BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
BENGALURU-560064

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

ADVANCE JAVA & J2EE (15CS553) ASSIGNMENT

"GROCERY BILLING JAVA APPLICATION THAT COMMUNICATES WITH MARIA DB USING TYPE-4 DRIVER"

By

Shubham Anand	--	*1BYI6IS050* ★
Shubham	--	*1BYI6IS049*
Shreya Raj	--	*1BY16IS048*
Shreeja Indudhara	--	*1BY16IS047*
Shreepooja H	--	*1BY16IS046*

Under the guidance of
Mrs. Veena N
Assistant Professor

INTRODUCTION

Java Programming Language

Java is a simple and yet powerful object oriented programming language and it is in many respects similar to C++. Java originated at Sun Microsystems, Inc. in 1991. It was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. It was developed to provide a platform-independent programming language.

Unlike many other programming languages including C and C++ when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.

Java was designed with a concept of 'write once and run everywhere'. Java Virtual Machine plays the central role in this concept. The JVM is the environment in which Java programs execute. It is a 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.

Java has powerful features. Some of them are :-

Since Java is an object oriented programming language it has following features:

- Reusability of Code
- Emphasis on data rather than procedure
- Data is hidden and cannot be accessed by external functions
- Objects can communicate with each other through functions
- New data and functions can be easily added
- Simple
- Reusability of Code
- Portable (Platform Independent)
- Distributed
- Robust
- Secure
- High Performance
- Dynamic
- Threaded
- Interpreted

MariaDB Database

MariaDB is a popular fork of MySQL created by MySQL's original developers. It grew out of concerns related to MySQL's acquisition by Oracle. It offers support for both small data processing tasks and enterprise needs. It aims to be a drop-in replacement for MySQL requiring only a simple uninstall of MySQL and an install of MariaDB. MariaDB offers the same features of MySQL and much more.

Key Features of MariaDB

- All of MariaDB is under GPL, LGPL, or BSD.
- MariaDB includes a wide selection of storage engines, including high-performance storage engines, for working with other RDBMS data sources.
- MariaDB uses a standard and popular querying language.
- MariaDB runs on a number of operating systems and supports a wide variety of programming languages.
- MariaDB offers support for PHP, one of the most popular web development languages.
- MariaDB offers Galera cluster technology.
- MariaDB also offers many operations and commands unavailable in MySQL, and eliminates/replaces features impacting performance negatively.

Java DataBase Connectivity

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases .The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as −

- Java Applications
- Java Applets
- Java Servlets
- Java Server Pages (JSPs)
- Enterprise JavaBeans (EJBs).

All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data. JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

There are 3 types of Statements Used in Java Aplication, as given below :

Statement:

It can be used for general-purpose access to the database. It is useful when you are using static SQL statements at runtime.

PreparedStatement:

It can be used when you plan to use the same SQL statement many times. The PreparedStatement interface accepts input parameters at runtime.

CallableStatement:

CallableStatement can be used when you want to access database stored procedures.

Today, there are five types of JDBC drivers in use:

Type 1: JDBC-ODBC bridge

Type 2: partial Java driver

Type 3: pure Java driver for database middleware

Type 4: pure Java driver for direct-to-database

Type 5: highly-functional drivers with superior performance

For most applications, the best choice is a pure Java driver, either Type 3, Type 4, or even Type 5.

Type 5 drivers (such as DataDirect JDBC drivers) offer advanced functionality and superior performance over other driver types.

Type 4 Drivers are the most common and are designed for a particular vendor's database

In contrast, a Type 3 driver is a single JDBC driver used to access a middleware server, which, in turn, makes the relevant calls to the database. A good example of Type 3 JDBC driver is the DataDirect SequeLink JDBC driver.

Type 1 drivers are used for testing JDBC applications against an ODBC data source. Type 2 drivers require a native database API to be used. Both Type 1 and Type 2 mix a Java-based API with another API ,

GROCERY BILLING SYSTEM

Billing systems are enabled by combining hardware and software. The hardware used is either a phone or a PC and the software is one which is designed by the company using the particular billing system. This method of billing is generally used by Glocery Billing System.

