# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018



A Mini Project Report on

## ELECTRICITY BILLING SYSTEM

Submitted in partial fulfillment of the requirements as a part of the DBMS lab for the V semester of degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

**Submitted by**

**Sidharth S Pai - 1RN20IS160**

**Surabhi S - 1RN20IS168**

**Under the Guidance of**

**Faculty Incharge**  **Lab Incharge**

**Dr. Prakasha S Dr. R Rajkumar**

##### Associate Professor Dept. of ISE, RNSIT

**Associate Professor Dept. of ISE, RNSIT**



**Department of Information Science and Engineering**

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, RR Nagar Post, Bengaluru – 560 098

2022 – 2023

**RNS INSTITUTE OF TECHNOLOGY**



Channasandra, Dr.Vishnuvardhan Road, RR Nagar Post, Bengaluru – 560 098

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the Mini project report entitled **ELECTRICITY BILLING SYSTEM** has been successfully completed by **SIDHARTH S PAI** bearing **USN 1RN20IS160 and SURABHI S** bearing **USN 1RN20IS168**, presently V semester student of **RNS Institute of Technology** in partial fulfillment of the requirements as a part of the DBMS Laboratory for the award of the degree **Bachelor of Engineering in Information Science and Engineering** under **Visvesvaraya Technological University, Belagavi** during academic year 2022 – 2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements as a part of DBMS Laboratory for the said degree.

|  |  |  |
| --- | --- | --- |
| **Dr. Prakasha S** | **Dr. R Rajkumar** | **Dr. Suresh L** |
| Faculty Incharge | Lab Incharge | Professor and HOD |
|  | **External Viva** |  |
| **Name of the Examiners** |  | **Signature with date** |
| 1. |  |  |
| 2. |  |  |

We, **SIDHARTH S PAI [USN: 1RN20IS160**] and **SURABHI S [USN: 1RN20IS168],** students of V Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Mini project entitled **ELECTRICITY BILLING SYSTEM** has been carried out by us and submitted in partial fulfilment of the requirements for the V Semester degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2022-2023.

Place: Bengaluru

**SIDHARTH S PAI**

**[1RN20IS160]**

**SURABHI S**

**[1RN20IS168]**

Electricity consumers are often faced with the problem of inaccuracy and delay in monthly billing due to some drawbacks. Thus, it is essential to have an efficient system for such purposes via electronic platform with consideration to proximity. The proposed system automates the conventional process of paying electricity bill by visiting the Electricity Board which is tiresome and time consuming. It is also designed to automate the electricity bill calculation and payment for user convenience. The system is developed with Java swings as the base programming language which can be used to develop websites, web applications and web services. The Microsoft Structured Query Language (SQL) server is also used for creating back-end database. The system would be having two logins: the administrative and user login. The administrator can view the user's account details and can add the customer's information of consuming units of energy of the current month in their account. The Admin must feed the system with the electricity usage data into respective user’s account. The system then calculates the electricity bill for every user and updates the information into their account every month. Users can then view their electricity bill and pay before the month end.

# ACKNOWLEDGEMENT

The fulfillment and rapture that go with the fruitful finishing of any assignment would be inadequate without the specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

We would like to profoundly thank **Management** of **RNS Institute of Technology** for providing such a healthy environment to carry out this Project work.

We would like to express our thanks to our Principal **Dr. M K Venkatesha** for his support and inspired us towards the attainment of knowledge.

We wish to place on record our words of gratitude to **Dr. Suresh L**, Professor and Head of the Department, Information Science and Engineering, for being the enzyme and master mind behind our Project work.

We would like to express our profound and cordial gratitude to our Laboratory Incharge **Dr. R Rajkumar**, Associate Professor, Department of Information Science and Engineering for his valuable guidance, constructive comments and continuous encouragement throughout the Project work.

We would like to express our profound and cordial gratitude to our Faculty Incharge **Dr. Prakasha S**, Associate Professor, Department of Information Science and Engineering for his valuable guidance in preparing Project report.

We would like to thank all other teaching and non-teaching staff of Information Science & Engineering who have directly or indirectly helped us to carry out the project work.

And lastly, We would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in carrying out this Project work.

**SIDHARTH S PAI (1RN20IS160) SURABHI S (1RN20IS168)**

## TABLE OF CONTENTS

**CERTIFICATE** i

DECLARATION ii

ABSTRACT iii

[ACKNOWLEDGEMENT iv](#_TOC_250001)

[TABLE OF CONTENTS v](#_TOC_250000)

1. INTRODUCTION 1
2. LITERATURE SURVEY 2
3. SYSTEM REQUIREMENTS 7
4. SYSTEM DESIGN 8
   1. Description of ER Diagram

4.2 Description of Relational Schema Diagram

1. IMPLEMENTATION 10
   1. Front-end Development
   2. Back-end Development
   3. SQL implementation
      1. Discussion of Code Segment
      2. Procedures and Triggers
2. **SNAPSHOTS** 17
3. **CONCLUSION AND FUTURE ENHANCEMENTS** 26
   1. Conclusion
   2. Future Enhancement

**REFERENCES** 27

# INTRODUCTION

Electricity Billing System is a software-based application.

1. This project aims at serving the department of electricity by computerizing the billing system.
2. It mainly focuses on the calculation of units consumed during the specified time and the money to be charged by the electricity offices.
3. 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.

# LITERATURE SURVEY

## 2.1 Traditional File System

In the early days of computing, data management and storage were a very new concept for organizations. The traditional approach to data handling offered a lot of the convenience of the manual approach to business processes (e.g. handwritten invoices & account statements, etc.) as well as the benefits of storing data electronically.

The traditional approach usually consisted of custom-built data processes and computer information systems tailored for a specific business function. An accounting department would have their own information system tailored to their needs, where the sales department would have an entirely separate system for their needs.

Initially, these separate systems were very simple to set up as they mostly mirrored the business process that departments had been doing for years but allowed them to do things faster with less work. However, once the systems were in use for so long, they became very difficult for individual departments to manage and rely on their data because there was no reliable system in place to enforce data standards or management.

Separate information systems for each business function also led to conflicts of interest within the company. Departments felt a great deal of ownership for the data that they collected, processed, and managed which caused many issues among company-wide collaboration and data sharing.

## 2.2 Pros and Cons of the Traditional Approach

***2.2.1 Pros***

### Simple

* Matched existing business processes and functions.
* Companies were not as interested in funding complicated information systems

Dept of ISE, RNSIT 2021-22 Page 2

**Electricity Billing System Literature Survey**

##### Initially low-cost

* Early computing was not viewed as beneficial for large funding.
* Systems were designed to be cheap in order to save on cost.

***2.2.2 Cons***

##### Separated ownership

* + Business functions had a high sense of data ownership.
  + Departments unwilling to share data for fear of minimizing their superiority.

##### Unmanaged redundancy

* + Multiple instances of the same data appeared throughout various files, systems, and databases.
  + Information updated in one place was not replicated to the other locations.
  + Disk space was very expensive, and redundancy had a big impact on storage.

##### Data in consistency

* + Redundant data stored in various locations was usually never stored the same way.
  + Formatting was not centrally managed.

##### Lack of data sharing

* + Same data stored in multiple locations.
  + Caused unnecessary doubling of efforts for processing and managing data.

##### High costs in the long run

* + Hiring data processors for each department was very expensive, and each position was typically working on the same thing just for a different area.
  + Doubling of work as well as excessive maintenance cost.

## 2.3 Downfall of Traditional Management System

Conceived in a relatively centralized era when software was deployed in static environments, legacy database architectures fail to support an increasingly mobile world where applications are accessed anytime, anywhere.

