ELECTRICITY BILLING SYSTEM

A Project Report

on

Electricity Billing System

By

LETLHOGONOLO

CHAUKE

AND

KHUMO MATHE

SYNOPSIS

Electricity consumers are often faced with the problem of inaccuracy and delay in monthly billing due to some drawbacks. Thus, it is essential to have an efficient system for such purposes via electronic platform with consideration to proximity.

The proposed system automates the conventional process of paying electricity bill by visiting the Electricity Board which is tiresome and time consuming. It is also designed to automate the electricity bill calculation and payment for user convenience. The system is developed with Java swings as the base programming language which can be used to develop websites, web applications and web services.

The Microsoft Structured Query Language (SQL) server is also used for creating back-end database. The system would be having two logins: the administrative and user login. The administrator can view the user's account details and can add the customer's information of consuming units of energy of the current month in their account.

The Admin must feed the system with the electricity usage data into respective user’s account. The system then calculates the electricity bill for every user and updates the information into their account every month. Users can then view their electricity bill and pay before the month end.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| CHAPTER NO. | CONTENT | PAGE NO. |
| 1. | Introduction   * 1. Preamble   2. Problem Statement | **1**  **2** |
| 2. | Analysis and System Requirements   * 1. Existing And Proposed System   2. Software and Hardware Requirements | **3**  **3** |
| 3. | System Design And Modeling   * 1. Preliminary Design      1. UML Diagram      2. Schema Diagram   2. Normalization      1. First Normal Form(1NF)      2. Second Normal Form(2NF)      3. Third Normal Form(3NF) | **5**  **6**  **10**  **12** |
| 4 | Implementation   * 1. Implementation of Operations   2. Implementation of SQL Statements   3. Algorithm or Pseudocode of Implementation | **14**  **14**  **16** |
| 5. | Testing   * 1. Testing Process   2. Testing Objective   3. Levels of Testing      1. Unit Testing      2. Integration Testing   5.3.2 System Testing | **20**  **20**  **20 – 26** |
| 6. | Discussion And Snapshots   * 1. Tables   2. Snapshots | **27**  **30** |
| 7. | Future Scope And Limitations | **51** |
| 8. | Conclusion | **52** |
| 9. | Bibliography | **53** |

## Chapter 1

**INTRODUCTION**

Electricity Billing System is a software-based application.

1. This project aims at serving the department of electricity by computerizing the billing system.
2. It mainly focuses on the calculation of units consumed during the specified time and the money to be charged by the electricity offices.
3. This computerized system will make the overall billing system easy, accessible, comfortable, and effective for consumers.

To design the billing system more service oriented and simple, the following features have been implemented in the project. The application has high speed of performance with accuracy and efficiency.

The software provides facility of data sharing, it does not require any staff as in the conventional system. Once it is installed on the system only the meter readings are to be given by the admin where customer can view all details, it has the provision of security restriction.

The electricity billing software calculates the units consumed by the customer and makes bills, it requires small storage for installation and functioning. There is provision for debugging if any problem is encountered in the system.

The system excludes the need of maintaining paper electricity bill, administrator does not have to keep a manual track of the users, users can pay the amount without visiting the office. Thus, it saves human efforts and resources.

## Preamble

We, the owners of our project, respect all customers and make them happy with our service. The main aim of our project is to satisfy customer by saving their time by payment process, maintaining records, and allowing the customer to view his/her records and permitting them to update their details. The firm handles all the work manually, which is very tedious and mismatched.

The objectives of our project are as follows:

* **To keep the information of consuming unit energy of current month.**
* **To keep the information of Consumer.**
* **To keep the information of consuming unit energy of previous month.**
* **To calculate the units consumed every month regularly.**
* **To generate the bills adding penalty and rent.**
* **To save the time by implementing payment process online.**
  1. **Problem Statement**

The manual system is suffering from a series of drawbacks. Since whole of the bills is to be maintained with hands the process of keeping and maintaining the information is very tedious and lengthy to customer. It is very time consuming and laborious process because, staff need to be visited the customers place every month to give the bills and to receive the payments. For this reason, we have provided features Present system is partially automated (computerized), existing system is quite laborious as one must enter same information at different places.

* 1. **Proposed Solution**
* This project system excludes the need of maintaining paper electricity bill as all the electricity bill records are managed electronically.
* Administrator doesn't have to keep a manual track of the users. The system automatically calculates the fine.
* Users don't have to visit to the office for bill payment.
* There is no need of delivery boy for delivery bills to user's place.
* Thus, it saves human efforts and resources.

**CHAPTER 2**

**ANALYSIS AND SYSTEM REQUIREMENT**

**2.1 Existing and Proposed System**

The conventional system of electricity billing is not so effective; one staff must visit each customer’s house to note the meter readings and collect the data. Then, another staff must compute the consumed units and calculate the money to be paid. Again, the bills prepared are to be delivered to customers. Finally, individual customer must go to electricity office to pay their dues.

Hence, the conventional electricity billing system is uneconomical, requires many staffs to do simple jobs and is a lengthy process overall. In order to solve this lengthy process of billing, a web based computerized system is essential. This proposed electricity billing system project overcomes all these drawbacks with the features. It is beneficial to both consumers and the company which provides electricity.

With the new system, there is reduction in the number of staffs to be employed by the company. The working speed and performance of the software is faster with high performance which saves time. Furthermore, there is very little chance of miscalculation and being corrupted by the staffs.

**2.2 Software & Hardware Requirements**

**Hardware Requirements:**

* Hardware Specification: -Processor Intel Pentium V or higher
* Clock Speed: -1.7 GHz or more
* System Bus: -64 bits
* RAM: 4GB or higher
* HDD: 500GB or higher

**Software Requirements:**

* Operating System: -Windows 8 or higher
* Software: Microsoft SQL Server
* Front End: -Java core/swings (Netbeans) and Eclipse
* Back End: -MySQL Workbench

