# ELECTRICITY BILLING SYSTEM

**A PROJECT REPORT**

***Submitted by***

**ELAIYANILA S (2303811710422037)**

***in partial fulfillment of requirements for the award of the course***

**CGB1201 - JAVA PROGRAMMING**

***In***

**COMPUTER SCIENCE AND ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**NOVEMBER- 2024**

# K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

**SAMAYAPURAM – 621 112**

## BONAFIDE CERTIFICATE

Certified that this project report on **"ELECTRICITY BILLING SYSTEM"** is the bonafide work of **ELAIYANILA S (2303811710422037)**who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR
[illegible]

CGB1201-JAVA PROGRAMMING
Mr. M. SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR
[illegible]

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| Dr.A.Delphin Carolina Rani, M.E.,Ph.D., | Mr. M. Saravanan, M.E., |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| PROFESSOR | ASSISTANT PROFESSOR |
| Department of CSE | Department of CSE |
| K.Ramakrishnan College of Technology (Autonomous) | K.Ramakrishnan College of Technology (Autonomous) |
| Samayapuram–621112. | Samayapuram–621112. |

Submitted for the viva-voce examination held on 02/12/2024

CGB1201-JAVA PROGRAMMING
Mr.MAARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR
[illegible]

CGB1201-JAVA PROGRAMMING
Dr.R.SENTHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.
[illegible]

INTERNAL EXAMINER EXTERNAL EXAMINER

# DECLARATION

I declare that the project report on **"ELECTRICITY BILLING SYSTEM"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

.

**Signature**

S. Elaiyanila

S.ELAIYANILA

Place: Samayapuram

Date:02/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. M. SARAVANAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

## MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

## VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

## MISSION OF DEPARTMENT

**M1: Industry Specific:** To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

**M2: Research:** To prepare students for research-oriented activities.

**M3: Society:** To empower students with the required skills to solve complex technological problems of society.

## PROGRAM EDUCATIONAL OBJECTIVES

**1. PEO1: Domain Knowledge**

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

**2. PEO2: Employability Skills and Research**

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

**3. PEO3: Ethics and Values**

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO 1: Domain Knowledge**

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

**PSO 2: Quality Software**

To apply software engineering principles and practices for developing quality software for scientific and business applications.

**PSO 3: Innovation Ideas**

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

**PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

# ABSTRACT

The **Electricity Billing System** developed using **Java Swing** provides a graphical user interface (GUI) to facilitate the calculation and management of electricity consumption and billing. The system allows users to enter essential customer details such as the customer name, ID, and units of electricity consumed. Based on these inputs, the system computes the electricity bill using a tiered pricing structure. The pricing model includes varying rates based on the number of units consumed, with different rates for consumption in the ranges of 0-100 units, 101-200 units, 201-300 units, and above 300 units.Additionally, the system incorporates an **18% Goods and Services Tax (GST)**, which is calculated based on the base bill amount. The final output includes the breakdown of the total bill, GST, and the overall total with GST applied.

## ABSTRACT WITH POs AND PSOs MAPPING

## CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

| ABSTRACT | POs MAPPED | PSOs MAPPED |
|---|---|---|
| The Electricity Billing System developed using Java Swing provides a graphical user interface (GUI) to facilitate the calculation and management of electricity consumption and billing. The system allows users to enter essential customer details such as the customer name, ID, and units of electricity consumed. Based on these inputs, the system computes the electricity bill using a tiered pricing structure. The pricing model includes varying rates based on the number of units consumed, with different rates for consumption in the ranges of 0-100 units, 101-200 units, 201-300 units, and above 300 units. Additionally, the system incorporates an 18% Goods and Services Tax (GST), which is calculated based on the base bill amount. The final output includes the breakdown of the total bill, GST, and the overall total with GST applied. | **PO1 -3**<br>**PO2 -3**<br>**PO3 -3**<br>**PO4 -3**<br>**PO5 -3**<br>**PO6 -3**<br>**PO7 -3**<br>**PO8 -3**<br>**PO9 -3**<br>**PO10 -3**<br>**PO11-3**<br>**PO12 -3** | **PSO1 -3**<br>**PSO2 -3**<br>**PSO3 -3** |

