# Introduction

The RESTful Django web application aims to provide access to South Africa’s protected areas, catering to potential investors and environmentalists. Leveraging the World Database on Protected Areas (WDPA), this project facilitates exploration, contribution, and management of protected area data. The WDPA, which is the only global database of protected areas, and it is one of the component databases of the Protected Planet Initiative. Protected Planet is a joint product of UNEP and IUCN, managed by UNEP-WCMC and the IUCN working with governments, communities and collaborating partners. The dataset includes diverse attributes such as location, size, designation type, and management status. By analyzing this dataset, we can identify trends, gaps, and conservation priorities specific to South Africa.

# Application

#### *Prerequisites*

development environment i.e. the operating system and python version

#### Running the Application

logging into the django-admin site i.e. username and password

## Data Model

The project contains two data models, **Location**, with province information and **ProtectedArea** which contains all the protected areas and the relevant information. There is a one-to-many relationship between the Location and ProtectedArea table.

Each protected area is represented by a model with various attributes

* wdpaid (int): The unique identifier for the protected area.
* wdpa\_pid (str): The PID (Protected Area ID) of the protected area.
* name (str): The name of the protected area.
* desig\_eng (str): The English designation of the protected area.
* desig\_type (str): The type of designation for the protected area.
* marine (str): Indicates whether the protected area is marine or not.
* rep\_m\_area (float): The reported marine area of the protected area.
* gis\_m\_area (float): The GIS (Geographic Information System) marine area of the protected area.
* rep\_area (float): The reported area of the protected area.
* gis\_area (float): The GIS area of the protected area.
* status (str): The status of the protected area.
* status\_yr (int): The year of the status of the protected area.
* gov\_type (str): The type of government responsible for the protected area.
* own\_type (str): The type of ownership for the protected area.
* mang\_auth (str): The managing authority of the protected area.
* sub\_loc (Location): The sub-location of the protected area (foreign key).
* parent\_iso3 (str): The ISO3 code of the parent location.

a ‘load and store’ python script to load your data from the .csv file or files and store it in a database, perform the necessary processing and return the JSON documents of your REST endpoints.

## Django REST Framework

## API Endpoints

Users can use the following links to access the APIs

1. **GET** List all protected areas? Provides an overview of all protected areas within South Africa. <http://127.0.0.1:8000/protected-areas/>
2. **GET** List all locations? Provides an overview of all provinces within South Africa. <http://127.0.0.1:8000/locations/>
3. **POST** Add supporting links for the protected areas. This enables users to contribute additional resources (e.g., research papers, images) related to specific protected areas.
4. **GET** Return protected areas that have been designated in each province? Allows modifications to existing protected area information. [http://127.0.0.1:8000/location-protected-areas/<sub\_loc>/](http://127.0.0.1:8000/location-protected-areas/%3csub_loc%3e/)
5. **GET** List of national parks, from largest to smallest? Offers sorted data, aiding statistical analysis or prioritization. <http://127.0.0.1:8000/national-parks/>
6. **POST** Add new location. Expands the database with new protected area entries.

## Testing

how to run the unit tests and the location of the data loading script

# Critical Evaluation