**CHAPTER 3**

**SYSTEM DESIGN AND MODELING**

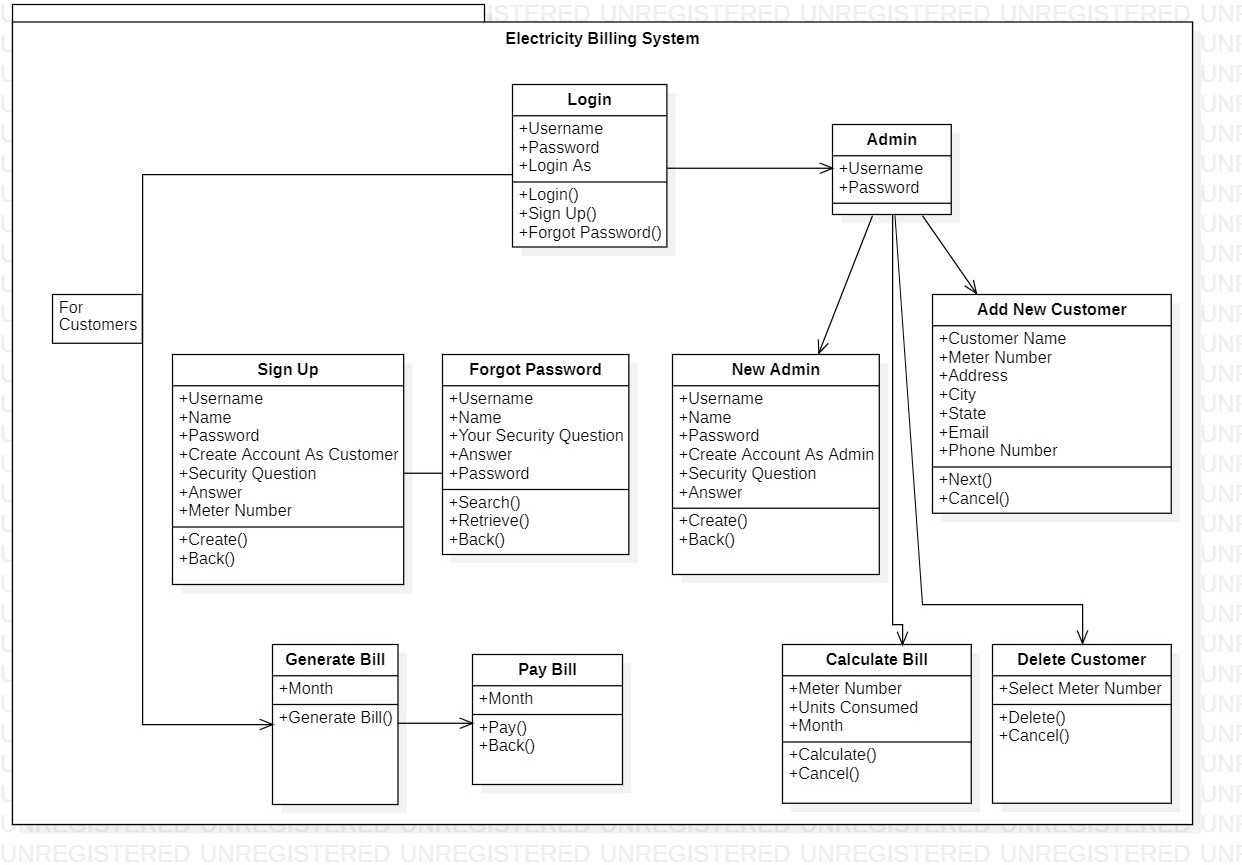
**3.1 Preliminary Design**

System design is an abstract representation of a system component and their relationship and which describe the aggregated functionally and performance of the system. It is also the plan or blueprint for how to obtain answer to the question being asked. The design specifies various type of approach.

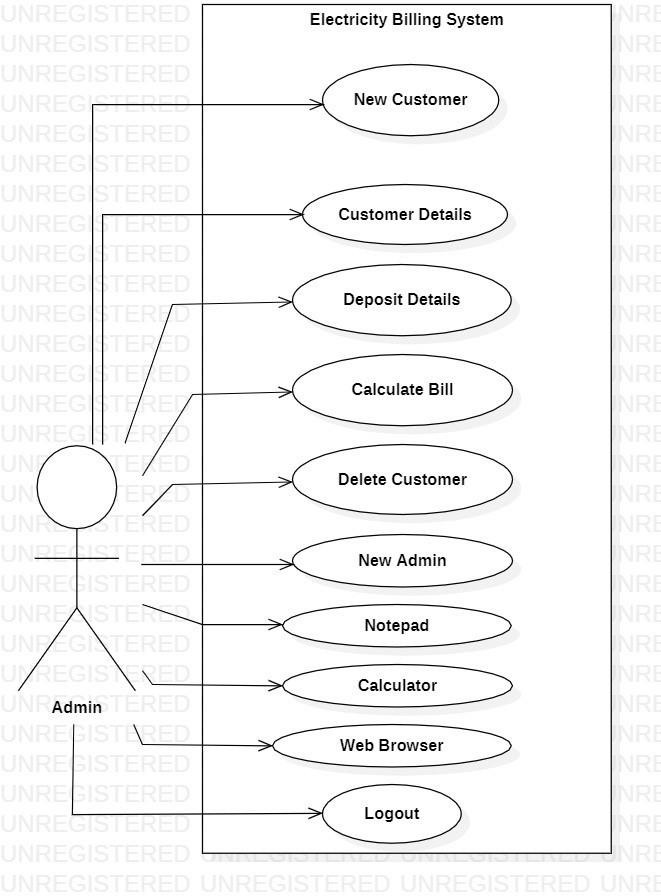
Database design is one of the most important factors to keep in mind if you are concerned with application performance management. By designing your database to be efficient in each call it makes and to effectively create rows of data in the database, you can reduce the amount of CPU needed by the server to complete your request, thereby ensuring a faster application.

**3.1.1 UML Diagram**

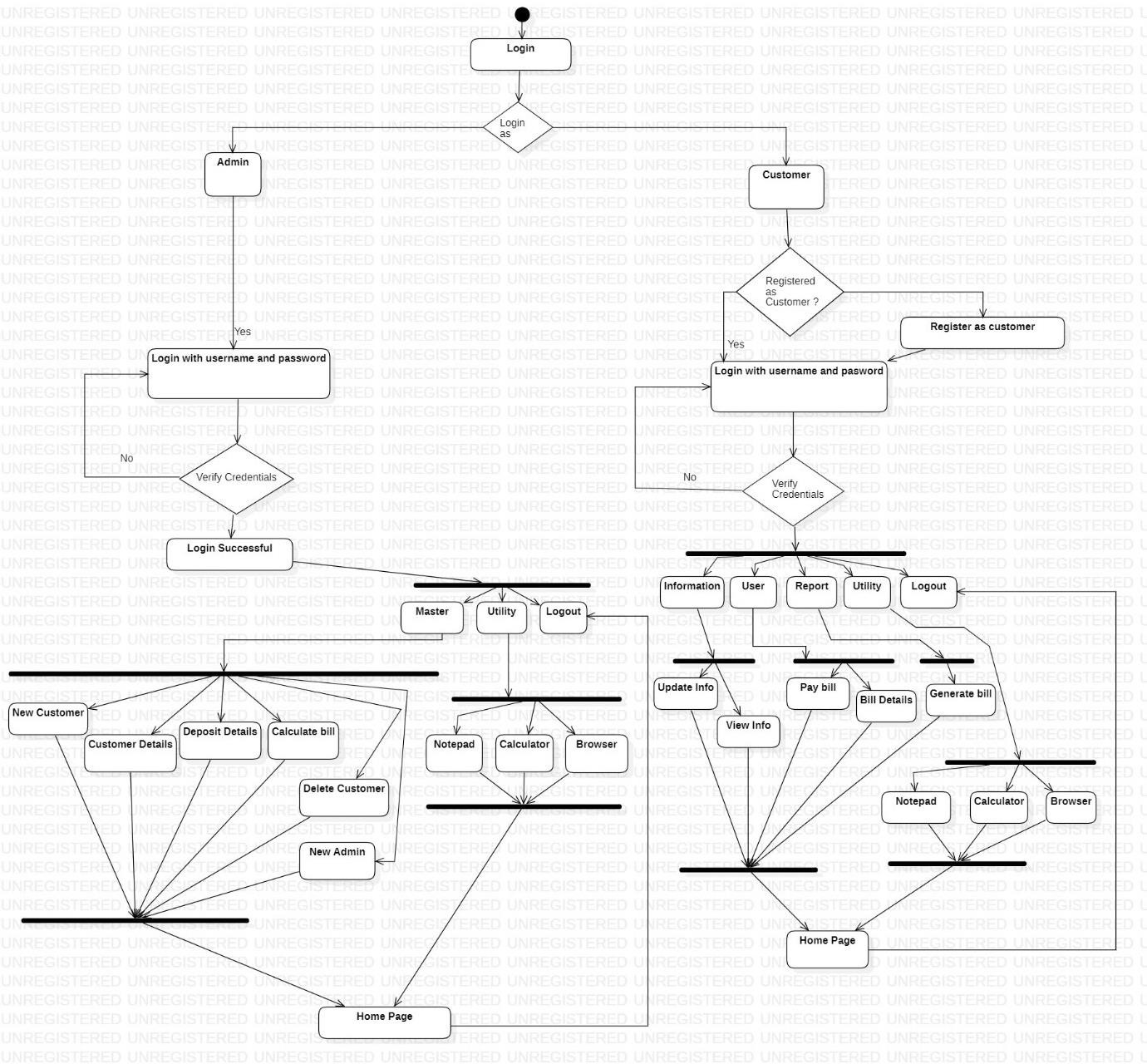
**Class Diagram:**



**Use Case Diagrams:**



**Activity Diagram:**



**3.1.2 Schema Diagram**

Database schema is described as database connections and constraints. It contains attributes. Every database has a state instances represent current set of databases with values. There are different types of keys in a database schema.

A primary key is a table column that can be used to uniquely identify every row of the table. Any column that has this property, these columns are called candidate key. A composite primary key is a primary key consisting of more than one column. A foreign is a column or combination of columns that contains values that are found in the primary key of some table.

All the attributes of each table are interconnected by foreign key which is primary key in another column and composite key. Primary key cannot be null. The fact that many foreign key values repeat simply reflects the fact that its one- to-many relationship. In one-to-many relationship, the primary key has the one value and foreign key has many values.

