

Design and Implementation of an Online Electricity Billing Application using Java Applets

A CAPSTONE PROJECT
Submitted By

Pyari srivastava.P
192224178

In Partial Fulfillment for the completion of the course

CSA0912

Programming in Java for Accessing Database

Sep 2024



SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CHENNAI - 602105
TAMIL NADU, INDIA



SAVEETHA
INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
(Declared as Deemed to be University under Section 3 of UGC Act 1956)



BONAFIDE CERTIFICATE

This is to certify that the project report entitled “**Design and Implementation of an Online Electricity Billing Application using Java Applets**” submitted by Pyari Srivastava.P, 192224178 to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements per this institution's regulations and in my appraisal meets the required standards for submission.

Dr.K.Jayasakthi Velmurugan

COURSE FACULTY

Department of Deep Learning.

Saveetha School of Engineering,

SIMATS, Chennai - 602105

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N M VEERAIYAN**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. DEEPAK NALLASWAMY**, SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. RAMYA DEEPAK**, Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B RAMESH**, Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **Dr.K.Jayasakthi Velmurugan**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

INDEX

BONAFIDE CERTIFICATE	1
ACKNOWLEDGEMENT	2
1. ABSTRACT	4
2. INTRODUCTION.....	4
3. ARCHITECTURE DIAGRAM	
4. FLOWCHART	7
5. UML DIAGRAM.....	8
6. CLASS DIAGRAM.....	8
7. CODE IMPLEMENTATION.....	
7.1 JAVA CODE	
7.2 BACK END	
8. OUTPUT SCREENSHOT	10
9. CONCLUSION	19
10. REFERENCES.....	21

1. ABSTRACT

This project demonstrates the potential of Java-based applications in streamlining utility billing processes, reducing manual errors, and improving the overall customer experience. Through the use of Java Applets, the application leverages cross-platform compatibility, enhancing accessibility for users across different environments. The result is a scalable, secure, and efficient solution for modern utility management. With the rapid advancement of technology, manual utility billing processes have become increasingly inefficient and prone to errors. This project proposes the design and implementation of an **Online Electricity Billing Application** using **Java Applets**, addressing the growing need for an automated, reliable, and user-friendly system. The system is designed to streamline the end-to-end billing process, from consumption monitoring to payment processing, providing both consumers and administrators with an effective platform for managing electricity usage and billing.

The application's core is built around Java Applets, providing a lightweight, interactive, and cross-platform interface. Consumers can register, log in securely, and access their real-time electricity consumption data, which is fetched dynamically from the back-end system. Based on this data, the system calculates monthly bills automatically, factoring in multi-tier tariff structures, energy slab rates, taxes, and applicable surcharges. Additionally, the application allows users to view their billing history, download invoices, and make payments online using secure gateways. Notifications are integrated to alert users about upcoming due dates, payment confirmations, and significant changes in consumption trends.

On the administrative side, the system provides comprehensive tools for managing customer profiles, electricity consumption records, tariff configurations, and payment logs. The administrators can generate and download detailed reports for operational analytics, monitor pending bills, and manage customer inquiries. The system is built with a strong emphasis on data integrity, security, and scalability. A centralized database, interfacing with the applet, stores all customer details, billing records, and transaction data, ensuring both reliability and the ability to handle growing volumes of data as the customer base expands. This project showcases how Java-based web applications can simplify the traditionally complex utility billing processes, reduce human errors, and enhance customer satisfaction through automation.

2. INTRODUCTION

The increasing demand for efficient and user-friendly utility management systems has led to the development of various automated billing solutions. This project focuses on the design and implementation of an online electricity billing application using Java Applets. The system automates the process of electricity bill calculation, customer data management, and payment integration, aiming to provide an intuitive and seamless experience for both the users and the administrators. The application employs a user-friendly interface built with Java Applets to allow customers to log in, view their electricity consumption details, and generate bills dynamically based on real-time data. The system calculates charges based on predefined tariffs, accounting for varying consumption slabs and surcharges. Additionally, the application offers secure payment processing and automated notifications to users about payment deadlines and consumption trends.

On the administrative side, the system facilitates the management of customer profiles, tracking of payments, and generation of reports for operational monitoring. The back-end is supported by a robust database for storing customer information and transaction history, ensuring data integrity and security. This project focuses on the design and implementation of an Online Electricity Billing Application using Java Applets to create a dynamic, web-based solution for managing electricity bills. The application provides users with the convenience of real-time access to their electricity consumption data, automated bill generation, and secure online payments. By utilizing Java Applets, which are lightweight and portable, the system can be easily deployed across various platforms and accessed through standard web browsers, offering flexibility and accessibility to a wide range of users.

Java Applets serve as the core front-end technology, allowing customers to interact seamlessly with the system. These applets retrieve real-time consumption data from the back-end system, where the customer's electricity usage is tracked. Based on predefined tariff structures, the applet calculates the bill automatically, ensuring that the amount accurately reflects the user's electricity consumption for the billing period. The application also facilitates features such as payment tracking, invoice generation, and consumption analysis, enhancing transparency and enabling users to make informed decisions about their energy usage.

For utility providers, the system offers a streamlined approach to managing large datasets, customer profiles, and payment records. By automating bill calculations and providing real-time data insights, utility companies can minimize the errors associated with manual data entry, reduce operational costs, and improve revenue collection efficiency. Additionally, administrators have access to tools that allow them to configure tariffs, monitor unpaid bills, generate consumption reports, and respond to customer inquiries.

