

DEVELOPMENT OF NDATA SCHOOL OF CLIMATE AND EARTH SCIENCES GEOPORTAL

ESTHER MAKWINJA
TASLEEM BAKILI
PEMPHERO KANDOJE
BILLY ERICK KACHINGWE
Thyolo, Malawi

Dissertation submitted to Ndata School of Climate and Earth Sciences of the Malawi University of Science and Technology in partial fulfilment of the requirements for the Bachelor of Science degree in Geo-Information and Earth Observation Science

July, 2021

DISSERTATION APPROVED BY:

SUPERVISOR:

Mr. Steven Gondwe Signature: 

ASSESSOR:

Title, Initials, Name Signature: _____

DISCLAIMER

This document describes work undertaken as part of a programme of study at Ndata School of Climate and Earth Sciences of the Malawi University of Science and Technology. All views and opinions expressed therein remain the sole responsibility of the author(s), and do not necessarily represent those of the School.

COPYRIGHT STATEMENT:

This copy of the dissertation has been supplied on condition that anyone who consults it is understood to recognize that its copyright rests with its author and that no quotation from the dissertation and no information derived from it may be published without acknowledgement.

ABSTRACT

The purpose of this study was to come with a web-based platform for Ndata School of Climate and Earth Science (NSCES) at the Malawi University of Science and Technology (MUST) and determine NSCES's current spatial data sharing patterns. Ndata school was chosen from a number of faculty departments because most activities within the faculty involve use of spatial data of various kinds that has been disorganized and unconfined to a central place for a long time since MUST began in 2014. There have also been data collection redundancies which has been costly since the same data has been collected now and again for different purposes within the school. The main objective was to develop a geoportal that will enhance spatial data sharing and effective management at NSCES i.e., data standards, availability and interoperability improvement thus creating an effective spatial data sharing community, Spatial Data Infrastructure (SDI) within the school. The methodology used implicates configuration of the database followed by server and web setup. A number of tools were used of which included an implementation using Django web framework of python that was used to come up with a user interface, a Geo-Server as spatial data server, Pycsw or Geo-Network as catalogue services tool and PostgreSQL along with PostGIS as database server. The project was implemented by following agile development approach wherein the implementation was done in phases and meetings concerning its development. The data to be used was collected from all NSCES programs through Head of Departments, class representatives and lecturers by reviewing the already existing data which will presented to the team as documents by one of the study participants. And lastly, the final product comprises of a working geoportal application with data downloading and other repository tabs in the end making spatial data availability and exchange more effective within MUST thus a contribution to the country as organizations dealing with spatial data can be managing its spatial data for effective use through this approach.

ACKNOWLEDGEMENTS

First and foremost, we would like to thank Lord God in heaven who has been our help from the beginning.

Secondly, special thanks go to our supervisor, Mr. S. Gondwe for the guidance and support during implementation of this project. We would like to give thanks to Dr. C. Chisenga who also rendered us with support and hints on how to carry on this project.

We would like to thank parents and families for the continuous support and encouragement throughout our academic life. To the Kachingwe family, Makwinja family, Bakili family and Kandonje family, we offer sincere appreciation and gratitude for the support and encouragement.

To our friends and classmates that have been right by our side throughout academic journey, we appreciate having you.

Finally, to our caring, loving and understanding beloved ones who patiently helped us with encouraging words, material things and best wishes that we might finish this project, we appreciate all the support.

Table of Contents

Table of Contents 3
1. introduction 10
1.1. Background 11
1.1. Problem Statement 13
1.2. Main Objective 13
1.3. Specific Objectives 13
1.4. Research Questions 13
1.5. Significance of Study 14
2. LITERATURE REVIEW 15
3. SYSTEM ANALYSIS AND DESIGN 18
3.1. Planning 18
3.1.1. Feasibility Analysis 18
3.1.2. Project Management and Selection 19
3.2. Analysis 19
3.2.1. User requirements 19
3.2.2. Functional requirements 20
3.2.3. Non-functional requirements 21
3.3. DESIGN 22
3.3.1. Data Flow Diagrams 22
3.3.2. Use Cases 23
3.3.3. Database design and development 35
3.3.4. The database designs 35
4. implementation 38
4.1. Software sourcing 38
4.2. Software installation 39
4.3. Software configuration 39
5. creating the django project 40
5.1. Database Development 40
6. data loading and geo-coding 42
7. uploading data 43
8. publishing raster data 45
8.1. Uploading Layers 46
9. viewing tables 46
10. interface and functionality 46
10.1. Navigation Bar 46
10.1.1. NSCES Geoportal Logo 47
47
10.1.2. Home 47

10.1.3. Spatial Data 48
10.1.4. Non-Spatial Data 49
10.1.5. Explore Maps 49
10.1.6. Contact Us 50
10.1.7. About 51
10.1.8. Help 51
10.1.9. Search 52
10.2. Authentication and Authorization 52
10.3. Log in 52
10.3.1. Interface for Log In 52
10.4. Sign Up 53
10.4.1. Interface for Sign Up 53
10.5. User Requirements 54
For a user to be registered, they are required to meet the following requirements: 54
10.6. Themes 55
11. RESULTS and DISCUSSION 56
12. Conclusions and Future Work 57
References 59

ACRONYMS/ABBREVIATIONS

MUST: Malawi University of Science and Technology

EPSG: European Petroleum Survey Group

CRS: Coordinate Referencing System

GIS Geographic Information Systems

NSCES: Ndata School of Climate and Earth Sciences

MUST: Malawi University of Science and Technology

SDI: Spatial Data Infrastructure

GIS: Geographical Information System

Pycsw: python catalogue service for the web

HTTP: Hypertext Transfer Protocol

WFS: Web Feature Service

WMS: Web Map Service

FOSS: Free and Open-Source Software

UNSDI: United Nation Spatial Data Infrastructure

OGC: Open Geospatial Consortium

ICT: Information and Communication Technology

IISD: International Institute for Sustainable Development

API: Application Programming Interface

ESRI: Environmental Systems Research Institute

HTML: Hypertext Markup Language

CSS: Cascading Style Sheets

LIST OF FIGURES

Figure 1: The geoportal portal reference architecture 17
Figure 2: use case 20
Figure 3: conceptual diagram 22
Figure 4: level 0 data flow diagram 23
Figure 5: database tables later added to data models. 37
Figure 6: database entity relationship model. 38
Figure 7: showing pgadmin panel 39
Figure 8: sample django project 40
Figure 9: showing pgadmin upon start 42
Figure 10: showing pgadmin site administration page 43
Figure 11: showing data in geoserver 45
Figure 12: NSCES Geoportal navigation menu 46
Figure 13: NSCES logo 47
Figure 14: Home page 48
Figure 15: Exploring layer page 48
Figure 16: Explore document page 49
Figure 17: Explore map page 50
Figure 18: contact us page 51
Figure 19: about page 51
Figure 20: help page 52
Figure 21: Log in page 53
Figure 22: register page 54
Figure 23: User profile page 55

LIST OF TABLES

Table 1: view data use case details 23
Table 2: zoom in and out and search use case details 25
Table 3: download and upload data use case details 27
Table 4:Data manipulation use case details 29
Table 5: data backup details 30
Table 6: user accounts management use case details 31
Table 7: Data type and formats obtained per department stored within the geoportal database 35

1. INTRODUCTION

An information system (IS) is a set of organized resources (materials, software, staff, data and procedures) that allow to collect, filter, process and disseminate data on a given environment (semoud, 2006). Information Systems are of great importance for all structures (institutions, organizations, companies, etc.). Information Systems are of great significance for all structures (institutions, organizations, companies, etc.). In reality, given the imperative part that information plays within the decision-making process in this modern age, each structure must commit a great part of its efforts and exercises to information and data collection, handling, stocking, and spread.

"GIS is computer software that links geographic information (where things are) with descriptive information (what they are). Not at all like a flat paper map, where what you see is what you get, GIS can display numerous layers of diverse data. Maps are a demonstrated strategy of communicating geographic information. When a decision needs to be made, GIS helps us gather place-based information and organize it on the digital map. We then use GIS to evaluate the decision once we fully understand the geographic consequences of our decision. We can then act in an informed, responsible manner." (ESRI, 2012)

GIS is benefiting organisations of all sizes and in almost every industry, this includes business, banking and financial services, defence and intelligence, education, governing of countries, health and human services, mapping and charting, natural resources management, public safety, transportation and utilities as well as communications.

To facilitate the sharing and reusing of geographic information many countries have implemented the use of Spatial Data Infrastructures (SDIs). An SDI is a framework of policies, institutional arrangements, technologies, data and people that enables the sharing

and effective use of geoinformation by standardizing formats and protocols for access and Interoperability. A key component of an SDI is that it facilitates accessibility
to geographic information datasets through a platform known as a Geoportal. A geoportal is an interface or a type of web portal used to find and access data and services via the internet according to Open Geospatial Consortium.
It acts as a gateway to digital geographic information content and services made available within the concept of an SDI. The Geoportal achieves this goal through the linking of geographic information-oriented websites and databases. Geoportals can be implemented at local level, regional level, national level, for public even private sector. Within the context of an SDI, it is propagated that there should be a single-entry point (a National Geoportal) to geographic information content and services for a nation.

1.1. Background

"With the development of technology, digital spatial data have become increasingly important. The public sector, private companies and the public attempt to collect and produce spatial data as a decision-making tool. Strategic planning, risk analysis, site selection, and route finding are some of the common applications. The collection and production of spatial data are time consuming and costly. Additionally, the production of spatial data from spatial raw data requires professionalism, expertise and advanced skills in the process. A number of important challenges in this area influence spatial data collection, such as lack of coordination and collaboration, duplicate or parallel activities, and spatial data collection within two or several organizations. These types of duplicated work eventually shall lead to waste of both financial and human resources." (Toomanian, 2012).
To overcome different aspects of the current challenges in spatial data handling, a Spatial Data Infrastructure (SDI) was introduced.
The term Spatial Data Infrastructure (SDI) was coined in 1993 by the US National Research Council to denote a framework of technologies, policies, and institutional arrangements, that together facilitate the creation, exchange, and use of geospatial data and related information resources across an information sharing community (Zorica et al., 2006).

"Such a framework can be implemented narrowly with the aim to enable the sharing of geospatial information within an organization or more broadly for use at a national, regional, or global level. In all cases, an SDI will provide an institutionally sanctioned, automated means for posting, discovering, evaluating and exchanging geospatial information by participating information producers and users" (ESRI, GIS best practices Spatial Data Infrastructure (SDI), September 2010).

Spatial Data Infrastructure works hand in hand with Geoportal. A distinction must be made between a Spatial Data Infrastructure and Geoportal. Geoportal is an integral part of the SDI and contributes to its construction. In SDI, the geoportal ensures three major functions: data publication, research and download.

Open Geospatial Consortium (OGC) defines Geoportal as « a human interface to a collection of online geospatial information resources, including data set and services" (OGC, 2004).

Ndata school of earth and climate sciences is one of the schools under Malawi University of Science and Technology that offers different programs. The programs are Bachelor of Science in Geo-Information and Earth Observation science, Disaster Risks Management, Earth Sciences, Meteorology, Sustainable Energy System. All these programs at some point at time are in need to use spatial data. Among these programs, spatial data is collected continuously. Apart from this, these programs are in continuous process of finding spatial data that it is somehow costly to find. This being the case, NSCES needs to have geoportal that will be used for easy sharing of spatial by uploading and downloading of spatial data remotely.

The main focus of the project was to develop an effective and efficient means of discovering spatial and non-spatial data from different sectors. GeoNode was chosen to develop the geoportal for NSCES with dynamic updates, background data loading and rich data visualization and controls. GeoNode was chosen because it is composed of following components;

- Django that python web framework
- Geoserver
- Pycsw
- PostgreSQL.

The geoportal incorporates the map services that were used to show the map features on the client side. Since the geoportal focused on the NSCES, the images shown on user interface and other functionalities were set to explain more about departments under NSCES when the geoportal is launched

1.1. Problem Statement

It has been a common challenge to find and access digital spatial resources which are widely distributed at different government agencies and websites (Li, Wang and Bhatia 2016). On the other hand, a lot of data redundancies exist, money and human resources are wasted in duplicated data collection and maintenance efforts (Rajabifard and Williamson 2001, Maguire and Longley 2005). These problems are recognized in NSCES as the lack of a geoportal has led to increase in cost for spatial data discovery and exchange in addition to low spatial availability and reliability.

1.2. Main Objective

The main objective was to create a Geoportal for NSCES which shall be used for uploading, downloading and visualizing spatial and non-spatial data for different sectors.

1.3. Specific Objectives

- To create geoportal interface where people can register and log in
- To create database containing spatial and non-spatial data of the geoportal
- To create upload and download options in the geoportal for data discovery

1.4. Research Questions

- Can the development of geoportal for NSCES improve spatial data availability at MUST?
- Can introduction of geoportal for NSCES reduce costs for finding spatial data at MUST?
- Can development of geoportal for NSCES reduce duplicated efforts in management and collection of spatial data?

1.5. Significance of Study

This study assesses the challenges that have been there because of lack of geoportal at MUST and assesses how introduction of geoportal will reduce these challenges. This study is expected to increase the knowledge on usability and importance of having geoportals for different organizations including MUST itself. The study will further serve as benchmark data for any further investigation, as a useful material for academic purposes, and as an added literature on the existing knowledge.

2. LITERATURE REVIEW

Living in an age of information with information being essential to tackle the issues of today's society is a modern age norm. Spatial information in particular is one of the most critical elements underpinning decision making for many disciplines. In the past used maps used to show where people and objects were located. Today this has evolved into a complex digital environment with sophisticated spatial and related textual databases, satellite positioning, communication networks like the internet as well as wireless applications.

The advances brought by Information and Communication Technology (ICT) helped to further confirm the importance of information systems. Their impressive capacity for processing huge volumes of data, interconnection between site webs, resources or geographically remote operators, account for the wide use of these advanced technologies in information processing and dissemination in real time, instead of using classical and slow mean of communication.

Sustainable development, according to the International Institute for Sustainable Development (IISD), "is development that meets the needs of the present without compromising the ability of future generations to meet their own needs. It contains within it two key concepts: the concept of needs, in particular the essential needs of the world's poor, to which overriding priority should be given; and the idea of limitations imposed by the state of technology and social organization on the environment's ability to meet present and future needs."

In general, sustainable development poses the challenge of managing large volumes of unstructured information and knowledge. For that, several methods have emerged: the semantic web techniques based on ontologies and metadata; knowledge engineering projects, wiki systems like Encyclopedia Ekopedia1; Spatial Data Infrastructure systems.

Here, the main focus will be on Spatial Data Infrastructure (SDI) as one way of sustainable development system.

A formal definition of a SDI was given by the US Federal Government in 2002 which states that a SDI is "the technology, policies, standards, human resources, and related activities necessary to acquire, process, distribute, use, maintain, and preserve spatial data." A SDI is potentially formed by the convergence of different geospatial data providers – each one of them enabling access to data through specific internet services. SDI is composed of three components: organizational component, functional component and technical component.

For Nebert & Lance (2002), SDI components include:

- Communication networks, forming the physical and logic infrastructure component
- Geospatial data as well as their sources, forming the geospatial data directory
- Geographic Information System (GIS) to communicate the different elements
- Interoperability norms and standards
- Institutional policies and arrangements
- Institutions, agencies, partner organizations

Geoportal has a role in Spatial Data Infrastructure for it is an integral part of the SDI and contributes to its construction. Geoportals could be divided into two groups: catalogue geoportals and application geoportals (Akinci & Cömert, 2007).

Catalogue geoportals are concerned primarily with organizing and managing access to Geographic Information. We have for example the FAO's geoportals (http://www.fao.org/geonetwork),and the UNSDI's (http://www.geonetwork.nl)

Application geoportals provide on-line dynamic geographic web services, for example, MapQuest provides routing services (www.mapquest.com)

The Geospatial Portal Reference Architecture, illustrated in Figure1, represents the guide of the Open Geospatial Consortium (OGC) for the implementation of a standardized geospatial portal (Horakova et al., 2007).

OGC specifies 4 types of services inherent for a complete implementation of a normalized geospatial geoportal. It also identifies the OpenGIS interoperability standards applicable to these services.

These four types are classified respectively as: portal services, catalogue services, representation services and data services.

- **Portal services**: provide access to geospatial, portal and users management and administration;
- **Catalogue services**: allow the collection, storing, and management of descriptive information on data stored in the database;
- **Data services**: provide access to spatial content in databases; they allow data encoding using a defined common interface;
- **Representation services**: allow geospatial information access and visualization through interactive mapping services.

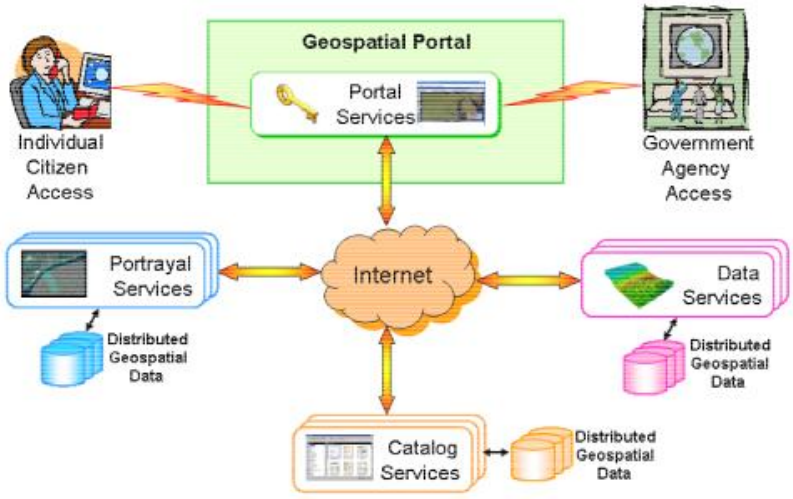


Figure 1: The geoportal portal reference architecture

3. SYSTEM ANALYSIS AND DESIGN

The methodology used in this research project includes planning, analysis, design and implementation. These have been included in the document as separate sections from each other. The system development approach used for the research project is agile.

