Helper: Database Connection

File: `helper/databaseConnection.js`

Function: `connectDB`

Description: Establishes a connection to the MongoDB database. This function connects to MongoDB using the connection string provided in the environment variables and handles any connection errors.

Steps:

1. **Connect to MongoDB:**
    - Use `mongoose.connect` with the connection string from `process.env.MONGO_URL`.
    - Log a success message if the connection is successful.
    - Log an error message and terminate the process if the connection fails.

Helper: Firebase Configuration

File: `helper/firebase.js`

Function: Firebase Initialization

Description: Initializes Firebase with credentials and configures the storage bucket. This setup is used for handling Firebase storage operations.

Steps:

1. **Initialize Firebase:**
    - Import and configure Firebase with service account credentials and storage bucket URL.
    - Initialize the Firebase app and set up the storage bucket.
2. **Export Bucket:**
    - Export the configured Firebase storage bucket for use in other parts of the application.

Usage:Helper: Image Compressor

File: `helper/imageCompressor.js`

Function: `processImage` and `processImageToWebP`

Description: Handles image processing tasks including downloading, compressing, and saving images in various formats.

Functions:

1. **`processImage(url, request_id, originalName)`**
    - **Description:** Downloads an image, compresses it based on its format, and saves it to the output directory.
    - **Steps:**
        - Create the output directory.
        - Download the image from the provided URL.
        - Compress the image according to its format using Sharp.
        - Save the processed image to the output directory.
        - Return the URL of the processed image.
2. **`processImageToWebP(url, request_id, originalName)`**
    - **Description:** Downloads an image, converts it to WebP format, and saves it to the output directory.
    - **Steps:**
        - Create the output directory.
        - Download the image from the provided URL.
        - Convert the image to WebP format using Sharp.
        - Save the WebP image to the output directory.
        - Return the URL of the WebP image.

Helper: Scheduler

File: `helper/scheduler.js`

Function: `process image` and `startScheduler`

Description: Manages image processing tasks using the Agenda library for job scheduling.

Functions:

1. **`process image` Job**
    - **Description:** Job definition for processing images.
    - **Steps:**
        - Retrieve image URLs and product details from the job data.
        - Process each image and save it in both standard and WebP formats.
        - Save the processed image URLs to the `Product` collection.
        - Update the request status to `completed` if all images are processed.
2. **`startScheduler`**
    - **Description:** Starts the Agenda scheduler.
    - **Steps:**
        - Initialize and start the Agenda scheduler.
        - Log a message indicating that the scheduler has started.

Helper: Validation

File: `helper/validation.js`

Functions: `validateCsvHeaders` and `validateCsvData`

Description: Validates CSV headers and data format to ensure proper input.

Functions:

1. **`validateCsvHeaders(headers)`**
    - **Description:** Checks if the CSV headers match the expected headers.
    - **Steps:**
        - Compare the provided headers against the expected headers.
        - Return `true` if headers match, otherwise `false`.
2. **`validateCsvData(csv_data)`**
    - **Description:** Validates the format of CSV data rows.
    - **Steps:**
        - Check if each row contains exactly three columns: serial number, product name, and image URLs.
        - Validate that each column contains non-empty strings.
        - Return `true` if all rows are valid, otherwise `false`.