Project Report

ON

SHOPPING MALL SERVICE DESK SIMULATOR USING JAVA (NetBeansIDE) AND MySQL

By: - Hemant Gupta

Hemant

Signature

TABLE OF CONTENTS

TITLE	Page No.

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Project Name: Shopping Mall Service Desk Simulator

Project Duration: 30 Days

Project Detail: It is a sample application designed individually in order to simulate major functioning of shopping mall service desk software in a java desktop application with help of core java programming using Net Beans IDE and back end database connectivity with MySQL. Under this topic we had gone through a practical development of JAVA Desktop Application with MySQL connectivity in backend using JAVA programming language and J2EE Architecture. This Project in JAVA Language of Shopping Mall Simulator is a simple application with very simple graphics developed with NetBeans IDE and MySQL. In this project you can go through all basic operation which a shopping mall go through in its business cycle just like you feel it elsewhere like in reliance digital or any other shopping mall.

This Project will aim on understanding the functioning of software used by service desk employee in shopping mall to add, delete and update customer and products information in databases.

CHAPTER 2: JAVA

Java is a general-purpose, concurrent, object-oriented, class-based, and the runtime environment (JRE) which consists of **JVM** which is the cornerstone of the Java platform.

Java has been used in different domains. Some of them are listed below:

- Banking: To deal with transaction management.
- Retail: Billing applications that you see in a store/restaurant are completely written in Java.
- Information Technology: Java is designed to solve implementation dependencies.
- Android: Applications are either written in Java or use Java API.
- Financial services: It is used in server-side applications.
- Stock market: To write algorithms as to which company they should invest in.
- Big Data: Hadoop MapReduce framework is written using Java.
- Scientific and Research Community: To deal with huge amount of data.

2.1 HISTORY

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. There are given significant points that describe the history of Java.

2.2 J2EE

This was originally known as Java 2 Platform, Enterprise Edition, which was later changed to Java Platform, Enterprise Edition (Java EE). This is one of the three computing platforms released by Sun Microsystems which was later acquired by Oracle Corporation. The other two platforms are Java Standard Edition (Java SE) and Java Micro Edition (Java ME). J2EE is nothing but an extension of the Java SE based on the Java programming language used for developing and deploying web-based enterprise applications. It consists of a set of APIs, services, and protocols that provide the functionality to develop multi-tiered web-based applications. It includes several technologies that extend the functionality of the Java SE APIs, such as Servlets, Connectors, Enterprise JavaBeans, etc.

It's mainly used for applications which run on servers and accessible through browsers like Chrome, Firefox, etc. It's also used for developing web applications over World Wide Web by creating standardized modular components to handle many aspects of programming. The J2EE architecture provides services to simplify the common challenges faced by developers while developing modern applications, thereby making it easier to implement industry-standard design patterns for greater efficiency and reliability.

The platform was known as *Java 2 Platform, Enterprise Edition* or *J2EE* from version 1.2, until the name was changed to *Java Platform, Enterprise Edition* or *Java EE* in version 1.5.

- J2EE 1.2 (December 12, 1999)
- J2EE 1.3 (September 24, 2001)
- J2EE 1.4 (November 11, 2003)
- Java EE 5 (May 11, 2006)
- Java EE 6 (December 10, 2009)
- Java EE 7 (May 28, 2013, but April 5, 2013 according to spec document. June 12, 2013 was the planned kickoff date)
- Java EE 8 (August 31, 2017)

2.3 JDBC

JDBC (Java Database Connectivity) is the Java API that manages connecting to a database, issuing queries and commands, and handling result sets obtained from the database. Released as part of JDK 1.1 in 1997, JDBC was one of the first components developed for the Java persistence layer.

JDBC was initially conceived as a client-side API, enabling a Java client to interact with a data source. That changed with JDCB 2.0, which included an optional package supporting server-side JDBC connections. Every new JDBC release since then has featured updates to both the client-side package (java.sql) and the server-side package (javax.sql). JDBC 4.3, the most current version as of this writing, was released as part of Java SE 9 in September 2017.

This article presents an overview of JDBC, followed by a hands-on introduction to using the JDBC API to connect a Java client with SQLite, a lightweight relational database.

How JDBC works

Developed as an alternative to the C-based ODBC (Open Database Connectivity) API, JDBC offers a programming-level interface that handles the mechanics of Java applications communicating with a database or RDBMS. The JDBC interface consists of two layers:

- The JDBC API supports communication between the Java application and the JDBC manager.
- The JDBC driver supports communication between the JDBC manager and the database driver.

JDBC is the common API that your application code interacts with. Beneath that is the JDBC-compliant driver for the database you are using.

CHAPTER 3: MySQL HISTORY

MySQL is an open-source relational database management system (RDBMS). MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, or a proprietary license.

Support can be obtained from the official manual. Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and in price. Additionally, a number of third party organisations exist to provide support and services, including MariaDB and Percona.

MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server".

MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from mSQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as mSQL. By keeping the API consistent with the mSQL system, many developers were able to use MySQL instead of the (proprietarily licensed) mSQL antecedent.

CHAPTER 4: OBJECTIVE

To learn JAVA application development and database connectivity with MySQL with the application of computer graphics, developing a desktop application can always be a better and smarter option. To have an idea of what we can achieve with relatively simple OOPs concepts and JAVA(j2ee) and perhaps the basis by which to extend the principles and create more interesting applications of our own.

