JAVA PROJECT REPORT

(Project Term January-May 2023)

Revolutionizing Electricity Billing and Usage Tracking System

Submitted by

Name: Master Ankur **Registration no: 12101680**

Course Code: CSE310

Under the Guidance of
Dr.A. Ranjith Kumar
Associate Professor

School of Computer Science and Engineering

LOVELY PROFESSIONAL UNIVERSITY

DECLARATION

We hereby declare that the project work entitled ("Electricity Management System") is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Dr. Ranjith Kumar A, during August to November 2022. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Name: Master Ankur

Registration no: 12101680

Signature of Student: Master Ankur

Date: 20-04-2023

TABLE OF CONTENTS

INTRODUCTION

An electricity billing system is a computerized system that is designed to calculate and manage the billing of electricity usage by consumers. This system is used by electric utility companies to track and monitor the consumption of electricity by residential, commercial, and industrial customers, and generate accurate bills based on the amount of electricity consumed. This billing engine uses this data to generate bills based on the customer's electricity usage and other relevant factors, such as time-of-use rates or demand charges. The meter reading system is used to collect data on the amount of electricity consumed by the customer. Alternatively, data may be collected manually by utility employees who read traditional analog meters.

The customer service portal is used to provide customers with access to their account information, including their billing history. This portal may also allow customers to submit service requests, report outages, and view their usage history. In addition to these core components, many electricity billing systems may also include additional features, such as analytics and reporting tools, automated payment processing, and integration with other utility systems such as outage management or customer relationship management. Overall, the electricity billing system plays a critical role in the management of utility services, helping to ensure that customers are billed accurately and on time, while also providing important data and insights for utility providers to manage their operations effectively.

SCOPE OF THE PROJECT

The scope of an electricity billing system typically includes the following:

Customer information management: This includes maintaining customer details such as name, address, contact information, and account details.

- Meter reading and billing: The system should be able to capture meter readings, calculate energy consumption, and generate bills based on the tariff rate.
- Tariff management: The system should be able to handle different tariff structures such as flat rates, time-of-use rates, and demand-based rates.
- Payment processing: The system should be able to accept and process payments from customers via various modes such as online payments, credit/debit card payments, and bank transfers.
- Disconnection and reconnection management: The system should be able to manage disconnection and reconnection of services based on customer payments or non-payments.

- Reporting and analytics: The system should be able to generate reports and provide analytics on energy consumption, revenue, and customer trends.
- Customer service: The system should provide customer support through various channels such as phone, email, and chat.

Overall, the scope of an electricity billing system covers the end-to-end process of managing customer accounts, billing, payments, and customer service.

- Integration with smart meters: The system can be integrated with smart meters to capture real-time energy consumption data and automate billing processes.

- Automated billing and payment reminders: The system can send automated billing and payment reminders to customers via email, SMS, or push notifications.

- Billing dispute management: The system should be able to handle billing disputes and provide a mechanism for customers to raise complaints and resolve disputes.

- Energy efficiency programs: The system can support energy efficiency programs by providing customers with personalized recommendations on how to reduce their energy consumption and save on their bills.

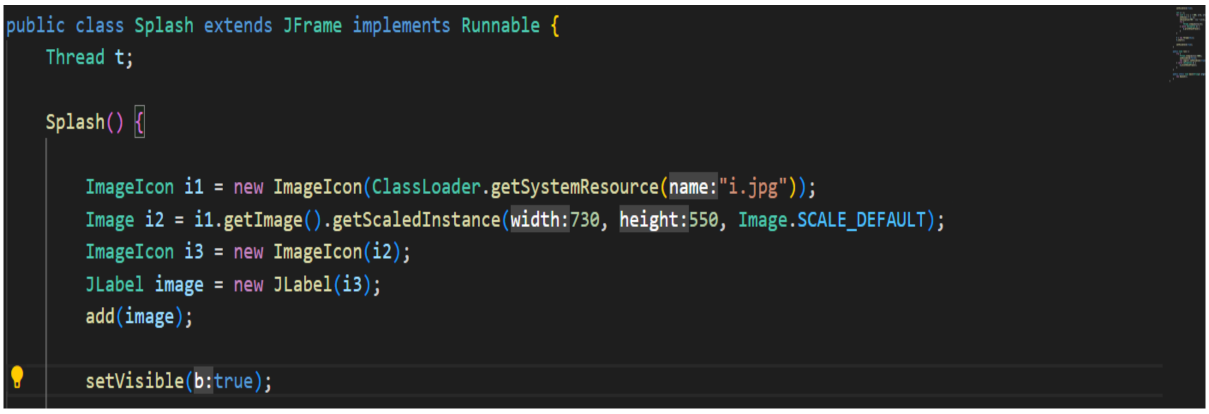
- Revenue protection: The system should be able to identify potential revenue loss due to meter tampering, theft, or billing errors and take appropriate measures to prevent it.

- Demand response management: The system can enable demand response programs by incentivizing customers to reduce their energy consumption during peak demand periods.

- These are some of the additional features that may be included in the scope of an electricity billing system, depending on the specific needs of the utility and its customers.

