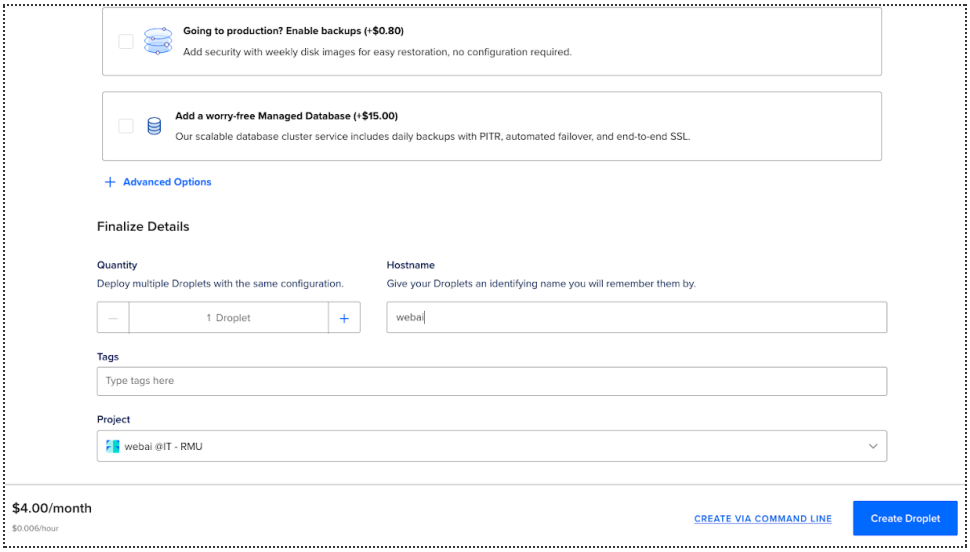
8. เลือกของ Droplets ประเภท Basic ให้เป็น **Choose Size (Droplet Type)** และ **CPU options** ให้เลือกเป็น **Regular Disk type: SSD** และขนาดของเครื่องตามที่ต้องการซึ่งใน workshop นี้ได้ใช้เป็นเครื่องที่มีขนาดเล็กที่สุดคือ $4 ต่อเดือน สเปคการเครื่องคือ Ram : 512MB, Vcpu : 1 core, SSD : 10 GB, Transfer : 500GB ต่อเดือน และสามารถ resize ได้ตามที่ต้องการ



9. กำหนดรหัสผ่านของเครื่อง ในหัวข้อ **Choose Authentication Method** ให้เป็นแบบ **Password** และทำการกำหนดรหัสผ่านตามเงื่อนไขที่กำหนดไว้



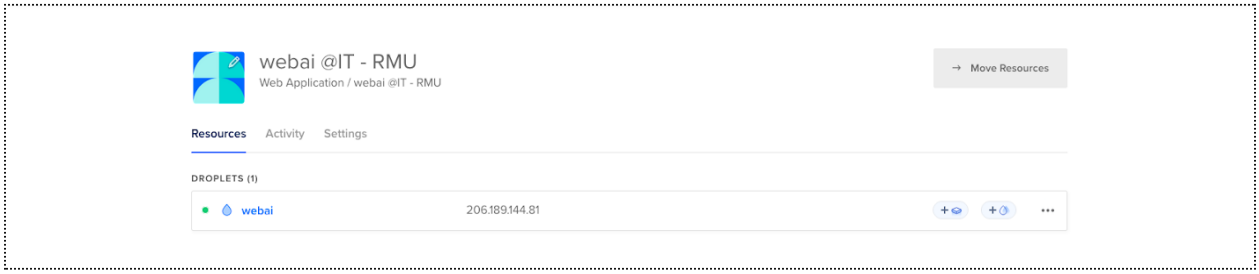
10. กำหนดชื่อเครื่อง Hostname ในหัวข้อ **Finalize Details** ให้เป็น **webai** และโปรเจคให้เลือกเป็นขื่อโปรเจคที่ได้สร้างไว้ในตอนต้นคือ **webai @IT - RMU**



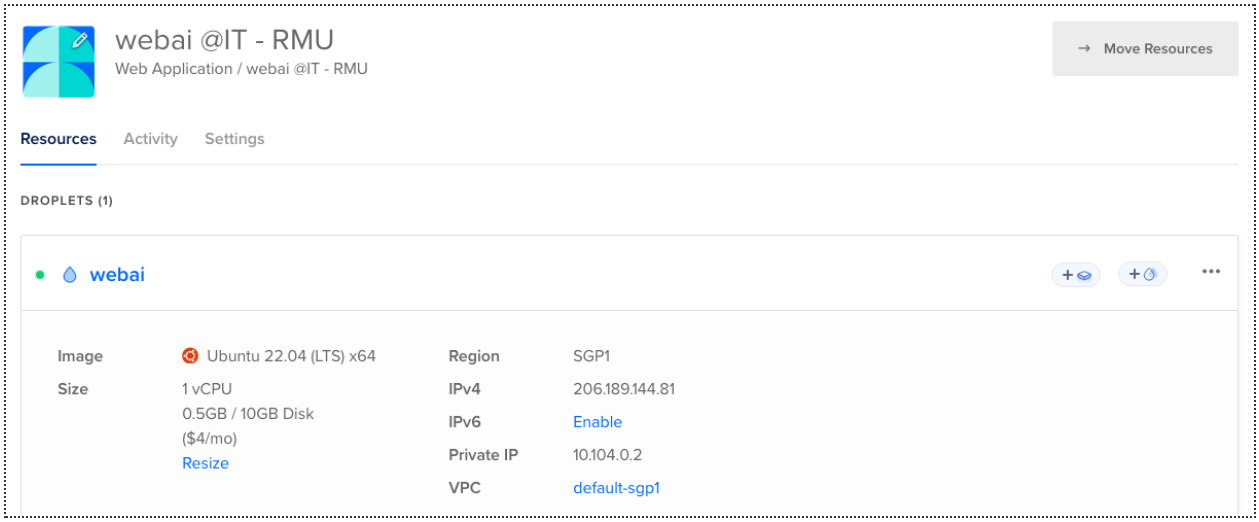
11. เมื่อสร้างเสร็จสมบูรณ์ จะแสดงดังภาพ โดยจะแสดงชื่อเครื่อง (Hostname) IP address ของเครื่อง

