Health Compass Scraping + AI Training Work Details

Project Overview

Project Name: Health-compass-phase-2

Objective:

The objective of this project is to develop an advanced, scalable system for scraping supplement and ingredient data from multiple online sources and storing this data in a structured database. Additionally, an intelligent, user-friendly AI bot is created to interact with users, providing detailed information, answering queries, and guiding users through the data collected. The bot is designed to offer accurate, real-time responses and assist users in navigating the large datasets of supplements and ingredients, improving their overall user experience.

Technology Stack:

- **Backend**: Node.js, Express.js
- **Database**: MongoDB
- **Bot**: GPT-based AI model (OpenAI GPT-3/4)
- **Deployment**: DigitalOcean

Database Schema Overview

1. DrugsDetails Schema

Purpose:
This schema stores the details of drugs, including their XML data converted into a structured format.

Key Fields:

- **setid**: A unique identifier for each drug entry (required for data integrity).
- **xmlJson**: The actual drug data, stored as an object (e.g., XML data parsed into JSON format).

Additional Features:

- **Timestamps**: Automatically stores the creation and modification times of each document.

Model: DrugsDetails

- Stores detailed information about each drug.
- Used for managing and retrieving drug data collected from various sources.

2. IngredientDetails Schema

Purpose:
This schema tracks ingredients associated with supplements and drugs.

Key Fields:

- **groupName**: The name of the ingredient group (e.g., vitamins, minerals).
- **hits**: An array that stores references or metadata related to ingredient hits, possibly including the number of times the ingredient is referenced or searched for.

Additional Features:

- **Timestamps**: Tracks when an ingredient entry was created or updated.

Model: IngredientDetails

- Helps categorize and track individual ingredients.
- Useful for searching and filtering ingredients across supplements or drugs.

3. SupplementDetails Schema

Purpose:
This schema stores details about various supplements, including their source and associated data.

Key Fields:

- **sourceId**: A unique identifier for the source of the supplement data.
- **data**: An object that stores all relevant data about the supplement, such as ingredients, usage instructions, dosage, etc.

Additional Features:

- **Timestamps**: Automatically records when a supplement entry is created or modified.

Model: SupplementDetails

- Designed for supplement data management.
- Can be used to manage and retrieve information about different supplements and their properties.

API Overview For Scraping

1. fetchAndStoreFactsheets (Ingredient Data Scraping)

Purpose:
This API fetches ingredient data from a third-party factsheet API and stores it in the database. It iterates over all ingredients, fetches additional information for each, and stores it if it doesn't already exist in the database.

Key Features:

- Fetches ingredient data for each ingredient group.
- Skips already existing entries to avoid duplicates.
- Handles API rate limits and retries in case of failures.
- Saves successfully fetched data to the IngredientDetails collection.

Usage:

- **Route**: GET /api/sync-ingredient-details
- **Response**: Success or failure with a count of saved and skipped entries.
- **Error Handling**: Logs errors and handles rate-limiting issues.

2. fetchAndStoreSupplementLabels (Supplement Data Scraping)

Purpose:
This API fetches supplement labels from a third-party API based on the sourceId and stores them in the database. The data is flattened into key-value pairs for easy storage and retrieval.

Key Features:

- Fetches supplement data based on a unique sourceId.
- Flattens nested data into a key-value structure for consistent storage.
- Skips already existing supplements to avoid duplicates.
- Handles API failures and retries with a delay to prevent flooding the API with requests.

- Saves data into the SupplementDetails collection.

Usage:

- **Route**: GET /api/sync-supplement-details
- **Response**: Success or failure with a count of saved and skipped supplements.
- **Error Handling**: Logs errors related to data fetching and retries.

3. fetchAndStoreDrugXML (Drug Data Scraping)

Purpose:
This API scrapes drug data from the FDA API in XML format, converts it to JSON, and stores it in the database. It processes data in batches and handles pagination.

Key Features:

- Fetches drug labels in batches (pagination) from the FDA's OpenFDA API.
- Converts XML data into a structured JSON format for easier processing.
- Checks if drug records already exist in the database to prevent duplicates.
- Saves new drug records to the DrugsDetails collection.
- Handles pagination, fetching data in increments of 1000 records at a time.

Usage:

- **Route**: GET /api/sync-drug-details
- **Response**: Success or failure with a count of processed and skipped drug records.
- **Error Handling**: Handles failed fetch attempts and logs errors accordingly.

4. fetchAndUpdateDrugClassification (Drug Classification Update)

Purpose:
This API updates the drug records with their respective classifications using the RxNorm API. It checks if the classification is already available and fetches it only if missing.

Key Features:

- Fetches drug classification data using RxNorm IDs.
- Updates drug entries in the database with classification details if not already present.
- Handles errors and failures gracefully, ensuring that only valid and up-to-date data is saved.
- Adds a 2-second delay between requests to avoid overwhelming the external API.

Usage:

- **Route**: GET /api/update-sync-drug-details
- **Response**: Success or failure with counts of processed, updated, and failed drug records.
- **Error Handling**: Catches errors during RxNorm API calls and skips any failed records.

API Overview For AI Bot

1. handleProductQuery (Bot Handler for User Queries)

Purpose:
This API processes user messages, validates them, and retrieves a response from the GPT-powered bot. The bot uses structured data (e.g., supplements, ingredients, drugs) to provide accurate answers to user queries.

Key Features:

- **Message Validation**: Ensures the user message is non-empty and valid.
- **GPT-Powered Response**: Sends the user's query to the GPT model to generate a response.
- **Error Handling**: Returns specific error messages if the bot cannot generate a valid reply or if any error occurs.
- **Response Structure**: Responds with the original query and the bot-generated reply.

Usage:

- **Route**: POST api/bot/ask
- **Response**: Returns the bot's reply or an error message if something goes wrong.

If you have any query regarding this project then kindly contact me through upwork.

Warm Regards,
Mr.Nikunj Goyani.
(Full-Stack Developer)

====XXXXX====