KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI

COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

SMART ELECTRICITY METER MONITORING AND PREDICTION: A CASE STUDY IN GHANA

BY:

Afari Prince

Asare Baffour Samuel Nana

Supervisor: Dr. T-S. M. A. Adjaidoo

September, 2022

DECLARATION

We hereby declare that except for specific references which have been properly acknowledged, this work is the result of our own research and it has not been submitted in part or whole for any other degree elsewhere.

Signature ……………………………………. Date ……………………………………

Afari Prince (Candidate)

Signature ……………………………………. Date ……………………………………

Asare Baffour Samuel Nana (Candidate)

Signature ……………………………………. Date ……………………………………

Dr. Mrs. T-S.M.A. Adjaidoo (Supervisor)

ABSTRACT

Electricity meters have gone through a lot of advancements, both technology-wise and in its deployment as well over the past years. Currently in Ghana, there has been the roll out of smart electricity meters and although it is not the only technology of electricity meters being used in the country, we have a great number of Ghanaian electricity consumers who have migrated from the traditional electricity meters to the use of smart meters since the Electricity Company of Ghana deployed them in the country. Smart meter data analytics, which deals with data gathering, transmission, processing, and interpretation that benefits all stakeholders, is one of the important variables that will make the implementation of smart meters worthwhile. Smart meters collect a large amount of data at regular intervals that can be used for data analytics and numerous insights such as electricity consumption forecasting or prediction, adopting Time of Use tariffs, detecting power thefts and other information that will benefit both electricity companies and in addition, the customer or the consumer. In this project, a web application was developed for monitoring and predicting electricity consumption patterns. The system takes as input, the meter number of the consumer or customer, queries the ECG API to provide usage readings or data related to that particular meter for further insights and also uses that data to predict a user's consumption pattern for the following months/weeks.

DEDICATION

We dedicate this project to our project supervisor, Dr. T-S. M. A. Adjaidoo and our mentor, Dr. Justice for their immense care, support and contribution towards this project. May God bless them abundantly.

ACKNOWLEDGEMENT

This work is dedicated to God Almighty for giving us the strength and the courage to go through with this project successfully.
We also want to say a big thank you to our project supervisor, Dr. T-S. M. A. Adjaidoo for giving us the opportunity to pursue this project.
We also want to say a special thank you to our parents, siblings, colleagues and friends for their prayers and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
DA	Data Analytics
ECG	Electricity Company of Ghana
XML	Extensible Mark-up Language
ARIMA	Auto Regressive Integrated Moving Average
RDBMS	Relational Database Management System
OpenTSDB	Open Time Series Database
ToU	Time of Use
API	Application Programming Interface
URL	Uniform Resource Locator

CHAPTER ONE: INTRODUCTION

1.1 BACKGROUND OF THE STUDY

Electricity has become widely recognized as the primary source of energy for most homes and businesses [1]. There is virtually no work that can be done without being connected to a power source. As a result, every town is keenly interested in the efficient execution of its electrical power generation, transmission, and distribution processes. Also, Rapid economic development has resulted in an increase in global electric power demand. Meanwhile, electricity is considered one of the most important drivers of economic progress and is considered indispensable in our everyday lives [2-4]. Now to mitigate this, the smart grid, which essentially is an electricity network/grid that allows for a two-way flow of electricity and data, with smart metering being a common first step has emerged as one of the solutions to this problem of increasing energy demand. As a result, forecasting electricity usage has become critical for not just Ghana but any country or region at all [5-6]. Development of an accurate and reliable forecasting model for electricity consumption, which might give significant information for electricity system operators or service providers in formulating electricity policies and plans, is critical for power system management. Smart electricity meters keep track of energy consumption every hour or less. The values from the meters are sent in real time to a central database, where they can be examined for various decision-making purposes. The availability of this data offers enormous potential for discovering energy usage patterns among various households using real-time data driven processes. It comes as no surprise that as at December, 2020, there was a total roll out of about 3.8 million smart meters by the Government of Ghana and the Electricity Company of Ghana [9]. Smart meters are popular for their ability to provide electricity consumption at smaller intervals, such as every 15 or 30 minutes, as well as bi-directional communication and remote operating capabilities. So, they record energy consumption in regular intervals and the data is transmitted or transferred in real-time to a central database, where it can then be analyzed for different decision making. As stated earlier, not only will a correct study of this data be beneficial to the customers or the consumers, but the utility or service providers as well. The advantages of smart metering installations are considerable for a variety of system stakeholders. The availability of this data offers enormous potential for discovering energy usage patterns among various households using real-time data-driven processes [7-8]. With the help of consumption patterns, smart energy meters play an essential role in reducing overall energy usage. Electric energy use is particularly complex and varied in everyday life. Electricity consumption, for example, varies greatly depending on the season, and consumption on working days and working days fluctuate as well depending on the availability of the user. At the same time, there will be anomalies in the electrical load, such as forgetting to turn off electrical equipment, appliance malfunction, and even electricity theft, among other things, resulting in a significantly higher electrical demand than usual. As a result, it's vital to spot anomalous consumption data. Abnormal detection can improve abnormal electric energy consumption to save energy and money as well especially in these times of increase in electricity tariffs and general standards of living, remind users to check for malfunctioning electrical appliances or change bad electricity usage patterns, reduce users' energy consumption costs, and raise electricity consumption safety awareness. Anomaly detection from the study of electricity consumption pattern also helps to detect probable power thefts so as to take the needed or necessary actions as soon as possible. Although all these are made possible from the use of smart meters and has enormous benefits over that of the traditional meters and post-paid meters some of these advantages have not fully been realized.

1.2 PROBLEM STATEMENT

Despite the numerous benefits of the smart meters that have been put throughout Ghana. The current isolated state or nature of these smart meters causes some inconvenience to customers. Customers do not get the exact consumption pattern information from ECG as should be the case and can therefore not make any informed decision with regards to the purchase and management of their energy consumption. Consumers or users must try as much as possible to purchase energy units without access to the utility or vending authority’s (ECG) service before the vendor stores close down. This usually happens when the ECG servers are down and so customers cannot make any purchase even with ECG's application, called the ECG Power. So, when in a pinch on weekends, holidays, or late at night, these customers must be willing to go large distances in quest of working suppliers or the vendor stores if they are in need of energy units and as well as be prepared to join long queues for purchase. Additionally, it is also unfortunate to note that, due to the same absence of connectivity, the consumer must always return to his premise to load up purchased energy units, even if there are other important activities that require his/her attention.

Now without a proper look and assessment of these inconveniences, one may conclude that these inconveniences are not really enormous and that they only affect the customer who finds himself or herself in such situation. A careful look at this would bring to bear that the utility or service provider, in this case ECG in the long run also gets affected. When a consumer is in critical need of energy units and has no other option except to suffer a power outage for several hours, he or she may resort to drastic means such as meter manipulation or meter tampering. One of the main motivations for meter manipulation has been found as consumer desperation [10]. Customers in desperation still venture into this unlawful activity with the aim of getting it to provide them with power for longer times [11]. They would not keep the "good news" of their hacking strategy to themselves if they were successful, but would gladly share it with others. This technique of energy theft would quickly become public knowledge, putting the utility or service provider (ECG) at danger of losing money and so long as there are desperate users or consumers, this cycle of power theft may not end. Here in Ghana, there have been reports of power thefts on these newly installed smart meters. [12-13].

From the discussion above, it is evident that both the utilities or service providers as well as the customers are affected by the nature and operation of smart meters and the whole purchase of energy units.

Now if customers are provided with a way of knowing their consumption pattern, then they would be informed as to how to manage their power consumption. Also, should they have a forecast or a prediction of their usage for the week or month ahead, then it would inform them as to how much energy units they are to purchase for a particular time so as to mitigate they being in the desperate search or need or them and not getting access due to the unavailability of service from ECG's application or the vendors and thereby curbing the power theft situation that may arise.

In addition, a careful analysis of the consumption pattern would be of use to the service providers in that they would be able to detect any anomaly that may arise and then take the necessary steps.

1.3 RESEARCH OBJECTIVES

1.3.1 GENERAL OBJECTIVES

The general objective of this project is to develop a system for smart meter monitoring and prediction of consumption pattern of electricity.

1.3.2 SPECIFIC OBJECTIVES

a. Provide energy consumption pattern monitoring system.

b. Provide a forecast of energy consumption.

1.4 SIGNIFICANCE OF THE STUDY

It must be stated that without a doubt, the smart energy meter offers significantly more appealing benefits than the traditional or Post-paid energy meter. Utility companies in Ghana, such as the ECG, have made significant reductions in outstanding debt from defaulting consumers as well as total administrative and operational costs since their creation [14,15]. All of the smart meter energy metering system's previously described benefits appear to benefit the utility more than the client. Even though the client benefits from having budget control, the inconvenience caused by the new system greatly surpasses this gain. Traveling back and forth between vendor shops when energy units run out is inconvenient for the client, especially since he is the one paying for the service. This isn't the case for the traditional meters since with that the customer uses power on credit and then makes payment later. It may seem that traditional meters are better than the smart meters looking at their user experience but then the traditional meters cannot guarantee the utility of the needed revenue needed for its daily operation, it is more prudent and better to still keep or maintain the smart meters. Also, as stated earlier, the increase in development of the economy has also led to increased power demand all over the country. Hence, the prediction of electricity consumption pattern and analysis of it has become very urgent and important for the country. The establishment of a proper, accurate and reliable prediction model for electricity consumption, which could provide valuable information for electricity system operators to formulate policies and plan for the proper distribution of electricity.

CHAPTER TWO: LITERATURE REVIEW

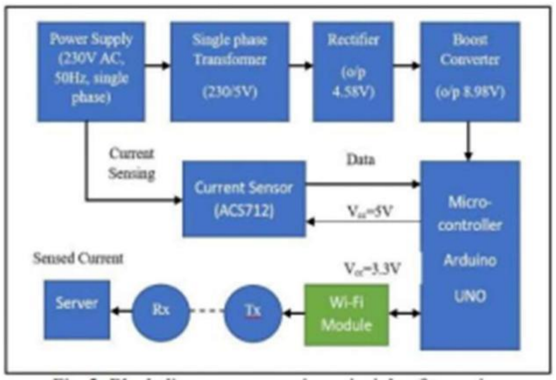
2.0 INTRODUCTION

Recent innovations in energy metering systems and applications have led to the conception and building of numerous kinds of meters working on distinct principles. This portion of the article provides reviews of smart water meters in general. It also outlines ideas and technologies applied in the currently existing smart power meters and their shortcomings, which led to this study effort.

2.1 RELATED WORKS

2.1.1 SMART METER ELECTRICITY MONITORING USING ISOCKET

In this article, a software program and hardware component known as the "i-socket" were developed to monitor the energy consumed by each gadget in a house and to compute the monthly power bill. Now, using this strategy, the user may select a desired monthly cost, and the software program will alert them when their energy use becomes too high. Additionally, this system uses an algorithm to decide the monitoring rate of a connected device to the socket based on the kind of device, making it an energy-saving device in and of itself. The ACS712 sensor was used as the current sensor, and the Arduino UNO microcontroller was used to control the hardware components. The power for the controllers and sensing devices was supplied via a boost converter. The software was created to track electricity expenditures. A block diagram showing how the hardware functions may be found below.


Power Supply (230V AC, 50Hz, single phase)
Single phase Transformer (230/5V)
Rectifier (o/p 4.58V)
Boost Converter (o/p 8.98V)
Current Sensing
Data
Current Sensor (ACS712)
V_{cc}=5V
Micro-controller Arduino UNO
Sensed Current
V_{cc}=3.3V
Server
Rx
Tx
Wi-Fi Module
Figure 2.1: Block diagram illustrating the operation

Below is also a circuit diagram of current sensor with microcontroller unit used in the project.

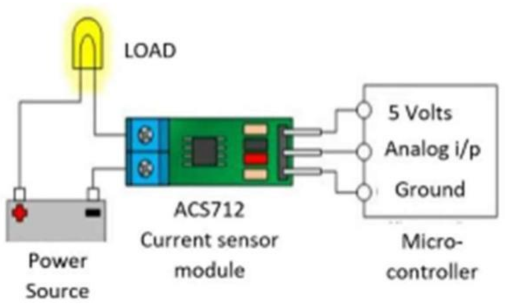


Figure 2.2: Circuit sensor circuit schematic using a microcontroller unit

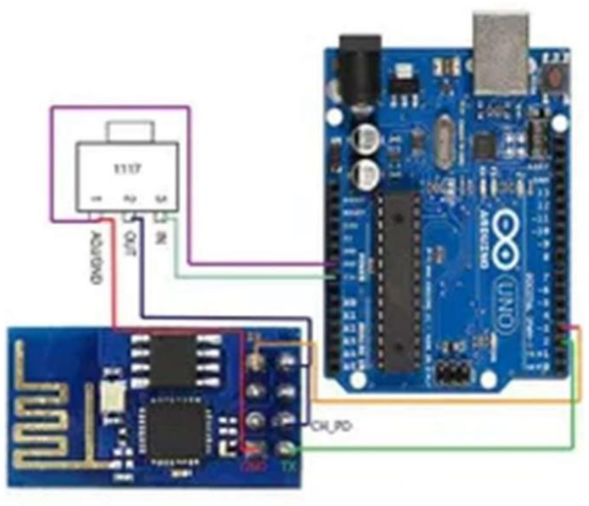


Figure 2.3: Wi-Fi module circuit schematic when linked to an Arduino UNO

THE SOFTWARE WORKFLOW

There are two portions in the software component. One is the program's user interface, while the other is its backend component. The user interface for tracking the trend of electricity bills is an Android app. Django is used to write the backend. The user interface is created using the software package Android Studio and is written in the Java 8 Standard Edition language and XML. Python is used to construct the backend portion of the application. Additionally, Firebase cloud, which serves as a database for storing data, is used to store the data. In order to maintain a single user's session, another application-level database, often SQLite, is used.

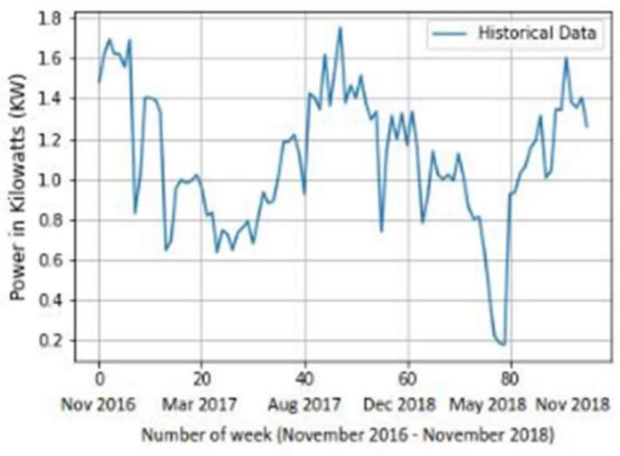


Figure 2.4: Observation of sampled data

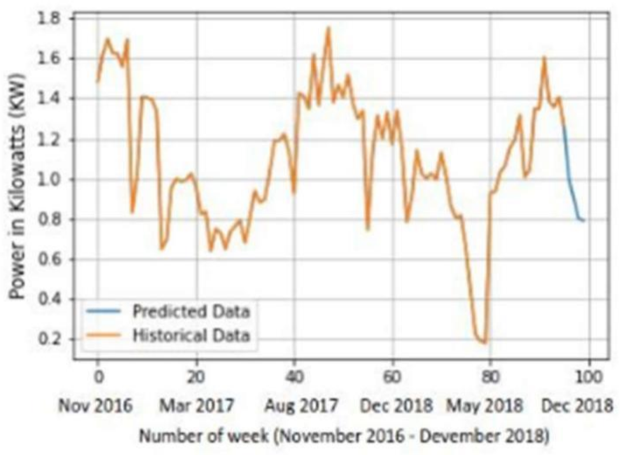


Figure 2.5: Predicted fulfilled using the ARIMA model

2.1.2 SMART METER DATA ANALYTICS USING OPENTSDB AND HADOOP

The goal of this research was to demonstrate how to use open-source technologies like OPENTSDB, HBase, and Hadoop to store time series data, run data analytics on the time series data to get useful insights about power use, and receive the results in a visual format like a graph. Data from smart meters is just a time series that is continuously collected at brief but consistent intervals of time. Smart meters, time series data, open-source tools for handling time series data, and limitations of traditional RDBMS for handling time series data were all covered in this article. Using open-source technologies and a publicly available test data set for smart meters, a prototype implementation of the smart meter data storage and analytics ecosystem was described in this research. [17]