The main purpose of a billing system is to make life easier for a customer. It is a basic customer care tool which also ensures that the company obtains payments in time. Before setting up a billing account, the personal information of customers is analysed. The customers' credit worthiness is the main information which an agency will pay attention to. After all this information is verified, the customers' billing systems are established.

Service

Service is the entity offered by the company and targeted to the customers. Each service is defined by an engineering employee as a service catalog which includes service type, name, billing policy, and its default rating profile. All of these attributes are described later in this article. After service ordering by the customer, it takes a unique ID which is attached with the account of the customer, provided by the provisioning system to the customer at a certain date.

Account

The customer account includes customer contact profile information, account type, login information, and payment method. Each customer account is linked in the system with specific services offered to this customer, and the customer will be billed depending on his usage of these services. Customer account belongs to a specific account type, which is related to some price plans determined by discounts and promotions.

Rating

One of the big issues in any billing system is how, when, and where the companies should bill their customers. Rating is the process of converting usage records from one form to another, like converting usage units to its cost. This process is essential in the billing process as it is the point of conversion of the service usage to revenue value to company, which is the target of companies.

Invoices and payments

Each customer in the system has an account balance, which affects any invoices requested by the customer and any payments done by the customer. Normally the invoices are generated as a result of service hosting or using, and payments are the customer payments as a result of the invoicing operation.

