

INTRODUCTION

1.1: Project Overview:

The **Electricity Management System** is a Java-based application designed to streamline and enhance the process of managing electricity usage, billing, and distribution. It provides an efficient platform for monitoring electricity consumption, generating bills, and managing payments, significantly reducing manual effort and minimizing errors. The system includes features for user registration, meter readings, and bill payment, as well as secure handling of service requests and disconnection/reconnection processes. A user-friendly graphical interface (built using Swing or JavaFX) ensures seamless navigation for both consumers and administrators. The backend integrates with relational databases like MySQL or PostgreSQL using JDBC, providing secure and reliable data storage and retrieval.

To ensure data integrity and confidentiality, the system incorporates robust security measures such as password encryption and role-based access control. Its scalable architecture allows for future enhancements, including mobile application support, integration with smart meters, and advanced features like energy usage analytics and personalized consumption insights. This comprehensive solution caters to the modern needs of utility management, offering efficiency, accuracy, and convenience to both electricity providers and consumers.

1.2: Objective

The primary objective of the **Electricity Management System** is to develop a secure, efficient, and user-friendly platform to manage electricity billing, payments, and consumption tracking. The system aims to automate tasks such as meter readings, bill generation, and payment processing, thereby reducing manual intervention and minimizing errors. It seeks to enhance the consumer experience by providing seamless access to billing information, secure payment options, and timely notifications, while empowering administrators with tools for effective monitoring, reporting, and data analysis.

Additionally, the system is designed to ensure data integrity, confidentiality, and scalability, making it adaptable to evolving requirements of the energy sector. By leveraging Java's robust features, the system delivers a reliable solution that meets the operational and security needs of modern electricity management operations.

MODULES:

1. Main Class:

The Main class serves as the entry point of the Electricity Management System. It initializes the system and provides a menu for users to log in, sign up, or access system functionalities. This class invokes other modules such as Login or SignUp to guide users through the workflow.

2. Login Class:

The Login class is responsible for authenticating users. It collects the username and password, verifies them against stored credentials in the database, and grants access upon successful authentication. If the credentials are incorrect, it prompts the user to try again. New users are redirected to the SignUp process if needed.

3. Connection Class:

The Connection class manages the database connection. It establishes a secure link between the application and the database (e.g., MySQL or PostgreSQL) and provides methods for executing queries such as validating login credentials, storing user data, or retrieving billing information.

4. SignUp Class:

The SignUp class handles the registration process for new users. It collects details such as name, contact information, and account setup preferences, securely storing them in the database. The class ensures all required fields are filled and validates the input before submission.

5. NewCustomer Class:

The NewCustomer class extends the registration process by gathering additional details like address, meter type, and connection preferences. This information is securely stored in the database and helps personalize user profiles for efficient system operation.

6. MeterInfo Class:

The MeterInfo class manages information about electricity meters assigned to users. It retrieves details such as meter ID, type, and usage history from the database. The class plays a critical role in ensuring accurate monitoring of electricity consumption.

7. CalculateBill Class:

The CalculateBill class computes the electricity bill based on the user's consumption data. It applies the applicable tariff rates, taxes, and surcharges to generate an accurate bill amount. The class ensures transparency by providing a detailed breakdown of charges.

8. GenerateBill Class:

The GenerateBill class generates an official electricity bill once calculations are completed. The bill includes details such as user information, meter readings, consumption, and total payable amount. It can be saved or printed for user convenience.

9. PayBill Class:

The PayBill class facilitates secure bill payments. It supports multiple payment options such as credit cards, debit cards, and online banking. The class validates payment details, processes the transaction securely, and provides a confirmation receipt.

10. ProfileManagement Class:

The ProfileManagement class allows users to update their account information, including contact details and preferences. It ensures that any changes are validated and securely updated in the database, enhancing personalization and user convenience.

11. BillingHistory Class:

The BillingHistory class provides users with an overview of their previous bills and payments. It retrieves historical data, such as bill amounts, payment dates, and consumption, helping users monitor and analyze their usage trends.

12. ServiceRequests Class:

The ServiceRequests class enables users to make requests such as meter upgrades, connection changes, or disconnection/reconnection services. It

collects request details, validates them, and forwards them to the appropriate administrator for processing.

13. Notifications Class:

The Notifications class keeps users informed about important updates such as upcoming due dates, service outages, or special offers. It retrieves notifications from the database and displays them to users in a timely manner.

14. OffersAndDiscounts Class:

The OffersAndDiscounts class showcases current discounts or promotions available, such as incentives for timely payments or energy-saving rewards. It retrieves these details from the database and displays them to help users make the most of available benefits.

CHAPTER 2: SYSTEM DESIGN

System Design is a critical phase in software development where the architecture and components of the system are structured and detailed. This chapter focuses on designing the **Electricity Management System**, outlining its structure, components, data flow, user interfaces, and interaction with databases.

2.1. System Architecture

The **Electricity Management System** follows a **Client-Server Architecture**, where:

1. **Client Side:** Users interact with the system through a user interface (UI), such as login screens, bill payment pages, and consumption details.
2. **Server Side:** The server processes requests, interacts with the database, performs calculations (e.g., bill generation), and sends responses back to the client.
3. **Database Layer:** Stores all user, meter, and billing data securely and ensures its integrity.

The architecture is divided into three main layers:

1. Presentation Layer:

- User-facing interface, displaying login screens, bill details, meter information, and profile management options.
- Provides intuitive navigation for users to view and update information.

2. Business Logic Layer:

- Contains the core functionalities such as processing meter data, calculating bills, managing customer profiles, and validating payments.
- Ensures smooth coordination between the client and the database layers.

3. Data Layer:

- Manages the database where user details, meter information, tax rates, and billing records are stored.
- Implements queries to retrieve, update, and delete data as per business requirements.

2.2. Database Design

The database is designed to efficiently store and manage all data related to users, meters, and billing. Below is the schema for the **Electricity Management System**:

Tables

- **Signup Table:**

Stores user authentication details.

Columns:

- meter_no (Primary Key)
- username
- name
- password
- usertype (e.g., Admin/User)

- **New_Customer Table:**

Stores customer details such as address and contact information.

Columns:

- name
- meter_no (Primary Key, references Signup.meter_no)
- address
- city
- state
- email
- phone_no

- **Meter_Info Table:**

Contains data about each electricity meter.

Columns:

- meter_number (Primary Key, references Signup.meter_no)
- meter_location
- meter_type
- phase_code
- bill_type (e.g., Residential/Commercial)
- days (e.g., number of days billed in a cycle)

- **Tax Table:**

Stores tax and fixed charges applicable to billing.

Columns:

- cost_per_unit
- meter_rent
- service_charge
- service_tax
- swachh_bharat
- fixed_tax

- **Bill Table:**

Tracks billing information for each customer.

Columns:

- meter_no (Primary Key, references Signup.meter_no)
- month
- unit (Units of electricity consumed)
- total_bill (Computed based on units and taxes)
- status (e.g., Paid/Pending)

Relationships

1. Signup ↔ New_Customer:

- A one-to-one relationship between user authentication and customer details, linked by meter_no.

2. Signup ↔ Meter_Info:

- A one-to-one relationship between a user and their meter, linked by meter_no.

3. Meter_Info ↔ Bill:

- A one-to-many relationship where a single meter can have multiple bills over time.

4. Tax ↔ Bill:

- Tax rates and charges are applied dynamically during bill calculations.

Data Flow

1. User Authentication:

- Users log in via the **Signup Table**, which authenticates their credentials and identifies their user type.