CHAPTER 5: SYSTEM REQUIREMENT ANALYSIS

5.1 SOFTWARE REQUIREMENTS:-

Operating system: WINDOWS

Application software: NetBeans IDE & MySQL Command

LineLanguage: JAVA and Sql

5.2 HARDWARE REQUIREMENTS:-

Hard Disk: - 32 GB

RAM: - 128 MB

Processor: - Any Pentium version

CHAPTER 6: USE CASE, CLASS & SEQUENCE DIAGRAM

USE CASE DIAGRAM

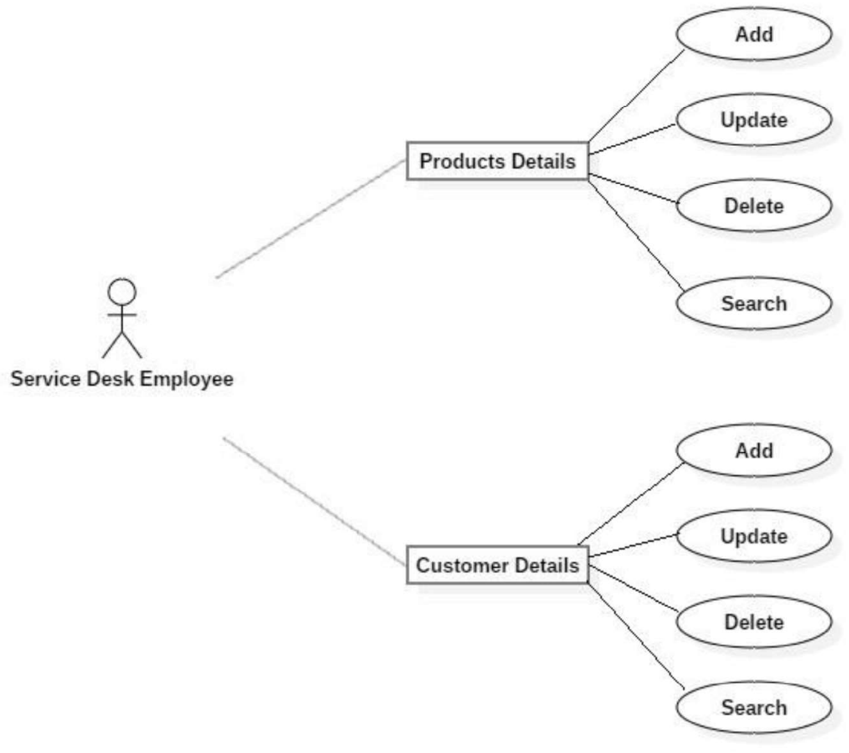


Figure 6.1 USE-CASE DIAGRAM

CLASS DIAGRAM

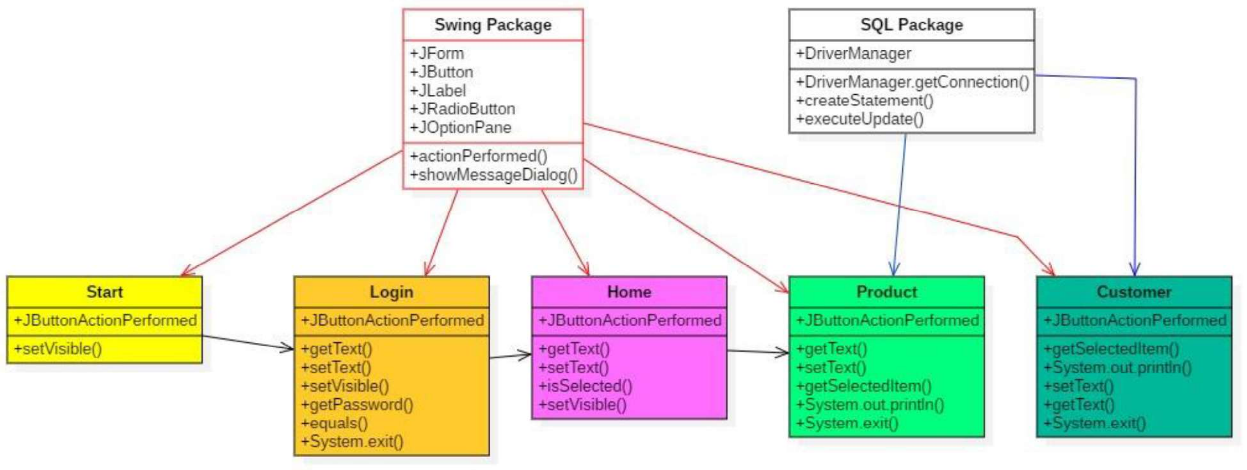


Figure 6.2 CLASS DIAGRAM

SEQUENCE DIAGRAM

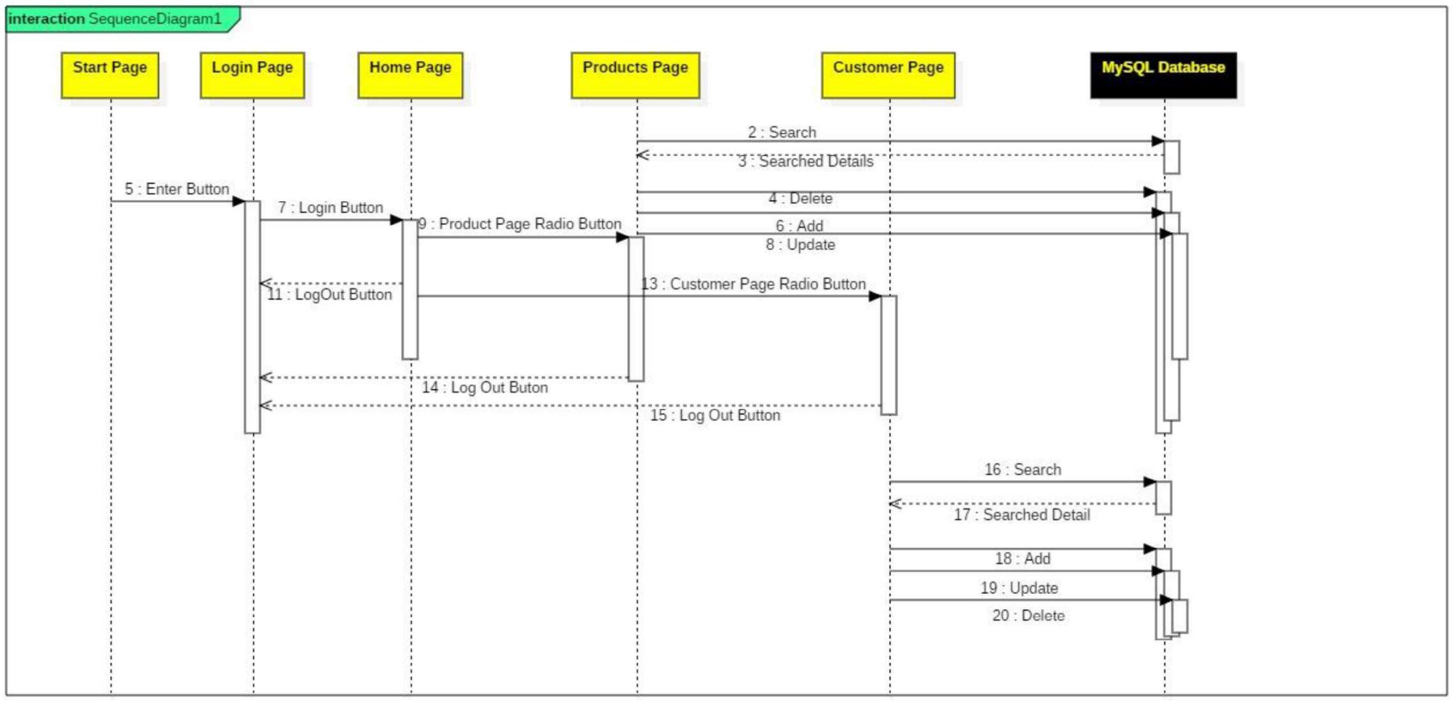
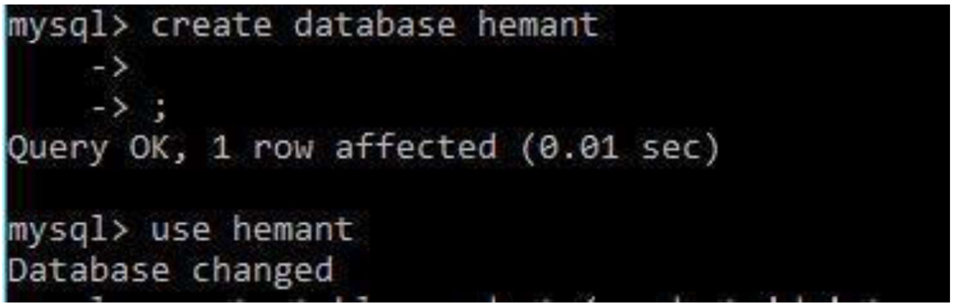


Figure 6.3 SEQUENCE DIAGRAM

CHAPTER 7: MySQL DATABASE

Database Name: - hemant



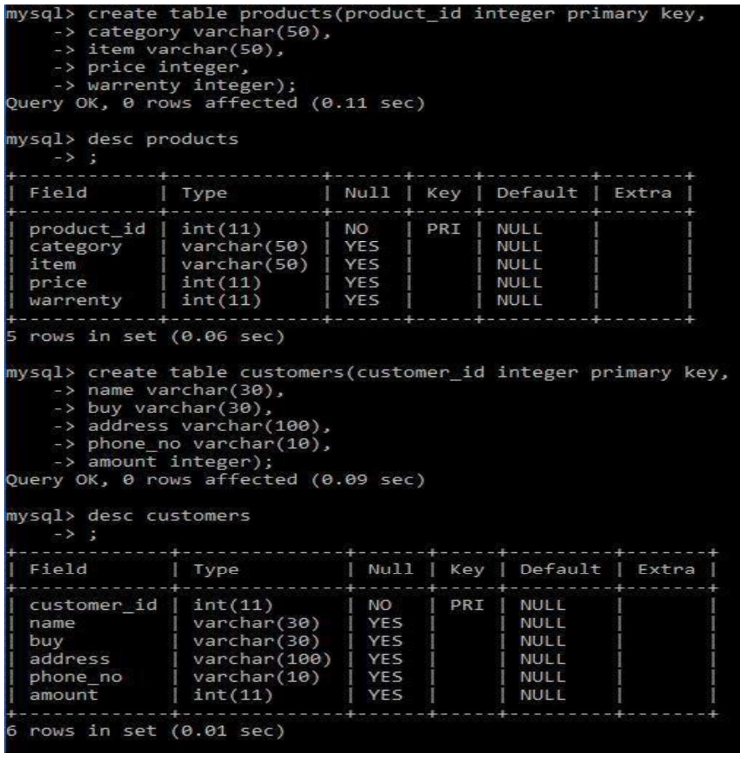
```
mysql> create database hemant
    ->
    -> ;
Query OK, 1 row affected (0.01 sec)

mysql> use hemant
Database changed
```

