ELECTRICITY BILLING SYSTEM

Bachelor of Technology
in
Computer Science & Engineering

By

P. HARSHA VARDHAN REDDY	2103A52180
E. SWAPNA	2103A52044
V.SHIVANI	2103A52188
M.NAGA VIVEK	2103A52056

Under the guidance of

Mr. SRAVAN

Assistant Professor , Department of CSE

Submitted to

(Assistant Professor Dept of CSE & AIML)

SR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the **Object Oriented Programming through Java - Course Project** Report entitled **"ELECTRICITY BILLING SYSTEM"** is a record of bonafide work carried out by the student P .HarshaVardhanReddy,E.Swapna,V.shivani,M.Naga vivek, bearing Roll No(s) 2103A52180, 2103A52044, 2103A52188, 2103A52056, during the academic year 2019-20 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the
S R Technological University,Warangal.

Lab In-charge **Head of the Department**

ACKNOWLEDGEMENT

We express our thanks to Course co-coordinator **Mr. Sravan, Asst. Prof.** for guiding us from the beginning through the end of the Course Project. We express our gratitude to Head of the department CS&AI, **Dr. M. Sheshikala, Associate Professor** for encouragement, support and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, School of Computer Science and Artificial Intelligence, **Dr C. V. Guru Rao**, for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

Table of Contents:

ABSTRACT

1.1 Overview:

The "Electricity Billing System for Enhanced Energy Management" is a novel and innovative solution designed to revolutionize the conventional electricity billing process. This system leverages cutting-edge technology to not only streamline the billing process but also empower consumers and utility providers with advanced features for more efficient energy consumption and management.

Traditional electricity billing systems often rely on outdated methods, such as manual meter reading and fixed-rate billing, leading to inaccuracies and lack of control for both consumers and providers. In contrast, the proposed smart billing system integrates modern technologies like smart meters, real-time data analytics, and IoT connectivity to offer a dynamic, intelligent, and user-friendly approach to electricity billing.

- Smart Meters
- Real-time Monitoring
- Time-of-Use Billing
- Bill Estimation Energy Efficiency Tips

1.1 Purpose:

- **Accurate Billing:** Ensure that consumers are billed accurately based on their actual electricity consumption, reducing billing disputes and enhancing customer satisfaction.

- **Cost Control:** Empower consumers to understand and control their energy costs by providing detailed information on their electricity usage and costs in real time.

- **Fair Pricing:** Implement pricing models that promote fairness and transparency, such as time-ofuse rates that encourage energy conservation during peak hours.

- **Resource Management:** Assist utility providers in optimizing resource allocation and planning by collecting and analyzing consumption data, enabling efficient energy distribution.

- **Energy Conservation:** Encourage energy conservation and the use of renewable energy sources by providing insights and incentives for reducing energy consumption and using green energy.

- **Billing Efficiency:** Streamline the billing process through automation, reducing administrative overhead and costs for utility providers.

Scope:

The scope of an electricity billing system involves the collection and processing of meter data, accurate billing calculations, user authentication, user-friendly interfaces, various payment methods, notifications to consumers, time-of-use billing, renewable energy integration, data security, analytics for utility providers, billing dispute resolution, historical data storage, regulatory compliance, customer support, payment gateway integration, demand forecasting, and support for incentive programs. This comprehensive scope ensures efficient and transparent billing processes, user empowerment, and resource management.

OBJECTIVE:

Electricity billing system is to accurately and efficiently measure and charge customers for their electricity consumption. This system aims to provide transparent, timely, and convenient billing services, ensuring that customers are billed based on their actual usage, promoting energy conservation, and enabling utilities to manage their revenue collection effectively.

Additionally, it should facilitate the integration of renewable energy sources and advanced metering technologies while minimizing errors and disputes in billing processes.

DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT

3.1 About Java:

Java is a general-purpose, class-based, object-oriented programming language designedfor having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Javaapplications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.

Here are some important Java applications:

- It is used for developing Android Apps
- Helps you to create Enterprise Software
- Wide range of Mobile java Applications
- Scientific Computing Applications
- Use for Big Data Analytics
- Java Programming of Hardware devices

3.2 About SQL:

SQL is Structured Query Language, which is a computer language for storing,manipulating and retrieving data stored in a relational database.

SQL in Java typically involves using Java Database Connectivity (JDBC) to interact with relational databases. JDBC is a Java-based API that provides a standard way to connect to databases, execute SQL queries, and manage the results. You can use JDBC to perform various database operations, such as connecting to a database, executing SQL statements, and processing query results.

- Data Retrieval: Applications use SQL SELECT statements to retrieve data from a database. This is often used to display information to users in a structured manner, like product listings, user profiles, or reports.

- Data Insertion: Applications use SQL INSERT statements to add new data to the database, such as adding new records to a customer or product table.

- Data Modification: Applications use SQL UPDATE statements to modify existing data in the database. This is commonly used to update user information or adjust inventory levels.

- Data Deletion: Applications use SQL DELETE statements to remove data from the database, such as deleting user accounts or removing outdated records.

3.3 Java swings:

Swings in Java refers to the Java Swing framework, which is a set of graphical user interface (GUI) components for building desktop applications. Swing provides a rich set of tools for creating windows, buttons, menus, and other GUI elements, allowing developers to create interactive and visually appealing applications.

- Swing is a part of the Java Foundation Classes (JFC) and is included in the Java Standard Edition (SE) library.

- Swing components are platform-independent, which means that Swing-based applications look and behave consistently on different operating systems.

- Swing components are more flexible and customizable compared to the earlier Abstract Window Toolkit (AWT) components in Java.

- Swing supports a wide range of GUI components, including buttons, labels, text fields, tables, and more.

- Swing applications are typically event-driven, responding to user interactions like button clicks or mouse events.

3.4 Java AWT:

AWT, or Abstract Window Toolkit, is a graphical user interface (GUI) library in Java. It is one of the foundational GUI libraries in Java and provides a set of classes and methods for creating and managing graphical user interfaces for desktop applications.

- Components: AWT provides a wide range of components (widgets) like buttons, labels, text fields, checkboxes, etc., that you can use to build a graphical user interface.

- Platform Independence: AWT is platform-independent, which means you can create Java applications with GUIs that work on different operating systems without modification.

- Event Handling: AWT supports event-driven programming, allowing you to define event handlers for user interactions like button clicks or mouse movements.

- Lightweight and Heavyweight Components: AWT components can be categorized into lightweight and heavyweight. Lightweight components are drawn by Java and are more flexible, while heavyweight components rely on the native operating system's GUI toolkit.

3.5 Graphical User Interface(GUI):

Graphical User Interface (GUI) programming in Java typically involves using the Java Swing or JavaFX libraries to create user-friendly interfaces for your applications.

- Swing and JavaFX: These are the two main libraries for building GUI applications in Java. Swing is the older of the two, while JavaFX is more modern and provides richer features. JavaFX is often preferred for new projects.

- Components: GUI applications consist of various graphical components such as buttons, labels, text fields, and more. These components are provided by Swing or JavaFX libraries and can be customized to suit your application's needs.

- Layout Managers: Layout managers are used to arrange and manage the positioning of components within a GUI. Common layout managers include BorderLayout, FlowLayout, and GridLayout.

- Event Handling: In GUI applications, you need to handle user interactions, such as button clicks or mouse events. This is done by registering event listeners and implementing event handling code.

