

สำหรับการทำระบบ Web Application ที่จัดเก็บและติดตามสถานะเอกสารตามรูปแบบที่คุณต้องการ นี่คือคำแนะนำเบื้องต้น:

1. การเลือกเทคโนโลยี

- **Backend:** ใช้ **Flask** หรือ **FastAPI** (ถนัดใน FastAPI ก็ดีเพราะรองรับการทำงานแบบ Asynchronous ซึ่งเหมาะสำหรับการอัปเดตสถานะเอกสารแบบเรียลไทม์)
- **Database:** ใช้ **PostgreSQL** หรือ **MySQL** สำหรับเก็บข้อมูลเอกสารและสถานะในแต่ละขั้นตอน
- **Frontend:** ใช้ **HTML/CSS** ร่วมกับ **JavaScript** (อาจใช้ **React** หรือ **Vue.js** เพื่อช่วยจัดการ UI) สำหรับแสดงข้อมูลและทำงานแบบ Interactive
- **Barcode/QR Code Generation:** ใช้ **Python Libraries** เช่น **qrcode** หรือ **barcode** ในการสร้างและพิมพ์รหัส QR หรือบาร์โค้ด

2. โครงสร้างฐานข้อมูล

สามารถใช้โครงสร้างตารางดังนี้: - **Documents:** เก็บข้อมูลเอกสาร เช่น `doc_id, title, sender_name, receiver_name, created_at, status` - **DocumentStatus:** เก็บสถานะของเอกสารในแต่ละขั้นตอน เช่น `doc_id, location, status, updated_at`

3. ฟังก์ชันหลักของระบบ

- **สร้างเอกสาร:** หน้าจอสำหรับผู้สร้างข้อมูลเอกสารใหม่ และระบบจะออกเลขรหัสเอกสาร พร้อมสร้าง QR Code
- **บันทึกสถานะ:** เมื่อเอกสารไปถึงแต่ละจุด (ผู้รับ-ส่งเอกสารต้นทาง, ปลายทาง) ผู้ใช้สามารถบันทึกสถานะได้ เช่น ตำแหน่ง, เวลา, ชื่อผู้รับเอกสาร
- **ค้นหาสถานะ:** สามารถค้นหาข้อมูลเอกสารตาม `doc_id, title, sender_name, receiver_name` เพื่อดูสถานะปัจจุบันของเอกสาร

4. การพัฒนา API

สร้าง API ที่สามารถทำหน้าที่ดังนี้: - **POST /documents:** สำหรับสร้างเอกสารใหม่ - **POST /documents/{id}/status:** สำหรับอัปเดตสถานะในแต่ละจุด - **GET /documents/{id}:** สำหรับดึงข้อมูลเอกสารและสถานะปัจจุบัน

5. การพิมพ์ QR Code/Barcode

ใช้ไลบรารีใน Python เช่น **qrcode** ในการสร้าง QR Code และพิมพ์รายละเอียดบนหน้าจอหรือไฟล์ PDF ที่สามารถพิมพ์ได้

6. การพัฒนา UI

ออกแบบหน้าเว็บสำหรับ: - **สร้างเอกสาร:** มีแบบฟอร์มให้กรอกข้อมูลและกดปุ่มเพื่อสร้าง QR Code พร้อมรายละเอียด - **บันทึกสถานะ:** สำหรับพนักงานในแต่ละจุดเพื่อบันทึกการรับเอกสาร - **ตรวจสอบสถานะ:** แสดงสถานะเอกสารล่าสุดเมื่อมีการค้นหา

7. ระบบตรวจสอบสถานะ

สามารถเพิ่มฟังก์ชันการแจ้งเตือน (เช่น ระบบ Notification หรือ Email) ให้ผู้ใช้ทราบสถานะเมื่อเอกสารถูกอัปเดต

ถ้าต้องการตัวอย่างโค้ดหรือความช่วยเหลือเพิ่มเติมในการเริ่มต้น สามารถแจ้งได้ครับ

คุณสามารถ สร้าง code เบื้องต้นทั้งหมดให้ผมได้ไหม

ได้รับ นี่คือนิยามโครงสร้างเบื้องต้นสำหรับ Web Application ในการติดตามสถานะเอกสาร ซึ่งใช้ **Flask** และ **SQLite** เป็นฐานข้อมูลเพื่อความสะดวกและความรวดเร็วในการพัฒนาเบื้องต้น

โครงสร้างโปรเจกต์

```
document_tracking/
├── app.py
├── templates/
│   ├── create_document.html
│   ├── update_status.html
│   └── check_status.html
├── static/
│   └── qr_codes/
├── models.py
└── requirements.txt
```

ขั้นตอนที่ 1: ติดตั้ง Dependencies

สร้างไฟล์ `requirements.txt` และเพิ่มรายการนี้

```
Flask
Flask-SQLAlchemy
qrcode[pil]
```

จากนั้นรันคำสั่งนี้ใน terminal:

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

ขั้นตอนที่ 2: สร้างฐานข้อมูลใน `models.py`

`models.py`

```
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
```

```
db = SQLAlchemy()
```

```
class Document(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    doc_id = db.Column(db.String(50), unique=True, nullable=False)
    title = db.Column(db.String(100), nullable=False)
    sender_name = db.Column(db.String(100), nullable=False)
    receiver_name = db.Column(db.String(100), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    status = db.Column(db.String(50), default="Created")
```

```
class DocumentStatus(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    doc_id = db.Column(db.String(50), db.ForeignKey('document.doc_id'),
    nullable=False)
    location = db.Column(db.String(50), nullable=False)
```

```
status = db.Column(db.String(50), nullable=False)
updated_at = db.Column(db.DateTime, default=datetime.utcnow)
```

ขั้นตอนที่ 3: สร้าง Flask Application ใน app.py

```
# app.py
from flask import Flask, render_template, request, redirect, url_for, flash
from models import db, Document, DocumentStatus
import qrcode
import os
```

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)
```

สร้างเส้นทางสำหรับสร้างเอกสารใหม่

```
@app.route('/create_document', methods=['GET', 'POST'])
def create_document():
```

```
    if request.method == 'POST':
        doc_id = request.form['doc_id']
        title = request.form['title']
        sender_name = request.form['sender_name']
        receiver_name = request.form['receiver_name']
```

สร้าง QR Code

```
    qr = qrcode.make(doc_id)
    qr_path = os.path.join('static', 'qr_codes', f'{doc_id}.png')
    qr.save(qr_path)
```

บันทึกข้อมูลเอกสาร

```
    new_document = Document(doc_id=doc_id, title=title,
sender_name=sender_name, receiver_name=receiver_name)
    db.session.add(new_document)
    db.session.commit()
```

```
    flash('Document created successfully!', 'success')
    return redirect(url_for('create_document'))
```

```
    return render_template('create_document.html')
```

เส้นทางสำหรับบันทึกสถานะเอกสาร

```
@app.route('/update_status/<string:doc_id>', methods=['GET', 'POST'])
def update_status(doc_id):
```

```
    document = Document.query.filter_by(doc_id=doc_id).first()
    if request.method == 'POST':
        location = request.form['location']
        status = request.form['status']
```

```

        # บันทึกสถานะ
        new_status = DocumentStatus(doc_id=doc_id, location=location,
status=status)
        db.session.add(new_status)
        db.session.commit()

        document.status = status
        db.session.commit()

        flash('Status updated successfully!', 'success')
        return redirect(url_for('update_status', doc_id=doc_id))

    return render_template('update_status.html', document=document)

# เส้นทางสำหรับตรวจสอบสถานะเอกสาร
@app.route('/check_status/<string:doc_id>')
def check_status(doc_id):
    document = Document.query.filter_by(doc_id=doc_id).first()
    statuses =
DocumentStatus.query.filter_by(doc_id=doc_id).order_by(DocumentStatus.updated
_at.desc()).all()
    return render_template('check_status.html', document=document,
statuses=statuses)

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
        app.run(debug=True)

```

ขั้นตอนที่ 4: สร้าง Template ใน templates/

```

create_document.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Create Document</title>
</head>
<body>
    <h1>Create New Document</h1>
    <form method="POST">
        Document ID: <input type="text" name="doc_id" required><br>
        Title: <input type="text" name="title" required><br>
        Sender Name: <input type="text" name="sender_name" required><br>
        Receiver Name: <input type="text" name="receiver_name" required><br>
        <button type="submit">Create</button>
    </form>
</body>
</html>

```

update_status.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Update Status</title>
</head>
<body>
  <h1>Update Status for Document {{ document.doc_id }}</h1>
  <form method="POST">
    Location: <input type="text" name="location" required><br>
    Status: <input type="text" name="status" required><br>
    <button type="submit">Update</button>
  </form>
</body>
</html>
```

check_status.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Check Status</title>
</head>
<body>
  <h1>Status for Document {{ document.doc_id }}</h1>
  <p>Title: {{ document.title }}</p>
  <p>Sender: {{ document.sender_name }}</p>
  <p>Receiver: {{ document.receiver_name }}</p>
  <h2>Statuses</h2>
  <ul>
    {% for status in statuses %}
      <li>{{ status.location }} - {{ status.status }} ({{
status.updated_at }})</li>
    {% endfor %}
  </ul>
</body>
</html>
```

การเริ่มต้นใช้งาน

1. สร้างฐานข้อมูล:

python app.py

2. เปิดเบราว์เซอร์และทดสอบ URL ต่อไปนี้:

- สร้างเอกสาร: http://localhost:5000/create_document
- อัปเดตสถานะเอกสาร: http://localhost:5000/update_status/
- ตรวจสอบสถานะเอกสาร: http://localhost:5000/check_status/

หมายเหตุ

- ระบบนี้เป็นโครงสร้างพื้นฐานเบื้องต้น คุณสามารถเพิ่มเติมฟังก์ชันหรือปรับปรุงตามความต้องการ เช่น เพิ่มระบบค้นหาจากชื่อเรื่อง, ระบบยืนยันการอัปเดตสถานะ, และฟีเจอร์การแจ้งเตือน

ช่วยสร้าง code ส่วนของการค้นหา และการแจ้งเตือนผ่าน email ด้วย

เพื่อเพิ่มฟีเจอร์การค้นหาและการแจ้งเตือนผ่าน Email ในระบบ Web Application นี้ เราสามารถทำได้โดยการเพิ่ม Endpoint สำหรับการค้นหาและฟังก์ชันสำหรับการส่ง Email เมื่อเอกสารมีการอัปเดตสถานะ

ขั้นตอนที่ 1: ติดตั้ง Flask-Mail

ใน requirements.txt ให้เพิ่ม Flask-Mail:

Flask-Mail

จากนั้นรันคำสั่ง:

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

ขั้นตอนที่ 2: การตั้งค่า Email ใน app.py

ใน app.py เพิ่มการตั้งค่าการใช้งาน Email และฟังก์ชันสำหรับส่ง Email แจ้งเตือน

```
from flask_mail import Mail, Message
```

```
# ตั้งค่า Flask-Mail
```

```
app.config['MAIL_SERVER'] = 'smtp.example.com' # แทนที่ด้วย SMTP Server ของคุณ
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'your_email@example.com' # แทนที่ด้วย Email ของคุณ
app.config['MAIL_PASSWORD'] = 'your_email_password' # แทนที่ด้วยรหัสผ่าน
app.config['MAIL_DEFAULT_SENDER'] = 'your_email@example.com'
```

```
mail = Mail(app)
```

ขั้นตอนที่ 3: ฟังก์ชันสำหรับส่ง Email

เพิ่มฟังก์ชัน send_status_update_email เพื่อส่ง Email แจ้งเตือนเมื่อมีการอัปเดตสถานะเอกสาร

```
def send_status_update_email(doc_id, title, status, receiver_email):
    try:
        msg = Message(
            subject=f"Status Update for Document ID {doc_id}",
            recipients=[receiver_email],
            body=f"Document '{title}' has been updated to status: {status}."
        )
        mail.send(msg)
        print("Email sent successfully!")
    except Exception as e:
        print(f"Failed to send email: {e}")
```

ขั้นตอนที่ 4: อัปเดต update_status เพื่อเรียกใช้การส่ง Email

ในฟังก์ชัน update_status เมื่อมีการอัปเดตสถานะเอกสาร ให้เรียกใช้ send_status_update_email เพื่อแจ้งเตือนผู้รับเอกสาร

```
@app.route('/update_status/<string:doc_id>', methods=['GET', 'POST'])
def update_status(doc_id):
    document = Document.query.filter_by(doc_id=doc_id).first()
    if request.method == 'POST':
        location = request.form['location']
        status = request.form['status']

        # บันทึกสถานะ
        new_status = DocumentStatus(doc_id=doc_id, location=location,
status=status)
        db.session.add(new_status)
        db.session.commit()

        document.status = status
        db.session.commit()

        # เรียกใช้ฟังก์ชันส่ง Email
        receiver_email = "receiver_email@example.com" # ระบุ Email ผู้รับเอกสาร
        send_status_update_email(document.doc_id, document.title, status,
receiver_email)

        flash('Status updated successfully and notification email sent!',
'success')
        return redirect(url_for('update_status', doc_id=doc_id))

    return render_template('update_status.html', document=document)
```

ขั้นตอนที่ 5: สร้างฟังก์ชันค้นหาใน app.py

เพิ่มฟังก์ชัน search_documents เพื่อค้นหาเอกสารโดยใช้ doc_id, title, sender_name, หรือ receiver_name

```
@app.route('/search', methods=['GET', 'POST'])
def search_documents():
    results = []
    if request.method == 'POST':
        query = request.form['query']

        # ค้นหาข้อมูลโดยใช้คำค้นหา
        results = Document.query.filter(
            (Document.doc_id.like(f"%{query}%")) |
            (Document.title.like(f"%{query}%")) |
            (Document.sender_name.like(f"%{query}%")) |
            (Document.receiver_name.like(f"%{query}%"))
```

```

    ).all()

    if not results:
        flash('No documents found.', 'warning')

    return render_template('search.html', results=results)

```

ขั้นตอนที่ 6: สร้าง Template สำหรับการค้นหา search.html

สร้างไฟล์ search.html ในโฟลเดอร์ templates/

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Search Documents</title>
</head>
<body>
    <h1>Search Documents</h1>
    <form method="POST">
        <input type="text" name="query" placeholder="Enter Document ID,
Title, Sender or Receiver Name" required>
        <button type="submit">Search</button>
    </form>

    {% if results %}
        <h2>Search Results</h2>
        <ul>
            {% for document in results %}
                <li>
                    <a href="{{ url_for('check_status',
doc_id=document.doc_id) }}">
                        {{ document.doc_id }} - {{ document.title }} ({{
document.status }})
                    </a>
                </li>
            {% endfor %}
        </ul>
    {% endif %}
</body>
</html>

```

การทดสอบระบบ

- เปิดเบราว์เซอร์และทดสอบฟีเจอร์ใหม่:
 - ค้นหาเอกสาร: <http://localhost:5000/search>
 - เมื่อค้นหาเจอเอกสาร สามารถคลิกลิงก์เพื่อตรวจสอบสถานะเอกสาร
 - เมื่ออัปเดตสถานะใน `/update_status/<doc_id>` ระบบจะส่ง Email แจ้งเตือนไปยังผู้รับเอกสาร

หมายเหตุ

- อย่าลืมเปลี่ยนข้อมูล Email Server, Username, Password และ Email ผู้รับตามจริง
- ไฟเจอร์เหล่านี้สามารถปรับปรุงเพิ่มเติมเพื่อให้รองรับการใช้งานได้จริง

ช่วยเพิ่มส่วนของ / ให้แสดงเป็นเมนู เพื่อใช้งานในส่วนต่าง ๆ โดยใช้ semantic ui ได้หรือไม่ ทั้งนี้ที่เป็น html ทั้งหมดขอให้เป็น semantic ui ด้วย

ได้ครับ ด้านล่างนี้เป็นการปรับปรุงให้หน้าแรก (เส้นทาง /) ของระบบแสดงเป็นเมนูเพื่อให้สามารถเข้าถึงฟังก์ชันต่าง ๆ ของระบบได้ง่ายขึ้น และใช้ **Semantic UI** สำหรับตกแต่ง UI รวมถึงการปรับเปลี่ยน HTML ของแต่ละหน้าด้วย Semantic UI

ขั้นตอนที่ 1: ติดตั้ง Semantic UI

1. เพิ่มลิงก์ Semantic UI CDN ลงใน HTML ของเรา เพื่อให้สามารถใช้งานได้ทันที

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
<script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.js"></script>
```

ขั้นตอนที่ 2: สร้างเมนูหลักในเส้นทาง /

ใน app.py ให้เพิ่มเส้นทาง / เพื่อแสดงเมนู

```
@app.route('/')
def home():
    return render_template('home.html')
```

ขั้นตอนที่ 3: สร้าง Template home.html สำหรับเมนูหลัก

สร้างไฟล์ home.html ในโฟลเดอร์ templates/

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document Tracking System</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
        <h2 class="ui center aligned header">Document Tracking System</h2>

        <div class="ui secondary pointing menu">
            <a class="item" href="{{ url_for('create_document') }}">Create
```

```

Document</a>
    <a class="item" href="{{ url_for('search_documents') }}">Search
Document</a>
    <a class="item" href="{{ url_for('update_status',
doc_id='sample') }}">Update Status</a>
    <a class="item" href="{{ url_for('check_status', doc_id='sample')
}}">Check Status</a>
</div>

<div class="ui segment">
    <h3>Welcome to Document Tracking System</h3>
    <p>Use the menu above to navigate through different
functions.</p>
</div>
</div>
</body>
</html>

```

ขั้นตอนที่ 4: ปรับปรุง Template ของแต่ละหน้าให้เป็น Semantic UI

create_document.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Create Document</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
        <h2 class="ui center aligned header">Create New Document</h2>
        <form class="ui form" method="POST">
            <div class="field">
                <label>Document ID</label>
                <input type="text" name="doc_id" placeholder="Document ID"
required>
            </div>
            <div class="field">
                <label>Title</label>
                <input type="text" name="title" placeholder="Title" required>
            </div>
            <div class="field">
                <label>Sender Name</label>
                <input type="text" name="sender_name" placeholder="Sender
Name" required>
            </div>
            <div class="field">
                <label>Receiver Name</label>

```

```

                <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
            </div>
            <button class="ui button primary" type="submit">Create</button>
        </form>
    </div>
</body>
</html>

```

update_status.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Update Status</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
        <h2 class="ui center aligned header">Update Status for Document {{
document.doc_id }}</h2>
        <form class="ui form" method="POST">
            <div class="field">
                <label>Location</label>
                <input type="text" name="location" placeholder="Location"
required>
            </div>
            <div class="field">
                <label>Status</label>
                <input type="text" name="status" placeholder="Status"
required>
            </div>
            <button class="ui button primary" type="submit">Update</button>
        </form>
    </div>
</body>
</html>

```

check_status.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Check Status</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>

```

```

<body>
  <div class="ui container">
    <h2 class="ui center aligned header">Status for Document {{
document.doc_id }}</h2>
    <div class="ui segment">
      <p><strong>Title:</strong> {{ document.title }}</p>
      <p><strong>Sender:</strong> {{ document.sender_name }}</p>
      <p><strong>Receiver:</strong> {{ document.receiver_name }}</p>
    </div>
    <h3>Statuses</h3>
    <div class="ui list">
      {% for status in statuses %}
        <div class="item">
          <i class="marker icon"></i>
          <div class="content">
            <div class="header">{{ status.location }}</div>
            <div class="description">{{ status.status }} - {{
status.updated_at }}</div>
          </div>
        </div>
      {% endfor %}
    </div>
  </div>
</body>
</html>

```

search.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Search Documents</title>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
  <div class="ui container">
    <h2 class="ui center aligned header">Search Documents</h2>
    <form class="ui form" method="POST">
      <div class="field">
        <input type="text" name="query" placeholder="Enter Document
ID, Title, Sender or Receiver Name" required>
      </div>
      <button class="ui button primary" type="submit">Search</button>
    </form>

    {% if results %}
      <h3>Search Results</h3>
      <div class="ui list">

```

```

        {% for document in results %}
            <div class="item">
                <i class="file icon"></i>
                <div class="content">
                    <a class="header" href="{{
url_for('check_status', doc_id=document.doc_id) }}">
                        {{ document.doc_id }} - {{ document.title }}
                    </a>
                    <div class="description">{{ document.status
}}</div>
                </div>
            </div>
        {% endfor %}
    </div>
</body>
</html>

```

ทดสอบการใช้งาน

- เปิดเบราว์เซอร์และเข้าที่ <http://localhost:5000/> คุณจะเห็นเมนูที่สามารถเข้าถึงฟังก์ชันต่าง ๆ ของระบบ

ต้องการให้ทุก html มีเมนูด้วยครับ

ได้ครับ ผมจะแก้ไขทุกไฟล์ HTML ให้มีเมนูร่วมด้วย เพื่อให้แต่ละหน้าสามารถเข้าถึงฟังก์ชันต่าง ๆ ของระบบได้ง่ายขึ้น
โดยจะสร้างเมนูที่มีลักษณะเหมือนกับหน้าแรก (home.html) ในทุกหน้าครับ

ปรับโครงสร้างให้ทุก Template มีเมนูร่วมกัน

เราจะสร้างเมนูในรูปแบบที่สามารถเรียกใช้ซ้ำได้โดยใช้ **Jinja2** ในการรวมไฟล์เมนูเข้าไปในทุกหน้า HTML ผ่านการใช้ `include` และสร้างเมนูเป็นไฟล์ `menu.html`

ขั้นตอนที่ 1: สร้างไฟล์ `menu.html`

สร้างไฟล์ `menu.html` ในโฟลเดอร์ `templates/` โดยไฟล์นี้จะมีแค่เมนูเท่านั้น และเราจะทำการรวมเข้าไปในทุกไฟล์ HTML อื่น ๆ

menu.html

```

<div class="ui secondary pointing menu">
    <a class="item" href="{{ url_for('home') }}">Home</a>
    <a class="item" href="{{ url_for('create_document') }}">Create
Document</a>
    <a class="item" href="{{ url_for('search_documents') }}">Search
Document</a>
    <a class="item" href="{{ url_for('update_status', doc_id='sample')
}}">Update Status</a>
    <a class="item" href="{{ url_for('check_status', doc_id='sample')
}}">Check Status</a>
</div>

```

ขั้นตอนที่ 2: ปรับไฟล์ HTML แต่ละไฟล์ให้ใช้เมนูร่วมกัน

โดยเราจะเพิ่มการ include ไฟล์ menu.html เข้าไปในแต่ละ Template ด้านล่างนี้

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document Tracking System</title>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
  <div class="ui container">
    <h2 class="ui center aligned header">Document Tracking System</h2>

    {% include 'menu.html' %}

    <div class="ui segment">
      <h3>Welcome to Document Tracking System</h3>
      <p>Use the menu above to navigate through different
functions.</p>
    </div>
  </div>
</body>
</html>
```

create_document.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Create Document</title>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
  <div class="ui container">
    <h2 class="ui center aligned header">Create New Document</h2>

    {% include 'menu.html' %}

    <form class="ui form" method="POST">
      <div class="field">
        <label>Document ID</label>
        <input type="text" name="doc_id" placeholder="Document ID">
      </div>
    </form>
  </div>
</body>
</html>
```

```

required>
    </div>
    <div class="field">
        <label>Title</label>
        <input type="text" name="title" placeholder="Title" required>
    </div>
    <div class="field">
        <label>Sender Name</label>
        <input type="text" name="sender_name" placeholder="Sender
Name" required>
    </div>
    <div class="field">
        <label>Receiver Name</label>
        <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
    </div>
    <button class="ui button primary" type="submit">Create</button>
</form>
</div>
</body>
</html>

```

update_status.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Update Status</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
        <h2 class="ui center aligned header">Update Status for Document {{
document.doc_id }}</h2>

        {% include 'menu.html' %}

        <form class="ui form" method="POST">
            <div class="field">
                <label>Location</label>
                <input type="text" name="location" placeholder="Location"
required>
            </div>
            <div class="field">
                <label>Status</label>
                <input type="text" name="status" placeholder="Status"
required>
            </div>

```

```

        <button class="ui button primary" type="submit">Update</button>
    </form>
</div>
</body>
</html>

```

check_status.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Check Status</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
        <h2 class="ui center aligned header">Status for Document {{
document.doc_id }}</h2>

        {% include 'menu.html' %}

        <div class="ui segment">
            <p><strong>Title:</strong> {{ document.title }}</p>
            <p><strong>Sender:</strong> {{ document.sender_name }}</p>
            <p><strong>Receiver:</strong> {{ document.receiver_name }}</p>
        </div>
        <h3>Statuses</h3>
        <div class="ui list">
            {% for status in statuses %}
                <div class="item">
                    <i class="marker icon"></i>
                    <div class="content">
                        <div class="header">{{ status.location }}</div>
                        <div class="description">{{ status.status }} - {{
status.updated_at }}</div>
                    </div>
                </div>
            {% endfor %}
        </div>
    </div>
</body>
</html>

```

search.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

```



```

<title>Search Documents</title>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
  <div class="ui container">
    <h2 class="ui center aligned header">Search Documents</h2>

    {% include 'menu.html' %}

    <form class="ui form" method="POST">
      <div class="field">
        <input type="text" name="query" placeholder="Enter Document
ID, Title, Sender or Receiver Name" required>
      </div>
      <button class="ui button primary" type="submit">Search</button>
    </form>

    {% if results %}
      <h3>Search Results</h3>
      <div class="ui list">
        {% for document in results %}
          <div class="item">
            <i class="file icon"></i>
            <div class="content">
              <a class="header" href="{{
url_for('check_status', doc_id=document.doc_id) }}">
                {{ document.doc_id }} - {{ document.title }}
              </a>
              <div class="description">{{ document.status
}}</div>
            </div>
          </div>
        {% endfor %}
      </div>
    {% endif %}
  </div>
</body>
</html>

```

สรุป

ตอนนี้ทุกหน้า HTML ได้มีเมนูหลักที่สามารถเข้าถึงฟังก์ชันต่าง ๆ ได้โดยใช้ **Semantic UI** สำหรับการจัดสไตล์

การ **update status** ต้องให้ผู้ใช้ระบุ **doc id** ก่อน หรือให้ผู้ใช้ค้นหา **doc id** ที่ต้องการ แล้วเลือกรายการนั้น เพื่อทำการปรับปรุงสถานะ

เพื่อปรับให้การอัปเดตสถานะเอกสารเริ่มจากการค้นหา **doc_id** ที่ต้องการและเลือกเอกสารนั้น ก่อนที่จะไปสู่หน้าสำหรับอัปเดตสถานะ เราสามารถปรับปรุงระบบให้ผู้ใช้ค้นหา ก่อน แล้วให้ระบบแสดงลิงก์หรือปุ่มเพื่อไปยังหน้าการอัปเดตสถานะตามเอกสารที่เลือก

ขั้นตอนที่ 1: ปรับปรุงการค้นหาให้มีลิงก์ไปยังการอัปเดตสถานะ

ในฟังก์ชัน `search_documents` ที่มีอยู่แล้ว ให้เราปรับปรุง `search.html` เพื่อให้ผู้ใช้สามารถเลือกเอกสารที่ต้องการและไปยังหน้าการอัปเดตสถานะได้

ปรับปรุง `search.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Search Documents</title>
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
  <div class="ui container">
    <h2 class="ui center aligned header">Search Documents</h2>

    {% include 'menu.html' %}

    <form class="ui form" method="POST">
      <div class="field">
        <input type="text" name="query" placeholder="Enter Document
ID, Title, Sender or Receiver Name" required>
      </div>
      <button class="ui button primary" type="submit">Search</button>
    </form>

    {% if results %}
      <h3>Search Results</h3>
      <div class="ui list">
        {% for document in results %}
          <div class="item">
            <i class="file icon"></i>
            <div class="content">
              <a class="header" href="{{
url_for('check_status', doc_id=document.doc_id) }}">
                {{ document.doc_id }} - {{ document.title }}
              </a>
              <div class="description">{{ document.status
}}</div>
              <a class="ui button" href="{{
url_for('update_status', doc_id=document.doc_id) }}">Update Status</a>
            </div>
          </div>
        {% endfor %}
      </div>
    {% endif %}
```

```
</div>
</body>
</html>
```

ในหน้านี้ หลังจากการค้นหาเอกสาร ผู้ใช้สามารถคลิกปุ่ม “Update Status”
ข้างแต่ละเอกสารที่พบเพื่อไปยังหน้าการอัปเดตสถานะของเอกสารนั้น ๆ ได้

