# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter:-1

## INTRODUCTION

Electricity Billing System is a software-based application. This project aims at serving the department of electricity by computerizing the billing system.
It mainly focuses on the calculation of units consumed during the specified time and the money to be charged by the electricity offices. 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.

### 1.1 Purpose :-

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.

### 1.2 Scope :-

The scope of our project are as follows:

- To keep the information of consuming unit energy of current month.
- To keep the information of Customer.
- 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.

## 1.3 Applicability:-

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.

# 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: -8 GB
- Storage: -512 GB
- Monitor: -LCD Monitor
- Keyboard: -Standard keyboard
- Mouse: -Compatible mouse

Software Requirements:-

- Operating System: -Windows 10
- Software: -Microsoft SQL Server
- Front End: -Java core/swings (NetBeans)
- Back End: -My SQL

# 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 :-

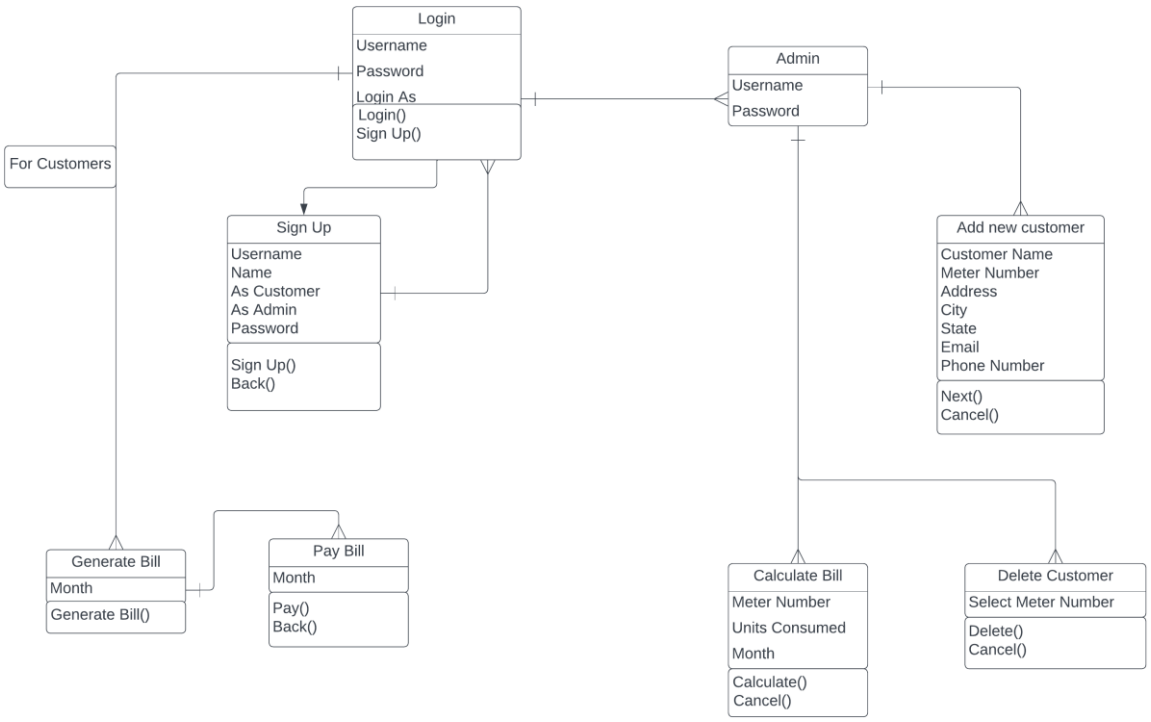



Fig. 3.1 Class Diagram

### 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 3.2 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 are6 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”, "swacch_bharat_cess”, "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 ".

**Schema Diagram:-**

Login

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

**Customer**

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

**Rent**

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

**Tax**

| Service Tax | Swacch bharat cess | GST |
|---|---|---|

**Bill**

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

**Meter Info**

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

Fig. 3.2 Schema diagram

## 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 first normal form:-

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

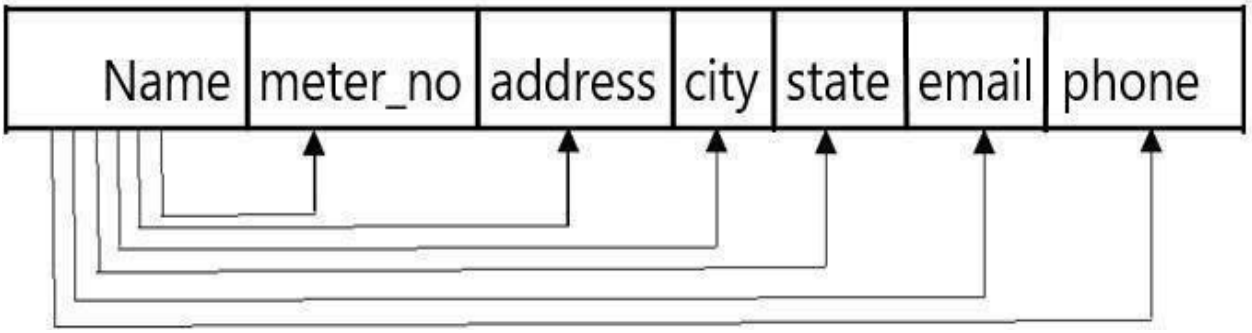



Fig. 3.3 Customer's info.

### 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.

### 3.2.3 Third 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

# IMPLIMENTATION

## 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.
Adding Tax: Here admin can add tax details.
Updating Customer: Here customer can update his/her details by using meter_no of the customer.
Delete Customer: Here admin can delete details based on meter number.

## 4.2 Implementation of SQL statements :-

**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: CREATETABLE table_name (column1datatype, column2datatype, column3 datatype, column datatype, 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),swacch_bharat_cess int (20), gst int (20));

- The following SQL statement creates a table “bill” table: create table bill (meter_no varchar (20), foreign key(meter_no) references customer(meter_no) on delete cascade, month varchar (20), units int (20), total_bill int (20), status varchar (40));

- 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));

**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 (“sai”,”12345”,” btm”,” Bangalore”, “Karnataka”, “sai@gmail.com”, “9876543333”);

**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 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 delete's a record in the “customer” table: delete from customer where meter_no=12345;

## 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 home page

**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.
If admin forgets password, it can be retrieved by giving username and answer for security question
After successful login the admin will be redirected to admin portal page where he/she can do 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 particulars 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. 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.

**Delete Customer:-**

- This Program will allow the admin to delete the customer info when meter_no is selected.
- If we need to delete the particulars that has been saved click onto delete option.
- If we need to cancel the particulars that has been entered click onto back 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.

**Update Info:-**

- This program will allow the customer to update the customer details. If customer address, city, state, email and phone number is updated.
- update the values into customer else print error
- update the details of customer 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 generatebill button.

NOTE: Utility (notepad, browser, calculator), query and logout is 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.

## 5.1 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.

## 5.2 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.

## 5.3 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.

### 5.3.1 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.

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

Table 5.1: Negative test case for phone number insertion

| Function Name | Input | Expected Output | Error | Resolved |
|---|---|---|---|---|
| Input Phone number | 9897778900 | Expected Output is Seen | _ | _ |

Table 5.2: Positive test case for phone number insertion

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

Table 5.3: Negative test case for email insertion

| Function Name | Input | Expected Output | Error | Resolved |
|---|---|---|---|---|
| Input email | aki123@gmail.com | Expected output is seen | _ | _ |

Table 5.4: Positive test case for email insertion

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

Table 5.5: Negative test case for customer name insertion

| Function Name | Input | Expected Output | Error | Resolved |
|---|---|---|---|---|
| Input customer name | Goku | Expected output is seen | _ | _ |

Table 5.6: Positive test case for customer name insertion

### 5.3.2 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.

| 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 | 12334(meter_no) January(month) | Must display full generated bill with total_bill | _ | _ |