3.6 Java Database connection(JDBC):

In Java, there are several ways to establish a connection to a database. The most common method involves using the JDBC (Java Database Connectivity) API, which provides a standard interface for connecting to various relational databases.Java, database connectivity is typically achieved using the Java Database Connectivity (JDBC) API, which provides a standard interface for accessing relational databases. JDBC

allows Java applications to interact with various database management systems, enabling tasks such as establishing connections, executing SQL queries, and processing result sets. To use JDBC, developers need to load the appropriate database driver, establish a connection to the database, create and execute SQL statements, and handle any resulting data or exceptions appropriately. Commonly used JDBC drivers include those provided by Oracle, MySQL, and PostgreSQL, among others.

REQUIREMENT ANALYSIS

1.1 E-R DIAGRAM:

ER Diagram: An Entity-Relationship (ER) diagram is a visual representation of the data model that describes how entities (or tables) are related to each other in a database. ER diagrams are commonly used in database design and serve as a blueprint for designing the structure of a relational database. They help in understanding the data requirements, relationships, and constraints within an application.

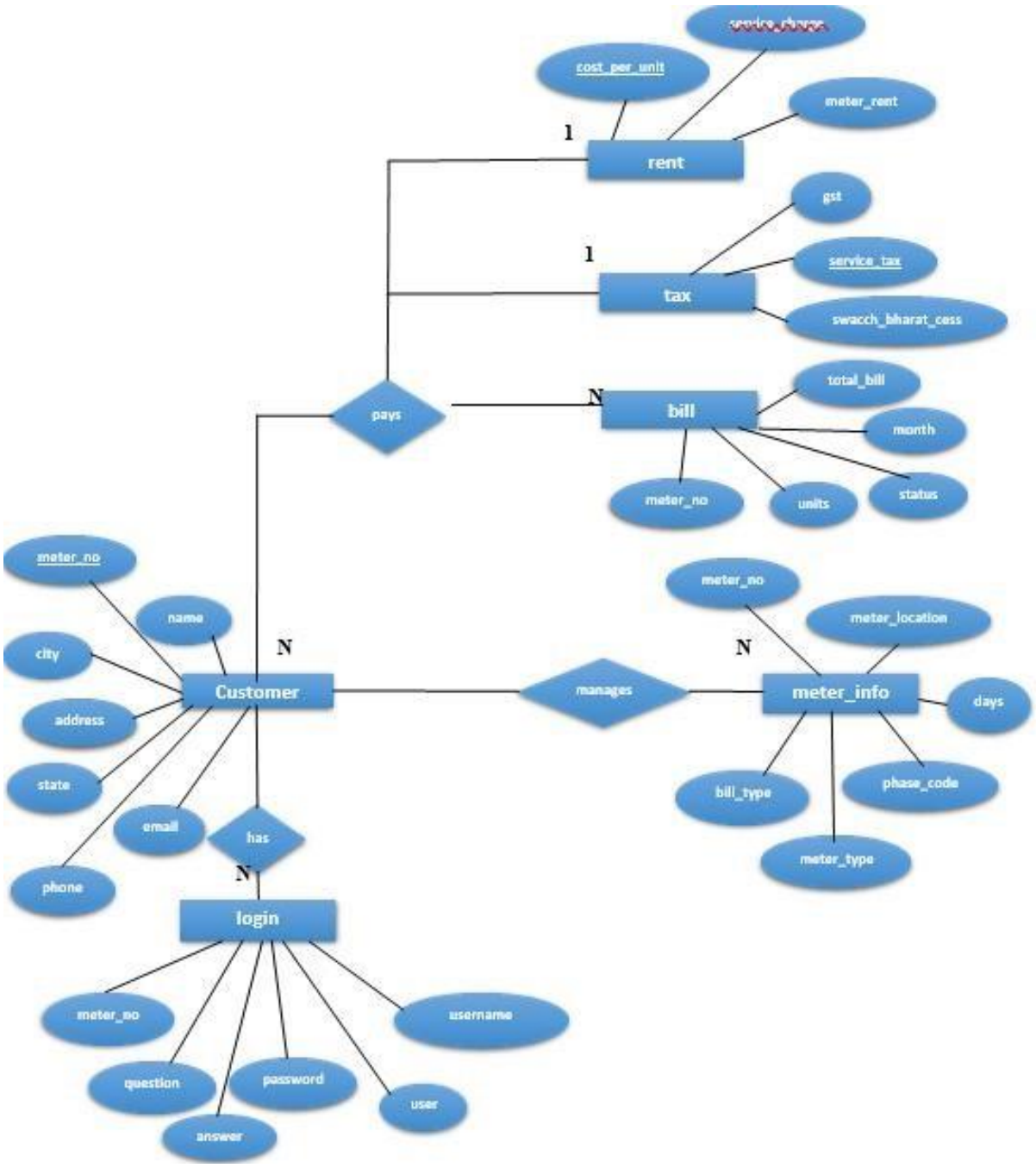


Figure 4.1: ER Diagram for electricity billing system

TABLE DESCRIPTION :

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 not equal to 10	Consume ()
Input phone number	98977agv	Phone number is invalid	Alphabets are being taken as input for phone number	_

Table 5.2: Positive test case for phone number insertion

Function Name	Input	Expected Output	Error	Resolved

Input phone number	9897778988	Expected output is seen	_	_

Table 5.3: Negative test case for email insertion

Function Name	Input	Expected Output	Error	Resolved
Input email	Sai1.in	Email is invalid	Email is not ina format given	Consume ()

Table 5.4: Positive test case for email insertion

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

Table 5.5: Negative test case for customer name 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 5.6: Positive test case for customer name insertion

Function Name	Input	Expected Output	Error	Resolved
Input customer name	Gowthu	Expected output is seen	_	_

5.1.1 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 havebeen 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	12334(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 deposit details	12334(meter_no) January(month)	Details not seen	Output not seen	Consume ()
Positive searching of total_bill	12334(meter_no) January(month)	Must display deposit details	_	_

5.1.1 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 caseswere 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 choice	The screen appears whenthe users run the program. 1.If admin login 2.If customer login	A page with differentmenu's appears. 1.Admin panel opens and 2.Customer panel opens
Step 2	The screen appears whenthe admin logs in and selects any one of the menus from the click ofthe mouse.	A window for adding new customer, insertingtax, calculate bill, view deposit details etc
Selection 1	❖ New Customer ❖ Customer Details ❖ Deposit Details ❖ Calculate Bill ❖ Tax Details ❖ Delete Customer	
Step 2.1	The screen appears whenthe customer login and selects any one of the menus from the click of the mouse	A window for generatingbill, update customer details, view details, generating bill
Selection 2	❖ Update Details ❖ View Details	
Selection 2a	❖Generate Bill	
Selection 2b	❖ Pay Bill ❖ Bill Details	

TABLE WITH VALUES

The given below table is a snapshot of backend view of the localhost and thestructures 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 withmeter_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.

```
mysql> show tables;
+----------------+
| Tables_in_elect |
+----------------+
| bill           |
| customer       |
| login          |
| meter_info     |
| rent           |
| tax            |
+----------------+
6 rows in set (0.03 sec)
```

FIG 6.1:List of tables

Login Table:

```
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)
```

FIG 6.2:login table description

Customer 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)
```

FIG 6.3: customer table description

Tax 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)
```

FIG 6.4: tax table description

Rent 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)
```

FIG 6.6: Rent table descripition

Meter_Info 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)
```

