

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI - 590014



A DBMS Mini Project Report On

“ELECTRICITY BILL MANAGEMENT SYSTEM”

Submitted in the partial fulfilment of the requirements for the award of the

*Degree of **Bachelor of Engineering in Information Science and***

Engineering

Submitted by,

NITHIN R

Under the guidance of

Mrs. SANDHYA RANI V

(Asst Prof, Dept. of ISE)

Project Guide



Department of Information Science and Engineering The Oxford College
of Engineering Bommanahalli, Bangalore - 68, 2020-2021.

THE OXFORD COLLEGE OF ENGINEERING

Bommanahalli, Hosur Road, Bangalore – 560068

(Affiliated to Visvesvaraya Technological University, Belgaum)

Department of Information Science and Engineering



CERTIFICATE

Certified that the project work entitled “**ELECTRICITY BILL MANAGEMENT SYSTEM**” carried out by **Nithin R Bonafede** student of The Oxford College of Engineering, Bangalore in partial fulfilment for the award of the Degree of **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during the year 2021-2022. The Mini project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Mrs. Sandhya Rani V.

Dr. Kanagavalli R

Dr. N Kannan

Project Guide

H.O.D, Dept. of ISE.

Principal, TOCE

External Viva

Name of the Examiners

Signature with Date

1. _____

2. _____

ACKNOWLEDGEMENT

A project is a job of great enormity and it can't be accomplished by an individual all by them. Eventually, we are grateful to several individuals whose professional guidance, assistance, and encouragement have made it a pleasant endeavour to undertake this project.

It gives us great pleasure in expressing our deep sense of gratitude to our respected Founder Chairman **Shri S. Narasa Raju** and the respected Chairman **Shri S.N.V.L Narasimha Raju** for having provided us with great infrastructure and well-furnished labs.

We take this opportunity to express our profound gratitude to our respected Principal **Dr. N Kannan** for his support.

We are grateful to the Vice-Principal and Head of the Department **Dr. R Kanagavalli** for her unfailing encouragement and suggestion given to us in the course of our project work.

Guidance and deadlines play a very important role in the successful completion of the project on time. We convey our gratitude to **Mrs. Sandhya Rani V**, Project Guide for having constantly guided and monitored the development of the project.

A note of thanks to the Department of Information Science & Engineering, both teaching and non-teaching staff for their co-operation extended to us.

We thank our parents for their constant support and encouragement. Last, but not least, we would like to thank our peers and friends.

Nithin R

Contents

1 Introduction

- 1.1 Overview
- 1.2 The objective of the Project
- 1.3 Problem & its solution

2 Development Tools & Environment

- 2.1 Overview
- 2.2 Java
- 2.3 UX Design
- 2.4 JavaFX
- 2.5 AnimateFX, CSS
- 2.6 SQL
- 2.7 Database System
- 2.8 JDBC

3 System Design

3.1 Front-End Design

- 3.1.1 Overview
- 3.1.2 Pseudocode for some complex methods

3.2 Database Design

- 3.2.1 Entity-Relationship Diagram
- 3.2.2 Database tables

4 Implementation

- 4.1 SQL Operations
- 4.2 Java Code

5 Testing

5.1 Testing Process

5.2 Testing Objective

6 Conclusion

7 Application Snapshots

8 Miscellaneous

Introduction

1.1 Overview

Our project entitled “**Electricity Bill Management System**” aims is to generate an electricity bill with all the charges. A manual system that is employed is extremely laborious and quite inadequate. It only makes the process more difficult.

Our project aims to develop a system that is meant to partially computerize the work performed in the Electricity Board like generating monthly electricity bills, a record of consuming units of energy, store records of the customer, and previous unpaid records.

1.2 The objective of the project

Our project aims at Business process automation, i.e. we have tried to computerize various processes of Electricity Billing systems. In the sector of electricity board we have computerizes their department and stock maintenance.

Scope of any software depends upon the following things:

- It satisfies the user requirement
- Be easy to understand by the user and operator
- Be easy to operate
- Have a good user interface

- Be expandable
- Delivered on schedule within the budget.

We have tried to make such type of software, which satisfies the above-given requirement.

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

The objective of our project is as follows:

- To keep the information of Customer.
- To keep the information of consuming unit of energy of the current month.
- To keep the information of paid bills.
- To keep the information of credit/debit cards for easier payment.
- To maintain the transaction history of the customer.

1.3 Problem & its solution

The old manual system was suffering from a series of drawbacks. Since the whole of the system was to be maintained with hands the process of keeping, maintaining, and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. there used to be lots of difficulties in associating any particular transaction with a particular context. If any information was to be found it was required to go through the different registers, documents there would never exist anything like report generation. There would always be unnecessary consumption of time while entering

records and retrieving records. One more problem was that it was very difficult to find errors while entering the records. Once the records were entered it was very difficult to update these records. At present, work done on the electricity board is performed manually which is a great headache for the department. The reason behind it is that there is a lot of information to be maintained and has to be kept in mind while running the business. For this reason, we have provided features Present system is partially automated (computerized) the existing system is quite laborious as one has to enter the same information at three different places.

The following points should be well considered:

Documents and reports that must be provided by the new system: there can also be a few reports, which can help management in decision-making and cost controlling, but since these reports do not get required attention, such kind of reports and information were also identified and given the required attention. Details of the information needed for each document and report. The required frequency and distribution for each document. Probable sources of information for each document and report. With the implementation of a computerized system, the task of keeping records in an organized manner will be solved. The greatest of all is the retrieval of information, which will be at the click of the mouse. So the proposed system helps in saving time in different operations and making information flow easy giving valuable reports.

