



# Energy Billing System

**Module Title:** Visual Object Software

**Module Code:** CSYM025

**Tutor Name:** Suraj Ajit

**Assignment Title:** Energy Billing System

**Student Name:** Parastoo Shojaei Sarcheshmeh

**Student ID:** 25828319

## Contents

1	Login Details (Username and Password) .....	3
2	UML Class Diagram.....	3
3	User Stories .....	4
4	Design and Implementation Process .....	5
4.1	Phase 1 – Console Application .....	5
4.2	Phase 2 – JavaFX GUI Development.....	5
4.3	Phase 3 – File Storage and Data Handling.....	6
4.4	Phase 4 – Final Integration and Object-Oriented Design .....	6
5	Screenshots of the System.....	6
5.1	Login Page.....	6
5.2	Main Layout and Menu.....	7
5.3	Home Page.....	7
5.4	Customers Page.....	8
5.5	Billing Page.....	9
5.6	Bills Page.....	10
6	Evidence of Testing.....	11
6.1	Blackbox Testing .....	11
6.1.1	Blackbox Testing Screenshots .....	12
6.2	Whitebox Testing .....	23
6.3	Bugs, Weaknesses .....	25
7	References.....	26

# 1 Login Details (Username and Password)

The application starts with a login page.

The login is only used to control access to the system before the main menu is opened.

Login credentials used in this system:

- Username: admin
- Password: 1234

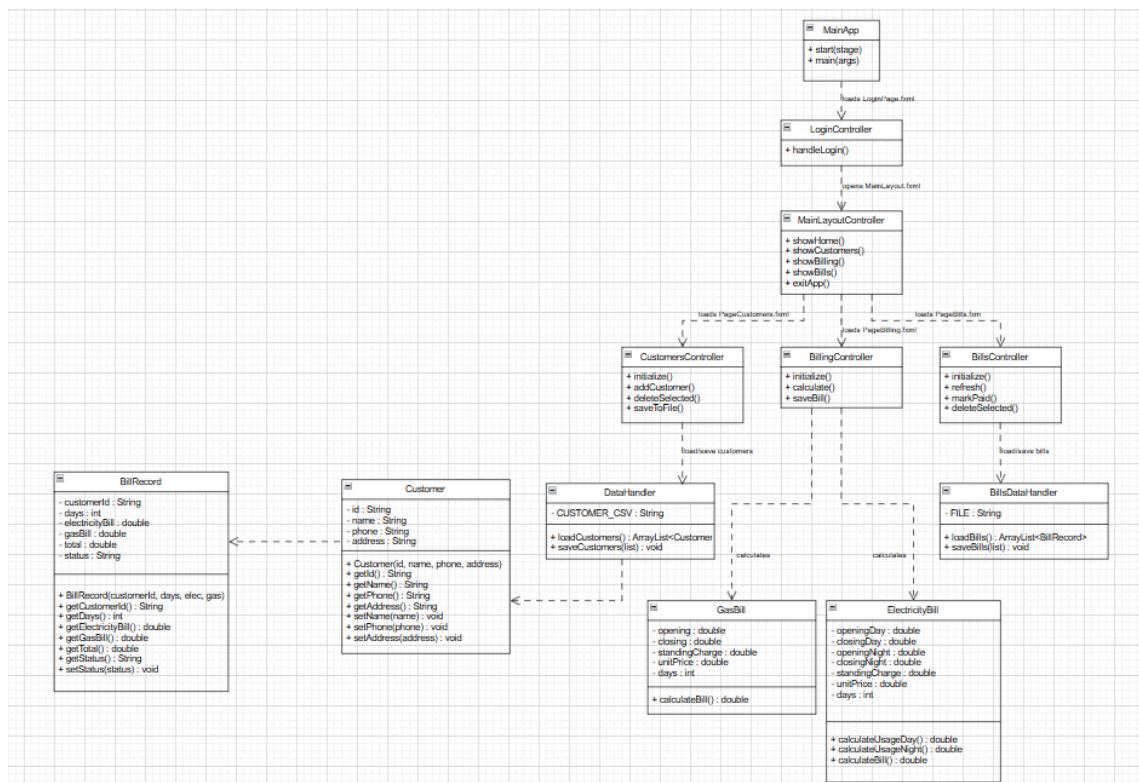
After a successful login, the system opens the main layout with the menu and all pages.

If the login details are incorrect, the system shows an error message.

The full implementation and detailed explanation of the login code are provided in Document 2 – Full Source Code Listing.

## 2 UML Class Diagram

The diagram shows the main structure of the Energy Billing System. The design is based on separating responsibilities into different classes. The model classes (Customer and BillRecord) represent the main data of the system. The calculation classes (ElectricityBill and GasBill) contain the billing formulas, while help classes (DataHandler and BillsDataHandler) manage CSV files. The controllers connect the JavaFX user interface to the logic and data. Full code and detailed explanations for all classes are provided in Document 2 – Full Source Code Listing.



### 3 User Stories

This section describes the main user stories that were implemented in the Energy Billing System.

User Story	Implemented	Comments
I want to log in to the system so that I can access the application.	Full	Simple login using username and password.
I want to see a home page after login so that I know the system is ready.	Full	Home page shows module and information.
I want to add new customers so that I can create bills for them.	Full	Customers added.
I want to view all customers in a table so that I can manage customer data easily.	Full	Customer data is stored in a CSV file
I want to delete customers so that I can remove incorrect data	Full	Delete Customers.
I want to calculate electricity and gas bills so that I can see the total cost.	Full	Calculations follow billing rules.
I want to save calculated bills so that I can access them later.	Full	Bills are stored in a CSV file.
I want to view all saved bills so that I can manage billing records.	Full	Bills shown in a Table View.
I want to mark bills as paid so that I can track payment status.	Full	Status updated and saved.
I want to delete bills so that I can remove old records.	Full	CSV file updated after deletion.
I want the system to save data to files so that information is not lost.	Full	File-based storage used.
I want to move between pages using a menu so that the system is easy to use.	Full	Handled by main layout controller.

## 4 Design and Implementation Process

This section explains how the system was designed and implemented step by step.