ขั้นตอนที่ 2: ปรับปรุงฟังก์ชัน `update_status` ให้รองรับการรับ `doc_id` จาก URL

ฟังก์ชัน `update_status` ใน `app.py` ถูกออกแบบมาให้ใช้ `doc_id` จาก URL แล้ว
ดังนั้นจึงไม่ต้องการเปลี่ยนแปลงเพิ่มเติมในส่วนนี้ ผู้ใช้สามารถคลิกจากหน้าค้นหาเอกสารเพื่อไปยังหน้าการอัปเดตสถานะได้โดยตรง

การทดสอบ

1. ผู้ใช้ไปที่หน้า <http://localhost:5000/search> เพื่อค้นหาเอกสารที่ต้องการ
2. เมื่อค้นหาเจอเอกสาร จะมีปุ่ม “Update Status” ให้คลิกข้างเอกสารนั้น
3. ผู้ใช้จะถูกนำไปยังหน้าการอัปเดตสถานะ (`/update_status/<doc_id>`) โดยไม่ต้องกรอก `doc_id` เอง

http://127.0.0.1:5000/update_status/

ให้ `default` เป็น การระบุ `doc id` เพื่อค้นหาถ้าเจอก็ให้ผู้ใช้บอ location และ `status` ต่อไปหรือไม่

ได้ครับ เราสามารถปรับเส้นทาง `/update_status/` ให้เริ่มต้นด้วยการค้นหา `doc_id` ก่อน และเมื่อผู้ใช้กรอก `doc_id` แล้ว
ระบบจะตรวจสอบว่ามีเอกสารอยู่ในระบบหรือไม่ ถ้าพบเอกสาร ระบบจะแสดงฟอร์มให้ผู้ใช้กรอก `location` และ `status`
เพื่ออัปเดตสถานะของเอกสาร

ขั้นตอนที่ 1: ปรับฟังก์ชัน `update_status` ใน `app.py`

เราจะปรับฟังก์ชัน `update_status` ให้รองรับการกรอก `doc_id` ก่อน จากนั้นจะแสดงฟอร์มสำหรับกรอก `location` และ `status` เฉพาะเมื่อ `doc_id` นั้นมีอยู่ในระบบ

```
@app.route('/update_status', methods=['GET', 'POST'])
def update_status():
    document = None
    if request.method == 'POST':
        doc_id = request.form['doc_id']

        # ค้นหาเอกสารจาก doc_id
        document = Document.query.filter_by(doc_id=doc_id).first()

    if document:
        # ถ้ามีเอกสาร ให้แสดงฟอร์มสำหรับกรอก location และ status
        if 'location' in request.form and 'status' in request.form:
            location = request.form['location']
            status = request.form['status']

            # บันทึกสถานะใหม่
            new_status = DocumentStatus(doc_id=doc_id, location=location,
status=status)
            db.session.add(new_status)
```

```

        db.session.commit()

        document.status = status
        db.session.commit()

        flash('Status updated successfully!', 'success')
        return redirect(url_for('update_status'))
    else:
        flash('Document ID not found. Please try again.', 'error')

    return render_template('update_status.html', document=document)

```

ขั้นตอนที่ 2: สร้าง Template `update_status.html` สำหรับค้นหาและอัปเดตสถานะ

ในไฟล์ `update_status.html` เราจะทำให้มีฟอร์มสำหรับการค้นหา `doc_id` และเมื่อพบเอกสาร ระบบจะแสดงฟอร์มให้ผู้ใช้กรอก `location` และ `status`

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Update Status</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
        <h2 class="ui center aligned header">Update Document Status</h2>

        {% include 'menu.html' %}

        <form class="ui form" method="POST">
            <div class="field">
                <label>Document ID</label>
                <input type="text" name="doc_id" placeholder="Enter Document
ID" required>
            </div>

            {% if document %}
                <div class="field">
                    <label>Location</label>
                    <input type="text" name="location" placeholder="Enter
Location" required>
                </div>
                <div class="field">
                    <label>Status</label>
                    <input type="text" name="status" placeholder="Enter
Status" required>

```

```

        </div>
        <button class="ui button primary" type="submit">Update
Status</button>
    {% else %}
        <button class="ui button primary" type="submit">Search
Document</button>
    {% endif %}
</form>

    {% if document %}
        <div class="ui message">
            <p>Document ID: {{ document.doc_id }}</p>
            <p>Title: {{ document.title }}</p>
            <p>Current Status: {{ document.status }}</p>
        </div>
    {% endif %}
</div>
</body>
</html>

```

การทำงานของระบบ

1. เมื่อผู้ใช้เข้าไปที่ http://127.0.0.1:5000/update_status จะเห็นฟอร์มสำหรับกรอก doc_id
2. เมื่อกรอก doc_id และกดปุ่ม “Search Document” ระบบจะตรวจสอบว่า doc_id มีอยู่ในระบบหรือไม่
 - ถ้าไม่พบ doc_id จะมีข้อความแจ้งว่าไม่พบเอกสาร
 - ถ้าพบเอกสาร จะมีฟอร์มให้กรอก location และ status เพิ่มเติม
3. เมื่อผู้ใช้กรอกข้อมูลครบและกด “Update Status” ระบบจะบันทึกสถานะใหม่ของเอกสารและรีเฟรชหน้า

ต้องการให้ส่วนของ **check status** เป็นเช่นเดียวกัน โดยให้มีช่องค้นหา กรณีเริ่มต้นก็ให้แสดงรายการเอกสารเรียงจากล่าสุดขึ้นมาก่อน ถ้ามีจำนวนมาก ให้สร้างเป็นหลายหน้าให้ผู้ใช้เลือก โดยเริ่มต้นที่ 50 รายการต่อหน้า

เพื่อให้ส่วนของ **Check Status** มีช่องค้นหา doc_id และเริ่มต้นด้วยการแสดงรายการเอกสารล่าสุด โดยแบ่งเป็นหลายหน้า (pagination) ตามที่คุณต้องการ เราจะใช้การค้นหาเอกสารพร้อมกับการแบ่งหน้า โดยกำหนดให้แสดง 50 รายการต่อหน้า

ขั้นตอนที่ 1: ปรับฟังก์ชัน **check_status** ใน **app.py**

เราจะเพิ่มการแบ่งหน้าและการค้นหาในฟังก์ชัน **check_status** โดยใช้ค่าเริ่มต้น 50 รายการต่อหน้า และรองรับการค้นหาเอกสารตาม doc_id

```

@app.route('/check_status', methods=['GET', 'POST'])
def check_status():
    search_query = None
    page = request.args.get('page', 1, type=int)
    per_page = 50

    if request.method == 'POST':
        search_query = request.form['query']

    if search_query:

```

```

        # ค้นหาเอกสารที่ตรงกับ query
        documents =
Document.query.filter(Document.doc_id.like(f"%{search_query}%")).paginate(pag
e=page, per_page=per_page)
    else:
        # แสดงรายการเอกสารล่าสุดทั้งหมด แบ่งเป็นหน้า
        documents =
Document.query.order_by(Document.created_at.desc()).paginate(page=page,
per_page=per_page)

    return render_template('check_status.html', documents=documents,
search_query=search_query)

```

ขั้นตอนที่ 2: สร้าง Template check_status.html พร้อมการแบ่งหน้าและค้นหา

ปรับแต่ง check_status.html ให้มีฟอร์มสำหรับค้นหา และแสดงรายการเอกสารล่าสุดพร้อมแบ่งหน้า (pagination)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Check Document Status</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
    <div class="ui container">
        <h2 class="ui center aligned header">Check Document Status</h2>

        {% include 'menu.html' %}

        <form class="ui form" method="POST">
            <div class="field">
                <input type="text" name="query" placeholder="Enter Document
ID to search" value="{{ search_query or '' }}">
            </div>
            <button class="ui button primary" type="submit">Search</button>
        </form>

        {% if documents.items %}
            <h3>Document List</h3>
            <div class="ui relaxed divided list">
                {% for document in documents.items %}
                    <div class="item">
                        <i class="file outline icon"></i>
                        <div class="content">
                            <a class="header" href="{{
url_for('check_status', doc_id=document.doc_id) }}">{{ document.doc_id }} -
{{ document.title }}</a>

```

```

        <div class="description">Status: {{
document.status }} | Created at: {{ document.created_at }}</div>
    </div>
</div>
{% endfor %}
</div>

<div class="ui pagination menu">
    {% if documents.has_prev %}
        <a class="item" href="{{ url_for('check_status',
page=documents.prev_num) }}">Previous</a>
    {% else %}
        <div class="disabled item">Previous</div>
    {% endif %}

    {% for page_num in documents.iter_pages(left_edge=1,
right_edge=1, left_current=2, right_current=2) %}
        {% if page_num %}
            {% if page_num == documents.page %}
                <div class="active item">{{ page_num }}</div>
            {% else %}
                <a class="item" href="{{ url_for('check_status',
page=page_num) }}">{{ page_num }}</a>
            {% endif %}
        {% else %}
            <div class="disabled item">...</div>
        {% endif %}
    {% endfor %}

    {% if documents.has_next %}
        <a class="item" href="{{ url_for('check_status',
page=documents.next_num) }}">Next</a>
    {% else %}
        <div class="disabled item">Next</div>
    {% endif %}
</div>
{% else %}
    <div class="ui message">
        <p>No documents found.</p>
    </div>
{% endif %}
</div>
</body>
</html>

```

การทำงานของระบบ

1. เมื่อผู้ใช้เข้าไปที่ http://127.0.0.1:5000/check_status ระบบจะแสดงรายการเอกสารล่าสุดเรียงจากใหม่ไปเก่า โดยแสดงหน้าแรก 50 รายการ
2. ผู้ใช้สามารถค้นหา doc_id ที่ต้องการในช่องค้นหา และกดปุ่ม “Search”

- หากพบเอกสาร ระบบจะแสดงรายการที่ค้นพบเท่านั้น
- หากไม่พบเอกสาร จะมีข้อความแจ้งเตือนว่าไม่พบเอกสาร

3. ระบบจะมีการแบ่งหน้า (pagination) ให้ผู้ใช้เลือกไปยังหน้าถัดไปหรือหน้าก่อนหน้า

ในกรณีที่เกิดปุ่ม update status จากการค้นหา ซึ่งมีการเรียก

`http://127.0.0.1:5000/update_status?doc_id=001`

จะต้องแก้ในนี้ อย่างไร ให้ทำการ update_status ด้วยถ้ามี doc id

```
@app.route('/update_status', methods=['GET', 'POST'])
def update_status():
    document = None
    if request.method == 'POST':
        doc_id = request.form['doc_id']

        # ค้นหาเอกสารจาก doc_id
        document = Document.query.filter_by(doc_id=doc_id).first()

        if document:
            # ถ้ามีเอกสาร ให้แสดงฟอร์มสำหรับกรอก location และ status
            if 'location' in request.form and 'status' in request.form:
                location = request.form['location']
                status = request.form['status']

                # บันทึกสถานะใหม่
                new_status = DocumentStatus(doc_id=doc_id, location=location,
status=status)
                db.session.add(new_status)
                db.session.commit()

                document.status = status
                db.session.commit()

                flash('Status updated successfully!', 'success')
                return redirect(url_for('update_status'))
            else:
                flash('Document ID not found. Please try again.', 'error')

        return render_template('update_status.html', document=document)
    ````
```

เราสามารถแก้ไขฟังก์ชัน `update\_status` ให้รองรับการตรวจสอบ `doc\_id` ที่ถูกส่งมาในพารามิเตอร์ของ URL (เช่น `http://127.0.0.1:5000/update\_status?doc\_id=001`) โดยเพิ่มการตรวจสอบ `request.args.get('doc\_id')` ในกรณีที่ เป็น `GET` request และทำให้ระบบแสดงข้อมูลเอกสารอัตโนมัติหากมี `doc\_id` ส่งมาใน URL

### ปรับโค้ดฟังก์ชัน `update\_status`



```

python
@app.route('/update_status', methods=['GET', 'POST'])
def update_status():
 document = None
 doc_id = request.args.get('doc_id') # รับค่า doc_id จาก query string

 # ถ้ามี doc_id ใน query string ให้ค้นหาข้อมูลเอกสารโดยอัตโนมัติ
 if doc_id:
 document = Document.query.filter_by(doc_id=doc_id).first()

 if request.method == 'POST':
 # กรณีผู้ใช้กรอก doc_id ในฟอร์มค้นหา
 doc_id = request.form.get('doc_id', doc_id) # ใช้ doc_id จากฟอร์มหรือ
 query string
 document = Document.query.filter_by(doc_id=doc_id).first()

 if document:
 # ถ้ามีเอกสาร และมีข้อมูล location และ status ให้บันทึกสถานะใหม่
 location = request.form.get('location')
 status = request.form.get('status')

 if location and status:
 # บันทึกสถานะใหม่
 new_status = DocumentStatus(doc_id=doc_id, location=location,
status=status)
 db.session.add(new_status)
 db.session.commit()

 document.status = status
 db.session.commit()

 flash('Status updated successfully!', 'success')
 return redirect(url_for('update_status', doc_id=doc_id))
 else:
 flash('Document ID not found. Please try again.', 'error')

 return render_template('update_status.html', document=document)

```

#### การทำงานของระบบ

1. ถ้าผู้ใช้เข้าที่ URL แบบมี doc\_id (เช่น `http://127.0.0.1:5000/update_status?doc_id=001`), ระบบจะใช้ doc\_id ที่รับจาก query string ในการค้นหาเอกสารโดยอัตโนมัติ
2. ถ้าพบเอกสาร ระบบจะแสดงฟอร์มสำหรับกรอก location และ status ให้ผู้ใช้กรอกข้อมูลเพื่ออัปเดตสถานะ
3. ถ้าไม่มี doc\_id หรือผู้ใช้ต้องการค้นหาเอกสารใหม่ สามารถกรอก doc\_id ในฟอร์มค้นหาได้

กรณีที่พบเอกสารให้แสดงค่า doc id ในช่องข้อความ `document.doc_id` ได้หรือไม่

ได้ครับ เราสามารถแสดงค่า doc\_id ในช่องข้อความของฟอร์มได้ในกรณีที่พบเอกสาร โดยจะเติมค่า doc\_id ในฟิลด์ของฟอร์มให้ผู้ใช้เห็นและไม่ต้องกรอกใหม่

ปรับปรุง update\_status.html เพื่อแสดงค่า doc\_id ในช่องข้อความ

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Update Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Update Document Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <label>Document ID</label>
 <input type="text" name="doc_id" placeholder="Enter Document
ID" value="{{ document.doc_id if document else '' }}" required>
 </div>

 {% if document %}
 <div class="field">
 <label>Location</label>
 <input type="text" name="location" placeholder="Enter
Location" required>
 </div>
 <div class="field">
 <label>Status</label>
 <input type="text" name="status" placeholder="Enter
Status" required>
 </div>
 <button class="ui button primary" type="submit">Update
Status</button>
 {% else %}
 <button class="ui button primary" type="submit">Search
Document</button>
 {% endif %}
 </form>

 {% if document %}
 <div class="ui message">
 <p>Document ID: {{ document.doc_id }}</p>
 <p>Title: {{ document.title }}</p>
 <p>Current Status: {{ document.status }}</p>
 </div>
 {% endif %}
```

```
</div>
</body>
</html>
```

#### การทำงานของระบบ

- เมื่อพบเอกสารจากการค้นหา ค่า `doc_id` จะถูกแสดงในฟิลด์ `Document ID` โดยอัตโนมัติ
- ถ้าไม่มีเอกสาร หรือผู้ใช้ต้องการค้นหาเอกสารใหม่ สามารถแก้ไข `doc_id` ในช่องข้อความได้

การแสดงความสถานะ ให้แสดงชื่อคนส่ง ชื่อผู้รับ และรายการเอกสารด้วย

ในกรณีที่การส่งเอกสาร 1 ครั้ง มีหลายเอกสารจะต้องเพิ่มเติมฐานข้อมูลอย่างไร

เพื่อรองรับการส่งเอกสารหลายรายการในการส่งครั้งเดียว

เราจำเป็นต้องปรับโครงสร้างฐานข้อมูลเพื่อให้สามารถจัดเก็บข้อมูลเอกสารหลายรายการที่เชื่อมโยงกับการส่งเอกสารครั้งเดียวได้

โดยสร้างความสัมพันธ์ระหว่างตาราง `Document` กับตารางใหม่สำหรับการส่งเอกสาร ซึ่งเราจะใช้โครงสร้างฐานข้อมูลแบบ **one-to-many** ดังนี้:

#### การออกแบบฐานข้อมูลใหม่

1. สร้างตารางใหม่ชื่อ **DocumentBatch** สำหรับเก็บข้อมูลการส่งเอกสารแต่ละครั้ง ซึ่งจะรวมถึงชื่อผู้ส่งและผู้รับ
2. แก้ไขตาราง **Document** เพื่อเชื่อมโยงเอกสารแต่ละรายการกับ `DocumentBatch` ด้วย `batch_id`

#### โครงสร้างฐานข้อมูล

- **DocumentBatch** (ตารางใหม่)
  - `batch_id`: รหัสของการส่งเอกสาร (Primary Key)
  - `sender_name`: ชื่อผู้ส่งเอกสาร
  - `receiver_name`: ชื่อผู้รับเอกสาร
  - `created_at`: วันและเวลาที่เริ่มการส่งเอกสาร
- **Document** (แก้ไขโครงสร้าง)
  - เพิ่ม `batch_id` เพื่อเชื่อมโยงกับ `DocumentBatch`

#### ตัวอย่างโค้ดการสร้างโมเดล

ในไฟล์ `models.py` ให้เพิ่มโมเดล `DocumentBatch` และอัปเดต `Document` ให้เชื่อมโยงกับ `DocumentBatch`

```
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
```

```
db = SQLAlchemy()
```

```
class DocumentBatch(db.Model):
 __tablename__ = 'document_batch'
 batch_id = db.Column(db.Integer, primary_key=True)
 sender_name = db.Column(db.String(100), nullable=False)
 receiver_name = db.Column(db.String(100), nullable=False)
 created_at = db.Column(db.DateTime, default=datetime.utcnow())
```

```
ความสัมพันธ์กับเอกสาร
```

```
documents = db.relationship('Document', backref='batch', lazy=True)
```

```
class Document(db.Model):
 __tablename__ = 'document'
 id = db.Column(db.Integer, primary_key=True)
 doc_id = db.Column(db.String(50), unique=True, nullable=False)
 title = db.Column(db.String(100), nullable=False)
 created_at = db.Column(db.DateTime, default=datetime.utcnow)
 status = db.Column(db.String(50), default="Created")

 # เชื่อมโยงกับ DocumentBatch
 batch_id = db.Column(db.Integer,
db.ForeignKey('document_batch.batch_id'), nullable=False)

class DocumentStatus(db.Model):
 __tablename__ = 'document_status'
 id = db.Column(db.Integer, primary_key=True)
 doc_id = db.Column(db.String(50), db.ForeignKey('document.doc_id'),
nullable=False)
 location = db.Column(db.String(50), nullable=False)
 status = db.Column(db.String(50), nullable=False)
 updated_at = db.Column(db.DateTime, default=datetime.utcnow)
```

ตัวอย่างการใช้งาน

1. สร้างการส่งเอกสารใหม่ โดยมีเอกสารหลายรายการที่เชื่อมโยงกัน

```
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 documents = request.form.getlist('documents') # รายชื่อเอกสารเป็น list

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(sender_name=sender_name,
receiver_name=receiver_name)
 db.session.add(batch)
 db.session.commit()

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for doc in documents:
 new_document = Document(doc_id=doc['doc_id'], title=doc['title'],
batch_id=batch.batch_id)
 db.session.add(new_document)

 db.session.commit()
 flash('Document batch created successfully!', 'success')
 return redirect(url_for('create_batch'))
```

```
return render_template('create_batch.html')
```

แก้ไข `check_status.html` ให้แสดงชื่อผู้ส่ง ชื่อผู้รับ และรายการเอกสารในกรณีที่มีการส่งหลายเอกสาร

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Check Document Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Check Document Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Document
ID to search" value="{{ search_query or '' }}">
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if documents.items %}
 <h3>Document List</h3>
 <div class="ui relaxed divided list">
 {% for document in documents.items %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <a class="header" href="{{
url_for('check_status', doc_id=document.doc_id) }}">{{ document.doc_id }} -
{{ document.title }}
 <div class="description">Status: {{
document.status }} | Created at: {{ document.created_at }}</div>
 <div class="description">Sender: {{
document.batch.sender_name }} | Receiver: {{ document.batch.receiver_name
}}</div>
 </div>
 </div>
 {% endfor %}
 </div>

 <!-- Pagination -->
 <div class="ui pagination menu">
```

```

 {% if documents.has_prev %}
 <a class="item" href="{% url_for('check_status',
page=documents.prev_num) %}">Previous
 {% else %}
 <div class="disabled item">Previous</div>
 {% endif %}

 {% for page_num in documents.iter_pages(left_edge=1,
right_edge=1, left_current=2, right_current=2) %}
 {% if page_num %}
 {% if page_num == documents.page %}
 <div class="active item">{{ page_num }}</div>
 {% else %}
 <a class="item" href="{% url_for('check_status',
page=page_num) %}">{{ page_num }}
 {% endif %}
 {% else %}
 <div class="disabled item">...</div>
 {% endif %}
 {% endfor %}

 {% if documents.has_next %}
 <a class="item" href="{% url_for('check_status',
page=documents.next_num) %}">Next
 {% else %}
 <div class="disabled item">Next</div>
 {% endif %}
 </div>
{% else %}
 <div class="ui message">
 <p>No documents found.</p>
 </div>
{% endif %}
</div>
</body>
</html>

```

สรุป

- **DocumentBatch** เก็บข้อมูลการส่งเอกสารและผู้ส่ง ผู้รับ
- **Document** เก็บข้อมูลเอกสารแต่ละรายการ โดยมี **batch\_id** เชื่อมโยงกับการส่ง
- ระบบ **check\_status** จะแสดงข้อมูลการส่งเอกสารทั้งหมดรวมถึงชื่อผู้ส่ง ผู้รับ และรายการเอกสาร

/create\_document ต้องใช้ข้อมูล

เมื่อมีการสร้างตาราง DocumentBatch เพื่อรองรับการส่งเอกสารหลายรายการในครั้งเดียว หน้าสำหรับสร้างเอกสาร (/create\_document) จึงไม่จำเป็นต้องใช้แยกต่างหากอีกต่อไป เพราะเราสามารถรวมการสร้างเอกสารแต่ละรายการภายในฟังก์ชัน /create\_batch ซึ่งจะทำหน้าที่สร้างการส่งเอกสารและเพิ่มเอกสารที่เกี่ยวข้องในคราวเดียวกันได้

การปรับฟังก์ชัน `/create_batch` ให้รองรับการสร้างเอกสารหลายรายการ

เราสามารถลบ `/create_document` ออก แล้วใช้ `/create_batch`

เพื่อสร้างการส่งเอกสารพร้อมกับรายการเอกสารทั้งหมดในครั้งเดียว โดยให้ผู้ใช้สามารถเพิ่มหลายเอกสารในแบบฟอร์มเดียว

ตัวอย่างฟังก์ชัน `/create_batch`

```
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายชื่อเอกสารที่เป็น List

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(sender_name=sender_name,
 receiver_name=receiver_name)
 db.session.add(batch)
 db.session.commit()

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for title in document_titles:
 new_document = Document(doc_id=str(uuid4()), title=title,
 batch_id=batch.batch_id)
 db.session.add(new_document)

 db.session.commit()
 flash('Document batch created successfully!', 'success')
 return redirect(url_for('create_batch'))

 return render_template('create_batch.html')
```

ตัวอย่าง Template `create_batch.html`

ไฟล์ `create_batch.html` จะช่วยให้ผู้ใช้สามารถกรอกชื่อผู้ส่ง ผู้รับ และเพิ่มรายการเอกสารที่ต้องการส่ง

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>
```

```

{% include 'menu.html' %}

<form class="ui form" method="POST">
 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" placeholder="Sender
Name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
 </div>

 <h4>Documents</h4>
 <div id="document-list">
 <div class="field">
 <label>Document Title</label>
 <input type="text" name="document_titles"
placeholder="Document Title" required>
 </div>
 </div>

 <button type="button" class="ui button"
onclick="addDocumentField()">Add Another Document</button>
 <button type="submit" class="ui button primary">Create
Batch</button>
</form>
</div>

<script>
function addDocumentField() {
 const documentList = document.getElementById('document-list');
 const newField = document.createElement('div');
 newField.classList.add('field');
 newField.innerHTML = '<label>Document Title</label><input
type="text" name="document_titles" placeholder="Document Title" required>';
 documentList.appendChild(newField);
}
</script>
</body>
</html>

```

การทำงานของระบบ

1. หน้านี้จะรับชื่อผู้ส่ง ชื่อผู้รับ และชื่อเอกสารหลายรายการ
2. ผู้ใช้สามารถกดปุ่ม “Add Another Document” เพื่อเพิ่มฟิลด์กรอกชื่อเอกสารหลายรายการในหน้าเดียว
3. เมื่อกดปุ่ม “Create Batch” ระบบจะสร้างการส่งเอกสารและรายการเอกสารทั้งหมดในครั้งเดียว



ต้องปรับ menu.html ใหม่ใหม่

อันนี้ของเดิม

```
<div class="ui secondary pointing menu">
 Home
 Create
Document
 Search
Document
 Update
Status
 Check
Status
</div>
```

เนื่องจากเราได้รวมฟังก์ชันการสร้างเอกสาร (/create\_document) เข้ากับฟังก์ชันการสร้างการส่งเอกสารหลายรายการ (/create\_batch) ดังนั้นเราจึงต้องปรับลิงก์ในเมนูให้ชี้ไปยัง /create\_batch แทน create\_document

ปรับปรุง menu.html

```
<div class="ui secondary pointing menu">
 Home
 Create Document
Batch
 Search
Document
 Update Status
 Check Status
</div>
```

คำอธิบาย

- แก้ไขลิงก์จาก create\_document เป็น create\_batch และเปลี่ยนข้อความเป็น “Create Document Batch” เพื่อสะท้อนถึงฟังก์ชันใหม่ในการสร้างการส่งเอกสารที่รองรับหลายรายการ
- ลิงก์อื่น ๆ ยังคงเหมือนเดิม

ทำไม doc\_id หายไป doc\_id หมายถึงเลขของใบนำส่ง (ใบนำส่งมีเอกสารหลายรายการได้)

การสร้าง document batch ต้องแสดง doc id และ เลขรายการเอกสารด้วย

เพื่อให้ฟังก์ชันการสร้าง Document Batch แสดง doc\_id ซึ่งเป็นเลขใบนำส่ง (ที่สามารถมีเอกสารหลายรายการได้) เราจะแก้ไขฟังก์ชัน create\_batch ให้สร้าง doc\_id อัตโนมัติเมื่อเริ่มการส่งเอกสาร และแสดง doc\_id พร้อมกับรายการเอกสารทั้งหมดในฟอร์มสร้างเอกสาร

ปรับปรุงฟังก์ชัน create\_batch ใน app.py

1. สร้าง doc\_id สำหรับใบนำส่ง
2. แสดง doc\_id ในฟอร์ม create\_batch.html พร้อมให้ผู้กรอกรายการเอกสารแต่ละรายการ

```
from uuid import uuid4
```

```

@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 doc_id = str(uuid4()).split('-')[0] # สร้าง doc_id แบบสุ่มเพื่อใช้เป็นเลขใบนำส่ง
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายชื่อเอกสารที่เป็น List

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name)
 db.session.add(batch)
 db.session.commit()

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for index, title in enumerate(document_titles, start=1):
 new_document = Document(
 doc_id=f"{doc_id}-{index}", # กำหนดเลขรายการเอกสารตามรูปแบบ doc_id
 title=title,
 batch_id=batch.batch_id
)
 db.session.add(new_document)

 db.session.commit()
 flash(f'Document batch with ID {doc_id} created successfully!',
 'success')
 return redirect(url_for('create_batch'))

 return render_template('create_batch.html', doc_id=doc_id)

```

ปรับ Template create\_batch.html ให้แสดง doc\_id และเลขรายการเอกสาร

ให้ฟอร์มแสดง doc\_id ใบนำส่ง และเพิ่มรายการเอกสารโดยใช้ฟอร์มแบบไดนามิกเพื่อให้ผู้ใช้สามารถเพิ่มเอกสารหลายรายการได้

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>

 {% include 'menu.html' %}

```

```

<div class="ui segment">
 <p>Batch Document ID (doc_id): {{ doc_id }}</p>
</div>

<form class="ui form" method="POST">
 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" placeholder="Sender
Name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
 </div>

 <h4>Documents in Batch</h4>
 <div id="document-list">
 <div class="field">
 <label>Document Title</label>
 <input type="text" name="document_titles"
placeholder="Document Title" required>
 </div>
 </div>

 <button type="button" class="ui button"
onclick="addDocumentField()">Add Another Document</button>
 <button type="submit" class="ui button primary">Create
Batch</button>
</form>
</div>

<script>
function addDocumentField() {
 const documentList = document.getElementById('document-list');
 const newField = document.createElement('div');
 newField.classList.add('field');
 newField.innerHTML = '<label>Document Title</label><input
type="text" name="document_titles" placeholder="Document Title" required>';
 documentList.appendChild(newField);
}
</script>
</body>
</html>

```

คำอธิบายการทำงานของระบบ

- เมื่อเข้าไปที่ /create\_batch ระบบจะสร้าง doc\_id ใหม่และแสดงในหน้าฟอร์มให้ผู้ใช้เห็น

- ผู้ใช้กรอกชื่อผู้ส่ง ชื่อผู้รับ และเพิ่มรายการเอกสารแต่ละรายการที่ต้องการ
- เมื่อกด “Create Batch” ระบบจะบันทึกการส่งเอกสาร (DocumentBatch) พร้อมกับรายการเอกสาร (Document) ที่มีเลขเอกสารแยกตาม doc\_id-ลำดับรายการ

127.0.0.1 - - [06/Nov/2024 18:20:29] "POST /create\_batch HTTP/1.1" 500 -  
Error on request:

Traceback (most recent call last):

```
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1971, in
_exec_single_context
 self.dialect.do_execute(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/default.py", line 919, in do_execute
 cursor.execute(statement, parameters)
sqlite3.OperationalError: no such table: document_batch
```

The above exception was the direct cause of the following exception:

Traceback (most recent call last):

```
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/werkzeug/serving.py", line 363, in run_wsgi
 execute(self.server.app)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/werkzeug/serving.py", line 324, in execute
 application_iter = app(environ, start_response)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 2213, in __call__
 return self.wsgi_app(environ, start_response)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 2193, in wsgi_app
 response = self.handle_exception(e)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 2190, in wsgi_app
 response = self.full_dispatch_request()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1486, in full_dispatch_request
 rv = self.handle_user_exception(e)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1484, in full_dispatch_request
 rv = self.dispatch_request()
File
```

```

"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1469, in dispatch_request
 return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
File "/Users/seal/Documents/GitHub/document_tracking/app.py", line 49, in
create_batch
 db.session.commit()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/scoping.py", line 597, in commit
 return self._proxied.commit()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 1972, in commit
 trans.commit(_to_root=True)
File "<string>", line 2, in commit
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/state_changes.py", line 139, in _go
 ret_value = fn(self, *arg, **kw)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 1257, in commit
 self._prepare_impl()
File "<string>", line 2, in _prepare_impl
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/state_changes.py", line 139, in _go
 ret_value = fn(self, *arg, **kw)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 1232, in _prepare_impl
 self.session.flush()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 4296, in flush
 self._flush(objects)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 4431, in _flush
 with util.safe_reraise():
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/util/langhelpers.py", line 146, in __exit__
 raise exc_value.with_traceback(exc_tb)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 4392, in _flush
 flush_context.execute()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.