    Hostname : webai

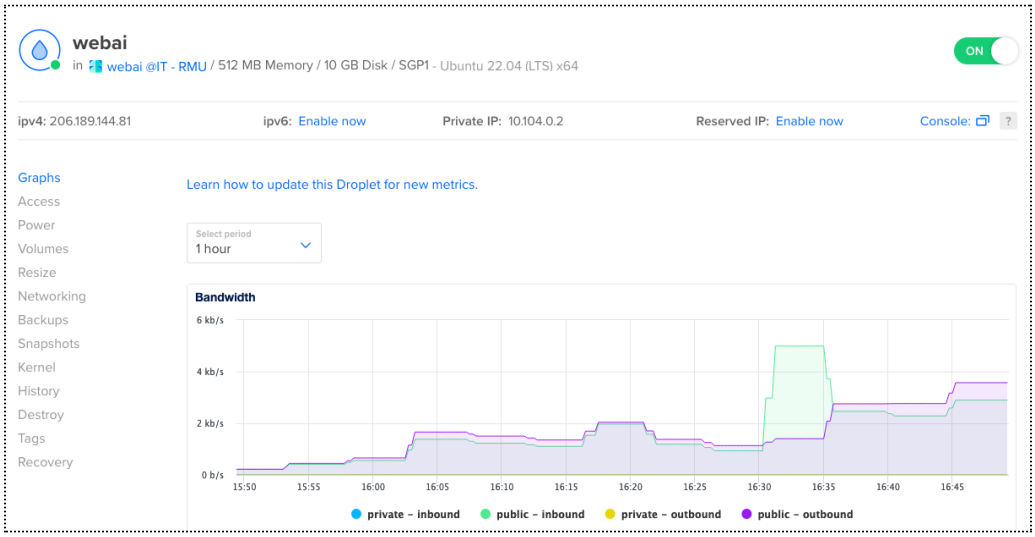
    Ip address : 206.189.144.81



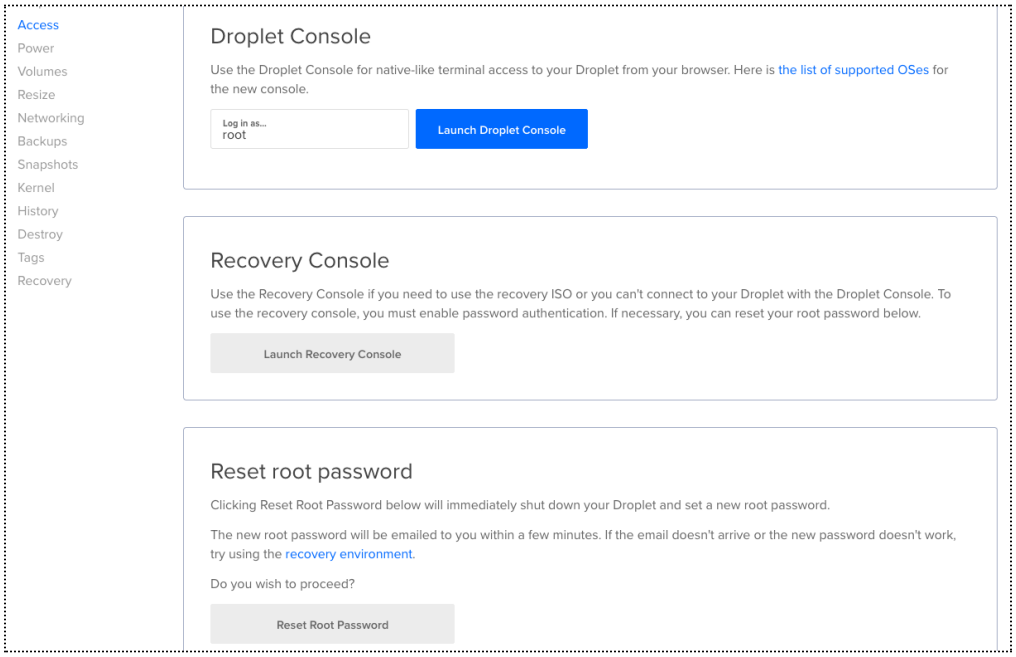
เมื่อคลิกที่เครื่องจะแสดงดังภาพ โดยจะแสดงรายละเอียดของเครื่องเพิ่มเติม



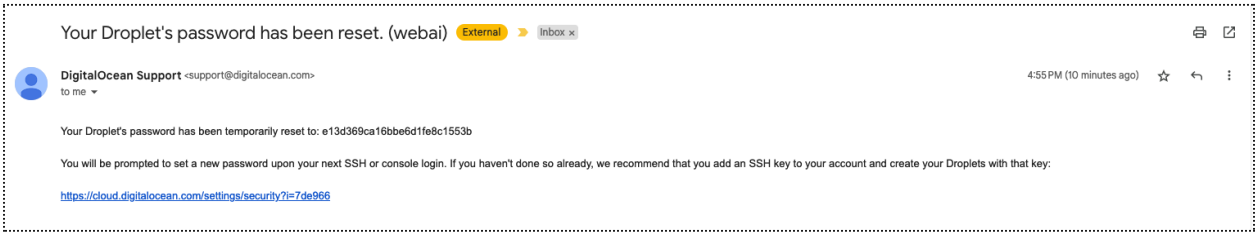
12. เมื่อคลิกที่ชื่อเครื่อง จะเป็นจัดการเครื่อง (manage) และส่วนของการติดตาม (monitoring) การทำงานของเครื่อง เช่น กราฟการใช้งาน Bandwidth กราฟการใช้งาน CPU หรือ การเขียนการอ่าน Disk เป็นการ ซึ่งจะไม่ลงรายละเอียดในหัวข้อนี้



13. กรณีต้องการรีเซ็ตรหัสผ่านของเครื่อง ให้คลิกที่ชื่อเครื่องที่ต้องการเปลี่ยน (เหมือนกับข้อที่ 12) เลือกที่เมนู Access และในส่วนของหัวข้อ **Reset root password** ให้คลิกที่ปุ่ม **Reset Root Password** **เมื่อทำการ reset เสร็จสมบูรณ์ระบบจะแจ้งรหัสผ่านใหม่ไปที่อีเมล**