Today software users want consistent improvements in usability and expect SaaS vendors to deliver new features and functionalities needed to achieve their business objectives. However, legacy database technologies fall short. in serving the needs of todays distributedand cloud environments for the following reasons:

* Inadequate fail over capabilities
* Insufficient provisions during peak demands Latency issues
* Lack of high availability at all times Increasing operational costs
* Inability to meet the demands of global markets

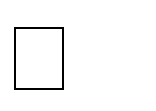
For all of these reasons, traditional databases are unable to deliver results in a rapidly growing environment where the workload is geographically distributed across heterogeneous datacenters. Upgrading to a more distributed data model is costly and complicated and your DBAs can’t just sit back and give up on this situation. Hence, due to these various reasons, the downfall of the traditional system was inevitable.

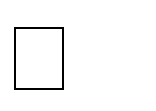
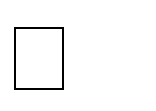
## 2.4 Introduction to the Database Management System

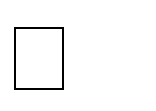
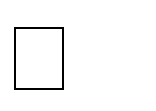
A database management system (DBMS) refers to the technology for creating and managing databases. Basically, a DBMS is a software tool to organize (create, retrieve, update and manage) data in a database.

The main aim of a DBMS is to supply a way to store and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database. A datum is a unit of Data. Meaningful data combines to form Information. Hence, information is interpreted data- data provided with semantics.MS ACCESS is one of the most common examples of database management software. Database systems are meant to handle large collection of information Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crash or attempts at unauthorized access.

## 2.5 Indicative areas for the use of a DBMS

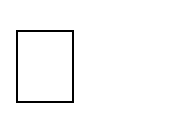
 Airlines: reservations, schedules etc.

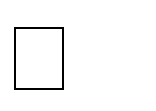
Telecom : calls made, customer details, network usage etc. Universities : registration, results, grades, etc.

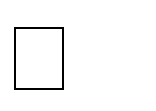
Sales: products, purchases, customers etc. Banking: all transactions

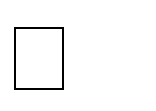
## 2.6 Advantages of a DBMS-

A Database Management System has many advantages over the traditional file system used in the earlier days, such as:

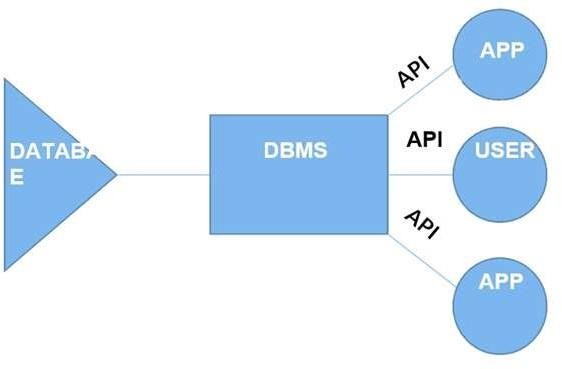
**Data independence**: Application programs should be as free or independent as possible from details of data representation and storage. DBMS can supply an abstract view of the data for insulating application code from such facts.

**Efficient data access**: DBMS utilize a mixture of sophisticated concepts and techniques for storing and retrieving data competently and this feature becomes important in cases where the data is stored on external storage devices.

**Data integrity and Security** : If data is accessed through the DBMS , the DBMS can enforce integrity constraint on the data

**Data administration**: When several users share the data, integrating the administration of data can offer major improvements. Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation toreduce redundancy and make the data to retrieve efficiently

**2.7 Components of a DBMS**



**Fig no. 2.1 Components of DBMS**

* **Users**: Users may be of any kind, such as data base administrators, system developers or database users.
* **Database application**: Database application may be Departmental ,Personal, Organizational and /or Internal
* **DBMS**: Software that allows users to create and manipulate database access.
* **Database**: Collection of logical data as a single unit.

## Chapter 3

**SYSTEM REQUIREMENTS**

The main purpose of this SRS document is to illustrate the requirements of the project Student information System and is intended to help any organization to maintain and manage its student’s personal data.

## 3.1 Hardware Requirements

* Processor : Intel Core i5
* RAM : 6GB
* Hard Disk :1TB

## 3.2 Software Requirements

##### Technologies Used:

* + Front End : JAVA CORE/SWINGS
  + Connection/Controller : JAVA
  + Back-End Database : MySQL
  + Text Editor : Sublime Text
  + Operating System : Windows10/Linux
  + Database Support : MySQL5
  + Back-End : MySQL Workbench.

# Chapter 4

**SYSTEM DESIGN**

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

## 4.1 Entity Relation Diagram

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database

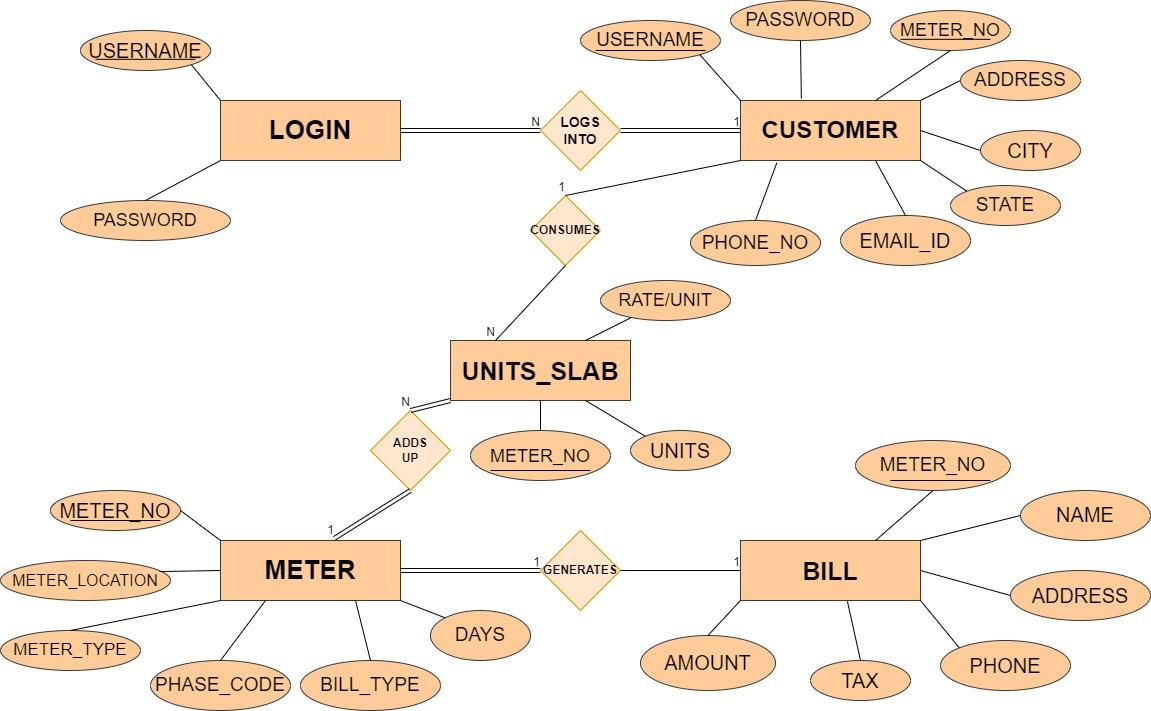


Fig .4.1 Entity Relational Diagram

## 4.2 Schema Diagram

A schema contains schema objects, which could be tables, columns, data types, store procedures, relationships, primary keys, foreign keys. A database schema can be represented in a visual diagram, which shows the database objects and their relationship with each other.