Development Tools & Environment

2.1 Overview

The system development environment shows the hardware and software, which was used for developing the software. The project was developed in:

OS – macOS Monterey

Java IDE – IntelliJ

Database Management – MySQL Workbench

Processor – Apple M1

Ram – 8 GB

Storage – 256 GB

2.2 Java

Java is a simple yet powerful object-oriented programming language and it is in many respects similar to C++. Java originated at Sun Microsystems, Inc. in 1991.

Java was designed with the concept of 'write once and run everywhere'. Java Virtual Machine plays a central role in this concept. The JVM is the environment in which Java programs execute. It is software that is implemented on top of real hardware and operating system. When the source code (.java files) is compiled, it is translated into byte codes and then placed into (.class) files. The JVM executes these bytecodes. So, Java byte codes can be thought of as the machine language of the JVM. A JVM can either interpret the bytecode one instruction at a time or the bytecode can be compiled further for the real microprocessor

using what is called a just-in-time compiler. The JVM must be implemented on a particular platform before compiled programs can run on that platform.

Types of Java Applications:

Web Application - Java is used to create server-side web applications. Currently, Servlet, JSP, Struts, JSF, etc. technologies are used.

Standalone Application - It is also known as the desktop application or window-based application. An application that we need to install on every machine or server such as media player, antivirus, etc. AWT and Swing are used in java for creating standalone applications.

Enterprise Application - An application that is distributed in nature, such as banking applications, etc. It has the advantage of high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

Mobile Application - Java is used to create an application software for mobile devices. Currently, Java ME is used for building applications for small devices, and also Java is a programming language for Google Android application development.

Features of Java:

Object-Oriented - Java supports the features of object-oriented programming. Its object model is simple and easy to expand.

Platform independent - C and C++ are platform dependency languages hence the application programs written in one Operating system cannot run in any other Operating system, but in platform independence language like Java application programs written in one Operating system can able run on any Operating system.

Simple - Java has included many features of C / C ++, which makes it easy to understand.

Secure - Java provides a wide range of protection from viruses and malicious programs. It ensures that there will be no damage and no security will be broken.

Portable - Java provides us with the concept of portability. Running the same program with Java on different platforms is possible.

Robust - During the development of the program, it helps us to find possible mistakes as soon as possible.

Multi-threaded - The multithreading programming feature in Java allows you to write a program that performs several different tasks simultaneously.

Distributed - Java is designed for distributed Internet environments as it manages the TCP/IP protocol.

2.3 UX Design

User Experience (UX) Design (UXD, UED, or XD) is the process of creating evidence-based, interaction designs between human users and products or websites. Design decisions in UX Design are driven by research, data analysis, and test results rather than aesthetic preferences and opinions. Unlike User Interface Design, which focuses solely on the design of a computer interface, UX Design encompasses all aspects of a user's perceived experience with a product or website, such as its usability, usefulness, desirability, brand perception, and overall performance. UX Design is also an element of the Customer Experience (CX), which encompasses all aspects and stages of a customer's experience and interaction with a company. Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.

Usability is attached to all tools used by humans and is extended to both digital and non-digital devices. Thus, it is a subset of user experience but not wholly contained. The section of usability that intersects with user experience design is related to humans' ability

to use a system or application. Good usability is essential to a positive user experience but does not alone guarantee it.

2.4 JavaFX

JavaFX is a software platform for creating and delivering desktop applications, as well as rich web applications that can run across a wide variety of devices. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS, as well as mobile devices running iOS and Android.

On desktops, JavaFX supports Windows Vista, Windows 7, Windows 8, Windows 10, macOS, and Linux operating systems. Beginning with JavaFX 1.2, Oracle has released beta versions for OpenSolaris. On mobile, JavaFX Mobile 1. x is capable of running on multiple mobile operating systems, including Symbian OS, Windows Mobile, and proprietary real-time operating systems.

JavaFX was intended to replace Swing as the standard GUI library for Java SE, but it has been dropped from new Standard Editions while Swing and AWT remain included, supposedly because JavaFX's market share has been "eroded by the rise of 'mobile first' and 'web first applications.'" With the release of JDK 11 in 2018, Oracle made JavaFX part of the OpenJDK under the OpenJFX project, to increase the pace of its development. Oracle support for JavaFX is also available for Java JDK 8 through March 2025.

Open-source JavaFX Ports works for iOS (iPhone and iPad) and Android and embedded (Raspberry Pi) and the related commercial software created under the name "Gluon" supports the same mobile platforms with additional features plus desktop. This allows a single source code base to create applications for the desktop, iOS, and Android devices

As of March 2014, JavaFX is deployed on Microsoft Windows, OS X, and Linux. Oracle has an internal port of JavaFX on iOS and Android. ARM support is available starting with JavaFX 8 On February 11, 2013.

2.5 AnimateFX & CSS

2.5.1 AnimateFX

A library of ready-to-use animations for JavaFX.

Features:

- Custom animations
- Custom interpolators
- Play/Stop animation
- Play an animation after another
- More to come

Maven

```
<repositories>
  <repository>
    <id>snapshots</id>
    <name>libs-snapshot</name>
    <url>https://oss.jfrog.org/artifactory/libs-snapshot</url>
  </repository>
</repositories>

<dependencies>
  <dependency>
    <groupId>io.github.typhon0</groupId>
    <artifactId>AnimateFX</artifactId>
    <version>1.2.2-SNAPSHOT</version>
  </dependency>
</dependencies>
```

Credits - <https://github.com/Typhon0/AnimateFX>

2.5.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup

language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/CSS is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

2.6 SQL

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables. SQL offers two main advantages over older read-write APIs such as ISAM or

VSAM. Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify how to reach a record, e.g. with or without an index.

Originally based upon relational algebra and tuple relational calculus, SQL consists of many types of statements, which may be informally classed as sublanguages, commonly: a data query language (DQL),[a] a data definition language (DDL),[b] a data control language (DCL), and a data manipulation language (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control. Although SQL is essentially a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages to use Edgar F. Codd's relational model. The model was described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986 and the International Organization for Standardization (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of standards, most SQL code requires at least some changes before being ported to different database systems.

