Asynchronous Processing Documentation

The image processing workflow is **asynchronous**, meaning tasks run in the background while the API remains responsive. This is achieved using **background workers** that handle image processing without blocking API requests.

How Asynchronous Processing Works

1.Request Creation & CSV Upload

- The user **uploads a CSV file** via the /upload/ API.
- A new **Request** document is created in the database with:
  - **status = "PENDING"**
  - A **unique request ID (requestId)**
- For each row in the CSV file, a **Product document** is created with:
  - The associated **requestId**
  - **inputImageUrls** extracted from the CSV
  - **outputImageUrls = []** (empty initially)

---

2. Background Worker Starts Processing

- The user triggers processing via /process/<requestId>/ API.
- The system updates the **Request status to "PROCESSING"**.
- A **worker function (process_images()) runs asynchronously** and:
  - Fetches all products linked to the requestId.
  - Processes each image from inputImageUrls.
  - Generates corresponding outputImageUrls.
  - Updates the database.

---

3. Image Processing Details

Each input image undergoes:
Fetching – Downloading the image via requests.get().
Processing – Simulating processing by appending -output to the image URL.
Saving Results – Storing processed URLs in outputImageUrls.

Example Processing
Before:

{
"inputImageUrls":["https://images.unsplash.com/photo-1521747116042-5a810fda9664"]
}

After:

{
"outputImageUrls":
["https://images.unsplash.com/photo-1521747116042-5a810fda9664-output"]
}

---

4. Status Update & Completion

- Once all images are processed, the **Request status updates to "COMPLETED"**.
- If any errors occur, the status is set to "FAILED".
- The user can check status via /status/<requestId>/ API.

Final Response Example

{
  "requestId": "123e4567-e89b-12d3-a456-426614174000",
  "status": "COMPLETED"
}

Worker Functions Overview

1. process_images(requestId)

- **Purpose:** Fetches products linked to a request and processes images.
- **Runs Asynchronously** to prevent blocking API responses.
- **Updates Database** once processing is done.

2. compress_image(image_url)

- **Purpose:** Simulates image compression by modifying the input URL.
- **Returns a new output URL.**

3. webhook_update(requestId, status)

- **Purpose:** Notifies external systems when processing completes.