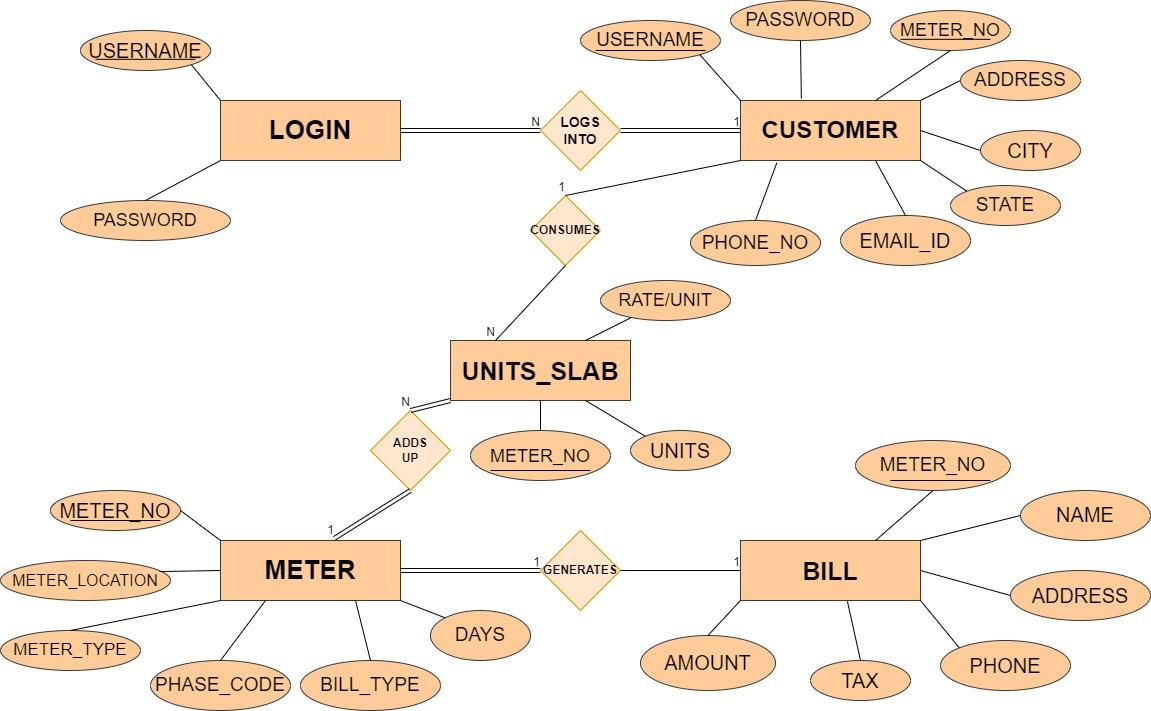


Fig 4.2 Schema diagram of Student Result Management System

### Chapter 5

### IMPLEMENTATION

**Swing**is a Java Foundation Classes [JFC] library and an extension of the Abstract Window Toolkit [AWT]. Swing offers much-improved functionality over AWT, new components, expanded components features, and excellent event handling with drag-and-drop support.

Swing has about four times the number of User Interface [UI] components as AWT and is part of the standard Java distribution. By today’s application GUI requirements, AWT is a limited implementation, not quite capable of providing the components required for developing complex GUI’s required in modern commercial applications. The AWT component set has quite a few bugs and really does take up a lot of system resources when compared to equivalent Swing resources. Netscape introduced its Internet Foundation Classes [IFC] library for use with Java. Its Classes became very popular with programmers creating GUI’s for commercial applications.

* Swing is a Set Of API ( API- Set Of Classes and Interfaces )
* Swing is Provided to Design Graphical User Interfaces
* Swing is an Extension library to the AWT (Abstract Window Toolkit)
* Includes New and improved Components that have been enhancing the looks and Functionality of GUIs’
* Swing can be used to build(Develop) The Standalone swing GUI Apps Also as Servlets And Applets
* It Employs model/view design architecture
* Swing is more portable and more flexible than AWT, The Swing is built on top of the AWT
* Swing is Entirely written in Java
* Java Swing Components are Platform-independent And The Swing Components are lightweight
* Swing Supports a Pluggable look and feels And Swing provides more powerful components
* such as tables, lists, Scrollpanes, Colourchooser, tabbedpane, etc
* Further Swing Follows MVC.

Many programmers think that JFC and Swing are one and the same thing, but that is not so.

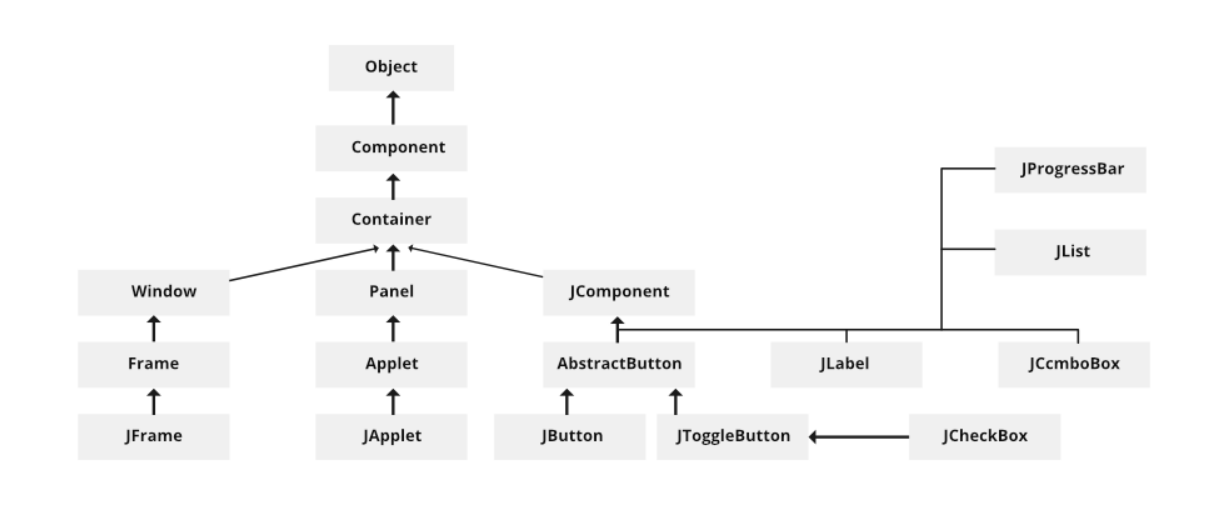
JFC contains Swing [A UI component package] and quite a number of other items:

* Cut and paste: Clipboard support
* Accessibility features: Aimed at developing GUI’s for users with disabilities
* The Desktop Colors Features Has been Firstly introduced in Java 1.1
* Java 2D: it has Improved colors, images, and also texts support

### ****Features Of Swing Class****

* Pluggable look and feel
* Uses MVC architecture
* Lightweight Components
* Platform Independent
* Advanced features such as JTable, JTabbedPane, JScollPane, etc.
* Java is a platform-independent language and runs on any client machine, the GUI look and feel, owned and delivered by a platform-specific O/S, simply does not affect an application’s GUI constructed using Swing components
* **Lightweight Components:** Starting with the JDK 1.1, its AWT-supported lightweight component development. For a component to qualify as lightweight, it must not depend on any non-Java [O/s based) system classes. Swing components have their own view supported by Java’s look and feel classes.
* **Pluggable Look and Feel:** This feature enables the user to switch the look and feel of Swing components without restarting an application. The Swing library supports components’ look and feels that remain the same across all platforms wherever the program runs. The Swing library provides an API that gives real flexibility in determining the look and feel of the GUI of an application
* **Highly customizable –**Swing controls can be customized in a very easy way as visual appearance is independent of internal representation.
* **Rich controls**– Swing provides a rich set of advanced controls like Tree TabbedPane, slider, colorpicker, and table controls.