Figure 7.1 DATABASE CREATION

CHAPTER 8: MySQL TABLES

Table Name:-

- products
- customers



```
mysql> create table products(product_id integer primary key,
    -> category varchar(50),
    -> item varchar(50),
    -> price integer,
    -> warrenty integer);
Query OK, 0 rows affected (0.11 sec)

mysql> desc products
    -> ;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| product_id | int(11)     | NO   | PRI | NULL    |       |
| category   | varchar(50) | YES  |     | NULL    |       |
| item       | varchar(50) | YES  |     | NULL    |       |
| price      | int(11)     | YES  |     | NULL    |       |
| warrenty   | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.06 sec)

mysql> create table customers(customer_id integer primary key,
    -> name varchar(30),
    -> buy varchar(30),
    -> address varchar(100),
    -> phone_no varchar(10),
    -> amount integer);
Query OK, 0 rows affected (0.09 sec)

mysql> desc customers
    -> ;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| customer_id | int(11)      | NO   | PRI | NULL    |       |
| name        | varchar(30)  | YES  |     | NULL    |       |
| buy         | varchar(30)  | YES  |     | NULL    |       |
| address     | varchar(100) | YES  |     | NULL    |       |
| phone_no    | varchar(10)  | YES  |     | NULL    |       |
| amount      | int(11)      | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

Figure 8.1 TABLE CREATION

CHAPTER 9: NET BEANS IDE CODING

9.1 Start.java

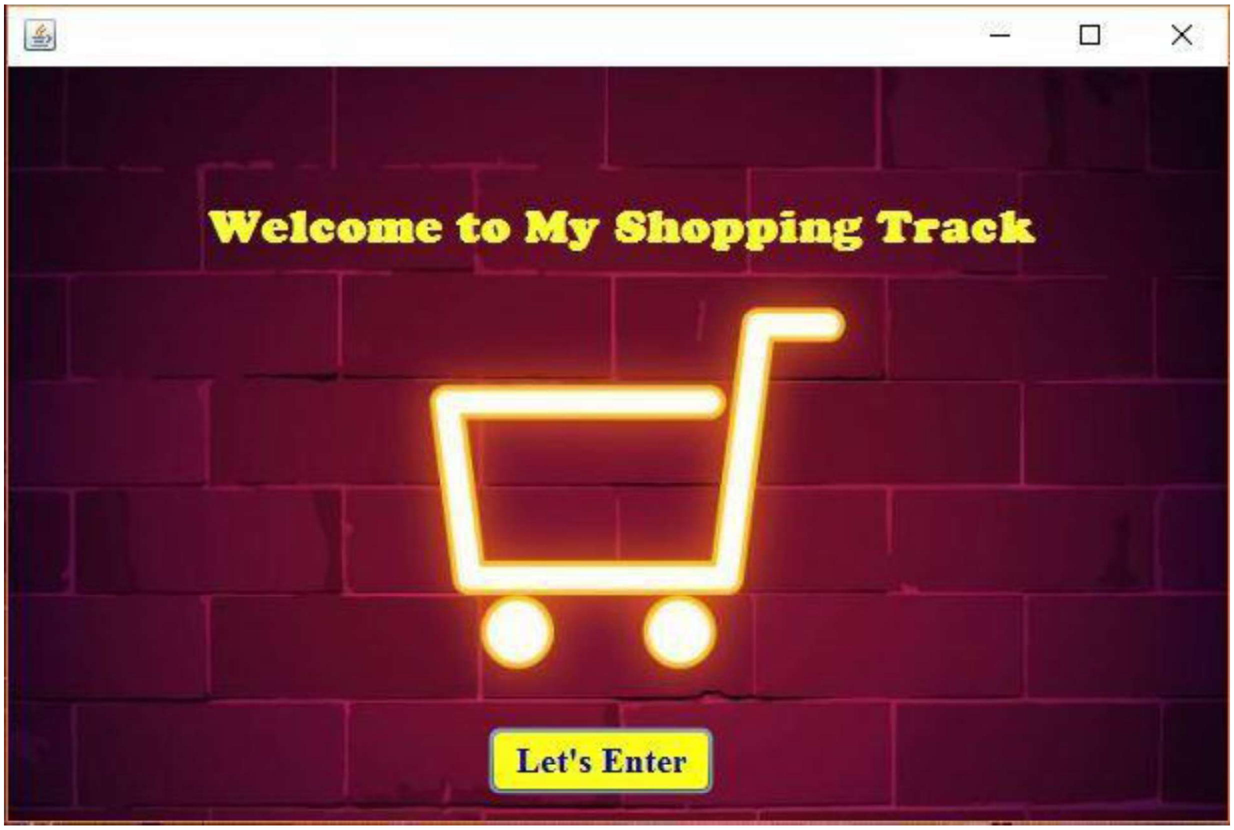


Figure 9.1.1 WELCOME PAGE

i) Let's Enter Button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        this.setVisible(false);

        new login().setVisible(true); }
```