3.1. Planning

Lack of a geoportal for Ndata school has resulted in several data efficiency and organization challenges as a result there has been a need to find an alternative way to store, share and organize data at a centralized place to ease access thus the project aims to develop a geoportal

3.1.1. Feasibility Analysis

3.1.1.1. Technical Feasibility

The portal development was technically possible as all the software's/most of them used were free and open-source software thus easily affordable/accessed via the internet. The only problem was technical expertise within the development team, as there was little/no experience with some of the software to be used in implementing the project but most of these software's were easy to learn on their functionalities thus a quick learning on the software usage and functionalities. The internet infrastructure for the software downloading was available.

3.1.1.2. Economic Feasibility

A simple cost-benefit analysis was performed based on estimates. And the conservation estimates show that the geoportal development and application have a good chance of

significantly enhancing the school's spatial data interoperability and access thus decreasing cost related to data access hence the project is likely to benefit the institution in terms of data collection storage, management, and access

Using equation ROI (return of investment) = (total benefits – total costs)/total cost.

3.1.1.3. Organizational Feasibility

In terms of organization perspective, the project has little risk as to the School's department heads, lecturers and students showed a strong interest in the project hence an opportunity that proposed users will be willing and ready to use the benefits from a system. The approval of the project from the school authorities showed that the geoportal will be welcomed by many within the School.

3.1.2. Project Management and Selection

3.1.2.1. Project Plan (work plan)

Due to the nature of the project and time available Agile model was used as the methodology approach. Very little time was available to implement the system (4 to 5 weeks) and as the development team was small as well (4 people) hence Agile worked best.

3.2. Analysis

From previous studies (done by GIS first graduating class) requirement assessment was done through document analysis. The documents contained the geoportal design requirements and user specifications.

The analysis was hence simplified lessening the work as limited time was available to develop the portal but there was a risk that some of these requirements might have changed over the year hence to tackle this challenge a simple observation was done on the spatial data interoperability within the School.

3.2.1. User requirements

User requirements simplified in term of use case models (diagrams)

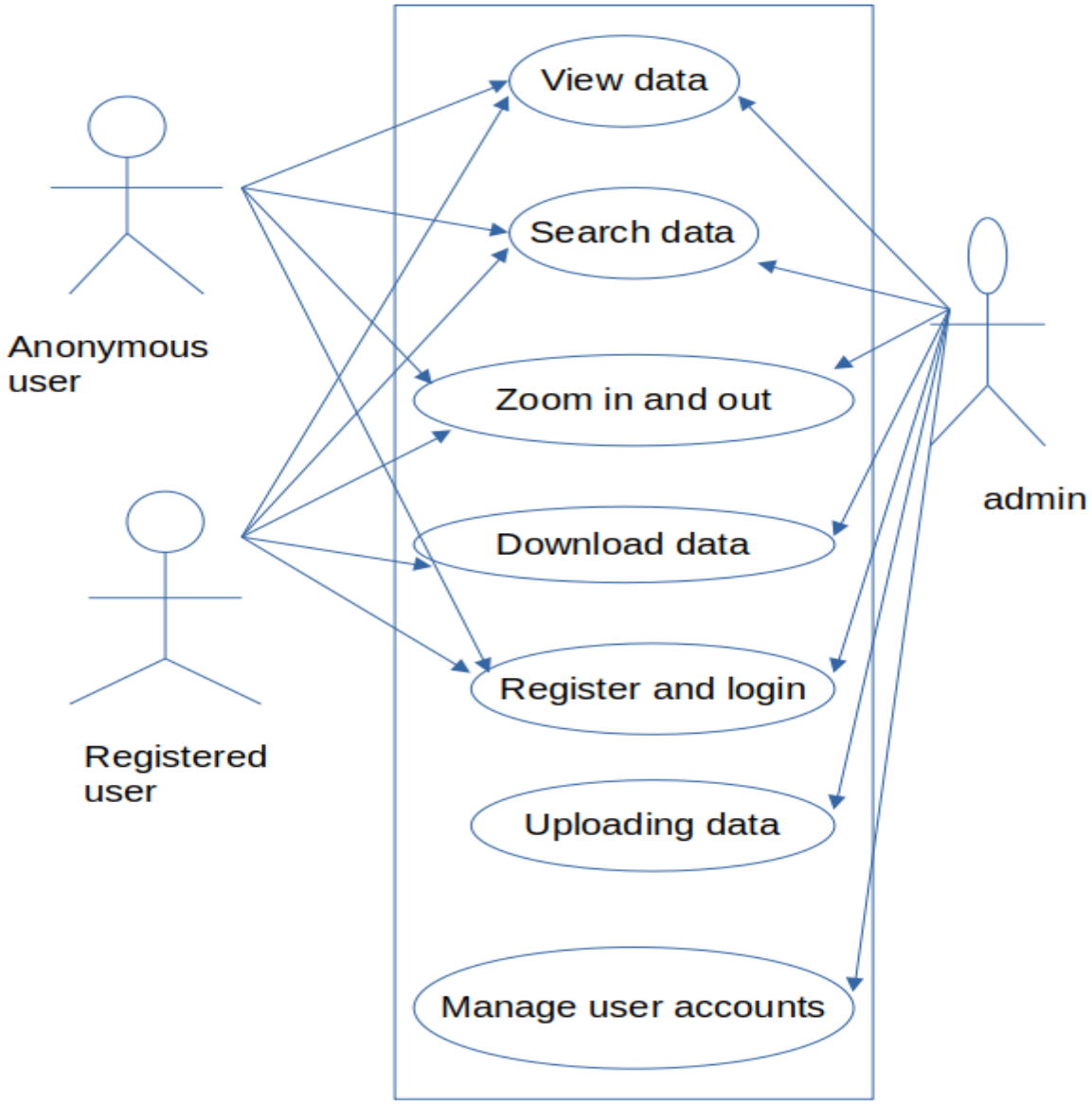


Figure 2: use case

3.2.2. Functional requirements

3.2.2.1. Search and browse

The system will allow users to search for data layers and maps

3.2.2.2. Data authentication

The system should be able to provide authentication functionalities

3.2.2.3. Data entry

Should be able to receive data uploaded by users

3.2.2.4. Data retrieval and visualization

The system should be able to connect to the database and retrieve any type of data required by users and display the data to users

3.2.2.5. Data update

The system should allow data to be updated if a need arises.

3.2.3. Non-functional requirements

3.2.3.1. Operational

Database constructed to facilitate searches by data types program/department name, types of data, department names/program names, and type of data formats.

The system will run almost on any web browser

In the event of data loss, the user can be able to recover data provided a backup is available.

3.2.3.2. Performance

Speed will be monitored and kept at an acceptable level

3.2.3.3. Security

To update/upload data user will need registered accounts and metadata restrictions and regulations

3.2.3.4. Cultural and political

Non required human ethic values will be violated within the course

3.3. DESIGN

3.3.1. Data Flow Diagrams

The creation of data flow diagrams goes under Process Modelling. Process Modelling is a method of describing a business process.

Firstly, the Conceptual Diagram:

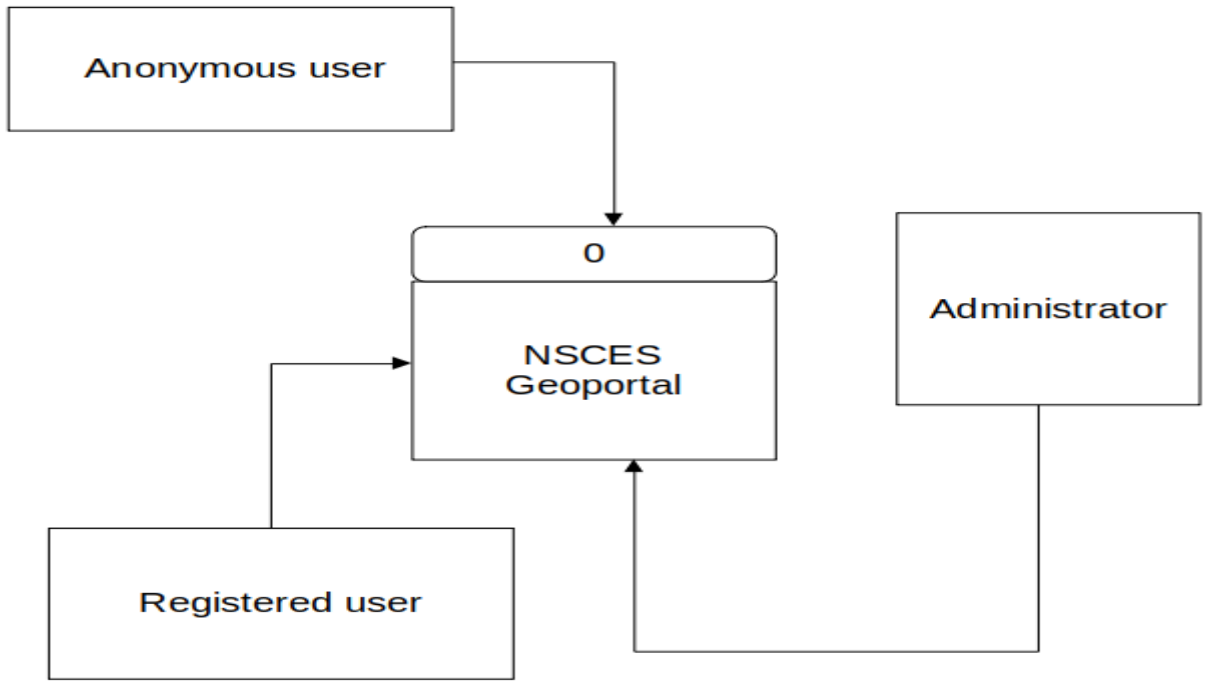


Figure 3: conceptual diagram

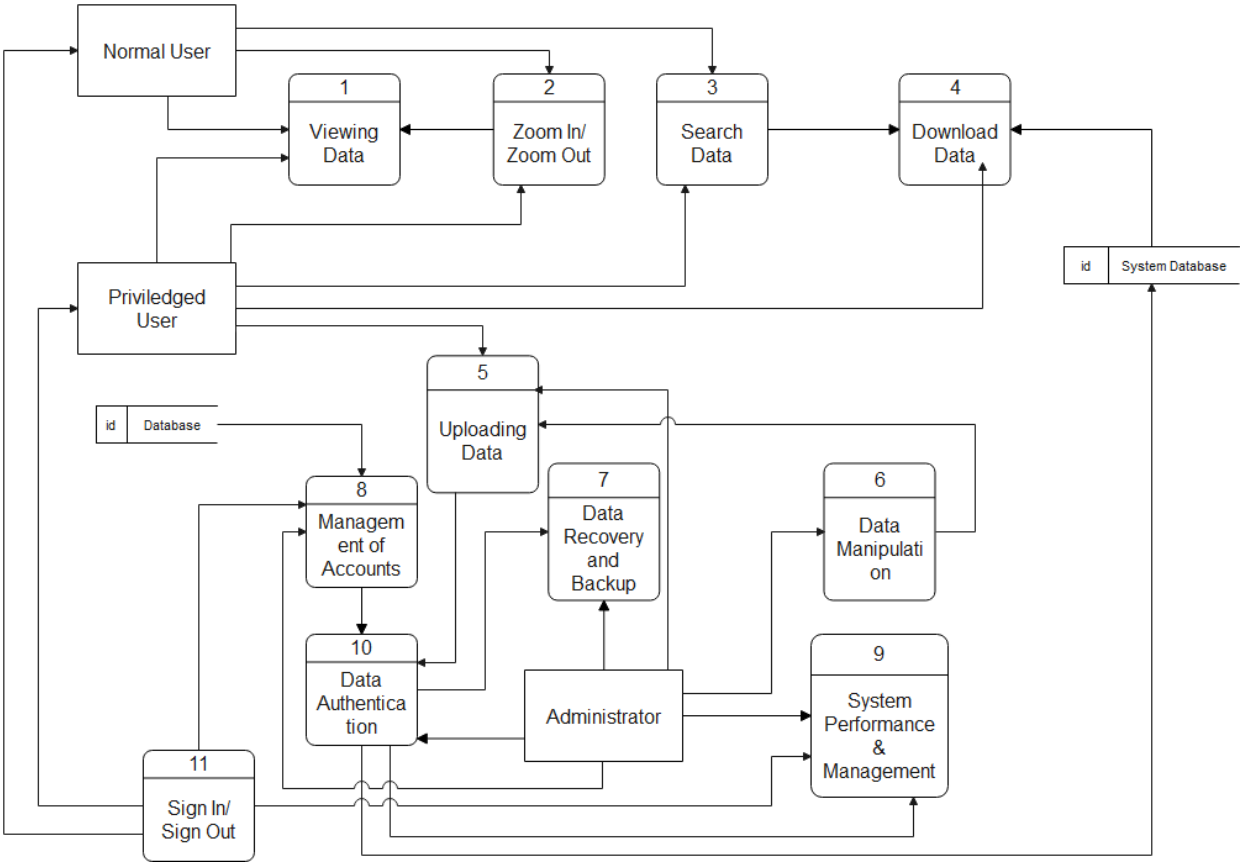


Figure 4: level 0 data flow diagram

3.3.2. Use Cases

The primary purpose of this project was to create geoportal to efficiently and effectively discover spatial data by uploading, publishing and downloading the data. The geoportal facilitates students, administration stuffs, other professionals to easily make use of spatial data found in the system. This chapter presents the use cases for the geoportal. Section describes the user interface of geoportal.

Table 1: view data use case details

use case	View data	ID : UC 1	Priority: high
Actor	Normal/ privileged/ administrator users		
Description	- The user or actor has to view the data such as vector or raster data. On the home page. - The users have to view a little metadata of the spatial data on the homepage.		

	- The user has to see also some of the data the database is keeping that may be of interest	
Trigger	- s/his project or assignment - An idea to looking for a place on a web map and to see available data. - for an administrator to see what type of data is available and what data should be added - for a privileged use to see and upload additional data	
Type	**External**	**Information for steps**
Preconditions **Normal course**	- the user has to have internet access - the user has to have a working internet browser - for a privileged user have to have an account - the administrator has to have his/her admin account	←internet access The database sends data to be viewed ← user's info sent to database and admin authorization account creation
	- the user will put the domain name or search the web on the browser and the home page will appear with a view window for spatial and tabular data - the home page will also have buttons for sign in and up	
Post conditions	- the home page will take data from the database and display it on the web browser - the users will sign in and sign up for more visualization - privileged users will be able to view the rest of the data not on the data on the home page	

Summary inputs	**Source**	**Output**	**destination**
Domain name on the browser Sign up and sign in	Database wms	Spatial and tabular data Vector and raster	Users interface & admin database

Table 2: zoom in and out and search use case details

use case	Zoom in and out	ID : UC 2	Priority:medium
Actor	Normal/ privileged/ administrator users		
Description	- a user will have access to a zoom tool on the view window where s/he can zoom in and out of the map - the zoom window will help to change the scale of the available spatial data i.e. vector/raster		
Trigger	- data visualization (the need to see clear objects) - An idea to looking for a place on a web map and to see available data. - for an administrator to see what type of data is available and what data should be added - for a privileged use to see and upload additional data		
Type	External		Information for steps
Preconditions	- the user has to have internet access - the user has to have a working internet browser - for a privileged user have to have an account - the administrator has to have his/her admin account		←internet access users have to log in or sign up ←users info sent to database and admin authorization account creation
Normal course	- the user will put the domain name or search the web on the browser and the home page will appear with a view window for spatial and tabular data - the view window will have the zooming tool		-the user will be able to click the zoom tool
Post conditions	- the home page will take data from the database and display it on the web browser - the users will sign in and sign up for more visualization - all users will be able to zoom the spatial data on their view window		
Summary inputs	Source	Output	destination

Domain name on the browser Sign up and sign in	Database Web map service	Zoomed in and out Vector and raster data	Users interface & admin database
use case	**Searching for data**	**ID : UC 3**	**Priority :high**

Actor	Normal/ privileged/ administrator users	
Description	- all the users will be able to search for spatial data available on the website - the home page of the web will have a search button for easy access	
Trigger	- to find specific area and details (metadata of a specific area or location) - for administrator and privileged user with an account can use the tool for easy finding and editing required information on the web	
Type	**External**	**Information for steps**
Preconditions	- the user must have internet access - a proper and fast web browser e.g., chrome - the use must have an account to access some of the important information on the web	←internet access users have to log in or sign up ← user search for specific information from the database -the data will be displayed on the web for a user to see
Normal course	- a normal user can search for data on the home page but the results will be minimal (e.g., on spatial location on the view window) - A privileged user can log in and access more data e.g., spatial and tabular.	
Postconditions	- the search tool on the web will trigger a retrieval process of data from the database - the user will be able to view and download the data depending on his or her privileges	

Summary inputs	**Source**	**Output**	**destination**

Domain name on the browser Sign up and sign in and a search tool or button	Database Web map service	The searched available Data in our database	Users interface & admin database

Table 3: download and upload data use case details

use case	Download data	ID: UC 4	Priority: high
Actor	privileged/ administrator users		
Description	- only the users will be able to download the data from the database - the users will require to enter specific details about themselves - the users will be able to search the data they want and download if they have internet		
Trigger	- a project a user is working on - maybe to share to some people who don't have access or have no account - for learning purpose and other personal or group task		
Type	External		Information for steps
Preconditions Normal course	- the user will have to create an account and the administrator have to respond ask quick - the user must search for the needed data - if the data is available the user can download and serve it on their computers or phones		←internet access users have to log in or sign up ←user's info sent to database and admin authorization account creation
	- the user creates an account by putting his details in spaces displayed by the web - administrator allows the creation of the account - the user search and download the data		-the user will be able to click the zoom tool The user will be able to download the data
Postconditions	- the database receives the needed information from the user - the user searches for specific data from the database and it will be displayed on the web - the user is interested in the data quality will download the data		

Summary inputs	Source	Output	destination
Domain name on the browser Sign up and sign in	Database Web map service	Specific data to be downloaded	Users interface & admin Database and the user's computer storage
use case	Uploading data	ID: UC 5	Priority: high

