Name: Bhavin Chavda

Email: bhavinchavda096@gmail.com

Contact: 9898418818

SDE Test Assignment Report

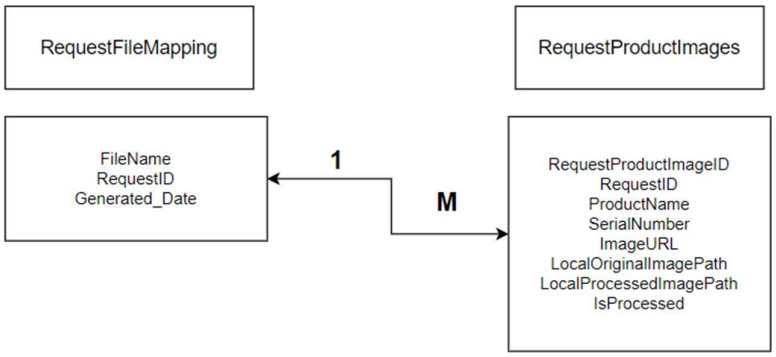
Objective: Build a system to efficiently process image data from CSV files.

As per the Requirement, first our system will take CSV file as input, then validate it and then proceed the images as given by comma separated URLs, stored it and return requestID per CSV file and user can also viewed their original Image and processed Image by respective RequestID

Tools & Technology Used:

- Python
- Flask
- MySQL
- Git
- Pandas
- PIL

System Design for Databases and code logic




```sql
use sde_test;

CREATE TABLE requestfilemapping (
    request_id INT PRIMARY KEY AUTO_INCREMENT,
    filename VARCHAR(255),
    current_date_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    newfilename VARCHAR(255)
);

ALTER TABLE requestfilemapping AUTO_INCREMENT = 5000;
```

```sql
use sde_test;

CREATE TABLE RequestProductImages (
    RequestProductImageID INT AUTO_INCREMENT PRIMARY KEY,
    RequestID INT,
    ProductName VARCHAR(255),
    SerialNumber VARCHAR(255),
    ImageURL VARCHAR(255),
    LocalOriginalImagePath VARCHAR(255),
    LocalProcessedImagePath VARCHAR(255),
    IsProcessed BOOLEAN DEFAULT 0
);
```

RequestFilemapping

There will be one RequestID generated per file

Example:

RequestID	FileName
5001	Book1.CSV
5002	Book2.CSV

Here we will be storing **newfilename** in database by adding CURRENT_TIME_STAMP at the end to avoid duplication as same file can be processed again and RequestID will be new and newfilename will be always new as per time.

```python
@app.route('/createRequestID', methods=['POST'])
def create_request_id():
    file = request.files['file']
    if file and file.filename.endswith('.csv'):
        originalfilename = file.filename

        # Save the file
        current_time = datetime.now().strftime('%Y%m%d_%H%M%S')
        filename, file_extension = os.path.splitext(file.filename)
        new_filename = f"{filename}_{current_time}{file_extension}"

        file.save(os.path.join('files/', new_filename))
        print(new_filename)

        # Save file details to the database
        conn = mysql.connection
        cursor = conn.cursor()

        # Insert the filename into the database
        query = "INSERT INTO requestfilemapping (filename, newfilename) VALUES (%s, %s)"
        cursor.execute(query, (originalfilename, new_filename))

        # Commit the transaction
        conn.commit()

        generated_request_id = cursor.lastrowid
        print(f"Generated Request ID: {generated_request_id}")

        # Close the cursor
        cursor.close()
```

Now there will a master table

RequestProductImages

here is an example how it will store the data

Suppose we have csv file data looks like below

Book1.csv

Serial number	Product	Images
1	A	url1,url2,url3
2	B	url1,url2,url3
3	C	url1,url2,url3

Now master table will looks like this for this CSV file
(Book1.csv → 5001)