9.2 login.java

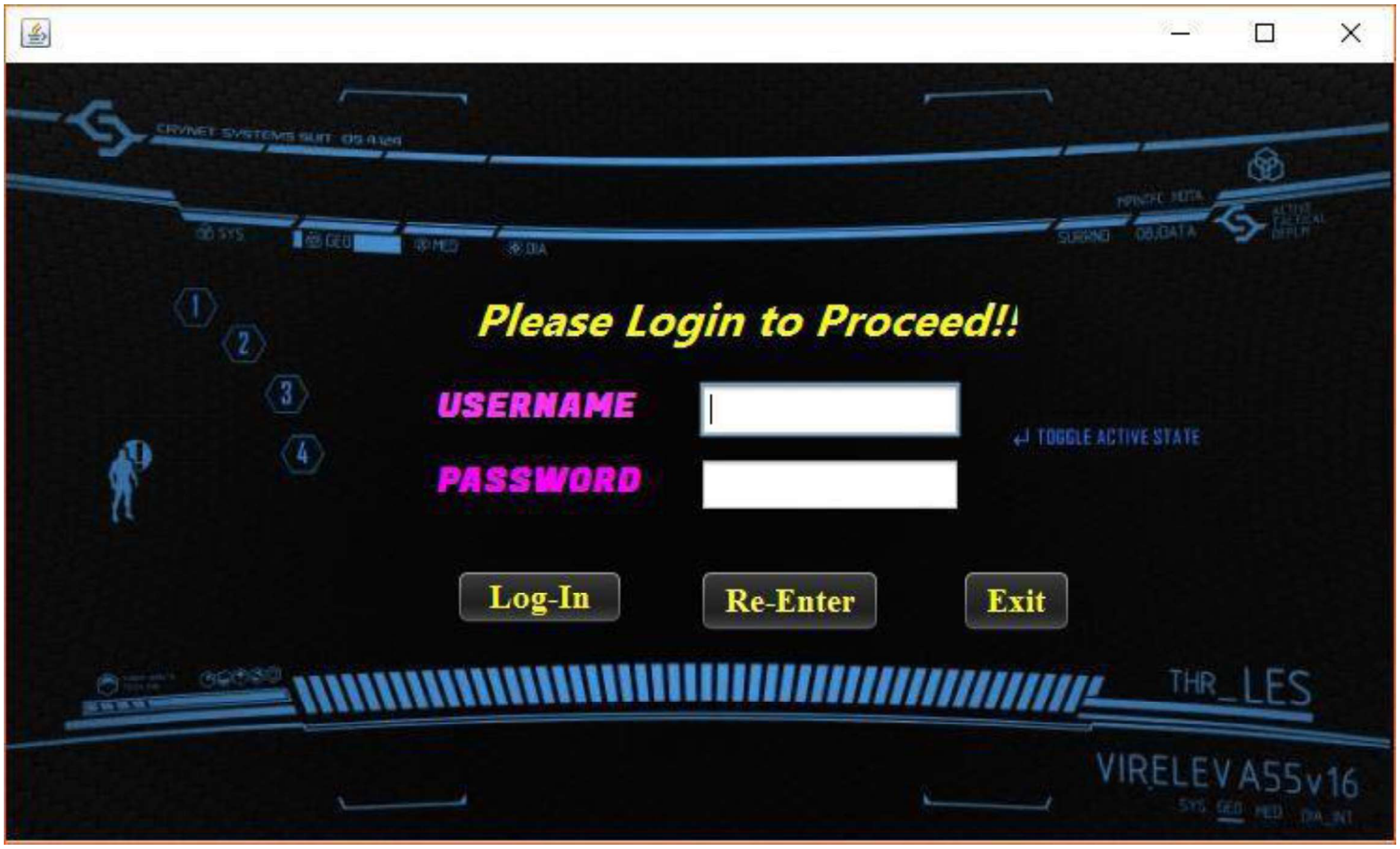


Figure 9.2.1 LOGIN PAGE

i) Log-In button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    String u=jTextField1.getText();
```

```
        String p= new String(jPasswordField1.getPassword());
      if  (u.equals("hemant") && p.equals("hemant"))
      {  new home().setVisible(true);
          this.setVisible(false);


      }
      else{
            JOptionPane.showMessageDialog(null," incorrect username or password");
      }
    }
```

ii) Re-Enter button

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

```
        // TODO add your handling code here:
        jTextField1.setText(" ");
        jPasswordField1.setText(" ");
```

}

iii) Exit Button:-

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

```
// TODO add your handling code here:
 System.exit(0);
```

}

9.3 home.java

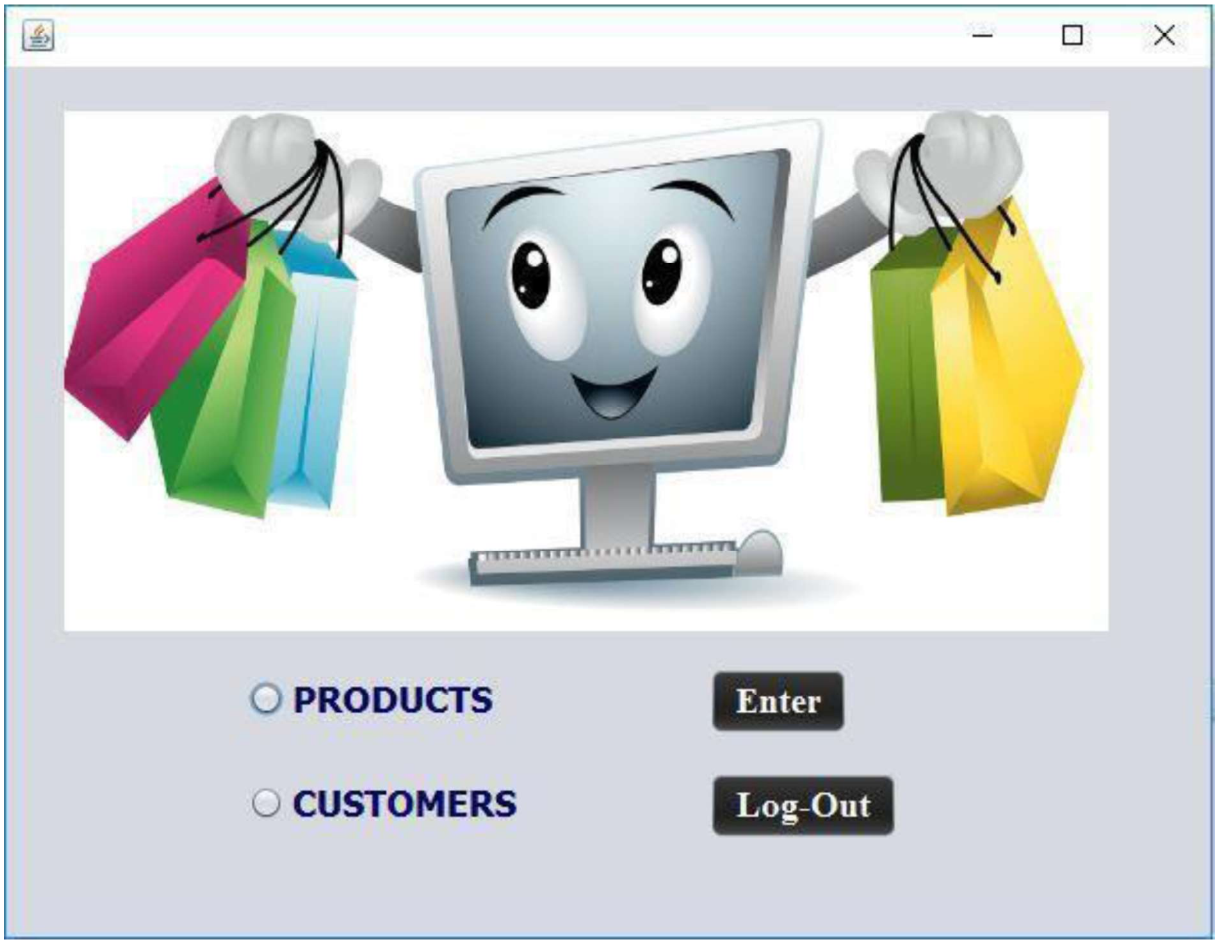


Figure 9.3.1

i) Enter Button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if (prd.isSelected()==true)
  {
     this.setVisible(false);
     new product().setVisible(true);
  }


if (cst.isSelected()==true)

{

    this.setVisible(false);

    new customer().setVisible(true);


}
```

}

ii) Log-Out Button:-

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

```
        // TODO add your handling code here:

        this.setVisible(false);

        new login().setVisible(true);
```

