

A GIS BASED ELECTRICTY DISTRIBUTION NETWORK AND FAULT TRACKING SYSTEM

MADALITSO KACHERE

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

DECLARATION

I, Madalitso Kachere declare that this research project represents my own work and that it has never been submitted to this institution or any other institution for any academic purpose. All sources of information have been appropriately acknowledged.

Signature:

Date: 12th July, 2021

DISSERTATION APPROVED BY:

SUPERVISOR:

Dr. Chikondi Chisenga Signature: _____

ASSESSOR:

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 acknowledgment.

Contents

DECLARATION 2

DISCLAIMER 4

COPYRIGHT STATEMENT: 5

List of Figures 8

LIST OF TABLES 9

ABSTRACT 10

ACKNOWLEDGEMENT 11

TERMINOLOGY DEFINITIONS 12

1. INTRODUCTION 15

1.1. Background 15

1.2. Problem statement 16

1.3. Main and Specific Objectives 18

1.3.1. Main Objective 18

1.3.2. Specific Objectives 19

1.4. Research questions 19

2. METHODOLOGY 20

3. REQUIREMENT SPECIFICATIONS 20

3.1. Introduction 20

3.2. User requirements definition 20

3.2.1. Functional requirements 20

3.2.2. Non-functional requirements 21

3.3. Use Case Analysis for a GIS web-based electricity management and fault tracking system 22

4. SYSTEM DESIGN 23

4.1. System Models and Architecture 23

4.1.1. Overview of the system and its subsystems explained 24

4.2. System Functional Requirements Definition 24

4.3. PRELIMINARY DESIGN 26
4.3.1. Design solution description 26
4.3.2. Process Model 27
4.3.3. Database Design Model 30
4.3.4. Database physical model 31
4.3.5. System's Subsystems 32
D. Visualization Subsystem 33
5. DETAILED / ARCHITECTURAL DESIGN 35
5.1. Overall Description of the Design Solution 35
5.2. System design 35
5.2.1. Welcome Page 35
5.2.2. Register Account Page 36
5.2.3. Login Page 39
5.2.4. Data Visualization Page 45
5.2.5. Login Page 49
5.2.6. Dashboard Page 51
5.2.7. Fault List Page 52
5.3. System Infrastructure 57
5.3.1. Overview 57
6. RESULTS AND DISCUSSIONS 60
6.1. DATA COLLECTED 60
6.2. Discussion 61
7. CONCLUSION AND RECOMMENDATIONS 63
8. REFERENCE 65
9. SYSTEM CODE. 66

List of Figures

Figure 1. Showing a Use case. 22
Figure 2. Showing the overview of the system and its subsystems. 23
Figure 3: Context Diagram 27
Figure 4: Level 0 DFD 28
Figure 5: Level 1 DFD 29
Figure 6. Showing the System's ERD 30
Figure 7. Shows a physical model of a database used in this system 31
Figure 8. Showing a User Log in subsystem Data Flow Diagram 32
Figure 9. Shows a Welcome Page 36
Figure 10. Shows a Registration Page 37
Figure 11. Shows a Log in page. 40
Figure 12. Showing a Dashboard 42
Figure 13. Shows fault reporting page 44
Figure 14. Shows an Administrator Welcome Page 46
Figure 15. shows the interface design of the admin registration page 47
Figure 16. Shows an Admin Log in page 50
Figure 17. Shows a dashboard for am admin 52
Figure 18. Shows a fault list page. 54
Figure 19. Shows a fault map 55
Figure 20. Shows a Fault map functionalities, with a pop-up. 56

LIST OF TABLES

Table 1. Showing how the facilities with facility ID and their location details and also poles connected to each facility. In this case, facilities with meter number 001 and 002 are connected to pole with pole ID, P1 while the other facilities are connected to P2....60
Table 2. Showing how the poles are connected to a specific transformer. Pole with pole_id P1 is connected from transformer with transformer_id T1. Pole. Pole with pole_id P2 is connected from transformer with transformer_id T2. The location details (latitude and longitudes) are for the poles. 60
Table 3. Shows location details of transformers.....62

ABSTRACT

The use of GIS in power system has greatly enhanced the efficiency in energy sector. With the prevalence of computer and development of information technology, Geographic Information Systems (GIS) have long used for a variety of applications in distribution network system. GIS are designed to support the analysis, management, manipulation and mapping of spatial data. The distributions of electrical energy to end users in most urban area is faced with divers' spatial problems particularly with the use of analogue system. In Malawi, there is no GIS based electricity distribution network and fault tracking system with respect to spatial data, making it too difficult for the electricity utility company to locate their electricity facilities in-terms of fault management. The development and usage of Web applications is one of the most popular directions of research and development in the field of software engineering. Web applications can be of great importance for enterprise information system (IS) development in the field of software engineering. This paper was entitled to design a geospatial database containing electricity distribution data in the study area, to develop a GIS-based web map for visualization and also to create a fault tracking mechanism in the study area to meet the demands of the customers in-terms of fault locating management. All the information was presented on a web GIS platform inform of a website, in this case, the user (customer) and system admin are interlinked in a systematic way to meet the demands of the system users. The system was modeled using waterfall software model development. The system was developed using Node JS, a software that uses JavaScript (JS) to develop web servers, PostgreSQL was used to create the database and it also acted as a database server, to visualize data on the web map, Open layers were used for the development of web map for visualization. At the end of the project, a successful GIS-based electricity management system was developed, whereby users are allowed to create an account, log in and be able to report a fault. The system allows administrators to view the report in the database and view the reported fault on the web map for easy management.

ACKNOWLEDGEMENT

Firstly, I would like to thank The Almighty Lord, for granting me this research idea and for allowing me to do it.

Secondly, I would like to express my special appreciation and thanks to my supervisor Dr. C. Chisenga, you have been a tremendous mentor for me. I would like to thank you for encouraging me go on with my research and also, for allowing me to believe in myself. Your advice on this research have been invaluable.

I would also like to thank my friends, Bryan Cacti_Nuts, bro you have been my mentor since day one and I owe you a lot, Timothy, your encouragement on me over this topic was fire chief. And to you George, I appreciate a lot. Mesho_Guy, Nu_T, Tron, Francis, Ronnie, Madalo Banda and Mike Kachembwe you are really an inspiration. I also want to thank you for your brilliant comments and suggestions, thanks to y'all friends.

A special thanks to my family. Words cannot express how grateful I am to my mother, all my sisters, cousins, all the sacrifices that you've made on my behalf. Your prayers for me were what sustained me thus far. And also, your financial help had me focused to continue this work. I owe you a lot FAM.

I would also like to thank my beloved girlfriend Taniel Tumpale Nkosi. Thank you for your support, you have been there for me always and especially I can't thank you enough for encouraging me not to give up when I was stuck throughout this experience.

Finally, I thank my God, my good Father, for letting me through all the difficulties. I have experienced Your guidance day by day. I will keep on trusting You for my future. Thank you, Lord.

TERMINOLOGY DEFINITIONS

Geographic Information System (GIS)

It is a technology used for capturing, storing, querying, analyzing, and displaying geographic data.

Web-Based System

It is an application that is accessed via Hypertext Transfer Protocol (HTTP). It is designed to run on the web browser

HTTP

It is a standard data protocol for transferring data from a web server to the web browsers.

JavaScript (JS)

It is a lightweight interpreted programming language designed for creating network-centric applications. It is complementary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML (HyperText Mark-up Language). It is open and cross-platform.

It is a scripting language used to create and control dynamic website content.

Application Programming Interface (API)

It is a set of programming code (extension or script or link) that enables data transmission between one software product and another. It also contains the terms of this data exchange.

API Calls

It is a process that takes place when you send a request after setting up your API with the correct endpoints. The information is transferred, processed, and feedback is returned.

Passport

It is authentication middleware for Node JS. It is developed to serve to authenticate the request

Popup

It is a small window graphical user interface (GUI) that display in the foreground of the visual interface.

Node Package Manager (NPM):

It is a Package manager for the Node JavaScript platform that puts modules in place so that node can find them, and manages dependency conflicts intelligently. It is extremely configurable to support a wide variety of use cases.

Extended JavaScript (EJS)

It is a simple templating language that allow you to generate HTML markup with JavaScript

HTML

It is the World Wide Web's core markup language used to organize word content on the web

GeoJSON

It is a Web Feature Service format for encoding different types of geographic data structures.

PostgreSQL

It is the object-relational database management system used to store spatial data

Server: a physical computer that runs without any interruption so that your website is available all the time for anyone who wants to see it

CSS (Cascading Styling Sheet)

It is a language used for describing the presentation written HTML document

It is also used for styling a web page.

Mongo Database

This is an open-source document database and leading NoSQL database. MongoDB is written in C++.

Object Relation Mapping (ORM)

This is a technique that allows us to query and change data from the database in an object-oriented way.

1. INTRODUCTION

1.1. Background

The use of electricity as a tool for socio-economic development of the nation cannot be overemphasized. Hence, there is a need to keep a comprehensive and accurate inventory of their physical assets, spatial location, both as part of normal service provision, extending the network and undertaking maintenance by the use of geospatial technique. A geographic information system (GIS) is defined as a computer system for capturing, storing, querying, analyzing, and displaying geospatial data (Chang, 2019). It originates from the science of geography and it integrates many types of data. A GIS can store this data by use of databases. A database is a repository capable of storing large amounts of data (de By, et al., 2001). GIS analyzes spatial location and organizes layers of information into visualizations using maps. With this unique capability, GIS reveals deeper insights into data, such as patterns and relationships between phenomena thereby helping different users make smarter decision. Geographic Information System (GIS) is one of typical information systems used in the electrical power supply companies (Kale and Lad, 2006). Therefore, there is need for electricity network to be organized. Organized electrical distribution system plays a role in proper management and maintenance of demand and supply and various electrical distributions are functioning in the world (Shaw et al., 2010; Erdogdu, 2019) and this idea was implemented and used to develop a system that was proposed in my research. Sustainable development is fueled by the energy sector and with the development of a GIS system, the reasons for power outage will purely be the responsibility of the concerned authority.

The development and usage of Web applications is one of the most popular directions of research and development in the field of software engineering. Web applications can be of great importance for enterprise information system (IS) development. Since the advancement of internet and geospatial

technologies, various efforts have been made to generate information and utilization of these techniques for the well-being of society. Today in the advancement of Web technology, various management systems have been developed for utility mapping, power distribution, Earth data and Planetary data dissemination and archival. Web GIS is also important in decision making process such as urban planning process (Mansourian et al., 2011), The above-mentioned applications highlighted a variety of specific GIS applications within different management system, hence forth, the proposed system made use of web characteristics to locate and visualize the reported fault in the system by a customer for quick management.

Efficient functioning of the transmitted power cannot be achieved without proper record keeping and monitoring of the distribution network system. Any organizations that expect to run efficient day-to-day operation and to manage and develop its service effectively must know what asset it has, where they are, their condition and how they are performing (Pickering, 1993). The knowledge about physical assets of the electricity distribution network is necessary to make strategic and operational decision. Hence to make such informed decision regarding the distribution and management of electricity and its facilities, information must be collected and analyzed to its full extent through GIS technique.

1.2. Problem statement

The distributions of electrical energy to end users in most urban areas is faced with divers' spatial problems particularly with the use of analogue system. In Malawi, there is no GIS based electricity distribution management system with respect to spatial data. According to Bill Meehan (2013a), the challenges faced by power industries and utility operators are many. These challenges are exponentially growing and will grow due to high rate of electricity usage in urbanized areas and due to the recent advancement in renewable energy technologies such as solar power, wind power and so on. With the advancement of such technology, the existing system models

may not be able to withstand the pressure and it is so delicate. For example, if an outage fault happens because of a tree falling over power lines during a windy rainfall in a remote location, it is hard to locate exactly where the problem is and if the remote location is hard to access, it becomes even more difficult to resolve the problem in a short period of time as per every customer's wish. With such challenges ahead, foresees a system to answer the questions such as where does the reported problem exist. Thus, GIS comes into play to answer such questions by integrating all the systems involved in a power industry. This is exactly why GIS is integrated with Power Systems not to resolve the problem directly but to have better overview of where and what is happening.

Poor service delivery by Malawi's electricity utility, ESCOM, is a major blow to their customers since it takes a long time to locate the fault geospatially and be managed. Due to this problem, this study was meant to improve the understanding and networking of electricity power distribution towards effective power distribution through a GIS web map. Most importantly, the power industry is expected to keep track of the electrical facilities (transformers, poles, and power lines etc.) involved in the distribution of energy to the end-users (customers). With the aid of GIS, variety of information can be better organized on a computer system linking the database to an output map. Emengini (2004) noted that knowledge about physical assets of the enterprise is necessary to make strategic and operation decisions. Thus, to take wise decisions vital to the operations, growth and management of electricity distribution facilities, information must be collected and analyzed to its full extent, such information contributes not only to efficient services, but also to the operation and maintenance of assets, and to the sensible planning of extensions and new works.

According to recent studies by Gao et al., (2009), they proposed that, GIS systems is made up of a software, hardware, modeling, database management and display as visualization for decision making. Based on that GIS systems can also be used in various ways, one of them being will be used to display the proposed research of electrical distribution network on a web map and being able to locate and manage

faults efficiently. To run a distribution system efficiently for providing a reliable service, it is required to manage geographic information, which can help engineers to operate the system as per the requirement (Vader and Kulkarni, 2006), also based on this explanation, electrical engineers can also use the same idea to manage electrical distribution network. GIS is one of the most important new technologies in power utilities which consider growth opportunities for fault analysis, optimization of networks, load forecasting, cost estimation and selection of suitable areas and etc.

High customer demands force different kinds of technologies and skills to be banked upon by utility companies in order to enhance the level of production. For this reason, GIS has become in effect, one of the tools for generating, arranging and managing geospatial information. Sipes (2007) shows a transition in recent years from the ways in which utility companies which have previously been engaging GIS as departmental solution to an enterprise solution where data is accessible by thousands of people. Korte (2001) demonstrates that GIS can be employed to some extend in network tracking to determine a particular path or route through a network. This however supports the fact that GIS could be employed in agencies that regulate networked services. Korte's idea was so great and therefore it was also used as an important tool in distribution power industries to determine the power distribution capacity of the system along the way or perhaps to determine the most favorable path alone a straight network. Henceforth Korte lacked to implement his idea in-terms of spatial location among the electrical distribution network to describe and locate exactly on where the facilities are and this lacked idea was implemented and used in my research.

1.3. Main and Specific Objectives

1.3.1. Main Objective

Was to develop a GIS-based system for effective management of electricity distribution as well as fault management.

1.3.2. Specific Objectives

- Was to create a geospatial database containing electricity distribution data.
- Was to create a GIS web map based for fault management and visualization

1.4. Research questions

- Can a geospatial database help to keep track of electricity distribution?
- Can a GIS web map-based help monitor, locate and visualize faults? (fault management).

1.5. Significance of study.

Over the past years, Malawi has been failing to adapt technological innovations, but I can not necessarily mean that the innovations are being in use as of now. GIS is one the most advanced technology that can be innovative to every sustainable department in Malawi. This paper incorporated the use of GIS in Electricity management so as to fill in the needed gap between customers and suppliers in terms of fault reporting and quick management.

2. METHODOLOGY

The methodology used in this research uses system development taking into account the waterfall model. The waterfall model is breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. In this paper, the system's development used the following stages; requirements specifications, design and implementation. These have been included in this document as they have been explained in details as separate chapters. They fall in that order respectively as the research project used a waterfall model in developing the system.

3. REQUIREMENT SPECIFICATIONS

3.1. Introduction

Requirement specification is the collection of the set of all requirements that are to be on the design and verification of the product. It is also the process of writing down the user and system requirements. For efficient management of electricity to end users, there is need for system development that should be able to locate how the electricity is distributed to end users for efficient and fast management.

A GIS- web based electricity distribution and fault tracking system will be developed, and this system is required to allow only registered users with the system to report faults and the admin will be able to view the query in the database and visualize on a web map and locate the fault per location of the facility.

3.2. User requirements definition

It specifies what the user needs to do with the system.

3.2.1. Functional requirements

It describes what the system should do. The proposed system should do the following:

- Enter data into the database.
- Update data in the data base.

- Register a customer in the system
- Log in
- Report faults by customers
- Visualize faults on a web map

3.2.2. Non-functional requirements

It describes how the system works. The proposed system will work depending on the following requirements:

a) Usability specifications

A GIS web-based electricity management and fault tracking system has a user friendly graphical user interface for easy navigation. The system also includes a help section where new users can learn how the system is all about.

b) Efficiency requirements

This is the extent to which the system handles capacity, throughput, and response time. For any user to access the proposed system, there is need for the user to have a gadget that has the following:

- Processor clock speed: good.
- Screen resolution: at least a resolution of 1080 by 720 pixels.
- Internet browser support.
- Internet connection
- At least 2 GB Ram

c) Reliability Requirements

The proposed system will be always available when needed and only available when the servers are ON.

d) Security

The authentication measures with the password will be created for all users of the system for reporting faults, one user, one email, one meter number.

3.3. Use Case Analysis for a GIS web-based electricity management and fault tracking system

Use case describes how users will interact with the system.

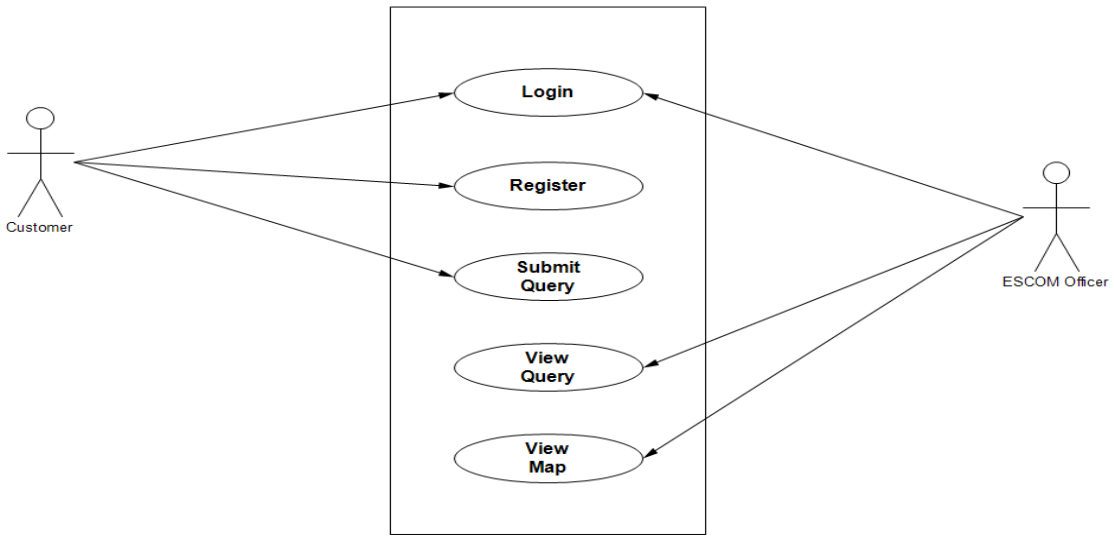


Figure 1. Showing a Use case.

In this case, a customer will register with the system first, then login in to submit the fault report. The system administrator, (ESCOM officer) will also login into the system, view the reported fault inform of a query in the database and visualize the fault on a web map depending on the attributes of the query.

4. SYSTEM DESIGN

4.1. System Models and Architecture

The proposed research termed "A GIS web-based electricity management and fault tracking system" has different functionalities and subsystems that provide different services to the user. These subsystems work as one entity to produce an operational GIS web-based electricity management and fault tracking system and meet the needs of the users. These separate entities are the user account management subsystem and data management subsystem which are further divided below:

- Login system
- Data Entry system
- Authentication
- Data visualization system

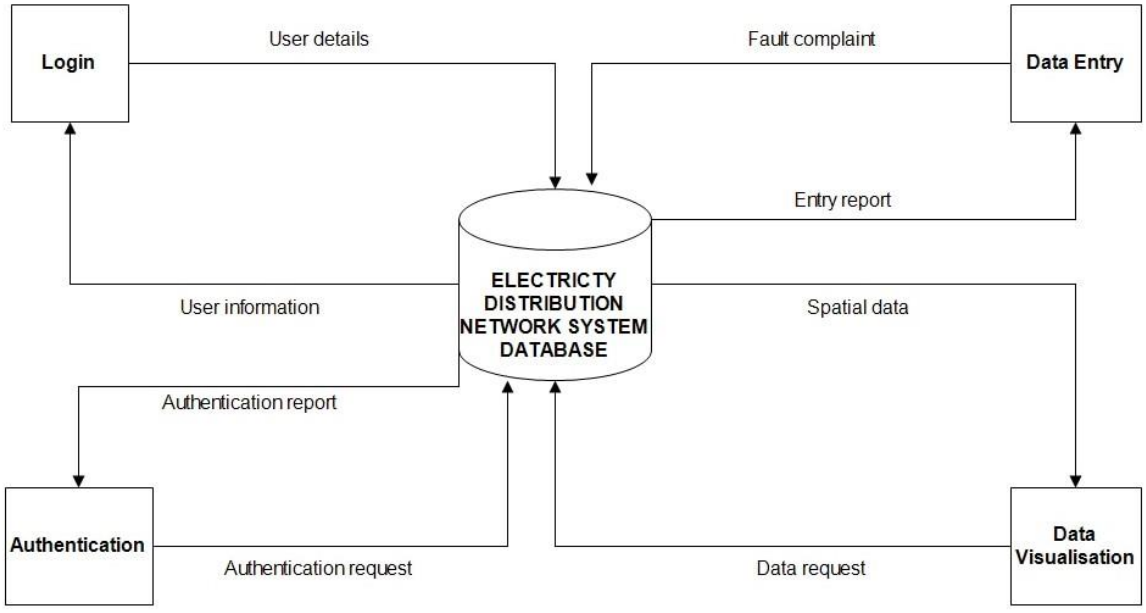


Figure 2. Showing the overview of the system and its subsystems.

