Overview

This application is a Node.js-based server using Express, designed for uploading CSV files containing product information, processing images associated with products, and generating compressed image URLs using Imgur. The application handles file uploads, image processing, error handling, and status tracking for each request.

Base URL

`http://localhost:3000`

Endpoints

1. Upload CSV File

- **Endpoint**: `/upload`
- **Method**: `POST`
- **Description**: Uploads a CSV file, validates its content, stores products in the database, and processes images.
- **Headers**:
  - `Content-Type: multipart/form-data`
- **Parameters**:
  - `file` (form-data): The CSV file containing product data.
- **Response**:
  - **200 OK**: `{"requestId": "unique-request-id"}`
  - **400 Bad Request**: `{"error": "Invalid CSV headers."}`
  - **500 Internal Server Error**: `{"error": "Error processing images."}`

2. Get Processing Status

- **Endpoint**: `/status/:requestId`
- **Method**: `GET`
- **Description**: Retrieves the status of the products for a given request ID.
- **Parameters**:
  - `requestId` (URL parameter): The unique identifier of the upload request.
- **Response**:
  - **200 OK**: `[{ "serialNumber": "123", "productName": "Sample Product", "status": "completed" }]`
  - **500 Internal Server Error**: `{"error": "Error fetching status information"}`

Asynchronous Workers Documentation

Image Processing Worker Function

Overview

This function handles the processing of images by downloading, compressing, and uploading them to Imgur. It operates asynchronously, allowing for non-blocking execution and improving performance.

Function: `processImages(products, requestId)`

- **Input**:
  - `products`: Array of product objects containing input image URLs.
  - `requestId`: Unique identifier for the request, used for status tracking.
- **Steps**:
  - Iterates through each product's image URLs.
  - Downloads images using `node-fetch`.
  - Validates image content type.
  - Compresses images using `sharp`.
  - Uploads images to Imgur using Axios and stores output URLs.
  - Updates the product status in the database.
  - Generates an output CSV with compressed image URLs if all processes succeed.
- **Error Handling**:
  - Errors during any image processing step will cancel the process and update the product status to "cancelled".

Helper Functions

- **`validateCSV(row)`**: Validates CSV rows against required headers and checks URL format.
- **`uploadToImgur(imageBuffer)`**: Uploads image buffers to Imgur and returns the URL.
- **`generateOutputCSV(data)`**: Creates an output CSV with the processed image URLs.

GitHub Repository

Repository Name: Image Processing Application

- **Repository Link**: GitHub Repository
  Contents:

  1. `models/product.js`: Mongoose schema for products.
  2. `index.js`: Main server file with Express setup and route definitions.
  3. `uploads/`: Directory for storing temporary file uploads.
  4. `compressed-images/`: Directory for output CSV files with compressed image URLs.
- **Setup**:
  1. Clone the repository.
  2. Run `npm install` to install dependencies.
  3. Set up environment variables for MongoDB and Imgur credentials.
  4. Start the server with `node server.js`.

How to Test

1. **Upload CSV**:
   - Choose the `/upload` endpoint.
   - Attach a CSV file in the `file` field.
   - Send the request and observe the response for `requestId`.
2. **Check Status**:
   - Use the `/status/:requestId` endpoint.
   - Replace `:requestId` with the request ID obtained from the upload response.
   - Send the request to check processing status.