2. Meter Management:

- The **Meter_Info Table** stores essential data like meter location and type, retrieved for calculating usage and bill details.

3. Billing:

- The **Bill Table** stores monthly usage, bill amounts, and payment status. Taxes and charges from the **Tax Table** are applied dynamically based on consumption data.

4. Updates:

- Users can update personal details in the **New_Customer Table**, while admins can update **Meter_Info** and **Tax Table** entries.

This design ensures a robust, scalable, and efficient system for managing electricity services, facilitating seamless interaction between users, administrators, and the database.

2.3. System Components

User Interface (UI):

The UI provides a user-friendly environment for interaction with the Electricity Management System. Key interfaces include:

- **Login Screen:** A secure interface for user authentication.
- **Dashboard:** Displays user information, recent bills, and quick links to key features.
- **Meter Information Screen:** Shows details about the user's electricity meter, including location and type.
- **Bill Calculation Screen:** Displays monthly electricity usage, cost per unit, and total payable amount.
- **Payment Screen:** Enables secure bill payment.
- **Billing History Screen:** Displays a record of past and pending bills for the user.

Business Logic:

This layer implements core functionalities and system workflows, such as:

- **Login/Authentication:** Verifies user credentials and securely manages login sessions.
- **Meter Management:** Handles the retrieval and display of meter details like type, location, and phase.
- **Bill Calculation:** Processes electricity usage data, applies taxes, and generates an accurate bill.
- **Payment Processing:** Validates payment methods and secures transaction completion.

- **Profile and Data Management:** Allows users to update their personal details and access relevant information.

Database Interaction:

Efficient database operations are critical for system functionality.

- **CRUD Operations:** Create, Read, Update, and Delete operations on user accounts, meter information, and billing data.
- **Query Execution:** SQL queries retrieve details such as meter data, usage, tax rates, and billing history.

2.4. Use Case Diagrams

Use case diagrams visually represent the system's core functionalities and interactions. Key use cases include:

- **User Authentication:** Log in with credentials.
- **View Meter Information:** Access details about the user's electricity meter.
- **Bill Calculation:** Calculate electricity costs based on usage.
- **Bill Payment:** Securely make payments for electricity bills.
- **View Billing History:** Check previous and pending bills.

2.5. Data Flow Diagram (DFD)

Level 1 DFD:

- The user logs in by providing credentials, which the system validates against the database.
- The user views meter details or billing information retrieved from the database.
- When the user processes payment, the system updates the billing and payment status in the database.

Level 2 DFD:

- **Meter Information Retrieval:** User requests meter details, and the system retrieves and displays them.
- **Bill Calculation:** The system fetches electricity usage, applies applicable taxes, and calculates the total bill.
- **Payment Processing:** User enters payment details, and the system validates and records the transaction.

2.6. Sequence Diagrams

Login Sequence:

1. User enters login credentials.
2. The system validates the credentials against the database.
3. Upon success, the user is redirected to the dashboard.

Bill Payment Sequence:

1. User selects a pending bill from the dashboard.
2. The system retrieves bill details and displays them.
3. User enters payment details and confirms payment.
4. The system processes the payment, updates the bill status, and provides a confirmation receipt.

2.7. Security Design

Security is critical in the Electricity Management System. Key aspects include:

- **Authentication:** Ensures only authorized users can access the system using secure login mechanisms.
- **Encryption:** Stores sensitive information, such as passwords, in an encrypted format.
- **Session Management:** Manages active user sessions securely to prevent unauthorized access.
- **Payment Security:** Implements secure payment gateways to protect transaction details.

2.8. System Flow

The system flow explains how the various components interact:

1. **User Login:** The system validates the user's credentials and grants access to the dashboard.
2. **Dashboard:** Users can view meter information, recent bills, and usage details.
3. **Bill Calculation:** The system calculates electricity bills based on usage and applicable taxes.
4. **Bill Payment:** Users securely pay pending bills via the payment interface.
5. **Billing History:** Users can review past and pending bills.
6. **Logout:** Users log out, securely closing their session.

This system flow ensures a seamless and efficient experience for both users and administrators.

CHAPTER 3: FUNCTIONAL REQUIREMENTS

Functional requirements define the specific behaviors, functions, and processes that the **Electricity Management System** must support. These requirements describe the system's interactions with users and other components, focusing on the desired capabilities and features needed to meet its objectives.

3.1. User Authentication

Description:

The system must support secure user authentication to ensure that only authorized individuals can access their accounts.

Functional Requirements:

- Users must log in using a valid **username** and **password**.
- The system must authenticate the user based on the provided credentials.
- Strong password policies must be enforced (e.g., minimum length, combination of characters).
- Users should be able to reset their password in case they forget it.
- Upon successful login, users should be redirected to their **dashboard**.
- The system should lock accounts after multiple failed login attempts, requiring secure intervention for unlocking.

3.2. Customer and Meter Management

Description:

The system must allow users to view and manage customer information and meter details.

Functional Requirements:

- Users must be able to view customer profiles, including **name**, **address**, **city**, **state**, **email**, and **phone number**.
- The system should display meter details, such as **meter number**, **location**, **type**, **phase code**, and **bill type**.

- Administrators must have the ability to update customer and meter information.

3.3. Bill Calculation

Description:

The system must calculate electricity bills based on usage, taxes, and applicable charges.

Functional Requirements:

- The system must retrieve usage data (units consumed) for each meter.
- It must apply predefined rates, taxes, and charges (e.g., cost per unit, service charges).
- The system must generate a summary of the bill, including **meter number, month, units consumed, tax breakdown, and total bill**.
- Users should be able to view the breakdown of charges before confirming the bill.

3.4. Bill Payment

Description:

The system must support secure and seamless payment processing for electricity bills.

Functional Requirements:

- Users should be able to choose from multiple payment options, such as **credit/debit cards, net banking, or digital wallets**.
- The system must validate payment details before processing the transaction.
- Upon successful payment, users must receive a confirmation of the transaction.
- Failed payments must not mark the bill as paid, and the status should remain pending.

3.5. Billing History

Description:

The system must allow users to view past and pending bills.

Functional Requirements:

- Users must be able to view a list of bills, including **meter number**, **month**, **units consumed**, and **payment status**.
- The system should allow users to filter bills by date or status (**paid/pending**).
- Users must be able to download a copy of previous bills for their records.

3.6. Error Handling and Validation

Description:

The system must validate inputs and handle errors gracefully.

Functional Requirements:

- The system must validate user inputs, such as **meter numbers**, **payment details**, and **customer data**.
- The system should display clear error messages for invalid inputs (e.g., "Invalid meter number").
- Unexpected errors, such as payment gateway failures, must be handled gracefully, and users should be informed of the issue.

3.7. Reporting and Analytics

Description:

The system must provide reporting features for administrators to monitor usage, billing, and revenue.

Functional Requirements:

- Administrators should be able to generate reports on customer usage and payments.
- The system must support daily, monthly, and yearly revenue reports.

- Reports must include details such as total units consumed, total revenue, and outstanding payments.
- Reports should be downloadable in formats like CSV or PDF.