RequestProductImageID	RequestID	ProductName	ImageURL	LocalOriginalImagePath	LocalProcessedImagePath
1	5001	A	url1	Generated By System	Generated By System
2	5001	A	url2	Generated By System	Generated By System
3	5001	A	url3	Generated By System	Generated By System
4	5001	B	url1	Generated By System	Generated By System
5	5001	B	url2	Generated By System	Generated By System
6	5001	B	url3	Generated By System	Generated By System
7	5001	C	url1	Generated By System	Generated By System
8	5001	C	url2	Generated By System	Generated By System
9	5001	C	url3	Generated By System	Generated By System

LocalOriginalImagePath Example:
Images\OriginalImages\5000\RequestID_5000_Product_B_Image2.jpg

LocalProcessedImagePath Example

Images\ProcessedImages\5000\RequestID_5000_Product_B_Image2_Processed.jpg

Important Note: System will download Images from URLS stored Original images, Then process these images one by one and then stored processed images and both the relative path will be stored in the table.

Function ProcessImageRequestProductWise :

Function download_image

Function create_processed_image_and_save

```
def ProcessImagesRequestProductWise(new_filename , generated_request_id):

    print("New File name is : " + new_filename)

    df = pd.read_csv('files/'+new_filename)

    # Open Connection
    conn = mysql.connection
    cursor = conn.cursor()

    for _, row in df.iterrows():
        serial_number = row['Serial Number']
        product = row['Product']
        images = row['Images'].split(',')

        cnt = 1
        for image in images:

            save_directory = generate_save_directory(generated_request_id)
            LocalOriginalImagePath = generate_local_original_image_path(generated_request_id, product,cnt)
            # print(f"Full Local Path: images\{generated_request_id}\{LocalOriginalImagePath}")
            save_path = os.path.join(save_directory, LocalOriginalImagePath)
            os.makedirs(save_directory, exist_ok=True)
            cnt=cnt+1
            # print(" SavePath : "+save_path )
            print(type(save_path))

            # Download Image into local to process it
            LocalOriginalImagePathToSave = download_image(image,save_path)

            #Process the Image

            LocalProcessedImagePath = create_processed_image_and_save(save_path,generated_request_id,product,cnt)

            boolIsprocessed = 1
```

```
            LocalProcessedImagePath = create_processed_image_and_save(save_path,generated_request_id,product,cnt)

            boolIsprocessed = 1
            if(LocalProcessedImagePath=="LocalProcessedImagePath"):
                boolIsprocessed = 1



            query = """
            INSERT INTO RequestProductImages (RequestID, ProductName, SerialNumber, ImageURL,
            LocalOriginalImagePath, LocalProcessedImagePath, IsProcessed)
            VALUES (%s, %s, %s, %s, %s, %s, %s)
            """
            cursor.execute(query, (generated_request_id, product, serial_number, image, LocalOriginalImagePathToSave, LocalProc


    conn.commit()
    cursor.close()



    return ""

def generate_save_directory(generated_request_id):
    return f"Images\OriginaImages\{generated_request_id}"

def generate_local_original_image_path(generated_request_id, product,cnt):
    return f"RequestID_{generated_request_id}_Product_{product}_Image{cnt}.jpg"
```

```
def download_image(url, save_path):

    try:
        # Send a GET request to the image URL
        response = requests.get(url, stream=True)
        response.raise_for_status()  # Check for request errors

        # Open the file in binary write mode
        with open(save_path, 'wb') as file:
            # Write the content to the file
            for chunk in response.iter_content(chunk_size=8192):
                file.write(chunk)

        return save_path
        # print(f"Image successfully downloaded and saved to {save_path}")

    except requests.exceptions.RequestException as e:
        return f"Failed to download image. Error: {e}"
```