Source Code

```
package containers;

import java.sql.*;
import java.util.*;
import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.text.ParseException;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import net.proteanit.sql.DbUtils;
import javax.swing.JTabbedPane;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.LayoutStyle.ComponentPlacement;
import java.awt.Font;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.SwingConstants;
import javax.swing.JComboBox;
import javax.swing.JTable;
import javax.swing.JScrollPane;

@SuppressWarnings("serial")
public class BillSystemFrame extends JFrame {

	private JPanel contentPane;
	private JTextField txtFirstName;
	private JTextField txtLastName;
	private JLabel lblCustomerid;
	private JButton btnSubmit;
	private JTextField txtCusContactNo;
	private JLabel lblCustomerId_1;
	private JLabel lblVegetable;
	private JComboBox<Object> comboBox;
	private JTextField textField;
	private JLabel lblFruits;
	private JComboBox<Object> comboBox_1;
	private JTextField textField_1;
	private JComboBox<Object> comboBox_2;
	private JLabel lblPulsesAndSpices;
	private JTextField textField_2;
	private JLabel lblStationary;
	private JComboBox<?> comboBox_3;
	private JTextField textField_3;
	private JLabel lblOthers;
	private JTextField textField_4;
	private JComboBox<?> comboBox_4;
	private JButton btnReset;
	private JLabel lblBillID;
	private Timestamp timestamp;
	private Connection con;
	private JButton btnAddVegetable;
	private JButton btnAddFruit;
	private JButton btnAddItem;
	private JButton btnAddItem_1;
	private JButton btnAddItem_2;
	private long timeI;
	private JButton btnTotal;
	private JTextField textField_5;
	private JButton btnSubmit_1;
	private long Cus_ID;
	private float iprice = 0;
	private float iQua = 0;
	private float iprice_1 = 0;
	private float iQua_1 = 0;
```

```
	private float iprice_2 = 0;
	private float iQua_2 = 0;
	private float iprice_3 = 0;
	private float iQua_3 = 0;
	private float iprice_4 = 0;
	private float iQua_4 = 0;
	private float gTotal = 0;
	private JLabel lblGrandTotal;
	private JScrollPane scrollPane;
	private JTable table;
	private JTable table_1;
	private JButton btnGloceryTransactionDetails;
	private JButton btnSearchCustomer;
	private JLabel lblCusconformation;
	private JLabel lblSubmitedToBill;

	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					BillSystemFrame frame = new BillSystemFrame();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	  // @throws Exception
	public BillSystemFrame() throws Exception, ParseException
	{
		String driver = "com.mysql.cj.jdbc.Driver";
		Class.forName(driver);
		con =
DriverManager.getConnection("jdbc:mysql://localhost:3305/BillingSystem?user=root&password=student");
		setTitle("Billing System");
		addComponents();
		addEvents();

	}
	//////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////
	////////////////////////////////////	Method for EventListener.
//////////////////////////////////////////
	//////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////
      private void addEvents() throws Exception
      {
        btnSubmit.addActionListener(new ActionListener()
        {
		public void actionPerformed(ActionEvent arg0) {

			lblCustomerid.setText("Customer ID : "+ txtCusContactNo.getText());
			lblCustomerid.setVisible(true);
			lblCustomerId_1.setText("Customer ID : "+ txtCusContactNo.getText());
			lblCustomerId_1.setVisible(true);

		    timestamp = new Timestamp(System.currentTimeMillis());
		    timeI =  timestamp.getTime();
		    lblBillID.setText("Bill ID : "+ timestamp.getTime());
			lblBillID.setVisible(true);
			try
				{
					long Cus_ID_1 = Long.parseLong(txtCusContactNo.getText());
					String sql = "SELECT * FROM bill where Cus_ID = '"+ Cus_ID_1 +"'";
					PreparedStatement pstmt = con.prepareStatement(sql);
					ResultSet rs = pstmt.executeQuery();
					if (rs.next()) {
						long Cus_ID_2 = Long.parseLong(txtCusContactNo.getText());
				PreparedStatement pstmt_21 = null;
				String C_Name = null;
						try {
```

```
                                                String sql_1 = " SELECT Cus_Name FROM customer WHERE
Cus_ID = '" + Cus_ID_2 + "' ";
                                                pstmt_21 = con.prepareStatement(sql_1);
                                                ResultSet rs_11 = pstmt_21.executeQuery();
                                                if (rs_11.next()) {
                                                        C_Name = rs_11.getString("Cus_Name");
                                                }
                                                lblCusconformation.setText("Our Company Welcome You
"+ C_Name);
                                        lblCusconformation.setVisible(true);

                                        } catch ( Exception e ){ System.out.println(e);}
                                } else {
                                        try
                                {
                            String query_1 = "insert into Customer values( ?, ?, ?)";
                                        PreparedStatement pstmt_20 = con.prepareStatement(query_1);

                                        Cus_ID = Long.parseLong(txtCusContactNo.getText());
                                        String Cus_Name = (String)txtFirstName.getText() +"
"+(String)txtLastName.getText();
                                        pstmt_20.setLong(1, Cus_ID);
                                        pstmt_20.setString(2, Cus_Name);
                                        pstmt_20.setString(3, null);
                                        int i = pstmt_20.executeUpdate();
                                        System.out.println(i +" records inserted");
