

**VISVESVARAYA TECHNOLOGICAL
UNIVERSITY, BELAGAVI 590018**



Project Report on
“ELECTRICITY BILLING SYSTEM”
By

Mouneshwar (1BM24CS175)
Nagaraj G M (1BM24CS176)
Nagarajun (1BM24CS178)
Mahesh Gopal Lamani (1BM24CS157)

Under the Guidance of
Monisha H M
Assistant Professor, Department of CSE
BMS College of Engineering

Work carried out at



Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING



CERTIFICATE

This is to certify that the OOPS with JAVA project titled “Your title” has been carried out by **Mouneshwar (1BM24CS175)**, **Nagaraj G M (1BM24CS176)**, **Nagarajun (1BM24CS178)**, **Mahesh Gopal Lamani (1BM24CS157)** during the academic year 2025-2026.

Signature of the guide

Monisha H M

Assistant Professor,

Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DECLARATION

We, **Mouneshwar (1BM24CS175), Nagaraj G M (1BM24CS176), Nagarajun (1BM24CS178), Mahesh Gopal Lamani (1BM24CS157)** students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this project work entitled "**ELECTRICITY BILLING SYSTEM**" has been carried out by us under the guidance of **Monisha H M**, Assistant Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Sep-Dec 2025. We also declare that to the best of our knowledge and belief, the project reported here is not from part of any other report by any other students.

Signature of the Candidates

Mouneshwar (1BM24CS175)

Nagaraj G M (1BM24CS176)

Nagarajun (1BM24CS178)

Mahesh Gopal Lamani (1BM24CS157)

TABLE OF CONTENTS

CHAPTER NO.	CONTENT	PAGE NO.
1.	Introduction	5
2.	Problem Statement	6
3.	Tools Used	7
4.	Overview of the project	9
5.	Implementation	14
6.	Testing	20
7.	OOP Concepts Used in Electricity Billing System	27
8.	Snapshots	30
9.	Future scope and Limitations / References	33

INTRODUCTION

Electricity Billing System is a software-based application.

- i. This project aims at serving the department of electricity by computerizing the billing system.
- ii. It mainly focuses on the calculation of units consumed during the specified time and the money to be charged by the electricity offices.
- iii. This computerized system will make the overall billing system easy, accessible, comfortable, and effective for consumers.

To design the billing system more service oriented and simple, the following features have been implemented in the project. The application has high speed of performance with accuracy and efficiency.

The software provides facility of data sharing, it does not require any staff as in the conventional system. Once it is installed on the system only the meter readings are to be given by the admin where customer can view all details, it has the provision of security restriction.

The electricity billing software calculates the units consumed by the customer and makes bills, it requires small storage for installation and functioning. There is provision for debugging if any problem is encountered in the system.

The system excludes the need of maintaining paper electricity bill, administrator does not have to keep a manual track of the users, users can pay the amount without visiting the office. Thus, it saves human efforts and resources.

Preamble

We, the owners of our project, respect all customers and make them happy with our service.

The main aim of our project is to satisfy customer by saving their time by payment process, maintaining records, and allowing the customer to view his/her records and permitting them to update their details.

The firm handles all the work manually, which is very tedious and mismatched.

The objectives of our project are as follows:

- **To keep the information of consuming unit energy of current month.**
- **To keep the information of Customer.**
- **To keep the information of consuming unit energy of previous month.**
- **To calculate the units consumed every month regularly.**
- **To generate the bills adding penalty and rent. □ To save the time by implementing payment process online.**

Problem Statement

The manual system is suffering from a series of drawbacks. Since whole of the bills is to be maintained with hands the process of keeping and maintaining the information is very tedious and lengthy to customer. It is very time consuming and laborious process because, staff need to be visited the customers place every month to give the bills and to receive the payments. For this reason, we have provided features Present system is partially automated (computerized), existing system is quite laborious as one must enter same information at different places.

Tools Used

Existing and Proposed System

The conventional system of electricity billing is not so effective; one staff must visit each customer's house to note the meter readings and collect the data. Then, another staff must compute the consumed units and calculate the money to be paid. Again, the bills prepared are to be delivered to customers. Finally, individual customer must go to electricity office to pay their dues.

Hence, the conventional electricity billing system is uneconomical, requires many staffs to do simple jobs and is a lengthy process overall. In order to solve this lengthy process of billing, a web based computerized system is essential. This proposed electricity billing system project overcomes all these drawbacks with the features. It is beneficial to both consumers and the company which provides electricity.

With the new system, there is reduction in the number of staffs to be employed by the company. The working speed and performance of the software is faster with high performance which saves time. Furthermore, there is very little chance of miscalculation and being corrupted by the staffs.