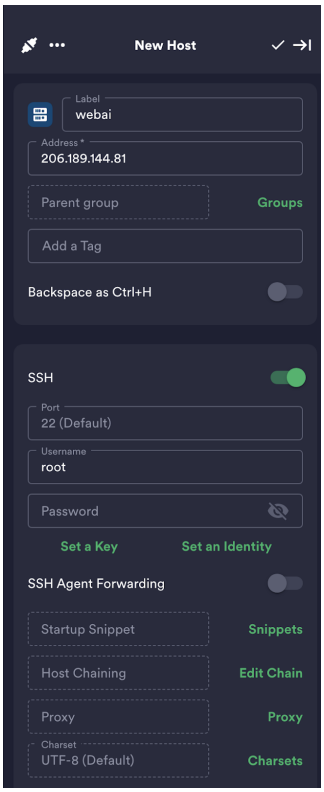


ตัวอย่างอีเมลเมื่อได้รับการของรีเซ็ตรหัสผ่าน



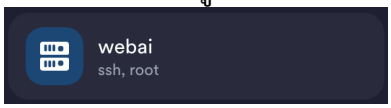
การอัพโหลดไฟล์โปรเจคและการแตกไฟล์

1. ใช้โปรแกรมสำหรับ SSH เข้าเครื่องที่ถนัด (ซึ่งจะแนะนำ termius) หรือ terminal ที่มาพร้อมกับเครื่อง mac, ubuntu หรือ cmd ที่มาพร้อมกับ windows ในตัวอย่างจะใช้โปรแกรม termius โดยวัตถุประสงค์หลักการใช้สำหรับการอัพโหลดไฟล์และเพื่อนคอนฟิกเครื่อง (สามารถใช้โปรแกรมอื่นได้ ที่สามารถดำเนินการตามวัตถุประสงค์ได้)
2. สร้าง Host คลิกที่ NEW HOST และกรอกข้อมูลที่จำเป็น ได้แก่

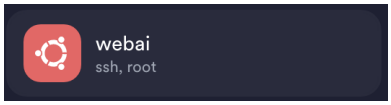


- Label ชื่อเครื่องที่จะแสดงในโปรแกรม
- Address คือ ip address ของเครื่องที่ได้จากขั้นตอนการสร้าง vm
- ในหัวข้อ SSH Username คือ root และ Password ที่ได้กำหนดไว้