Figure 3.1.2 is a Schema diagram of Electricity Billing System which has six tables i.e., login, customer, tax, rent, bill, and meter\_info where each table contain attributes some with primary key, foreign key. In the login table there are 6 attributes "meter\_no”, "username”, “password”, "user”, "question”, "answer”. The customer table has 7 attributes "name”, "meter\_no"(primary key), "address”, "city”, "state”, “email”, "phone”. The rent table has 3 attributes "cost\_per\_unit"(primary key), " meter\_rent”, "service\_charge". The tax table has 3 attributes " service\_tax”, "nersa\_tarrif”, "gst”. The bill table has 5 attributes "meter\_no"(foreign key that references the primary key of the customer table meter\_no), "month”, "units”,"total\_bill”, "status”. The meter\_info table has 6 attributes "meter\_no"(foreign key that references the primary key of the customer table meter\_no), "meter\_location”, "meter\_type”, "phase\_code”, " bill\_type”, "days ".

**3.1.2 Schema Diagram**

**Login**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Meter No | Username | Password | User | Question | Answer |

**Customer**

**Rent**

**Tax**

**Bill**

**Meter Info**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Meter No | Address | City | State | Email | Phone |

|  |  |  |
| --- | --- | --- |
| Cost Per Unit | Meter Rent | Service Rent |

|  |  |  |
| --- | --- | --- |
| Service Tax | Nersa\_tarrif | GST |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Meter No | Month | Units | Total Bill | Status |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Meter No | Meter Location | Meter Type | Phase Code | Bill Type | Days |

**3.2 Normalization**

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

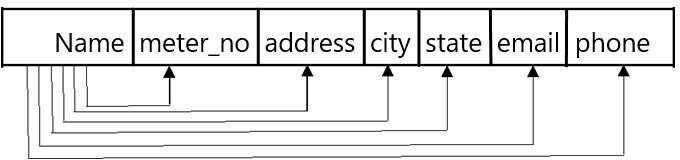
Let’s discuss about anomalies first then we will discuss normal forms with examples. Anomalies in DBMS There are three types of anomalies that occur when the database is not normalized. These are –Insertion, update and deletion anomaly.

**3.2.1 First Normal Form(1NF)**

As per the rule of the first normal form

* All rows must be unique (no duplicate rows).
* Each Cell must only contain a signle value (not a list).
* Each value should be non-divisible (can’t be split down further).

Customer



**3.2.2 Second Normal Form(2NF)**

As per the rule of second normal form,

* Database must be in First Normal Form.
* Non partial dependency-All non-prime attributes should be fully functionally dependent on the candidate key.

**3.2.3Third Normal Form(3NF)**

As per the rule of third normal form,

* Database must be in First and Second Normal Form.
* Non transitive dependency-All fields must only be determinable by the primary/composite key, not by other keys.

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 Implementation of Operations**

* **Adding Customer:** Here admin can add new customer to the customer list who started using electricity bill system.
* **Searching Deposit Details:** Here admin can search according to meter number and month to view deposit details.
* **Viewing Details:** Here admin and user can view customer details and about details.
* **Updating Customer:** Here customer can update his/her details by using meter\_no of the customer.

**4.2 Implementation of SQL Statements**

**Insert Statement**

* The INSERT INTO statement is used to insert new records in a table.
* The INSERT INTO syntax would be as follows: INSERT INTO table\_name VALUES (value1, value2, value3, ...).
* The following SQL statement insert's a new record in the “customer” table: Insert into customer VALUES (“Lebogang”,”800617”,”1127 Block F”,”Pretoria”,“Gauteng”,“[Lebogang67@gmail.com](mailto:Lebogang67@gmail.com)”, “27816173346”).

**Update Statement**

* An SQL UPDATE statement changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a condition.
* The UPDATE syntax would be as follows: UPDATE table\_name SET column\_name = value, column\_name = value…[WHERE condition]

The following SQL statement update's a new record in the “customer” table: UPDATE TABLE customer SET email= [su@gmail.com](mailto:su@gmail.com) WHERE meter\_no =”12345”.

**Delete Statement**

* The DELETE statement is used to delete existing records in a table.
* The DELETE syntax would be as follows: DELETE FROM table\_name WHERE condition
* The following SQL statement deletes a record in the “customer” table: delete from customer where meter\_no = 12345.

**Create Statement**

* The CREATE TABLE Statement is used to create tables to store data. Integrity Constraints like primary key, unique key, foreign key can be defined for the columns while creating the table.
* The syntax would be as follows: CREATE TABLE table\_name (column1 data-type, column2 data-type, column3 data-type, PRIMARY KEY (one or more columns))
* The following SQL statement creates a table “customer” table: create table customer (name varchar (30), meter\_no varchar (20) primary key, address varchar (50), city varchar (20), state varchar (30), email varchar (30), phone varchar (30));
* The following SQL statement creates a table “login” table: create table login (meter\_no varchar(30), username varchar(30), password varchar(30), user varchar(30), question varchar(40), answer varchar(30));
* The following SQL statement creates a table “tax” table: create table tax (cost\_per\_unit int(20) primary key, meter\_rent int(20), service\_charge int(20), service-tax int(20), nersa\_tarrif int(20), gst int(20));
* The following SQL statement creates a table “meter\_info” table: create table meter\_info (meter\_no varchar(30), foreign\_key(meter\_no) references customer(meter\_no) on delete cascade, meter\_location varchar(10), meter\_type varchar(15), phase\_code int(5), bill\_type varchar(10), days int(5));

**4.3 Algorithm or Pseudocode of Implementation**

**Explanation of Algorithm or Pseudocode of System**

* Start System
* Enter Login name and password
* On clicking the login button
* Connect to database
* Query database to know whether user credentials are correct
* If not, deny access and return login page with an error message
* If correct, check if credentials for administrator
* If yes, allow login
* Set admin session, re-direct administrator to admin login page
* If no, allow login set user session
* Re-direct user to user homepage

**Algorithm or pseudocode of admin:**

**Login:**

* This program will allow the admin to enter the username and password.
* If the entered credentials are correct, then the login will be successful otherwise need to be signup.
* After successful login the admin will be redirected to admin portal page where he/she can do the following activities.

**New Customer:**

* This program will allow the admin to enter the customer details and automatically generates unique meter number.
* If customer name, address, city, state, email and phone number is entered, insert the values into customer.

else print error

while(next = true)

enter the meter\_info

details else print

meter\_info error

* Submit the details of customer that has been entered by clicking onto next button.
* If we need to cancel the particulanrs that has been entered click onto cancel option.
* If we need to submit the particulars that has been entered click onto submit option.

**Customer Details**

* This program will allow the admin to view customer details.
* If we need to print the particulars that has been viewed click onto print option

**Deposit Details**

* This program will allow the admin to view bill details. If we need to sort the particulars based on meter\_no and month.
* If we need to search the particulars that has been viewed click onto search option.
* If we need to print the particulars that has been viewed click onto print option.

**Tax Details**

* This program will allow the admin to add tax details and insert the values into tax.
* Else print error.
* Submit the details of tax that has been entered by clicking onto submit button.
* If we need to cancel the particulars that has been entered click onto cancel option

**Calculate Bill:**

* This program will allow the admin to calculate total bill when units consumed are inserted where meter\_no and month is selected.
* Insert the values into bill else print error.
* Submit the details of tax that has been entered by clicking onto submit button.
* If we need to cancel the particulars that has been entered click onto cancel option.

**Algorithm or Pseudocode of Customer:**

**Login:**

* This program will allow the customer to enter the username and password. If the entered credentials are correct, then the login will be successful otherwise need to be signup with the meter\_no which is given by admin.
* After successful login the customer will be redirected to customer portal page where he/she can do the following activities.

**Update Info:**

* This program will allow the customer to update the customer details. If customer address, city, province, email and phone number is updated.
* Update the values into customer else print error.
* Update the details of cutomer that has been updated by clicking onto update button.
* If we need to cancel the particulars that has been updated, click onto back option.

**View Info:**

* This program will allow the customer to view his/her own details.
* If we need to go back from the particulars that has been viewed click onto back option.

**Pay Bill:**

* This program will allow the customer to view bill details and redirects to pay the bill where status will be updated.
* If we need to cancel the particulars that has been viewed click onto back option.
* If we need to pay the bill amount that has been viewed click onto pay option.