```

```

10/site-packages/sqlalchemy/orm/unitofwork.py", line 466, in execute
 rec.execute(self)
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/unitofwork.py", line 642, in execute
 util.preloaded.orm_persistence.save_obj(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/persistence.py", line 93, in save_obj
 _emit_insert_statements(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/persistence.py", line 1048, in
_emit_insert_statements
 result = connection.execute(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1422, in execute
 return meth(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/sql/elements.py", line 514, in
_execute_on_connection
 return connection._execute_clauseelement(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1644, in
_execute_clauseelement
 ret = self._execute_context(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1850, in _execute_context
 return self._exec_single_context(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1990, in
_exec_single_context
 self._handle_dbapi_exception(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 2357, in
_handle_dbapi_exception
 raise sqlalchemy_exception.with_traceback(exc_info[2]) from e
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1971, in
_exec_single_context
 self.dialect.do_execute(
 File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.

```

```
10/site-packages/sqlalchemy/engine/default.py", line 919, in do_execute
 cursor.execute(statement, parameters)
sqlalchemy.exc.OperationalError: (sqlite3.OperationalError) no such table:
document_batch
[SQL: INSERT INTO document_batch (batch_id, sender_name, receiver_name,
created_at) VALUES (?, ?, ?, ?)]
[parameters: ('11e57b31', 'บข.', 'งข.', '2024-11-06 11:20:29.956156')]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

Error นี้เกิดจากการที่ฐานข้อมูลไม่มีตาราง `document_batch`

ซึ่งอาจเกิดจากการที่ยังไม่ได้สร้างตารางใหม่หลังจากแก้ไขโครงสร้างฐานข้อมูล คุณสามารถแก้ไขได้โดยการสร้างตารางใหม่ในฐานข้อมูล ดังนี้:

### วิธีแก้ไข

1. สร้างตารางในฐานข้อมูล: ใช้คำสั่ง `db.create_all()` ใน Flask เพื่อสร้างตารางที่กำหนดไว้ในโครงสร้างฐานข้อมูลล่าสุด

ขั้นตอนการสร้างตารางใหม่

เปิด Python shell หรือรันโค้ดนี้ภายใน Flask app:

```
from app import db
db.create_all()
```

2. รีสตาร์ทเซิร์ฟเวอร์: หากคุณยังไม่ได้รีสตาร์ท Flask server หลังจากปรับโครงสร้างฐานข้อมูล ให้รีสตาร์ท server ด้วย
3. ตรวจสอบอีกครั้ง: หลังจากสร้างตารางใหม่แล้ว ลองทดสอบฟังก์ชัน `/create_batch` อีกครั้ง

หมายเหตุ: คำสั่งนี้จะสร้างตารางใหม่ในฐานข้อมูล แต่จะไม่ลบข้อมูลเดิมในฐานข้อมูล

```
(venv) seal@PornthePs-MacBook-Pro document_tracking % python3
Python 3.10.11 (v3.10.11:7d4cc5aa85, Apr 4 2023, 19:05:19) [Clang 13.0.0
(clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from app import db
>>> db.create_all()
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
 File
"/Users/seal/Documents/GitHub/document_tracking/venv/lib/python3.10/site-
packages/flask_sqlalchemy/extension.py", line 900, in create_all
 self._call_for_binds(bind_key, "create_all")
 File
"/Users/seal/Documents/GitHub/document_tracking/venv/lib/python3.10/site-
packages/flask_sqlalchemy/extension.py", line 871, in _call_for_binds
 engine = self.engines[key]
 File
"/Users/seal/Documents/GitHub/document_tracking/venv/lib/python3.10/site-
packages/flask_sqlalchemy/extension.py", line 687, in engines
 app = current_app._get_current_object() # type: ignore[attr-defined]
 File
"/Users/seal/Documents/GitHub/document_tracking/venv/lib/python3.10/site-
```

```
packages/werkzeug/local.py", line 519, in _get_current_object
 raise RuntimeError(unbound_message) from None
RuntimeError: Working outside of application context.
```

This typically means that you attempted to use functionality that needed the current application. To solve this, set up an application context with `app.app_context()`. See the documentation for more information.

Error นี้เกิดจากการที่ `db.create_all()` ถูกเรียกใช้นอก **application context** ซึ่ง Flask ต้องการ context นี้เพื่อเข้าถึงการตั้งค่าและสร้างตารางในฐานข้อมูล

คุณสามารถแก้ปัญหานี้ได้โดยการเพิ่ม **application context** ก่อนเรียกใช้ `db.create_all()` ด้วยคำสั่ง `app.app_context()` ดังนี้:

```
from app import app, db

with app.app_context():
 db.create_all()
```

#### วิธีการรัน

1. เปิด Python shell ใหม่
2. รันคำสั่งนี้เพื่อสร้างตารางในฐานข้อมูล

#### IntegrityError

```
sqlalchemy.exc.IntegrityError: (sqlite3.IntegrityError) datatype mismatch
[SQL: INSERT INTO document_batch (batch_id, sender_name, receiver_name,
created_at) VALUES (?, ?, ?, ?)]
[parameters: ('9c3cb1a1', 'บข.', 'งช.', '2024-11-06 11:24:33.989411')]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

Traceback (most recent call last)

File

```
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1971, in
_exec_single_context
self.dialect.do_execute(
```

File

```
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/default.py", line 919, in do_execute
cursor.execute(statement, parameters)
```

The above exception was the direct cause of the following exception:

File

```
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 2213, in __call__
return self.wsgi_app(environ, start_response)
```

File

```
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 2193, in wsgi_app
response = self.handle_exception(e)
```

File



```
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 2190, in wsgi_app
response = self.full_dispatch_request()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1486, in full_dispatch_request
rv = self.handle_user_exception(e)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1484, in full_dispatch_request
rv = self.dispatch_request()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1469, in dispatch_request
return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
File "/Users/seal/Documents/GitHub/document_tracking/app.py", line 49, in
create_batch
db.session.commit()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/scoping.py", line 597, in commit
return self._proxied.commit()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 1972, in commit
trans.commit(_to_root=True)
File "<string>", line 2, in commit
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/state_changes.py", line 139, in _go
ret_value = fn(self, *arg, **kw)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 1257, in commit
self._prepare_impl()
File "<string>", line 2, in _prepare_impl
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/state_changes.py", line 139, in _go
ret_value = fn(self, *arg, **kw)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 1232, in _prepare_impl
self.session.flush()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 4296, in flush
self._flush(objects)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
```

```
10/site-packages/sqlalchemy/orm/session.py", line 4431, in _flush
with util.safe_reraise():
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/util/langhelpers.py", line 146, in __exit__
raise exc_value.with_traceback(exc_tb)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/session.py", line 4392, in _flush
flush_context.execute()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/unitofwork.py", line 466, in execute
rec.execute(self)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/unitofwork.py", line 642, in execute
util.preloaded.orm_persistence.save_obj(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/persistence.py", line 93, in save_obj
_emit_insert_statements(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/persistence.py", line 1048, in
_emit_insert_statements
result = connection.execute(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1422, in execute
return meth(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/sql/elements.py", line 514, in
_execute_on_connection
return connection._execute_clauseelement(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1644, in
_execute_clauseelement
ret = self._execute_context(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1850, in _execute_context
return self._exec_single_context(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1990, in
_exec_single_context
self._handle_dbapi_exception(
```

```

File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 2357, in
_handle_dbapi_exception
raise sqlalchemy_exception.with_traceback(exc_info[2]) from e
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/base.py", line 1971, in
_exec_single_context
self.dialect.do_execute(
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/engine/default.py", line 919, in do_execute
cursor.execute(statement, parameters)
sqlalchemy.exc.IntegrityError: (sqlite3.IntegrityError) datatype mismatch
[SQL: INSERT INTO document_batch (batch_id, sender_name, receiver_name,
created_at) VALUES (?, ?, ?, ?)] [parameters: ('9c3cb1a1', 'บข.', 'งข.',
'2024-11-06 11:24:33.989411')] (Background on this error at:
https://sqlalche.me/e/20/gkpj)
The debugger caught an exception in your WSGI application. You can now look
at the traceback which led to the error.
To switch between the interactive traceback and the plaintext one, you can
click on the "Traceback" headline. From the text traceback you can also
create a paste of it. For code execution mouse-over the frame you want to
debug and click on the console icon on the right side.

```

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

```

dump() shows all variables in the frame
dump(obj) dumps all that's known about the object
Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback
interpreter.

```

Error นี้บ่งชี้ว่าเกิดปัญหาความไม่ตรงกันของประเภทข้อมูล (datatype mismatch) ในการเพิ่มข้อมูลลงในตาราง `document_batch` ซึ่งอาจเกิดจากการกำหนดประเภทข้อมูลของคอลัมน์ในตารางไม่ตรงกับข้อมูลที่พยายามเพิ่มลงไป ตัวอย่างเช่น `batch_id` อาจถูกกำหนดให้เป็น `Integer` แต่ข้อมูลที่กำลังพยายามใส่เป็น `String`

#### วิธีการตรวจสอบและแก้ไข

1. ตรวจสอบการกำหนดประเภทข้อมูลของคอลัมน์: ตรวจสอบในโมเดล `DocumentBatch` ว่ากำหนดประเภทข้อมูลของ `batch_id` เป็น `String` (หรือ `db.String`) หาก `batch_id` ต้องการเก็บ UUID หรือข้อความ และตรวจสอบคอลัมน์อื่น ๆ ให้แน่ใจว่าตรงกับข้อมูลที่กำลังบันทึกลงไป
2. ตัวอย่างการปรับโมเดล `DocumentBatch`

ตรวจสอบว่า `batch_id` และคอลัมน์อื่น ๆ กำหนดประเภทข้อมูลให้ตรงกับข้อมูลที่ใส่ลงไป ดังนี้:

```

class DocumentBatch(db.Model):
 __tablename__ = 'document_batch'

```

```

 batch_id = db.Column(db.String(50), primary_key=True) # กำหนดเป็น
String ถ้า batch_id เป็นข้อความ
 sender_name = db.Column(db.String(100), nullable=False)
 receiver_name = db.Column(db.String(100), nullable=False)
 created_at = db.Column(db.DateTime, default=datetime.utcnow)

ความสัมพันธ์กับเอกสาร
 documents = db.relationship('Document', backref='batch', lazy=True)

```

3. ลบฐานข้อมูลเดิมและสร้างใหม่ (ถ้าจำเป็น): หากคุณสามารถทำการปรับปรุงโมเดลแล้ว ให้ลบไฟล์ฐานข้อมูลเดิม (ถ้าเป็น SQLite ให้ลบไฟล์ .db) และเรียก db.create\_all() ใหม่เพื่อให้โครงสร้างฐานข้อมูลอัปเดต

4. ทดสอบอีกครั้ง: หลังจากแก้ไขและสร้างฐานข้อมูลใหม่แล้ว ลองรันแอปพลิเคชันอีกครั้งเพื่อตรวจสอบว่าปัญหาได้รับการแก้ไขแล้ว

```

app.py
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_mail import Mail, Message
from models import db, Document, DocumentStatus, DocumentBatch
import qrcode
import os
from uuid import uuid4

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)

ตั้งค่า Flask-Mail
app.config['MAIL_SERVER'] = 'smtp.example.com' # แทนที่ด้วย SMTP Server ของคุณ
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'your_email@example.com' # แทนที่ด้วย Email ของคุณ
app.config['MAIL_PASSWORD'] = 'your_email_password' # แทนที่ด้วยรหัสผ่าน
app.config['MAIL_DEFAULT_SENDER'] = 'your_email@example.com'

mail = Mail(app)

def send_status_update_email(doc_id, title, status, receiver_email):
 try:
 msg = Message(
 subject=f"Status Update for Document ID {doc_id}",
 recipients=[receiver_email],
 body=f"Document '{title}' has been updated to status: {status}."
)
 mail.send(msg)
 print("Email sent successfully!")
 except Exception as e:

```

```

 print(f"Failed to send email: {e}")

@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 doc_id = str(uuid4()).split('-')[0] # สร้าง doc_id แบบสุ่มเพื่อใช้เป็นเลขใบนำส่ง
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายชื่อเอกสารที่เป็น list

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name)
 db.session.add(batch)
 db.session.commit()

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for index, title in enumerate(document_titles, start=1):
 new_document = Document(
 doc_id=f"{doc_id}-{index}", # กำหนดเลขรายการเอกสารตามรูปแบบ doc_id
 title=title,
 batch_id=batch.batch_id
)
 db.session.add(new_document)

 db.session.commit()
 flash(f'Document batch with ID {doc_id} created successfully!',
 'success')
 return redirect(url_for('create_batch'))

 return render_template('create_batch.html', doc_id=doc_id)

สร้างเส้นทางสำหรับสร้างเอกสารใหม่
@app.route('/create_document', methods=['GET', 'POST'])
def create_document():
 if request.method == 'POST':
 doc_id = request.form['doc_id']
 title = request.form['title']
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']

 # สร้าง QR Code
 qr = qrcode.make(doc_id)
 qr_path = os.path.join('static', 'qr_codes', f'{doc_id}.png')
 qr.save(qr_path)

```

```

 # บันทึกข้อมูลเอกสาร
 new_document = Document(doc_id=doc_id, title=title,
sender_name=sender_name, receiver_name=receiver_name)
 db.session.add(new_document)
 db.session.commit()

 flash('Document created successfully!', 'success')
 return redirect(url_for('create_document'))

 return render_template('create_document.html')

เส้นทางสำหรับบันทึกสถานะเอกสาร
@app.route('/update_status', methods=['GET', 'POST'])
def update_status():
 document = None
 doc_id = request.args.get('doc_id') # รับค่า doc_id จาก query string

 # ถ้ามี doc_id ใน query string ให้ค้นหาข้อมูลเอกสารโดยอัตโนมัติ
 if doc_id:
 document = Document.query.filter_by(doc_id=doc_id).first()

 if request.method == 'POST':
 # กรณีผู้ใช้กรอก doc_id ในฟอร์มค้นหา
 doc_id = request.form.get('doc_id', doc_id) # ใช้ doc_id จากฟอร์มหรือ
query string
 document = Document.query.filter_by(doc_id=doc_id).first()

 if document:
 # ถ้ามีเอกสาร และมีข้อมูล location และ status ให้บันทึกสถานะใหม่
 location = request.form.get('location')
 status = request.form.get('status')

 if location and status:
 # บันทึกสถานะใหม่
 new_status = DocumentStatus(doc_id=doc_id, location=location,
status=status)
 db.session.add(new_status)
 db.session.commit()

 document.status = status
 db.session.commit()

 flash('Status updated successfully!', 'success')
 return redirect(url_for('update_status', doc_id=doc_id))
 else:
 flash('Document ID not found. Please try again.', 'error')

 return render_template('update_status.html', document=document)

```

```

เส้นทางสำหรับตรวจสอบสถานะเอกสาร
@app.route('/check_status', methods=['GET', 'POST'])
def check_status():
 search_query = None
 page = request.args.get('page', 1, type=int)
 per_page = 50

 if request.method == 'POST':
 search_query = request.form['query']

 if search_query:
 # ค้นหาเอกสารที่ตรงกับ query
 documents =
Document.query.filter(Document.doc_id.like(f"%{search_query}%")).paginate(pag
e=page, per_page=per_page)
 else:
 # แสดงรายการเอกสารล่าสุดทั้งหมด แบ่งเป็นหน้า
 documents =
Document.query.order_by(Document.created_at.desc()).paginate(page=page,
per_page=per_page)

 return render_template('check_status.html', documents=documents,
search_query=search_query)

@app.route('/search', methods=['GET', 'POST'])
def search_documents():
 results = []
 if request.method == 'POST':
 query = request.form['query']

 # ค้นหาข้อมูลโดยใช้คำค้นหา
 results = Document.query.filter(
 (Document.doc_id.like(f"%{query}%")) |
 (Document.title.like(f"%{query}%")) |
 (Document.sender_name.like(f"%{query}%")) |
 (Document.receiver_name.like(f"%{query}%"))
).all()

 if not results:
 flash('No documents found.', 'warning')

 return render_template('search.html', results=results)

@app.route('/')

```

```
def home():
 return render_template('home.html')

if __name__ == '__main__':
 with app.app_context():
 db.create_all()
 app.run(debug=True)
```

ต้องปรับส่วนอื่นอะไรบ้าง

จากโค้ดที่คุณให้มา มีบางส่วนที่สามารถปรับปรุงเพื่อให้ทำงานได้ดีขึ้นตามโครงสร้างฐานข้อมูลใหม่ โดยเฉพาะการลบ `create_document` เนื่องจากฟังก์ชันนี้ได้ถูกรวมเข้าไปใน `create_batch` แล้ว นอกจากนี้ ยังมีการตั้งค่าต่าง ๆ ที่สามารถปรับให้สอดคล้องกันได้:

### 1. ลบฟังก์ชัน `create_document`

เนื่องจาก `create_document` ได้ถูกรวมไว้ใน `create_batch` แล้ว (เพื่อสร้างใบนำส่งและเอกสารหลายรายการในคราวเดียว) ฟังก์ชันนี้ไม่จำเป็นอีกต่อไป คุณสามารถลบฟังก์ชัน `create_document` ออกจาก `app.py` ได้เลย

### 2. อัปเดตเมนูใน `menu.html`

ตรวจสอบว่าเมนูใน `menu.html` ได้รับการอัปเดตให้ชี้ไปยัง `create_batch` แทน `create_document`:

```
<div class="ui secondary pointing menu">
 Home
 Create Document
Batch
 Search
Document
 Update Status
 Check Status
</div>
```

### 3. ตรวจสอบการกำหนดค่าและสร้างตารางใหม่

ถ้าคุณเปลี่ยนแปลงโครงสร้างของโมเดล `DocumentBatch` หรือ `Document` ใน `models.py` แล้ว อย่าลืมสร้างฐานข้อมูลใหม่โดยลบไฟล์ฐานข้อมูลเก่า (เช่น `documents.db`) และใช้คำสั่ง `db.create_all()` เพื่อสร้างตารางใหม่ทั้งหมด (ในกรณีของ SQLite)

```
if __name__ == '__main__':
 with app.app_context():
 db.create_all()
 app.run(debug=True)
```



#### 4. ตรวจสอบฟังก์ชัน update\_status

ฟังก์ชันนี้มีการอัปเดตสถานะของเอกสารโดยใช้ doc\_id จาก URL parameter (doc\_id ที่ส่งมากับ URL) ซึ่งทำงานได้ตามที่ต้องการ แต่ควรตรวจสอบให้แน่ใจว่า DocumentStatus และ Document มีการเชื่อมโยง batch\_id ที่ถูกต้องด้วย

#### 5. แก้ไขการสร้างเอกสารให้ใช้ doc\_id แบบไบนาส

ในฟังก์ชัน create\_batch ให้ doc\_id ของเอกสารมีรูปแบบตามโครงสร้างที่ตั้งไว้ (doc\_id-ลำดับ) ซึ่งจะรวม batch\_id จาก DocumentBatch

#### 6. ตรวจสอบการตั้งค่า Flask-Mail

หากต้องการทดสอบการส่งอีเมลใน send\_status\_update\_email อย่าลืมตั้งค่า SMTP server ให้ถูกต้อง เช่น MAIL\_SERVER, MAIL\_PORT, MAIL\_USERNAME, และ MAIL\_PASSWORD

ช่วยสร้าง code ใหม่ของ app.py ทั้งหมดให้ด้วย

แน่นอนครับ นี่คือโค้ดทั้งหมดของ app.py ที่ปรับปรุงตามที่ได้กล่าวไว้:

```
app.py
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_mail import Mail, Message
from models import db, Document, DocumentStatus, DocumentBatch
import qrcode
import os
from uuid import uuid4
from datetime import datetime
```

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)
```

```
ตั้งค่า Flask-Mail
app.config['MAIL_SERVER'] = 'smtp.example.com' # แทนที่ด้วย SMTP Server ของคุณ
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'your_email@example.com' # แทนที่ด้วย Email ของคุณ
app.config['MAIL_PASSWORD'] = 'your_email_password' # แทนที่ด้วยรหัสผ่าน
app.config['MAIL_DEFAULT_SENDER'] = 'your_email@example.com'
```

```
mail = Mail(app)
```

```
ฟังก์ชันส่งอีเมลแจ้งเตือนเมื่ออัปเดตสถานะ
```

```
def send_status_update_email(doc_id, title, status, receiver_email):
 try:
 msg = Message(
 subject=f"Status Update for Document ID {doc_id}",
```

```

 recipients=[receiver_email],
 body=f"Document '{title}' has been updated to status: {status}."
)
 mail.send(msg)
 print("Email sent successfully!")
except Exception as e:
 print(f"Failed to send email: {e}")

ฟังก์ชันสร้างการส่งเอกสาร (Document Batch) พร้อมเอกสารใน batch
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 doc_id = str(uuid4()).split('-')[0] # สร้าง doc_id แบบสุ่มเพื่อใช้เป็นเลขใบนำส่ง
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายชื่อเอกสารที่เป็น List

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name)
 db.session.add(batch)
 db.session.commit()

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for index, title in enumerate(document_titles, start=1):
 new_document = Document(
 doc_id=f"{doc_id}-{index}", # กำหนดเลขรายการเอกสารตามรูปแบบ doc_id
 title=title,
 batch_id=batch.batch_id
)
 db.session.add(new_document)

 db.session.commit()
 flash(f'Document batch with ID {doc_id} created successfully!',
 'success')
 return redirect(url_for('create_batch'))

 return render_template('create_batch.html', doc_id=doc_id)

ฟังก์ชันอัปเดตสถานะเอกสาร
@app.route('/update_status', methods=['GET', 'POST'])
def update_status():
 document = None
 doc_id = request.args.get('doc_id') # รับค่า doc_id จาก query string

 # ถ้ามี doc_id ใน query string ให้ค้นหาข้อมูลเอกสารโดยอัตโนมัติ
 if doc_id:
 document = Document.query.filter_by(doc_id=doc_id).first()

```

```

if request.method == 'POST':
 # กรณีผู้ใช้กรอก doc_id ในฟอร์มค้นหา
 doc_id = request.form.get('doc_id', doc_id) # ใช้ doc_id จากฟอร์มหรือ
query string
 document = Document.query.filter_by(doc_id=doc_id).first()

 if document:
 # ถ้ามีเอกสาร และมีข้อมูล Location และ status ให้บันทึกสถานะใหม่
 location = request.form.get('location')
 status = request.form.get('status')

 if location and status:
 # บันทึกสถานะใหม่
 new_status = DocumentStatus(doc_id=doc_id, location=location,
status=status)
 db.session.add(new_status)
 db.session.commit()

 document.status = status
 db.session.commit()

 flash('Status updated successfully!', 'success')
 return redirect(url_for('update_status', doc_id=doc_id))
 else:
 flash('Document ID not found. Please try again.', 'error')

 return render_template('update_status.html', document=document)

ฟังก์ชันตรวจสอบสถานะเอกสาร พร้อมการค้นหาและการแบ่งหน้า
@app.route('/check_status', methods=['GET', 'POST'])
def check_status():
 search_query = None
 page = request.args.get('page', 1, type=int)
 per_page = 50

 if request.method == 'POST':
 search_query = request.form['query']

 if search_query:
 # ค้นหาเอกสารที่ตรงกับ query
 documents =
Document.query.filter(Document.doc_id.like(f"%{search_query}%")).paginate(pag
e=page, per_page=per_page)
 else:
 # แสดงรายการเอกสารล่าสุดทั้งหมด แบ่งเป็นหน้า
 documents =
Document.query.order_by(Document.created_at.desc()).paginate(page=page,

```

```
per_page=per_page)
```

```
 return render_template('check_status.html', documents=documents,
search_query=search_query)
```

```
ฟังก์ชันค้นหาเอกสารจาก doc_id, title, sender_name, receiver_name
```

```
@app.route('/search', methods=['GET', 'POST'])
```

```
def search_documents():
```

```
 results = []
```

```
 if request.method == 'POST':
```

```
 query = request.form['query']
```

```
ค้นหาข้อมูลโดยใช้คำค้นหา
```

```
 results = Document.query.filter(
 (Document.doc_id.like(f"%{query}%")) |
 (Document.title.like(f"%{query}%")) |
 (Document.batch.sender_name.like(f"%{query}%")) |
 (Document.batch.receiver_name.like(f"%{query}%"))
).all()
```

```
 if not results:
```

```
 flash('No documents found.', 'warning')
```

```
 return render_template('search.html', results=results)
```

```
หน้าแรก
```

```
@app.route('/')
```

```
def home():
```

```
 return render_template('home.html')
```

```
if __name__ == '__main__':
```

```
 with app.app_context():
```

```
 db.create_all()
```

```
 app.run(debug=True)
```

สิ่งที่อัปเดตในโค้ดนี้:

1. ลบฟังก์ชัน **create\_document**: ฟังก์ชันนี้ไม่จำเป็นแล้ว เพราะการสร้างเอกสารถูกจัดการภายใน **create\_batch**
2. เพิ่ม **DocumentBatch**: **create\_batch** ถูกใช้เพื่อสร้าง **DocumentBatch** พร้อมกับรายการเอกสารที่เกี่ยวข้อง
3. ตรวจสอบ **doc\_id** ใน **update\_status**: ใช้ **doc\_id** จาก query string เพื่อค้นหาเอกสารและแสดงฟอร์มให้อัปเดตสถานะ
4. ฟังก์ชัน **check\_status** และ **search\_documents**: รองรับการค้นหาเอกสารและการแบ่งหน้า