3.8. Backup and Recovery

Description:

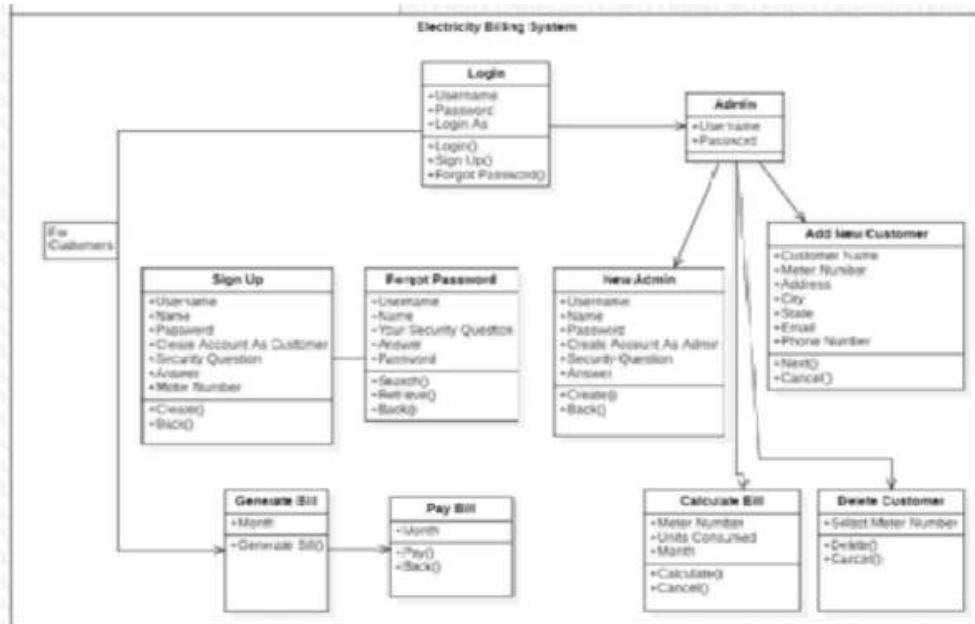
The system must support regular backups and ensure data recovery in case of failures.

Functional Requirements:

- The system must perform regular backups of customer data, meter information, and billing records.
- In the event of a system failure, data should be restored from the most recent backup with minimal downtime.
- The backup process must ensure transactional integrity to avoid discrepancies.

ER Diagram

P



Program Code

Code For Electricity Management System

Login Page:

```
package electricity.billing.system;

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;

public class Login extends JFrame implements ActionListener {
    JTextField userText,passwordText;
    Choice loginChoice;

    JButton loginButton, cancelButton, signupButton;

    Login(){
        super("Login");
        getContentPane().setBackground(Color.white);
        JLabel username = new JLabel("UserName");
        username.setBounds(300,60,100,20);
        add(username);

        userText = new JTextField();
        userText.setBounds(400,60,150,20);
        add(userText);

        JLabel password = new JLabel("Password");
        password.setBounds(300,100,100,20);
        add(password);

        passwordText = new JTextField();
        passwordText.setBounds(400,100,150,20);
        add(passwordText);
```

```
JLabel login = new JLabel("Loggin In As");  
login.setBounds(300,140,100,20);  
add(login);
```

```
loginChoice = new Choice();  
loginChoice.add("Admin");  
loginChoice.add("Customer");  
loginChoice.setBounds(400,140,150,20);  
add(loginChoice);
```

```
loginButton = new JButton("Login");  
loginButton.setBounds(330,180,100,20);  
loginButton.addActionListener(this);  
add(loginButton);
```

```
cancelButton = new JButton("Cancel");  
cancelButton.setBounds(460,180,100,20);  
cancelButton.addActionListener(this);  
add(cancelButton);
```

```
signupButton = new JButton("Signup");  
signupButton.setBounds(400,215,100,20);  
signupButton.addActionListener(this);  
add(signupButton);
```

```
ImageIcon profileOne = new  
ImageIcon(ClassLoader.getResource("icon/profile.png"));  
Image profileTow =  
profileOne.getImage().getScaledInstance(250,250,Image.SCALE_DEFAULT);  
ImageIcon fprofileOne = new ImageIcon(profileTow);  
JLabel profileLable = new JLabel(fprofileOne);  
profileLable.setBounds(5,5,250,250);  
add(profileLable);
```

```
setSize(640,300);  
setLocation(400,200);  
setLayout(null);
```

```

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==loginButton){
            String susername = userText.getText();
            String spassword = passwordText.getText();
            String suser = loginChoice.getSelectedItem();

            try{
                database d = new database();
                String queryy = "select * from Signup where username =
"+susername+" and password = "+spassword+" and usertype='"+suser+"'";
                ResultSet resultSet = d.statement.executeQuery(queryy);

                if (resultSet.next()){
                    String meter = resultSet.getString("meter_no");
                    setVisible(false);
                    new main_class(suser,meter);
                }else {
                    JOptionPane.showMessageDialog(null ,"Invalid Login");
                }

            }catch (Exception E){
                E.printStackTrace();
            }
        } else if (e.getSource()==cancelButton) {
            setVisible(false);
        } else if (e.getSource()==signupButton) {
            setVisible(false);
            new Signup();
        }

    }

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

```

SignUp Page:

```
package electricity.billing.system;

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

public class Signup extends JFrame implements ActionListener {
    Choice loginASCho;
    TextField meterText, EmployerText, userNameText, nameText, passwordText;
    JButton create, back;
    Signup() {
        super("Signup Page");
        getContentPane().setBackground(new Color(168,203,255));

        JLabel createAs = new JLabel("Create Account As");
        createAs.setBounds(30,50,125,20);
        add(createAs);

        loginASCho = new Choice();
        loginASCho.add("Admin");
        loginASCho.add("Customer");
        loginASCho.setBounds(170,50,120,20);
        add(loginASCho);

        JLabel meterNo = new JLabel("Meter Number");
        meterNo.setBounds(30,100,125,20);
        meterNo.setVisible(false);
        add(meterNo);

        meterText = new TextField();
        meterText.setBounds(170,100,125,20);
        meterText.setVisible(false);
        add(meterText);

        JLabel Employer = new JLabel("Employer ID");
        Employer.setBounds(30,100,125,20);
```

```

Employer.setVisible(true);
add(Employer);

EmployerText = new TextField();
EmployerText.setBounds(170,100,125,20);
EmployerText.setVisible(true);
add(EmployerText);

JLabel userName = new JLabel("UserName");
userName.setBounds(30,140,125,20);
add(userName);

userNameText = new TextField();
userNameText.setBounds(170,140,125,20);
add(userNameText);

JLabel name = new JLabel("Name");
name.setBounds(30,180,125,20);
add(name);

nameText = new TextField("");
nameText.setBounds(170,180,125,20);
add(nameText);

meterText.addFocusListener(new FocusListener() {
    @Override
    public void focusGained(FocusEvent e) {

    }

    @Override
    public void focusLost(FocusEvent e) {
        try {
            database c = new database();
            ResultSet resultSet = c.statement.executeQuery("select * from
Signup where meter_no = '"+meterText.getText()+"'");
            if (resultSet.next()){
                nameText.setText(resultSet.getString("name"));
            }
        }
    }
});

```

```

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

JLabel password = new JLabel("Password");
password.setBounds(30,220,125,20);
add(password);

passwordText = new TextField();
passwordText.setBounds(170,220,125,20);
add(passwordText);

loginAScho.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        String user = loginAScho.getSelectedItem();
        if (user.equals("Customer")){
            Employer.setVisible(false);
            nameText.setEditable(false);
            EmployerText.setVisible(false);
            meterNo.setVisible(true);
            meterText.setVisible(true);
        } else {
            Employer.setVisible(true);
            EmployerText.setVisible(true);
            meterNo.setVisible(false);
            meterText.setVisible(false);
        }
    }
});

create = new JButton("Create");
create.setBackground(new Color(66,127,219));
create.setForeground(Color.black);
create.setBounds(50,285,100,25);

