SpamLookupify

1. Overview

This project is a Django-based application that allows users to manage their contacts, report spam numbers, and search for contacts based on names and phone numbers. It features user registration, login, and a global search functionality.

2. Features

- User registration and login
- Manage contacts (create, read, update, delete)
- Report spam contacts
- Search contacts by name or phone number
- View contact details with spam likelihood
- rate limit available (Uncomment throttle classes in views.py to add rate limit)

3. Requirements

- Python 3.8 or higher
- Django 3.2 or higher
- Django REST Framework
- PostgreSQL or any other supported database

4. Getting Started

Step 1: Download and Extract

1. Download the project zip file from the provided source.
2. Extract the contents to your desired location.

Step 2: Setup the Environment

A. Create a virtual environment

Command: `python -m venv .venv`

B. Activate the virtual environment:

On Windows:

Command: `.venv\Scripts\activate`

On macOS/Linux:

Command: `source .venv/bin/activate`

C. Install the required packages:

Command: `pip install -r requirements.txt`

Step 3: Configure Database

1. If you wish to use the default SQLite database, you can leave the database settings in `settings.py` as is.
2. To use a different database (e.g., PostgreSQL, MySQL), update the database settings in `settings.py` to match your database configuration.

   check postgres db config

   Run the following commands to set up the database:

   Command: `python manage.py makemigrations`
   Command: `python manage.py migrate`

Step 3.1: Generate Test Data

1. populate_db.py script in SpamLookupify/management/commands helps to populate test data

   Command: `python manage.py populate_db`

Step 4: Create a Superuser (Optional)

To access the admin panel:

Command: `python manage.py createsuperuser`

Step 5: Run the Development Server

Start the server:

Command: `python manage.py runserver`

Open your browser and navigate to `http://127.0.0.1:8000/` to access the application.

Step 6: Testing

1. Through Test file

```
Command: python manage.py test
```

2. Through Postman:

Important: 1. for first time setup please hit login api after register api to generate csrftoken and sessionid

API Endpoints (Test using Postman):

User Authentication

- **Register User**
  - **URL:** `/api/register/`
  - **Method:** `POST`

  Body:

```
{
  "username": "string",
  "password": "string",
  "phone_number": "string",
  "email": "string"
}
```

- **Login User**
  - **URL:** `/api/login/`
  - **Method:** `POST`

  Body:

```
{
  "username": "string",
  "password": "string"
}
```

- **Logout User**
  - **URL:** `/api/logout/`
  - **Method:** `POST`

Contacts Management

- **List/Create Contacts**
  - **URL:** `/api/contacts/`

- **Method:** `GET` / `POST`

Body (for POST):

```
{
  "name": "string",
  "phone_number": "string"
}
```

- **Retrieve/Update/Delete Contact**
  - **URL:** `/api/contacts/<id>/`
  - **Method:** `GET` / `PUT` / `DELETE`

Spam Reporting

- **Report Spam**
  - **URL:** `/api/report-spam/`
  - **Method:** `POST`

Body:

```
{
  "phone_number": "string"
}
```

Search Functionality

- **Search Contacts by Name**
  - **URL:** `/api/search/`
  - **Method:** `GET`
  - **Query Parameters:**
    - `query`: string

- **Search Contacts by Phone Number**
  - **URL:** `/api/search-phone/`
  - **Method:** `GET`
  - **Query Parameters:**
    - `phone_number`: string

Postgres DB config

1. Install Prerequisites

Install PostgreSQL:
Ensure you have PostgreSQL installed on your machine.

Install psycopg2:
Use the command below to install the PostgreSQL adapter:
`Command: pip install psycopg2-binary`

2. Update settings.py

update the DATABASES setting in settings.py as follows:

`Command: cd Spamlookupify_project and open settings.py`

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'your_database_name',  # Replace with your database name
        'USER': 'your_database_user',  # Replace with your database user
        'PASSWORD': 'your_database_password',  # Replace with your database password
        'HOST': 'localhost',  # Set to empty string for localhost
        'PORT': '',  # Set to empty string for default
    }
}
```

3. Create the PostgreSQL Database

You need to create the database in PostgreSQL if you haven't already. You can do this using the PostgreSQL command-line tool or a GUI like pgAdmin. Here's how to do it using the command line:

```
# Log into PostgreSQL (default user is usually 'postgres')
```

```
      Command: psql -U postgres

# Create the database

      Command: CREATE DATABASE your_database_name;

# Exit psql

      Command: \q
```