**Swing Classes Hierarchy**



**5.1**

**5.1 JAVA**

### The MVC Connection

* In general, a visual component is a composite of**three distinct aspects:**
  1. The way that the component looks when rendered on the screen
  2. The way such that the component reacts to the user
  3. The state information associated With the component
* Over the years, one component architecture has proven itself to be exceptionally effective:- **Model-View-Controller** or **MVC** for short.
* In MVC terminology, the **model** corresponds to the state information associated with the Component
* The **view** determines how the component is displayed on the screen, including any aspects of the view that are affected by the current state of the model.
* The **controller** determines how the component reacts to the user

The simplest Swing components have capabilities far beyond AWT components as follows:

* Swing buttons and labels can be displaying images instead of or in addition to text
* The borders around most Swing components can be changed easily. For example, it is easy to put a 1 pixel border around the outside of a Swing label
* Swing components do not have to be rectangular.

## 5.3 SQL (Structured Query Language)

**SQL (Structured Query Language)** 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). In comparison to older read/write APIs like [SAM or VSAM, SQL offers two main advantages: first, it introduced the concept of accessing many records with one single command; and second, 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 a data definition language, data manipulation language, and data control language. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL became a standard of the American National Standards Institute (ANSI) in 1986 and of 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 such standards, most SQL code is not completely portable among different data base systems without adjustments.

## 5.4 Code Snippets

### 5.4.1 Connectivity to Database

**import** java.sql.\*;

**public** **class** Conn {

Connection c;

Statement s;

Conn() {

**try** {

c = DriverManager.*getConnection*("jdbc:mysql:///ebs", "root", "$Indsan0911");

s = c.createStatement();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

### 5.4.2 Insert a New Customer

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.util.\*;

**import** java.awt.event.\*;

**public** **class** NewCustomer **extends** JFrame **implements** ActionListener{

JTextField tfname, tfaddress, tfstate, tfcity, tfemail, tfphone;

JButton next, cancel;

JLabel lblmeter;

NewCustomer() {

setSize(700, 500);

setLocation(400, 200);

JPanel p = **new** JPanel();

p.setLayout(**null**);

p.setBackground(**new** Color(173, 216, 230));

add(p);

JLabel heading = **new** JLabel("New Customer");

heading.setBounds(180, 10, 200, 25);

heading.setFont(**new** Font("Tahoma", Font.***PLAIN***, 24));

p.add(heading);

JLabel lblname = **new** JLabel("Customer Name");

lblname.setBounds(100, 80, 100, 20);

p.add(lblname);

tfname = **new** JTextField();

tfname.setBounds(240, 80, 200, 20);

p.add(tfname);

JLabel lblmeterno = **new** JLabel("Meter Number");

lblmeterno.setBounds(100, 120, 100, 20);

p.add(lblmeterno);

lblmeter = **new** JLabel("");

lblmeter.setBounds(240, 120, 100, 20);

p.add(lblmeter);

Random ran = **new** Random();

**long** number = ran.nextLong() % 1000000;

lblmeter.setText("" + Math.*abs*(number));

JLabel lbladdress = **new** JLabel("Address");

lbladdress.setBounds(100, 160, 100, 20);

p.add(lbladdress);

tfaddress = **new** JTextField();

tfaddress.setBounds(240, 160, 200, 20);

p.add(tfaddress);

JLabel lblcity = **new** JLabel("City");

lblcity.setBounds(100, 200, 100, 20);

p.add(lblcity);

tfcity = **new** JTextField();

tfcity.setBounds(240, 200, 200, 20);

p.add(tfcity);

JLabel lblstate = **new** JLabel("State");

lblstate.setBounds(100, 240, 100, 20);

p.add(lblstate);

tfstate = **new** JTextField();

tfstate.setBounds(240, 240, 200, 20);

p.add(tfstate);

JLabel lblemail = **new** JLabel("Email");

lblemail.setBounds(100, 280, 100, 20);

p.add(lblemail);

tfemail = **new** JTextField();

tfemail.setBounds(240, 280, 200, 20);

p.add(tfemail);

JLabel lblphone = **new** JLabel("Phone Number");

lblphone.setBounds(100, 320, 100, 20);

p.add(lblphone);

tfphone = **new** JTextField();

tfphone.setBounds(240, 320, 200, 20);

p.add(tfphone);

next = **new** JButton("Next");

next.setBounds(120, 390, 100,25);

next.setBackground(Color.***BLACK***);

next.setForeground(Color.***WHITE***);

next.addActionListener(**this**);

p.add(next);

cancel = **new** JButton("Cancel");

cancel.setBounds(250, 390, 100,25);

cancel.setBackground(Color.***BLACK***);

cancel.setForeground(Color.***WHITE***);

cancel.addActionListener(**this**);

p.add(cancel);

setLayout(**new** BorderLayout());

add(p, "Center");

ImageIcon i1 = **new** ImageIcon(ClassLoader.*getSystemResource*("icon/hicon1.jpg"));

Image i2 = i1.getImage().getScaledInstance(150, 300, Image.***SCALE\_DEFAULT***);

ImageIcon i3 = **new** ImageIcon(i2);

JLabel image = **new** JLabel(i3);

add(image, "West");

getContentPane().setBackground(Color.***WHITE***);

setVisible(**true**);

}

**public** **void** actionPerformed(ActionEvent ae) {

**if** (ae.getSource() == next) {

String name = tfname.getText();

String meter = lblmeter.getText();

String address = tfaddress.getText();

String city = tfcity.getText();

String state = tfstate.getText();

String email = tfemail.getText();

String phone = tfphone.getText();

String query1 = "insert into customer values('"+name+"', '"+meter+"', '"+address+"', '"+city+"', '"+state+"', '"+email+"', '"+phone+"')";

String query2 = "insert into login values('"+meter+"', '', '"+name+"', '', '')";

**try** {

Conn c = **new** Conn();

c.s.executeUpdate(query1);

c.s.executeUpdate(query2);

JOptionPane.*showMessageDialog*(**null**, "Customer Details Added Successfully");

setVisible(**false**);

// new frame

**new** MeterInfo(meter);

} **catch** (Exception e) {

e.printStackTrace();

}

} **else** {

setVisible(**false**);

}

}

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

**new** NewCustomer();

}

}

**5.4.3 View the contents of Students**

import javax.swing.\*;

import java.awt.\*;

import java.sql.\*;

import java.awt.event.\*;

public class ViewInformation extends JFrame implements ActionListener{

JButton cancel;

ViewInformation(String meter) {

setBounds(350, 150, 850, 650);

getContentPane().setBackground(Color.*WHITE*);

setLayout(null);

JLabel heading = new JLabel("VIEW CUSTOMER INFORMATION");

heading.setBounds(250, 0, 500, 40);

heading.setFont(new Font("Tahoma", Font.*PLAIN*, 20));

add(heading);

JLabel lblname = new JLabel("Name");

lblname.setBounds(70, 80, 100, 20);

add(lblname);

JLabel name = new JLabel("");

name.setBounds(250, 80, 100, 20);

add(name);

JLabel lblmeternumber = new JLabel("Meter Number");

lblmeternumber.setBounds(70, 140, 100, 20);

add(lblmeternumber);

JLabel meternumber = new JLabel("");

meternumber.setBounds(250, 140, 100, 20);

add(meternumber);

JLabel lbladdress = new JLabel("Address");

lbladdress.setBounds(70, 200, 100, 20);

add(lbladdress);

JLabel address = new JLabel("");

address.setBounds(250, 200, 100, 20);

add(address);

JLabel lblcity = new JLabel("City");

lblcity.setBounds(70, 260, 100, 20);

add(lblcity);

JLabel city = new JLabel("");

city.setBounds(250, 260, 100, 20);

add(city);

JLabel lblstate = new JLabel("State");

lblstate.setBounds(500, 80, 100, 20);

add(lblstate);