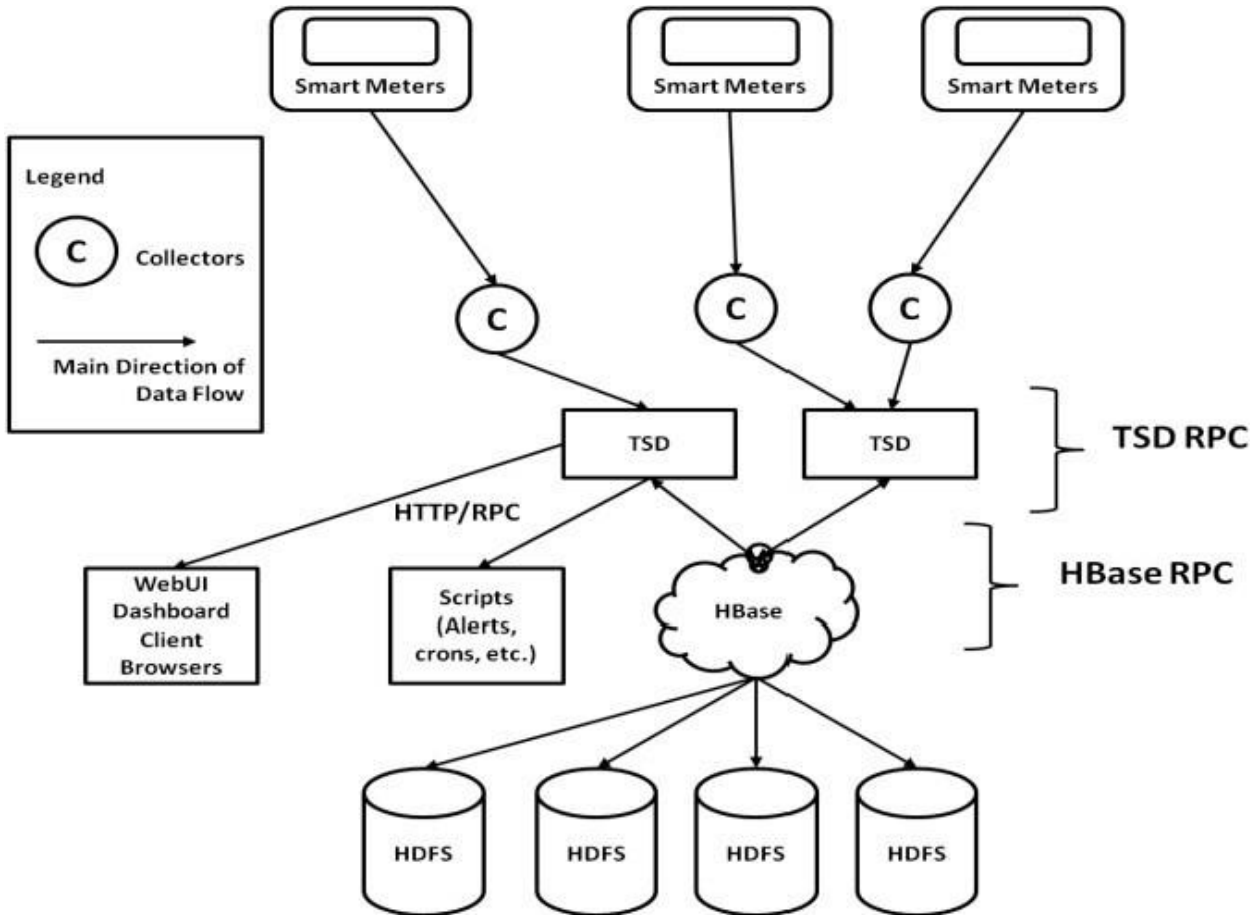


Figure 2.6: Architectural picture of OpenTSDB combined with HBase and Hadoop

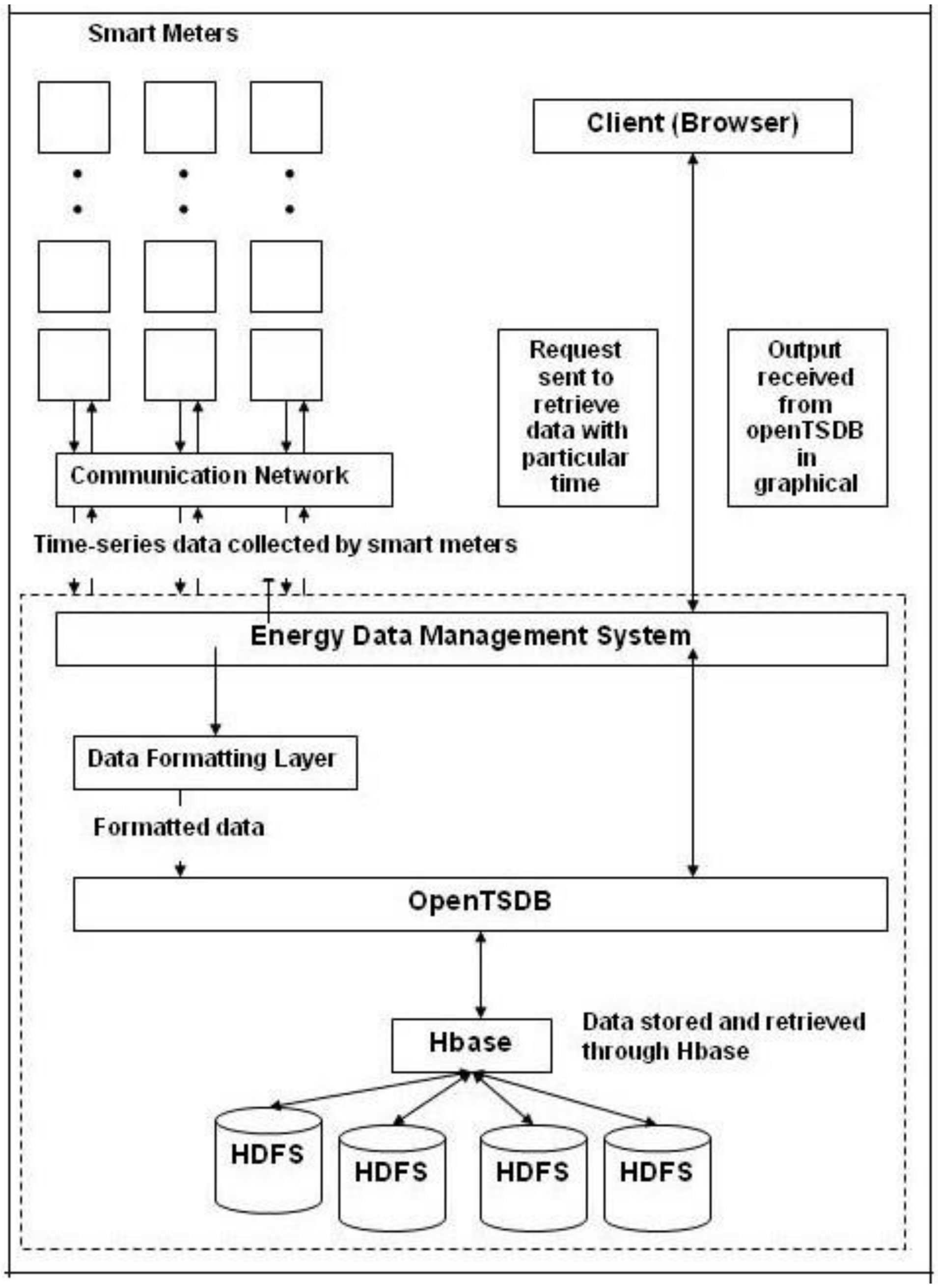
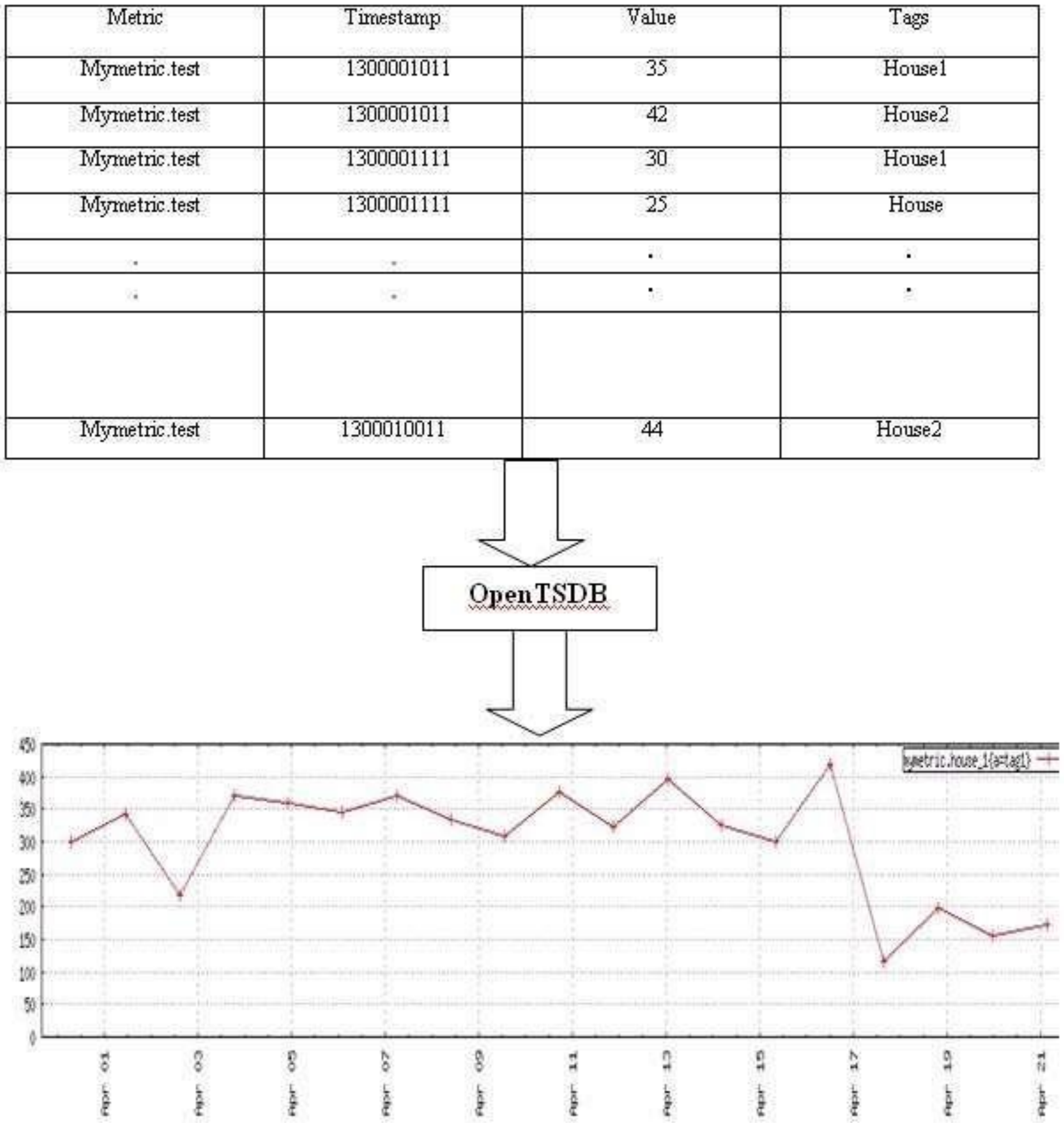
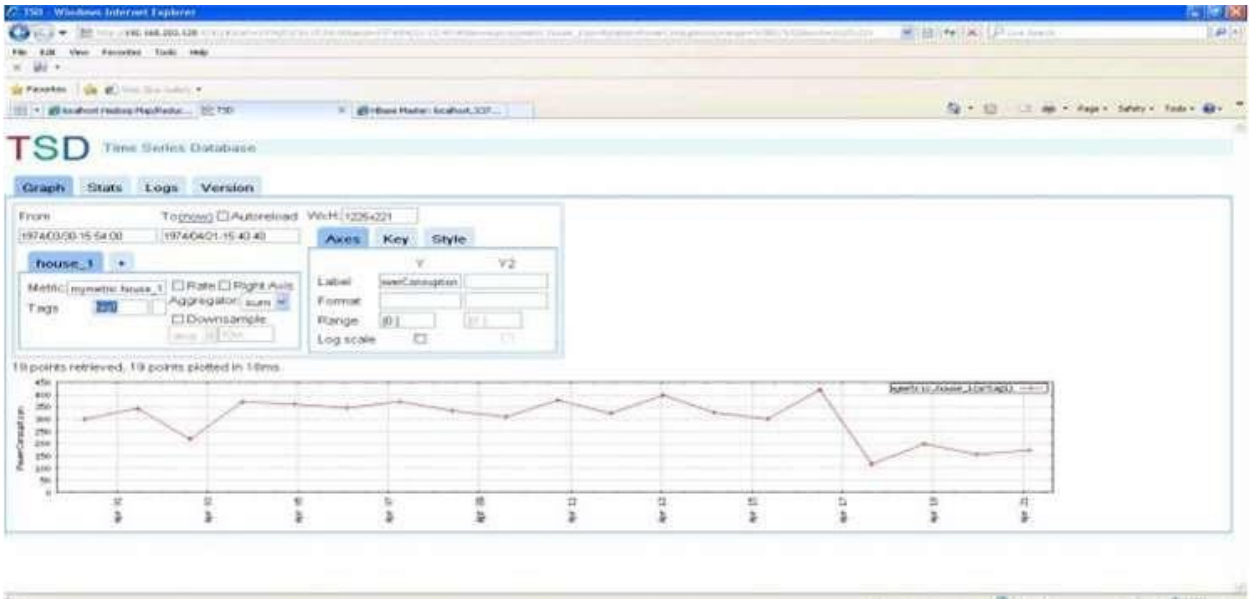

Smart Meters
Client (Browser)
Request sent to retrieve data with particular time
Output received from openTSDB in graphical
Communication Network
Time-series data collected by smart meters
Energy Data Management System
Data Formatting Layer
Formatted data
OpenTSDB
Hbase
Data stored and retrieved through Hbase
HDFS
HDFS
HDFS
HDFS

Figure 2.7: OpenTSDB work flow diagram



Metric	Timestamp	Value	Tags
Mymetric.test	1300001011	35	House1
Mymetric.test	1300001011	42	House2
Mymetric.test	1300001111	30	House1
Mymetric.test	1300001111	25	House
.	.	.	.
.	.	.	.
Mymetric.test	1300010011	44	House2

Figure 2.8: A example of input data and output graph created by OpenTSDB



From 1974/03/30-15:54:00 To(now) ☐ Autoreload WxH: 1226x221 1974/04/01-15:40:40

house_1

Metric: mymetric.house_1 ☐ Rate ☐ Right Axis
Tags
Aggregator: sum
☐ Downsample

	Y	Y2
Label	wetConsuption	
Format		
Range	[0:]	
Log scale	☐	☐

19 points retrieved, 19 points plotted in 18ms

Figure 2.9: A picture of OpenTSDB web UI

2.1.3 SMART ELECTRICITY METER DATA INTELLIGENCE FOR FUTURE ENERGY SYSTEMS: A SURVEY

Since the early 2000s, smart meters have been used in a number of countries throughout the world. Numerous stakeholders are anticipated to benefit economically, socially, and environmentally from the smart meter, a crucial component of the smart grid. The real benefits of smart meters have been the subject of much debate. Smart meter data analytics, which deals with data collection, transmission, processing, and interpretation that benefits all stakeholders, is one of the crucial factors that will determine the success of smart meters. This research provides a thorough review of smart energy meters and how they are used, focusing on key metering process elements, various stakeholder interests, and the technologies used to address stakeholder needs. The report also discusses challenges and opportunities brought on by the rise of big data and the rising use of cloud environments. The availability of time interval data and one of the most helpful analytics tools for the SG have made it possible to anticipate accurately and in the near future. For selecting short-term operations as well as mid-term scheduling, accurate estimates are essential, but decision-makers also need to have an understanding of the customers they must serve for long-term planning. In the literature, several statistical and machine learning techniques have been used in different applications of load forecasting. The use of time series analysis and neural networks for short- and medium-term forecasting has been made. [18]

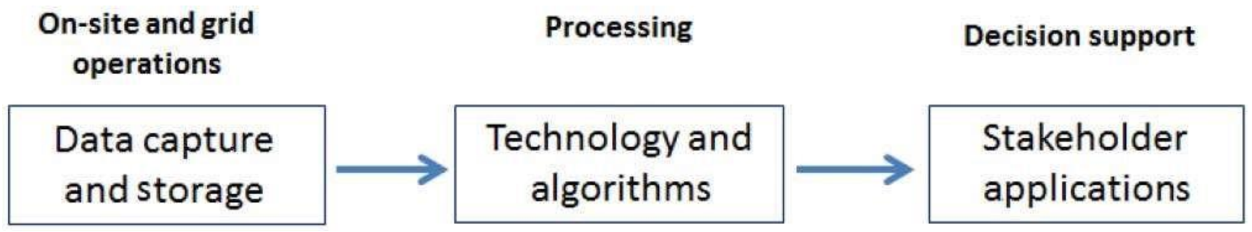


Figure 2.10: Key components of energy meter data intelligence.

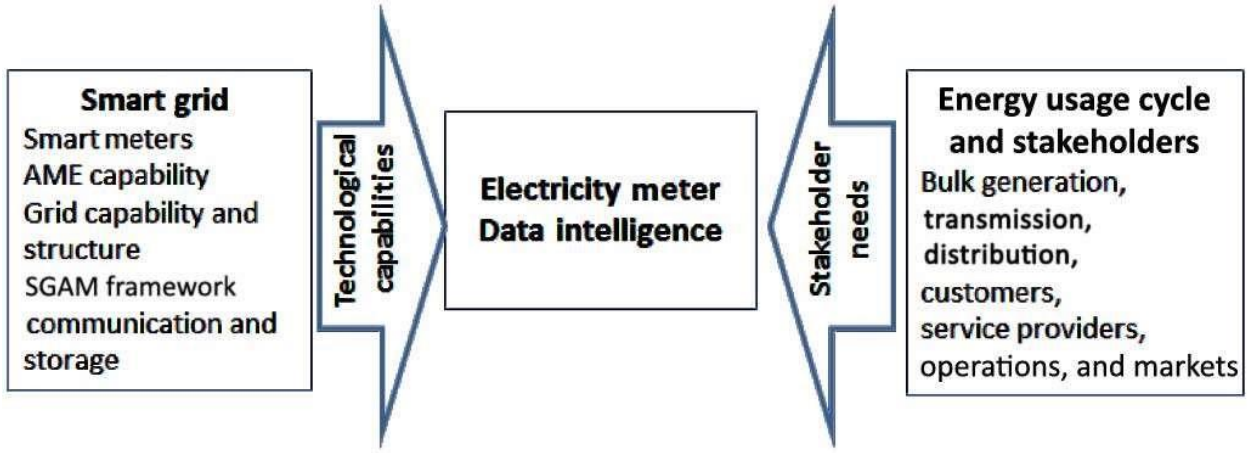

Smart grid
Smart meters
AME capability
Grid capability and structure
SGAM framework
communication and storage
Technological capabilities
Electricity meter Data intelligence
Stakeholder needs
Energy usage cycle and stakeholders
Bulk generation, transmission, distribution, customers, service providers, operations, and markets

Figure 2.11: Environment for intelligent smart meter data.

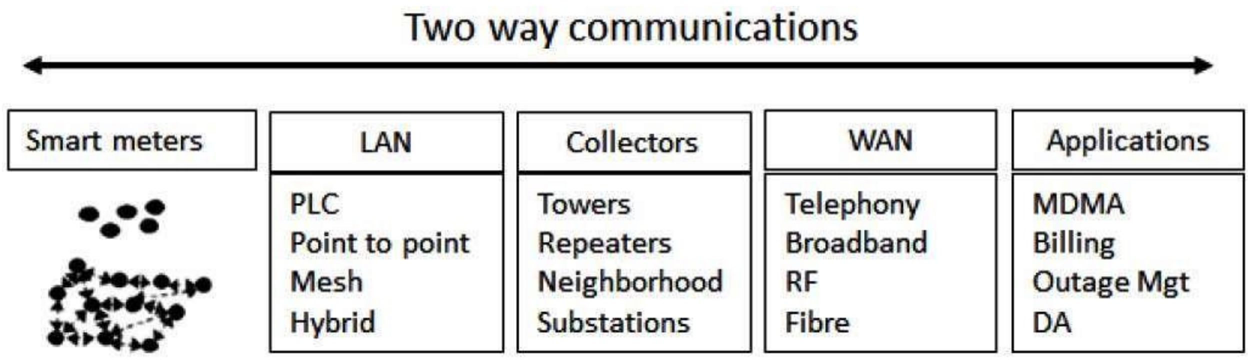


Figure 2.12: Smart-metering process

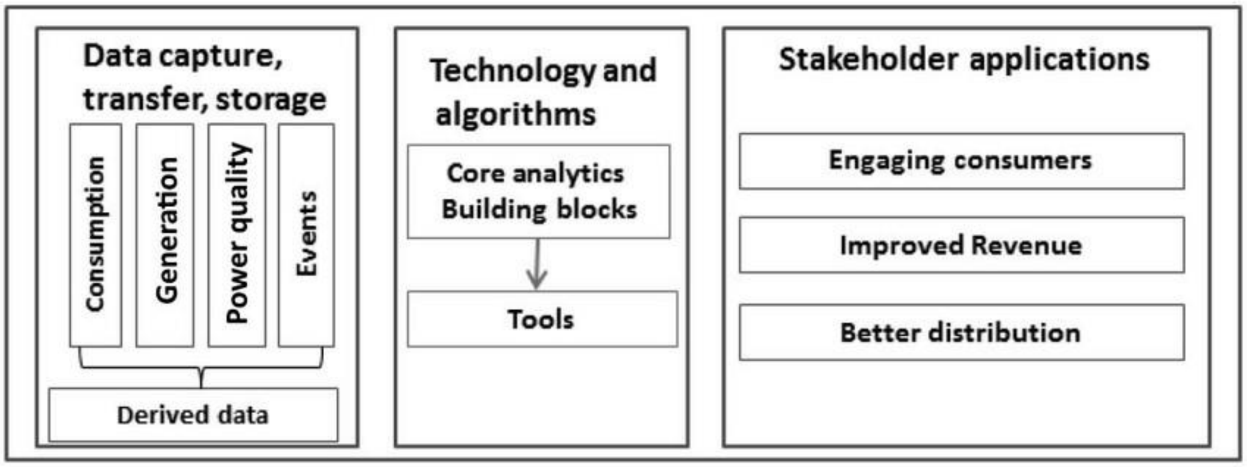


Figure 2.13: Framework for smart meter data intelligence.