Table 5.7: Test case on basis of generation of bill

| 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 | _ | _ |

Table 5.8: Test case on basis of deposit details

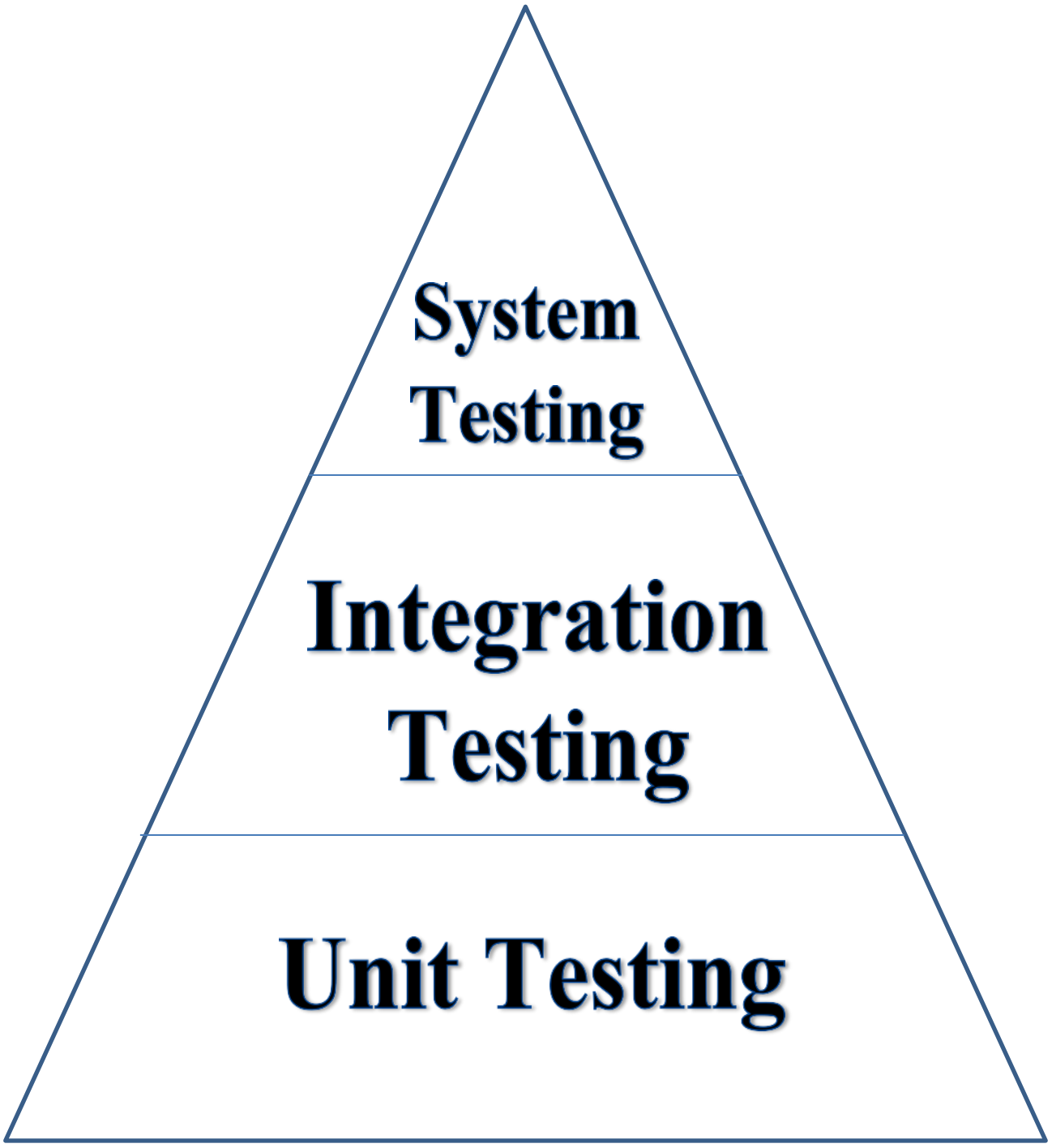



Fig. 5.1 Testing Diagram

**5.3.3 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.

# CHAPTER:-6

## DISCUSSION AND SNAPSHOTS

### 6.1 TABLES:

The given below tables are the snapshots of backend view of the localhost and the structures of the tables presents in Electricity Billing System.The tables presents login,cutomer,tax,bill,meter info. Etc.

- The login table is used to store the details of login's admin and customer with meter number.
- The customer table is used to store details of customer.
- The tax table is used to store tax values.
- The rent table is used to store rent values.
- The bill table is used to store details of bill of meter.
- The meter_info table is used to store information of meter placed in customer's home.

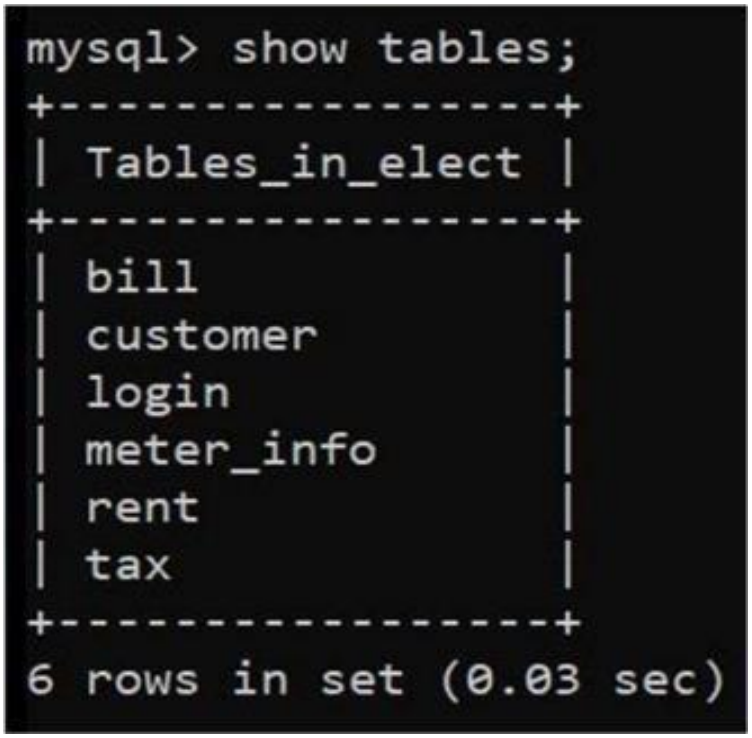



Table 6.1: List of tables

```
mysql> desc login;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| meter_no | varchar(30) | YES  |     | NULL    |       |
| username | varchar(30) | YES  |     | NULL    |       |
| password | varchar(30) | YES  |     | NULL    |       |
| user     | varchar(30) | YES  |     | NULL    |       |
| question | varchar(40) | YES  |     | NULL    |       |
| answer   | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Table 6.2: Login table

```
mysql> desc customer;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| name     | varchar(30) | YES  |     | NULL    |       |
| meter_no | varchar(20) | NO   | PRI | NULL    |       |
| address  | varchar(50) | YES  |     | NULL    |       |
| city     | varchar(20) | YES  |     | NULL    |       |
| state    | varchar(30) | YES  |     | NULL    |       |
| email    | varchar(30) | YES  |     | NULL    |       |
| phone    | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Table 6.3: Customer Table

```
mysql> desc tax;
+--------------------+------+------+-----+---------+-------+
| Field              | Type | Null | Key | Default | Extra |
+--------------------+------+------+-----+---------+-------+
| service_tax        | int  | NO   | PRI | NULL    |       |
| swacch_bharat_cess | int  | YES  |     | NULL    |       |
| gst                | int  | YES  |     | NULL    |       |
+--------------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Table 6.4: Tax Table

```
mysql> desc rent;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| cost_per_unit   | int  | NO   | PRI | NULL    |       |
| meter_rent      | int  | YES  |     | NULL    |       |
| service_charge  | int  | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Table 6.5: Rent Table

```
mysql> desc bill;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| meter_no   | varchar(20) | YES  | MUL | NULL    |       |
| month      | varchar(20) | YES  |     | NULL    |       |
| units      | int         | YES  |     | NULL    |       |
| total_bill | int         | YES  |     | NULL    |       |
| status     | varchar(40) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Table 6.6: Bill Table

```
mysql> desc meter_info;
+---------------+------------+------+-----+---------+-------+
| Field         | Type       | Null | Key | Default | Extra |
+---------------+------------+------+-----+---------+-------+
| meter_no      | varchar(30)| YES  | MUL | NULL    |       |
| meter_location| varchar(10)| YES  |     | NULL    |       |
| meter_type    | varchar(15)| YES  |     | NULL    |       |
| phase_code    | int        | YES  |     | NULL    |       |
| bill_type     | varchar(10)| YES  |     | NULL    |       |
| days          | int        | YES  |     | NULL    |       |
+---------------+------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Table 6.7: Meter_Info Table

## 6.2 SNAPSHOTS

**Sign Up Screen:-**

Here New customers 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.