FIG 6.7: meter_info table description

IMPLEMENTATION

```
package electricity.billing.system;

import javax.swing.*;
```

```
import java.awt.*;

import java.sql.*;

import java.awt.event.*;

public class PayBill extends JFrame implements ActionListener{

    Choice cmonth;

    JButton pay, back;

    String meter;

    PayBill(String meter) {

        this.meter = meter;

        setLayout(null);

        setBounds(300, 150, 900, 600);



        JLabel heading = new JLabel("Electicity Bill");

        heading.setFont(new Font("Tahoma", Font.BOLD, 24));

        heading.setBounds(120, 5, 400, 30);

        add(heading);

        JLabel lblmeternumber = new JLabel("Meter Number");

        lblmeternumber.setBounds(35, 80, 200, 20);

        add(lblmeternumber);

        JLabel meternumber = new JLabel("");
```

```
meternumber.setBounds(300, 80, 200, 20);

add(meternumber);

JLabel lblname = new JLabel("Name");

lblname.setBounds(35, 140, 200, 20);

add(lblname);

JLabel labelname = new JLabel("");

labelname.setBounds(300, 140, 200, 20);

add(labelname);

 JLabel lblmonth = new JLabel("Month");

lblmonth.setBounds(35, 200, 200, 20);

add(lblmonth);

cmonth = new Choice();

cmonth.setBounds(300, 200, 200, 20);

cmonth.add("January");

cmonth.add("February");

cmonth.add("March");

cmonth.add("April");

cmonth.add("May");

cmonth.add("June");

cmonth.add("July");
```

```
 cmonth.add("August");

 cmonth.add("September");

 cmonth.add("October");

 cmonth.add("November");

 cmonth.add("December");

 add(cmonth);

 JLabel lblunits = new JLabel("Units");

 lblunits.setBounds(35, 260, 200, 20);

 add(lblunits);

JLabel labelunits = new JLabel("");

 labelunits.setBounds(300, 260, 200, 20);

 add(labelunits);

JLabel lbltotalbill = new JLabel("Total Bill");

 lbltotalbill.setBounds(35, 320, 200, 20);

 add(lbltotalbill);

 JLabel labeltotalbill = new JLabel("");

 labeltotalbill.setBounds(300, 320, 200, 20);

 add(labeltotalbill);

 JLabel lblstatus = new JLabel("Status");

 lblstatus.setBounds(35, 380, 200, 20);
```

```
add(lblstatus);

JLabel labelstatus = new JLabel("");

labelstatus.setBounds(300, 380, 200, 20);

labelstatus.setForeground(Color.RED);

add(labelstatus);

try {

    Conn c = new Conn();

    ResultSet rs = c.s.executeQuery("select * from customer where meter_no = '"+meter+"'");

    while(rs.next()) {

        meternumber.setText(meter);

        labelname.setText(rs.getString("name"));

    }

rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month = 'January'");

    while(rs.next()) {

        labelunits.setText(rs.getString("units"));

        labeltotalbill.setText(rs.getString("totalbill"));

        labelstatus.setText(rs.getString("status"));

    }

} catch (Exception e) {

    e.printStackTrace();
```

```
        }

cmonth.addItemListener(new ItemListener(){

            @Override

            public void itemStateChanged(ItemEvent ae) {

                try {

                    Conn c = new Conn();

                    ResultSet rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month
= '"+cmonth.getSelectedItem()+"'");

                    while(rs.next()) {

                        labelunits.setText(rs.getString("units"));

                        labeltotalbill.setText(rs.getString("totalbill"));

                        labelstatus.setText(rs.getString("status"));

                    }

                } catch (Exception e) {

                    e.printStackTrace();

                }

            }

        });

         pay = new JButton("Pay");

        pay.setBackground(Color.BLACK);

        pay.setForeground(Color.WHITE);
```

```
    pay.setBounds(100, 460, 100, 25);

    pay.addActionListener(this);

    add(pay);

    back = new JButton("Back");

    back.setBackground(Color.BLACK);

    back.setForeground(Color.WHITE);

    back.setBounds(230, 460, 100, 25);

    back.addActionListener(this);

    add(back);

   getContentPane().setBackground(Color.WHITE);

  ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/bill.png"))

    Image i2 = i1.getImage().getScaledInstance(600, 300, Image.SCALE_DEFAULT);

    ImageIcon i3 = new ImageIcon(i2);

    JLabel image = new JLabel(i3);

    image.setBounds(400, 120, 600, 300);

    add(image);

    setVisible(true);

} public void actionPerformed(ActionEvent ae) {

    if (ae.getSource() == pay) {

        try {
```

```
            Conn c = new Conn();

            c.s.executeUpdate("update bill set status = 'Paid' where meter_no = '"+meter+"' AND
month='"+cmonth.getSelectedItem()+"'");

        } catch (Exception e) {

            e.printStackTrace();

        }

        setVisible(false);

        new Paytm(meter);

    } else {

        setVisible(false);

    }

  }public static void main(String[] args){

    new PayBill("");

  }

}
```

- **LOGIN Implementation**

```
package electricity.billing.system;

import javax.swing.*;

import java.awt.*;

import java.sql.*;

import java.awt.event.*;
```

```
public class PayBill extends JFrame implements ActionListener{

  Choice cmonth;

    JButton pay, back;

    String meter;

    PayBill(String meter) {

        this.meter = meter;

        setLayout(null);

        setBounds(300, 150, 900, 600);

 JLabel heading = new JLabel("Electicity Bill");

        heading.setFont(new Font("Tahoma", Font.BOLD, 24));

        heading.setBounds(120, 5, 400, 30);

add(heading);

 JLabel lblmeternumber = new JLabel("Meter Number");

        lblmeternumber.setBounds(35, 80, 200, 20);

        add(lblmeternumber);

      JLabel meternumber = new JLabel("");

        meternumber.setBounds(300, 80, 200, 20);

        add(meternumber);

      JLabel lblname = new JLabel("Name");

        lblname.setBounds(35, 140, 200, 20);
```

```
 add(lblname);
JLabel labelname = new JLabel("");
 labelname.setBounds(300, 140, 200, 20);
 add(labelname);
JLabel lblmonth = new JLabel("Month");
 lblmonth.setBounds(35, 200, 200, 20);
 add(lblmonth);
cmonth = new Choice();
 cmonth.setBounds(300, 200, 200, 20);
 cmonth.add("January");
 cmonth.add("February");
 cmonth.add("March");
 cmonth.add("April");
 cmonth.add("May");
 cmonth.add("June");
 cmonth.add("July");
 cmonth.add("August");
 cmonth.add("September");
 cmonth.add("October");
 cmonth.add("November");
```

```
cmonth.add("December");

add(cmonth);

JLabel lblunits = new JLabel("Units");

lblunits.setBounds(35, 260, 200, 20);

add(lblunits);

JLabel labelunits = new JLabel("");

labelunits.setBounds(300, 260, 200, 20);

add(labelunits);

JLabel lbltotalbill = new JLabel("Total Bill");

lbltotalbill.setBounds(35, 320, 200, 20);

add(lbltotalbill);

JLabel labeltotalbill = new JLabel("");

labeltotalbill.setBounds(300, 320, 200, 20);

add(labeltotalbill);

JLabel lblstatus = new JLabel("Status");

lblstatus.setBounds(35, 380, 200, 20);

add(lblstatus);

JLabel labelstatus = new JLabel("");

labelstatus.setBounds(300, 380, 200, 20);

labelstatus.setForeground(Color.RED);
```

```
        add(labelstatus);
        try {
            Conn c = new Conn();
            ResultSet rs = c.s.executeQuery("select * from customer where meter_no = '"+meter+"'");
            while(rs.next()) {
                meternumber.setText(meter);
                labelname.setText(rs.getString("name"));
            }
rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month = 'January'");
            while(rs.next()) {
                labelunits.setText(rs.getString("units"));
                labeltotalbill.setText(rs.getString("totalbill"));
                labelstatus.setText(rs.getString("status"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }  cmonth.addItemListener(new ItemListener(){
            @Override
            public void itemStateChanged(ItemEvent ae) {
                try {
```

```
                Conn c = new Conn();

                ResultSet rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month
= '"+cmonth.getSelectedItem()+"'");

                while(rs.next()) {

                    labelunits.setText(rs.getString("units"));

                    labeltotalbill.setText(rs.getString("totalbill"));

                    labelstatus.setText(rs.getString("status"));

                }

            } catch (Exception e) {

                e.printStackTrace();

            }

        }

});

    pay = new JButton("Pay");

      pay.setBackground(Color.BLACK);

      pay.setForeground(Color.WHITE);

      pay.setBounds(100, 460, 100, 25);

      pay.addActionListener(this);

      add(pay);

      back = new JButton("Back");

      back.setBackground(Color.BLACK);
```

```
        back.setForeground(Color.WHITE);

        back.setBounds(230, 460, 100, 25);

        back.addActionListener(this);

        add(back);

       getContentPane().setBackground(Color.WHITE);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/bill.png"));

        Image i2 = i1.getImage().getScaledInstance(600, 300, Image.SCALE_DEFAULT);

        ImageIcon i3 = new ImageIcon(i2);

        JLabel image = new JLabel(i3);

        image.setBounds(400, 120, 600, 300);

        add(image);

  setVisible(true);

    }

   public void actionPerformed(ActionEvent ae) {

        if (ae.getSource() == pay) {

            try {

                Conn c = new Conn();

                c.s.executeUpdate("update bill set status = 'Paid' where meter_no = '"+meter+"' AND
month='"+cmonth.getSelectedItem()+"'");

            } catch (Exception e) {

                e.printStackTrace();
```

```
            }
            setVisible(false);
            new Paytm(meter);
        } else {
            setVisible(false);
        }
    }
  public static void main(String[] args){
        new PayBill("");
    }
}
```