Actor	privileged/ administrator users	
Description	- uploading data means adding new data to the database - the data must be spatial (tabular, vector or raster) - on a user with an account will be able to perform the task but with authorization from the administrator	
Trigger	- the need to increase available data to be shared with other users on the platform - the need to improve the quality of the data present on the database - The idea of satisfying the user's needs on the website.	
Type	External	Information for steps
Preconditions	- a user must create an account unless he or she is an administrator - the user must upload relevant information on the web - the information will be authorized into the database by the administrator	←internet access users have to log in or sign up ←user's info sent to database and admin authorization account creation -user will upload the data into the database and wait for the administrator's approval
Normal course	- the user will create an account and the administrator will approve - the data that a privileged user is uploading will go through an administrator for approval	
Postconditions	- the administrator will approve the creation of the user's account - the administrator will receive and analyses the data a user is trying to upload - the administrator will send the data into the database and will appear on the	

	website		
Summary inputs	**Source**	**Output**	**destination**
Domain name on the browser Sign up and sign in New data to be upload	Database Web map service Users	Zoomed in and out Vector and raster data From the new data	Users interface & admin Database administrator

Table 4:Data manipulation use case details

use case	**Data manipulation**	**ID: UC 6**	**Priority: high**
Actor	Administrator		
Description	- the administrator will be able to manipulate the data in the database (change metadata, upload new vector and raster data) - the administrator will be able to authorize the data to be uploaded		
Trigger	- assessment of available data - increase the availability of spatial data to the users - increase the quality of the data - to control data to be uploaded		
Type	**External**		**Information for steps**
Preconditions **Normal course**	- the user has to have internet access - the administrator need to have an account special for him - the administrator has to know what quality data should be added or uploaded on the web		←internet access users have to log in or sign up Have to know how to create maps have to know qualities of good spatial

	- the administrator will log in to his admin's account - the administrator will be able to create additional maps to be added to the database for users - the administrator should be able to judge the maps trying to be uploaded by other users	data
Postconditions	- The administrator must try and add data to improve the data quality and availability for users - The administrator should be able to authorize data uploading for other users	

Summary inputs	**Source**	**Output**	**Destination**
Domain name on the browser Sign up and sign in	Database Web map service Administrator and users	Creation of maps by the administrator and also authorization of new data	Users interface & admin database

Table 5: data backup details

use case	**Data Recovery and backup**	**ID: UC 7**	**Priority: high**
Actor	Administrator		
Description	- The administrator should be able to keep back up data in media devices so that should be always available to other uses of the geoportal - The administrator should be able to recovery (retrieve)deleted data from the geoportal		
Trigger	- The need for data availability to the normal users by the administrator - The need to store the important data in case the system is down - The need to provide an alternative to the users when severs are down hence the backup data		
Type	**Internal**		**Information for steps**

Preconditions **Normal course**	- The administrator must have two separate servers up and running in case one has problems - The administrator must ensure that back power for the servers is available in case they are a blackout - The administrator must know how he/she can recover deleted or miss placed data in the system	← hardware devices for storage e.g. sever Knowledge on how to recover lost data in the geoportal database Backup power to keep the system up and running
	- The administrator must buy or own two separate servers for data backup - The administrator must know how to recover lost data in the system. - The administrator must have backup generators	
Postconditions	- Recovery and publishing of lost data - The system running all times - The data backed up in two servers	

Summary inputs	**Source**	**Output**	**destination**
Backup power, backup servers, and administrator knowledge to recover data	Database Web map service Administrator and users	Recovered data and a system that runs throughout	Users interface & admin Database

Table 6: user accounts management use case details

use case	**Management of user accounts**	**ID: UC 8**	**Priority: high**
Actor	Administrator		
Description	- The administrator will be required to manage user accounts - The users will be having restrictions on what they can do on the geoportal depending on their level of account		
Trigger	- Need to control quality available data and data to be uploaded - The need for the administrator to easily arrange or modify database		

Type	Internal	Information for steps
Preconditions **Normal course**	- A normal user must have an internet connection and a web browser - The normal user must create an account to gain some or more privileges of the geoportal from the administrator - Normal users account details must be sent to the administrator's email to evaluate	←the admin will be needed to have an email to be used in receiving details of the user who is creating an account All users plus admin will need internet access
	- The administrator must have an internet connection - The administrator must log into his Gmail account and evaluate users details - The administrator must then log into his geoportal account and assign privileges to the users who created accounts,	
Postconditions	- The administrator should receive and evaluate the account details of the other users - The administrator should be able to grant the users privileges on the website depending on the user's profile	

Summary inputs	Source	Output	destination
Users details Admin credentials	Email Database and the website	User email message and details Admin privileged to users	Admin email and users page

use case	System performance monitoring	ID: UC 9	Priority: high
Actor	Administrator		

Description	- The administrator should be able to perform system monitoring, this will help in the maintenance and improvement of the system - The administrator should be able to notice all the users trouble and be able to maintain	
Trigger	- The need to give the users what they want efficiently - The need to provide a fast system to the user. I.e. a system that can process much information at once depending on the user's needs	
Type	**External**	**Information for steps**
Preconditions **Normal course**	- The administrator should have a way to take not on errors in the system from the user's side - The administrator should know how to correct the errors and improve the system performance to the needs of the normal users	←all users should have the internet to communicate on how the system can be improved. All should have ideas towards improving the system The user should know or have help improving the needed codes of the system
	- The system should send any malfunctions to the admin's account as a warning - The system will have a button where the user can report his/her needs to the admin so that during the updates the user's opinion should be included	
Postconditions	- The admin should receive the errors or malfunction of the system - The admin should take into considerations the needs of the user through a button like as a suggestion box for the user - The admin should be able to fix all the problem	

Summary inputs	**Source**	**Output**	**destination**
Ideas and knowledge on how to improve the performance	User and administrator and also the system	Improvement of the system performance	Users and admin plus the system

use case	**Data authentication**	**ID :UC 10**	**Priority :high**

Actor	Administrator	
Description	- The administrator and privileged users will be needed to check the quality of data that will be uploaded on the geoportal - The users will upload data on the web but it will not be direct the administrator have to evaluate the quality, accuracy, and some more things before the data appear in the web database	
Trigger	- The need to provide quality data to users - To prevent unwanted data in the system - To enforce rules among users	
Type	**External**	**Information for steps**
Preconditions **Normal course**	- The users should have knowledge on what kind of data they can upload - The administrator has to know what type of data a user is requesting to upload - The administrator should know what type of data a user may need from the geoportal	←internet access users have to log in or sign up Have to know how to create maps have to know qualities of good spatial data
	- The user will upload data on the website - The admin will evaluate the data depending on accuracy, duplication, quality and its metadata plus value to users	
Post conditions	- The admin will receive and evaluate data to be uploaded by the user - The admin will have the right to deny or accept the data - The admin will also have the right to give reasons to the user on why he made the decision on the data	

Summary inputs	**Source**	**Output**	**destination**
Data from user Rules from the admin	User with an account Administrator	Uploaded data and comment from administrator	Users

3.3.3. Database design and development

The database development flow followed a conceptual data model development first then a logical model design. The conceptual model demonstrates the general picture of the data organization and structure, on the other hand the logical data model included the detailed design of the database schema representation of relationships, attributes and data behaviour. The development workflow of the database was as illustrated below

3.3.4. The database designs

For effective management, maintenance and retrieval of the data, database design facilitation is very important. The following describes the design and the implementation of the database for the NSCES geoportal

a) Conceptual model

This is the main picture behind the logical database model represented by the Ndata School departments, students and heads of departments and from the collected data, the datasets acquired to provide solutions for the Ndata School of Climate and Earth Sciences data management were as the following classes

Table 7: Data type and formats obtained per department stored within the geoportal database

DEPARTMENT	TYPE OF DATA COLLECTED	DATA FORMAT (vector, raster or tabular)
Earth Sciences	-Mineral data -Soil data -Lithology (rock mass types) -Geological structures (faults, shear zones)	-vector and raster -vector -vector -vector

Climate Sciences	-Rainfall data -Temperature -Wind -Hazards (floods, earthquakes, landslides) -Pests (locusts, armyworms) -Outbreaks	-tabular -tabular -tabular -vector or raster -vector or raster -vector
Water Resource Managements	-Ground water -Water table -Water sources -Water quality data (water pH, impurities, salinity) -Water conductivity	-raster -tabular -vector -tabular -tabular
Energy Resources	-Power lines -Hydrothermal (hot springs, geysers) -Solar radiation -Water falls	-vector -vector -tabular -vector

The database system comprised of the of grouped data tables depending on the similarity and manipulations to be performed within the table. Each table represent the entities and within the tables are attributes which are the table records and each entity contain key attributes uniquely identifying the entity as shown below

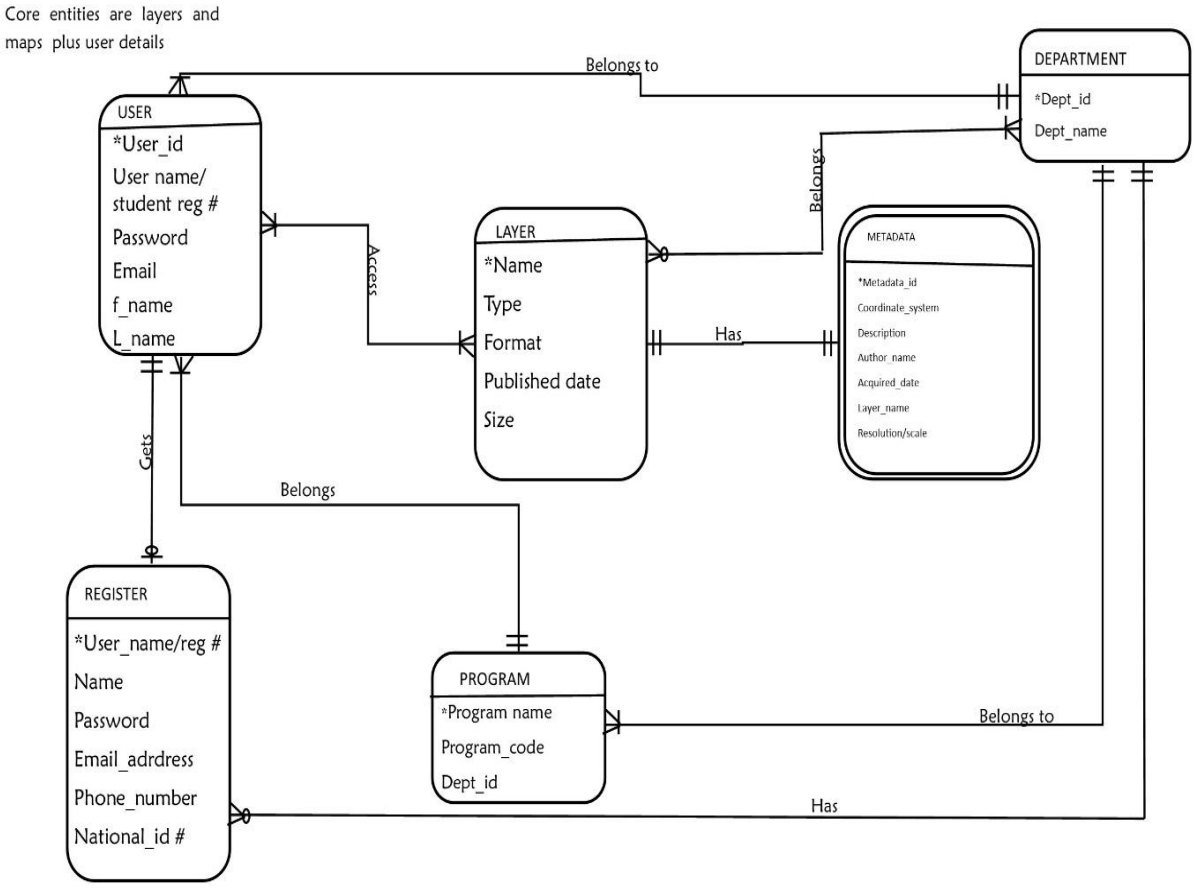


Figure 5: database tables later added to data models.

b) Logical data model

The physical design was then configured by the geonode data tables to create a complete database structure of the NSCES geoportal. The following diagram is the detailed data model of the geonode that was configured by new created models in figure 3

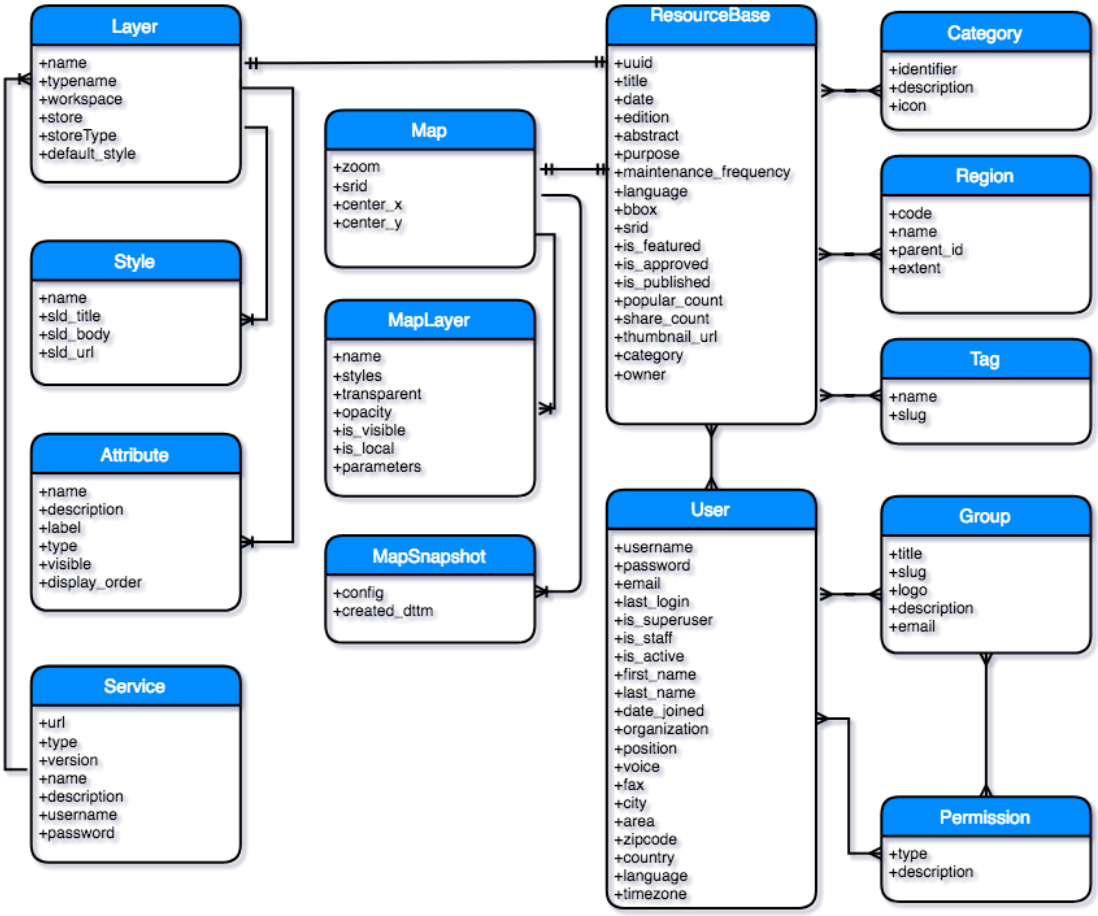


Figure 6: database entity relationship model.

4. IMPLEMENTATION

4.1. Software sourcing

The following FOSS were downloaded via the internet

- Python, Django
- GeoServer
- PostgreSQL/PostGIS
- Pycsw
- GeoExplorer
- Sublime Text
- Bootstrap

4.2. Software installation

- Python, Django installation
- GeoServer
- PostgreSQL/PostGIS
- Pycsw
- Apache
- GeoExplorer

4.3. Software configuration

- Database (PostgreSQL and PostGIS) configuration with GeoNode

  In database construction GeoNode entities was combined with other figure 7 entities whilst others eliminated according to the data requirement for the School thus the GeoNode model was modified remaining with the following tables as displayed in PostgreSQL (pg admin).

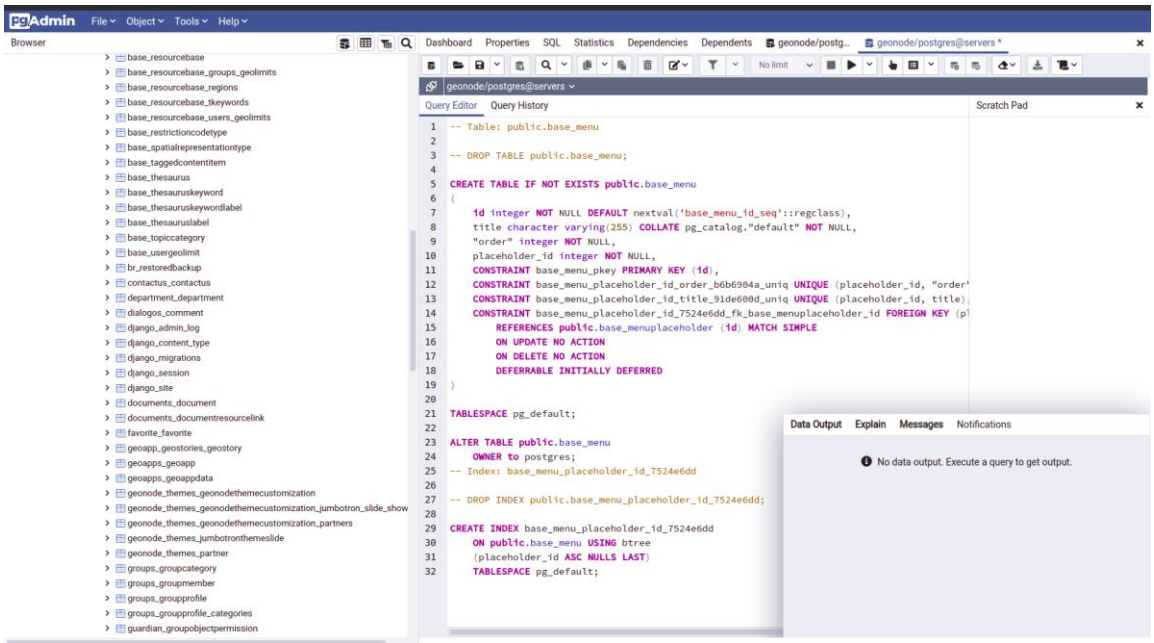


Figure 7: showing pgadmin panel

5. CREATING THE DJANGO PROJECT

- To start the Django project the following command was run from the directory in which the system files were to be stored
- Python django-admin.py startproject geoportal
- Running the command resulted in a file tree created in the specified directory as follows

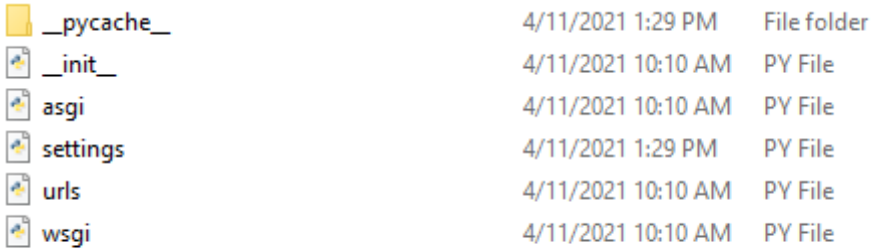


Figure 8: sample django project

- Where the _init_ file is a python empty file insinuating that the directory should be considered a python package throughout. The settings.py is another python file for configuration of the Django project, followed by the urls.py file which is the declaration for the Django project powered site content. Lastly the wsgi.py file, another python file which is an entry point for WSGI-compatible web servers to serve the project.
- After starting the project, a development server was run by running the command: *python manage.py runserver.* At http://localhost:8000 a local server for the running the project is created

  PostgreSQL database

5.1. Database Development

Database development was the first step in the implementation of the project. This is because all data present in the geoportal are stored in the database. PostgreSQL was used to store non spatial

data and PostGIS which is PostgreSQL spatial extension was used to store spatial data for the project. PostgreSQL is designed and built using the principles of relational management system where data is stored in tables making its advantages in storing, querying, manipulation and management of geospatial data.