Note: 1- Low, 2-Medium, 3- High

# TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|---|---|---|
| | **ABSTRACT** | |
| **1** | **INTRODUCTION** | |
| | 1.1 Objective | 1 |
| | 1.2 Overview | 1 |
| | 1.3 Java Programming concepts | 2 |
| **2** | **PROJECT METHODOLOGY** | |
| | 2.1 Proposed Work | 4 |
| | 2.2 Block Diagram | 4 |
| **3** | **MODULE DESCRIPTION** | |
| | 3.1 Bill Calculation | 5 |
| | 3.2 User Interface | 5 |
| | 3.3 GST Calculation | 6 |
| | 3.4 Error Handling | 6 |
| **4** | **CONCLUSION & FUTURE SCOPE** | |
| | 4.1 Conclusion | 7 |
| | 4.2 Future Scope | 7 |
| | **REFERENCES** | |
| | **APPENDIX A (SOURCE CODE)** | 9 |
| | **APPENDIX B (SCREENSHOTS)** | 12 |

# CHAPTER 1

# INTRODUCTION

## 1.1 Objective

The objective of the Electricity Billing System code is to provide a user-friendly graphical interface for calculating electricity bills based on the number of units consumed. The system follows a tiered pricing structure, where the cost per unit varies depending on the amount of electricity used, with rates of Rs. 5 for the first 100 units, Rs. 7 for the next 100 units, Rs. 10 for the subsequent 100 units, and Rs. 12 for units exceeding 300. Additionally, the system calculates an 18% GST on the total bill and adds it to the final amount. Users input their details, including their name, ID, and units consumed, and the system displays the total bill amount, GST, and the final amount to be paid.

## 1.2 Overview

An Electricity Billing System is a software application designed to calculate and manage the billing process for electricity consumption. It tracks the amount of electricity used by a customer and generates an invoice based on predefined pricing structures and additional charges like taxes. Typically, the system accepts customer details (such as name, ID, and address) and the number of units consumed over a given period. The billing process involves applying tiered pricing, where the cost per unit of electricity may vary depending on the total consumption, often with higher rates applied to larger quantities of electricity used.In addition to the base charges for consumption, the system may also calculate taxes, such as GST (Goods and Services Tax), and apply any additional fees or surcharges if necessary. After performing these calculations, the system generates a bill that can be presented to the customer, showing the breakdown of the total cost, taxes, and final amount due.

## 1.3 Java Programming Concepts

**The basic concepts of Object-Oriented Programming (OOP) are:**

- ✓ **Class and Object:** A class is a blueprint, and an object is an instance of the class.
- ✓ **Encapsulation:** Bundles data and methods into a single unit (class) while restricting direct access to data.
- ✓ **Inheritance:** Enables a class (child) to inherit properties and methods from another class (parent), promoting code reuse.
- ✓ **Polymorphism:** Allows methods to perform differently based on the object context (e.g., method overloading and overriding).
- ✓ **Abstraction:** Hides implementation details and exposes only essential features, simplifying system design**.**

**Project related concepts:**

**Classes :**

- Classes in OOP are blueprints for creating objects. Here , Electricity Billing System Swing is a class. It encapsulates the logic for calculating electricity bills, handling user input, and displaying output.

**Method:**

- Methods define the behavior or actions that a class can perform. In the code, the method calculateBill() calculates the total bill based on the number of units consumed**.**

## Error Handling:

### try-catch Block:

- **Error handling** is crucial for ensuring the program does not crash when unexpected input is provided. In the code, a try-catch block is used in the actionPerformed() method to catch any NumberFormatException errors that occur if the user enters non-numeric values in the "Units Consumed" field.

## Input Validation:

- **Input validation** ensures that the data entered by the user is correct. In the code, this is handled by the try-catch block, where the program ensures that the units consumed input is a valid numeric value. If it's not, an error message is shown, alerting the user to provide valid input.

## Mathematical Operations:

### GST Calculation:

- **GST (Goods and Services Tax)** is calculated as 18% of the total electricity bill.