4.1.1. Overview of the system and its subsystems explained

A. Login Subsystem

In GIS web-based electricity management and fault tracking system the main users are the customers and the system admins (officers). However, to ensure integrity and security of the system, the system will have a login subsystem that will enable only registered users to log into the system. The following are the functionalities of this subsystem:

- User account creation
- Registered Users Logging into system

B. Data Entry Subsystem

A system cannot exist without data moving to its processes. So, a GIS web-based electricity management and fault tracking system has this subsystem mainly aimed at the administrators (data clerks) to enter the data collected on the ground and submit it into the system through the database for storage.

C. Data Visualization Subsystem

This is one of the core subsystems in the GIS web-based electricity management and fault tracking system. Since, data needs to be visualized on a web map. The main users of this subsystem are the system admin.

D. Authentication Subsystem

For security concerns, this subsystem aims at securing the system from external users and attacks. This helps to avoid unauthorized users from using system resources without any permission.

4.2. System Functional Requirements Definition

1. Authentication

The system should provide authentication to avoid unauthorized use and inaccurate data entry into the system by intruders during registration.

2. Database Transactions

A. Data-entry

The system should be able to receive every entry submitted by data-entry administrators every day. These entries upon processed will be entered into the database. The system should be able to handle many data entry requests all at once.

B. User Retrieval

The system should be able to connect to the database containing the user information, faults and send this information where it has been requested for visualization on a web map.

C. Fault Data Retrieval

The system should be able to query the database and retrieve all or specific information from the database and send it to the back-end in a meaningful format such as points on a map to be visualized by the administrator.

3. Visualize spatial data

The system, per requirement, should be able to query data from the database and retrieve the data (information) and manipulate the information into GeoJSON format which can be visualized on a web map page as reported faults.

4.3. PRELIMINARY DESIGN

4.3.1. Design solution description

The GIS web-based electricity management and fault tracking system has several components, these components will work hand in as an integrated system to provide the user's needs. The architectural design of the system includes the following four components: User Login, Authentication, Data entry, and Visualization. Besides these components, the system will also have a Database developed in the PostgreSQL 13 server responsible for storing all the data flowing throughout the system.

The User Login component has the ability to allow a user, in this case a customer as well as the system administrator to login into the system or create a new account to be used to access the system. The User Login component will also have an interface with the Database management system to enable data creation, storage, and retrieval.

The Authentication component will check the system's requirements and make sure that only authenticated users have access into the system and that the system's contents are protected and also, non-authenticated users should be redirected to a login or register page. There will also be an interface between the Authentication component and the Database management system whereby when a user is accessing and navigating through the system, the authentication component should check within the authorized users in the database.

The Visualization component will allow logged-in users to visualize the spatial data in terms of faults reported by several users (customers) on a map as points and be able to locate the reported fault based on location. The component will also have an interface with the database so that the component can query and retrieve data from the database about the location of the reported fault.

The Data Entry component as per systems' design will allow the back-end users to enter the spatial data (electrical identity and location) collected within their distribution network

into the system. The component will also have an interface to the database that will be able to send the data to the database for storage.

The interconnections between these system's components have been illustrated in the System Models section of this document. The software specification and explanations of how the individual parts of the design solution satisfy the user and system requirements have also been presented in the subsequent sections.

4.3.2. Process Model

The above sub-systems contain several individual processes, but these processes communicate with each other within the entire system. Below is an illustration of how data will flow between different processes within the system. The illustration is in form of the following Data Flow Diagrams:

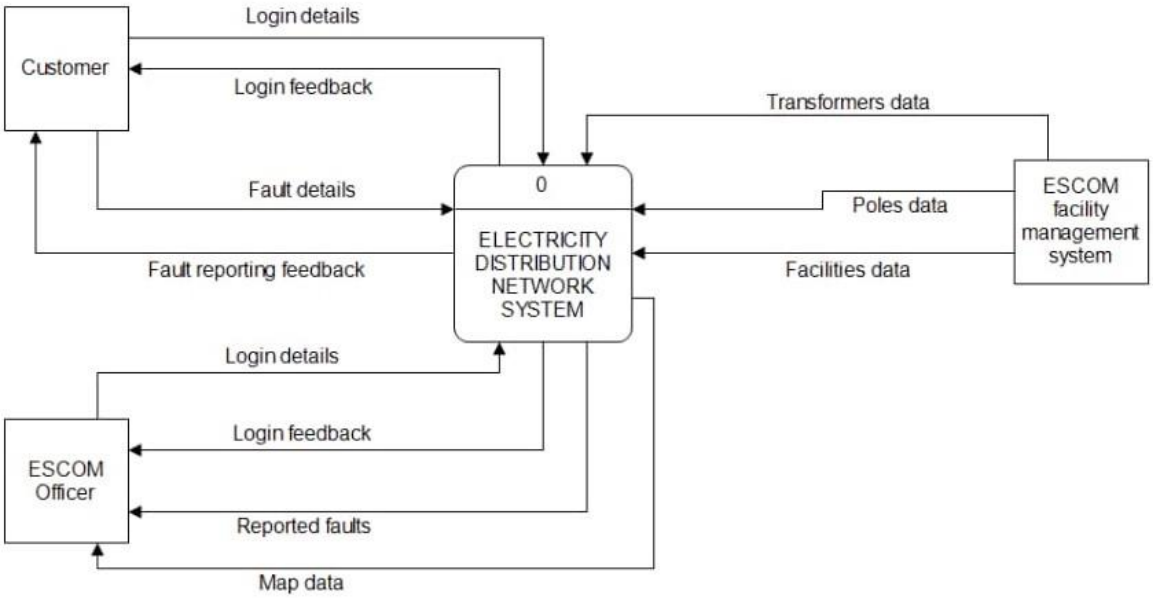


Figure 3: Context Diagram

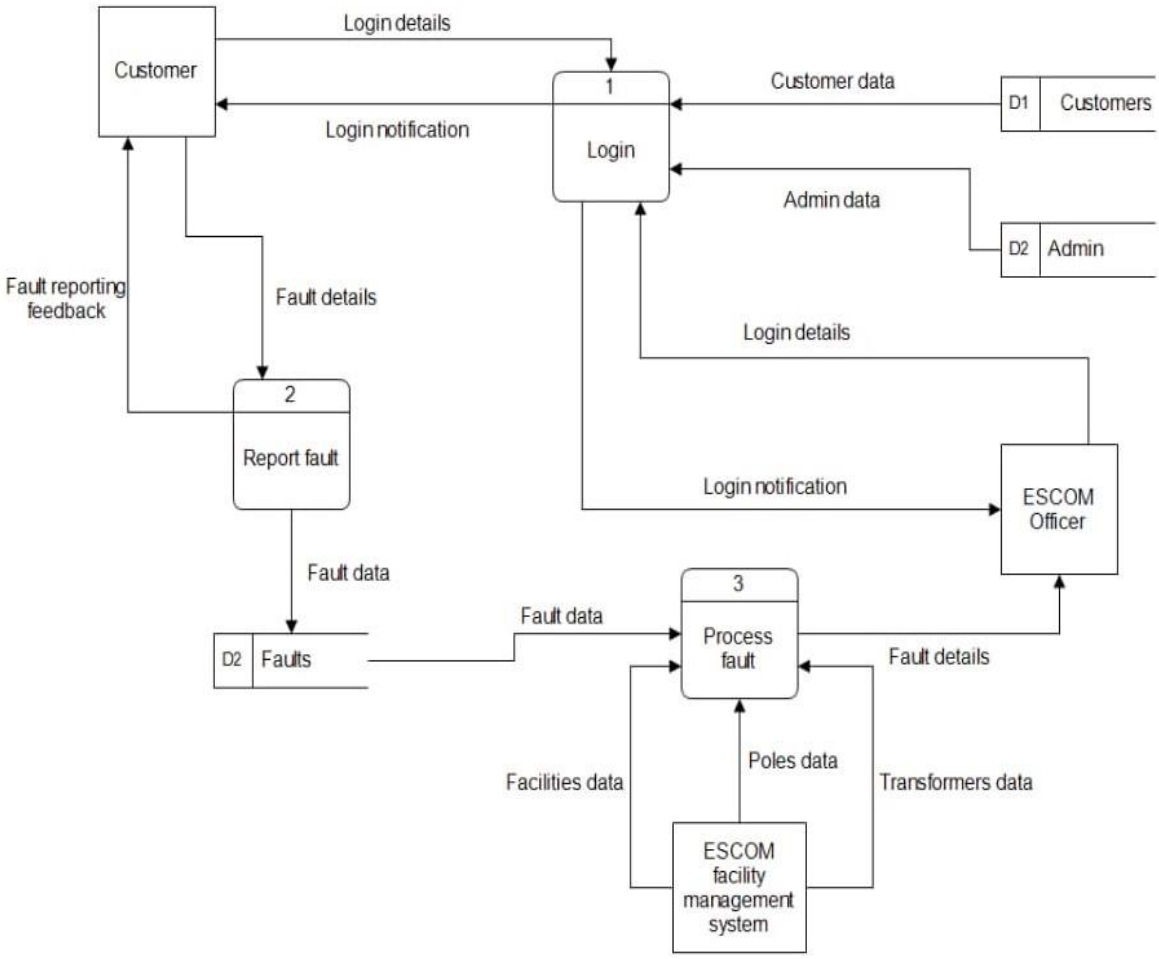


Figure 4: Level 0 DFD

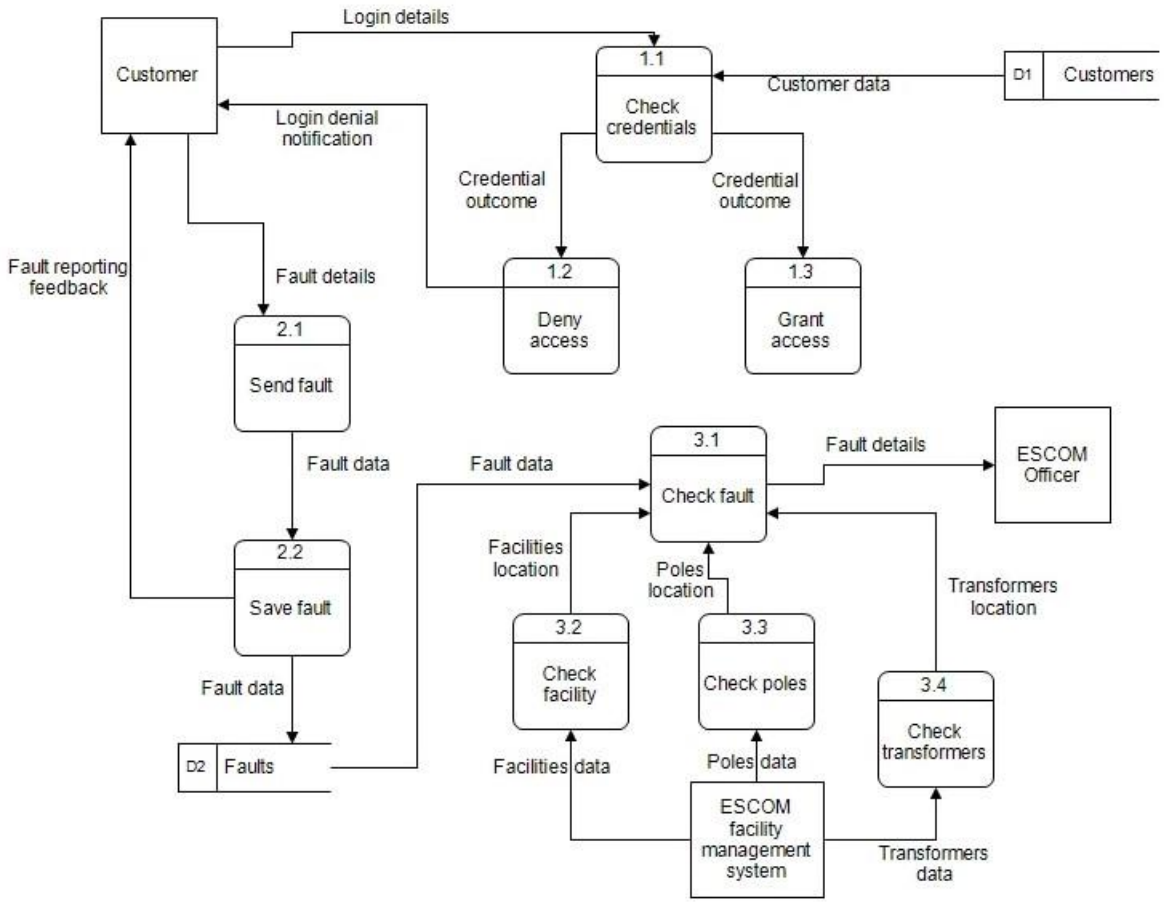


Figure 5: Level 1 DFD

4.3.3. Database Design Model

The system's database was created using PostgreSQL 13 and it is composed of five different tables (subsystems) that work hand-in-hand to provide a complete, functional and working system to achieve the system's goals. These subsystems will work by interconnecting with the database, to store all the data needed and involved with the tables. The database design using an entity-relationship diagram (ERD) of the system is shown below.

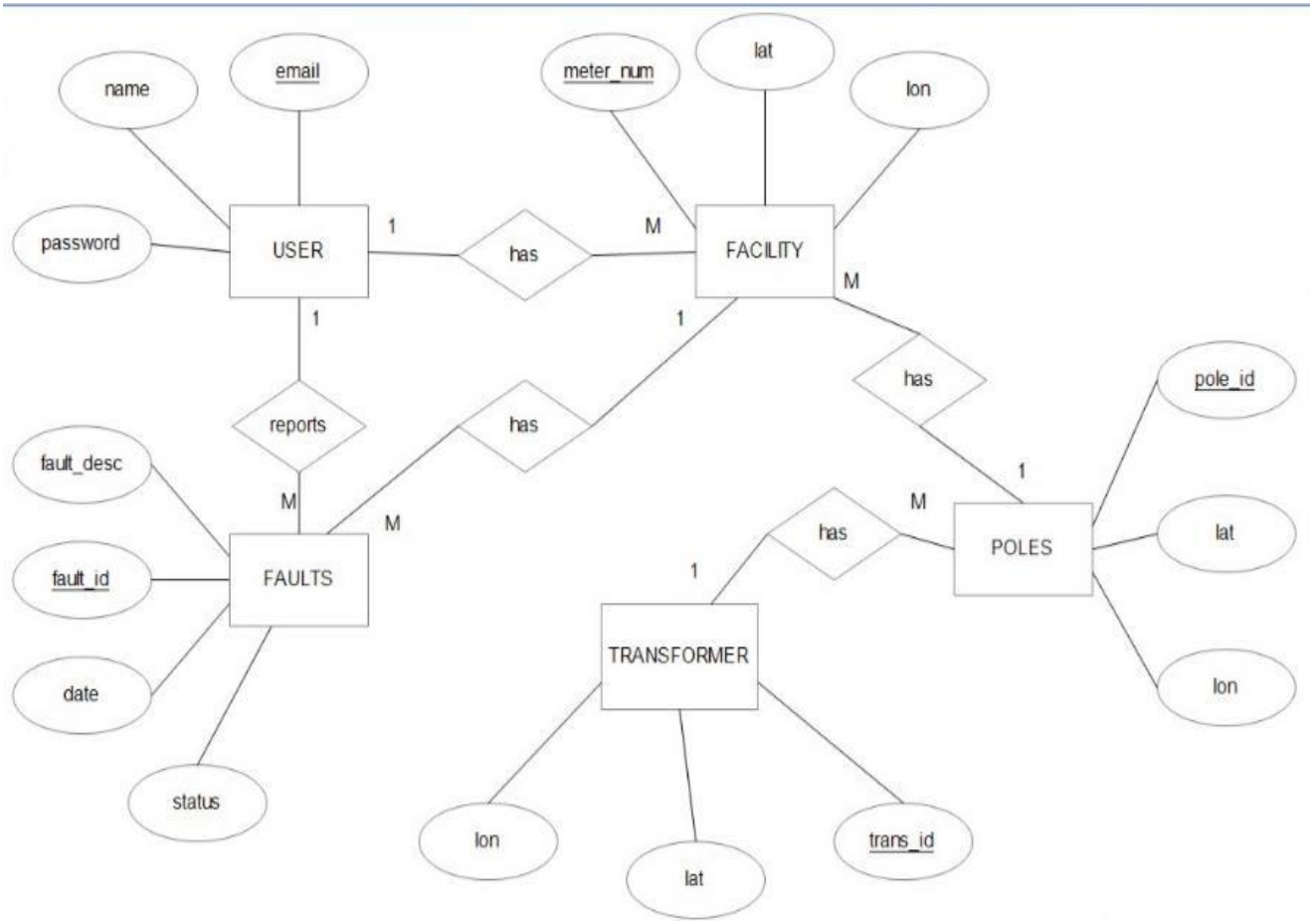


Figure 6. Showing the System's ERD

In this case, a user has a facility, one user can have many facilities but one facility can not belong to many users. A facility is connected from poles, one pole can connect many facilities but one facility can only be connected from one pole. A pole is connected from a transformer. One transformer can connect many poles but one

pole can be connected from one transformer. A user can report a fault, one user can report many faults nut one fault cannot be reported by many users.

4.3.4. Database physical model

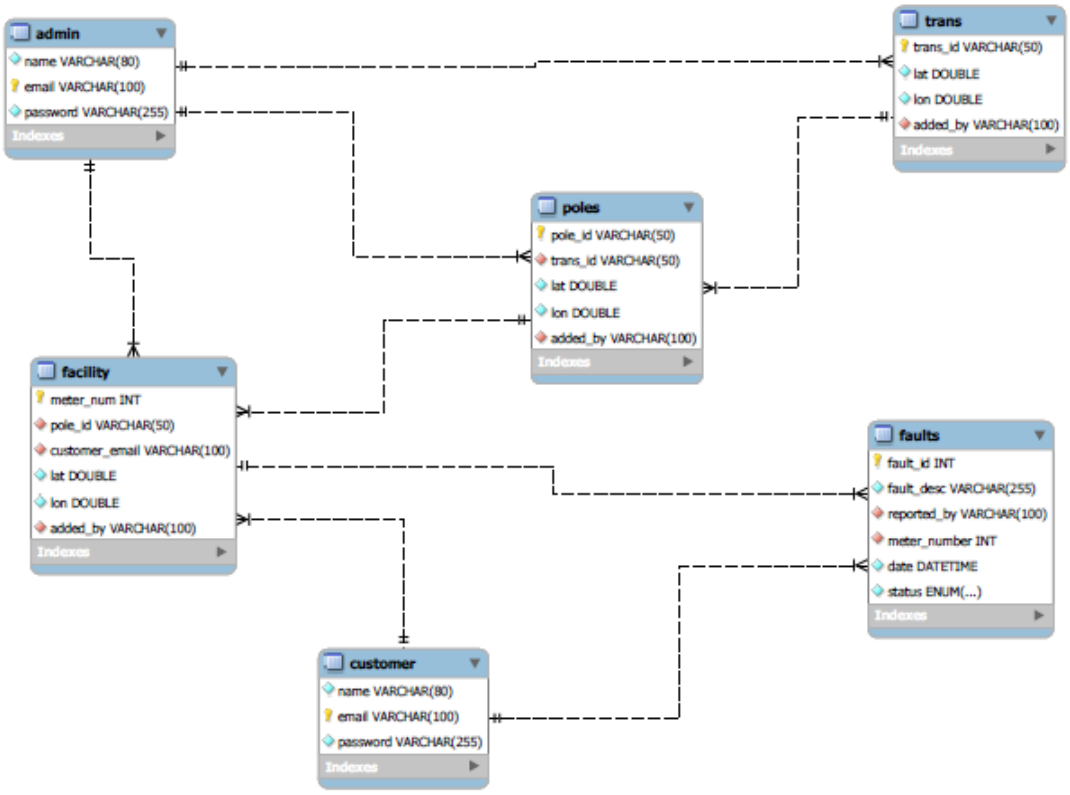


Figure 7. Shows a physical model of a database used in this system

In the above database, the following is happening;

- One admin can add many transformers
- One admin can add many poles
- One admin can add many facilities
- One pole can connect many facilities
- One customer can have many facilities
- One customer can report many faults
- One facility can have many faults

4.3.5. System's Subsystems

4.3.5.1. User Log in subsystem

The GIS web-based electricity management and fault tracking system has a login subsystem that will have the ability to manage all the users in the user entity in the database. This subsystem will help the creation of new users and allows registered users to log in into the system.