                                        lblCusconformation.setText("Welcome! To Our Company First ");
                                        lblCusconformation.setVisible(true);
                                        }
                                catch ( SQLException e ){ System.out.println(e);}
                }
                        } catch (Exception e) { System.out.println(e);   }
                        }
                });

        btnSubmit_1.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try
                                {
                                        PreparedStatement pstmt_16 = con.prepareStatement("insert into bill
values( ?, ?, ?, ?, NOW())");
                                        pstmt_16.setLong(1,timeI);
                                        pstmt_16.setLong(2,Cus_ID);
                                        float tQua = iQua + iQua_1 + iQua_2 + iQua_3 + iQua_4;
                                        pstmt_16.setFloat(3, tQua);
                                        gTotal = iprice*iQua + iprice_1*iQua_1 + iprice_2*iQua_2 +
iprice_3*iQua_3 + iprice_4*iQua_4;
                                        pstmt_16.setFloat(4, gTotal);
                                        int i = pstmt_16.executeUpdate();
                                        System.out.println(i+" records inserted");
                                        lblSubmitedToBill.setVisible(true);
                                        } catch ( Exception e ) { System.out.println(e); }
                        }
                });

        btnTotal.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                lblGrandTotal.setText("Grand Total : INR  "+ (iprice*iQua + iprice_1*iQua_1 +
iprice_2*iQua_2 + iprice_3*iQua_3 + iprice_4*iQua_4) );
                        lblGrandTotal.setVisible(true);
                        }
                });

        btnReset.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                textField.setText(null);
                                textField_1.setText(null);
                                textField_2.setText(null);
                                textField_3.setText(null);
                                textField_4.setText(null);
                                txtFirstName.setText("First Name");
                                txtLastName.setText("Last Name");
```

```
                        txtCusContactNo.setText("");
                        lblSubmitedToBill.setVisible(false);
                        lblCusconformation.setVisible(false);
                        lblCustomerid.setVisible(false);
                Statement st_5 = null;
                        try
                                {
                                        st_5 = con.createStatement();
                                        ResultSet rs_10 = st_5.executeQuery("SELECT Cus_ID FROM
customer");
                                        Long Long_1=null;
                                while (rs_10.next()) {
                                Long_1 = rs_10.getLong("Cus_ID");
                                                System.out.println(Long_1);
                                                try {
                                                        String sql_10 = " DELETE FROM billdesctemp
WHERE Cus_ID = '" + Long_1 + "' ";
                                                        PreparedStatement pstmt_15 =
con.prepareStatement(sql_10);
                                                        int i = pstmt_15.executeUpdate();
                                                        System.out.println(i+" records deleted");
                                                        }
                                                catch ( Exception e ){ System.out.println(e);}
                                        }
                                }catch ( Exception e ){ System.out.println(e);}
                                try
                                {
                                        String sql = "SELECT * FROM billdesctemp";
                                        PreparedStatement pstmt = con.prepareStatement(sql);
                                        ResultSet rs = pstmt.executeQuery();
                                        table.setModel(DbUtils.resultSetToTableModel(rs));
                        } catch (Exception e) { System.out.println(e);   }
                        }
                });

        btnSearchCustomer.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try
                                {
                                        long Cus_ID_1 = Long.parseLong(textField_5.getText());
                                        String sql = "SELECT * FROM bill where Cus_ID = '"+ Cus_ID_1 +"'";
                                        PreparedStatement pstmt = con.prepareStatement(sql);
                                        ResultSet rs = pstmt.executeQuery();
                                        if (rs.next()) {
                                                try
                                                {
                                                        String sql_1 = "select * from bill where Cus_ID = '"+
Cus_ID_1 +"'";
                                                        PreparedStatement pstmt_1 =
con.prepareStatement(sql_1);
                                                        ResultSet rs_1 = pstmt_1.executeQuery();
                                                        table_1.setModel(DbUtils.resultSetToTableModel(rs_1));
                                                } catch (Exception e) { System.out.println(e);   }
                                                } else {
                                                        textField_5.setText("Not In Database");
                                                }