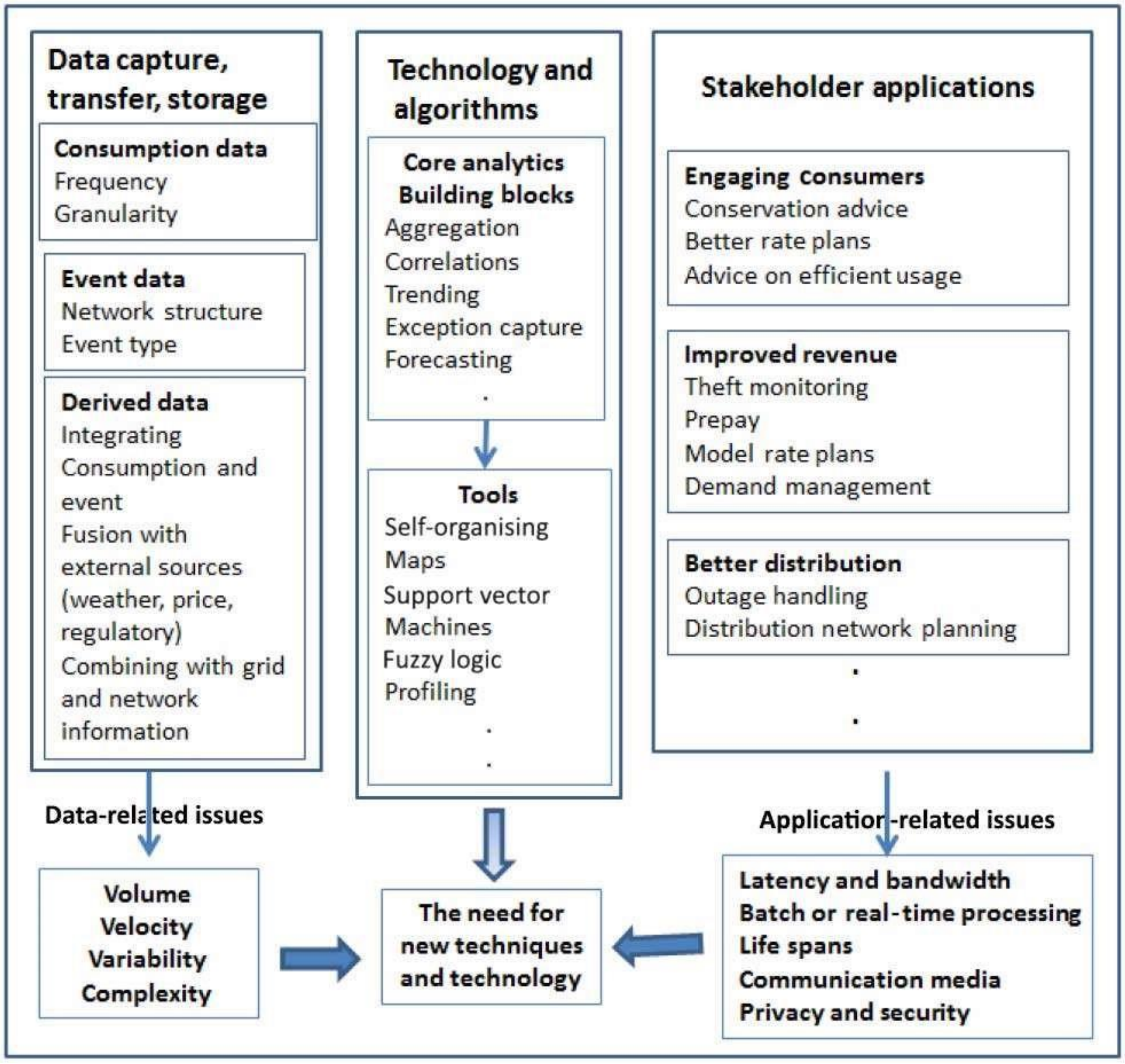


Figure 2.14: Smart-metering framework and new impacts.

2.1.4 WATT'S UP AT HOME? SMART METER DATA ANALYTICS FROM A CONSUMER-CENTRIC PERSPECTIVE

As a result, we look at the range of services in this task that are appropriate for end users' needs. The fundamental advantage of smart meters over traditional metering devices is their ability to communicate consumption data to remote data processing systems. These devices' data collection makes it possible to create a variety of novel use cases in addition to automatically capturing a customer's power use for billing purposes. But the great majority of these services aim to improve the efficiency of the whole electrical grid. Estimates of household energy use or solar production, for example, are helpful for improved power plant producing scheduling. Similar to this, identifying odd patterns of utilization may indicate energy theft and serve as the basis for associated investigations. Although individuals directly manage how much electricity they use, the range of use cases that are advantageous to individual users is still far less than that which is advantageous to the grid as a whole. By briefly discussing the technological foundations of smart meters and their possible impact on future developments, we highlight the enormous potentials of using smart meter data from a user-centric perspective. A number of open research questions in this area that are brought on by the shortcomings of contemporary data processing and transmission methods are also discussed. We believe the findings of their investigation will considerably enhance data processing services and ultimately enhance the use of smart meters. [19]

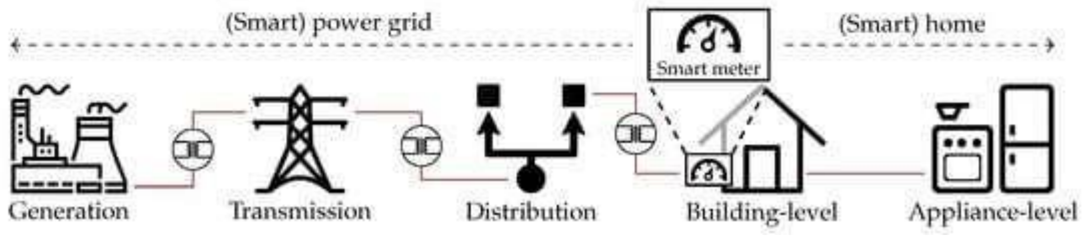


Figure 2.15: Location of smart meters within the electrical power grid (icons by Icon Fonts; CC BY 3.0).

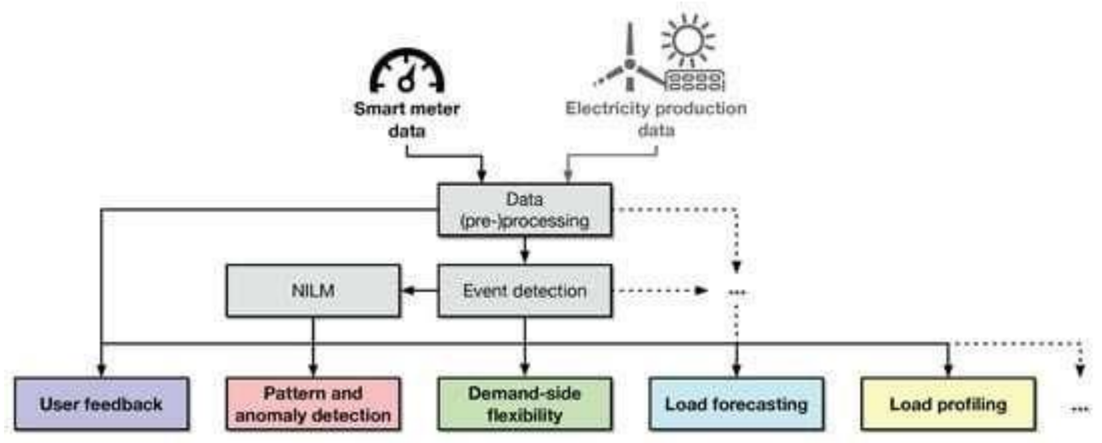


Figure 2.16: Overview of consumer-centric services enabled by smart meter data and their proper data

2.1.5 SMART METER DATA ANALYTICS: SYSTEMS, ALGORITHMS AND BENCHMARKING

This research examined software performance in relation to smart meter analytics. First, a performance benchmark was created that includes common smart meter analytics tasks. These include the architecture we provide for live anomaly detection, model building, and offline feature extraction. As actual smart meter data collection is difficult due to privacy concerns, we also propose a method for creating large realistic data sets from a little seed of real data. Thirdly, five sample platforms were used to construct the proposed benchmark: MATLAB, a relational DBMS with an integrated machine learning toolset; PostgreSQL/MADlib; System C, a main memory column store; and Hive and Spark/Spark Streaming, two distributed data processing platforms. We compare the performance of the five platforms in terms of application development effort on a multi-core machine and a cluster of 16 commodity servers. Special focus was placed on On-Line Anomaly Detection. The last approach, which is also the only online technique in our benchmark, handles anomaly detection. The four algorithms that came before it examine a historical data set to find consumption patterns and build consumption models. On the other hand, anomaly detection examines a day's worth of data at a time. Below, we provide a technique for detecting anomalies in smart meter data and describe how we used it to compare the data to the specified benchmark. We focus on two categories of abnormalities. If the present power usage of a particular home differs significantly from its historical average, we classify this as a self-anomaly. If the family's consumption today significantly deviates from the group average consumption today, we flag it as a group oddity. In this research, a group is defined as the region physically surrounding the given family. Alternative reasonable definitions do exist, however, including clustering the households based on their daily routines, with each cluster being a collection of homes with a similar daily schedule. From the standpoint of a performance benchmark, the precise definition of a group is unimportant since we assume that we have access to a data collection that includes each household's ID and group number. [20]

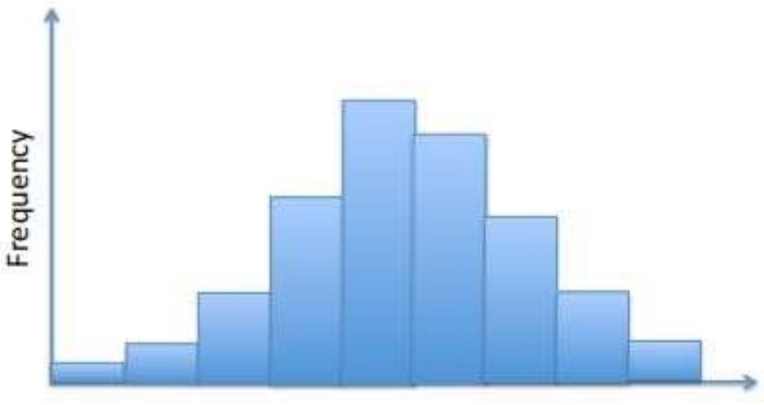


Figure 2.17: Example of a consumption histogram.

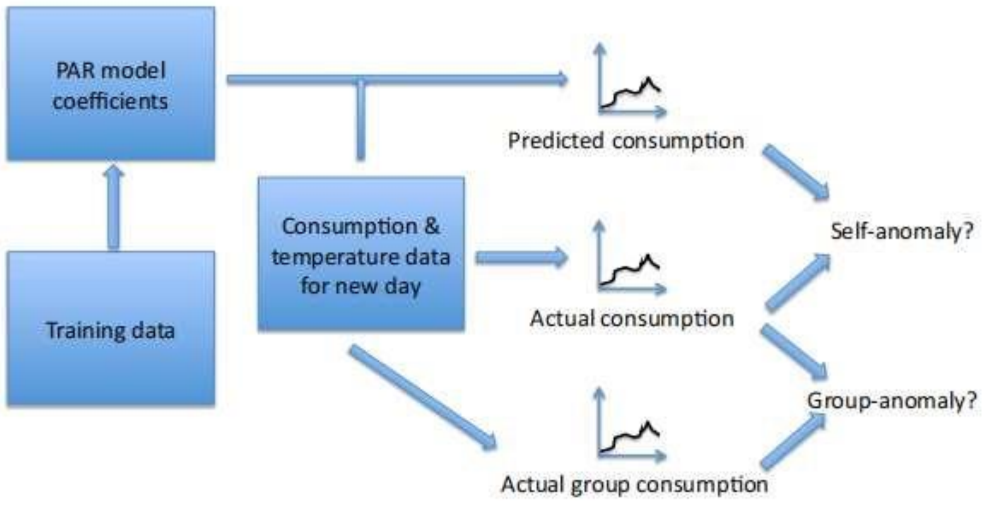


Figure 2.18: Overview of anomaly detection.

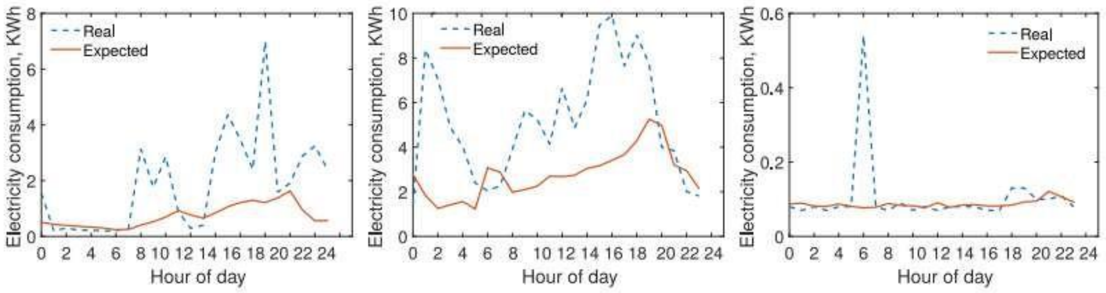


Figure 2.19: Three examples of anomalies detected by the self-anomaly algorithm.

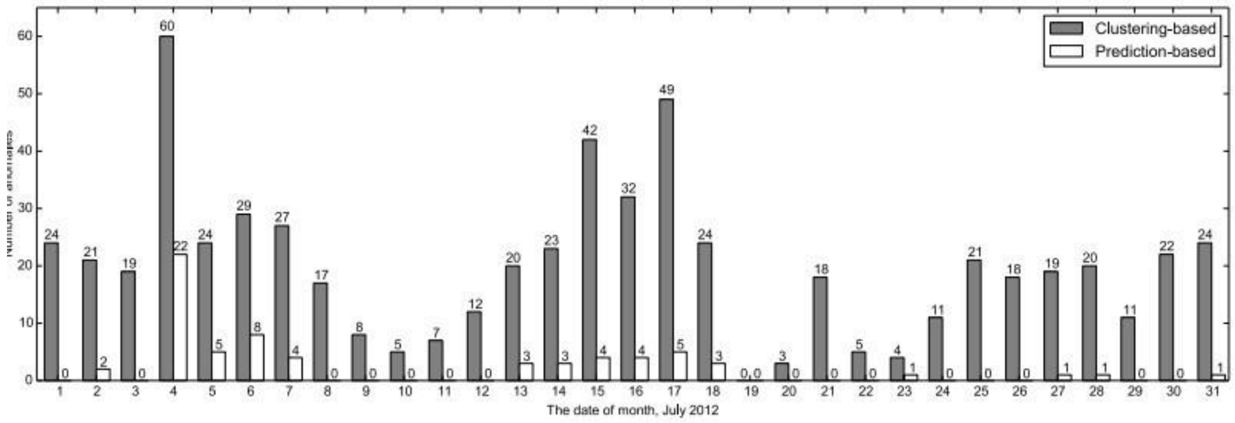


Figure 2.20: Anomalies flagged by our algorithm (Prediction-based) and the Clustering-based algorithm in the month of July

2.1.6 ANALYSIS OF SMART METER DATA FOR ELECTRICITY

This article offers many techniques for analyzing load data, such as analyses of daily load profiles and their similarities, analyses of load density, and analyses of seasonal and irregular elements in the load time series. It attempts to aid customers in comprehending trends in power use. It offers many analytic techniques for home load data, such as the comparison of daily load profiles, analysis of load density, and analysis of seasonal and irregular elements in the load time series. In order to better understand the volatility and unpredictability of the load profiles, this study focuses on descriptive analytics in the context of the smart grid. Systems for descriptive analytics enable the description of certain traits of a home based on its power use. Data from 1000 residential customers' smart meters from the previous year are included in the dataset utilized for this investigation. The data has a 15-min time frame. To enable further analysis, the 15-min energy captured by the smart meters are translated into load. Figure 1 displays 15-minute loads for a sample client (customer X). [21]

CHAPTER THREE: RESEARCH METHODOLOGY

3.1 INTRODUCTION

This chapter details the research methodology for the study. In this section, the method and software development model used to develop the analytics platform and achieve the project objectives is discussed. The method used to develop the software is explained in details together with figures and flow charts.

3.2 DATA ANALYTICS

Analyzing data sets to find patterns and draw conclusions about the information they contain is known as Data analytics. Data analytics is increasingly carried out with the use of specialist hardware and software. Tools and approaches for data analytics are widely used in the commercial sectors to help firms make better business decisions. Analytics technologies are also used by scientists and researchers to support or refute scientific models, theories, and hypotheses. Data mining is a kind of advanced data analytics that involves sifting through massive data sets to find trends, patterns, and connections. Predictive analytics is an additional method that seeks to foresee customer behavior, equipment issues, and other potential business circumstances and events. By employing automated algorithms to process data sets more quickly than data analysts can using conventional analytical modeling, machine learning may also be used for data analytics. Therefore, we would use machine learning for our data analytics. [22]

3.2.1 TOOLS FOR DATA ANALYTICS

Python and PostgreSQL would be used in this project. Below are some reasons why python would be used for data analytics:

1. Python is a well-known multipurpose programming language that is extensively used for its flexibility and large library of useful libraries that are useful for analytics and complicated computations.

2. Because of Python's flexibility, there are dozens of libraries devoted to analytics, such as the widely used Python Data Analysis Library (also known as Pandas).

3. The NumPy library, which offers hundreds of mathematical computations, operations, and functions, is largely where Python's data analytics modules stem from. [23]

4. Matplotlib would then be used to create graphs for the analytics findings to make them easier to analyze.

Also, PostgreSQL would be used for these reasons:

The powerful, open-source PostgreSQL object-relational database system uses the SQL language together with a variety of tools to stably store and expand even the most demanding data requirements. PostgreSQL has a number of features that are intended to help programmers create applications, system administrators protect data integrity and build fault-tolerant systems, and you manage your data no matter the size of the dataset. In addition to being open source and free, PostgreSQL is also incredibly extensible. Without having to rebuild your database, you may add new functions, define your own data types, and even write code in multiple other programming languages![24] Most significantly, it is easy to use.

3.3 SOFTWARE DEVELOPMENT LIFE CYCLE

A software process or software methodology is a set of related activities that leads to the production of a software. The activities may involve the development of the software from the scratch, or, modifying an existing system. Any software methodology must include the following four activities:

- Software specification or requirement engineering: The main functionalities of the software and the constraints around them are defined.
- Software design and implementation: The software is designed and programmed.
- Software verification and validation: The software must conform to its specification and meet the customer’s needs.

- Software evolution or maintenance: The software is modified to meet customer and market requirement changes [25].