4.3.5.2. Data Flow Diagram

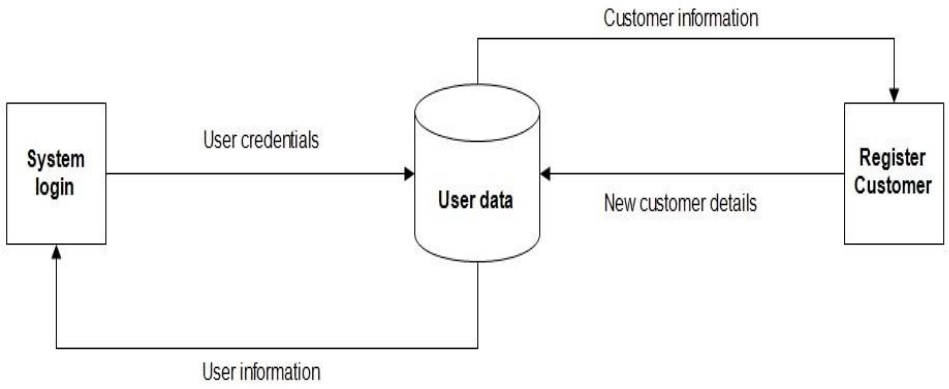


Figure 8. Showing a User Log in subsystem Data Flow Diagram

4.3.5.3. Software specification

A. Creating a User account

Function: Create a new user

Description: a new user to be registered in the system

Input: Full name, email, meter number, password

Output: Login page

Destination: database

Source: user

B. **Logging into system**

Function: Login into the system

Description: user enters into the system

Input: correct email and password

Output: User Dashboard

Destination: database

Source: user

C. **Data Entry subsystem**

The system cannot work without data. This subsystem is eligible to allow users (customers and administrators) to input data into the system for effective functionality. Front-end users will enter data into the system's database in form of name, email, password, and reported faults whereas back-end users will enter data into the database by entering data into the remaining four entities namely;

- Facilities (meter_num, pole_id, lat, lon),
- Poles (pole_id, trans_id, lat, lon)
- Transformers (trans_id, lat, lon)

D. **Visualization Subsystem**

This is another system's functionality, that will allow back-end users to visualize data on a web map page inform of points. This data will be retrieved from the database mainly from users (customers) and faults entities. As explained earlier, per requirement, it should be able to query data from the database and retrieve the data

(information) and manipulate the information into GeoJSON format which can be plotted (visualized)on a web map page as reported faults in form of points.

Software Specification

1. View faults

   Function : View reported faults

   Description : User (back-end) enters into the systems database

   Input : correct email and password

   Output : Admin dashboard

   Destination : Visualization page (web-map)

   Source : Database

5. DETAILED / ARCHITECTURAL DESIGN

5.1. Overall Description of the Design Solution

In the software specification, the modules of the device models are also defined. The same modules have illustrated the architecture of the interface models (interface screenshots).

The system is made up of several modules that will work together to provide the consumer with the services they need, geographic information system. User registration, authentication, fault reporting, and data entry are all parts of the system's architectural design.

5.2. System design

5.2.1. Welcome Page

This is a page that introduces users the system. Users are allowed to either create an account (Sign Up) or login (Sign in) into the system. This page has several buttons that are directed to specific response. Among the buttons there is Home, About Us, Contact Us, Sign Up, Log in as well as Report here button. Sign Up, Report Here and Log in buttons are linked to registration and log in forms.

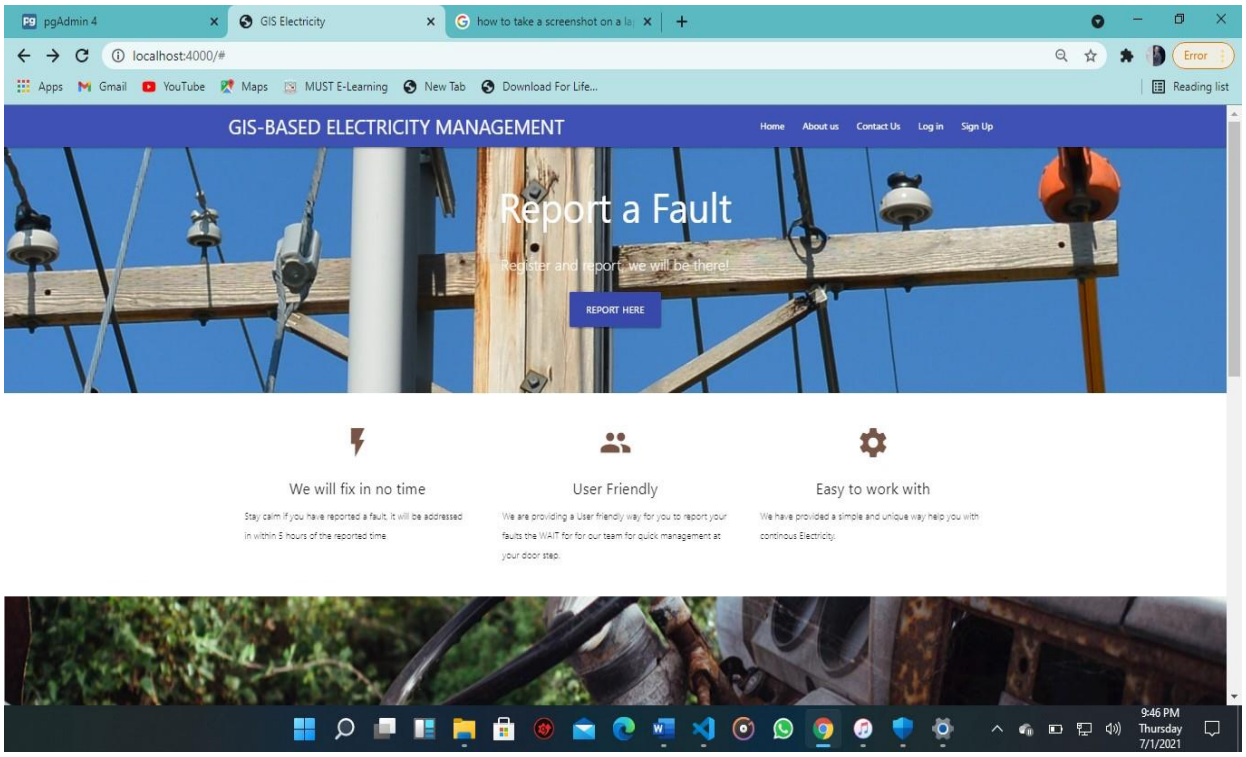


Figure 9. Shows a Welcome Page

5.2.2. Register Account Page

5.2.2.1. Software specification

Function: Register form

Description: The user enters his/her name, meter number, email, and password to register into the system.

Input: Full name, meter number, email, and password

Output: registered user

Destination: database

Source: User (keyboard)

Precondition: none

5.2.2.2. Interface Design

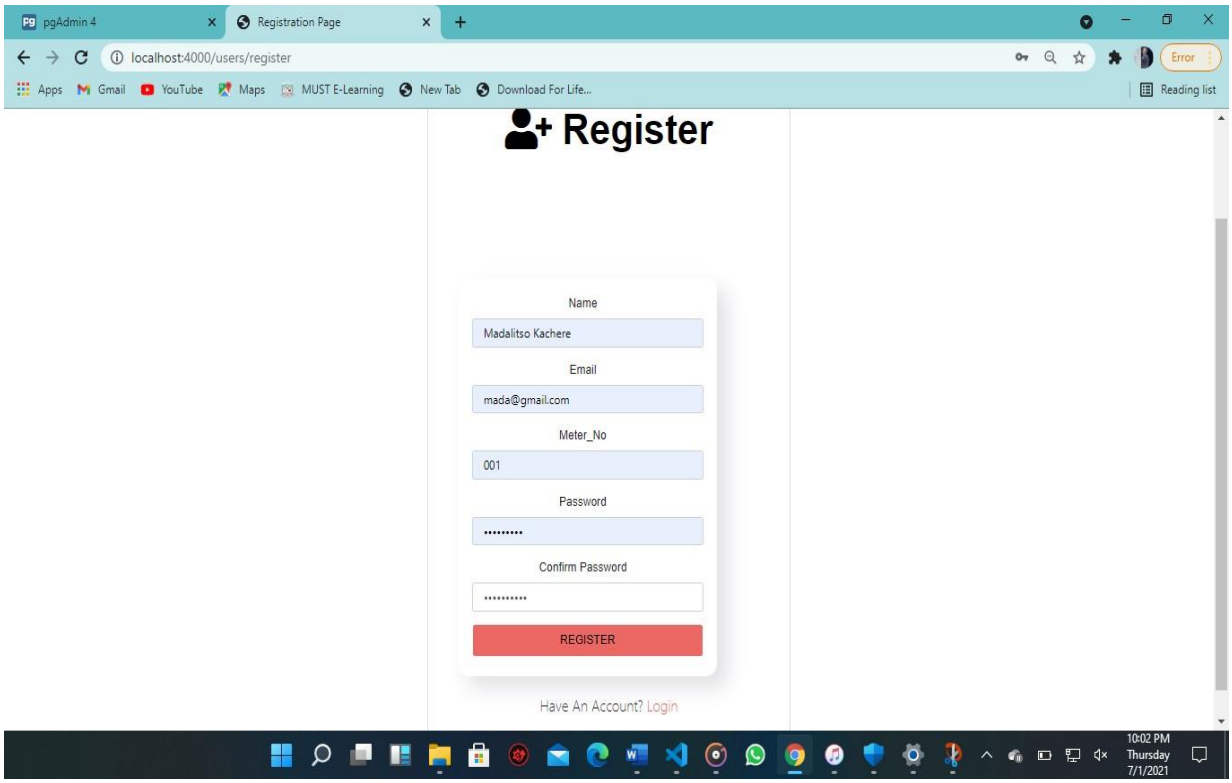


Figure 10. Shows a Registration Page

Full name: The name of the user (Customer).

Meter number: This meter number that will be registered.

User email: This is the email that will be registered.

Password: any combinations of characters such as *H#--cy18 #@*, i.e., must not less than 6 characters.

Register: A Button to submit the information for authorization. If a user already exists with the provided information then it fails, then a message appears on top of the form.

5.2.2.3. Data structure design

Data type, name, and format
Name: VARCHAR, meter number: VARCHAR, email: VARCHAR, Password: VARCHAR

5.2.2.4. Algorithm design

Read name

Read meter number

Read email

Read password

If the choice is register

Check if all input boxes are filled

Connect to database

If the connection is successful, then

Access user table

Read email and meter number from the user table

If email or meter number exist then

Return error

Else username doesn't exist

Write the name, meter number, email, and password to the user table

5.2.2.5. Database table

Users {name VARCHAR (80) email VARCHAR (80) meter_num VARCHAR (80) password VARCHAR (128)} Primary key – email

5.2.3. Login Page

5.2.3.1. Software specification

Function: Login form

Description: Registered user enters his/her email and password to login into the system.

Input: correct email and password

Output: logged-in user

Destination: database

Source: User credentials (keyboard)

Precondition: correct email and password

5.2.3.1. Interface Design

As a security control measure, all users must log in to the system, by providing their email and password. Upon arriving on the login page, the following user interface will appear.

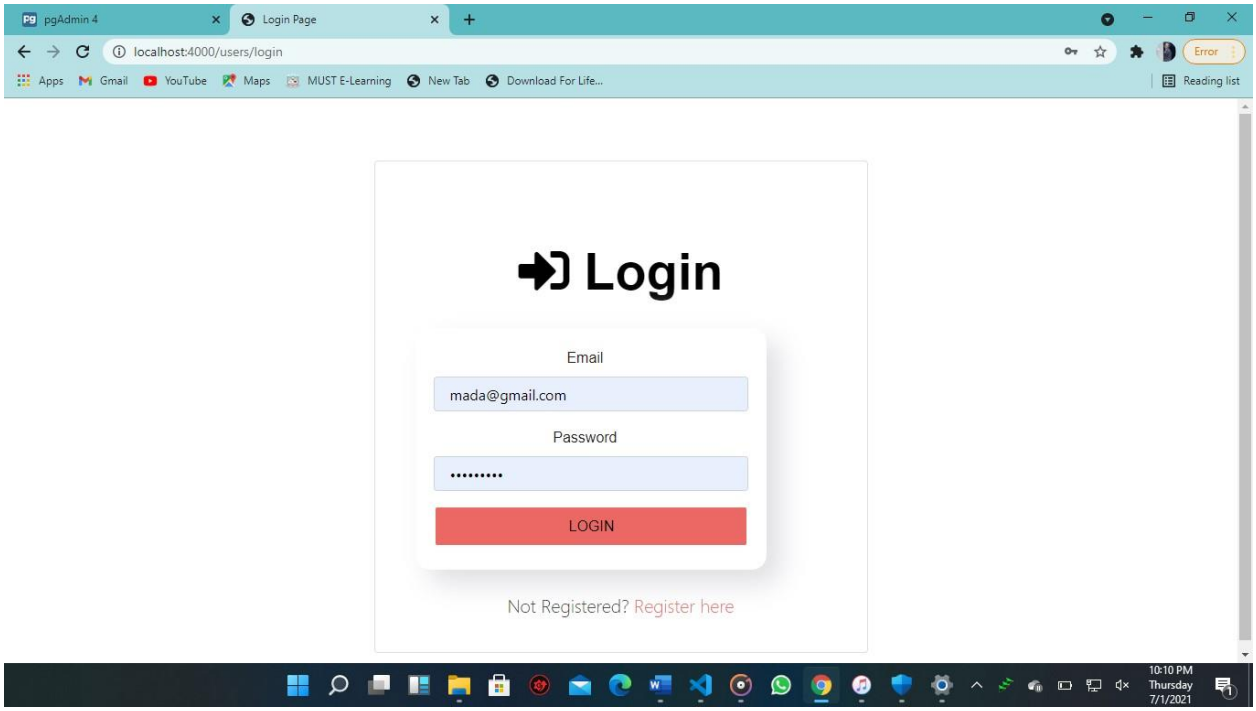


Figure 11. Shows a Log in page.

Username: This is the email that was registered.

Password: any combinations of characters such as *M#--blks24 @#*, i.e., must not less than 6 characters.

Log in Here: A Button to submit the credentials for authorization. If authorization fails, then a message appears on top of the username label.

Register: A link to the registering page

5.2.3.2. Data structure design

Data type, name, and format
email: VARCHAR, Password: VARCHAR

5.2.3.3. Algorithm design

Read email

Read password

If the choice is login

Check if all input boxes are filled

Connect to database

If the connection is successful, then

Access user table

Read email and password from the user table

If email and password exist then

Redirect to the dashboard page

Else username and password don't exist

Show error message

5.2.3.4. Database table

Users {name VARCHAR (80) email VARCHAR (80) meter_num VARCHAR (80) password VARCHAR (128)}

Primary key – email

5.2.3.5. Dashboard Page

When registered user are successfully logged in into the system, the user will be welcomed with a dashboard where they can either report a fault or log out of the system with the navigation buttons.

5.2.3.6. Interface design

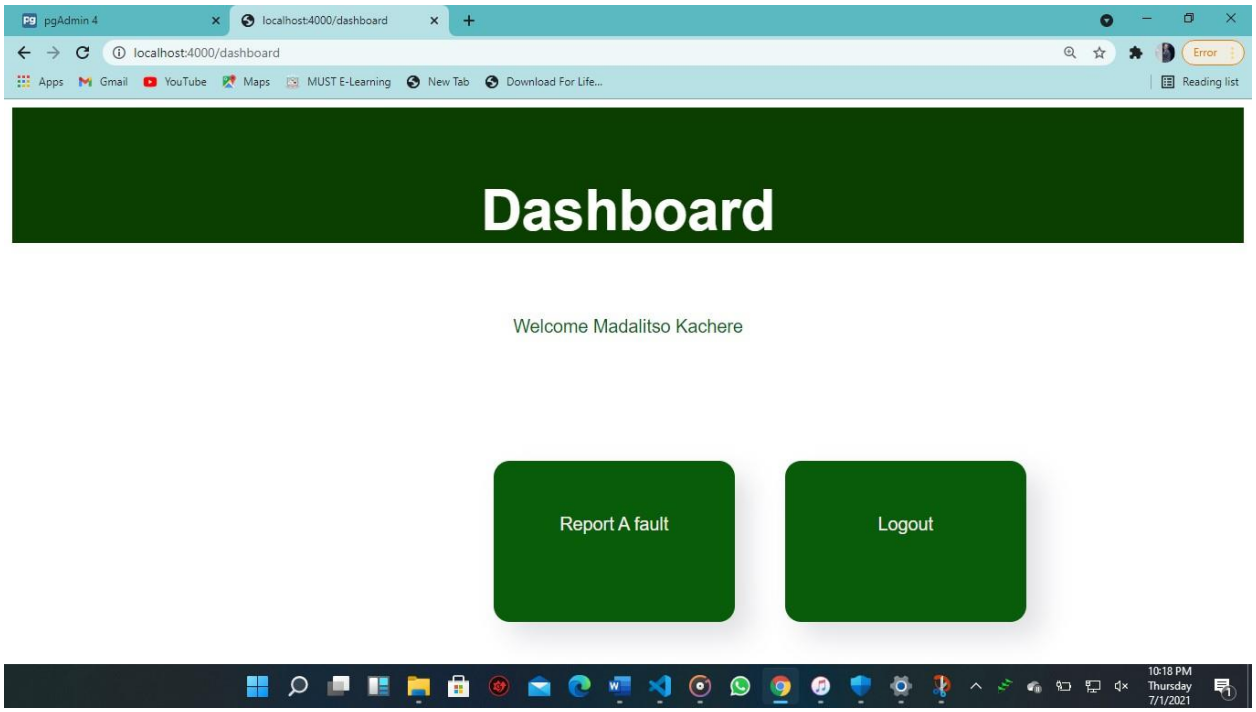


Figure 12. Showing a Dashboard

Based on the dashboard page, the user can select a functionality to access, either to report a fault or log out of the system.

5.2.3.7. Algorithm design

If choice = report fault, then

Redirect to fault reporting page

If choice = logout, then

Redirect to home page

5.2.3.8. Fault reporting Page

This page enables a registered user (customer) with a. The user fills in a reporting form and then submits it for storage in the database where an administrator can access it.

5.2.3.9. Software specification

Function: Fault reporting form

Description: The User reports a fault into the system.

Input: name, meter number, date, fault description.

