

ITM PROJECT

Project ITM : Reverse Image Search for Online Shopping

เอกสาร Technical documentation และ Project team lesson learn

นำเสนอ

อาจารย์ ดร.บุญรัตน์ เพลิมรอด

ผู้จัดทำ

นายธีรภัทร	อักษรนันท์	6320500573
นายอาทิตย์	มหากลั่น	6320501600
นายชนกานต์	ศรีสุรติพร	6320502371
นายปวิช	สังข์วารีย์	6320502452

เอกสาร Technical documentation และ Project team lesson learn นี้เป็นส่วน
หนึ่งเป็นรายวิชา

การบริหารจัดการเทคโนโลยีสารสนเทศ

(Management of Information Technology) 02204372-60

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตกำแพงแสน

คณะวิศวกรรมศาสตร์ กำแพงแสน ภาควิชาวิศวกรรมคอมพิวเตอร์

สารบัญ

Technical documentation	1
การ setup Environment สำหรับโปรเจค	2
การทำงานของ Code	3
Model	17
Project team lesson learn	22

Technical documentation

เทคโนโลยี หรือ ซอฟต์แวร์	ฟังก์ชัน	รายละเอียด
VSCODE	IDE	โปรแกรมประยุกต์ซอฟต์แวร์ที่ช่วยให้โปรแกรมเมอร์พัฒนาซอฟต์แวร์ได้อย่างมีประสิทธิภาพ ในที่นี้ใช้ในการทำทุกอย่าง
XAMPP	Frameworks & Library	XAMPP Version: 7.4.29
PYTHON	Languages	ใช้สำหรับการ Coding ทั้งทางด้าน AI แล้วก็ Backend
flask	Frameworks & Library	Python Framework ช่วยในการทำงานเป็นรูปแบบมากยิ่งขึ้น ตามแนวทาง MVC
html5	Languages	ภาษาที่ใช้ในการทำโครงหน้าเว็บไซต์
css3	Languages	ภาษาที่ใช้ในการตกแต่งหน้าเว็บให้มีความสวยงามมากยิ่งขึ้น
jupyter	Frameworks & Library	เครื่องมือที่ใช้เขียนcodeในการที่เราจะสร้าง machine learning model
conda	Frameworks & Library	Conda เป็นที่ setting Environment library ที่จำเป็นใน Python

การ setup Environment สำหรับโปรเจค

1. การลง library ที่ใช้ในการทำโปรเจค
 - a. pip install flask
 - b. pip install PyMySQL
 - c. pip install tensorflow
 - d. pip install --upgrade tensorflow-hub
 - e. pip install pandas
 - f. pip install Ponytile
 - g. หากมีปัญหา
 - i. Downgrade the protobuf package to 3.20.x or lower.
 - ii. Set `PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python`
(but this will use pure-Python parsing and will be much slower).
ใช้งาน `pip install protobuf==3.20.*`
2. ทำการ Clone ตัว Repository
3. การนำฐานข้อมูลลงในตัว XAMPP
 - a. ทำการสร้างฐานข้อมูลใน <http://localhost/phpmyadmin/> ตั้งชื่อ database ว่า itm_database ใช้ utf8_general_ci
 - b. import : file "itm_databaseVfinal.sql" ใน folder: Database
4. ทำการเปลี่ยน path ตามความต้องการใน file "app.py" path ทั้งหมดต้องอยู่ใน folder: *Flask_Run_Application*
5. ทำการ run file "app.py" ใน folder: Flask_Run_Application

การทำงานของ Code

การทำงานของ Code จะถูกแบ่งออกเป็น 3 การทำงานก็คือ

1. การโชว์หน้าและฟังก์ชันการทำงานก่อนการเข้าสู่ระบบ
2. การทำงานตาม Action ต่างๆของการกด Submit ตาม Form ใน HTML
3. การโชว์หน้าและฟังก์ชันการทำงานหลังการเข้าสู่ระบบ

การโชว์หน้าและฟังก์ชันการทำงานก่อนการเข้าสู่ระบบ

```
1 @app.route("/")
2 def indexWithoutUser():
3     return render_template('index.html')
```

- การทำงานของ route("/") คือ การเปิดหน้า index.html หรือหน้าแรกของตัวเว็บไซต์ โดยไม่มีการรับหรือส่งค่าตัวแปรใดๆไปเลย

```
1 @app.route("/shopping")
2 def shoppingWithoutUser():
3     conn = openConnection()
4     cur = conn.cursor()
5     sql = "SELECT * FROM `product` NATURAL JOIN product_category"
6     cur.execute(sql)
7     result_product = cur.fetchall()
8     conn.close()
9     return render_template('pageWithoutUser/shop.html', product=result_product)
```

- การทำงานของ route("/shopping") คือ การเปิดหน้าของ shop.html หรือหน้าแสดงสินค้าทั้งหมดที่มีอยู่ในฐานข้อมูล ไฟล์อยู่ที่ Folder : pageWithoutUser และมีการ Query นำสินค้า (Product) ที่มีประเภทสินค้า (Product Category) ทั้งหมดออกมาด้วย แล้วทำการส่งข้อมูลที่ได้จากฐานข้อมูลไปที่หน้า shop.html

```
1 @app.route("/search")
2 def searchWithoutUser():
3     return render_template('pageWithoutUser/search.html')
```

- การทำงานของ route("/search") คือ การเปิดหน้าของ search.html หรือหน้าแสดงฟังก์ชันการค้นหาไฟล์อยู่ที่ Folder : pageWithoutUser โดยไม่มีการรับหรือส่งค่าตัวแปรใดๆไปเลย

```
1 @app.route("/sign_in")
2 def signInPage():
3     return render_template('pageWithoutUser/signin.html')
```

- การทำงานของ route("/sign_in") คือ การเปิดหน้าของ signin.html หรือหน้าแสดงฟังก์ชันการเข้าสู่ระบบ ไฟล์อยู่ที่ Folder : pageWithoutUser โดยไม่มีการรับหรือส่งค่าตัวแปรใดๆไปเลย

```
1 @app.route("/sign_up")
2 def signUpPage():
3     return render_template('pageWithoutUser/signup.html')
```

- การทำงานของ route("/sign_up") คือ การเปิดหน้าของ signup.html หรือหน้าแสดงฟังก์ชันการสมัคร User หรือ Admin เข้าไปที่ฐานข้อมูล ไฟล์อยู่ที่ Folder : pageWithoutUser โดยไม่มีการรับหรือส่งค่าตัวแปรใดๆไปเลย

```
1 @app.route("/aboutUs")
2 def aboutUsPage():
3     return render_template('pageWithoutUser/aboutUs.html')
```

