**คู่มือสำหรับนักพัฒนา**

1. **ส่วนของ Back-End**

ภายในโฟลเดอร์ backend ซึ่งถูกพัฒนาด้วย Python – Django ซึ่งสามารถ clone ได้จาก <https://github.com/SuteeSaraphan/IPAuTSoNS> ประกอบไปด้วยไฟล์และโฟลเดอร์ดังรูปที่ 1

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, ออกแบบ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 1 ไฟล์และโฟลเดอร์ภายในโฟลเดอร์ backend

**1.1 โฟลเดอร์ env**

โฟลเดอร์ env เป็นโฟลเดอร์ที่ถูกสร้างขึ้นมาเพื่อสร้าง environment ในการพัฒนาให้กับ โฟลเดอร์ IPAutSoNsAPI เพื่อลดปัญหาที่อาจจะเกิดขึ้นจาก version ของ library เสริมที่ติดตั้งเข้ามาเพื่อ ใช้งาน และ ลดปัญหาการหา path file ไม่พบอีกด้วย โดยมีไฟล์และโฟลเดอร์ดังรูปที่ 2

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 2 ไฟล์และโฟลเดอร์ภายในโฟลเดอร์ env

**1.2 โฟลเดอร์ IPAutSoNsAPI**

โฟลเดอร์ IPAutSoNsAPI เป็นโฟลเดอร์หลักของการเก็บไฟล์ที่ใช้งานในการสร้างส่วน Backend โดยจะมี Django project ที่ถูกสร้างเอาไว้ภายใน โดยมีไฟล์และโฟลเดอร์ดังรูปที่ 3

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 3 ไฟล์และโฟลเดอร์ภายในโฟลเดอร์ IPAutSoNsAPI

โดยตัว project Django จะแบ่งเป็น 2 ส่วนหลักคือ โฟลเดอร์ IPAutSoNsAPI และ โฟลเดอร์ api และไฟล์คำสั่งย่อยอื่นๆ ประกอบไปด้วย

**1.2.1 โฟลเดอร์ api**

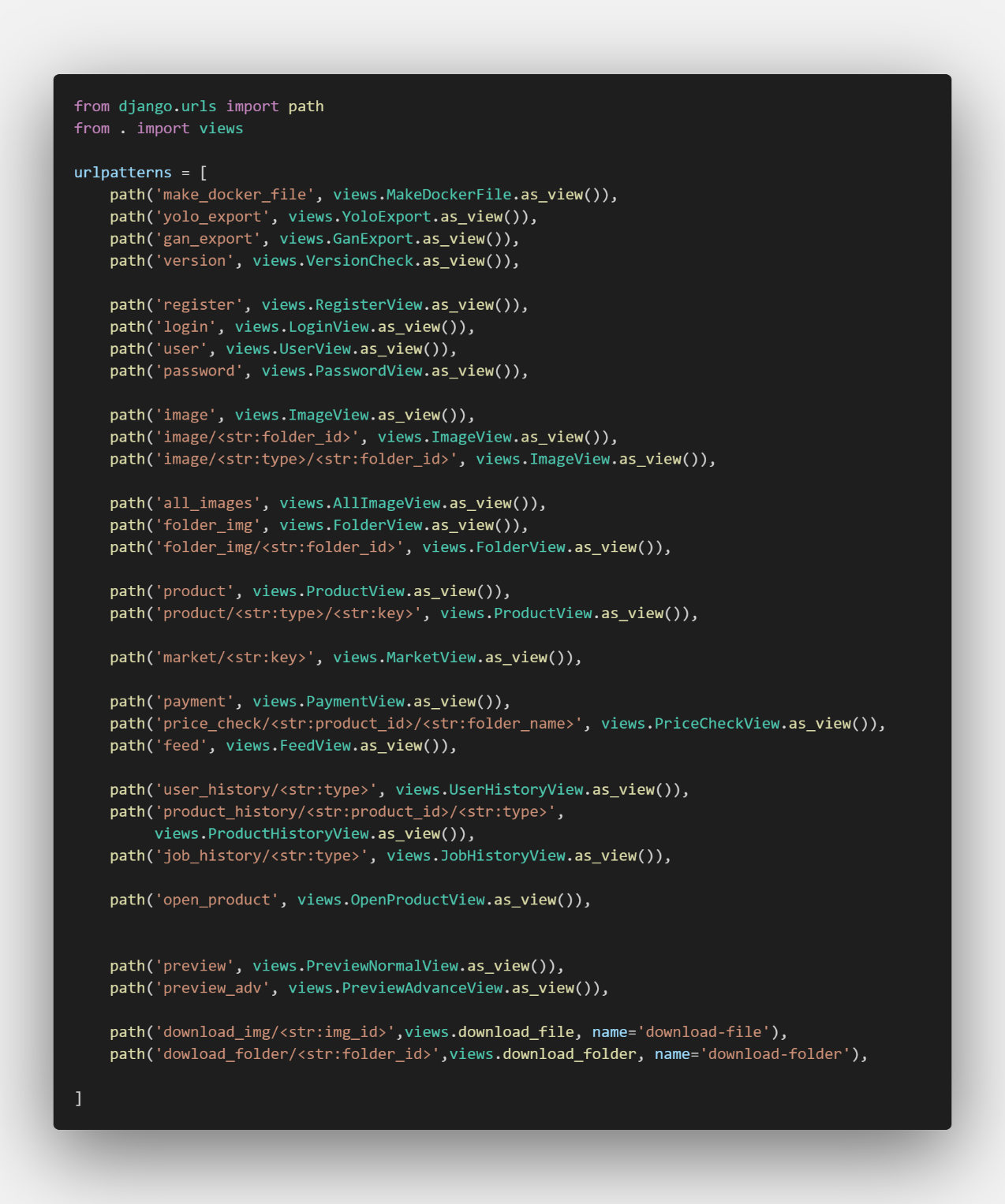
โดยโฟลเดอร์จะเป็นส่วนหลักของการทำงานด้าน API ที่เชื่อมต่อเข้ากับ Frontend , การประมวลผลและการจัดเก็บข้อมูลต่างๆ ของ Web application โดยมีไฟล์และโฟลเดอร์ดัง รูปที่ 4