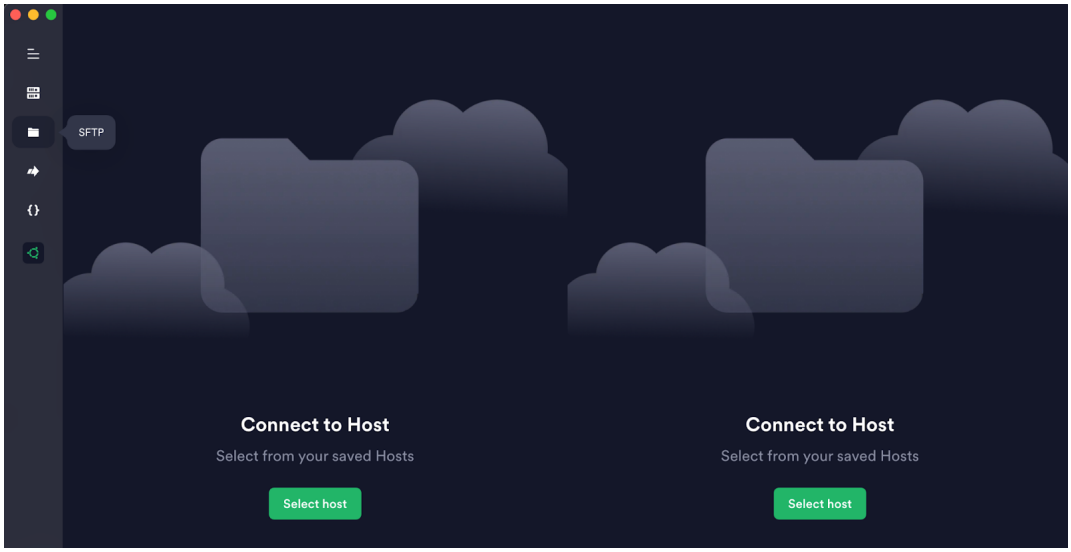
เมื่อสร้างเสร็จสมบูรณ์จะแสดงเป็นดังภาพ



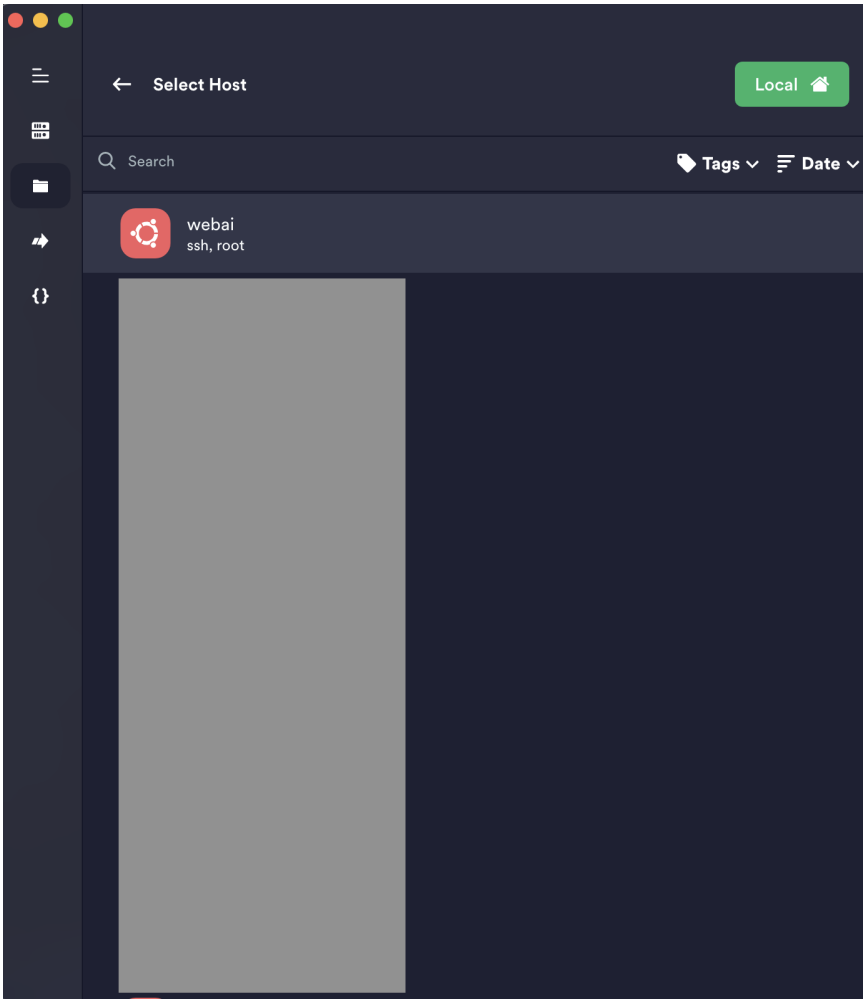
กรณีที่มีการเชื่อมต่อแล้วจะแสดงดังภาพ



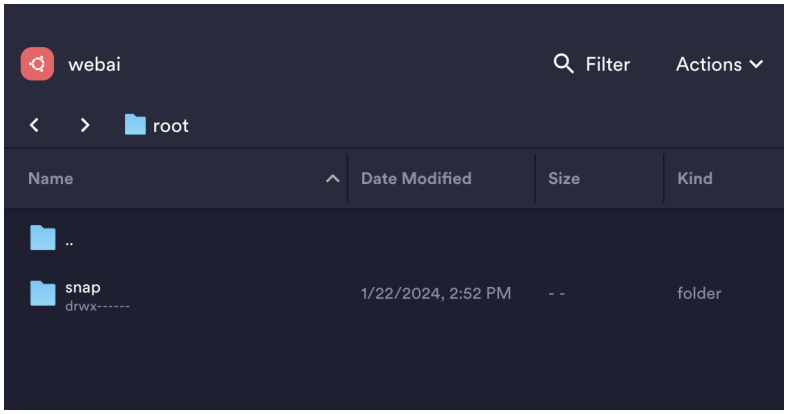
3. การอัพโหลดไฟล์โปรเจคสามารถใช้โปรแกรม termius คลิกที่ไอคอน จากนั้นคลิกที่ Select host



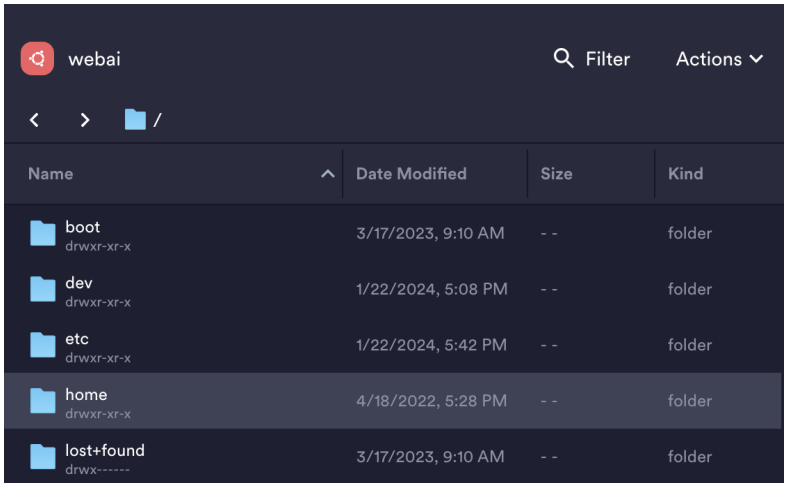
4. เลือก Host ที่ต้องการ ซึ่งในตัวอย่างจะเป็น Host ที่ชื่อ webai เป็นชื่อที่ตั้งไว้ในขั้นตอนของการเพิ่ม Host



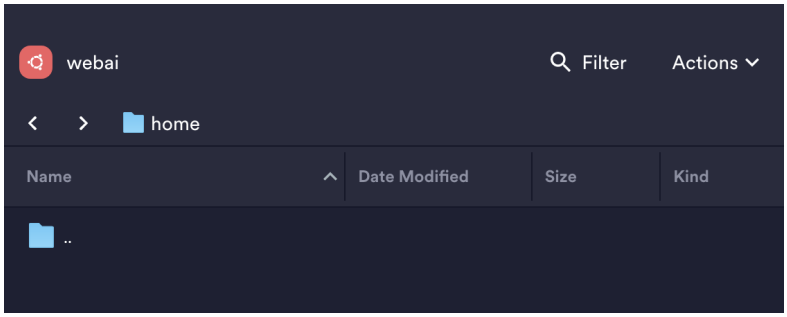
เลือก Host ชื่อ webai หรือที่ตั้งไว้



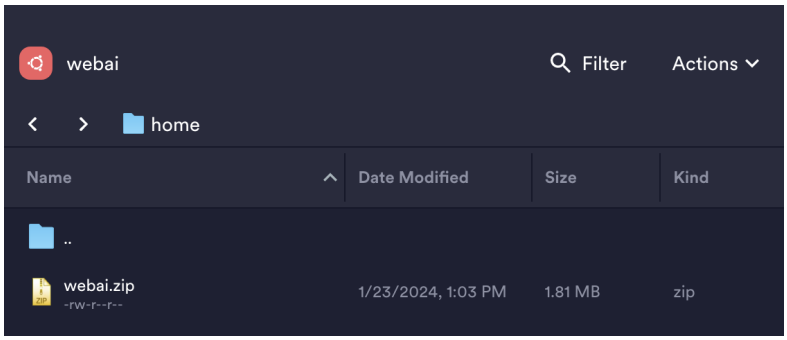
คลิกที่ Folder .... 
เพื่อย้อนกลับไปเข้า Folder /



จากนั้นคลิกที่ Folder Home  จะได้ดังภาพ



5. ทำการอัพโหลดไฟล์ โดยใช้การดึงไฟล์ .zip ที่เตรียมไว้ใส่ใน Folder Home จะได้ดังภาพ