### Arithmetic Operations:

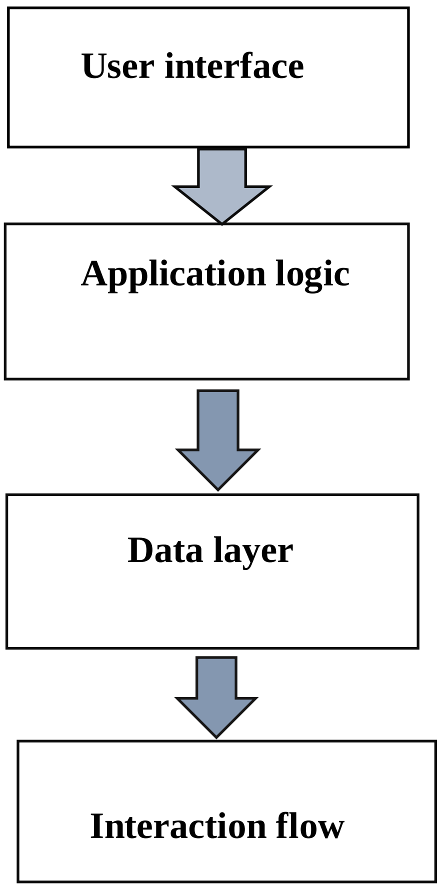
- The code uses various **arithmetic operations** to calculate the electricity bill based on units consumed:
    - **Multiplication** to calculate the cost for each tier (e.g., unitsConsumed * 5 for the first 100 units).
    - **Addition** to sum up the costs for each tier (e.g., 100 * 5 + (unitsConsumed - 100) * 7).

# CHAPTER 2
# PROJECT METHODOLOGY

## 2.1 Proposed Work

The Electricity Billing System can be enhanced in several ways to improve its functionality, user experience, and maintainability. Firstly, the user interface (UI) can be improved by adding better input validation, tooltips, and help text for clearer guidance on entering data. Implementing a Clear button to reset input fields and output labels would also enhance usability. Additionally, specific error messages can be introduced to give users better feedback on incorrect inputs. A bill history feature could be added to track previous bills, and multiple tariff plans could be incorporated to cater to different customer categories like residential, commercial, or industrial users. This would allow the user to select a tariff plan and calculate the bill based on that selection.

## 2.2 Block Diagram



# CHAPTER 3
# MODULE DESCRIPTION

## 3.1 Module 1 Heading

### Bill Calculation:

The Bill Calculation Module handles the core logic for calculating the electricity bill based on the units consumed by the customer. The system follows a tiered pricing structure, applying different rates depending on the amount of electricity consumed. For example, units up to 100 are billed at Rs. 5 per unit, the next 100 units are charged at Rs. 7 per unit, the next 100 units at Rs. 10 per unit, and any units above 300 are charged at Rs. 12 per unit. The method uses conditional statements (if-else) to apply the correct rate to each tier and calculates the total bill accordingly. This calculation is returned to the user for further processing.

## 3.2 Module 2 Heading

### User Interface :

The User Interface (UI) Module is responsible for creating the graphical layout through which users interact with the application. Using Java's Swing library, the module sets up a main window (JFrame) and organizes components like labels, text fields, and buttons inside a panel (JPanel). This interface allows users to enter their customer information, such as name, customer ID, and units consumed, and then click a button to calculate the bill. The results, including the total bill amount, GST, and the total amount with GST, are displayed dynamically in the UI, providing a seamless and interactive experience for the user.

## 3.3 Module 3 Heading

**GST Calculation:**

The GST Calculation Module handles the calculation of Goods and Services Tax (GST), which is typically applied to the electricity bill at a rate of 18%. Once the basic bill is calculated, this module computes the GST amount by multiplying the bill by the GST percentage (18%). The total amount to be paid by the customer is then calculated by adding the GST to the original bill. These values are then displayed to the user on the interface, giving them a complete breakdown of the bill, including taxes.

## 3.4 Module 4 Heading

**Error Handling :**

The Error Handling Module ensures the robustness of the system by managing potential errors, particularly when users input invalid data. For example, if the user enters non-numeric characters for the units consumed, the program catches the resulting NumberFormatException and displays an error message to the user, prompting them to enter valid data. This helps maintain the stability of the application, preventing it from crashing and providing the user with clear feedback on what went wrong.