Output: Successful fault report submission

Destination: database

Source: User

Precondition: logged in User

Postcondition: no empty field

5.2.3.10. Interface Design

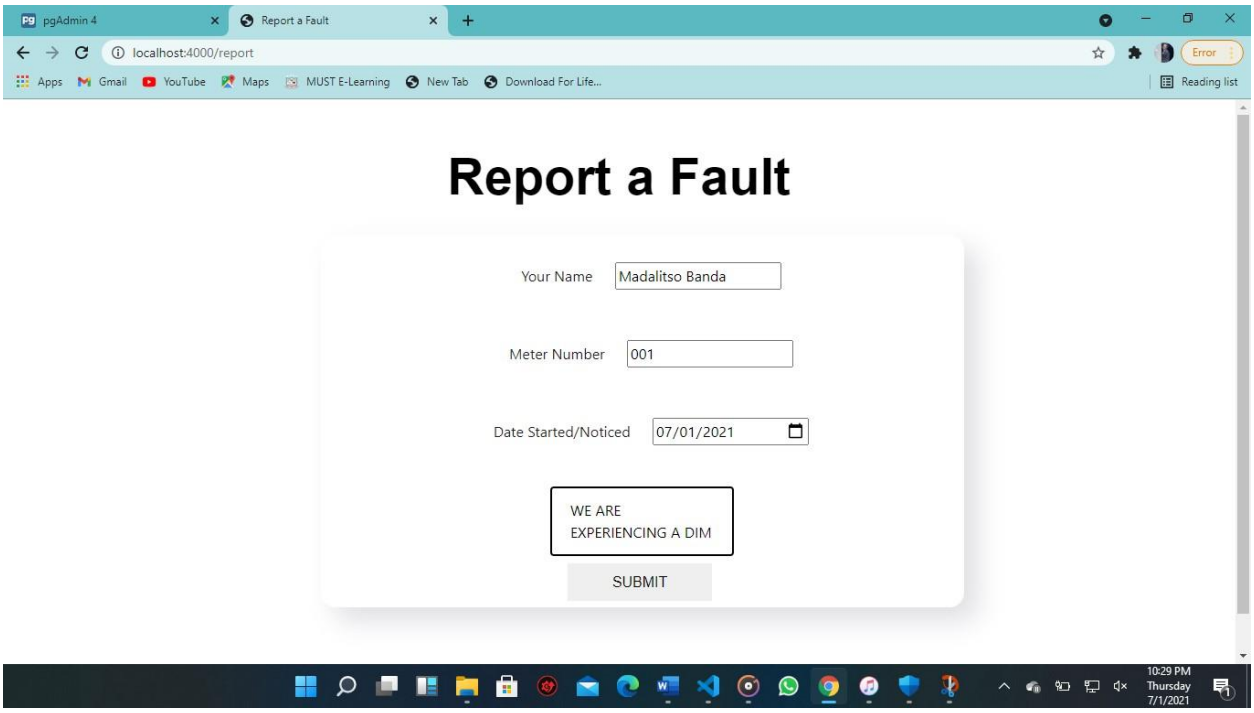


Figure 13. Shows fault reporting page

Registered customer (User) is entitled to fill in the form with full name, meter number, date and description of the fault the submit.

Name: This is the name of the logged in User.

Meter Number: This is the meter number of the User's facility.

Date: Shows date where the fault started or noticed.

Fault Description: Tells

Submit: Sends the entered data to the database.

5.2.3.11. Data structure design

Data type, name, and format
Name: VARCHAR, Meter Number VARCHAR, Date DATE, Fault Description VARCHAR

5.2.3.12. Algorithm design

If Authenticated

Read Name

Read Meter Number

Read Date

Read Fault Description

If the choice is to submit

Check if all input boxes are filled

Connect to Database

If the connection is successful, then

Write Name, Meter number, Date, Fault Description and Time to faults table

5.2.3.13. Database table

Faults {fault_desc character varying (150), reported_by character varying (40), meter_num character varying (50), date Date, status character varying (20)}

5.2.4. Data Visualization Page

This is a Back-end page that is used by system administrators to access the system. The admin is entitled to register, login with the system, view fault list as well as view map to locate the faults on a web-map page.

5.2.4.1. Welcome Page

This is a page that introduces the admin to the system. The admin is allowed to either create an account (Sign Up) or login (Sign in) into the system. This page has a two buttons, Register Account and Login. These buttons are linked to registration and log in forms.

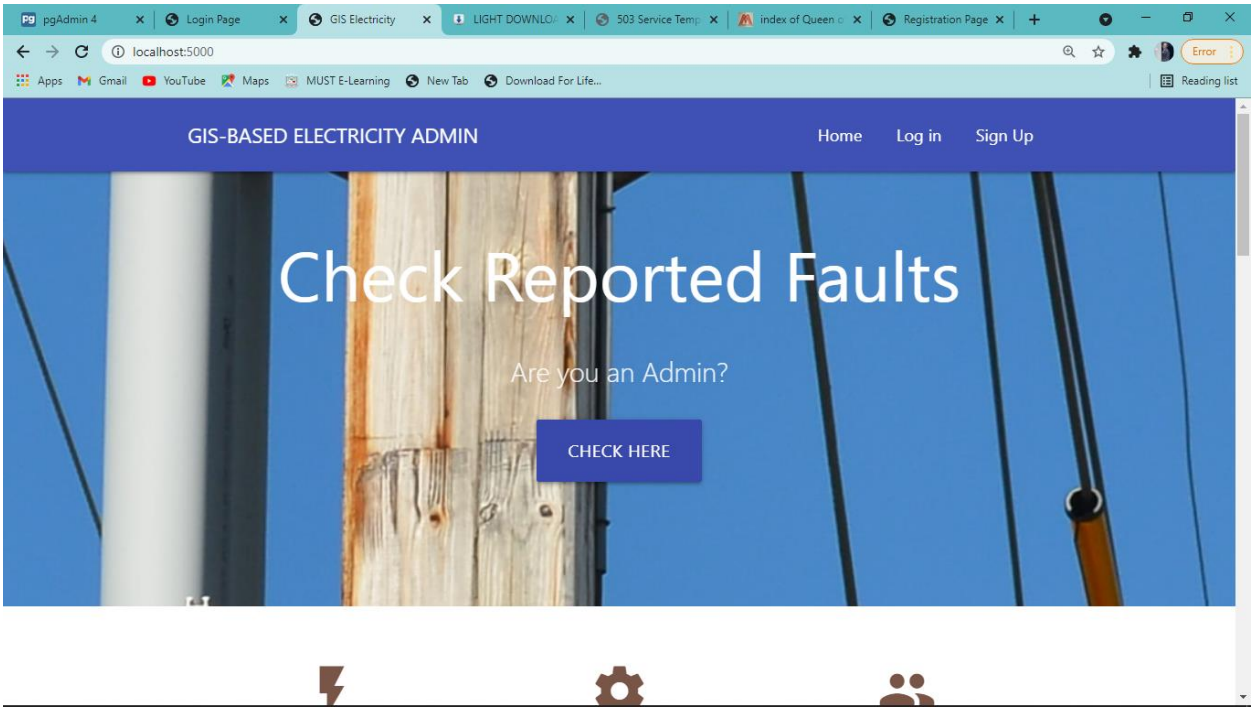


Figure 14. Shows an Administrator Welcome Page

5.2.4.2. Register Page

5.2.4.3. Software specification

Function: Register form

Description: The Admin enters his/her name, email, and password to register into the system.

Input: Full name, email, and password

Output: registered Admin

Destination: database

Source: User (keyboard)

Precondition: none

5.2.3.1. Interface Design

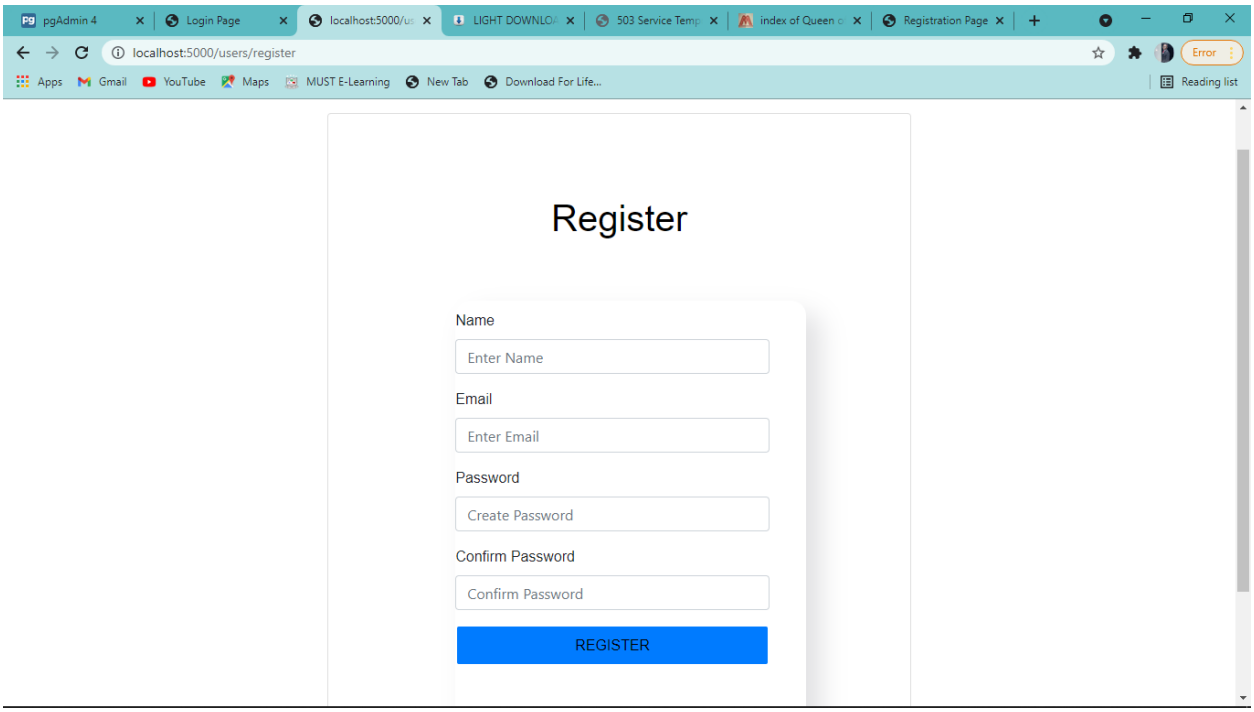


Figure 15. shows the interface design of the admin registration page

Full name: The name of the user (Admin).

User email: This is the email that will be registered.

Password: any combinations of characters such as *H#--ght49 #@*, i.e., must not less than 6 characters.

Register: A Button to submit the information for authorization. If a user already exists with the provided information, then it fails, then a message appears on top of the form.

5.2.3.2. Data structure design

Data type, name, and format
Name: VARCHAR, email: VARCHAR, Password: VARCHAR

5.2.3.3. Algorithm design

Read name

Read email

Read password

If the choice is register

Check if all input boxes are filled

Connect to database

If the connection is successful, then

Access Admin table

Read email and from the admin table

If email exist then

Return error

Else username doesn't exist

Write the name, email, and password to the admin table

5.2.3.4. Database table

Admin {name VARCHAR (80) email VARCHAR (80), password VARCHAR (128)}

Primary key – email

5.2.5. Login Page

5.2.5.1. Software specification

Function: Login form

Description: Registered user (Admin) enters his/her email and password to login into the system.

Input: correct email and password

Output: logged-in user

Destination: database

Source: User credentials (keyboard)

Precondition: correct email and password

5.2.3.1. Interface Design

As a security control measure, all users must log in to the system, by providing their email and password. Upon arriving on the login page, the following user interface will appear

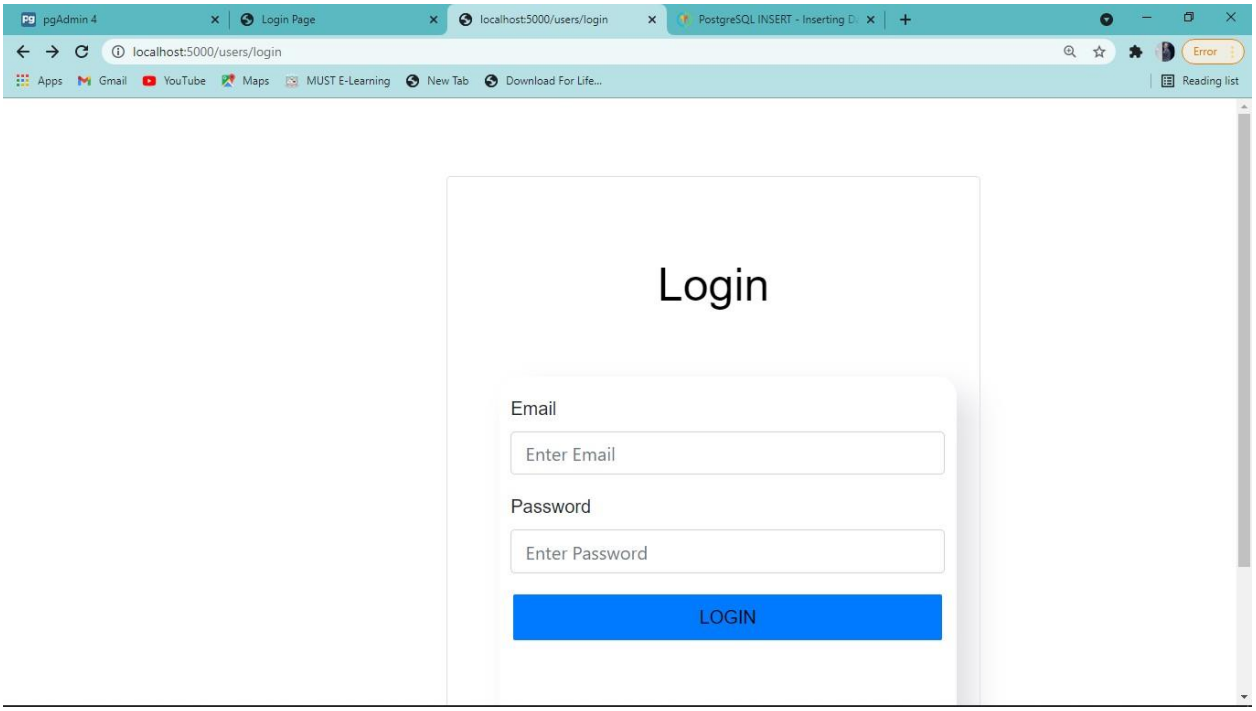


Figure 16. Shows an Admin Log in page

Username: This is the email that was registered.

Password: any combinations of characters such as *M#--blks24 @#*, i.e., must not less than 6 characters.

Log in Here: A Button to submit the credentials for authorization. If authorization fails, then a message appears on top of the username label.

Register: A link to the registering page

5.2.3.2. Data structure design

Data type, name, and format
email: VARCHAR, Password: VARCHAR

5.2.3.3. Algorithm design

Read email

Read password

If the choice is login

Check if all input boxes are filled

Connect to database

If the connection is successful, then

Access Admin table

Read email and password from the admin table

If email and password exist then

Redirect to the dashboard page

Else username and password don't exist

Show error message

5.2.3.4. Database table

Admin {name VARCHAR (80) email VARCHAR (80)) password VARCHAR (128)}

Primary key – email

5.2.6. Dashboard Page

When a registered Admin is successfully logged in into the system, the admin will be welcomed with a dashboard page where he/she can either select to view fault list or view fault map.

5.2.6.1. Interface design

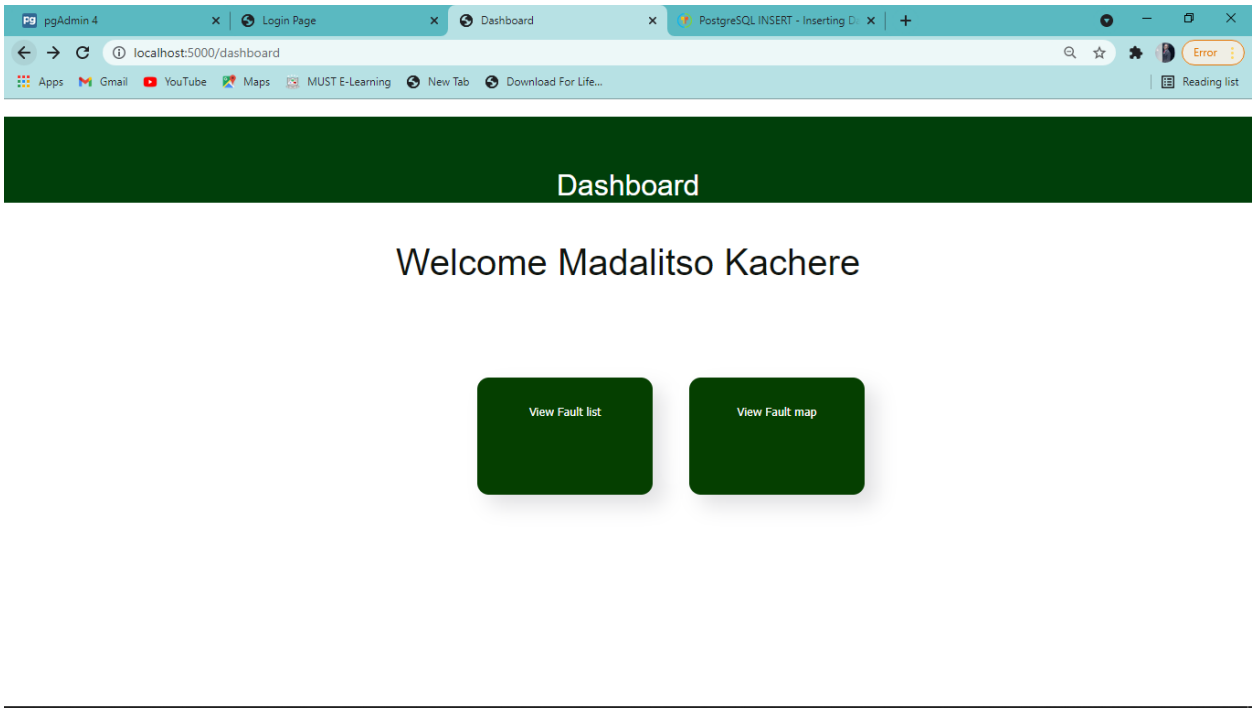


Figure 17. Shows a dashboard for am admin

Based on the dashboard page, the admin can select a functionality to access, either to fault list or log view fault map.

5.2.6.2. Algorithm design

If choice = view fault list, then

Redirect to fault list page

If choice = view fault map, then

Redirect to fault map page

5.2.7. Fault List Page

This is a page that contains a list of faults reported by a customer(user).

5.2.7.1. Software specification

Function: faults form

Description: List of faults reported by users (Customers)

Input: Name, Meter number, Date, Fault Description

Output: Reported fault

Destination: database

Source: User credentials (keyboard)

Precondition: correct email and password

5.2.7.2. Data structure design

Data type, name, and format
Name: VARCHAR, Meter Number VARCHAR, Date DATE, fault Description VARCHAR

5.2.7.3. Database table

Faults {fault_desc character varying (150), reported_by character varying (40), meter_num character varying (50), date Date, status character varying (20)}

5.2.7.4. Interface Design

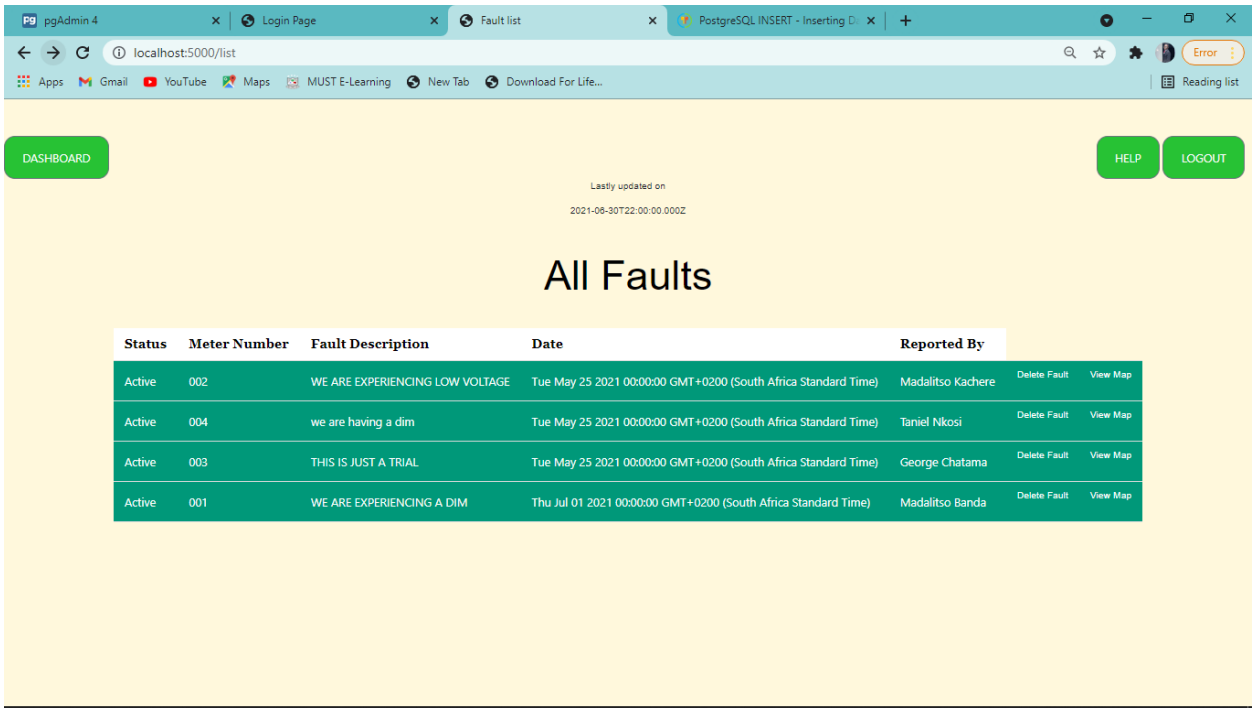


Figure 18. Shows a fault list page.