}

9.4 product.java

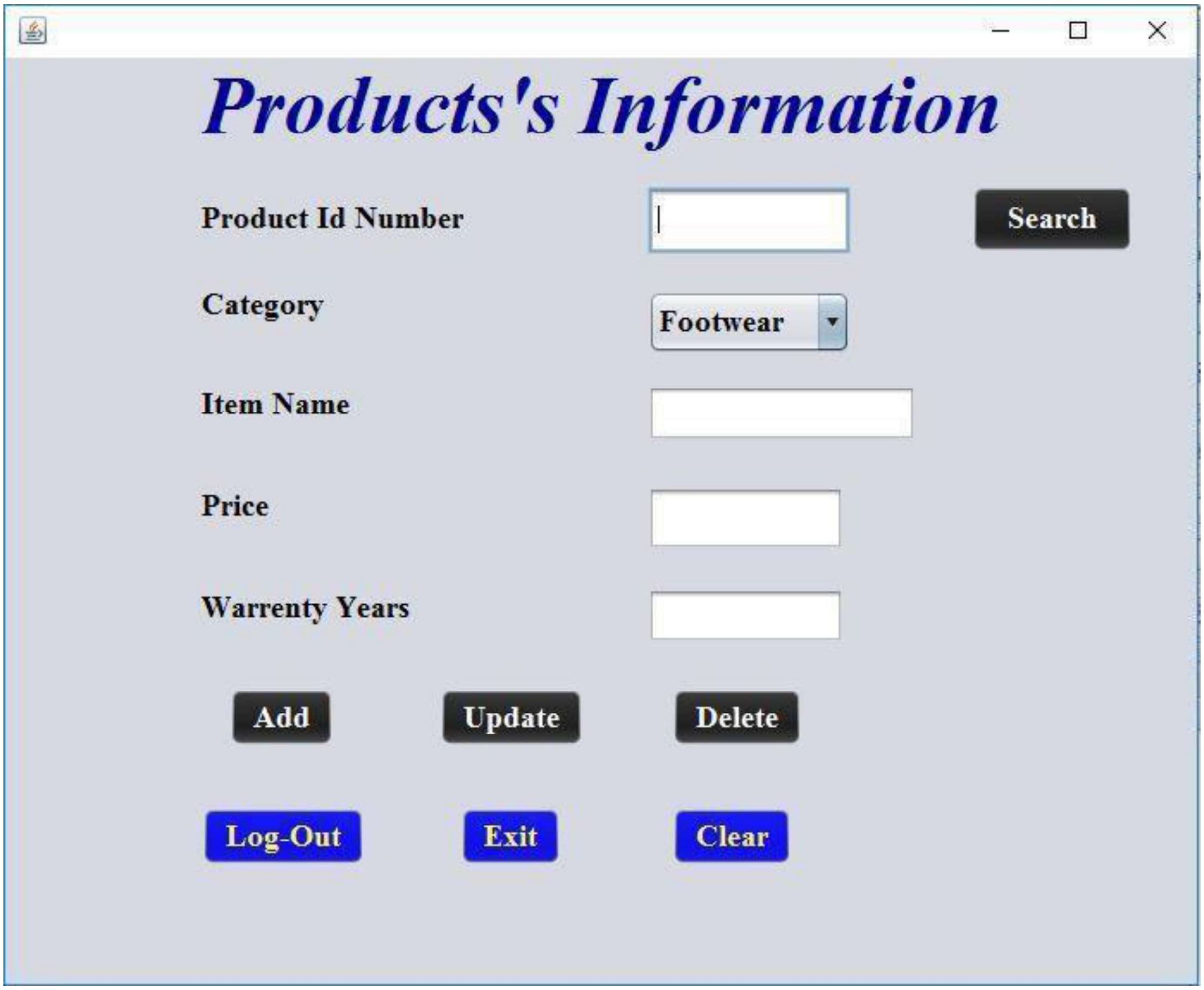


Figure 9.4.1

i) Add button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    try
```

```
  {

Class.forName("java.sql.Driver");

con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");

stmt=con.createStatement();


p = Integer.parseInt(ProductId.getText());

pr= Integer.parseInt(Price.getText());

w= Integer.parseInt(Warrenty.getText());

c= (String)Category.getSelectedItem();

n=Item.getText();


query = "insert into products values ("+p+",'"+c+"','"+n+"',"+pr+","+w+");";

System.out.println(query);


stmt.executeUpdate(query);


 JOptionPane.showMessageDialog(null, "RECORD ADDED SUCCESSFULLY");

 }

 catch (Exception e1)


{
JOptionPane.showMessageDialog(null, "RECORD CANT BE ADDED");

}

 }
```

ii) Search Button:-

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        try

 {

 Class.forName("java.sql.Driver");

 con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");

 stmt=con.createStatement();



  p=Integer.parseInt(ProductId.getText());

   query="select * from products where product_id="+p+";";

  System.out.println(query);

  rs= stmt.executeQuery(query);

  if(rs.next())

  {



        c=rs.getString("category");

        n=rs.getString("item");
        pr=rs.getInt("price");

        w=rs.getInt("warrenty");


        Category.setSelectedItem(" "+c);

        Item.setText(" "+n);

        Price.setText(" "+pr);

        Warrenty.setText(" "+w);

        }

        else

       {

       JOptionPane.showMessageDialog(null, "Record doesnt exist");

        }

        }

     catch(Exception e1)

      {

        JOptionPane.showMessageDialog(null, "Error in Connectivity");

      }

    }
```

iii) Update button:-

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

```
        // TODO add your handling code here:
```

```
        try

    {

    Class.forName("java.sql.Driver");

    con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");

    stmt=con.createStatement();


    p= Integer.parseInt(ProductId.getText());

    pr = Integer.parseInt(Price.getText());

    w= Integer.parseInt(Warrenty.getText());

    n= Item.getText();

    c=(String)Category.getSelectedItem();

query = "update products set category='"+c+"',item='"+n+"',price="+pr+",warrenty="+w+" where
product_id="+p+";";

    System.out.println(query);

    stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(null, "RECORD UPDATED SUCCESSFULLY");

    }

  catch (Exception e1)

  {

    JOptionPane.showMessageDialog(null, "RECORD CANT BE UPDATED");

   }
   }
```

iv) Delete Button:-

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

```
       // TODO add your handling code here:
       try
   {
   Class.forName("java.sql.Driver");
   con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");
   stmt=con.createStatement();

   p=Integer.parseInt(ProductId.getText());
   query= "delete from products where product_id="+p+";";
   System.out.println(query);
   stmt.executeUpdate(query);
   JOptionPane.showMessageDialog(null, "RECORD DELETED SUCCESSFULLY");
   }
   catch (Exception e)
   {
   JOptionPane.showMessageDialog(null, "RECORD CANT BE DELETED");
  }
   }
```

v) Log-Out Button:-

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        this.setVisible(false);

        new login().setVisible(true);

    }
```

vi) Exit Button:-

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        System.exit(0);

    }
```

vii) Clear Button:-

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        ProductId.setText(" ");

        Item.setText(" ");

        Price.setText(" ");

        Warrenty.setText(" ");

    }
```

9.5 customer.java

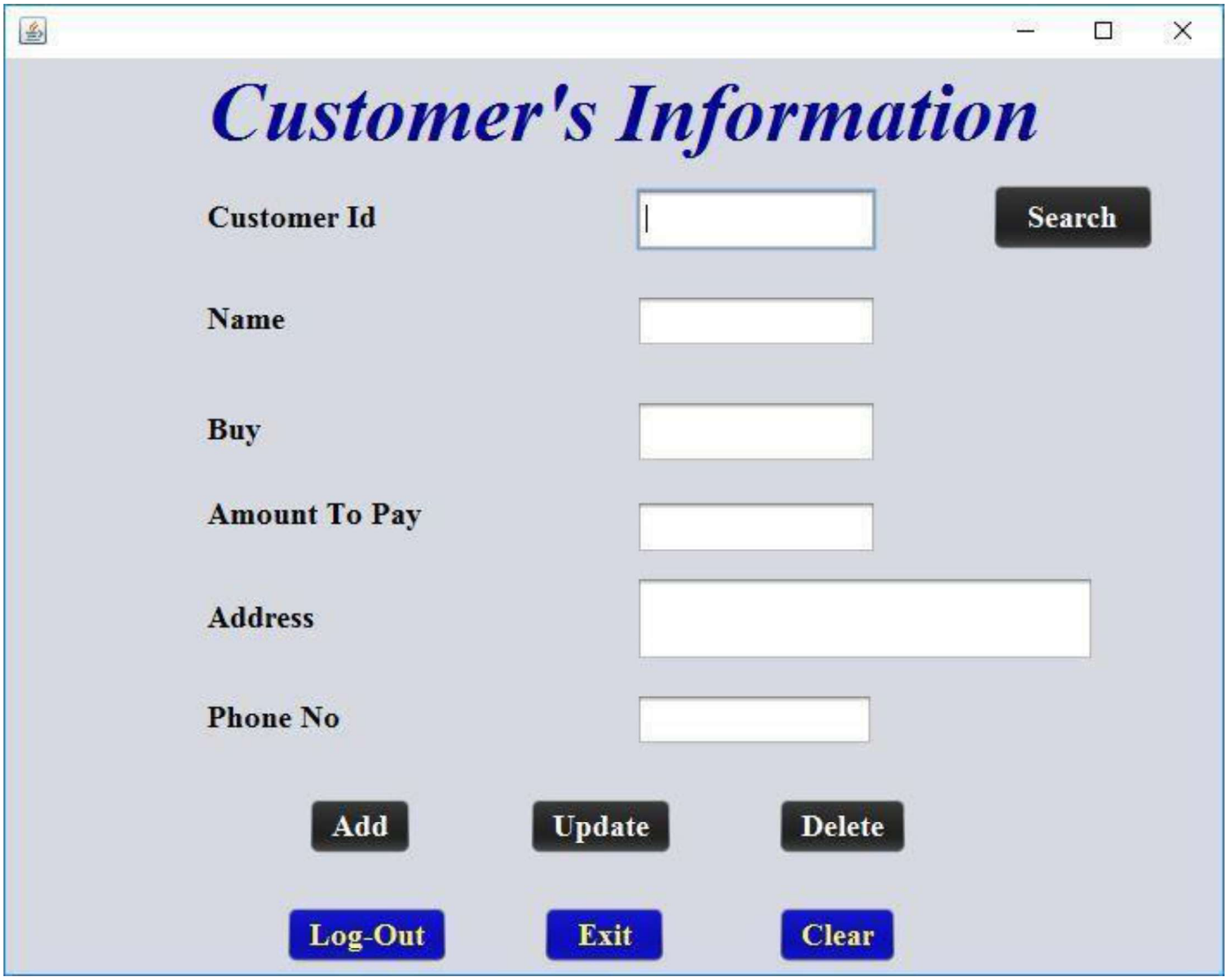


Figure 9.5.1

i) Add Button

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        try
```

```
 {
Class.forName("java.sql.Driver");
 con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");
 stmt=con.createStatement();
 p=Phoneno.getText();
 c=Integer.parseInt(CustomerId.getText());
 b=Buy.getText();
 a=Address.getText();
 n=Name.getText();
 am=Integer.parseInt(Pay.getText());
 query="insert into customers values("+c+",'"+n+"','"+b+"','"+a+"','"+p+"',"+am+");";
  System.out.println(query);
 stmt.executeUpdate(query);
 JOptionPane.showMessageDialog(null, "RECORD ADDED");
 }
 catch (Exception e1)
{
 JOptionPane.showMessageDialog(null, "RECORD CANT BE ADDED");
}
 }
```