# CHAPTER 4
# CONCLUSION & FUTURE SCOPE

## 4.1 CONCLUSION

In conclusion, the Electricity Billing System built with Java and Swing provides an efficient, user-friendly solution for calculating electricity bills based on consumption. The system is designed with four key modules: Bill Calculation, User Interface (UI), GST Calculation, and Error Handling. Each module serves a distinct function that ensures the smooth operation of the application, from calculating the tiered billing amount to applying taxes and displaying results in an interactive graphical interface.The Bill Calculation Module accurately computes the bill based on different consumption tiers, while the UI Module allows users to input data and view the results clearly. The GST Calculation Module adds transparency by including the GST on the total bill, and the Error Handling Module ensures the program remains stable and user-friendly, even when invalid input is provided.By leveraging Java's Swing library for the user interface and handling essential tasks such as calculations and error management, the system offers a seamless experience for users. This modular design also ensures the system is maintainable, extensible, and adaptable for future improvements, such as adding more features or modifying pricing structures. Overall, the Electricity Billing System demonstrates a clear and structured approach to solving real-world problems using object-oriented programming principles**.**

## 4.2 FUTURE SCOPE

Further enhancements could include creating a user account system to track billing history, offering energy consumption insights, and providing mobile application support for greater accessibility. The system could also be expanded to manage multiple utilities like water, gas, and internet bills, offering a consolidated view of all utility charges. Implementing security features like data encryption and user authentication would ensure that sensitive information is protected.

## REFERENCES

**Java Books:**

**"Effective Java" by Joshua Bloch**

A comprehensive book on best practices in Java programming. It covers topics such as object-oriented design, error handling, and Java collections, all of which are crucial for building robust and efficient applications like the Electricity Billing System.

**Link**: Effective Java.

**"Head First Java" by Kathy Sierra and Bert Bates**

An excellent book for beginners, explaining core Java concepts such as classes, objects, methods, and error handling in an engaging way. It provides a strong foundation for developing Java applications.

**Link**: Head First Java

**Websites:**

**GeeksforGeeks - Java Tutorials**

**URL**: https://www.geeksforgeeks.org/java/

A comprehensive collection of tutorials on Java, covering topics like classes, objects, inheritance, encapsulation, and more. Great for learning the core concepts of Java and applying them in projects like EPMS.

**W3Schools - Java Tutorial**

**URL:** https://www.w3schools.com/java/

A beginner-friendly resource that offers tutorials on Java programming, including object-oriented principles and core Java concepts.

# APPENDIX A
# (SOURCE CODE)

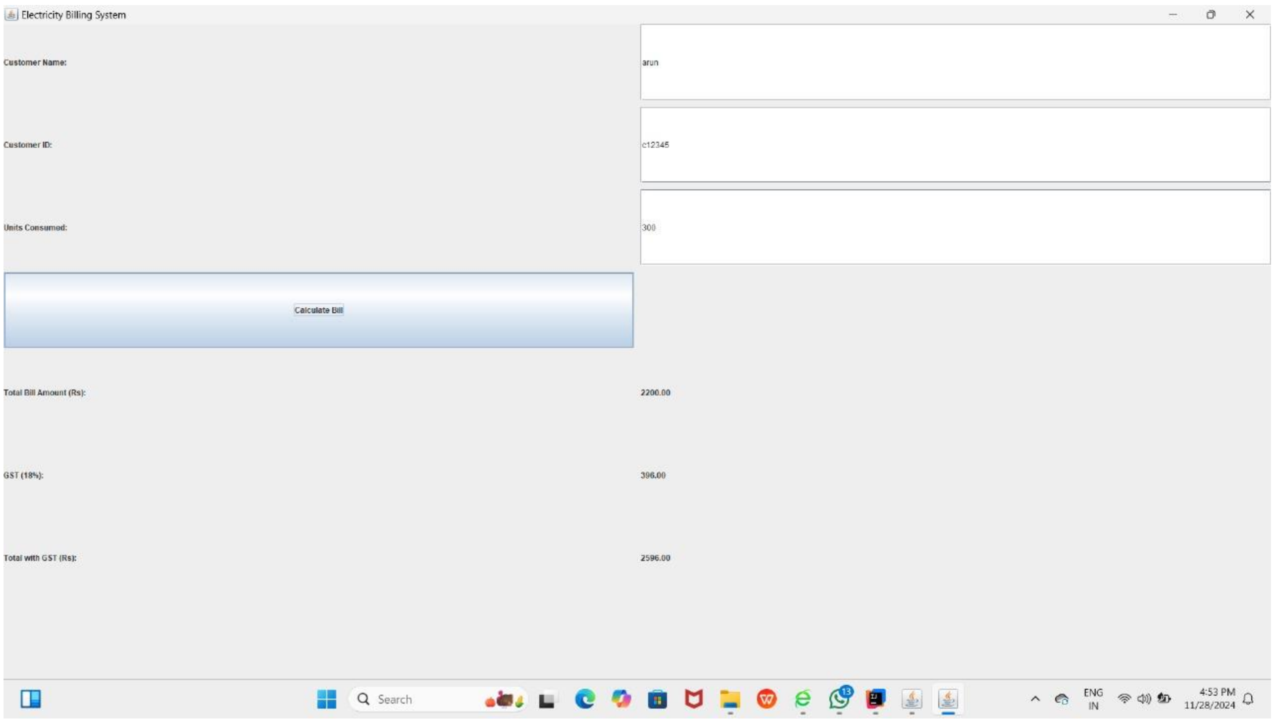
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ElectricityBillingSystemSwing {