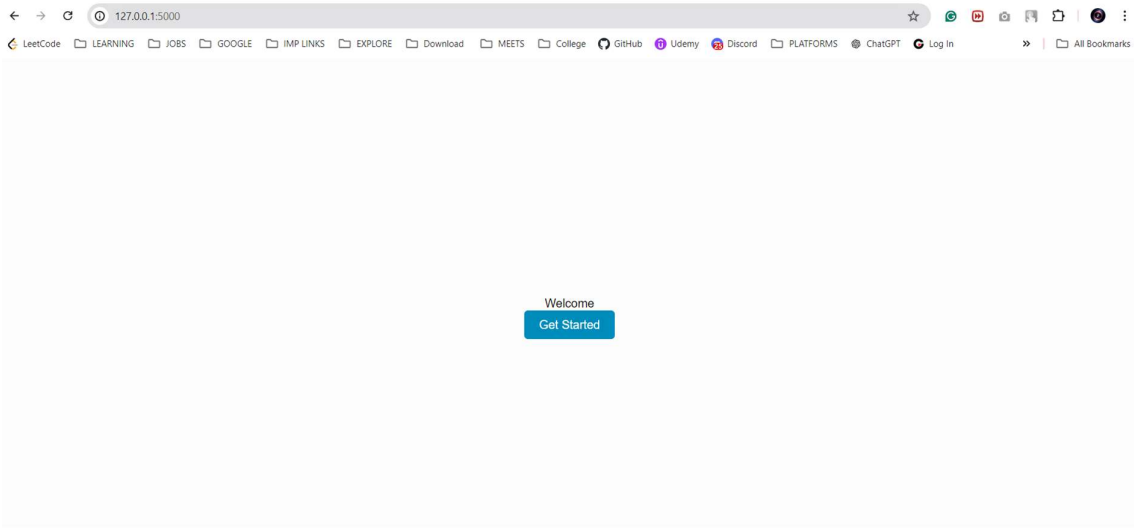
```python
def create_processed_image_and_save(save_path,generated_request_id,product,cnt):

    output_directory = generate_save_processed_directory(generated_request_id)
    os.makedirs(output_directory, exist_ok=True)

    print(type(save_path))

    try:
        file_name, file_extension = os.path.splitext(os.path.basename(save_path))

        new_file_name = f"{file_name}_Processed{file_extension}"

        print(new_file_name)
        # Create the new save path
        new_save_path = os.path.join(output_directory, new_file_name)
        print(new_save_path)
        # Open the original image
        with Image.open(save_path) as img:
            if file_extension.lower() in ['.jpg', '.jpeg']:
                img.save(new_save_path, quality=50, optimize=True)
            else:
                # For other formats, use default saving options
                img.save(new_save_path)

        return new_save_path

    except Exception as e:
        # return f"Failed to process image. Error: {e}"
        return "ERROR OCCURED"

    return ""
```

Now user can view their Process using RequestID and will show the system path (combined by os.current_directory and relative path stored in database)

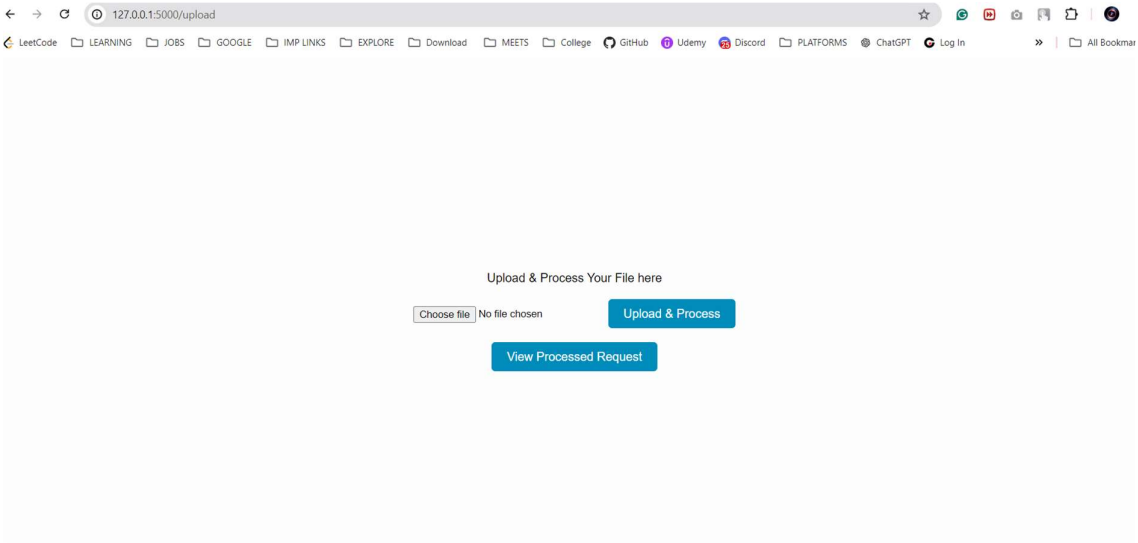
```python
@app.route('/requestDetailpage/<int:request_id>')
def request_page_detail(request_id):

    current_directory = os.path.dirname(os.path.abspath(__file__))

    conn = mysql.connection
    cursor = conn.cursor()

    query = "SELECT LocalOriginalImagePath, LocalProcessedImagePath FROM RequestProductImages WHERE RequestID = %s"
    cursor.execute(query, (request_id,))
    all_requests_images = cursor.fetchall()

    all_requests_images = [
        {
            'LocalOriginalImagePath': os.path.join(current_directory, row[0]),
            'LocalProcessedImagePath': os.path.join(current_directory, row[1])
        }
        for row in all_requests_images
    ]

    conn.commit()
    cursor.close()

    # return f"Request ID is found{request_id}"
    return render_template('requestDetail.html' , request_id=request_id , all_requests_images=all_requests_images)
```