- การทำงานของ route("/aboutUs") คือ การเปิดหน้าของ aboutUs.html หรือหน้าแสดงผู้ที่เกี่ยวข้องทำการทำโปรเจค และคำอธิบายที่เกี่ยวข้อง ไฟล์อยู่ที่ Folder : pageWithoutUser โดยไม่มีการรับหรือส่งค่าตัวแปรใดๆไปเลย

การทำงานตาม Action ต่างๆของการกด Submit ตาม Form ใน HTML

```
1
2 @app.route('/logout')
3 def logout():
4     session.pop('user', None)
5     return redirect(url_for('signInPage'))
```

- การทำงานของ route("/logout") คือ การทำงานในการ ออกจากระบบของ User หรือ Admin โดยการทำการนำ user ที่ login อยู่ นั้นออกจาก session ที่มีชื่อว่า 'user'
- หลังจากการออกจากระบบเสร็จ จะกลับไปยังหน้า signin.html หรือหน้าแสดงฟังก์ชันการเข้าสู่ระบบ ไฟล์อยู่ที่ Folder : pageWithoutUser ตาม url_for('signInPage') [signInPage เป็นชื่อ method ของ route("/sign_in")]

```

1 @app.route("/sign_in/actions", methods=['POST'])
2 def signInLogin():
3     if request.method == 'POST':
4         email = request.form['email']
5         password = request.form['password']
6         conn = openConnection()
7         cur = conn.cursor()
8         # hashed_password = generate_password_hash(password, method='sha256')
9         # print(hashed_password)
10        # sql = "SELECT * FROM `user` NATURAL JOIN `user_type` WHERE user.user_email = %s AND user.user_password = %s"
11        sql = "SELECT * FROM `user` NATURAL JOIN `user_type` WHERE user.user_email = %s"
12        cur.execute(sql, (email))
13        result = cur.fetchone()
14        # if result = none return to sign in page
15        if result == None :
16            flash("Don't have this User in database", "danger")
17            return redirect(url_for('signInPage'))
18        #
19        userType = result[6]
20        conn.close()
21        if result and check_password_hash(result[5], password):
22            # if result :
23            session['user'] = result[1]
24            if userType == "ADMIN":
25                return redirect(url_for('adminPageUser', userType_name=result[6], user_id=result[1]))
26            elif (userType == "USER"):
27                return redirect(url_for('userPageShopping', userType_name=result[6], user_id=result[1]))
28        else:
29            flash("Please check your password", "warning")
30            return redirect(url_for('signInPage'))
31    else:
32        return redirect(url_for('signInPage'))

```

- การทำงานของ route("/sign_in/actions") คือ การทำงานของการเข้าสู่ระบบ ผ่านทางหน้าของ signin.html หรือหน้าแสดงฟังก์ชันการเข้าสู่ระบบ
- การทำงานจะรับเป็น method == "POST" โดยการรับ การกรอกต่อไปนี้
 - email = มาจากช่อง input HTML ผ่านทาง form['email']
 - password = มาจากช่อง input HTML ผ่านทาง form['password']
- เมื่อได้ค่ามาแล้ว ทำการเชื่อมต่อกับฐานข้อมูล เพื่อตรวจหา email ว่ามีอยู่ในระบบหรือไม่
 - หากไม่เจอ : ก็จะกลับไปหน้า signin.html ตาม url_for('signInPage') [signInPage เป็นชื่อ method ของ route("/sign_in")] แล้วส่ง message flashing เป็น ("Don't have this User in database")
 - หากเจอ : ก็ตรวจสอบตัวรหัสผ่านต่อในขั้นต่อไป
 - หากไม่ถูก : กลับไปหน้า signin.html แล้วส่ง message flashing เป็น ("Please check your password")
 - หากถูก : ก็จะถือว่าเป็นประเภท User แบบใด เป็น Admin หรือ User
- เป็น Admin ทำการเปิดหน้า admin ที่แสดง User ทั้งหมด ตาม url_for('adminPageUser') [adminPageUser เป็นชื่อ method ของ route("/<string:userType_name>/<string:user_id>/users")] พร้อมส่งค่าของทางด้าน userType_name กับ user_id ไปด้วย
- เป็น User ทำการเปิดหน้า user หน้าแรกของร้านค้า ที่แสดงสินค้า ตาม url_for('userPageShopping') [userPageShopping เป็นชื่อ method ของ route("/<string:userType_name>/<string:user_id>/shopping")] พร้อมส่งค่าของทางด้าน userType_name กับ user_id ไปด้วย


```

1 @app.route("/sign_up/actions", methods=['POST'])
2 def signUpLogin():
3     firstname = request.form['firstname']
4     lastname = request.form['lastname']
5     email = request.form['email']
6     password = request.form['password']
7     user = 2
8     hashed_password = generate_password_hash(password, method='sha256')
9     conn = openConnection()
10    cur = conn.cursor()
11    sql = "INSERT INTO 'user' ('user_email', 'user_firstname', 'user_lastname', 'user_password', 'userType_id') VALUES (%s, %s, %s, %s, %s)"
12    cur.execute(sql, (email, firstname, lastname, hashed_password, user))
13    conn.commit()
14    conn.close()
15    flash("Register Success", "success")
16    # INSERT INTO 'user' ('user_email', 'user_firstname', 'user_lastname', 'user_password', 'userType_id') VALUES ('{value-1}', '{value-2}', '{value-3}', '{value-4}', '{value-5}', '{value-6}')
17    return redirect(url_for('signInPage'))