PostgreSQL databases are used within the project as the primary back-end to store project shapefiles and facilitates the use of the Django API functionality. The database contained the user's details table, layer table, and maps tables with the metadata details table. These tables are created as classes in the Django app models file each class representing each table in the system then the classes are migrated into the project connected database by executing *make migrations*, then *migrate* to move the models into our PostgreSQL database and the models appeared as follows in the PostgreSQL database

As previously designed, the database models included layers, maps, user and metadata information, authentication and security parameters shapefiles, web-specific files like documents, etc.

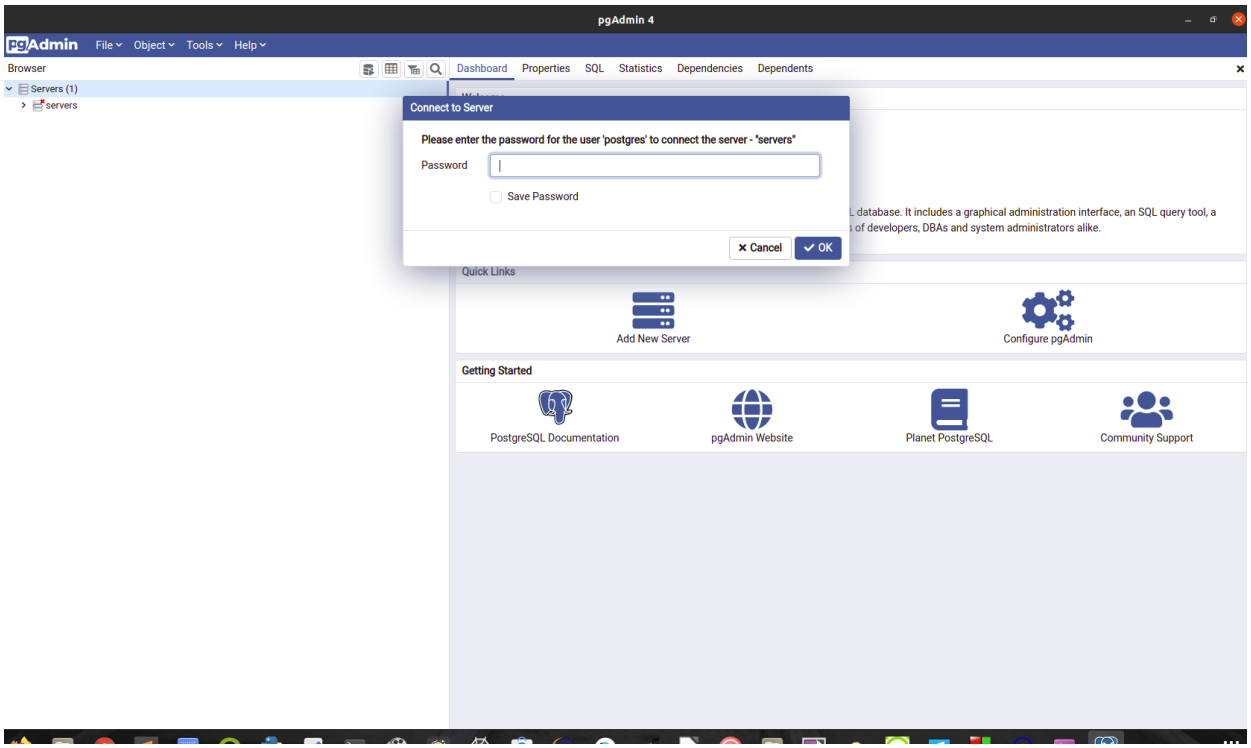


Figure 9: showing pgadmin upon start

6. DATA LOADING AND GEO-CODING

The datasets received were imported into database by using pgAdmin which is User Graphical Interface for the PostgreSQL database management system. Other databases were imported into the database by using administration panel of the geoportal. During the import process the data were standardized and field aliases and types were assigned to enable easy understanding of the variability of data in the geoportal.

Below is the site administration page, for users' accounts management, and some of the supported administrative functions.

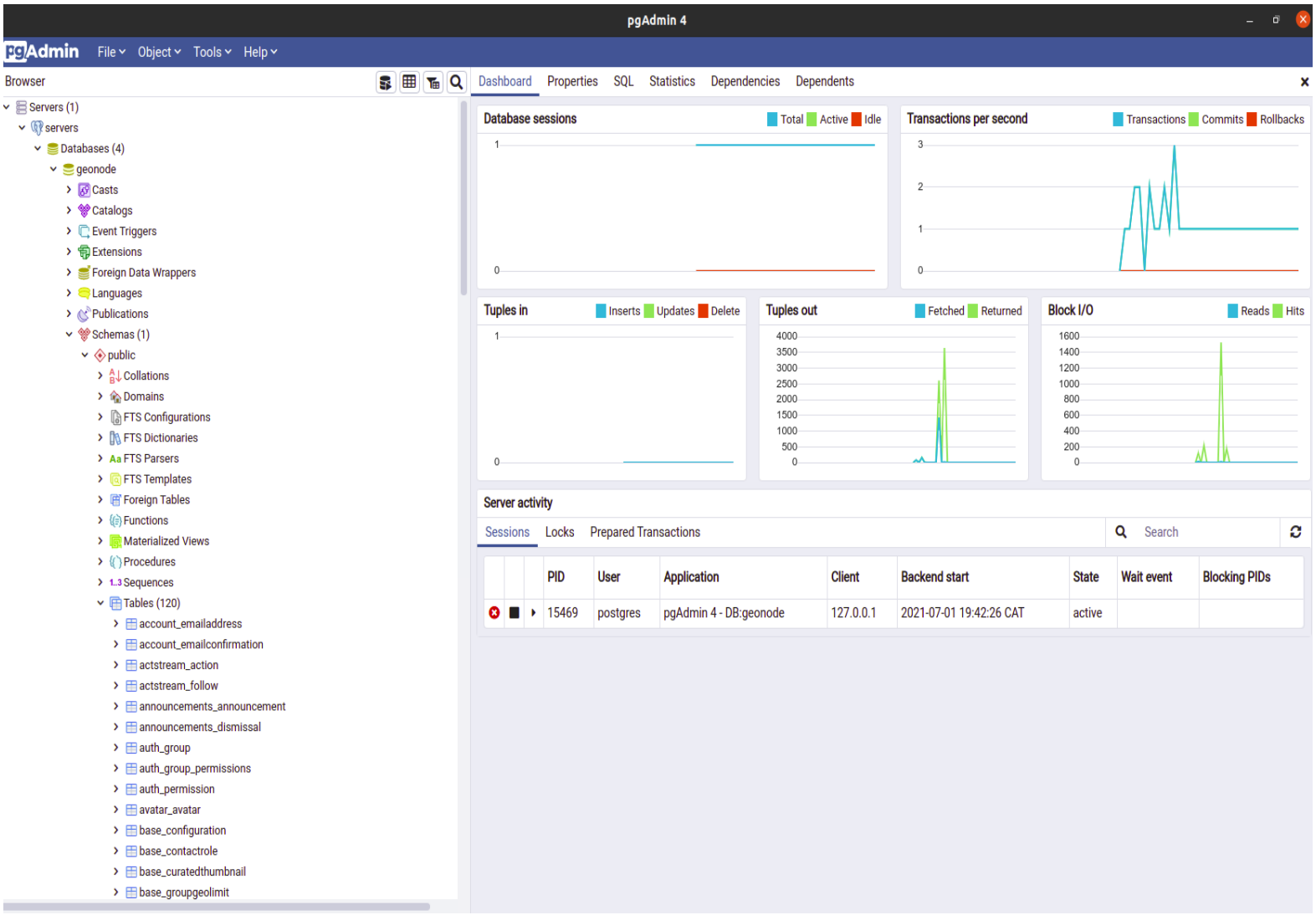


Figure 10: showing pgadmin site administration page

7. UPLOADING DATA

This is the process that will be followed when uploading data on the geoportal, whether from a end user interface, the admin page, pgadmin or geoserver. The data will be uploaded by administrator for security reasons and data reliability. The system will be managing data from four departments; namely; earth sciences, climate sciences, water

resources and energy resources. Therefore, administrator is expected to have experts in all these departments to get reliable data from.

The GeoServer will store vector and raster data. The vector data could be shapefiles of ESRI format with a .shp extension. The raster data could be in different formats, for example, ArcGrid a coverage format, and GeoTiff an image format with geographic information it is an example of a TIFF file.

Process of adding Shapefiles in GeoServer:

- The administrator must make sure that the shapefile should contain mandatory files mostly the ones with these extensions: .shp, .dbf, .shx and .prj. .shp is main shapefile that gives features their geometry, .dbf is used to store attribute data and object IDs, .shx is used to search forward and backwards for it is index file and .prj contains the SRS data that determine the data projection of the shapefile (SRID) and contains the metadata associated with shapefiles coordinate and projection system.
- Figure 11: showing data in geoserver

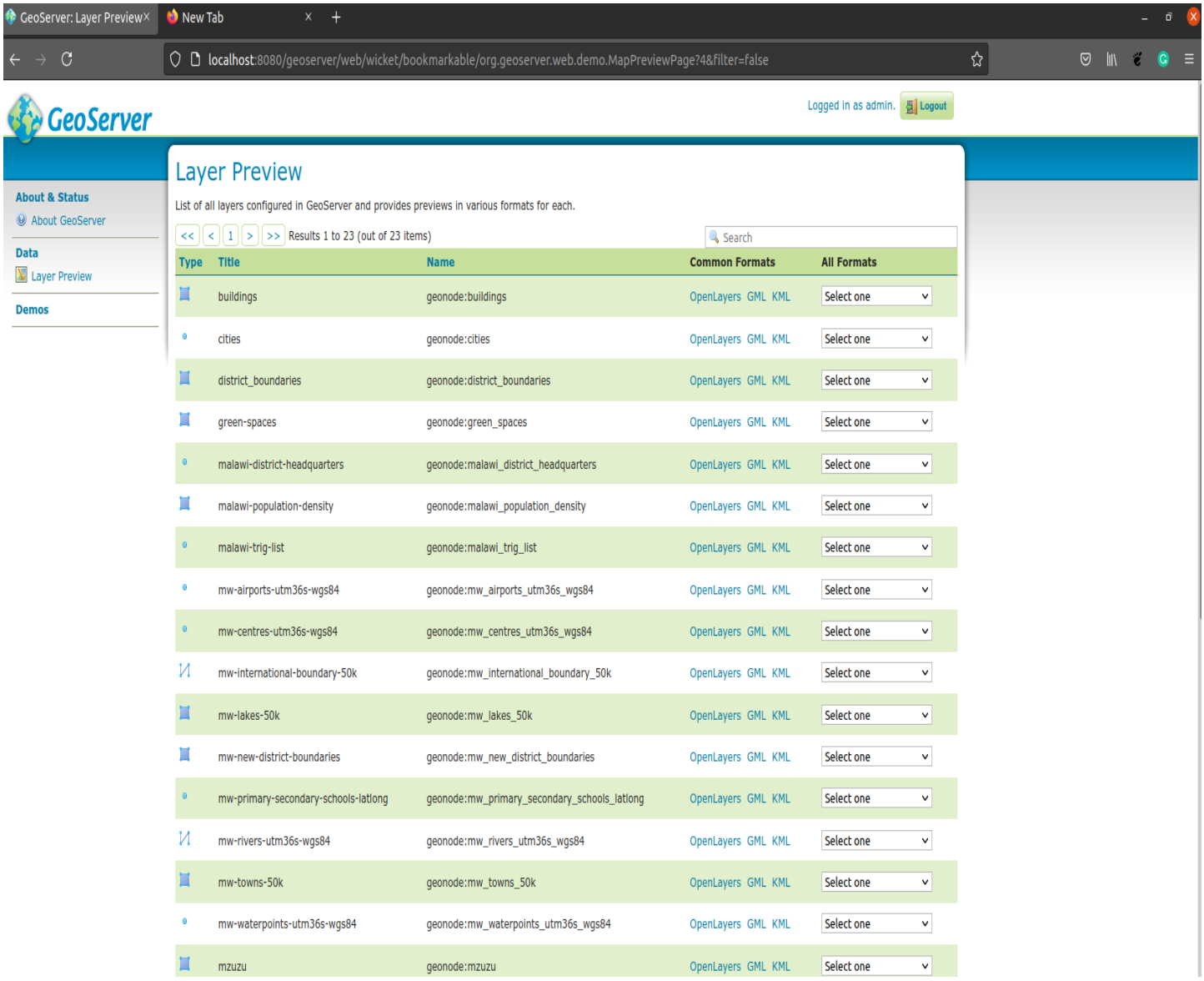


Figure 11: showing data in geoserver

8. PUBLISHING RASTER DATA

Raster data are commonly used to present satellite images, scanned maps, and digital elevation models. Some of the raster data format ArcGRID is a binary format created by ESRI used with ArcGIS. GeoTIFF is a TIFF file commonly used as a storage format for an aerial picture, has spatial data SRS and the bounding box. Other raster data formats are GTOPO 30, IMAGE MOSIAC, WORLD IMAGE.

Adding Raster Data;

Raster data could be uploaded in geoserver using PostGIS and publish using geoserver.

8.1. Uploading Layers

A layer is the most important thing in a geoportal, it represents spatial information so it can be displayed inside a map.

9. VIEWING TABLES

The attribute table contains supporting information about the layers or spatial objects. The data is in form of numbers or text.

10. INTERFACE AND FUNCTIONALITY

The user interface was designed and organized based on clarity and consistency by grouping related components together. Its design helped that the common tasks could be done easily as simplifying the data acquisition process. Below shows the geoportal layout.



Figure 12: NSCES Geoportal navigation menu

10.1. Navigation Bar

The navigation bar has 11 elements. They are as follows:

- NSCES Geoportal Logo
- Home Option
- Spatial Datasets
- Documents
- Interactive Maps
- Contact Us

- About
- Help
- Search
- Sign In
- Sign Up

10.1.1. NSCES Geoportal Logo

The geoportal has its specific logo which is used to represent the Geoportal and make it unique from other Geoportals. It is comprised of a globe with the words "NSCES geoportal" and beneath it a phrase which reads out "making spatial data discoverable".



Figure 13: NSCES logo

10.1.2. Home

This is an option of which is used to send a user back to the homepage of the geoportal in a scenario whereby a user wants to return the homepage from another page.

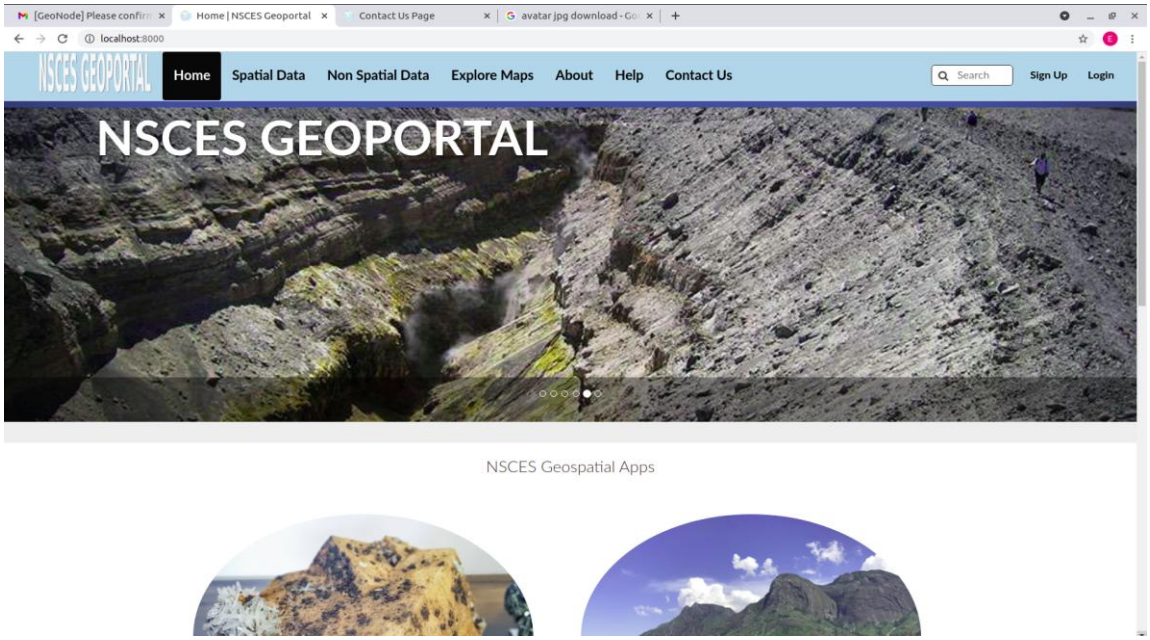


Figure 14: Home page

10.1.3. Spatial Data

This is where users access various spatial datasets based on the user's purpose or preference.