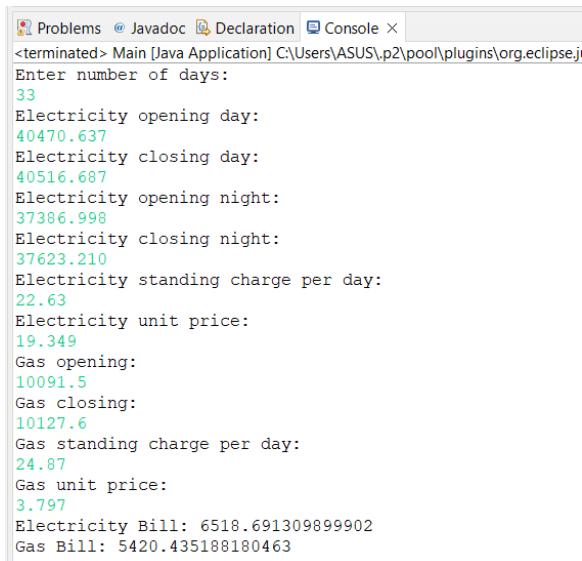
### 4.1 Phase 1 – Console Application

The project started as a simple console application. In this phase, the goal was to make sure that the electricity and gas billing calculations were correct.

The program asked the user to enter meter readings, prices, and the number of days, and then printed the electricity bill and gas bill in the console.

At this stage, there was no graphical interface.

The full console code and explanation are provided in Document 2 – Full Source Code Listing.



```
Problems @ Javadoc Declaration Console X
<terminated> Main [Java Application] C:\Users\ASUS\p2\pool\plugins\org.eclipse.jdt.core\src\com\example\billing\Billing.java
Enter number of days:
33
Electricity opening day:
40470.637
Electricity closing day:
40516.687
Electricity opening night:
37386.998
Electricity closing night:
37623.210
Electricity standing charge per day:
22.63
Electricity unit price:
19.349
Gas opening:
10091.5
Gas closing:
10127.6
Gas standing charge per day:
24.87
Gas unit price:
3.797
Electricity Bill: 6518.691309899902
Gas Bill: 5420.435188180463
```

### 4.2 Phase 2 – JavaFX GUI Development

After the console version was working correctly, the project was converted into a JavaFX application. In this phase, I designed the user interface using Scene Builder. Each page of the system was first created visually in Scene Builder by placing buttons, labels, text fields, tables, and layouts.

After designing the layout, Scene Builder automatically generated the FXML code for each page.

The controllers were used to connect the interface to the program logic. Each controller is responsible for one page only.

The main layout uses a menu to switch between pages, so the user stays in one window and the content changes dynamically.

The full JavaFX code and FXML explanations are provided in Document 2 – Full Source Code Listing.

### **4.3 Phase 3 – File Storage and Data Handling**

In the next phase, file-based storage was added to the system. Instead of using a database, I used CSV files.

Customer data is stored in customers.csv, and bill data is stored in bills.csv.

Special classes were created to handle file reading and writing.

When the application starts, data is loaded from the files, and when the user saves or updates data, the files are updated automatically.

### **4.4 Phase 4 – Final Integration and Object-Oriented Design**

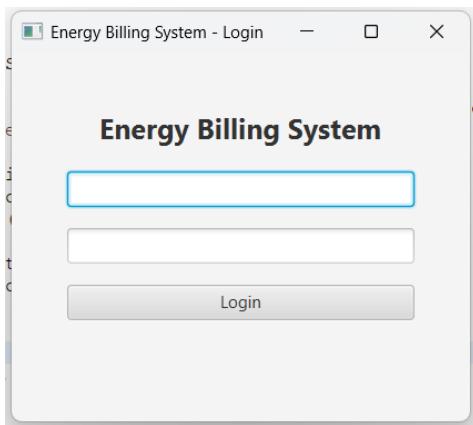
In the final, all parts of the system were connected together. The controllers, model classes, calculation classes, and file handlers were integrated to create one complete application.

Each class has a clear role: model classes store data, calculation classes perform billing logic, controllers manage user interaction, and data handler classes manage file storage.

## **5 Screenshots of the System**

This section shows screenshots of the main features of the System.

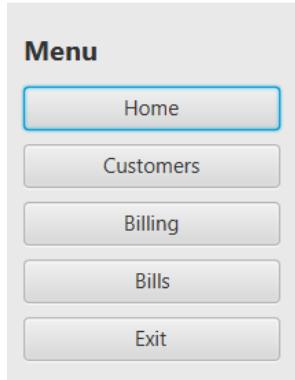
### **5.1 Login Page**



This shows the login page of the system. The user must enter a username and password to access the application. If the login details are correct, the system opens the main menu.

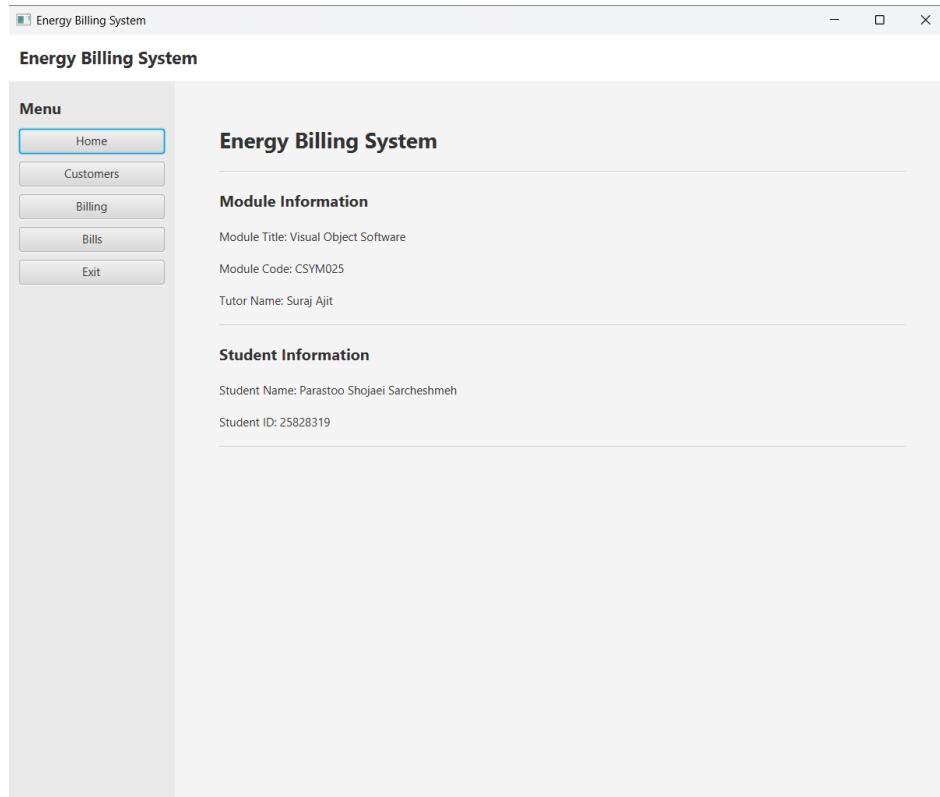
If the details are wrong, an error message is shown.