On this Page, the Admin can view the list of all faults reported. The list updates every time a new fault is reported and the admin can delete a fault, view help, log out of the system or return to a dashboard page.

5.2.7.5. View Fault Map Page

This is a real-time GIS core functionality that provides visualization of reported faults on a digital map. This view contains a map containing facilities, poles and transformers whereby they all have coordinate systems depending on where they are situated.

5.2.7.6. Software Specification

Function: get recent Fault reports

Description: The map loads recent reported faults

Input: none

Output: map

Destination: Admin

Source: System (on page load)

Precondition: logged in user

5.2.7.7. Interface Design

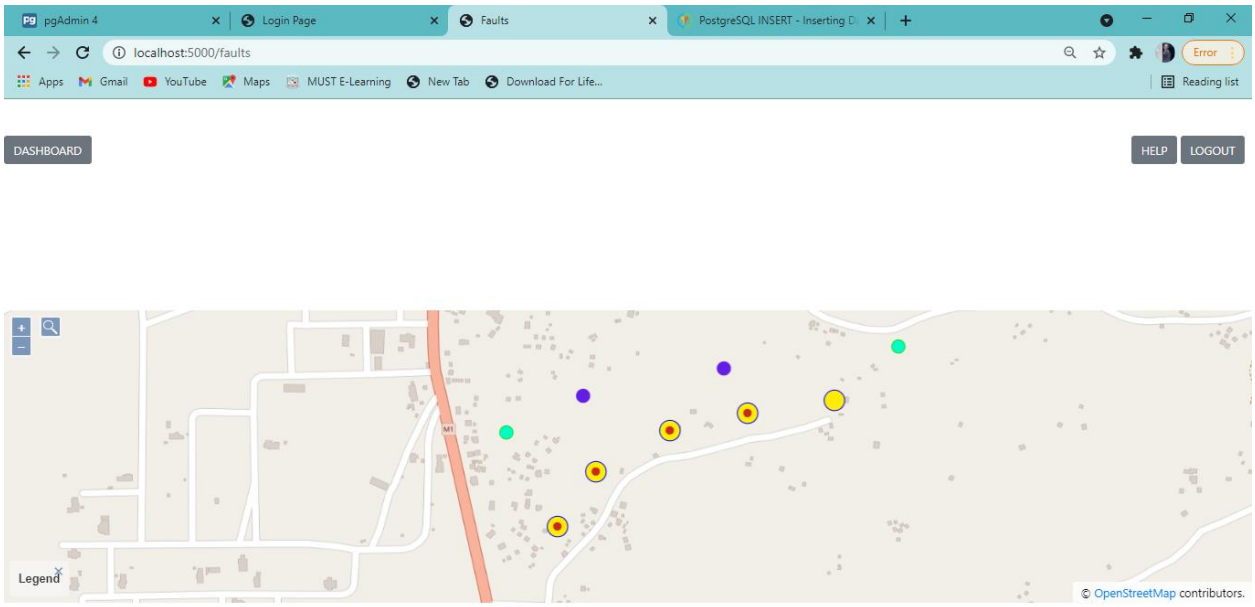


Figure 19. Shows a fault map

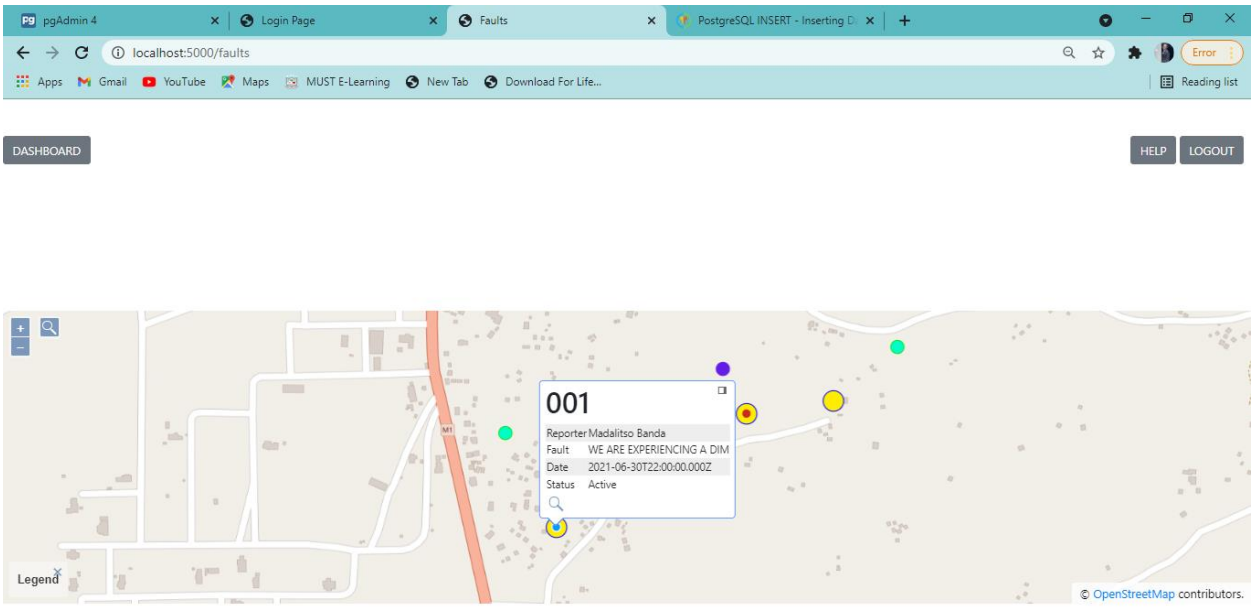


Figure 20. Shows a Fault map functionalities, with a pop-up.

Based on figure 18 and figure 19, the two blue points represents transformers, the other two purple points represents poles. The other five yellow points represents facilities.

When a user reports a fault, a red point overlays with a facility point depending on the shared locational coordinates of the user's facility, and meter number. When a reported facility is clicked on a map, a pop-up window shows as in figure 19, showing the meter number of the facility, name of the reporter (customer/user), date, as well as status of the fault. The other one yellow point without overlayed points represents a facility that have not reported any faults.

Once an admin is done locating a fault on a fault map, there is a need to manage it by sending a team of engineers or electricians to manage the fault, he/she can return to a dashboard page then navigating to a fault list page then delete the fault and finally log out of the system or refresh the fault list page to know other reported faults.

5.3. System Infrastructure

5.3.1. Overview

The user interface, web server, application server, and database server are all part of the system's 4-tier architecture. Node JavaScript is used to create the application and web servers. Node JavaScript is a scalable network application runtime that is based on an asynchronous event driven JavaScript runtime. PostgreSQL version 13 and MongoDB are the database servers of choice. In comparison to PostgreSQL, which needs the use of long, sophisticated query strings, Mongo DB provides simple and easy functions for querying information from the database. The records are stored in PostgreSQL since they don't need to be retrieved as quickly as user data, which is constantly needed by the system to authenticate users to use system resources. EJS was used to create the user interface, which allows dynamic content to be injected into the pages, making each user's experience unique. CSS3 was used to style the user interface, which allowed for several interactive and appealing page features.

5.4. Web Server

The webserver is developed from the ground up using node JavaScript and associated frameworks, as mentioned in the overview. The webserver's job is to serve as a conduit between the front end, or user interface, and the application server. It responds to all of the user's queries.

a) **Routes**

- "/": This gets the home page of the system. Once a user enters the main URL, the system calls this route and the system renders the home page.
- "/dashboard": This gets the dashboard page. After logging in, the system calls this route and renders the dashboard for the user.
- "/help": This gets the help page. The system calls this route and renders the help page.
- "/entry": This gets the data entry form page. Once a user clicks on a link to enter data, the system calls this route. After a user submits the form, the

system posts the route and then sends the data to the application server using the Axios module.

- "/senu": This gets the visualization page. Once a user clicks on a link to visualize data, the system calls this route, and then the system renders the map onto the user interface after getting the map data from the application server.

b) **Modules used specification**

- Axios: used to make API calls to the application server when requesting and sending data such as the faults reported by clients
- Bcryptjs: used to encrypt and decrypt user passwords.
- ejs: used as a templating engine for the web pages.
- express: web application framework used to build the server.
- express-session: used to make sessions for each user.
- mongoose: used to connect to the mongo DB database.
- passport: used in the authentication process of the users.
- passport-local: used in the authentication process of the users

5.2.4. Application Server

As eluded in the overview, the Application server is built from scratch by the use of node JavaScript and its frameworks. The purpose of this server is to act as the center and a bridge between user requests and the actual database. It queries data from the database and makes it available through its routes, so the webserver can get this data by making API calls to the server then receive the data in JSON and GeoJSON formats.

a) **Routes**

- **Getter.js**

This is used to share data between different entities in the database. In other words, it is also used to retrieve and query data in the database

- **Index.js**

This is mostly used to render the reporting page of a user to be successful and also it makes sure that the details are shown on a fault map.

- **User.js**

Makes it possible that the user authentication is always initiated.

b) Modules used specification

- express: web application framework used to build the server.
- Body-parser: used when sending data in the routes. It enables the system to use the JSON format.
- Cors: used in implanting communications with other servers, for example when the application server is communicating with the webserver.
- Moment-timezone: used in converting time to other time zones.
- Pg: used to connect with the PostgreSQL database.
- Sequelize: used as an ORM for the PostgreSQL database.

6. RESULTS AND DISCUSSIONS

6.1. DATA COLLECTED

Data and information collected in this research is mainly dummy data which is created to match the requirements of this study. Below is a table of the information used in this research.

Table 1. Showing how the facilities with facility ID and their location details and also poles connected to each facility. In this case, facilities with meter number 001 and 002 are connected to pole with pole ID, P1 while the other facilities are connected to P2.

Facility_ID or Meter number	Pole_ID	Latitude	Longitude
001	P1	34.970838	-15.877925
002	P1	34.971484	-15.877017
003	P2	34.972705	-15.876336
004	P2	34.973992	-15.876047
005	P2	34.97543	-15.87584

Table 2. Showing how the poles are connected to a specific transformer. Pole with pole_id P1 is connected from transformer with transformer_id T1. Pole. Pole with pole_id P2 is connected from transformer with transformer_id T2. The location details (latitude and longitudes) are for the poles.

Pole_ID	Tranformer_ID	Latitude	Longitude
P1	T1	34.971267	-15.875758
P2	T2	34.973606	-15.875304

Table 3. Shows location details of transformers.

Tranformer_ID	Latitude	Longitude
T1	34.970001	-15.876377
T2	34.976503	-15.874953

6.2. Discussion

The GIS web-based electricity distribution and fault tracking system was built to allow remote access from any device with an internet connection, meaning that a user could potentially use all of the system resources without the need of any special expensive hardware but rather use readily available devices. The system's User Interface is mobile responsive, meaning that, it takes the size of the mobile device's display size in use to access the system and show the contents depending on the display size. The spatial data was mostly stored in the PostgreSQL database, which was hosted online, allowing the system to connect to it using networking protocols. MongoDB allowed for quick retrieval of user data, which was necessary in authentication scenarios. With the usage of the system's visualization subsystem, the integration of the subsystems resulted in a working system that could monitor real-time reported faults on a map depicting the spatial location of the faults happening at different power facilities.

The system is mainly developed to help clients register with the system and log in and be able to report the fault providing the date of which the fault started and a detailed description of the faults as they wait for up to date and quick management of the reported faults. On the other hand, the visualization subsystem allows an admin to log into the system, view fault list and also visualize reported faults on map. On the map, the reported faults have a red point overlayed with a yellow point that represents a facility. The facility has a spatial location whereby this spatial location will be used to track and locate the fault reported. However, when a reported fault point is clicked, a pop-up widow is displayed.

The pop-up window contains name of the client (reporter), meter number, fault description as well as date reported.

7. CONCLUSION AND RECOMMENDATIONS

The study clearly demonstrates the capabilities of a geographically enabled information system to control the electrical distribution network and management of faults. The spatial data of the power distribution network used in this study is primarily dummy data, which is currently acquired, processed, managed, stored in a geographical database (spatial database) for example this study used a PostgreSQL database, but this sort of data can also be digitalized. Geospatial Information Science (GIS) is very essential to economic planning and national development, therefore, in this research, GIS is used to allow Electricity utility company to locate and manage their services by locating where their services are being provided and also provide a real-time and up to date method of managing faults through a GIS-web based map for visualization. This is supported by Alamu and Ejiobih (2002), who concluded that a well-maintained utility information infrastructure provides up-to-date information on what is where, the state of it, the consequences of other actions on it, and how it can be harnessed for the most efficient use of people and resources. GIS has been shown to be one of the technologies used to improve the delivery of networked services. This also shows that, independent of cost, GIS has significantly enhanced the manner in which services are delivered in terms of time.

Finally, it was discovered that GIS applications have not yet reached their full potential, and that there is still opportunity for more research and development in the field of utility service delivery, such as in the case of power distribution and fault management.

For effective electricity distribution networks and fault management, the following suggestions are made;

1. More efforts should be made to include refined and scientific methodologies into the administration of power distribution networks and fault management through Geographic Information Systems (GIS) technologies.

2. Training programs on the integration of GIS in the administration and monitoring of energy distribution should be organized for technical online personnel of electricity distribution.
3. There is need to have a spatial location of the electricity facilities to fast and efficient management during disasters.
4. There is need to implement a GIS-web based real-time monitoring of electricity distribution in Malawi.
5. There is need to provide a better customer service through GIS real-time monitoring.

8. REFERENCE

- GIS Overview. Retrieved from: https://www.esri.com/en-us/what-is-gis/overview
- Pickering, D.J.M and Bannister, D.H. (1993): Utility Mapping and Record Keeping for Infrastructure.
- Korte, G (2001). The GIS book: how to implement, manage and assess the value of Geography
- Alamu, E. O. and Ejiobih, H. C. (2002). Utility Information Infrastructure Needs in Utility Organisations in Nigeria: A Case Study of Niger State Water Board. Proceedings of the Technical Session of the 37th Annual General Conference and Meeting of Nigerian Institution of Surveyors, Owerri, Imo State, Nigeria, pp. 85-88.
-
- Emengini, E. J. (2004). Application of Geographic Information System (GIS) to Utility Information Management: A Case Study of Onitsha-North L.G.A., Anambra State, Nigeria. Unpublished M.Sc. Thesis Submitted to the Department of Surveying and Geoinformatics, Nnamdi Azikiwe University, Awka, Anambra State, Nigeria
- Kale U. D., Lad R., 2006 GIS integration with SCADA, DMS & AMR in Electrical Utility, Map India Conference
- Sipes, J. L. (2007). Spatial Tech- GIS for the utilities Industry. Cadalyst Publicationss
- Gao, S., Mioc, F., and Coleman, D.J (2009), Online GIS services for mapping and sharing disease information, International Journal of Health Geographics, 8(3), pp 1-16. doi:10.1186/1476-072X-8-3

- Pickle, Eddie. (2012). The Benefits of Open Source/OpenGeoSuite. FOSS4G. Washington, DC: FOSS4G North America Conference

9. SYSTEM CODE.

A. Front-End

1. Application Server

a) App.js

```
// importing libraries
const express = require('express');
const passport = require('passport');
const flash = require('connect-flash');
const session = require('express-session');
const app = express();

// Passport Config
require('./config/passport')(passport);


// EJS

app.set('view engine', 'ejs');

// Express body parser
app.use(express.urlencoded({ extended: true }));
app.use(express.static(__dirname + '/public'));
// Express session
app.use(
  session({
    secret: 'secret',
    resave: true,
    saveUninitialized: true
  })
);

// Passport middleware
app.use(passport.initialize());
app.use(passport.session());
```

```
// Connect flash
app.use(flash());

// Global variables
app.use(function(req, res, next) {
  res.locals.success_msg = req.flash('success_msg');
  res.locals.error_msg = req.flash('error_msg');
  res.locals.error = req.flash('error');
  next();
});

// Routes
app.use('/', require('./routes/index.js'));
app.use('/data', require('./routes/getter.js'));
app.use('/users', require('./routes/users.js'));

const PORT = process.env.PORT || 4000;

app.listen(PORT, console.log(`Server running on  ${PORT}`));
```

2. Configuration

a. Auth.js

```
module.exports = {
  ensureAuthenticated: function(req, res, next) {
    if (req.isAuthenticated()) {
      return next();
    }
    req.flash('error_msg', 'Please log in to view that resource');
    res.redirect('/users/login');
  },
  forwardAuthenticated: function(req, res, next) {
    if (!req.isAuthenticated()) {
      return next();
    }
    res.redirect('/dashboard');
  }
};
```

b. Connect.js

```
const {Pool} = require('pg')
// connecting to a database
const db = new Pool({
    database:'Electricity Management',
    password:'1234',
    port:5433,
    user:'postgres',
})

module.exports = db;
```

c. keys.js

```
// importing mongodatabase
dbPassword = 'mongodb+srv://YOUR_USERNAME_HERE:'+ encodeURIComponent('YOUR_PASSWO
RD_HERE') + '@CLUSTER_NAME_HERE.mongodb.net/test?retryWrites=true';

// initializing mongodb
module.exports = {
    mongoURI: dbPassword
};
```

d. passport.js

```
const LocalStrategy = require('passport-local').Strategy;
const bcrypt = require('bcryptjs');
const db = require('./connect')
// Load User model


module.exports = function(passport) {
  passport.use(
    new LocalStrategy({ usernameField: 'email' }, (email, password, done) => {

      // Match user
      db.query(`select * from users where "email"='${email}'`,(err,results)=>{
        if(err){
          console.log(err);
          return done(null, false, { message: 'That email is not registered' });
        }else{
          console.log(results.rows);
          const user=results.rows[0]
          bcrypt.compare(password, user.password, (err, isMatch) => {
```

```
            if (err) throw err;
            if (isMatch) {
              return done(null, user);
            } else {
              return done(null, false, { message: 'Password incorrect' });
            }
          });

        }
      })
    })
  );

  passport.serializeUser(function(user, done) {
    console.log('serial');
    done(null, user.meter_num);
  });

  passport.deserializeUser(function(id, done) {
    db.query(`select * from users where "meter_num"='${id}'`,(error,results)=>{
      if(error){
        console.log('error on deserial');
      }else{
        const user=results.rows[0]
        done(error,user)
      }
    })
  });
};
```

3. models

a. Users.js

```
// importing  a mongoose
const mongoose = require('mongoose');

// creating a schema
const UserSchema = new mongoose.Schema({
  name: {
```

```
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  date: {
    type: Date,
    default: Date.now
  }
});
// creating a user table and a schema
const User = mongoose.model('User', UserSchema);

//matching a user
module.exports = User;
```

4. Routes

a. getter.js

```
//
const router = require('express').Router()

//connecting to a database
const db = require('../config/connect')

// getting into trans table in the database

router.get('/trans',(req,res)=>{
    db.query('select * from trans',(err,data)=>{
        if(err){
            console.log(err);
        }
        else{
            var coll={
                "name":"transformers",
                "type":"FeatureCollection",
```

```
                "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OGC:1.
3:CRS84"}},
                "features":[]
            }
            data.rows.forEach(element=>{
                var feature={
                    "type":'Feature',
                    "properties":element,
                    "geometry":{"type":"Point","coordinates":[element.lat,element
.lon]}
                }
                coll.features.push(feature)
            })
            res.send(coll)
        }
    })
})
// getting into poles table
router.get('/pole',(req,res)=>{
    db.query('select * from poles',(err,data)=>{
        if(err){
            console.log(err);
        }
        else{
            var coll={
                "name":"poles",
                "type":"FeatureCollection",
                "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OGC:1.
3:CRS84"}},
                "features":[]
            }
            data.rows.forEach(element=>{
                var feature={
                    "type":'Feature',
                    "properties":element,
                    "geometry":{"type":"Point","coordinates":[element.lat,element
.lon]}
                }
                coll.features.push(feature)
            })
            res.send(coll)
        }
    })
})
//getting into facilities table
```

```
router.get('/fac',(req,res)=>{
    db.query('select * from facility',(err,data)=>{
        if(err){
            console.log(err);
        }
        else{
            var coll={
                "name":"facility",
                "type":"FeatureCollection",
                "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OGC:1.
3:CRS84"}},
                "features":[]
            }
            data.rows.forEach(element=>{
                var feature={
                    "type":'Feature',
                    "properties":element,
                    "geometry":{"type":"Point","coordinates":[element.lat,element
.lon]}
                }
                coll.features.push(feature)
            })
            res.send(coll)
        }
    })
})
// getting into faults table

router.get('/faults',(req,res)=>{
    const qry = `select *
    from faults,facility
    where faults."meter_num"=facility."meter_num"
    `
    console.log(qry);
    db.query(qry,(err,data)=>{
            if(err){
                console.log(err);
            }
            else{
                var coll={
                    "name":"faults",
                    "type":"FeatureCollection",
                    "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OG
C:1.3:CRS84"}},
                    "features":[]
```

```
                }
                data.rows.forEach(element=>{
                    var feature={
                        "type":'Feature',
                        "properties":element,
                        "geometry":{"type":"Point","coordinates":[element.lat,ele
ment.lon]}
                    }
                    coll.features.push(feature)
                })
                res.send(coll)
            }

    })
})




module.exports = router
```

b. index.js

```
const express = require('express');
const router = express.Router();
const { ensureAuthenticated, forwardAuthenticated } = require('../config/auth');
const db = require('../config/connect')
// Welcome Page
router.get('/', forwardAuthenticated, (req, res) => res.render('welcome'));

// Dashboard
router.get('/dashboard', ensureAuthenticated, (req, res) =>
  res.render('dashboard', {
    user: req.user
  })
);
router.get('/report',ensureAuthenticated,(req,res)=>{
  res.render('report',{user:req.user})
})
router.post('/report',ensureAuthenticated,(req,res)=>{
  console.log(req.body);

// inserting data into faults table
```

```
  db.query(`INSERT INTO faults("fault_desc","reported_by","meter_num","date","sta
tus") VALUES('${req.body.fault}','${req.body.name}','${req.body.mtno}','${req.bod
y.date}','Active')`,(err,data)=>{
      if(err){
          console.log(err);
      }else{
        req.flash('success_msg', 'Successfully Reported Fault');
          res.redirect('/dashboard');
      }
  })
})

router.get('/faults',(req,res)=>{
  res.render('faultmap')
})//show faults data on a faultmap

module.exports = router;
```

c. users.js

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const passport = require('passport');


const { forwardAuthenticated } = require('../config/auth');
const db = require('../config/connect');