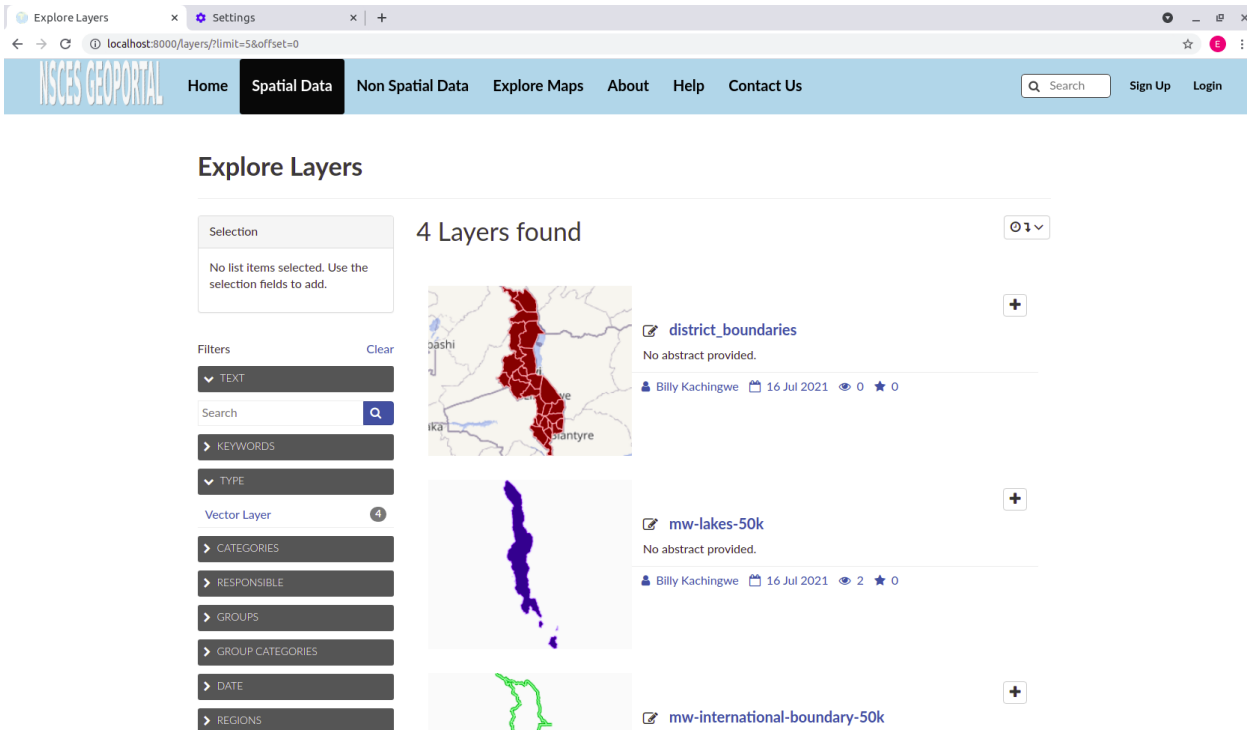


Figure 15: Exploring layer page

10.1.4. Non-Spatial Data

This is where users access various kinds of documents, to be specific, tabular data in form of a document.

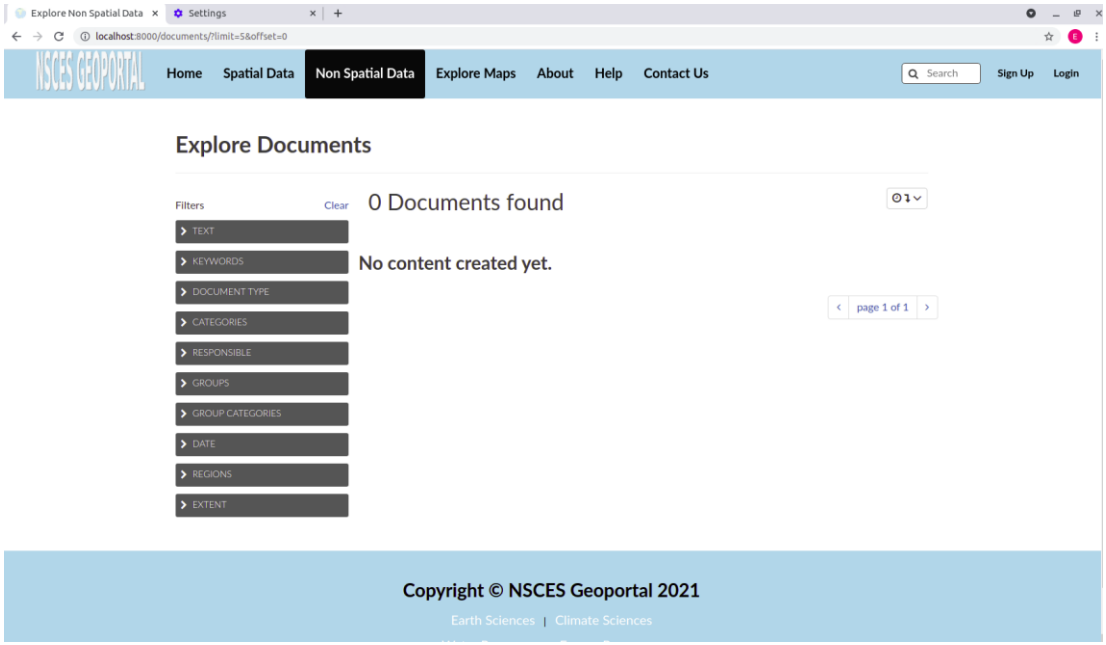


Figure 16: Explore document page

10.1.5. Explore Maps

This is whereby a user can bring out various maps and visualize them based on their preference. The user is given access to various layers of which may be explored.

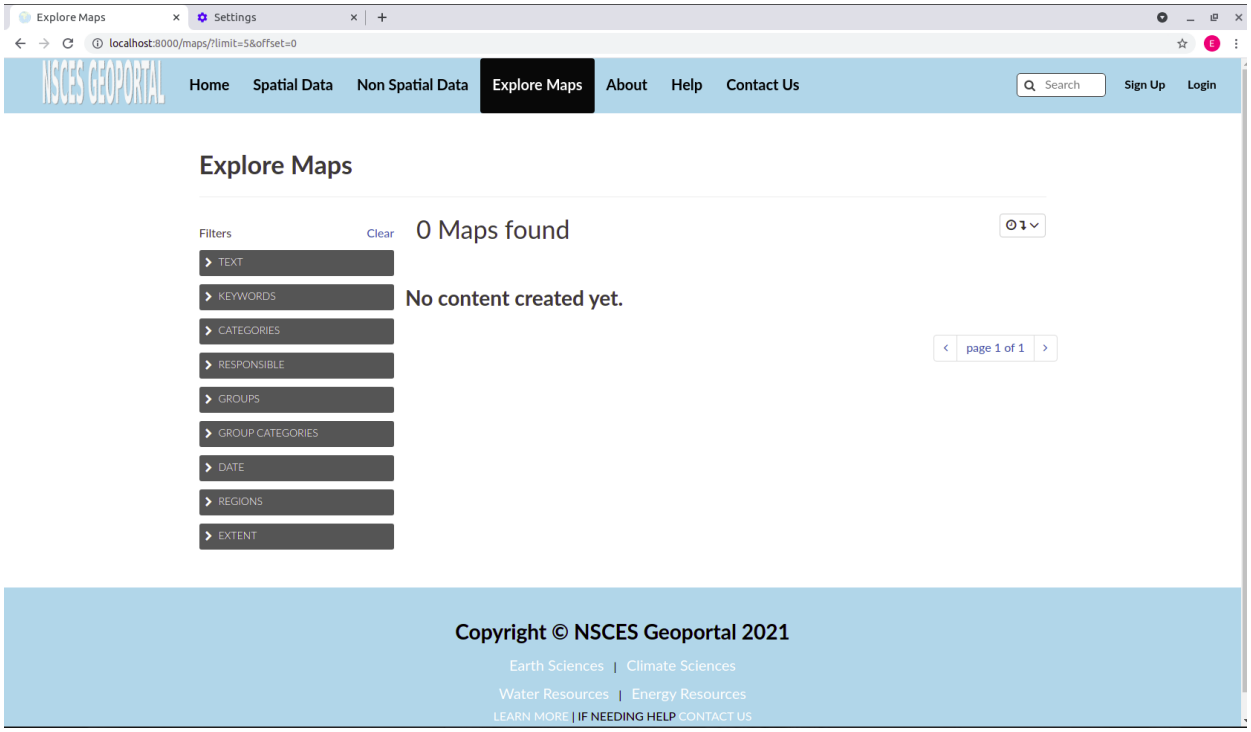


Figure 17: Explore map page

10.1.6. Contact Us

This is where users are given means of contacting with the administrators of the Geoportal. There are some phone numbers and email addresses of the administrators.

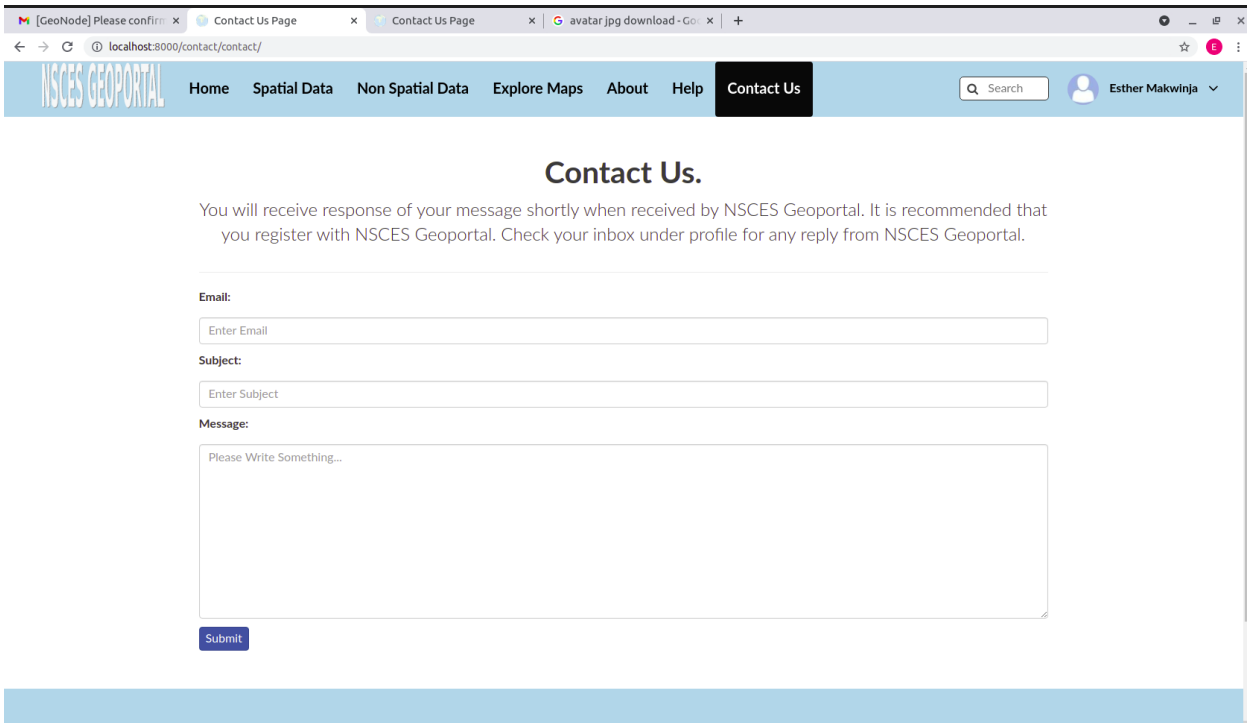


Figure 18: contact us page

10.1.7. About

This is where there is an explanation of how the geoportal came up to be and some details on the developers.

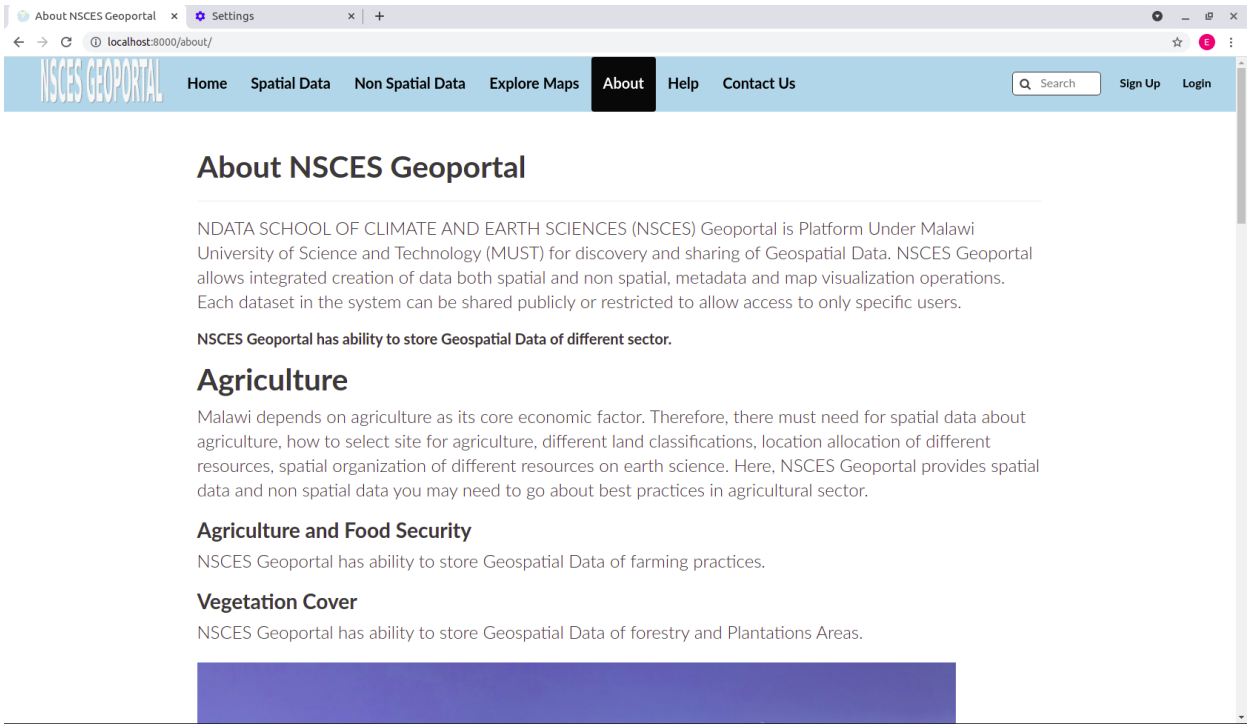


Figure 19: about page

10.1.8. Help

This is where users can go when they are needed for the help of some sort. The Help option also has a brief user manual to guide users on how to work on the geoportal.

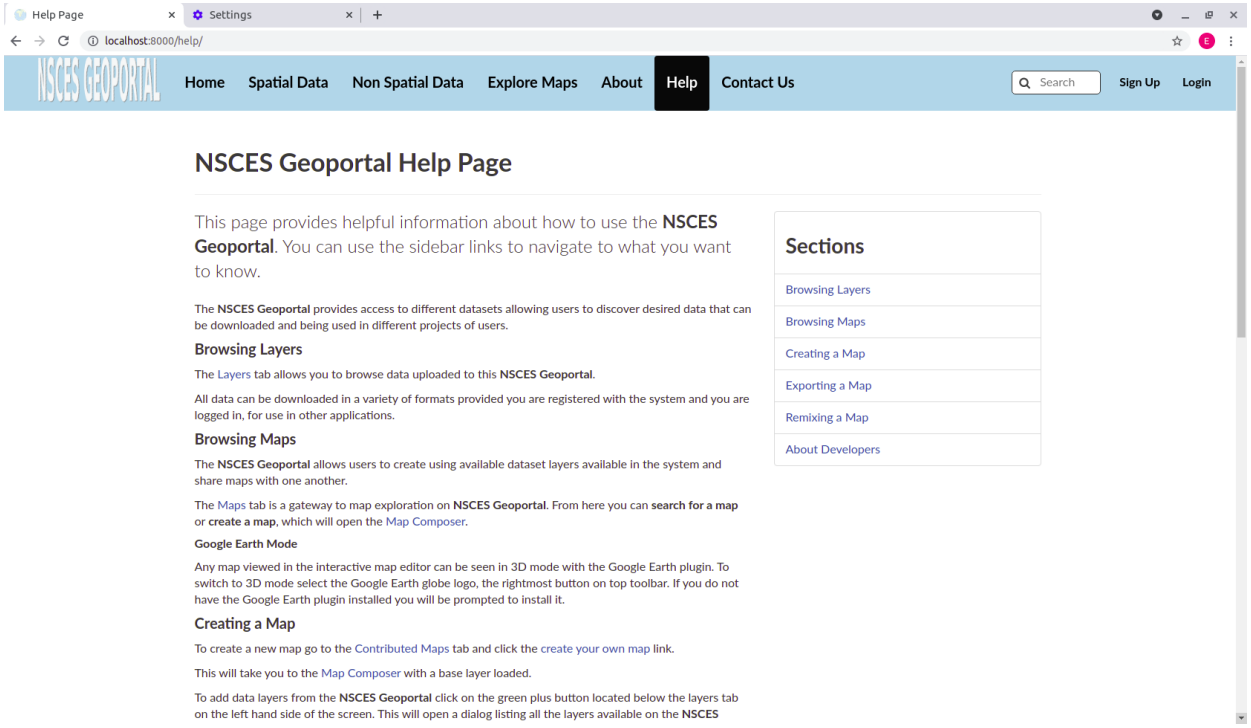


Figure 20: help page

10.1.9. Search

Positioned on the far right of the navigation bar, the search directory with a magnifying glass as its icon is used to search directly what the user wants of which is present in the geoportal's database.