```
if name == 'main': with app.app_context(): db.create_all() app.run(debug=True)
```

อธิบายการทำงานส่วนนี้ให้ด้วย

ส่วนนี้ของโค้ดทำหน้าที่ในการเริ่มต้นแอปพลิเคชัน Flask และตรวจสอบว่าฐานข้อมูลได้ถูกตั้งค่าไว้อย่างถูกต้องแล้วหรือไม่ โดยใช้ฟังก์ชัน `db.create_all()` ภายใต้ **application context** เพื่อให้แน่ใจว่าตารางทั้งหมดในฐานข้อมูลถูกสร้างขึ้นก่อนที่จะรันแอปพลิเคชัน

อธิบายที่ละเอียด

```
if __name__ == '__main__':
```

- ตรวจสอบว่าไฟล์นี้กำลังรันโดยตรง (ไม่ใช่ถูกนำไป import ในไฟล์อื่น) เพื่อให้แน่ใจว่าการเรียกใช้โค้ดในบล็อกนี้จะเกิดขึ้นเฉพาะเมื่อไฟล์นี้เป็นไฟล์หลักที่รันแอป

```
with app.app_context():
 db.create_all()
```

- `app.app_context()`: การสร้าง **application context** ขึ้นมา เพื่อให้โค้ดที่ต้องการอ้างอิงถึง `app` สามารถเข้าถึงการตั้งค่าและฐานข้อมูลของแอปนี้ได้
- `db.create_all()`: เมื่ออยู่ภายใต้ **application context** แล้ว คำสั่งนี้จะสร้างตารางที่จำเป็นทั้งหมดในฐานข้อมูลตามโมเดลที่กำหนดไว้ (เช่น `Document`, `DocumentBatch`, `DocumentStatus`) หากตารางเหล่านั้นยังไม่ถูกสร้าง

```
app.run(debug=True)
```

- `app.run(debug=True)`: รันเซิร์ฟเวอร์ Flask ในโหมด **debug** ซึ่งช่วยให้สามารถดูข้อผิดพลาดในระหว่างการพัฒนา โดยเมื่อเซิร์ฟเวอร์ทำงาน จะสามารถเข้าถึงแอปพลิเคชันผ่านเบราว์เซอร์ที่ `http://127.0.0.1:5000/`

AttributeError

AttributeError: Neither 'InstrumentedAttribute' object nor 'Comparator' object associated with Document.batch has an attribute 'sender\_name'

Traceback (most recent call last)

File

"/Users/seal/Documents/GitHub/url\_shortener/user\_management/venv/lib/python3.10/site-packages/sqlalchemy/orm/attributes.py", line 472, in `__getattr__`  
return getattr(self.comparator, key)

File

"/Users/seal/Documents/GitHub/url\_shortener/user\_management/venv/lib/python3.10/site-packages/sqlalchemy/util/langhelpers.py", line 1328, in `__getattr__`  
return self.\_fallback\_getattr(key)

File

"/Users/seal/Documents/GitHub/url\_shortener/user\_management/venv/lib/python3.10/site-packages/sqlalchemy/util/langhelpers.py", line 1297, in `_fallback_getattr`  
raise AttributeError(key)

The above exception was the direct cause of the following exception:

File

"/Users/seal/Documents/GitHub/url\_shortener/user\_management/venv/lib/python3.10/site-packages/flask/app.py", line 2213, in `__call__`  
return self.wsgi\_app(environ, start\_response)

File

"/Users/seal/Documents/GitHub/url\_shortener/user\_management/venv/lib/python3.10/site-packages/flask/app.py", line 2193, in `wsgi_app`  
response = self.handle\_exception(e)

```

File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 2190, in wsgi_app
response = self.full_dispatch_request()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1486, in full_dispatch_request
rv = self.handle_user_exception(e)
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1484, in full_dispatch_request
rv = self.dispatch_request()
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/flask/app.py", line 1469, in dispatch_request
return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
File "/Users/seal/Documents/GitHub/document_tracking/app.py", line 133, in
search_documents
(Document.batch.sender_name.like(f"%{query}%")) |
File
"/Users/seal/Documents/GitHub/url_shortener/user_management/venv/lib/python3.
10/site-packages/sqlalchemy/orm/attributes.py", line 474, in __getattr__
raise AttributeError(
AttributeError: Neither 'InstrumentedAttribute' object nor 'Comparator'
object associated with Document.batch has an attribute 'sender_name'
The debugger caught an exception in your WSGI application. You can now look
at the traceback which led to the error.
To switch between the interactive traceback and the plaintext one, you can
click on the "Traceback" headline. From the text traceback you can also
create a paste of it. For code execution mouse-over the frame you want to
debug and click on the console icon on the right side.

```

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

```

dump() shows all variables in the frame
dump(obj) dumps all that's known about the object
Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback
interpreter.

```

Error นี้เกิดขึ้นเนื่องจากการพยายามเข้าถึง `sender_name` ของ `Document.batch` โดยตรง แต่ `batch` ใน `Document` เป็นความสัมพันธ์แบบ **relationship** ซึ่งต้องอ้างอิงผ่าน `join` หรือ **query filter** โดยอ้อม

ในการแก้ไข ให้ใช้ `join` เพื่อเชื่อม `Document` กับ `DocumentBatch` และจากนั้นให้เข้าถึง `sender_name` ในเงื่อนไขการค้นหาแทน

ปรับปรุงโค้ดในฟังก์ชัน `search_documents`

ให้ใช้ `join` กับ `DocumentBatch` ในการค้นหาผู้ส่งและผู้รับใน `sender_name` และ `receiver_name` ดังนี้:

```

@app.route('/search', methods=['GET', 'POST'])
def search_documents():
 results = []
 if request.method == 'POST':
 query = request.form['query']

 # ค้นหาข้อมูลโดยใช้คำค้นหา โดย join กับ DocumentBatch
 results = Document.query.join(DocumentBatch).filter(
 (Document.doc_id.like(f"%{query}%")) |
 (Document.title.like(f"%{query}%")) |
 (DocumentBatch.sender_name.like(f"%{query}%")) |
 (DocumentBatch.receiver_name.like(f"%{query}%"))
).all()

 if not results:
 flash('No documents found.', 'warning')

 return render_template('search.html', results=results)

```

คำอธิบายการปรับปรุง

- `Document.query.join(DocumentBatch)`: ทำการ join ตาราง DocumentBatch เข้ากับ Document เพื่อให้สามารถเข้าถึงฟิลด์ใน DocumentBatch เช่น `sender_name` และ `receiver_name` ในเงื่อนไขการค้นหาได้
- `.filter(...)`: กรองผลลัพธ์ตามคำค้นหาที่ระบุในฟิลด์ `doc_id`, `title`, `sender_name` และ `receiver_name`

โดยการใช้ join แบบนี้ จะช่วยให้การค้นหาทำงานได้ถูกต้อง

การค้นหาต้องแสดง `batch_id` จาก `document_batch` เป็นหลักก่อน แล้วจึงแสดงรายการ ใน `document doc_id` ที่อยู่ภายใต้ `batch_id` นั้น

เพื่อให้การค้นหาแสดง `batch_id` จาก `DocumentBatch` เป็นหลักก่อน และแสดงรายการเอกสาร (`doc_id`) ภายใต้ `batch_id` นั้น ๆ คุณสามารถปรับโค้ดให้ทำการค้นหา `DocumentBatch` ก่อน และดึงรายการเอกสารที่อยู่ในแต่ละ `batch_id` จากนั้นแสดงผลในรูปแบบที่ต้องการได้ ดังนี้

ปรับปรุงโค้ดในฟังก์ชัน `search_documents`

ฟังก์ชันนี้จะทำการค้นหา `batch_id` จาก `DocumentBatch` ก่อน จากนั้นดึงรายการเอกสาร (`doc_id`) ใน `Document` ที่เกี่ยวข้องกับ `batch_id` นั้น

```

@app.route('/search', methods=['GET', 'POST'])
def search_documents():
 results = []
 if request.method == 'POST':
 query = request.form['query']

 # ค้นหา DocumentBatch ที่ตรงกับ query ใน batch_id, sender_name หรือ receiver_name
 batches = DocumentBatch.query.filter(

```

```

 (DocumentBatch.batch_id.like(f"%{query}%")) |
 (DocumentBatch.sender_name.like(f"%{query}%")) |
 (DocumentBatch.receiver_name.like(f"%{query}%"))
).all()

 # ดึงเอกสารทั้งหมดที่อยู่ภายใต้แต่ละ batch
 for batch in batches:
 documents =
Document.query.filter_by(batch_id=batch.batch_id).all()
 results.append({
 'batch': batch,
 'documents': documents
 })

 if not results:
 flash('No documents found.', 'warning')

 return render_template('search.html', results=results)

```

การปรับแต่ง Template search.html

Template นี้จะแสดงผล batch\_id และข้อมูลผู้ส่งและผู้รับเป็นหลัก โดยรายการ doc\_id ที่อยู่ภายใต้แต่ละ batch\_id จะแสดงอยู่ภายใน

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Search Documents</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Documents</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Document
Batch ID, Sender or Receiver Name" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if results %}
 <h3>Search Results</h3>

```



```

 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="file icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}</p>
 <p>Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at }}</p>
 <div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in result.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID:
{{ document.doc_id }}</p>
 <p>Title: {{
document.title }}</p>
 <p>Status: {{
document.status }}</p>
 </div>
 </div>
 {% endfor %}
 </div>
 </div>
 </div>
 {% endfor %}
 </div>
 {% else %}
 <div class="ui message">
 <p>No documents found.</p>
 </div>
 {% endif %}
</div>
</body>
</html>

```

คำอธิบายการทำงาน

1. ค้นหา **DocumentBatch** ที่มี **batch\_id**, **sender\_name** หรือ **receiver\_name** ตรงกับ query
2. ดึงรายการเอกสาร (ในตาราง **Document**) สำหรับแต่ละ **batch\_id** ที่ค้นพบ
3. ใน **Template search.html**:
  - แสดง **batch\_id** และข้อมูลของผู้ส่งและผู้รับ
  - แสดงรายการ **doc\_id** และรายละเอียดเอกสาร (**title**, **status**) ที่อยู่ภายใต้ **batch\_id** นั้น

การใช้ `uuid` ก็เป็นเรื่องดี แต่เราสามารถทำ `id` ที่ง่ายกว่านี้ได้ไหม จะได้ไม่ยุ่งยากกับผู้ใช้ในกรณีที่เขาต้องพิมพ์เลข `batch_id` ในการค้นหา เช่น อาจใช้ ตัวอักษร สองตัวขึ้นต้น ที่เหลือเป็นตัวเลข และจำนวนหลักควรเป็นเท่าไรถึงจะไม่มากเกินไป

คุณสามารถสร้าง `batch_id` ที่เรียบง่ายขึ้นได้โดยใช้ตัวอักษร 2 ตัวตามด้วยตัวเลข โดยการผสมของรหัสตัวอักษรและตัวเลข เช่น `AB123456` ซึ่งจะมีความยาวที่พอเหมาะ (เช่น 8 หลัก) เพื่อให้ผู้ใช้สามารถจำและพิมพ์ได้ง่าย

ตัวอย่างการสร้าง `batch_id` ที่ง่ายขึ้น

ในฟังก์ชัน `create_batch` คุณสามารถสร้าง `batch_id` โดยใช้ตัวอักษรสองตัวแรกและตัวเลขสุ่มตามที่กำหนด:

```
import random
import string

def generate_batch_id():
 prefix = ''.join(random.choices(string.ascii_uppercase, k=2)) # ตัวอักษรสองตัว
 suffix = ''.join(random.choices(string.digits, k=6)) # ตัวเลขหกตัว
 return f"{prefix}{suffix}"
```

จากนั้นใน `create_batch` สามารถใช้ฟังก์ชันนี้เพื่อสร้าง `batch_id`:

```
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 doc_id = generate_batch_id() # สร้าง batch_id ใหม่
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายชื่อเอกสารที่เป็น List

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name)
 db.session.add(batch)
 db.session.commit()

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for index, title in enumerate(document_titles, start=1):
 new_document = Document(
 doc_id=f"{doc_id}-{index}", # กำหนดเลขรายการเอกสารตามรูปแบบ doc_id
 title=title,
 batch_id=batch.batch_id
)
 db.session.add(new_document)

 db.session.commit()
 flash(f'Document batch with ID {doc_id} created successfully!',
 'success')
 return redirect(url_for('create_batch'))

 return render_template('create_batch.html', doc_id=doc_id)
```

### ข้อดีของรูปแบบนี้

- รูปแบบของ `batch_id` เช่น AB123456 เป็นที่จดจำง่ายและพิมพ์ง่ายกว่ารูปแบบ UUID
- ความยาว 8 ตัวอักษร (2 ตัวอักษร + 6 ตัวเลข) เป็นจำนวนที่ไม่มากเกินไป

ตอนกดปุ่ม `create document batch` โปรแกรมแสดง Batch Document ID DH856923

แต่พอกดปุ่มสร้าง `create batch` โปรแกรมสร้าง Batch Document ID อื่นที่ไม่ใช่ DH856923 จะแก้ไขอย่างไร

เนื่องจากในโค้ดปัจจุบัน `batch_id` ถูกสร้างใหม่ในฟังก์ชัน `generate_batch_id()` เมื่อโหลดหน้า `create_batch.html` และไม่มีการบันทึกค่านี้ไว้ในฟังก์ชัน POST ทำให้เมื่อกดปุ่มสร้าง `batch` โปรแกรมจะสร้าง `batch_id` ใหม่

เราสามารถแก้ไขปัญหานี้ได้โดยการเก็บค่า `batch_id` ไว้ในแบบฟอร์มและส่งไปยังเซิร์ฟเวอร์เมื่อกดปุ่ม `Create Batch` เพื่อให้ใช้ `batch_id` ที่แสดงไว้เดิม

### วิธีการแก้ไข

1. แก้ไขโค้ดในฟังก์ชัน `create_batch` ให้สร้าง `batch_id` ครั้งเดียวและส่ง `batch_id` เดิมไปกับแบบฟอร์ม
2. เพิ่มฟิลด์ `batch_id` ซ่อนในแบบฟอร์มเพื่อเก็บค่า `batch_id` ที่สร้างไว้เดิม

โค้ดที่ปรับปรุงใน `app.py`

```
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 if request.method == 'POST':
 # รับ batch_id จากฟอร์ม
 doc_id = request.form['batch_id']
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายชื่อเอกสารที่เป็น List

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name)
 db.session.add(batch)
 db.session.commit()

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for index, title in enumerate(document_titles, start=1):
 new_document = Document(
 doc_id=f"{doc_id}-{index}", # กำหนดเลขรายการเอกสารตามรูปแบบ doc_id
 title=title,
 batch_id=batch.batch_id
)
 db.session.add(new_document)

 db.session.commit()
 flash(f'Document batch with ID {doc_id} created successfully!',
 'success')
```

```

 return redirect(url_for('create_batch'))

กรณี GET สร้าง batch_id ใหม่
doc_id = generate_batch_id() # สร้าง batch_id ใหม่และส่งไปที่หน้า create_batch.html
return render_template('create_batch.html', doc_id=doc_id)

```

## ปรับปรุง Template create\_batch.html

ในฟอร์ม create\_batch.html เพิ่มฟิลด์ซ่อน (hidden input) เพื่อส่ง batch\_id ที่สร้างไว้ไปพร้อมกับการส่งข้อมูล:

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>

 {% include 'menu.html' %}

 <div class="ui segment">
 <p>Batch Document ID: {{ doc_id }}</p>
 </div>

 <form class="ui form" method="POST">
 <!-- ฟิลด์ซ่อนเพื่อเก็บ batch_id -->
 <input type="hidden" name="batch_id" value="{{ doc_id }}">

 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" placeholder="Sender
Name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
 </div>

 <h4>Documents in Batch</h4>
 <div id="document-list">
 <div class="field">
 <label>Document Title</label>
 <input type="text" name="document_titles"

```

```

placeholder="Document Title" required>
 </div>
</div>

 <button type="button" class="ui button"
onclick="addDocumentField()">Add Another Document</button>
 <button type="submit" class="ui button primary">Create
Batch</button>
</form>
</div>

<script>
 function addDocumentField() {
 const documentList = document.getElementById('document-list');
 const newField = document.createElement('div');
 newField.classList.add('field');
 newField.innerHTML = '<label>Document Title</label><input
type="text" name="document_titles" placeholder="Document Title" required>';
 documentList.appendChild(newField);
 }
</script>
</body>
</html>

```

คำอธิบาย

- **ฟิลด์ซ่อน (hidden input)** ในแบบฟอร์มจะส่งค่า `batch_id` ที่สร้างขึ้นในตอนแรกไปยังเซิร์ฟเวอร์ ทำให้ `batch_id` ไม่เปลี่ยนแปลงเมื่อกดปุ่ม Create Batch
- โค้ดนี้จะแสดง `batch_id` เดิมที่สร้างในหน้าแรก (GET request) ในหน้าสร้าง Document Batch

ส่วนของ

```

ฟังก์ชันตรวจสอบสถานะเอกสาร พร้อมการค้นหาและการแบ่งหน้า
@app.route('/check_status', methods=['GET', 'POST'])
def check_status():
 search_query = None
 page = request.args.get('page', 1, type=int)
 per_page = 50

 if request.method == 'POST':
 search_query = request.form['query']

 if search_query:
 # ค้นหาเอกสารที่ตรงกับ query
 documents =
Document.query.filter(Document.doc_id.like(f"%{search_query}%")).paginate(pag
e=page, per_page=per_page)
 else:
 # แสดงรายการเอกสารล่าสุดทั้งหมด แบ่งเป็นหน้า
 documents =

```

```
Document.query.order_by(Document.created_at.desc()).paginate(page=page,
per_page=per_page)
```

```
return render_template('check_status.html', documents=documents,
search_query=search_query)
```

ต้องปรับใหม่ ให้การแสดงผล เป็นแบบการค้นหาด้วย คือแสดง **batch** ก่อน ตามด้วยรายการ **document** และแสดงรายละเอียดอื่นให้ครบด้วย ผู้ส่ง, ผู้รับ, สถานะ วันที่เวลา (แสดงรูปแบบ 2024-11-06 11:28:14)

คุณสามารถปรับปรุงฟังก์ชัน **check\_status** ให้แสดงผลตามที่ต้องการ โดยแสดง **batch** ก่อนตามด้วยรายการ **document** ในแต่ละ **batch** และแสดงรายละเอียดต่าง ๆ เช่น ผู้ส่ง, ผู้รับ, สถานะ, และวันที่เวลา

โค้ดที่ปรับปรุงจะแสดง **batch** ก่อน จากนั้นจะแสดงรายการ **document** ที่เกี่ยวข้องกับ **batch\_id** นั้น ๆ พร้อมข้อมูลที่จำเป็น

ปรับปรุงฟังก์ชัน **check\_status** ใน **app.py**

```
from datetime import datetime
```

```
ฟังก์ชันตรวจสอบสถานะเอกสาร พร้อมการค้นหาและการแบ่งหน้า
```

```
@app.route('/check_status', methods=['GET', 'POST'])
```

```
def check_status():
```

```
 search_query = None
```

```
 page = request.args.get('page', 1, type=int)
```

```
 per_page = 50
```

```
 if request.method == 'POST':
```

```
 search_query = request.form['query']
```

```
 if search_query:
```

```
 # ค้นหา batch ที่ตรงกับ query
```

```
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).paginate(page=page, per_page=per_page)
```

```
 else:
```

```
 # แสดง batch ล่าสุดทั้งหมด แบ่งเป็นหน้า
```

```
 batches =
```

```
DocumentBatch.query.order_by(DocumentBatch.created_at.desc()).paginate(page=p
age, per_page=per_page)
```

```
 return render_template('check_status.html', batches=batches,
search_query=search_query)
```

ปรับปรุง **Template check\_status.html**

ให้แสดง **batch\_id** ของแต่ละการส่งเอกสารเป็นหลัก และแสดงรายการ **document** แต่ละรายการใน **batch** นั้น รวมถึงรายละเอียดต่าง ๆ

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Check Document Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Check Document Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender or Receiver" value="{{ search_query or '' }}">
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if batches.items %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for batch in batches.items %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ batch.batch_id }}</h4>
 <p>Sender: {{ batch.sender_name
}}</p>
 <p>Receiver: {{
batch.receiver_name }}</p>
 <p>Created At: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in batch.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID:
{{ document.doc_id }}</p>
 <p>Title: {{
document.title }}</p>
 <p>Status: {{
document.status }}</p>

```

```

 <p>Last Updated:
{{ document.updated_at.strftime('%Y-%m-%d %H:%M:%S') if document.updated_at
else 'N/A' }}</p>
 </div>
 </div>
 {% endfor %}
 </div>
 </div>
 </div>
 {% endfor %}
 </div>

<!-- Pagination -->
<div class="ui pagination menu">
 {% if batches.has_prev %}
 <a class="item" href="{% url_for('check_status',
page=batches.prev_num) %}">Previous
 {% else %}
 <div class="disabled item">Previous</div>
 {% endif %}

 {% for page_num in batches.iter_pages(left_edge=1,
right_edge=1, left_current=2, right_current=2) %}
 {% if page_num %}
 {% if page_num == batches.page %}
 <div class="active item">{{ page_num }}</div>
 {% else %}
 <a class="item" href="{% url_for('check_status',
page=page_num) %}">{{ page_num }}
 {% endif %}
 {% else %}
 <div class="disabled item">...</div>
 {% endif %}
 {% endfor %}

 {% if batches.has_next %}
 <a class="item" href="{% url_for('check_status',
page=batches.next_num) %}">Next
 {% else %}
 <div class="disabled item">Next</div>
 {% endif %}
</div>
{% else %}
 <div class="ui message">
 <p>No documents found.</p>
 </div>
{% endif %}
</div>

```



```
</body>
</html>
```

#### คำอธิบายการทำงาน

- ค้นหา **DocumentBatch** ก่อน: หากมีการค้นหา จะใช้ `search_query` ในการกรองข้อมูลจาก `batch_id`, `sender_name` หรือ `receiver_name` ของ `DocumentBatch`
- แสดง **batch** และ **documents** ที่เกี่ยวข้อง: ใน `check_status.html` จะแสดงรายละเอียด `batch_id`, ผู้ส่ง (`sender_name`), ผู้รับ (`receiver_name`), วันที่สร้าง (`created_at`) และรายการ `document` ในแต่ละ `batch`
- รูปแบบวันที่และเวลา: แสดงวันที่ในรูปแบบ `YYYY-MM-DD HH:MM:SS`

Batch ID: GU771281

Sender: วช.

Receiver: บช.

Created At: 2024-11-06 11:48:01

Documents in this Batch:

Document ID: GU771281-1

Title: งบประมาณ

Status: Created

Last Updated: N/A

เนื่องจาก Status: Created เป็น status ของ batch ควรปรับ database ใหม่ และการแสดงผลด้วย การแสดงผลควรเป็นแบบ

Batch ID: GU771281 Sender: วช. Receiver: บช. Created At: 2024-11-06 11:48:01 Status: Created Last Updated: N/A

Documents in this Batch: Document ID: GU771281-1 Title: งบประมาณ

เพื่อให้โครงสร้างข้อมูลเหมาะสมมากขึ้นและแสดงสถานะ (Status) ที่ระดับ batch พร้อมแสดงรายการเอกสาร (documents) ที่เกี่ยวข้องโดยไม่รวมสถานะในแต่ละเอกสาร คุณสามารถปรับโครงสร้างฐานข้อมูลและการแสดงผลให้สอดคล้องกับรูปแบบนี้ได้

#### การปรับปรุงฐานข้อมูล

- เพิ่มฟิลด์ **status** และ **updated\_at** ในตาราง `DocumentBatch` แทนที่จะอยู่ในระดับเอกสาร (Document) เพื่อเก็บสถานะของการส่งเอกสารและเวลาที่อัปเดตล่าสุด
- ลบฟิลด์ **status** และ **updated\_at** ออกจากตาราง `Document` เนื่องจากสถานะและการอัปเดตล่าสุดจะเก็บไว้ที่ระดับ `batch` เท่านั้น

การอัปเดตโครงสร้างโมเดลใน `models.py`

```
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
```

```
db = SQLAlchemy()
```

```
class DocumentBatch(db.Model):
 __tablename__ = 'document_batch'
 batch_id = db.Column(db.String(50), primary_key=True)
 sender_name = db.Column(db.String(100), nullable=False)
 receiver_name = db.Column(db.String(100), nullable=False)
 created_at = db.Column(db.DateTime, default=datetime.utcnow)
 status = db.Column(db.String(50), default="Created") # สถานะของ batch
 updated_at = db.Column(db.DateTime) # เวลาที่อัปเดตล่าสุด

 # ความสัมพันธ์กับเอกสาร
 documents = db.relationship('Document', backref='batch', lazy=True)
```

```
class Document(db.Model):
 __tablename__ = 'document'
 id = db.Column(db.Integer, primary_key=True)
 doc_id = db.Column(db.String(50), unique=True, nullable=False)
 title = db.Column(db.String(100), nullable=False)
 created_at = db.Column(db.DateTime, default=datetime.utcnow)
 batch_id = db.Column(db.String(50),
db.ForeignKey('document_batch.batch_id'), nullable=False)
```

การแก้ไขฟังก์ชัน `check_status`

ฟังก์ชัน `check_status` ควรแสดงผล batch พร้อมสถานะของมัน  
และแสดงรายการเอกสารแต่ละรายการโดยไม่รวมสถานะที่ระดับเอกสาร

```
@app.route('/check_status', methods=['GET', 'POST'])
def check_status():
 search_query = None
 page = request.args.get('page', 1, type=int)
 per_page = 50

 if request.method == 'POST':
 search_query = request.form['query']

 if search_query:
 # ค้นหา batch ที่ตรงกับ query
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).paginate(page=page, per_page=per_page)
 else:
 # แสดง batch ล่าสุดทั้งหมด แบ่งเป็นหน้า
 batches =
DocumentBatch.query.order_by(DocumentBatch.created_at.desc()).paginate(page=p
```

```
age, per_page=per_page)
```

```
 return render_template('check_status.html', batches=batches,
search_query=search_query)
```

ปรับปรุง Template check\_status.html

การแสดงผลใน Template จะเน้นการแสดงผล batch พร้อมกับ documents ในแต่ละ batch โดยแสดงสถานะของ batch เท่านั้น

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Check Document Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Check Document Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender or Receiver" value="{{ search_query or '' }}">
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if batches.items %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for batch in batches.items %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ batch.batch_id }}</h4>
 <p>Sender: {{ batch.sender_name
}}
 Receiver: {{
batch.receiver_name }}</p>
 <p>Created At: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: {{ batch.status
}}</p>
 <p>Last Updated: {{
```

```

batch.updated_at.strftime('%Y-%m-%d %H:%M:%S') if batch.updated_at else 'N/A'
}}</p>

<div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in batch.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID:

{{ document.doc_id }}</p>

 <p>Title: {{
document.title }}</p>

 </div>
 </div>
 {% endfor %}
</div>

<!-- Pagination -->
<div class="ui pagination menu">
 {% if batches.has_prev %}
 <a class="item" href="{% url_for('check_status',
page=batches.prev_num) %}">Previous
 {% else %}
 <div class="disabled item">Previous</div>
 {% endif %}

 {% for page_num in batches.iter_pages(left_edge=1,
right_edge=1, left_current=2, right_current=2) %}
 {% if page_num %}
 {% if page_num == batches.page %}
 <div class="active item">{{ page_num }}</div>
 {% else %}
 <a class="item" href="{% url_for('check_status',
page=page_num) %}">{{ page_num }}
 {% endif %}
 {% else %}
 <div class="disabled item">...</div>
 {% endif %}
 {% endfor %}

 {% if batches.has_next %}
 <a class="item" href="{% url_for('check_status',
page=batches.next_num) %}">Next
 {% else %}
 <div class="disabled item">Next</div>
 {% endif %}

```