ii) Search Button:-

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try
    {
    Class.forName("java.sql.Driver");
    con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");
    stmt=con.createStatement();
    c=Integer.parseInt(CustomerId.getText());
    query="select * from customers where customer_id="+c+";";
   System.out.println(query);
   rs= stmt.executeQuery(query);
   if(rs.next())
   {
        n=rs.getString("name");
        b=rs.getString("buy");
        a=rs.getString("address");
        p=rs.getString("phone_no");
        am=rs.getInt("amount");

    Name.setText(" "+n);
    Buy.setText(" "+b);

    Address.setText(" "+a);

    Phoneno.setText(" "+p);

    Pay.setText(" "+am);

}

  else

   {

    JOptionPane.showMessageDialog(null, "RECORD DOSENT EXISTS");

   }

    }

    catch(Exception e1)

    {

        JOptionPane.showMessageDialog(null, "ERROR IN CONNECTIVITY");

     }

    }
```

iii) Update Button:-

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

```
        // TODO add your handling code here:

        try

       {
```

```
      Class.forName("java.sql.Driver");

      con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");

      stmt=con.createStatement();



       p=Phoneno.getText();

      c=Integer.parseInt(CustomerId.getText());

      b=Buy.getText();

        a=Address.getText();

        n=Name.getText();

        am=Integer.parseInt(Pay.getText());

query = "update customers set
name='"+n+"',buy='"+b+"',address='"+a+"',phone_no='"+p+"',amount="+am+" where
customer_id="+c+";";

     System.out.println(query);

    stmt.executeUpdate(query);

    JOptionPane.showMessageDialog(null, "RECORD UPDATED");

    }

   catch (Exception e1)

   {

    JOptionPane.showMessageDialog(null, "RECORD CANT BE UPDATED");

   }

    }
```

iv) Delete Button

```
  private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      try
   {
     Class.forName("java.sql.Driver");
    con=DriverManager.getConnection("jdbc:mysql://localhost/hemant","root","root");
    stmt=con.createStatement();
   c=Integer.parseInt(CustomerId.getText());
   query= "delete from customers where customer_id="+c+";";
   System.out.println(query);
   stmt.executeUpdate(query);
   JOptionPane.showMessageDialog(null, "RECORD DELETED");
   }
   catch (Exception e)
   {
JOptionPane.showMessageDialog(null, "RECORD CANT BE DELETED");
   }
 }
```

v) Log-Out Button:-

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
       this.setVisible(false);
        new login().setVisible(true);
    }
```

vi) Exit Button:-

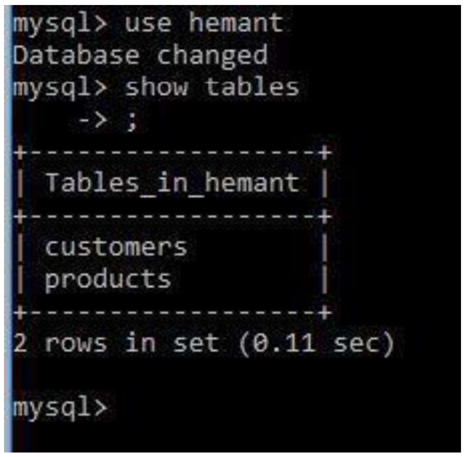
```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        System.exit(0);
    }
```

vii) Clear Button:-

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        CustomerId.setText(" ");
        Name.setText(" ");
        Buy.setText(" ");
        Phoneno.setText(" ");
        Address.setText(" ");
        Pay.setText(" ");
 }
```

CHAPTER 10: TESTING AND SCREENSHOTS OF PROJECT

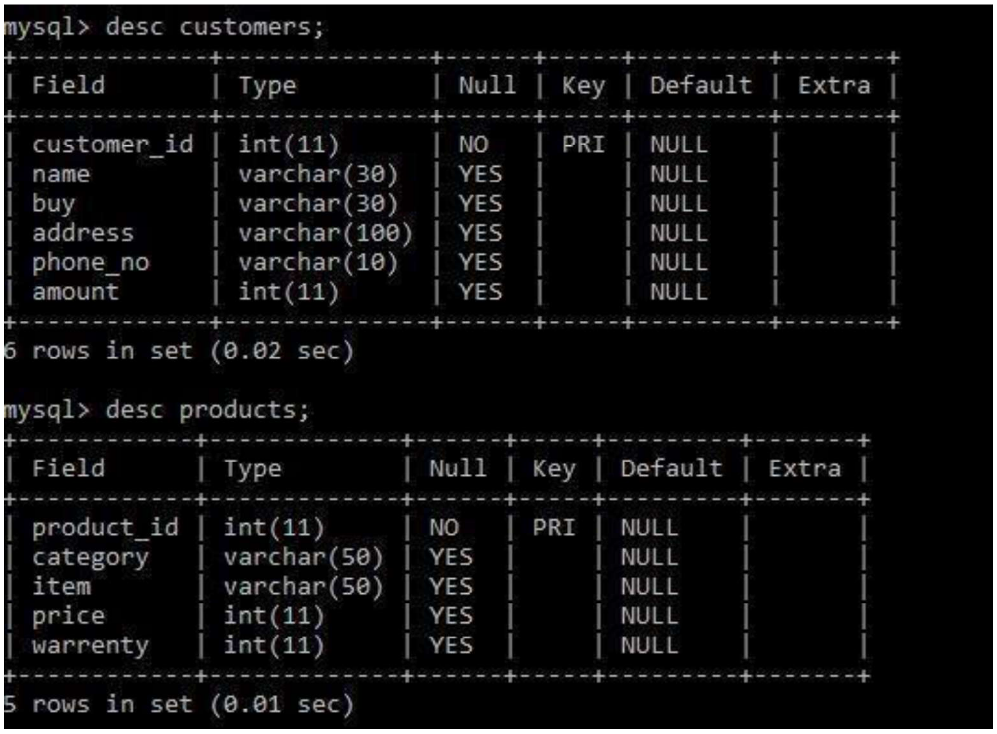
1. Show tables