```



```

create.addActionListener(this);
add(create);

back = new JButton("Back");
back.setBackground(new Color(66,127,219));
back.setForeground(Color.black);
back.setBounds(180,285,100,25);
back.addActionListener(this);
add(back);

ImageIcon boyIcon = new
ImageIcon(ClassLoader.getResource("icon/boy.png"));
Image boyImg =
boyIcon.getImage().getScaledInstance(250,250,Image.SCALE_DEFAULT);
ImageIcon boyIcon2 = new ImageIcon(boyImg);
JLabel boyLabel = new JLabel(boyIcon2);
boyLabel.setBounds(320,30,250,250);
add(boyLabel);

setSize(600,380);
setLocation(500,200);
setLayout(null);
setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == create) {
        String sloginAs = loginASCho.getSelectedItem();
        String susername = userNameText.getText();
        String sname = nameText.getText();
        String spassword = passwordText.getText();
        String smeter = meterText.getText();
        try {
            database c = new database();
            String query = null;
            if (sloginAs.equals("Admin")) {
                query = "insert into signup value('" + smeter + "', '" + susername +
                    "', '" + sname + "', '" + spassword + "', '" + sloginAs + "')";
            } else {

```

```

        query = "update signup set username = '"+susername+"', password
= '"+spassword+"', usertype = '"+sloginAs+" where meter_no = '"+smeter+""";
    }
    c.statement.executeUpdate(query);

    JOptionPane.showMessageDialog(null,"Account Created");
    setVisible(false);
    new Login();

    } catch (Exception E){
        E.printStackTrace();
    }
    } else if (e.getSource()==back) {
        setVisible(false);
        new Login();
    }
}

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

```

Main Page:

```

package electricity.billing.system;

import net.proteanit.sql.DbUtils;

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

public class bill_details extends JFrame {
    String meter;
    bill_details(String meter){
        this.meter =meter;
        setSize(700,650);
    }
}

```

```

        setLocation(400,150);
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JTable table = new JTable();

        try{
            database c = new database();
            String query_bill = "select * from bill where meter_no = '"+meter+"'";
            ResultSet resultSet = c.statement.executeQuery(query_bill);
            table.setModel(DbUtils.resultSetToTableModel(resultSet));
        }catch (Exception e){
            e.printStackTrace();
        }
        JScrollPane sp = new JScrollPane(table);
        sp.setBounds(0,0,700,650);
        add(sp);

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

```

NewCustomer Page:

```

package electricity.billing.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;

public class newCustomer extends JFrame implements ActionListener {

```

```

JLabel heading ,meternumText,customerName,meterNum,
address,city,state,email,phone;
JButton next, cancel;
TextField nameText, addressText, cityText, stateText ,emailText,phoneText;
newCustomer(){
    super("New Customer");
    setSize(700,500);
    setLocation(400,200);

    JPanel panel = new JPanel();
    panel.setLayout(null);
    panel.setBackground(new Color(252,186,3));
    add(panel);

    heading = new JLabel("New Customer");
    heading.setBounds(180,10,200,20);
    heading.setFont(new Font("Tahoma",Font.BOLD,20));
    panel.add(heading);

    customerName = new JLabel("New Customer");
    customerName.setBounds(50,80,100,20);
    panel.add(customerName);

    nameText = new TextField();
    nameText.setBounds(180,80,150,20);
    panel.add(nameText);

    meterNum = new JLabel("Meter Number");
    meterNum.setBounds(50,120,100,20);
    panel.add(meterNum);

    meternumText = new JLabel("");
    meternumText.setBounds(180,120,150,20);
    panel.add(meternumText);

    Random ran = new Random();
    long number = ran.nextLong() % 1000000;
    meternumText.setText(""+ Math.abs(number) );

    address = new JLabel("Address ");

```

```
address.setBounds(50,160,100,20);  
panel.add(address);
```

```
addressText = new TextField();  
addressText.setBounds(180,160,150,20);  
panel.add(addressText);
```

```
city = new JLabel("City ");  
city.setBounds(50,200,100,20);  
panel.add(city);
```

```
cityText = new TextField();  
cityText.setBounds(180,200,150,20);  
panel.add(cityText);
```

```
state = new JLabel("State ");  
state.setBounds(50,240,100,20);  
panel.add(state);
```

```
stateText = new TextField();  
stateText.setBounds(180,240,150,20);  
panel.add(stateText);
```

```
email = new JLabel("Email ");  
email.setBounds(50,280,100,20);  
panel.add(email);
```

```
emailText = new TextField();  
emailText.setBounds(180,280,150,20);  
panel.add(emailText);
```

```
phone = new JLabel("Phone ");  
phone.setBounds(50,320,100,20);  
panel.add(phone);
```

```
phoneText = new TextField();  
phoneText.setBounds(180,320,150,20);  
panel.add(phoneText);
```

```

        next = new JButton("Next");
        next.setBounds(120,390,100,25);
        next.setBackground(Color.black);
        next.setForeground(Color.white);
        next.addActionListener(this);
        panel.add(next);

        cancel = new JButton("Cancel");
        cancel.setBounds(230,390,100,25);
        cancel.setBackground(Color.black);
        cancel.setForeground(Color.white);
        cancel.addActionListener(this);
        panel.add(cancel);

        setLayout(new BorderLayout());
        add(panel,"Center");

        ImageIcon i1 = new
        ImageIcon(ClassLoader.getResource("icon/boy.png"));
        Image i2 =
        i1.getImage().getScaledInstance(230,200,Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel imgLable = new JLabel(i3);
        add(imgLable,"West");

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==next){
            String sname = nameText.getText();
            String smeter = meternumText.getText();
            String saddress = addressText.getText();
            String scity = cityText.getText();
            String sstate = stateText.getText();
            String email = emailText.getText();
            String sphone = phoneText.getText();

```

```

        String query_customer = "insert into new_customer
values('"+sname+"','"+smeter+"','"+saddress+"','"+scity+"','"+sstate+"','"+eemail
+ "','"+sphone+"')";
        String query_signup = "insert into Signup
values('"+smeter+"','"+sname+"','"+sphone+"')";

        try {
            database c = new database();
            c.statement.executeUpdate(query_customer);
            c.statement.executeUpdate(query_signup);

            JOptionPane.showMessageDialog(null, "Customer details added
successfully");
            setVisible(false);
            new meterInfo(smeter);
        } catch (Exception E) {
            E.printStackTrace();
        }
    } else {
        setVisible(false);
    }
}

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

```

Calculate Bill Page:

```

package electricity.billing.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

```

```

import java.sql.ResultSet;

public class calculate_bill extends JFrame implements ActionListener{
    JLabel nameText, addressText;
    TextField unitText;
    Choice meternumCho, monthCho;
    JButton submit, cancel;
    calculate_bill(){

        JPanel panel = new JPanel();
        panel.setLayout(null);
        panel.setBackground(new Color(214,195,247));
        add(panel);

        JLabel heading = new JLabel("Calculate Electricity Bill");
        heading.setBounds(70,10,300,20);
        heading.setFont(new Font("Tahoma",Font.BOLD,20));
        panel.add(heading);

        JLabel meternum = new JLabel("Meter Number");
        meternum.setBounds(50,80,100,20);
        panel.add(meternum);

        meternumCho = new Choice();
        try {
            database c = new database();
            ResultSet resultSet = c.statement.executeQuery("select * from
new_customer");
            while (resultSet.next()){
                meternumCho.add(resultSet.getString("meter_no"));
            }
        } catch (Exception E){
            E.printStackTrace();
        }
        meternumCho.setBounds(180,80,100,20);
        panel.add(meternumCho);

        JLabel name = new JLabel("Name");
        name.setBounds(50,120,100,20);
        panel.add(name);
    }
}