จัดเตรียมและตั้งค่า Ubuntu Server 22.04

1. อัพเดต Ubuntu และ Package

```
sudo apt update
```

2. ตรวจสอบเวอร์ชั่น Python

```
python3 -V

Output
Python 3.10.4
```

3. ติดตั้ง Pip

```
sudo apt install python3-pip
```

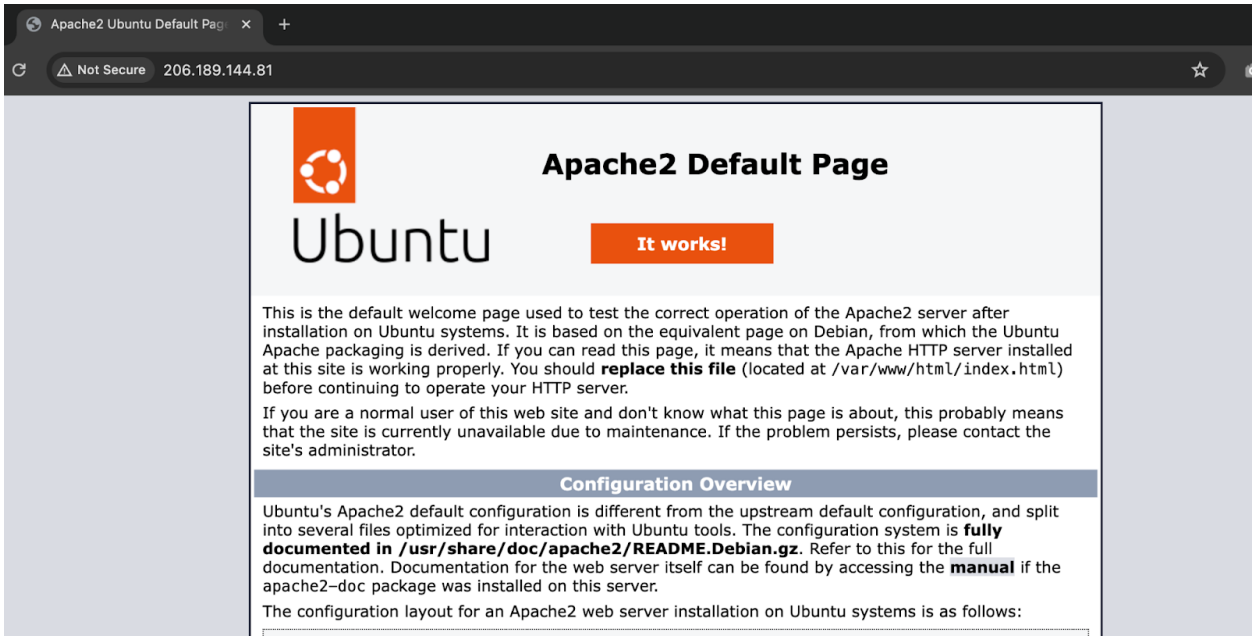
4. ติดตั้ง Apache2 และ wsgi (สำหรับทำ web server)

```
sudo apt-get install apache2 libapache2-mod-wsgi-py3
```

5. ติดตั้ง Virtualenv

```
pip3 install virtualenv
```

ทดสอบเรียก web server โดยใช้หมายเลข ip address ของ vm ที่สร้างขึ้น หลังจากการติดตั้ง Apache2 หากแสดงดังภาพคือถูกต้อง



เริ่มติดตั้งโปรเจค webai

1. สร้าง Folder ชื่อว่า webai ภายใต้ folder Home

```
cd ../home
mkdir webai
```

2. ทำการแตกไฟล์ Extracting หรือ unzip โดยใช้คำสั่ง unzip webai.zip -d webai กรณีที่ยังไม่มีคำสั่ง unzip ให้ติดตั้งโดยใช้คำสั่ง apt install unzip และเรียกใช้งาน unzip webai.zip -d webai อีกครั้งจะได้ดังภาพ

```
unzip webai.zip -d webai

// สำหรับติดตั้งคำสั่ง unzip
apt install unzip
```

```
root@webai:/home# unzip webai.zip -d /webai
Archive:  webai.zip
   creating: /webai/webai/
  inflating: /webai/webai/db.sqlite3
   creating: /webai/webai/backendai/
  inflating: /webai/webai/backendai/admin.py
  inflating: /webai/webai/backendai/models.py
  inflating: /webai/webai/backendai/tests.py
  inflating: /webai/webai/backendai/views.py
  inflating: /webai/webai/backendai/apps.py
  inflating: /webai/webai/backendai/testimage.py
```

3. สร้าง Virtual Environments โดยใช้ชื่อ env
โดยให้ตำแหน่งไปอยู่ที่ /home/webai ซึ่ง folder webai ที่ได้สร้างขึ้นในขั้นตอนก่อนหน้า

```
cd /home/webai
virtualenv env
```

```
4  หรือกรณีที่ไม่สามารถสร้างได้
5  python3 -m virtualenv env
```

4. เรียกใช้งาน Virtual Environments ที่สร้างขึ้น (Activate)

```
1  source env/bin/activate
```

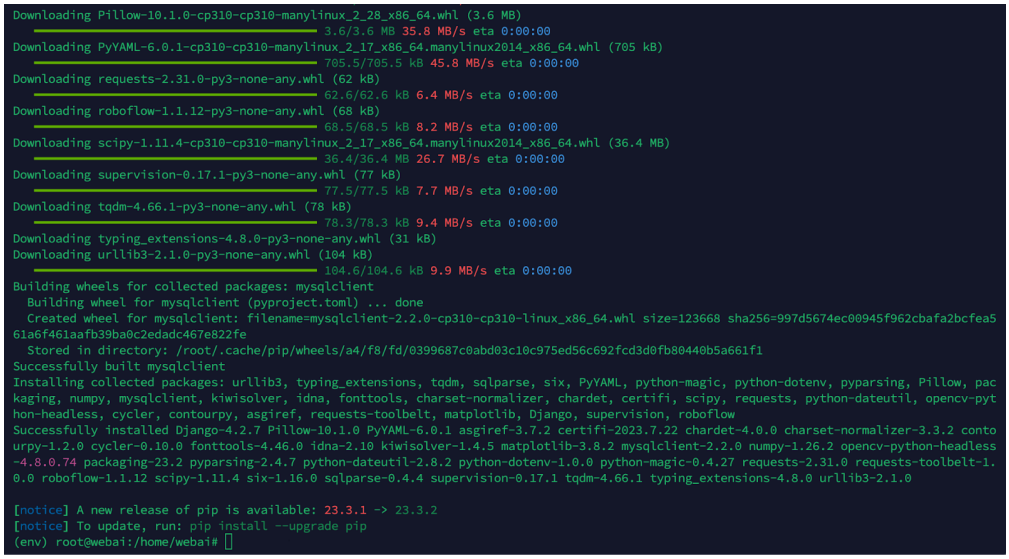
5. ติดตั้ง Libraries จากไฟล์ requirements.txt หรือชื่อไฟล์ที่เตรียมไว้

```
1  pip install -r requirements.txt
```

กรณีที่ติดตั้งแล้วมีปัญหาของ mysqlclient ให้ทำการติดตั้งคำสั่งด้านล่างนี้ก่อน เมื่อติดตั้งเสร็จสมบูรณ์แล้ว ให้ทำการติดตั้ง Libraries ใหม่อีกรอบ

```
1  apt install python3-dev default-libmysqlclient-dev build-essential pkg-config
```