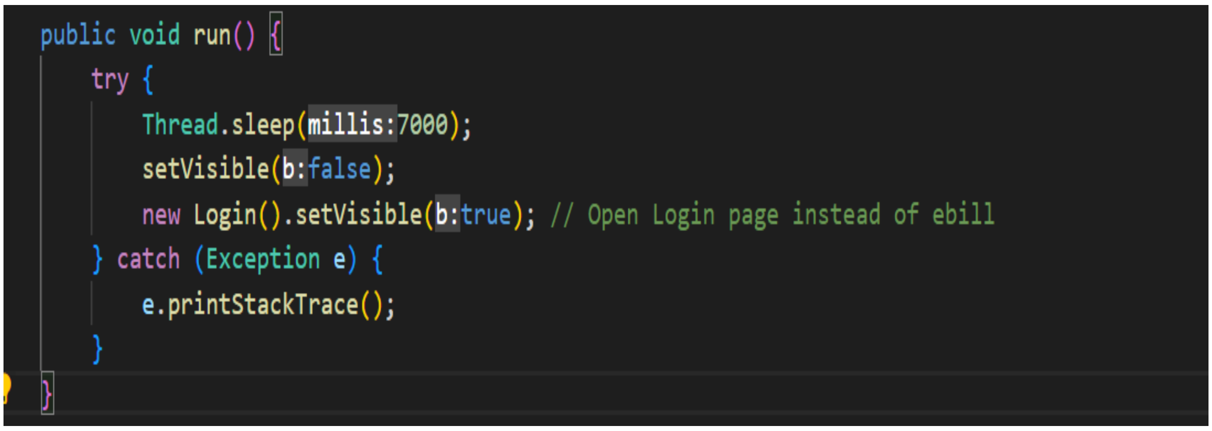
MODULES

SPLASH



```
public class Splash extends JFrame implements Runnable {
    Thread t;

    Splash() {

        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource(name:"i.jpg"));
        Image i2 = i1.getImage().getScaledInstance(width:730, height:550, Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel image = new JLabel(i3);
        add(image);

        setVisible(b:true);
```

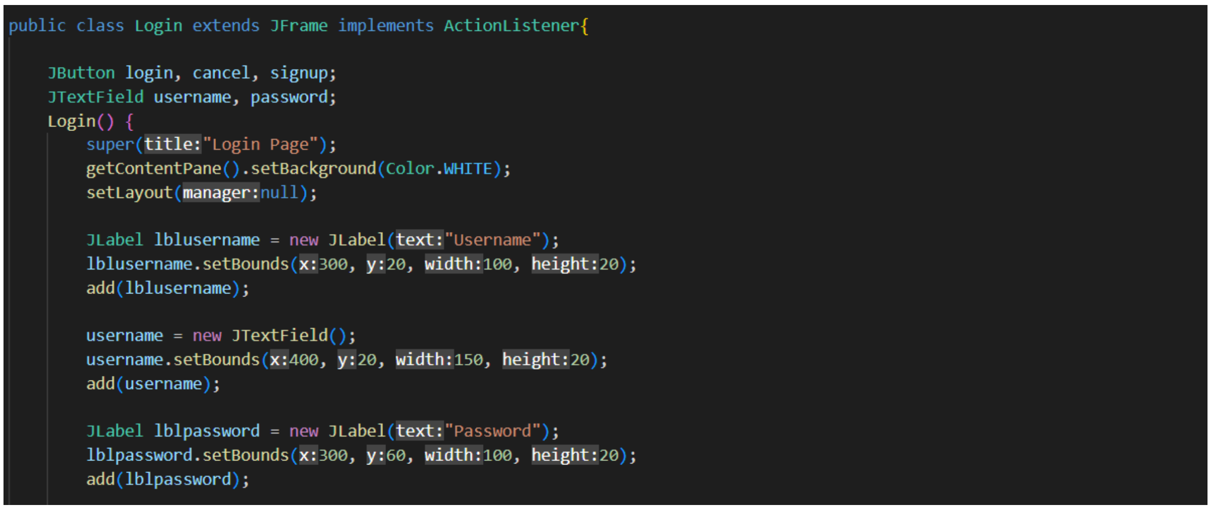
The setVisible(true) method call is used to make the frame visible on the screen. In this code, it sets the Splash frame visible, so the user can see it on their screen. The add(image) call before it adds a JLabel component containing the ImageIcon to the frame's content pane, which displays the image within the frame. So together, these lines create a frame window with an image displayed on it and make it visible to the user.



```
public void run() {
    try {
        Thread.sleep(millis:7000);
        setVisible(b:false);
        new Login().setVisible(b:true); // Open Login page instead of ebill
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

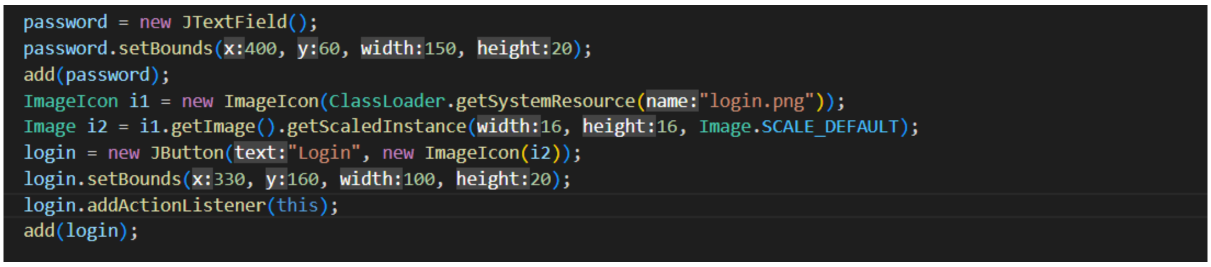
This code defines a run() method for the Splash class, which is executed when the t thread is started (as defined in the constructor). The run () method contains a try-catch block that pauses the thread for 7 seconds (using Thread.sleep(7000)). Then it sets the visibility of the Splash frame to false (using setVisible(false)) to hide it from the user. After that, it creates a new Login object and sets it visible (using new Login().setVisible(true)). This opens a new window (or page) showing the login screen to the user.

LOGIN