```



```

nameText = new JLabel("");
nameText.setBounds(180,120,150,20);
panel.add(nameText);

JLabel address = new JLabel("Address");
address.setBounds(50,160,100,20);
panel.add(address);

addressText = new JLabel("");
addressText.setBounds(180,160,150,20);
panel.add(addressText);

try {
    database c= new database();
    ResultSet resultSet = c.statement.executeQuery("select * from
new_customer where meter_no = '"+meternumCho.getSelectedItem()+" ");
    while (resultSet.next()){
        nameText.setText(resultSet.getString("name"));
        addressText.setText(resultSet.getString("address"));
    }
} catch (Exception E){
    E.printStackTrace();
}
meternumCho.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        try {
            database c= new database();
            ResultSet resultSet = c.statement.executeQuery("select * from
new_customer where meter_no = '"+meternumCho.getSelectedItem()+" ");
            while (resultSet.next()){
                nameText.setText(resultSet.getString("name"));
                addressText.setText(resultSet.getString("address"));
            }
        } catch (Exception E){
            E.printStackTrace();
        }
    }
});

```

```
JLabel unitconsumed = new JLabel("Unit Consumed");
unitconsumed.setBounds(50,200,100,20);
panel.add(unitconsumed);
```

```
unitText = new TextField();
unitText.setBounds(180,200,150,20);
panel.add(unitText);
```

```
JLabel month = new JLabel("Month");
month.setBounds(50,240,100,20);
panel.add(month);
```

```
monthCho = new Choice();
monthCho.add("January");
monthCho.add("February");
monthCho.add("March");
monthCho.add("April");
monthCho.add("May");
monthCho.add("June");
monthCho.add("July");
monthCho.add("August");
monthCho.add("September");
monthCho.add("October");
monthCho.add("November");
monthCho.add("December");
monthCho.setBounds(180,240,150,20);
panel.add(monthCho);
```

```
submit = new JButton("Submit");
submit.setBounds(80,300,100,25);
submit.setBackground(Color.black);
submit.setForeground(Color.white);
submit.addActionListener(this);
panel.add(submit);
```

```
cancel = new JButton("Cancel");
cancel.setBounds(220,300,100,25);
cancel.setBackground(Color.black);
```

```

cancel.setForeground(Color.white);
cancel.addActionListener(this);
panel.add(cancel);

setLayout(new BorderLayout());
add(panel,"Center");
ImageIcon imageIcon = new
ImageIcon(ClassLoader.getResource("icon/budget.png"));
Image image =
imageIcon.getImage().getScaledInstance(250,200,Image.SCALE_DEFAULT);
ImageIcon imageIcon1 = new ImageIcon(image);
JLabel imageLabel = new JLabel(imageIcon1);
add(imageLabel,"East");

setSize(650,400);
setLocation(400,200);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == submit){
        String smeterNo = meternumCho.getSelectedItem();
        String sunit = unitText.getText();
        String smonth = monthCho.getSelectedItem();

        int totalBill =0;
        int units=Integer.parseInt(sunit);
        String query_tax = "select * from tax";
        try{
            database c = new database();
            ResultSet resultSet = c.statement.executeQuery(query_tax);
            while (resultSet.next()){
                totalBill += units *
Integer.parseInt(resultSet.getString("cost_per_unit"));
                totalBill += Integer.parseInt(resultSet.getString("meter_rent"));
                totalBill += Integer.parseInt(resultSet.getString("service_charge"));
                totalBill += Integer.parseInt(resultSet.getString("swacch_bharat"));
                totalBill += Integer.parseInt(resultSet.getString("fixed_tax"));
            }
        }
    }
}

```

```

        }
    } catch (Exception E){
        E.printStackTrace();
    }
    String query_total_bill = "insert into bill values('"+smeterNo+"',
    '"+smonth+"','"+sunit+"', '"+totalBill+"','Not Paid')";
    try{
        database c = new database();
        c.statement.executeUpdate(query_total_bill);

        JOptionPane.showMessageDialog(null,"Customer Bill Updated
        Successfully");
        setVisible(false);
    } catch (Exception E){
        E.printStackTrace();
    }

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

```

Generate Bill Page:

```

package electricity.billing.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;

public class generate_bill extends JFrame implements ActionListener {

```

```

Choice searchmonthcho;
String meter;
JTextArea area;
JButton bill;
generate_bill(String meter){
    this.meter=meter;
    setSize(500,700);
    setLocation(500,30);
    setLayout( new BorderLayout());
    JPanel panel = new JPanel();

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

    JLabel meter_no = new JLabel(meter);

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

    area= new JTextArea(50,15);
    area.setText("\n \n \t ----- Click on the ----- \n \t -----
----- Generate Bill");
    area.setFont(new Font("Senserif",Font.ITALIC,15));
    JScrollPane pane = new JScrollPane(area);
    bill = new JButton("Generate Bill");
    bill.addActionListener(this);

    add(pane);

    panel.add(heading);

```

```

        panel.add(meter_no);
        panel.add(searchmonthcho);
        add(panel,"North");
        add(bill,"South");

        setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        try{
            database c = new database();
            String smonth = searchmonthcho.getSelectedItem();
            area.setText("\n Power Limited \n Electricity Bill For Month of
            "+smonth+",2023\n\n\n");
            ResultSet resultSet = c.statement.executeQuery("select * from
            new_customer where meter_no='"+meter+"'");
            if (resultSet.next()){
                area.append("\n Customer Name      :
            "+resultSet.getString("name"));
                area.append("\n Customer Meter Number:
            "+resultSet.getString("meter_no"));
                area.append("\n Customer Address   :
            "+resultSet.getString("address"));
                area.append("\n Customer City       : "+resultSet.getString("city"));
                area.append("\n Customer State    : "+resultSet.getString("state"));
                area.append("\n Customer Email     :
            "+resultSet.getString("email"));
                area.append("\n Customer Phone Number :
            "+resultSet.getString("phone_no"));

            }

            resultSet = c.statement.executeQuery("select * from meter_info where
            meter_number='"+meter+"'");
            if (resultSet.next()){
                area.append("\n Customer Meter Location      :

```

```

"+resultSet.getString("meter_location"));
        area.append("\n Customer Meter Type:
"+resultSet.getString("meter_type"));
        area.append("\n Customer Phase Code :
"+resultSet.getString("phase_code"));
        area.append("\n Customer Bill Type :
"+resultSet.getString("bill_type"));
        area.append("\n Customer Days : "+resultSet.getString("Days"));

    }
    resultSet = c.statement.executeQuery("select * from tax");
    if (resultSet.next()){
        area.append("\n Cost Per Unit :
"+resultSet.getString("cost_per_unit"));
        area.append("\n Meter Rent: "+resultSet.getString("meter_rent"));
        area.append("\n Service Charge :
"+resultSet.getString("service_charge"));
        area.append("\n Service Tax :
"+resultSet.getString("service_tax"));
        area.append("\n Swacch Bharat :
"+resultSet.getString("swacch_bharat"));
        area.append("\n Fixed Tax : "+resultSet.getString("fixed_tax"));

    }
    resultSet = c.statement.executeQuery("select * from bill where meter_no
= '"+meter+"' and month = '"+searchmonthcho.getSelectedItemAt()+'");
    if (resultSet.next()) {
        area.append("\n Current Month : " +
resultSet.getString("month"));
        area.append("\n Units Consumed: " + resultSet.getString("unit"));
        area.append("\n Total Charges : " +
resultSet.getString("total_bill"));
        area.append("\n Total Payable: "+resultSet.getString("total_bill"));
    }

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