```
mysql> use hemant
Database changed
mysql> show tables
    -> ;
+------------------+
| Tables_in_hemant |
+------------------+
| customers        |
| products         |
+------------------+
2 rows in set (0.11 sec)

mysql>
```

Figure 10.1 Show Table

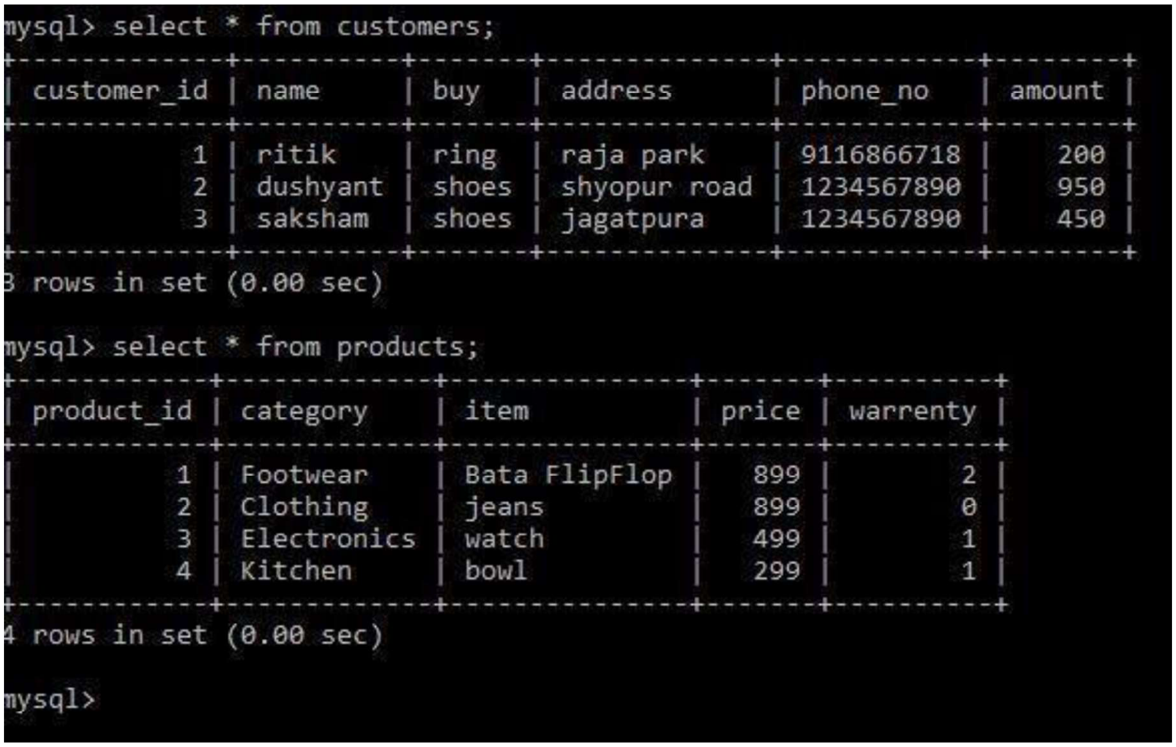
2. Desc Table



```
mysql> desc customers;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| customer_id | int(11)      | NO   | PRI | NULL    |       |
| name        | varchar(30)  | YES  |     | NULL    |       |
| buy         | varchar(30)  | YES  |     | NULL    |       |
| address     | varchar(100) | YES  |     | NULL    |       |
| phone_no    | varchar(10)  | YES  |     | NULL    |       |
| amount      | int(11)      | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
6 rows in set (0.02 sec)

mysql> desc products;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| product_id | int(11)     | NO   | PRI | NULL    |       |
| category   | varchar(50) | YES  |     | NULL    |       |
| item       | varchar(50) | YES  |     | NULL    |       |
| price      | int(11)     | YES  |     | NULL    |       |
| warrenty   | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Figure 10.2 Desc. Table

3. Show table content



```
mysql> select * from customers;
+-------------+----------+-------+--------------+------------+--------+
| customer_id | name     | buy   | address      | phone_no   | amount |
+-------------+----------+-------+--------------+------------+--------+
|           1 | ritik    | ring  | raja park    | 9116866718 |    200 |
|           2 | dushyant | shoes | shyopur road | 1234567890 |    950 |
|           3 | saksham  | shoes | jagatpura    | 1234567890 |    450 |
+-------------+----------+-------+--------------+------------+--------+
3 rows in set (0.00 sec)

mysql> select * from products;
+------------+-------------+----------------+-------+----------+
| product_id | category    | item           | price | warrenty |
+------------+-------------+----------------+-------+----------+
|          1 | Footwear    | Bata FlipFlop  |   899 |        2 |
|          2 | Clothing    | jeans          |   899 |        0 |
|          3 | Electronics | watch          |   499 |        1 |
|          4 | Kitchen     | bowl           |   299 |        1 |
+------------+-------------+----------------+-------+----------+
4 rows in set (0.00 sec)

mysql>
```