## 5.2 Main Layout and Menu



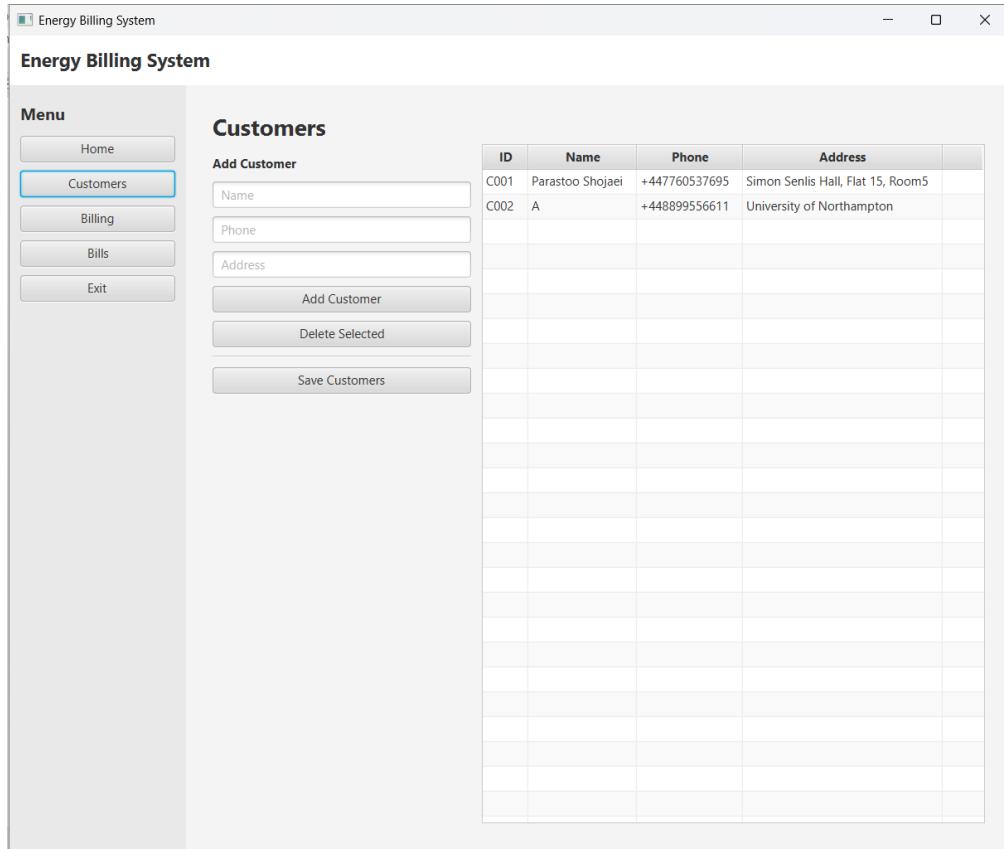
This shows the main layout of the system after login. The menu is displayed on the left side of the window and allows the user to navigate between pages.

## 5.3 Home Page

A screenshot of the Energy Billing System home page. The window title is "Energy Billing System". On the left, there is a sidebar titled "Menu" with buttons for "Home" (selected), "Customers", "Billing", "Bills", and "Exit". The main content area is titled "Energy Billing System" and contains sections for "Module Information" (Module Title: Visual Object Software, Module Code: CSYM025, Tutor Name: Suraj Ajit) and "Student Information" (Student Name: Parastoo Shojaei Sarcheshmeh, Student ID: 25828319).

The Home page is the first page shown after login.

## 5.4 Customers Page



This shows the Customers page where the user can add new customers by entering name, phone, and address. All customers are displayed in a Table View. The user can also delete customers and save the data to a CSV file.

