API Documentation for StoreApp

This document provides a detailed explanation of the API, its endpoints, the technologies used, and how to interact with it.

Overview

The `StoreApp` is built using Django and Django Rest Framework (DRF). It allows users to fetch store data based on geographical coordinates. The application also supports bulk CSV uploads using Celery and Redis for background task processing. SQLite is used as the database for simplicity.

Technologies Used

- **Core Language** : Python
- **Framework:** Django, Django Rest Framework (DRF)
- **Task Queue:** Celery
- **Message Broker/Worker:** Redis
- **Database:** SQLite

API Endpoints

1. Fetch Stores by Location

- **Endpoint:** `/storeapp/store/`
- **Method:** `GET`
- **Description:** Retrieves a list of stores near the specified latitude and longitude.

Query Parameters

Parameter	Type	Description	Required
`Latitude`	float	Latitude of the user's location	Yes
`Longitude`	float	Longitude of the user's location	Yes

Example Request

GET http://127.0.0.1:8000/storeapp/store/?latitude=53.25039&longitude=10.41092

Example Response


```
{
   "available_stores": [
      {
         "id": 16,
         "latitude": 53.25039,
         "longitude": 10.41092,
         "availability_radius": 4.0,
         "open_hour": "09:00:00",
         "close_hour": "20:00:00",
         "rating": 4.4
      },
      {
         "id": 6959,
         "latitude": 53.2478611,
         "longitude": 10.4032051,
         "availability_radius": 3.0,
         "open_hour": "08:00:00",
```

```
    "close_hour": "23:00:00",
    "rating": 4.0
},
{
    "id": 7620,
    "latitude": 53.2511304,
    "longitude": 10.4165358,
    "availability_radius": 5.0,
    "open_hour": "08:00:00",
    "close_hour": "23:00:00",
    "rating": 4.7
},
{
    "id": 8876,
    "latitude": 53.2479667,
    "longitude": 10.4272847,
    "availability_radius": 2.0,
    "open_hour": "10:00:00",
    "close_hour": "18:00:00",
    "rating": 4.6
},
{
    "id": 9189,
    "latitude": 53.244877,
    "longitude": 10.421371,
    "availability_radius": 3.0,
    "open_hour": "09:00:00",
    "close_hour": "20:00:00",
    "rating": 3.7
```

```
    },
    {
      "id": 9356,
      "latitude": 53.2639,
      "longitude": 10.39996,
      "availability_radius": 3.0,
      "open_hour": "09:00:00",
      "close_hour": "20:00:00",
      "rating": 4.3
    }
  ]
}
```


2. Upload Stores via CSV

- **Source :** https://s3.amazonaws.com/test.jampp.com/dmarasca/takehome.csv
- **Description:** Allows uploading of a CSV file containing store data. The file is processed asynchronously using Celery.

Setup Instructions

1. Install Dependencies

Ensure you have Python, Redis, and SQLite installed. Then install required Python packages:

pip install django djangorestframework celery redis geopy requests

2. Configure Celery

In `settings.py`, configure Celery:

```
CELERY_BROKER_URL = 'redis://127.0.0.1:6379'
CELERY_ACCEPT_CONTENT = ['application/json']
CELERY_RESULT_SERIALIZER = 'json'
CELERY_TASK_SERIALIZER = 'json'
CELERY_TIMEZONE = 'Asia/Kolkata'

CELERY_RESULT_BACKEND = 'django-db'

CELERY_BEAT_SCHEDULER = 'django_celery_beat.schedulers:DatabaseScheduler'
```

Run Redis:

redis-server

Start the Celery worker:

celery -A project_name worker --loglevel=info

2. Run Server

Start the Django development server:

python manage.py runserver