```
public class Login extends JFrame implements ActionListener{

    JButton login, cancel, signup;
    JTextField username, password;
    Login() {
        super(title:"Login Page");
        getContentPane().setBackground(Color.WHITE);
        setLayout(manager:null);

        JLabel lblusername = new JLabel(text:"Username");
        lblusername.setBounds(x:300, y:20, width:100, height:20);
        add(lblusername);

        username = new JTextField();
        username.setBounds(x:400, y:20, width:150, height:20);
        add(username);

        JLabel lblpassword = new JLabel(text:"Password");
        lblpassword.setBounds(x:300, y:60, width:100, height:20);
        add(lblpassword);
```

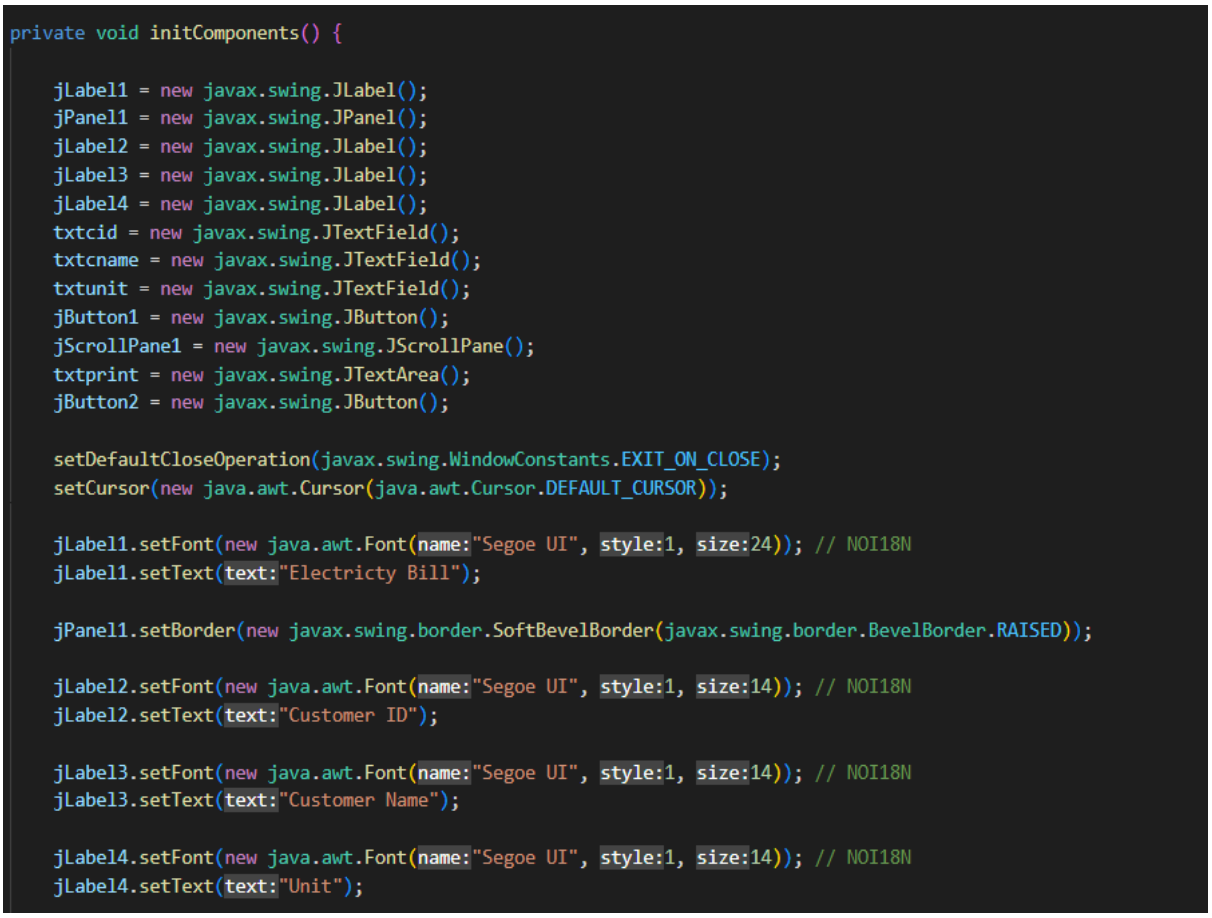
This code defines a JLabel component called lblpassword, which displays the text "Password" on the login page. This JLabel component is added to the Login frame using the add () method and positioned on the frame using the setBounds() method. Create and add a new JLabel with the text "Password" and position it on the login page at coordinates (300, 60) with a width of 100 pixels and a height of 20 pixels. This code is defining the layout and positioning of the UI components on the login page.