10.2. Authentication and Authorization

Data found in the geoportal are protected, in such a way that users are expected to be authenticated that they are the one they claim they are for them to make full use of geoportal. Apart from that, users are expected to be authorized to desired data and resources in the geoportal by registering with the geoportal and always log in to the geoportal.

10.3. Log in

Users are expected to use registered emails and passwords to log in to the geoportal. The geoportal verifies the user information in the user table of the geoportal.

10.3.1. Interface for Log In

It can be found using following URL; http://localhost:8000/account/login/

This is whereby a user can log in into the geoportal when the user has an already registered account.

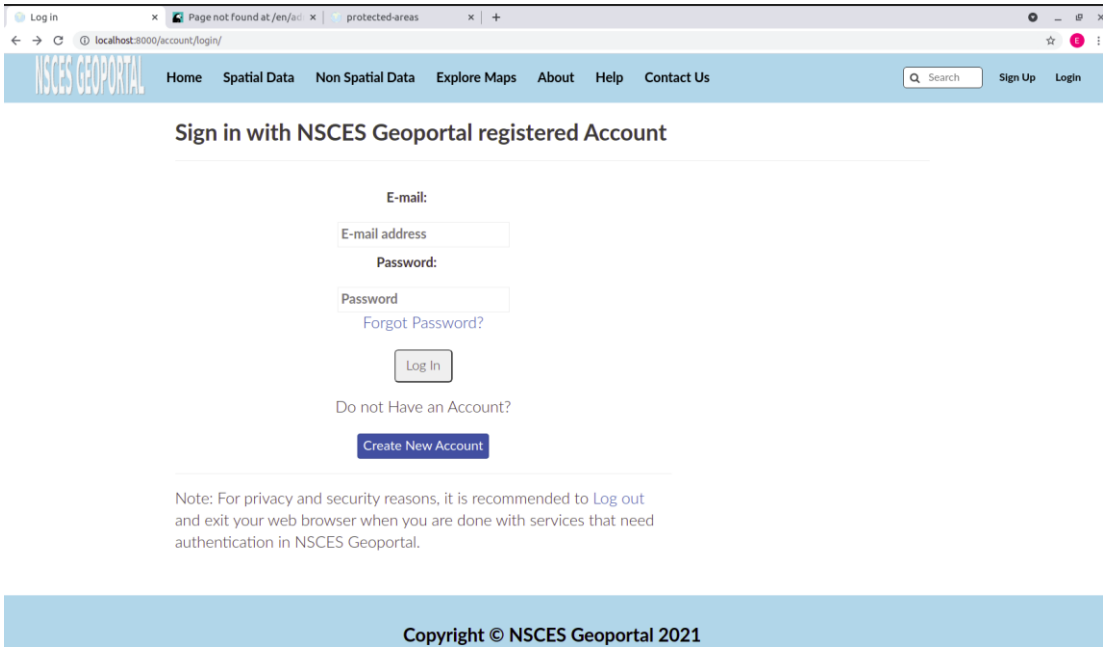


Figure 21: Log in page

10.4. Sign Up

This is whereby a user can register himself as a user.

10.4.1. Interface for Sign Up

It can be found by following URL; http://localhost:8000/account/signup/

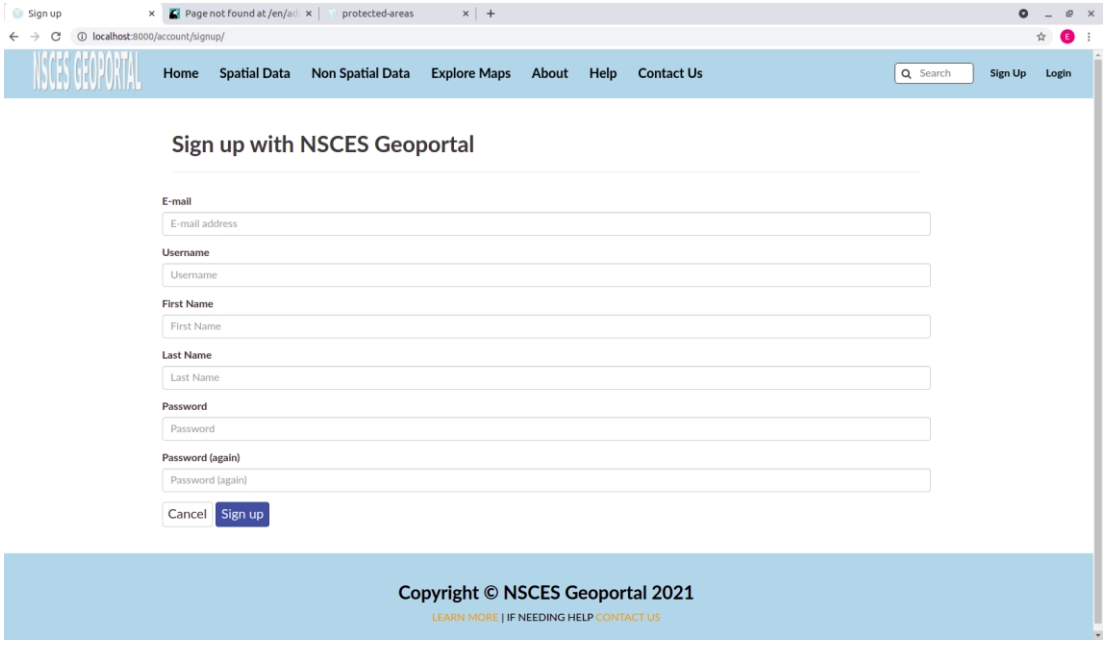


Figure 22: register page

10.5. User Requirements

For a user to be registered, they are required to meet the following requirements:

- Email
- Username
- First Name
- Last Name
- Password

Upon filling the registration forms with the user information, a Register button is present right below the boxes.

Upon successfully logged in to the geoportal, the user is taken to user profile page where user can fill in optional information. For example, position, organization, location, voice (mobile number) and brief description about self.

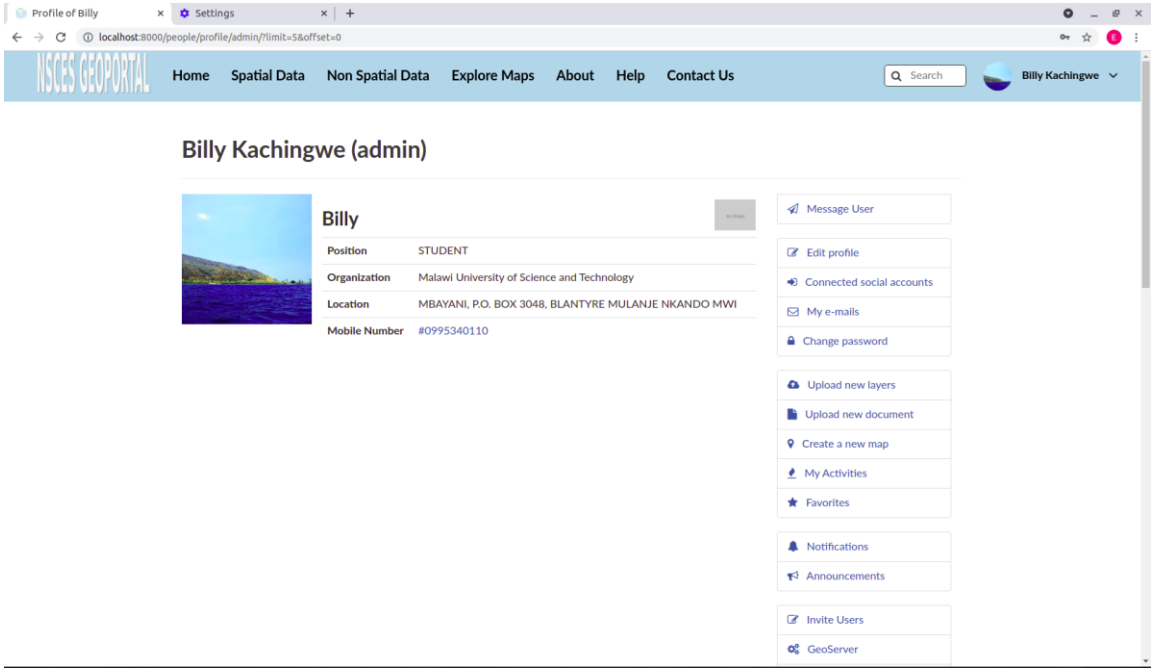


Figure 23: User profile page

10.6. Themes

These are various layers grouped up based on the category they belong to. As for the NSCES Geoportal, the layers will be categorized based on the type of data present under the specified thumbnail. There are four themes of which are:

- Earth Science
- Disaster
- Climate Science
- Water

Each theme shall have its thumbnail image of which shall portray some relevance to the layers of data present in it.

System Infrastructure

The geoportal is designed on a 4-tier architecture which comprises the user interface, web server, application server and database server. The application server is built using Django Python Web framework. Django was designed to build scalable web applications using models, views, templates. The database server is PostgreSQL because of its maturity. PostgreSQL uses mature spatial extension PostGIS for storing spatial databases. The user

interface was designed using same Django Python web framework. Django allows creation of dynamic web applications. Django combines usefulness of HTML, CSS and JavaScript to have interactive web applications. The web server that is of choice to host the geoportal was Apache http server by collaboration with mod_wsgi extension that fits with Django applications.

11. RESULTS AND DISCUSSION

In Malawi, more especially at Malawi University of Science and Technology, there is no central system that can work online for managing, uploading, sharing, downloading spatial data for different fields. Hence, spatial data is just scattered among staffs, students that make the data not easily accessed and discovered. This geoportal is there to fill the gap where students, professional and other stake holders can make use of geoportal to download data for relevant to their needs.

The geoportal provides a full measure of security where users are expected to register with the geoportal for them to make full use of geoportal functionalities. To unregistered users, they can only go into the system, explore layers found in the geoportal, but they cannot download any datasets in the system. Therefore, users are expected to register with geoportal, the link is sent to their emails which upon clicking they activate their accounts. Users are expected to be logged in for them to download data found in the geoportal. All other functionalities for example uploading are done by the geoportal administrators.

The geoportal allowed remote access of spatial data and non-spatial data across internet from any device given internet connection is there. Geoportal used PostgreSQL for database engine because of its high spatial data capabilities for it has spatial extension called PostGIS. Under same back end, geoserver was used for publishing spatial data on web page. Geoserver has built in web services that facilitate these functionalities, one of these web services is WMS that is used to publish map tiles/ images on web page in this case geoportal user interface, WFS is another web service that is used to publish and share feature attributes of the maps published across internet. All data that were not stored in

the database were stored in geoserver. Both PostgreSQL and GeoServer are FOSS that helped to reduce costs on implementation of the geoportal. The geoportal was also integrated with PyCSW which was used to store metadata of the data found in the geoportal. To make geoportal user interface interactive, python web framework called Django was used. Python was chosen for its easy to use, it has huge community of developers, it can be used in all operating systems for example windows, Linux and Mac OS. Among all web frameworks in python, Django web framework was chosen because of its MVT functionality.

12. CONCLUSIONS AND FUTURE WORK

The project was developed for the Ndata School of Climate and Earth Sciences (NSCES) under Malawi University of Science and Technology. NSCES Geoportal was required as a means by which students, professionals and other stake holders can use to discover spatial data effectively and efficiently for their relevant projects in their respective fields.

A requirement analysis under agile approach was used in all system implementation and this analysis determined the functional and non-functional requirements. A review of similar organization and their methods of development of geoportals were conducted. Combining the information in both analyses, a geoportal was developed to facilitate discovery of spatial data to different users. The geoportal enabled users to download spatial data, visualize spatial layers in the system, query and save data documents of different sectors for example urban planning, forestry, water resources, energy resources, climate sciences and earth sciences.

The geoportal was developed using geonode that makes use of following components; Django python web framework, geoserver, pycsw and PostgreSQL alongside PostGIS. The functionalities this geoportal presents enables different users from different sectors to easily access spatial data relevant to their fields. It also demonstrated how a web-based application can be utilized to organize, view, manage and distribute data with a wide audience.

Deployment of more geoportals in Malawi can be used to effectively and efficiently make spatial data of different fields. Water resources, Energy resources, forestry, construction, urban planning to be more discovered. Data is important for many transactions to take place, therefore having readily available datasets means ability to successfully to different projects. The NSCES geoportal is an ongoing project in progress thus many features can be added to the system as time goes to meet most of the user requirements time and again hence making spatial data discoverable to all.

REFERENCES

- ESRI. (JULY 2012). WHAT IS GIS.
- ESRI. (September 2010). GIS best practices Spatial Data Infrastructure (SDI).
- Ian Williamson, A. R.-E. (2003). Developing Spatial Data Infrastructures: From concept to reality.
- Meaden, G. J. (2013). Advances in geographic information systems and remote sensing for fisheries and aquaculture.
- Toomanian, A. (2012). Methods to Improve and Evaluate Spatial Data Infrastructures.
- Mabeya M. Waithaka E. (2014). Developing a Geoportal for Country. International Journal of Science and Technology. ISSN (online) 2319-7064
- Hu. Y. (2017). *Spatial Data Infrastructures*. University Consortium for Geographic Information Science. Retried from http://www.Researchgate.net/publications/316523213
- Unknown. (2014). *GIS Enterprise Geoportal project: software recommendations report.* Division of Enterprise Technology (det)
- Medolinska.K. et el. (2017). *Local GIS. Development and Assessment of the Geoportal for local governments and local communities. Case study of a small town in Poland.* MG-RSD.
- Tait. M. (2005) implementing Geoportals: application of Distributed GIS, in Computers. Environment and Urban Systems, Vol 29.pp33-47
- Mabeya .M and Wathaika .H. (2017). *Developing a geoportal for country SDI: Case study of Nairobi country.* IJSR. Retrieved from http://www.Researchgate.net/publication.
- Prasek. J, Valta. J and Hrebicek. J. (2013). *National Insipre Geoportal of the Czech Republic.* Masaryk University
- Zurbaran et el. (2nd ED). (2018). *PostGIS Cookbook.* Packt publishing Ltd.

- Nebert. D. D.(ed). (2004). Developing SDI: the SDi Cookbook. Global spatial data infrastructure. Retrieved from http://www.gsdi.org/docs

- Elizabeth T. et el. (2012). Scholars Geoportal: *A new Platform for Geoportal Data Discovery*. Exploration and access in Ontario University. Western libraries Publication. 87
- Richard T. Watson (2007). Information Systems.
- (GTplan), W. G. (2009 – 2013). Spatial Data Infrastructure (SDI) Manual for the America.

APPENDIX

SAMPLE CODE

(a) Frontend implementation

i. html

html was used to implement user interface of the system. It was designed in such a way that files programmed in html can be understood by web browsers.

login.html

```
{% extends "account/base.html" %}
{% load i18n %}
{% load bootstrap_tags %}
{% load account socialaccount %}
{% block title %}{% trans "Log in" %}{% endblock %}
{% block head_title %}{% trans "Log in" %}{% endblock %}


{% block body_outer %}
  <div class="page-header">
    <h2>{% trans "Sign in with NSCES Geoportal registered Account" %}</h2>
  </div>
<div class="row">
  <div class="col-md-8">
    {% get_providers as socialaccount_providers %}
    {% if socialaccount_providers %}
      <p>{% blocktrans with site.name as site_name %}Please sign in with one
      of your existing third party accounts. Or, <a href="{{ signup_url }}">sign up</a>
      for a {{ site_name }} account and sign in below:{% endblocktrans %}</p>
      <div class="socialaccount_ballot">
        <ul class="socialaccount_providers">
          {% include "socialaccount/snippets/provider_list.html" with process="login" %}
        </ul>
```

```
    <div class="login-or">{% trans 'or' %}</div>
  </div>
  {% include "socialaccount/snippets/login_extra.html" %}

{% else %}
  <p>{% blocktrans %}{% endblocktrans %}</p>
{% endif %}
<form role="form" method="POST" action="{% url 'account_login' %}" autocapitalize="off" {% if form.is_multipart %} enctype="multipart/form-data"{% endif %}>
  {% csrf_token %}
  <div class="text-center input-lg "><strong>{{ form.login.label_tag }}</strong></div>
  <div class="text-center input-lg"><strong>{{ form.login }}</strong></div>
  <div class="text-center input-lg"><strong>{{ form.password.label_tag }}</strong></div>
 <div class="text-center input-lg"><strong>{{ form.password }}</strong></div>
  {% if redirect_field_value %}
    <input type="hidden" name="{{ redirect_field_name }}" value="{{ redirect_field_value }}" />
  {% endif %}
  <div class="form-group">
    <div class="text-center lead"><a href="{% url 'account_reset_password' %}">{% trans "Forgot Password?" %}</a>
    </div>
  </div>
  {% if form.errors %}
  {% for field in form %}
  {% for error in field.errors %}
  <div class="alert alert-error text-danger text-center">
    <strong>{{ error|escape }}</strong>
  </div>
  {% endfor %}
  {% endfor %}
```

```
    {% for error in form.non_field_errors %}
    <div class="alert alert-error text-danger text-center">
      <strong>{{ error|escape }}</strong>
    </div>
    {% endfor %}
    {% endif %}
    <div class="text-center lead"><p><button class="btn-lg" type="submit">{% trans "Log In" %}</button></div>
  </form>
  <div class="text-center lead"><p>Do not Have an Account?</p></div>
  <div class="text-center lead"><a href="{{ signup_url }}"  class="btn btn-primary btn-lg">Create New Account</a></div>
  <strong><hr></strong>
  <p class="lead"><strong>Note:</strong> For privacy and security reasons, it is recommended to <a href="{% if SITEURL|default:""|slice:"-1:" == "/" %}{{SITEURL}}{% else %}{{SITEURL}}/{% endif %}account/logout/">{% trans "Log out" %}</a> and exit your web browser when you are done with services that need authentication in NSCES Geoportal.</p>
  </div>
</div>
{% endblock body_outer %}
```

