**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter** | **Chapter Title** | **Section Title** | **Page no.** |
|  | **ABSTRACT** | Abstract | 6 |
| **I** | **INTRODUCTION** | Overview | 7 |
| Objectives and Scope | 7 |
| Importance Of Energy Tracking System | 7 |
| Overview of Project Accomplishments | 8 |
| **II** | **PROBLEM IDENTIFICATION** | Problem Statement | 9 |
| **III** | **METHODOLOGY** | Project Design and Architecture | 10 - 11 |
| Implementation | 11 - 12 |
| Features and Functionality | 12 - 13 |
| Testing and Results | 13 - 14 |
| IV | **CONCLUSION** | Summary | 15 |
|  |  | Scope for Future Enhancements | 15-16 |
| **V** | **CODE SNIPPETS** | Code | 17 - 23 |
| Outputs Screenshot | 24 |
| **VI** | **BIBLOGRAPHY** | | 25 |

# ABSTRACT

This project aims to develop an interactive and user-friendly "Electricity Consumption Calculator" to empower users in managing and optimizing their household energy consumption. The primary objective is to provide users with a tool that facilitates real-time calculations, fosters energy awareness, and contributes to sustainable energy practices.

Inefficient energy consumption and a lack of awareness about electricity usage patterns pose significant challenges for individuals seeking to manage their energy costs and reduce environmental impact. Existing tools often lack user-friendliness and fail to provide instantaneous feedback, hindering informed decision-making.

The project adopts a Java-based approach, utilizing JavaFX for an intuitive graphical interface. It incorporates predefined appliance data, allowing users to input information such as the number of appliances, usage time, and duration. Real-time calculations provide immediate feedback on electricity consumption and costs. The system supports appliance customization, enabling users to modify predefined data or add entirely new devices. Robust testing methodologies, including unit tests and user acceptance testing, ensure the accuracy, usability, and responsiveness of the application.

Anticipated outcomes include precise calculations, an intuitive user interface, and valuable insights garnered from user feedback. The solution is poised to enhance energy awareness, empower users to make informed decisions, and contribute to sustainable energy practices in residential and commercial settings.

Top of Form

# INTRODUCTION

### 1.1 Overview of the Project:

The "Electricity Consumption Calculator" project aims to address the critical need for efficient energy management in households. As the demand for electricity continues to rise, it becomes imperative for individuals to monitor and understand their electricity consumption patterns. This project introduces a user-friendly JavaFX application designed to assist users in estimating the electricity consumption and associated costs of various household appliances.

### 1.2 Objectives and Scope:

The primary objective of this project is to develop a functional news aggregator utilizing Java programming language and JavaFX for the user interface. The project focuses on integrating these technologies to create a seamless and user-friendly energy tracking system.

"Electricity Consumption Calculator" project aims to achieve a balance between practical utility and educational value. Through a user-friendly interface and a diverse range of applicable appliances, the project seeks to empower users to make informed decisions about their energy consumption in both a practical and educational context.

### 1.3 Importance of Energy Tracking System:

An energy tracking system holds paramount importance across diverse domains. By meticulously monitoring energy consumption, it enables significant cost savings through the identification and rectification of inefficiencies, facilitating accurate budgeting and financial planning. Moreover, the system plays a pivotal role in environmental sustainability, empowering users to actively reduce their carbon footprint and adopt resource-conserving practices. For businesses, energy tracking enhances operational efficiency by optimizing processes, predicting maintenance needs, and ensuring compliance with regulatory standards. Transparent reporting of energy usage not only aligns with corporate social responsibility goals but also incentivizes behavioral changes towards energy conservation. The system contributes to grid management and emergency preparedness, ensuring a reliable and resilient energy infrastructure. Ultimately, the multifaceted benefits of an energy tracking system underscore its critical role in promoting economic savings, environmental stewardship, and operational excellence.

**1.4 Overview of Project Accomplishments:**

Throughout the course of this project, we have successfully designed and implemented an interactive interface using JavaFX, allowing users to input details for a diverse range of household appliances. The application facilitates real-time calculations, offering instantaneous feedback on electricity consumption and costs. One of the project's notable achievements is the integration of predefined appliance data, allowing users to customize information and even add entirely new devices, providing unparalleled flexibility.