Fig. 6.1 Sign Up Screen for Admin


Fig. 6.2 Sign Up Screen for Customer

**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.

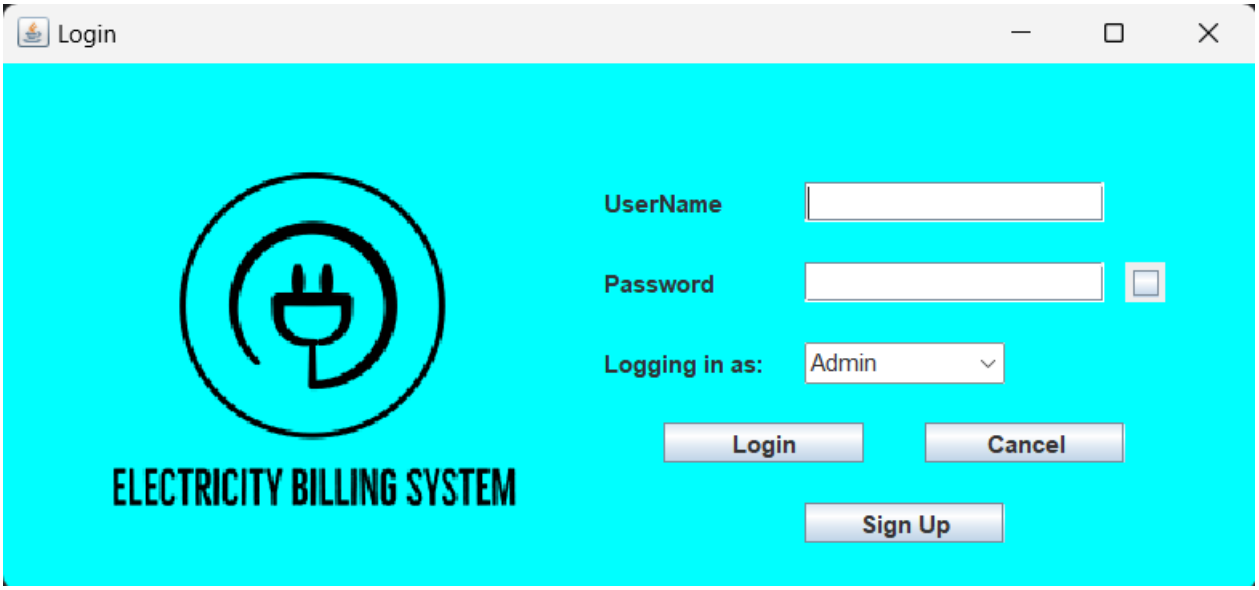



Fig. 6.3 Login Screen

**Admin's Home Screen:**

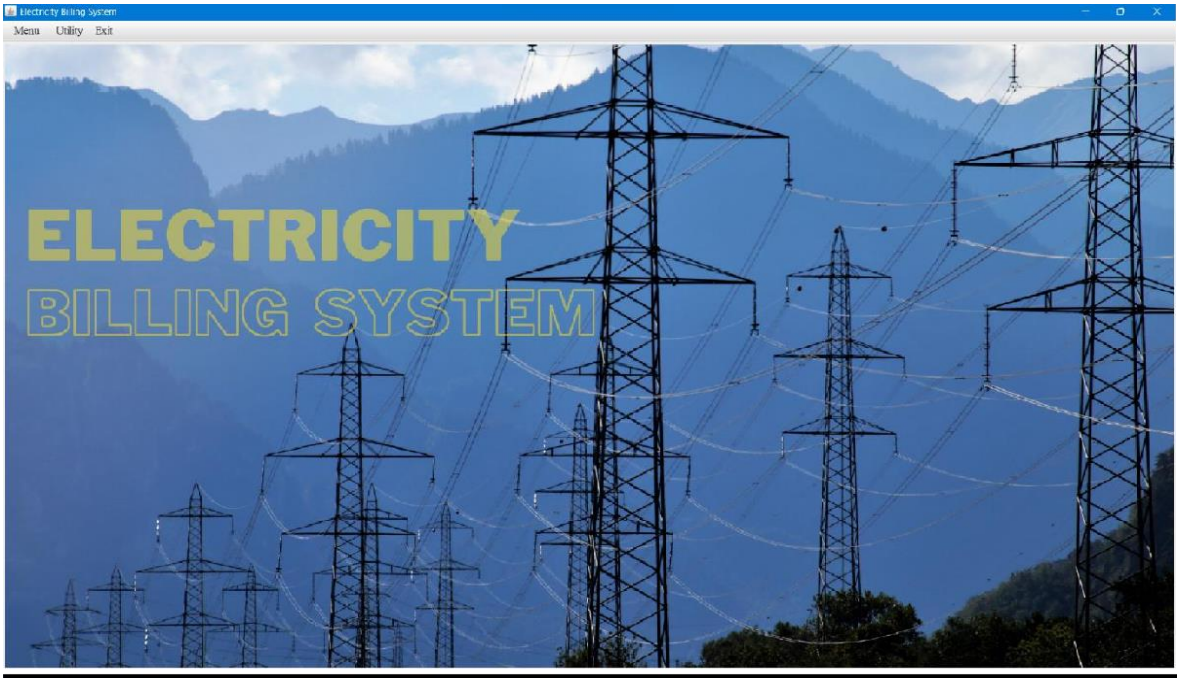



Fig. 6.4 Admin's Home Screen

Menu:

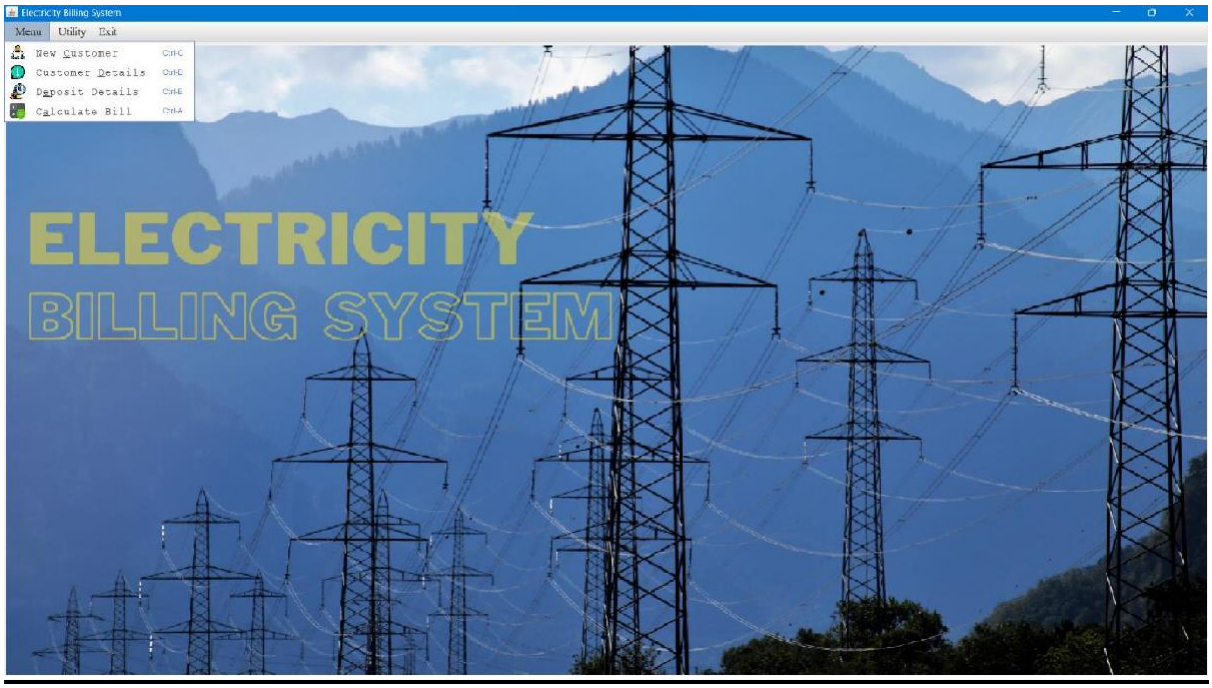



Fig. 6.5 Menu

**Utility:**

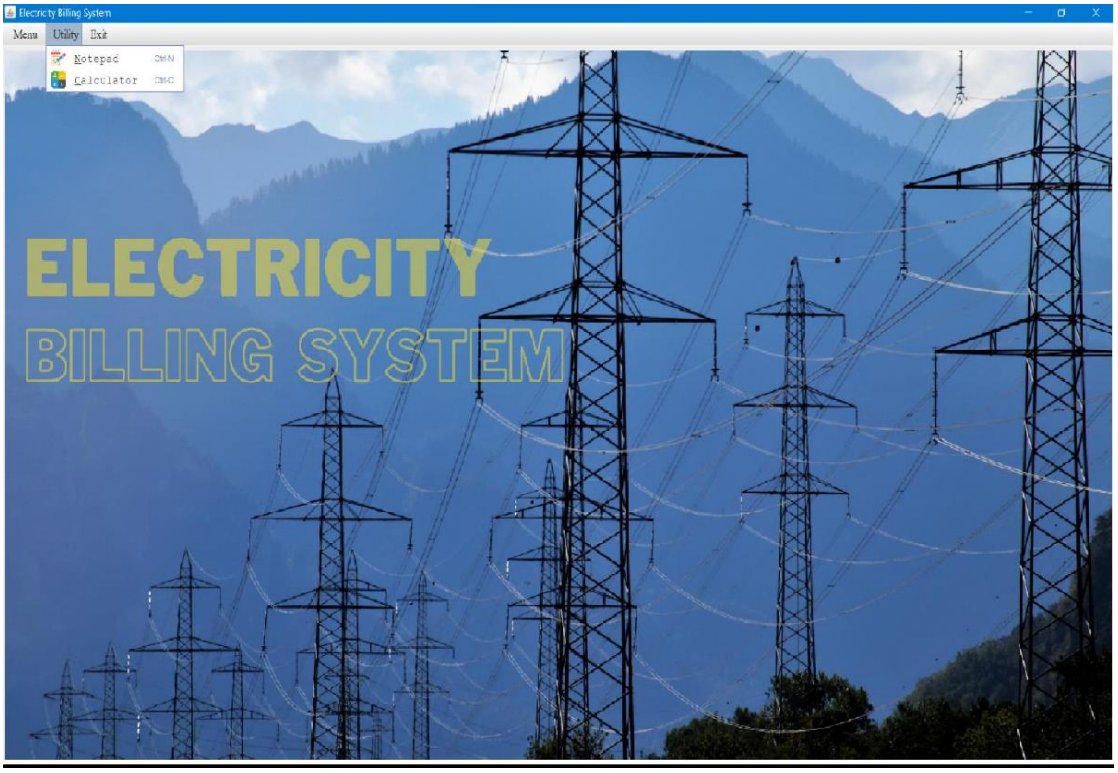



Fig. 6.6 Utility

**Exit:**


Fig. 6.7 Exit

After clicking exit user will reach to login screen and he/she can login again as a customer or admin.

**New Customer Screen:-**

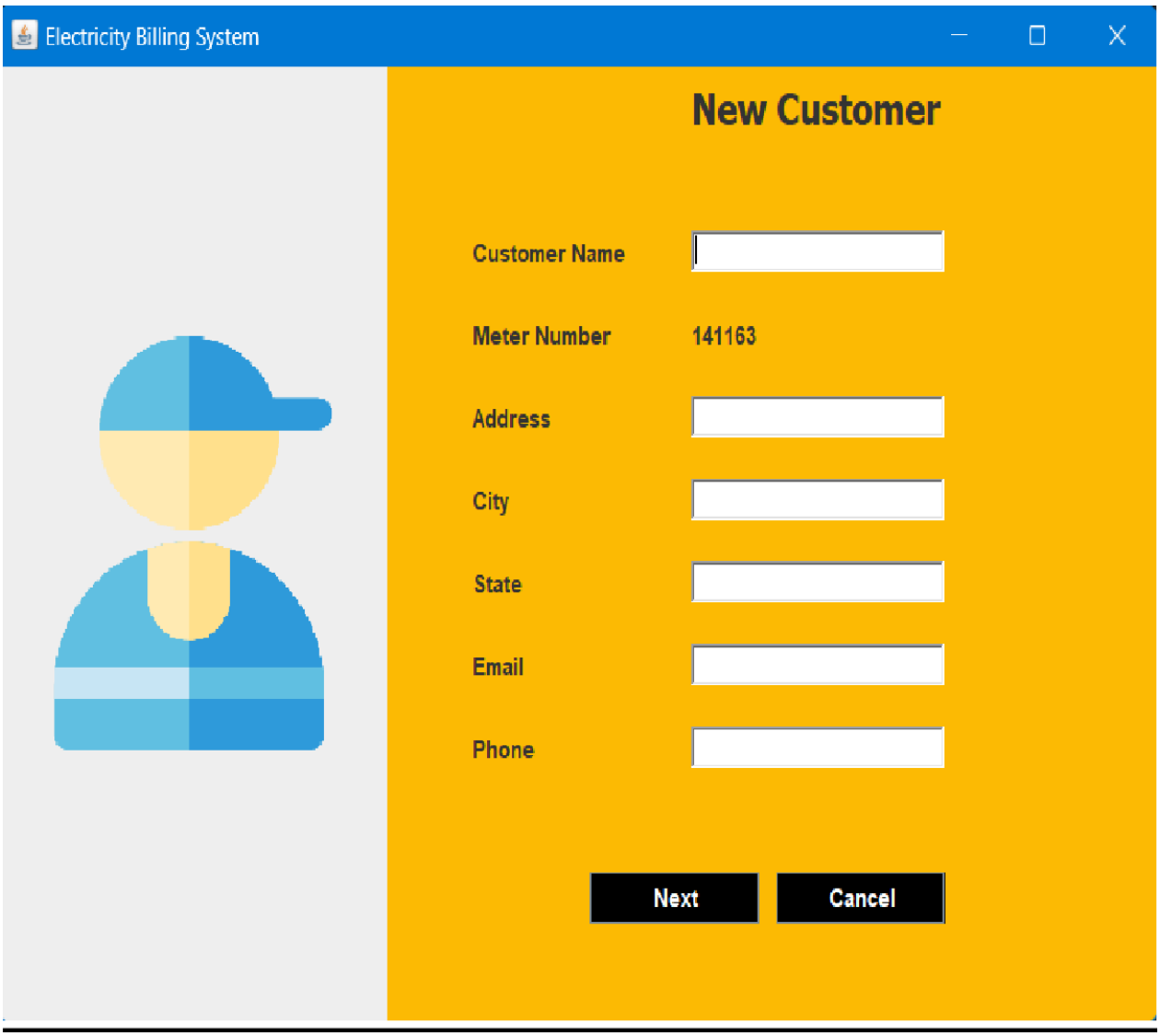



Fig. 6.8 Customer Screen

Here admin registers new users.