// Login Page
router.get('/login', forwardAuthenticated, (req, res) => res.render('login'));

// Register Page
router.get('/register', forwardAuthenticated, (req, res) => res.render('register'
));

// Register
router.post('/register', (req, res) => {
  console.log(req.body);
  const { name, email,meter_num, password, password2 } = req.body;
  let errors = [];

 if (!name || !email || !meter_num|| !password || !password2) {
```

```
    errors.push({ msg: 'Please enter all fields' });
  }

  if (password != password2) {
    errors.push({ msg: 'Passwords do not match' });
  }

  if (password.length < 6) {
    errors.push({ msg: 'Password must be at least 6 characters' });
  }

  if (errors.length > 0) {
    res.render('register', {
      errors,
      name,
      email,
      meter_num,
      password,
      password2
    });
  } else {
    db.query(`select * from users where "meter_num"='${meter_num}'`,(error,result
s)=>{
      if(error){
        console.log(error);
      }
      else{
        console.log(results.rows);
        const user=results.rows[0]
        if (user) {
          errors.push({ msg: 'Meter Number already registered' });
          res.render('register', {
            errors,
            name,
            email,
            meter_num,
            password,
            password2
          });
        } else {
          const newUser = {
            name,
            email,
            meter_num,
            password
```

```
          };

          bcrypt.genSalt(10, (err, salt) => {
            bcrypt.hash(newUser.password, salt, (err, hash) => {
              if (err) throw err;
              newUser.password = hash;
              db.query(`insert into users("name","email","meter_num","password")
values('${newUser.name}','${newUser.email}','${newUser.meter_num}','${newUser.pas
sword}')`,(error,results)=>{
                if(error){
                  console.log(error);
                }else{
                  req.flash(
                    'success_msg',
                    'You are now registered and can log in'
                  );
                  res.redirect('/users/login');
                }
              })
            });
          });
        }
      }
    })
  }
});

// Login
router.post('/login', (req, res, next) => {
  passport.authenticate('local', {
    successRedirect: '/dashboard',
    failureRedirect: '/users/login',
    failureFlash: true
  })(req, res, next);
});

// Logout
router.get('/logout', (req, res) => {
  req.logout();
  req.flash('success_msg', 'You are logged out');
  res.redirect('/users/login');
});

module.exports = router;
```

5. views (Web-server)

a. messages.ejs

```
 <% if(locals.errors){ %> <% errors.forEach(function(error) { %>
<div class="alert alert-warning alert-dismissible fade show" role="alert">
  <%= error.msg %>
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
<% }); %> <% } %> <% if(success_msg != ''){ %>
<div class="message" role="alert">
  <%= success_msg %>
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
<% } %> <% if(error_msg != ''){ %>
<div class="alert alert-danger alert-dismissible fade show" role="alert">
  <%= error_msg %>
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
<% } %> <% if(error != ''){ %>
<div class="alert alert-danger alert-dismissible fade show" role="alert">
  <%= error %>
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
<% } %>
```

b. dashboard.ejs

```
<style>
    body{

    }
    h1{
        padding-top: 70px;
        text-align: center;
        font-family: Helvetica;
        font-size: 56px;
        color: rgb(255, 255, 255);
    }
    p{
        font-family: Helvetica;
        font-size: 26px;
        padding-top: 70px;
        text-align: center;
    }
    .lead{
        color: rgb(7, 92, 32);
    }
    .center{
        padding-top: 70px;
        align-items: center;
        margin-left: 35%;
    }
    .report_container{
        font-size: 20px;
        margin-left: 50px;
        background-color: #095c09;
        float: left;
        width: 25%;
        height:8em;
        width:12em;
        display:-webkit-box;
        display:-ms-flexbox;
        display:flex;
        -webkit-box-orient:vertical;
        -webkit-box-direction:normal;
        -ms-flex-direction:column;
        flex-direction:column;
        position:relative;
        -webkit-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
        -o-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
```

```
        transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
        border-radius:16px;
        overflow:hidden;
        -webkit-box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
        box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
    }
    .report_container:hover{
        box-shadow:  10px 10px 17px #e1e1e3, -15px -15px 27px #7f9e7f;
    }
    p{
        color: rgb(247, 238, 238);
        font-family: Helvetica;
        font-size: 18px;
        padding: 35px;
        text-align: center;
    }
    header{
        background:rgb(10, 63, 0);
    }
    .message{
        position: relative;
    }


</style>


<body>
    <header>
        <h1 class="mt-4">Dashboard</h1>


    </header>

    <p class="lead mb-3">Welcome <%= user.name %></p>


    <div class="center">

        <a href="/report" class="links">
            <div class="report_container">
                <p>Report A fault</p>
            </div></a>
```

```
        <a href="/users/logout" class="links">
            <div class="report_container">
                <p>Logout</p>
            </div></a>
    </div>

    <% include ./partials/messages %>




</body>
```

c. faultmap.ejs / Vizualization page

```
<!DOCTYPE html>
<html>

<head>
    <title>Faults</title>




    <!-- jQuery -->
    <script type="text/javascript" src="https://code.jquery.com/jquery-
1.11.0.min.js"></script>

    <!-- Openlayers -->
    <link rel="stylesheet" href="https://openlayers.org/en/latest/css/ol.css" />
    <script type="text/javascript" src="https://openlayers.org/en/latest/build/ol
.js"></script>
    <script
        src="https://cdn.polyfill.io/v2/polyfill.min.js?features=requestAnimation
Frame,Element.prototype.classList,URL,Object.assign">
    </script>

    <!-- ol-ext -->
    <link rel="stylesheet" href="https://cdn.rawgit.com/Viglino/ol-
ext/master/dist/ol-ext.min.css" />
    <script type="text/javascript" src="https://cdn.rawgit.com/Viglino/ol-
ext/master/dist/ol-ext.min.js"></script>
```

```
    <style>
        .map {
            height: 400px;
            width: 100%;
        }

        .overlay-container {
            background-color: blueviolet;
            width: 150px;
            color: rgb(0, 255, 200);
            text-align: center;
            border-radius: 1px;
            padding: 2px 0;
            position: absolute;
            z-index: 1;
            bottom: 100%;
        }
    </style>


</head>

<body>

    <!-- DIV pour la carte -->
    <div id="map" style="width:100%; margin-top: 200px; height:400px"></div>

    <script type="text/javascript">
        // Layers
        var layers = [
            new ol.layer.Tile({
                source: new ol.source.OSM()
            })
        ]
        var myExtent = [34.961673,-15.883286 ,34.982369,-15.867291 ]

        // The map
        var map = new ol.Map({
            target: 'map',
            view: new ol.View({
                zoom: 9,
                center: [34.12, -13.55],
             extent:myExtent,
                projection: 'EPSG:4326'
            }),
```

```
        layers: layers
    });

    map.getView().fit(myExtent , map.getSize());
    var url = '/data/trans'
    var vsource = new ol.source.Vector({
        url: url,
        format: new ol.format.GeoJSON()
    })



    var trans = new ol.layer.VectorImage({
        source: vsource,
        visible: true,
        title: 'transformers',
        style: new ol.style.Style({
            image: new ol.style.Circle({
            fill: new ol.style.Fill({
                color: 'rgb(0, 255, 200)'
                }),
            stroke: new ol.style.Stroke({
                color: 'rgb(5,230,10)',
                width: 1.25
            }),
            radius: 9
            })
        })
    });


    var url2 = '/data/pole'
    var vsource2 = new ol.source.Vector({
        url: url2,
        format: new ol.format.GeoJSON()
    })
    var pole = new ol.layer.VectorImage({
        source: vsource2,
        visible: true,
        title: 'poles'
        ,
        style: new ol.style.Style({
            image: new ol.style.Circle({
            fill: new ol.style.Fill({
                color: 'rgb(100, 30, 230)'
```

```
                }),
            stroke: new ol.style.Stroke({
                color: 'rgb(100, 30, 230)',
                width: 1.25
            }),
            radius: 9
            })
        })
    });


    var url3 = '/data/fac'
    var vsource3 = new ol.source.Vector({
        url: url3,
        format: new ol.format.GeoJSON()
    })
    var fac = new ol.layer.VectorImage({
        source: vsource3,
        visible: true,
        title: 'facilities',
        style: new ol.style.Style({
            image: new ol.style.Circle({
            fill: new ol.style.Fill({
                color: 'rgb(255, 238, 0)'
                }),
            stroke: new ol.style.Stroke({
                color: 'rgb(0, 0, 255)',
                width: 1.25
            }),
            radius: 14
            })
        })
    });



    var url4 = '/data/faults'
    var vsource4 = new ol.source.Vector({
        url: url4,
        format: new ol.format.GeoJSON()
    })
    var faults = new ol.layer.VectorImage({
        source: vsource4,
        visible: true,
        title: 'faults'
```

```
    ,
    style: new ol.style.Style({
        image: new ol.style.Circle({
        fill: new ol.style.Fill({
            color: 'rgb(206, 37, 37)'
            }),
        stroke: new ol.style.Stroke({
            color: 'rgb(206, 37, 37)',
            width: 1.25
        }),
        radius: 5
        })
    })
});









map.addLayer(fac);
map.addLayer((faults))
map.addLayer(trans);
map.addLayer(pole);
console.log(trans);
console.log(fac);
console.log(pole);
console.log(faults);

var legend = new ol.legend.Legend({
    title: 'Legend'
})
var legendCtrl = new ol.control.Legend({
    legend: legend,
    collapsed: false
});
map.addControl(legendCtrl);
var select = new ol.interaction.Select({})
map.addInteraction(select)
var search = new ol.control.SearchFeature(
{
    source: vsource4,
    property:'meter_num'
```

```
        });
    map.addControl(search);
    // Select feature when click on the reference index
    search.on('select', function(e)
        {   select.getFeatures().clear();
            select.getFeatures().push (e.search);
            var p = e.search.getGeometry().getFirstCoordinate();
            map.getView().animate({ center:p });
        });

        // Select feature when click on the reference index


        var popup = new ol.Overlay.PopupFeature({
            popupClass: 'default anim',
            select: select,
            canFix: true,
            template: {
                title: function (f) {
                        return f.get('meter_num');
                    },
                attributes: // [ 'region', 'arrond', 'cantons', 'communes', 'pop'
 ]
                {
                    'reported_by':{title:'Reporter'},
                    'fault_desc':{title:'Fault'},
                    'date': {title: 'Date'},
                    'status':{title:'Status'}

                }
        }
        })
        map.addOverlay(popup)
    </script>

</body>

</html>
```

d. layout.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css
/all.css" integrity="sha384-
UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/" crossorigin="an
onymous">
    <link
      rel="stylesheet"
      href="https://bootswatch.com/4/journal/bootstrap.min.css"
    />
    <title>Node.js & Passport Login App</title>
  </head>
  <body>
    <div class="container"><%- body %></div>

    <script
      src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
      integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min
.js"
      integrity="sha384-
wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js
"
      integrity="sha384-
B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
      crossorigin="anonymous"
    ></script>
  </body>
</html>
```

e. login.ejs / Log in page

```
<!DOCTYPE html>
<html lang="en">
```

```
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css
/all.css" integrity="sha384-
UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/" crossorigin="an
onymous">
    <link
      rel="stylesheet"
      href="https://bootswatch.com/4/journal/bootstrap.min.css"
    />
    <title>Login Page</title>
    <style>
      body{
        padding: 20px;
        align-items: center;
        text-align: center;
      }
        .form-group{



        }
        .content-container{
          margin-top: 15px;
          margin-left: 5%;
          display: inline-block;
          padding: 20px;
          height:fit-content;
          width:24em;
          display:-webkit-box;
          display:-ms-flexbox;
          display:flex;
          -webkit-box-orient:vertical;
          -webkit-box-direction:normal;
          -ms-flex-direction:column;
          flex-direction:column;
          position:relative;
          -webkit-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
          -o-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
          transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
          border-radius:16px;
          overflow:hidden;
          -webkit-box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
```

```
          box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
        }
        h1{
          padding-top: 70px;
          text-align: center;
          font-family: Helvetica;
          font-size: 56px;
          color: rgb(0, 0, 0);
      }
        label{
            text-align: left;
            text-decoration: solid;
            text-shadow: '8px';
            size: 16px;
            font-family: Helvetica;
            margin-right: 14px;
        }
        button[type=submit]{
            margin-top: 20px;
            display:inline-block;
            padding:0.5em 3em;
            border:0.16em solid #FFFFFF;
            margin:0 0.3em 0.3em 0;
            box-sizing: border-box;
            text-decoration:none;
            text-transform:uppercase;
            font-family:'Roboto',sans-serif;
            font-weight:400;
            color: black;
            text-align:center;
            transition: all 0.15s;
        }

    </style>
  </head>
  <body>
    <div class="container">
      <div class="row mt-5">
        <div class="col-md-6 m-auto">
          <div class="card card-body">
            <h1 class="text-center mb-3"><i class="fas fa-sign-in-
alt"></i>  Login</h1>
            <% include ./partials/messages %>
            <div class="content-container">
              <form action="/users/login" method="POST">
```

```
              <div class="form-group">
                <label for="email">Email</label>
                <input
                  type="email"
                  id="email"
                  name="email"
                  class="form-control"
                  placeholder="Enter Email"
                />
              </div>
              <div class="form-group">
                <label for="password">Password</label>
                <input
                  type="password"
                  id="password"
                  name="password"
                  class="form-control"
                  placeholder="Enter Password"
                />
              </div>
              <button type="submit" class="btn btn-primary btn-
block">Login</button>
            </form>

          </div>

          <p class="lead mt-4">
            Not Registered? <a href="/users/register">Register here</a>
          </p>
        </div>
      </div>
    </div>
  </div>

  <script
    src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
    integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
    crossorigin="anonymous"
  ></script>
  <script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min
.js"
    integrity="sha384-
wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut"
```

```
      crossorigin="anonymous"
    ></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js
"
      integrity="sha384-
B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
      crossorigin="anonymous"
    ></script>

</body>
```

f. register.ejs / registration page

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css
/all.css" integrity="sha384-
UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/" crossorigin="an
onymous">
    <link
      rel="stylesheet"
      href="https://bootswatch.com/4/journal/bootstrap.min.css"
    />
    <title>Registration Page</title>
    <style>
      body{
        padding: 20px;
        align-items: center;
        text-align: center;
      }
        .form-group{



        }
        .content-container{
          margin-top: 150px;
          margin-left: 5%;
          display: inline-block;
          padding: 20px;
```

```
      height:fit-content;
      width:24em;
      display:-webkit-box;
      display:-ms-flexbox;
      display:flex;
      -webkit-box-orient:vertical;
      -webkit-box-direction:normal;
      -ms-flex-direction:column;
      flex-direction:column;
      position:relative;
      -webkit-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
      -o-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
      transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
      border-radius:16px;
      overflow:hidden;
      -webkit-box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
      box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
    }
    h1{
      padding-top: 70px;
      text-align: center;
      font-family: Helvetica;
      font-size: 56px;
      color: rgb(0, 0, 0);
  }
    label{
        text-align: left;
        text-decoration: solid;
        text-shadow: '8px';
        size: 16px;
        font-family: Helvetica;
        margin-right: 14px;
    }
    button[type=submit]{
        margin-top: 20px;
        display:inline-block;
        padding:0.5em 3em;
        border:0.16em solid #FFFFFF;
        margin:0 0.3em 0.3em 0;
        box-sizing: border-box;
        text-decoration:none;
        text-transform:uppercase;
        font-family:'Roboto',sans-serif;
        font-weight:400;
        color: black;
```

```
            text-align:center;
            transition: all 0.15s;
        }

    </style>
</head>
<body>
    <div class="container">

        <div class="row mt-5">
          <div class="col-md-6 m-auto">
            <div class="card card-body">
              <h1 class="text-center mb-3">
                <i class="fas fa-user-plus"></i> Register
              </h1>
              <% include ./partials/messages %>
              <div class="content-container">
                <form action="/users/register" method="POST">
                  <div class="form-group">
                    <label for="name">Name</label>
                    <input
                      type="name"
                      id="name"
                      name="name"
                      class="form-control"
                      placeholder="Enter Name"
                      value="<%= typeof name != 'undefined' ? name : '' %>"
                    />
                  </div>
                  <div class="form-group">
                    <label for="email">Email</label>
                    <input
                      type="email"
                      id="email"
                      name="email"
                      class="form-control"
                      placeholder="Enter Email"
                      value="<%= typeof email != 'undefined' ? email : '' %>"
                    />
                  </div>
                  <div class="form-group">
                    <label for="meter_num">Meter_No</label>
                    <input
                      type="text"
                      id="meter_num"
```

```
                    name="meter_num"
                    class="form-control"
                    placeholder="Enter Meter Number"
                    value="<%= typeof meter_num != 'undefined' ? meter_num : ''
 %>"
                  />
                </div>
                <div class="form-group">
                  <label for="password">Password</label>
                  <input
                    type="password"
                    id="password"
                    name="password"
                    class="form-control"
                    placeholder="Create Password"
                    value="<%= typeof password != 'undefined' ? password : '' %
>"
                  />
                </div>
                <div class="form-group">
                  <label for="password2">Confirm Password</label>
                  <input
                    type="password"
                    id="password2"
                    name="password2"
                    class="form-control"
                    placeholder="Confirm Password"
                    value="<%= typeof password2 != 'undefined' ? password2 : ''
 %>"
                  />
                </div>
                <button type="submit" class="btn btn-primary btn-block">
                  Register
                </button>
              </form>

            </div>

            <p class="lead mt-
4">Have An Account? <a href="/users/login">Login</a></p>
          </div>
        </div>
      </div>
```