The choice of **Java Applets** for this project leverages several key advantages, including the ability to run in a browser without the need for external installations, cross-platform compatibility, and support for real-time interactivity. While Java Applets have seen a decline in use due to evolving web technologies, they offer a simple, fast-loading solution for lightweight applications in certain legacy systems and restricted environments where modern web frameworks may not be feasible.

This paper will explore the technical architecture of the application, key functionalities such as bill generation, payment integration, and security protocols, as well as the challenges encountered during the development process. Ultimately, this project aims to present a comprehensive solution to modernizing electricity billing systems, reducing human intervention, and improving the overall experience for both consumers and utility providers.

3. ARCHITECTURE DIAGRAM

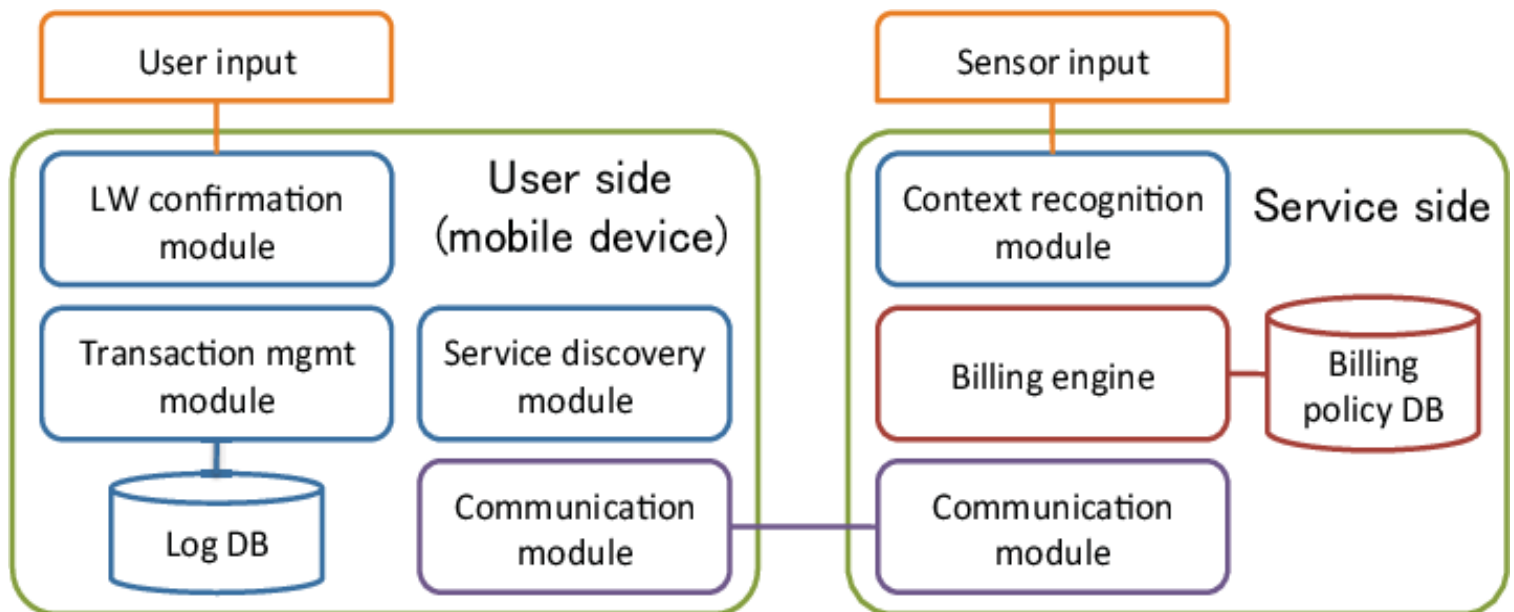


Fig1

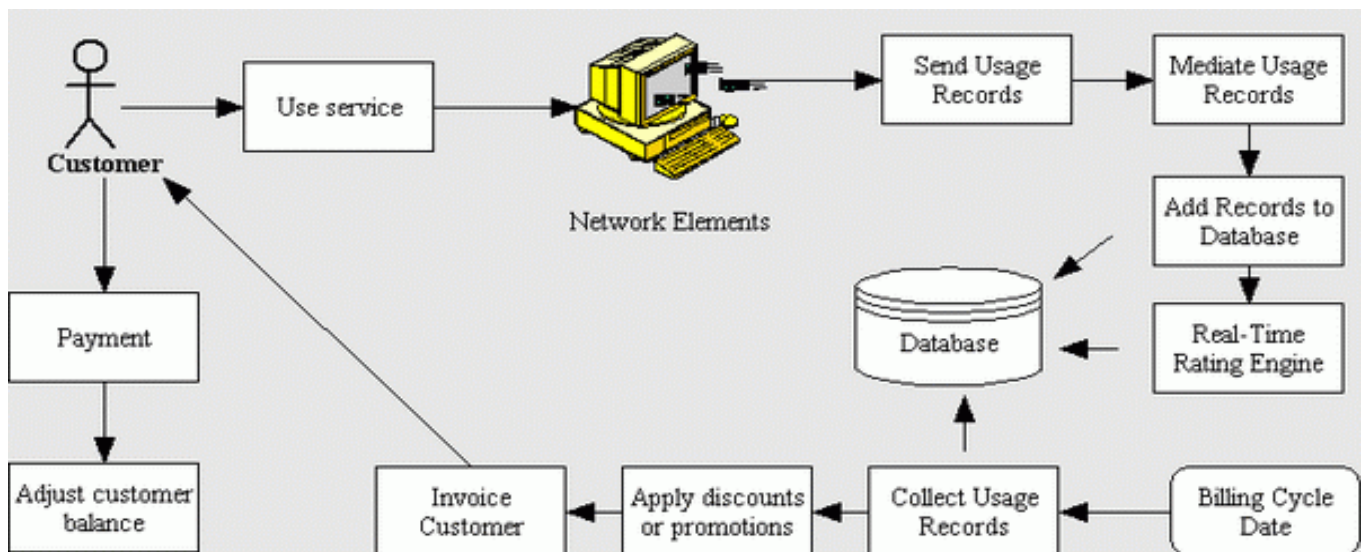


Fig2

The two images provided outline different architecture components, which can be applied to the design of an Online Electricity Billing Application using Java Applets.