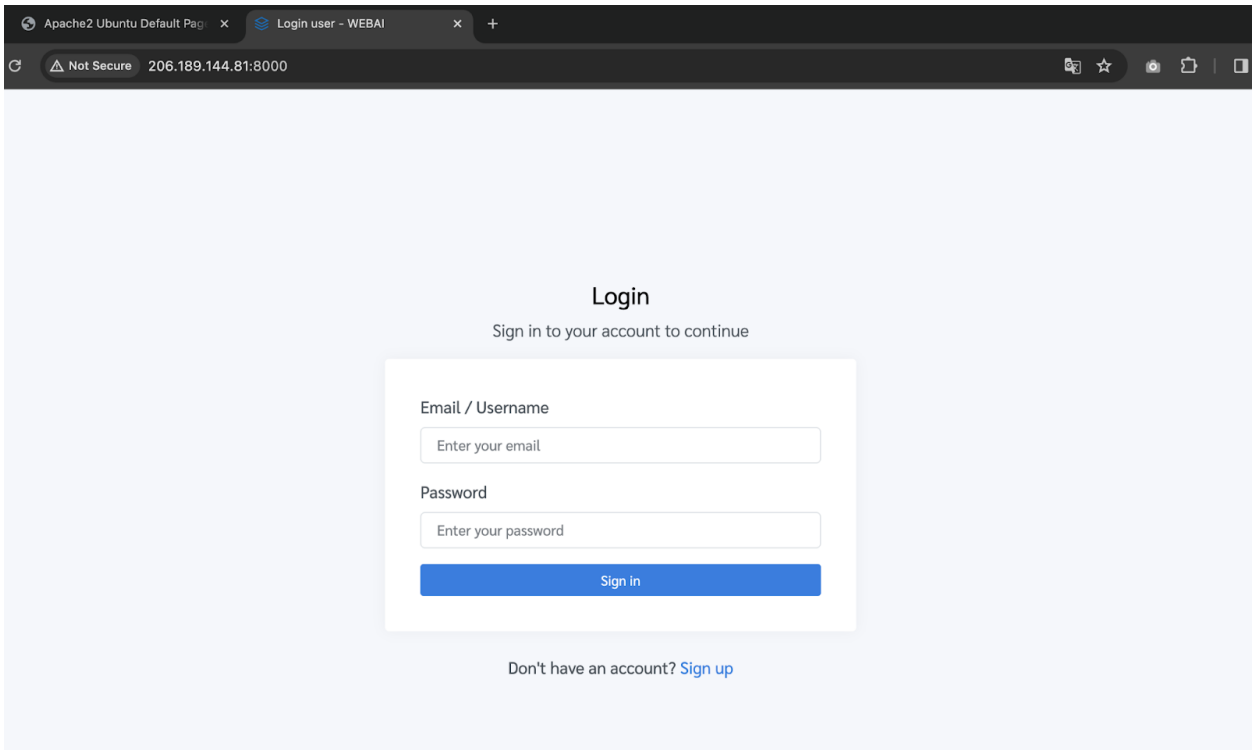
เมื่อติดตั้งเสร็จสมบูรณ์จะแสดงดังภาพ



6. ทดสอบการเรียกใช้งานระบบผ่าน Ip address โดยทำการอนุญาตให้เรียกใช้งานผ่าน port 8000 และทำการ run server เหมือนกับการ run server ที่อยู่ระหว่างการพัฒนาระบบ

```
1  // อนุญาตให้เรียกใช้งานผ่าน port 8000
2  sudo ufw allow 8000
3
4  // คำสั่ง run server โดย your_server_ip กำหนดให้เป็นของเครื่อง
5  python manage.py runserver your_server_ip:8000
6
7  // เรียกใช้งานผ่าน ip address
8  http://your_server_ip:8000
```

เมื่อเรียกใช้งาน หากถูกต้องจะได้ดังภาพ



7. การทำ Collectstatic File

```
1  python manage.py collectstatic
```

8. การอนุญาตและให้สิทธิ์การเข้าถึง

```
1  sudo chown www-data:www-data /home/webai/*
2  sudo chmod -R 775 /home/webai/webai/
3  sudo chmod -R 777 /home/webai/webai/static/*
```

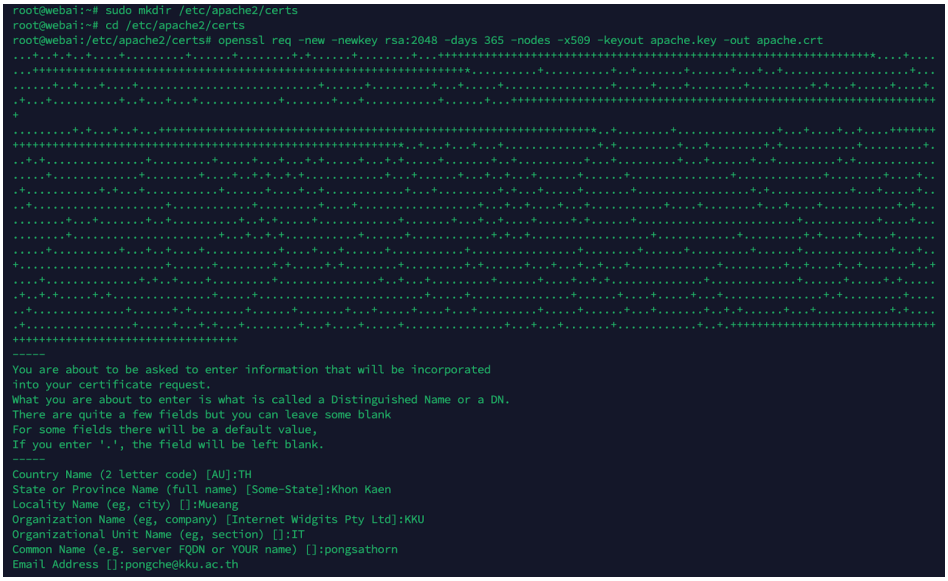
9. สร้าง certs สำหรับการเรียกใช้งานแบบ HTTPS โดยใช้ Openssl (ควรจะเป็น HTTPS เท่านั้น)

```
1  sudo mkdir /etc/apache2/certs
2  cd /etc/apache2/certs
3
4  openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout apache.key -out apache.crt
5
6  // เมื่อลอง ls จะพบกับไฟล์ 2 ไฟล์ ได้แก่
7  apache.crt  apache.key
8
```

โดยรายละเอียดให้ระบุเป็น

- Country Name (2 letter code) [AU]:TH
- State or Province Name (full name) [Some-State]:Khon Kaen

- Locality Name (eg, city) []:Mueang
- Organization Name (eg, company) [Internet Widgits Pty Ltd]:KKU
- Organizational Unit Name (eg, section) []:IT
- Common Name (e.g. server FQDN or YOUR name) []:pongsathorn
- Email Address []:pongche@kku.ac.th