JLabel state = new JLabel("");

state.setBounds(650, 80, 100, 20);

add(state);

JLabel lblemail = new JLabel("Email");

lblemail.setBounds(500, 140, 100, 20);

add(lblemail);

JLabel email = new JLabel("");

email.setBounds(650, 140, 100, 20);

add(email);

JLabel lblphone = new JLabel("Phone");

lblphone.setBounds(500, 200, 100, 20);

add(lblphone);

JLabel phone = new JLabel("");

phone.setBounds(650, 200, 100, 20);

add(phone);

try {

Conn c = new Conn();

ResultSet rs = c.s.executeQuery("select \* from customer where meter\_no = '"+meter+"'");

while(rs.next()) {

name.setText(rs.getString("name"));

address.setText(rs.getString("address"));

city.setText(rs.getString("city"));

state.setText(rs.getString("state"));

email.setText(rs.getString("email"));

phone.setText(rs.getString("phone"));

meternumber.setText(rs.getString("meter\_no"));

}

} catch (Exception e) {

e.printStackTrace();

}

cancel = new JButton("Cancel");

cancel.setBackground(Color.*BLACK*);

cancel.setForeground(Color.*WHITE*);

cancel.setBounds(350, 340, 100, 25);

add(cancel);

cancel.addActionListener(this);

ImageIcon i1 = new ImageIcon(ClassLoader.*getSystemResource*("icon/viewcustomer.jpg"));

Image i2 = i1.getImage().getScaledInstance(600, 300, Image.*SCALE\_DEFAULT*);

ImageIcon i3 = new ImageIcon(i2);

JLabel image = new JLabel(i3);

image.setBounds(20, 350, 600, 300);

add(image);

setVisible(true);

}

public void actionPerformed(ActionEvent ae) {

setVisible(false);

}

public static void main(String[] args) {

new ViewInformation("");

}

}

### 5.4.4 Admin and Customer Login

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** java.sql.\*;

**public** **class** Login **extends** JFrame **implements** ActionListener{

JButton login, cancel, signup;

JTextField username, password;

Choice logginin;

Login() {

**super**("Login Page");

getContentPane().setBackground(Color.***WHITE***);

setLayout(**null**);

JLabel lblusername = **new** JLabel("Username");

lblusername.setBounds(300, 20, 100, 20);

add(lblusername);

username = **new** JTextField();

username.setBounds(400, 20, 150, 20);

add(username);

JLabel lblpassword = **new** JLabel("Password");

lblpassword.setBounds(300, 60, 100, 20);

add(lblpassword);

password = **new** JTextField();

password.setBounds(400, 60, 150, 20);

add(password);

JLabel loggininas = **new** JLabel("Loggin in as");

loggininas.setBounds(300, 100, 100, 20);

add(loggininas);

logginin = **new** Choice();

logginin.add("Admin");

logginin.add("Customer");

logginin.setBounds(400, 100, 150, 20);

add(logginin);

ImageIcon i1 = **new** ImageIcon(ClassLoader.*getSystemResource*("icon/login.png"));

Image i2 = i1.getImage().getScaledInstance(16, 16, Image.***SCALE\_DEFAULT***);

login = **new** JButton("Login", **new** ImageIcon(i2));

login.setBounds(330, 160, 100, 20);

login.addActionListener(**this**);

add(login);

ImageIcon i3 = **new** ImageIcon(ClassLoader.*getSystemResource*("icon/cancel.jpg"));

Image i4 = i3.getImage().getScaledInstance(16, 16, Image.***SCALE\_DEFAULT***);

cancel = **new** JButton("Cancel", **new** ImageIcon(i4));

cancel.setBounds(450, 160, 100, 20);

cancel.addActionListener(**this**);

add(cancel);

ImageIcon i5 = **new** ImageIcon(ClassLoader.*getSystemResource*("icon/signup.png"));

Image i6 = i5.getImage().getScaledInstance(16, 16, Image.***SCALE\_DEFAULT***);

signup = **new** JButton("Signup", **new** ImageIcon(i6));

signup.setBounds(380, 200, 100, 20);

signup.addActionListener(**this**);

add(signup);

ImageIcon i7 = **new** ImageIcon(ClassLoader.*getSystemResource*("icon/second.jpg"));

Image i8 = i7.getImage().getScaledInstance(250, 250, Image.***SCALE\_DEFAULT***);

ImageIcon i9 = **new** ImageIcon(i8);

JLabel image = **new** JLabel(i9);

image.setBounds(0, 0, 250, 250);

add(image);

setSize(640, 300);

setLocation(400, 200);

setVisible(**true**);

}

**public** **void** actionPerformed(ActionEvent ae) {

**if** (ae.getSource() == login) {

String susername = username.getText();

String spassword = password.getText();

String user = logginin.getSelectedItem();

**try** {

Conn c = **new** Conn();

String query = "select \* from login where username = '"+susername+"' and password = '"+spassword+"' and user = '"+user+"'";

ResultSet rs = c.s.executeQuery(query);

**if** (rs.next()) {

String meter = rs.getString("meter\_no");

setVisible(**false**);

**new** Project(user, meter);

} **else** {

JOptionPane.*showMessageDialog*(**null**, "Invalid Login");

username.setText("");

password.setText("");

}

} **catch** (Exception e) {

e.printStackTrace();

}

} **else** **if** (ae.getSource() == cancel) {

setVisible(**false**);

} **else** **if** (ae.getSource() == signup) {

setVisible(**false**);

**new** Signup();

}

}

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

**new** Login();

}

}

**5.4.5 Bill Details of Customer**

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.sql.\*;

**import** net.proteanit.sql.DbUtils;

**public** **class** BillDetails **extends** JFrame{

BillDetails(String meter) {

setSize(700, 650);

setLocation(400, 150);

getContentPane().setBackground(Color.***WHITE***);

JTable table = **new** JTable();

**try** {

Conn c = **new** Conn();

String query = "select \* from bill where meter\_no = '"+meter+"'";

ResultSet rs = c.s.executeQuery(query);

table.setModel(DbUtils.*resultSetToTableModel*(rs));

} **catch** (Exception e) {

e.printStackTrace();

}

JScrollPane sp = **new** JScrollPane(table);

sp.setBounds(0, 0, 700, 650);

add(sp);

setVisible(**true**);