Software & Hardware Requirements

Hardware Requirements:

- Hardware Specification: -Processor Intel Pentium V or higher
- Clock Speed: -1.7 GHz or more
- System Bus: -64 bits
- RAM: -16GB
- HDD: -2TB
- Monitor: -LCD Monitor
- Keyboard: -Standard keyboard
- Mouse: -Compatible mouse

Software Requirements:

- Operating System: -Windows 10
- Software: -Microsoft SQL Server
- Front End: -Java core/swings (NetBeans)
- Back End: -My SQL

Overview of the project

Preliminary Design

System design is an abstract representation of a system component and their relationship and which describe the aggregated functionality and performance of the system. It is also the plan or blueprint for how to obtain answer to the question being asked. The design specifies various type of approach.

Database design is one of the most important factors to keep in mind if you are concerned with application performance management. By designing your database to be efficient in each call it makes and to effectively create rows of data in the database, you can reduce the amount of CPU needed by the server to complete your request, thereby ensuring a faster application

Schema Diagram

Database schema is described as database connections and constraints. It contains attributes. Every database has a state instances represent current set of databases with values. There are different types of keys in a database schema.

A primary key is a table column that can be used to uniquely identify every row of the table. Any column that has this property, these columns are called candidate key. A composite primary key is a primary key consisting of more than one column. A foreign is a column or combination of columns that contains values that are found in the primary key of some table.

All the attributes of each table are interconnected by foreign key which is primary key in another column and composite key. Primary key cannot be null. The fact that many foreign key values repeat simply reflects the fact that its one- to-many relationship. In one-to-many relationship, the primary key has the one value and foreign key has many values.

Figure 3.1.2 is a Schema diagram of Electricity Billing System which has six tables i.e., login, customer, tax, rent, bill, and meter_info where each table contain attributes some with primary key, foreign key. In the login table there are

6 attributes "meter_no", "username", "password", "user", "question", "answer". The customer table has 7 attributes "name", "meter_no"(primary key), "address",

"city", "state", "email", "phone". The rent table has 3 attributes "cost_per_unit"(primary key), "meter_rent", "service_charge". The tax table has 3 attributes "service_tax", "swacch_bharat_cess", "gst". The bill table has 5 attributes "meter_no"(foreign key that references the primary key of the customer table meter_no), "month", "units", "total_bill", "status". The meter_info table has 6 attributes "meter_no"(foreign key that references the primary key of the customer table meter_no), "meter_location", "meter_type", "phase_code", "bill_type", "days".

Schema Diagram

Login

Meter No	Username	Password	User	Question	Answer
----------	----------	----------	------	----------	--------

Customer

Name	Meter No	Address	City	State	Email	Phone
------	----------	---------	------	-------	-------	-------



Rent

Cost Per Unit	Meter Rent	Service Rent
---------------	------------	--------------

Tax

Service Tax	Swacch bharat cess	GST
-------------	--------------------	-----

Bill

Meter No	Month	Units	Total Bill	Status
----------	-------	-------	------------	--------

Meter Info

Meter No	Meter Location	Meter Type	Phase Code	Bill Type	Days
----------	----------------	------------	------------	-----------	------



Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

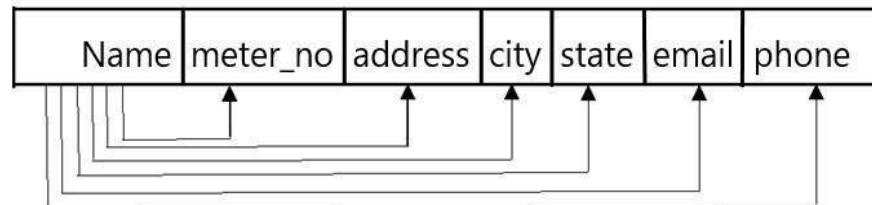
Let's discuss about anomalies first then we will discuss normal forms with examples. Anomalies in DBMS There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly.

First normal form(1NF)

As per the rule of first normal form,

- All rows must be unique (no duplicate rows).
- Each Cell must only contain a single value (not a list).
- Each value should be non-divisible (can't be split down further).

Customer



Second normal form(2NF)

As per the rule of second normal form,

- ✓ Database must be in First Normal Form.
- ✓ Non partial dependency-All non-prime attributes should be fully functionally dependent on the candidate key.