10. คอนฟิก Apache

โดยการสร้างไฟล์ webai.conf เพื่อเก็บไฟล์คอนฟิกของ Apache

```
1  touch /etc/apache2/sites-available/webai.conf
2
3  sudo vi /etc/apache2/sites-available/webai.conf
```

จากนั้นคัดลอกค่านำไปใส่ โดยที่แก้ไขในข้อความพื้นหลังสีเหลืองให้ถูกต้อง

```
1  <VirtualHost *:80>
2    ServerAdmin me@pongthorn.in.th
3    ServerName webai.pongthorn.in.th
4    ServerAlias webai.pongthorn.in.th
5
6    Redirect permanent / https://webai.pongthorn.in.th/
7  </VirtualHost>
8
9  <VirtualHost *:443>
10   ServerAdmin me@pongthorn.in.th
11   ServerName webai.pongthorn.in.th
12   ServerAlias webai.pongthorn.in.th
13
14   Alias /static /home/webai/webai/static
15   <Directory /home/webai/webai/static>
```

```
        Require all granted
    </Directory>

    <Directory /home/webai/webai/webai>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    WSGIDaemonProcess webai python-home=/home/webai/env python-path=/home/webai
    WSGIProcessGroup webai
    WSGIScriptAlias / /home/webai/webai/webai/wsgi.py
    WSGIApplicationGroup %{GLOBAL}

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    SSLEngine on
    SSLCertificateFile /etc/apache2/certs/apache.crt
    SSLCertificateKeyFile /etc/apache2/certs/apache.key
</VirtualHost>

```

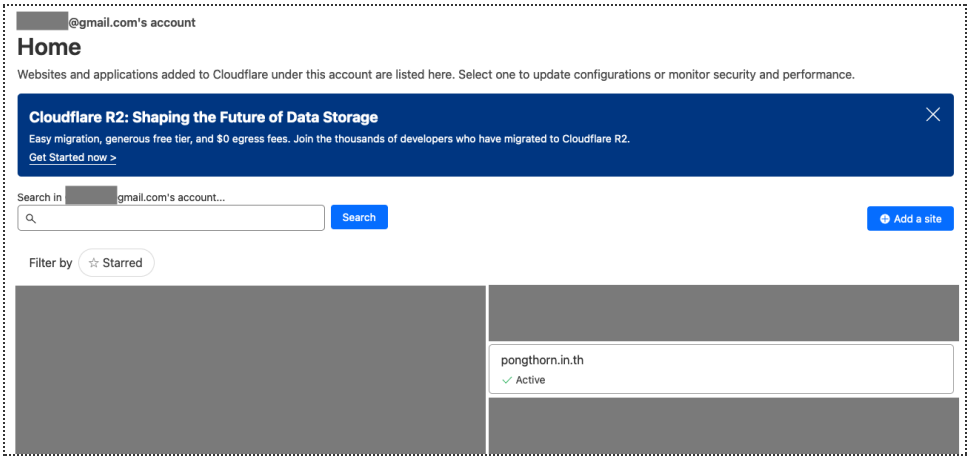
จากนั้นให้ดำเนินการต่อโดยใช้คำสั่งต่อไปนี้

```
sudo a2dissite 000-default.conf
sudo a2ensite django.conf
sudo a2enmod ssl
sudo a2enmod rewrite
sudo apache2ctl configtest
sudo systemctl restart apache2
```

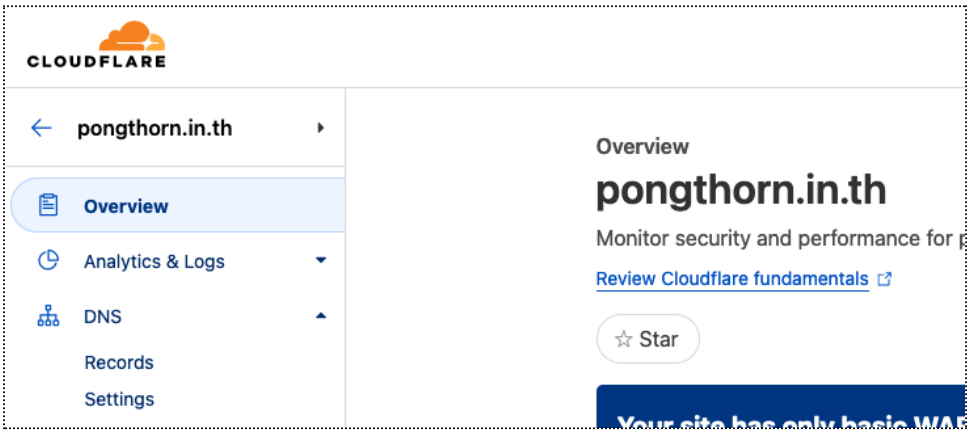
การตั้งค่าให้เรียกใช้งานผ่าน Domain Name

การใช้งานระบบผ่านการเรียก domain name สามารถทำได้หลายวิธีซึ่งวิธีที่จะทำนี้เป็นการกำหนดใน DNS ของ domain เจ้านั้น ๆ โดยในตัวอย่างจะทำผ่าน cloudflare กรณีที่มีการเพิ่ม domain เข้าไปอยู่ใน cloudflare เรียบร้อยแล้ว

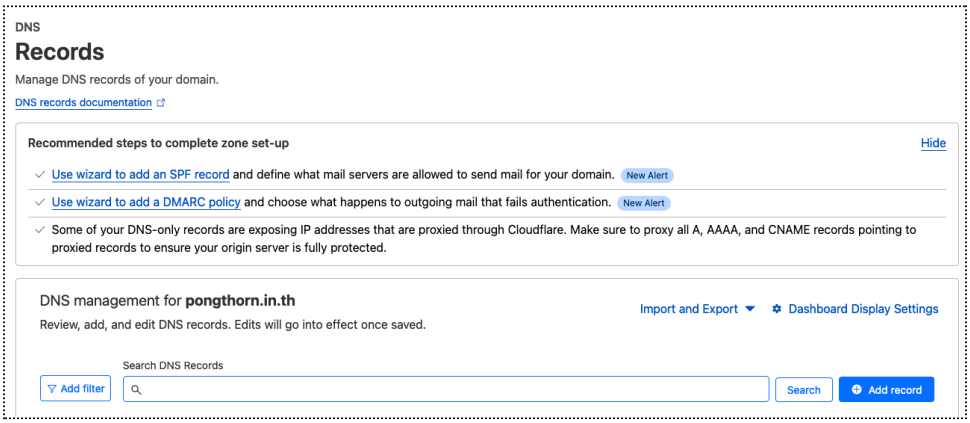
1. หลังจาก login ให้เลือก Domain ที่ต้องการใช้งาน ในที่นี้คือ pongthorn.in.th