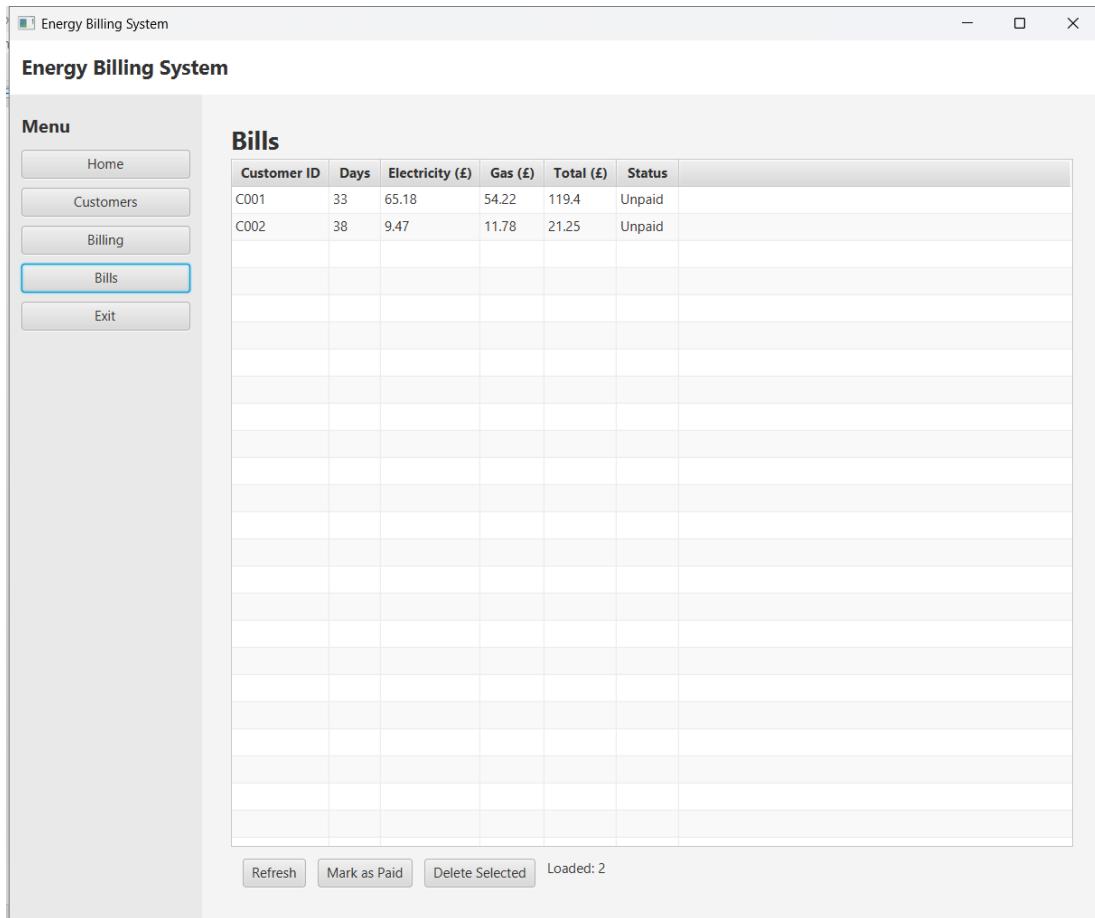
## 5.5 Billing Page

The screenshot shows the 'Billing' page of the Energy Billing System. On the left, a vertical menu bar titled 'Menu' contains five buttons: 'Home', 'Customers', 'Billing' (which is highlighted with a blue border), 'Bills', and 'Exit'. The main content area is titled 'Billing' and includes two input fields: 'Customer ID:' with a dropdown arrow and 'Days:' with an empty text box. Below this is a section titled 'Electricity Inputs' containing six input fields: 'Opening Day:', 'Closing Day:', 'Opening Night:', 'Closing Night:', 'Standing (p/day):', and 'Unit Price (p:)'. Further down is a section titled 'Gas Inputs' with four input fields: 'Opening:', 'Closing:', 'Standing (p/day):', and 'Unit Price (p:)'. At the bottom of the page are two buttons: 'Calculate' and 'Save Bill'. Below these buttons, the system displays three bill amounts: 'Electricity Bill: -', 'Gas Bill: -', and 'Total: -'.

This shows the Billing page where the user can calculate electricity and gas bills.

The user selects a customer, enters meter readings, prices, and number of days, and then clicks the calculate button. The system displays the electricity bill, gas bill, and total amount in pounds.

## 5.6 Bills Page



This screenshot shows the Bills page where all saved bills are displayed in a table.

The user can refresh the list, mark bills as paid, or delete selected bills.

## 6 Evidence of Testing

First, I did blackbox testing by using the system like a user and checking the output on the GUI. Second, I did whitebox testing using JUnit to test the main calculation classes with sample values.

### 6.1 Blackbox Testing

Blackbox testing checks the system from the user's point of view. In this testing, I entered different inputs in the GUI and checked if the system behaved correctly.

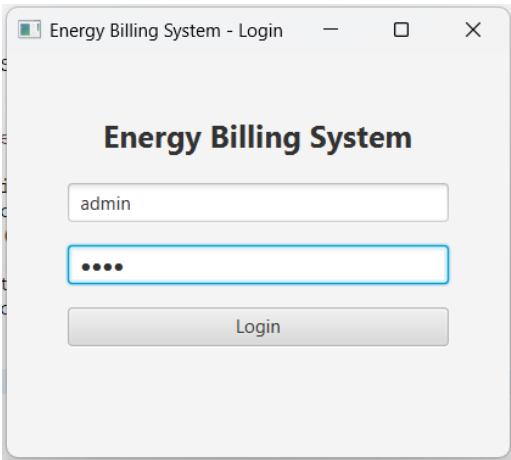
Feature Tested	Test Steps	Result	Pass/Fail
Login (correct)	Enter admin / 1234 and click Login	Main menu opens	Pass
Login (wrong)	Enter wrong details and click Login	Error message shown	Pass
Navigation	Click Home / Customers / Billing / Bills	Correct page loads	Pass
Add Customer	Fill all fields, click Add	Customer appears in table	Pass
Add Customer (missing)	Leave one field empty, click Add	Warning shown	Pass
Delete Customer	Select a customer and delete	Customer removed	Pass
Save Customers	Click Save	customers.csv updated	Pass
Billing Calculate (valid)	Enter valid numeric inputs, click Calculate	Bills shown	Pass
Billing Calculate (invalid)	Enter non-numeric input, click Calculate	"Invalid input"	Pass
Save Bill (no calculation)	Select customer but do not calculate	"Calculate first"	Pass
Save Bill (valid)	Select customer + calculate + save	Bill saved to file	Pass
Bills Refresh	Open Bills page and click Refresh	Table reloads	Pass
Mark Paid (no selection)	Click Mark Paid without selecting	Message shown	Pass
Mark Paid (selected)	Select bill row and click Mark Paid	Status becomes Paid	Pass
Delete Bill (selected)	Select bill row and delete	Removed + file updated	Pass