```

```

    }

    public static void main(String[] args) {
        new generate_bill("");
    }
}

```

Pay Bill Page:

```

package electricity.billing.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.sql.ResultSet;

public class pay_bill extends JFrame implements ActionListener {
    Choice searchmonthcho;
    String meter;
    JButton pay,back;

    pay_bill(String meter){
        this.meter=meter;
        setSize(900,600);
        setLocation(300,150);
        setLayout(null);

        JLabel heading = new JLabel("Pay Bill");
        heading.setFont(new Font("Tahoma",Font.BOLD,24));
        heading.setBounds(120,5,400,30);
        add(heading);

        JLabel meterNumber = new JLabel("Meter Number");
        meterNumber.setBounds(35,80,200,20);
    }
}

```



```
add(meterNumber);
```

```
JLabel meterNumberText = new JLabel("");  
meterNumberText.setBounds(300,80,200,20);  
add(meterNumberText);
```

```
JLabel name = new JLabel("Name");  
name.setBounds(35,140,200,20);  
add(name);
```

```
JLabel nameText = new JLabel("");  
nameText.setBounds(300,140,200,20);  
add(nameText);
```

```
JLabel month = new JLabel("Month");  
month.setBounds(35,200,200,20);  
add(month);
```

```
searchmonthcho = new Choice();  
searchmonthcho.add("January");  
searchmonthcho.add("February");  
searchmonthcho.add("March");  
searchmonthcho.add("April");  
searchmonthcho.add("May");  
searchmonthcho.add("June");  
searchmonthcho.add("July");  
searchmonthcho.add("August");  
searchmonthcho.add("September");  
searchmonthcho.add("October");  
searchmonthcho.add("November");  
searchmonthcho.add("December");  
searchmonthcho.setBounds(300,200,150,20);  
add(searchmonthcho);
```

```
JLabel unit = new JLabel("Unit");  
unit.setBounds(35,260,200,20);
```

```
add(unit);
```

```
JLabel unitText = new JLabel("");  
unitText.setBounds(300,260,200,20);  
add(unitText);
```

```
JLabel totalBill = new JLabel("Total Bill");  
totalBill.setBounds(35,320,200,20);  
add(totalBill);
```

```
JLabel totalBillText = new JLabel("");  
totalBillText.setBounds(300,320,200,20);  
add(totalBillText);
```

```
JLabel status = new JLabel("Status");  
status.setBounds(35,380,200,20);  
add(status);
```

```
JLabel statusText = new JLabel("");  
statusText.setBounds(300,380,200,20);  
statusText.setForeground(Color.RED);  
add(statusText);
```

```
try{  
    database c = new database();  
    ResultSet resultSet = c.statement.executeQuery("select * from  
new_customer where meter_no = '"+meter+"'");  
    while (resultSet.next()){  
        meterNumberText.setText(meter);  
        nameText.setText(resultSet.getString("name"));  
    }  
}catch (Exception e){  
    e.printStackTrace();  
}
```

```
searchmonthcho.addItemListener(new ItemListener() {
```

```

@Override
public void itemStateChanged(ItemEvent e) {
    database c = new database();
    try {
        ResultSet resultSet = c.statement.executeQuery("select * from bill
where meter_no = '"+meter+"' and month =
 '"+searchmonthcho.getSelectedItemAt()+"'");
        while (resultSet.next()){
            unitText.setText(resultSet.getString("unit"));
            totalBillText.setText(resultSet.getString("total_bill"));
            statusText.setText(resultSet.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);

```

```

setVisible(true);
}

```

```

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource()==pay){
        try {

```

```

        database c = new database();
        c.statement.executeUpdate("update bill set status = 'Paid' where
meter_no = '"+meter+"' and month = '"+searchmonthcho.getSelectedItemAt()+"'");
    } catch (Exception E){
        E.printStackTrace();
    }
    setVisible(false);
    new payment_bill(meter);
} else {
    setVisible(false);
}
}

public static void main(String[] args) {
    new pay_bill("");
}
}

```

Update Information Page:

```

package electricity.billing.system;

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.RectangularShape;
import java.sql.ResultSet;

public class update_information extends JFrame implements ActionListener {
    JLabel nametext;
    JTextField addressText,cityText,stateText,emailText,phoneText;
    String meter;
    JButton update,cancel;
    update_information(String meter){
        this.meter=meter;
        setBounds(400,150,777,450);
        getContentPane().setBackground(new Color(229,255,227));
        setLayout(null);
    }
}

```

```
JLabel heading = new JLabel("Update Customer Information");  
heading.setBounds(50,10,400,40);  
heading.setFont(new Font("serif",Font.BOLD,20));  
add(heading);
```

```
JLabel name = new JLabel("Name");  
name.setBounds(30,70,100,20);  
add(name);
```

```
nametext = new JLabel("");  
nametext.setBounds(150,70,200,20);  
add(nametext);
```

```
JLabel meterNo = new JLabel("Meter Number");  
meterNo.setBounds(30,110,100,20);  
add(meterNo);
```

```
JLabel meterText = new JLabel("");  
meterText.setBounds(150,110,100,20);  
add(meterText);
```

```
JLabel address = new JLabel("Address");  
address.setBounds(30,150,100,20);  
add(address);
```

```
addressText = new JTextField();  
addressText.setBounds(150,150,200,20);  
add(addressText);
```

```
JLabel city = new JLabel("City");  
city.setBounds(30,190,100,20);  
add(city);
```

```
cityText = new JTextField();  
cityText.setBounds(150,190,200,20);  
add(cityText);
```

```
JLabel state = new JLabel("State");
```

```

state.setBounds(30,230,100,20);
add(state);

stateText = new JTextField();
stateText.setBounds(150,230,200,20);
add(stateText);

JLabel email = new JLabel("Email");
email.setBounds(30,270,100,20);
add(email);

emailText = new JTextField();
emailText.setBounds(150,270,200,20);
add(emailText);

JLabel phone = new JLabel("Phone");
phone.setBounds(30,310,100,20);
add(phone);

phoneText = new JTextField();
phoneText.setBounds(150,310,200,20);
add(phoneText);

try{
    database c = new database();
    ResultSet resultSet = c.statement.executeQuery("select * from
new_customer where meter_no = '"+meter+"'");
    if (resultSet.next()){
        nametext.setText(resultSet.getString("name"));
        meterText.setText(resultSet.getString("meter_no"));
        addressText.setText(resultSet.getString("address"));
        cityText.setText(resultSet.getString("city"));
        stateText.setText(resultSet.getString("state"));
        emailText.setText(resultSet.getString("email"));
        phoneText.setText(resultSet.getString("phone_no"));
    }
} catch (Exception e){
    e.printStackTrace();
}

```

```

update = new JButton("Update");
update.setBackground(new Color(33,106,145));
update.setForeground(Color.white);
update.setBounds(50,360,120,25);
update.addActionListener(this);
add(update);

```

```

cancel = new JButton("Cancel");
cancel.setBackground(new Color(33,106,145));
cancel.setForeground(Color.white);
cancel.setBounds(200,360,120,25);
cancel.addActionListener(this);
add(cancel);

```

```

        ImageIcon imageIcon = new
ImageIcon(ClassLoader.getResource("icon/update.png"));
        Image image =
imageIcon.getImage().getScaledInstance(400,410,Image.SCALE_DEFAULT);
        ImageIcon imageIcon1 = new ImageIcon(image);
        JLabel imgLabel = new JLabel(imageIcon1);
        imgLabel.setBounds(360,0,400,410);
        add(imgLabel);

        setVisible(true);

    }