```

- การทำงานของ route("/sign_up/actions") คือ การทำงานของการสมัคร User หรือ Admin เข้าสู่ฐานข้อมูลผ่านทางหน้าเว็บไซต์ ผ่านทางหน้าของ signup.html
- การทำงานจะรับเป็น method == "POST" มีการรับค่าผ่านตัวของ form ผ่านทาง HTML มาเก็บในตัวแปรดังต่อไปนี้
 - firstname = มาจากช่อง input HTML ผ่านทาง form['firstname']
 - lastname = มาจากช่อง input HTML ผ่านทาง form['lastname']
 - email = มาจากช่อง input HTML ผ่านทาง form['email']
 - password = มาจากช่อง input HTML ผ่านทาง form['password']
 - กำหนด user = 2 (ค่าคงที่ในการกำหนดว่า user นั้นเป็น User ไม่ใช่ Admin)
- ทำการเพิ่มข้อมูลที่รับเข้ามา ไปยังฐานข้อมูล
- ส่ง message flashing เป็น ("Register Success") แล้วกลับไปยังหน้า signin.html ตาม url_for('signInPage') [signInPage เป็นชื่อ method ของ route("/sign_in")]

```

1 # action adding product
2 @app.route("/<string:userType_name>/<string:user_id>/insert_product", methods=['POST'])
3 def adminPageActionAddingProduct(userType_name, user_id):
4     if request.method == 'POST':
5         product_name = request.form['product_name']
6         product_color = request.form['product_color']
7         product_cost = request.form['product_cost']
8         product_category = request.form['product_category']
9         file_image = request.files['product_image']
10        filename = secure_filename(product_name + ".jpg")
11
12        # path folder for save img for upload
13        file_image.save(
14            'D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/img_product_into_db/' + filename)
15
16        # path folder for save img for call img into database with feature
17        folder_path = "D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/img_product_into_db/{}".format(
18            filename)
19        feature = extract(folder_path)
20        js = json.dumps(feature.tolist())
21        product_vector = js
22
23        con = openConnection()
24        cur = con.cursor()
25        sql = "INSERT INTO 'product' ('product_name', 'product_img_name', 'product_cost', 'product_vector', 'product_color', 'productCategory_id') VALUES (%s, %s, %s, %s, %s, %s)"
26        cur.execute(sql, (product_name, filename, product_cost, product_vector, product_color, product_category))
27        con.commit()
28        con.close()
29        flash("Adding product success", "success")
30        return redirect(url_for('adminPageProduct', userType_name=userType_name, user_id=user_id))
31

```

- การทำงานของ route("/<string:userType>/<string:user_id>/insert_product") คือ การทำงานของ Admin ในการเพิ่มข้อมูลในส่วนของสินค้า (Product)
- การทำงานจะรับเป็น method == "POST" มีการรับค่าผ่านตัวของ form ผ่านทาง HTML มาเก็บในตัวแปรดังต่อไปนี้
 - product_name = มาจากช่อง input HTML ผ่านทาง form['product_name']

- product_color = มาจากช่อง input HTML ผ่านทาง form['product_color']
- product_cost = มาจากช่อง input HTML ผ่านทาง form['product_cost']
- product_category = มาจากช่อง input HTML ผ่านทาง form['product_category']
- file_image = มาจากช่อง input HTML ผ่านทาง form['product_image']
- มีการกำหนด filename ให้เป็นชื่อของรูปภาพตามตัวแปร product_name หลังจากนั้นทำการนำภาพที่ได้ไปเก็บเข้าไปที่ Folder : ../static/img_product_into_db
- จากนั้นทำการนำภาพที่ save ไปทำการเพิ่ม vector ของรูปภาพนั้นไปเก็บในตัวแปร product_vector
- ทำการเพิ่มข้อมูลทั้งหมดเข้าไปที่ฐานข้อมูล ตามตัวแปรที่เก็บข้อมูลมาทั้งหมด และ message flashing เป็น ("Adding product success") ส่งไปที่ url_for('adminPageProduct') [adminPageProduct เป็นชื่อ method ของ route("/<string:userType>/<string:user_id>/products")] ซึ่งเป็นหน้าแสดงสินค้าทั้งหมดในฐานข้อมูล

```

1 @app.route("/<string:userType_name>/<string:user_id>/editProduct/<string:product_id>")
2 def adminPageEditProduct(userType_name, user_id, product_id):
3     conn = openConnection()
4     cur = conn.cursor()
5     sql = "SELECT * FROM 'product' NATURAL JOIN product_category WHERE product_id = %s"
6     cur.execute(sql, (product_id))
7     result_product = cur.fetchone()
8     conn.close()
9
10    conn = openConnection()
11    cur = conn.cursor()
12    sql = "SELECT * FROM product_category"
13    cur.execute(sql)
14    result_product_category = cur.fetchall()
15    conn.close()
16    return render_template("pageWithUser/Adminpage/Editproduct/editproduct.html", data_id = user_id, data_userType = userType_name, product = result_product, product_category = result_product_category )

```

- การทำงานของ route("/<string:userType>/<string:user_id>/<string:product_id>") คือ การทำงานของ Admin ในการแก้ไขข้อมูลในส่วนของสินค้า (Product) โดยการรับค่าของ product_id นั้นๆมาด้วย
- ทำการหา product ตาม product_id ที่ได้รับเข้ามาแล้ว ทำการเปิดหน้าต่างของ editproduct.html ซึ่งทำการส่งค่า product_id ที่ต้องการแก้ไขและ product_category สำหรับการทำให้ dropdown

```

1 @app.route("/<string:userType_name>/<string:user_id>/editProduct/<string:product_id>/<string:old_filename>", methods=['POST'])
2 def adminPageActionEditProduct(userType_name, user_id, product_id, old_filename):
3     folder_path = os.path.join("D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/img_product_into_db/", old_filename+".jpg")
4     if os.path.exists(folder_path):
5         os.remove(folder_path)
6         # delete old file in database
7         product_name = request.form['product_name']
8         product_color = request.form['product_color']
9         product_cost = request.form['product_cost']
10        product_category = request.form['product_category']
11        file_image = request.files['product_image']
12        filename = secure_filename(product_name+".jpg")
13
14        # path folder for save img for upload
15        file_image.save(
16            'D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/img_product_into_db/' + filename)
17
18        # path folder for save img for coll img into database with feature
19        folder_path = "D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/img_product_into_db/{}".format(
20            filename)
21        feature = extract(folder_path)
22        js = json.dumps(feature.tolist())
23        product_vector = js
24
25        conn = openConnection()
26        cur = conn.cursor()
27        sql = "UPDATE 'product' SET 'product_name' = %s, 'product_img_name' = %s, 'product_cost' = %s, 'product_vector' = %s, 'product_color' = %s, 'productCategory_id' = %s WHERE product_id = %s"
28        cur.execute(sql, (product_name, filename, product_cost, product_vector, product_color, product_category, product_id))
29        conn.commit()
30        conn.close()
31
32        return redirect(url_for('adminPageProduct', userType_name = userType_name, user_id = user_id))