**Bill Details:**

* This program will allow the customer to view bill details.
* If we need to print the particulars that has been viewed click onto print option.

**Generate Bill:**

* This program will allow the customer to generate bill when meter\_no and month is selected.
* Generate the details by clicking on generate bill button.

NOTE: Utility(notepad and calculator) query and logout are given to both customer and admin portals.

**CHAPTER 5**

**TESTING**

This chapter gives the outline of all the testing methods that are carried out to get a bug free application.

## Testing process

Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, compiles with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested. In some cases, test cases are done based on the system requirements specified for the product/software, which is to be developed.

## Testing objectives

The main objectives of testing process are as follows:

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has high probability of finding an as yet undiscovered error.
* A successful test is one that uncovers an as yet undiscovered error.

## Levels of Testing

Different levels of testing are used in the testing process; each level of testing aims to test different aspects of the system. The basic levels are unit testing, integration testing, system testing and acceptance testing.

* + 1. **Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design the module. The software built, is a collection of individual modules. In this kind of testing exact flow of control for each module was verified. With detailed design consideration used as a guide, important control paths are tested to uncover errors within the boundary of the module.

**Table 5.1: Negative Test Case for phone number insertion**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Input** | **Expected Output** | **Error** | **Resolved** |
| Input  phone number | 98977 | Phone  number is invalid | Length  of  phone number is | Consume  () |
| not equal to | |
| 10 |  |
| Input | 98977agv | Phone | Alphabets | \_ |
| phone | number is invalid | are being take n as input for pho ne number |
| number |

**Table 5.2: Positive Test Case for phone number insertion**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Input** | **Expected Output** | **Error** | **Resolved** |
| Input | 27817159963 | Expected | \_ | \_ |
| Phone Number |  | Output is Seen |

**Table 5.3: Negative Test Case for email insertion**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Input** | **Expected Output** | **Error** | **Resolved** |
| Input email | Sai1.i n | Email is invalid | Email is not in  a format given | Consume () |

**Table 5.4: Positive Test Case for email insertion**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Input** | **Expected Output** | **Error** | **Resolved** |
| Input email | [sizwe123@gmail.com](mailto:sizwe123@gmail.com) | Expected | \_ | \_ |
| output is seen |

**Table 5.5: Negative test Case for customer name insertion**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Input** | **Expected Output** | **Error** | **Resolved** |
| Input | Sana123 | Name is | Numbers are | Consume () |
| customer | invalid | being taken as |
|  |  | input for |
| name |  | name |

**Table 5.6: Positive Test Case fot customer name insertion**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function**  **Name** | **Input** | **Expected**  **Output** | **Error** | **Resolved** |
| Input  customer name | Lebogang | Expected output is seen | **\_** | **\_** |

* + 1. **Integration Testing**

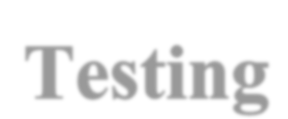
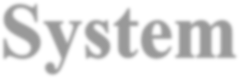
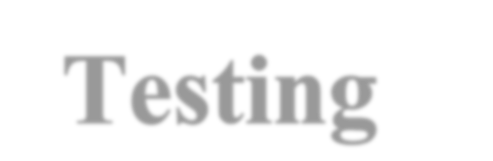
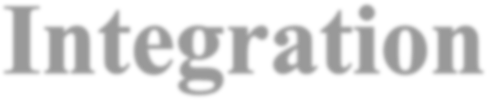
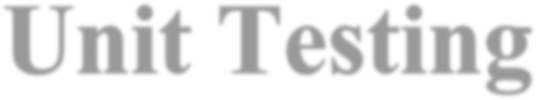
The second level of testing is called integration testing. In this, many class-tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly. We have been identified and debugged.

**Table 5.7: Test case on basis of generation of bill**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Input** | **Expected Output** | **Error** | **Resolved** |
| Negative  searching of total\_bill | 12334(meter\_no) January(month) | Details seen  but not total\_bill | Output not seen | Consume () |
| Positive  searching of total\_bill | 800617(meter\_no) January(month) | Must display  full generated  bill with total\_bill | **\_** | **\_** |

**Table 5.8: Test Case on basis of deposit details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Input** | **Expected Output** | **Error** | **Resolved** |
| Negative  searching of depositedetails | 12334(meter\_no) January(month) | Details not seen | Output not seen | Consume () |
| Positive  searching of total\_bill | 12334(meter\_no) January(month) | Must display depositedetails | \_ | \_ |



**Testing Diagram**

* + 1. **System Testing**

Here the entire application is tested. The reference document for this process is the requirement document, and the goal is to see IF the application meets its requirements. Each module and component of ethereal was thoroughly tested to remove bugs through a system testing strategy. Test cases were generated for all possible input sequences and the output was verified for its correctness.

**Table 5.9: Test Cases for the project**

|  |  |  |
| --- | --- | --- |
| **Steps** | **Action** | **Expected output** |
| Step1 | The screen appears when | A page with different |
| choice | the users run the program.   1. If admin login 2. If customer login | menu’s appears.  1.Admin panel opens and 2.Customer panel opens |
| Step 2 | The screen appears when | A window for adding |
| the admin logs in and selects any one of the  menus from the click of the mouse. | new customer, inserting tax, calculate bill, view deposit details etc. |
| Selection 1 | ❖New Customer  ❖Customer  Details  ❖Deposit Details  ❖Calculate Bill  ❖Tax Details  ❖Delete Customer  ❖New Admin |  |

|  |  |  |
| --- | --- | --- |
| Step 2.1 | The screen appears when the customer login and selects any one of the  menus from the click of the mouse | A window for generating bill, update customer details, view details, generating bill |
| Selection 2 | ❖Update Details  ❖View Details |  |
| Selection 2a | ❖Generate Bill |  |
| Selection 2b | ❖Pay Bill  ❖Bill Details |  |

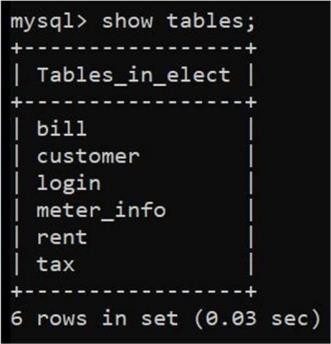
**CHAPTER 6**

**DISCUSSION AND SNAPSHOTS**

**6.1 TABLES:**

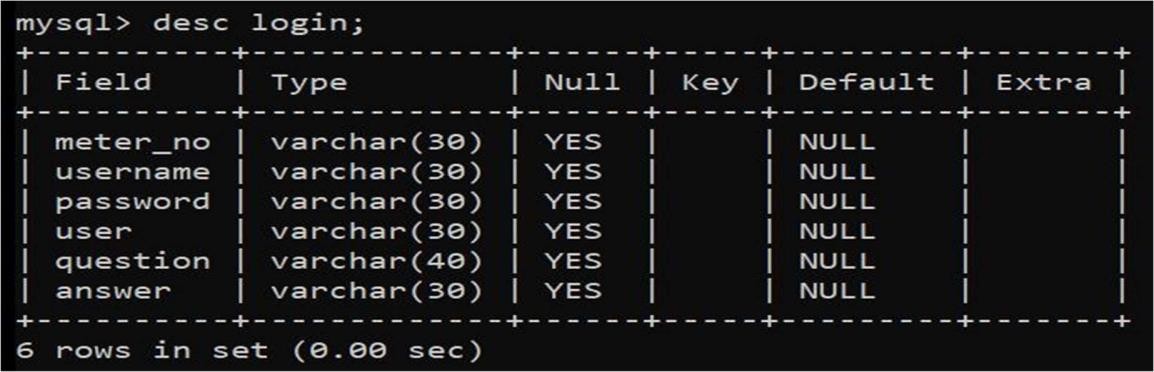
The given below table is a snapshot of backend view of the localhost and the structures of the tables present in Electricity Billing System. The tables present are login, customer, tax, bill, meter\_info.

* + - The login is used to store the details of login’s admin and customer with meter\_no.
    - The customer is used to store details of customer.
    - The tax is used to store tax values.
    - The rent is used to store rent values.
    - The bill is used to store details of bill of meter.
    - The meter\_info is used to store information of meter placed.