```

 {% endif %}
 </div>
 {% else %}
 <div class="ui message">
 <p>No documents found.</p>
 </div>
 {% endif %}
</div>
</body>
</html>

```

## คำอธิบายการทำงาน

- สถานะของ **batch**: แสดงสถานะและเวลาที่อัปเดตล่าสุด (Last Updated) ที่ระดับ batch ไม่ใช่ที่ระดับเอกสาร (document) อีกต่อไป
- แสดงเอกสารภายใต้ **batch**: เอกสารแต่ละรายการ (document) จะถูกแสดงเฉพาะ doc\_id และ title เท่านั้น

ยังจำเป็นต้องมี document status ด้วยครับ class DocumentStatus(db.Model): **tablename = 'document\_status'** id = db.Column(db.Integer, primary\_key=True) doc\_id = db.Column(db.String(50), db.ForeignKey('document.doc\_id'), nullable=False) location = db.Column(db.String(50), nullable=False) status = db.Column(db.String(50), nullable=False) updated\_at = db.Column(db.DateTime, default=datetime.utcnow)

หรือควรตัด status ใน batch ออก เนื่องจาก batch จะต้องเกิดสถานะทั้งหมด 4 จุดอย่างที่บอกครั้งแรก คือ เมื่อสร้าง(ส่ง), จุดที่สอง จุดรับ-ส่ง ต้นทาง, จุดที่สาม จุดรับ-ส่ง ปลายทาง, จุดที่สี่ จุดรับ

ในกรณีนี้ ควรคงโมเดล DocumentStatus ไว้เพื่อบันทึกสถานะที่แตกต่างกันในแต่ละจุด (4 จุด) แทนที่จะมีสถานะเดียวใน DocumentBatch เพราะ DocumentBatch จะต้องผ่านแต่ละสถานะตามขั้นตอนที่ระบุ ซึ่ง DocumentStatus จะช่วยบันทึกและติดตามสถานะในแต่ละจุดของกระบวนการได้อย่างละเอียดและยืดหยุ่นมากกว่า

## การปรับปรุงการทำงาน

โมเดล DocumentStatus จะมีหน้าที่: - บันทึกจุดสถานะต่าง ๆ ของ batch และ document - ระบุจุดที่เอกสารนั้นอยู่ (location) และอัปเดต status พร้อมเวลา (updated\_at) เพื่อสะท้อนการเปลี่ยนแปลงในแต่ละจุด

## การแสดงผลใน Template

ให้แสดงสถานะในแต่ละจุดของเอกสาร โดยใช้ DocumentStatus เพื่อแสดงเส้นทางสถานะที่เอกสารต้องผ่านในแต่ละจุด

ในแต่ละจุดจะมีการลงข้อมูลดังนี้

ผู้ส่งเอกสารต้นทาง ผู้รับเอกสารต้นทาง พนักงานคัดแยกเอกสาร ผู้ส่งเอกสารปลายทาง ผู้รับเอกสารปลายทาง ลงชื่อ.....  
 ลงชื่อ..... ลงชื่อ..... ลงชื่อ..... ลงชื่อ.....  
 ...../...../..... เวลา ..... ...../...../..... เวลา ..... ...../...../..... เวลา .....  
 ...../...../..... เวลา..... ...../...../..... เวลา .....

ขอให้ปรับ code และ database ให้ใหม่ทั้งหมดด้วย

ตามข้อมูลที่คุณให้มา ในการบันทึกสถานะของเอกสารในแต่ละจุด

เราจะปรับปรุงฐานข้อมูลและโค้ดให้รองรับการลงข้อมูลของสถานะที่ต้องการในแต่ละจุด (5 จุด) โดยแต่ละสถานะจะมีรายละเอียดของการลงชื่อ

(signed\_by) และวันที่เวลาของการลงชื่อ (signed\_at) ดังนั้น  
เราจำเป็นต้องปรับโครงสร้างฐานข้อมูลและโค้ดให้รองรับการจัดเก็บข้อมูลตามขั้นตอนนี้

การออกแบบฐานข้อมูลใหม่

1. ปรับปรุง *DocumentBatch* เพื่อให้เก็บข้อมูล *batch\_id*, ผู้ส่ง, และผู้รับ รวมถึงข้อมูลวันที่เริ่มต้นการส่ง

```
class DocumentBatch(db.Model):
 __tablename__ = 'document_batch'
 batch_id = db.Column(db.String(50), primary_key=True)
 sender_name = db.Column(db.String(100), nullable=False)
 receiver_name = db.Column(db.String(100), nullable=False)
 created_at = db.Column(db.DateTime, default=datetime.utcnow)

 # ความสัมพันธ์กับเอกสารและสถานะ
 documents = db.relationship('Document', backref='batch', lazy=True)
 statuses = db.relationship('DocumentStatus', backref='batch', lazy=True)
```

2. ปรับปรุง *DocumentStatus* เพื่อให้เก็บข้อมูลสถานะในแต่ละจุด พร้อมข้อมูลการลงชื่อและเวลา

```
class DocumentStatus(db.Model):
 __tablename__ = 'document_status'
 id = db.Column(db.Integer, primary_key=True)
 batch_id = db.Column(db.String(50),
db.ForeignKey('document_batch.batch_id'), nullable=False)
 location = db.Column(db.String(50), nullable=False) # เช่น "ผู้ส่งเอกสารต้นทาง"
 signed_by = db.Column(db.String(100), nullable=True) # ผู้ลงชื่อ
 signed_at = db.Column(db.DateTime, nullable=True) # วันที่และเวลาที่ลงชื่อ
```

3. ปรับปรุง *Document* เพื่อเก็บข้อมูลเอกสารที่เกี่ยวข้องกับ *batch\_id* โดยไม่ต้องมีสถานะของเอกสารแยก

```
class Document(db.Model):
 __tablename__ = 'document'
 id = db.Column(db.Integer, primary_key=True)
 doc_id = db.Column(db.String(50), unique=True, nullable=False)
 title = db.Column(db.String(100), nullable=False)
 created_at = db.Column(db.DateTime, default=datetime.utcnow)
 batch_id = db.Column(db.String(50),
db.ForeignKey('document_batch.batch_id'), nullable=False)
```

ตัวอย่างโค้ดการอัปเดตสถานะใน *app.py*

เราสามารถเพิ่มโค้ดให้รองรับการอัปเดตข้อมูลการลงชื่อและเวลาในแต่ละจุดได้

```
from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for, flash
from models import db, DocumentBatch, Document, DocumentStatus

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)
```

```

@app.route('/update_status/<batch_id>', methods=['GET', 'POST'])
def update_status(batch_id):
 batch = DocumentBatch.query.get_or_404(batch_id)

 if request.method == 'POST':
 # รับข้อมูลสถานะใหม่จากฟอร์ม
 location = request.form['location']
 signed_by = request.form['signed_by']
 signed_at = datetime.strptime(request.form['signed_at'], '%Y-%m-%d
%H:%M:%S')

 # สร้างหรืออัปเดตสถานะของสถานะนั้น
 status = DocumentStatus.query.filter_by(batch_id=batch_id,
location=location).first()
 if status:
 # ถ้ามีอยู่แล้วให้แก้ไขข้อมูล
 status.signed_by = signed_by
 status.signed_at = signed_at
 else:
 # ถ้ายังไม่มีให้สร้างใหม่
 status = DocumentStatus(batch_id=batch_id, location=location,
signed_by=signed_by, signed_at=signed_at)
 db.session.add(status)

 db.session.commit()
 flash('Status updated successfully!', 'success')
 return redirect(url_for('update_status', batch_id=batch_id))

 # รับสถานะปัจจุบันทั้งหมดของ batch นี้เพื่อแสดงในหน้า
 statuses = DocumentStatus.query.filter_by(batch_id=batch_id).all()
 return render_template('update_status.html', batch=batch,
statuses=statuses)

if __name__ == '__main__':
 with app.app_context():
 db.create_all()
 app.run(debug=True)

```

การแสดงผลใน Template update\_status.html

Template นี้จะแสดงฟอร์มให้ผู้ใช้สามารถกรอกข้อมูลการลงชื่อและเวลาในแต่ละจุด

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Update Document Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-

```

```

ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Update Document Status for Batch
ID: {{ batch.batch_id }}</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <label>Location</label>
 <select name="location" required>
 <option value="ผู้ส่งเอกสารต้นทาง">ผู้ส่งเอกสารต้นทาง</option>
 <option value="ผู้รับเอกสารต้นทาง">ผู้รับเอกสารต้นทาง</option>
 <option value="พนักงานคัดแยกเอกสาร">พนักงานคัดแยกเอกสาร</option>
 <option value="ผู้ส่งเอกสารปลายทาง">ผู้ส่งเอกสารปลายทาง</option>
 <option value="ผู้รับเอกสารปลายทาง">ผู้รับเอกสารปลายทาง</option>
 </select>
 </div>
 <div class="field">
 <label>Signed By</label>
 <input type="text" name="signed_by" placeholder="Enter Name"
required>
 </div>
 <div class="field">
 <label>Signed At</label>
 <input type="text" name="signed_at" placeholder="YYYY-MM-DD
HH:MM:SS" required>
 </div>
 <button class="ui button primary" type="submit">Update
Status</button>
 </form>

 <h3>Current Statuses</h3>
 <div class="ui relaxed divided list">
 {% for status in statuses %}
 <div class="item">
 <div class="content">
 <p>Location: {{ status.location
}}</p>
 <p>Signed By: {{ status.signed_by
}}</p>
 <p>Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') if status.signed_at else 'N/A'
}}</p>
 </div>
 </div>
 {% endfor %}

```



```

 </div>
 </div>
</body>
</html>

```

#### คำอธิบายการทำงาน

1. การปรับฐานข้อมูล: ฐานข้อมูลได้รับการปรับให้ DocumentStatus จัดเก็บข้อมูลการลงชื่อและเวลาสำหรับแต่ละจุดที่เอกสารต้องผ่าน
2. การอัปเดตสถานะ: ผู้ใช้สามารถเลือกจุดสถานะ (เช่น ผู้ส่งเอกสารต้นทาง) ลงชื่อ (signed\_by) และบันทึกวันที่เวลาที่ (signed\_at)
3. การแสดงสถานะปัจจุบัน: แสดงสถานะของเอกสารในแต่ละจุด

ต้องเปลี่ยนอะไรใน

```

<div class="ui secondary pointing menu">
 Home
 Create Document
Batch
 Search
Document
 Update Status
 Check Status
</div>

```

เนื่องจากฟังก์ชัน update\_status ตอนนี้องค์การ batch\_id เพื่ออัปเดตสถานะเอกสารเฉพาะของ batch นั้น การเปลี่ยนแปลงนี้หมายความว่าในเมนู เราจำเป็นต้องนำทางไปยังหน้า update\_status พร้อมกับ batch\_id ที่ระบุ หากไม่มี batch\_id ในลิงก์ตรง ๆ เราอาจปรับให้ลิงก์ในเมนูนำผู้ใช้ไปยังหน้าอื่นที่พวกเขาสามารถค้นหา batch\_id ก่อน จากนั้นจึงนำพวกเขาไปยังหน้าอัปเดตสถานะ

#### การแก้ไขในเมนู

1. ปรับลิงก์ Update Status ให้ไปที่หน้า search\_documents ก่อน เพื่อให้ผู้ใช้ค้นหา batch\_id ก่อนทำการอัปเดตสถานะ หรือ
2. ถ้า batch\_id เป็นข้อมูลที่สามารถระบุได้ให้ทำการส่ง batch\_id เป็นพารามิเตอร์ไปยัง update\_status

ตัวเลือกที่ 1 เป็นวิธีที่ง่ายที่สุด

#### โค้ดที่ปรับปรุงแล้ว

```

<div class="ui secondary pointing menu">
 Home
 Create Document
Batch
 Search
Document
 <!-- เปลี่ยนลิงก์ Update Status ให้ไปที่หน้า Search Document -->
 Update
Status
 Check Status
</div>

```

## อธิบายการเปลี่ยนแปลง

- แทนที่จะลิงก์ไปที่ `update_status` โดยตรง เราลิงก์ไปที่ `search_documents` ก่อน
- ในหน้า `search_documents` ผู้ใช้จะสามารถค้นหา `batch_id` ได้ จากนั้นสามารถเลือก `batch_id` เพื่อนำพวกเขาไปยังหน้า `update_status` พร้อม `batch_id`

เนื่องจาก `Status: Created` เป็น `status` ของ `batch` ควรปรับการแสดงผลด้วย การแสดงผลควรเป็นแบบ

Batch ID: GU771281 Sender: วช. Receiver: บข. Created At: 2024-11-06 11:48:01 Status: Created Last Updated: N/A

Documents in this Batch: Document ID: GU771281-1 Title: งบประมาณ

ทั้งการแสดงผลของ `search document`, และ `check status`

ในการปรับการแสดงผลเพื่อให้แสดง `status` ของ `batch` แทนที่จะเป็นของ `document` เราจำเป็นต้องปรับปรุงโค้ดในทั้งหน้า `search_documents` และ `check_status` ให้แสดงข้อมูลในรูปแบบที่กำหนด คือแสดง `Batch ID`, `Sender`, `Receiver`, `Created At`, `Status` และ `Last Updated` สำหรับ `batch` และแสดงข้อมูลของ `document` ภายใต้อัตโนมัติ `batch` นั้น

## การปรับปรุงในฟังก์ชัน `search_documents` และ `check_status`

เราจะปรับให้ทั้งสองฟังก์ชันดึงข้อมูล `batch` ที่รวมทั้งสถานะ (`status`) และเวลาอัปเดตล่าสุด (`updated_at`) มาแสดงเป็นหลัก และแสดงรายการ `document` ภายใต้อัตโนมัติ `batch`

ตัวอย่างโค้ด `search_documents` ใน `app.py`

```
@app.route('/search', methods=['GET', 'POST'])
def search_documents():
 results = []
 if request.method == 'POST':
 query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{query}%")) |
 (DocumentBatch.sender_name.like(f"%{query}%")) |
 (DocumentBatch.receiver_name.like(f"%{query}%"))
).all()

 # ดึงเอกสารทั้งหมดที่อยู่ภายใต้อัตโนมัติ batch
 for batch in batches:
 documents =
 Document.query.filter_by(batch_id=batch.batch_id).all()
 results.append({
 'batch': batch,
 'documents': documents
 })

 if not results:
 flash('No documents found.', 'warning')
```

```
 return render_template('search.html', results=results)
```

ตัวอย่างโค้ด check\_status ใน app.py

```
@app.route('/check_status', methods=['GET', 'POST'])
```

```
def check_status():
```

```
 search_query = None
```

```
 page = request.args.get('page', 1, type=int)
```

```
 per_page = 50
```

```
 if request.method == 'POST':
```

```
 search_query = request.form['query']
```

```
 if search_query:
```

```
 # ค้นหา batch ที่ตรงกับ query
```

```
 batches = DocumentBatch.query.filter(
```

```
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
```

```
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
```

```
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
```

```
).paginate(page=page, per_page=per_page)
```

```
 else:
```

```
 # แสดง batch ล่าสุดทั้งหมด แบ่งเป็นหน้า
```

```
 batches =
```

```
DocumentBatch.query.order_by(DocumentBatch.created_at.desc()).paginate(page=p
age, per_page=per_page)
```

```
 return render_template('check_status.html', batches=batches,
search_query=search_query)
```

การปรับปรุง Template search.html และ check\_status.html

Template ทั้งสองนี้จะแสดง batch เป็นหลัก พร้อมกับแสดง status และ updated\_at ใน batch และแสดงรายการ document ภายใต้ batch นั้น

Template ที่ปรับปรุง: search.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
 <meta charset="UTF-8">
```

```
 <title>Search Documents</title>
```

```
 <link rel="stylesheet"
```

```
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
```

```
</head>
```

```
<body>
```

```
 <div class="ui container">
```

```
 <h2 class="ui center aligned header">Search Documents</h2>
```

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

```

<form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender or Receiver" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
</form>

{% if results %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: {{
result.batch.status }}</p>
 <p>Last Updated: {{
result.batch.updated_at.strftime('%Y-%m-%d %H:%M:%S') if
result.batch.updated_at else 'N/A' }}</p>
 <div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in result.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID:
{{ document.doc_id }}</p>
 <p>Title: {{
document.title }}</p>
 </div>
 </div>
 {% endfor %}
 </div>
 </div>
 </div>
 {% endfor %}
 </div>
{% else %}
 <div class="ui message">
 <p>No documents found.</p>
 </div>

```

```

 {% endif %}
 </div>
</body>
</html>

Template ที่ปรับปรุง: check_status.html
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Check Document Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Check Document Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender or Receiver" value="{{ search_query or '' }}">
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if batches.items %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for batch in batches.items %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ batch.batch_id }}</h4>
 <p>Sender: {{ batch.sender_name
}}
 Receiver: {{
batch.receiver_name }}</p>
 <p>Created At: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: {{ batch.status
}}</p>
 <p>Last Updated: {{
batch.updated_at.strftime('%Y-%m-%d %H:%M:%S') if batch.updated_at else 'N/A'
}}</p>
 <div class="ui list">

```

```

 <h5>Documents in this Batch:</h5>
 {% for document in batch.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID: {{
document.doc_id }}</p>

 <p>Title: {{
document.title }}</p>

 </div>
 </div>
 {% endfor %}
 </div>
</div>
{% endif %}
</div>

<!-- Pagination -->
<div class="ui pagination menu">
 {% if batches.has_prev %}
 <a class="item" href="{% url_for('check_status',
page=batches.prev_num) %}">Previous
 {% else %}
 <div class="disabled item">Previous</div>
 {% endif %}

 {% for page_num in batches.iter_pages(left_edge=1,
right_edge=1, left_current=2, right_current=2) %}
 {% if page_num %}
 {% if page_num == batches.page %}
 <div class="active item">{{ page_num }}</div>
 {% else %}
 <a class="item" href="{% url_for('check_status',
page=page_num) %}">{{ page_num }}
 {% endif %}
 {% else %}
 <div class="disabled item">...</div>
 {% endif %}
 {% endfor %}

 {% if batches.has_next %}
 <a class="item" href="{% url_for('check_status',
page=batches.next_num) %}">Next
 {% else %}
 <div class="disabled item">Next</div>
 {% endif %}
</div>
{% else %}

```

```

 <div class="ui message">
 <p>No documents found.</p>
 </div>
 {% endif %}
</div>
</body>
</html>

```

คำอธิบาย

- แสดงข้อมูล **batch** เป็นหลัก: แสดง **batch\_id**, ผู้ส่ง, ผู้รับ, วันที่สร้าง, สถานะ

เมื่อมีการสร้าง **batch** จะต้องมีการเพิ่มรายการเข้าไปใน **document\_status** ทันที โดยระบุ **location** ตาม **sender\_name**, **signed\_by** คือ ชื่อผู้ส่ง, **signed\_at** ให้ใช้จากเวลาเดียวกับ **created\_at** ของ **document\_batch**

เพื่อให้มีการสร้างรายการสถานะ (**document\_status**) ทันทีเมื่อสร้าง **batch** เราสามารถปรับฟังก์ชัน **create\_batch** ให้เพิ่มข้อมูลนี้ได้ โดยเมื่อ **DocumentBatch** ถูกสร้างแล้ว เราจะเพิ่มรายการใน **DocumentStatus** ตามที่กำหนดไว้

ปรับปรุงโค้ดใน **create\_batch** ใน **app.py**

```

from datetime import datetime
from models import db, DocumentBatch, Document, DocumentStatus

@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 doc_id = generate_batch_id() # สร้าง batch_id ใหม่
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 # รายชื่อเอกสารที่เป็น List
 created_at = datetime.utcnow() # ใช้เวลาเดียวกันในการสร้าง document batch และ
 # document status

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name, created_at=created_at)
 db.session.add(batch)
 db.session.commit()

 # เพิ่มสถานะเริ่มต้นใน DocumentStatus ทันทีเมื่อสร้าง batch
 initial_status = DocumentStatus(
 batch_id=batch.batch_id,
 location=sender_name,
 signed_by=sender_name,
 signed_at=created_at
)
 db.session.add(initial_status)

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch

```

```

for index, title in enumerate(document_titles, start=1):
 new_document = Document(
 doc_id=f"{doc_id}-{index}", # กำหนดเลขรายการเอกสารตามรูปแบบ doc_id
 title=title,
 batch_id=batch.batch_id
)
 db.session.add(new_document)

 db.session.commit()
 flash(f'Document batch with ID {doc_id} created successfully!',
'success')
 return redirect(url_for('create_batch'))

return render_template('create_batch.html', doc_id=doc_id)

```

คำอธิบายการทำงานที่เพิ่มขึ้น

- สร้างสถานะเริ่มต้นใน **DocumentStatus**: เมื่อสร้าง **DocumentBatch** แล้ว เราเพิ่มสถานะเริ่มต้นใน **DocumentStatus** ทันทีโดยใช้ข้อมูลต่อไปนี้:
    - **batch\_id**: ใช้ **batch\_id** ของ **DocumentBatch** ที่เพิ่งสร้าง
    - **location**: ตั้งค่าเป็น **sender\_name** ของผู้ส่ง
    - **signed\_by**: ตั้งค่าเป็น **sender\_name** เช่นเดียวกัน
    - **signed\_at**: ตั้งค่าเป็น **created\_at** ของ **DocumentBatch**
- การ update status เมื่อผู้ใช้ระบุ batch id หรือค้นหา batch id แล้วให้ผู้ใช้ทำการกดปุ่ม update status เพื่อระบุข้อมูล document status ดังนี้

**location** ให้ผู้ใช้เลือก ผู้รับเอกสารต้นทาง, พนักงานคัดแยกเอกสาร, ผู้ส่งเอกสารปลายทาง, ผู้รับเอกสารปลายทาง อย่างใดอย่างหนึ่ง เพื่อจัดเก็บลง **location** และ รับข้อมูลลงชื่อ เพื่อใส่ใน **signed\_by** เมื่อกด update ก็ให้ระบุเวลา ลงใน **signed\_at**

โดยการค้นหา อาจมีหลาย batch id ขึ้นมาได้

เพื่อให้ผู้ใช้สามารถอัปเดตสถานะของเอกสารได้ตามที่ต้องการ (ระบุ batch\_id หรือค้นหา batch\_id และเลือก location เพื่อบันทึกข้อมูลใน **DocumentStatus**) เราสามารถสร้างหน้า **update\_status** ที่ให้ผู้ใช้เลือก location และบันทึกข้อมูลการลงชื่อ (**signed\_by**) และบันทึกเวลา (**signed\_at**) ได้ตามที่กำหนด

โครงสร้าง **app.py** สำหรับ **update\_status**

1. ฟังก์ชัน **search\_batches** เพื่อค้นหา batch\_id ที่ผู้ใช้ต้องการอัปเดต
2. ฟังก์ชัน **update\_status** ที่ให้ผู้ใช้เลือก location และบันทึกข้อมูลการลงชื่อ

โค้ดใน **app.py**

```

from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for, flash
from models import db, DocumentBatch, DocumentStatus

```

```

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'

```



```

db.init_app(app)

ฟังก์ชันค้นหา batch เพื่อเตรียมสำหรับการอัปเดตสถานะ
@app.route('/search_batches', methods=['GET', 'POST'])
def search_batches():
 search_query = None
 batches = []

 if request.method == 'POST':
 search_query = request.form['query']
 batches = DocumentBatch.query.filter(DocumentBatch.batch_id.like(f"%{search_query}%")).all()

 if not batches:
 flash('No batches found.', 'warning')

 return render_template('search_batches.html', batches=batches,
 search_query=search_query)

ฟังก์ชันสำหรับอัปเดตสถานะของเอกสาร
@app.route('/update_status/<batch_id>', methods=['GET', 'POST'])
def update_status(batch_id):
 batch = DocumentBatch.query.get_or_404(batch_id)

 if request.method == 'POST':
 location = request.form['location']
 signed_by = request.form['signed_by']
 signed_at = datetime.utcnow()

 # ตรวจสอบว่ามีกรบันทึกสถานะสำหรับ Location นั้นหรือยัง
 status = DocumentStatus.query.filter_by(batch_id=batch_id,
 location=location).first()
 if status:
 # ถ้ามีอยู่แล้ว ให้แก้ไขข้อมูลการลงชื่อและเวลาที่ลงชื่อ
 status.signed_by = signed_by
 status.signed_at = signed_at
 else:
 # ถ้าไม่มี ให้สร้างสถานะใหม่สำหรับ Location นั้น
 status = DocumentStatus(
 batch_id=batch_id,
 location=location,
 signed_by=signed_by,
 signed_at=signed_at
)
 db.session.add(status)

 db.session.commit()
 flash('Status updated successfully!', 'success')

```

```

 return redirect(url_for('search_batches'))

 return render_template('update_status.html', batch=batch)

if __name__ == '__main__':
 with app.app_context():
 db.create_all()
 app.run(debug=True)

```

## Template สำหรับ search\_batches.html

หน้านี้จะให้ผู้ใช้ค้นหา batch\_id และแสดงรายการ batch ที่ค้นพบ ให้ผู้ใช้กดปุ่ม Update Status เพื่อนำไปที่หน้า update\_status

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Search Batches</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Batches</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID
to search" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if batches %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for batch in batches %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ batch.batch_id }}</h4>
 <p>Sender: {{ batch.sender_name }}
 Receiver: {{
batch.receiver_name }}</p>
 <p>Created At: {{

```

```

batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <a class="ui button" href="{ {
url_for('update_status', batch_id=batch.batch_id) }}">Update Status
 </div>
 </div>
 {% endfor %}
</div>
{% else %}
 <div class="ui message">
 <p>No batches found.</p>
 </div>
{% endif %}
</div>
</body>
</html>

```

## Template สำหรับ update\_status.html

หน้านี้ให้ผู้ใช้เลือก location และป้อนข้อมูลการลงชื่อ (signed\_by) แล้วบันทึกเวลา (signed\_at)

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Update Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Update Document Status for Batch
ID: {{ batch.batch_id }}</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <label>Location</label>
 <select name="location" required>
 <option value="ผู้รับเอกสารต้นทาง">ผู้รับเอกสารต้นทาง</option>
 <option value="พนักงานคัดแยกเอกสาร">พนักงานคัดแยกเอกสาร</option>
 <option value="ผู้ส่งเอกสารปลายทาง">ผู้ส่งเอกสารปลายทาง</option>
 <option value="ผู้รับเอกสารปลายทาง">ผู้รับเอกสารปลายทาง</option>
 </select>
 </div>
 <div class="field">
 <label>Signed By</label>
 <input type="text" name="signed_by" placeholder="Enter Name"
required>

```

```

 </div>
 <button class="ui button primary" type="submit">Update
Status</button>
 </form>

 <h3>Current Statuses</h3>
 <div class="ui relaxed divided list">
 {% for status in batch.statuses %}
 <div class="item">
 <div class="content">
 <p>Location: {{ status.location
}}</p>
 <p>Signed By: {{ status.signed_by
}}</p>
 <p>Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') if status.signed_at else 'N/A'
}}</p>
 </div>
 </div>
 {% endfor %}
 </div>
</body>
</html>

```

คำอธิบายการทำงาน

1. หน้า **search\_batches**: ผู้ใช้สามารถค้นหา batch\_id ได้ และจะแสดงรายการ batch ที่ค้นพบพร้อมปุ่ม Update Status สำหรับการอัปเดตสถานะ
2. หน้า **update\_status**: ผู้ใช้สามารถเลือก location และป้อนชื่อ (signed\_by) เพื่อบันทึกลงใน DocumentStatus พร้อมกับเวลา (signed\_at)

ขอ menu.html ใหม่ด้วย

แน่นอนครับ นี่คือโค้ด menu.html ใหม่ที่สามารถใช้กับฟังก์ชันต่าง ๆ ในโปรเจกต์ของคุณ โดยใช้ Semantic UI เพื่อสร้างเมนูสำหรับนำทางไปยังหน้าเพจที่จำเป็น:

```

<div class="ui secondary pointing menu">
 Home
 Create Document
Batch
 Search and Update
Status
 Check Status
</div>

```

คำอธิบาย

- **Home**: ลิงก์ไปยังหน้าแรก
- **Create Document Batch**: ลิงก์ไปยังหน้าสร้าง batch ใหม่

- **Search and Update Status:** ลิงก์ไปยังหน้า `search_batches` ที่ให้ผู้ใช้สามารถค้นหา `batch_id` และอัปเดตสถานะเอกสาร
- **Check Status:** ลิงก์ไปยังหน้า `check_status` เพื่อดูสถานะปัจจุบันของเอกสารใน `batch`

โค้ดนี้จะให้ผู้ใช้สามารถเข้าถึงแต่ละฟังก์ชันได้ง่ายและชัดเจน

การค้นหาและปรับปรุงสถานะ รายการที่แสดงต้องแสดงรายการ `document` ใน `batch` ด้วย

ถ้าต้องการให้การค้นหาและปรับปรุงสถานะแสดงรายการ `document` ทั้งหมดภายใน `batch` ที่ค้นหาได้ คุณสามารถปรับปรุงหน้า `search_batches.html` ให้แสดงรายการ `document` ภายในแต่ละ `batch` นั้น ๆ เพื่อให้ผู้ใช้เห็นข้อมูลของเอกสารทั้งหมดที่เกี่ยวข้องกับ `batch` ที่ค้นหาได้

ปรับปรุงฟังก์ชัน `search_batches` ใน `app.py`

ให้ดึงข้อมูล `Document` ที่เกี่ยวข้องกับ `batch` เพื่อให้เห็นใน `search_batches.html` ด้วย

```
@app.route('/search_batches', methods=['GET', 'POST'])
def search_batches():
 search_query = None
 results = []

 if request.method == 'POST':
 search_query = request.form['query']
 batches = DocumentBatch.query.filter(DocumentBatch.batch_id.like(f"%{search_query}%")).all()

 for batch in batches:
 documents = Document.query.filter_by(batch_id=batch.batch_id).all()
 results.append({
 'batch': batch,
 'documents': documents
 })

 if not results:
 flash('No batches found.', 'warning')

 return render_template('search_batches.html', results=results,
 search_query=search_query)
```

ปรับปรุง Template `search_batches.html`

ให้แสดงข้อมูล `batch` และรายการ `document` ภายใน `batch` นั้น รวมถึงปุ่ม `Update Status` สำหรับไปยังหน้าอัปเดตสถานะ

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Search and Update Status</title>
```

```

 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Batches and Update
Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID
to search" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if results %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: {{
result.batch.status }}</p>
 <p>Last Updated: {{
result.batch.updated_at.strftime('%Y-%m-%d %H:%M:%S') if
result.batch.updated_at else 'N/A' }}</p>

 <div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in result.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID:
{{ document.doc_id }}</p>
 <p>Title: {{
document.title }}</p>

```

```

 </div>
 </div>
 {% endfor %}
</div>
<a class="ui button" href="{
url_for('update_status', batch_id=result.batch.batch_id) }}">Update
Status
</div>
</div>
{% endfor %}
</div>
{% else %}
<div class="ui message">
<p>No batches found.</p>
</div>
{% endif %}
</div>
</body>
</html>

```

คำอธิบายการทำงาน

- ฟังก์ชัน **search\_batches**: ค้นหา batch ตาม batch\_id ที่ผู้ใช้ป้อนใน search\_query และดึงรายการ document ที่เกี่ยวข้องกับแต่ละ batch เพื่อแสดงผล
- หน้า **search\_batches.html**: แสดงข้อมูล batch เช่น batch\_id, ผู้ส่ง (sender\_name), ผู้รับ (receiver\_name), วันที่สร้าง (created\_at), สถานะ (status), เวลาที่อัปเดตล่าสุด (Last Updated) และแสดงรายการ document ทั้งหมดใน batch นั้น ๆ โดยแสดง doc\_id และ title ของแต่ละ document พร้อมปุ่ม Update Status ที่จะนำผู้ใช้ไปยังหน้าสำหรับอัปเดตสถานะของ batch นั้น

เอ็งอย่าได้หน้าลืมหูลงสิ

การค้นหาต้องค้นจากอะไรก็ได้ ไม่ว่าจะเป็น batch id sender receiver

ขอโทษด้วยครับที่ลืมข้อกำหนดนั้น! ผมจะแก้ไขฟังก์ชัน search\_batches ให้รองรับการค้นหาจาก batch\_id, sender\_name, หรือ receiver\_name ได้ เพื่อให้ผู้ใช้สามารถค้นหาจากเงื่อนไขใดก็ได้ตามที่คุณกำหนด

ปรับปรุงฟังก์ชัน search\_batches ใน app.py

ฟังก์ชันนี้จะค้นหา batch ตามเงื่อนไขที่ผู้ใช้ระบุใน search\_query ซึ่งอาจเป็น batch\_id, sender\_name, หรือ receiver\_name

```

@app.route('/search_batches', methods=['GET', 'POST'])
def search_batches():
 search_query = None
 results = []

 if request.method == 'POST':
 search_query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query ใน `batch_id`, `sender_name`, หรือ

```

```

`receiver_name`
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).all()

 for batch in batches:
 # ดึงข้อมูลเอกสารทั้งหมดที่เกี่ยวข้องกับ batch
 documents =
Document.query.filter_by(batch_id=batch.batch_id).all()
 results.append({
 'batch': batch,
 'documents': documents
 })

 if not results:
 flash('No batches found.', 'warning')

 return render_template('search_batches.html', results=results,
search_query=search_query)

```

Template สำหรับ search\_batches.html

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Search and Update Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Batches and Update
Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender, or Receiver" value="{{ search_query or '' }}" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if results %}
 <h3>Search Results</h3>

```



```

 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: {{
result.batch.status }}</p>
 <p>Last Updated: {{
result.batch.updated_at.strftime('%Y-%m-%d %H:%M:%S') if
result.batch.updated_at else 'N/A' }}</p>

 <div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in result.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID:
{{ document.doc_id }}</p>
 <p>Title: {{
document.title }}</p>
 </div>
 </div>
 {% endfor %}
 </div>
 <a class="ui button" href="{{
url_for('update_status', batch_id=result.batch.batch_id) }}">Update
Status
 </div>
 </div>
 {% endfor %}
 </div>
{% else %}
 <div class="ui message">
 <p>No batches found.</p>
 </div>
{% endif %}
</div>
</body>
</html>

```

## คำอธิบาย

- การค้นหาจากหลายฟิลด์: ตอนนี้ `search_batches` สามารถค้นหาจาก `batch_id`, `sender_name`, หรือ `receiver_name` ได้โดยใช้ LIKE กับทั้งสามฟิลด์
- แสดงผล: เมื่อพบผลลัพธ์ จะแสดงข้อมูล `batch` และรายการ `document` ทั้งหมดภายใน `batch` นั้น รวมถึงปุ่ม `Update Status` สำหรับไปที่หน้าอัปเดตสถานะ

การ `update status` เมื่อ `update` แล้ว และกลับมาที่ ค้นหาและปรับปรุงสถานะ ไม่ควรขึ้นว่า `No batches found`. ควร `query batch id` ที่ `update` นั้นขึ้นมาและแสดงสถานะ ของ `document_status` ในการแสดงผล ในส่วนของ `Status`: และ `Last Updated: N/A`

เพื่อให้การอัปเดตสถานะทำงานตามที่ต้องการ หลังจากที่ทำกรอัปเดตสถานะเรียบร้อยแล้ว เมื่อผู้ใช้กลับมาที่หน้า “ค้นหาและปรับปรุงสถานะ” (`search_batches.html`) ระบบจะต้องแสดง `batch` ที่เพิ่งทำการอัปเดต พร้อมกับแสดงสถานะล่าสุดจาก `DocumentStatus` โดยไม่ต้องขึ้นข้อความว่า “No batches found.”

## ปรับปรุงฟังก์ชัน `update_status` ใน `app.py`

ให้ `update_status` ทำการรีไ้เรกต์กลับไปยัง `search_batches` พร้อมส่ง `batch_id` ที่เพิ่งอัปเดต เพื่อแสดงผลลัพธ์ทันที

```
@app.route('/update_status/<batch_id>', methods=['GET', 'POST'])
def update_status(batch_id):
 batch = DocumentBatch.query.get_or_404(batch_id)

 if request.method == 'POST':
 location = request.form['location']
 signed_by = request.form['signed_by']
 signed_at = datetime.utcnow()

 # ตรวจสอบว่ามีกรบันทึกสถานะสำหรับ Location นั้นหรือยัง
 status = DocumentStatus.query.filter_by(batch_id=batch_id,
 location=location).first()
 if status:
 # ถ้ามีอยู่แล้ว ให้แก้ไขข้อมูลการลงชื่อและเวลาที่ลงชื่อ
 status.signed_by = signed_by
 status.signed_at = signed_at
 else:
 # ถ้าไม่มี ให้สร้างสถานะใหม่สำหรับ Location นั้น
 status = DocumentStatus(
 batch_id=batch_id,
 location=location,
 signed_by=signed_by,
 signed_at=signed_at
)
 db.session.add(status)

 db.session.commit()
 flash('Status updated successfully!', 'success')
 return redirect(url_for('search_batches', query=batch_id)) #
รีไ้เรกต์ไปที่หน้า search_batches พร้อม batch_id
```

```
return render_template('update_status.html', batch=batch)
```

ปรับปรุงฟังก์ชัน `search_batches` ให้รองรับการค้นหา `batch_id` ที่เพิ่งอัปเดต

ในฟังก์ชัน `search_batches` เราจะตรวจสอบว่ามี `query` ใน `request.args` หรือไม่ หากมี จะใช้ `batch_id` นี้เพื่อค้นหาและแสดงผลลัพธ์ทันที โดยดึงข้อมูลสถานะล่าสุดจาก `DocumentStatus` มาด้วย

```
@app.route('/search_batches', methods=['GET', 'POST'])
def search_batches():
 search_query = request.args.get('query', None)
 results = []

 if request.method == 'POST' or search_query:
 # รับค่า query จากฟอร์มหรือจากพารามิเตอร์
 if not search_query:
 search_query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query ใน `batch_id`, `sender_name`, หรือ
 # `receiver_name`
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).all()

 for batch in batches:
 # ดึงข้อมูลเอกสารทั้งหมดที่เกี่ยวข้องกับ batch
 documents =
 Document.query.filter_by(batch_id=batch.batch_id).all()

 # ดึงสถานะล่าสุดจาก DocumentStatus
 latest_status =
 DocumentStatus.query.filter_by(batch_id=batch.batch_id).order_by(DocumentStat
 us.signed_at.desc()).first()
 status_info = {
 'status': latest_status.location if latest_status else 'N/A',
 'last_updated': latest_status.signed_at.strftime('%Y-%m-%d
 %H:%M:%S') if latest_status and latest_status.signed_at else 'N/A'
 }

 results.append({
 'batch': batch,
 'documents': documents,
 'status_info': status_info
 })

 if not results:
 flash('No batches found.', 'warning')
```

```
 return render_template('search_batches.html', results=results,
search_query=search_query)
```

ปรับปรุง Template search\_batches.html ให้แสดง Status และ Last Updated ของ DocumentStatus

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Search and Update Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Batches and Update
Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender, or Receiver" value="{{ search_query or '' }}" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if results %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: {{
result.status_info.status }}</p>
 <p>Last Updated: {{
result.status_info.last_updated }}</p>
 </div>
 </div>
 {% endfor %}
 </div>
 {% endif %}
 </div>
</body>
</html>
```

```

 <div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in result.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID: {{
document.doc_id }}</p>
 <p>Title: {{
document.title }}</p>
 </div>
 </div>
 {% endfor %}
 </div>
 <a class="ui button" href="{{
url_for('update_status', batch_id=result.batch.batch_id) }}">Update
Status
 </div>
 {% endfor %}
{% else %}
 <div class="ui message">
 <p>No batches found.</p>
 </div>
{% endif %}
</div>
</body>
</html>

```

#### คำอธิบายการทำงาน

- การรีไดเรกต์หลังอัปเดต: หลังจากอัปเดตสถานะ ระบบจะรีไดเรกต์กลับไป `search_batches` พร้อม `batch_id` ที่เพิ่งอัปเดตในพารามิเตอร์ `query`
- การค้นหาจาก **query**: หากมี `query` ใน `request.args` ระบบจะค้นหา `batch` และแสดงข้อมูล `DocumentStatus` ล่าสุดของ `batch` นั้น
- การแสดง **Status** และ **Last Updated**: Template จะแสดงสถานะและเวลาที่อัปเดตล่าสุดของ `DocumentStatus`

[http://127.0.0.1:5000/check\\_status](http://127.0.0.1:5000/check_status)

ยังแสดง **status** ไม่ถูกต้อง ขอให้ทำการแสดงเหมือนกรณีการค้นหา

เพื่อให้การแสดงผลในหน้า `/check_status` มีความสอดคล้องกับหน้า `search_batches` และแสดง **status** และ **last\_updated** ของ `DocumentStatus` สำหรับแต่ละ `batch` อย่างถูกต้อง เราจะปรับปรุงฟังก์ชัน `check_status` ใน `app.py` เพื่อให้ดึงข้อมูลสถานะล่าสุดจาก `DocumentStatus` และแสดงผลในรูปแบบเดียวกับหน้า `search_batches.html`

ปรับปรุงฟังก์ชัน `check_status` ใน `app.py`

ฟังก์ชันนี้จะดึงข้อมูล `batch` ทั้งหมด (หรือจากการค้นหา) พร้อมดึงข้อมูลสถานะล่าสุดจาก `DocumentStatus` เพื่อนำไปแสดงในหน้า `check_status.html`

```
@app.route('/check_status', methods=['GET', 'POST'])
def check_status():
 search_query = request.args.get('query', None)
 page = request.args.get('page', 1, type=int)
 per_page = 50
 results = []

 if request.method == 'POST' or search_query:
 # รับค่า query จากฟอร์มหรือจากพารามิเตอร์
 if not search_query:
 search_query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query ใน `batch_id`, `sender_name`, หรือ
 # `receiver_name`
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).paginate(page=page, per_page=per_page)
 else:
 # แสดง batch ทั้งหมด โดยเรียงลำดับจากล่าสุด แบ่งเป็นหน้า
 batches = DocumentBatch.query.order_by(DocumentBatch.created_at.desc()).paginate(page=page, per_page=per_page)

 for batch in batches.items():
 # ดึงข้อมูลเอกสารทั้งหมดที่เกี่ยวข้องกับ batch
 documents = Document.query.filter_by(batch_id=batch.batch_id).all()

 # ดึงสถานะล่าสุดจาก DocumentStatus
 latest_status = DocumentStatus.query.filter_by(batch_id=batch.batch_id).order_by(DocumentStatus.signed_at.desc()).first()
 status_info = {
 'status': latest_status.location if latest_status else 'N/A',
 'last_updated': latest_status.signed_at.strftime('%Y-%m-%d %H:%M:%S') if latest_status and latest_status.signed_at else 'N/A'
 }

 results.append({
 'batch': batch,
 'documents': documents,
 'status_info': status_info
 })
```

```
 return render_template('check_status.html', results=results,
search_query=search_query, batches=batches)
```

## ปรับปรุง Template check\_status.html

ให้แสดงข้อมูล batch และรายการ document ภายใน batch พร้อมสถานะ (status) และเวลาที่อัปเดตล่าสุด (last\_updated) ในรูปแบบเดียวกับ search\_batches.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Check Document Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Check Document Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender, or Receiver" value="{{ search_query or '' }}" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if results %}
 <h3>Document Status</h3>
 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: {{
result.status_info.status }}</p>
 <p>Last Updated: {{
```

```

result.status_info.last_updated }}</p>

 <div class="ui list">
 <h5>Documents in this Batch:</h5>
 {% for document in result.documents %}
 <div class="item">
 <i class="file outline icon"></i>
 <div class="content">
 <p>Document ID: {{
{{ document.doc_id }}</p>

 <p>Title: {{
document.title }}</p>

 </div>
 </div>
 {% endfor %}
 </div>

 </div>
 {% endfor %}
 </div>

 <!-- Pagination -->
 <div class="ui pagination menu">
 {% if batches.has_prev %}
 <a class="item" href="{{ url_for('check_status',
page=batches.prev_num) }}">Previous
 {% else %}
 <div class="disabled item">Previous</div>
 {% endif %}

 {% for page_num in batches.iter_pages(left_edge=1,
right_edge=1, left_current=2, right_current=2) %}
 {% if page_num %}
 {% if page_num == batches.page %}
 <div class="active item">{{ page_num }}</div>
 {% else %}
 <a class="item" href="{{ url_for('check_status',
page=page_num) }}">{{ page_num }}
 {% endif %}
 {% else %}
 <div class="disabled item">...</div>
 {% endif %}
 {% endfor %}

 {% if batches.has_next %}
 <a class="item" href="{{ url_for('check_status',
page=batches.next_num) }}">Next
 {% else %}
 <div class="disabled item">Next</div>

```



```

 {% endif %}
 </div>
 {% else %}
 <div class="ui message">
 <p>No documents found.</p>
 </div>
 {% endif %}
</div>
</body>
</html>

```

#### คำอธิบายการทำงาน

- การค้นหาจากหลายฟิลด์: ฟังก์ชัน `check_status` สามารถค้นหาตาม `batch_id`, `sender_name`, หรือ `receiver_name` และดึงข้อมูล `DocumentStatus` ล่าสุดเพื่อแสดงสถานะและเวลาที่อัปเดต
- แสดงสถานะและเวลาที่อัปเดต: Template จะแสดงสถานะล่าสุด (`status`) และเวลาที่อัปเดต (`last_updated`) ของ `batch` ในรูปแบบที่สอดคล้องกับหน้า `search_batches.html`

ในการสร้าง `batch` ขอให้ทำการสร้าง `qr code` ให้ด้วย และเมื่อสร้าง `batch` แล้วขอให้แสดงผลด้านล่างของจอภาพ แสดงข้อมูลต่าง ๆ ที่สร้างขึ้น รวมทั้ง `qr code` ด้วย การแสดงผลเช่น

Batch ID: QQ496874

Sender: บข Receiver: วข

Created At: 2024-11-06 12:19:16

Status: ผู้ส่งเอกสารต้นทาง

Last Updated: 2024-11-06 12:36:34

Documents in this Batch:

Document ID: QQ496874-1

Title: หนังสืองบประมาณ Document ID: QQ496874-2

Title: หนังสือส่งกระทรวงการคลัง

โดยสถานะ `status` ตอนสร้างจะต้องเป็น ผู้ส่งเอกสารต้นทาง (ใน `documentn_status` field location เท่ากับ ผู้ส่งเอกสารต้นทาง)

เพื่อให้การสร้าง `batch` สร้าง `QR code` และแสดงผลที่หน้าจอ เราจะปรับฟังก์ชัน `create_batch` ให้ทำการสร้าง `QR code` จาก `batch_id` และบันทึกไว้ในโฟลเดอร์ `static/qr_codes` นอกจากนี้ เราจะเพิ่มโค้ดให้แสดงผลข้อมูลที่สร้างขึ้น รวมถึง `QR code` หลังการสร้าง

ปรับปรุงโค้ดใน `app.py` สำหรับการสร้าง `QR Code` และแสดงผล

```

import qrcode
import os
from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for, flash

```

```

from models import db, DocumentBatch, Document, DocumentStatus

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)

ฟังก์ชันสำหรับสร้าง batch พร้อมสร้าง QR Code และบันทึกสถานะเริ่มต้น
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 doc_id = generate_batch_id() # สร้าง batch_id แบบง่าย
 created_at = datetime.utcnow() # เวลาเดียวกันสำหรับทั้ง batch และสถานะเริ่มต้น

 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายการชื่อเอกสารใน batch นี้

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name, created_at=created_at)
 db.session.add(batch)
 db.session.commit()

 # สร้างสถานะเริ่มต้นใน DocumentStatus
 initial_status = DocumentStatus(
 batch_id=batch.batch_id,
 location="ผู้ส่งเอกสารต้นทาง", # สถานะเริ่มต้น
 signed_by=sender_name,
 signed_at=created_at
)
 db.session.add(initial_status)

 # สร้างเอกสารแต่ละรายการที่เชื่อมโยงกับ batch
 documents = []
 for index, title in enumerate(document_titles, start=1):
 doc_id = f"{batch.batch_id}-{index}"
 new_document = Document(doc_id=doc_id, title=title,
 batch_id=batch.batch_id)
 db.session.add(new_document)
 documents.append(new_document)

 # สร้าง QR Code สำหรับ batch นี้
 qr = qrcode.make(batch.batch_id)
 qr_path = os.path.join('static', 'qr_codes', f"{batch.batch_id}.png")
 qr.save(qr_path)

```

```

 db.session.commit()
 flash(f'Document batch with ID {batch.batch_id} created
successfully!', 'success')

 # ส่งข้อมูลไปแสดงในหน้าเว็บหลังสร้าง batch
 return render_template('create_batch.html', batch=batch,
documents=documents, qr_path=qr_path)

 return render_template('create_batch.html', doc_id=doc_id)

```

Template create\_batch.html สำหรับการแสดงผลหลังสร้าง Batch

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" required>
 </div>
 <div class="field">
 <label>Document Titles</label>
 <input type="text" name="document_titles" placeholder="Title
1, Title 2, ..." required>
 </div>
 <button class="ui button primary" type="submit">Create
Batch</button>
 </form>

 {% if batch %}
 <div class="ui segment">
 <h3>Batch Created Successfully</h3>
 <p>Batch ID: {{ batch.batch_id }}</p>
 <p>Sender: {{ batch.sender_name }}</p>
 </div>
 {% endif %}
 </div>
</body>
</html>

```

```

Receiver: {{ batch.receiver_name }}</p>
 <p>Created At: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: ผู้ส่งเอกสารต้นทาง</p>
 <p>Last Updated: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>

 <h4>Documents in this Batch:</h4>

 {% for document in documents %}
 Document ID: {{ document.doc_id
}}
 Title: {{ document.title }}
 {% endfor %}

 <h4>QR Code</h4>

 </div>
 {% endif %}
</div>
</body>
</html>

```

คำอธิบายการทำงาน

1. การสร้าง **QR Code**: หลังจากสร้าง DocumentBatch และบันทึกสถานะเริ่มต้นใน DocumentStatus ระบบจะสร้าง QR code จาก batch\_id และบันทึกไว้ในโฟลเดอร์ static/qr\_codes ในรูปแบบไฟล์ PNG
2. การแสดงผลในหน้าเว็บ: หลังจากสร้าง batch เรียบร้อย ข้อมูลของ batch จะถูกส่งไปแสดงผลในหน้า create\_batch.html รวมถึง QR code ที่สร้างขึ้น
3. ข้อมูลที่แสดง: ข้อมูล Batch ID, ผู้ส่ง (Sender), ผู้รับ (Receiver), เวลาสร้าง (Created At), สถานะ (Status), เวลาอัปเดตล่าสุด (Last Updated), รายการเอกสารภายใน batch, และ QR code

การสร้างเอกสาร document doc\_id เอาแบบเดิมสักรับจะเปลี่ยนทำไม ไม่ได้ให้เปลี่ยน มันต้องมีหลายรายการ หลายบรรทัด

ขอโทษด้วยครับที่เข้าใจผิด ผมจะปรับโค้ดกลับไปใช้รูปแบบเดิมของ doc\_id สำหรับแต่ละ document เพื่อให้แต่ละรายการสามารถมีหลายเอกสารได้ โดยจะเพิ่มบรรทัดแยกแต่ละรายการตามที่กำหนด

โค้ดที่ปรับปรุงใน app.py สำหรับการสร้าง Document หลายรายการโดยใช้ doc\_id รูปแบบเดิม

```

import qrcode
import os
from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for, flash
from models import db, DocumentBatch, Document, DocumentStatus

```

```

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)

```

```

ฟังก์ชันสำหรับสร้าง batch พร้อมสร้าง QR Code และบันทึกสถานะเริ่มต้น
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 batch_id = generate_batch_id() # สร้าง batch_id แบบง่าย
 created_at = datetime.utcnow() # เวลาเดียวกันสำหรับทั้ง batch และสถานะเริ่มต้น

 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายการเอกสารใน batch นี้

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=batch_id, sender_name=sender_name,
 receiver_name=receiver_name, created_at=created_at)
 db.session.add(batch)
 db.session.commit()

 # สร้างสถานะเริ่มต้นใน DocumentStatus
 initial_status = DocumentStatus(
 batch_id=batch.batch_id,
 location="ผู้ส่งเอกสารต้นทาง", # สถานะเริ่มต้น
 signed_by=sender_name,
 signed_at=created_at
)
 db.session.add(initial_status)

 # สร้างเอกสารแต่ละรายการที่เชื่อมโยงกับ batch
 documents = []
 for index, title in enumerate(document_titles, start=1):
 doc_id = f"{batch.batch_id}-{index}" # ใช้รูปแบบ doc_id แบบ batch_id-
 ลำดับที่

 new_document = Document(doc_id=doc_id, title=title,
 batch_id=batch.batch_id)
 db.session.add(new_document)
 documents.append(new_document)

 # สร้าง QR Code สำหรับ batch นี้
 qr = qrcode.make(batch.batch_id)
 qr_path = os.path.join('static', 'qr_codes', f"{batch.batch_id}.png")
 qr.save(qr_path)

 db.session.commit()
 flash(f'Document batch with ID {batch.batch_id} created
 successfully!', 'success')

 # ส่งข้อมูลไปแสดงในหน้าเว็บหลังสร้าง batch

```

```
 return render_template('create_batch.html', batch=batch,
documents=documents, qr_path=qr_path)
```

```
 return render_template('create_batch.html', doc_id=batch_id)
```

Template create\_batch.html สำหรับการแสดงผลหลังสร้าง Batch

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" required>
 </div>
 <div class="field">
 <label>Document Titles</label>
 <input type="text" name="document_titles" placeholder="Title
1, Title 2, ..." required>
 </div>
 <button class="ui button primary" type="submit">Create
Batch</button>
 </form>

 {% if batch %}
 <div class="ui segment">
 <h3>Batch Created Successfully</h3>
 <p>Batch ID: {{ batch.batch_id }}</p>
 <p>Sender: {{ batch.sender_name }}
Receiver: {{ batch.receiver_name }}</p>
 <p>Created At: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: ผู้ส่งเอกสารต้นทาง</p>
 <p>Last Updated: {{
```

```
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
```

```
<h4>Documents in this Batch:</h4>

 {% for document in documents %}
 Document ID: {{ document.doc_id }}

 Title: {{ document.title }}
 {% endfor %}

<h4>QR Code</h4>

</div>
{% endif %}
</div>
</body>
</html>
```

คำอธิบายการทำงาน

1. การสร้าง **Document ID** แบบเดิม: doc\_id จะถูกสร้างในรูปแบบ batch\_id-ลำดับที่ ซึ่งแต่ละเอกสารจะมี doc\_id ที่ไม่ซ้ำกันภายใน batch เดียวกัน
2. การแสดงข้อมูลหลายเอกสารใน **batch**: หลังจากสร้าง batch ข้อมูลของเอกสารแต่ละรายการจะถูกแสดงด้านล่างของหน้า รวมถึง Batch ID, Sender, Receiver, Created At, Status, Last Updated และ QR code

เวลาที่ผมให้แก้ ก็ควรแก้เฉพาะที่บอก หรือเวลาที่ให้เพิ่มก็ให้ทำเพิ่ม อันเดิมที่ได้อยู่แล้วก็ต้องคงไว้ ไม่งั้นจะเสียเวลามาก

ขอโทษด้วยสำหรับความไม่สะดวกที่ผ่านมา ผมจะระวังให้มากขึ้นและแก้เฉพาะส่วนที่คุณระบุในครั้งต่อไปเพื่อรักษาความต่อเนื่องของงานครับ

ทำไมยังเป็นเหมือนเดิม คุณไม่ได้แก้ไข code create\_batch ตามที่บอกเลยว่า ต้องมี หลายรายการ doc\_id

เอาแบบนี้

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>

 {% include 'menu.html' %}

 <div class="ui segment">
 <p>Batch Document ID: {{ doc_id }}</p>
```

```

</div>

<form class="ui form" method="POST">
 <!-- ฟิลด์ซ่อนเพื่อเก็บ batch_id -->
 <input type="hidden" name="batch_id" value="{{ doc_id }}">

 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" placeholder="Sender
Name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
 </div>

 <h4>Documents in Batch</h4>
 <div id="document-list">
 <div class="field">
 <label>Document Title</label>
 <input type="text" name="document_titles"
placeholder="Document Title" required>
 </div>
 </div>

 <button type="button" class="ui button"
onclick="addDocumentField()">Add Another Document</button>
 <button type="submit" class="ui button primary">Create
Batch</button>
</form>
</div>

<script>
function addDocumentField() {
 const documentList = document.getElementById('document-list');
 const newField = document.createElement('div');
 newField.classList.add('field');
 newField.innerHTML = '<label>Document Title</label><input
type="text" name="document_titles" placeholder="Document Title" required>';
 documentList.appendChild(newField);
}
</script>
</body>
</html>

