Objective:

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

1. **Receive**:- Accept CSV file containing below format

   Input CSV Format:

   Column 1:- Serial Number

   Column 2:- Product Name:- This will be a name of product against which we will store input and output images

   Column 3:- Input Image Urls:- In this column we will have comma separated image urls

S. No.	Product Name	Input Image Urls
1.	SKU1	https://www.public-image-url1.jpg, https://www.public-image-url2.jpg, https://www.public-image-url3.jpg
2.	SKU2	https://www.public-image-url1.jpg, https://www.public-image-url2.jpg, https://www.public-image-url3.jpg

2. **Validate**:- Ensure the CSV data is correctly formatted.
3. **Process**:- Asynchronously process images which means the image will be compressed by 50% of its original quality.
4. **Store**:- Save processed image data and associated product information to a database.

5. **Respond**:-
    a. Initially:- Provide a unique request ID to the user immediately after file submission.
    b. Later:- Offer a separate API to check processing status using the request ID.

Requirements:

Asynchronous APIs:

1. Upload API:- Accepts the CSV, Validate the Formatting and returns a unique request ID.
2. Status API:- Allows users to query processing status with the request ID.
3. Bonus Point:- Create a webhook flow so that after processing all the images you can trigger the webhook endpoint

    Output CSV Format:

    Column 1:- Serial Number

    Column 2:- Product Name:- This will be a name of product against which we will store input and output images.

    Column 3:- Input Image Urls:- In this column we will have comma separated image urls.

    Column 4:- Output Image Urls:- In this column we will have comma separated output image urls in the same sequence as input.

S. No.	Product Name	Input Image Urls	Output Image Urls
1.	SKU1	https://www.public-image-url1.jpg, https://www.public-image-url2.jpg, https://www.public-image-url3.jpg	https://www.public-image-output-url1.jpg, https://www.public-image-output-url2.jpg, https://www.public-image-output-url3.jpg
2.	SKU2	https://www.public-image-url1.jpg, https://www.public-image-url2.jpg, https://www.public-image-url3.jpg	https://www.public-image-output-url1.jpg, https://www.public-image-output-url2.jpg, https://www.public-image-output-url3.jpg

Low-Level Design (LLD):

- Create a detailed technical design document.
- Include a visual diagram of the system (using Draw.io or similar).
- Describe the role and function of each component.

Components to Include:

- **Image Processing Service Interaction**: Integrate with the async image processing service.
- **Webhook Handling**: Process callbacks from the image processing service.

- **Database Interaction**: Store and track the status of each processing request.
- **API Endpoints**:
    - Upload API: Accept CSV files and return a unique request ID.
    - Status API: Check the processing status using the request ID.

Database Schema:

Design a database schema to store product data and track the status of each processing request.

API Documentation: Clear specifications for API Documentation.

Asynchronous Workers Documentation: Description of worker functions.

GitHub Repository: Containing all project code.

Postman Collection: Publicly accessible link for testing the APIs.

Submission Guidelines

- **Tech Stack:-** Use Node Js or Python
- **Databases**:- SQL or NoSQL
- After completing the project submit your assignment using this google form

Note:- Don't use "*SPYNE*" keyword in your git repo