In addition to the user-centric design, the project team has rigorously tested the application through unit tests and user acceptance testing. This testing phase ensures the accuracy, usability, and responsiveness of the "Electricity Consumption Calculator." Further, the project emphasizes the importance of user feedback and aims to continuously enhance the application based on insights gathered from users.

As we delve into the various aspects of this project, including its features, functionalities, and testing methodologies, the overarching goal is to provide users with a powerful yet accessible tool for managing and optimizing their household energy consumption. The subsequent sections of this report will delve into the specific details of the project, outlining the methodologies employed, features implemented, and the expected outcomes of this innovative "Electricity Consumption Calculator."

# PROBLEM IDENTIFICATION

### 2.1 Problem Statement:

The absence of a comprehensive and user-friendly energy tracking system poses a significant challenge, hindering individuals and organizations from effectively managing and optimizing their energy consumption. Without a streamlined solution, identifying and rectifying inefficiencies becomes challenging, leading to unnecessary costs and environmental impact. Businesses lack the tools necessary to strategically plan and reduce peak energy usage, resulting in operational inefficiencies and increased expenses. Moreover, the absence of transparent energy reporting impedes regulatory compliance and hinders efforts towards corporate social responsibility. In the absence of a robust energy tracking system, there is a critical gap in enabling informed decision-making, promoting sustainable practices, and contributing to a resilient and efficient energy ecosystem. Addressing this problem is crucial for realizing cost savings, enhancing environmental stewardship, and achieving operational excellence.

# METHODOLOGY

**3.1 Project Design and Architecture**

### 3.1.1 Project planning and scope definition:

Objective Refinement here means clarified project objectives, emphasizing the development of a user-friendly Electricity Consumption Calculator for effective energy tracking and scope definition here is to clearly outlined the scope of the project, including features such as real-time calculations, predefined appliance data, and a JavaFX-based user interface.

### 3.1.2 User Interface (UI):

In the "Electricity Consumption Calculator" project, JavaFX was instrumental in crafting a dynamic and user-friendly interface. Leveraging JavaFX components like labels, text fields, and buttons, we organized the UI elements using layout panes such as **Grid Pane** and **V Box** for an intuitive grid-based structure. The versatility of JavaFX allowed us to create a visually appealing design, further enhanced through CSS styling for consistent aesthetics. Labels provided clear indications for user input, while text fields allowed seamless data entry. Buttons, crucial for user interaction, were strategically placed for accessibility. The responsive layout, achievable through JavaFX, ensured adaptability across different screen sizes. Altogether, JavaFX facilitated the creation of an interactive and aesthetically pleasing UI, enhancing the overall user experience in the "Electricity Consumption Calculator" application.

### 3.1.3 System Design:

Functional Requirements Specification is to define the detailed functional requirements, including data input, real-time calculations, and result presentation and user interface design is created wireframes and prototypes to design an intuitive JavaFX-based interface, emphasizing simplicity and accessibility.

**3.1.4 Data Preprocessing:**

Predefined Appliance Data is a compiled list of common household appliances with associated

power ratings and electricity rates, incorporating this data into the application for user convenience.

### 3.1.5 Application Development:

We Implemented and developed the application using JavaFX, utilizing its features for graphical interface design, event handling, and seamless user interaction and the algorithm is implemented for real-time calculations of electricity consumption and associated costs based on user input.

## 3.2 Implementation:

### 3.2.1 Technology Stack:

We Utilized Java for its versatility, platform independence, and extensive community support. User Interface Framework is employed JavaFX for GUI development, leveraging its rich set of components and ease of integration with Java applications.

**3.2.1 Data Model:**

Appliance data list is implemented to store predefined appliance data, including names, power ratings, and electricity rates and dynamic data storage is allowed for dynamic storage of user-inputted data, such as the number of appliances, usage times, and number of days.

### 3.2.2 User Interface Development with JavaFX:

We Utilized JavaFX components such as labels, text fields, buttons, and layout panes for constructing an intuitive and visually appealing interface and applied CSS for basic styling, ensuring a consistent and visually appealing appearance.