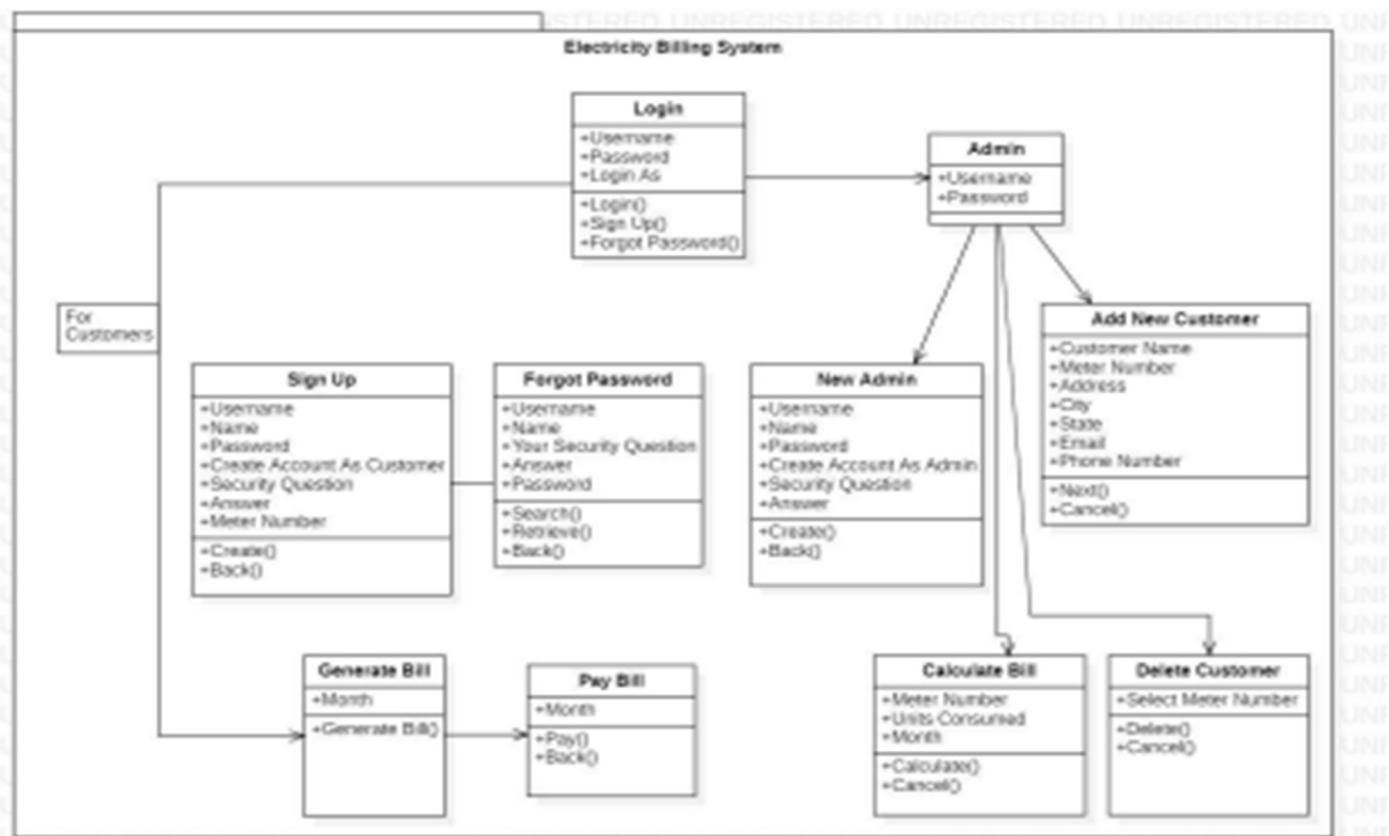
Third normal form(3NF)