2.7 Database System

In computing, a database is an organized collection of data stored and accessed electronically. Small databases can be stored on a file system, while large databases are hosted on computer clusters or cloud storage. The design of databases spans formal techniques and practical considerations including data modeling, efficient data representation and storage, query languages, security and privacy of sensitive data, and distributed computing issues including supporting concurrent access and fault tolerance.

A database management system (DBMS) is the software that interacts with end-users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The total of the database, the DBMS, and the associated applications can be referred to as a database system. Often the term "database" is also used loosely to refer to any of the DBMS, the database system, or an application associated with the database. Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, collectively referred to as NoSQL because they use different query languages.

2.8 JDBC

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented toward relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

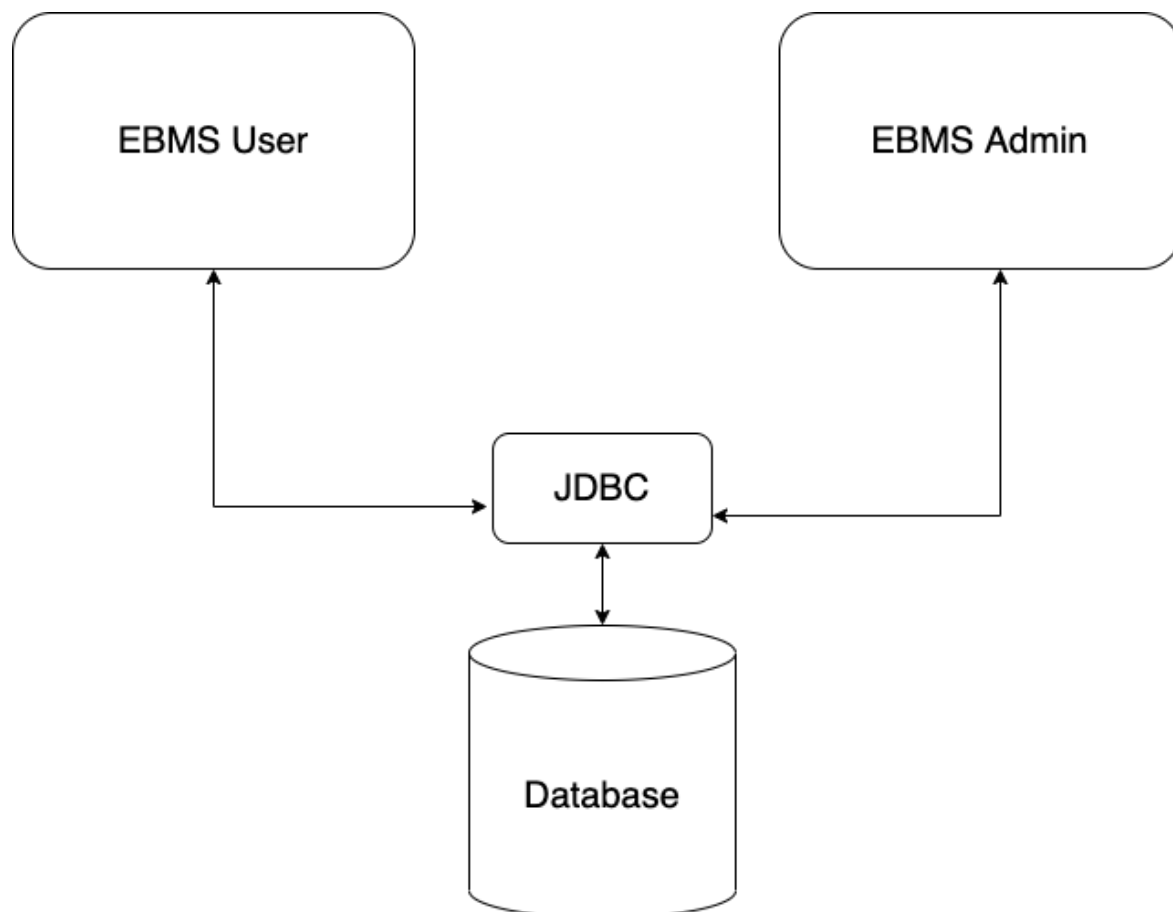
```
Connection conn = DriverManager.getConnection(
    "jdbc:somejdbcvender:other data needed by some jdbc vendor",
    "myLogin",
    "myPassword");
try {
    /* you use the connection here */
} finally {
    //It's important to close the connection when you are done with it
    try {
        conn.close();
    } catch (Throwable e) { /* Propagate the original exception
                           instead of this one that you want just
logged */
        logger.warn("Could not close JDBC Connection",e);
    }
}
```


System Design

3.1 Front-End Design

3.1.1 Overview

There are two applications in the project one app for the user and the other for the admin to manage the customers. This project shows DBMS support for multiple users.



3.1.2 Pseudocode for some complex methods

Code to delete messages in the inbox.

```

if(count == 0){
    waring_txt_inbox.setText("No Messages to Delete!");
} else {
    count = 0;
    String del = "DELETE FROM inbox";
    ButtonType good_btn = new ButtonType("Delete");
    ButtonType bad_btn = new ButtonType("Cancel");
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION, "",
good_btn, bad_btn);
    alert.setHeaderText(null);
    alert.setContentText("Are you sure you want to empty your
inbox, messages will be deleted permanently!!!");
    alert.initStyle(StageStyle.UNDECORATED);
    alert.initStyle(StageStyle.TRANSPARENT);
    alert.getDialogPane().setStyle("-fx-background-color:
#2e9cca; -fx-border-color: #2e9cca; -fx-border-radius: 30; -
fx-background-radius: 30; -fx-border-width: 10;");

alert.getDialogPane().getScene().setFill(Color.TRANSPARENT);
    new FadeIn(alert.getDialogPane()).play();
    Optional<ButtonType> result = alert.showAndWait();
    result.ifPresent(res -> {
        if (res.equals(good_btn)) {
            try {
                this.s = this.c.createStatement();
                this.s.executeUpdate(del);
                r.next();
                waring_txt_inbox.setText("Messages deleted
Successfully!");
                inbox_table.getItems().clear();
                list = null;
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    });
}

