**Provide scripts or instructions for setting up OpenSearch and indexing the data.**

**1. Docker Compose Configuration**

The Docker Compose file (docker-compose.yml) is used to set up OpenSearch and OpenSearch Dashboards. It defines two services:

* **OpenSearch Node**: This service runs the OpenSearch server, allowing you to store and search your data. It runs on port 9200.
* **OpenSearch Dashboards**: This service provides a user interface for managing and visualizing your OpenSearch data. It runs on port 5601.

**2. Flask Application**

The Flask application provides a RESTful API to interact with your OpenSearch instance. Here's a breakdown of the key components:

* **Imports**:
  + Flask: The web framework for creating the application.
  + OpenSearch: The client library to interact with OpenSearch.
  + CORS: A middleware to enable Cross-Origin Resource Sharing, allowing your frontend (running on a different port) to communicate with the Flask backend.
* **OpenSearch Client Setup**: The client is configured to connect to the OpenSearch instance running locally on port 9200.
* **Routes**:
  + **/search**: This endpoint allows users to search for recipes using keywords. It accepts query parameters for the search term, pagination (page number and size), and constructs a multi-match query for the specified fields (title, ingredients, categories).
  + **/filter**: This endpoint allows users to filter recipes based on various criteria, such as title, category, calories, and rating. It constructs a query with the specified filters and pagination.
  + **/categories**: This endpoint retrieves unique categories from the indexed recipes using an aggregation query.

**3. Data Indexing**

The data indexing script uses the Pandas library to read a CSV file containing recipe data. The script processes the data and formats it for bulk indexing into OpenSearch. It:

* Cleans the data by removing duplicates.
* Breaks the data into chunks (to manage memory usage) and sends it to OpenSearch using the bulk indexing API.

**Running the Setup**

**Step 1: Start OpenSearch**

1. **Create the Docker Compose File**: Save the Docker Compose configuration to a file named docker-compose.yml.
2. **Start the Services**: In your terminal, navigate to the directory containing the docker-compose.yml file and run:

bash

Copy code

docker-compose up

This command will start OpenSearch and OpenSearch Dashboards. You can access the OpenSearch Dashboards UI at http://localhost:5601.

**Step 2: Set Up the Flask Application**

1. **Create the Flask Application File**: Save the Flask code to a file named app.py (or any name you prefer).
2. **Run the Flask Application**: In a new terminal window, navigate to the directory containing your app.py file and run:

bash

Copy code

python app.py

This will start the Flask server, and you can access the API at http://localhost:5000.

**Step 3: Index Your Data**

1. **Prepare Your Dataset**: Ensure you have the CSV file containing your recipes ready (named epi\_r.csv).
2. **Run the Indexing Script**: In the Flask application, include the data indexing logic, or create a separate script to read the CSV and index the data. When you run this script, it will send the data to OpenSearch.

**Step 4: Test Your API**

You can use tools like Postman or your web browser to test the endpoints:

* **Search Recipes**: Send a GET request to http://localhost:5000/search?q=your\_search\_query.
* **Filter Recipes**: Send a GET request to http://localhost:5000/filter?q=your\_search\_query&category=your\_category.
* **Get Categories**: Send a GET request to http://localhost:5000/categories.

**Conclusion**

By following the steps above, you can set up OpenSearch using Docker, create a Flask application to interact with OpenSearch, and index your recipe dataset. This setup provides a solid foundation for developing your EpiRecipes Search Platform project.