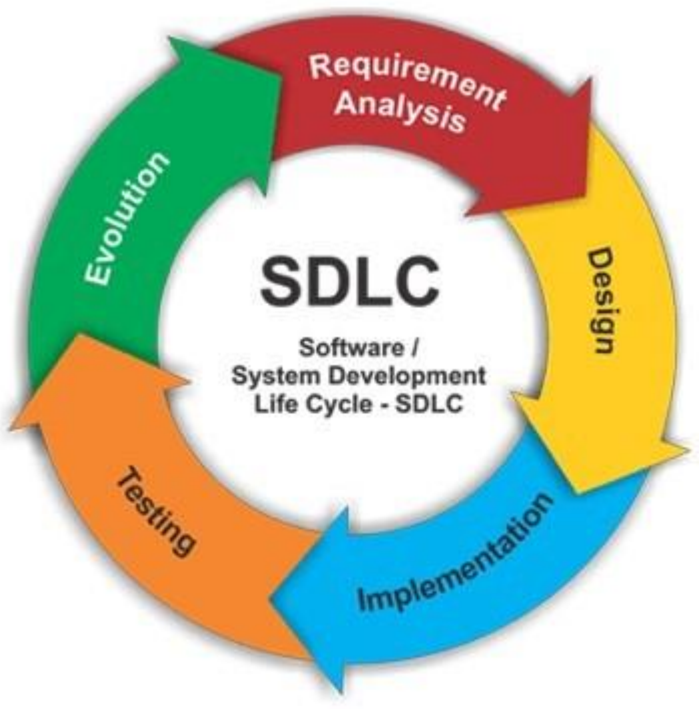


Figure 3.1: Software Development Life Cycle

3.3.1 SOFTWARE DEVELOPMENT MODEL: ITERATIVE AND INCREMENTAL METHOD

A software can be developed using many different software development models. A streamlined illustration of a software process is a software development model. The incremental model is used in this project. The principle behind incremental development is to create a working prototype, test it with users, then refine it across numerous iterations until a workable solution is created. A process's steps are not sequential but rather intertwined, and feedback is integrated into each step. Software that is created progressively is more affordable and simpler to modify while it is being created. The fundamental concept behind this approach is to construct a system via repeated cycles (iterative) and in more manageable chunks at a time (incremental), enabling software engineers to benefit from what was discovered during the creation of previous components or versions of the system.[26]

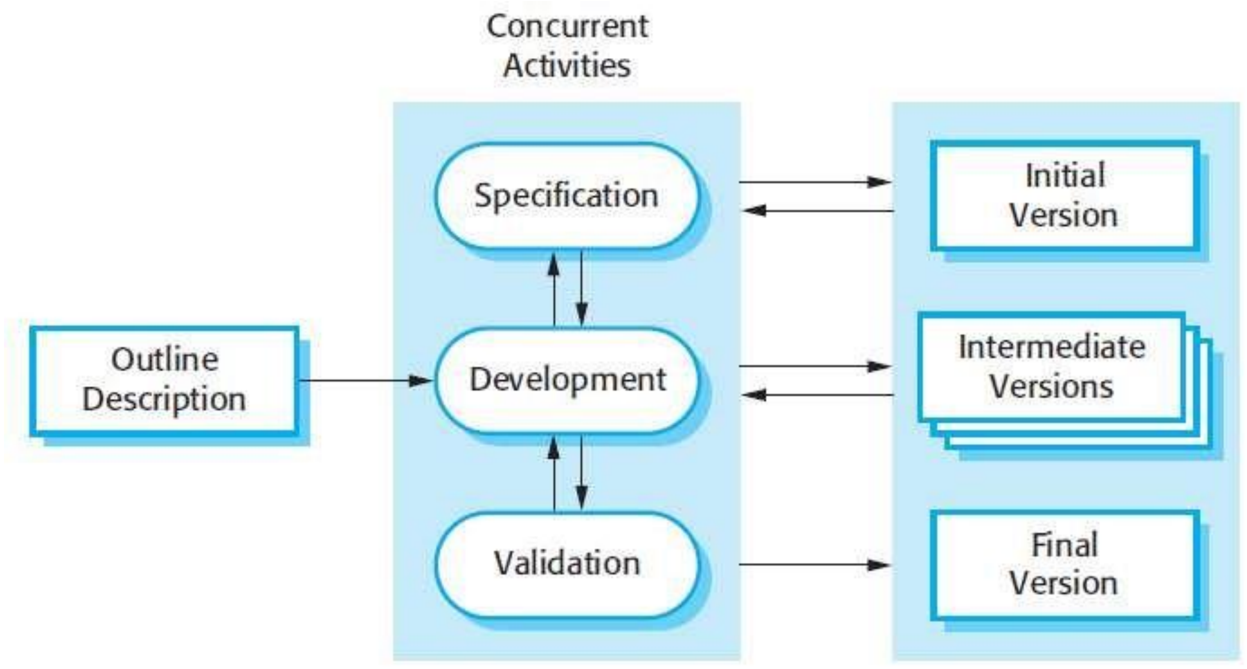


Figure 3.2: Incremental method

3.3.2 TOOLS FOR SOFTWARE DEVELOPMENT

1. HTML AND CSS

   HTML and CSS are the languages to be employed in this project for the software development part. These would be used in the front-end development because it is easy to code with. And also, because the project is a prototype.

2. JAVASCRIPT

   JavaScript is a scripting language used primarily by Web browsers to create a dynamic and interactive experience for the user. Most of the functions and applications that make the Internet indispensable to modern life are coded in some form of JavaScript. Thus, JavaScript would be used.

3. DJANGO

A high-level Python web framework called Django allows the quick creation of safe and dependable websites. Django, which was created by seasoned programmers, handles a lot of the pain associated with web development, allowing you to concentrate on developing your app without having to recreate the wheel. It is open source and free, has a strong community, excellent documentation, and a variety of free and paid support options.

Advantages

- Versatile
- Secure
- Maintainable
- Scalable

4. POSTGRESQL

With numerous capabilities that reliably store and grow the most complex data workloads, PostgreSQL is a robust, open-source object-relational database system that utilizes and extends the SQL language. Numerous capabilities in PostgreSQL are designed to support developers in creating applications, administrators in safeguarding data integrity and creating fault-tolerant systems, and you in managing your data regardless of the size of the dataset. PostgreSQL is not only open source and free, but it is also very extendable. You may create new functions, specify your own data types, and even write code in several programming languages without having to recompile your database, for instance! [24] Above all else, it is simple to use.

3.4. DESIGN METHODOLGY

To have a clearer definition of the system being built/implemented for the user and its functionalities, the following use cases are employed in the elicitation of these requirements or functionalities. This usually aids in presenting in natural language the expectations, goals and benefits of the system being built to the regular customer or consumer. The use cases are presented below:

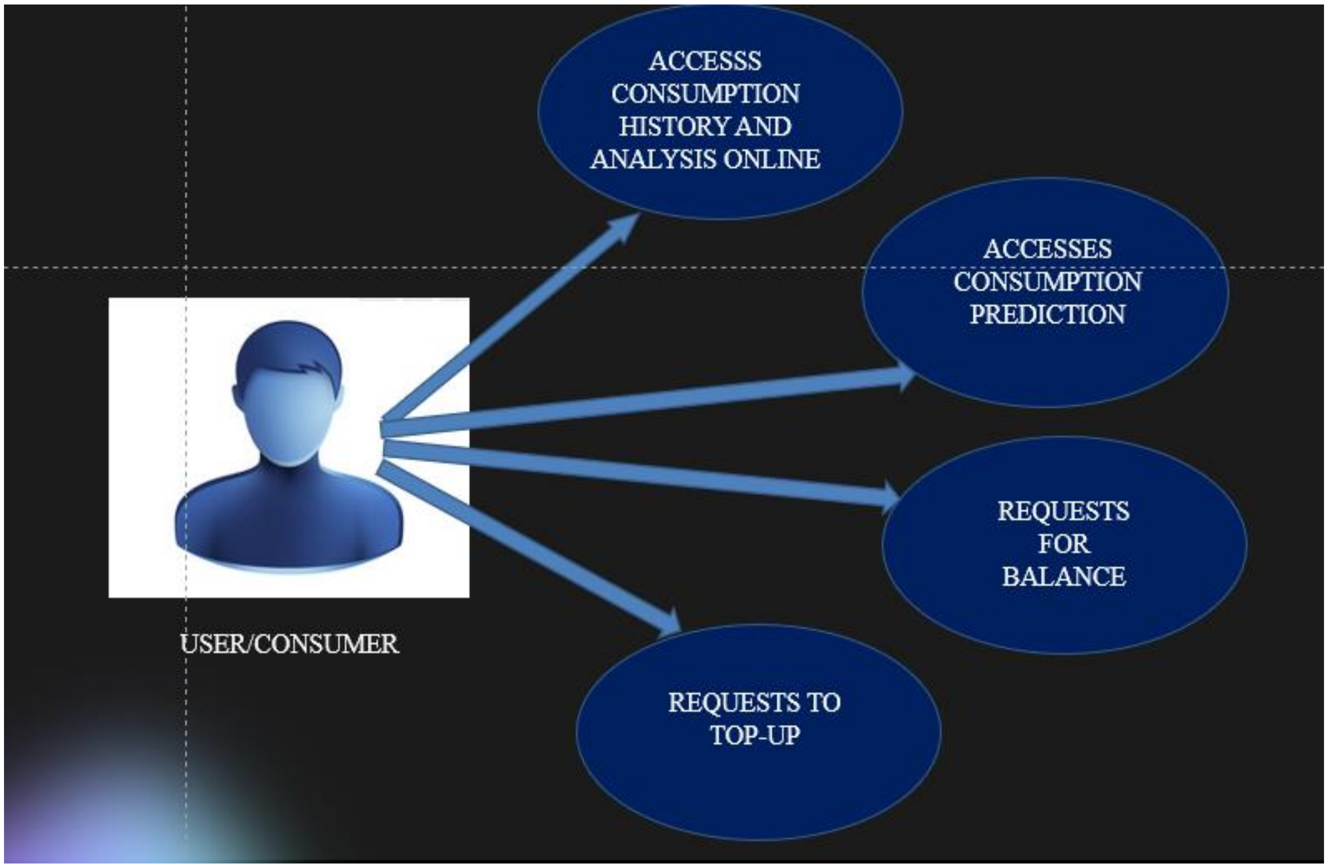

ACCESSS CONSUMPTION HISTORY AND ANALYSIS ONLINE
ACCESSES CONSUMPTION PREDICTION
REQUESTS FOR BALANCE
REQUESTS TO TOP-UP
USER/CONSUMER

Figure 3.3: Use case for system design

The design of any complex system first begins with the design of the individual modules that make up the system and then assembling these modules as one system. It is very necessary to this effect break down the design of the smart meter data analytics system into functional parts to make the design process more systematic and easier. There are three major functional parts identified with the Smart Meter Analytics System which is aimed at providing both parties (Utility Company and Consumers) with consumption data and also provide a prediction of user consumption for the subsequent weeks or months. The main parts of the system are;

- The analytics model
- The prediction model

- The user interface/ visualization model/ web application.

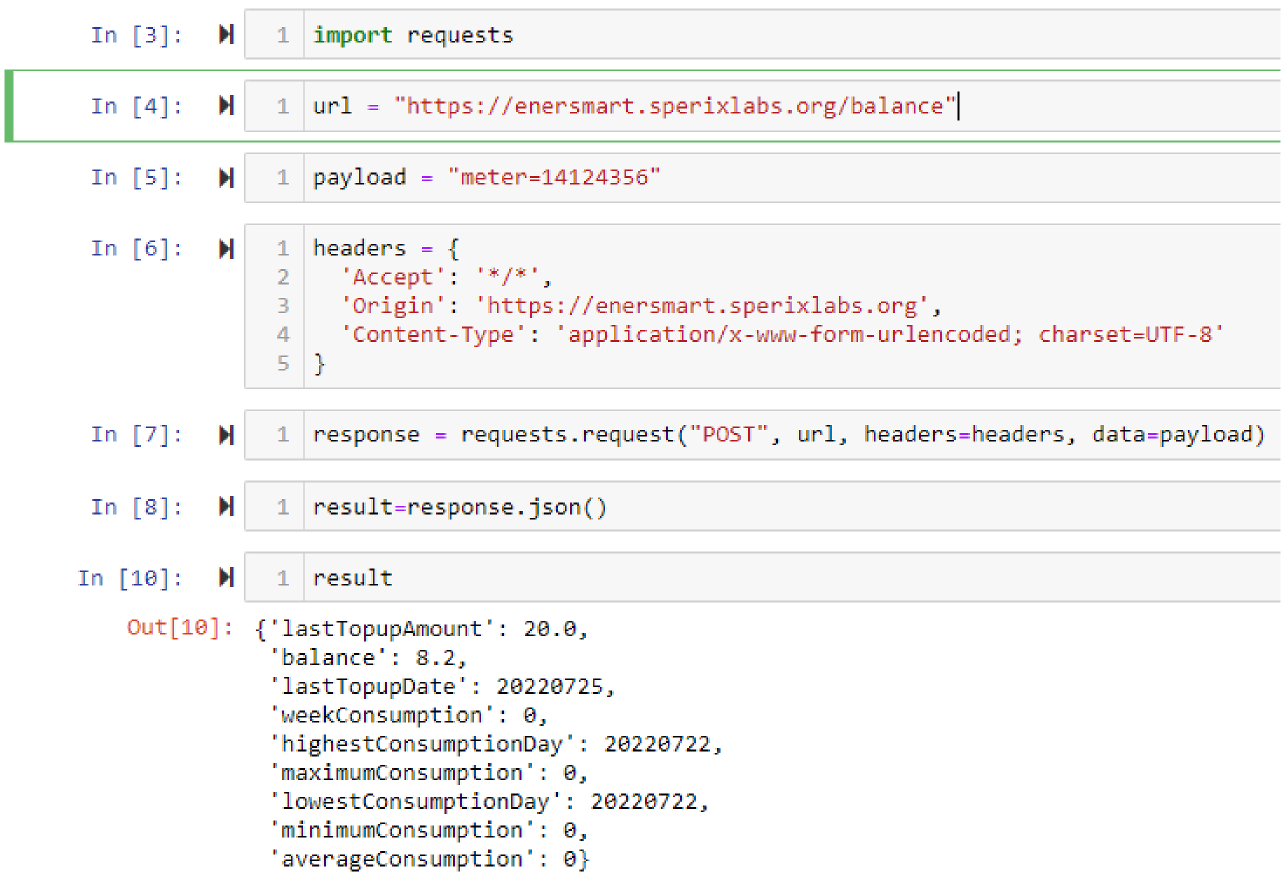
To implement these functional parts or sections of the project, the following processes or steps were used;

a. Data gathering
b. Data cleaning and normalization
c. Storing data
d. Building the analytics model
e. Implementing the forecasting model
f. Providing a visualization for results
g. Providing a user interface for system / integration of analytics with a system

A brief introduction of how each of these guidelines is followed in this research is provided in the following subsections.

3.4.1 DATA GATHERING

Data gathering which may also be referred to as data collection is the act of gathering, gauging, and analyzing precise data from a range of pertinent sources in order to address issues, provide answers, assess results, and predict trends and possibilities. Data collection forms the basis of the entire project since every other implementation is dependent on the availability of data for analysis and processing. The source of customer data is primarily from the utility company (ECG). Through the ECG API, various consumption data of customers (four users) were gathered for analysis. Querying of the API only returns the current balance of the consumer and has no field for the past consumption of the consumer. As a result, consumer data had to be logged for about two months (mainly between the months of June and August).



```
In [3]:  import requests

In [4]:  url = "https://enersmart.sperixlabs.org/balance"

In [5]:  payload = "meter=14124356"

In [6]:  headers = {
           'Accept': '*/*',
           'Origin': 'https://enersmart.sperixlabs.org',
           'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8'
         }

In [7]:  response = requests.request("POST", url, headers=headers, data=payload)

In [8]:  result=response.json()

In [10]: result

Out[10]: {'lastTopupAmount': 20.0,
          'balance': 8.2,
          'lastTopupDate': 20220725,
          'weekConsumption': 0,
          'highestConsumptionDay': 20220722,
          'maximumConsumption': 0,
          'lowestConsumptionDay': 20220722,
          'minimumConsumption': 0,
          'averageConsumption': 0}
```

Figure 3.3: Code snippet for API query

The url for the API is https://enersmart.sperixlabs.org/balance . This returns the following fields; the last top up amount, balance, last top up date, week consumption, highest consumption day, maximum consumption, lowest consumption day, minimum consumption and the average consumption of the user or the customer. Although not all the fields return true values, the essential fields are always present, such as the balance and the timestamp. There are days when the server is offline and hence affects the data collection process, however such cases were addressed in the data cleaning and normalization process.

3.4.2 DATA CLEANING AND NORMALIZATION

A very essential part of data collection and analysis is data cleaning and normalization since it is key to ensuring the quality and integrity of the data collected without necessarily tampering with the data. A database's whole contents are reviewed as part of the data cleansing process, and any information that is missing, inaccurate, duplicated, or irrelevant is either updated or removed. Some anomalies that come with the data collection process and how they were addressed are;

a. Servers being offline: for some days within the month, especially the early/ first days of the month, the data isn't available for collection since the servers are offline. As a result the fields in the database for these days are mostly null. For such cases the average consumption of the user is usually computed and used in place of the null values.
b. Customer TopUp: usually at some points within the week, customers top up to increase their energy units. This often affects the computation for the customer's consumption calculation and will be talked about extensively in the subsequent chapter. The average consumption of the customer prior to that date is a gain computed and used instead.

3.4.3 DATA STORAGE

Data storage is also very essential in the whole process. Form figure 3.3, the query results are stored or returned in a json object. The results are compiled and pushed or imported into the local database created with POSTGRESQL. The advantages or reasons for using POSTGRESQL are spelt out in the earlier part of the chapter. The data is pushed into the database and is then normalized to remove duplicates, redundancies as much as possible and to make it as accurate as possible.

3.4.4 BUILDING THE ANALYTICS MODEL

After the data has been collected, cleaned and rid of all redundancies, then comes the analytics or the section for the analysis. Extracting usable information from data and making decisions based on that analysis are the goals of data analysis. There are a number of data analysis techniques however for the purpose of this study, two of these techniques were looked at and they are as follows;

a. The Statistical Analysis: here, data collection, analysis, interpretation, presentation, and modeling are all included in statistical analysis. A set of data or a sample of data is analyzed. There are two categories under this type of data analysis and they are inferential analysis and descriptive analysis. This project also makes use of the descriptive analysis category. With this, it analyses complete data or a sample of summarized numerical data, showing the mean or average, the deviation as well for the data and it basically answers the question of what can we see from the data?

b. Predictive Analysis: this form of analysis basically answers the question, what will happen as a result of the data? Essentially, it combines the knowledge from all earlier analyses to decide how to respond to a current issue or choice.

To some extent the data gathering process and cleaning process may all be considered as part of the data analysis process.

3.4.5 IMPLEMENTING THE FORECASTING MODEL

After data has been analyzed and meaningful insights drawn, a forecasting model will be built to predict the user’s consumption pattern for the subsequent weeks or months. A pre-existing forecasting model will be studied and used to implement such a functionality. The Auto Regressive Integrated Moving Average forecasting model will be used in this regard. Although the ARIMA model is often difficult to implement, it however has a number of advantages over the other forecasting models and was preferred for this project.

ARIMA

In an Autoregressive Integrated Moving Average (ARIMA) model the data gathered is difference to make it stationary. A model that demonstrates stationarity demonstrates that the data remain constant across time. The goal of differencing is to eliminate any patterns or seasonal structures that are present because most economic and market data exhibit trends. Seasonality, or when data exhibit recurring, predictable trends over the course of a year, may have a negative impact on the regression model. Many of the calculations throughout the process cannot be performed with great efficiency if a trend develops and stationarity is not obvious. The model combines two predictive models, the Autoregressive (AR) model and the Moving Average (MA) model, making it very efficient for forecasting future consumption patterns or forecasting in general despite being difficult to implement.