```

Code for Adding new Customers

```

if(count == 0){
    txt_box.setText("No Customers!!!");
} else {
    list.clear();
    name_search.setText("");
    txt_box.setText("");
    cus_table.getItems().clear();
    try {
        this.s = this.c.createStatement();
        //String s = "SELECT * FROM CUSTOMER_DETAILS";
        //this.r = this.s.executeQuery(s);
        CallableStatement callableStatement = c.prepareCall("{CALL
DISPLAY_CUSTOMERS}");
        callableStatement.execute();
        this.r = callableStatement.getResultSet();
        while (r.next()) {
            list.add(new getCustomerDetails(r.getInt("c_id"),
r.getString("name"), r.getString("email"), r.getString("occupation"),
r.getBigDecimal("phone"), r.getString("dob"), r.getInt("flat_no"),
r.getString("address"), r.getInt("pincode"), r.getString("state"),
r.getString("city")));
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    try {
        cid_col.setCellValueFactory(new
PropertyValueFactory<>("c_id"));
        name_col.setCellValueFactory(new
PropertyValueFactory<>("name"));
        email_col.setCellValueFactory(new
PropertyValueFactory<>("email"));
        occ_col.setCellValueFactory(data -> new
SimpleStringProperty(data.getValue().getOcc()));
        phone_col.setCellValueFactory(new
PropertyValueFactory<>("phone"));
        dob_col.setCellValueFactory(new PropertyValueFactory<>("dob"));
        flat_col.setCellValueFactory(new
PropertyValueFactory<>("flat_no"));
        address_col.setCellValueFactory(new
PropertyValueFactory<>("address"));
        pincode_col.setCellValueFactory(data -> new
SimpleStringProperty(data.getValue().getPin_code()));
        state_col.setCellValueFactory(new
PropertyValueFactory<>("state"));
        city_col.setCellValueFactory(new
PropertyValueFactory<>("city"));
        cus_table.setItems(list);
    } catch (Exception e){
        System.out.println(e);
    }
}

```

Code for user Home page

```

Image image = new Image("file:/Users/nithinr/ESMS App/ESMS
User/src/main/images/icons/icon-modified.png");
    Image image1 = new Image("file:/Users/nithinr/ESMS
App/ESMS User/src/main/images/icons/Untitled design-
modified.png");
    Image acc = new Image("file:/Users/nithinr/ESMS App/ESMS
User/src/main/images/icons/acc.png");
    Image bill = new Image("file:/Users/nithinr/ESMS App/ESMS
User/src/main/images/icons/bill.png");
    Image his = new Image("file:/Users/nithinr/ESMS App/ESMS
User/src/main/images/icons/history.png");
    Image ch_pass = new Image("file:/Users/nithinr/ESMS
App/ESMS User/src/main/images/icons/ch_pass.png");
    Image logout = new Image("file:/Users/nithinr/ESMS
App/ESMS User/src/main/images/icons/logout.png");
    Image issue = new Image("file:/Users/nithinr/ESMS App/ESMS
User/src/main/images/icons/issue.png");
    Image img2 = new Image("file:/Users/nithinr/ESMS App/ESMS
User/src/main/images/icons/billPay.png");
    Image img3 = new Image("file:/Users/nithinr/ESMS App/ESMS
User/src/main/images/icons/user.png");
    try {
        user_c_img.setImage(img3);
        c_bill_img.setImage(img2);
    } catch (Exception e){

    }
    acc_img.setImage(acc);
    icon2.setImage(image);
    bill_img.setImage(bill);
    user_icon.setImage(image1);
    his_img.setImage(his);
    pass_ch_img.setImage(ch_pass);
    logout_img.setImage(logout);
    complaint_img.setImage(issue);
    try {
        c =
DriverManager.getConnection("jdbc:mysql://localhost:3306/ESM",
"root", "MySQL@123");
    } catch (SQLException e) {
        System.out.println("error " + e.getMessage());
    }
    try {
        BufferedReader bufferedReader = new BufferedReader(new
FileReader("/Users/nithinr/ESMS App/ESMS
User/src/main/java/com/example/esms_user/c_Id"));
        c_id = Integer.parseInt(bufferedReader.readLine());