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 4 ไฟล์และโฟลเดอร์ภายในโฟลเดอร์ api

1. model.py เป็นไฟล์ที่ใช้สำหรับกำหนดโครงสร้างของ database โดยสามารถกำหนดโครงสร้าง Entity และ Attribute ของ Entity ได้โดยจะอยู่ในรูปแบบของ Class
2. serializers.py เป็นไฟล์ที่ใช้สำหรับปรับรูปแบบการเข้าถึง Model ต่างๆเพื่อให้ง่ายต่อการเรียกใช้ Model
3. url.py เป็นไฟล์สำหรับตั้ง URL เพื่อเข้าถึง API เรียกใช้ Functions ต่างๆ โดยสามารถใช้ method ได้หลากหลายในการเข้าถึง เช่น GET,POST,PUT,DELETE เป็นต้น ซึ่งสามารถกำหนดให้รับค่า parameter เพื่อนส่งต่อไปให้ Functions เรียกใช้งานได้อีกด้วย โดยส่วนปนะสอบสามารถดูได้ที่รูปที่ 5



รูปที่ 5 URL ทั้งหมดที่ใช้งาน API ได้

1. view.py เป็นไฟล์ที่รบรวมการทำงานประมวลผลและจัดเก็บข้อมูลทั้งหมดเอาไว้โดยจะถูกแบ่งออกเป็น class ตามการทำงานที่เกี่ยวข้อง

**1.2.2 โฟลเดอร์ IPAutSoNsAPI**

เป็นโฟลเดอร์หลักของ Django Project ที่ได้มีการเก็บไฟล์ config ค่าต่างๆ ของ

backend เอาไว้ โดยมีไฟล์และโฟลเดอร์ดังรูปที่ 6

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, ออกแบบ

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 6 ไฟล์และโฟลเดอร์ที่อยู่ในโฟลเดอร์ IPAutSoNsAPI

1. setting.py ไฟล์รวมการตั้งค่าทั้งหมดของ Project โดยสามารถปรับค่าต่างๆ เฉพาะด้านได้ภายในไฟล์นี้ เช่น การเชื่อมต่อ Database, การกำหนด IP address สำหรับใช้งาน Server ไปจนถึงการกำหนด path เก็บไฟล์
2. url.py ไฟล์ไว้สำหรับการตั้งค่า URL ที่จะให้เข้าถึงการใช้งาน API