                                        } catch (Exception e) { System.out.println(e);   }
                                        }
                        });

        btnAddVegetable.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                        try
                        {
                                String vegetable = null;

                                PreparedStatement pstmt = con.prepareStatement("insert into billdesctemp values( ?, ?,
?, ?, ?, ?, NOW())");
                                PreparedStatement pstmt_2 = con.prepareStatement("insert into totalbilldesc values( ?, ?,
?, ?, ?, ?, NOW())");

                                pstmt.setLong(1,timeI);  //1 specifies the first parameter in the query
```

```
                        pstmt_2.setLong(1, timeI);
                        pstmt.setLong(2, Long.parseLong(txtCusContactNo.getText()));
                        pstmt_2.setLong(2, Long.parseLong(txtCusContactNo.getText()));

                        if(comboBox.getSelectedItem().toString() != null)
                                vegetable = (String)comboBox.getItemAt(comboBox.getSelectedIndex());
                        else
                                System.out.println("nulle");

                        pstmt.setString(3, vegetable);
                        pstmt_2.setString(3, vegetable);
                        PreparedStatement pstmt_1 = null;
                        try {
                                        String sql_1 = " SELECT Price FROM vegetables WHERE Vegetables = '" +
vegetable + "' ";
                                        pstmt_1 = con.prepareStatement(sql_1);
                                        ResultSet rs_5 = pstmt_1.executeQuery();
                                        if (rs_5.next()) {
                                                iprice = rs_5.getFloat("Price");
                                        }
                                } catch ( Exception e ){ System.out.println(e);}

                        pstmt.setFloat(4, iprice);
                        pstmt_2.setFloat(4, iprice);

                   iQua = Float.parseFloat(textField.getText());

                        pstmt.setFloat(5, iQua);
                        pstmt_2.setFloat(5, iQua);

                        Float calPrice = iprice * iQua ;

                        pstmt.setFloat(6, calPrice);
                        pstmt_2.setFloat(6, calPrice);

                        int i = pstmt.executeUpdate();
                        int j = pstmt_2.executeUpdate();

                        System.out.println(i+j +" records inserted");
                        }
                catch ( Exception e ){ System.out.println(e);}
                try
                        {
                                String sql = "SELECT * FROM billdesctemp";
                                PreparedStatement pstmt = con.prepareStatement(sql);
                                ResultSet rs = pstmt.executeQuery();
                                table.setModel(DbUtils.resultSetToTableModel(rs));
                        } catch (Exception e) { System.out.println(e);   }
        }
        });

        btnAddFruit.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try
                        {
                                String fruit = null;
                                PreparedStatement pstmt_3 = con.prepareStatement("insert into billdesctemp
values( ?, ?, ?, ?, ?, ?, NOW())");
                                PreparedStatement pstmt_5 = con.prepareStatement("insert into totalbilldesc
values( ?, ?, ?, ?, ?, ?, NOW())");

                                pstmt_3.setLong(1,timeI);  //1 specifies the first parameter in the query
                                pstmt_5.setLong(1, timeI);
                                pstmt_3.setLong(2, Long.parseLong(txtCusContactNo.getText()));
                                pstmt_5.setLong(2, Long.parseLong(txtCusContactNo.getText()));
                                if(comboBox_1.getSelectedItem().toString() != null)
                                fruit = (String)comboBox_1.getItemAt(comboBox_1.getSelectedIndex());
                        else
                                System.out.println("nulle");

                        pstmt_3.setString(3, fruit);
                        pstmt_5.setString(3, fruit);
```

```
                        PreparedStatement pstmt_4 = null;
                                try {
                                                String sql_1 = " SELECT Price FROM fruits WHERE Fruits = '" +
fruit + "' ";
                                                pstmt_4 = con.prepareStatement(sql_1);
                                                ResultSet rs_6 = pstmt_4.executeQuery();
                                                if (rs_6.next()) {
                                                        iprice_1 = rs_6.getFloat("Price");
                                                }
                                        } catch ( Exception e ){ System.out.println(e);}

                                pstmt_3.setFloat(4, iprice_1);
                                pstmt_5.setFloat(4, iprice_1);
                                iQua_1 = Float.parseFloat(textField_1.getText());
                                pstmt_3.setFloat(5, iQua_1);
                                pstmt_5.setFloat(5, iQua_1);
                                Float calPrice = iprice_1 * iQua_1 ;
                                pstmt_3.setFloat(6, calPrice);
                                pstmt_5.setFloat(6, calPrice);
                                int i = pstmt_3.executeUpdate();
                                int j = pstmt_5.executeUpdate();
                                System.out.println(i+j +" records inserted");