Admin enters Customer's Name, Address, City, State, Email and Phone Number.

**Meter Info Screen:-**

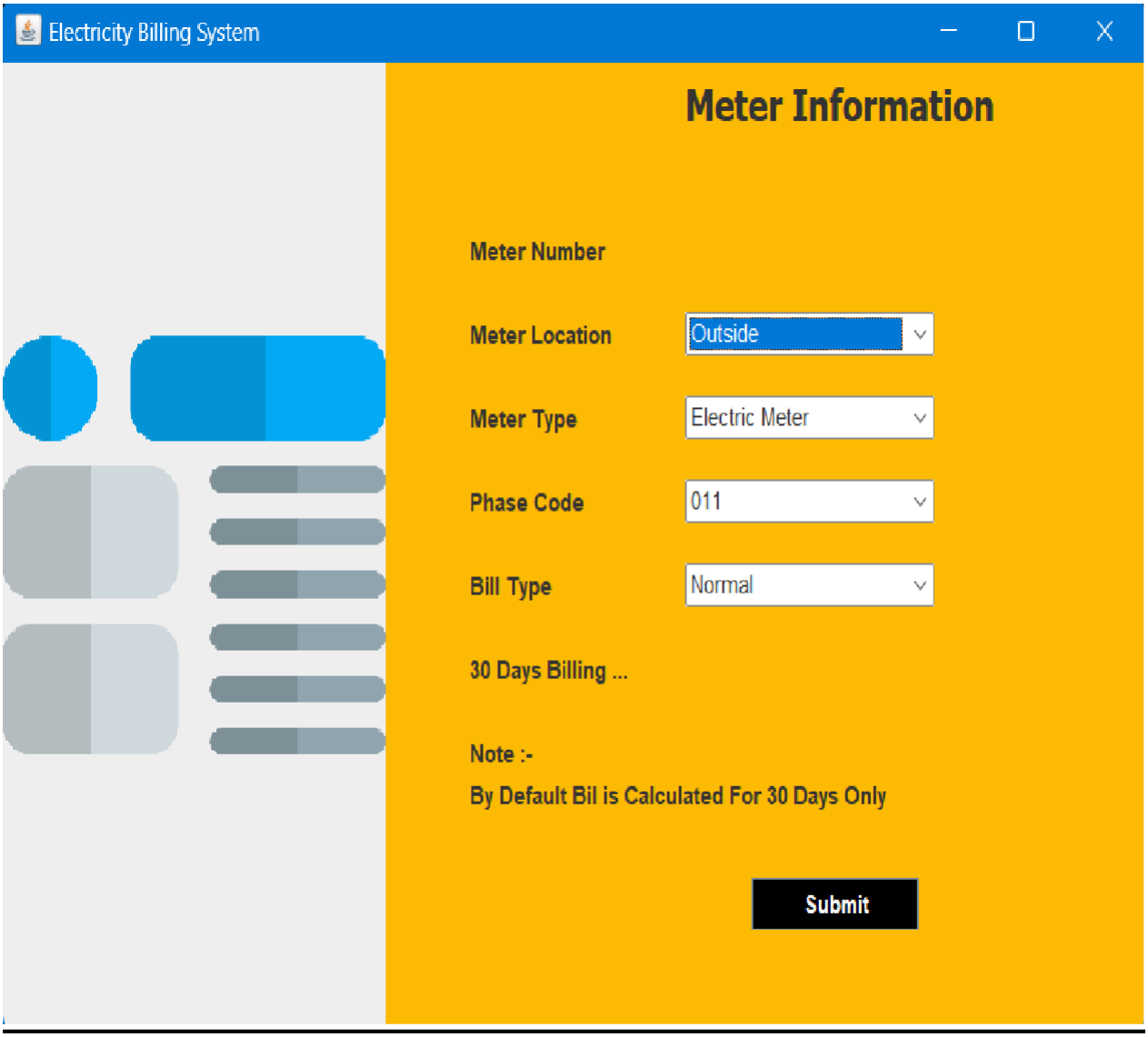



Fig. 6.9 Meter Info. Screen

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:-**

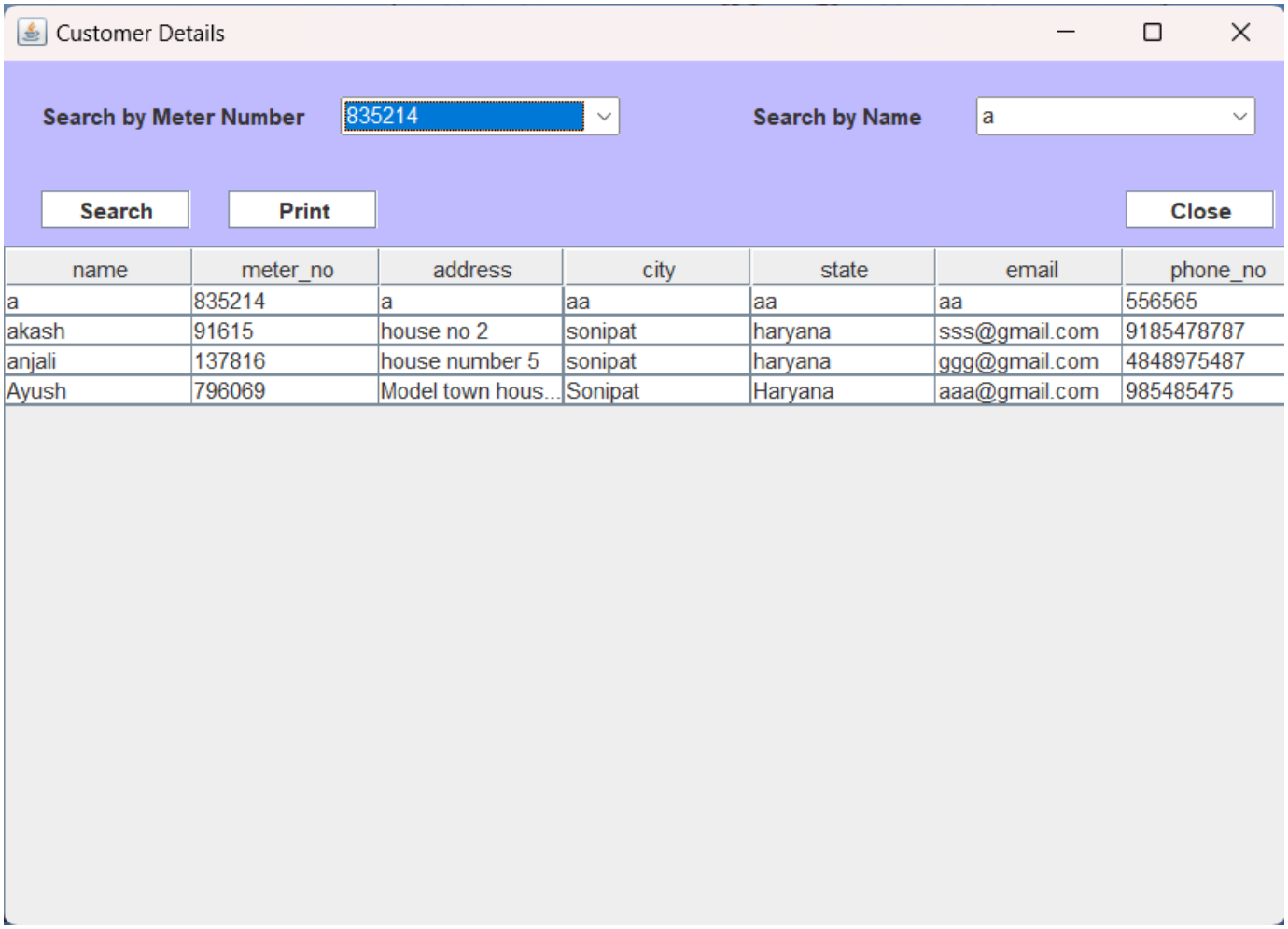



Fig. 6.10 Customer Details Screen

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

**Deposit Details Screen:-**

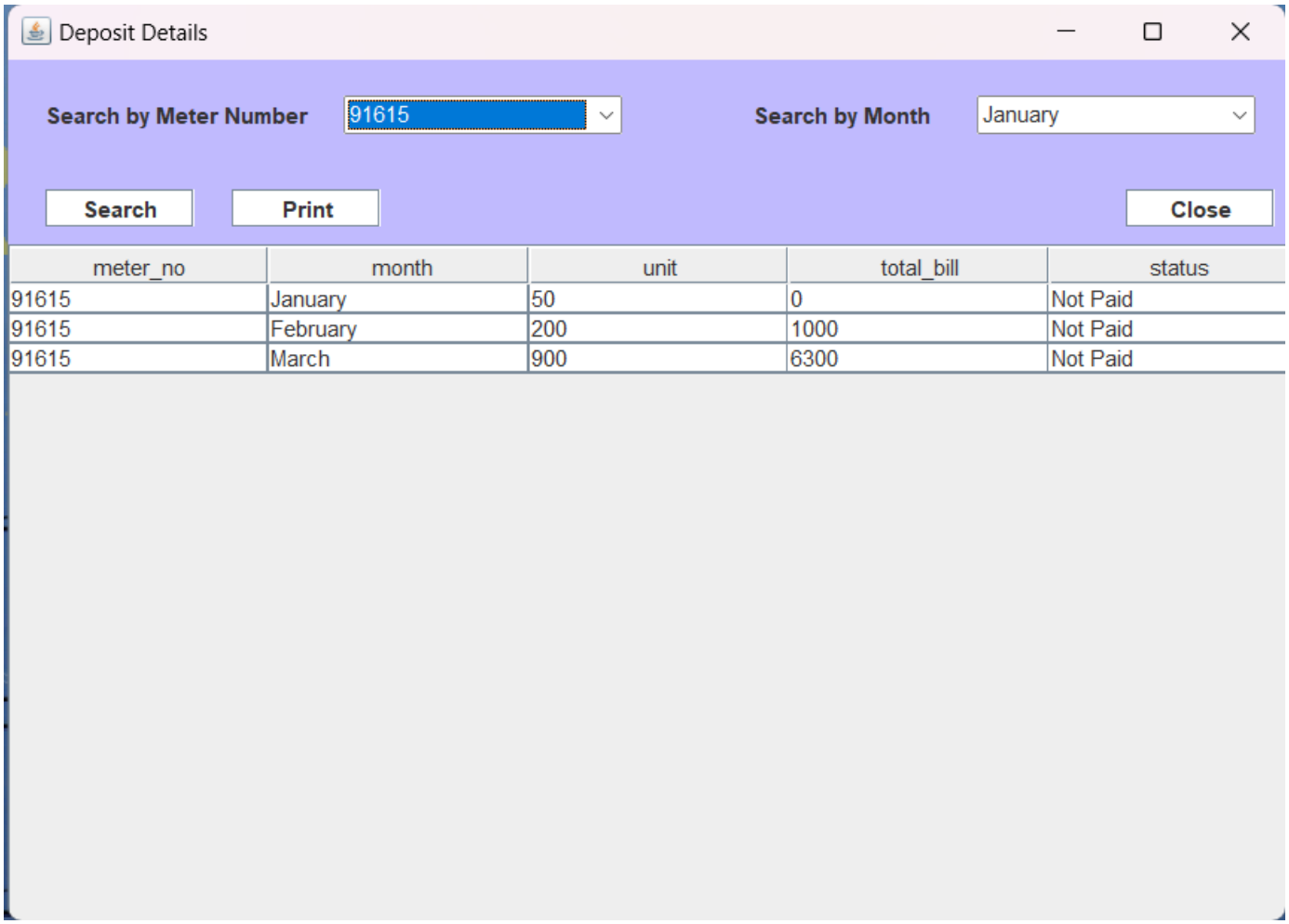



Fig. 6.11 Deposit Detail Screen