    // Method to calculate the electricity bill based on units consumed
    public static double calculateBill(double unitsConsumed) {
        double bill = 0;
        if (unitsConsumed <= 100) {
            bill = unitsConsumed * 5; // Rs. 5 per unit for the first 100 units
        } else if (unitsConsumed <= 200) {
            bill = 100 * 5 + (unitsConsumed - 100) * 7; // Rs. 7 per unit for 101-200 units
        } else if (unitsConsumed <= 300) {
            bill = 100 * 5 + 100 * 7 + (unitsConsumed - 200) * 10; // Rs. 10 per unit for 201-300 units
        } else {
            bill = 100 * 5 + 100 * 7 + 100 * 10 + (unitsConsumed - 300) * 12; // Rs. 12 per unit for above 300
units
        }
        return bill;
    }

    public static void main(String[] args) {
        // Create the main frame
        JFrame frame = new JFrame("Electricity Billing System");
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create panels for the layout
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(8, 2, 10, 10));
```

```
// Add components for input fields
JLabel nameLabel = new JLabel("Customer Name:");
JTextField nameField = new JTextField();

JLabel idLabel = new JLabel("Customer ID:");
JTextField idField = new JTextField();

JLabel unitsLabel = new JLabel("Units Consumed:");
JTextField unitsField = new JTextField();

JButton calculateButton = new JButton("Calculate Bill");

JLabel billLabel = new JLabel("Total Bill Amount (Rs):");
JLabel billAmountLabel = new JLabel();

JLabel gstLabel = new JLabel("GST (18%):");
JLabel gstAmountLabel = new JLabel();

JLabel totalLabel = new JLabel("Total with GST (Rs):");
JLabel totalWithGstLabel = new JLabel();

// Add components to the panel
panel.add(nameLabel);
panel.add(nameField);
panel.add(idLabel);
panel.add(idField);
panel.add(unitsLabel);
panel.add(unitsField);
panel.add(calculateButton);
panel.add(new JLabel()); // Empty cell
panel.add(billLabel);
panel.add(billAmountLabel);
panel.add(gstLabel);
panel.add(gstAmountLabel);
panel.add(totalLabel);
panel.add(totalWithGstLabel);
```

```
        // Add panel to frame
        frame.add(panel);

        // Button action listener to calculate the bill
        calculateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    String customerName = nameField.getText();
                    String customerId = idField.getText();
                    double unitsConsumed = Double.parseDouble(unitsField.getText());

                    double billAmount = calculateBill(unitsConsumed);
                    double gst = billAmount * 0.18;
                    double totalWithGST = billAmount + gst;

                    billAmountLabel.setText(String.format("%.2f", billAmount));
                    gstAmountLabel.setText(String.format("%.2f", gst));
                    totalWithGstLabel.setText(String.format("%.2f", totalWithGST));

                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(frame, "Please enter valid inputs.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        // Set frame visibility
        frame.setVisible(true);
    }
}
```

# APPENDIX B

# (SCREENSHOTS)



Electricity Billing System

| | |
|---|---|
| Customer Name: | arun |
| Customer ID: | c12345 |
| Units Consumed: | 300 |
| Total Bill Amount (Rs): | 2200.00 |
| GST (18%): | 396.00 |
| Total with GST (Rs): | 2596.00 |