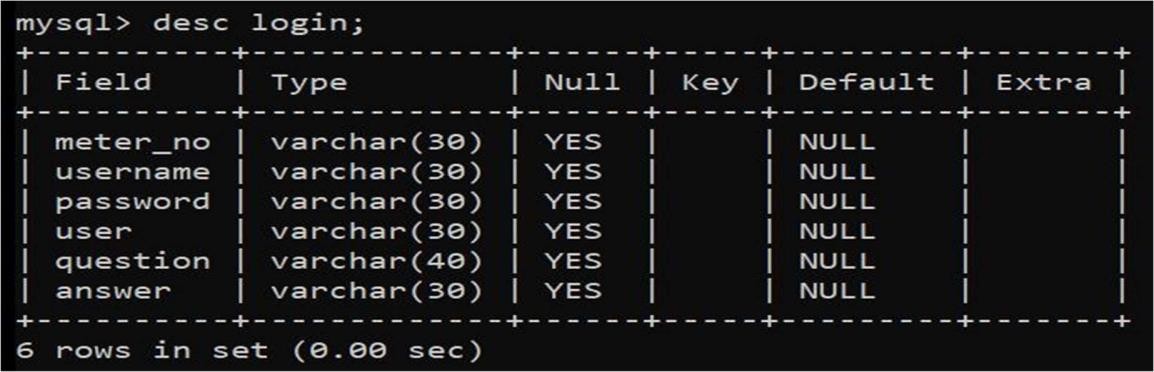
**FIG 6.1: LIST OF TABLES**

**LOGIN Table:**



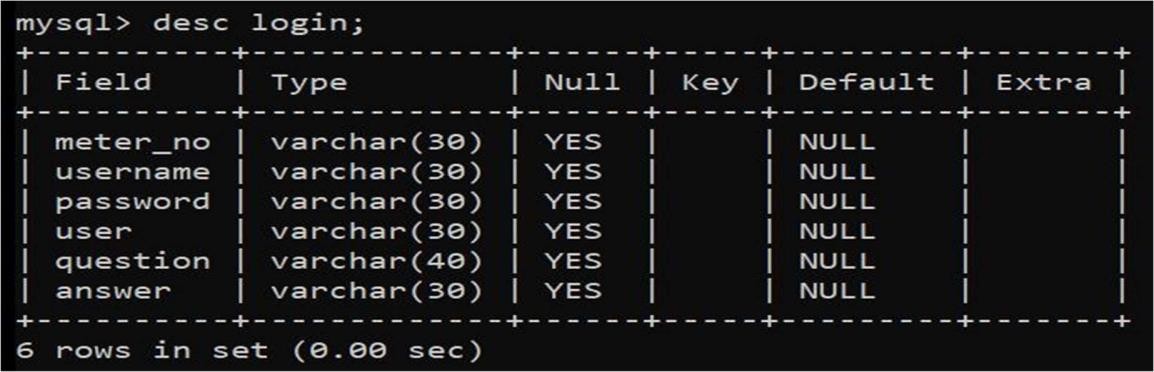
**FIG 6.2: Login table description**

**CUSTOMER Table:**



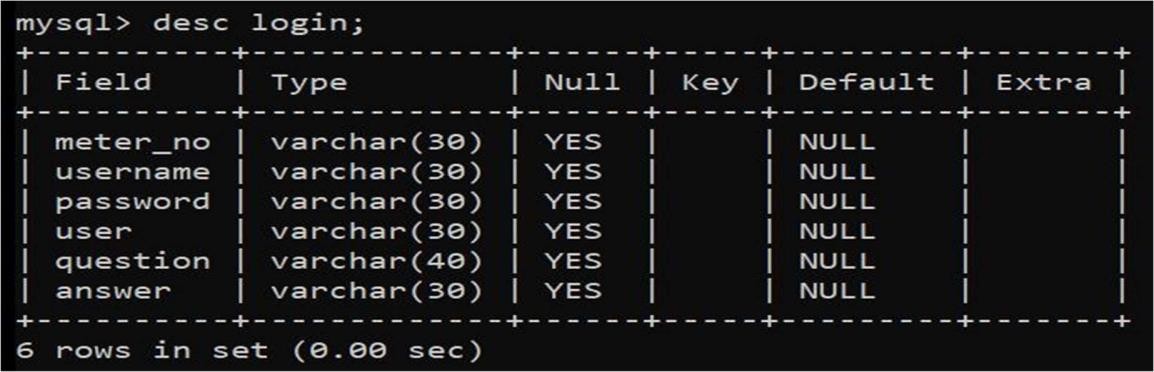
**FIG 6.3: Customer table description**

**TAX Table**



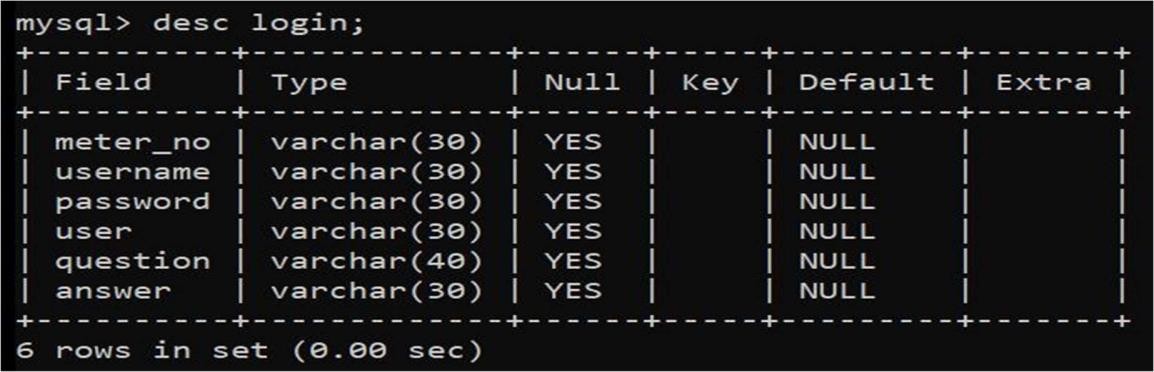
**FIG 6.4: TAX Table description**

**Rent Table:**



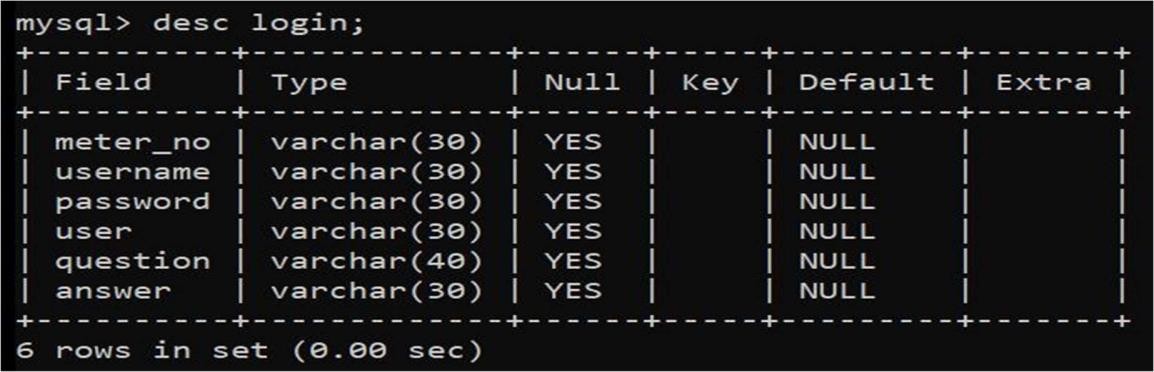
**FIG 6.5: Rent table description**

**Bill Table:**



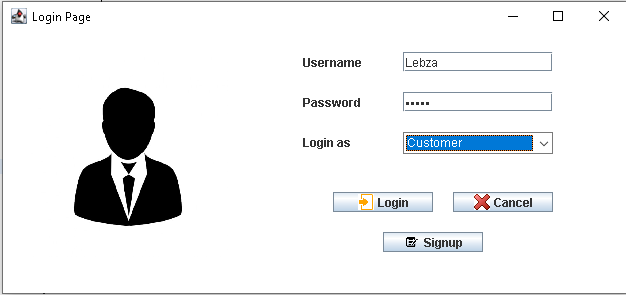
**FIG 6.6: Bill Table description**

**Meter\_Info Table:**



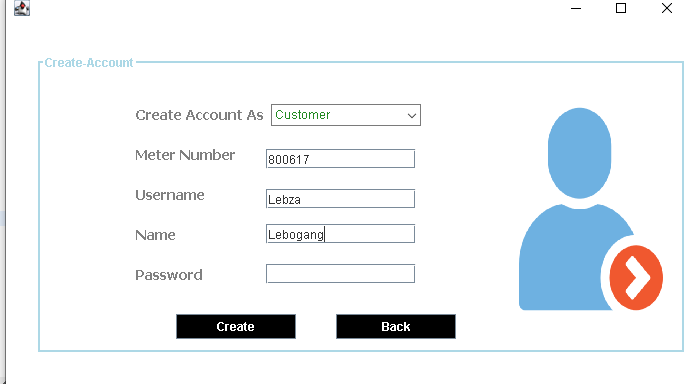
**FIG 6.7: Meter\_Info Table description**