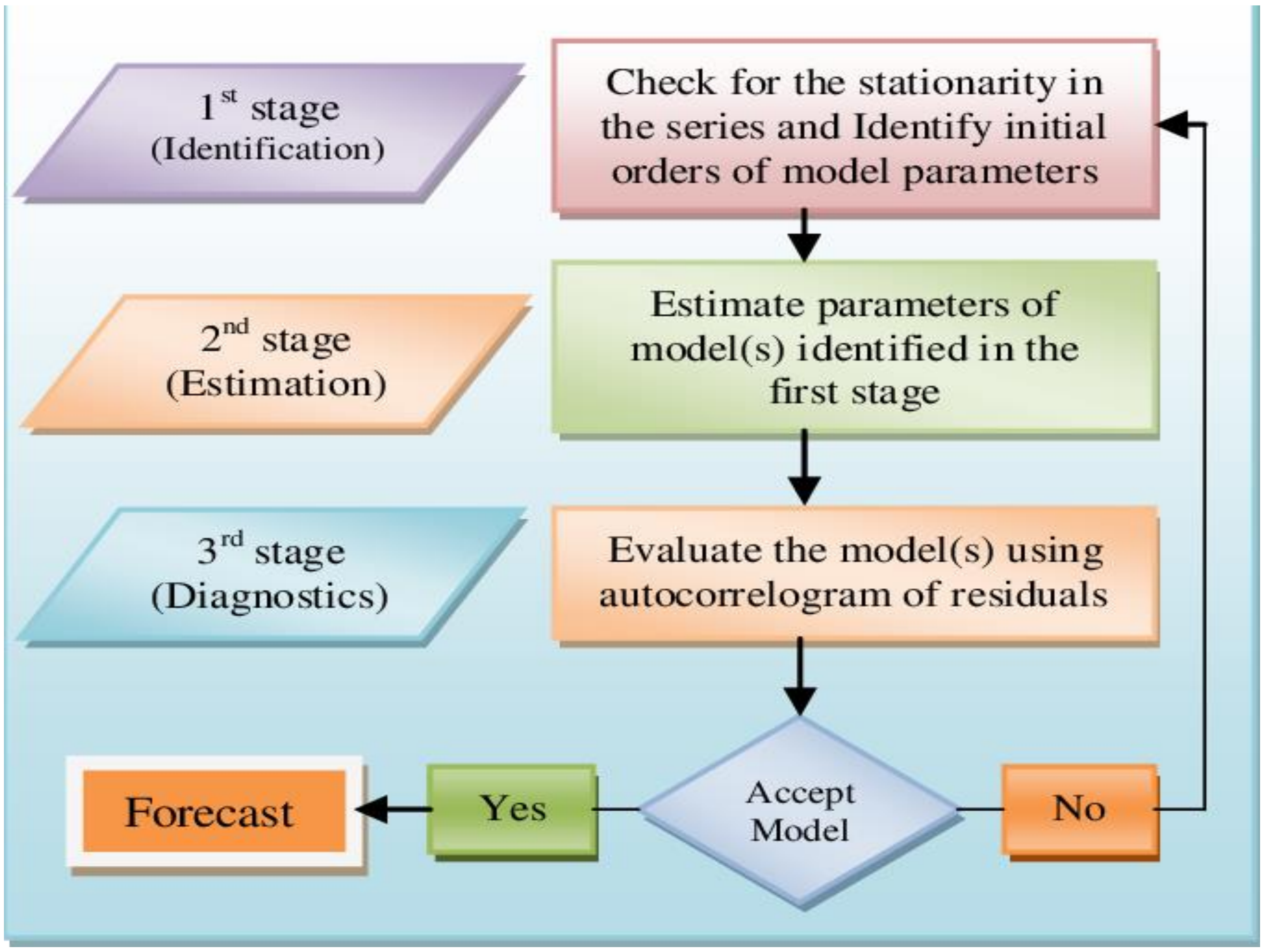

1st stage
(Identification)
Check for the stationarity in the series and Identify initial orders of model parameters
2nd stage
(Estimation)
Estimate parameters of model(s) identified in the first stage
3rd stage
(Diagnostics)
Evaluate the model(s) using autocorrelogram of residuals
Forecast
Yes
Accept Model
No

Figure 3.4: Schematic presentation of methodology for ARIMA forecasting model

REASONS FOR CHOOSING ARIMA

- Only requires the prior data of a time series to generalize the forecast.
- Performs well on short term forecasts.
- Models non-stationary time series.

3.4.6 VISUALIZATION

Another very important part of data analysis is data visualization. The depiction of data through the use of typical graphics, such as infographics, charts, and even animations, is known as data visualization. These informational visual representations make complex data relationships and data-driven insights simple to comprehend [27]. Basically, data visualization helps to better appreciate the results of data that has been analyzed. There are a number of data visualization types and representation types out there. This project makes use of line graphs to depict trends of data(consumption). For tools for data visualization, there are popular ones such as Tableau, Echarts, Vega and Microsoft power BI, among others. However, the python library, matplotlib was employed for this purpose.

MATPLOTLIB

Python's Matplotlib toolkit provides a complete tool for building static, animated, and interactive visualizations. Matplotlib makes difficult things possible and simple things easy in terms of data visualization.

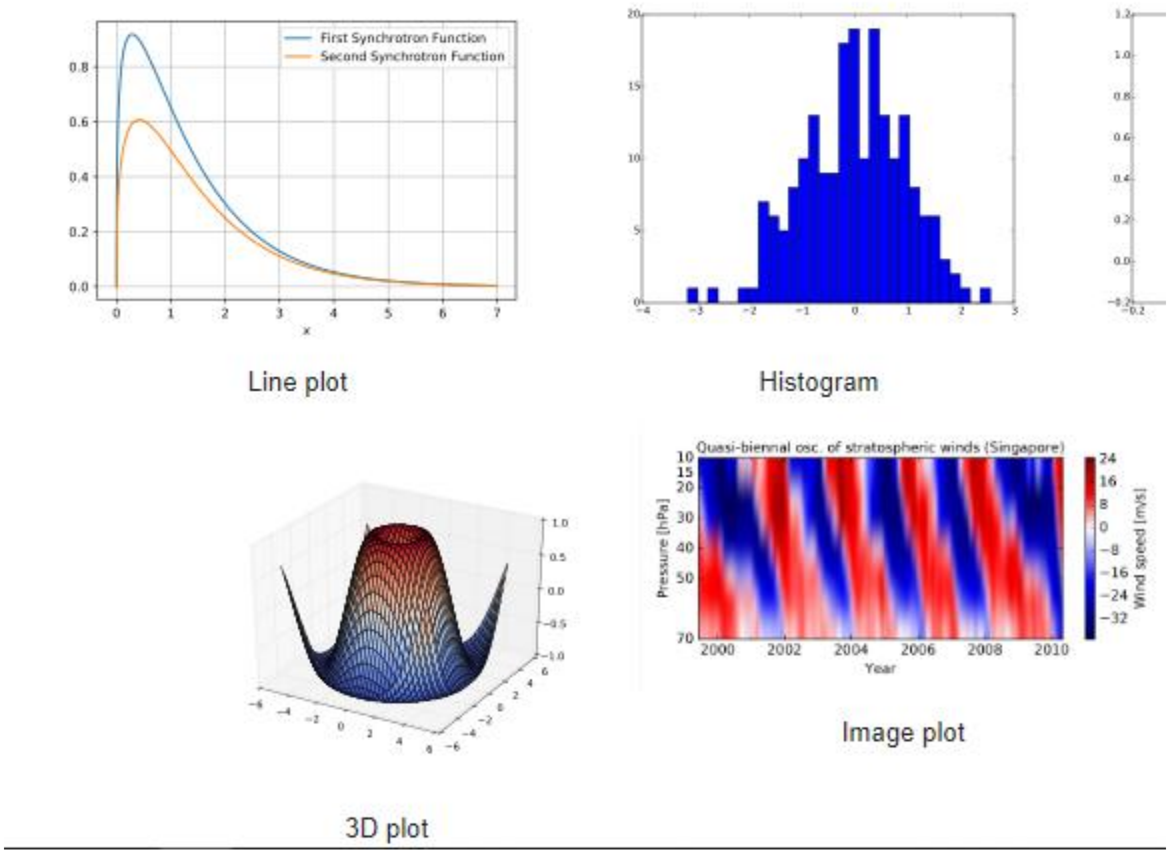


Figure 3.5: Visualization with Matplotlib

It has the following advantages and that's why it was chosen[28];

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

3.4.7 EMBEDDED ANALYTICS

Generally, analytics embedding or embedded analytics is the process of incorporating analytical and data visualization features into a software program. By integrating reports and dashboards, they assist the end user in analyzing their data within the software application. Users can detect and reduce business risks using the information provided from this study, as well as discover new opportunities, which aids in corporate growth. It basically makes it easier for the users or the customers to interact better with the system developed. It also has the advantage of increasing productivity and usability of the system.

4.0 DESIGN IMPLEMENTATION

The entire implementation of the work is categorized into three parts apart from the data collection, cleaning and storage stages; the descriptive analytics, predictive analysis and the embedded analytics. As described in the methodology section, the data was continually logged by making frequent API request to the ECG API for individual consumers' consumption balance and top-up values among other details. The data after being cleaned was stored in the database for further analysis.

4.1 PREDICTIVE ANALYSIS

Future values can be forecasted from past series values with the ARIMA model. Again, we are dealing with time series values in this project. Time series can basically be defined as a sequence where a particular metric is recorded over regular time intervals. The frequency of this time series was that of daily. After the values have been logged, cleaned and stored, the next step for us to know the values the series is going to take is to make the forecast. Also, for time series forecasting, if only the previous values of the time series are used to predict future values, then it is termed as the univariate time series forecasting.

The ARIMA model is a forecasting algorithm based on the idea that the previous values of the series can be used to predict the future values.

Any implementation of the ARIMA model is characterized by 3 terms: p, d, and q.

P is the order of the Autoregressive (AR) term, which is basically the number of immediately preceding values in the series that are used to predict the values at the present time [29], the q is the order of the Moving Average term, also depicting the number of the number of lagged forecast errors that should go into the ARIMA model [30] and finally the d value is the number of differencing that is needed in order to make the time series stationary.

Now, we look at the flowchart to describe the workflow of the implementation of the predictive model;

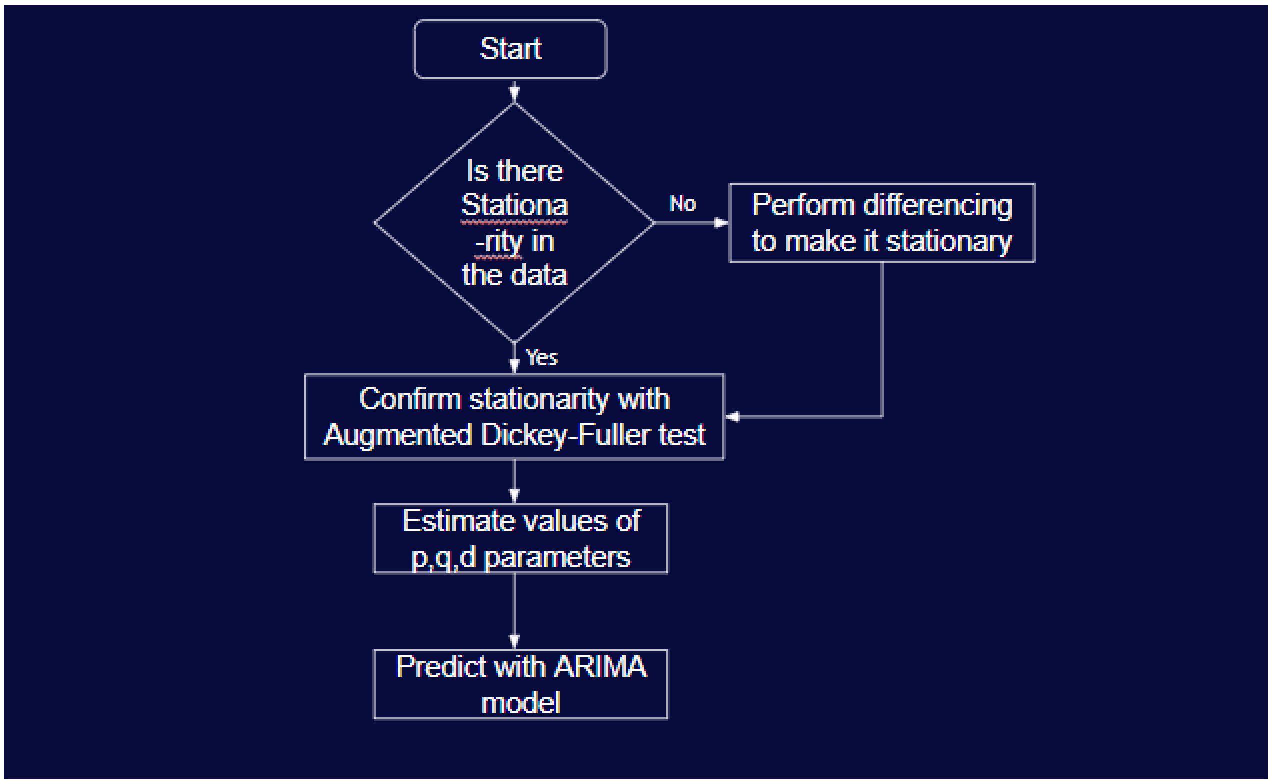
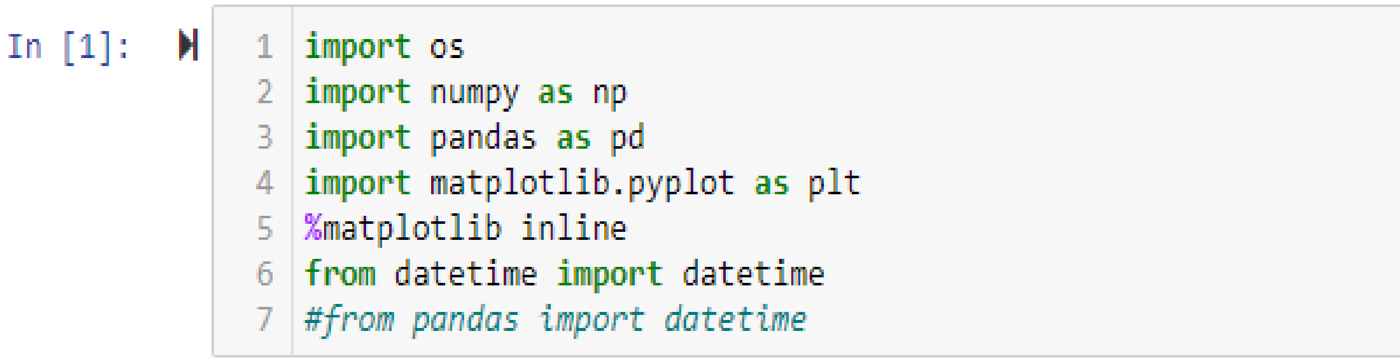

Start
Is there Stationa -rity in the data
No
Perform differencing to make it stationary
Yes
Confirm stationarity with Augmented Dickey-Fuller test
Estimate values of p,q,d parameters
Predict with ARIMA model

Figure 4.1: flowchart for predictive analysis

4.2 STATIONARITY

A stationary time series data is one whose properties do not depend on time, thus one which does not have any trends, seasonality or anything of the sort as the trend and seasonality will affect the value of the series at different times.

For every implementation, the first step is to get access to the data or import it.

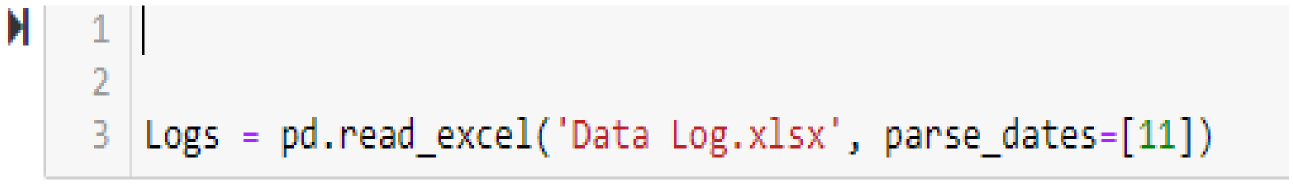


```
In [1]:
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from datetime import datetime
#from pandas import datetime
```

Figure 4.2: importing dependencies

We make use of the python libraries for data analysis, NumPy, pandas, matplotlib among others.

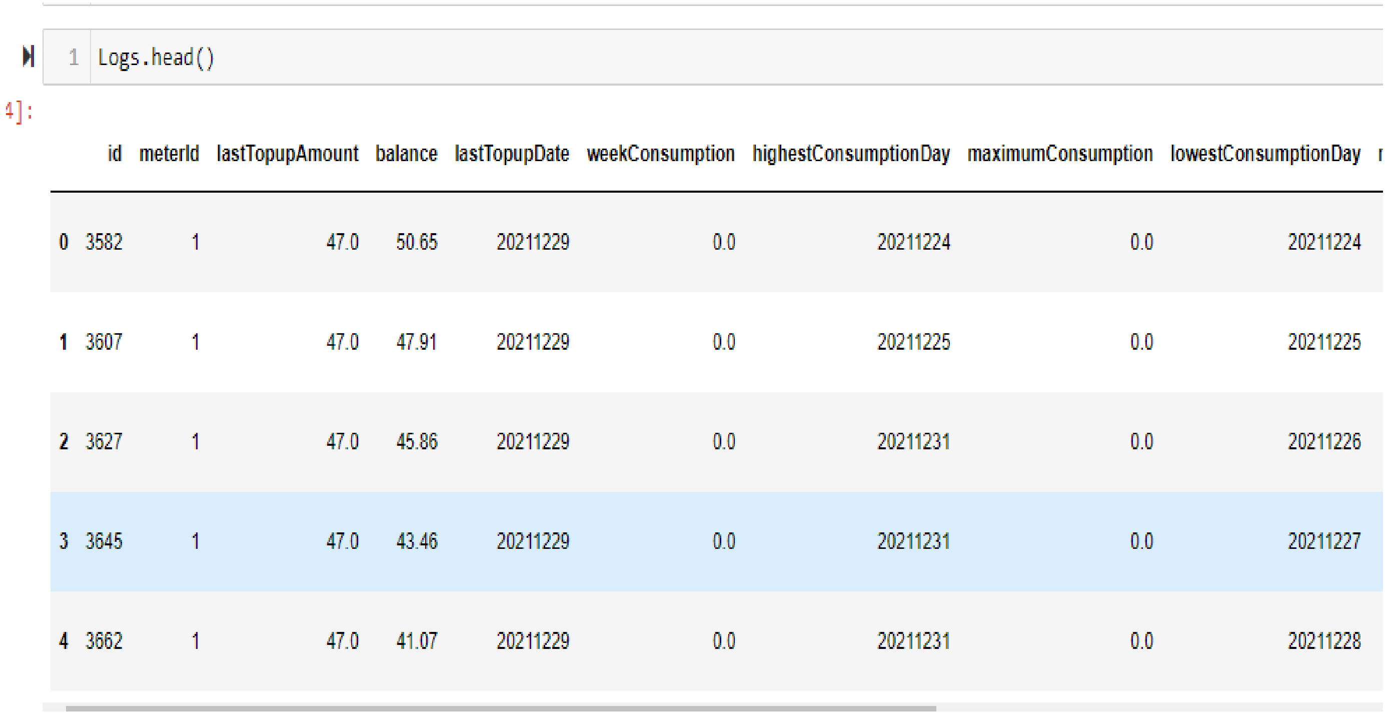
Also, to implement and train the model, we make use of a logged data for one customer. We import the consumption results into the jupyter notebook in a form of a pandas data frame.



```python


Logs = pd.read_excel('Data Log.xlsx', parse_dates=[11])
```

Figure 4.3: importing consumption data of customer

We have the eleventh column of the data frame to be timestamped but upon reading it, it is returned as a string so we pass it as a data value to make it a data as such.



```python
Logs.head()
```

```
4]:
```

	id	meterId	lastTopupAmount	balance	lastTopupDate	weekConsumption	highestConsumptionDay	maximumConsumption	lowestConsumptionDay
0	3582	1	47.0	50.65	20211229	0.0	20211224	0.0	20211224
1	3607	1	47.0	47.91	20211229	0.0	20211225	0.0	20211225
2	3627	1	47.0	45.86	20211229	0.0	20211231	0.0	20211226
3	3645	1	47.0	43.46	20211229	0.0	20211231	0.0	20211227
4	3662	1	47.0	41.07	20211229	0.0	20211231	0.0	20211228

Figure 4.4: Content of data frame

The content of the data frame is as shown in figure 4.4. The next in building the ARIMA model is to make the time series stationary. Again, the AR term in ARIMA indicates that it is a linear regression model that uses its own lags as predictors and linear regression models work best when the predictors are not correlated and are independent of each other. The easiest test in determining the stationarity is to visualize the data. Any form of trend or seasonality in the visualization means the time series is not stationary.