### 3.2.3 Real Time Calculations:

We implemented event handlers for user input fields to capture real-time changes and calculation logic is developed for calculating electricity consumption and cost based on user inputs, considering predefined appliance data.

### 3.2.4 Predefined Appliance Data:

We employed arrays and lists to store predefined appliance data, ensuring ease of access and modification and for initialization of process we integrated a process to load default appliance data at the start of the application.

### 3.2.5 User Input Handling:

We implemented event listeners on input fields for the number of appliances, usage time, and number of days to capture and process user input and we incorporated exception handling to manage potential errors in user input, providing a smoother user experience.

**3.2.6 Testing:**

Conducted unit tests to ensure the functionality of individual methods and classes and User Acceptance Testing (UAT) to engaged potential users in sessions to validate the application's usability and identify any user experience issues.

**3.3 Features and Functionality:**

**3.3.1 Appliance Selection:**

Predefined Appliance Data: Users can choose from a predefined list of common household appliances, each with associated power ratings and electricity rates. Dynamic Addition: The application allows users to dynamically add and customize appliances beyond the predefined list.

**3.3.2 User Input:**

**Number of Appliances:** Users can input the quantity of each selected appliance.

**Usage Time:** Users specify the average daily usage time (in hours) for each appliance.

**Number of Days:** Users input the total number of days for which they want to calculate electricity

consumption and cost.

**3.3.3 Real-Time Calculations:**

The application dynamically calculates the total electricity consumption based on the entered data, considering power ratings, usage times, and the number of appliances and Real-time cost estimation is provided, taking into account electricity rates and the calculated consumption.

**3.3.4 User-Friendly Interface:**

The JavaFX-based interface is designed for simplicity and ease of use. Prompts and placeholders guide users on what information is required in each input field.

**3.4 Testing and Results:**

## 3.4.1 Testing Approach:

## Objective: Verify the functionality of individual methods and classes.

## Tests: Confirm accurate calculation of electricity consumption and cost.

## Validate input validation and error handling.

## Ensure proper updating of labels and real-time feedback.

## 3.4.2 User Acceptance Testing (UAT):

## Objective: Evaluate the application's usability and user experience.

## Tests: Verify the intuitiveness of the user interface for inputting data.

## Assess the real-time updating of labels and feedback.

## Evaluate the customization features, including adding custom appliances.

## 3.4.3 Expected Results:

## Calculation Accuracy: Unit tests should confirm that the calculations for electricity consumption and cost are accurate, considering predefined and user-inputted data.

## Input Validation: Validate that the application appropriately handles valid and invalid numeric inputs, providing clear error messages when necessary.

## Label Updating: Confirm that labels dynamically update in real-time as users input data, providing immediate and accurate feedback.

## 3.4.4 Performance Evaluation:

## Response Time:

## Objective: Assess the application's responsiveness under varying levels of user input.

## Measurement: Measure the time taken for the application to recalculate and update results as users modify input fields.

## Resource Utilization:

## Objective: Evaluate the application's resource usage during regular operations.

## Measurement: Monitor memory and CPU usage to ensure the application operates efficiently.

# CONCLUSION

**4.1 Conclusion:**

The Electricity Consumption Calculator is a comprehensive tool designed to assist users in estimating their electricity usage and associated costs based on the usage patterns of various appliances. By providing a user-friendly interface and real-time calculations, the application empowers users to make informed decisions about energy consumption and cost management.

During the development of this project, several key features were implemented, including the input of appliance details, dynamic calculations, and the display of individual and total electricity costs. The integration of JavaFX ensured a responsive and intuitive user interface.

**4.2 Scope for Future Enhancements:**

While the current version of the Electricity Consumption Calculator successfully meets its objectives, there are several areas where enhancements and additional features could be implemented to further improve its functionality and user experience. The scope for future enhancements includes:

**4.2.1 Real-time Data Updates:** Implement a mechanism to receive real-time updates on electricity rates and appliance power consumption. This could involve integrating with external APIs or databases that provide live data.

**4.2.2 User Authentication and Profile Management:** Introduce user authentication and profile management to allow users to save and retrieve their appliance data for future reference. This could enhance the personalization of the application.