**6.1 SNAPSHOTS**

****

**Login Screen**

**Here Customer and Admin can login to their respective accounts. The dropdown menu allows to choose whether to login as an admin or as a customer.**

****

**Sign Up Screen**

**Here New customers and admins will signup to access their accounts.**

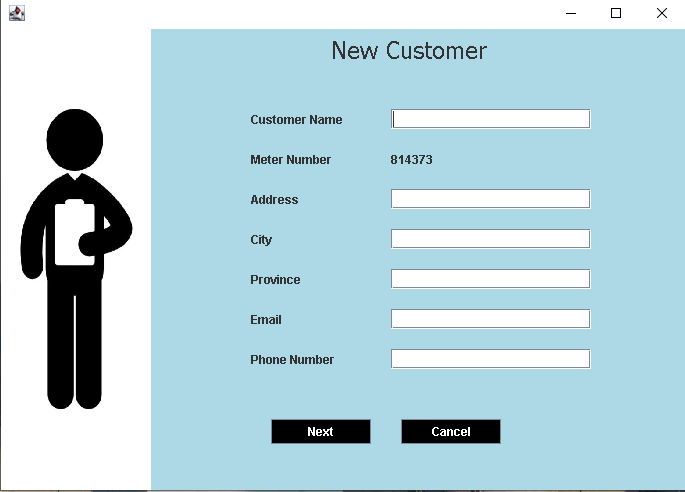
**User have to enter username, name, password, choose security question and answer to that question.**

**Every user must enter their unique Meter Number to complete their signup process.**



**Admin’s Home Screen**

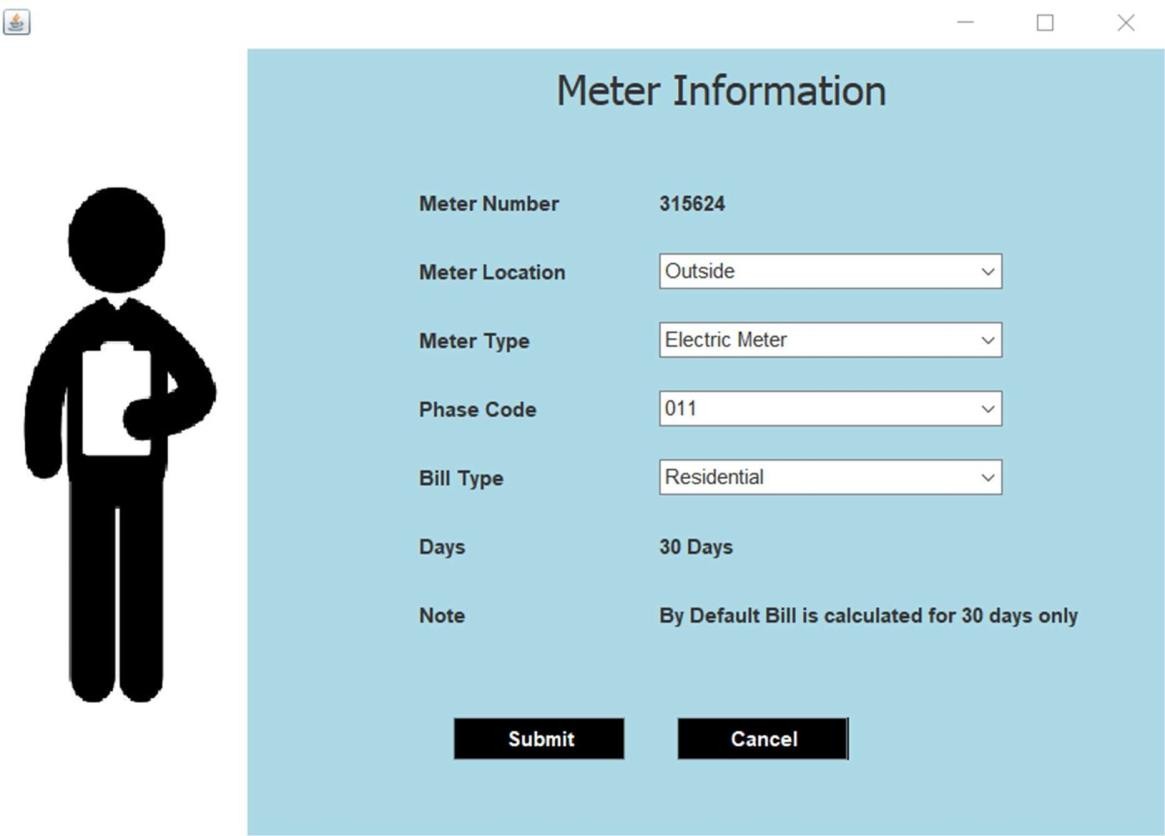
**Admin lands on this page after successful login.**

****

**New Customer Screen**

**Here admin registers new users.**

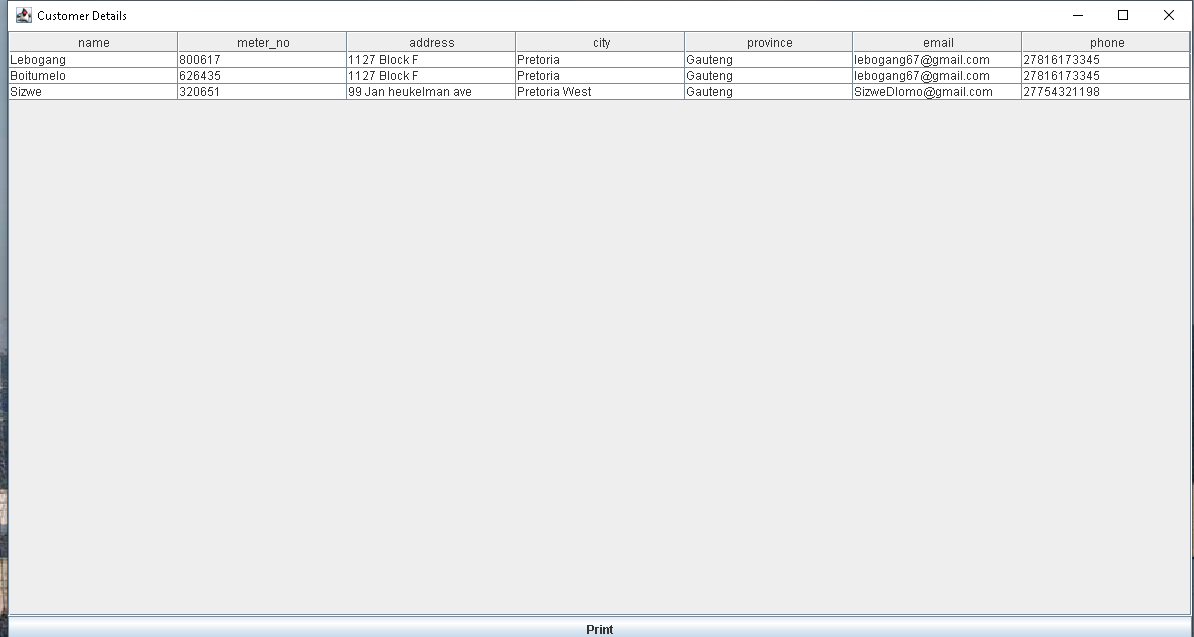
**Admin enters Customer’s Name, Address, City, Province, Email and Phone Number.**