signup.html

```
 <div class="page-header">
      <h2 class="text-center">{% trans "Sign up with NSCES Geoportal" %}</h2>
      <p class="lead text-center">When you submit the form, activation link will be sent to your registered email.</p>

  </div>
  <div class="row">
    {% get_providers as socialaccount_providers %}
    {% if socialaccount_providers %}
       <p>{% blocktrans with site.name as site_name %}Sign up with one
```

```
            of your existing third party accounts{% endblocktrans %}</p>
        {% include "socialaccount/snippets/provider_list.html" with process="signup" %}
        {% include "socialaccount/snippets/login_extra.html" %}
        <hr>
     {% endif %}
   </div>
   {% if account_geonode_local_signup %}
      <div class="row">
       <form id="signup_form" method="post" action="{% url "account_signup" %}" autocapitalize="off" {% if form.is_multipart %} enctype="multipart/form-data"{% endif %}>
           <fieldset>
              {% csrf_token %}
              <div class="form-group">
              <p>{{ form | as_bootstrap }}</p>
             </div>
              {% if redirect_field_value %}
                 <input type="hidden" name="{{ redirect_field_name }}" value="{{ redirect_field_value }}" />
              {% endif %}
              <div class="form-actions">
                <button type="reset" class="btn btn-default btn-lg">{% trans "Cancel" %}</button>
                 <button type="submit" class="btn btn-primary btn-lg">{% trans "Sign up" %}</button>
              </div>
           </fieldset>
        </form>
      </div>
```

earth.html

```
{% extends "geonode_base.html" %}
{% load static from staticfiles %}
```

```
{% load i18n %}
{% block title %} {% trans "Earth Sciences Page" %} {% endblock %}


{% block body_outer %}

<div class="page-header">
  <h1>{% trans "Earth Sciences Department" %} <a href="{% url "home" %}" class="btn btn-default pull-right">&laquo;{% trans "Back to Home" %}</a></h1>

</div>
<p class="lead">{% trans "NDATA SCHOOL OF CLIMATE AND EARTH SCIENCES (NSCES) Geoportal is Platform Under Malawi University of Science and Technology (MUST) for discovery and sharing of Geospatial Data. NSCES Geoportal allows integrated creation of data both spatial and non spatial, metadata and map visualization operations. Each dataset in the system can be shared publicly or restricted to allow access to only specific users." %}</p>

<div class="pic"><p><h1>Geology</h1></p><p class="lead">{% trans " Geology is the primary earth science. Geology deals with the composition of earth materials, earth structures and earth processes." %}</p></div></p>
<p><h3>Vegetation Cover</h3></p>
<p class="lead">{% trans "NSCES Geoportal has ability to store Geospatial Data of forestry and Plantations Areas." %}</p>
<img src="{% static "geonode/img/geology.jpg" %}"  data-holder-rendered="true" class="img-responsive" height="200" ></p>

<div class="col-md-4">
  <ul class="list-group">
    <li class="list-group-item"><p class="="lead">{% trans "Earth Sciences Dept stores Geospatial Data about: " %}</p></li>
    <li class="list-group-item"><a href="#">{% trans "Natural Hazards" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Fuels" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Minerals" %}</a></li>
```

```
    <li class="list-group-item"><a href="#">{% trans "Organisms of the planet" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Environmental Features" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Physical Features" %}</a></li>
  </ul>
</div>
<p><img  src="{% static "geonode/img/mineral.jpg" %}"  class="img-responsive" alt="" height="200"><div><p><h1>Environmental Features</h1></p>
  <p class="lead">{% trans "Malawi as country is one of nations that is blessed with many physical features; mountains, forests, water bodies, valleys, plain areas,plateaus and likes of different physical features. These physical feautures influence many sectors; agricultural, forestry, water conservation. Apart from that, they influence kinds of analysis that take place in different projects. Hence spatial data and non spatial data about different physical features is needed to go about above  processes and analysis. NSCES Geoportal provides us with spatial data that can be needed to fulfill above mentioned." %}</p>

{% endblock %}
```

climate.html

```
{% extends "geonode_base.html" %}
{% load static from staticfiles %}
{% load i18n %}
{% block title %} {% trans "Climate Sciences Page" %} {% endblock %}


{% block body_outer %}

<div class="page-header">
  <h1>{% trans "Climate Sciences Department" %}<a href="{% url "home" %}" class="btn btn-default pull-right">&laquo;{% trans "Back to Home" %}</a></h1>
</div>
```

{% trans "NDATA SCHOOL OF CLIMATE AND EARTH SCIENCES (NSCES) Geoportal is Platform Under Malawi University of Science and Technology (MUST) for discovery and sharing of Geospatial Data. NSCES Geoportal allows integrated creation of data both spatial and non spatial, metadata and map visualization operations. Each dataset in the system can be shared publicly or restricted to allow access to only specific users." %}

<h4>{% trans "NSCES Geoportal has ability to store Geospatial Data of different sector." %}</h4>
<h1>Meteorology</h1> {% trans "Meteorology is study of the atmosphere and how processes in the atmosphere dtermine Earth's weather and climate. Meteorology is a very practical science because everyone is concerned about the weather." %}
<h3>Agriculture and Food Security</h3>
{% trans "NSCES Geoportal has ability to store Geospatial Data of farming practices." %}
<h3>Vegetation Cover</h3>
{% trans "NSCES Geoportal has ability to store Geospatial Data of forestry and Plantations Areas." %}
<img src="{% static "geonode/img/mulanje_tea.jpg" %}" class="img-responsive" alt="" >

  <ul class="list-group">
    <li class="list-group-item"><h3>{% trans "NSCES Geoportal stores Geospatial Data about: " %}</h3>
    <li class="list-group-item">{% trans "Precipitation" %}
    <li class="list-group-item">{% trans "Wind Speed" %}
    <li class="list-group-item">{% trans "Solar Radiation" %}
    <li class="list-group-item">{% trans "Cloud Coverage" %}
    <li class="list-group-item">{% trans "Humidity" %}
    <li class="list-group-item">{% trans "Temperature" %}

<img src="{% static "geonode/img/climate_data.jpg" %}" class="img-responsive" alt="" > <h1>Climate Sciences</h1>
{% trans "Malawi as country is one of nations that is blessed with many physical features; mountains, forests, water bodies, valleys, plain areas,plateaus and likes of different physical features. These physical feautures influence many sectors; agricultural, forestry, water conservation. Apart from that, they influence kinds of analysis that take place in different projects. Hence spatial data and non spatial data about different physical features is needed to go about above processes and analysis. NSCES Geoportal provides us with spatial data that can be needed to fulfill above mentioned." %}

{% endblock %}
water.html

{% extends "geonode_base.html" %}
{% load static from staticfiles %}
{% load i18n %}
{% block title %} {% trans "Water Resources Page" %} {% endblock %}

{% block body_outer %}

<h1>{% trans "Water Resources Management Department" %}«{% trans "Back to Home" %}</h1>

{% trans "NDATA SCHOOL OF CLIMATE AND EARTH SCIENCES (NSCES) Geoportal is Platform Under Malawi University of Science and Technology (MUST) for discovery and sharing of Geospatial Data. NSCES Geoportal allows integrated creation of data both spatial and non spatial, metadata and map visualization operations. Each dataset in the system can be shared publicly or restricted to allow access to only specific users." %}

<h4>{% trans "MANAGING WATER QUALITY." %}</h4>

<h1>Water Resources</h1> Managing the quality of both surface water and groundwater. This is vital for sustaining ecosystems which is important factor to protect considerable diversity of water bodies.
<img src="{% static "geonode/img/water_bodies.jpeg" %}" class="img-responsive" alt="" >

<h1>Water</h1>
{% trans "Water is one of basic neccesities provided by nature. Therefore, there must be conserving ways to make it available for drinking, irrigation and other important activities. Hence, NSCES Geoportal has both spatial data and non spatial data about water bodies, boreholes, rivers, lakes, dams and other water bodies found in Malawi to help us easily access to spatial data for different analysis and projects. Land is where human beings and other animals dwell and practice different activities like agriculture, construction and many transformations." %}

  <ul class="list-group">
    <li class="list-group-item"><h3>{% trans "Water Quality Issues: " %}</h3>
    <li class="list-group-item">{% trans "Salination" %}
    <li class="list-group-item">{% trans "Nutrient Enrichment" %}
    <li class="list-group-item">{% trans "Low Dissolved Oxygen" %}
    <li class="list-group-item">{% trans "Acidification" %}
    <li class="list-group-item">{% trans "Shallow Groundwater" %}
    <li class="list-group-item">{% trans "Erosion and Sedimentation" %}

<img src="{% static "geonode/img/malawi-water_0.jpg" %}" class="img-responsive" alt="" > <h1>Water Resources Quality Improvement Plans</h1>

{% trans "Water quality improvement plans are developed to improve current water quality in the lakes, rivers, streams and water catchment areas around Malawi. These plans include water allocation plans, salinity management, water resource recovery and improved usage of groundwaters." %}

{% endblock %}

energy.html

{% extends "geonode_base.html" %}

{% load static from staticfiles %}

{% load i18n %}

{% block title %} {% trans "Energy Resources Page" %} {% endblock %}

{% block body_outer %}

<h1>{% trans "Energy Resources Management Department" %}«{% trans "Back to Home" %}</h1>

{% trans "NDATA SCHOOL OF CLIMATE AND EARTH SCIENCES (NSCES) Geoportal is Platform Under Malawi University of Science and Technology (MUST) for discovery and sharing of Geospatial Data. NSCES Geoportal allows integrated creation of data both spatial and non spatial, metadata and map visualization operations. Each dataset in the system can be shared publicly or restricted to allow access to only specific users." %}

<img src="{% static "geonode/img/hydropower.jpg" %}" class="img-responsive" alt="" >

<h1>Energy Resources</h1> Malawi has rich renewable generation potential, including hydropower, solor, wind and biomass. Those resources have not been well utilized yet, and if unlocked, could provide a diverse portfolio of resources that produce cheap, reliable services through bulk power systems.

```
<div class="col-md-4">
  <ul class="list-group">
    <li class="list-group-item"><h3>{% trans "Energy Resources issues: " %}</h3></li>
    <li class="list-group-item"><a href="#">{% trans "Hydropower" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Solar Energy" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Wind Energy" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Biomass" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Geothermal Energy" %}</a></li>
    <li class="list-group-item"><a href="#">{% trans "Thermal Energy" %}</a></li>
  </ul>
</div>
<p><img  src="{% static "geonode/img/images.jpeg" %}"  class="img-responsive" alt="" >

<p><h1>Energy Resources</h1></p>
<p class="lead">{% trans "NSCES Geoportal has ability to store Geospatial Data about Malawi Energy Resources." %}</p>
<p class="lead">{% trans "Malawi as a nation has rich renewable generation potential, that include hydropower, solar, wind, geothermal and biomass. These resources have not yet been well utilized and if unlocked, they could provide a diverse portfolio of resources that produce cheap, reliable services through numerous power systems. NSCES Geportal provides spatial data, non spatial data and maps about energy and related resources." %}</p></div></p>
{% endblock %}
```

(b) Frontend implementation

CSS files

These files were used to give style to user interface by modifying colours, arranging items in their needful place. Bootstrap which is CSS framework was used.

```
<style type="text/css">
      /* Additional styles by custom themes */
      {% if custom_theme.body_color %}
```

```
body {
  background: {{ custom_theme.body_color }};
  {% if custom_theme.body_text_color %}
  color: {{ custom_theme.body_text_color }};
  {% endif %}
}
.home #partners {
  background: {{ custom_theme.body_color }};
  padding: 2em 0;
  text-align: center;
}
{% endif %}
{% if custom_theme.navbar_color %}
.home .navbar-inverse , .navbar-inverse {
  background-color: {{ custom_theme.navbar_color }};
}
{% endif %}
{% if custom_theme.navbar_text_color %}
.navbar-inverse .navbar-nav > li > a {
  color: {{ custom_theme.navbar_text_color }};
  font-weight: 600;
  padding-top: 25px;
  padding-bottom: 25px;
}
{% endif %}
{% if custom_theme.navbar_text_hover %}
.navbar-inverse .navbar-nav > li > a:hover {
  background-color: {{ custom_theme.navbar_text_hover }};
}
{% endif %}
{% if custom_theme.navbar_text_hover_focus %}
.navbar-inverse .navbar-nav > .open > a,
.navbar-inverse .navbar-nav > .open > a:hover,
.navbar-inverse .navbar-nav > .open > a:focus{
```

```
  background-color: {{ custom_theme.navbar_text_hover_focus }};
}
{% endif %}
{% if custom_theme.navbar_dropdown_menu %}
.navbar-nav .dropdown-menu {
  background-color: {{ custom_theme.navbar_dropdown_menu }};
  border-top: 1px solid {{ custom_theme.navbar_dropdown_menu }};
}
{% endif %}
{% if custom_theme.navbar_dropdown_menu_text %}
.navbar-nav .dropdown-menu a {
  color: {{ custom_theme.navbar_dropdown_menu_text }};
}
{% endif %}
{% if custom_theme.navbar_dropdown_menu_hover %}
.navbar-nav .dropdown-menu li a:hover {
  background-color: {{ custom_theme.navbar_dropdown_menu_hover }};
  color: {{ custom_theme.navbar_dropdown_menu_text }};
}
{% endif %}
{% if custom_theme.navbar_dropdown_menu_divider %}
.navbar-nav .dropdown-menu .divider {
  background-color: {{ custom_theme.navbar_dropdown_menu_divider }};
}
{% endif %}
{% if custom_theme.logo %}
.navbar-brand {
  background-image: url({{ custom_theme.logo.url }});
  background-repeat: no-repeat;
  background-position: center;
  background-size: contain;
}
{% endif %}
{% if custom_theme.jumbotron_color %}
```

```
.home .jumbotron {
  background-color: {{ custom_theme.jumbotron_color }}
}
{% endif %}
{% if custom_theme.jumbotron_title_color %}
.home .jumbotron h1,
.home .jumbotron h2,
.home .jumbotron h3 {
  color: {{ custom_theme.jumbotron_title_color }}
}
{% endif %}
{% if custom_theme.jumbotron_text_color %}
.home .jumbotron p {
  color: {{ custom_theme.jumbotron_text_color }};
  font-weight: normal;
  font-size: xx-large;
  font-family: 'Open Sans','Helvetica Neue',Arial,sans-serif;
}
{% endif %}
{% if custom_theme.jumbotron_welcome_hide %}
.home .jumbotron .container {
  visibility: hidden;
}
{% endif %}
{% if custom_theme.jumbotron_bg %}
.home .jumbotron .container {
  position: relative;
  z-index: 1;
}
.home .jumbotron:after {
  content: "";
  background-image: url({{ custom_theme.jumbotron_bg.url }});
  background-repeat: no-repeat;
  background-position: center;
```

```
  background-size: cover;
  {% if not custom_theme.jumbotron_welcome_hide %}
  opacity: 0.5;
  {% endif %}
  top: 0;
  left: 0;
  bottom: 0;
  right: 0;
  position: absolute;
  z-index: 0;
}
{% endif %}

{% if custom_theme.search_bg_color or custom_theme.search_title_color %}
.home .big-search {
  background: {{ custom_theme.search_bg_color }};
  color: {{ custom_theme.search_title_color }};
  padding-bottom: 3em;
  padding-top: 1em;
}
.home .big-search .btn-default {
  background: transparent;
  border: none;
  color: {{ custom_theme.search_link_color }};
}
{% endif %}

{% if custom_theme.copyright_color %}
.footer-copyright {
  background-color: {{ custom_theme.copyright_color }};
}
{% endif %}
{% if custom_theme.footer_bg_color or custom_theme.footer_text_color or
custom_theme.footer_href_color %}
```

```
        footer {
          background-color: {{ custom_theme.footer_bg_color }};
          color: {{ custom_theme.footer_text_color }};
          padding: 1.5em;
        }
        footer a {
          color: {{ custom_theme.footer_href_color }};
        }
        {% endif %}
      </style>
```

(c) Frontend implementation

javascript files

Javascript was used to make web pages interactive.

```
<script>
// Add active class to the current button (highlight it)
var header = document.getElementById("myDIV");
var btns = header.getElementsByClassName("btn1");
for (var i = 0; i < btns.length; i++) {
  btns[i].addEventListener("click", function() {
  var current = document.getElementsByClassName("active");
  current[0].className = current[0].className.replace(" active", "");
  this.className += " active";
  });
}
</script>
```