```
password = new JTextField();
password.setBounds(x:400, y:60, width:150, height:20);
add(password);
ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource(name:"login.png"));
Image i2 = i1.getImage().getScaledInstance(width:16, height:16, Image.SCALE_DEFAULT);
login = new JButton(text:"Login", new ImageIcon(i2));
login.setBounds(x:330, y:160, width:100, height:20);
login.addActionListener(this);
add(login);
```

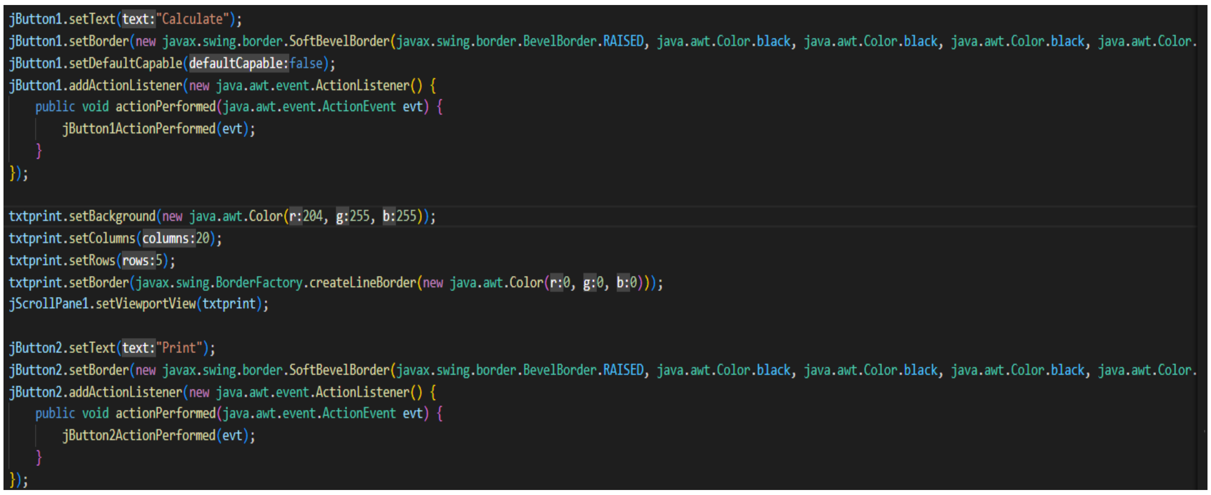
JTextField component called password which is added to the frame and positioned on it using the setBounds() method. ImageIcon object called i1 is created from an image file "login.png" located in the system resources using the ClassLoader.getSystemResource() method. Then, the image is scaled down to 16x16 pixels using the getScaledInstance() method to create a new Image object called i2. JButton component called login is created with the text "Login" and an icon i2 using the JButton(String text, Icon icon) constructor. The button is positioned on the login page using the setBounds () method. The ActionListener interface is implemented by the Login class, so the login button is registered with the addActionListener () method and this reference (which refers to the current Login object) is passed as the listener object. Overall, this code defines the password field, a login button with an icon and registers an action listener to be executed when the login button is clicked.

DASHBOARD



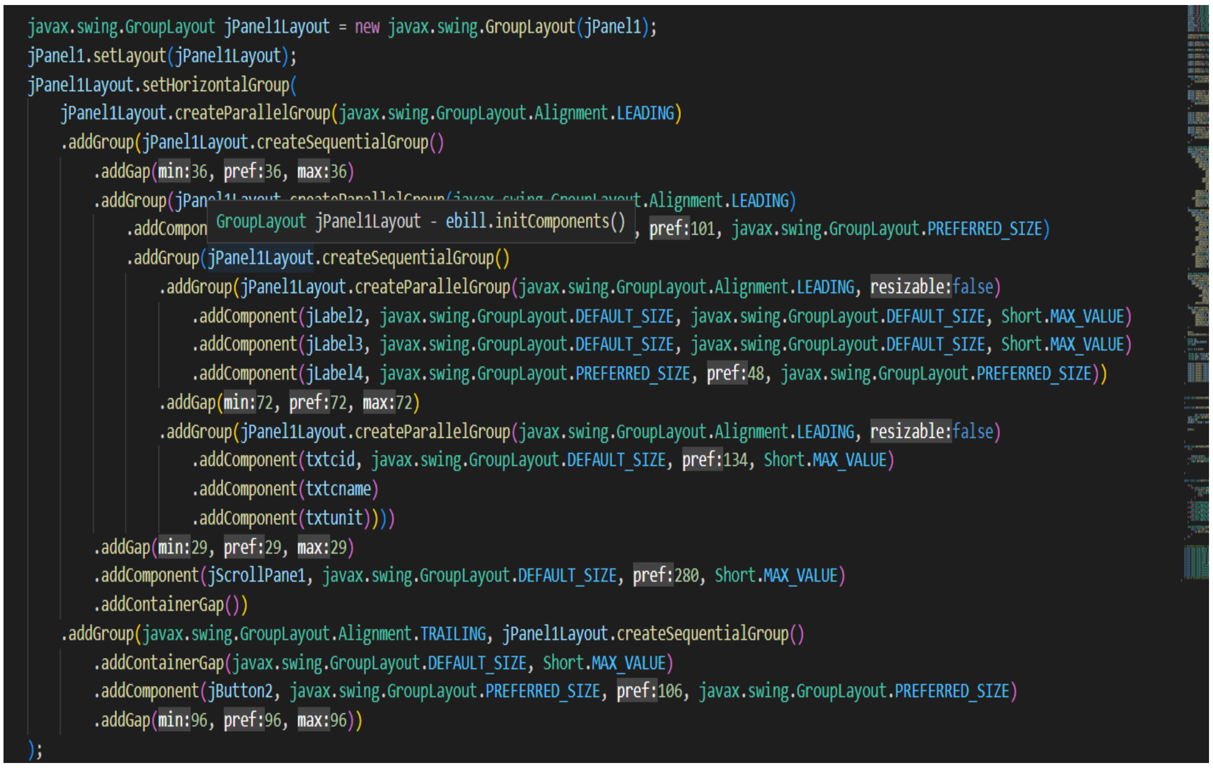
```java
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jPanel1 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    txtcid = new javax.swing.JTextField();
    txtcname = new javax.swing.JTextField();
    txtunit = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    txtprint = new javax.swing.JTextArea();
    jButton2 = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