### 6.1.1 Blackbox Testing Screenshots

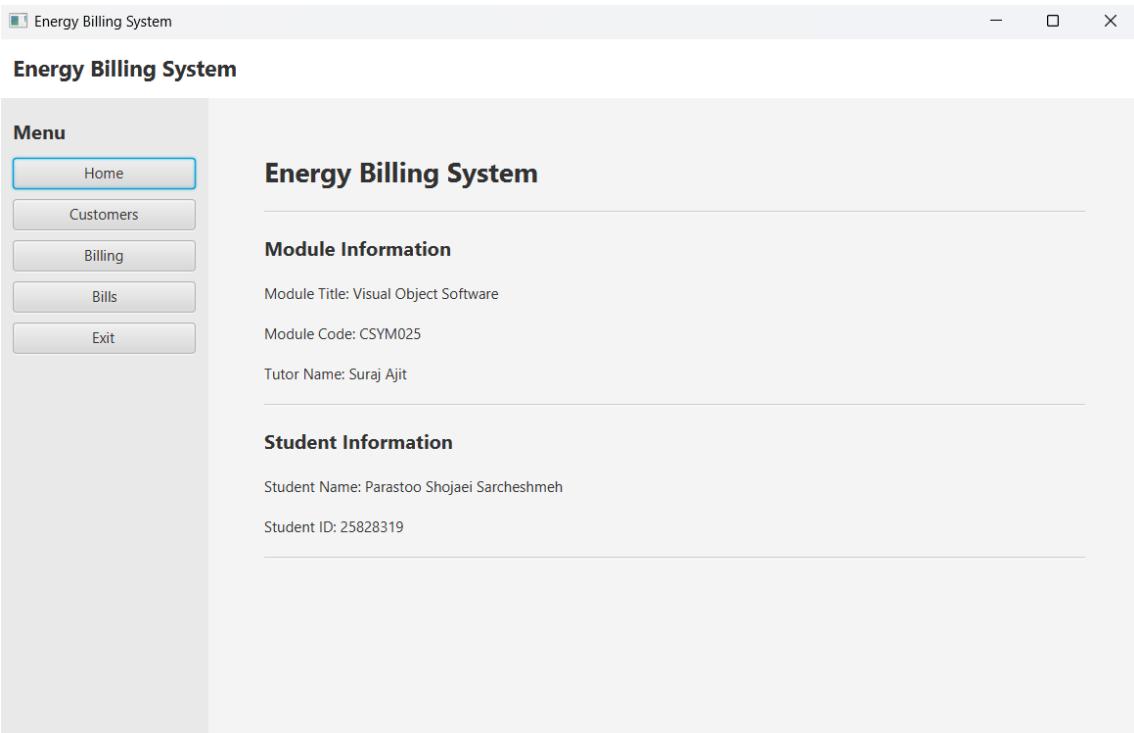
The following screenshots show evidence of the Blackbox tests listed in the table.

#### Login Success

This screenshot shows a successful login after entering the correct username and password.



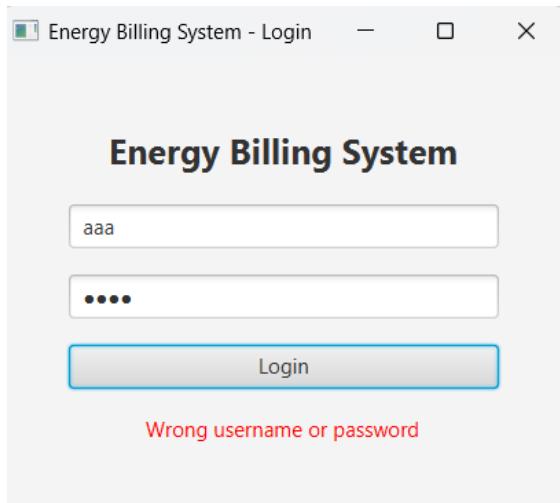
The screenshot shows the 'Energy Billing System - Login' window. It contains two text input fields: the first is labeled 'Username' and contains 'admin', and the second is labeled 'Password' and contains '\*\*\*\*\*'. Below these is a 'Login' button. The window has standard minimize, maximize, and close buttons at the top right.

The screenshot shows the main 'Energy Billing System' interface. On the left is a vertical menu bar with buttons for 'Home' (which is highlighted), 'Customers', 'Billing', 'Bills', and 'Exit'. The main content area is titled 'Energy Billing System' and contains two sections: 'Module Information' and 'Student Information'. Under 'Module Information', it lists the Module Title as 'Visual Object Software', the Module Code as 'CSYM025', and the Tutor Name as 'Suraj Ajit'. Under 'Student Information', it lists the Student Name as 'Parastoo Shojaei Sarcheshmeh' and the Student ID as '25828319'.

## Login Failure Message

This shows the error message when incorrect login details are entered.



## Add Customer

This shows a new customer added to the table and the confirmation message displayed by the system.

The screenshot shows a Windows application window titled "Energy Billing System". On the left is a "Menu" sidebar with buttons for Home, Customers, Billing, Bills, and Exit. The main area is titled "Customers" and contains a sub-section "Add Customer". In the "Add Customer" section, there are three input fields: the first is "B", the second is "12345678", and the third is "UON", which is highlighted with a blue border. Below these fields are three buttons: "Add Customer", "Delete Selected", and "Save Customers". To the right of the "Add Customer" section is a table with four columns: ID, Name, Phone, and Address. The table has two rows of data:

ID	Name	Phone	Address
C001	Parastoo Shojaei	+447760537695	Simon Senlis Hall, Flat 15, Room5
C002	A	+448899556611	University of Northampton

Energy Billing System

**Menu**

- Home
- Customers
- Billing
- Bills
- Exit

**Customers**

**Add Customer**

Name	Parastoo Shojaei	Phone	+447760537695
Phone	A	+448899556611	University of Northampton
Address	B	12345678	UON
<b>Add Customer</b>			
Delete Selected			
Save Customers			

Customer added: C003

## Missing Fields Validation

This shows the warning message when add a customer without filling all required fields.

Energy Billing System

**Menu**

- Home
- Customers
- Billing
- Bills
- Exit

**Customers**

**Add Customer**

B	Parastoo Shojaei	+447760537695	Simon Senlis Hall, Flat 15, Room5
Phone	A	+448899556611	University of Northampton
UON			
<b>Add Customer</b>			
Delete Selected			
Save Customers			

Please fill all fields.