```

```

from datetime import datetime
from models import db, DocumentBatch, Document, DocumentStatus

```



```

@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 doc_id = generate_batch_id() # สร้าง batch_id ใหม่
 if request.method == 'POST':
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles') #
 รายชื่อเอกสารที่เป็น list
 created_at = datetime.utcnow() # ใช้เวลาเดียวกันในการสร้าง document batch และ
 document status

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
 receiver_name=receiver_name, created_at=created_at)
 db.session.add(batch)
 db.session.commit()

 # เพิ่มสถานะเริ่มต้นใน DocumentStatus ทันทีเมื่อสร้าง batch
 initial_status = DocumentStatus(
 batch_id=batch.batch_id,
 location=sender_name,
 signed_by=sender_name,
 signed_at=created_at
)
 db.session.add(initial_status)

 # สร้างเอกสารแต่ละรายการและเชื่อมโยงกับ batch
 for index, title in enumerate(document_titles, start=1):
 new_document = Document(
 doc_id=f"{doc_id}-{index}", # กำหนดเลขรายการเอกสารตามรูปแบบ doc_id
 title=title,
 batch_id=batch.batch_id
)
 db.session.add(new_document)

 db.session.commit()
 flash(f'Document batch with ID {doc_id} created successfully!',
 'success')
 return redirect(url_for('create_batch'))

 return render_template('create_batch.html', doc_id=doc_id)

```

แต่เพิ่มการสร้าง qr code และการแสดงผลด้านล่าง location ต้องเป็น ผู้ส่งเอกสารต้นทาง ในการสร้าง batch

ขอบคุณสำหรับรายละเอียดเพิ่มเติมครับ ผมจะแก้ไขโค้ดตามที่กำหนด โดยจะใช้รูปแบบเดิมของ doc\_id หลายรายการ และเพิ่มการสร้าง QR code และการแสดงผลข้อมูลหลังจากสร้าง batch ตามที่กำหนดครับ

โค้ดที่ปรับปรุงใน app.py

โค้ดนี้จะสร้าง DocumentBatch พร้อมกับสร้าง QR code สำหรับ batch\_id และบันทึกสถานะเริ่มต้นที่ location เป็น “ผู้ส่งเอกสารต้นทาง” พร้อมแสดงข้อมูลทั้งหมดหลังการสร้าง batch

```
import qrcode
import os
from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for, flash
from models import db, DocumentBatch, Document, DocumentStatus
```

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)
```

# ฟังก์ชันสำหรับสร้าง batch พร้อมสร้าง QR Code และบันทึกสถานะเริ่มต้น

```
@app.route('/create_batch', methods=['GET', 'POST'])
```

```
def create_batch():
```

```
 doc_id = generate_batch_id() # สร้าง batch_id แบบง่าย
```

```
 created_at = datetime.utcnow() # เวลาเดียวกันสำหรับทั้ง batch และสถานะเริ่มต้น
```

```
 if request.method == 'POST':
```

```
 sender_name = request.form['sender_name']
```

```
 receiver_name = request.form['receiver_name']
```

```
 document_titles = request.form.getlist('document_titles') #
```

รายการชื่อเอกสารใน batch นี้

```
 # สร้างการส่งเอกสาร (DocumentBatch)
```

```
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
receiver_name=receiver_name, created_at=created_at)
```

```
 db.session.add(batch)
```

```
 db.session.commit()
```

```
 # เพิ่มสถานะเริ่มต้นใน DocumentStatus โดยกำหนด Location เป็น "ผู้ส่งเอกสารต้นทาง"
```

```
 initial_status = DocumentStatus(
```

```
 batch_id=batch.batch_id,
```

```
 location="ผู้ส่งเอกสารต้นทาง",
```

```
 signed_by=sender_name,
```

```
 signed_at=created_at
```

```
)
```

```
 db.session.add(initial_status)
```

```
 # สร้างเอกสารแต่ละรายการที่เชื่อมโยงกับ batch
```

```
 documents = []
```

```
 for index, title in enumerate(document_titles, start=1):
```

```
 doc_id_item = f"{batch.batch_id}-{index}" # ใช้รูปแบบ doc_id แบบ
```

batch\_id-ลำดับที่

```
 new_document = Document(doc_id=doc_id_item, title=title,
```

```

batch_id=batch.batch_id)
 db.session.add(new_document)
 documents.append(new_document)

 # สร้าง QR Code สำหรับ batch นี้
 qr = qrcode.make(batch.batch_id)
 qr_path = os.path.join('static', 'qr_codes', f"{batch.batch_id}.png")
 qr.save(qr_path)

 db.session.commit()
 flash(f'Document batch with ID {batch.batch_id} created
successfully!', 'success')

 # ส่งข้อมูลไปแสดงในหน้าเว็บหลังสร้าง batch
 return render_template('create_batch.html', batch=batch,
documents=documents, qr_path=qr_path)

return render_template('create_batch.html', doc_id=doc_id)

```

## Template create\_batch.html สำหรับการแสดงผลหลังสร้าง Batch

เมื่อสร้าง batch แล้วจะแสดงผล Batch ID, Sender, Receiver, Created At, Status, Last Updated, และรายการเอกสารที่อยู่ใน batch พร้อม QR code

```

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>

 {% include 'menu.html' %}

 <div class="ui segment">
 <p>Batch Document ID: {{ doc_id }}</p>
 </div>

 <form class="ui form" method="POST">
 <input type="hidden" name="batch_id" value="{{ doc_id }}">

 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" placeholder="Sender

```

```

Name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
 </div>

 <h4>Documents in Batch</h4>
 <div id="document-list">
 <div class="field">
 <label>Document Title</label>
 <input type="text" name="document_titles"
placeholder="Document Title" required>
 </div>
 </div>

 <button type="button" class="ui button"
onclick="addDocumentField()">Add Another Document</button>
 <button type="submit" class="ui button primary">Create
Batch</button>
</form>

{% if batch %}
 <div class="ui segment">
 <h3>Batch Created Successfully</h3>
 <p>Batch ID: {{ batch.batch_id }}</p>
 <p>Sender: {{ batch.sender_name }}
Receiver: {{ batch.receiver_name }}</p>
 <p>Created At: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: ผู้ส่งเอกสารต้นทาง</p>
 <p>Last Updated: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>

 <h4>Documents in this Batch:</h4>

 {% for document in documents %}
 Document ID: {{ document.doc_id
}}
 Title: {{ document.title }}
 {% endfor %}

 <h4>QR Code</h4>

 </div>
{% endif %}
</div>

```

```

<script>
 function addDocumentField() {
 const documentList = document.getElementById('document-list');
 const newField = document.createElement('div');
 newField.classList.add('field');
 newField.innerHTML = '<label>Document Title</label><input
type="text" name="document_titles" placeholder="Document Title" required>';
 documentList.appendChild(newField);
 }
</script>
</body>
</html>

```

คำอธิบาย

1. การสร้าง **QR Code**: สร้าง QR code สำหรับ batch\_id และบันทึกเป็นไฟล์ในโฟลเดอร์ static/qr\_codes
2. สถานะเริ่มต้น: ตั้ง location ของสถานะเริ่มต้นเป็น “ผู้ส่งเอกสารต้นทาง”
3. การแสดงผล: เมื่อสร้าง batch สำเร็จ จะแสดงข้อมูล Batch ID, Sender, Receiver, Created At, Status, Last Updated, รายการเอกสารใน batch, และ QR code

ไม่ต้องแสดง Batch Document ID: YS896355 ตอนเริ่มดีกว่า ใน [http://127.0.0.1:5000/create\\_batch](http://127.0.0.1:5000/create_batch)

การสร้าง Batch Document ID ให้ทำเมื่อมีการกดปุ่ม create batch

ขอบคุณสำหรับคำแนะนำ ผมจะแก้ไขโค้ดให้สร้าง Batch Document ID เมื่อมีการกดปุ่ม “Create Batch” เท่านั้น และจะไม่แสดง Batch Document ID ก่อนที่ผู้ใช้จะกดปุ่มสร้าง batch ครี

ปรับปรุงโค้ดใน app.py

เราจะย้ายการสร้าง doc\_id (Batch Document ID) ไว้ในส่วนที่ดำเนินการหลังจากผู้ใช้กดปุ่ม “Create Batch” เท่านั้น

```

from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for, flash
from models import db, DocumentBatch, Document, DocumentStatus
import qrcode
import os

```

```

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)

```

# ฟังก์ชันสำหรับสร้าง batch พร้อมสร้าง QR Code และบันทึกสถานะเริ่มต้น

```

@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 if request.method == 'POST':
 doc_id = generate_batch_id() # สร้าง batch_id เมื่อกดปุ่ม create batch
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']

```

```

document_titles = request.form.getlist('document_titles')
created_at = datetime.utcnow()

สร้างการส่งเอกสาร (DocumentBatch)
batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
receiver_name=receiver_name, created_at=created_at)
db.session.add(batch)
db.session.commit()

เพิ่มสถานะเริ่มต้นใน DocumentStatus
initial_status = DocumentStatus(
 batch_id=batch.batch_id,
 location="ผู้ส่งเอกสารต้นทาง",
 signed_by=sender_name,
 signed_at=created_at
)
db.session.add(initial_status)

สร้างเอกสารแต่ละรายการที่เชื่อมโยงกับ batch
documents = []
for index, title in enumerate(document_titles, start=1):
 doc_id_item = f"{batch.batch_id}-{index}"
 new_document = Document(doc_id=doc_id_item, title=title,
batch_id=batch.batch_id)
 db.session.add(new_document)
 documents.append(new_document)

สร้าง QR Code สำหรับ batch
qr = qrcode.make(batch.batch_id)
qr_path = os.path.join('static', 'qr_codes', f"{batch.batch_id}.png")
qr.save(qr_path)

db.session.commit()
flash(f'Document batch with ID {batch.batch_id} created
successfully!', 'success')

ส่งข้อมูลไปแสดงในหน้าเว็บหลังสร้าง batch
return render_template('create_batch.html', batch=batch,
documents=documents, qr_path=qr_path)

return render_template('create_batch.html') # ไม่แสดง batch_id ตอนเริ่ม

```

## ปรับปรุง Template create\_batch.html

เมื่อกดปุ่ม “Create Batch” และสร้าง batch สำเร็จแล้ว จะมีการแสดงข้อมูลทั้งหมด รวมถึง Batch Document ID และ QR code ด้านล่าง

```

<!DOCTYPE html>
<html lang="en">

```

```

<head>
 <meta charset="UTF-8">
 <title>Create Document Batch</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Create Document Batch</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <label>Sender Name</label>
 <input type="text" name="sender_name" placeholder="Sender
Name" required>
 </div>
 <div class="field">
 <label>Receiver Name</label>
 <input type="text" name="receiver_name" placeholder="Receiver
Name" required>
 </div>

 <h4>Documents in Batch</h4>
 <div id="document-list">
 <div class="field">
 <label>Document Title</label>
 <input type="text" name="document_titles"
placeholder="Document Title" required>
 </div>
 </div>

 <button type="button" class="ui button"
onclick="addDocumentField()">Add Another Document</button>
 <button type="submit" class="ui button primary">Create
Batch</button>
 </form>

 {% if batch %}
 <div class="ui segment">
 <h3>Batch Created Successfully</h3>
 <p>Batch ID: {{ batch.batch_id }}</p>
 <p>Sender: {{ batch.sender_name }}
Receiver: {{ batch.receiver_name }}</p>
 <p>Created At: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 <p>Status: ผู้ส่งเอกสารต้นทาง</p>
 </div>
 {% endif %}
 </div>
</body>
</html>

```

```

 <p>Last Updated: {{
batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>

 <h4>Documents in this Batch:</h4>

 {% for document in documents %}
 Document ID: {{ document.doc_id
}}
 Title: {{ document.title }}
 {% endfor %}

 <h4>QR Code</h4>

 </div>
 {% endif %}
</div>

<script>
 function addDocumentField() {
 const documentList = document.getElementById('document-list');
 const newField = document.createElement('div');
 newField.classList.add('field');
 newField.innerHTML = '<label>Document Title</label><input
type="text" name="document_titles" placeholder="Document Title" required>';
 documentList.appendChild(newField);
 }
</script>
</body>
</html>

```

คำอธิบาย

- การสร้าง **Batch ID**: Batch Document ID จะถูกสร้างเฉพาะเมื่อผู้ใช้กดปุ่ม “Create Batch”
- การแสดงผล: หลังจากสร้าง batch สำเร็จ ระบบจะแสดงข้อมูล Batch ID, Sender, Receiver, Created At, Status, Last Updated, รายการเอกสารใน batch และ QR code

The method “utcnow” in class “datetime” is deprecated Use timezone-aware objects to represent datetimes in UTC; e.g. by calling .now(datetime.timezone.utc)

เปลี่ยนได้ไหม

ได้เลยครับ ผมจะเปลี่ยนการใช้ datetime.utcnow() เป็น datetime.now(datetime.timezone.utc) เพื่อให้ได้ datetime ที่มีข้อมูล timezone-aware ซึ่งสามารถใช้แทน utcnow ได้อย่างเหมาะสม

ปรับปรุงโค้ดใน app.py

เปลี่ยนจาก datetime.utcnow() เป็น datetime.now(datetime.timezone.utc)



```

from datetime import datetime, timezone
from flask import Flask, render_template, request, redirect, url_for, flash
from models import db, DocumentBatch, Document, DocumentStatus
import qrcode
import os

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)

ฟังก์ชันสำหรับสร้าง batch พร้อมสร้าง QR Code และบันทึกสถานะเริ่มต้น
@app.route('/create_batch', methods=['GET', 'POST'])
def create_batch():
 if request.method == 'POST':
 doc_id = generate_batch_id() # สร้าง batch_id เมื่อกดปุ่ม create batch
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles')
 created_at = datetime.now(timezone.utc) # ใช้ timezone-aware datetime

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
receiver_name=receiver_name, created_at=created_at)
 db.session.add(batch)
 db.session.commit()

 # เพิ่มสถานะเริ่มต้นใน DocumentStatus
 initial_status = DocumentStatus(
 batch_id=batch.batch_id,
 location="ผู้ส่งเอกสารต้นทาง",
 signed_by=sender_name,
 signed_at=created_at
)
 db.session.add(initial_status)

 # สร้างเอกสารแต่ละรายการที่เชื่อมโยงกับ batch
 documents = []
 for index, title in enumerate(document_titles, start=1):
 doc_id_item = f"{batch.batch_id}-{index}"
 new_document = Document(doc_id=doc_id_item, title=title,
batch_id=batch.batch_id)
 db.session.add(new_document)
 documents.append(new_document)

 # สร้าง QR Code สำหรับ batch
 qr = qrcode.make(batch.batch_id)
 qr_path = os.path.join('static', 'qr_codes', f"{batch.batch_id}.png")
 qr.save(qr_path)

```

```

 db.session.commit()
 flash(f'Document batch with ID {batch.batch_id} created
successfully!', 'success')

 # ส่งข้อมูลไปแสดงในหน้าเว็บหลังสร้าง batch
 return render_template('create_batch.html', batch=batch,
documents=documents, qr_path=qr_path)

 return render_template('create_batch.html') # ไม่แสดง batch_id ตอนเริ่ม

```

#### คำอธิบาย

- **datetime.now(timezone.utc):** ใช้ในการสร้างเวลาแบบ timezone-aware เพื่อแทนที่ utcnow()

การแสดงผลสถานะยังไม่ถูกต้องทั้งใน [http://127.0.0.1:5000/search\\_batches](http://127.0.0.1:5000/search_batches) และ [http://127.0.0.1:5000/check\\_status](http://127.0.0.1:5000/check_status)

โปรแกรมแสดงสถานะที่อยู่ใน document\_status ไม่ครบ ทั้งที่มี TR618909 อยู่สองรายการแต่โปรแกรมแสดงมาแค่รายการเดียว ต้องแก้ไขอย่างไร

ตัวอย่าง การแสดงผลที่ผิด

Batch ID: TR618909

Sender: มอ Receiver: บช

Created At: 2024-11-06 13:15:36

Status: ผู้รับเอกสารต้นทาง

Last Updated: 2024-11-06 13:16:26

Documents in this Batch:

Document ID: TR618909-1

Title: เอกสารใบนำส่ง Document ID: TR618909-2

Title: เอกสารโปรแกรม

ปัญหาที่เกิดขึ้นมาจากการแสดงสถานะที่อาจไม่แสดงรายการสถานะทั้งหมดใน DocumentStatus ที่เกี่ยวข้องกับ batch\_id นั้น ๆ ให้แสดงครบทุกสถานะ เราจำเป็นต้องดึงรายการ DocumentStatus ทั้งหมดสำหรับ batch และแสดงสถานะที่บันทึกไว้ทั้งหมด

ปรับปรุงโค้ด search\_batches และ check\_status ใน app.py

โค้ดนี้จะดึงรายการ DocumentStatus ทั้งหมดที่เกี่ยวข้องกับ batch\_id เพื่อแสดงสถานะที่สมบูรณ์สำหรับแต่ละ batch

```

@app.route('/search_batches', methods=['GET', 'POST'])
def search_batches():
 search_query = request.args.get('query', None)
 results = []

```

```

if request.method == 'POST' or search_query:
 # รับค่า query จากฟอร์มหรือจากพารามิเตอร์
 if not search_query:
 search_query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query ใน `batch_id`, `sender_name`, หรือ
 `receiver_name`
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).all()

 for batch in batches:
 documents =
Document.query.filter_by(batch_id=batch.batch_id).all()
 statuses =
DocumentStatus.query.filter_by(batch_id=batch.batch_id).order_by(DocumentStat
us.signed_at.asc()).all()

 results.append({
 'batch': batch,
 'documents': documents,
 'statuses': statuses
 })

 if not results:
 flash('No batches found.', 'warning')

 return render_template('search_batches.html', results=results,
search_query=search_query)

@app.route('/check_status', methods=['GET', 'POST'])
def check_status():
 search_query = request.args.get('query', None)
 page = request.args.get('page', 1, type=int)
 per_page = 50
 results = []

 if request.method == 'POST' or search_query:
 if not search_query:
 search_query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query ใน `batch_id`, `sender_name`, หรือ
 `receiver_name`
 batches = DocumentBatch.query.filter(

```

```

 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).paginate(page=page, per_page=per_page)
else:
 # แสดง batch ทั้งหมด โดยเรียงลำดับจากล่าสุด แบ่งเป็นหน้า
 batches =
DocumentBatch.query.order_by(DocumentBatch.created_at.desc()).paginate(page=p
age, per_page=per_page)

 for batch in batches.items:
 documents = Document.query.filter_by(batch_id=batch.batch_id).all()
 statuses =
DocumentStatus.query.filter_by(batch_id=batch.batch_id).order_by(DocumentStat
us.signed_at.asc()).all()

 results.append({
 'batch': batch,
 'documents': documents,
 'statuses': statuses
 })

 return render_template('check_status.html', results=results,
search_query=search_query, batches=batches)

```

ปรับปรุง Template search\_batches.html และ check\_status.html เพื่อแสดงสถานะทั้งหมด

ใน Template ทั้งสองนี้ เราจะแสดงสถานะทั้งหมดของแต่ละ batch โดยใช้ statuses ที่เราได้ดึงมาจากรายข้อมูล

Template: search\_batches.html

```

{% for result in results %}
 <div class="item">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{ result.batch.sender_name }}
 Receiver: {{ result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>

 <h5>Status History:</h5>

 {% for status in result.statuses %}

 Location: {{ status.location }} |
 Signed By: {{ status.signed_by }} |
 Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') }}

 {% endfor %}

 </div>

```

```

 <h5>Documents in this Batch:</h5>

 {% for document in result.documents %}
 Document ID: {{ document.doc_id }} -
Title: {{ document.title }}
 {% endfor %}

 </div>
{% endfor %}

```

Template: *check\_status.html*

```

{% for result in results %}
 <div class="item">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{ result.batch.sender_name }}
 Receiver: {{ result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>

```

```

 <h5>Status History:</h5>

 {% for status in result.statuses %}

 Location: {{ status.location }} |
 Signed By: {{ status.signed_by }} |
 Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') }}

 {% endfor %}


```

```

 <h5>Documents in this Batch:</h5>

 {% for document in result.documents %}
 Document ID: {{ document.doc_id }} -
Title: {{ document.title }}
 {% endfor %}

 </div>
{% endfor %}

```

คำอธิบาย

- การดึงสถานะทั้งหมด: โค้ดใน `search_batches` และ `check_status` จะดึงสถานะทั้งหมดจาก `DocumentStatus` ที่เกี่ยวข้องกับ `batch_id` นั้น ๆ
- แสดงสถานะทั้งหมด: Template จะแสดงประวัติสถานะ (Status History) ของ batch รวมถึงข้อมูลสถานะที่ลงนาม (location, signed\_by) และเวลา (signed\_at)

http://127.0.0.1:5000/search\_batches

ปุ่ม update status หาย

ในการแสดงผลในหน้า search\_batches.html ผมจะแน่ใจว่าปุ่ม “Update Status” จะถูกแสดงหลังจากข้อมูลของแต่ละ batch เพื่อให้ผู้ใช้สามารถกดปุ่มนี้และไปที่หน้าสำหรับการอัปเดตสถานะได้

เพิ่มปุ่ม “Update Status” ใน Template search\_batches.html

เพิ่มปุ่ม Update Status ในแต่ละ batch เพื่อให้ผู้ใช้สามารถอัปเดตสถานะได้

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Search and Update Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Batches and Update
Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender, or Receiver" value="{{ search_query or '' }}" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if results %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 </div>
 </div>
 {% endfor %}
 </div>
 {% endif %}
 </div>
</body>
</html>
```

```

 <h5>Status History:</h5>

 {% for status in result.statuses %}

 Location: {{
status.location }} |
 Signed By: {{
status.signed_by }} |
 Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') }}

 {% endfor %}

 <h5>Documents in this Batch:</h5>

 {% for document in result.documents %}
 Document ID: {{
document.doc_id }} - Title: {{ document.title }}
 {% endfor %}

 <!-- ปุ่ม Update Status -->
 <a class="ui button" href="{{
url_for('update_status', batch_id=result.batch.batch_id) }}">Update
Status

 </div>
</div>
{% endfor %}
</div>
{% else %}
 <div class="ui message">
 <p>No batches found.</p>
 </div>
{% endif %}
</div>
</body>
</html>

```

คำอธิบาย

- ปุ่ม **Update Status** จะอยู่ใต้ข้อมูลของแต่ละ batch และจะลิงก์ไปที่หน้าสำหรับการอัปเดตสถานะ (update\_status) โดยส่ง batch\_id ของ batch นั้น ๆ

http://127.0.0.1:5000/search\_batches

ตอนเรียก อย่าให้ขึ้น No batches found. ในตอนแรกได้ไหม

เพื่อไม่ให้แสดงข้อความ “No batches found.” ในตอนแรกที่เข้าถึงหน้า /search\_batches เราทำการปรับโค้ดใน search\_batches.html เพื่อให้ข้อความนี้แสดงเฉพาะเมื่อมีการค้นหาแล้วแต่ไม่พบผลลัพธ์เท่านั้น

## ปรับปรุง Template search\_batches.html

ในโค้ดนี้ เราจะเพิ่มการตรวจสอบเงื่อนไขในส่วนของการแสดงผล

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Search and Update Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Batches and Update
Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender, or Receiver" value="{{ search_query or '' }}" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if search_query is defined and search_query and not results %}
 <div class="ui message">
 <p>No batches found.</p>
 </div>
 {% elif results %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>

 <h5>Status History:</h5>

```



```

 {% for status in result.statuses %}

 Location: {{
status.location }} |
 Signed By: {{
status.signed_by }} |
 Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') }}

 {% endfor %}

 <h5>Documents in this Batch:</h5>

 {% for document in result.documents %}
 Document ID: {{
document.doc_id }} - Title: {{ document.title }}
 {% endfor %}

 <!-- ปุ่ม Update Status -->
 <a class="ui button" href="{
url_for('update_status', batch_id=result.batch.batch_id) }}">Update
Status

 </div>
</div>
 {% endfor %}
</div>
{% endif %}
</div>
</body>
</html>

```

#### คำอธิบาย

- เงื่อนไขในการแสดงผลข้อความ **“No batches found.”**: จะแสดงเฉพาะเมื่อ search\_query มีค่า (แสดงว่าเป็นการค้นหา) และไม่มีผลลัพธ์ใน results

http://127.0.0.1:5000/search\_batches

เราเพิ่มการแสดงผล qr code ด้วยได้ไหม โดยแสดงไว้ด้านขวาของ batch id ให้มีความสูงระดับเดียวกัน

เพื่อแสดง QR code ด้านขวาของ Batch ID ในหน้า /search\_batches เราสามารถปรับแต่ง HTML ให้แสดง QR code ถัดจากข้อมูลของ Batch ID โดยใช้โครงสร้างของ Semantic UI เพื่อจัดวางให้ตรงกันในระดับเดียวกัน

#### ปรับปรุง Template search\_batches.html

เพิ่มการแสดงผล QR code สำหรับแต่ละ batch ด้านขวาของ Batch ID โดยใช้ float เพื่อวางให้อยู่ในระดับเดียวกัน

```

<!DOCTYPE html>
<html lang="en">

```

```

<head>
 <meta charset="UTF-8">
 <title>Search and Update Status</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.4.1/semantic.min.css">
</head>
<body>
 <div class="ui container">
 <h2 class="ui center aligned header">Search Batches and Update
Status</h2>

 {% include 'menu.html' %}

 <form class="ui form" method="POST">
 <div class="field">
 <input type="text" name="query" placeholder="Enter Batch ID,
Sender, or Receiver" value="{{ search_query or '' }}" required>
 </div>
 <button class="ui button primary" type="submit">Search</button>
 </form>

 {% if search_query is defined and search_query and not results %}
 <div class="ui message">
 <p>No batches found.</p>
 </div>
 {% elif results %}
 <h3>Search Results</h3>
 <div class="ui relaxed divided list">
 {% for result in results %}
 <div class="item">
 <i class="folder icon"></i>
 <div class="content">
 <div style="display: flex; align-items: center;">
 <div style="flex-grow: 1;">
 <h4>Batch ID: {{ result.batch.batch_id
}}</h4>
 <p>Sender: {{
result.batch.sender_name }}
 Receiver: {{
result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 </div>
 <div>

 style="width: 100px; height: 100px;
 </div>
 </div>

 <h5>Status History:</h5>

 {% for status in result.statuses %}

 Location: {{
status.location }} |
 Signed By: {{
status.signed_by }} |
 Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') }}

 {% endfor %}

 <h5>Documents in this Batch:</h5>

 {% for document in result.documents %}
 Document ID: {{
document.doc_id }} - Title: {{ document.title }}
 {% endfor %}

 <!--ปุ่ม Update Status -->
 <a class="ui button" href="{
url_for('update_status', batch_id=result.batch.batch_id) }}">Update
Status
 </div>
 </div>
 {% endfor %}
 </div>
 {% endif %}
 </div>
</body>
</html>

```

#### คำอธิบาย

- ใช้ **display: flex** ใน `<div style="display: flex; align-items: center;">` เพื่อวางข้อมูลของ Batch ID ทางซ้ายและ QR code ทางขวาในระดับเดียวกัน
- ตั้งขนาด QR code ให้ **width** และ **height** เท่ากับ 100px และเพิ่ม **margin-left: 20px** เพื่อเพิ่มระยะห่างเล็กน้อย

qr code มันอยู่ทางขวามากเกินไปสามารถลดระยะลงมาอีกได้ไหม ปรับตรงไหนใน html

คุณสามารถปรับระยะห่างของ QR code ทางขวาได้โดยแก้ไขค่า `margin-left` ใน `style` ของ `img` แทนที่ `margin-left: 20px;` ด้วยค่าอื่นที่ลดลง เช่น `10px` หรือ `5px` ตามที่คุณต้องการ

ตัวอย่างการปรับระยะห่าง