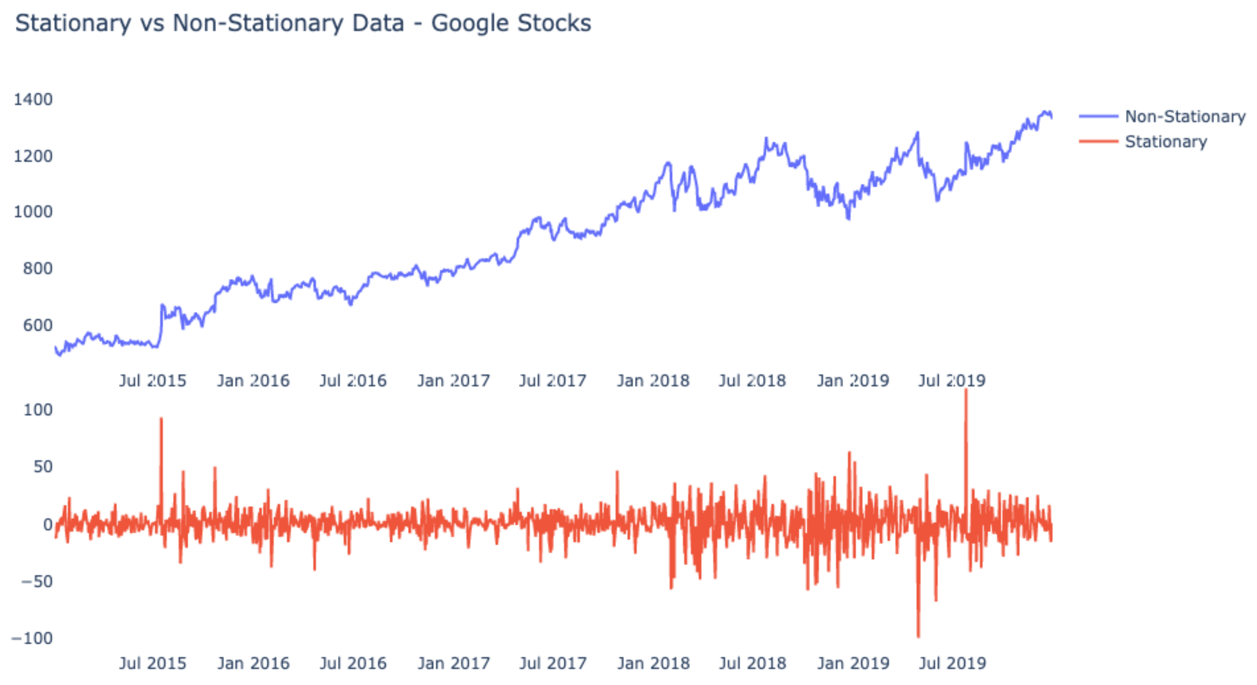


Figure 4.5: Stationary vs non-Stationary data

From the diagram in figure 4.1 we realize the difference between the two plots. The non-stationary data has an upward trend whiles the stationary is relatively the same across the entire time interval.

In making the time series stationary, the most common practice is to use the differencing approach. That is, to subtract the previous value from the current value. From the results of the query that is stored, we have the query returning the balance of the customer, to difference this value is to produce the consumption of the customer. That is, **(Balance at day 2 – Balance at day 1 = Consumption for day 1).** Essentially, the consumption values would have already been a result of a differentiation. Hence this computation is made to produce the consumption of the user since it is what is going to be predicted, its past values ought to be computed for.

Hence this further makes the data the more stationary as would be illustrated below.

```
Out[18]: <AxesSubplot:xlabel='date'>
```

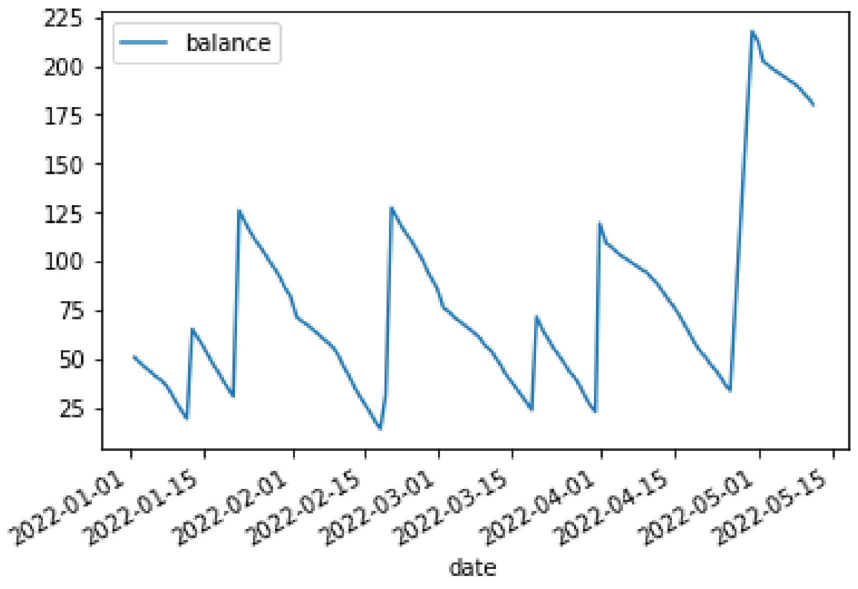


Figure 4.6: plot of balance against time

The figure above shows a plot of the balance as against time. There are some spikes in the values and are also visible in this graph as well. The reasons for such effects have been duly explained in the data collection and cleaning section in this same chapter.

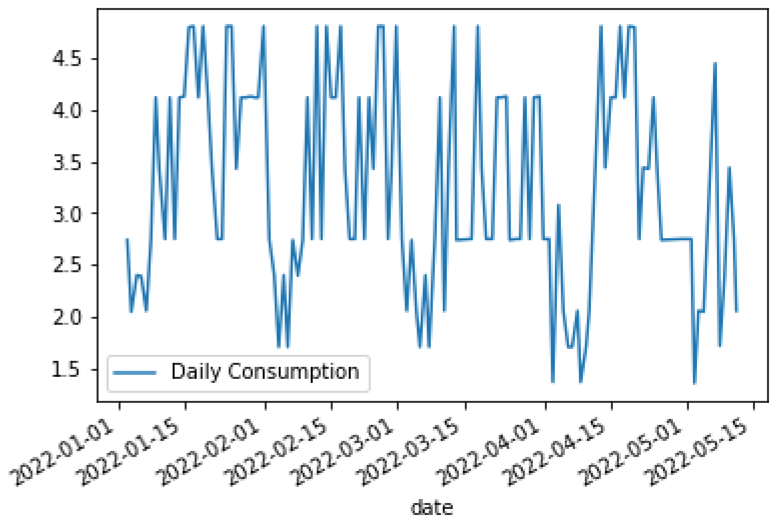
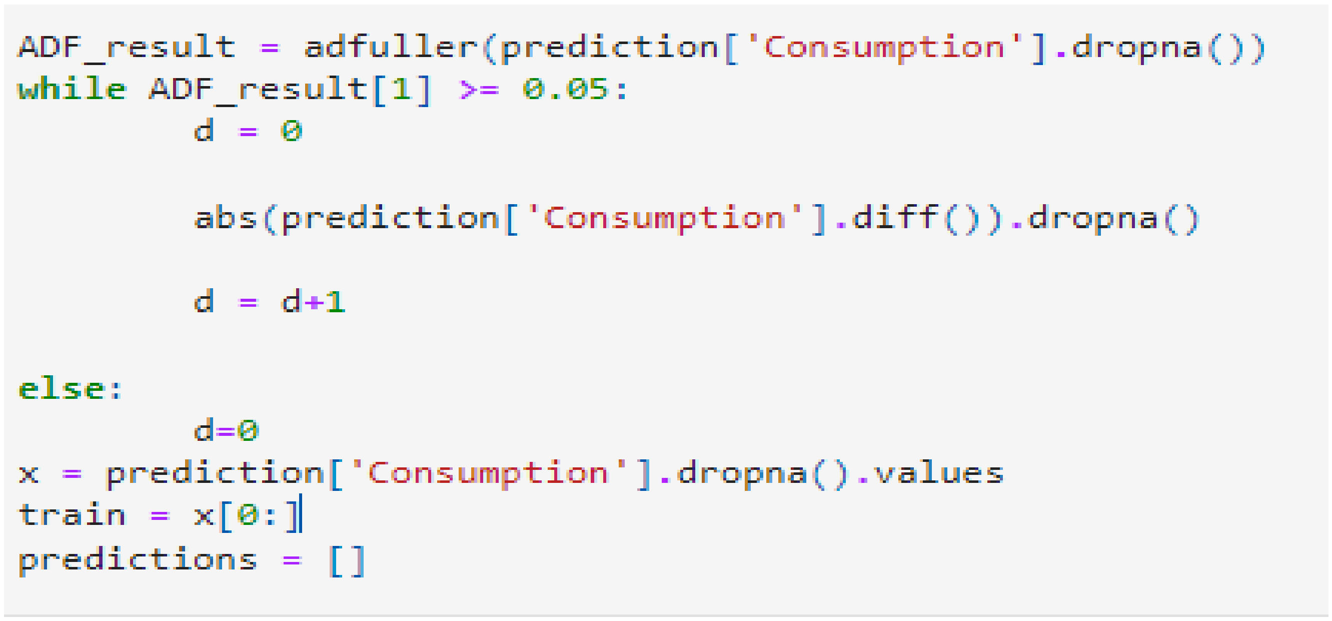


Figure 4.7: plot of consumption against time

The above figure also shows the plot of the consumption of the user as against time. This looks more stationary visually as compared to the graph of the balance although further tests are going to be made to validate this observation so as to make the data ready for training the model.

The order of differentiation once again determines or is equal to the value of ‘d’. Hence for an already stationary data, the value of d = 0. To further validate the stationarity of the series, we used the Augmented Dickey Fuller rest (adfuller()) from the stats models package. Now, the null hypothesis of the Augmented Dickey Fuller test is that time series is non-stationary. Hence, if the p-value of this very test is less than the significance level, 0.05, then we can ignore the null hypothesis and hence conclude that the series is stationary.

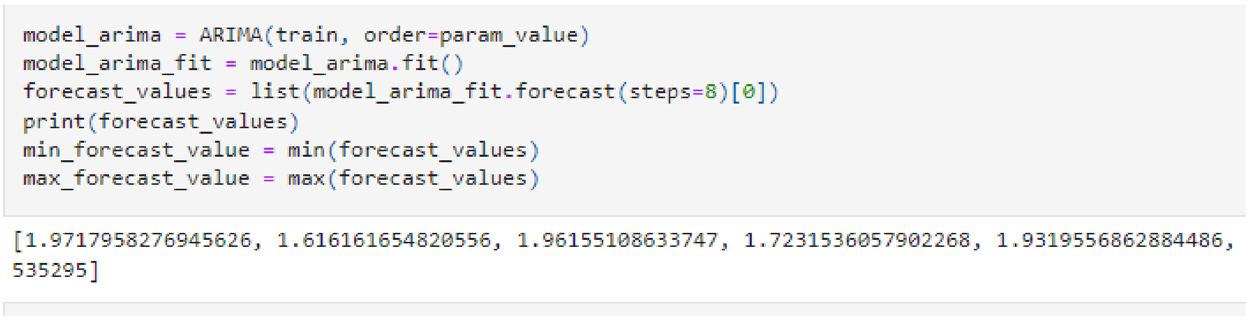


```
ADF_result = adfuller(prediction['Consumption'].dropna())
while ADF_result[1] >= 0.05:
        d = 0

        abs(prediction['Consumption'].diff()).dropna()

        d = d+1

else:
        d=0
x = prediction['Consumption'].dropna().values
train = x[0:]
predictions = []
```

Figure 4.8: code snippet for ADF test

Figure 4.7 above shows the code snippet for the stationarity testing. As stated earlier, we make use of the stats model package which provides a reliable implementation of the Augmented Dickey Fuller test through the adfuller () function in statsmodels.tsa.stattools. The function returns the critical value cutoffs, the number of lags considered for the test, the value of the test statistic and also the p-value, with the p-value being at index one. So, we take the index one value of the ADF_result and compare it with 0.05, which is the significance level.

After the stationarity has been confirmed, we move on to estimate the values of p and q and then eventually make our predictions. For any model of prediction, we need to train it with enough available data so as to make prediction as reliable as possible. We use the entire collated log of the consumption to train the model for prediction of individual consumer consumptions over subsequent months or weeks.



```
model_arima = ARIMA(train, order=param_value)
model_arima_fit = model_arima.fit()
forecast_values = list(model_arima_fit.forecast(steps=8)[0])
print(forecast_values)
min_forecast_value = min(forecast_values)
max_forecast_value = max(forecast_values)
```

```
[1.9717958276945626, 1.616161654820556, 1.96155108633747, 1.7231536057902268, 1.9319556862884486,
535295]
```

Figure 4.9: sample ARIMA test results

From the above diagram in figure 4.8, the estimated parameters of p,d and q are passed into the Arima model, along with the training dataset to make forecasts of future values. The step size for the implementation refers to the number of values to be returned as predictions. Hence for a logged data of Monday to Sunday, a step size of seven will be a prediction of consumption from the next day, Monday to Sunday, giving a total of 7 days. This is the case for data that is logged daily. For a monthly record of data, a step size of four would be a forecast of the next four months from the last month in which the data was logged.

4.3 DESCRIPTIVE ANALYSIS

Under the descriptive analysis section, what we seek to achieve is to answer the question, ‘What can we see from the data?’. That is, it is geared towards drawing more insights from the data.

Below is the flowchart describing the workflow for the descriptive analysis section.

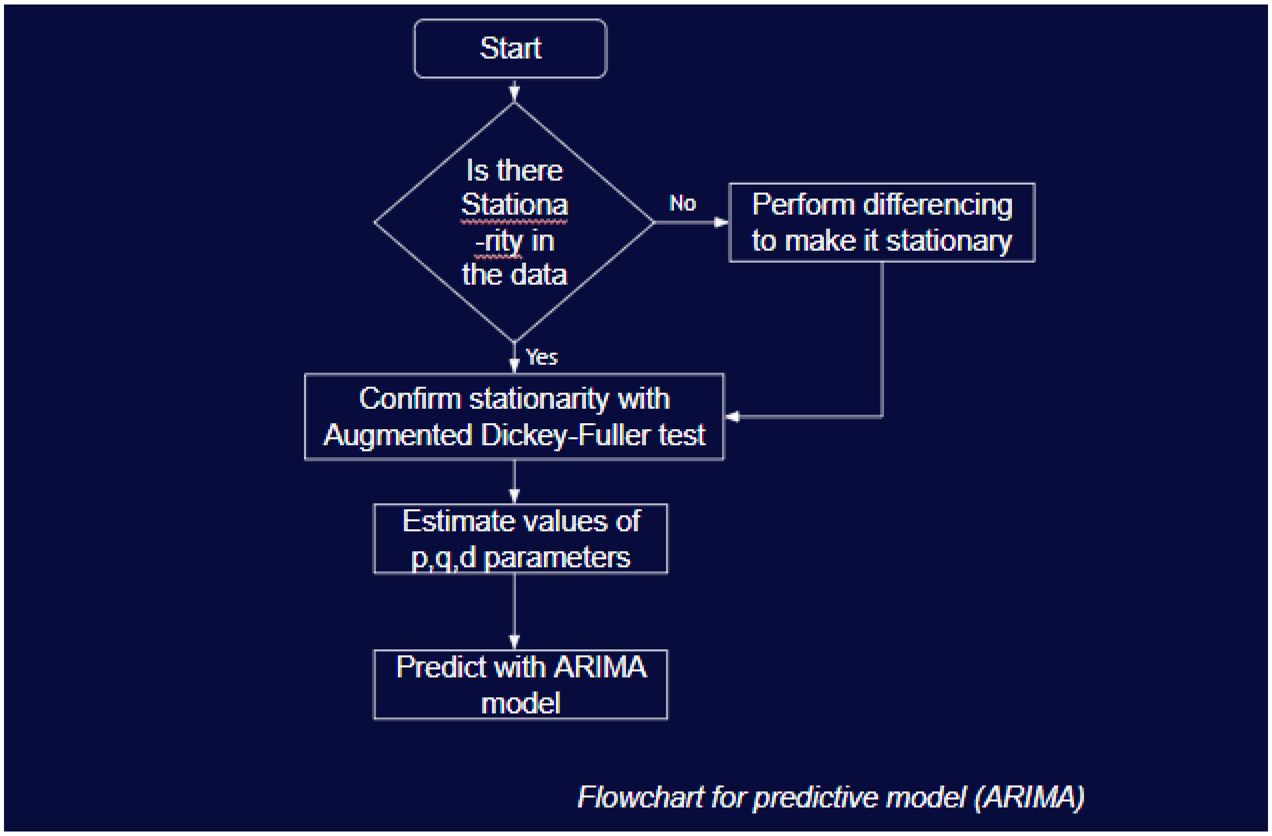

Start
Is there Stationa -rity in the data
No
Perform differencing to make it stationary
Yes
Confirm stationarity with Augmented Dickey-Fuller test
Estimate values of p,q,d parameters
Predict with ARIMA model
Flowchart for predictive model (ARIMA)

Figure 4.10: Descriptive analysis flowchart

4.3.1 CONSUMPTION PATTERN FOR THE WEEK

If a user logs in, the data is retrieved from the database using the meter number. If the mode of prediction is week (from the webapp), the last seven data values for balance is selected along with the date values. It is graphed using **plotly** for a better and enhanced visual representation and other features.

4.3.2 CONSUMPTION PATTERN FOR THE WEEK

If a user logs in, the data is retrieved from the database using the meter number. If the mode of prediction (from the webapp) is month, the last thirty data values for balance is selected along with the date values. It is graphed using **plotly** for a better and enhanced visual representation and other features.

4.4 WEB APPLICATION

The background analysis performed and everything else was integrated into a web application in order to increase ease of usability and general user experience in a process called embedded analytics.

As stated earlier, python Django is employed for building the backend of the application, with html, JavaScript and css being employed for building the frontend of the application.

There is a sign up and sign in page for the smart meter data analysis web app. Thus, when a user registers, the data is logged using the user's meter number. A user registers with the username, password, meter number and mobile number. The data is stored in a database under each meter number as the table. The login page has option for the type of prediction mode the user wants.
If a user tries to sign in without signing up, an error would be thrown for the user to sign up.
But if a user has already registered, he can sign in. The user then enters the username, password and selects the type of consumption pattern he/she wants. The meter number is fetched using the user's meter number and the data is fetched from the database. The consumption pattern is graphed using plotly and the prediction is done for the day, week and month.
Because there are no records for a user's consumption at the start, the user's consumption data would have to be logged for some time, at least a week. However, for the purpose of reliable analytics the higher the volume of accumulated data, the better the analytics results. When the consumption and prediction pattern is displayed, the user can log out of the system and other users can also log in to get their consumption and prediction patterns. Also, error handling is applied during the registration and login.