**4.2.3 Usage History and Analytics:** Incorporate a feature to track and visualize historical electricity usage data. Users could benefit from insights and analytics on their consumption patterns over time.

**4.2.4 Notification System:** Implement a notification system to alert users when their electricity consumption is significantly higher than usual or when electricity rates change. This proactive approach can assist users in managing their energy usage effectively.

**4.2.5 Energy Saving Recommendations:** Integrate machine learning algorithms or predefined rules to provide users with personalized recommendations on how to optimize their energy consumption and reduce costs.

**4.2.6 Multi-Platform Support:** Extend the application to support multiple platforms, including mobile devices and web browsers, to increase accessibility and reach a broader user base.

**4.2.7 Localization:** Introduce localization features to support multiple languages and regional settings, making the application more user-friendly for a diverse audience.

**4.2.8 Integration with Smart Devices:** Explore integration with smart home devices to automate the collection of appliance usage data, further reducing manual input and enhancing accuracy. Incorporating these enhancements will not only elevate the capabilities of the Electricity Consumption Calculator but also position it as a more versatile and valuable tool for users seeking real-time insights into their energy consumption habits.

**Code Snippets**

**package com.example.firstproject;**

**import javafx.application.Application;**

**import javafx.geometry.Insets;**

**import javafx.geometry.Pos;**

**import javafx.scene.Scene;**

**import javafx.scene.control.\*;**

**//import javafx.scene.layout.Background;**

**import javafx.scene.layout.GridPane;**

**import javafx.scene.layout.StackPane;**

**import javafx.scene.layout.VBox;**

**import javafx.stage.Stage;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class ElectricityConsumptionCalculatorFX extends Application {**