```

- การทำงานของ
route("/<string:userType>/<string:user_id>/editProduct/<string:product_id>/<string:old_filename>") คือ การทำงานฟังก์ชันในส่วนของการ กด Submit การแก้ไข Product นั้นๆ
- โดยมีการรับค่าของตัว product_id และ old_filename หรือก็คือ ชื่อไฟล์รูปภาพเก่าของสินค้า (Product) ที่ต้องการแก้ไข
- การทำงานในส่วนแรกจะเป็นการลบ ไฟล์รูปภาพเก่าที่อยู่ Folder : ../static/img_product_into_db
- มีการรับค่าผ่านตัวของ form ผ่านทาง HTML มาเก็บในตัวแปรดังต่อไปนี้
 - product_name = มาจากช่อง input HTML ผ่านทาง form['product_name']
 - product_color = มาจากช่อง input HTML ผ่านทาง form['product_color']
 - product_cost = มาจากช่อง input HTML ผ่านทาง form['product_cost']
 - product_category = มาจากช่อง input HTML ผ่านทาง form['product_category']
 - file_image = มาจากช่อง input HTML ผ่านทาง form['product_image']
- การทำงานทั้งหมดคล้ายกับการเพิ่มสินค้าใหม่เข้าฐานข้อมูล แต่แตกต่างกันตรงที่ เป็นการ Update สินค้าตาม product_id ที่ได้รับเข้ามา
- หลังจากนั้นทำการกลับไปหน้าแสดงสินค้าทั้งหมดของ Admin ตาม url_for('adminPageProduct') [adminPageProduct เป็นชื่อ method ของ route("/<string:userType>/<string:user_id>/products")]

```

1 @app.route("/<string:userType_name>/<string:user_id>/deleteProduct/<string:product_id>/<string:product_name>")
2 def adminPageActionDeleteProduct(userType_name, user_id, product_id, product_name):
3     folder_path = os.path.join("D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/img_product_into_db/", product_name)
4     if os.path.exists(folder_path):
5         os.remove(folder_path)
6         # delete old file in database
7     conn = openConnection()
8     cur = conn.cursor()
9     sql = "DELETE FROM `product` WHERE product_id = %s"
10    cur.execute(sql, (product_id))
11    conn.commit()
12    conn.close()
13    flash("Delete product success", "success")
14    return redirect(url_for('adminPageProduct', userType_name = userType_name, user_id = user_id))

```

- การทำงานของ
route("/<string:userType>/<string:user_id>/deleteProduct/<string:product_id>/<string:product_name>") คือ การทำงานฟังก์ชันในส่วนของการ กด Submit การลบ Product นั้นๆ
- โดยมีการรับค่าของตัว product_id และ product_name หรือก็คือ ชื่อไฟล์รูปภาพเก่าของสินค้า (Product) ที่ต้องการลบ
- ทำการลบสินค้า (Product) นั้นๆ ตามที่ product_id ที่ได้รับเข้ามำกำหนด
- ทำการกลับไปหน้าแสดงสินค้าทั้งหมดของ Admin ตาม url_for('adminPageProduct') [adminPageProduct เป็นชื่อ method ของ route("/<string:userType>/<string:user_id>/products")] และ message flashing เป็น ("Delete product success")

```
1 @app.route('/search', methods=['POST'])
2 def searchAction():
3     file = request.files['fileUpload']
4     filename = secure_filename("search.jpg")
5     file.save('D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/search_upload/' + filename)
6     folder_path = "D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/search_upload/{}".format(filename)
7     df = searchVector(extract(folder_path))
8     id = df['id'].values.tolist()
9     result = np.empty((0, 8))
10    con = openConnection()
11    cur = con.cursor()
12    sql = "SELECT * FROM `product` NATURAL JOIN product_category WHERE product_id = %s"
13    for i in id[:4]: #best of ...
14        cur.execute(sql, (i,))
15        row = cur.fetchall()
16        result = np.append(result, np.array(row), axis=0)
17    result = tuple(map(tuple, result))
18    return render_template('pageWithoutUser/resultSearch.html', datas=result)
```

- การทำงานของ route("/search") หรือ def searchAction() คือ การทำฟังก์ชันในส่วนของการนำ Model มาใช้ในการค้นหาสินค้าที่มีลักษณะใกล้เคียงกับรูปภาพที่นำเข้าสู่ระบบ ในส่วนของหน้าที่ไม่ได้เข้าสู่ระบบของ User
- การทำงาน คือ การนำรูปภาพที่ได้มาทำ ตั้งชื่อเป็น "search.jpg" ไปเก็บที่ Folder : ../static/search_upload
- หลังจากนั้นทำการ Query โดยการเรียงลำดับในส่วน of รูปภาพที่มีความใกล้เคียงมากที่สุด ไปยังน้อยที่สุด
- กลับไปยังหน้า resultSearch.html ใน Folder : pageWithoutUser โดยการส่ง product_id ทั้งหมดที่ได้มาไปด้วยเพื่อใช้สำหรับแสดงผลลัพธ์ต่อไป

```

1 app.route('/<string:userType_name>/<string:user_id>/search', methods=['POST'])
2 def searchActionWithUser(userType_name, user_id):
3     conn = openConnection()
4     cur = conn.cursor()
5     # sql = "SELECT * FROM 'product' NATURAL JOIN product_category"
6     sql = "SELECT product_id FROM 'product' NATURAL JOIN product_category WHERE product_id IN ( SELECT product_id FROM favoriteproduct WHERE user_id = %s)"
7     cur.execute(sql,(user_id))
8     result_product = cur.fetchall()
9     conn.close()
10    # print(result_product)
11    # print(type(result_product))
12    # for save in result_product :
13    #     print(save)
14    file = request.files['fileUpload']
15    filename = secure_filename("search.jpg")
16    file.save('D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/search_upload/'+ filename)
17    folder_path = "D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Flask_Run_Application/static/search_upload/{}/".format(filename)
18    df = searchVector(extract(folder_path))
19    id = df['id'].values.tolist()
20    # print(id)
21    # for result_id in id:
22    #     i = 0
23    #     while (i< len(result_product)):
24    #         print(f"result_id {result_id} | save {result_product[i][0]}")
25    #         print(result_product[i][0] == result_id)
26    #         i += 1
27    result = np.empty((0, 0))
28    con = openConnection()
29    cur = con.cursor()
30    sql = "SELECT * FROM 'product' NATURAL JOIN product_category WHERE product_id = %s"
31    # AND product_id NOT IN ( SELECT product_id FROM favoriteproduct WHERE user_id = %s)
32    for save in id :
33        print(save)
34
35    i = 0
36    listSave = []
37    # cur.execute(sql,(id[i]))
38    # row = cur.fetchall()
39    # result = np.append(result,np.array(row),axis=0)
40    while (i<len(id)): #best of ...
41        favorite = False
42        saveID = id[i]
43        y = 0
44        while (y<len(result_product)):
45            print(f"result_product = {result_product[y][0]} | id = {id[i]}")
46            print(result_product[y][0] == id[i])
47            if result_product[y][0] == id[i]:
48                favorite = True
49                saveID = id[i]
50            y += 1
51        if not favorite :
52            listSave.append(saveID)
53        i += 1
54
55    print(listSave)
56
57    if len(listSave) >= 4 :
58        for i in listSave[:4]: #best of ...
59            cur.execute(sql,(i))
60            row = cur.fetchall()
61            result = np.append(result,np.array(row),axis=0)
62    elif len(listSave) > 0 and len(listSave) < 4:
63        for i in listSave: #best of ...
64            cur.execute(sql,(i))
65            row = cur.fetchall()
66            result = np.append(result,np.array(row),axis=0)
67
68    result = tuple(map(tuple, result))
69    return render_template('pageWithUser/Userpage/userResultSearch.html', data_userType=userType_name, data_id=user_id, datas=result)