- **PAY BILL Implementation**

```
package electricity.billing.system;
import javax.swing.*;
import java.awt.*;
import java.sql.*;
import java.awt.event.*;
public class PayBill extends JFrame implements ActionListener{
    Choice cmonth;
    JButton pay, back;
```

```
    String meter;

    PayBill(String meter) {

        this.meter = meter;

        setLayout(null);

        setBounds(300, 150, 900, 600);

  JLabel heading = new JLabel("Electicity Bill");

        heading.setFont(new Font("Tahoma", Font.BOLD, 24));

        heading.setBounds(120, 5, 400, 30);

        add(heading);

  JLabel lblmeternumber = new JLabel("Meter Number");

        lblmeternumber.setBounds(35, 80, 200, 20);

        add(lblmeternumber);

 JLabel meternumber = new JLabel("");

        meternumber.setBounds(300, 80, 200, 20);

        add(meternumber);

  JLabel lblname = new JLabel("Name");

        lblname.setBounds(35, 140, 200, 20);

        add(lblname);

    JLabel labelname = new JLabel("");

        labelname.setBounds(300, 140, 200, 20);
```

```
    add(labelname);
JLabel lblmonth = new JLabel("Month");
    lblmonth.setBounds(35, 200, 200, 20);
    add(lblmonth);
    cmonth = new Choice();
    cmonth.setBounds(300, 200, 200, 20);
    cmonth.add("January");
    cmonth.add("February");
    cmonth.add("March");
    cmonth.add("April");
    cmonth.add("May");
    cmonth.add("June");
    cmonth.add("July");
    cmonth.add("August");
    cmonth.add("September");
    cmonth.add("October");
    cmonth.add("November");
    cmonth.add("December");
    add(cmonth);
    JLabel lblunits = new JLabel("Units");
```

```
lblunits.setBounds(35, 260, 200, 20);

add(lblunits);

JLabel labelunits = new JLabel("");

labelunits.setBounds(300, 260, 200, 20);

add(labelunits);

JLabel lbltotalbill = new JLabel("Total Bill");

lbltotalbill.setBounds(35, 320, 200, 20);

add(lbltotalbill);

JLabel labeltotalbill = new JLabel("");

labeltotalbill.setBounds(300, 320, 200, 20);

add(labeltotalbill);

JLabel lblstatus = new JLabel("Status");

lblstatus.setBounds(35, 380, 200, 20);

add(lblstatus);

JLabel labelstatus = new JLabel("");

labelstatus.setBounds(300, 380, 200, 20);

labelstatus.setForeground(Color.RED);

add(labelstatus);

try {

    Conn c = new Conn();
```

```
            ResultSet rs = c.s.executeQuery("select * from customer where meter_no = '"+meter+"'");

            while(rs.next()) {

                meternumber.setText(meter);

                labelname.setText(rs.getString("name"));

            }

   rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month = 'January'");

            while(rs.next()) {

                labelunits.setText(rs.getString("units"));

                labeltotalbill.setText(rs.getString("totalbill"));

                labelstatus.setText(rs.getString("status"));

            }

} catch (Exception e) {

            e.printStackTrace();

        }

  cmonth.addItemListener(new ItemListener(){

            @Override

            public void itemStateChanged(ItemEvent ae) {

                try {

                    Conn c = new Conn();

                    ResultSet rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month
= '"+cmonth.getSelectedItem()+"'");
```

```
            while(rs.next()) {
                labelunits.setText(rs.getString("units"));
                labeltotalbill.setText(rs.getString("totalbill"));
                labelstatus.setText(rs.getString("status"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
pay = new JButton("Pay");
pay.setBackground(Color.BLACK);
pay.setForeground(Color.WHITE);
pay.setBounds(100, 460, 100, 25);
pay.addActionListener(this);
add(pay);
back = new JButton("Back");
back.setBackground(Color.BLACK);
back.setForeground(Color.WHITE);
back.setBounds(230, 460, 100, 25);
```

```
        back.addActionListener(this);

        add(back);

       getContentPane().setBackground(Color.WHITE);

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/bill.png"));

        Image i2 = i1.getImage().getScaledInstance(600, 300, Image.SCALE_DEFAULT);

        ImageIcon i3 = new ImageIcon(i2);

        JLabel image = new JLabel(i3);

        image.setBounds(400, 120, 600, 300);

        add(image);

       setVisible(true);

    }

   public void actionPerformed(ActionEvent ae) {

        if (ae.getSource() == pay) {

            try {

                Conn c = new Conn();

                c.s.executeUpdate("update bill set status = 'Paid' where meter_no = '"+meter+"' AND
month='"+cmonth.getSelectedItem()+"'");

            } catch (Exception e) {

                e.printStackTrace();

            }

            setVisible(false);
```

```
            new Paytm(meter);
        } else {
            setVisible(false);
        }
    }  public static void main(String[] args){
        new PayBill("");
    }
}
```