**private final List<String> applianceNames = new ArrayList<>();**

**private final List<Double> powerRatings = new ArrayList<>();**

**private final List<Integer> numberOfAppliances = new ArrayList<>();**

**private final List<Double> usageTimes = new ArrayList<>();**

**private final List<Double> electricityRates = new ArrayList<>();**

**private final List<Integer> numberOfDays = new ArrayList<>();**

**private final Label totalCostLabel = new Label();**

**private final List<Label> consumptionLabels = new ArrayList<>();**

**private final List<Label> costLabels = new ArrayList<>();**

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

**launch(args);**

**}**

**@Override**

**public void start(Stage primaryStage) {**

**primaryStage.setTitle("Electricity Consumption Calculator");**

**GridPane grid = new GridPane();**

**grid.setHgap(10);**

**grid.setVgap(10);**

**grid.setPadding(new Insets(20, 20, 20, 20));**

**Label titleLabel = new Label("Welcome to the Electricity Consumption Calculator (Standard values for India)");**

**titleLabel.setStyle("-fx-font-size: 16pt;");**

**GridPane.setConstraints(titleLabel, 0, 0, 4, 1);**

**initializeAppliances();**

**for (int i = 0; i < applianceNames.size(); i++) {**

**addUIComponents(grid, i);**

**}**

**Button calculateButton = new Button("Calculate");**

**calculateButton.setOnAction(e -> calculateTotalBill());**

**GridPane.setConstraints(calculateButton, 0, applianceNames.size() + 2);**

**totalCostLabel.setStyle("-fx-font-size: 14pt;");**

**GridPane.setConstraints(totalCostLabel, 0, applianceNames.size() + 3, 4, 1);**

**VBox contentLayout = new VBox(20); // Vertical box with 20-pixel spacing**

**contentLayout.setAlignment(Pos.CENTER);**

**contentLayout.getChildren().addAll(titleLabel, grid, calculateButton, totalCostLabel);**

**StackPane root = new StackPane(contentLayout);**

**Scene scene = new Scene(root, 800, 600);**

**primaryStage.setScene(scene);**

**primaryStage.show();**

**}**

**private void initializeAppliances() {**

**String[] predefinedAppliances = {**

**"Ceiling Fan", "Incandescent Bulb", "CFL Bulb", "LED Bulb", "Refrigerator",**

**"Television", "Air Conditioner", "Washing Machine", "Microwave Oven", "Electric Heater",**

**"Laptop", "Desktop Computer", "Router", "Water Heater", "Coffee Maker"**

**};**

**double[] predefinedPowerRatings = {**

**0.075, 0.1, 0.015, 0.005, 0.15, 0.1, 1.5, 0.8, 1.2, 2.0,**

**0.05, 0.1, 0.02, 2.0, 1.0**

**}; // Power ratings in kW**

**double[] predefinedElectricityRates = {**

**5.0, 3.0, 4.0, 2.0, 7.0, 4.0, 8.0, 7.0, 5.0, 6.0,**

**2.0, 4.0, 1.0, 7.0, 5.0**

**}; // Electricity rates per kWh**

**for (int i = 0; i < predefinedAppliances.length; i++) {**

**applianceNames.add(predefinedAppliances[i]);**

**powerRatings.add(predefinedPowerRatings[i]);**

**electricityRates.add(predefinedElectricityRates[i]);**

**// Initialize lists with default values**

**numberOfAppliances.add(0);**

**usageTimes.add(0.0);**

**numberOfDays.add(0);**

**}**

**}**

**private void addUIComponents(GridPane grid, int index) {**

**Label nameLabel = new Label(applianceNames.get(index));**

**GridPane.setConstraints(nameLabel, 0, index + 1);**

**TextField numAppliancesField = new TextField();**

**numAppliancesField.setPromptText("Number of Appliance");**

**GridPane.setConstraints(numAppliancesField, 1, index + 1);**

**TextField usageTimeField = new TextField();**

**usageTimeField.setPromptText("Usage per day (hours)");**

**GridPane.setConstraints(usageTimeField, 2, index + 1);**

**TextField numDaysField = new TextField();**

**numDaysField.setPromptText("Number of Days");**

**GridPane.setConstraints(numDaysField, 3, index + 1);**

**Label consumptionLabel = new Label();**

**consumptionLabels.add(consumptionLabel);**

**GridPane.setConstraints(consumptionLabel, 4, index + 1);**

**Label costLabel = new Label();**

**costLabels.add(costLabel);**

**GridPane.setConstraints(costLabel, 5, index + 1);**

**final String applianceName = applianceNames.get(index);**

**grid.getChildren().addAll(nameLabel, numAppliancesField, usageTimeField, numDaysField, consumptionLabel, costLabel);**

**numAppliancesField.setOnKeyReleased(e -> {**

**try {**

**numberOfAppliances.set(index, Integer.parseInt(numAppliancesField.getText()));**

**} catch (NumberFormatException | IndexOutOfBoundsException ignored) {**

**// Handle the exception or ignore it**

**}**

**});**

**usageTimeField.setOnKeyReleased(e -> {**

**try {**

**usageTimes.set(index, Double.parseDouble(usageTimeField.getText()));**

**} catch (NumberFormatException | IndexOutOfBoundsException ignored) {**

**// Handle the exception or ignore it**

**}**

**});**

**numDaysField.setOnKeyReleased(e -> {**

**try {**

**numberOfDays.set(index, Integer.parseInt(numDaysField.getText()));**

**} catch (NumberFormatException | IndexOutOfBoundsException ignored) {**

**// Handle the exception or ignore it**

**}**

**});**

**}**

**private void calculateTotalBill() {**

**double totalCost = 0;**

**for (int i = 0; i < applianceNames.size(); i++) {**

**double powerRating = powerRatings.get(i);**

**int numAppliances = numberOfAppliances.get(i);**

**double usageTime = usageTimes.get(i);**

**double electricityRate = electricityRates.get(i);**

**int numDays = numberOfDays.get(i);**

**double electricityConsumption = calculateElectricityConsumption(powerRating, numAppliances, usageTime, numDays);**

**double electricityCost = calculateElectricityCost(electricityConsumption, electricityRate);**

**totalCost += electricityCost;**

**// Update the labels for each appliance**

**consumptionLabels.get(i).setText("Consumption: " + electricityConsumption + " kWh");**

**costLabels.get(i).setText("Cost: Rs. " + electricityCost);**

**}**

**totalCostLabel.setText("Total Electricity Bill for all appliances: Rs. " + totalCost);**

**}**

**private double calculateElectricityConsumption(double powerRating, int numAppliances, double usageTime, int numDays) {**