```

- การทำงานของ route (“/<string:userType>/<string:user_id>/search”) คือ การทำฟังก์ชันในส่วนของการนำ Model มาใช้ในการค้นหาสินค้าที่มีลักษณะใกล้เคียงกับรูปภาพที่นำเข้าสู่ระบบ ในส่วนของหน้าที่มีการเข้าสู่ระบบของ User
- การทำงานในส่วนแรก คือ การหาสินค้าทั้งที่ User นั้นได้ทำการเพิ่มสินค้าที่ชื่นชอบไปเรียบร้อยแล้ว
- การทำงานในส่วนต่อมา คือ การนำรูปภาพที่ได้มาทำ ตั้งชื่อเป็น “search.jpg” ไปเก็บที่ Folder :
../static/search_upload และทำการเรียง product_id ที่มีภาพใกล้เคียงกับรูปภาพที่นำเข้าไปจากมากที่สุดไปน้อยสุด
- จากนั้นทำการ Query สินค้าทั้งหมดที่มีอยู่ในระบบมา แล้วทำการเปรียบเทียบกับสินค้าที่ User นั้นๆ ทำการเพิ่มสินค้าที่ชื่นชอบไปเรียบร้อยแล้ว
 - หากมีสินค้า : สินค้าอื่นๆที่ไม่จะถูกเพิ่มไปในตัวแปร listSave
 - หากไม่มีสินค้า : สินค้าอื่นๆก็就会被เพิ่มไปในตัวแปร listSave

- จากนั้นทำการดูว่า listSave มีความยาวมากกว่าหรือเท่ากับ 4 หรือไม่
 - หากมากกว่าหรือเท่ากับ 4 : ก็ทำการส่งข้อมูลไปเพียง 4 ตัวแรกใน listSave เท่านั้น
 - หากน้อยกว่า 4 : ก็ทำการส่งข้อมูลทั้งหมดใน listSave
- ทำการเพิ่มหน้า userResultSearch.html โดยการส่ง product_id ทั้งหมดที่ได้มาไปด้วยเพื่อใช้สำหรับแสดงผลต่อไป

```
1 @app.route('/<string:userType_name>/<string:user_id>/favorite/<string:product_id>')
2 def actionFavorite(userType_name, user_id, product_id):
3     conn = openConnection()
4     cur = conn.cursor()
5     sql = "INSERT INTO `favoriteproduct`(`user_id`, `product_id`) VALUES (%s, %s)"
6     cur.execute(sql, (user_id, product_id))
7     conn.commit()
8     conn.close()
9     return redirect(url_for('userPageShopping', userType_name = userType_name, user_id = user_id))
```

- การทำงานของ route("/<string:userType>/<string:user_id>/favorite/<string:product_id>") คือ การทำฟังก์ชันในการเพิ่มสินค้าที่ชื่นชอบให้กับ User นั้นๆ
- การทำงาน คือ การรับค่าตัวของ product_id มาแล้วทำการเพิ่มข้อมูลเข้าในฐานข้อมูล โดยการส่ง user_id กับ product_id ไปทำการเพิ่ม
- หลังจากนั้นทำการเพิ่มหน้า แสดงสินค้า ตาม url_for('userPageShopping') [userPageShopping เป็นชื่อ method ของroute("/<string:userType_name>/<string:user_id>/shopping")]

```
1 @app.route('/<string:userType_name>/<string:user_id>/removeFromfavorite/<string:product_id>')
2 def actionRemoveFavorite(userType_name, user_id, product_id):
3     conn = openConnection()
4     cur = conn.cursor()
5     sql = "DELETE FROM `favoriteproduct` WHERE user_id = %s AND product_id = %s"
6     cur.execute(sql, (user_id, product_id))
7     conn.commit()
8     conn.close()
9     return redirect(url_for('userPageFavorite', userType_name = userType_name, user_id = user_id))
```

- การทำงานของ route("/<string:userType>/<string:user_id>/removeFromfavorite/<string:product_id>") คือ การทำฟังก์ชันในการลบสินค้าที่ชื่นชอบให้กับ User นั้นๆ
- การทำงาน คือ การรับค่าตัวของ product_id มาแล้วทำการลบข้อมูลเข้าในฐานข้อมูล โดยการส่ง user_id กับ product_id ไปทำการลบ

- หลังจากนั้นทำการเพิ่มหน้า แสดงสินค้า ตาม url_for('userPageShopping') [userPageShopping เป็นชื่อ method ของroute("/<string:userType_name>/<string:user_id>/shopping")]

การโชว์หน้าและฟังก์ชันการทำงานหลังการเข้าสู่ระบบ

```
1 @app.route("/<string:userType_name>/<string:user_id>/users")
2 def adminPageUser(userType_name, user_id):
3     if 'user' in session:
4         conn = openConnection()
5         cur = conn.cursor()
6         sql = "SELECT * FROM 'user' NATURAL JOIN user_type WHERE user_id != %s"
7         cur.execute(sql, (user_id))
8         result_user = cur.fetchall()
9         conn.close()
10        return render_template('pageWithUser/Adminpage/user/user.html', data_user=result_user, data_id=user_id, data_userType=userType_name)
11    else:
12        return redirect(url_for('signInPage'))
```

- การทำงานของ route("/<string:userType>/<string:user_id>/users") คือ การแสดงหน้า user.html หรือ หน้าแสดง user ทั้งหมดในระบบ จากการเข้าสู่ระบบที่เรียบร้อยของตัว Admin
- การ Query คือ การที่เรียกรายชื่อ User ทั้งหมดในฐานข้อมูลยกเว้นตัวเอง ขึ้นมาแสดงให้เห็น
- โดยมีการตรวจสอบการเข้าถึงด้วยตัวของ sessions ที่มีชื่อ "user"

```
1 @app.route("/<string:userType_name>/<string:user_id>/products")
2 def adminPageProduct(userType_name, user_id):
3
4     if 'user' in session:
5         conn = openConnection()
6         cur = conn.cursor()
7         sql = "SELECT * FROM 'product' NATURAL JOIN product_category"
8         cur.execute(sql)
9         result_product = cur.fetchall()
10        conn.close()
11        return render_template('pageWithUser/Adminpage/product/product.html', data_product=result_product, data_id=user_id, data_userType=userType_name)
12    else:
13        return redirect(url_for('signInPage'))
```