```
    </div>

    <script
      src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
      integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min
.js"
      integrity="sha384-
wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js
"
      integrity="sha384-
B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
      crossorigin="anonymous"
    ></script>
  </body>
```

g. report.ejs / fault reporting page

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css
/all.css" integrity="sha384-
UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/" crossorigin="an
onymous">
    <link
      rel="stylesheet"
```

```
  href="https://bootswatch.com/4/journal/bootstrap.min.css"
/>
<title> Report a Fault</title>
<style>
    body{
        text-align: center;
        padding: 30px;

    }
    #card{
        display: inline-block;
        padding: 25px 50px 75px 100px;
    }
    header{

    }
    h1{
        text-align: center;
        text-decoration: solid;
        text-shadow: '8px';
        color: black;
        font-size: 56px;
        font-family: Helvetica;
    }
    label{
        padding: 20px;
        text-align: left;
        text-decoration: solid;
        text-shadow: '8px';
        size: 18px;
    }
    textarea{
        margin-top: 20px;
        width: fit-content;
        height: fit-content;
        padding: 12px 20px;
        box-sizing: border-box;
        border: 2px solid rgb(0, 0, 0);
        border-radius: 4px;
        background-color: #ffffff;
        resize: none;
    }
    input[type=text]{
        margin-top: 30px;
    }
```

```
      input[type=date]{
          margin-top: 30px;
      }
      input[type=submit]{
          margin-top: 20px;
          display:inline-block;
          padding:0.5em 3em;
          border:0.16em solid #FFFFFF;
          margin:0 0.3em 0.3em 0;
          box-sizing: border-box;
          text-decoration:none;
          text-transform:uppercase;
          font-family:'Roboto',sans-serif;
          font-weight:400;
          color: black;
          text-align:center;
          transition: all 0.15s;
      }
      input[type=submit]:hover{
  color:rebeccapurple;
  border-color:white;
  }
  input[type=submit]:active{
  color:#BBBBBB;
  border-color:#BBBBBB;
  }
  .content{
  height:fit-content;
    width:44em;
    display:-webkit-box;
    display:-ms-flexbox;
    display:flex;
    -webkit-box-orient:vertical;
    -webkit-box-direction:normal;
    -ms-flex-direction:column;
    flex-direction:column;
    position:relative;
    -webkit-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
    -o-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
    transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
    border-radius:16px;
    overflow:hidden;
    -webkit-box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
    box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
      }
```

```
    </style>
  </head>
  <body>
    <div class="container">
        <header>
            <h1 class="mt-4">
                Report a Fault
            </h1>
        </header>
        <div id="card">
            <div id="heading">

            </div>
           <div class="content">
                <form method="POST">
                    <label for="name">Your Name</label>
                    <input type="text" id="name" name="name" value="<%=user.name
%>">
                    <br>
                    <label for="mtno">Meter Number</label>
                    <input type="text" id="mtno" name="mtno" value="<%=user.meter
_num %>">
                    <br>
                    <label for="fdate">Date Started/Noticed</label>
                    <input type="date" id="fdate" name="date" placeholder="">
                    <br>
                    <textarea id="fault" name="fault" placeholder="fault is....">
</textarea>
                    <br>
                    <input type="submit" value="Submit">

                </form>
            </div>

        </div>

    </div>

    <script
      src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
      integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
      crossorigin="anonymous"
    ></script>
```

```
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min
.js"
      integrity="sha384-
wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js
"
      integrity="sha384-
B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
      crossorigin="anonymous"
    ></script>

</body>
```

h. welcome.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <title>GIS Electricity</title>

  <!-- CSS  -->
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="style
sheet">
  <link href="css/materialize.css" type="text/css" rel="stylesheet" media="screen
,projection"/>
  <link href="css/style.css" type="text/css" rel="stylesheet" media="screen,proje
ction"/>
</head>
<body>
  <script src="css/file.js">

  </script>
  <nav class="indigo" role="navigation">
    <div class="nav-wrapper container">
      <a id="logo-container" href="#" class="brand-logo white-text" style="font-
size: 16px;">GIS-BASED ELECTRICITY MANAGEMENT </a>
```

```
      <ul class="right hide-on-med-and-down">
        <li><a class="white-text" href="#">Home</a></li>
        <li><a class="white-text" href="#">About us</a></li>
        <li><a class="white-text" href="#">Contact Us</a></li>
        <li><a class="white-text" href="users/login">Log in</a></li>
        <li><a class="white-text" href="users/register">Sign Up</a></li>
      </ul>

      <ul id="nav-mobile" class="sidenav indigo darken-3">
        <li><a class="white-text" href="#">Home</a></li>
        <li><a class="white-text" href="#">About us</a></li>
        <li><a class="white-text" href="#">Contact Us</a></li>
        <li><a class="white-text" href="users/login">Log in</a></li>
        <li><a class="white-text" href="users/register">Sign Up</a></li>
      </ul>
      <a href="#" data-target="nav-mobile" class="sidenav-
trigger"><i class="material-icons">menu</i></a>
    </div>
  </nav>

  <div id="index-banner" class="parallax-container">
    <div class="section no-pad-bot">
      <div class="container">
        <br><br>
        <h1 class="header center white-text text-darken-2">Report a Fault</h1>
        <div class="row center">
          <h5 class="header col s12 light">Register and report, we will be there!
</h5>
        </div>
        <div class="row center">
          <a href="/users/register" id="download-button" class="btn-large waves-
effect waves-light indigo darken-1">Report here</a>
        </div>
        <br><br>

      </div>
    </div>
    <div class="parallax"><img src="image.jpg" alt="Unsplashed background img 1">
</div>
  </div>


  <div class="container">
    <div class="section">
```

```
      <!--   Icon Section   -->
      <div class="row">
        <div class="col s12 m4">
          <div class="icon-block">
            <h2 class="center brown-text"><i class="material-
icons">flash_on</i></h2>
            <h5 class="center">We will fix in no time</h5>

            <p class="light">Stay calm if you have reported a fault, it will be a
ddressed in within 5 hours of the reported time</p>
          </div>
        </div>

        <div class="col s12 m4">
          <div class="icon-block">
            <h2 class="center brown-text"><i class="material-
icons">group</i></h2>
            <h5 class="center">User Friendly</h5>

            <p class="light">We are providing a User friendly way for you to repo
rt your faults the WAIT for for our team for quick management at your door step.<
/p>
          </div>
        </div>

        <div class="col s12 m4">
          <div class="icon-block">
            <h2 class="center brown-text"><i class="material-
icons">settings</i></h2>
            <h5 class="center">Easy to work with</h5>

            <p class="light">We have provided a simple and unique way help you wi
th continous Electricity.</p>
          </div>
        </div>
      </div>

    </div>
  </div>


  <div class="parallax-container valign-wrapper">
    <div class="section no-pad-bot">
      <div class="container">
        <div class="row center">
```

```
          <h5 class="header col s12 light">A modern responsive front-
end framework based on Material Design</h5>
        </div>
      </div>
    </div>
    <div class="parallax"><img src="background2.jpg" alt="Unsplashed background i
mg 2"></div>
  </div>

  <div class="container">
    <div class="section">

      <div class="row">
        <div class="col s12 center">
          <h3><i class="mdi-content-send brown-text"></i></h3>
          <h4>Contact Us</h4>
          <p class="left-
align light">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam scel
erisque id nunc nec volutpat. Etiam pellentesque tristique arcu, non consequat ma
gna fermentum ac. Cras ut ultricies eros. Maecenas eros justo, ullamcorper a sapi
en id, viverra ultrices eros. Morbi sem neque, posuere et pretium eget, bibendum
sollicitudin lacus. Aliquam eleifend sollicitudin diam, eu mattis nisl maximus se
d. Nulla imperdiet semper molestie. Morbi massa odio, condimentum sed ipsum ac, g
ravida ultrices erat. Nullam eget dignissim mauris, non tristique erat. Vestibulu
m ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;</p
>
        </div>
      </div>

    </div>
  </div>


  <div class="parallax-container valign-wrapper">
    <div class="section no-pad-bot">
      <div class="container">
        <div class="row center">
          <h5 class="header col s12 light">A modern responsive front-
end framework based on Material Design</h5>
        </div>
      </div>
    </div>
    <div class="parallax"><img src="background3.jpg" alt="Unsplashed background i
mg 3"></div>
  </div>
```

```
  <footer class="page-footer teal">
    <div class="container">
      <div class="row">
        <div class="col l6 s12">
          <h5 class="white-text">Company Bio</h5>
          <p class="grey-text text-lighten-
4">We are a team of college students working on this project like it's our full t
ime job. Any amount would help support and continue development on this project a
nd is greatly appreciated.</p>


        </div>
        <div class="col l3 s12">
          <h5 class="white-text">Settings</h5>
          <ul>
            <li><a class="white-text" href="#">Home</a></li>
            <li><a class="white-text" href="#">About us</a></li>
            <li><a class="white-text" href="#">Contact Us</a></li>
            <li><a class="white-text" href="users/login">Log in</a></li>
            <li><a class="white-text" href="users/register">Sign Up</a></li>
          </ul>
        </div>
        <div class="col l3 s12">
          <h5 class="white-text">Connect</h5>
          <ul>

          </ul>
        </div>
      </div>
    </div>
    <div class="footer-copyright">
      <div class="container">
      Made with <a class="brown-text text-lighten-
3" href="http://materializecss.com">Materialize</a>
      </div>
    </div>
  </footer>


  <!--  Scripts-->
  <script src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="js/materialize.js"></script>
  <script src="js/init.js"></script>
```

```
  </body>
</html>
```

B. Back-end (Administrator)

1. Application Server

a. app.js

```
const express = require('express');
const passport = require('passport');
const flash = require('connect-flash');
const session = require('express-session');
const app = express();

// Passport Config
require('./config/passport')(passport);


// EJS

app.set('view engine', 'ejs');

// Express body parser
app.use(express.urlencoded({ extended: true }));
app.use(express.static(__dirname + '/public'));
//app.set('views', path.join(__dirname, 'views'));
// Express session
app.use(
  session({
    secret: 'secret',
    resave: true,
    saveUninitialized: true
  })
);

// Passport middleware
app.use(passport.initialize());
```

```
app.use(passport.session());



// Connect flash
app.use(flash());

// Global variables
app.use(function(req, res, next) {
  res.locals.success_msg = req.flash('success_msg');
  res.locals.error_msg = req.flash('error_msg');
  res.locals.error = req.flash('error');
  next();
});

// Routes
app.use('/', require('./routes/index.js'));
app.use('/data', require('./routes/getter.js'));
app.use('/users', require('./routes/users.js'));

const PORT = process.env.PORT || 5000;

app.listen(PORT, console.log(`Server running on  ${PORT}`));
```

2. configuration

a. auth.js

```
module.exports = {
  ensureAuthenticated: function(req, res, next) {
    if (req.isAuthenticated()) {
      return next();
    }
    req.flash('error_msg', 'Please log in to view that resource');
    res.redirect('/users/login');
  },
  forwardAuthenticated: function(req, res, next) {
    if (!req.isAuthenticated()) {
      return next();
    }
    res.redirect('/dashboard');
```

```
  }
};
```

b. connect.js

```
const {Pool} = require('pg')
// connecting with a database
const db = new Pool({
    database:'Electricity Management',
    password:'1234',
    port:5433,
    user:'postgres',
})

module.exports = db;
```

c. keys.js

```
dbPassword = 'mongodb+srv://YOUR_USERNAME_HERE:'+ encodeURIComponent('YOUR_PASSWO
RD_HERE') + '@CLUSTER_NAME_HERE.mongodb.net/test?retryWrites=true';

module.exports = {
    mongoURI: dbPassword
};
```

d. passport.js

```
const LocalStrategy = require('passport-local').Strategy;
const bcrypt = require('bcryptjs');
const db = require('./connect')
// Load User model
```

```
module.exports = function(passport) {
  passport.use(
    new LocalStrategy({ usernameField: 'email' }, (email, password, done) => {

      // Match user
      db.query(`select * from admin where "email"='${email}'`,(err,results)=>{
        if(err){
          console.log(err);
          return done(null, false, { message: 'That email is not registered' });
        }else{
          console.log(results.rows);
          const user=results.rows[0]
          bcrypt.compare(password, user.password, (err, isMatch) => {
            if (err) throw err;
            if (isMatch) {
              return done(null, user);
            } else {
              return done(null, false, { message: 'Password incorrect' });
            }
          });

        }
      })
    })
  );

  passport.serializeUser(function(user, done) {
    console.log('serial');
    done(null, user.email);
  });

  passport.deserializeUser(function(id, done) {
    db.query(`select * from admin where "email"='${id}'`,(error,results)=>{
      if(error){
        console.log('error on deserial');
      }else{
        const user=results.rows[0]
        done(error,user)
      }
    })
  });
};
```

3. models

a. user.js

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  date: {
    type: Date,
    default: Date.now
  }
});

const User = mongoose.model('User', UserSchema);

module.exports = User;
```

4. routes / webserver.

a. getter.js

```
const router = require('express').Router()
const db = require('../config/connect')



router.get('/trans',(req,res)=>{
    db.query('select * from trans',(err,data)=>{
        if(err){
            console.log(err);
        }
        else{
            var coll={
                "name":"transformers",
                "type":"FeatureCollection",
                "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OGC:1.
3:CRS84"}},
                "features":[]
            }
            data.rows.forEach(element=>{
                var feature={
                    "type":'Feature',
                    "properties":element,
                    "geometry":{"type":"Point","coordinates":[element.lat,element
.lon]}
                }
                coll.features.push(feature)
            })
            res.send(coll)
        }
    })
})

router.get('/pole',(req,res)=>{
    db.query('select * from poles',(err,data)=>{
        if(err){
            console.log(err);
        }
        else{
            var coll={
```

```
                "name":"poles",
                "type":"FeatureCollection",
                "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OGC:1.
3:CRS84"}},
                "features":[]
            }
            data.rows.forEach(element=>{
                var feature={
                    "type":'Feature',
                    "properties":element,
                    "geometry":{"type":"Point","coordinates":[element.lat,element
.lon]}
                }
                coll.features.push(feature)
            })
            res.send(coll)
        }
    })
})

router.get('/fac',(req,res)=>{
    db.query('select * from facility',(err,data)=>{
        if(err){
            console.log(err);
        }
        else{
            var coll={
                "name":"facility",
                "type":"FeatureCollection",
                "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OGC:1.
3:CRS84"}},
                "features":[]
            }
            data.rows.forEach(element=>{
                var feature={
                    "type":'Feature',
                    "properties":element,
                    "geometry":{"type":"Point","coordinates":[element.lat,element
.lon]}
                }
                coll.features.push(feature)
            })
            res.send(coll)
        }
    })
```

```
})


router.get('/faults',(req,res)=>{
    const qry = `select *
    from faults,facility
    where faults."meter_num"=facility."meter_num"
    `
    console.log(qry);
    db.query(qry,(err,data)=>{
            if(err){
                console.log(err);
            }
            else{
                var coll={
                    "name":"faults",
                    "type":"FeatureCollection",
                    "crs":{"type":"name","properties":{"name":"urn:ogc:def:crs:OG
C:1.3:CRS84"}},
                    "features":[]
                }
                data.rows.forEach(element=>{
                    var feature={
                        "type":'Feature',
                        "properties":element,
                        "geometry":{"type":"Point","coordinates":[element.lat,ele
ment.lon]}
                    }
                    coll.features.push(feature)
                })
                res.send(coll)
            }

    })
})
router.get('/latest',(req,res)=>{
    db.query(`select "date" from faults order by "date" desc limit 1`,(err,data)=
>{
      if(err){
        console.log(err);
      }else{
        res.send(data.rows[0])
      }
    })
  })
```

```
module.exports = router
```

b. index.js

```
const express = require('express');
const router = express.Router();
const axios = require('axios')
const { ensureAuthenticated, forwardAuthenticated } = require('../config/auth');
const db = require('../config/connect')
// Welcome Page
router.get('/', forwardAuthenticated, (req, res) => res.render('welcome'));

// Dashboard
router.get('/dashboard', ensureAuthenticated, (req, res) =>
  res.render('dashboard', {
    user: req.user
  })
);


router.get('/faults',(req,res)=>{
  res.render('faultmap')
})


router.get("/list",(req,res)=>{


  db.query("select * from faults",(err,data)=>{
    if(err){
      console.log(err);
    }else{
      const faults = data.rows
      res.render("faultlist",{faults})
    }
```

```
  })

})


router.get('/delete/:id',(req,res)=>{
  db.query(`DELETE FROM faults WHERE "meter_num" ='${req.params.id}';`,(err,data)
=>{
    if(err){
      console.log(err);
    }else{
      res.redirect("/list")
    }
  })
})




module.exports = router;
```

c. user.js

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const passport = require('passport');


const { forwardAuthenticated } = require('../config/auth');
const db = require('../config/connect');

// Login Page
router.get('/login', forwardAuthenticated, (req, res) => res.render('login'));

// Register Page
router.get('/register', forwardAuthenticated, (req, res) => res.render('register'
));

// Register
```

```
router.post('/register', (req, res) => {
  console.log(req.body);
  const { name, email, password, password2 } = req.body;
  let errors = [];

 if (!name || !email || !password || !password2) {
    errors.push({ msg: 'Please enter all fields' });
  }

  if (password != password2) {
    errors.push({ msg: 'Passwords do not match' });
  }

  if (password.length < 6) {
    errors.push({ msg: 'Password must be at least 6 characters' });
  }

  if (errors.length > 0) {
    res.render('register', {
      errors,
      name,
      email,
      password,
      password2
    });
  } else {
    db.query(`select * from admin where "email"='${email}'`,(error,results)=>{
      if(error){
        console.log(error);
      }
      else{
        console.log(results.rows);
        const user=results.rows[0]
        if (user) {
          errors.push({ msg: 'Email already registered' });
          res.render('register', {
            errors,
            name,
            email,
            password,
            password2
          });
        } else {
          const newUser = {
            name,
```

```
            email,
            password
          };

          bcrypt.genSalt(10, (err, salt) => {
            bcrypt.hash(newUser.password, salt, (err, hash) => {
              if (err) throw err;
              newUser.password = hash;
              db.query(`insert into admin("name","email","password") values('${ne
wUser.name}','${newUser.email}','${newUser.password}')`,(error,results)=>{
                if(error){
                  console.log(error);
                }else{
                  req.flash(
                    'success_msg',
                    'You are now registered and can log in'
                  );
                  res.redirect('/users/login');
                }
              })
            });
          });
        }
      }
    })
  }
});

// Login
router.post('/login', (req, res, next) => {
  passport.authenticate('local', {
    successRedirect: '/dashboard',
    failureRedirect: '/users/login',
    failureFlash: true
  })(req, res, next);
});

// Logout
router.get('/logout', (req, res) => {
  req.logout();
  req.flash('success_msg', 'You are logged out');
  res.redirect('/users/login');
});

module.exports = router;
```

5. views/ webserver UI

a. dashboard.ejs

```
<title>Dashboard</title>
<style>

    h1{
        padding-top: 70px;
        text-align: center;
        font-family: Helvetica;
        font-size: 56px;
        color: white;
    }
    .center{
        padding-top: 70px;
        align-items: center;
        margin-left: 35%;
    }
    .report_container p{
        color: white;
        font-family: Tahoma;
        font-size: medium;
    }
    .report_container{
        background:rgb(5, 63, 0) ;

        font-size: 20px;
        margin-left: 50px;
        float: left;
        width: 25%;
        height:8em;
        width:12em;
```

```
        display:-webkit-box;
        display:-ms-flexbox;
        display:flex;
        -webkit-box-orient:vertical;
        -webkit-box-direction:normal;
        -ms-flex-direction:column;
        flex-direction:column;
        position:relative;
        -webkit-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
        -o-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
        transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
        border-radius:16px;
        overflow:hidden;
        -webkit-box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
        box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
    }
    .report_container:hover{
        box-shadow:  10px 10px 17px #7f9e7f, -15px -15px 27px #7f9e7f;
    }
    p{
        color: rgb(15, 19, 15);
        font-family: Helvetica;
        font-size: 50px;
        padding-top: 70px;
        padding: 35px;
        text-align: center;
    }
    header{
        background:rgb(0, 63, 10);
    }
    .message{
        position: relative;
    }


</style>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/cs
s/bootstrap.min.css">

<body>
    <header>
        <h1 class="mt-4">Dashboard</h1>


    </header>
```

```
    <p class="lead-mb-3">Welcome <%= user.name %></p>


    <div class="center">

        <a href="/list" class="links">
            <div class="report_container">
                <p>View Fault list</p>
            </div></a>


        <a href="/faults" class="links">
            <div class="report_container">
                <p>View Fault map</p>
            </div></a>
    </div>

    <% include ./partials/messages %>



    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.m
in.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.b
undle.min.js"></script>
</body>
```

b. faultlist.ejs

```
<head>
    <title>Fault list</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dis
t/css/bootstrap.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.21.1/axios.min.js
" integrity="sha512-
bZS47S7sPOxkjU/4Bt0zrhEtWx0y0CRkhEp8IckzK+ltifIIE9EMIMTuT/mEzoIMewUINruDBIR/jJnbg
uonqQ==" crossorigin="anonymous"></script>
</head>
<style>
```

```
    h2{
        padding-top: 30px;
        text-align: center;
        font-family: Helvetica;
        font-size: 56px;
        color: rgb(0, 0, 0);
    }
    .center{
        padding-top: 30px;
        align-items: center;
    }
    html, body {
    width: 100%;
    background: cornsilk;
    }
    table {
    margin: 0 auto;
    }
    .faults table{
    border-collapse: collapse;
    margin: 25px 0;
    font-size: 0.9em;
    font-family: sans-serif;
    min-width: 400px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
    }
    .faults tr{
    background-color: #009879;
    color: #ffffff;
    text-align: left;
    }
.faults td {
    padding: 12px 15px;
}
.faults tr {
    border-bottom: 1px solid #dddddd;
}
.faults th{
    font-family: Georgia;
    font-size: large;
    padding: 12px 15px;
    color: rgb(0, 0, 0);
    background: rgb(255, 255, 255);
}
a{
```

