Low-Level Design (LLD) Document

1. Overview

This document describes the detailed technical design for the image processing system. The system handles CSV uploads, processes images, and exports data. It consists of various components that interact with each other to achieve the desired functionality.

2. Components

2.1 API Endpoints

- **Upload API**
    - **Function:** Accepts CSV files and triggers the image processing workflow.
    - **Endpoints:** `/upload/`
    - **Request Method:** `POST`
    - **Inputs:** CSV file containing image URLs and product details.
    - **Outputs:** Unique request ID and processing status message.
- **Status API**
    - **Function:** Retrieves the status of image processing requests.
    - **Endpoints:** `/status/<request_id>/`
    - **Request Method:** `GET`
    - **Inputs:** Request ID.
    - **Outputs:** Status of the processing request, including details about the number of processed rows and any errors encountered.
- **Export API**
    - **Function:** Exports processed product data to a CSV file.
    - **Endpoints:** `/export/<request_id>/`
    - **Request Method:** `GET`
    - **Inputs:** Request ID.
    - **Outputs:** CSV file containing product data with input and output image URLs

2.2 Image Processing Service Interaction

- **Function:** Processes images asynchronously based on the URLs provided in the CSV file.
- **Interaction:**
    - **Upload API** sends image URLs to the processing service.
    - **Processing Service** performs image resizing and other transformations.
    - **Processing Service** saves processed images and returns URLs.
    - **Update Database** with processed image URLs.

.

2.4 Database Interaction

- **Function:** Stores and tracks the status of each processing request and related product data.
- **Models:**
  - **ImageProcessingRequest**
    - **Fields:** `request_id`, `status`
  - **Product**
    - **Fields:** `serial_number`, `product_name`, `input_image_urls`, `output_image_urls`, `request`
- **Interaction:**
  - **Upload API** creates a new processing request record.
  - **Processing Service** updates the database with processed image URLs.
  - **Status API** retrieves processing request status from the database.
  - **Export API** fetches product data from the database for export.

3. Visual Diagram

Here's a structured outline to create your diagram:

1. **Central Component:**
   - **API Endpoints**
     - **Upload API**
     - **Status API**
     - **Export API**
2. **Connected Components:**
   - **Image Processing Service**
     - Connects to **Upload API** (for receiving image URLs).
     - Optionally, connect to **Webhook Handling** (for status updates).
   - **Database**
     - Connects to **Upload API** (for creating and updating records).
     - Connects to **Status API** (for retrieving request status).
     - Connects to **Export API** (for fetching data to export).
3. **Diagram Layout:**
   - **Top Level:**
     - **Upload API** → **Image Processing Service** and **Database**
   - **Middle Level:**
     - **Status API** → **Database**
     - **Export API** → **Database**

3. Visual Diagram

Here's a structured outline to create your diagram:

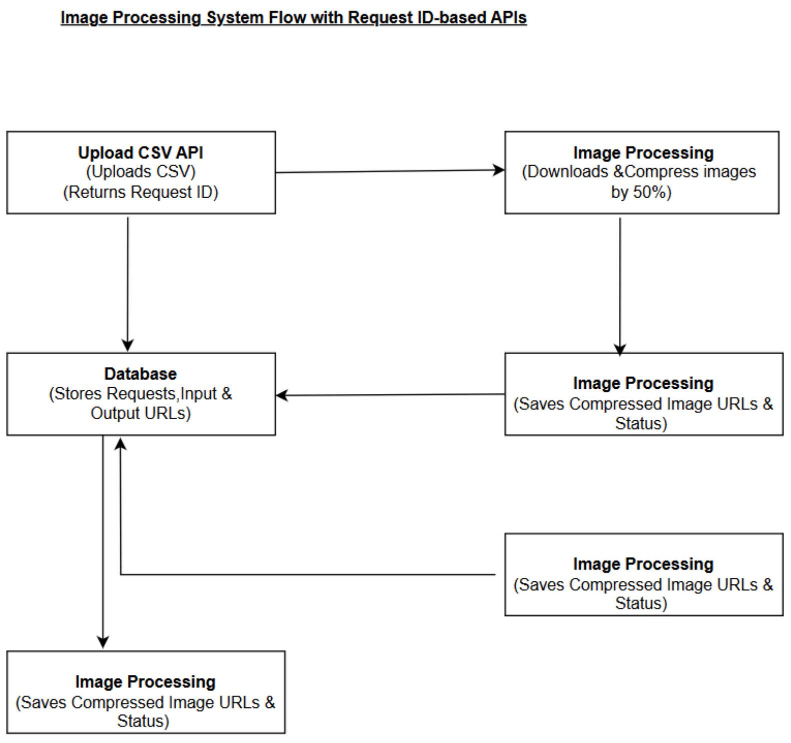
Image Processing System Flow with Request ID-based APIs
Upload CSV API
(Uploads CSV)
(Returns Request ID)
Image Processing
(Downloads &Compress images by 50%)
Database
(Stores Requests,Input & Output URLs)
Image Processing
(Saves Compressed Image URLs & Status)
Image Processing
(Saves Compressed Image URLs & Status)
Image Processing
(Saves Compressed Image URLs & Status)