    jLabel1.setFont(new java.awt.Font(name:"Segoe UI", style:1, size:24)); // NOI18N
    jLabel1.setText(text:"Electricty Bill");

    jPanel1.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    jLabel2.setFont(new java.awt.Font(name:"Segoe UI", style:1, size:14)); // NOI18N
    jLabel2.setText(text:"Customer ID");

    jLabel3.setFont(new java.awt.Font(name:"Segoe UI", style:1, size:14)); // NOI18N
    jLabel3.setText(text:"Customer Name");

    jLabel4.setFont(new java.awt.Font(name:"Segoe UI", style:1, size:14)); // NOI18N
    jLabel4.setText(text:"Unit");
```

This code initializes the components of a graphical user interface (GUI) for an electricity bill application. It sets up a window with a title "Electricity Bill" and three labels with text "Customer ID", "Customer Name", and "Unit". The labels are associated with text fields for the user to input values for these fields. The GUI also has two buttons: "jButton1" and "jButton2". "jButton1" is labeled "Calculate" and "jButton2" is labelled "Clear". The GUI also has a scrollable text area "txtprint" that displays the result of the calculation.



```java
jButton1.setText(text:"Calculate");
jButton1.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED, java.awt.Color.black, java.awt.Color.black, java.awt.Color.black, java.awt.Color.
jButton1.setDefaultCapable(defaultCapable:false);
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

txtprint.setBackground(new java.awt.Color(r:204, g:255, b:255));
txtprint.setColumns(columns:20);
txtprint.setRows(rows:5);
txtprint.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(r:0, g:0, b:0)));
jScrollPane1.setViewportView(txtprint);

