**Dummy JSON API Integration with MongoDB and ETL Process**

This project demonstrates an Express.js application that integrates with the Dummy JSON API to:

1. Fetch and store user data into a **MongoDB** database.
2. Generate **reports** in CSV and PDF format.
3. Implement an **ETL (Extract, Transform, Load)** process to transform raw data into meaningful metrics.
4. Send **email notifications** when specific conditions (e.g., user count exceeding a threshold) are met.

**Table of Contents**

* [Prerequisites](#prerequisites)
* [Installation](#installation)
* [Environment Variables](#environment-variables)
* [Database Setup](#database-setup)
* [Running the Application](#running-the-application)
* [API Endpoints](#api-endpoints)
  + [Fetch Users](#fetch-users)
  + [ETL Process: User Metrics](#etl-process-user-metrics)
  + [Generate Reports (CSV/ PDF)](#generate-reports-csv--pdf)
  + [Check User Count (Email Alerts)](#check-user-count-email-alerts)
* [Technologies Used](#technologies-used)

**Prerequisites**

Before you begin, ensure you have met the following requirements:

* Node.js installed on your local machine
* MongoDB installed locally or have a cloud instance (e.g., MongoDB Atlas)
* A Dummy JSON API account (Dummy JSON is used here)
* An email service for email notifications

**Installation**

1. Clone this repository:

git clone https://github.com/your-repo/dummy-json-api-integration.git

cd dummy-json-api-integration

1. Install all dependencies:

npm install

**Environment Variables**

# MongoDB Connection URI

MONGO\_URI=mongodb://localhost:27017/dummyjsondb

* **MONGO\_URI**: The connection string for your MongoDB instance.
* **EMAIL\_USER** and **EMAIL\_PASS**: Email credentials for sending alerts

**Database Setup**

Ensure you have **MongoDB** installed and running locally or use a cloud service like **MongoDB Atlas**. This application will store user data fetched from the Dummy JSON API into MongoDB.

**Running the Application**

To run the application locally:

**node start**

The server will run on http://localhost:3000.

**API Endpoints**

**Fetch Users**

* **GET** /api/fetch-users

This endpoint fetches user data from the Dummy JSON API and stores it in the MongoDB database.

**Response**:

{

"message": "Users fetched and stored in MongoDB"

}

**ETL Process: User Metrics**

* **GET** /api/users/metrics

This endpoint runs an **ETL process** to extract raw user data, transform it to calculate metrics (like average age and gender distribution), and return the results.

**Response**:

{

"totalUsers": 100,

"avgAge": 32.5,

"genderDistribution": {

"male": 60,

"female": 40

}

}

**Generate Reports (CSV/ PDF)**

**CSV Report**

* **GET** /api/report/csv

Generates and downloads a CSV report of all users stored in MongoDB.

**Response**: CSV file download (users-report.csv).

**PDF Report**

* **GET** /api/report/pdf

Generates and downloads a PDF report of all users stored in MongoDB.

**Response**: PDF file download (users-report.pdf).

**Check User Count (Email Alerts)**

* **GET** /api/check-users

This endpoint checks the current number of users stored in MongoDB. If the user count exceeds a threshold (e.g., 100 users), an **email alert** is sent.

**Response**:

{

"message": "User count checked",

"userCount": 105

}

**Email Notification**: If the number of users exceeds the threshold, an email is sent with the following message:

Subject: User Count Alert

The number of users has exceeded 100. Current count: 105.

**Technologies Used**

* **Node.js**: JavaScript runtime environment
* **Express.js**: Web framework for Node.js
* **MongoDB**: NoSQL database for storing user data
* **Mongoose**: ODM library for MongoDB
* **Axios**: For making HTTP requests to the Dummy JSON API
* **pdfkit**: Library for generating PDF reports
* **json2csv**: Library for generating CSV reports
* **Nodemailer**: For sending email notifications