Consumer (User) use cases of the application

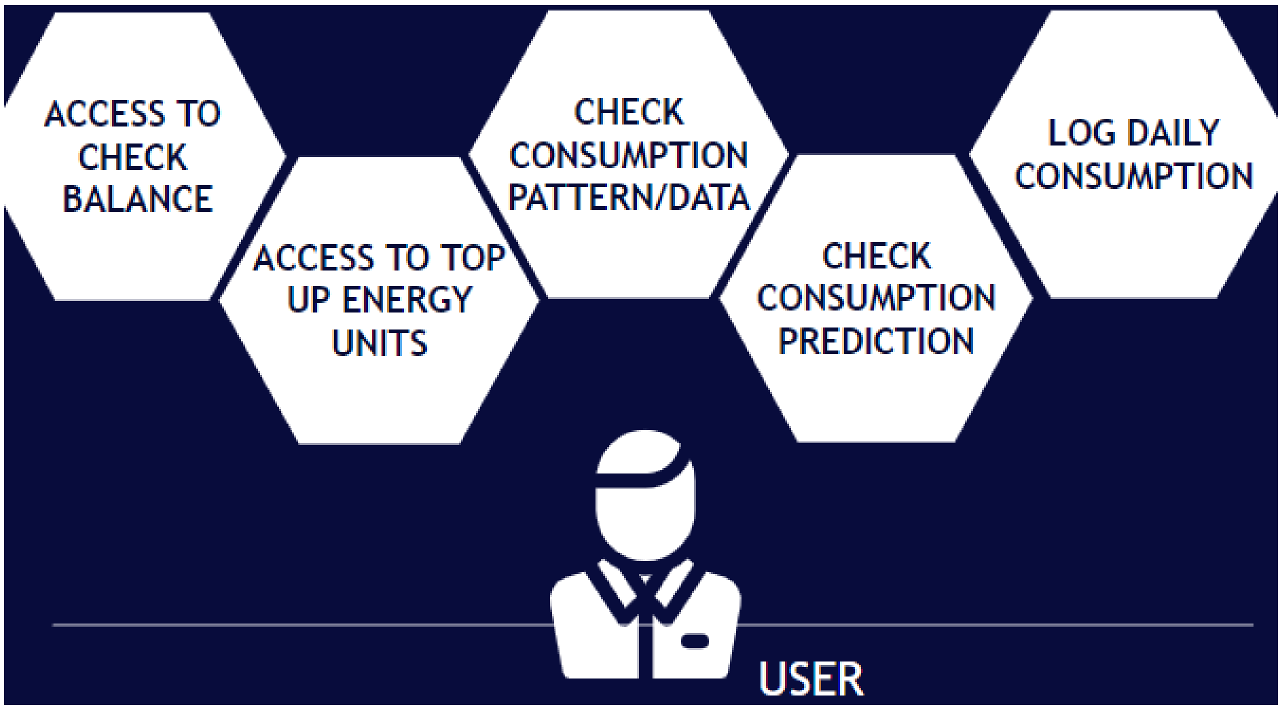

ACCESS TO CHECK BALANCE
ACCESS TO TOP UP ENERGY UNITS
CHECK CONSUMPTION PATTERN/DATA
CHECK CONSUMPTION PREDICTION
LOG DAILY CONSUMPTION
USER

Figure 4.11: Consumer use cases (excluding login and authentication)

5.0 RESULTS AND DISCUSSION

Various functionalities were duly implemented and have been tested. The various use cases of the consumer were implemented and are as follows;

5.1 SIGN UP PAGE

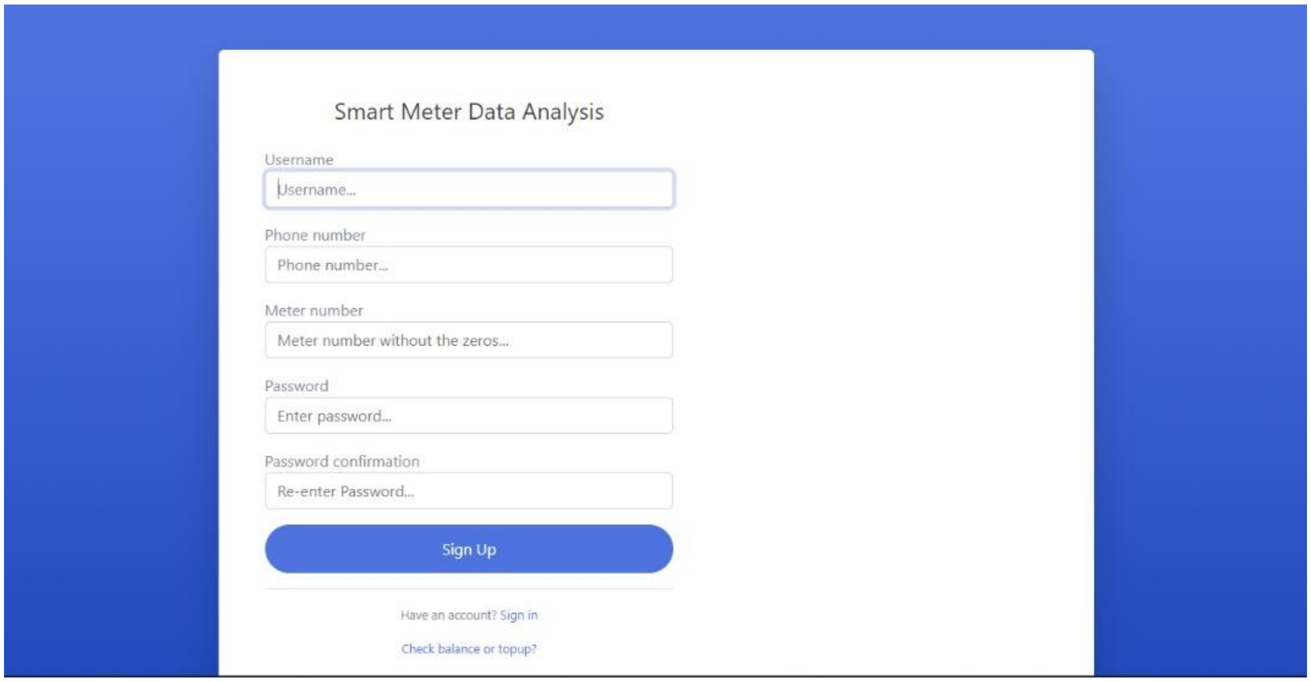


Figure 5.0: Signup page

The signup page is the first page accessible to the user when he loads the application. The page consists of a form that requires;

a. Username: the username is the name of the consumer.
b. The phone number: this is the field for the contact of the user.
c. Meter number: this is a very essential detail of the signup page. The meter number takes an eight-digit input which represents the meter number of the customer or the user.
d. User Password: the password of the user

5.2 SIGN IN PAGE

After the user signs up, the sign in page is loaded and then the user has to input his details.

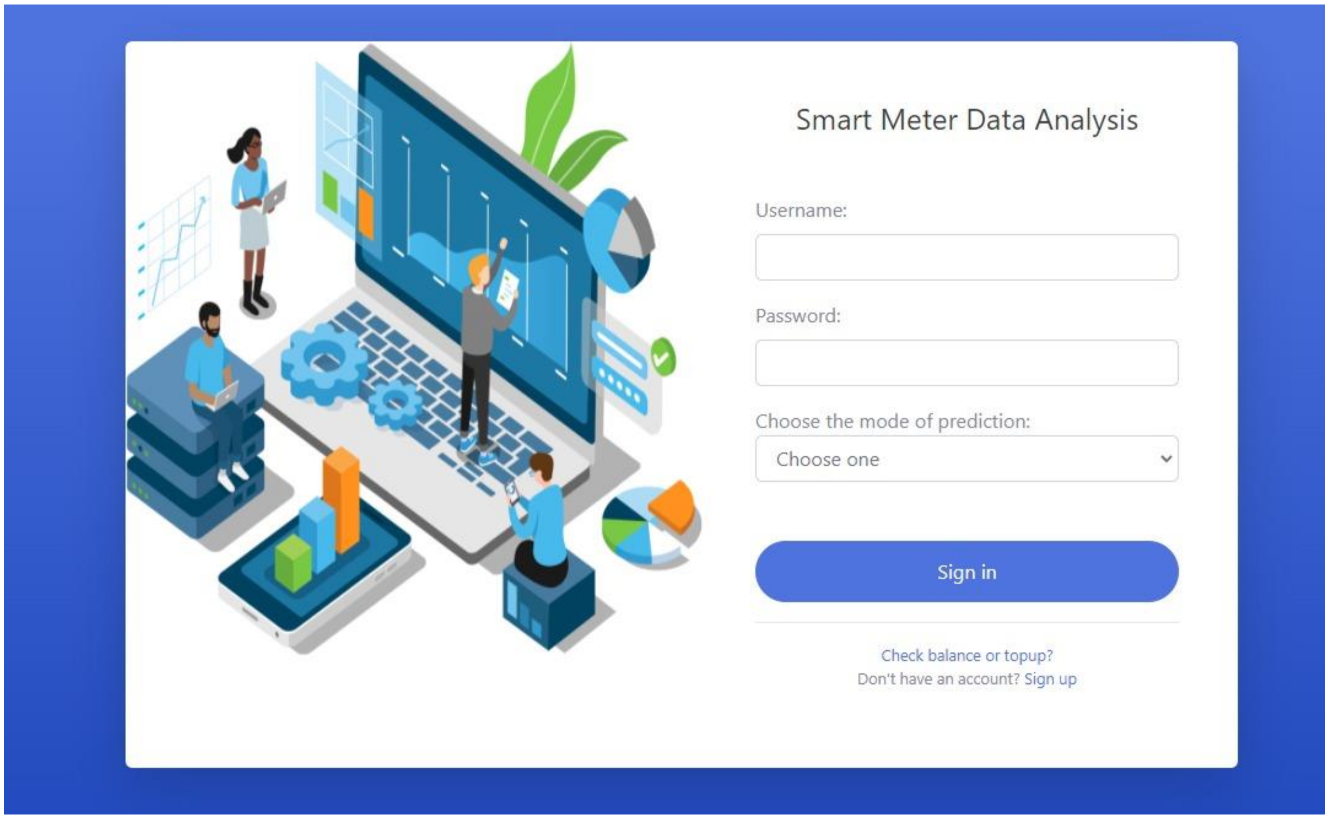
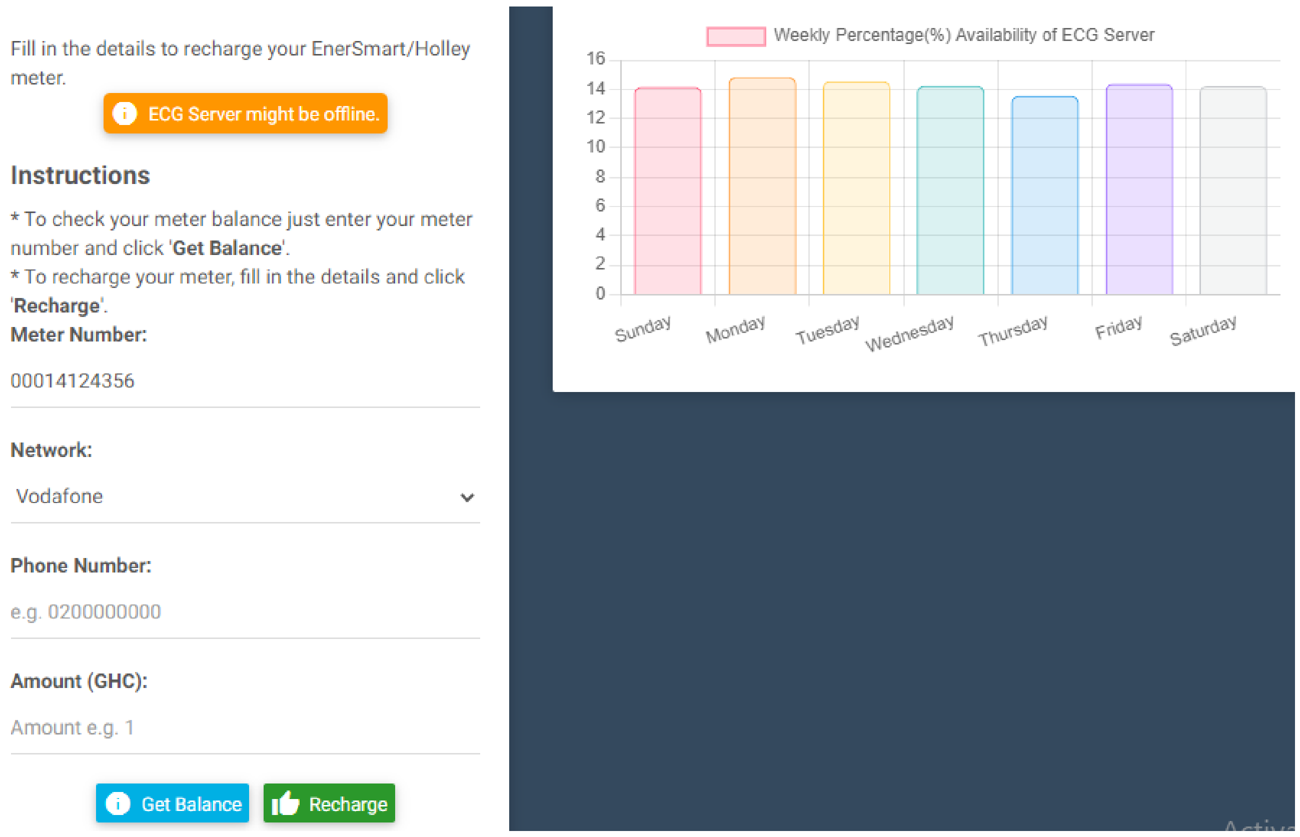

Smart Meter Data Analysis
Username:
Password:
Choose the mode of prediction:
Choose one
Sign in
Check balance or topup?
Don't have an account? Sign up

Figure 5.1: Sign-in page

The sign-in page consists of the username, the password or the user that was set during the signup process and then the option of choosing the mode of prediction. The mode could either be weekly based or monthly based representation. When these details are provided, the user can then sign in into his or her dashboard which shows the consumption pattern of the user in a visualized manner (graph). Then also, from the sign in page, the user can choose to sign up if he doesn’t have an account already or check balance or top-up. Upon clicking on the Check balance or top up option, the user is redirected to the Ener Smart recharge platform to check his or her balance or top up, however it doesn’t require submission of the signup details.


Fill in the details to recharge your EnerSmart/Holley meter.

ECG Server might be offline.

Instructions

* To check your meter balance just enter your meter number and click '**Get Balance**'.
* To recharge your meter, fill in the details and click '**Recharge**'.

Meter Number:

00014124356

Network:

Vodafone

Phone Number:

e.g. 0200000000

Amount (GHC):

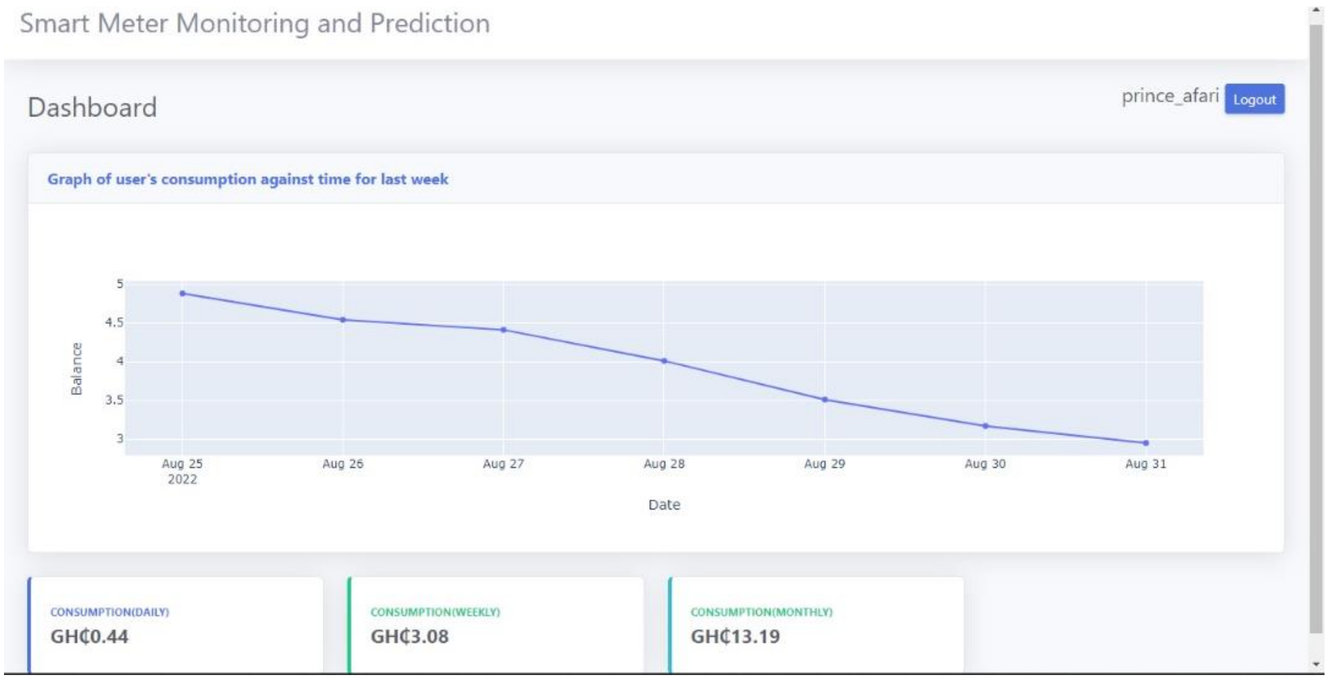
Amount e.g. 1

Figure 5.2: Checking balance and topping up

The redirection brings the user to check his balance or top up.

5.3 ANALYTICS AND VISUALIZATION

After signing in and choosing the preferred mode, the user dashboard shows the various consumption details as well as the consumption prediction.



CONSUMPTION(DAILY) GH₵0.44

CONSUMPTION(WEEKLY) GH₵3.08

CONSUMPTION(MONTHLY) GH₵13.19

Figure 5.3: User dashboard

The graph shows the consumption pattern of the user and also has a hover feature that shows the details of consumption at any point of the graph. It also shows the prediction for the daily consumption, the weekly consumption and the monthly consumption.