jButton2.setText(text:"Print");
jButton2.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED, java.awt.Color.black, java.awt.Color.black, java.awt.Color.black, java.awt.Color.
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});
```

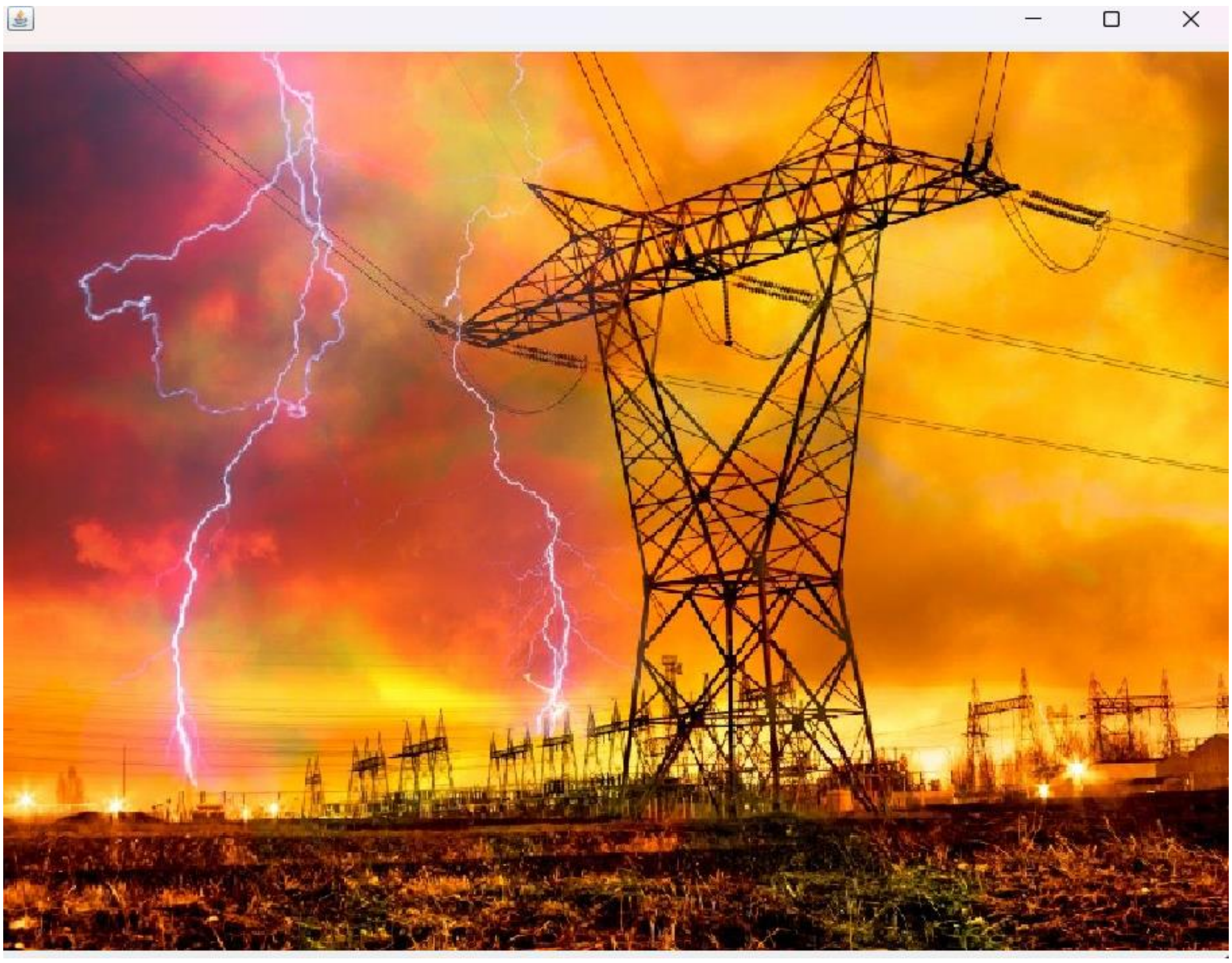
This code initializes and sets properties for three GUI components: a JButton named jButton1, a JTextArea named txtprint, and another JButton named jButton2. jButton1 is set to display the text "Calculate" and has a raised SoftBevelBorder. It is also set to be the default button when the frame is focused. An ActionListener is added to handle the button click event. txtprint is a JTextArea that displays the result of the electricity bill calculation. It is given a background color and border, and is set to allow five rows of text. jButton2 is another JButton set to display the text "Print" and has a raised SoftBevelBorder. An ActionListener is added to handle the button click event.

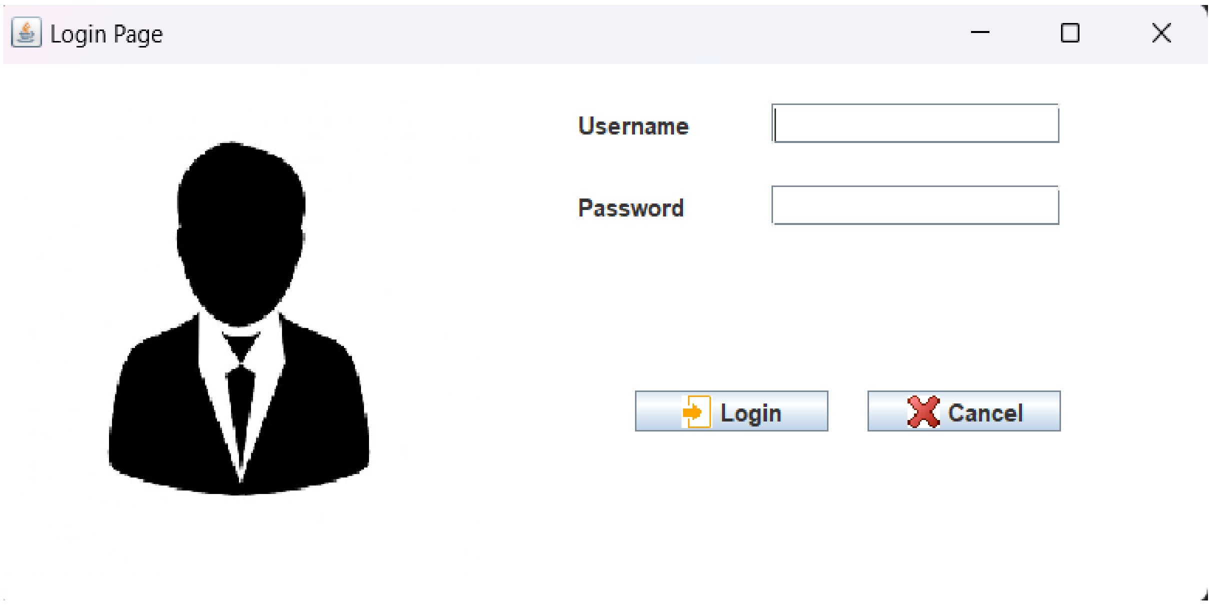


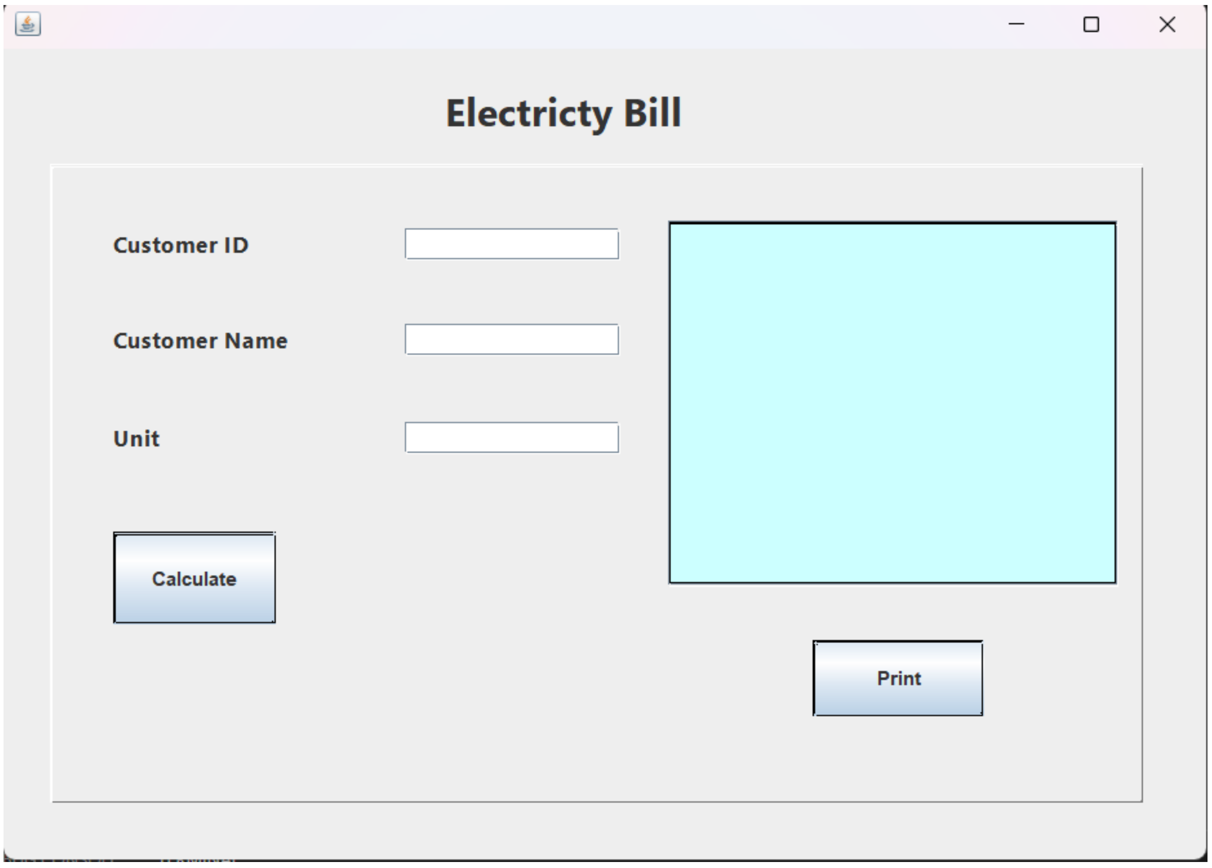
```java
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(min:36, pref:36, max:36)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addCompon GroupLayout jPanel1Layout - ebill.initComponents() , pref:101, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, resizable:false)
                    .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, pref:48, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(min:72, pref:72, max:72)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, resizable:false)