## Delete Customer

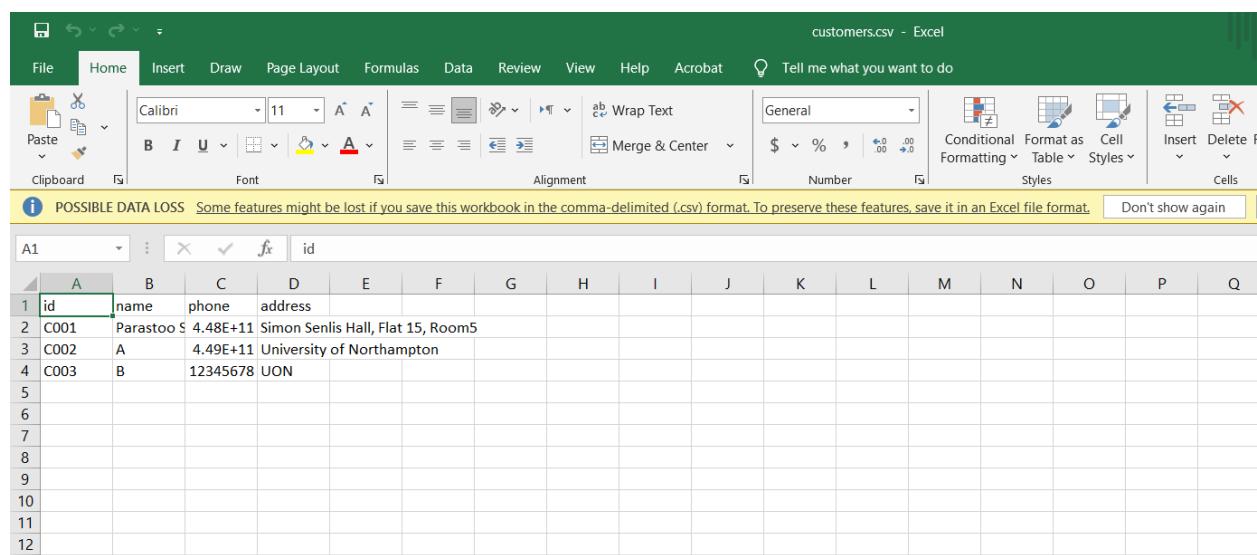
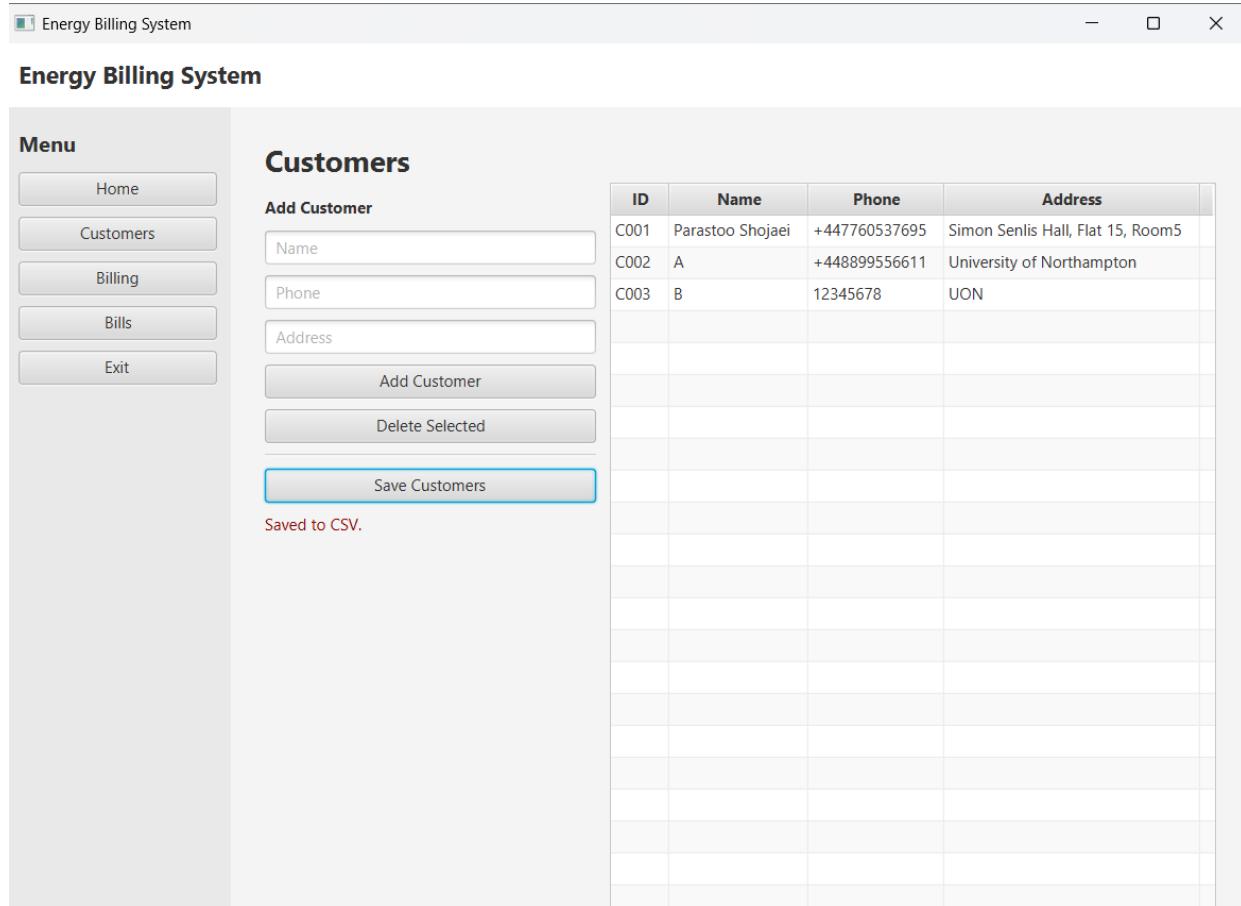
This shows the customer table after a selected customer has been deleted.

The screenshot shows the 'Energy Billing System' application window. On the left, a vertical menu bar titled 'Menu' contains five items: 'Home', 'Customers', 'Billing', 'Bills', and 'Exit'. The main area is titled 'Customers' and contains a sub-section 'Add Customer' with three input fields: 'Name' (Parastoo Shojaei), 'Phone' (+447760537695), and 'Address' (Simon Senlis Hall, Flat 15, Room5). Below these are two buttons: 'Add Customer' and 'Delete Selected', with 'Delete Selected' being highlighted with a blue border. A third button, 'Save Customers', is also present. At the bottom of the main area, a message 'Customer deleted.' is displayed in red text. To the right of the 'Add Customer' section is a large, empty table with four columns: 'ID', 'Name', 'Phone', and 'Address'. The first two rows of the table are visible, showing data for 'C001' and 'C002'. The row for 'C002' is highlighted with a grey background.

ID	Name	Phone	Address
C001	Parastoo Shojaei	+447760537695	Simon Senlis Hall, Flat 15, Room5
C002	A	+448899556611	University of Northampton

## Save Customers to CSV

This shows the confirmation message after customer data is saved to the CSV file.



## Bill Calculation Result

This shows the electricity bill, gas bill, and total amount after calculation.

The screenshot shows the 'Energy Billing System' application window. On the left is a vertical menu bar with options: Home, Customers, Billing (which is selected and highlighted in blue), Bills, and Exit. The main area is titled 'Billing' and contains two sections: 'Electricity Inputs' and 'Gas Inputs'. Under 'Electricity Inputs', there are six input fields: 'Opening Day' (40470.637), 'Closing Day' (40516.687), 'Opening Night' (37386.998), 'Closing Night' (37623.210), 'Standing (p/day)' (22.63), and 'Unit Price (p)' (19.349). Under 'Gas Inputs', there are four input fields: 'Opening' (10091.5), 'Closing' (10127.6), 'Standing (p/day)' (24.87), and 'Unit Price (p)' (3.797). At the bottom, there are three buttons: 'Calculate' (highlighted in blue), 'Save Bill', and a status message 'Calculated'. Below these buttons, the calculated bills are displayed: Electricity Bill: £65.18, Gas Bill: £54.22, and Total: £119.40.