```

```

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource()==update){
        String saddress = addressText.getText();
        String scity = cityText.getText();
        String sstate = stateText.getText();
        String semail = emailText.getText();
        String sphone = phoneText.getText();

        try {
            database c = new database();
            c.statement.executeUpdate("update new_customer set address

```

```
=""+saddress+"" , city = ""+scity+"" , state = ""+sstate+"" , email = ""+semail+"" ,
phone_no =""+sphone+"" where meter_no = ""+meter+""");
```

```

        JOptionPane.showMessageDialog(null,"User Information Updated
Successfully");
        setVisible(false);
    } catch (Exception E){
        E.printStackTrace();
    }
    } else {
        setVisible(false);
    }
}

public static void main(String[] args) {
    new update_information("");
}
}

```

Database Connectivity:

```

package electricity.billing.system;

import java.sql.*;

public class database {
    Connection connection;
    Statement statement;
    database(){
        try {
            connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bill_system","root"
,"Kish@2006");
            statement = connection.createStatement();

            System.out.println("Connected to the database successfully");

```



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

SQL QUERIES:

```
create Schema bill_system;
```

```
use bill_system;
```

```
create table Signup(meter_no varchar(20),username varchar(20),
name varchar(30),password varchar(30),usertype varchar(20));
```

```
select*from Signup;
```

```
create table new_customer(name varchar(30),meter_no varchar(20),
address varchar(50),city varchar(30),state varchar(20),
email varchar(30),phone_no varchar(12));
```

```
select*from new_customer;
```

```
create table meter_info(meter_number varchar(30),meter_location
varchar(30),meter_type varchar(30),phase_code varchar(30),bill_type
varchar(30),days varchar(10));
```

```
select*from meter_info;
```

```
create table tax(cost_per_unit varchar(20),meter_rent varchar(20),
service_charge varchar(20),service_tax varchar(20),swacch_bharat_acss
varchar(20),
```

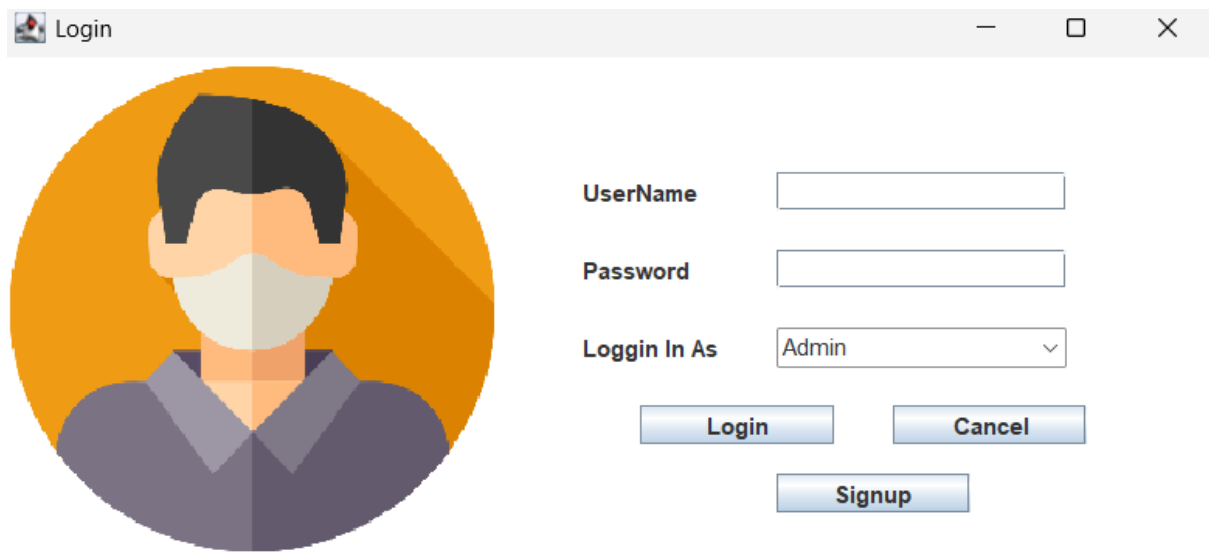
```
fixed_tax varchar(20));
```

```
select*from tax;
```

```
insert into tax values('10', '45', '20', '58', '5', '18');
```

```
create table bill(meter_no varchar(20),month varchar(20),unit  
varchar(20),total_bill varchar(20),status varchar(20));  
select*from bill;
```

PROJECT SCREENSHOTS



A screenshot of a web application window titled "Login". The window has a standard OS-style title bar with minimize, maximize, and close buttons. On the left side, there is a large circular profile picture of a person with dark hair and a beard, wearing a grey shirt, set against an orange background. To the right of the profile picture, there are three input fields: "UserName" (a text box), "Password" (a text box), and "Login In As" (a dropdown menu currently showing "Admin"). Below these fields are three buttons: "Login" and "Cancel" are side-by-side, and "Signup" is centered below them.

Login

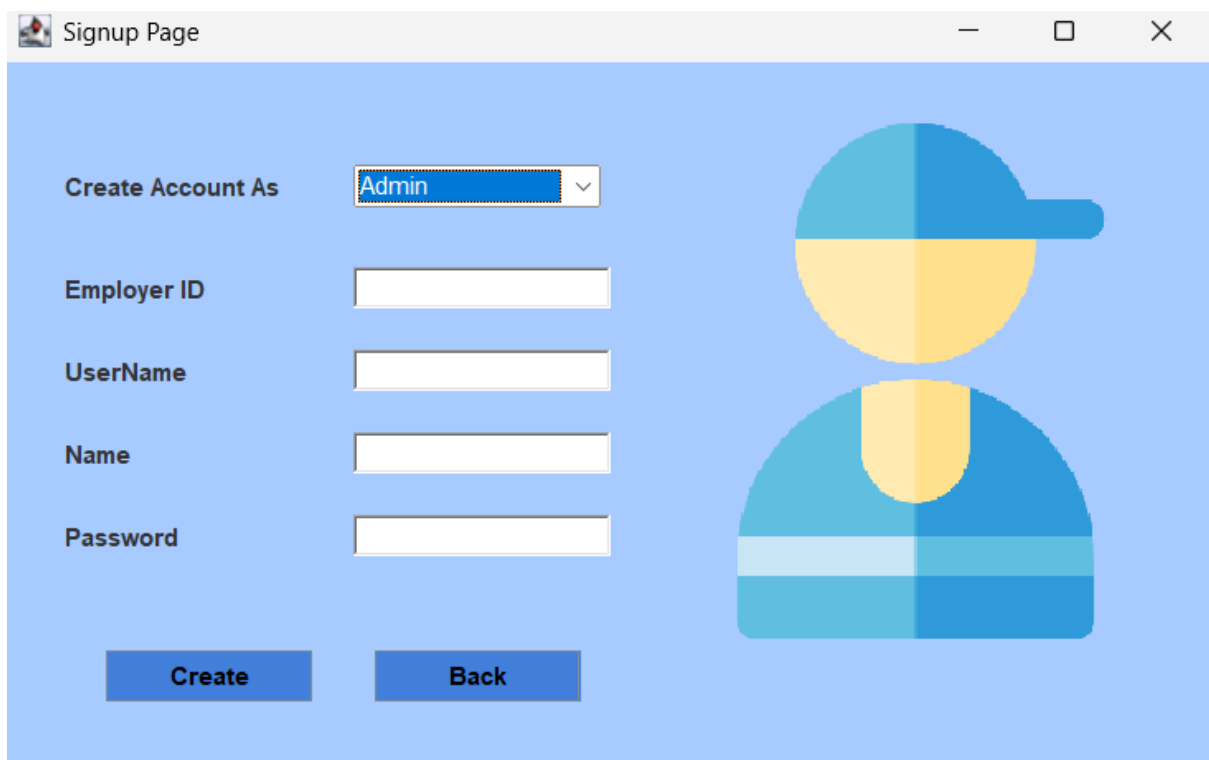
UserName

Password

Login In As

Login Cancel

Signup



A screenshot of a web application window titled "Signup Page". The window has a standard OS-style title bar. The background is a solid light blue. On the left side, there are five input fields: "Create Account As" (a dropdown menu showing "Admin"), "Employer ID" (a text box), "UserName" (a text box), "Name" (a text box), and "Password" (a text box). Below these fields are two buttons: "Create" and "Back". On the right side of the window, there is a large circular profile picture of a person with blonde hair, wearing a blue shirt and a blue cap, set against a yellow background.