I have also created basic front-end so let’s see how our system works over there step by step

1) Home Page

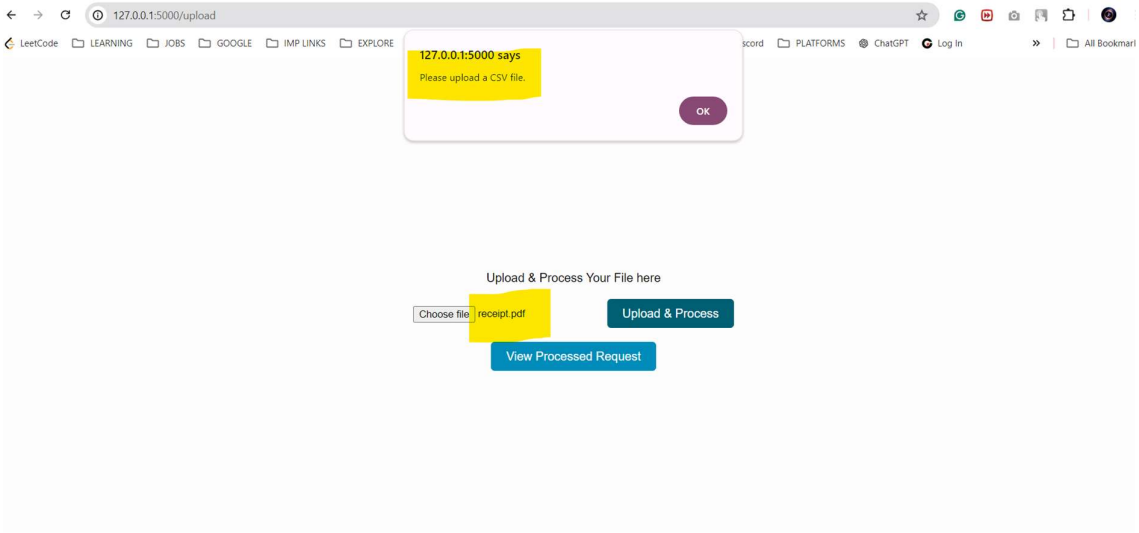


2) Click on Get Started

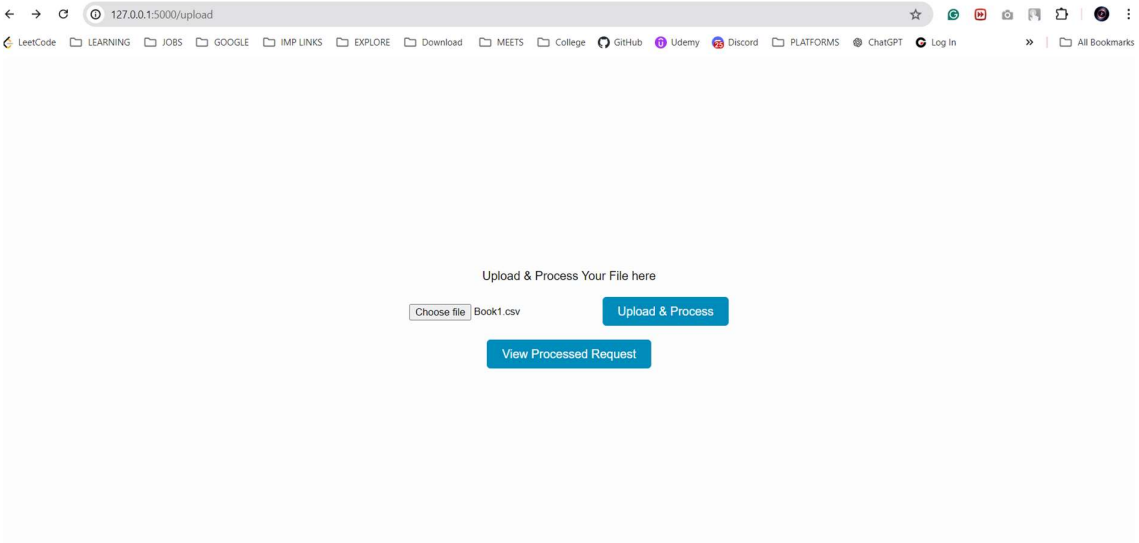


3) Upload the CSV File and Click button for process

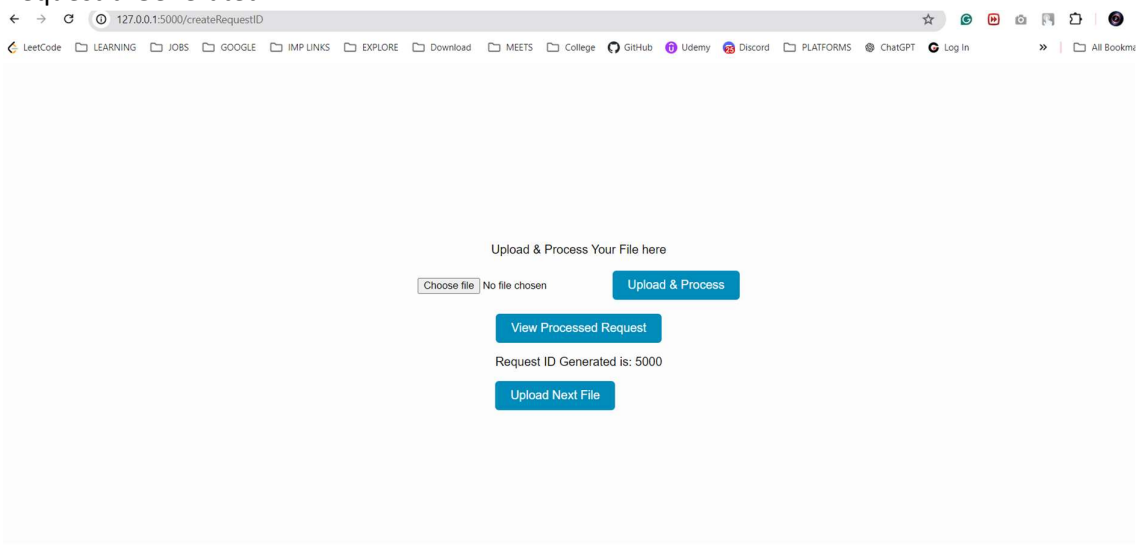
Validate CSV



Upload Correct CSV

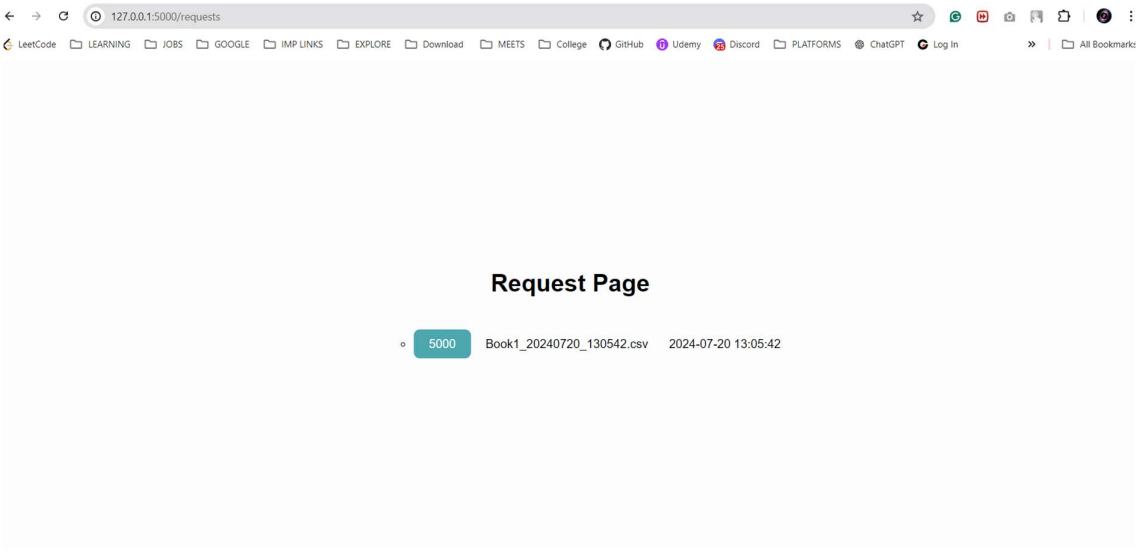


RequestId Generated