User-side (Client Layer): The customer interacts with the system via Java Applets, viewing their electricity consumption, generating bills, and making payments.

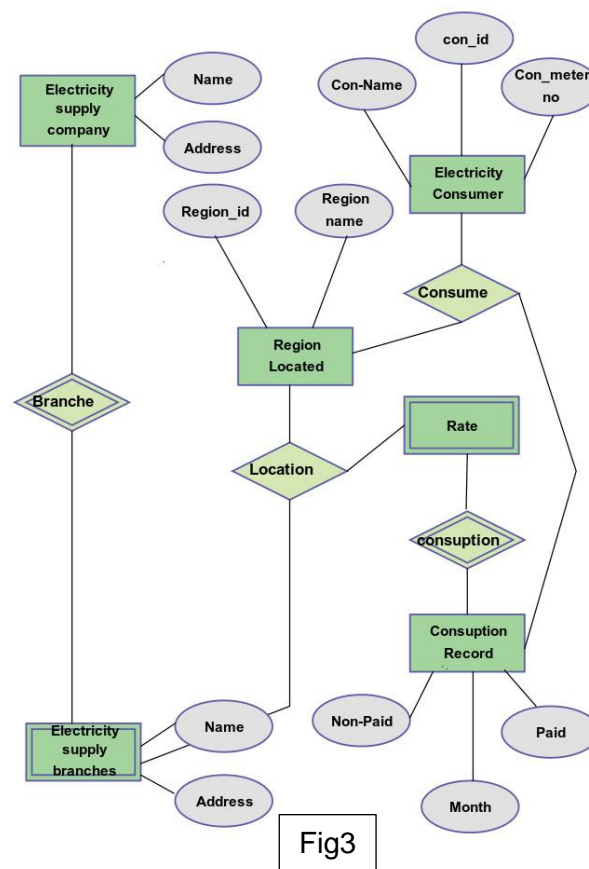
Service-side (Application Layer): This layer handles consumption data processing, bill calculation, and policy enforcement using modules like the billing engine and real-time rating engine.

Database Layer: Both diagrams highlight the importance of a centralized database where billing policies, transaction records, and customer data are stored for future reference and operational integrity.

Real-Time Processing: The integration of usage records from smart meters ensures that bill generation is accurate and updated in real-time, reflecting actual consumption.

This architecture efficiently streamlines the electricity billing process, ensuring that customers have a smooth experience while administrators can manage the system with real-time data analytics.

4. FLOWCHART



The diagram shows how consumers are linked to specific regions and branches, and how their electricity usage is recorded. Each consumer has a unique meter number, and their monthly consumption is tracked with attributes indicating whether the bill is paid or unpaid. This structure facilitates efficient tracking of consumption, billing, and payment statuses for each customer.