**return powerRating \* numAppliances \* usageTime \* numDays;**

**}**

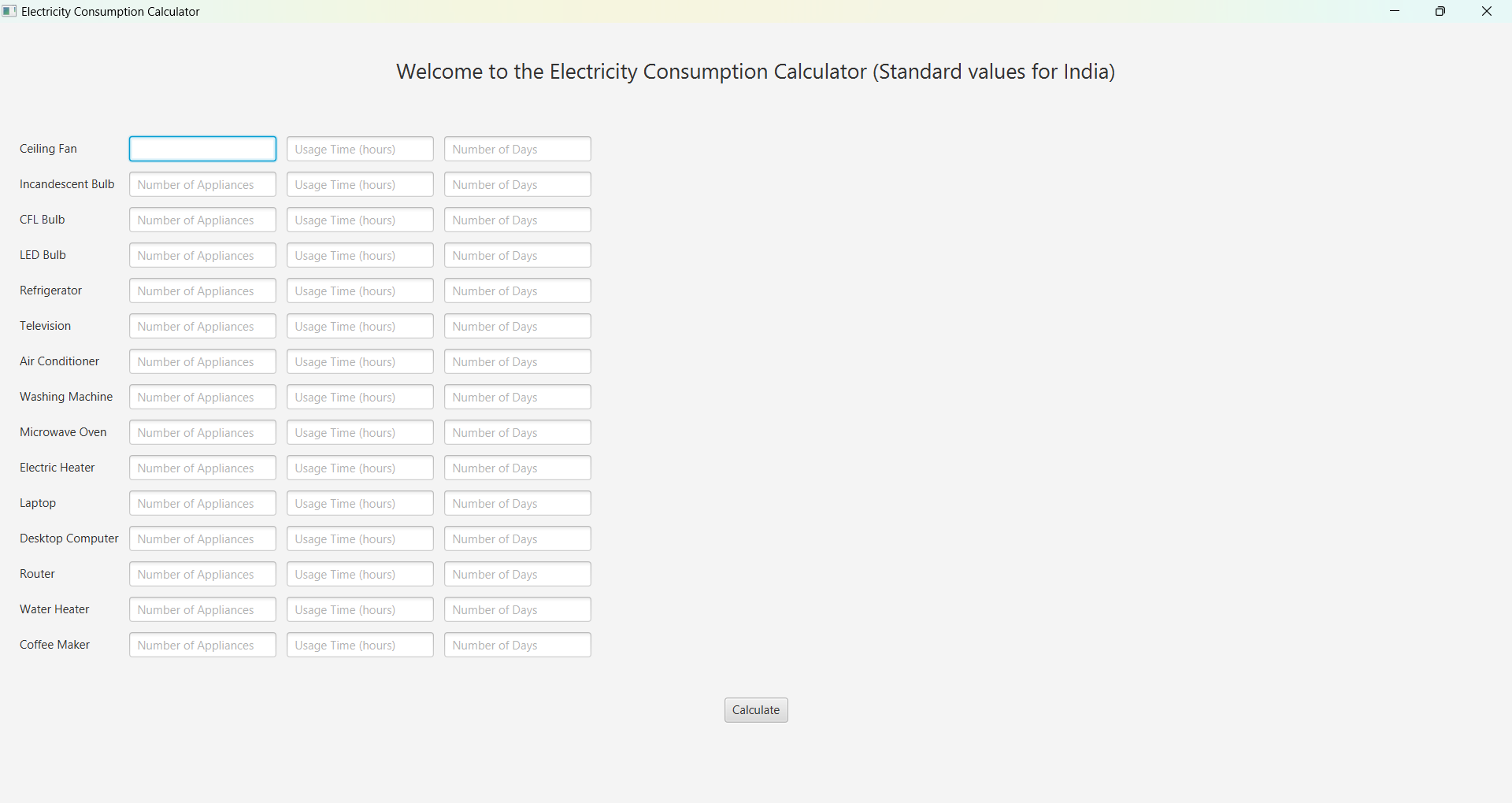
**private double calculateElectricityCost(double electricityConsumption, double electricityRate) {**