Figure 10.3 Show table content

4. Welcome Page

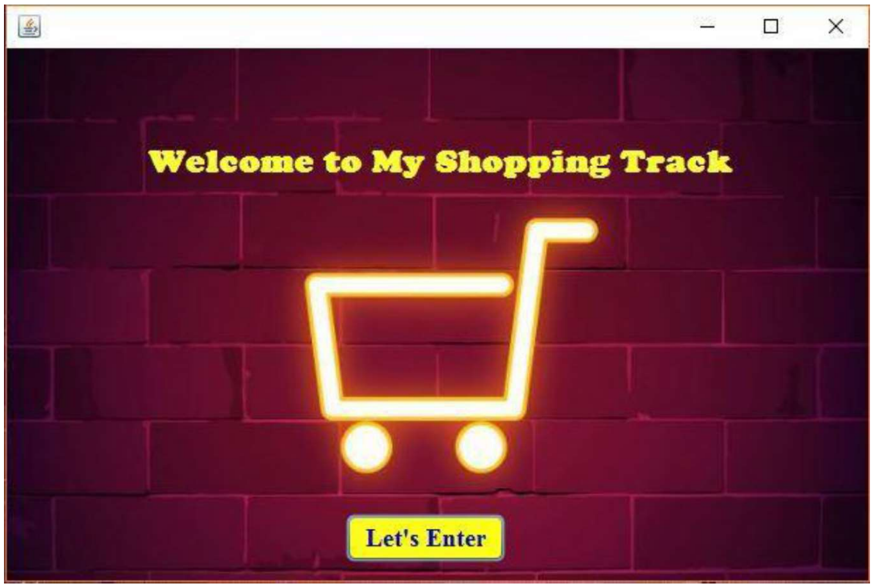


Figure 10.4 Welcome page snapshot

5. Login Page

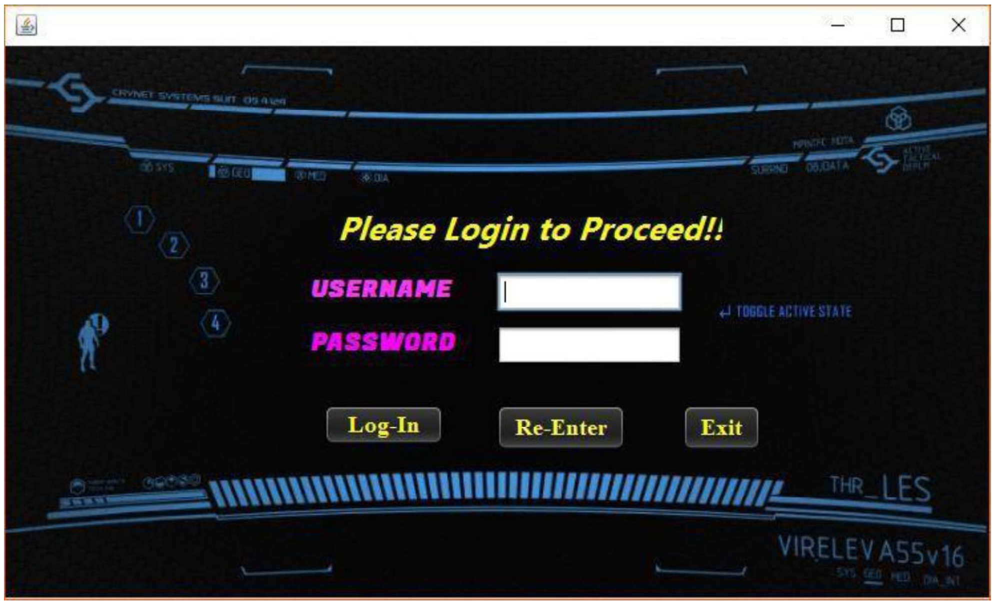


Figure 10.5 Login Page snapshot

6. Home Page

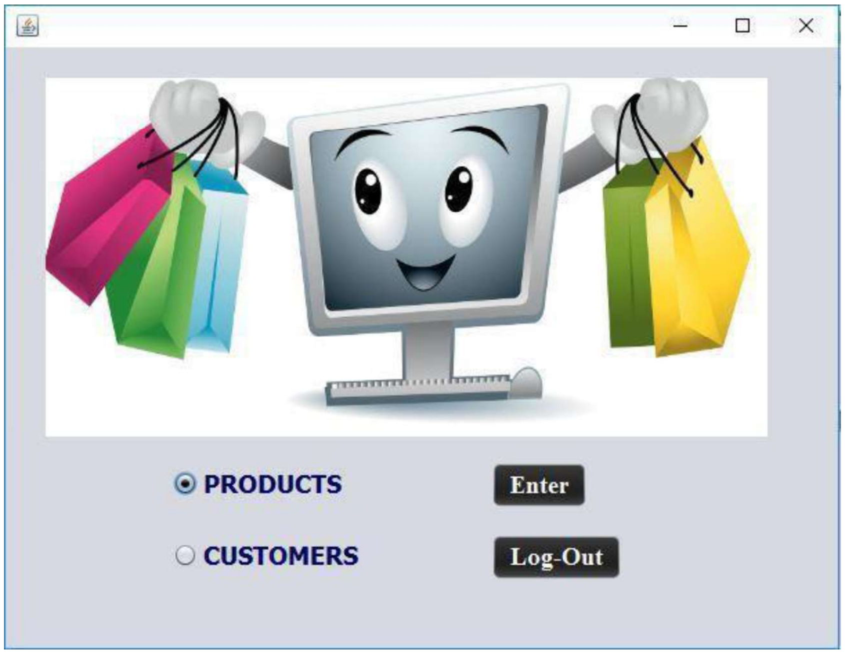
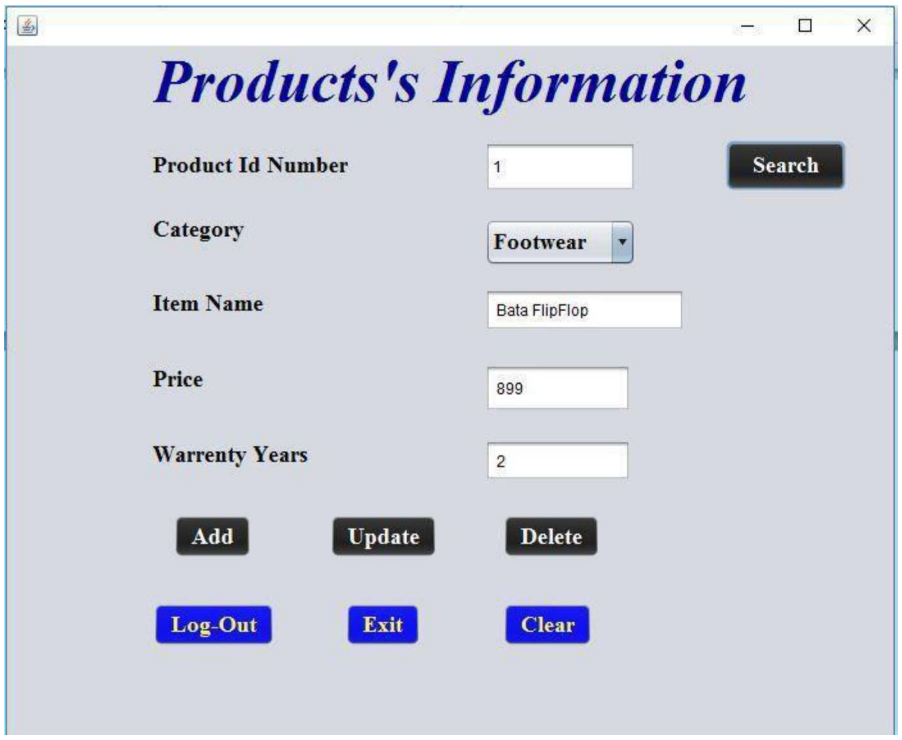


Figure 10.6 Home Page snapshot

7. Product's Page

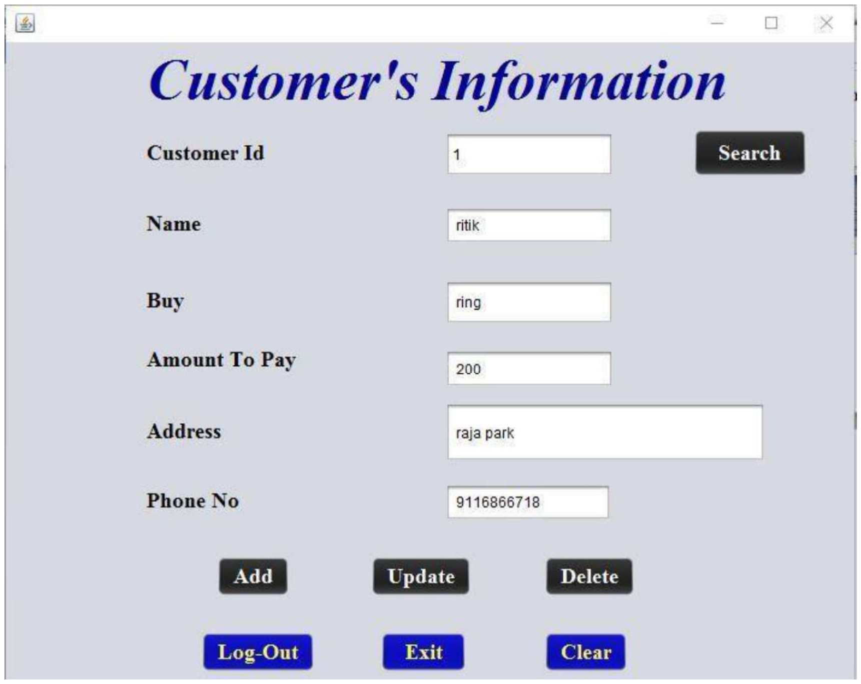


Products's Information

Product Id Number	1
Category	Footwear
Item Name	Bata FlipFlop
Price	899
Warrenty Years	2

Figure 10.7 Product Page snapshot

8. Customer's Page



Customer's Information

Customer Id	1
Name	ritik
Buy	ring
Amount To Pay	200
Address	raja park
Phone No	9116866718

Figure 10.8 Customer Page snapshot

CHAPTER 11: FUTURE OF THE PROJECT

This Project will aim on understanding the functioning of software used by service desk employee in shopping mall to add, delete and update customer and products information in databases and how it can be more improved using advanced technologies like artificial intelligence and this project will help new programmers and students to understand the use of J2EE after learning basic java and to deal with graphics, databases and connectivity and how a desktop application can be useful and fruitful in multiple dynamic aspects.

CHAPTER 12: CONCLUSION

This is to Conclude that this project simulates the basic fundamental functioning of how shopping mall's inventory, stock, customer's information is managed by its employees using one software. This data is also used for billing purpose too.

CHAPTER 13: BIBLIOGRAPHY

Reference Books:

- JAVA The Complete Reference 8th Edition
- Head First JAVA

Reference Websites

- https://www.tutorialspoint.com/awt/awt_graphics_class.htm
- https://www.webopedia.com/TERM/J/J2EE.html
- https://www.oracle.com/java/technologies/appmodel.html
- http://zetcode.com/db/mysqljava/
- https://en.wikipedia.org/wiki/MySQL
- https://dev.mysql.com/doc/connectors/en/connector-j-reference-type-conversions.html
- https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-usagenotes-connect-drivermanager.html
- https://www.mysqltutorial.org/mysql-jdbc-tutorial/