5. UML DIAGRAM

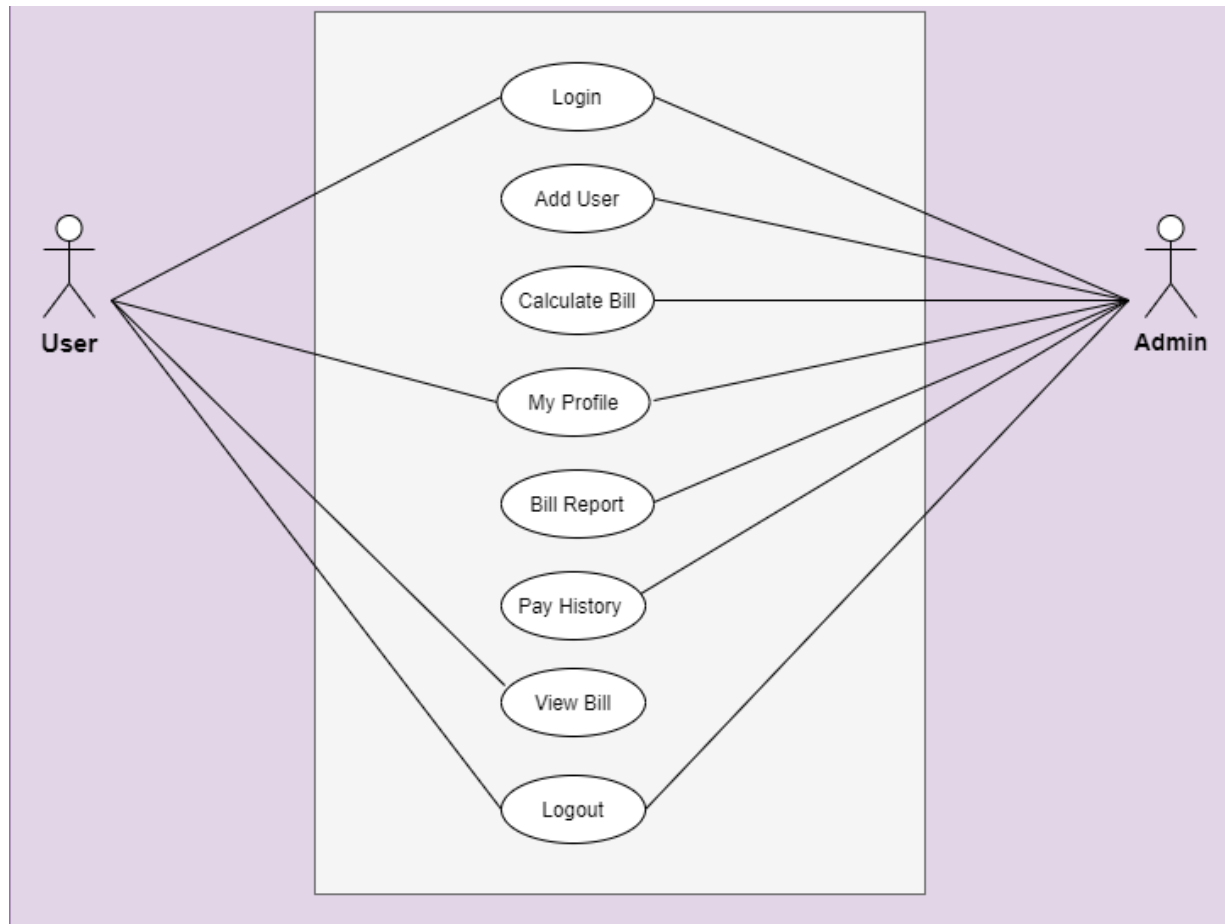
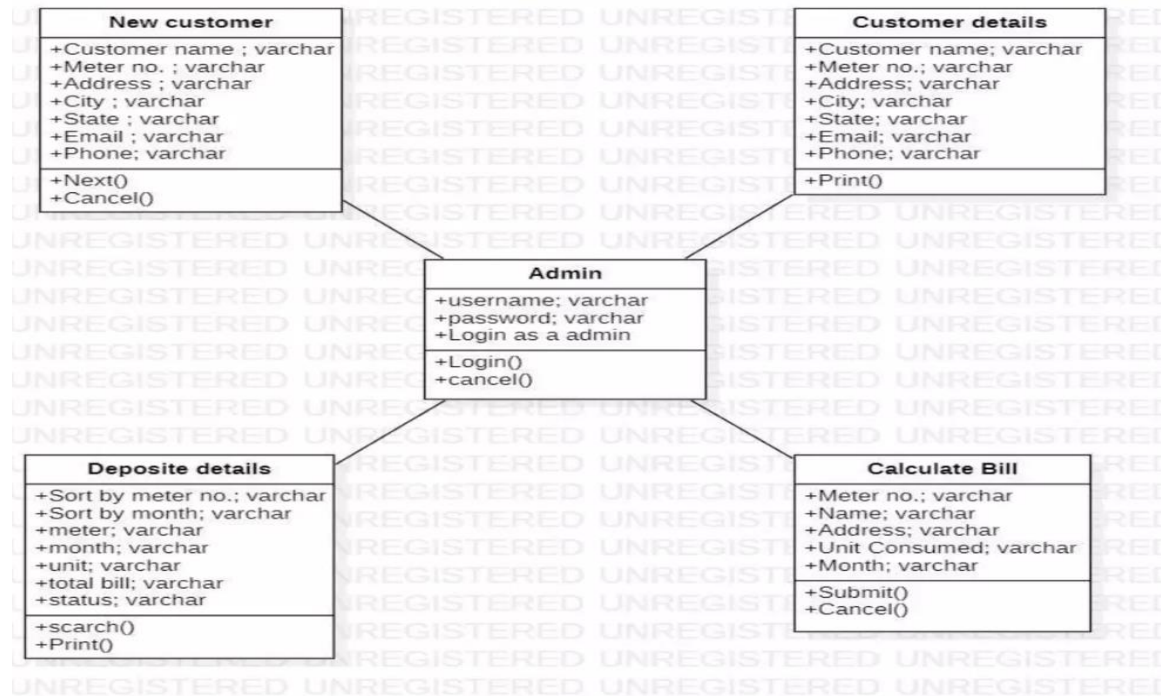