As per the rule of third normal form,

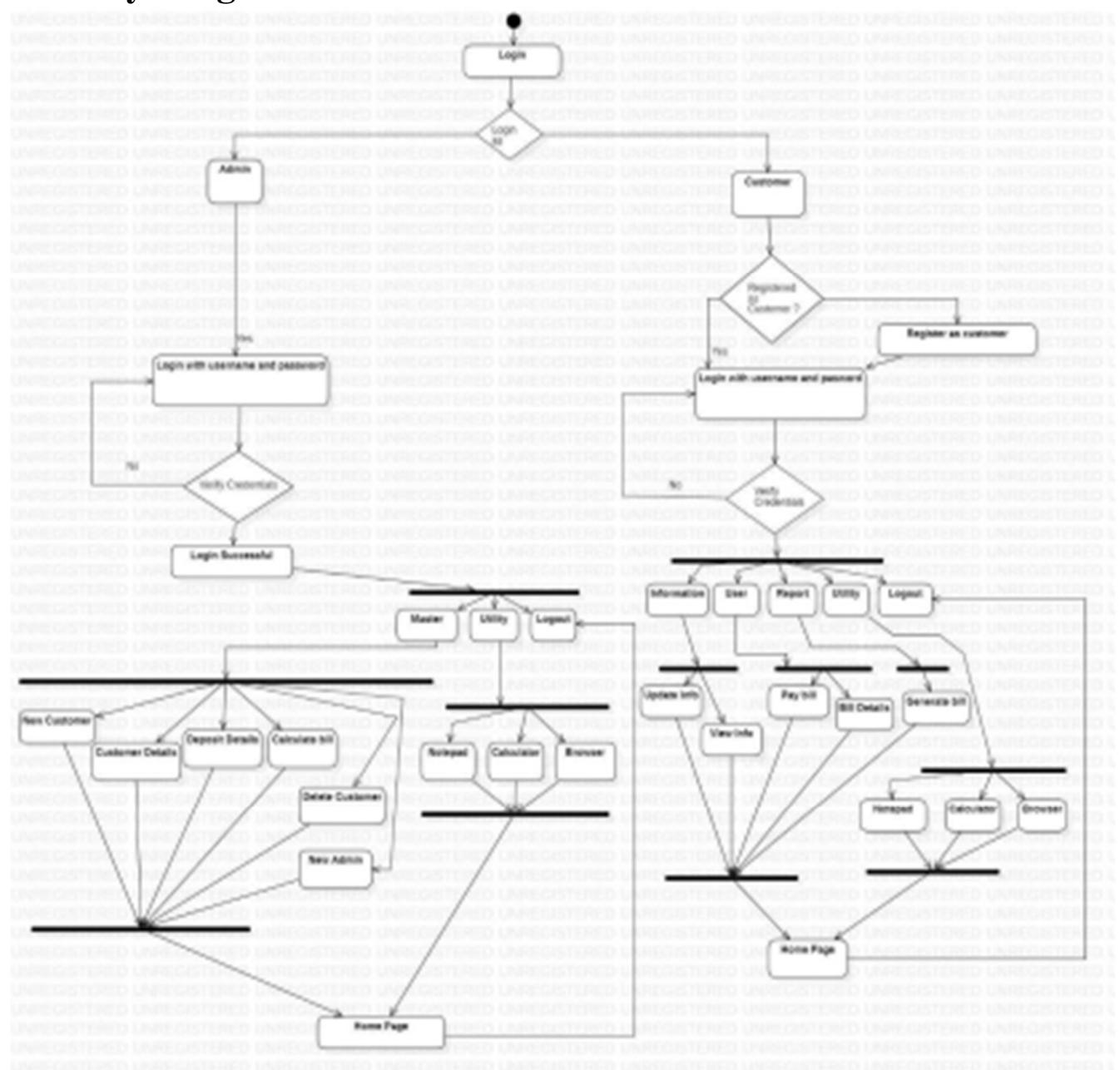
- Database must be in First and Second Normal Form.
- Non transitive dependency-All fields must only be determinable by the primary/composite key, not by other keys.

UML Diagram

Class Diagram:



Activity Diagram:



Implementation

Implementation of operations

- ❖ **Adding Customer:** Here admin can add new customer to the customer list who started using electricity bill system.
- ❖ **Searching Deposit Details:** Here admin can search according to meter number and month to view deposit details.
- ❖ **Viewing Details:** Here admin and user can view customer details and about details.
- ❖ **Adding Tax:** Here admin can add tax details.
- ❖ **Updating Customer:** Here customer can update his/her details by using meter_no of the customer.
- ❖ **Delete Customer:** Here admin can delete details based on meter number.

Implementation of SQL statements

Insert statement:

- The INSERT INTO statement is used to insert new records in a table.
- The INSERT INTO syntax would be as follows: INSERT INTO table_name VALUES (value1, value2, value3, ...).
- The following SQL statement insert's a new record in the “customer” table: Insert into customer VALUES (“sai”, “12345”, “ btm”, “ Bangalore”, “Karnataka”, “sai@gmail.com”, “9876543333”).

Update statement:

- An SQL UPDATE statement changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a condition.
- The UPDATE syntax would be as follows: UPDATE table_name SET column_name =value, column_name=value... [WHERE condition].
- The following SQL statement update's a new record in the “customer” table: UPDATE TABLE customer SET email= su@gmail.com WHERE meter_no =”12345”.

Delete statement:

- The DELETE statement is used to delete existing records in a table.
- The DELETE syntax would be as follows: DELETE FROM table_name WHERE condition.
 - The following SQL statement delete's a record in the “customer” table:
delete from customer where meter_no=12345.