**1.2.3 โฟลเดอร์ yaml\_file**

เป็นโฟลเดอร์สำหรับเก็บไฟล์ประเภท yaml เพื่อเรียกสั่งงาน order image processing job ซึ่งจะสามารถปรับเปลี่ยนย้ายที่ได้หากผู้พัฒนาต้องการ โดยรูปที่ 7 คือรูปภายในโฟล์เดอร์ yaml\_file และรูปที่ 8 คือตัวอย่างไฟล์ yaml ที่ใช้สำหรับสั่งงาน

**รูปภาพประกอบด้วย ข้อความ, ตัวอักษร, ภาพหน้าจอ, จำนวน

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 7 ไฟล์ที่อยู่ในโฟลเดอร์ yaml\_file

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 8 ตัวอย่างไฟล์ yaml สำหรับสั่งงาน

**1.2.4 ไฟล์ manage.py**

ไฟล์สำหรับสั่งเปิดใช้งาน backend API server ให้สามารถเข้าใช้งานได้ผ่าน URL ที่ ผู้พัฒนานั้นตั้งเอาไว้

**1.2.5 ไฟล์ preview\_adv\_api.py**

ไฟล์สำหรับการสั่งงานแบบ Preview เพื่อให้ได้รูปที่ผ่านการประมวลผลแบบเร็วที่สุดโดยที่ไม่ห่วงคุณภาพ เพื่อให้สามารถนำไปแสดงผลให้ผู้ใช้ได้ไวที่สุด โดยจะเป็นการ Preview ในรูปแบบใช้งาน weight จากสินค้าที่มีผู้ใช้เพิ่มขึ้นมา โดยรูปที่ 9 คือภายในของไฟล์

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 9 ภายในของไฟล์ preview\_adv\_api.py

**1.2.6 ไฟล์ preview\_api.py**

ไฟล์สำหรับการสั่งงานแบบ Preview เพื่อให้ได้รูปที่ผ่านการประมวลผลแบบเร็วที่สุดโดยที่ไม่ห่วงคุณภาพ เพื่อให้สามารถนำไปแสดงผลให้ผู้ใช้ได้ไวที่สุด โดยจะเป็นการ Preview ในรูปแบบใช้งาน weight จากสินค้าที่มีเป็นของทาง Web applicationโดยรูปที่ 10 คือภายในของไฟล์

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 10 ภายในของไฟล์ preview\_api.py

**1.2.7 ไฟล์ yaml\_run.py**

ไฟล์สำหรับสั่งงานการทำงาน Image processing โดยผ่านการสั่งให้ไฟล์ yaml ให้ทำงาน ซึ่งจะสามารถส่งค่าสถานการณ์สั่งงานกลับมาได้ว่าสั่งงานสำเร็จหรือไม่ โดยรูปภาพที่ 11 คือตัวอย่างภายในไฟล์

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, ซอฟต์แวร์มัลติมีเดีย

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 11 ภายในของไฟล์ yaml\_run.py

**1.3 ไฟล์ Dockerfile**

ใช้สำหรับการสร้างหรือ build Docker image เพื่อใช้สำหรับการ Deploy ส่วน Backend ในเครื่อง Server จริงโดยรูปภาพที่ 12 คือตัวอย่างภายในไฟล์

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 12 ภายในของไฟล์ Dockerfile

**1.4 ไฟล์ Readme.txt**

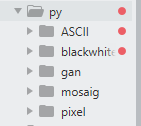
เป็นไฟล์สำหรับแสดงรายละเอียดของโฟลเดอร์ backend

**1.5 ไฟล์ req.txt**

เป็นไฟล์สำหรับจัดการ package ต่าง ๆ ที่ใช้ในโปรเจค

1. **ส่วนของ Front-End**
2. **ส่วนของ Application ประมวลผลภาพ**

ภายในโฟลเดอร์แอพพลิเคชั่นพัฒนาด้วย Python ประกอบด้วยโฟลเดอร์ และ ไฟล์ทั้งหมดตามโครงสร้างดังรูปที่ สามารถ clone ได้จาก https://github.com/SuteeSaraphan/IPAuTSoNS



รูป ..