Fig4

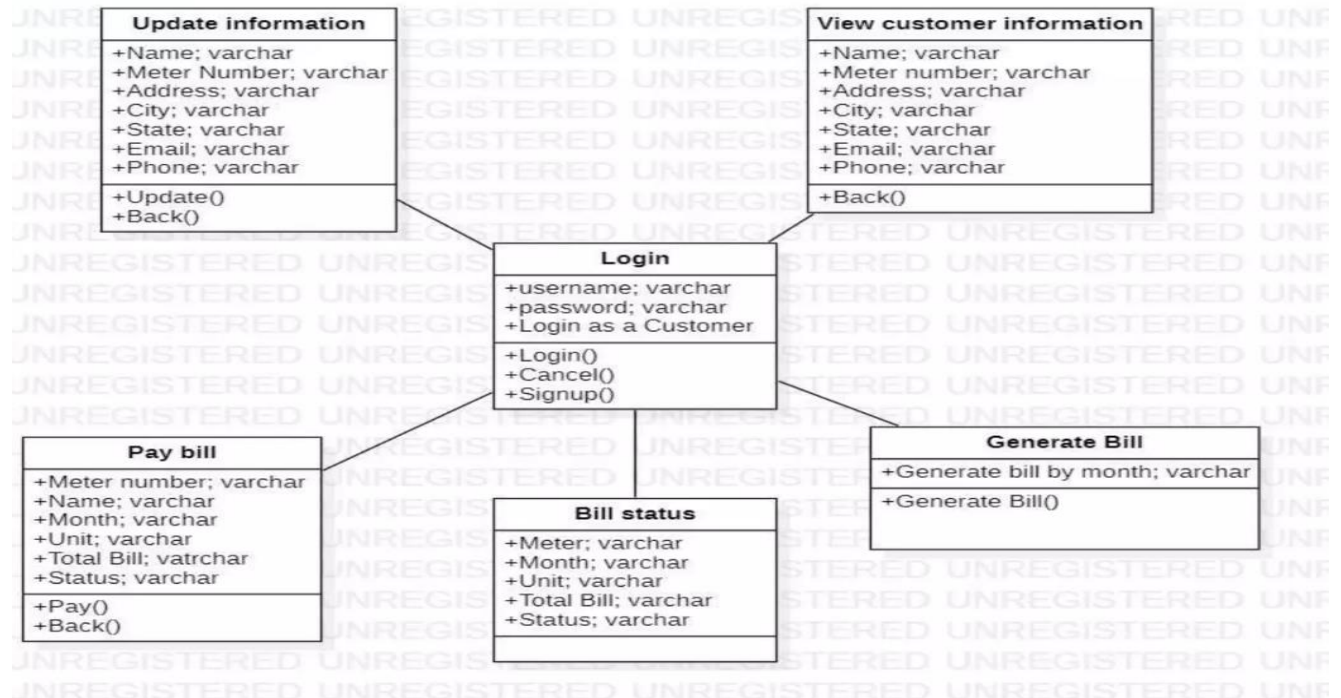
This UML diagram outlines a system for managing customer billing and payments through cards. A Customer is associated with multiple Cards, which can be either Debit or Credit cards, each having attributes like card number, balance, and limits. The Elect_Dept generates bills through the Bill_Section, where details such as bill ID, amount, and due dates are managed. The customer can make payments using their card, and the Bank_Consortium is responsible for checking card details and processing payments. The Verification process confirms the card's validity and limits, while a Mobile_Device sends SMS notifications to the customer regarding payment status. This system ensures that billing and card verification are handled efficiently, with clear communication to the customer via mobile.

6. CLASS DIAGRAM

Admin class diagram



Customer class diagram



This class diagram represents a customer billing system centered around user login and management. The Login class manages user authentication with attributes for username and password, providing methods to log in, sign up, or cancel. The Update Information and View Customer Information classes allow users to either update or view their personal details such as name, meter number, address, and contact information, with methods to update or navigate back. The Pay Bill class handles bill payment, capturing details like the meter number, month, units consumed, total bill, and payment status. The Bill Status class shows the current billing details, while the Generate Bill class is responsible for creating bills by month. All these classes work together to support customer account management and billing operations.

7. CODE IMPLEMENTATION

LAST BILLING DATA:-

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class LastBill extends JFrame implements ActionListener
{
    JLabel l1;
    JTextArea t1;
    JButton b1;
    Choice c1;
    JPanel p1;
    LastBill(){
        setSize(500,900);
        setLayout(new BorderLayout());

        p1 = new JPanel();

        l1 = new JLabel("Generate Bill");

        c1 = new Choice();

        c1.add("1001");
        c1.add("1002");
        c1.add("1003");
        c1.add("1004");
        c1.add("1005");
        c1.add("1006");
        c1.add("1007");
        c1.add("1008");
        c1.add("1009");
        c1.add("1010");
```

```

t1 = new JTextArea(50,15);
JScrollPane jsp = new JScrollPane(t1);
t1.setFont(new Font("Serif",Font.ITALIC,18));
b1 = new JButton("Generate Bill");
p1.add(l1);
p1.add(c1);
add(p1,"North");
add(jsp,"Center");
add(b1,"South");
b1.addActionListener(this);

setLocation(350,40);
}
public void actionPerformed(ActionEvent ae){
    try{
        conn c = new conn();

        ResultSet rs = c.executeQuery("select * from emp where
meter_number="+c1.getSelectedItem());

        if(rs.next()){
            t1.append("\n Customer Name:"+rs.getString("name"));
            t1.append("\n Meter Number: "+rs.getString("meter_number"));
            t1.append("\n Address: "+rs.getString("address"));
            t1.append("\n State: "+rs.getString("state"));
            t1.append("\n City: "+rs.getString("city"));
            t1.append("\n Email: "+rs.getString("email"));
            t1.append("\n Phone Number "+rs.getString("phone"));
            t1.append("\n-----");
            t1.append("\n");
        }

        t1.append("Details of the Last Bills\n\n\n");
        rs = c.executeQuery("select * from bill where meter_number="+c1.getSelectedItem());
        while(rs.next()){
            t1.append(" "+ rs.getString("month") + " " + rs.getString("amount") + "\n");
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