- **Update Information**

```
package electricity.billing.system;
import javax.swing.*;
import java.awt.*;
import java.sql.*;
import java.awt.event.*;
public class PayBill extends JFrame implements ActionListener{
   Choice cmonth;
    JButton pay, back;
    String meter;
    PayBill(String meter) {
        this.meter = meter;
```

```
setLayout(null);

setBounds(300, 150, 900, 600);

 JLabel heading = new JLabel("Electicity Bill");

heading.setFont(new Font("Tahoma", Font.BOLD, 24));

heading.setBounds(120, 5, 400, 30);

add(heading);

JLabel lblmeternumber = new JLabel("Meter Number");

lblmeternumber.setBounds(35, 80, 200, 20);

add(lblmeternumber);

JLabel meternumber = new JLabel("");

meternumber.setBounds(300, 80, 200, 20);

add(meternumber);

JLabel lblname = new JLabel("Name");

lblname.setBounds(35, 140, 200, 20);

add(lblname);

JLabel labelname = new JLabel("");

labelname.setBounds(300, 140, 200, 20);

add(labelname);

JLabel lblmonth = new JLabel("Month");

lblmonth.setBounds(35, 200, 200, 20);
```

```
add(lblmonth);

cmonth = new Choice();

cmonth.setBounds(300, 200, 200, 20);

cmonth.add("January");

cmonth.add("February");

cmonth.add("March");

cmonth.add("April");

cmonth.add("May");

cmonth.add("June");

cmonth.add("July");

cmonth.add("August");

cmonth.add("September");

cmonth.add("October");

cmonth.add("November");

cmonth.add("December");

add(cmonth);

JLabel lblunits = new JLabel("Units");

lblunits.setBounds(35, 260, 200, 20);

add(lblunits);

JLabel labelunits = new JLabel("");
```

```
labelunits.setBounds(300, 260, 200, 20);

add(labelunits);

JLabel lbltotalbill = new JLabel("Total Bill");

lbltotalbill.setBounds(35, 320, 200, 20);

add(lbltotalbill);

JLabel labeltotalbill = new JLabel("");

labeltotalbill.setBounds(300, 320, 200, 20);

add(labeltotalbill);

 JLabel lblstatus = new JLabel("Status");

lblstatus.setBounds(35, 380, 200, 20);

add(lblstatus);

JLabel labelstatus = new JLabel("");

labelstatus.setBounds(300, 380, 200, 20);

labelstatus.setForeground(Color.RED);

add(labelstatus);

try {

    Conn c = new Conn();

    ResultSet rs = c.s.executeQuery("select * from customer where meter_no = '"+meter+"'");

    while(rs.next()) {

        meternumber.setText(meter);
```

```
            labelname.setText(rs.getString("name"));

        }

    rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month = 'January'");

        while(rs.next()) {

            labelunits.setText(rs.getString("units"));

            labeltotalbill.setText(rs.getString("totalbill"));

            labelstatus.setText(rs.getString("status"));

        }

    } catch (Exception e) {

        e.printStackTrace();

    }

    cmonth.addItemListener(new ItemListener(){

        @Override

        public void itemStateChanged(ItemEvent ae) {

            try {

                Conn c = new Conn();

                ResultSet rs = c.s.executeQuery("select * from bill where meter_no = '"+meter+"' AND month
= '"+cmonth.getSelectedItem()+"'");

                while(rs.next()) {

                    labelunits.setText(rs.getString("units"));

                    labeltotalbill.setText(rs.getString("totalbill"));
```

```
                labelstatus.setText(rs.getString("status"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
pay = new JButton("Pay");
pay.setBackground(Color.BLACK);
pay.setForeground(Color.WHITE);
pay.setBounds(100, 460, 100, 25);
pay.addActionListener(this);
add(pay);
back = new JButton("Back");
back.setBackground(Color.BLACK);
back.setForeground(Color.WHITE);
back.setBounds(230, 460, 100, 25);
back.addActionListener(this);
add(back);
getContentPane().setBackground(Color.WHITE);
```

```
    ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/bill.png"));

    Image i2 = i1.getImage().getScaledInstance(600, 300, Image.SCALE_DEFAULT);

    ImageIcon i3 = new ImageIcon(i2);

    JLabel image = new JLabel(i3);

    image.setBounds(400, 120, 600, 300);

    add(image);

  setVisible(true);

  }

 public void actionPerformed(ActionEvent ae) {

    if (ae.getSource() == pay) {

        try {

            Conn c = new Conn();

            c.s.executeUpdate("update bill set status = 'Paid' where meter_no = '"+meter+"' AND
month='"+cmonth.getSelectedItem()+"'");

        } catch (Exception e) {

            e.printStackTrace();

        }

        setVisible(false);

        new Paytm(meter);

    } else {

        setVisible(false);
```

```
        }

public static void main(String[] args){

        new PayBill("");

    }

}
```