```

```

        bufferedReader.close();
        String sql = "SELECT name, email, address FROM
CUSTOMER_DETAILS WHERE c_id=" + c_id + ";";
        s = c.createStatement();
        r = s.executeQuery(sql);
        r.next();
        try {
            add_txt.setText(r.getString("address"));
            name_txt.setText(r.getString("name"));
            email_txt.setText(r.getString("email"));
        } catch (Exception e) {

        }
    } catch (Exception e) {

    }
    try {
        BufferedReader bufferedReader = new BufferedReader(new
FileReader("/Users/nithinr/ESMS App/ESMS
User/src/main/java/com/example/esms_user/c_Id"));
        c_id = Integer.parseInt(bufferedReader.readLine());
        bufferedReader.close();
        String sql = "SELECT * FROM BILLS WHERE c_id=" + c_id
+ ";";
        String bill_no = "";
        String amt_pay = "";
        s = c.createStatement();
        r = s.executeQuery(sql);
        r.next();
        bill_no = r.getString("billno");
        amt_pay = r.getString("amt");
        try {
            no_txt.setText(bill_no);
            amt_txt.setText(amt_pay);
        } catch (Exception e) {

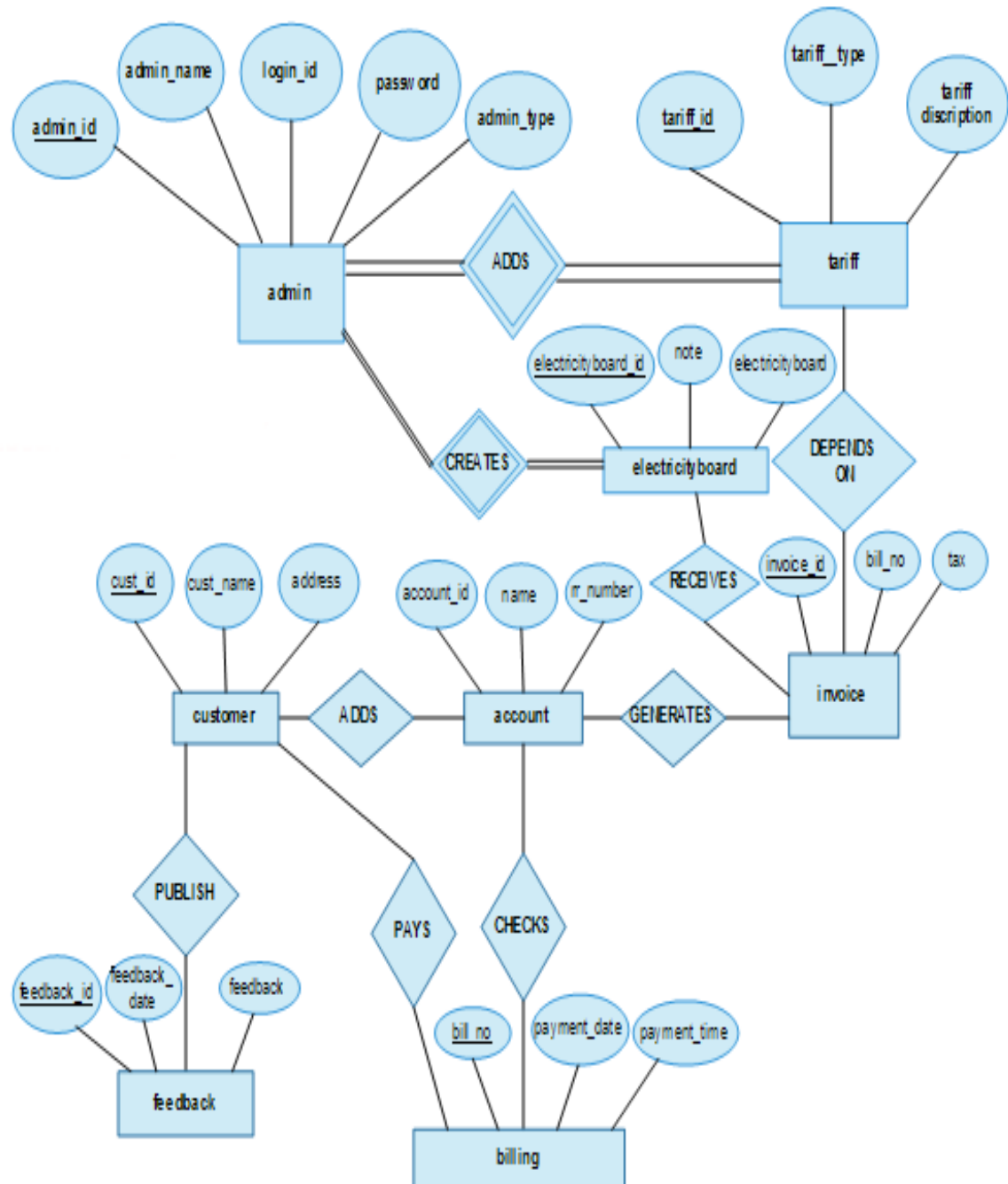
        }
    } catch (Exception e) {
        try {
            no_txt.setText("All Bills Paid Good Job!");
            r_txt.setVisible(false);
            pay_btn.setVisible(false);
        } catch (Exception e1) {

        }
    }
}
}

```


3.2 Database Design

3.2.1 Entity-Relationship Diagram




3.2.2 Database tables


Admin table

admin_login	
 name	char
password	varchar


Inbox

inbox_admin	
name	char
email	varchar
message	int
 c_id	int


User Login

user_login	
 c_id	int
name	char
password	varchar


Customer Details

customer_details	
 c_id	int
name	char
email	varchar
occupation	char
phone	bigint
dob	date
flat_no	int
address	varchar
pincode	int
state	char
city/village	char


Bills

BILLS	
 c_id	int
name	char
amt_topay	int
due_date	date

Bills Paid

BILLS_PAID	
 c_id	int
name	char
bill_amt	int
bill_paid_date	date

Card

CARD	
 c_id	int
card_num	varchar
card_name	varchar

Implementation

4.1 SQL Operations

Insertion

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

```
INSERT INTO TABLE_NAME (column1, column2,
column3,...column N)
VALUES (value1, value2, value3,...value N);
```

Here, column1, column2, column3,...column N are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The **SQL INSERT INTO** syntax will be as follows –

```
INSERT INTO TABLE_NAME VALUES
(value1,value2,value3,...value N);
```

Inserting New Customer Values

```
if(valid){
    String insert_cus = "INSERT INTO
CUSTOMER_DETAILS(c_id,name,email,occupation,phone,dob,flat_no,a
ddress,pincode,state,city) " +
    "VALUES " +
    "(" + c_id + "," + name_field + "," + email_field + "," +
occupation_field + "," + phone_field + "," + date_field + "," + flat_no_field
+ "," + address + "," + pincode_field + "," + state + "," + city_field + "));";
    String insert_user = "INSERT INTO user_login(c_id, user_name,
password)" +
    "VALUES (" + c_id + "," + user_name_field + "," + pass_field
+ "));";
```

```

        this.s = this.c.createStatement();
        this.s.executeUpdate(insert_cus);
        this.s.executeUpdate(insert_user);
        text_waring.setText("Successfully added new Connection");
    }

```

Deletion

The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise, all the records would be deleted.

Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows –

```

DELETE FROM table_name
WHERE [condition];

```

You can combine N number of conditions using AND or OR operators.

Deleting Existing customer

```

if (res.equals(good_btn)) {

    String sql1 = "DELETE FROM CUSTOMER_DETAILS WHERE
c_id=" + c_id + ";";

    String sql2 = "DELETE FROM user_login WHERE c_id=" + c_id
+ ";";

    try {

        this.s = this.c.createStatement();

        this.s.executeUpdate(sql1);

        this.s.executeUpdate(sql2);

    } catch (SQLException e) {

        e.printStackTrace();
    }
}

```

```

    }

    del_cus.setText("Deleted Successfully!");

}

```

Stored Procedure

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure so that the stored procedure can act based on the parameter value(s) that is passed.

Stored Procedure Syntax

```

CREATE PROCEDURE procedure_name
AS
sql_statement
GO;

```

Execute a Stored Procedure

```
EXEC procedure_name;
```

Stored Procedure for getting Customer Details

Java Code

```

this.s = this.c.createStatement();

        CallableStatement callableStatement = c.prepareCall("{CALL
DISPLAY_CUSTOMERS}");

        callableStatement.execute();

        this.r = callableStatement.getResultSet();

        while (r.next()) {

                list.add(new                getCustomerDetails(r.getInt("c_id"),
r.getString("name"),    r.getString("email"),    r.getString("occupation"),
r.getBigDecimal("phone"),    r.getString("dob"),    r.getInt("flat_no"),

```

```

r.getString("address"),    r.getInt("pincode"),    r.getString("state"),
r.getString("city"));
    }

```

SQL code

```

USE `ESM`;

DROP PROCEDURE IF EXISTS `DISPLAY_CUSTOMERS`;

DELIMITER $$

USE `ESM` $$

CREATE PROCEDURE `DISPLAY_CUSTOMERS` ()

BEGIN

    SELECT * FROM CUSTOMER_DETAILS;

END$$

DELIMITER ;

```

Trigger

Trigger: A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Syntax:

```

create trigger [trigger_name]

[before | after]

{insert | update | delete}

on [table_name]

[for each row]

```

[trigger_body]

Explanation of syntax:

create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.

[before | after]: This specifies when the trigger will be executed.

{insert | update | delete}: This specifies the DML operation.

on [table_name]: This specifies the name of the table associated with the trigger.

[for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.

[trigger_body]: This provides the operation to be performed as the trigger is fired

SQL Code

USE ESM;

DROP TRIGGER IF EXISTS `a`;

DELIMITER //

USE `ESM`//

CREATE TRIGGER `ESM`.`a`

AFTER INSERT

ON new_table DELIMITER \$\$

FOR EACH ROW

BEGIN

INSERT INTO AB VALUES();

END//

DELIMITER ;

4.2 Java Code

Code for switching application tabs

```

public void new_cus(ActionEvent event) throws IOException {
    try {
        Stage stage1 = (Stage) new_conn_btn.getScene().getWindow();
        stage1.close();

        Parent root =
        FXMLLoader.load(getClass().getResource("new_conn_pane.fxml"));

        stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        scene = new Scene(root, 1280, 720);
        scene.setFill(Color.TRANSPARENT);
        stage.setScene(scene);
        stage.show();
        new FadeIn(root).play();
    } catch (Exception e){
        System.out.println(e);
    }
}

public void my_acc_pane(ActionEvent event) throws IOException{
    Stage stage1 = (Stage) inbox_btn.getScene().getWindow();
    stage1.close();

    Parent root = FXMLLoader.load(getClass().getResource("admin_UI.fxml"));

```

```

stage = (Stage) ((Node)event.getSource()).getScene().getWindow();

scene = new Scene(root, 1280, 720);

scene.setFill(Color.TRANSPARENT);

stage.setScene(scene);

stage.show();

new FadeIn(root).play();
}

```

```

public void inbox(ActionEvent event) throws IOException {

    Stage stage1 = (Stage) my_acc_btn.getScene().getWindow();

    stage1.close();

    Parent root = FXMLLoader.load(getClass().getResource("inbox_pane.fxml"));

    stage = (Stage) ((Node)event.getSource()).getScene().getWindow();

    scene = new Scene(root, 1280, 720);

    scene.setFill(Color.TRANSPARENT);

    stage.setScene(scene);

    stage.show();

    new FadeIn(root).play();

}

```

```

public void bills_pane(ActionEvent event) throws IOException{

    Stage stage1 = (Stage) bills_btn.getScene().getWindow();

    stage1.close();

    Parent root = FXMLLoader.load(getClass().getResource("bills_pane.fxml"));

```



```

stage = (Stage) ((Node)event.getSource()).getScene().getWindow();

scene = new Scene(root, 1280, 720);

scene.setFill(Color.TRANSPARENT);

stage.setScene(scene);

stage.show();

new FadeIn(root).play();
}

```

```

public void customer_details(ActionEvent event) throws IOException{

    Stage stage1 = (Stage) customer_deatils_btn.getScene().getWindow();

    stage1.close();

    Parent root =
FXMLLoader.load(getClass().getResource("customer_details.fxml"));

    stage = (Stage) ((Node)event.getSource()).getScene().getWindow();

    scene = new Scene(root, 1280, 720);

    scene.setFill(Color.TRANSPARENT);

    stage.setScene(scene);

    stage.show();

    new FadeIn(root).play();

}

```

Testing

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

5.1 Testing process

Testing is an integral part of software development. The testing process, in a way, certifies, whether the product, that is developed, compiles with the standards, that it was designed to. The testing process involves the 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.