public static void main(String[] args){
    new LastBill().setVisible(true);
}
}

```

CALCULATE BILL:-

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class calculate_bill extends JFrame implements ActionListener
{
    JLabel l1,l2,l3,l4,l5;
    JTextField t1;
    Choice c1,c2;
    JButton b1,b2;
    JPanel p;
    calculate_bill(){

        p = new JPanel();
        p.setLayout(new GridLayout(4,2,30,30));
        p.setBackground(Color.WHITE);

        l1 = new JLabel("Calculate Electricity Bill");
        l2 = new JLabel("Meter No");
        l3 = new JLabel("Units Cosumed");
        l5 = new JLabel("Month");

        t1 = new JTextField();

        c1 = new Choice();
        c1.add("1001");
        c1.add("1002");
        c1.add("1003");
        c1.add("1004");
        c1.add("1005");
        c1.add("1006");
        c1.add("1007");
        c1.add("1008");
        c1.add("1009");
        c1.add("1010");

        c2 = new Choice();
        c2.add("January");
        c2.add("February");
        c2.add("March");
        c2.add("April");
        c2.add("May");
        c2.add("June");
        c2.add("July");
        c2.add("August");
        c2.add("September");
        c2.add("October");
        c2.add("November");
```

```

c2.add("December");

b1 = new JButton("Submit");
b2 = new JButton("Cancel");
b1.setBackground(Color.BLACK);
b1.setForeground(Color.WHITE);
b2.setBackground(Color.BLACK);
b2.setForeground(Color.WHITE);
ImageIcon i1 = new ImageIcon(ClassLoader.getResource("icon/hicon2.jpg"));
Image i2 = i1.getImage().getScaledInstance(180, 270, Image.SCALE_DEFAULT);
ImageIcon i3 = new ImageIcon(i2);
l4 = new JLabel(i3);
l1.setFont(new Font("Serif", Font.PLAIN, 26));
//Move the label to center
l1.setHorizontalAlignment(JLabel.CENTER);
p.add(l2);
p.add(c1);
p.add(l5);
p.add(c2);
p.add(l3);
p.add(t1);
p.add(b1);
p.add(b2);

setLayout(new BorderLayout(30,30));

add(l1, "North");
add(p, "Center");
add(l4, "West");

b1.addActionListener(this);
b2.addActionListener(this);

getContentPane().setBackground(Color.WHITE);
setSize(650,500);
setLocation(350,220);
}
public void actionPerformed(ActionEvent ae){
    String a = c1.getSelectedItem();
    String b = t1.getText();
    String c = c2.getSelectedItem();

    int p1 = Integer.parseInt(b);

    int p2 = p1*7;
    int p3 = p2+50+12+102+20+50;

    String q = "insert into bill values('"+a+"','"+c+"','"+b+"','"+p3+"')";

    try{

```

```

        conn c1 = new conn();
        c1.s.executeUpdate(q);
        JOptionPane.showMessageDialog(null,"Bill Updated");
    }catch(Exception aee){
        aee.printStackTrace();
    }

}

public static void main(String[] args){
    new calculate_bill().setVisible(true);
}
}

CUSTOMER DETAILS:-
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class customer_details extends JFrame implements ActionListener{

    JTable t1;
    JButton b1;
    String x[] = {"Emp Name","Meter No","Address","State","City","Email","Phone"};
    String y[][] = new String[20][8];
    int i=0, j=0;
    customer_details(){
        super("Customer Details");
        setSize(1200,650);
        setLocation(200,200);

        try{
            conn c1 = new conn();
            String s1 = "select * from emp";
            ResultSet rs = c1.s.executeQuery(s1);
            while(rs.next()){
                y[i][j++]=rs.getString("name");
                y[i][j++]=rs.getString("meter_number");
                y[i][j++]=rs.getString("address");
                y[i][j++]=rs.getString("state");
                y[i][j++]=rs.getString("city");
                y[i][j++]=rs.getString("email");
                y[i][j++]=rs.getString("phone");
                i++;
                j=0;
            }
            t1 = new JTable(y,x);

```

```

    }catch(Exception e){
        e.printStackTrace();
    }

    b1 = new JButton("Print");
    add(b1,"South");
    JScrollPane sp = new JScrollPane(t1);
    add(sp);
    b1.addActionListener(this);
}
public void actionPerformed(ActionEvent ae){
    try{
        t1.print();
    }catch(Exception e){}
}
public static void main(String[] args){
    new customer_details().setVisible(true);
}
}

```

BILL GENERATION:-

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class generate_bill extends JFrame implements ActionListener{
    JLabel l1;
    JTextArea t1;
    JButton b1;
    Choice c1,c2;
    JPanel p1;
    generate_bill(){
        setSize(500,900);
        setLayout(new BorderLayout());

        p1 = new JPanel();

        l1 = new JLabel("Generate Bill");

        c1 = new Choice();
        c2 = new Choice();

        c1.add("1001");
        c1.add("1002");
        c1.add("1003");
        c1.add("1004");
        c1.add("1005");
        c1.add("1006");
        c1.add("1007");
    }
}

```



```

c1.add("1008");
c1.add("1009");
c1.add("1010");

c2.add("January");
c2.add("February");
c2.add("March");
c2.add("April");
c2.add("May");
c2.add("June");
c2.add("July");
c2.add("August");
c2.add("September");
c2.add("October");
c2.add("November");
c2.add("December");
t1 = new JTextArea(50,15);
JScrollPane jsp = new JScrollPane(t1);
t1.setFont(new Font("Serif",Font.ITALIC,18));

b1 = new JButton("Generate Bill");
p1.add(l1);
p1.add(c1);
p1.add(c2);
add(p1,"North");

add(jsp,"Center");
add(b1,"South");

b1.addActionListener(this);

setLocation(350,40);
}
public void actionPerformed(ActionEvent ae){
    try{
        conn c = new conn();

        String month = c2.getSelectedItem();
        t1.setText("\tReliance Power Limited\nELECTRICITY BILL FOR THE MONTH OF
"+month+" ,2018\n\n\n");