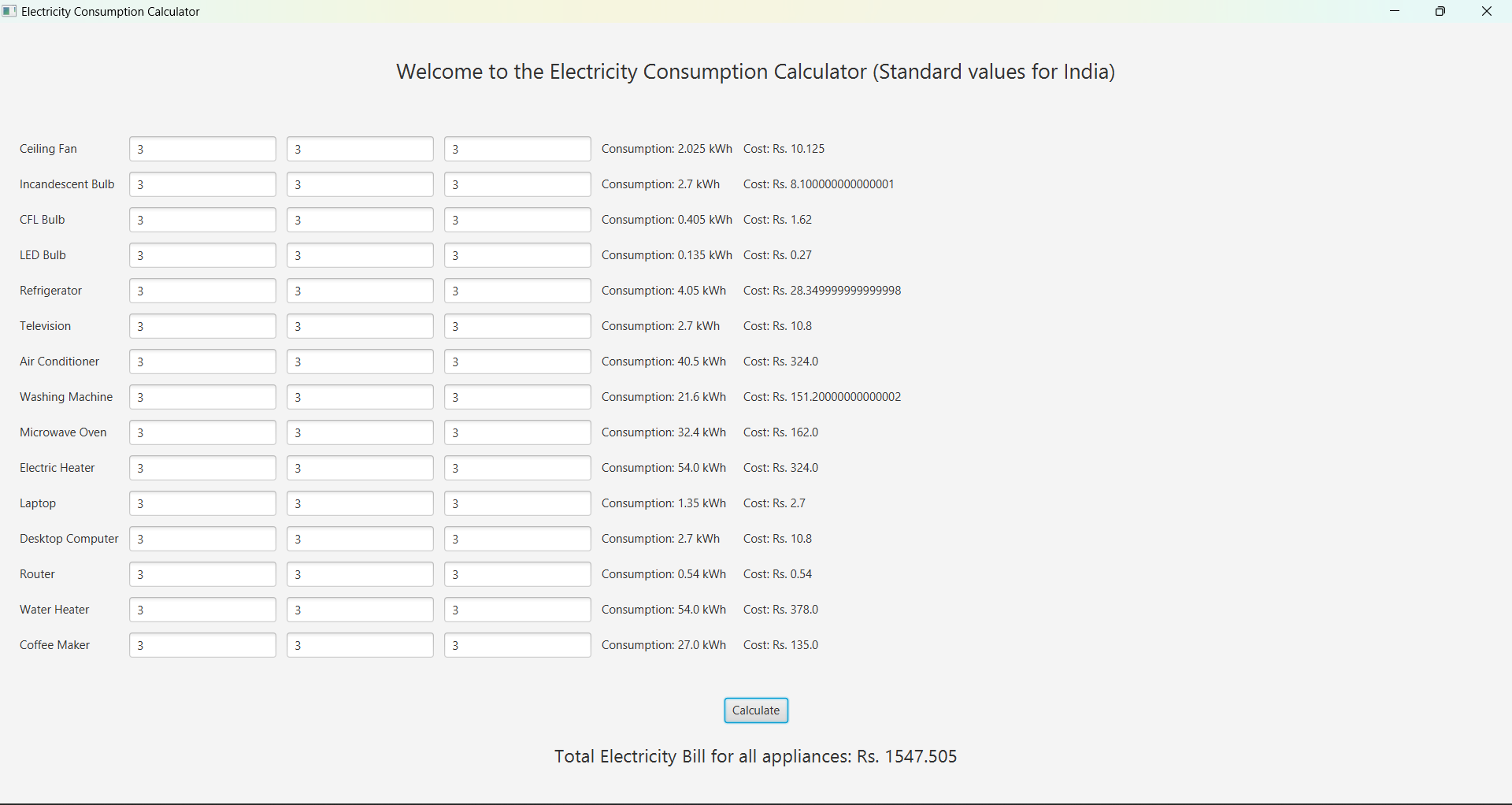
**return electricityConsumption \* electricityRate;**

**}**

**}**

**Output Screenshots**

****

****

# BIBLOGRAPHY

|  |  |  |
| --- | --- | --- |
| **1.** | JAVA Language | https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html |
| **2.** | JAVA FX (JFX) | https://docs.oracle.com/javafx/2/ |