5.2 Testing objectives

The main objectives of the testing process are as follows:

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

Conclusion

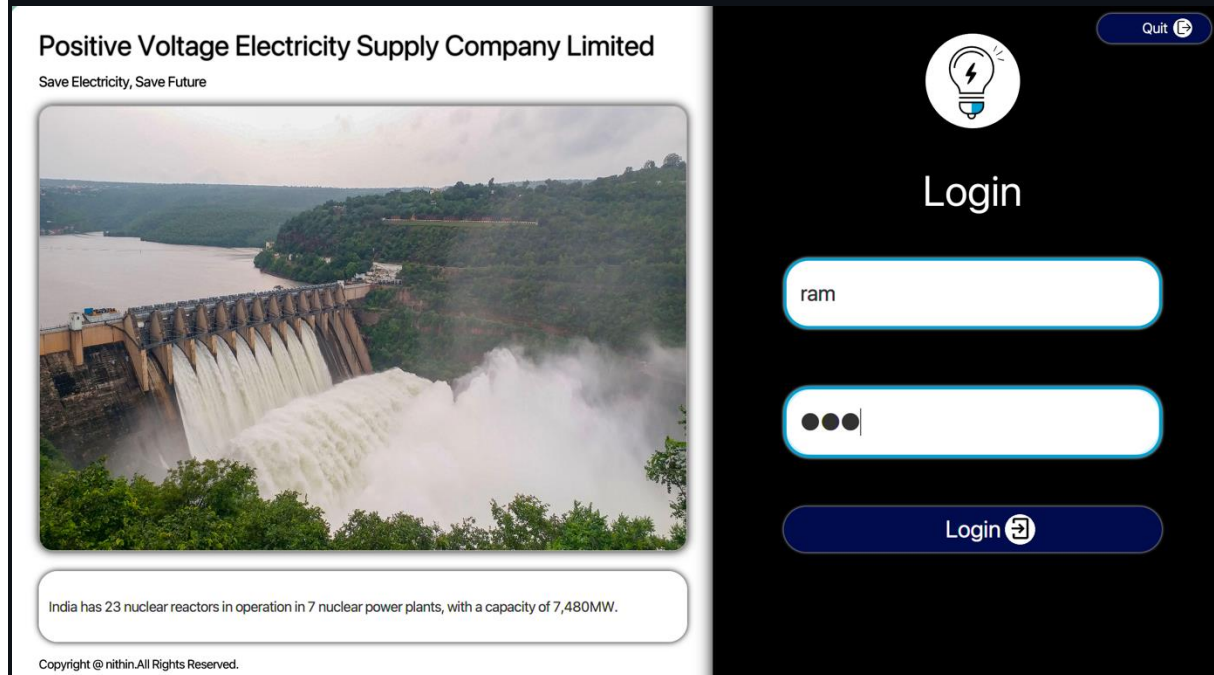
We have tried to develop a system that can be a great help for the owner of the referred electricity department to receive bills from the customer. Despite all our efforts, there are some bugs in the system, which are still to be removed. This is possible by the testing being done in the system. We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them. In the last, we would like to thank all the persons involved in the development of the system directly or indirectly. We are also thankful to YouTube, Stack overflow, Geeks For Geeks, Java point, Tutorials point, W3 Schools for the help in developing this project. By developing this project we enhanced the following skills Java, JavaFX, CSS, SQL, DBMS, UX Design, OOP which are a boost to our resume. We hope that the project will serve the purpose for which it is developed thereby underlining the success of the process. We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them. In the last, we would like to thank all the persons involved in the development of the system directly or indirectly. We hope that the project will serve the purpose for which it is developed thereby underlining the success of the process.