**3.1 โฟลเดอร์ ASCII**

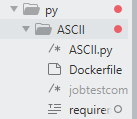
a. ascii.py สำหรับเป็น Script ในการประมวลผลภาพ

b. Dockerfile ในการสร้าง Python Script ให้อยู่ในรูปแบบของ Docker Image

c. jobtestcom1.yaml เป็นคำสั่งสำหรับสั่งงานประมวลผลในรูปแบบ Job ของ Kubernetes เพื่อทดสอบ Python Script ตัวนี้บนรูปแบบ Docker Image

d. requirement.txt เป็นไฟล์ในการเก็บ Requirement ที่จำเป็นของ Python Script นี้เพื่อนำไปใช้งานกับ Dockerfile

โดยการทำงานหลักของโปรแกรมจะต้องใส่ Parameter เป็น JobID, UserID, Folder(ภาพที่ต้องการประมวผล), NewFolder(ตำแหน่งใหม่ที่ต้องการเก็บภาพ) โดยตัว Folder ข้างในของ ASCII จะแสดงดังรูปที่...



รูป ..

**3.2 โฟลเดอร์ blackwhite**

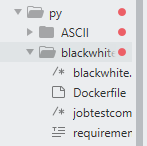
a. blackwhite.py สำหรับเป็น Script ในการประมวลผลภาพ

b. Dockerfile ในการสร้าง Python Script ให้อยู่ในรูปแบบของ Docker Image

c. jobtestcom1.yaml เป็นคำสั่งสำหรับสั่งงานประมวลผลในรูปแบบ Job ของ Kubernetes เพื่อทดสอบ Python Script ตัวนี้บนรูปแบบ Docker Image

d. requirement.txt เป็นไฟล์ในการเก็บ Requirement ที่จำเป็นของ Python Script นี้เพื่อนำไปใช้งานกับ Dockerfile

โดยการทำงานหลักของโปรแกรมจะต้องใส่ Parameter เป็น JobID, UserID, Folder(ภาพที่ต้องการประมวผล), NewFolder(ตำแหน่งใหม่ที่ต้องการเก็บภาพ) โดยตัว Folder ข้างในของ blackwhite จะแสดงดังรูปที่...



รูป ..

**3.3 โฟลเดอร์ mosaig**

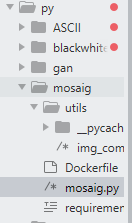
a. mosaig.py สำหรับเป็น Script ในการประมวลผลภาพ

b. Dockerfile ในการสร้าง Python Script ให้อยู่ในรูปแบบของ Docker Image

c. requirement.txt เป็นไฟล์ในการเก็บ Requirement ที่จำเป็นของ Python Script นี้เพื่อนำไปใช้งานกับ Dockerfile

d. ในโฟลเดอร์ utils ประกอบด้วยไฟล์ img\_common\_util.py สำหรับเป็น Function ในการ convert\_image\_to\_tensor, convert\_tensor\_to\_image เพื่อทำงานประมวลใน mosaig.py

โดยการทำงานหลักของโปรแกรมจะต้องใส่ Parameter เป็น JobID, UserID, Folder(ภาพที่ต้องการประมวผล), SelectImage(ภาพที่ต้องการเลือกให้เป็นภาพหลักในการทำ Mosaig), NewFolder(ตำแหน่งใหม่ที่ต้องการเก็บภาพ) โดยตัว Folder ข้างในของ mosaig จะแสดงดังรูปที่...



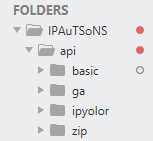
รูป ..

**3.4 โฟลเดอร์ pixel**

**3.5 โฟลเดอร์ gan**

1. **ส่วนของ API**

ภายในโฟลเดอร์ API พัฒนาด้วย Python ประกอบด้วยโฟลเดอร์ และ ไฟล์ทั้งหมดตามโครงสร้างดังรูปที่ สามารถ clone ได้จาก <https://github.com/SuteeSaraphan/IPAuTSoNS>



รูป ..

**4.1 โฟลเดอร์ basic**

a. ascii.py สำหรับเป็น Script ในการประมวลผลภาพ

b. Dockerfile ในการสร้าง Python Script ให้อยู่ในรูปแบบของ Docker Image

**4.2 โฟลเดอร์ gan**

**4.3 โฟลเดอร์ ipyolor**

**4.4 โฟลเดอร์ zip**

1. **ส่วนของ Yaml ในการทำงาน Kubernetes**
2. **ส่วนของ MongoDB**