**Meter Info Screen**

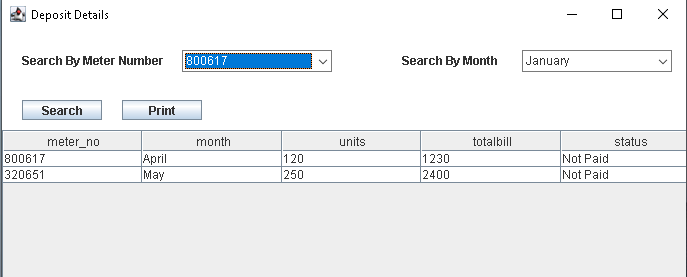
**Here Admin selects the location and type of meter installed at the customers end.**

**Admin also selects the phase code and Bill type i.e. Residential or Commercial/ Industrial.**

****

**Customer Details Screen**

**Here Admins can see the details of all registered customers. Admin can print these details in pdf format if the wish.**

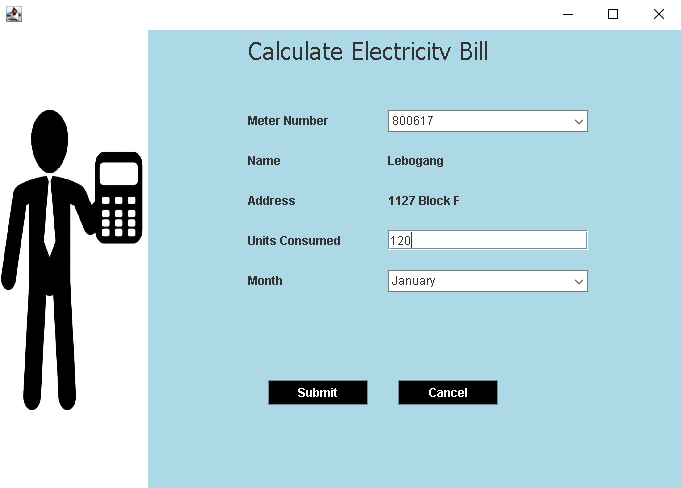
****

**Deposit Details Screen**

**Here Admin can check the status whether customers have paid their bills or not.**

**His list can be sorted according to individual user’s meter number or according to month.**

**Admin can print these details in pdf format if the wish.**

****

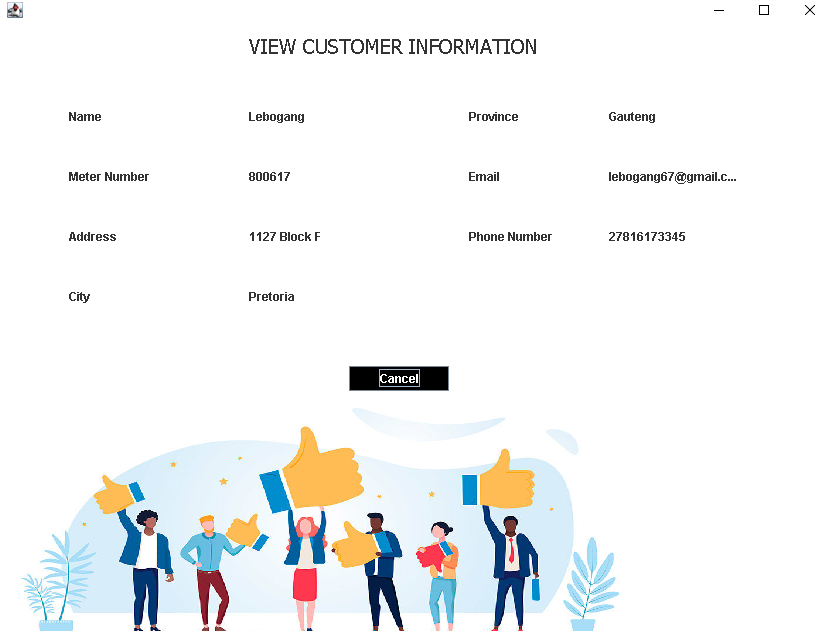
**Calculate Bill Screen**

**Here admin calculate the bill of users by selecting appropriate meter number, units consumed and month.**



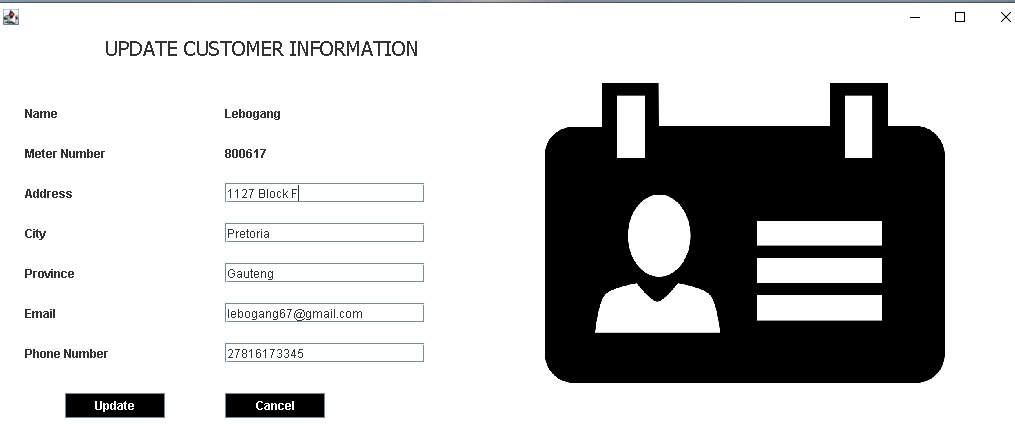
**Customer’s Home Screen**

**Customer lands on this page after successful login**

****

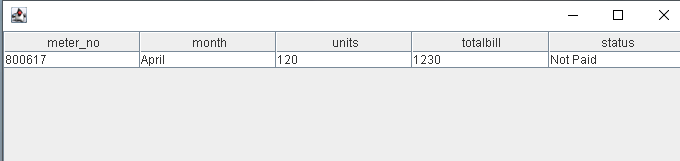
**View Customer Info Screen**

**Here customer can see their entered information such as their name, meter number, address, city , state, email id and phone number.**

****

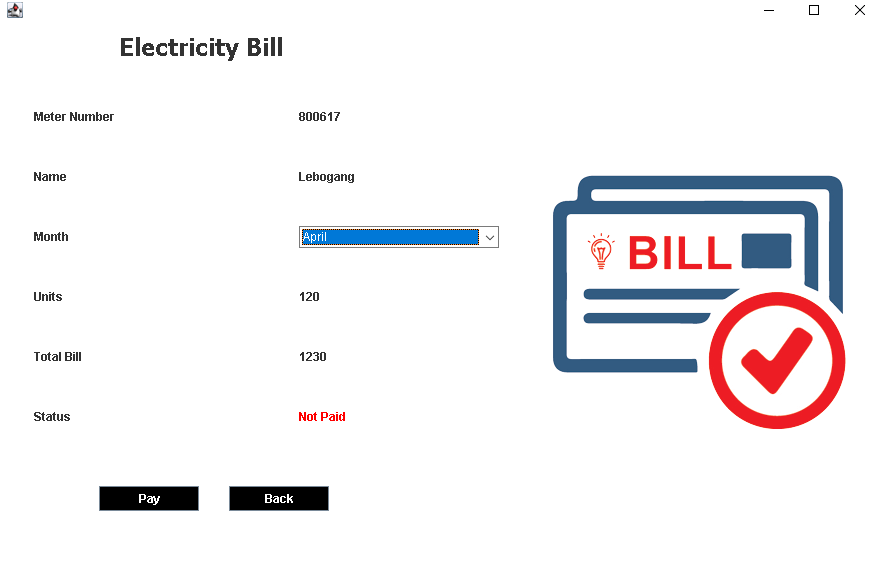
**Update Customer Info Screen**

**Here customer can update their entered information if any correction is needed such as their address, city, state, email id and phone number.**