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:-**

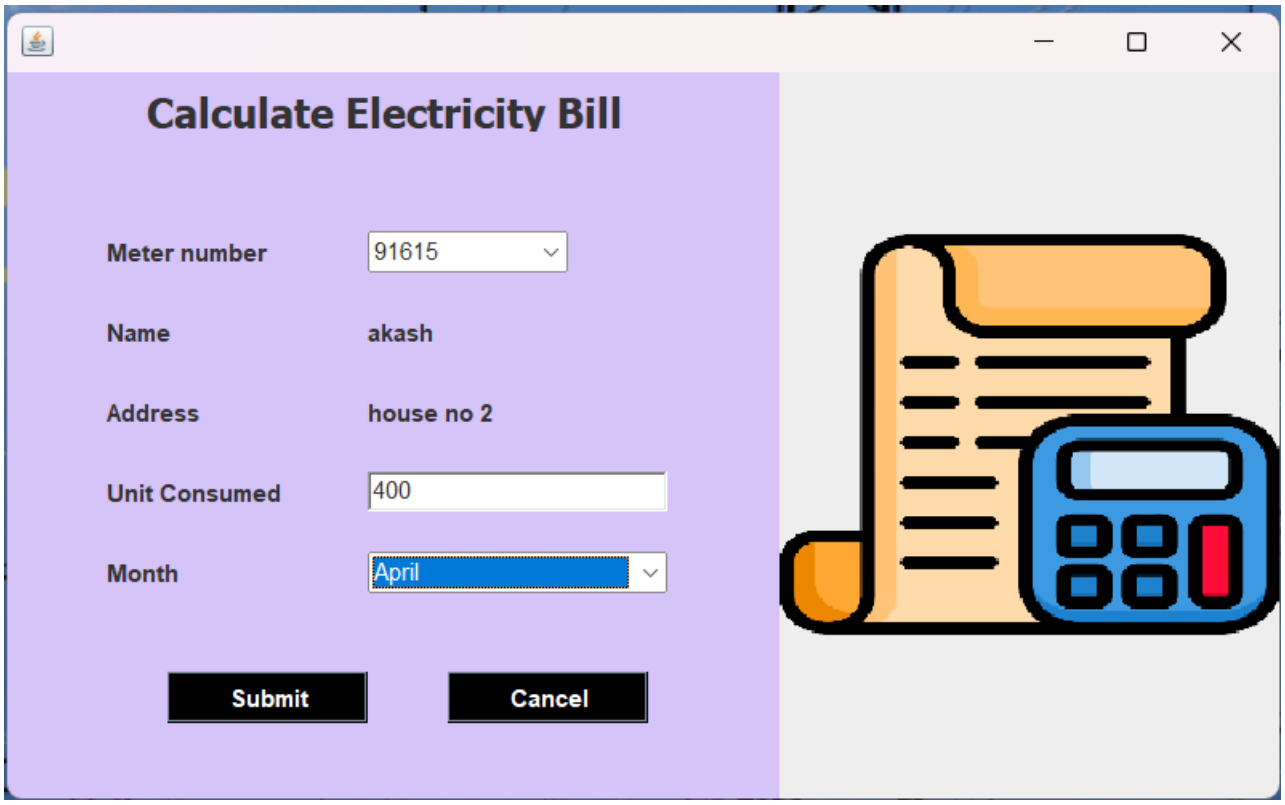



Fig. 6.12 Calculating Bill Screen

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

**Customer's Home Screen:-**

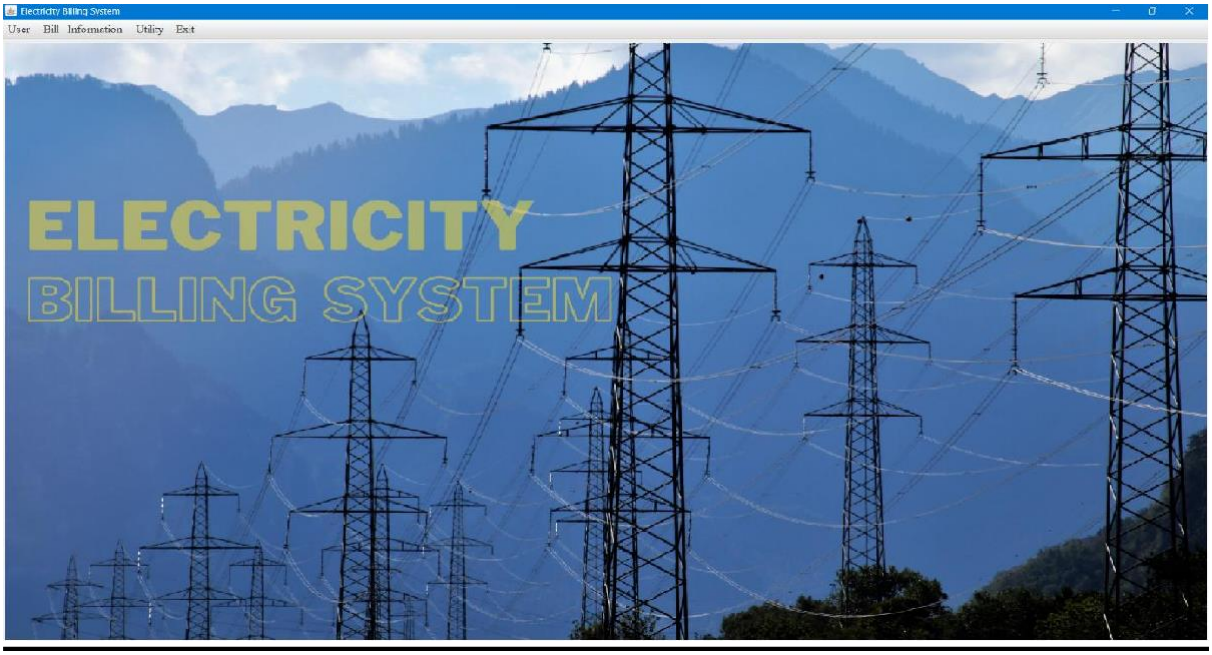
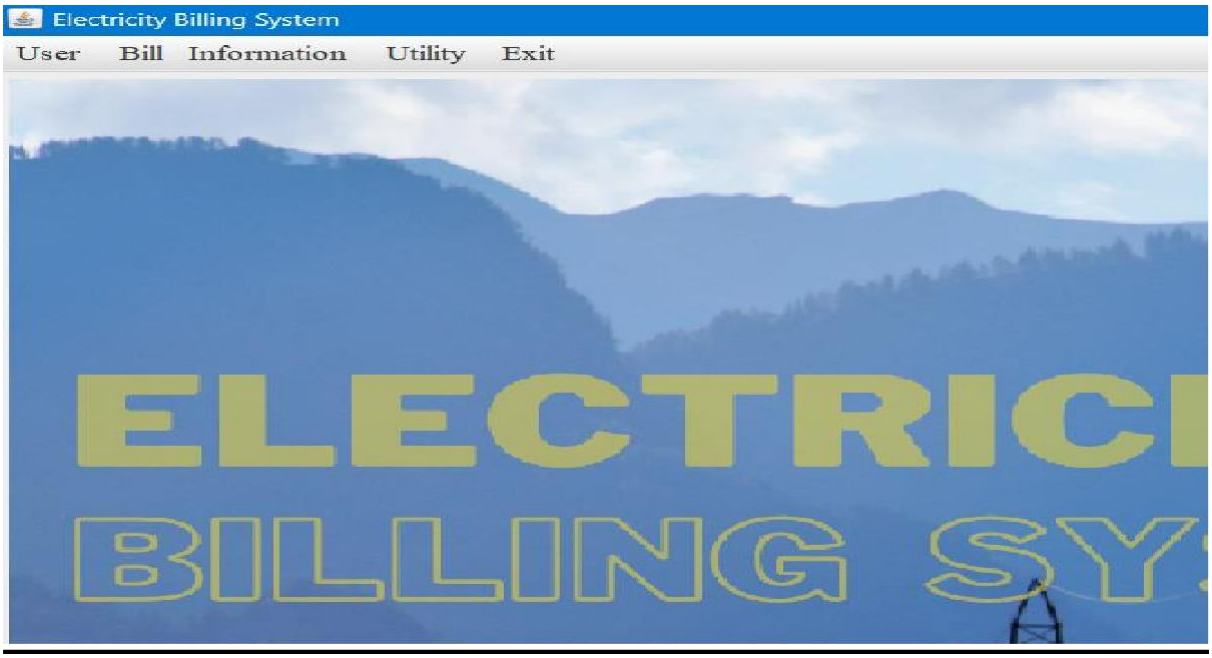



Fig. 6.13 Customer's Home Screen




Customer's after Login Screen (Fig. 6.14)

Customer lands on this (Fig. 6.14) page after successful login.

