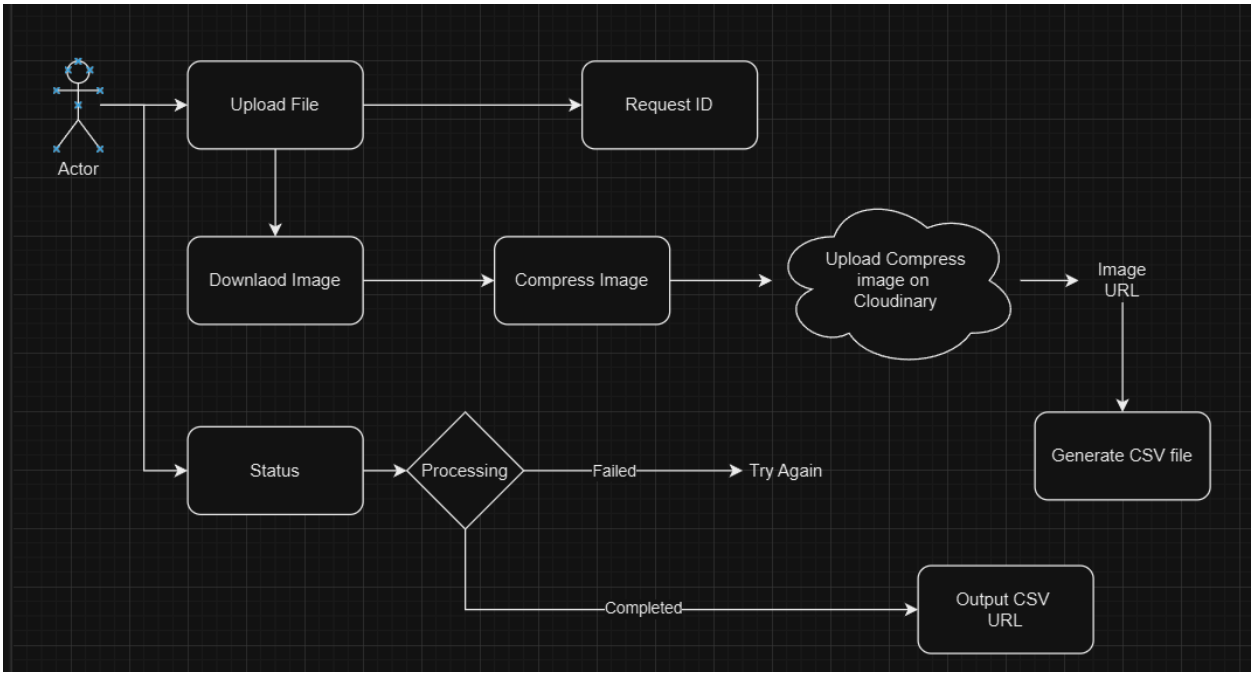
# CSV image Processing Assignment

## 1. User Flow Diagram




## 2. Architecture Overview

Server (Express): Acts as the web server, handles incoming requests and routes them appropriately.

Database (MongoDB): Stores information about image processing tasks, their status, and URLs for the processed images.

Image Processor (Sharp): Compresses images locally before uploading them to the cloud.

Cloud Storage (Cloudinary): Stores processed images and returns publicly accessible URLs.

CSV Parser (csv-parser): Parses CSV files containing image URLs for processing.

Task Progress Tracker: Tracks the progress of image processing jobs and provides status updates to the client.

# 3. Component Description

## 3.1 Server

Technology: Node.js, Express

Responsibility: The server handles HTTP requests and serves APIs for uploading CSV files, processing images, and retrieving task status.

Key Files:

server.js: Entry point of the application, initializes the Express server, and sets up API routes.

upload.js: Handles the image upload and processing logic.

status.js: Handles requests to check the status of image processing tasks.

## 3.2 Database

Technology: MongoDB (via Mongoose)

Responsibility: Stores task metadata, including the request ID, product names, input URLs, output URLs (compressed image URLs), task status, and timestamps.

Key Model:

ProcessingRequest: A schema representing the image processing request. It stores the request ID, product name, input and output URLs, status, and a link to the CSV containing compressed image URLs.

## 3.3 Image Processor

Technology: Sharp, Axios, Cloudinary SDK

Responsibility: Handles image compression using the Sharp library. The images are fetched using Axios, compressed locally, and uploaded to Cloudinary. The URLs of the compressed images are stored and included in the output CSV.

Key Functions:

compressImage: Compresses and resizes an image.

uploadToCloud: Uploads the compressed image to Cloudinary and returns the public URL.

## 3.4 CSV Processing

Technology: csv-parser

Responsibility: Parses the CSV file containing image URLs, cleans the image URLs, and initiates the image processing workflow.

Key Functionality:

generateOutputCSV: Creates the final output CSV containing the original image URLs and the corresponding compressed URLs.

### 3.5 Task Progress Tracker

Technology: JavaScript (via Express Middleware)

Responsibility: Tracks the progress of each image processing task and stores the current status (in-progress, completed, etc.) in memory. It also updates the database after the processing completes.

# 4. API Endpoints

### 4.1 /upload

Method: POST

Description: This API allows the user to upload a CSV file containing image URLs for processing. The system responds with a requestId, which can be used to track the progress of the request.

Input: CSV file containing SNO, Product Name, and Input Image Urls.

Output: JSON response containing the requestId.

### 4.2 /status/

Method: GET

Description: Returns the status of the image processing task. If completed, it provides URLs to the compressed images and the link to download the output CSV.

Input: requestId as a path parameter.

Output: JSON response with status, outputImageUrls, and outputCSVUrl (if applicable).

## 5. Sequence of Operations

### 5.1 CSV Upload & Image Processing

The user uploads a CSV file containing image URLs via the /upload endpoint.

The server processes the CSV file and extracts the image URLs.

A new processing request entry is created in MongoDB with the status set to "pending".

The image processing workflow is started asynchronously:

Each image is compressed using Sharp.

The compressed image is uploaded to Cloudinary.

The output CSV containing the URLs of compressed images is generated.

The MongoDB record is updated with the status "completed" and the link to the output CSV.

The response to the client contains the requestId.

### 5.2 Status Check

The client can query the /status/:requestId endpoint to check the status of the request.

If the request is completed, the endpoint returns the URLs of the compressed images and the link to download the output CSV.

## 6. Error Handling

### 6.1 CSV Validation

If the uploaded CSV file does not follow the expected structure, the server returns a 400 error with a relevant message.

### 6.2 Image Processing Failures

If an error occurs during image processing (e.g., network issues, invalid image URLs), the task status is updated in MongoDB with the error message.

### 6.3 File Handling

Uploaded files are removed from the uploads/ directory after processing.

Compressed images are removed from the local compressed_images/ directory after being uploaded to Cloudinary.