Create statement:

- The CREATE TABLE Statement is used to create tables to store data. Integrity Constraints like primary key, unique key, foreign key can be defined for the columns while creating the table.
- The syntax would be as follows: CREATETABLE table_name (column1datatype, column2datatype, column3 datatype, column datatype, PRIMARY KEY (one or more columns)).
 - The following SQL statement creates a table “customer” table:
create table customer (name varchar (30), meter_no varchar (20) primary key, address varchar (50), city varchar (20), state varchar (30), email varchar (30), phone varchar (30));
 - The following SQL statement creates a table “login” table: create table login (meter_no varchar (30), username varchar (30), password varchar (30), user varchar (30), question varchar (40), answer varchar (30));
 - The following SQL statement creates a table “tax” table: create table tax (cost_per_unit int (20) primary key, meter_rent int (20), service_charge int (20), service_tax int (20), swacch_bharat_cess int (20), gst int (20));
 - The following SQL statement creates a table “bill” table: create table bill (meter_no varchar (20), foreign key(meter_no) references customer(meter_no) on delete cascade, month varchar (20), units int (20), total_bill int (20), status varchar (40));
 - The following SQL statement creates a table “meter_info” table: create table meter_info (meter_no varchar (30), foreign key(meter_no) references customer(meter_no) on delete cascade, meter_location varchar (10), meter_type varchar (15), phase_code int (5), bill_type varchar (10), days int (5));

Algorithm or pseudocode of implementation

Explanation of Algorithm or pseudocode of system:

- ✓ Start system
- ✓ Enter login name and password
- ✓ On clicking the login button
- ✓ Connect to database
- ✓ Query database to know whether user credentials are correct
- ✓ If not, deny access and return login page with an error message
- ✓ If correct, check if credentials for administrator
- ✓ If yes, allow login
- ✓ Set admin session, re-direct administrator to admin login page
- ✓ If no, allow login set user session
- ✓ Re-direct user to user home page

Algorithm or pseudocode of admin:

Login: This program will allow the admin to enter the username and password.

- If the entered credentials are correct, then the login will be successful otherwise need to be signup.
- If admin forgets password, it can be retrieved by giving username and answer for security question.
- After successful login the admin will be redirected to admin portal page where he/she can do following activities.

New Customer:

- This program will allow the admin to enter the customer details and automatically generates unique meter number. □
- If customer name, address, city, state, email and phone number is entered, insert the values into customer □

else print

error

while

next=true

```
enter the meter_info details else  
print  
meter_info error
```

- Submit the details of customer that has been entered by clicking onto next button.
- If we need to cancel the particulars that has been entered click onto cancel option.
- If we need to submit the particulars that has been entered click onto submit option.

Customer Details:

- This program will allow the admin to view customer details.
- If we need to print the particulars that has been viewed click onto print option.

Deposit Details:

- This program will allow the admin to view bill details. If we need to sort the particulars based on meter_no and month.
- If we need to search the particulars that has been viewed click onto search option.
- If we need to print the particulars that has been viewed click onto print option.

Tax Details:

- This program will allow the admin to add tax details. insert the values into tax
- else print error
- Submit the details of tax that has been entered by clicking onto submit button.
- If we need to cancel the particulars that has been entered click onto cancel option.

CalculateBill:

- This program will allow the admin to calculate total_bill when units consumed are inserted where meter_no and month is selected.
- Insert the values into bill else print error

- Submit the details of tax that has been entered by clicking onto submit button.
- If we need to cancel the particulars that has been entered click onto cancel option.

Delete Customer:

- This Program will allow the admin to delete the customer info when meter_no is selected.
- If we need to delete the particulars that has been saved click onto delete option.
- If we need to cancel the particulars that has been entered click onto back option.

Algorithm or pseudocode of Customer:

Login:

- This program will allow the customer to enter the username and password. If the entered credentials are correct, then the login will be successful otherwise need to be signup with the meter_no which is given by admin.
- If customer forgets password, it can be retrieved by giving username and answer for security question. After successful login the customer will be redirected to customer portal page where he/she can do following activities.

UpdateInfo1:

- This program will allow the customer to update the customer details. If customer address, city, state, email and phone number is updated.
- update the values into customer else print error
- update the details of customer that has been updated by clicking onto update button.
- If we need to cancel the particulars that has been updated, click onto back option.

View Info:

- This program will allow the customer to view his/her own details.
- If we need to go back from the particulars that has been viewed click onto back option.

Pay Bill:

- This program will allow the customer to view bill details and redirects to pay.
- the bill where status will be updated.
- If we need to cancel the particulars that has been viewed click onto back option.
- If we need to pay the bill amount that has been viewed click onto pay option.

Bill Details:

- This program will allow the customer to view bill details.□
- If we need to print the particulars that has been viewed click onto print option.□

Generate Bill:

- This program will allow the customer to generate bill when meter_no and month is selected.□
- Generate the details by clicking on generatebill button.□

NOTE: Utility (notepad, browser, calculator), query and logout is given to both customer and admin portals.

Testing

This chapter gives the outline of all the testing methods that are carried out to get a bug free application.

Testing process

Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, compiles with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested. In some cases, test cases are done based on the system requirements specified for the product/software, which is to be developed.

