Spam Search API Documentation

This project provides a RESTful API for managing users and reporting spam activity. With this API, users can create accounts, log in, report spam, and search spam reports by name or phone number.

Prerequisites

To set up and run this project, make sure you have the following tools installed:

- Docker
- Docker Compose
- Go (version 1.18 or higher)

Project Setup

Step 1: Build and Start the Application with Docker

1. Use Docker Compose to set up and run the application and its dependencies (e.g., a PostgreSQL database).

   ```bash
   docker-compose up --build
   ```

2. This command will:
   - Build and start the API container.
   - Set up the necessary services (e.g., PostgreSQL database).

3. Once the containers are running, the API will be available at `http://localhost:8080`.

Step 2: Run the Go Application (Development Mode)

For development, you can run the application directly using Go. This setup allows hot-reloading and easier debugging:

```bash
go run main.go -e development
```

API Endpoints

Base URL

All API endpoints are accessible under:

http://localhost:8080/api/v1

User Endpoints

1. Create a New User

- URL: `/users/create`
- Method: `POST`
- Request Body:

```json
{
  "name": "John Doe",
```

```
    "email": "john.doe@example.com",
    "phone_number": "+123456789",
    "password": "yourpassword"
  }
  ```
- Response:
  - Success: 200 OK
    ```json
    { "message": "User created successfully" }
    ```
  - Error: 409 Conflict if the user already exists or other error codes for validation issues.

2. User Login

- URL: `/users/login`
- Method: `POST`
- Request Body:
  ```json
  {
    "phone_number": "+123456789",
    "password": "yourpassword"
  }
  ```
- Response:
  - Success: 200 OK
    ```json
    { "message": "Login successful", "response": { "access_token": "token" } }
    ```
```
```

  - Error: 401 Unauthorized for invalid credentials.

Spam Report Endpoints (Authenticated)

> Note: For all spam report endpoints, include an `Authorization` header with a Bearer token obtained during login.

1. Report Spam

- URL: `/spam/report`
- Method: `POST`
- Authorization: Required (`Bearer <access_token>`)
- Request Body:

  ```json
  {
    "name": "Spam Caller",
    "phone_number": "+123456789",
    "spam_likelihood": 75
  }
  ```

- Response:
  - Success: 200 OK

    ```json
    { "message": "Spam report created successfully" }
    ```

  - Error: Validation errors as applicable.

2. Search Spam by Name

- URL: `/spam/search/name`
- Method: `GET`
- Authorization: Required (`Bearer <access_token>`)
- Query Parameters:
  - name: The name to search for (e.g., `?name=John`)
- Response:
  - Success: 200 OK with JSON containing the search results.
  - Error: Validation or token errors as applicable.

3. Search Spam by Phone Number

- URL: `/spam/search/phone`
- Method: `GET`
- Authorization: Required (`Bearer <access_token>`)
- Query Parameters:
  - phone_number: The phone number to search for (e.g., `?phone_number=+123456789`)
- Response:
  - Success: 200 OK with JSON containing the search results.
  - Error: Validation or token errors as applicable.

Code Structure Overview

- Controllers:
  - `UserController`: Manages user-related actions, like creating users and logging in.
  - `SpamReportsController`: Manages spam reporting and search functions.

- Middleware:

  - `AuthTokenMiddleware`: Ensures requests to `/spam` endpoints are authenticated.

- Routes: Defined in `main.go`, grouped under `/api/v1` with `/users` and `/spam` routes for user and spam functionalities.

Example Usage

Below are some example `curl` commands to interact with the API.

1. **Create a New User**

   ```bash
   curl -X POST http://localhost:8080/api/v1/users/create -H "Content-Type: application/json" -d '{"name": "John Doe", "email": "john@example.com", "phone_number": "+123456789", "password": "password123"}'
   ```

2. **User Login to Obtain Token**

   ```bash
   curl -X POST http://localhost:8080/api/v1/users/login -H "Content-Type: application/json" -d '{"phone_number": "+123456789", "password": "password123"}'
   ```

3. **Report Spam (Authenticated)**

   ```bash
   curl -X POST http://localhost:8080/api/v1/spam/report -H "Authorization: Bearer <access_token>" -H "Content-Type: application/json" -d '{"name": "Spam Caller", "phone_number": "+123456789", "spam_likelihood":
   ```

80}'

```

4. **Search Spam by Name (Authenticated)**

```bash
curl -X GET "http://localhost:8080/api/v1/spam/search/name?name=John" -H "Authorization: Bearer <access_token>"
```

This should help clarify the setup and usage for building, running, and interacting with the Spam Search API. Let me know if you need further help!
```