}

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

**new** BillDetails("");

}

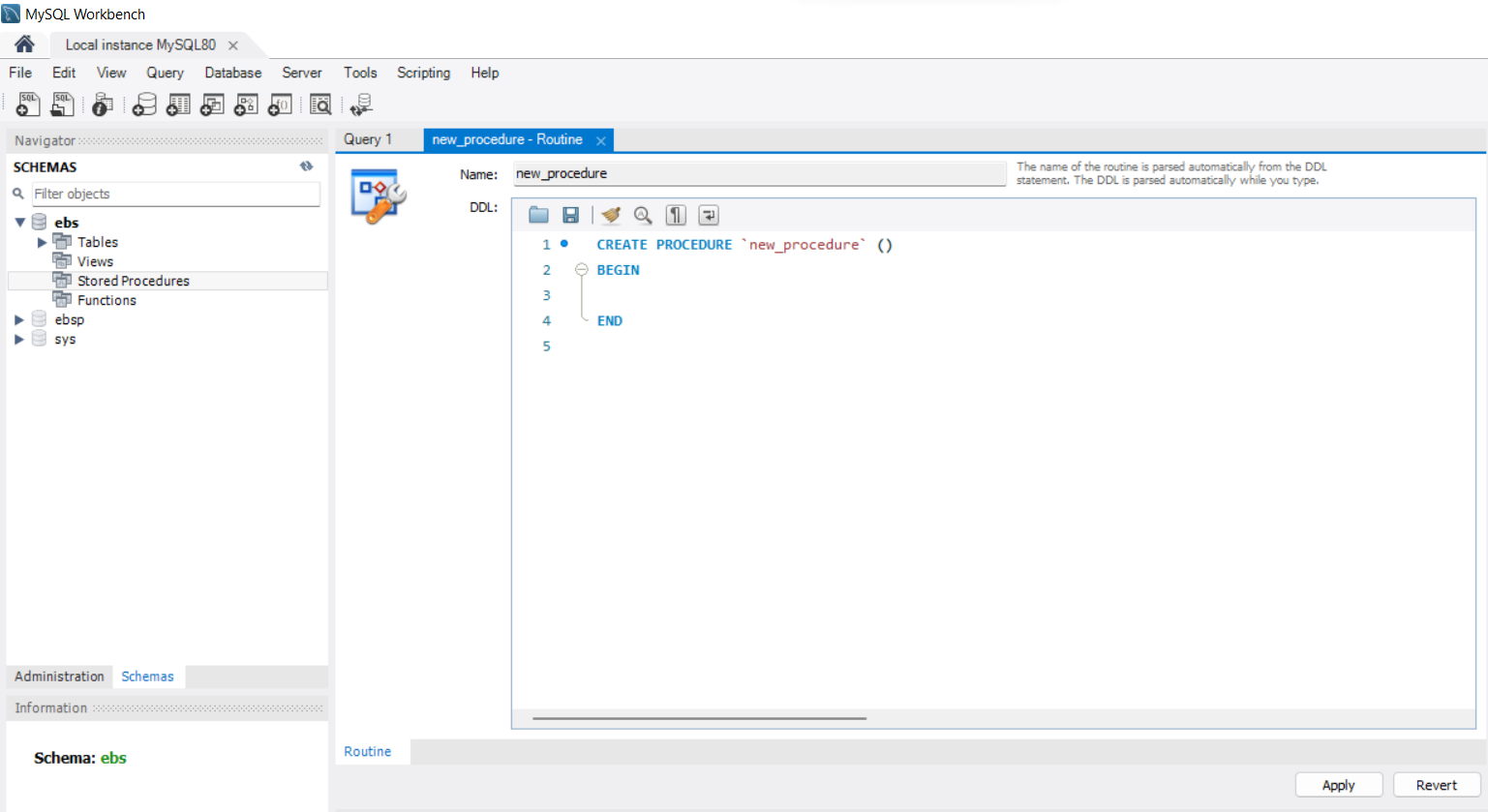
}

## 5.5 Stored Procedure

A procedure in SQL (often referred to as stored procedure), is a reusable unit that encapsulates

the specific business logic of the application. A SQL procedure is a group of SQL statements

and logic, compiled and stored together to perform a specific task



* 1. Stored Procedure of Electricity Billing System

## 5.6 Trigger

**Trigger** is a statement that a system executes automatically when there is any modification to

the database. In a trigger, we first specify when the trigger is to be executed and then the action to be performed when the trigger executes. Triggers are used to specify certain integrity and referential constraints that cannot be specified using the constraint mechanism of SQL.

## 

* 1. Trigger of Electricity Billing System

**Chapter 6**

**SNAPSHOTS**

## 6.1 The Login page

#### This is the admin’s and customer’s login page. All the users have access to this page. Here the admin or the customer have to first sign up and then login in into the system. After doing this their account will be created.



Fig no .6.1 Login Page of Electricity Billing System

## 6.2 The Account Creation Page

## This is the account creation page. The admin or the customer are directed to this page after signing up. After the filling the required details the admin or the customer’s account will be successfully we created and will be added into the database.

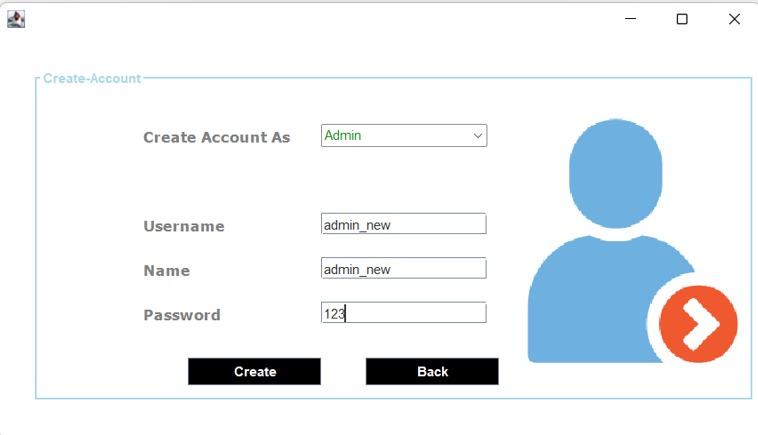
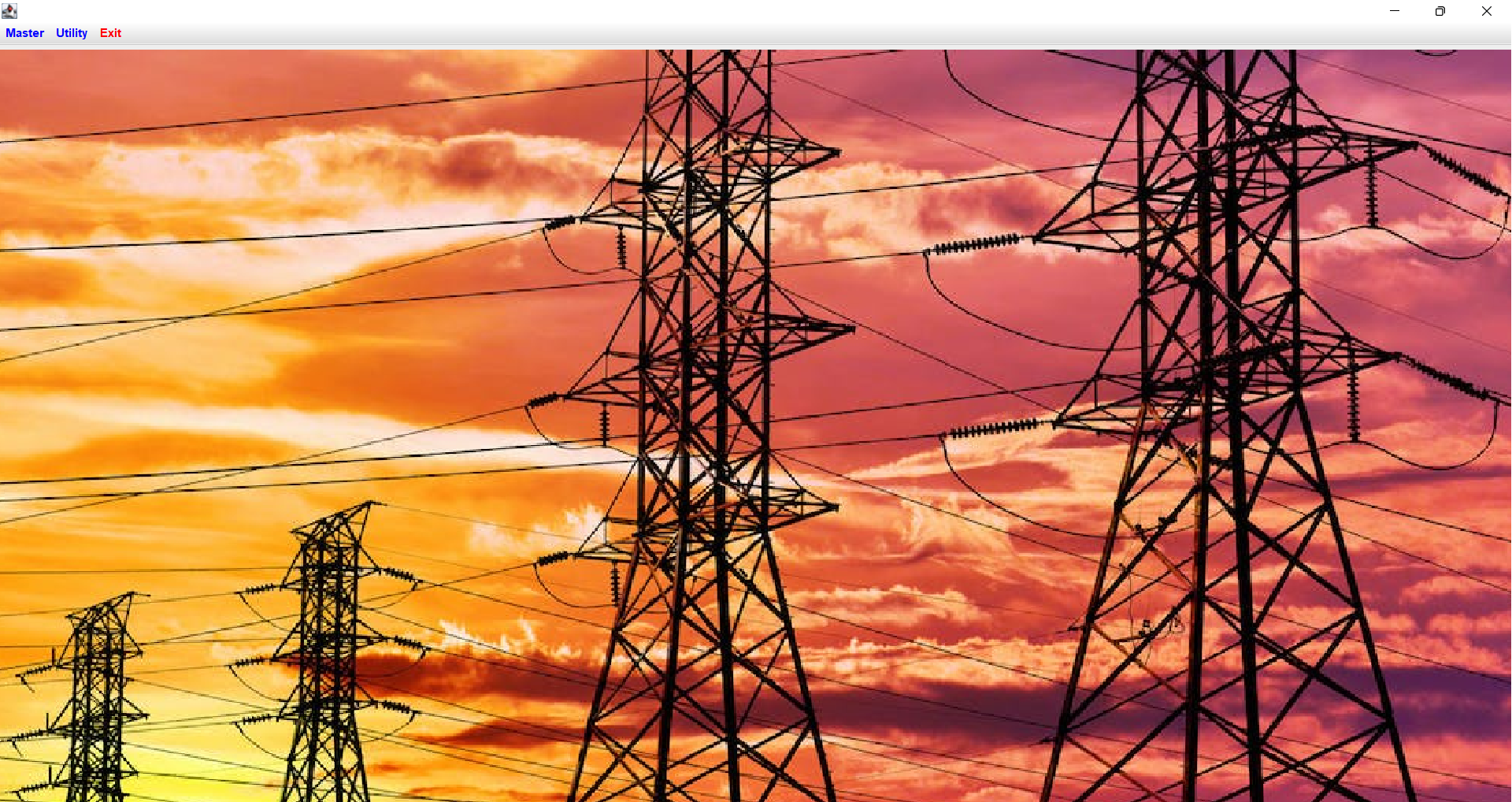
****