- การทำงานของ route("/<string:userType>/<string:user_id>/products") คือ การแสดงหน้า product.html หรือ product ทั้งหมดในระบบ จากการเข้าสู่ระบบที่เรียบร้อยของตัว Admin
- การ Query คือ การที่เรียกรายชื่อสินค้าทั้งหมดในฐานข้อมูล ขึ้นมาแสดงให้เห็น
- โดยมีการตรวจสอบการเข้าถึงด้วยตัวของ sessions ที่มีชื่อ "user"

```

1 @app.route("/<string:userType_name>/<string:user_id>/products/addProduct")
2 def adminPageAddProduct(userType_name, user_id):
3     if 'user' in session:
4         con = openConnection()
5         cur = con.cursor()
6         cur.execute("SELECT * FROM 'product_category'")
7         product_category_result = cur.fetchall()
8         return render_template('pageWithUser/Adminpage/Addproduct/addproduct.html', data_id=user_id, data_userType=userType_name, product_category=product_category_result)
9     else:
10        return redirect(url_for('signInPage'))

```

- การทำงานของ route("/<string:userType>/<string:user_id>/addProduct") คือ การแสดงหน้า addproduct.html หรือ หน้าสำหรับการเพิ่มสินค้าผ่านเว็บไซต์เข้าสู่ฐานข้อมูล จากการเข้าสู่ระบบที่เรียบร้อยของตัว Admin
- การ Query คือ การที่เรียกรายชื่อประเภทสินค้าทั้งหมดในฐานข้อมูล ขึ้นมาเพื่อใช้สำหรับการทำ dropdown
- โดยมีการตรวจสอบการเข้าถึงด้วยตัวของ sessions ที่มีชื่อ "user"

```

1 @app.route("/<string:userType_name>/<string:user_id>/shopping")
2 def userPageShopping(userType_name, user_id):
3     if 'user' in session:
4         conn = openConnection()
5         cur = conn.cursor()
6         # sql = "SELECT * FROM 'product' NATURAL JOIN 'product_category'"
7         sql = "SELECT * FROM 'product' NATURAL JOIN 'product_category' WHERE product_id NOT IN ( SELECT product_id FROM favoriteproduct WHERE user_id = %s)"
8         cur.execute(sql, (user_id))
9         result_product = cur.fetchall()
10        conn.close()
11        return render_template('pageWithUser/Userpage/user.html', data_id=user_id, data_userType=userType_name, product=result_product)
12    else:
13        return redirect(url_for('signInPage'))

```

- การทำงานของ route("/<string:userType>/<string:user_id>/shopping") คือ การแสดงหน้า user.html หรือ หน้าแสดงสินค้าทั้งหมดที่มีอยู่ในระบบ จากการเข้าสู่ระบบที่เรียบร้อยของตัว User
- การ Query คือ การที่เรียกสินค้าทั้งหมดที่มีอยู่ในระบบและไม่ได้อยู่ในสินค้าที่ชื่นชอบของ User นั้นๆ
- โดยมีการตรวจสอบการเข้าถึงด้วยตัวของ sessions ที่มีชื่อ "user"

```

1 @app.route("/<string:userType_name>/<string:user_id>/search")
2 def userPageSearch(userType_name, user_id):
3     if 'user' in session:
4         return render_template('pageWithUser/Userpage/userSearch.html', data_id=user_id, data_userType=userType_name)
5     else:
6         return redirect(url_for('signInPage'))

```

- การทำงานของ route("/<string:userType>/<string:user_id>/search") คือ การแสดงหน้า userSearch.html หรือ หน้าแสดงฟังก์ชันการทำงานในการเพิ่มรูปเข้าระบบเพื่อการค้นหา
- โดยมีการตรวจสอบการเข้าถึงด้วยตัวของ sessions ที่มีชื่อ "user"


```

1 @app.route("/<string:userType_name>/<string:user_id>/about_us")
2 def userPageAboutUs(userType_name, user_id):
3     if 'user' in session:
4         return render_template('pageWithUser/Userpage/userAboutUs.html', data_id=user_id, data_userType=userType_name)
5     else:
6         return redirect(url_for('signInPage'))

```

- การทำงานของ route("/<string:userType>/<string:user_id>/about_us") คือ การแสดงหน้า userAboutUs.html หรือ หน้าแสดงผู้จัดทำและคำอธิบายที่เกี่ยวข้อง
- โดยมีการตรวจสอบการเข้าถึงด้วยตัวของ sessions ที่มีชื่อ "user"

```

1 @app.route("/<string:userType_name>/<string:user_id>/favorite")
2 def userPageFavorite(userType_name, user_id):
3     if 'user' in session:
4         conn = openConnection()
5         cur = conn.cursor()
6         sql = "SELECT * FROM 'product' NATURAL JOIN product_category WHERE product_id IN ( SELECT product_id FROM favoriteproduct WHERE user_id = %s)"
7         cur.execute(sql, (user_id))
8         result = cur.fetchall()
9         conn.close()
10        return render_template('pageWithUser/Userpage/userFavorite.html', data_id=user_id, data_userType=userType_name, product = result)
11    else:
12        return redirect(url_for('signInPage'))

```

- การทำงานของ route("/<string:userType>/<string:user_id>/favorite") คือ การแสดงหน้า userFavorite.html หรือ หน้าแสดงสินค้าทั้งหมดที่ User นั้นๆ ได้เพิ่มเข้าสินค้าที่ชื่นชอบ
- การ Query คือ การที่เรียกสินค้าทั้งหมดที่มีอยู่ในระบบและอยู่ในสินค้าที่ชื่นชอบของ User นั้นๆ
- โดยมีการตรวจสอบการเข้าถึงด้วยตัวของ sessions ที่มีชื่อ "user"

```

1 model = keras.models.load_model(
2     'D:/ITM_Group4_Reverse_Image_Search_for_Online_Shopping/Model/preweight/efficientnet.h5', custom_objects={'KerasLayer': hub.KerasLayer})
3 model.summary()

```

- เป็นการโหลดโมเดลจากไฟล์เครื่องมาเติมไว้

```

1 def extract(file):
2     IMAGE_SHAPE = (224, 224)
3     file = Image.open(file).convert('L').resize(IMAGE_SHAPE)
4     file = np.stack((file,)*3, axis=-1)
5     file = np.array(file)/255.0
6     embedding = model.predict(file[np.newaxis, ...])
7     vgg16_feature_np = np.array(embedding)
8     flattened_feature = vgg16_feature_np.flatten()
9     return flattened_feature