        ResultSet rs = c.s.executeQuery("select * from emp where
meter_number="+c1.getSelectedItem());

        if(rs.next()){
            t1.append("\n Customer Name:"+rs.getString("name"));
            t1.append("\n Meter Number: "+rs.getString("meter_number"));
            t1.append("\n Address: "+rs.getString("address"));
            t1.append("\n State: "+rs.getString("state"));
            t1.append("\n City: "+rs.getString("city"));
        }
    }
}

```

```

        t1.append("\n   Email:           "+rs.getString("email"));
        t1.append("\n   Phone Number  "+rs.getString("phone"));
        t1.append("\n-----");
        t1.append("\n");
    }

    rs = c.s.executeQuery("select * from tax");

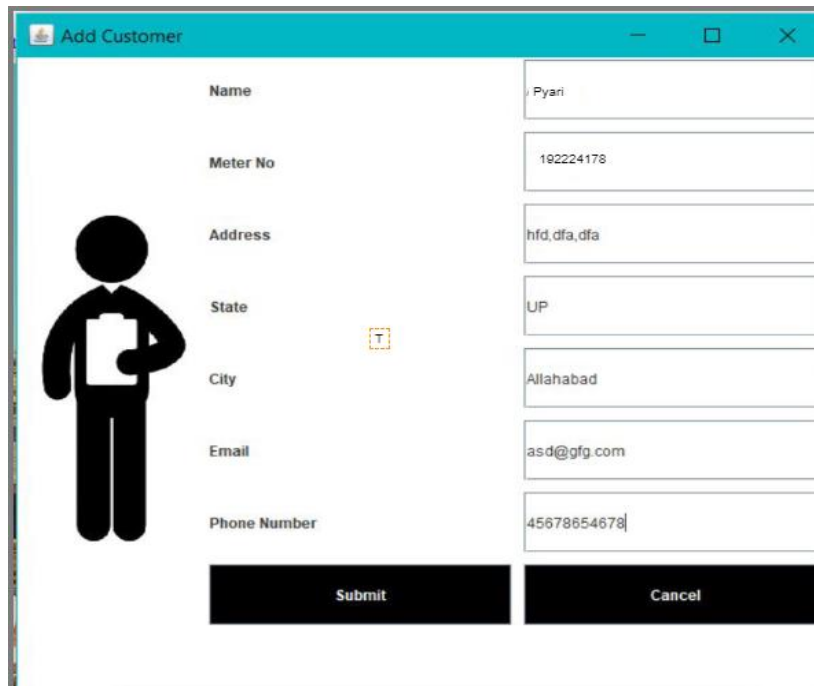
    if(rs.next()){
        t1.append("\n   Meter Location:"+rs.getString("meter_location"));
        t1.append("\n   Meter Type:    "+rs.getString("meter_type"));
        t1.append("\n   Phase Code:   "+rs.getString("phase_code"));
        t1.append("\n   Bill Type:    "+rs.getString("bill_type"));
        t1.append("\n   Days:        "+rs.getString("days"));
        t1.append("\n");
        t1.append("-----");
        t1.append("\n\n");
        t1.append("\n   Meter Rent:\t\t"+rs.getString("meter_rent"));
        t1.append("\n   MCB Rent:  \t\t"+rs.getString("mcb_rent"));
        t1.append("\n   Service Tax:\t"+rs.getString("service_rent"));
        t1.append("\n   GST@9%:\t\t"+rs.getString("gst"));
        t1.append("\n");
    }
    rs = c.s.executeQuery("select * from bill where meter_number="+c1.getSelectedItem());

    if(rs.next()){
        t1.append("\n   Current Month :\t"+rs.getString("month"));
        t1.append("\n   Units Consumed:\t"+rs.getString("units"));
        t1.append("\n   Total Charges :\t"+rs.getString("amount"));
        t1.append("\n-----");
        t1.append("\n   TOTAL PAYABLE :\t"+rs.getString("amount"));
    }
}
}catch(Exception e){
    e.printStackTrace();
} }