SNAPSHOTS

FIG 6.9: Login page

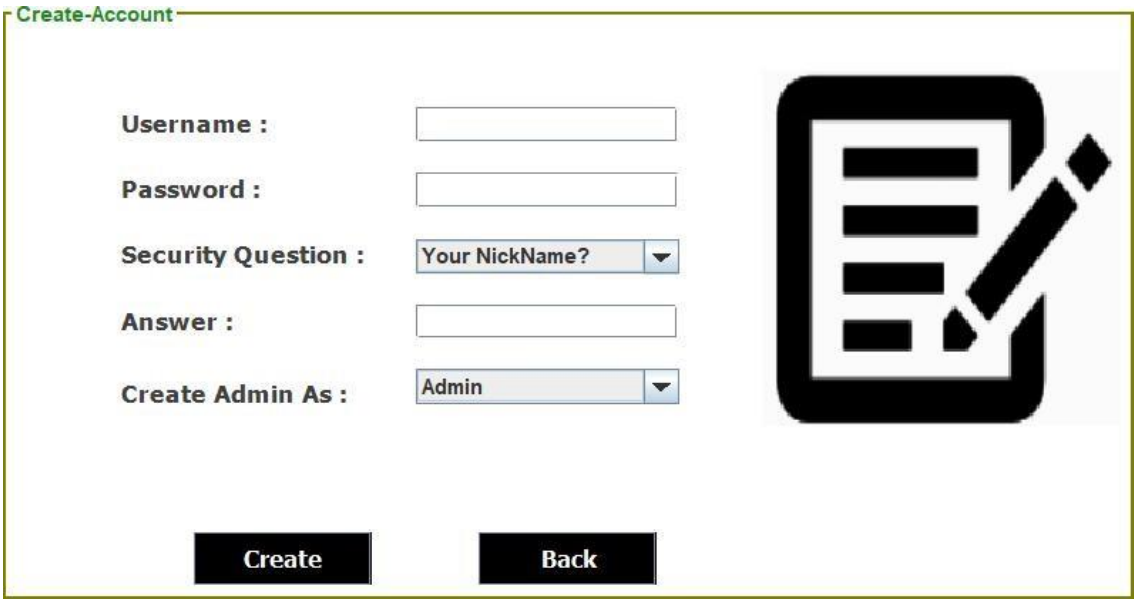


FIG 6.10: Sign up page

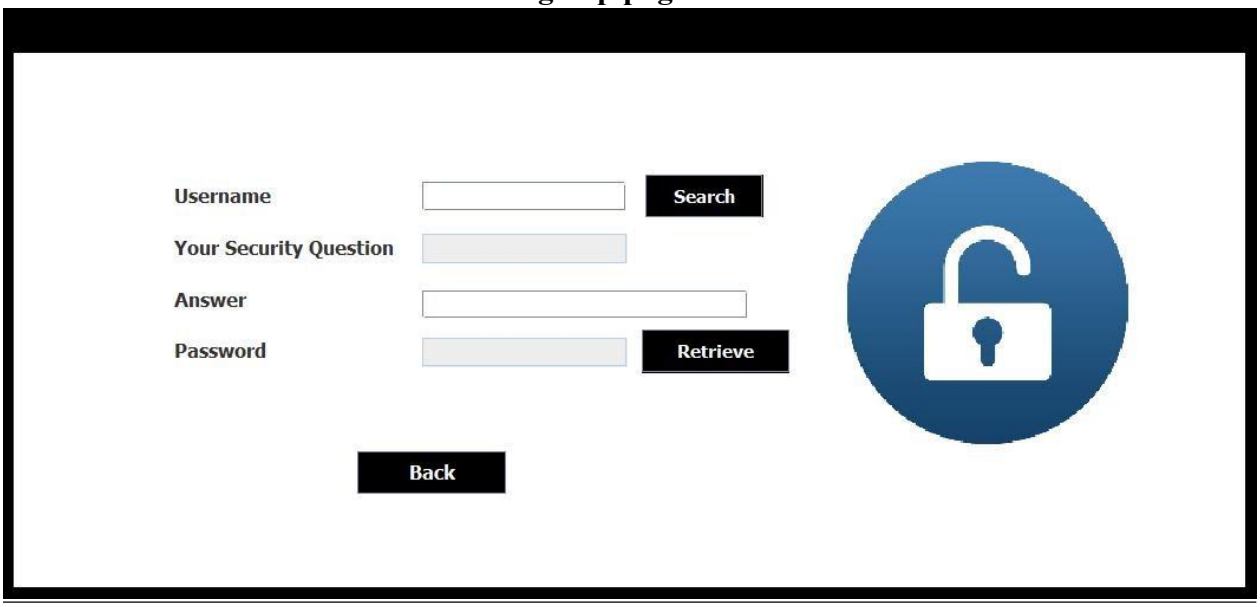


FIG 6.11: Forgot Password page

FIG 6.12: Admin home page

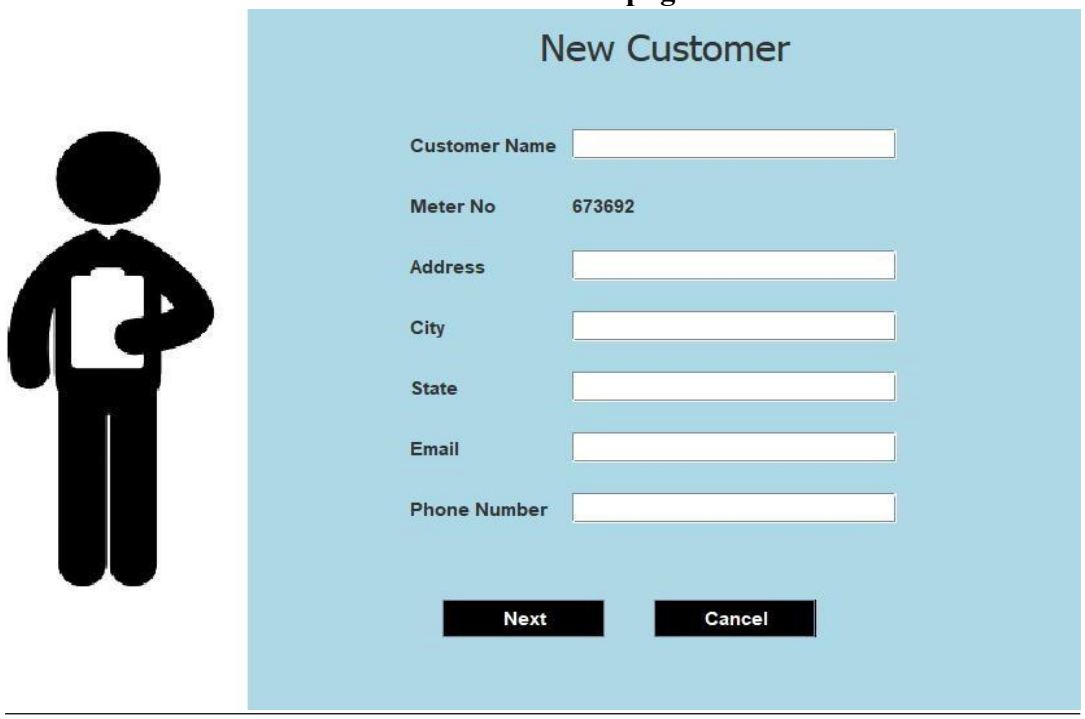


FIG 6.13: New customer page

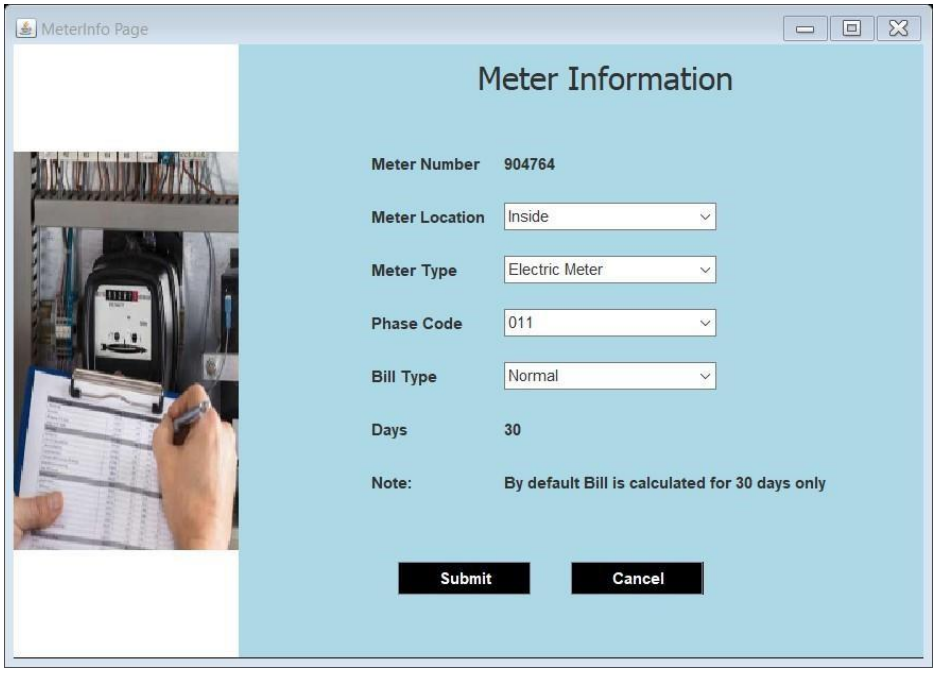


FIG 6.14: Meter

FIG 6.15: Customer Details page