```

- เป็นฟังก์ชันที่ใช้ในการ feature extraction จากรูปภาพแล้ว return ออกมาเป็น feature array ขนาด 1280

```

1 def searchVector(vector):
2     con = openConnection()
3     cur = con.cursor()
4     cur.execute("Select * From product")
5     rows = cur.fetchall()
6     data = {
7         "id": [], # row[0]
8         "distance": [] # row[4] = vector
9     }
10    df = pd.DataFrame(data)
11    metric = 'cosine'
12    for row in rows:
13        r = np.array(json.loads(row[4]))
14        print(type(r))
15        dc = distance.cdist([vector], [r], metric)[0]
16        df.loc[len(df.index)] = [row[0], dc]
17    return df.sort_values(by='distance')

```

- เป็นฟังก์ชันที่ใช้ในการค้นหารูปโดยรับพารามิเตอร์เป็น feature ของรูปภาพจากนั้นจะ Connect database เพื่อดึงข้อมูลและ feature ออกมาแล้วมาวัดแบบ cosine กับ feature ที่รับมา จากนั้นใส่ค่า distance และ Id ของสินค้าใน dataframe สุดท้ายก็ return dataframe ที่ sort จาก distance ออกไป

Model

```
In [1]: import cv2
import os
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow import keras
from PIL import Image
from scipy.spatial import distance
import pandas as pd
import json

#Display image
import glob
import random
import base64
from io import BytesIO
from IPython.display import HTML

os.environ['TFHUB_CACHE_DIR'] = '/home/user/workspace/tf_cache'
```

Libraries

- os ใช้ในการเรียกไฟล์ต่างๆภายในเครื่อง
- np ใช้ในเปลี่ยนรูปภาพให้เป็นarray เพื่อที่จะInputไปในตัวModel
- tensorflow ใช้สร้าง, save, load modelออกมาใช้งาน
- tensorflow_hub ใช้เรียก Pre-trained Model จาก url
- Image ใช้อ่านรูปจากpath
- scipy ใช้ในคำนวณหา distance ของ vector
- pandas ใช้ในการจัดเรียงข้อมูลในรูปแบบตาราง
- Json ใช้ dumps ข้อมูลให้เป็น json

Pre-Trained Model

```
In [2]: model_url = "https://tfhub.dev/tensorflow/efficientnet/lite0/feature-vector/2"

IMAGE_SHAPE = (224, 224)

layer = hub.KerasLayer(model_url, input_shape=IMAGE_SHAPE+(3,))
model = tf.keras.Sequential([layer])
```

```
In [8]: model.save('./preweight/efficientnet.h5')
```

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

```
In [9]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
keras_layer (KerasLayer)	(None, 1280)	3413024

=====
 Total params: 3,413,024
 Trainable params: 0
 Non-trainable params: 3,413,024
 =====

เรียก layer จาก url แล้วนำมาใส่ใน model sequential จากนั้นก็ save เป็น .h5 เพื่อนำไปใช้งานในเว็บ

Model input=224x224x3 output = 1280

```
In [3]: IMAGE_SHAPE = (224, 224)
test_model = keras.models.load_model('./preweight/efficientnet.h5', custom_objects={'KerasLayer': hub.KerasLayer})

WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
```

```
In [4]: test_model.summary()

Model: "sequential"

Layer (type) Output Shape Param #
=====
keras_layer (KerasLayer) (None, 1280) 3413024
=====
Total params: 3,413,024
Trainable params: 0
Non-trainable params: 3,413,024
```

Load model จาก file มาใช้งาน

```
In [5]: def extract(file):
file = Image.open(file).convert('L').resize(IMAGE_SHAPE)
display(file)

file = np.stack((file,)*3, axis=-1)

file = np.array(file)/255.0

embedding = test_model.predict(file[np.newaxis, ...])
# print(embedding)
vgg16_feature_np = np.array(embedding)
flattended_feature = vgg16_feature_np.flatten()

#print(len(flattended_feature))
#print(flattended_feature)
#print('-----')
return flattened_feature
```

Fuction ที่ใช้ในการ extract feature รับ input เป็น file path จากนั้น fuction อ่านรูปจาก path และเปลี่ยนเป็น array โดย numpy และ /255 เพื่อ normalized แล้วให้ model predict และ flatten เพื่อเปลี่ยน array หลายมิติเป็นมิติเดียว ขนาด 1280

```
In [6]: def extractAll(folder):
images = []
data = {
    "feature": [],
    "name": [],
    "file": []
}
df = pd.DataFrame(data)
for filename in os.listdir(folder):
    img = extract(os.path.join(folder,filename))
    if img is not None:
        # images.append(np.array(img))
        df.loc[len(df.index)] = [img,filename,os.path.join(folder,filename)]
return df
```

Fuction extractAll ใช้ในการ extract รูปทั้ง folder ใส่ใน data frame มี column feature, name, filename

```
In [7]: def findClosest(x,df):
metric = 'cosine'
data = {
    "distance": [],
    "name": [],
    "file": []
}
result_df = pd.DataFrame(data)
for i in df.index:
    dc = distance.cdist([x], [df['feature'][i]], metric)[0]
    if dc is not None:
        result_df.loc[len(result_df.index)] = [dc,df['name'][i],df['file'][i]]
return result_df.sort_values(by='distance')
```

Fuction findClosest input x คือ featrue ที่ต้องการค้นหา และ df คือ data frame ที่มี featrueเก็บอยู่ จากนั้นก็นำมาเทียบกันแล้วเรียงลำดับใน dataframe

```
In [8]: pd.set_option('display.max_colwidth', -1)
```

```
def get_thumbnail(path):  
    i = Image.open(path)  
    i.thumbnail((150, 150), Image.LANCZOS)  
    return i  
  
def image_base64(im):  
    if isinstance(im, str):  
        im = get_thumbnail(im)  
    with BytesIO() as buffer:  
        im.save(buffer, 'jpeg')  
        return base64.b64encode(buffer.getvalue()).decode()  
  
def image_formatter(im):  
    return f"<img src='data:image/jpeg;base64,{image_base64(im)}'>"
```

```
C:\Users\JoEy.S\AppData\Local\Temp\ipykernel_24464\1694978098.py:1: FutureWarning: Passing a negative integer is deprecated in version 1.0 and will not be supported in future version. Instead, use None to not limit the column width.  
pd.set_option('display.max_colwidth', -1)
```

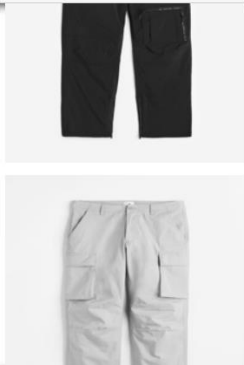
ใช้อ่าน file path แล้วแปลงเป็น format ที่ HTML อ่านแล้วสามารถแสดงรูปภาพได้

เรียก function มาใช้งาน