Energy Billing System

**Menu**

- Home
- Customers
- Billing
- Bills
- Exit

**Billing**

Customer ID: C003 Days: 33

**Electricity Inputs**

Opening Day:	40470.637
Closing Day:	40516.687
Opening Night:	37386.998
Closing Night:	37623.210
Standing (p/day):	22.63
Unit Price (p):	19.349

**Gas Inputs**

Opening:	10091.5
Closing:	10127.6
Standing (p/day):	24.87
Unit Price (p):	3.797

**Calculated**

Electricity Bill: £65.18  
Gas Bill: £54.22  
Total: £119.40

## Invalid Input Handling

This shows the error message when invalid (non-numeric) input is entered in billing fields.

The screenshot shows a Windows application window titled "Energy Billing System". The menu on the left includes "Home", "Customers", "Billing" (which is selected), "Bills", and "Exit". The main area is titled "Billing" and contains two sections: "Electricity Inputs" and "Gas Inputs". In the "Electricity Inputs" section, the "Opening Day:" field contains the non-numeric value "12" (which is valid). The "Closing Day:" field contains the non-numeric value "14" (which is valid). The "Opening Night:" field contains the non-numeric value "AB" (which is invalid). The "Closing Night:" field is empty. The "Standing (p/day):" field is empty. The "Unit Price (p):" field is empty. Below these fields, there are four empty input fields for "Gas Inputs": "Opening:", "Closing:", "Standing (p/day):", and "Unit Price (p):". At the bottom, there are three buttons: "Calculate" (highlighted in blue), "Save Bill", and "Invalid input" (which is displayed in red text). Below the buttons, the bill summary is shown: Electricity Bill: -, Gas Bill: -, Total: -.

Customer ID: C003 Days: 34

**Electricity Inputs**

Opening Day: 12  
Closing Day: 14  
Opening Night: AB  
Closing Night:  
Standing (p/day):  
Unit Price (p):

---

**Gas Inputs**

Opening:  
Closing:  
Standing (p/day):  
Unit Price (p):

**Buttons:**

Calculate Save Bill Invalid input

**Bill Summary:**

Electricity Bill: -  
Gas Bill: -  
Total: -

## Bill Saved

This shows the confirmation message after a bill is saved successfully.

The screenshot shows the 'Energy Billing System' application window. On the left is a vertical menu bar with options: Home, Customers, Billing (which is selected and highlighted in blue), Bills, and Exit. The main area is titled 'Billing' and contains two sections: 'Electricity Inputs' and 'Gas Inputs'. Under 'Electricity Inputs', there are six input fields: 'Opening Day' (40470.637), 'Closing Day' (40516.687), 'Opening Night' (37386.998), 'Closing Night' (37623.210), 'Standing (p/day)' (22.63), and 'Unit Price (p)' (19.349). Under 'Gas Inputs', there are four input fields: 'Opening' (10091.5), 'Closing' (10127.6), 'Standing (p/day)' (24.87), and 'Unit Price (p)' (3.797). At the bottom of the main area, there are two buttons: 'Calculate' and 'Save Bill'. The 'Save Bill' button is highlighted with a blue border. To its right, the text 'Bill saved' is displayed. Below these buttons, the calculated bills are shown: 'Electricity Bill: £65.18', 'Gas Bill: £54.22', and 'Total: £119.40'.

Energy Billing System

**Menu**

- Home
- Customers
- Billing
- Bills
- Exit

**Billing**

Customer ID: C003 Days: 33

**Electricity Inputs**

Opening Day:	40470.637
Closing Day:	40516.687
Opening Night:	37386.998
Closing Night:	37623.210
Standing (p/day):	22.63
Unit Price (p):	19.349

---

**Gas Inputs**

Opening:	10091.5
Closing:	10127.6
Standing (p/day):	24.87
Unit Price (p):	3.797

---

**Bill saved**

Electricity Bill: £65.18  
Gas Bill: £54.22  
Total: £119.40

## Bills Page Loaded

This shows the table loaded from the CSV file and the number of records displayed.

## Bill Marked as Paid

This screenshot shows a bill after its status has been changed to “Paid”.

Customer ID	Days	Electricity (£)	Gas (£)	Total (£)	Status
C001	33	65.18	54.22	119.4	Unpaid
C002	38	9.47	11.78	21.25	Unpaid
C003	33	65.18	54.22	119.4	Paid

Customer ID	Days	Electricity (£)	Gas (£)	Total (£)	Status
C001	33	65.18	54.22	119.4	Unpaid
C002	38	9.47	11.78	21.25	Unpaid
C003	33	65.18	54.22	119.4	Paid

## Bill Deleted

This shows the table after a selected bill has been deleted from the system.

The screenshot shows a Windows application window titled "Energy Billing System". On the left is a vertical menu bar with the following options: Home, Customers, Billing, Bills (which is currently selected), and Exit. The main content area is titled "Bills" and contains a table with the following data:

Customer ID	Days	Electricity (£)	Gas (£)	Total (£)	Status
C001	33	65.18	54.22	119.4	Unpaid
C002	38	9.47	11.78	21.25	Unpaid

At the bottom of the main area, there are three buttons: "Refresh", "Mark as Paid", and "Delete Selected". The "Delete Selected" button is highlighted with a blue border. To its right, the text "Deleted." is displayed in a small font.

## 6.2 Whitebox Testing

In this project, the most important logic is the bill calculation code, so I created JUnit tests for ElectricityBill and GasBill.

The main reason for using JUnit was to confirm that the calculation results match expected values using fixed inputs, and to make sure rounding and VAT are applied correctly.

I created a separate package called application.tests to keep unit tests separate from the main application code.

For each test, I create a bill object with known input values.

Then I call calculateBill() and compare the result to the expected value in pounds

If the expected value matches, the test passes. If not, the test fails.

