Design Document: Image Processing System

1. Overview

The purpose of this document is to outline the design of a system that processes image data from CSV files. The system will receive a CSV file containing product information and associated image URLs, compress the images, and store both the original and processed images. Additionally, the system will provide a mechanism to track the status of each processing request.

2. Functional Requirements

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.

3.System Workflow

File Upload:

- The system will accept a CSV file upload via an API.
- Upon submission, the system will generate a unique alphanumeric `request_id` for tracking the request.

Image Processing:

- The system will asynchronously download images from the provided URLs.
- Images will be resized and compressed to 50% of their original quality.
- Both the original and compressed images will be stored in designated directories.

Request Status Tracking:

- The system will track the processing status of each request (e.g., `processing=0`, `success=1`, `Failed=2`).
- Users can query the status of a request using the `request_id`.

Error Handling:

- The system will handle errors such as invalid CSV format, unreachable image URLs, and processing failures. Errors will be logged and reported in the request status.

4. System Components

File Upload API:

- Endpoint to upload the CSV file.
- Returns a unique `request_id`.

Status Check API:

- Endpoint to query the status of a request using `request_id`.

Image Processing Service

Compress images to 50% of original quality using an image processing library (e.g., Jimp)

6. Database Schema

CREATE TABLE csv_processing_requests (

id INT AUTO_INCREMENT PRIMARY KEY,

request_id VARCHAR(255) NOT NULL UNIQUE,

file_name VARCHAR(255) NOT NULL

status TINYINT NOT NULL DEFAULT 0 COMMENT '0=>processing, 1=>success, 2=>failed',

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

```
);

CREATE TABLE csv_products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    request_id VARCHAR(255) NOT NULL,
    product_name VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE csv_product_images (
    id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT NOT NULL,
    product_image_url VARCHAR(255) NOT NULL,
    product_compressed_image_url VARCHAR(255) DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Conclusion

This design document outlines a robust system for processing image data

from CSV files, handling large volumes of data efficiently, and providing mechanisms for status tracking and error handling. The architecture is scalable, secure, and reliable, ensuring smooth operations in production environments.