Fig no. 6.2 Account Creation Page of Electricity Billing System

## 6.3 Admin Home Page

#### Once the admin sign’s up, he’ll be directed to the admin’s home page. Only he has the access to this home page. He can create a new customer, view the customer’s details, view deposit details and calculate the bill of the customer.



## Fig no. 6.3 Admin’s Home Page of Electricity Billing System

## 6.4 New Customer Insertion Page

## Here the admin can create a new customer’s account by filling all the required details of the customer. After this the customer’s account will be successfully created.

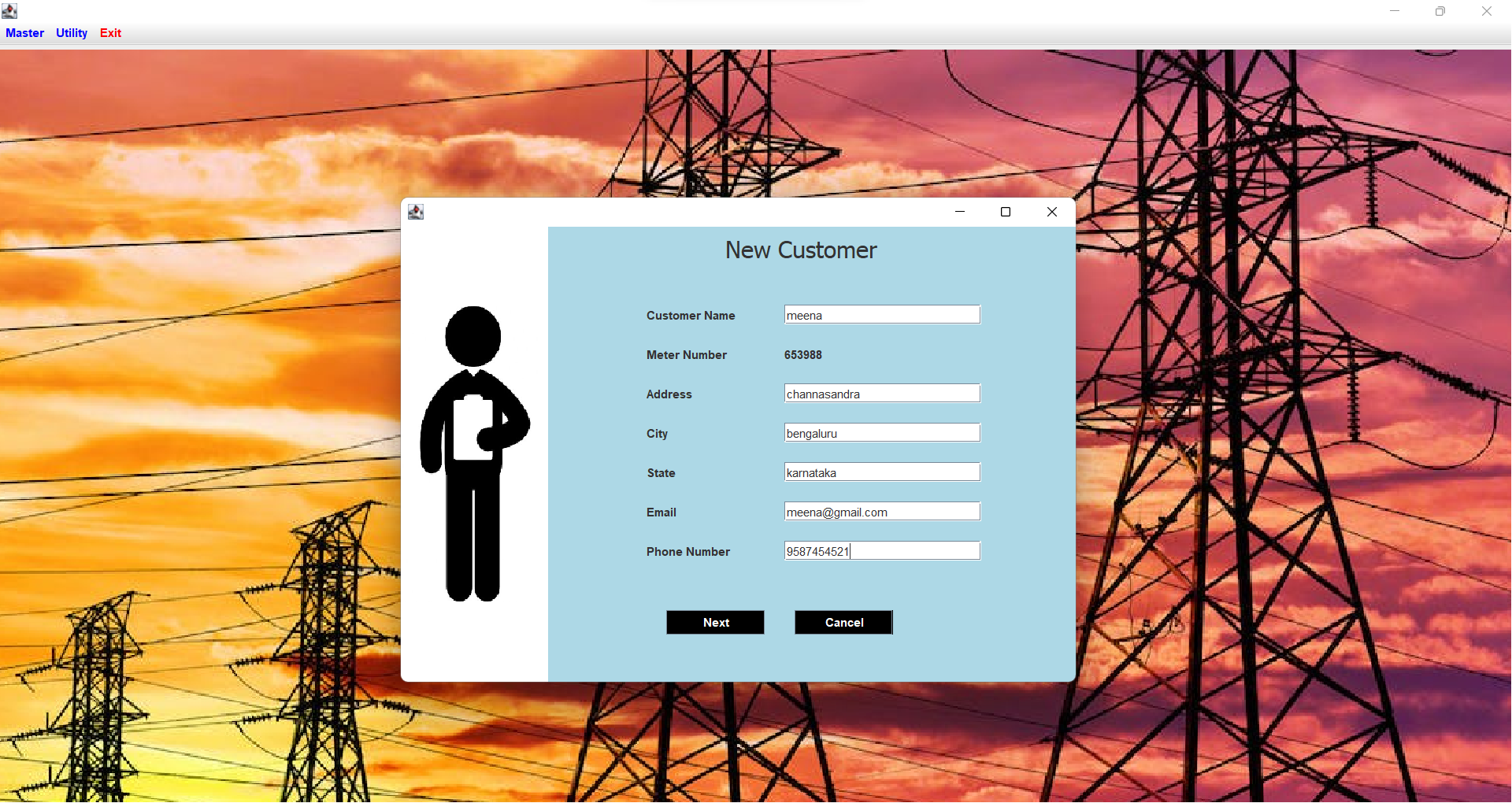
****

Fig no. 6.4 New Customer Creation Page of Electricity Billing System

**6.5 Meter Information Page**

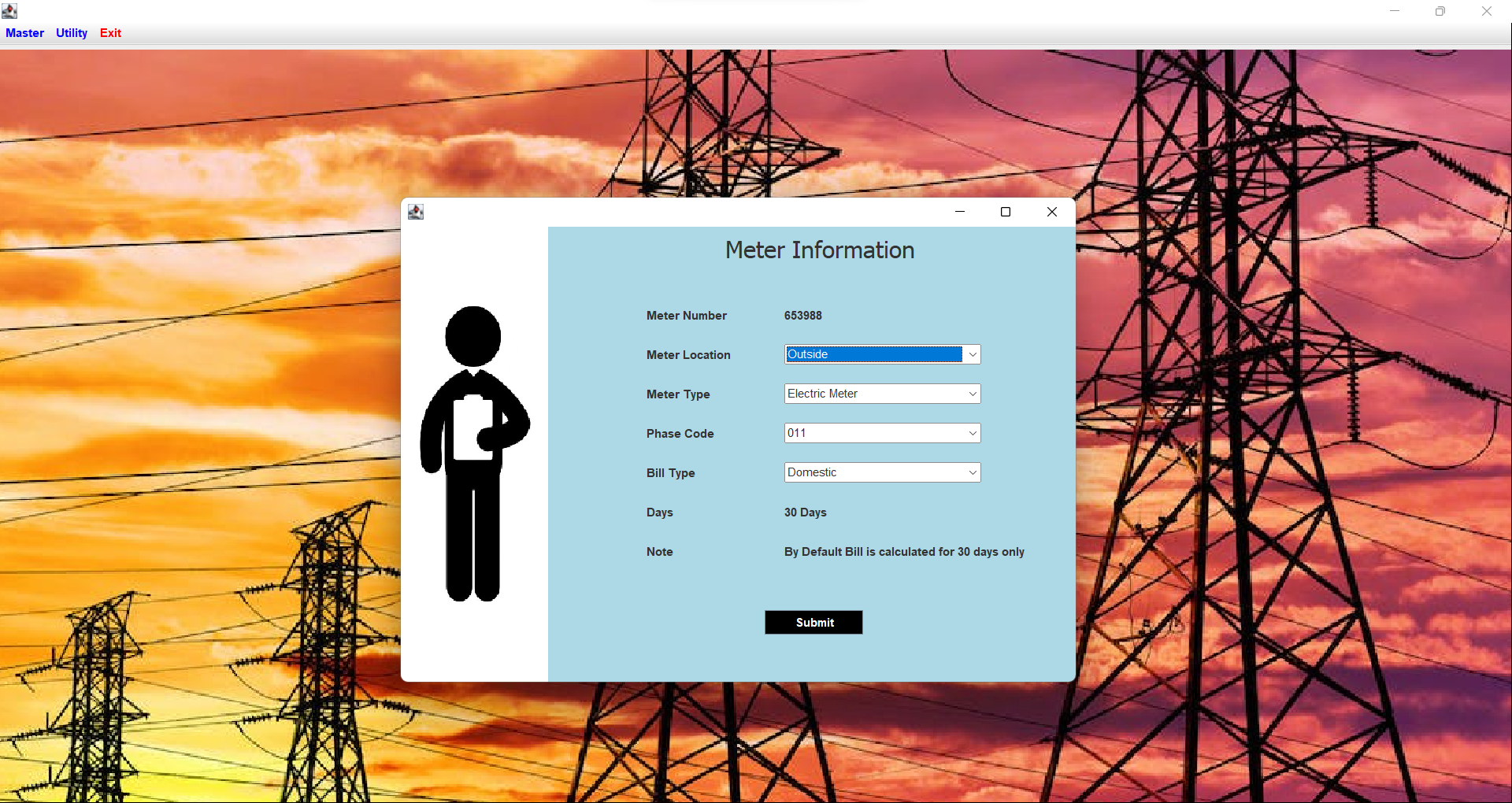
Everytime a customer’s account is created, a unique meter number is given to the customer. The customer should provide their meter location, meter type, phase code and bill type to the admin 