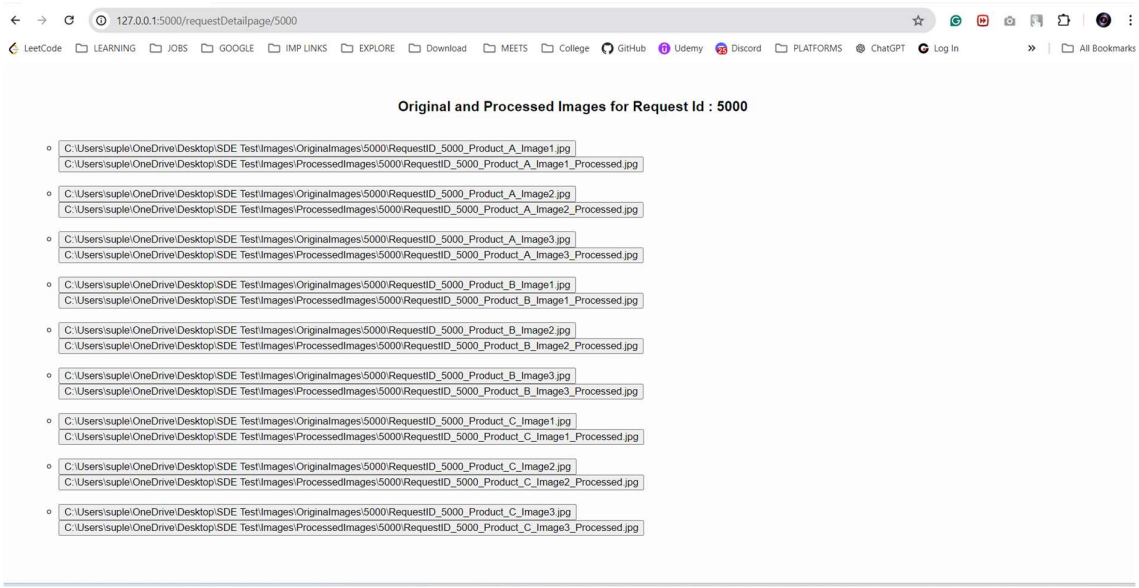


4) Go to Your RequestID to see the Status

Check on Processed Request By Clicking on “View Processed Request”



Click on you RequestID(It will show full system path for both original and processed Image)



Limitations and Future Extension

As per the requested Assignment security modules have not been implemented and there might be some validation issues as well but as per requirement images have been processed properly from given csv file

Also please note that I have merged API with simple front-end best way possible by keeping the deadline in the mind and API return type has been set accordingly and can be changed easily for standalone backend application.

Thank you for giving me this opportunity, it was great experience and fun to create this application for this assignment and feel free to reach out to me for any code issues