```
    color: rgb(255, 255, 255);
}
p{
        font-family: Helvetica;
        font-size: small;
        text-align: center;
    }
    .links{
        position:absolute;
    top:50px;
    right: 10px;
    }
    .btn-secondary{
    background-color: rgb(39, 194, 52);
    color: white;
    border-radius: 15px;
    padding: 1em 1.5em;
    text-decoration: none;
    text-transform: uppercase;
    }
.update{
    padding-top: 50px;
}

</style>
<body>
    <header>
        <div class="update">
            <a href="/dashboard" class="btn btn-secondary">Dashboard</a>
            <p>Lastly updated on</p><p id="latest"></p>
          </div>
          <div class="links">

            <a href="/help" class="btn btn-secondary">Help</a>
            <a href="/users/logout" class="btn btn-secondary">Logout</a>

        </div>
        <h2>All Faults</h2>
    </header>
    <div class="center">
        <table class="faults">
            <tr>
                <th>
                    Status
                </th>
```

```
                <th>
                    Meter Number
                </th>
                <th>
                    Fault Description
                </th>
                <th>
                    Date
                </th>
                <th>
                    Reported By
                </th>

            </tr>
        <% if (faults.length > 0 ){ %>
            <% faults.forEach(fault=>{%>

                <tr>
                    <td> <%=fault.status %></td>
                    <td> <%=fault.meter_num %></td>
                    <td> <%=fault.fault_desc %></td>
                    <td> <%=fault.date %></td>
                    <td> <%=fault.reported_by %></td>
                    <td><p><a href="/delete/<%=fault.meter_num %>">Delete Fault</
a></p></td>
                    <td><p><a href="/faults">View Map</a></p></td>

                </tr>

        <% }) %>
        <% } %>
        </table>

    </div>
    <script>
        axios.get('/data/latest')
            .then(response => {
                console.log(response.data);
                document.getElementById("latest").innerHTML = response.data.date;
        });

    </script>
```

```
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"></script>

</body>
```

c. faultmap.ejs / visualization page

```
<!DOCTYPE html>
<html>

<head>
    <title>Faults</title>




    <!-- jQuery -->
    <script type="text/javascript" src="https://code.jquery.com/jquery-1.11.0.min.js"></script>

    <!-- Openlayers -->
    <link rel="stylesheet" href="https://openlayers.org/en/latest/css/ol.css" />
    <script type="text/javascript" src="https://openlayers.org/en/latest/build/ol.js"></script>
    <script
        src="https://cdn.polyfill.io/v2/polyfill.min.js?features=requestAnimationFrame,Element.prototype.classList,URL,Object.assign">
    </script>

    <!-- ol-ext -->
    <link rel="stylesheet" href="https://cdn.rawgit.com/Viglino/ol-ext/master/dist/ol-ext.min.css" />
    <script type="text/javascript" src="https://cdn.rawgit.com/Viglino/ol-ext/master/dist/ol-ext.min.js"></script>
    <style>
        .map {
            height: 400px;
```

```
            width: 100%;
        }

        .overlay-container {
            background-color: blueviolet;
            width: 150px;
            color: rgb(0, 255, 200);
            text-align: center;
            border-radius: 1px;
            padding: 2px 0;
            position: absolute;
            z-index: 1;
            bottom: 100%;
        }
        .links{
        position:absolute;
    top:50px;
    right: 10px;
    }
    .btn-secondary{
    background-color: rgb(39, 194, 52);
    color: white;
    border-radius: 15px;
    padding: 1em 1.5em;
    text-decoration: none;
    text-transform: uppercase;
    }
.update{
    padding-top: 50px;
}
p{
    font-family: Helvetica;
    font-size: small;
    text-align: center;
    }
    </style>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/cs
s/bootstrap.min.css">
</head>

<body>
    <header>
        <div class="update">
            <a href="/dashboard" class="btn btn-secondary">Dashboard</a>
```

```
      </div>
      <div class="links">

        <a href="/help" class="btn btn-secondary">Help</a>
        <a href="/users/logout" class="btn btn-secondary">Logout</a>

    </div>

</header>


<!-- DIV pour la carte -->
<div id="map" style="width:100%; margin-top: 200px; height:400px"></div>

<script type="text/javascript">
    // Layers
    var layers = [
        new ol.layer.Tile({
            source: new ol.source.OSM()
        })
    ]
    var myExtent = [34.961673,-15.883286 ,34.982369,-15.867291 ]

    // The map
    var map = new ol.Map({
        target: 'map',
        view: new ol.View({
            zoom: 9,
            center: [34.12, -13.55],
         extent:myExtent,
            projection: 'EPSG:4326'
        }),
        layers: layers
    });

    map.getView().fit(myExtent , map.getSize());
    var url = '/data/trans'
    var vsource = new ol.source.Vector({
        url: url,
        format: new ol.format.GeoJSON()
    })



    var trans = new ol.layer.VectorImage({
```

```
        source: vsource,
        visible: true,
        title: 'transformers',
        style: new ol.style.Style({
            image: new ol.style.Circle({
            fill: new ol.style.Fill({
                color: 'rgb(0, 255, 200)'
                }),
            stroke: new ol.style.Stroke({
                color: 'rgb(5,230,10)',
                width: 1.25
            }),
            radius: 9
            })
        })
    });


    var url2 = '/data/pole'
    var vsource2 = new ol.source.Vector({
        url: url2,
        format: new ol.format.GeoJSON()
    })
    var pole = new ol.layer.VectorImage({
        source: vsource2,
        visible: true,
        title: 'poles'
        ,
        style: new ol.style.Style({
            image: new ol.style.Circle({
            fill: new ol.style.Fill({
                color: 'rgb(100, 30, 230)'
                }),
            stroke: new ol.style.Stroke({
                color: 'rgb(100, 30, 230)',
                width: 1.25
            }),
            radius: 9
            })
        })
    });


    var url3 = '/data/fac'
    var vsource3 = new ol.source.Vector({
```

```
            url: url3,
            format: new ol.format.GeoJSON()
        })
        var fac = new ol.layer.VectorImage({
            source: vsource3,
            visible: true,
            title: 'facilities',
            style: new ol.style.Style({
                image: new ol.style.Circle({
                fill: new ol.style.Fill({
                    color: 'rgb(255, 238, 0)'
                    }),
                stroke: new ol.style.Stroke({
                    color: 'rgb(0, 0, 255)',
                    width: 1.25
                }),
                radius: 14
                })
            })
        });



        var url4 = '/data/faults'
        var vsource4 = new ol.source.Vector({
            url: url4,
            format: new ol.format.GeoJSON()
        })
        var faults = new ol.layer.VectorImage({
            source: vsource4,
            visible: true,
            title: 'faults'
            ,
            style: new ol.style.Style({
                image: new ol.style.Circle({
                fill: new ol.style.Fill({
                    color: 'rgb(206, 37, 37)'
                    }),
                stroke: new ol.style.Stroke({
                    color: 'rgb(206, 37, 37)',
                    width: 1.25
                }),
                radius: 5
                })
            })
```

```
    });









    map.addLayer(fac);
    map.addLayer((faults))
    map.addLayer(trans);
    map.addLayer(pole);
    console.log(trans);
    console.log(fac);
    console.log(pole);
    console.log(faults);

    var legend = new ol.legend.Legend({
        title: 'Legend'
    })
    var legendCtrl = new ol.control.Legend({
        legend: legend,
        collapsed: false
    });
    map.addControl(legendCtrl);
    var select = new ol.interaction.Select({})
    map.addInteraction(select)
    var search = new ol.control.SearchFeature(
    {
        source: vsource4,
        property:'meter_num'
    });
map.addControl(search);
// Select feature when click on the reference index
search.on('select', function(e)
    {   select.getFeatures().clear();
        select.getFeatures().push (e.search);
        var p = e.search.getGeometry().getFirstCoordinate();
        map.getView().animate({ center:p });
    });

    // Select feature when click on the reference index
```

```
        var popup = new ol.Overlay.PopupFeature({
            popupClass: 'default anim',
            select: select,
            canFix: true,
            template: {
                title: function (f) {
                        return f.get('meter_num');
                    },
                attributes: // [ 'region', 'arrond', 'cantons', 'communes', 'pop'
 ]
                {
                    'reported_by':{title:'Reporter'},
                    'fault_desc':{title:'Fault'},
                    'date': {title: 'Date'},
                    'status':{title:'Status'}

                }
        }
        })
        map.addOverlay(popup)
    </script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.j
s"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundl
e.min.js"></script>
</body>

</html>
```

d. layout.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css
/all.css" integrity="sha384-
UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/" crossorigin="an
onymous">
      <!-- Compiled and minified CSS -->
```

```
     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/material
ize/1.0.0/css/materialize.min.css">

      <!-- Compiled and minified JavaScript -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/mate
rialize.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js
"></script>
    <title>Node.js & Passport Login App</title>
  </head>
  <body>
    <div class="container"><%- body %></div>

    <script
      src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
      integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min
.js"
      integrity="sha384-
wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js
"
      integrity="sha384-
B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
      crossorigin="anonymous"
    ></script>
  </body>
</html>
```

e. Login.ejs

```
<style>
body{
```

```
padding: 20px;
align-items: center;
text-align: center;
}
.form-group{



}
.cardbody{
  margin-left: 20%;
  width: fit-content;
  height: fit-content;
}
.content-container{

  margin-top: 40px;
  margin-left: 25px;
  display: inline-block;
  padding-top: 16px;
  padding-left: 10px;
  padding-right: 10px;
  height:20em;
  width:25em;
  display:-webkit-box;
  display:-ms-flexbox;
  display:flex;
  -webkit-box-orient:vertical;
  -webkit-box-direction:normal;
  -ms-flex-direction:column;
  flex-direction:column;
  position:relative;
  -webkit-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
  -o-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
  transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
  border-radius:16px;
  overflow:hidden;
  -webkit-box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
  box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
}
h1{
  padding-top: 50px;
  text-align: center;
  font-family: Helvetica;
  font-size: 40px;
```

```
  color: rgb(0, 0, 0);
}
label{
    text-align: left;
    text-decoration: solid;
    text-shadow: '8px';
    size: 16px;
    font-family: Helvetica;
    margin-right: 14px;
}
button[type=submit]{
    margin-top: 20px;
    display:inline-block;
    padding:0.5em 3em;
    border:0.16em solid #FFFFFF;
    margin:0 0.3em 0.3em 0;
    box-sizing: border-box;
    text-decoration:none;
    text-transform:uppercase;
    font-family:'Roboto',sans-serif;
    font-weight:400;
    color: black;
    text-align:center;
    transition: all 0.15s;
}

</style>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/cs
s/bootstrap.min.css">


<body>
  <div class="row mt-5">
    <div class="col-md-6 m-auto">
      <div class="card card-body cardbody">
        <h1 class="text-center mb-3"><i class="fas fa-sign-in-
alt"></i>  Login</h1>
        <% include ./partials/messages %>
        <div class="content-container custom">
          <form action="/users/login" method="POST">
            <div class="form-group">
              <label for="email">Email</label>
              <input
                type="email"
                id="email"
```

```
                name="email"
                class="form-control"
                placeholder="Enter Email"
              />
            </div>
            <div class="form-group">
              <label for="password">Password</label>
              <input
                type="password"
                id="password"
                name="password"
                class="form-control"
                placeholder="Enter Password"
              />
            </div>
            <button type="submit" class="btn btn-primary btn-
block">Login</button>
          </form>

        </div>

        <p class="lead mt-4">
          Not Registered? <a href="/users/register">Register here</a>
        </p>
      </div>
    </div>
  </div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.j
s"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundl
e.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
</body>
```

f. Register.ejs / registration page

```
<style>
  body{
    padding: 20px;
    align-items: center;
    text-align: center;
  }
    .form-group{



  }

  .cardbody{
  margin-left: 20%;
  width: fit-content;
  height: fit-content;
}
    .content-container{
      margin-top: 50px;
      margin-left: 20%;
      display: inline-block;
      height:30em;
      width:24em;
      padding-left: 6x;
      padding-top: 10px;
      padding-right: 40px;
      display:-webkit-box;
      display:-ms-flexbox;
      display:flex;
      -webkit-box-orient:vertical;
      -webkit-box-direction:normal;
      -ms-flex-direction:column;
      flex-direction:column;
      position:relative;
      -webkit-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
      -o-transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
      transition:all 0.4s cubic-bezier(0.645, 0.045, 0.355, 1);
      border-radius:16px;
      overflow:hidden;
      -webkit-box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
      box-shadow:  15px 15px 27px #e1e1e3, -15px -15px 27px #ffffff;
    }
    h1{
      padding-top: 70px;
      text-align: center;
```

```
      font-family: Helvetica;
      font-size: 56px;
      color: rgb(0, 0, 0);
  }
    label{
        text-align: left;
        text-decoration: solid;
        text-shadow: '8px';
        size: 16px;
        font-family: Helvetica;
        margin-right: 14px;
    }
    button[type=submit]{
        margin-top: 20px;
        display:inline-block;
        padding:0.5em 3em;
        border:0.16em solid #FFFFFF;
        margin:0 0.3em 0.3em 0;
        box-sizing: border-box;
        text-decoration:none;
        text-transform:uppercase;
        font-family:'Roboto',sans-serif;
        font-weight:400;
        color: black;
        text-align:center;
        transition: all 0.15s;
    }

</style>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/cs
s/bootstrap.min.css">

<body>
  <div class="row mt-5">
    <div class="col-md-6 m-auto">
      <div class="card card-body">
        <h1 class="text-center mb-3">
          <i class="fas fa-user-plus"></i> Register
        </h1>
        <% include ./partials/messages %>
        <div class="content-container">
          <form action="/users/register" method="POST">
            <div class="form-group">
              <label for="name">Name</label>
              <input
```

```
            type="name"
            id="name"
            name="name"
            class="form-control"
            placeholder="Enter Name"
            value="<%= typeof name != 'undefined' ? name : '' %>"
          />
        </div>
        <div class="form-group">
          <label for="email">Email</label>
          <input
            type="email"
            id="email"
            name="email"
            class="form-control"
            placeholder="Enter Email"
            value="<%= typeof email != 'undefined' ? email : '' %>"
          />
        </div>

        <div class="form-group">
          <label for="password">Password</label>
          <input
            type="password"
            id="password"
            name="password"
            class="form-control"
            placeholder="Create Password"
            value="<%= typeof password != 'undefined' ? password : '' %>"
          />
        </div>
        <div class="form-group">
          <label for="password2">Confirm Password</label>
          <input
            type="password"
            id="password2"
            name="password2"
            class="form-control"
            placeholder="Confirm Password"
            value="<%= typeof password2 != 'undefined' ? password2 : '' %>"
          />
        </div>
        <button type="submit" class="btn btn-primary btn-block">
          Register
        </button>
```

```
          </form>

        </div>

        <p class="lead mt-
4">Have An Account? <a href="/users/login">Login</a></p>
      </div>
    </div>
  </div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.j
s"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundl
e.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
</body>
```

g. welcome.ejs / welcome page

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <title>GIS Electricity</title>

  <!-- CSS  -->
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="style
sheet">
  <link href="css/materialize.css" type="text/css" rel="stylesheet" media="screen
,projection"/>
  <link href="css/style.css" type="text/css" rel="stylesheet" media="screen,proje
ction"/>
</head>
<body>
  <nav class="indigo" role="navigation">
    <div class="nav-wrapper container">
```

```
      <a id="logo-container" href="#" class="brand-logo white-text" style="font-
size: 18px;">GIS-BASED ELECTRICITY ADMIN</a>
      <ul class="right hide-on-med-and-down">
        <li><a class="white-text" href="#">Home</a></li>


        <li><a class="white-text" href="/users/login">Log in</a></li>
        <li><a class="white-text" href="/users/register">Sign Up</a></li>
      </ul>

      <ul id="nav-mobile" class="sidenav">
        <li><a href="#">Navbar Link</a></li>
      </ul>
      <a href="#" data-target="nav-mobile" class="sidenav-
trigger"><i class="material-icons">menu</i></a>
    </div>
  </nav>

  <div id="index-banner" class="parallax-container">
    <div class="section no-pad-bot">
      <div class="container">
        <br><br>
        <h1 class="header center white-text text-darken-
2">Check Reported Faults</h1>
        <div class="row center">
          <h5 class="header col s12 light">Are you an Admin?</h5>
        </div>
        <div class="row center">
          <a href="/users/login" id="download-button" class="btn-large waves-
effect waves-light indigo darken-1">Check Here</a>
        </div>
        <br><br>

      </div>
    </div>
    <div class="parallax"><img src="image.jpg" alt="Unsplashed background img 1">
</div>
  </div>


  <div class="container">
    <div class="section">

      <!--   Icon Section   -->
      <div class="row">
```

```
        <div class="col s12 m4">
          <div class="icon-block">
            <h2 class="center brown-text"><i class="material-
icons">flash_on</i></h2>
            <h5 class="center">RESPOND TO FAULTS</h5>

            <p class="light">Make sure that all the all the Reported Faults are l
ooked into.</p>
          </div>
        </div>

        <div class="col s12 m4">
          <div class="icon-block">
            <h2 class="center brown-text"><i class="material-
icons">settings</i></h2>
            <h5 class="center">LOCATE REPORTED FAULTS</h5>

            <p class="light">Make sure that all the Reported Faults are located o
n map</p>
          </div>
        </div>

        <div class="col s12 m4">
          <div class="icon-block">
            <h2 class="center brown-text"><i class="material-
icons">group</i></h2>
            <h5 class="center">SEND A TEAM OF ELECTRICIANS</h5>

            <p class="light"> Make sure you send a team of Electricitians to go a
nd fix the Reported Faults based on their location.</p>
          </div>
        </div>
      </div>

    </div>
  </div>


  <div class="parallax-container valign-wrapper">
    <div class="section no-pad-bot">
      <div class="container">
        <div class="row center">
          <h5 class="header col s12 light">A modern responsive front-
end framework based on Material Design</h5>
        </div>
```

```
      </div>
    </div>
    <div class="parallax"><img src="background2.jpg" alt="Unsplashed background i
mg 2"></div>
  </div>

  <div class="container">
    <div class="section">

      <div class="row">
        <div class="col s12 center">
          <h3><i class="mdi-content-send brown-text"></i></h3>
          <h4>Contact Us</h4>
          <p class="left-
align light">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam scel
erisque id nunc nec volutpat. Etiam pellentesque tristique arcu, non consequat ma
gna fermentum ac. Cras ut ultricies eros. Maecenas eros justo, ullamcorper a sapi
en id, viverra ultrices eros. Morbi sem neque, posuere et pretium eget, bibendum
sollicitudin lacus. Aliquam eleifend sollicitudin diam, eu mattis nisl maximus se
d. Nulla imperdiet semper molestie. Morbi massa odio, condimentum sed ipsum ac, g
ravida ultrices erat. Nullam eget dignissim mauris, non tristique erat. Vestibulu
m ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;</p
>
        </div>
      </div>

    </div>
  </div>


  <div class="parallax-container valign-wrapper">
    <div class="section no-pad-bot">
      <div class="container">
        <div class="row center">
          <h5 class="header col s12 light">A modern responsive front-
end framework based on Material Design</h5>
        </div>
      </div>
    </div>
    <div class="parallax"><img src="background3.jpg" alt="Unsplashed background i
mg 3"></div>
  </div>

  <footer class="page-footer teal">
    <div class="container">
```

```
      <div class="row">
        <div class="col l6 s12">
          <h5 class="white-text">Company Bio</h5>
          <p class="grey-text text-lighten-
4">We are a team of college students working on this project like it's our full t
ime job. Any amount would help support and continue development on this project a
nd is greatly appreciated.</p>


        </div>
        <div class="col l3 s12">
          <h5 class="white-text">Settings</h5>
          <ul>
            <li><a class="white-text" href="#!">Link 1</a></li>
            <li><a class="white-text" href="#!">Link 2</a></li>
            <li><a class="white-text" href="#!">Link 3</a></li>
            <li><a class="white-text" href="#!">Link 4</a></li>
          </ul>
        </div>
        <div class="col l3 s12">
          <h5 class="white-text">Connect</h5>
          <ul>
            <li><a class="white-text" href="#!">Link 1</a></li>
            <li><a class="white-text" href="#!">Link 2</a></li>
            <li><a class="white-text" href="#!">Link 3</a></li>
            <li><a class="white-text" href="#!">Link 4</a></li>
          </ul>
        </div>
      </div>
    </div>
    <div class="footer-copyright">
      <div class="container">
      Made by <a class="brown-text text-lighten-
3" href="http://materializecss.com">Materialize</a>
      </div>
    </div>
  </footer>


  <!--  Scripts-->
  <script src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="js/materialize.js"></script>
  <script src="js/init.js"></script>

  </body>
```

```
</html>
```