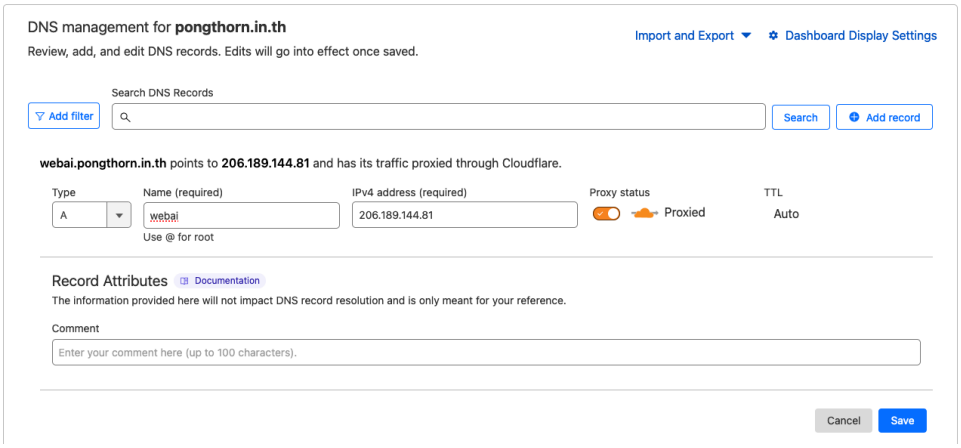
2. คลิกที่เมนู DNS



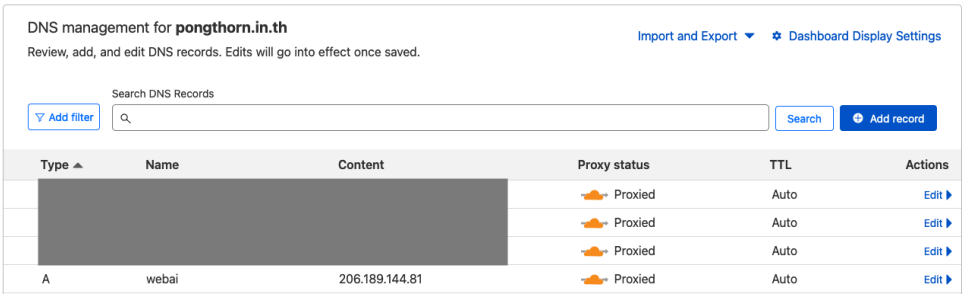
3. จากนั้นคลิกที่ปุ่ม 



4. กำหนดค่า Type ให้เป็น A และ Name ให้เป็นชื่อที่ต้องการ และ IPv4 address ที่ได้จากเครื่อง vm ที่สร้างขึ้น เช่น webai ก็จะได้เป็น subdomain => webai.pongthorn.in.th กรณีที่ต้องการให้กำหนดเป็น root domain ในหัวข้อของ Name ให้ระบุเป็น @ เท่านั้น จากนั้นทำการบันทึกคลิกที่ปุ่ม 

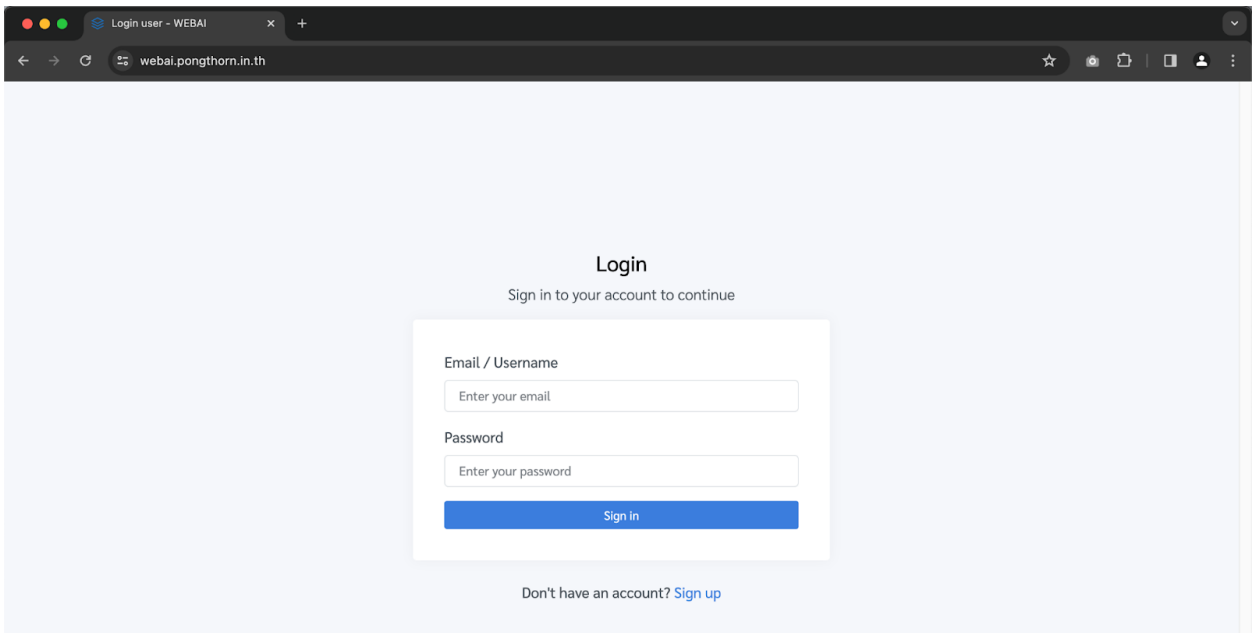


จะได้ดังภาพ และเสร็จเรียบร้อย



ในขั้นตอนของการติดตั้งระบบในหัวข้อ 37 - 38 หากไม่สามารถติดตั้งได้ หรือติดปัญหา Error ให้ Google Chat มาที่ pongche@kku.ac.th หรือติดต่อทางอีเมลที่ me@pongthorn.in.th ครับ

เมื่อติดตั้งได้สมบูรณ์ และเรียกใช้งานผ่าน webai.pongthorn.in.th



== ==
ฉบับพื้นฐาน
“ลองเรียนรู้ ไม่รักก็เปลี่ยนใหม่”
== ==