```
In [31]: r = extract(r"C:/Sheet/Test/sneaker (1).jpg")
```



```
In [9]: images = extractAll(r"C:/Sheet/Test")
```



```
In [32]: result = findClosest(r,images)
```

C:\Users\JoEy.S\anaconda3\envs\tmrcnnt3\lib\site-packages\pandas\core\dtypes\cast.py:948: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
element = np.asarray(element)

```
In [33]: result
```






Out[33]:	distance	name	file
29	[1.1102230246251565e-16]	sneaker (1).jpg	C:\Sheet\Test\sneaker (1).jpg
32	[0.14205340358352636]	sneaker (3).jpg	C:\Sheet\Test\sneaker (3).jpg
31	[0.23115667130312123]	sneaker (2).jpg	C:\Sheet\Test\sneaker (2).jpg
37	[0.3198013265832468]	sneaker (8).jpg	C:\Sheet\Test\sneaker (8).jpg
35	[0.38151941451441107]	sneaker (6).jpg	C:\Sheet\Test\sneaker (6).jpg
36	[0.3889888465864849]	sneaker (7).jpg	C:\Sheet\Test\sneaker (7).jpg
30	[0.39931405444172663]	sneaker (10).jpg	C:\Sheet\Test\sneaker (10).jpg
38	[0.4101554380257083]	sneaker (9).jpg	C:\Sheet\Test\sneaker (9).jpg
33	[0.42626819647712116]	sneaker (4).jpg	C:\Sheet\Test\sneaker (4).jpg
42	[0.4287947712296447]	test_sneaker (1).jpg	C:\Sheet\Test\test_sneaker (1).jpg
34	[0.43165794571764204]	sneaker (5).jpg	C:\Sheet\Test\sneaker (5).jpg
15	[0.5051153535105344]	sandals (1).jpg	C:\Sheet\Test\sandals (1).jpg
17	[0.54153259753613]	sandals (3).jpg	C:\Sheet\Test\sandals (3).jpg
43	[0.5615610592319076]	test_sneaker (2).jpg	C:\Sheet\Test\test_sneaker (2).jpg

```
In [34]: result['image'] = result.file.map(lambda f: get_thumbnail(f))
result.head()
```

Out[34]:	distance	name	file	image
29	[1.1102230246251565e-16]	sneaker (1).jpg	C:\Sheet\Test\sneaker (1).jpg	<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=150x107 at 0x23D9FFF84F0>
32	[0.14205340358352636]	sneaker (3).jpg	C:\Sheet\Test\sneaker (3).jpg	<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=150x107 at 0x23DEF0481C0>
31	[0.23115667130312123]	sneaker (2).jpg	C:\Sheet\Test\sneaker (2).jpg	<PIL.WebPImagePlugin.WebPImageFile image mode=RGB size=150x107 at 0x23DEF0480D0>
37	[0.3198013265832468]	sneaker (8).jpg	C:\Sheet\Test\sneaker (8).jpg	<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=150x107 at 0x23DEF048160>
35	[0.38151941451441107]	sneaker (6).jpg	C:\Sheet\Test\sneaker (6).jpg	<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=150x107 at 0x23DEF048850>

แปลง file name ให้ HTML สามารถแสดงรูปได้

```
In [35]: HTML(result[['name','distance', 'image']]).to_html(formatters={ 'image': image_formatter}, escape=False))
```

Out[35]:	name	distance	image
29	sneaker (1).jpg	[1.1102230246251565e-16]	
32	sneaker (3).jpg	[0.14205340358352636]	
31	sneaker (2).jpg	[0.23115667130312123]	
37	sneaker (8).jpg	[0.3198013265832468]	
			

ใช้ HTML แสดงตารางพร้อมรูป

Project team lesson learn

หัวข้อ	ดี	แย่	แนวทางแก้ไข
บทบาทหน้าที่และความรับผิดชอบตามส่วนงาน	<ul style="list-style-type: none"> ทุกคนมีหน้าที่ที่ชัดเจน รับผิดชอบในส่วนของตัวเอง การทำงานสามารถทำได้พร้อมๆกัน 	<ul style="list-style-type: none"> คนอื่นไม่สามารถลงไปช่วยแก้ไขงานในส่วนนั้นได้ หากงานในส่วนนั้นมีขนาดใหญ่คนที่รับผิดชอบก็จะเกิดความเหนื่อยล้า 	<ul style="list-style-type: none"> เมื่อหน้าที่ความรับผิดชอบให้เกิดความสอดคล้องกัน งานในส่วนต่างๆควรจะมีคนรับผิดชอบ อย่างน้อย 2 คน
การบริหารรูปแบบชัดเจน	<ul style="list-style-type: none"> การทำงานเป็นระบบ สามารถทำให้งานเดินตามแผนที่วางไว้ได้ ได้รับงานแน่นอน 	<ul style="list-style-type: none"> เกิดความกดดันในการทำงาน ลูกทีมมีความเหนื่อยล้าเป็นพิเศษ 	<ul style="list-style-type: none"> ยืดหยุ่นการทำงาน รับฟังความเห็นของลูกทีมในเรื่องการส่งงาน
การวางแผนการทำงานตามที่วางเอาไว้	<ul style="list-style-type: none"> ทุกคนรับทราบถึงขั้นตอน และกระบวนการทำงานทั้งหมด ทุกคนเห็นภาพรวมของโปรเจก และเป้าหมายในการทำงาน 	<ul style="list-style-type: none"> สมาชิกในทีมตอนวางแผนมีความเห็นที่ต่างกัน อาจทำให้เกิดการสื่อสารที่ผิดพลาดและเข้าใจผิดบางจุด 	<ul style="list-style-type: none"> การสื่อสารกับสมาชิกในทีมให้ชัดเจนและเข้าใจในทิศทางเดียวกัน
การอัปเดตความคืบหน้าบ่อยๆ	<ul style="list-style-type: none"> ได้เห็นว่าจะงานให้ไปได้ถึงไหน วางแผนให้งานต่อไป งานไปในทิศทางเดียวกัน 	<ul style="list-style-type: none"> การนัดประชุมเวลาของแต่ละคนในทีม ไม่ตรงกัน 	<ul style="list-style-type: none"> การนัดวันและเวลาในการประชุมให้มีความหลากหลายและลงตัวกับคนในทีม