**View Customer Info Screen:-**

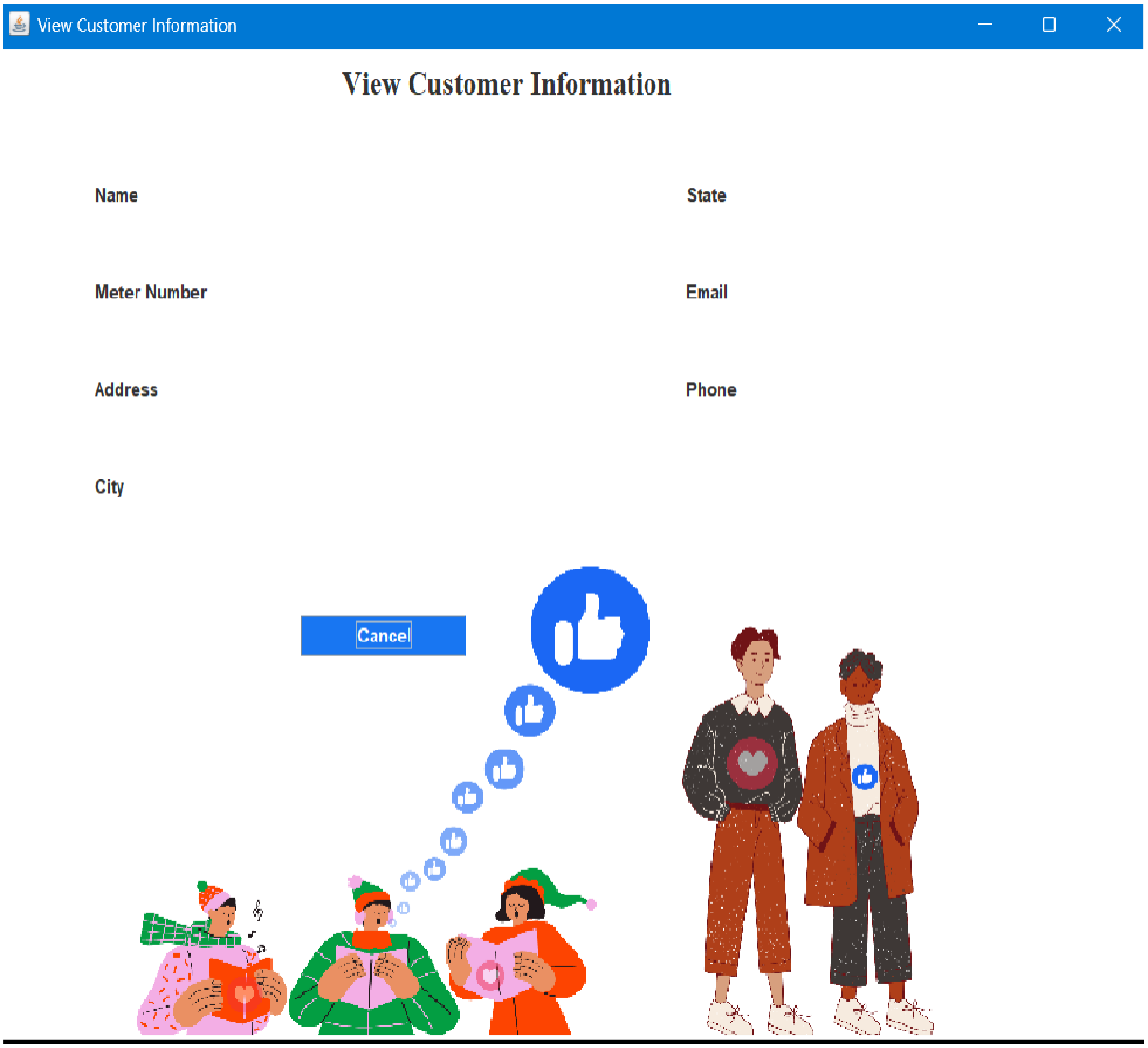



Fig. 6.15 Customer's Info. Screen

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:-**

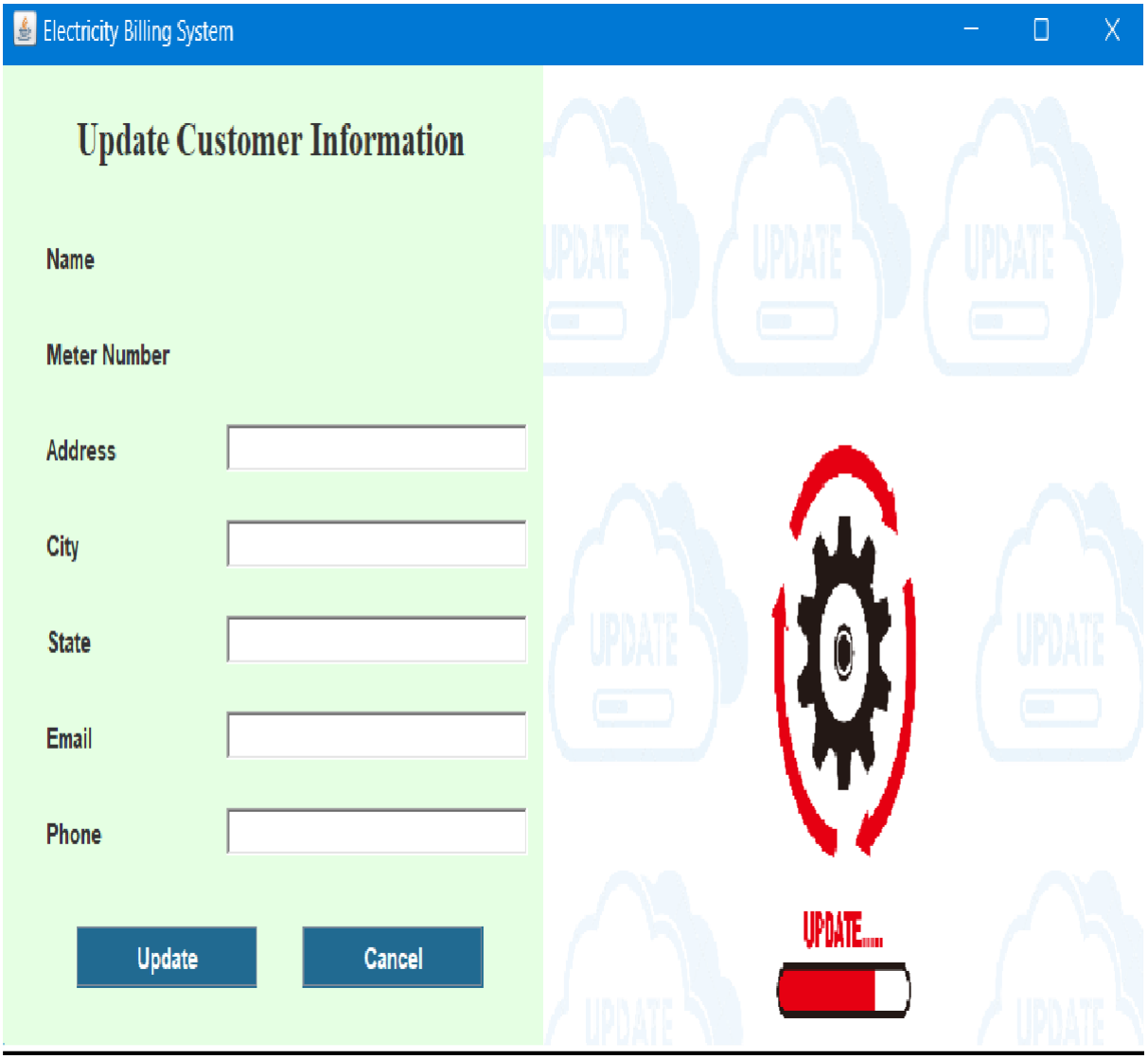



Fig. 6.16 Updating Info. Screen

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:-**