Testing objectives

The main objectives of testing process are as follows:

- Testing is a process of executing a program with the intent of finding an error. • A good test case is one that has high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

Levels of Testing

Different levels of testing are used in the testing process; each level of testing aims to test different aspects of the system. The basic levels are unit testing, integration testing, system testing and acceptance testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design the module. The software built, is a collection of individual modules. In this kind of testing exact flow of control for each module was verified. With detailed design consideration used as a guide, important control paths are tested to uncover errors within the boundary of the module.

Table 5.1: Negative test case for phone number insertion

Function Name	Input	Expected Output	Error	Resolved
Input phone number	98977	Phone number is invalid	Length of phone number is not equal to 10	Consume 0
Input phone number	98977agv	Phone number is invalid	Alphabets are being taken as input for phone number	—

Table 5.2: Positive test case for phone number insertion

Function Name	Input	Expected Output	Error	Resolved
Input Phone Number	989777 8988	Expected Output is Seen	—	—

Table 5.3: Negative test case for email insertion

Function Name	Input	Expected Output	Error	Resolved
Input email	Sai1.i n	Email is invalid	Email is not in a format given	Consume ()

Table 5.4: Positive test case for email insertion

Function Name	Input	Expected Output	Error	Resolved
Input email	aki123@gmail.com	Expected output is seen	—	—

Table 5.5: Negative test case for customer name insertion

Function Name	Input	Expected Output	Error	Resolved
Input customer name	Sana123	Name is invalid	Numbers are being taken as input for name	Consume ()

Table 5.6: Positive test case for customer name insertion

Function Name	Input	Expected Output	Error	Resolved
Input customer name	Gowthu	Expected output is seen	—	—

Integration Testing

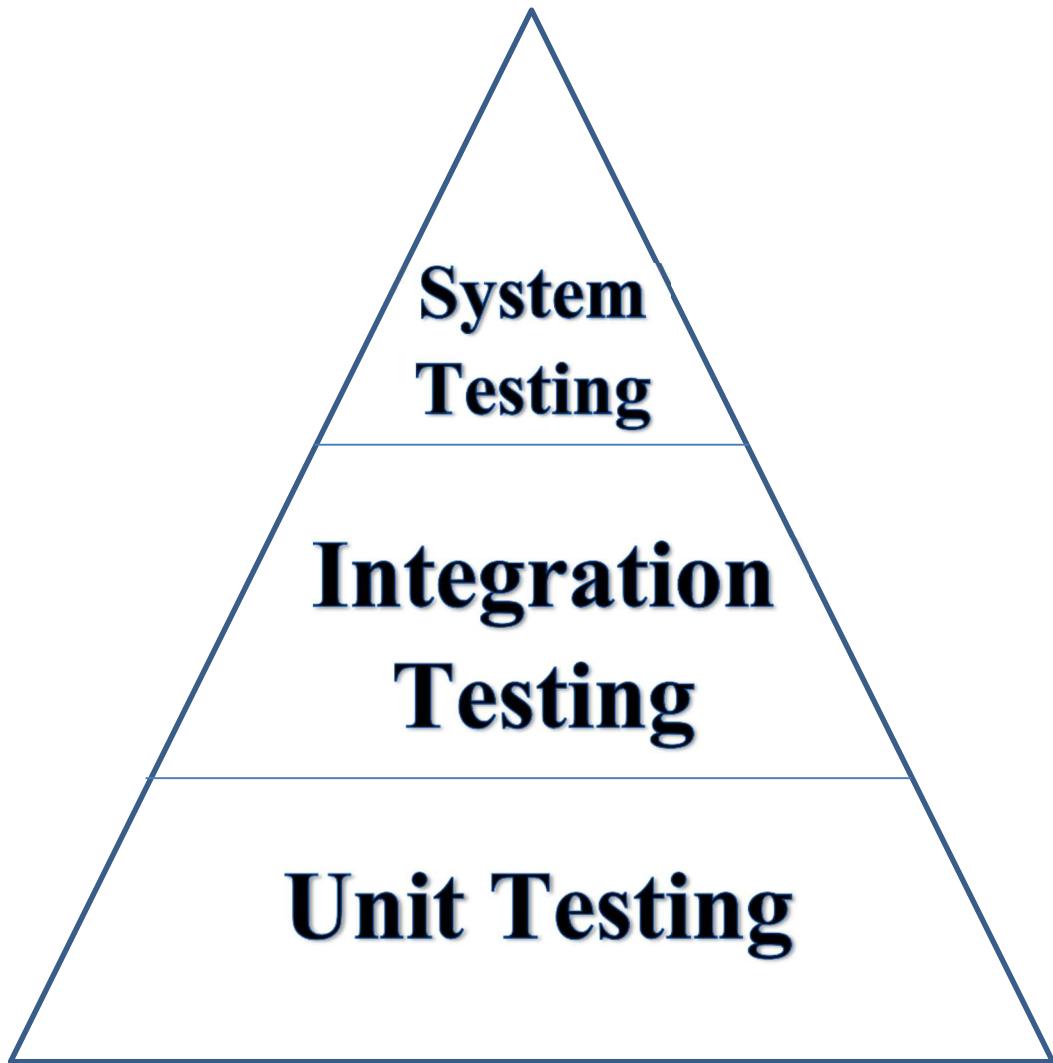
The second level of testing is called integration testing. In this, many class-tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly. We have been identified and debugged.