FIG 6.16: Tax Details page

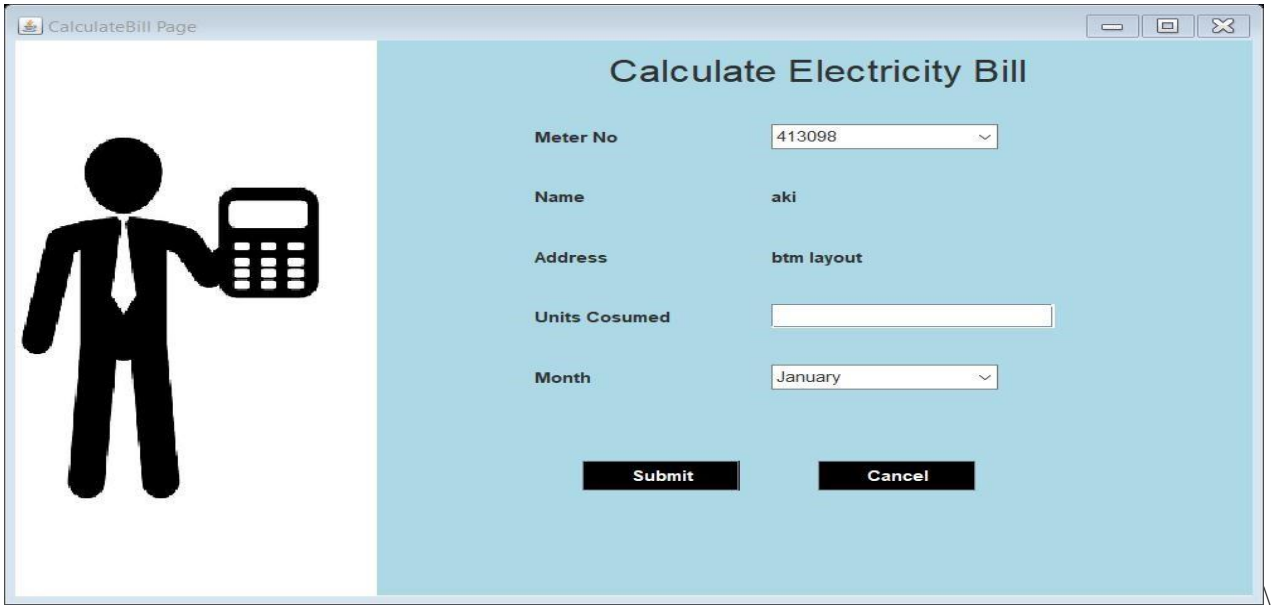


FIG 6.17: Calculate Bill page

FIG 6.18: Delete Customer page

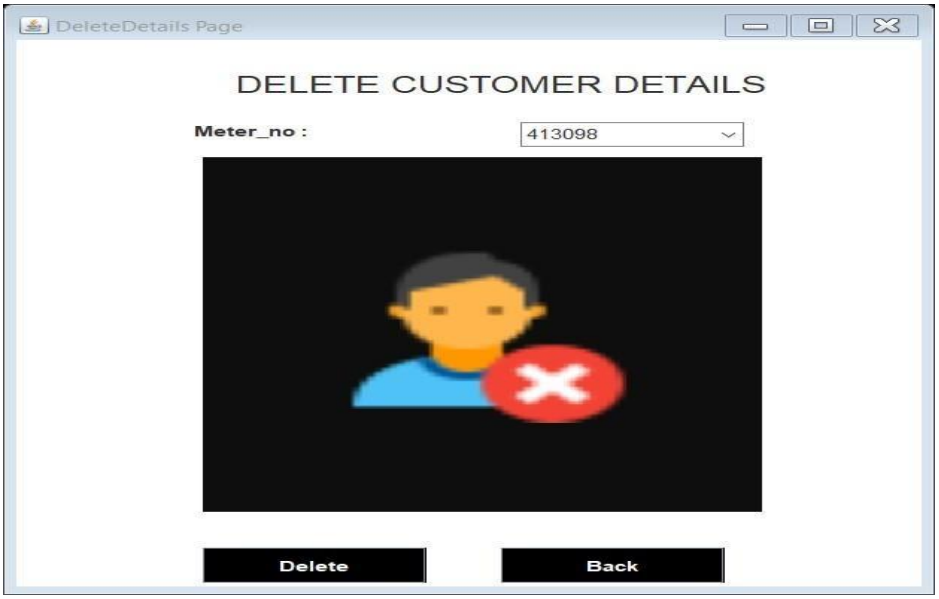


FIG 6.19: Customer Home page

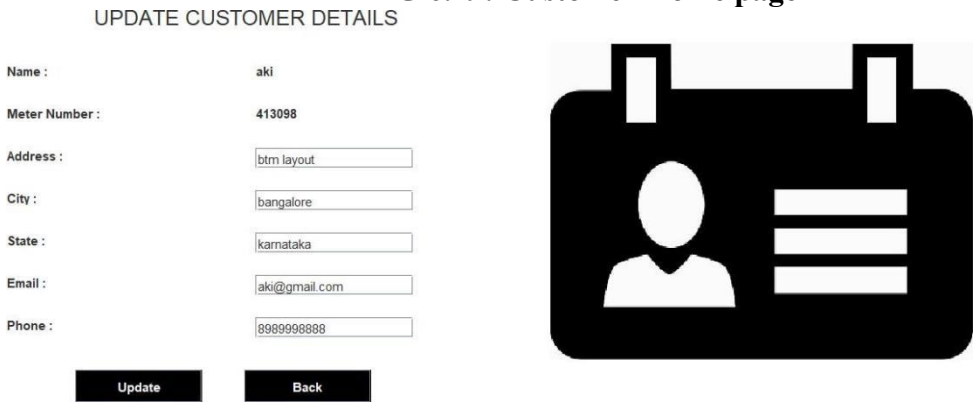


FIG 6.20: Update customer details page

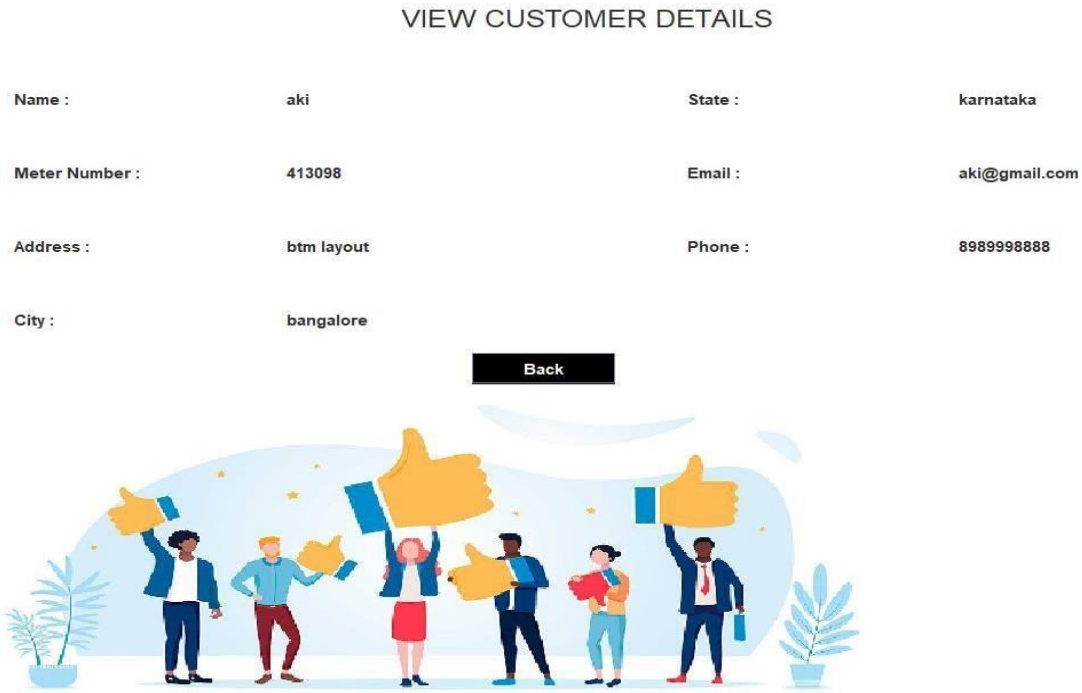
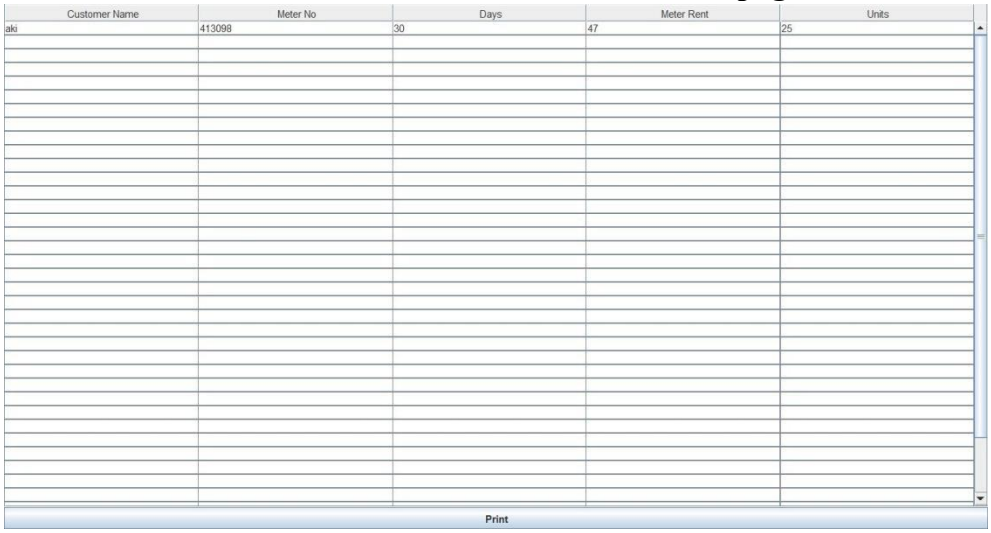