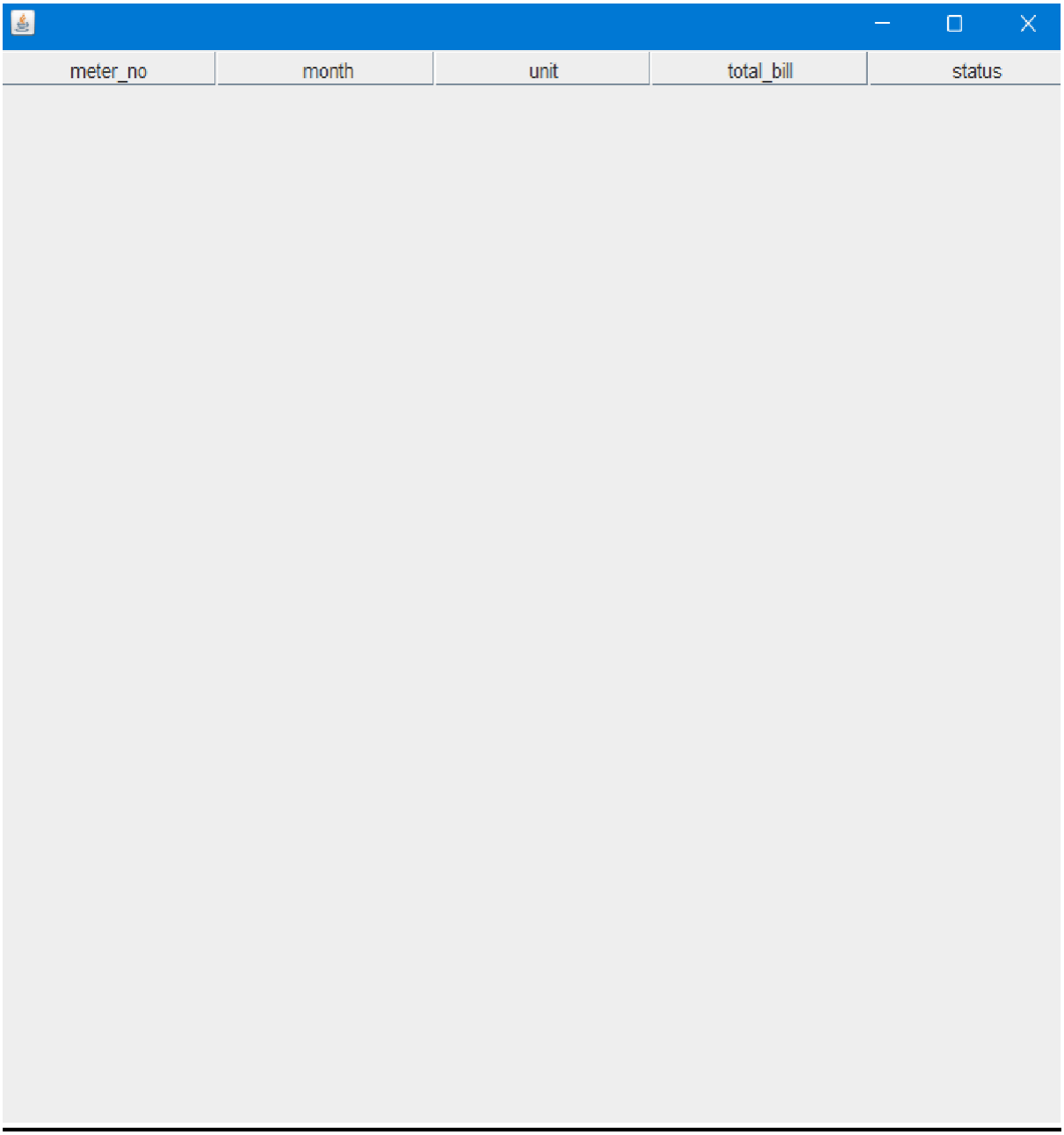



Fig. 6.17 Bill Detail Screen

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

**Pay Bill Screen:-**

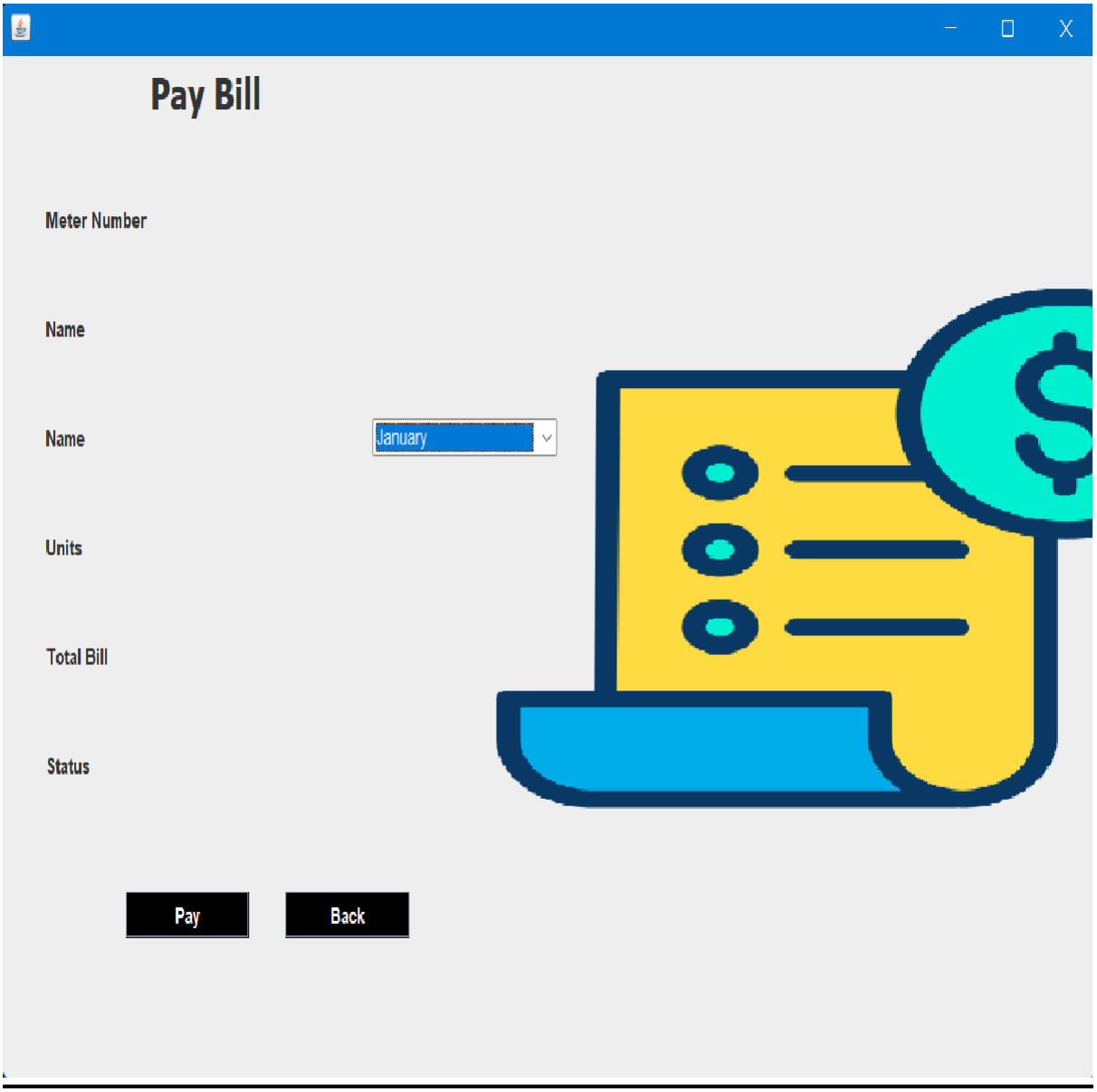



Fig. 6.18 Pay Bill Screen

Here customers pay their bills by selecting appropriate month.

