Low-Level Design Document for Processing the Images

**1. System Overview**

This system efficiently processes image data from CSV files. It provides an asynchronous pipeline to:

* Accept CSV files
* Validate data
* Compress images by 50%
* Store processed images in a database
* Provide APIs for upload, status checks, and webhook notifications

**2. System Architecture**

**Components**

1. **Flask API Server**: Handles CSV upload, status checks, and webhook notifications.
2. **Celery Worker**: Processes images asynchronously.
3. **SQL Server Database**: Stores request and image data.
4. **Storage (Local / Cloud)**: Saves compressed images.
5. **Webhook Service**: Notifies users upon completion.

**Flow Diagram :**

A diagram of a software server

AI-generated content may be incorrect.

**3. Database Schema**

**Requests Table**

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| request\_id | UUID | Unique ID for each request |
| status | VARCHAR | Pending, Processing, Completed |
| timestamp | DATETIME | Request timestamp |

**Products Table**

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| product\_id | UUID | Unique ID for each product |
| request\_id | UUID | Foreign key linking to Requests |
| product\_name | VARCHAR | Product name from CSV |

**Images Table**

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| image\_id | UUID | Unique ID for each image |
| product\_id | UUID | Foreign key linking to Products |
| input\_image\_url | TEXT | Original image URL |
| output\_image\_url | TEXT | Compressed image URL |
| status | VARCHAR | Processing / Completed |

**Processing Logs Table**

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| log\_id | UUID | Unique log entry |
| request\_id | UUID | Foreign key linking to Requests |
| image\_id | UUID | Foreign key linking to Images |
| step | VARCHAR | Step in the processing pipeline |
| status | VARCHAR | Success/Failure |
| timestamp | DATETIME | Log timestamp |

**4. API Endpoints**

**1. Upload API (POST /upload)**

* Accepts CSV file
* Validates format
* Returns request\_id

**2. Status API (GET /status/{request\_id})**

* Checks processing status
* Returns Pending, Processing, or Completed

**3. Webhook API (POST /webhook)**

* Triggered when processing is complete
* Sends notification with request details

**5. Asynchronous Image Processing**

* **Task Queue**: Celery in Python
* **Compression**: Python Imaging Library
* **Storage Options**: Local Storage