Table 5.7: Test case on basis of generation of bill

Function Name	Input	Expected Output	Error	Resolved
Negative searching of total_bill	12334(meter_no) January(month)	Details seen but not total_bill	Output not seen	Consume()
Positive searching of total_bill	12334(meter_no) January(month)	Must display full generated bill with total_bill	—	—

Table 5.8: Test case on basis of deposit details

Function Name	Input	Expected Output	Error	Resolved
Negative searching of depositedetails	12334(meter_no) January(month)	Details not seen	Output not seen	Consume()
Positive searching of total_bill	12334(meter_no) January(month)	Must display depositedetails	—	—



Testing Diagram

System testing

Here the entire application is tested. The reference document for this process is the requirement document, and the goal is to see IF the application meets its requirements. Each module and component of ethereal was thoroughly tested to remove bugs through a system testing strategy. Test cases were generated for all possible input sequences and the output was verified for its correctness.

Table 5.9: Test cases for the project

Steps	Action	Expected output
Step1 choice	The screen appears when the users run the program. 1. If admin login 2. If customer login	A page with different menu's appears. 1.Admin panel opens and 2.Customer panel opens
Step 2	The screen appears when the admin logs in and selects any one of the menus from the click of the mouse.	A window for adding new customer, inserting tax, calculate bill, view deposit details etc.
Selection 1	❖ New Customer ❖ Customer Details ❖ Deposit Details ❖ Calculate Bill ❖ Tax Details ❖ Delete Customer ❖ New Admin	

Step 2.1	The screen appears when the customer login and selects any one of the menus from the click of the mouse	A window for generating bill, update customer details, view details, generating bill
Selection 2	<ul style="list-style-type: none"> ❖ Update Details ❖ View Details 	
Selection 2a	<ul style="list-style-type: none"> ❖ Generate Bill 	
Selection 2b	<ul style="list-style-type: none"> ❖ Pay Bill ❖ Bill Details 	

OOP Concepts Used in Electricity Billing System

The Electricity Billing System is developed using Java (Swing + JDBC) and follows Object Oriented Programming (OOP) principles. These concepts help in designing a structured, secure, reusable, and real-world-oriented software system. The major OOP concepts used in this project are explained below.

1. Class

Definition:

A class is a blueprint for creating objects. It defines properties (variables) and behaviors (methods).

Usage in this Project:

- Customer class – stores customer details such as name, meter number, address, phone, and email
- Admin class – handles administrative operations
- Bill class – contains bill details like units consumed, month, total amount, and payment status
- MeterInfo class – stores meter-related information
- Login class – manages username, password, and user role

Each real-world entity in the system is represented using a class.

2. Object

Definition:

An object is an instance of a class.

Usage in this Project:

- Each customer registered in the system is an object of the Customer class
- Each generated bill is an object of the Bill class

Database records are represented as objects in the application.

3. Encapsulation

Definition:

Encapsulation is the process of wrapping data and methods together into a single unit (class) and restricting direct access to data.

Usage in this Project:

- Variables are declared as private
- Data is accessed and modified using public getter and setter methods

Example:

- `private String meterNo;`
- `getMeterNo()` and `setMeterNo()`

Encapsulation improves data security and prevents unauthorized access.

4. Abstraction

Definition:

Abstraction shows only essential features to the user and hides internal implementation details.

Usage in this Project:

- Users can view and pay bills without knowing how the bill is calculated
- The internal logic for unit calculation, tax, and total bill generation is hidden

This makes the system easy to use and understand.

5. Inheritance

Definition:

Inheritance allows a class to acquire the properties and methods of another class.

Usage in this Project:

- A common User class can act as a parent class
- Admin and Customer classes act as child classes

Common functionalities like login credentials are reused, reducing code duplication.

6. Polymorphism

Definition:

Polymorphism allows the same method to perform different actions based on the object or context.

Usage in this Project:

- login() method behaves differently for Admin and Customer
- calculateBill() method can generate different bills for residential and commercial users

This increases flexibility and scalability.

7. Association

Definition:

Association represents a relationship between two classes.

Usage in this Project:

- One Customer is associated with multiple Bill objects
- Customer is associated with MeterInfo

This represents real-world relationships such as one customer having many bills.

8. Reusability

Definition:

Reusability means using the same code multiple times without rewriting it.

Usage in this Project:

- Database connection classes
- Common validation methods for email and phone number.