**Generate/ Show Bill Screen:-**

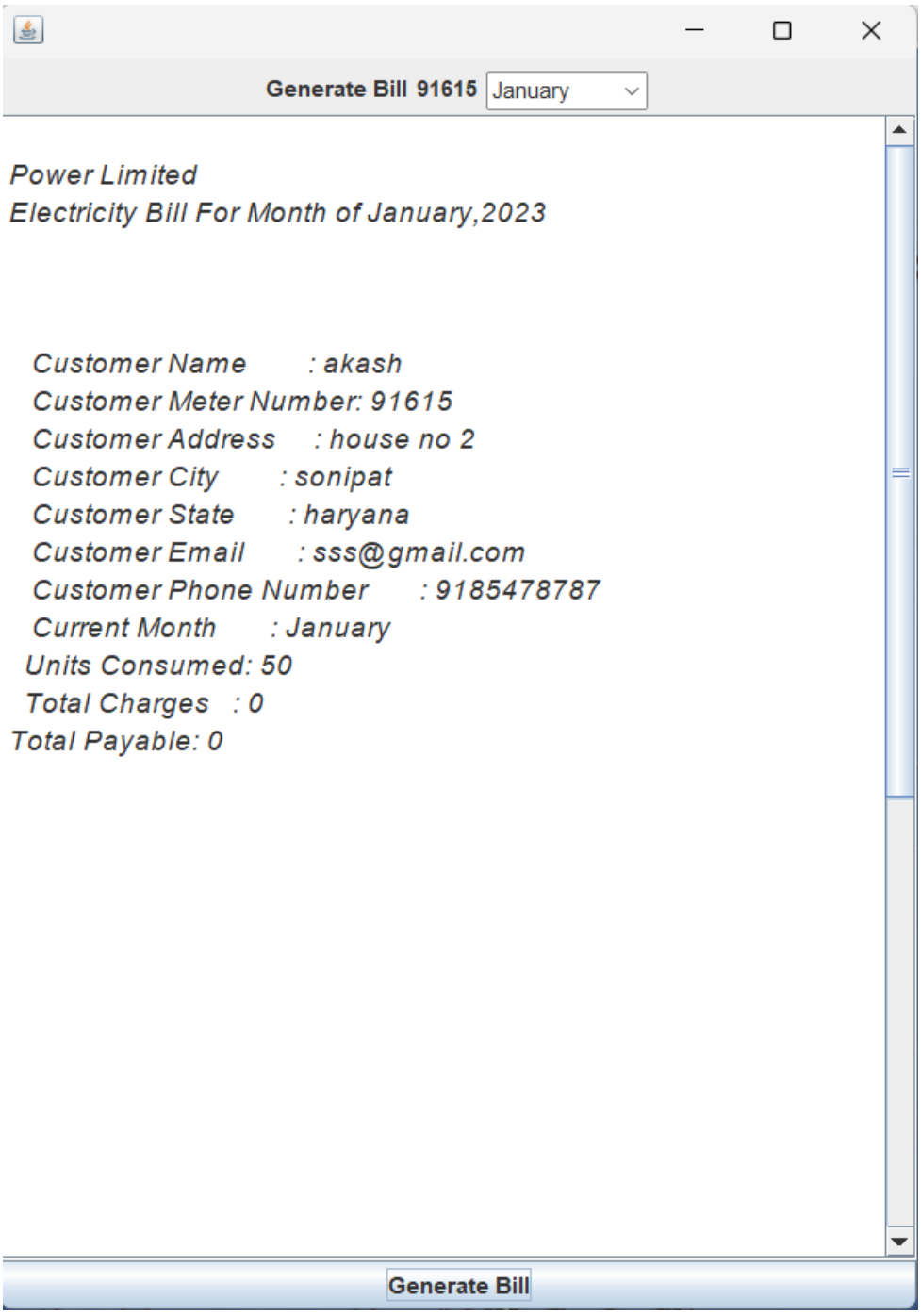



Fig. 6.19 Show Bill Screen

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

**Notepad Screen:-**

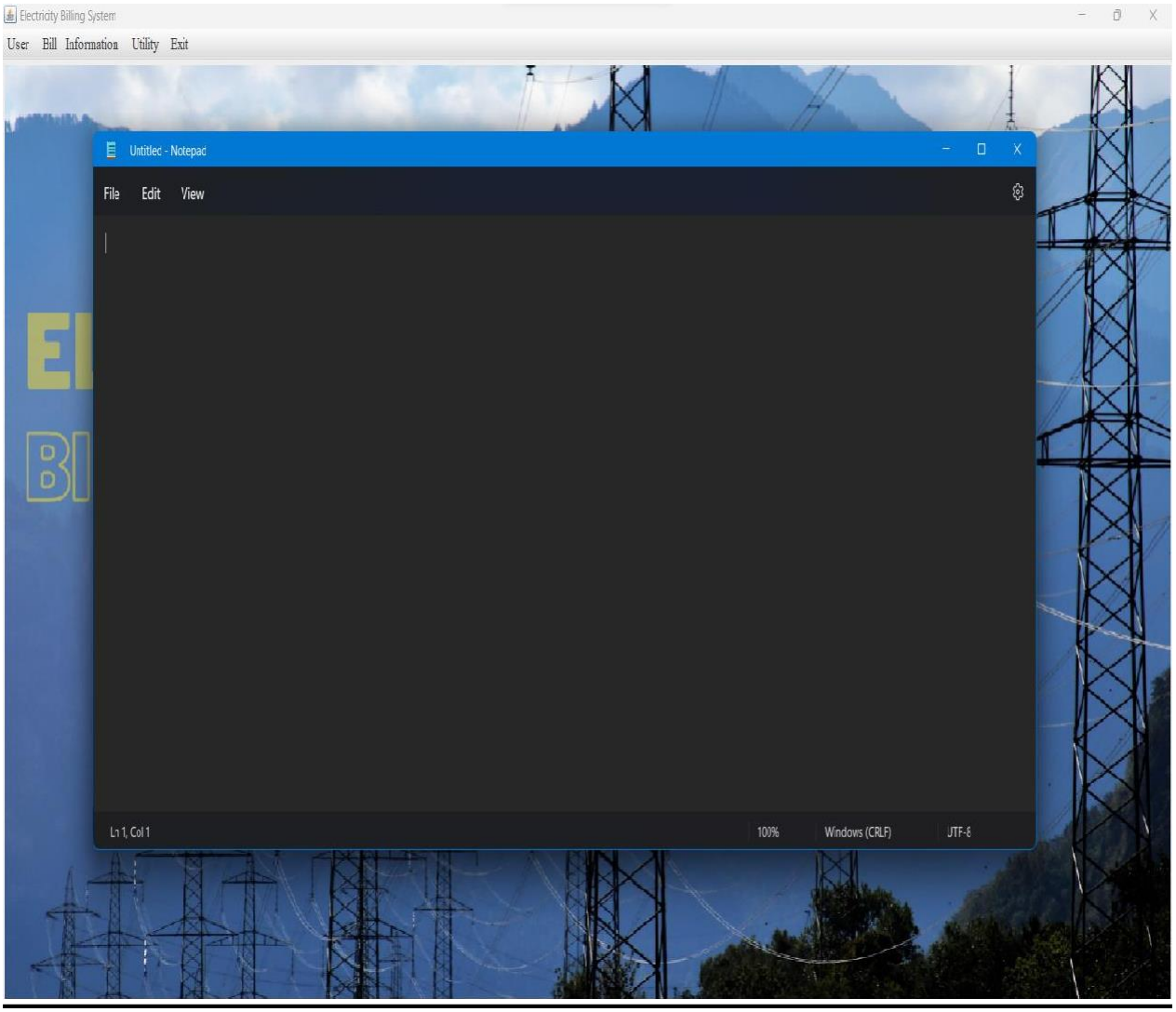



Fig. 6.20 Notepad Screen

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

This feature is available to both Admins and Customers.

**Calculator Screen:-**

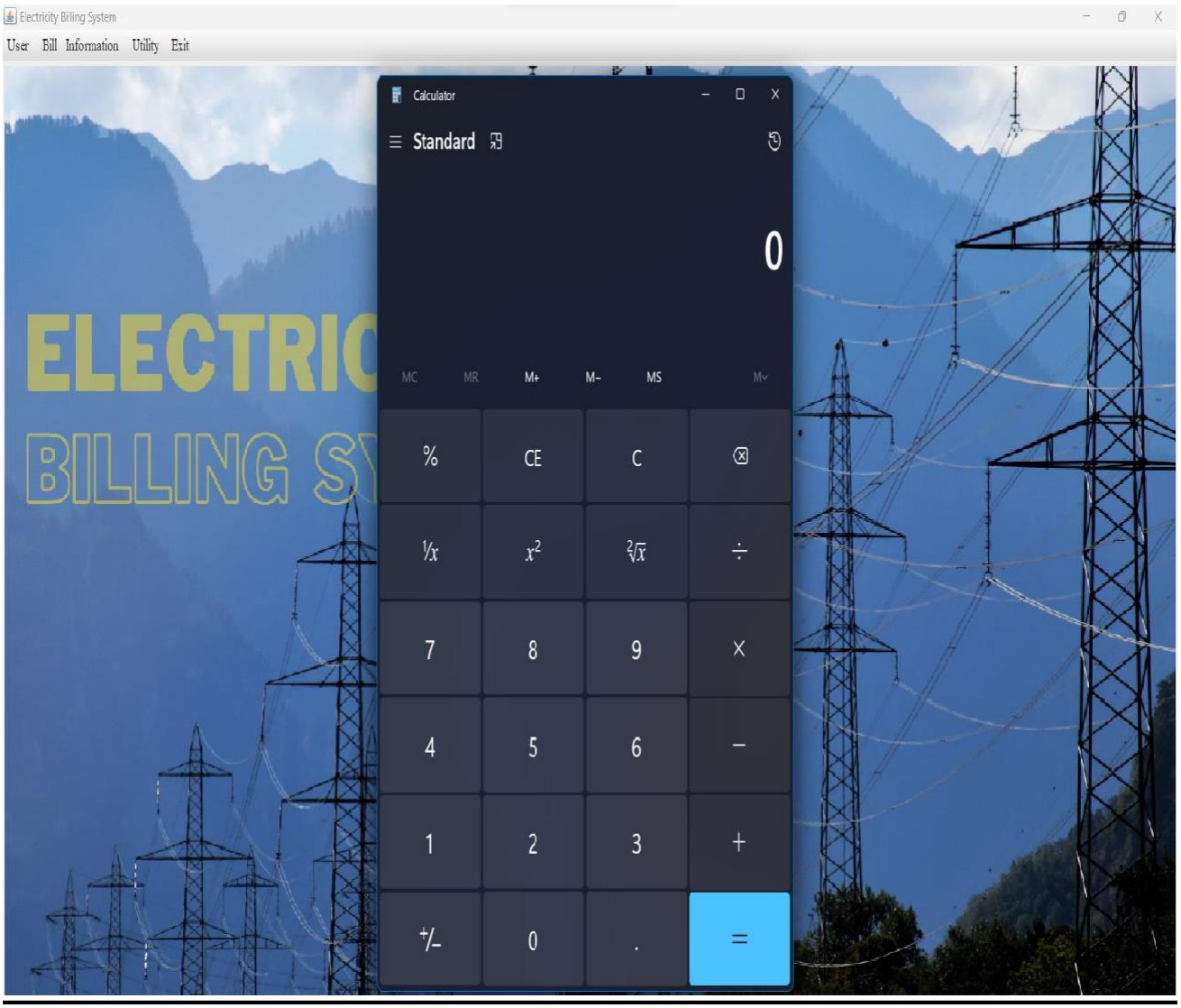



Fig. 6.21 Calculator Screen

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:-

- **Future Enhancements**: The software can be further enhanced to include features like predictive analysis for power consumption, integration with smart meters for real-time consumption tracking, and providing personalized energy-saving tips to consumers.
- **Scalability:** Given that Java is platform-independent, the software can be deployed across various platforms, making it highly scalable. It can handle the billing process for a small locality as well as for large cities.
- **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. Sharing of newly written code within a project and reuse of previously written code on new projects.
- **Understandability:** 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 is 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:

- **Storage Requirements:** It requires storage for installation and functioning, which might be a constraint for systems with limited storage.
- **Security Restrictions:** There might be security restrictions that could limit certain functionalities.
- **Offline Customers:** Reaching offline customers who do not access the internet can make the process difficult

# CHAPTER:-8

## CONCLUSION

In conclusion, the Electricity Billing System in Java is a software-based application that aims to serve the department of electricity by computerizing the billing system. It focuses on the calculation of units consumed during the specified time and the money to be paid to electricity offices. This computerized system makes the overall billing system easy, accessible, comfortable, and effective for consumers.

The electricity billing software calculates the units consumed by the customer and makes bills. It requires small storage for installation and functioning. There is a provision for debugging if any problem is encountered in the system. This project would replace the existing traditional and analog type of electricity billing system ensuring security, ease, and comfort in billing.

The system excludes the need for maintaining paper electricity bills, and users can pay the amount without visiting the office. It saves human efforts and resources. The software provides the facility of data sharing and 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 the customer can view all details.

The major goal of this project is to please customers by saving them time via the payment process, maintaining records, and allowing them to see and amend their information. Thus, it is a step towards modernizing the electricity billing procedure. However, it’s important to note that the effectiveness of this system can vary depending on the specific implementation and use case. Future enhancements can focus on addressing any limitations and improving user experience.