(d) Back-end implementation

models.py

These files are used to implement database, they behave same way SQL behaves in implementing database only that models.py are programmed using python programming language.

```
from uuid import uuid4
```

```
import logging
from allauth.account.adapter import get_adapter
from django.conf import settings
from django.db import models
from django.db.models import signals
from django.urls import reverse
from django.contrib.sites.models import Site
from django.utils.translation import ugettext_lazy as _
from django.contrib.auth.models import AbstractUser, UserManager
from django.contrib.auth.signals import user_logged_in, user_logged_out
from taggit.managers import TaggableManager
from geonode.base.enumerations import COUNTRIES
from geonode.groups.models import GroupProfile
from allauth.account.signals import user_signed_up
from allauth.socialaccount.signals import social_account_added
from .utils import format_address
from .signals import (
    do_login,
    do_logout,
    profile_post_save,
    update_user_email_addresses,
    notify_admins_new_signup)
from .languages import LANGUAGES
from .timezones import TIMEZONES
logger = logging.getLogger(__name__)
class ProfileUserManager(UserManager):
    def get_by_natural_key(self, username):
        return self.get(username=username)
class Profile(AbstractUser):
    organization = models.CharField(
        _('Organization Name'),
        max_length=255,
        blank=True,
        null=True,
```

```
        help_text=_('name of the responsible organization'))
    profile = models.TextField(
        _('Profile'),
        null=True,
        blank=True,
        help_text=_('introduce yourself'))
    position = models.CharField(
        _('Occupation'),
        max_length=255,
        blank=True,
        null=True,
        help_text=_('role or position of the responsible person'))
    voice = models.CharField(_('Mobile Number'), max_length=255, blank=True,
null=True, help_text=_(
        'telephone number by which individuals can speak to the responsible organization
or individual'))
    fax = models.CharField(_('Facsimile'), max_length=255, blank=True, null=True,
help_text=_(
        'telephone number of a facsimile machine for the responsible organization or
individual'))
    delivery = models.CharField(
        _('Delivery Point'),
        max_length=255,
        blank=True,
        null=True,
        help_text=_('physical and email address at which the organization or individual may
be contacted'))
    city = models.CharField(
        _('District'),
        max_length=255,
        blank=True,
        null=True,
        help_text=_('city of the location'))
    area = models.CharField(
```

```
        _('Administrative Area'),
        max_length=255,
        blank=True,
        null=True,
        help_text=_('state, province of the location'))
    zipcode = models.CharField(
        _('Postal Code'),
        max_length=255,
        blank=True,
        null=True,
        help_text=_('ZIP or other postal code'))
    country = models.CharField(
        _('Country'),
        choices=COUNTRIES,
        max_length=3,
        blank=True,
        null=True,
        help_text=_('country of the physical address'))
    keywords = TaggableManager(_('keywords'), blank=True, help_text=_(
        'commonly used word(s) or formalised word(s) or phrase(s) used to describe the
subject \
            (space or comma-separated'))
    language = models.CharField(
        _("language"),
        max_length=10,
        choices=LANGUAGES,
        default=settings.LANGUAGE_CODE
    )
    timezone = models.CharField(
        _('Timezone'),
        max_length=100,
        default="",
        choices=TIMEZONES,
        blank=True,
```

```
)

def __init__(self, *args, **kwargs):
    super(Profile, self).__init__(*args, **kwargs)
    self._previous_active_state = self.is_active

def get_absolute_url(self):
    return reverse('profile_detail', args=[self.username, ])

def __str__(self):
    return str(self.username)

@staticmethod
def class_name(value):
    return value.__class__.__name__

objects = ProfileUserManager()
USERNAME_FIELD = 'username'

def group_list_public(self):
    return GroupProfile.objects.exclude(
        access="private").filter(groupmember__user=self)

def group_list_all(self):
    return GroupProfile.objects.filter(groupmember__user=self).distinct()

def is_member_of_group(self, group_slug):
    """
    Returns if the Profile belongs to a group of a given slug.
    """
    return self.groups.filter(name=group_slug).exists()

def keyword_list(self):
    """
```

```
    Returns a list of the Profile's keywords.
    """
    return [kw.name for kw in self.keywords.all()]

@property
def name_long(self):
    if self.first_name and self.last_name:
        return f'{self.first_name} {self.last_name} ({self.username})'
    elif (not self.first_name) and self.last_name:
        return f'{self.last_name} ({self.username})'
    elif self.first_name and (not self.last_name):
        return f'{self.first_name} ({self.username})'
    else:
        return self.username

@property
def full_name_or_nick(self):
    if self.first_name and self.last_name:
        return f'{self.first_name} {self.last_name}'
    else:
        return self.username

@property
def first_name_or_nick(self):
    return self.first_name if self.first_name else self.username

@property
def location(self):
    return format_address(self.delivery, self.zipcode,
                          self.city, self.area, self.country)

def save(self, *args, **kwargs):
    super(Profile, self).save(*args, **kwargs)
    self._notify_account_activated()
```

```
        self._previous_active_state = self.is_active

    def _notify_account_activated(self):
        """Notify user that its account has been activated by a staff member"""
        became_active = self.is_active and not self._previous_active_state
        if became_active and self.last_login is None:
            try:
                # send_notification(users=(self,), label="account_active")

                from invitations.adapters import get_invitations_adapter
                current_site = Site.objects.get_current()
                ctx = {
                    'username': self.username,
                    'current_site': current_site,
                    'site_name': current_site.name,
                    'email': self.email,
                    'inviter': self,
                }

                email_template = 'pinax/notifications/account_active/account_active'
                adapter = get_invitations_adapter()
                adapter.send_invitation_email(email_template, self.email, ctx)
            except Exception:
                import traceback
                traceback.print_exc()

    def send_mail(self, template_prefix, context):
        if self.email:
            get_adapter().send_mail(template_prefix, self.email, context)
def get_anonymous_user_instance(user_model):
    return user_model(pk=-1, username='AnonymousUser')
""" Connect relevant signals to their corresponding handlers. """
user_logged_in.connect(do_login)
user_logged_out.connect(do_logout)
```

```
social_account_added.connect(
    update_user_email_addresses,
    dispatch_uid=str(uuid4()),
    weak=False
)
user_signed_up.connect(
    notify_admins_new_signup,
    dispatch_uid=str(uuid4()),
    weak=False
)
signals.post_save.connect(
    profile_post_save,
    sender=settings.AUTH_USER_MODEL
)
```

(e) Back-end implementation

views.py

These files are used to define other functionalities of user html files (template files) programmed in python. They communicate with urls.py to give meaningful approach.

```
from allauth.account.views import SignupView
from django.contrib.auth import get_user_model
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from django.shortcuts import render, redirect, get_object_or_404, redirect
from django.urls import reverse
from django.utils.translation import ugettext as _
from django.contrib.sites.models import Site
from django.conf import settings
from django.http import HttpResponseForbidden
from django.db.models import Q
from geonode.tasks.tasks import send_email
from geonode.people.forms import ProfileForm
from geonode.base.auth import get_or_create_token
```

```
from geonode.people.forms import ForgotUsernameForm
from geonode.base.views import user_and_group_permission
from django.views import View
from django import forms
from allauth.account.forms import SignupForm
from dal import autocomplete
from geonode.people.forms import CustomSignupForm
from django.contrib.sites.shortcuts import get_current_site
from django.utils.encoding import force_bytes, force_text
from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
from django.template.loader import render_to_string
from .tokens import account_activation_token
from django.contrib.auth.models import User
from django.core.mail import EmailMessage
from django.contrib import messages
from .forms import SignupForm
class SetUserLayerPermission(View):
    def get(self, request):
        return user_and_group_permission(request, 'profile')
    def post(self, request):
        return user_and_group_permission(request, 'profile')
class CustomSignupView(SignupView):
    def get_context_data(self, **kwargs):
        ret = super(CustomSignupView, self).get_context_data(**kwargs)
        ret.update({'account_geonode_local_signup':
settings.SOCIALACCOUNT_WITH_GEONODE_LOCAL_SINGUP})
        return ret
def signup(request):
    if request.method == 'POST':
        form = SignupForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.is_active = False
            user.save()
```

```
            current_site = get_current_site(request)
            mail_subject = 'Activate your NSCES Geoportal account.'
            message = render_to_string('people/acc_active_email.html', {
                'user': user,
                'domain': current_site.domain,
                'uid':urlsafe_base64_encode(force_bytes(user.pk)),
                'token':account_activation_token.make_token(user),
            })
            to_email = form.cleaned_data.get('email')
            email = EmailMessage(
                        mail_subject, message, to=[to_email]
            )
            email.send()
            return redirect('people/account_confirm')
    else:
        form = SignupForm(instance=profile)
    return render(request, 'people/signup.html', {'form': form})
#activate email to login
def activate(request, uidb64, token):
    try:
        uid = force_text(urlsafe_base64_decode(uidb64))
        user = User.objects.get(pk=uid)
    except(TypeError, ValueError, OverflowError, User.DoesNotExist):
        user = None
    if user is not None and account_activation_token.check_token(user, token):
        user.is_active = True
        user.save()
        login(request, user)
        return redirect('account_login')
        #return HttpResponse('Thank you for your email confirmation. Now you can login
your account.')
    else:
        return HttpResponse('Activation link is invalid!')
#edit profile after logged in
```

```
@login_required
def profile_edit(request, username=None):
    if username is None:
        try:
            profile = request.user
            username = profile.username
        except get_user_model().DoesNotExist:
            return redirect("profile_browse")
    else:
        profile = get_object_or_404(get_user_model(), Q(is_active=True), username=username)
    if username == request.user.username or request.user.is_superuser:
        if request.method == "POST":
            form = ProfileForm(request.POST, request.FILES, instance=profile)
            if form.is_valid():
                form.save()
                messages.success(request, (f"Profile {username} updated."))
                return redirect(
                    reverse(
                        'profile_detail',
                        args=[
                            username]))
        else:
            form = ProfileForm(instance=profile)
        return render(request, "people/profile_edit.html", {
            "profile": profile,
            "form": form,
        })
    else:
        return HttpResponseForbidden(
            'You are not allowed to edit other users profile')
def profile_detail(request, username):
    profile = get_object_or_404(get_user_model(), Q(is_active=True), username=username)
```

```
    # combined queryset from each model content type
    access_token = None
    if request and request.user:
        access_token = get_or_create_token(request.user)
        if access_token and not access_token.is_expired():
            access_token = access_token.token
        else:
            access_token = None
    return render(request, "people/profile_detail.html", {
        'access_token': access_token,
        "profile": profile,
    })


def forgot_username(request):
    """ Look up a username based on an email address, and send an email
    containing the username if found"""
    username_form = ForgotUsernameForm()
    message = ''
    site = Site.objects.get_current()
    email_subject = _("Your username for " + site.name)
    if request.method == 'POST':
        username_form = ForgotUsernameForm(request.POST)
        if username_form.is_valid():
            users = get_user_model().objects.filter(
                email=username_form.cleaned_data['email'])
            if users:
                username = users[0].username
                email_message = email_subject + " : " + username
                send_email(email_subject, email_message, settings.DEFAULT_FROM_EMAIL,
                           [username_form.cleaned_data['email']], fail_silently=False)
                message = _("Your username has been emailed to you.")
            else:
                message = _("No user could be found with that email address.")
```

```
    return render(request, 'people/forgot_username_form.html', context={
        'message': message,
        'form': username_form
    })
```

(f) Back-end implementation

urls.py

These files are also written in python that are used to be link between views.py files and template files written in html to so that they might communicate.

```
from django.conf.urls import url
from django.views.generic import TemplateView
from .views import ProfileAutocomplete, SetUserLayerPermission
from . import views
from django.urls import path


urlpatterns = [  # 'geonode.people.views',
    url(r'^$', TemplateView.as_view(template_name='people/profile_list.html'),
        name='profile_browse'),
    url(r"^edit/$", views.profile_edit, name="profile_edit"),
    url(r"^edit/(?P<username>[^/]*)$",
        views.profile_edit, name="profile_edit"),
    url(r"^profile/(?P<username>[^/]*)/$",
        views.profile_detail, name="profile_detail"),
    url(r'^forgotname', views.forgot_username, name='forgot_username'),
    url(r'^autocomplete/$',
        ProfileAutocomplete.as_view(), name='autocomplete_profile'),
    url(r'^layer/permission/$',
        SetUserLayerPermission.as_view(), name='set_user_layer_permissions'),
]
```

(g) Back-end implementation

settings.py

Any web application written in django web framework which is written in python must have settings.py file. This file is used to make web application working because it defines

apps used, languages to be used in web application, email configurations, changing permissions, database configuration.

```
import os
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = ''

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'contactus',
    'forum',
    'pages',
    'account',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
```

```
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'geoportal.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'geoportal.wsgi.application'
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_URL = '/static/'
#email settings
#set  https://myaccount.google.com/lesssecureapps
#set https://accounts.google.com/DisplayUnlockCaptcha
```

```
#set https://myaccount.google.com/apppasswords
EMAIL_BACKEND='django.core.mail.backends.smtp.EmailBackend'
#EMAIL_HOST="smtp.gmail.com"
#EMAIL_PORT=587
#EMAIL_HOST_USER="kachingwebilly@gmail.com"
#EMAIL_HOST_PASSWORD=
#EMAIL_USE_TLS=True
#EMAIL_USE_SSL=False
#make sure password is sucured.

#email activation setup
EMAIL_HOST="smtp.gmail.com"
EMAIL_HOST_USER="kachingwebilly@gmail.com"
EMAIL_USE_TLS=True
EMAIL_PORT=587
EMAIL_HOST_PASSWORD=""
DEFAULT_FROM_EMAIL = "kachingwebilly@gmail.com"
```

(h) Back-end implementation

admin.py

This file is used to make apps created in website that has been programmed using Django web framework to be registered to admin panel that is created by default the moment django project is being created. It is also programmed using python programming language.

```
from django.conf import settings
from django.conf.urls import url
from django.contrib import admin
from django.contrib import messages
from django.contrib.auth import get_user_model
from django.contrib.auth.forms import AdminPasswordChangeForm
from django.utils.translation import ugettext_lazy as _
from django.utils.decorators import method_decorator
from django.views.decorators.csrf import csrf_protect
```

```
from django.db import transaction
from django.views.decorators.debug import sensitive_post_parameters
from django.contrib.admin.options import IS_POPUP_VAR
from django.shortcuts import get_object_or_404
from django.utils.html import escape
from django.template.response import TemplateResponse
from django.http import HttpResponseRedirect, Http404
from django.core.exceptions import PermissionDenied
from django.forms import modelform_factory
from geonode.base.admin import set_user_and_group_layer_permission
from .models import Profile
from .forms import ProfileCreationForm, ProfileChangeForm


csrf_protect_m = method_decorator(csrf_protect)
sensitive_post_parameters_m = method_decorator(sensitive_post_parameters())


class ProfileAdmin(admin.ModelAdmin):
    modelform_factory(get_user_model(), fields='__all__')
    add_form_template = 'admin/auth/user/add_form.html'
    change_user_password_template = None
    fieldsets = (
        (None, {'fields': ('username', 'password')}),
        (_('Personal info'), {'fields': ('first_name', 'last_name', 'email')}),
        (_('Permissions'), {'fields': ('is_active', 'is_staff', 'is_superuser',
                                       'groups')}),
        (_('Important dates'), {'fields': ('last_login', 'date_joined')}),
        (_('Extended profile'), {'fields': ('organization', 'profile',
                                            'position', 'voice', 'fax',
                                            'delivery', 'city', 'area',
                                            'zipcode', 'country',
                                            'keywords')}),
    )
```

```
add_fieldsets = (
    (None, {
        'classes': ('wide',),
        'fields': ('username', 'password1', 'password2')}
     ),
)
form = ProfileChangeForm
add_form = ProfileCreationForm
change_password_form = AdminPasswordChangeForm
list_display = (
    'id', 'username', 'organization',
    'email', 'first_name', 'last_name',
    'is_staff', 'is_active')
list_filter = ('is_staff', 'is_superuser', 'is_active', 'groups')
search_fields = (
    'username', 'organization', 'profile',
    'first_name', 'last_name', 'email')
readonly_fields = ("groups", )
ordering = ('username',)
filter_horizontal = ('groups', 'user_permissions',)
actions = [set_user_and_group_layer_permission]
def get_fieldsets(self, request, obj=None):
    if not obj:
        return self.add_fieldsets
    return super(ProfileAdmin, self).get_fieldsets(request, obj)
def get_form(self, request, obj=None, **kwargs):
    """
    Use special form during user creation
    """
    defaults = {}
    if obj is None:
        defaults.update({
            'form': self.add_form,
            'fields': admin.utils.flatten_fieldsets(self.add_fieldsets),
```

```
        })
    defaults.update(kwargs)
    return super(ProfileAdmin, self).get_form(request, obj, **defaults)
def get_urls(self):
    return [  # ",
        url(r'^(\d+)/password/$',
            self.admin_site.admin_view(self.user_change_password))
    ] + super(ProfileAdmin, self).get_urls()

def lookup_allowed(self, lookup, value):
    # See #20078: we don't want to allow any lookups involving passwords.
    if lookup.startswith('password'):
        return False
    return super(ProfileAdmin, self).lookup_allowed(lookup, value)

@sensitive_post_parameters_m
@csrf_protect_m
@transaction.atomic
def add_view(self, request, form_url='', extra_context=None):
    if not self.has_change_permission(request):
        if self.has_add_permission(request) and settings.DEBUG:
            # Raise Http404 in debug mode so that the user gets a helpful
            # error message.
            raise Http404(
                'Your user does not have the "Change user" permission. In '
                'order to add users, Django requires that your user '
                'account have both the "Add user" and "Change user" '
                'permissions set.')
        raise PermissionDenied
    if extra_context is None:
        extra_context = {}
    username_field = self.model._meta.get_field(self.model.USERNAME_FIELD)
    defaults = {
        'auto_populated_fields': (),
```

```
            'username_help_text': username_field.help_text,
        }
        extra_context.update(defaults)
        return super(ProfileAdmin, self).add_view(request, form_url,
                                                  extra_context)
    @sensitive_post_parameters_m
    def user_change_password(self, request, id, form_url=''):
        if not self.has_change_permission(request):
            raise PermissionDenied
        user = get_object_or_404(self.get_queryset(request), pk=id)
        if request.method == 'POST':
            form = self.change_password_form(user, request.POST)
            if form.is_valid():
                form.save()
                change_message = self.construct_change_message(
                    request,
                    form,
                    None)
                self.log_change(request, user, change_message)
                msg = _('Password changed successfully.')
                messages.success(request, msg)
                return HttpResponseRedirect('..')
        else:
            form = self.change_password_form(user)

        fieldsets = [(None, {'fields': list(form.base_fields)})]
        adminForm = admin.helpers.AdminForm(form, fieldsets, {})
        context = {
            'title': _('Change password: %s') % escape(user.get_username()),
            'adminForm': adminForm,
            'form_url': form_url,
            'form': form,
            'is_popup': IS_POPUP_VAR in request.GET,
            'add': True,
```

```
            'change': False,
            'has_delete_permission': False,
            'has_change_permission': True,
            'has_absolute_url': False,
            'opts': self.model._meta,
            'original': user,
            'save_as': False,
            'show_save': True,
        }
        return TemplateResponse(request,
                                self.change_user_password_template or
                                'admin/auth/user/change_password.html',
                                context)  # , using=self.admin_site.name)
    def response_add(self, request, obj, post_url_continue=None):

        if '_addanother' not in request.POST and IS_POPUP_VAR not in request.POST:
            mutable = request.POST._mutable
            request.POST._mutable = True
            request.POST['_continue'] = 1
            request.POST._mutable = mutable
        return super(ProfileAdmin, self).response_add(request, obj,
                                                      post_url_continue)
admin.site.register(Profile, ProfileAdmin)
```

(i) Back-end implementation

wsgi.py

This file is used by development server. Websites programmed using Django web framework can be run at localhost, therefore, wsgi.py written in python programming language is used to load development server at localhost.

```
"""

It exposes the WSGI callable as a module-level variable named ``application``.
For more information on this file, see
```

```
https://docs.djangoproject.com/en/2.2/howto/deployment/wsgi/
"""
import os
from django.core.wsgi import get_wsgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'geoportal.settings')
application = get_wsgi_application()
```