public static void main(String[] args){
    new generate_bill().setVisible(true);
}
}

```

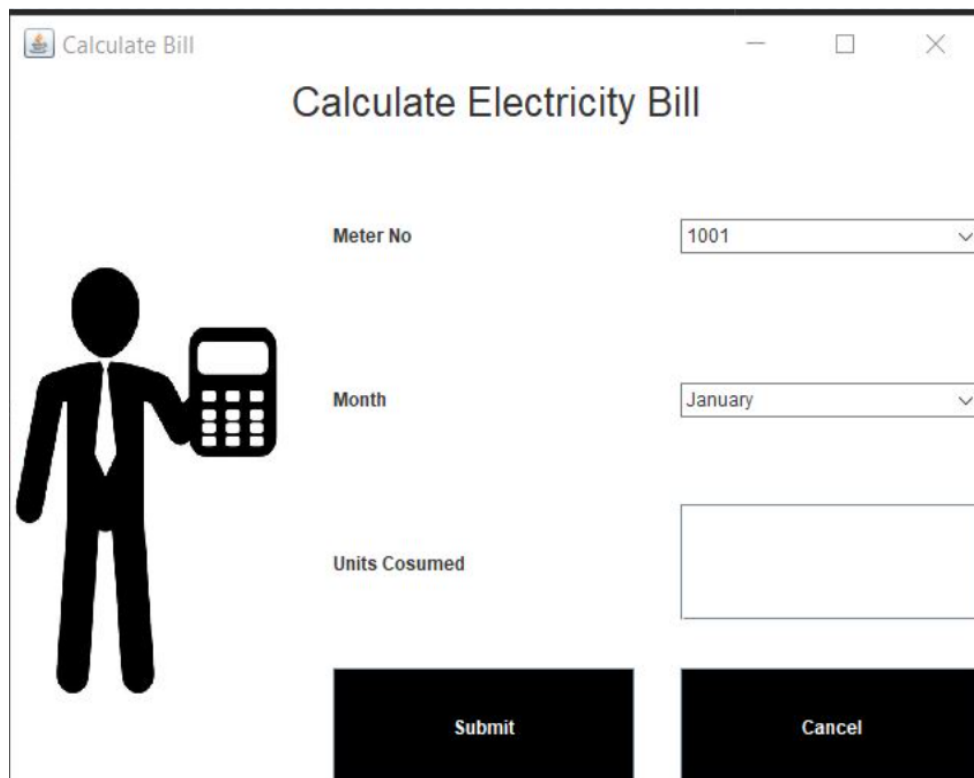
8. OUTPUT SCREENSHOT



The screenshot shows a window titled "Add Customer" with a teal header bar. On the left is a black silhouette of a person holding a document. The form contains the following fields and values:

Field	Value
Name	Pyari
Meter No	192224178
Address	hfd, dfa, dfa
State	UP
City	Allahabad
Email	asd@gfg.com
Phone Number	45678854678

At the bottom are two black buttons: "Submit" and "Cancel". A small yellow box with the letter "T" is positioned between the "State" and "City" labels.



The screenshot shows a window titled "Calculate Bill" with a white header bar. On the left is a black silhouette of a person holding a calculator. The form contains the following fields and values:

Field	Value
Meter No	1001
Month	January
Units Cosumed	

At the bottom are two black buttons: "Submit" and "Cancel".

Customer Details						
Emp Name	Meter No	Address	State	City	Email	Phone
Adarsh	1234	5 h malviya road	UP	Allahabad	xyz@gfg.com	3453245432
vishal	27482	743 aedhad	MP	Nemuch	wer@fgfd.com	6789938433
ansh	678233	nahs jd h 2e w	Delhi	Delhi	hjk@hj.com	456787654
Adarsh	5678987	hfd,dfa,dfa	UP	Allahabad	asd@gfg.com	45678654678
Print						

Generate Bill	
Generate Bill	1001 January
Reliance Power Limited ELECTRICITY BILL FOR THE MONTH OF January ,2018	
Customer Name: Adarsh Meter Number: 1001 Address: 5 H Malviya Road State: UP City: Allahabad Email: Ada@gmail.com Phone Number 567898765	
Meter Location: UP Meter Type: Electric Phase Code: 123 Bill Type: Ebill Days: 10	
Meter Rent: 1234 MCB Rent: 21 Service Tax: 321 GST@9%: 500	
Current Month : January Units Consumed: 45526 Total Charges : 318916	
TOTAL PAYABLE : 318916	
Generate Bill	

9. CONCLUSION

In conclusion, the Online Electricity Billing System outlined in the ERD provides an efficient framework for managing the billing process. By linking electricity consumers to their respective regions and branches, and tracking their consumption through unique identifiers, the system ensures accurate billing and streamlined payment management. The inclusion of rate differentiation and payment status tracking allows for dynamic billing based on location and consumption patterns. This database structure enhances transparency and reliability, ensuring that both the electricity providers and consumers have a clear and efficient means of monitoring usage and payments. Overall, the system simplifies the complex process of electricity billing, making it scalable and user-friendly.

the Online Electricity Billing System provides a comprehensive and scalable solution for managing electricity consumption and billing processes in an efficient and transparent manner. The system's structure, as represented in the ER diagram, captures key entities such as the Electricity Supply Company, its various Branches, and the individual Consumers with their unique identifiers and meter numbers. By linking consumers to specific regions and tracking their monthly electricity usage, the system ensures that consumption is recorded accurately and billed accordingly. The dynamic relationship between consumption records, regional rates, and payment status (whether a bill is paid or unpaid) adds flexibility, enabling the system to accommodate varying tariffs or policies depending on location and usage patterns.

Moreover, the system's design allows for real-time updates, ensuring that users can easily monitor their consumption, check their outstanding bills, and make payments online. This not only enhances customer experience but also reduces administrative overhead for electricity providers. The clear connection between the Consumption Records and Billing Rates ensures that billing policies are applied fairly and consistently, while also allowing for easy integration of different pricing models, such as peak and off-peak rates, promotional discounts, or surcharges.

Overall, the system is designed to be user-friendly, reliable, and adaptable, making it suitable for managing large-scale operations across multiple regions. Its robust database structure allows electricity providers to efficiently track and manage consumer data, consumption records, and payment statuses, while offering a seamless interface for customers to interact with. This architecture ultimately simplifies the complex process of electricity billing, ensuring accuracy, transparency, and ease of use for all stakeholders involved.

10. REFERENCES

- 1 Deloitte Insights, "2024 Power and Utilities Industry Outlook," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].
- 2 Mordor Intelligence, "Utility Billing Software Market Overview," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024]. MaxBill, "Electric Billing Software," 2024.
- 3 MaxBill, "Electric Billing Software," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].
- 4 Oracle, "Billing and Revenue Management System," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].
- 5 VertexOne, "Customer-to-Cash Technology Platform," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].
- 6 Harris Computer Systems, "Utility Billing Solutions," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024]
- 7 CUSI, "Utility Billing Solutions," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].
- 8 Tridens, "Monetization," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].
- 9 MuniBilling, "Core Platform," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].
- 10 Emerging Trends in Utility Billing Software, "Analysis of Mergers and Acquisitions," 2024. [Online]. Available: [insert link]. [Accessed: Sep. 26, 2024].