### ElectricityBillTest.java:

```
package application.tests;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import application.ElectricityBill;
public class ElectricityBillTest {
    @Test
    void shouldMatchSampleStatement_electricity() {
        int days = 33;
        ElectricityBill e = new ElectricityBill(
            40470.637, 40516.687,
            37386.998, 37623.210,
            22.63, // standing charge (p/day)
            19.349, // unit price (p)
            days
        );
        double pence = e.calculateBill();
        double pounds = pence / 100.0;
        assertEquals(65.18, pounds, 0.0001);
    }
}
```

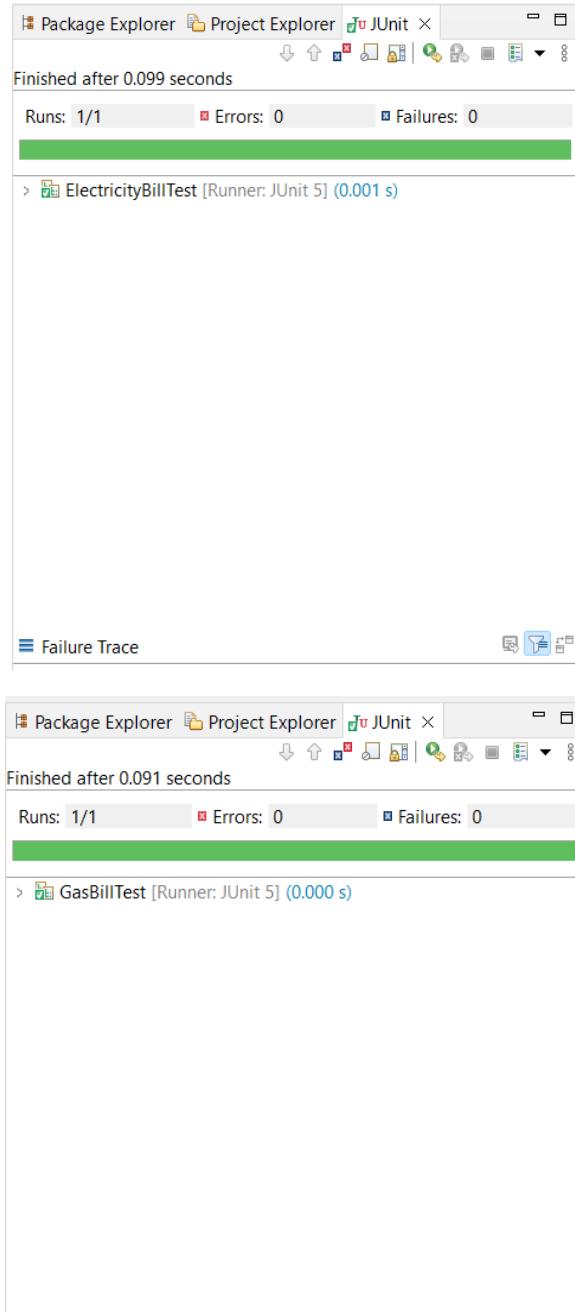
### GasBillTest.java:

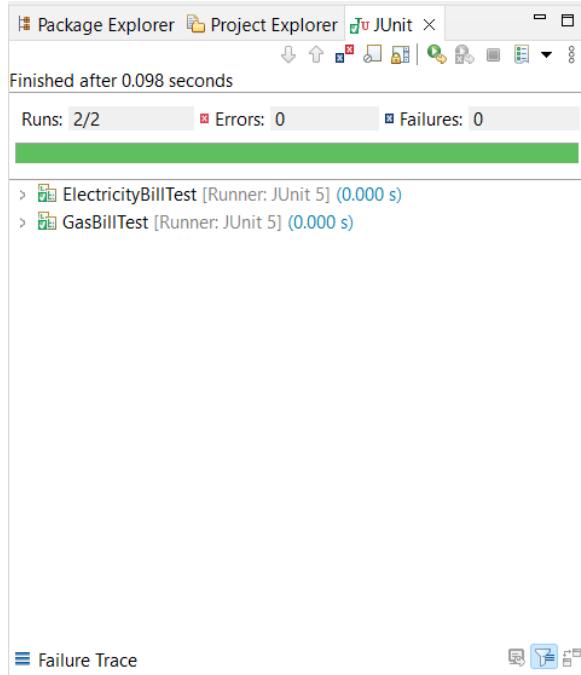
```
package application.tests;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import application.GasBill;
public class GasBillTest {

    @Test
    void shouldMatchSampleStatement_gas() {
        int days = 33;
        GasBill g = new GasBill(
            10091.5, 10127.6,
            24.87,
            3.797,
            days
        );
    }
}
```

```
) ;  
        double pence = g.calculateBill();  
        double pounds = pence / 100.0;  
        assertEquals(54.22, pounds, 0.0001);  
    }  
}
```

I also included screenshots showing the unit tests running successfully.





### 6.3 Bugs, Weaknesses

The main system functions work correctly, but the current version has some limitations. These are not critical errors, but they are areas that could be improved in a future version.

- **Login is fixed (no user management):** The system uses one fixed username and password (admin/1234). The user cannot create new accounts, change passwords, or manage different roles (admin/user).
- **No “Edit/Update” feature for customers:** Customers can be added and deleted, but there is no feature to edit an existing customer’s name, phone, or address in the GUI.
- **No “Edit/Update” feature for bills:** Bills can be saved, deleted, and marked as paid, but bills cannot be edited after they are created.
- **Customer deletion does not check related bills:** A customer can be deleted even if bills exist for that customer. In a real system, the system should warn the user or prevent deletion.
- **No search or filter:** The system does not include searching customers or filtering bills (for example by status Paid/Unpaid).
- **No confirmation dialogs:** Delete actions remove rows immediately. A confirmation popup could be added to avoid accidental deletion.

## 7 References

I developed this project using the concepts and examples taught during lectures.

I used some example code provided by the tutor as a reference.

I also used the official Java and JavaFX documentation when I needed help with layouts, TableView, event handling, and application structure. These resources were mainly used for learning syntax.

The following resources were used during the development of this project:

- Oracle (2024). Java Platform, Standard Edition Documentation.  
<https://docs.oracle.com/javase/>
- Oracle (2024). JavaFX Documentation.  
<https://openjfx.io/javadoc/>
- JUnit Team (2024). JUnit 5 User Guide.  
<https://junit.org/junit5/docs/current/user-guide/>
- University of Northampton. CSYM025 – Visual Object Software