FIG 6.21: View Customer Details page



Customer Name	Meter No	Days	Meter Rent	Units
aki	413098	30	47	25

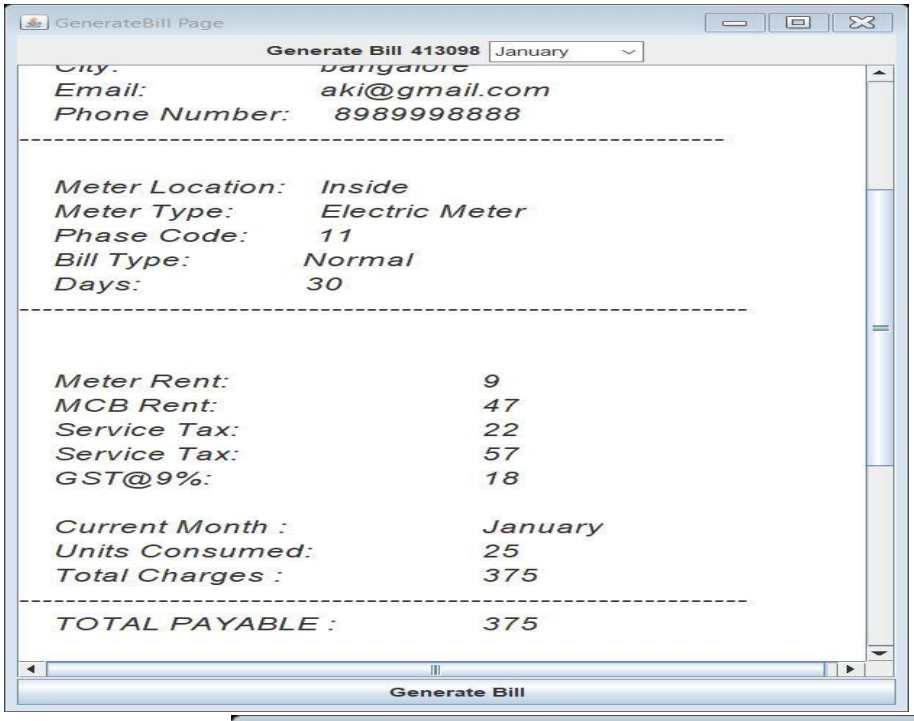
Print

FIG 6.22: Query page

FIG 6.23: Pay Bill page

FIG 6.24: Paytm page FIG

6.26: Generate Bill page



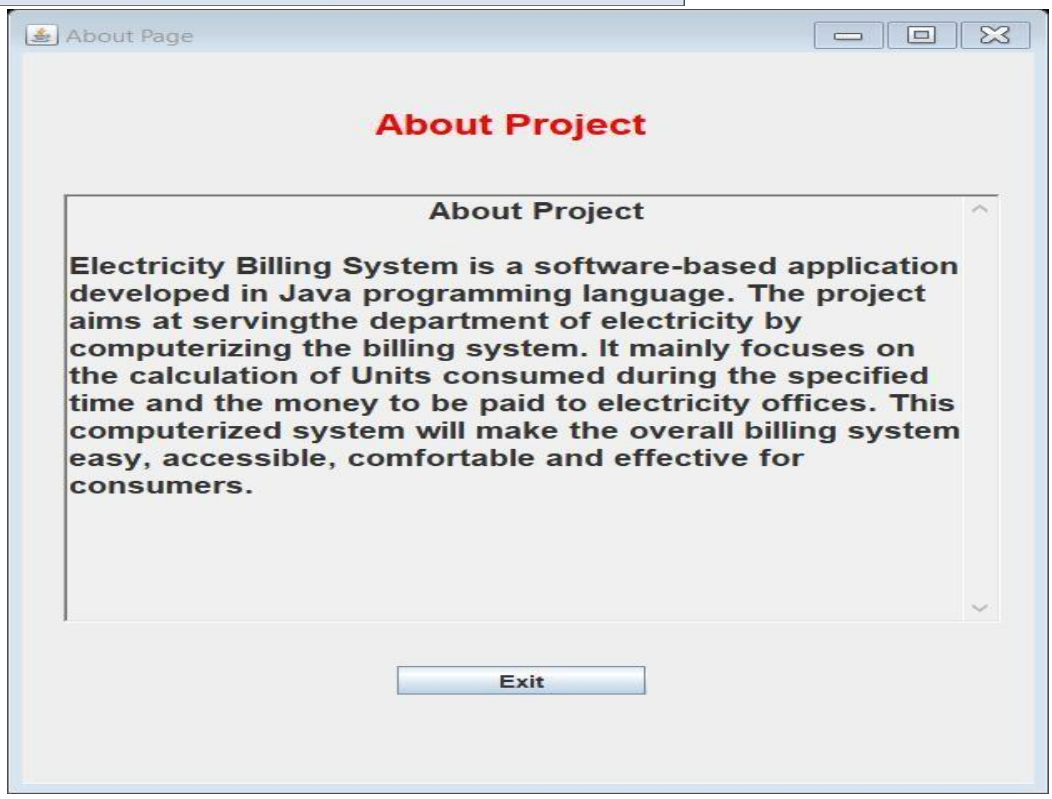


FIG 6.28: About page

CONCLUSION

The project entitled as **Electricity Billing System** is the system that deals withthe issues related to a particular institution.
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.

BIBLIOGRAPHY

REFERENCES

Database Management Systems 3rd Edition by Raghu Ramakrishnan (TEXTBOOK).

Websites:

- https://www.youtube.com/watch?v=iWitVuW2D1o&t=4s
- www.stackoverflow.com
- www.google.com
- http://www.javatpoint.com/