Application Snapshot

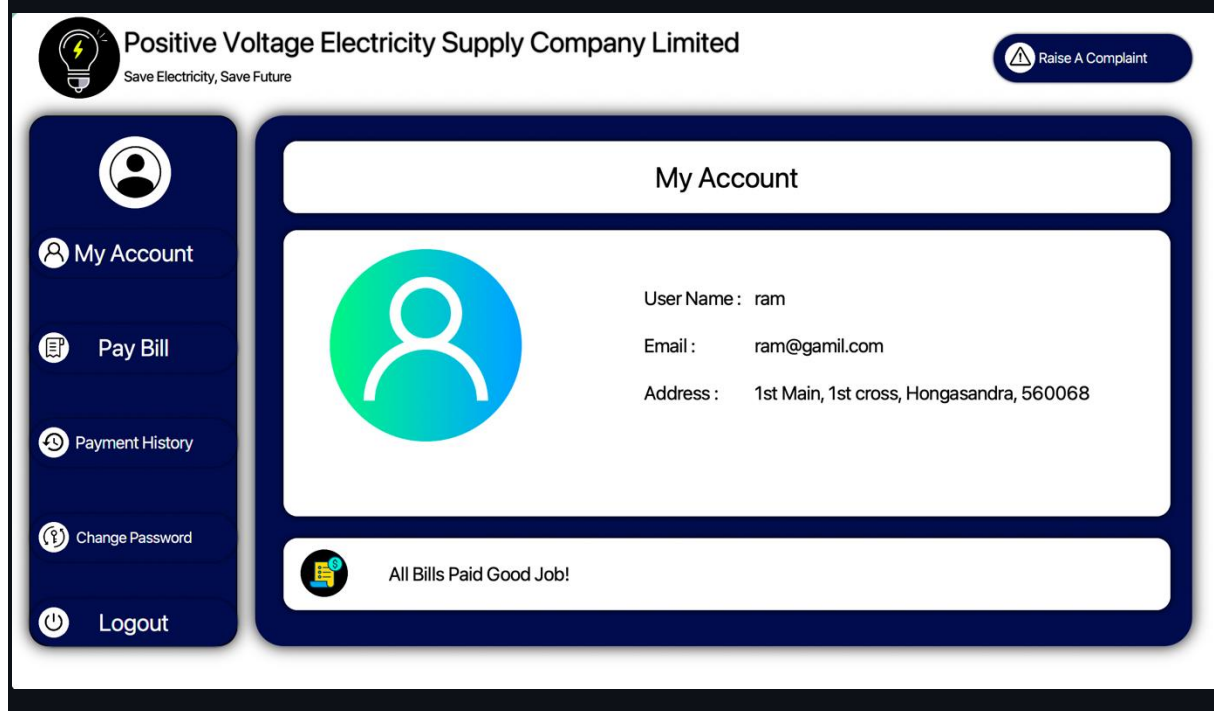
User APP

Login Page


Here There is Image SildeShow & SildeShow about facts on the Indian Power Grid.





User UI



Payment


Positive Voltage Electricity Supply Company Limited
 Save Electricity, Save Future


 Raise A Complaint



My Account

Pay Bill

Payment History


Change Password

Logout

Pay


Bill Details


Bill No	Name	Bill Date	Kilo Watt
2022-01-09606100	ram	2022-01-09	132




₹ 792

Pay


Positive Voltage Electricity Supply Company Limited
 Save Electricity, Save Future


 Raise A Complaint



My Account

Pay Bill

Payment History

Change Password

Logout

Card Details

Cancel

Card Number

Card Name

CVV


Save Card

₹ 792


Pay


Transaction History

More History will come by doing more transactions like in PhonePe.




Positive Voltage Electricity Supply Company Limited
Save Electricity, Save Future


 Raise A Complaint




My Account




Pay Bill



Payment History







Change Password



Logout

Transaction History

	2022-01-08858100	618	2022-01-09	Download Invoice
	2022-01-09364100	624	2022-01-09	Download Invoice
	2022-01-09800100	624	2022-01-09	Download Invoice
	2022-01-09461100	600	2022-01-09	Download Invoice

Change Password



Positive Voltage Electricity Supply Company Limited
Save Electricity, Save Future

 Raise A Complaint



My Account



Pay Bill



Payment History



Change Password



Logout

Current Password


New Password

[Generate A password](#)


Confirm Password

[Submit](#)

Complaint Section


Positive Voltage Electricity Supply Company Limited
 Save Electricity, Save Future

Raise A Complaint


 My Account
 Pay Bill
 Payment History
 Change Password
 Logout


Complaints will be seen and surely solved. Please raise your complaint below. Please be Respectful Thank You.

Send


ADMIN APP

Login Page


Here There is Image SildeShow & SildeShow about facts on the Indian Power Grid.

Electricity Bill Management System


Save Electricity, Save Future



India's hydroelectric power potential is estimated at 148,700 MW at 60% load factor.


 Quit

User Name

Password

Login

Copyright @ nithin.All Rights Reserved.

Admin UI

Electricity Bill Management System

Save Electricity, Save Future

My Account

New Connection

Bills

Customer Details

Inbox

Logout

My Account

Admin page Name - admin Password -admin

Copyright @ Nithin R, The Oxford College Of Engineering. Built using Java, JavaFx, AnimateFx. IDE used IntelliJ, Scene Builder, SQL - MySQL workbench. CSS used for additional styling. Database name ESM, 7 tables are there.Database Username = root, Password = MySQL@123

New Connection

Regular Expressions are used for checking the inputs.

Electricity Bill Management System

Save Electricity, Save Future

My Account

New Connection

Bills

Customer Details

Inbox

Logout

New Connection

New Connection Request Form

All Fileds are Mandatory

Name & Email

Name

Email

Phone Number

Consumer Demographic Info

Bills

Electricity Bill Management System

Save Electricity, Save Future

- My Account
- New Connection
- Bills
- Customer Details
- Inbox
- Logout

Bills

Enter the Customer Id for Billing

125

Search

Bill Generated for 125

Enter the Kilo Watt customer has consumed

123

Submit

Customer Details

Search By Name

Search Show All Clear

Terminate Connection

Customer ID	Name	Email	Occupation	Phone
104	nithin	nithin@gmail.com	student	20
105	nithin	nithin@gmail.com	student	20
117	nithin	nithin@gmail.com	student	20
119	ram	ram@gamil.com	police	19
120	roy	roy	actor	19
125	Nithin R	nithinr101010@gmail.com	Student	20

Inbox

Electricity Bill Management System

Save Electricity, Save Future

My Account

New Connection

Bills

Customer Details

Inbox

Logout

Inbox

Search By Name

Search

Show

Clear

Delete All

Action	Customer Id	Name	Email	
Delete	119	ram	ram@gamil.com	learn DSALGO
Delete	120	roy	roy	learn mem stack

Electricity Bill Management System

Save Electricity, Save Future

My Account

New Connection

Bills

Customer Details

Inbox

Logout

Inbox

Search By Name

Search

Show

Clear

Delete All

Are you sure you want to Log out any unsaved changes will be lost!

Log Out

Cancel

Action	Customer Id	Name	Email	
Delete	119	ram	ram@gamil.com	learn DSALGO
Delete	120	roy	roy	learn mem stack

Miscellaneous

This project is available on GitHub

<https://github.com/Nithin0001/Electricity-Bill-Management-System>

AnimateFx

<https://github.com/Typhon0/AnimateFX>

JavaFX

<https://openjfx.io/>

IntelliJ

<https://www.jetbrains.com/idea/>

Scene Builder

<https://gluonhq.com/products/scene-builder/>

MySQL Workbench

<https://www.mysql.com/products/workbench/>

Database Name - ESM

Username - root

Password - MySQL@123