Since there aren't any readily available data of a customer, we begin to log user's consumption at occasional times or specific times when he signs up so as to make the analysis after enough data has been collated. Once the user signs up and tries to access his consumption pattern among other parameters, an empty dashboard is returned as shown below;

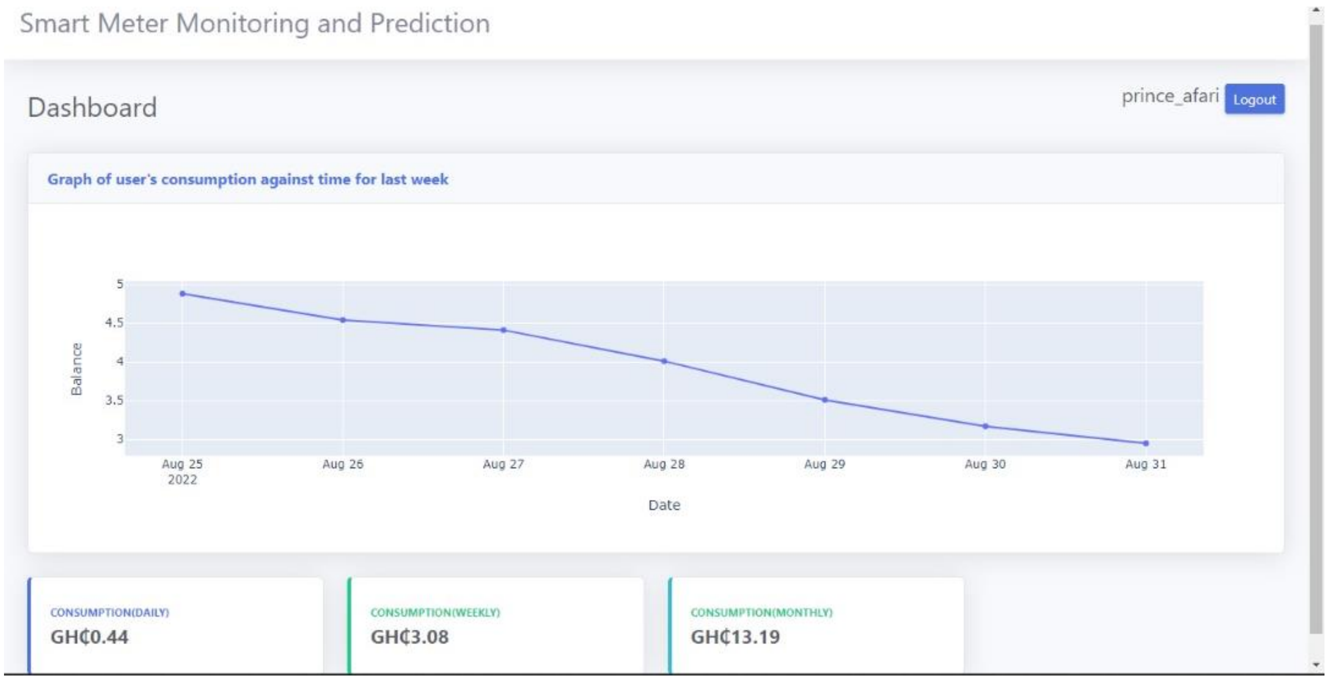


Figure 5.4: User dashboard without logged data

5.4 TESTING

The system was setup and tested for the performance of the various functionalities of the system on a number of meters and user details too. New users were created with the requisite details. The new users were able to sign up and check the functionality of all the implementations on the user signup page. The users were also able to log in into the system using their details and already existing meter numbers were assigned to them. The meter IDs/numbers were successfully loaded into the database and also the consumption data for these users were successfully retrieved for analysis. The web application was also successful in redirecting users to purchase energy units or to top up as well. All of these were successfully done and reflected in the customer's dashboard.

5.5 EVALUATION

From the results obtained, it indicates that the overall system functionality worked very well. The expected theoretical result was for the application to successfully get user details, log and store consumption data and also retrieve data from the database for various analysis to be performed on it. Results also showed that the ECG server goes offline on some occasions so an error is returned when you try to access meter data on those days. The consumption for such days is considered to be zero and the average consumption of the previous days up until that day is computed for and assigned to that date thereof.

6.0 CONCLUSION AND RECOMMENDATION

6.1CONCLUSION

This project highlighted the drawbacks of the smart electricity metering systems already in use in Ghana. It proposed a system (application) for monitoring electricity consumption and forecasting which was designed in order to deal with the various identified inconveniences. The various inconveniences highlighted were as follows;

a. Unavailability of consumption data.

   User consumption data is very essential for efficient management of electricity consumption. Users can more easily reduce their electricity usage and energy costs if they have access to accurate information on how much energy they are using. Both home and commercial energy users may attest to this. Consumers may save money in their house by learning how their energy use varies depending on the day of the week or season. Users can modify how their appliances are used to reduce their energy costs or upgrade their home with energy-saving technologies. [31]
   From the developed system, users can monitor how their consumption varies as the weeks/months go by. So are able to better manage their spending so as to save money and also reduce energy loss.

b. Inconveniences faced by consumers

   In the case where the ECG servers are offline, customers would have to travel to vendors to purchase energy units and then load it onto the meters. Sometimes, these customers would have to travel very far distances and join queues to purchase these units and in the case of weekends, holidays, late hours of the night or any special activity, these vendors may be unavailable. And as stated earlier, this situation may have a negative effect both on the customer and the utility company as well. To mitigate this, a preview of the customers consumption pattern may advise the customer on consumption management, coupled with a forecast of his consumption over a specified period may reduce the inconveniences customers go through to some extent since they now have an idea or are informed of how much they're going to consume over the next weeks or months.

With these inconveniences effectively addressed, there is a high chance that there would be a higher customer experience, satisfaction and comfort in the usage of the deployed smart meters, thus along with customer education, would help in eliminating customer desperation and anxieties that often lead to meter tampering and power theft. Ultimately, this would help increase revenue, reduce electricity loss and reduce meter tampering and power theft which eventually leads to the frequent need for ECG to change tampered meters.

6.2 CHALLENGES ENCOUNTERED

The main challenges that were encountered during the course of this project include the following:

1. Unavailability of user consumption data
2. Unavailability of ECG servers on some occasions

6.3 FUTURE WORKS

Another system could be built for the utility company to also remotely monitor customer status in-terms of consumption, remotely transmit tariffs, notifications and control information, provide consumption history and analysis online, among other benefits. The system could also be augmented to be able to function as a fully-fledged smart energy meter monitoring system.

6.4 RECOMMENDATIONS

The system designed (application) is simple to use, cost effective, user friendly and of high benefits to the individual consumers, both residential and commercial consumers. It could be implemented to replace the existing system. Also, utilities in Ghana such as the Electricity Company of Ghana (ECG), the Northern Electricity Distribution Company, (NEDCo) and the Enclave power company Ltd (EPC) could adopt this system to increase effectiveness in operation. [32]

REFERENCES

[1] Alan W. Hodges and Mohammad Rahmani, “Economic Impacts of Generating Electricity, Wood to Energy”, pp.1-8, September 2007.

[2] Xiao, L.; Wang, C.; Liang, T.; Shao, W. A combined model based on multiple seasonal patterns and modified firefly algorithm for electrical load forecasting. Energy Policy 2016, 167, 135–153. [CrossRef]

[3] Li, S.; Ma, X.; Yang, C. A novel structure-adaptive intelligent grey forecasting model with full-order time power terms and its application. Comput. Ind. Eng. 2018, 120, 53–67. [CrossRef]

[4] Zhao, H.; Guo, S.; Xue, W. Urban saturated power load analysis based on a novel combined forecasting model. Information 2015, 6, 69–88. [CrossRef]

[5] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, “A survey on smart grid potential applicatio00ns and communication requirements,” IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp. 28–42, 2013

[6] https://www.i-scoop.eu/industry-4-0/smart-grids-electrical-grid/

[7] F. Benzi, N. Anglani, E. Bassi, and L. Frosini, “Electricity smart meters interfacing the households,” IEEE Transactions on Industrial Electronics, vol. 58, no. 10, pp. 4487–4494, 2011. [8] S. Nambi, E. Pournaras, and R. V. Prasad, “Temporal self-regulation of energy demand,” IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp. 1196–1205, 2016.

[9] https://www.ghanaweb.com/GhanaHomePage/business/ECG-to-add-700-000-smart-metersacross-the-country-by-end-of-2020-Bawumia-870853

[10] Gerhard Eisenbeiss, “The Cat and Mouse Game of Meter Tampering”, http://www.ee.co.za/article/cat-mouse-game-meter-tampering.html, Elster, EE Publishers (Pty) Limited, April 2015.

[11] P. Sekhar, “Secured Techno- Economic Growth of India: Unleashing Hidden Growth Potential”, Micro Media Marketing Pvt. Ltd, October 2014.

[12] GhanaWeb,“Power Thief Nabbed”, http://www.ghanaweb.com/GhanaHomePage/NewsArchive/Power-thief-nabbed358250, Daily Guide, May 2015

[13] Ghana Web, “Electrician Jailed Three Months for Tampering with Prepaid Meter”, http://www.ghanaweb.com/GhanaHomePage/crime/artikel.php?ID=243290, GNA, June 2012

[14] K. Sheelasobanarani, S. Dinesh Raja, B. Dhanaraj, K. Manickam and K. Karthick Raja, “A Prepaid Energy Meter for Efficient Power Management”, International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 3, March 2014

[15] Francis Edzorna Mensah, “ECG to procure Smart Prepaid Meters to help increase its cash flow”, http://radioxyzonline.com/ecg-to-procure-smart-prepaid-meters-to-helpincrease-its-cashflow/, Radio XYZ, June 201

[16] https://www.sciencedirect.com/science/article/pii/S0148296319303078

[17] S. Prasad, S. B. Avinash,” Smart meter data analytics using OPENTSDB and HADOOP”, Proc. IEEE Conf. Innov. Smart Grid Technol. Asia (ISGT Asia’13), pp. 1-6, 2013.

[18] D. Alahakoon, X. Yu, “Smart electricity Meter data intelligence for Future Energy Systems: A Survey”, IEEE Trans. Industrial Informatics, vol. 12, issue 1, pp. 425-436, Feb. 2016.

[19] https://www.researchgate.net/publication/348913611_Watt’s_up_at_Home_Smart_Meter_Data_Analytics_from_a_Consumer-Centric_Perspective

[20] https://www.researchgate.net/publication/310665777_Smart_Meter_Data_Analytics_Systems_Algorithms_and_Benchmarking

[21] https://ieeexplore.ieee.org/document/8469896

[22] https://www.techtarget.com/searchdatamanagement/definition/data-analytics

[23] https://www.sisense.com/glossary/python-for-data-analysis/

[24] https://www.postgresql.org/about/

[25] O. Elgabry, “Software Engineering — Software Process and Software Process Models (Part 2),” Medium Corporation, 2017. [Online]. Available: https://medium.com/omarelgabrysblog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc.

[26] M. Sami, “Software Development Life Cycle Models and Methodologies - Mohamed Sami,” Melsatar, 2012. [Online]. Available: https://melsatar.blog/2012/03/15/software- development-life-cycle-models-and-methodologies/.

[27] https://www.ibm.com/cloud/learn/data-visualization
[28] https://matplotlib.org/
[29] https://online.stat.psu.edu/stat462/node/188/
[30] https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

[31] https://nebeskie.com/blogs/DataAnalytics.htm

[32] https://www.energymin.gov.gh/sector-overview

APPENDIX: VIEWS.PY FILE

```
from django.shortcuts import redirect, render
from django.contrib.contenttypes.models import ContentType
from django.db import connection
from django.contrib import messages
import pandas as pd
import plotly.express as px
import statistics as stats
import numpy as np
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
import warnings
import itertools
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth import login, logout
from .forms import SignUpForm
from .models import User
import datetime
import string
from django.views.decorators.csrf import csrf_exempt

letters = string.ascii_letters + string.punctuation
```

```
def logout_view(request):
    if request.method == 'POST':
        logout(request)
        return redirect('smart_app:signin')
```

```
def signup_view(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        user_username = request.POST.get('username')
        meter = str(request.POST.get('meter_number'))
        phone = str(request.POST.get('phone_number'))
        meter_1 = meter
        meter = 't' + meter

    #-----------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-------------------------------
        if User.objects.filter(phone_number=phone).exists():
            messages.info(request, 'Phone number exists.')
            return redirect('smart_app:signup')
        elif User.objects.filter(meter_number= meter).exists():
            messages.info(request, 'Meters number exists.')
            return redirect('smart_app:signup')
    #---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
------------------------------
        else:
            for item in phone:
                if item in letters:
                    messages.info(request, 'Please enter a valid phone number.')
                    return redirect('smart_app:signup')
            for item in meter_1:
                if item in letters:
                    messages.info(request, 'Please enter a valid meter number.')
                    return redirect('smart_app:signup')
        if form.is_valid():
            form.save()
            user_1 = User.objects.get(username = user_username)
            meter_number = user_1.meter_number
            meter_number = 't' + str(meter_number)
            cursor = connection.cursor()
            date_value = str(datetime.datetime.now().strftime("%Y-%m-%d"))
            cursor.execute("CREATE TABLE %s (balance REAL NOT NULL, date DATE NOT
NULL);" % meter_number)
            # cursor.execute("INSERT INTO %s VALUES (0, to_date(%s::text, 'YYYY-MM-
DD'));" % (meter_number, date_value))
            cursor.execute("INSERT INTO %s VALUES (0, '%s')" % (meter_number,
date_value))
            SignUpForm()
            return redirect('smart_app:signin')
    else:
        form = SignUpForm()
    return render(request, 'smartapp/register.html',{'form': form})
```

```
@csrf_exempt
def signin_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(data = request.POST)
        user_username = request.POST.get('username')
        analysis_type = request.POST['analysis']
        pass_word = request.POST.get('password')

        #Check if user exists
        if not User.objects.filter(username=user_username).exists():
            messages.info(request, "Username does not exist")
            return redirect('smart_app:signin')
        elif analysis_type == "":
            messages.info(request, "Select a mode of prediction")
            return redirect('smart_app:signin')
        else:
            user_1 = User.objects.get(username = user_username)
            meter_number = user_1.meter_number
            meter_number = 't' + str(meter_number)
            if form.is_valid():
                user = form.get_user()
                display_results(meter_number, analysis_type)
                chart ,pred_type,title = display_results(meter_number, 
analysis_type)
                login(request, user)
                form = AuthenticationForm()
                return render(request, 'smartapp/results.html', {'chart': chart, 
'pred_type': pred_type, 'title': title,'user': user})
    else:
        form = AuthenticationForm()
    return render(request, 'smartapp/signin.html',{'form': form})


def display_results(meter_number, analysis_type):
    balance = []
    date = []
    cursor = connection.cursor()
    cursor.execute("SELECT date, balance FROM " + meter_number)
    fetch = cursor.fetchall()
    fetch_len = len(fetch)
    for item in fetch:
        date.append(item[0])
        balance.append(item[1])
    if fetch_len > 1:
        if analysis_type == 'week':
            fig = px.line(
                x = date[-7:],
                y = balance[-7:],
                # title = "Graph of user's consumption against time for last week",
                labels= {'x' : 'Date', 'y' : 'Balance'},
                markers=True,
            )
            fig.update_layout(title={
                'font_size' : 22,
                'xanchor' : 'center',
                'x' : 0.5,
            })

            chart = fig.to_html()
            title = "Graph of user's consumption against time for last week"
            pred_type = prediction_pattern(balance, date)
            return chart, pred_type, title
```

```
        elif analysis_type == 'month':
            fig = px.line(
                x = date[-30:],
                y = balance[-30:],
                # title = "Graph of user's consumption against time for last 
month",
                labels= {'x' : 'Date', 'y' : 'Balance'},
                markers=True,
            )

            fig.update_layout(title={
                'font_size' : 22,
                'xanchor' : 'center',
                'x' : 0.5,
            })

            chart = fig.to_html()
            title = "Graph of user's consumption against time for last month"
            pred_type = prediction_pattern(balance, date)
            return chart, pred_type, title

    else:
        if analysis_type == 'week':
            fig = px.line(
                x = date[:],
                y = balance[:],
                # title = "Graph of user's consumption against time for last week",
                labels= {'x' : 'Date', 'y' : 'Balance'},
                markers=True,
            )

            fig.update_layout(title={
                'font_size' : 22,
                'xanchor' : 'center',
                'x' : 0.5,
            })

            chart = fig.to_html()
            title = "Graph of user's consumption against time for last week"
            pred_day = 'GH₵0.00'
            pred_week = 'GH₵0.00'
            pred_month = 'GH₵0.00'
            pred_type = [
                {'day' : pred_day, 'week' : pred_week, 'month' :pred_month}
            ]
            return chart, pred_type, title
```

```
        elif analysis_type == 'month':
            fig = px.line(
                x = date[:],
                y = balance[:],
                # title = "Graph of user's consumption against time for last 
month",
                labels= {'x' : 'Date', 'y' : 'Balance'},
                markers=True,
            )

            fig.update_layout(title={
                'font_size' : 22,
                'xanchor' : 'center',
                'x' : 0.5,
            })

            chart = fig.to_html()
            title = "Graph of user's consumption against time for last month"
            pred_day = 'GH₵0.00'
            pred_week = 'GH₵0.00'
            pred_month = 'GH₵0.00'
            pred_type = [
                {'day' : pred_day, 'week' : pred_week, 'month' :pred_month}
            ]
            return chart, pred_type, title
```

```
def prediction_pattern(balance, Date):
    smart_meter_data = pd.DataFrame(list(zip(Date, balance)), columns=['date', 
'balance'])
    #------------------------------------------------------------------------------
--------------------------------
    smart_meter_data['Consumption'] = 
abs(smart_meter_data['balance'].diff()).dropna()
    smart_meter_data['Consumption'] = 
smart_meter_data['Consumption'].replace(np.nan, 0)
    #------------------------------------------------------------------------------
------------------
    for i in range(len(smart_meter_data['Consumption'])):
        if smart_meter_data['Consumption'][i] >= 6:
    #         smart_meter_data['Consumption'] = 
smart_meter_data['Consumption'].replace(smart_meter_data['Consumption'][i], 5)
            mean_consumption = stats.mean(smart_meter_data['Consumption'][:i])
            if mean_consumption <= 6:
                smart_meter_data['Consumption'] = 
smart_meter_data['Consumption'].replace(smart_meter_data['Consumption'][i], 
mean_consumption)
            else:
                smart_meter_data['Consumption'] = 
smart_meter_data['Consumption'].replace(smart_meter_data['Consumption'][i], 5)
    #------------------------------------------------------------------------------
------------------   -------------------
    smart_meter_data['Consumption'] = smart_meter_data['Consumption'].dropna()
    prediction = smart_meter_data[['Consumption','date']]
    prediction.index = pd.to_datetime(prediction.date)
    #------------------------------------------------------------------------------
----------------------------------------------------------
    ADF_result = adfuller(prediction['Consumption'].dropna())
    while ADF_result[1] >= 0.05:
            d = 0

            abs(prediction['Consumption'].diff()).dropna()

            d = d+1

    else:
            d=0
    x = prediction['Consumption'].dropna().values
    train = x[0:]
    predictions = []
    #------------------------------------------------------------------------------
----------------------------------
```

```
    warnings.filterwarnings('ignore')
    p=d=q = range(0,3)
    pdq = list(itertools.product(p,d,q))
    param_values = []
    error_values = []
    for param in pdq:
        try:
            model_arima = ARIMA(train, order=param)
            model_arima_fit = model_arima.fit()
            # print(param,model_arima_fit.aic)
            param_values.append(param)
            error_values.append(model_arima_fit.aic)
        except:
            continue
    min_error_value = min(error_values)
    error_value_index = error_values.index(min_error_value)
    param_value = param_values[error_value_index]
    # param_value = (0,0,2)
    #--------------------------------------------------------------------------------
----------------------------------------
    model_arima = ARIMA(train, order=param_value)
    model_arima_fit = model_arima.fit()
    forecast_values = []
    forecast_values = model_arima_fit.forecast(steps=8)[0]
    # print(forecast_values)
    min_forecast_value = np.min(forecast_values)
    max_forecast_value = np.max(forecast_values)
    #--------------------------------------------------------------------------------
----------------------------------------
    print('Your average daily consumption is between GH₵%s and GH₵%s'%
(round(min_forecast_value,2), round(max_forecast_value,2)))
    print('Your average weekly consumption is GH₵%s and GH₵%s'%(round(7 *
min_forecast_value,2),round(7 * max_forecast_value,2)))
    print('Your average monthly consumption is GH₵%s and GH₵%s'%(round(30 *
min_forecast_value,2),round(30 * max_forecast_value,2)))

    pred_day = 'GH₵%s'% round(min_forecast_value,2)
    pred_week = 'GH₵%s' %round(7 * min_forecast_value,2)
    pred_month = 'GH₵%s'%round(30 * min_forecast_value,2)
    pred_type = [
                {'day' : pred_day, 'week' : pred_week, 'month' :pred_month}
            ]
    return pred_type
```