****

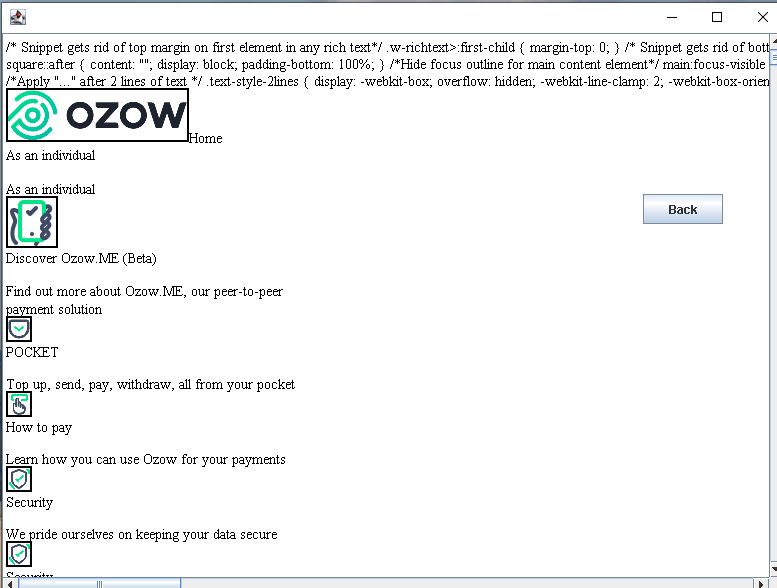
**Bill Details Screen for Customers**

**Here every customer can check the status of their bills, whether they have paid the bills or not.**

****

**Pay Bill Screen**

**Here customers pay their bills by selecting appropriate month.**

****

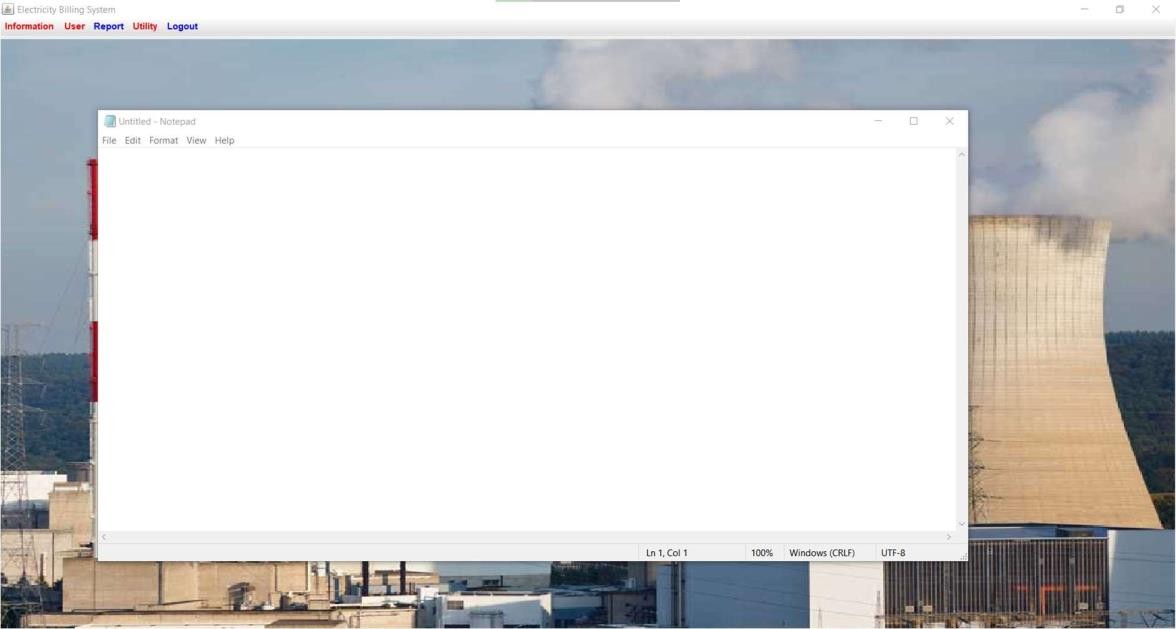
**Ozow Gateway Screen**

**Customers can pay via Paytm gateway.**

**Customers can pay via net banking, wallet, Debit or credit cards.**

**Generate/ Show Bill Screen**

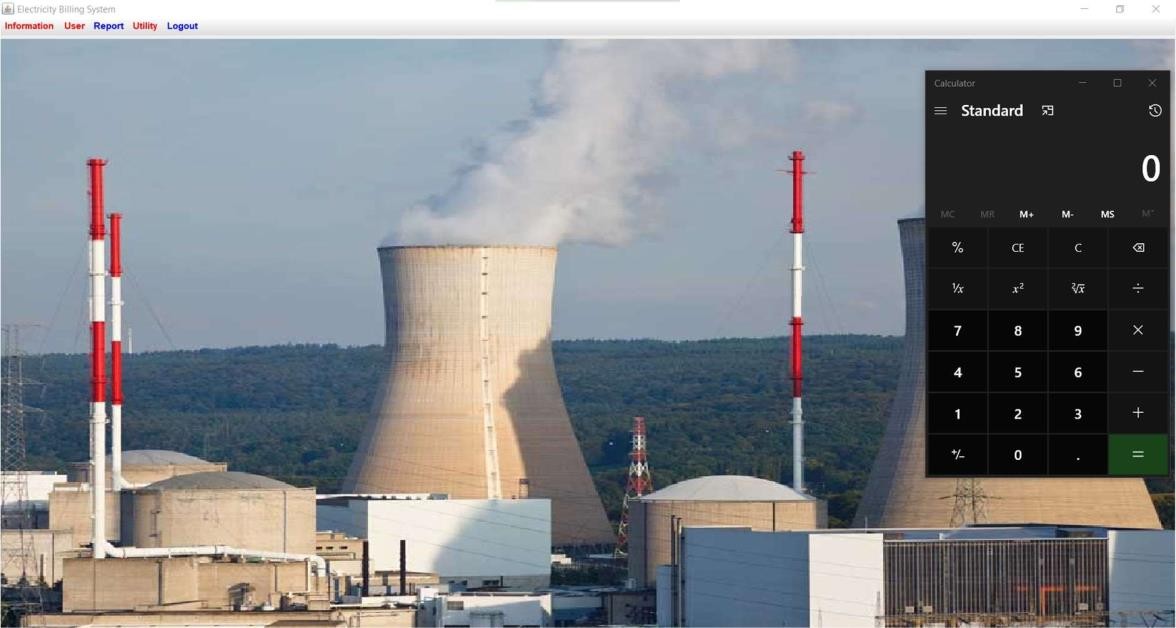
**Here customer can generate / see their bill in a proper breakdown of entire amount.**



**Notepad Screen**

**When user clicks on notepad option under utilities section, its launches the notepad.**

**This feature is available to both Admins and Customers.**



**Calculator Screen**

**When user clicks on calculator option under utilities section, its launches the calculator.**

**This feature is available to both Admins and Customers.**

**CHAPTER 7**

**FUTURE SCOPE AND LIMITATIONS**

**SOFTWARE SCOPE:**

* **Extensibility**: This software is extendable in ways that its original developers may not expect. The following principles enhances extensibility like hide data structure, avoid traversing multiple

Links or methods avoid case statements on object type and distinguish public and private operations.

* **Reusability**: Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort Over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability:

Sharing of newly written code within a project and reuse of previously written code on new projects.

* **Understand ability:** A method is understandable if someone other than the creator of the method can understand the code (as well as the creator after a time lapse). We use the method, which small and coherent helps to accomplish this.
* **Cost-effectiveness:** Its cost is under the budget and make within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement.

Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

##### LIMITATIONS:

**This application cannot be accessed remotely.**

* + **This application requires knowledgeable person to use this application.**
  + **This application does not have journals.**

**CHAPTER 8**

**CONCLUSION**

After all the hard work is done for electricity bill management system is here. It is a software which helps the user to work with the billing cycles, paying bills, managing different DETAILS under which are working etc.

This software reduces the amount of manual data entry and gives greater efficiency. The User Interface of it is very friendly and can be easily used by anyone.

It also decreases the amount of time taken to write details and other modules.

**CHAPTER 9**

**BIBLIOGRAPHY**

**REFERENCES**

**WEBSITES**

* <https://www.github.com>
* [www.stackoverflow.com](http://www.stackoverflow.com)
* [www.google.com](http://www.google.com)
* https://www.javatpoint.com