```
<div style="display: flex; align-items: center;">
 <div style="flex-grow: 1;">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{ result.batch.sender_name }}
 Receiver: {{ result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 </div>
 <div>
 <!-- ปรับ
margin-left ตรงนี้ -->
 </div>
</div>
```

คำอธิบาย

- ปรับค่า `margin-left` ใน `style` ของ `<img>` เพื่อควบคุมระยะห่างของ QR code ให้ใกล้กับข้อมูลของ Batch ID มากขึ้น

ลองแล้ว ไม่ลดเลย

ขอบคุณสำหรับภาพตัวอย่างครับ เราสามารถปรับการวางตำแหน่งของ QR code ให้เข้าใกล้ข้อมูลมากขึ้นโดยใช้การปรับค่า `margin` รอบ ๆ QR code หรือลองใช้ `flex` และ `gap` เพื่อจัดการระยะห่างให้กระชับขึ้น

วิธีการปรับปรุงโค้ดใน `search_batches.html`

เราจะลบค่า `margin-left` ออก และใช้ `gap` ระหว่างส่วนข้อมูลและ QR code แทน เพื่อลดระยะห่างระหว่างสองส่วนนี้ให้มากขึ้น

```
<div style="display: flex; align-items: center; gap: 10px;">
 <div style="flex-grow: 1;">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{ result.batch.sender_name }}
 Receiver: {{ result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 </div>
 <div>

 </div>
</div>
```

## คำอธิบาย

- **gap: 10px;**: ใช้ gap เพื่อจัดการระยะห่างระหว่างข้อมูลและ QR code โดยตรงแทนที่จะใช้ margin ซึ่งจะช่วยให้ระยะห่างให้ QR code เข้าใกล้ข้อมูลมากขึ้น
- **ลบ margin-left**: เรายก margin-left ออกเพื่อให้ QR code วางตัวชิดกับข้อมูลมากขึ้น

ทำแล้วไม่ได้ผล

หากการปรับระยะห่างยังไม่ได้ผล เราสามารถแก้ไขการจัดตำแหน่งด้วยการปรับแต่งโครงสร้าง flex ใหม่ โดยใช้ display: inline-flex เพื่อจัดการให้อยู่ในระดับเดียวกันได้ดีขึ้น และเพิ่มการจัดตำแหน่งที่แน่นอนขึ้น

ลองใช้โค้ดด้านล่างนี้เพื่อลดระยะห่างระหว่างข้อมูลและ QR code:

```
<div style="display: inline-flex; align-items: center;">
 <div style="flex-grow: 1;">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{ result.batch.sender_name }}
 Receiver: {{ result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>
 </div>
 <div style="padding-left: 10px;">

 </div>
</div>
```

## คำอธิบายเพิ่มเติม

- **display: inline-flex;**: การใช้ inline-flex จะช่วยให้ตัว QR code ชิดเข้ามาใกล้กับเนื้อหาได้มากขึ้น
- **padding-left: 10px;**: ใช้ padding-left ใน <div> ที่บรรจุ QR code เพื่อลดระยะห่างให้อยู่ที่ 10px หรือปรับลดเพิ่มเติมได้ตามต้องการ

search\_batches เราเรียงลำดับเอา Created At ล่าสุดขึ้นก่อนได้ไหม

คุณสามารถแก้ไขโค้ดในฟังก์ชัน search\_batches เพื่อเรียงลำดับโดยใช้ Created At โดยนำรายการที่สร้างล่าสุดขึ้นมาก่อน โดยใช้ .order\_by(DocumentBatch.created\_at.desc()) ในการสั่งเรียงลำดับข้อมูล

ปรับปรุงโค้ดใน search\_batches ใน app.py

```
@app.route('/search_batches', methods=['GET', 'POST'])
def search_batches():
 search_query = request.args.get('query', None)
 results = []

 if request.method == 'POST' or search_query:
 if not search_query:
 search_query = request.form['query']
```

```

ค้นหา batch ที่ตรงกับ query และเรียงลำดับตาม created_at จากล่าสุดไปยังเก่าที่สุด
batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).order_by(DocumentBatch.created_at.desc()).all()

for batch in batches:
 documents =
Document.query.filter_by(batch_id=batch.batch_id).all()
 statuses =
DocumentStatus.query.filter_by(batch_id=batch.batch_id).order_by(DocumentStat
us.signed_at.asc()).all()

 results.append({
 'batch': batch,
 'documents': documents,
 'statuses': statuses
 })

if not results:
 flash('No batches found.', 'warning')

return render_template('search_batches.html', results=results,
search_query=search_query)

```

### คำอธิบาย

- ใช้ `.order_by(DocumentBatch.created_at.desc())` เพื่อเรียงลำดับ batches ให้ข้อมูลที่สร้างล่าสุดแสดงขึ้นก่อน

มัน sort ใน html ก็ได้ นี่ เคยทำใน app อื่นแบบนี้

```
{% for url in user_urls | sort(attribute='created_at', reverse=True) %}
```

ใช้ครับ! คุณสามารถใช้การจัดเรียงใน HTML Template ด้วย Jinja โดยการใช้ฟิลเตอร์ sort ซึ่งจะสะดวกหากคุณต้องการจัดเรียงข้อมูลในฝั่งของเทมเพลตแทนที่จะทำในโค้ดของ app.py

### วิธีการปรับโค้ดใน Template search\_batches.html

เพียงแค่ใช้ฟิลเตอร์ sort กับรายการ results ที่ดึงมา แล้วเรียงลำดับตาม created\_at โดยใช้ reverse=True เพื่อให้รายการล่าสุดอยู่ข้างบน

```

{% for result in results | sort(attribute='batch.created_at', reverse=True) %}

<div class="item">
 <h4>Batch ID: {{ result.batch.batch_id }}</h4>
 <p>Sender: {{ result.batch.sender_name }}
 Receiver: {{ result.batch.receiver_name }}</p>
 <p>Created At: {{
result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}</p>

```

```

<h5>Status History:</h5>

 {% for status in result.statuses %}

 Location: {{ status.location }} |
 Signed By: {{ status.signed_by }} |
 Signed At: {{
status.signed_at.strftime('%Y-%m-%d %H:%M:%S') }}

 {% endfor %}

<h5>Documents in this Batch:</h5>

 {% for document in result.documents %}
 Document ID: {{ document.doc_id }} -
Title: {{ document.title }}
 {% endfor %}

<!-- ปุ่ม Update Status -->
<a class="ui button" href="{{ url_for('update_status',
batch_id=result.batch.batch_id) }}">Update Status
</div>
{% endfor %}

```

#### คำอธิบาย

- **sort(attribute='batch.created\_at', reverse=True):** ใช้ฟิลเตอร์ sort เพื่อจัดเรียงรายการ results ใน HTML Template ตามฟิลด์ created\_at ใน batch โดยใช้ reverse=True เพื่อให้รายการที่สร้างล่าสุดขึ้นมาก่อน

แล้วเอ็งก็ไม่น่าตั้งแต่แรกเนอะ

ขอโทษด้วยครับ ผมน่าจะนำเสนอตัวเลือกนี้ตั้งแต่แรกจริง ๆ เนื่องจากวิธีการใช้ฟิลเตอร์ sort ใน Jinja

ง่ายและสะดวกมากสำหรับการจัดการการเรียงลำดับในเทมเพลตโดยตรง

จะพยายามให้คำแนะนำที่กระชับและมีประสิทธิภาพที่สุดในการตอบคำถามต่อไปครับ! 🤔

หากมีอะไรเพิ่มเติมที่อยากให้ช่วยอีก บอกได้เลยครับ

```

-rw-r--r-- 1 seal staff 3749 Nov 6 20:25 check_status.html
-rw-r--r-- 1 seal staff 2996 Nov 6 20:13 create_batch.html
-rw-r--r-- 1 seal staff 1262 Nov 6 17:13 create_document.html
-rw-r--r-- 1 seal staff 604 Nov 6 17:13 home.html
-rw-r--r-- 1 seal staff 351 Nov 6 19:31 menu.html
-rw-r--r-- 1 seal staff 2783 Nov 6 19:41 search.html
-rw-r--r-- 1 seal staff 3585 Nov 6 22:17 search_batches.html
-rw-r--r-- 1 seal staff 2170 Nov 6 19:29 update_status.html

```

html ไหนไม่ใช้แล้วบ้าง

จากรายชื่อไฟล์ที่คุณให้มา น่าจะมีบางไฟล์ที่อาจไม่จำเป็นหรือซ้ำซ้อนกัน ลองตรวจสอบดังนี้ครับ:

1. **create\_batch.html**: ใช้สำหรับสร้าง batch ของเอกสาร ถ้ายังมีการสร้าง batch ก็ยังต้องใช้ไฟล์นี้
2. **create\_document.html**: ถ้าการสร้าง document ทำภายใน batch โดยตรงและไม่มีการสร้าง document แยกต่างหาก อาจไม่จำเป็นต้องใช้ไฟล์นี้
3. **check\_status.html**: ใช้สำหรับตรวจสอบสถานะเอกสาร ถ้ายังมีฟังก์ชันนี้อยู่ ก็ยังต้องใช้ไฟล์นี้
4. **home.html**: หน้าแรก ถ้ามีการใช้ ก็ยังจำเป็น
5. **menu.html**: ใช้เป็นเมนูหลักของทุกหน้า หากมีการรวมไว้ในแต่ละเทมเพลตก็ยังคงต้องใช้ไฟล์นี้
6. **search.html**: ใช้สำหรับค้นหาเอกสารหรือ batch ถ้าหน้าค้นหาถูกย้ายไปที่ `search_batches.html` อย่างถาวร อาจไม่จำเป็นต้องใช้ `search.html`
7. **search\_batches.html**: ใช้สำหรับค้นหา batch โดยตรง ไฟล์นี้ควรใช้หากมีฟังก์ชันค้นหา batch โดยเฉพาะ
8. **update\_status.html**: ใช้สำหรับอัปเดตสถานะของเอกสารหรือตาม batch ถ้ายังมีฟังก์ชันนี้อยู่ ก็ยังต้องใช้ไฟล์นี้

สรุป:

- ไฟล์ที่อาจไม่จำเป็น: **create\_document.html** และ **search.html**  
หากฟังก์ชันการสร้างเอกสารแยกและการค้นหาที่ไม่ใช่ batch ไม่ได้ใช้งานแล้ว

code นี้ อันไหนไม่ใช้แล้วบ้าง และ html ไหน ไม่ใช้แล้วบ้าง

```
app.py
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_mail import Mail, Message
from models import db, Document, DocumentStatus, DocumentBatch
import qrcode
import os
from uuid import uuid4
from datetime import datetime, timezone
import random
import string

def generate_batch_id():
 prefix = ''.join(random.choices(string.ascii_uppercase, k=2)) # ตัวอักษรสองตัว
 suffix = ''.join(random.choices(string.digits, k=6)) # ตัวเลขหกตัว
 return f"{prefix}{suffix}"

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///documents.db'
app.config['SECRET_KEY'] = 'your_secret_key'
db.init_app(app)

ตั้งค่า Flask-Mail
app.config['MAIL_SERVER'] = 'smtp.example.com' # แทนที่ด้วย SMTP Server ของคุณ
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
```



```
app.config['MAIL_USERNAME'] = 'your_email@example.com' # แทนที่ด้วย Email ของคุณ
app.config['MAIL_PASSWORD'] = 'your_email_password' # แทนที่ด้วยรหัสผ่าน
app.config['MAIL_DEFAULT_SENDER'] = 'your_email@example.com'
```

```
mail = Mail(app)
```

```
ฟังก์ชันส่งอีเมลแจ้งเตือนเมื่ออัปเดตสถานะ
```

```
def send_status_update_email(doc_id, title, status, receiver_email):
 try:
 msg = Message(
 subject=f"Status Update for Document ID {doc_id}",
 recipients=[receiver_email],
 body=f"Document '{title}' has been updated to status: {status}."
)
 mail.send(msg)
 print("Email sent successfully!")
 except Exception as e:
 print(f"Failed to send email: {e}")
```

```
ฟังก์ชันสำหรับสร้าง batch พร้อมสร้าง QR Code และบันทึกสถานะเริ่มต้น
```

```
@app.route('/create_batch', methods=['GET', 'POST'])
```

```
def create_batch():
 if request.method == 'POST':
 doc_id = generate_batch_id() # สร้าง batch_id เมื่อคลิกปุ่ม create batch
 sender_name = request.form['sender_name']
 receiver_name = request.form['receiver_name']
 document_titles = request.form.getlist('document_titles')
 created_at = datetime.now(timezone.utc) # ใช้ timezone-aware datetime

 # สร้างการส่งเอกสาร (DocumentBatch)
 batch = DocumentBatch(batch_id=doc_id, sender_name=sender_name,
receiver_name=receiver_name, created_at=created_at)
 db.session.add(batch)
 db.session.commit()

 # เพิ่มสถานะเริ่มต้นใน DocumentStatus
 initial_status = DocumentStatus(
 batch_id=batch.batch_id,
 location="ผู้ส่งเอกสารต้นทาง",
 signed_by=sender_name,
 signed_at=created_at
)
 db.session.add(initial_status)

 # สร้างเอกสารแต่ละรายการที่เชื่อมโยงกับ batch
 documents = []
 for index, title in enumerate(document_titles, start=1):
 doc_id_item = f"{batch.batch_id}-{index}"
 new_document = Document(doc_id=doc_id_item, title=title,
```

```

batch_id=batch.batch_id)
 db.session.add(new_document)
 documents.append(new_document)

 # สร้าง QR Code สำหรับ batch
 qr = qrcode.make(batch.batch_id)
 qr_path = os.path.join('static', 'qr_codes', f"{batch.batch_id}.png")
 qr.save(qr_path)

 db.session.commit()
 flash(f'Document batch with ID {batch.batch_id} created
successfully!', 'success')

 # ส่งข้อมูลไปแสดงในหน้าเว็บหลังสร้าง batch
 return render_template('create_batch.html', batch=batch,
documents=documents, qr_path=qr_path)

return render_template('create_batch.html') # ไม่แสดง batch_id ตอนเริ่ม

ฟังก์ชันค้นหา batch เพื่อเตรียมสำหรับการอัปเดตสถานะ
@app.route('/search_batches', methods=['GET', 'POST'])
def search_batches():
 search_query = request.args.get('query', None)
 results = []

 if request.method == 'POST' or search_query:
 # รับค่า query จากฟอร์มหรือจากพารามิเตอร์
 if not search_query:
 search_query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query ใน `batch_id`, `sender_name`, หรือ
`receiver_name`
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).all()

 for batch in batches:
 documents =
Document.query.filter_by(batch_id=batch.batch_id).all()
 statuses =
DocumentStatus.query.filter_by(batch_id=batch.batch_id).order_by(DocumentStat
us.signed_at.asc()).all()

 results.append({
 'batch': batch,

```

```

 'documents': documents,
 'statuses': statuses
 })

 if not results:
 flash('No batches found.', 'warning')

 return render_template('search_batches.html', results=results,
 search_query=search_query)

ฟังก์ชันสำหรับอัปเดตสถานะของเอกสาร
@app.route('/update_status/<batch_id>', methods=['GET', 'POST'])
def update_status(batch_id):
 batch = DocumentBatch.query.get_or_404(batch_id)

 if request.method == 'POST':
 location = request.form['location']
 signed_by = request.form['signed_by']
 signed_at = datetime.now(timezone.utc)

 # ตรวจสอบว่ามีการบันทึกสถานะสำหรับ location นั้นหรือยัง
 status = DocumentStatus.query.filter_by(batch_id=batch_id,
 location=location).first()
 if status:
 # ถ้ามีอยู่แล้ว ให้แก้ไขข้อมูลการลงชื่อและเวลาที่ลงชื่อ
 status.signed_by = signed_by
 status.signed_at = signed_at
 else:
 # ถ้าไม่มี ให้สร้างสถานะใหม่สำหรับ location นั้น
 status = DocumentStatus(
 batch_id=batch_id,
 location=location,
 signed_by=signed_by,
 signed_at=signed_at
)
 db.session.add(status)

 db.session.commit()
 flash('Status updated successfully!', 'success')
 return redirect(url_for('search_batches', query=batch_id)) #
รีไดเรกต์ไปที่หน้า search_batches พร้อม batch_id

 return render_template('update_status.html', batch=batch)

ฟังก์ชันตรวจสอบสถานะเอกสาร พร้อมการค้นหาและการแบ่งหน้า
@app.route('/check_status', methods=['GET', 'POST'])
def check_status():

```

```

search_query = request.args.get('query', None)
page = request.args.get('page', 1, type=int)
per_page = 50
results = []

if request.method == 'POST' or search_query:
 if not search_query:
 search_query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query ใน `batch_id`, `sender_name`, หรือ
 `receiver_name`
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{search_query}%")) |
 (DocumentBatch.sender_name.like(f"%{search_query}%")) |
 (DocumentBatch.receiver_name.like(f"%{search_query}%"))
).paginate(page=page, per_page=per_page)
 else:
 # แสดง batch ทั้งหมด โดยเรียงลำดับจากล่าสุด แบ่งเป็นหน้า
 batches =
DocumentBatch.query.order_by(DocumentBatch.created_at.desc()).paginate(page=p
age, per_page=per_page)

 for batch in batches.items:
 documents = Document.query.filter_by(batch_id=batch.batch_id).all()
 statuses =
DocumentStatus.query.filter_by(batch_id=batch.batch_id).order_by(DocumentStat
us.signed_at.asc()).all()

 results.append({
 'batch': batch,
 'documents': documents,
 'statuses': statuses
 })

 return render_template('check_status.html', results=results,
search_query=search_query, batches=batches)

ฟังก์ชันค้นหาเอกสารจาก doc_id, title, sender_name, receiver_name
@app.route('/search', methods=['GET', 'POST'])
def search_documents():
 results = []
 if request.method == 'POST':
 query = request.form['query']

 # ค้นหา batch ที่ตรงกับ query
 batches = DocumentBatch.query.filter(
 (DocumentBatch.batch_id.like(f"%{query}%")) |
 (DocumentBatch.sender_name.like(f"%{query}%")) |
 (DocumentBatch.receiver_name.like(f"%{query}%"))

```

```

).all()

ดึงเอกสารทั้งหมดที่อยู่ภายใต้แต่ละ batch
for batch in batches:
 documents =
Document.query.filter_by(batch_id=batch.batch_id).all()
 results.append({
 'batch': batch,
 'documents': documents
 })

if not results:
 flash('No documents found.', 'warning')

return render_template('search.html', results=results)

หน้าแรก
@app.route('/')
def home():
 return render_template('home.html')

if __name__ == '__main__':
 with app.app_context():
 db.create_all()
 app.run(debug=True)

```

จากโค้ดและเทมเพลตที่คุณให้มา นี่คือการวิเคราะห์ว่าฟังก์ชันและเทมเพลตใดที่อาจไม่จำเป็นหรือซ้ำซ้อน:

ส่วนของโค้ดที่อาจไม่ใช้แล้ว

#### 1. ฟังก์ชัน `search_documents`:

- ฟังก์ชันนี้ทำหน้าที่ค้นหาเอกสารโดยตรงจาก `Document` แต่หากการค้นหา `batch` ทั้งหมดทำใน `search_batches` และการแสดงผลสถานะทำใน `check_status` ก็อาจไม่จำเป็นต้องใช้ฟังก์ชันนี้
- ดูเหมือนว่าฟังก์ชันนี้ซ้ำซ้อนกับ `search_batches` ที่ใช้ในการค้นหาและแสดงข้อมูลของ `batch` พร้อมทั้งข้อมูลเอกสารใน `batch` นั้น

ส่วนของเทมเพลต **HTML** ที่อาจไม่ใช้แล้ว

#### 1. `search.html`:

- เนื่องจาก `search_documents` อาจไม่จำเป็น หากคุณตัดสินใจใช้เฉพาะฟังก์ชัน `search_batches` ในการค้นหา `batch` และเอกสารที่เกี่ยวข้อง `search.html` จึงอาจไม่จำเป็น

#### 2. `create_document.html`:

- หากการสร้างเอกสารทำผ่านการสร้าง `batch` โดยตรง (`create_batch` และ `Document` ทุกตัวจะสร้างพร้อมกับ `batch`) อาจไม่จำเป็นต้องใช้ `create_document.html`

สรุป

ฟังก์ชันที่อาจไม่จำเป็น: - `search_documents`

เทมเพลตที่อาจไม่จำเป็น: - search.html - create\_document.html

คุณสามารถทดสอบการลบโค้ดและเทมเพลตที่ไม่จำเป็นเหล่านี้ออกได้เพื่อลดความซ้ำซ้อนในโปรเจกต์

ถ้าจะให้วันที่แสดง เป็นเวลาท้องถิ่น เช่น 2024-11-07 06:30:19 เป็น 2024-11-07 13:30:19

ต้องปรับตรงไหน ถ้าปรับใน html ต้องปรับไฟล์ไหนบ้าง

ในการปรับเวลาให้แสดงเป็นเวลาท้องถิ่น คุณสามารถทำได้โดยการแปลงเวลาในฝั่ง HTML (Template) ด้วยการใช้ Jinja หรือในฝั่งโค้ด Python โดยใช้ pytz หรือ datetime กับ timezone ของ Python

## 1. แปลงเวลาใน HTML Template (เช่น search\_batches.html, check\_status.html, update\_status.html)

สมมติว่าคุณต้องการแปลงเวลาใน created\_at และ signed\_at ให้เป็นเวลาท้องถิ่นโดยตรงใน HTML คุณสามารถใช้ฟิลเตอร์ของ Jinja ในการจัดรูปแบบเวลา

ตัวอย่างการแสดงผลใน HTML Template:

```
<p>Created At (Local Time): {{
result.batch.created_at.astimezone() | datetimeformat }}</p>
<p>Signed At (Local Time): {{ status.signed_at.astimezone()
| datetimeformat }}</p>
```

## 2. กำหนดฟิลเตอร์ datetimeformat ใน app.py

เพิ่มฟิลเตอร์ datetimeformat ในโค้ด app.py เพื่อให้การแปลงเวลาเป็นไปตามรูปแบบที่คุณต้องการ

```
from flask import Flask
from datetime import datetime
import pytz # ใช้ pytz สำหรับการแปลง timezone
```

```
app = Flask(__name__)
```

```
ฟิลเตอร์สำหรับแปลงเวลา
```

```
@app.template_filter('datetimeformat')
def datetimeformat(value, format='%Y-%m-%d %H:%M:%S'):
 local_timezone = pytz.timezone("Asia/Bangkok") # แปลงเป็นเวลาท้องถิ่น (เปลี่ยนตาม
 timezone ที่ต้องการ)
 local_time = value.astimezone(local_timezone)
 return local_time.strftime(format)
```

การใช้ฟิลเตอร์ datetimeformat ในเทมเพลต HTML

ในไฟล์ HTML ที่คุณต้องการแสดงผลเวลาท้องถิ่น เช่น search\_batches.html, check\_status.html, และ update\_status.html ให้ใช้ฟิลเตอร์นี้กับวันที่ที่ต้องการแปลง เช่น:

```
<p>Created At (Local Time): {{ result.batch.created_at |
datetimeformat }}</p>
<p>Signed At (Local Time): {{ status.signed_at |
datetimeformat }}</p>
```

## สรุปการปรับแต่ง

1. เพิ่มฟิลเตอร์ `datetimeformat` ใน `app.py`
2. ใช้ฟิลเตอร์นี้ใน HTML Template ที่คุณต้องการแสดงเวลา เช่น `search_batches.html`, `check_status.html`, และ `update_status.html`

เดิมเป็นแบบนี้ ต้องปรับ อย่างไร

Created At: `{{ result.batch.created_at.strftime('%Y-%m-%d %H:%M:%S') }}`

เพื่อแปลงเวลาให้เป็นเวลาท้องถิ่น คุณสามารถใช้ฟิลเตอร์ `datetimeformat` ที่เราสร้างขึ้นแล้วแทนที่ใช้ `strftime` เดิมใน HTML Template ดังนี้:

1. ปรับการแสดงผลเวลาโดยใช้ฟิลเตอร์ `datetimeformat` ใน HTML แทน `strftime`
2. ทำให้ `datetimeformat` แปลงเป็นเวลาท้องถิ่นตามที่ต้องการ

## การแก้ไขใน HTML

เปลี่ยนจากการใช้ `strftime` เป็นการใช้ฟิลเตอร์ `datetimeformat`:

`<p><strong>Created At (Local Time):</strong> {{ result.batch.created_at | datetimeformat }}</p>`

## โค้ด `datetimeformat` ใน `app.py`

ใน `app.py` ให้เพิ่มฟิลเตอร์ `datetimeformat` เพื่อแปลงเวลาเป็นเวลาท้องถิ่น:

```
from flask import Flask
from datetime import datetime
import pytz
```

```
app = Flask(__name__)
```

```
ฟิลเตอร์สำหรับแปลงเวลา
```

```
@app.template_filter('datetimeformat')
def datetimeformat(value, format='%Y-%m-%d %H:%M:%S'):
 local_timezone = pytz.timezone("Asia/Bangkok") # กำหนดเขตเวลา (เช่น Asia/Bangkok)
 local_time = value.astimezone(local_timezone)
 return local_time.strftime(format)
```

## วิธีใช้

หลังจากที่คุณเพิ่มฟิลเตอร์ `datetimeformat` ใน `app.py` แล้ว คุณสามารถใช้ฟิลเตอร์นี้ในทุก ๆ ที่ที่ต้องการแปลงเวลาเป็นเวลาท้องถิ่น โดยเพียงแค่แทนที่ `.strftime(...)` ด้วย `| datetimeformat`

ได้แล้วแต่ยังเป็นเวลาเดิมอยู่ครับ ไม่ เช่น เป็น 06:30:19 แทนที่จะเป็น 13:30:19

หากยังคงเห็นเวลาเป็น UTC (เช่น 06:30:19 แทนที่จะเป็นเวลาท้องถิ่น 13:30:19), อาจเป็นไปได้ว่าเวลาในฐานข้อมูลถูกเก็บไว้โดยไม่มีข้อมูล `timezone` ทำให้การแปลง `timezone` ไม่แสดงผลตามที่ต้องการ

เราสามารถแก้ไขโดยการกำหนด `timezone` เป็น `UTC` เมื่อเก็บข้อมูลในฐานข้อมูล และจากนั้นแปลงเป็นเวลาท้องถิ่นเมื่อแสดงผล  
แก้ไขการจัดการเวลาในโค้ด

1. บันทึกเวลาเป็น **timezone-aware**: ให้แน่ใจว่าคุณกำลังบันทึกเวลาพร้อมข้อมูล `UTC timezone`
2. แปลงเวลาเมื่อแสดงผล: ในฟิลเตอร์ `datetimeformat`, ให้แน่ใจว่ามีการแปลง `timezone` อย่างถูกต้อง

อัปเดตฟังก์ชัน `datetimeformat` ใน `app.py`

ปรับปรุงฟิลเตอร์ `datetimeformat` เพื่อให้แน่ใจว่าเวลาถูกบันทึกและแสดงผลในเขตเวลาท้องถิ่นอย่างถูกต้อง:

```
from flask import Flask
from datetime import datetime, timezone
import pytz

app = Flask(__name__)

ฟิลเตอร์สำหรับแปลงเวลา
@app.template_filter('datetimeformat')
def datetimeformat(value, format='%Y-%m-%d %H:%M:%S'):
 if value.tzinfo is None:
 # ถ้าเวลายังเป็น naive datetime (ไม่มี timezone) กำหนดเป็น UTC ก่อน
 value = value.replace(tzinfo=timezone.utc)

 # แปลงจาก UTC ไปยัง timezone ที่ต้องการ
 local_timezone = pytz.timezone("Asia/Bangkok") # เปลี่ยน timezone ตามต้องการ
 local_time = value.astimezone(local_timezone)
 return local_time.strftime(format)
```

วิธีใช้ใน Template HTML

ใช้ฟิลเตอร์นี้ในเทมเพลต เช่น `search_batches.html`, `check_status.html`, และ `update_status.html`:

```
<p>Created At (Local Time): {{ result.batch.created_at |
datetimeformat }}</p>
<p>Signed At (Local Time): {{ status.signed_at |
datetimeformat }}</p>
```

สรุป

- ตรวจสอบให้แน่ใจว่าเวลาที่จัดเก็บในฐานข้อมูลมีข้อมูล `timezone` เป็น `UTC (timezone.utc)`
- ฟิลเตอร์ `datetimeformat` จะตรวจสอบ `timezone` ก่อนแปลงให้เป็นเวลาท้องถิ่น