1. Technical Design Document

1.1 Overview

The system processes image data from a CSV file, compresses the images, and stores the results. It includes asynchronous APIs for uploading the CSV, checking the processing status, and triggering a webhook after processing is complete.

1.2 Architecture Overview

The architecture includes the following components:

- **Flask API Server**: Handles incoming API requests.
- **Celery Worker**: Processes image compression tasks asynchronously.
- **SQLAlchemy Database**: Tracks the status of image processing requests.
- **Redis**: Acts as the message broker for Celery.
- **Image Storage**: Stores the processed images (local storage or cloud-based).

1.3 Components

1. **API Layer (Flask)**:
    - **Upload API**: Accepts CSV files, validates input, and queues a processing task.
    - **Status API**: Checks the processing status of a given request.
    - **Webhook**: Sends a notification when processing is complete.
2. **Asynchronous Processing (Celery + Redis)**:
    - **Celery Task**: Handles downloading, compressing, and saving images.
    - **Redis**: Serves as the broker for managing task queues.
3. **Database Layer (SQLAlchemy + SQLite)**:
    - **ProcessingRequest Model**: Represents a request for processing images, storing input URLs, output URLs, and status.
4. **Image Handling (Pillow)**:
    - **Image Downloading**: Retrieves images from provided URLs.
    - **Image Compression**: Reduces image quality to 50%.
    - **Image Storage**: Saves compressed images locally or to cloud storage.

1.4 Data Flow

1. **Client Request**: The client uploads a CSV file via the Upload API.
2. **Task Queueing**: The Flask server queues a task to process images using Celery.
3. **Image Processing**: The Celery worker downloads, compresses, and stores images.
4. **Status Update**: The processed image URLs and status are updated in the database.
5. **Webhook Notification**: Once processing is complete, a webhook is triggered to notify the client.

1.5 Database Schema

sql
Copy code

```
CREATE TABLE processing_request (
    id TEXT PRIMARY KEY,
    status TEXT NOT NULL,
    product_name TEXT NOT NULL,
    input_urls TEXT NOT NULL,
    output_urls TEXT
);
```

- **id**: Unique identifier for each request.
- **status**: Current processing status (e.g., Pending, Completed).
- **product_name**: Name of the product associated with the images.
- **input_urls**: Comma-separated list of input image URLs.
- **output_urls**: Comma-separated list of output (processed) image URLs

Component Roles and Functions

1. **Flask API Server**:
    - **Upload API**: Accepts CSV files, validates data, and queues tasks.
    - **Status API**: Returns the current status of a processing request.
    - **Webhook Trigger**: Sends a POST request to a webhook endpoint upon task completion.
2. **Celery Worker**:
    - **Task Processing**: Executes image processing tasks asynchronously.
    - **Image Handling**: Downloads images, compresses them using Pillow, and stores the results.
3. **SQLAlchemy Database**:
    - **Request Tracking**: Keeps track of each processing request, including input and output URLs, and processing status.
4. **Redis**:
    - **Message Broker**: Manages task queues for Celery, ensuring tasks are processed in a distributed, asynchronous manner.
5. **Pillow**:
    - **Image Compression**: Reduces the file size of images to optimize storage and bandwidth usage.
6. **Webhook**:
    - **Notification**: Alerts the client once all images in a request are processed, enabling real-time updates.