Fig no. 6.5 Meter information Page of Electricity Billing System

## 6.6 Customers Details Page

## The admin can see the customer’s details in this page.

## 

## Fig no. 6.6 Customer’s Details of Electricity Billing System

## 6.7 Calculation of Bill

## The bill of the customer is calculated by the admin by providing the admin with the number of units used by the customer.

## 

## Fig no. 6.7 Bill Calculation of Electricity Billing System

## 6.8 Customer Home Page

## Once the customer login’s in, he is directed to the customer home page. Here he can upadate his information, view his bill status, pay his bill and also generate his electricity bill.

## 

## Fig no. 6.8 Customer’s Home page of Electricity Billing system

## 6.9 Bill Payment Status Page

## The customer can know about their bill status and pay the bill online.

## 

## Fig no. 6.9 Bill Payment Status of Electricity Billing System

## 6.10 Bill Generation Page

## The customer gets their electricity bill after the payment.

## 

## Fig no. 6.10 Electricity Bill of Electricity Billing System

**6.11 Table Structure**

### 6.6.1 Login Table

### Table name : Login

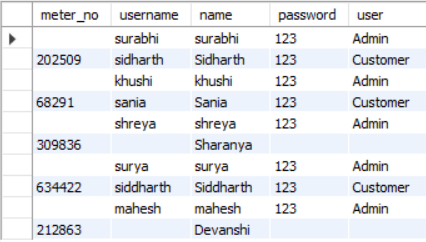
****

Table 6.11.1 shows the login page which has five attributes: meter\_no, username, name, password and user.

Here meter\_no is the primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Key** |
| Meter\_no | Interger | 20 | Primary |
| Username | Varchar | 30 |  |
| Name | Varchar | 30 |  |
| Password | Varchar | 20 |  |
| User | Varchar | 20 |  |

**6.11.2 Customer Table**

**Table name : Customer**

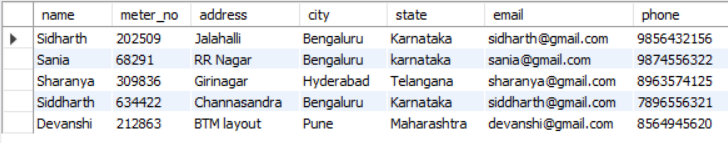
****

Table 6.11.2 shows Customer table which has seveb attributes : name, meter\_no, address, city, state, email, phone.

Here meter\_no is the primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Key** |
| Name | Varchar | 20 |  |
| Meter\_no | interger | 20 | Primary |
| Address | Varchar | 50 |  |
| City | Varchar | 30 |  |
| Email | Varchar | 30 |  |
| Phone | Integer | 10 |  |

**6.11.3 Meter\_info Table**

**Table name : meter\_info**

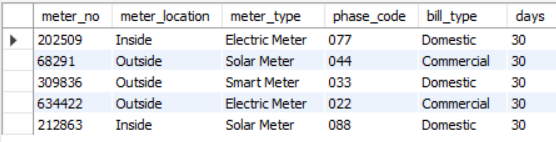
****

Table 6.11.2 shows the meter\_info table which has six attributes : meter\_no, meter\_location, meter\_type, phase\_code, bill\_type and days.

Here meter\_no is the primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Key** |
| Meter\_no | Integer | 20 | Primary |
| Meter\_location | Varchar | 20 |  |
| Meter\_type | Varchar | 20 |  |
| Phase\_code | Varchar | 20 |  |
| Bill\_type | Varchar | 20 |  |
| Days | Interger | 20 |  |

**6.11.4 Tax Table**

**Table Name : Tax**

****

Table 6.11.4 shows the Tax table which has six attributes : cost\_per\_unit, meter\_rent, service\_charge, service\_tax, swacch\_bharat\_cess, fixed\_tax.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Key** |
| Cost\_per\_unit | Integer | 20 |  |
| Meter\_rent | Integer | 20 |  |
| Service\_charge | Integer | 20 |  |
| Service\_tax | Integer | 20 |  |
| Swacch\_bharat\_cess | Integer | 20 |  |
| Fixed\_tax | Interger | 20 |  |

**6.11.5 Bill Table**

**Table name : Bill**

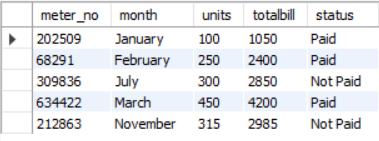
****

Table 6.11.5 shows the Bill table which has five attributes : meter\_no, month, units, totalbill, status.

Here meter\_no is the primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Key** |
| Meter\_no | Integer | 20 | Primary |
| Month | Varchar | 20 |  |
| Units | Integer | 20 |  |
| Totalbill | Integer | 20 |  |
| Status | Varchar | 20 |  |

## Chapter 7

**CONCLUSION**

After all the hard work is done for electricity bill management system is here. It is a software which helps the user to work with the billing cycles, paying bills, managing different details under which are working etc.

This software reduces the amount of manual data entry and gives greater efficiency. The User Interface of it is very friendly and can be easily used by anyone.

It also decreases the amount of time taken to write details and other modules.

**FUTURE ENHANCEMENTS**

* Our project is a fairly simple Database consisting of a few fields, adding more information and being able to manage more student information with added functionality.
* Hosting the website on an online web server.
* Payment verification through an authorized client can also be implemented.
* All in all, making it a more robust system with added functionalities to enhance the overall project features.

.

**REFERENCES**

1. Fundamentals of Database Systems, RamezElmasri and Shamkant B. Navathe, 7th Edition,2017,Pearson.
2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill.
3. W3 Schools (JAVA SWINGS reference) –https:/[/www.w3schools.com/](http://www.w3schools.com/)html
4. Geeksforgeeks –https://www.geeksforgeeks.org/introduction-to-java-swing/
5. The MySQL Documentation –https://dev.mysql.com/doc/
6. Stack Overflow –https://stackoverflow.com/
7. Wikipedia –https:/[/www.wikipedia.org/](http://www.wikipedia.org/)