Signup Page

Create Account As

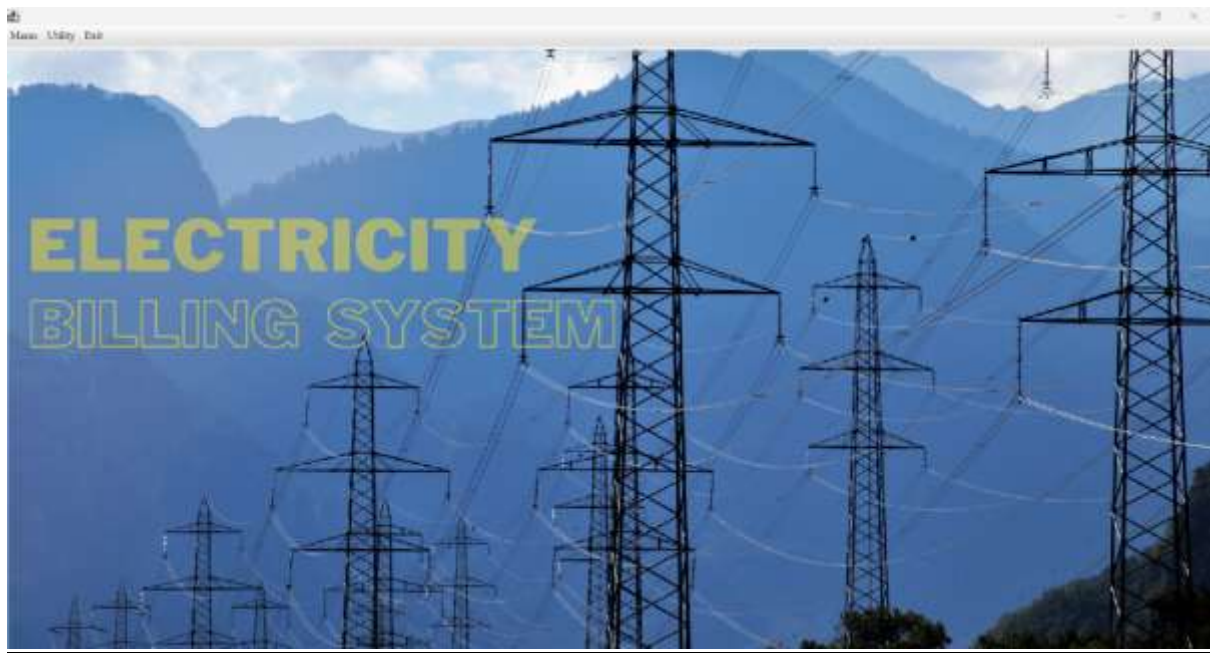
Employer ID

UserName

Name

Password

Create Back

A screenshot of a software application window titled "New Customer" with a standard Windows-style title bar. The window is divided into two main sections. On the left is a light gray sidebar containing a stylized icon of a person wearing a blue and yellow cap and a blue and yellow shirt. The main area on the right has a solid yellow background and is titled "New Customer" in bold black text. It contains a form with the following fields: "New Customer" (empty text input), "Meter Number" (text input containing "481467"), "Address" (empty text input), "City" (empty text input), "State" (empty text input), "Email" (empty text input), and "Phone" (empty text input). At the bottom of the form are two black buttons with white text: "Next" and "Cancel".

name	meter_no	address	city	state	email	phone_no
Kannan	218746	Plot no.77	Chennai	Tamilnadu	Kannan@gmail.c...	8975603423
Kannan	223331	plot n0o.89	Chennai	Tamilnadu	kannan@gmail.c...	89769053

RESULTS AND DISCUSSION

The result of the Electricity Management System refers to the successful implementation and output of a functional system that collects, processes, stores, and analyzes electricity consumption and billing data. This system enables users to efficiently manage large volumes of electricity usage data (customer details, meter readings, billing records, etc.) while providing meaningful insights and visualizations.

The results of the Electricity Management System would include:

- Efficient data storage for large-scale electricity usage and billing records.
- Actionable insights through data analysis and visualizations, such as usage trends and payment behaviors.
- Scalability to handle an increasing number of customers and data over time.

- A user-friendly interface for exploring customer data, tracking electricity usage, and managing payments.
- Real-time or batch report generation for stakeholders (e.g., utility companies, administrators).

This system will simplify the process of monitoring electricity consumption, managing bills, and making informed decisions based on data-driven insights. It will enhance operational efficiency and improve customer satisfaction by ensuring accurate billing and providing easy access to relevant information.

CONCLUSIONS

The Electricity Management System represents a comprehensive solution for collecting, processing, storing, and analyzing electricity consumption and billing data. By leveraging advanced technologies, including backend frameworks (like Python and Flask), a robust database system (such as MySQL or PostgreSQL), and an intuitive frontend interface, this system provides valuable tools for utility companies, administrators, and consumers to manage electricity efficiently and transparently.

Key Highlights:

Comprehensive Data Collection:

The system collects a wide range of electricity-related data, including customer details, meter readings, billing records, payment statuses, and consumption patterns. This data is seamlessly integrated into the system through automated processes, ensuring accuracy and scalability.

Efficient Data Storage and Management:

Electricity data is organized into a well-structured database, allowing for efficient queries and quick access to information. Whether utilizing a relational database like MySQL or PostgreSQL, the system ensures that high volumes of customer and meter data can be stored, processed, and retrieved without performance issues.

Data Processing and Analysis:

The backend functionalities provide advanced data processing capabilities, such as calculating bills based on usage, tracking payment statuses, and analyzing consumption trends over time. This enables utility providers to detect inefficiencies, identify peak usage periods, and optimize resource allocation.

Actionable Insights and Reporting:

Dashboards and visualizations offer meaningful insights, such as energy usage trends, billing summaries, and payment histories. Reports generated from this data assist utility companies in identifying patterns, reducing energy waste, and enhancing decision-making. Consumers can also leverage these insights to better manage their electricity consumption.

Scalability and Flexibility:

The system is designed to grow alongside the number of users and data volume. Whether serving a small community or a large metropolitan area, its architecture ensures seamless scaling and adaptability to evolving requirements, such as adding new customers, updating tax policies, or integrating renewable energy data.

User-Friendly Interface:

The frontend interface, built with user-centric tools, ensures that both administrators and consumers can easily interact with the system.

Administrators can efficiently manage customer accounts, monitor payments, and generate reports, while consumers can access their billing details, track usage, and make payments effortlessly.

REFERENCES:

- Database System Concepts(6th Edition) By Abraham Silberschatz, Henry F. Korth,S. Sudarshan.

WEBSITES:

- Www.Geeksforgeeks.Com
- Wwww.W3schools.Com