### **Low-Level Design (LLD) Document**

#### **1. Overview**

This system processes image data from CSV files asynchronously. It includes functionalities for uploading CSV files, processing images, tracking processing status, and handling webhook notifications. The system involves a web API for interacting with users and background tasks for processing.

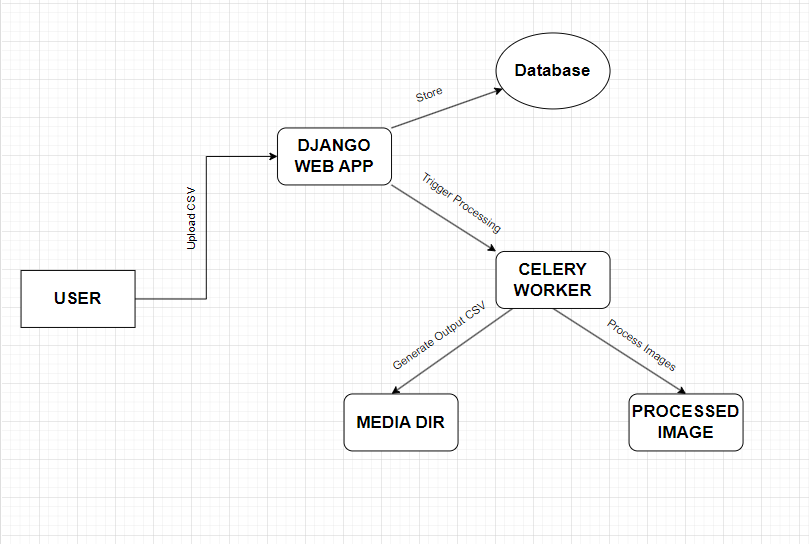
#### **2. Components**

1. **Django Web Application**
2. **Celery Worker for Asynchronous Tasks**
3. **Database**
4. **Webhook Notification System**
5. **CSV File Handling**

#### **3. Architecture Diagram**

Here's a high-level description of the system components and their interactions:

1. **User Interaction**: Users upload CSV files through the Django API.
2. **File Storage**: Uploaded CSV files are stored in the media directory.
3. **Asynchronous Processing**: Celery workers process images as specified in the CSV.
4. **Database**: Stores image processing requests and processed image data.
5. **Webhook Notifications**: Notifications are sent to an external system or endpoint upon completion.



#### **4. Detailed Design**

##### **4.1 Models**

**ImageProcessing Model**

* **Purpose**: Track each image processing request.
* **Fields**:
  + request\_id (UUIDField): Unique identifier for the request.
  + status (CharField): Status of the processing (e.g., 'pending', 'completed').
  + created\_at (DateTimeField): Timestamp of when the request was created.
  + csv\_file\_url (URLField): URL to the generated CSV file (optional).

**ProcessedImage Model**

* **Purpose**: Store information about processed images.
* **Fields**:
  + request (ForeignKey): Links to the ImageProcessing request.
  + product\_name (CharField): Name of the product.
  + original\_url (URLField): URL of the original image.
  + processed\_image\_path (ImageField): Path to the processed image file.

##### **4.2 Serializers**

**ImageProcessingSerializer**

* **Fields**:
  + request\_id
  + status

**ProcessedImageSerializer**

* **Fields**:
  + product\_name
  + original\_url
  + processed\_image\_path

**WebhookSerializer**

* **Fields**:
  + request\_id (UUIDField)
  + status (ChoiceField: 'completed', 'failed')

##### **4.3 Views**

**UploadCSVView**

* **Purpose**: Handle CSV file uploads and initiate image processing.
* **Methods**:
  + POST: Validates and saves the uploaded CSV file, creates an ImageProcessing record, and triggers asynchronous processing.

**StatusView**

* **Purpose**: Provide the status of a processing request.
* **Methods**:
  + GET: Retrieves the status of the request using request\_id.

**WebhookView**

* **Purpose**: Receive notifications from the image processing service.
* **Methods**:
  + POST: Updates the status of the request in the database and optionally generates an output CSV file.

##### **4.4 Asynchronous Tasks**

**process\_images (Celery Task)**

* **Purpose**: Process images asynchronously.
* **Steps**:
  + Read the CSV file.
  + Download and process each image (e.g., compressing images).
  + Save processed images and update the status of the ImageProcessing record.
  + Optionally, notify via a webhook.

##### **4.5 Webhook Notification**

* **Purpose**: Notify an external system when image processing is complete.
* **Endpoint**: POST to http://localhost:8000/webhook/processing\_complete/

**Payload**:  
json  
Copy code  
{

"request\_id": "b6a94e1e-5f2a-4b6e-9f9c-b8bb06527d1f",

"status": "completed"

}

##### **4.6 File Handling**

* **Upload Directory**: Files are stored in the uploads directory under MEDIA\_ROOT.
* **Output Directory**: Generated CSV files are stored in the outputs directory under MEDIA\_ROOT.

#### **4.7. API Endpoints**

1. **Upload CSV**
   * **URL**: /upload/
   * **Method**: POST
   * **Request Body**: Multipart file upload.
   * **Response**: JSON with request\_id.
2. **Check Status**
   * **URL**: /status/<uuid:request\_id>/
   * **Method**: GET
   * **Response**: JSON with request\_id and status.
3. **Webhook Notification**
   * **URL**: /webhook/processing\_complete/
   * **Method**: POST
   * **Request Body**: JSON with request\_id and status.

#### **7. Database Schema**

**ImageProcessing Table**

* request\_id (UUID, Primary Key)
* status (CharField)
* created\_at (DateTimeField)
* csv\_file\_url (URLField, optional)

**ProcessedImage Table**

* id (AutoField, Primary Key)
* request\_id (ForeignKey to ImageProcessing)
* product\_name (CharField)
* original\_url (URLField)
* processed\_image\_path (ImageField)