SNAPSHOTS

SNAPSHOTS:



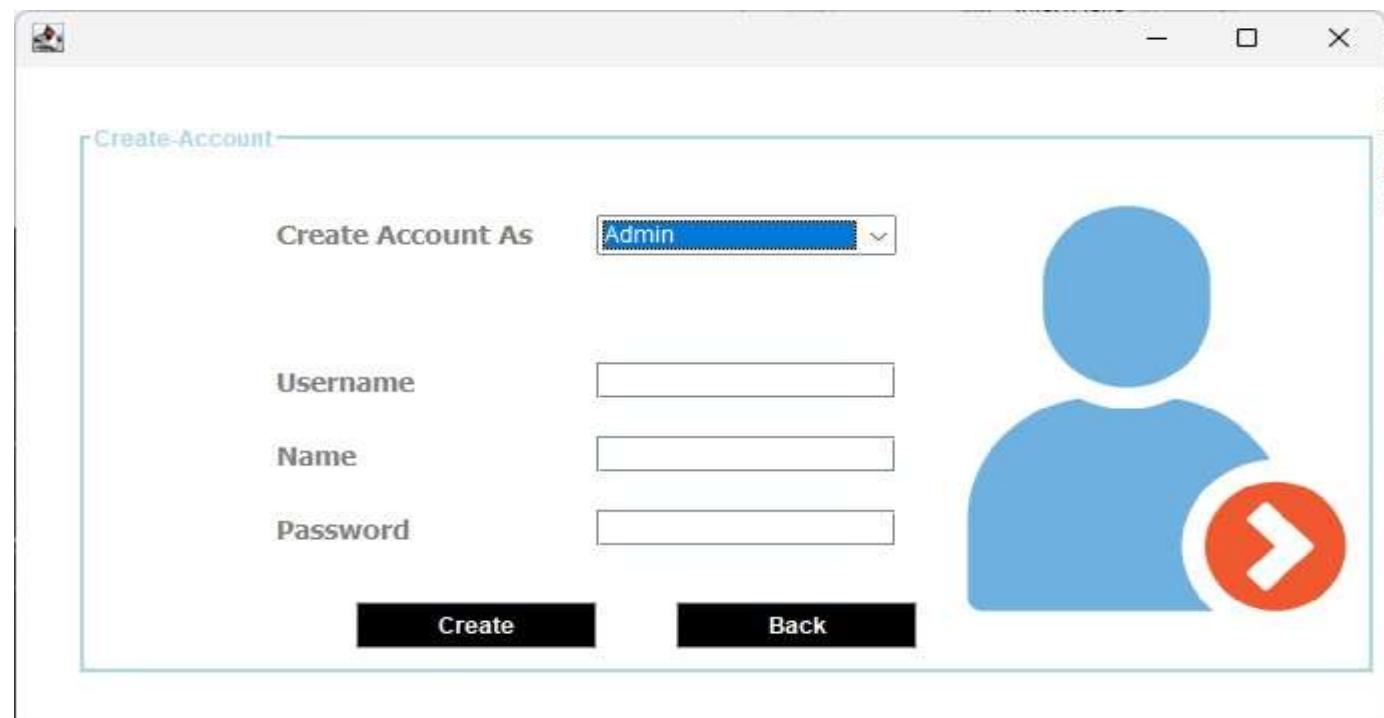
Splash page

Before, going to the login page. It will be generated for 10 seconds. After that login page generated.



Login Screen

Here Customer and Admin can login to their respective accounts. The dropdown menu allows to choose whether to login as an admin or as a customer.



Sign Up Screen

Here New customers will signup to access their accounts.
User have to enter username, name, password, choose security question and answer to that question.
Every user must enter their unique Meter Number to complete their signup process.



Customer's Home Screen

Customer lands on this page after successful login.

Future scope and Limitations

SOFTWARE SCOPE:

- **Extensibility:** This software is extendable in ways that its original developers may not expect. The following principles enhances extensibility like hide data structure, avoid traversing multiple links or methods avoid case statements on object type and distinguish public and private operations.
- **Reusability:** Reusability is possible as and when required in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability: Sharing of newly written code within a project and reuse of previously written code on new projects.
- **Understandability:** A method is understandable if someone other than the creator of the method can understand the code (as well as the creator after a time lapse). We use the method, which is small and coherent helps to accomplish this.
- **Cost-effectiveness:** Its cost is under the budget and made within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement.

Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

LIMITATIONS:

This application cannot be accessed remotely.

- **This application requires knowledgeable person to use this application.**
- **This application does not have journals.**

References

Book Reference

Database Management Systems 3rd Edition by Raghu Ramakrishnan (TEXTBOOK).

Websites

- <https://www.youtube.com/watch?v=iWitVuW2D1o&t=4s>
- <http://www.github.com>
- www.stackoverflow.com