                    .addComponent(txtcid, javax.swing.GroupLayout.DEFAULT_SIZE, pref:134, Short.MAX_VALUE)
                    .addComponent(txtcname)
                    .addComponent(txtunit))))
        .addGap(min:29, pref:29, max:29)
        .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, pref:280, Short.MAX_VALUE)
        .addContainerGap())
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, pref:106, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(min:96, pref:96, max:96))
);
```

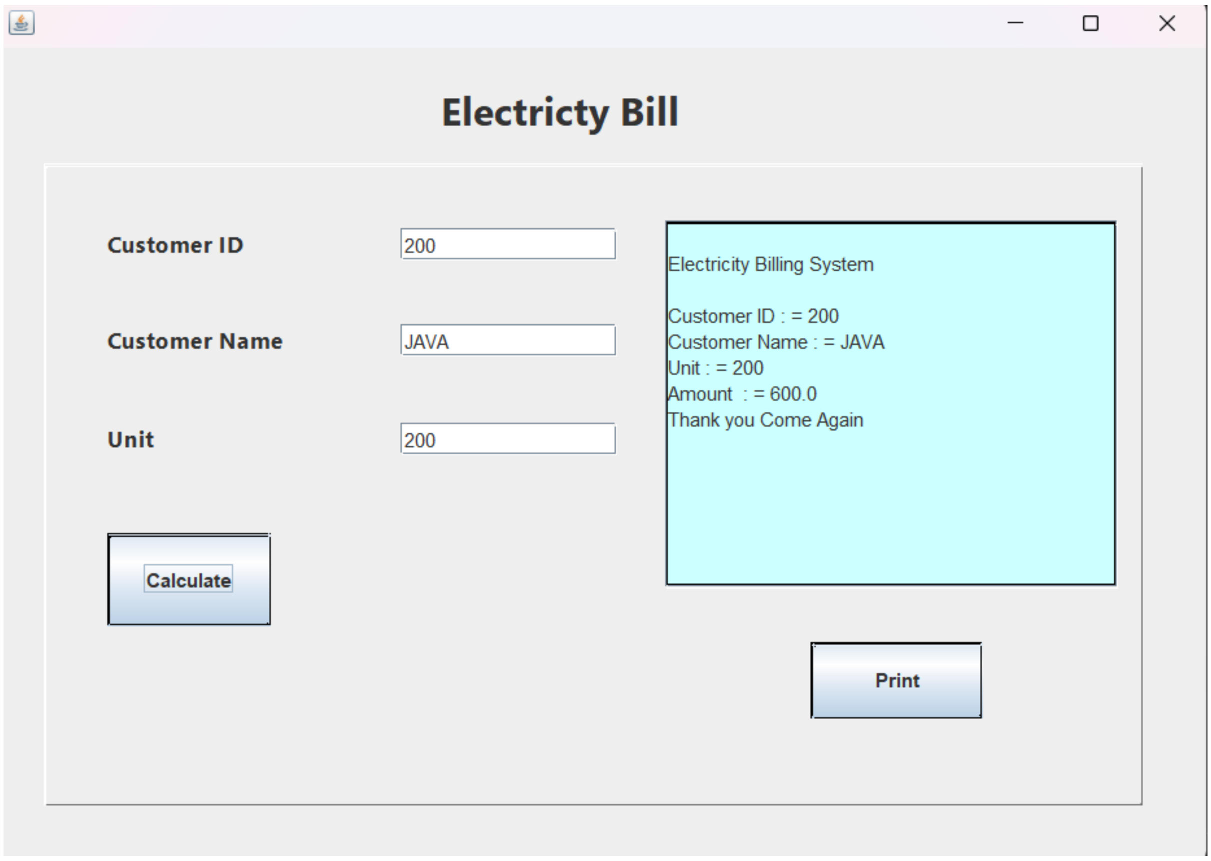
This code block creates the layout of the main panel of the GUI. It uses the GroupLayout manager to position the various components on the panel. The panel contains labels and text fields for the customer ID, customer name, and units. It also contains a "Calculate" button to compute the electricity bill and a "Print" button to print the bill. The panel is divided into two parts using a split pane. The left part contains the labels and text fields, while the right part contains a text area for displaying the bill details. The split pane is positioned using horizontal and vertical grouping. The horizontal grouping aligns the split pane and the "Print" button, while the vertical grouping aligns the split pane and the "Calculate" button.

OUTPUT OF THE PROJECT




Login Page
Username
Password
Login
Cancel


Electricty Bill
Customer ID
Customer Name
Unit
Calculate
Print

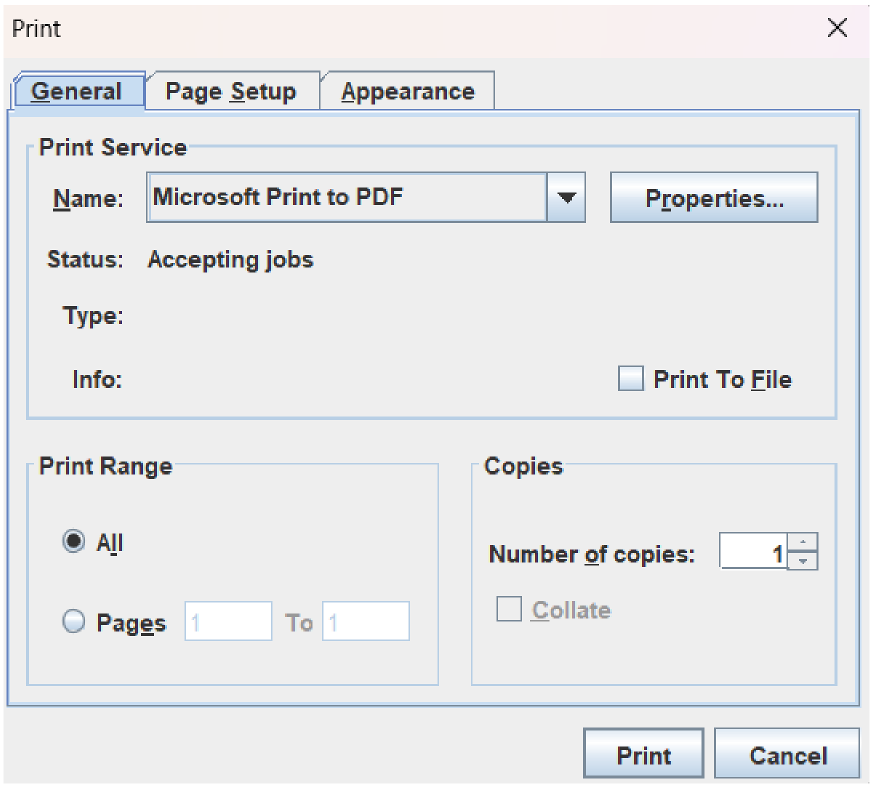


Customer ID 200

Customer Name JAVA

Unit 200

```
Electricity Billing System

Customer ID : = 200
Customer Name : = JAVA
Unit : = 200
Amount  : = 600.0
Thank you Come Again
```



Print Service

Name: Microsoft Print to PDF

Status: Accepting jobs

Type:

Info:

Print To File [ ]

Print Range

[✓ All]

Pages 1 To 1

Copies

Number of copies: 1

Collate [ ]

CONCLUSION

After all the hard work is done for this project is here. It is software that helps the user to work with the billing cycles, paying bills. 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.

Based on the provided information, it seems that the electricity management system is designed to monitor and control the energy consumption of a building or facility. The system collects data from various sensors and meters to provide real-time information about energy usage, and uses this data to optimize energy consumption and reduce waste. The system also allows for remote monitoring and control, providing the ability to adjust settings and schedules from a central location.

In addition to energy management, the system also appears to include features for monitoring the health and status of electrical equipment, as well as providing alerts for any issues or failures. Overall, the electricity management system seems to be a useful tool for improving energy efficiency and reducing costs in a building or facility. By providing real-time data and remote control capabilities, it can help organizations optimize their energy consumption and improve their overall sustainability.