                                }
                                catch ( Exception e ){ System.out.println(e);}
                                try
                                {
                                        String sql = "SELECT * FROM billdesctemp";
                                        PreparedStatement pstmt = con.prepareStatement(sql);
                                        ResultSet rs = pstmt.executeQuery();
                                        table.setModel(DbUtils.resultSetToTableModel(rs));
                                } catch (Exception e) { System.out.println(e);   }
                                }
                        });

        btnAddItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try
                                {
                                String kitchen_items = null;
                                PreparedStatement pstmt_6 = con.prepareStatement("insert into billdesctemp
values( ?, ?, ?, ?, ?, ?, NOW())");
                                PreparedStatement pstmt_8 = con.prepareStatement("insert into totalbilldesc
values( ?, ?, ?, ?, ?, ?, NOW())");

                                pstmt_6.setLong(1,timeI);  //1 specifies the first parameter in the query
                                pstmt_8.setLong(1, timeI);

                                pstmt_6.setLong(2, Long.parseLong(txtCusContactNo.getText()));
                                pstmt_8.setLong(2, Long.parseLong(txtCusContactNo.getText()));

                                if(comboBox_2.getSelectedItem().toString() != null)
                                kitchen_items = (String)comboBox_2.getItemAt(comboBox_2.getSelectedIndex());
                        else
                                System.out.println("nulle");

                        pstmt_6.setString(3, kitchen_items);
                        pstmt_8.setString(3, kitchen_items);

                        PreparedStatement pstmt_7 = null;
                                try {
                                                String sql_2 = " SELECT Price FROM kitchen WHERE Items = '" +
kitchen_items + "' ";
                                                pstmt_7 = con.prepareStatement(sql_2);
                                                ResultSet rs_7 = pstmt_7.executeQuery();
                                                if (rs_7.next()) {
                                                        iprice_2 = rs_7.getFloat("Price");
                                                }
                                        } catch ( Exception e ){ System.out.println(e);}

                                pstmt_6.setFloat(4, iprice_2);
                                pstmt_8.setFloat(4, iprice_2);
```

```
                                iQua_2 = Float.parseFloat(textField_2.getText());

                                pstmt_6.setFloat(5, iQua_2);
                                pstmt_8.setFloat(5, iQua_2);

                                Float calPrice = iprice_2 * iQua_2 ;

                                pstmt_6.setFloat(6, calPrice);
                                pstmt_8.setFloat(6, calPrice);

                                int i = pstmt_6.executeUpdate();
                                int j = pstmt_8.executeUpdate();

                                System.out.println(i+j +" records inserted");
                                }
                                catch ( Exception e ){ System.out.println(e);}
                                try
                                {
                                        String sql = "SELECT * FROM billdesctemp";
                                        PreparedStatement pstmt = con.prepareStatement(sql);
                                        ResultSet rs = pstmt.executeQuery();
                                        table.setModel(DbUtils.resultSetToTableModel(rs));
                        } catch (Exception e) { System.out.println(e);   }

                                }
                        });

        btnAddItem_1.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try
                                {
                                        String stationary_items = null;

                                        PreparedStatement pstmt_9 = con.prepareStatement("insert into billdesctemp
values( ?, ?, ?, ?, ?, ?, NOW())");
                                        PreparedStatement pstmt_11 = con.prepareStatement("insert into totalbilldesc
values( ?, ?, ?, ?, ?, ?, NOW())");

                                        pstmt_9.setLong(1,timeI);  //1 specifies the first parameter in the query
                                        pstmt_11.setLong(1, timeI);

                                        pstmt_9.setLong(2, Long.parseLong(txtCusContactNo.getText()));
                                        pstmt_11.setLong(2, Long.parseLong(txtCusContactNo.getText()));

                                        if(comboBox_3.getSelectedItem().toString() != null)
                                        stationary_items =
(String)comboBox_3.getItemAt(comboBox_3.getSelectedIndex());
                                else
                                        System.out.println("nulle");

                                pstmt_9.setString(3, stationary_items);
                                pstmt_11.setString(3, stationary_items);

                                PreparedStatement pstmt_10 = null;
                                        try {
                                                        String sql_2 = " SELECT Price FROM stationary WHERE Items = '" +
stationary_items + "' ";
                                                        pstmt_10 = con.prepareStatement(sql_2);
                                                        ResultSet rs_8 = pstmt_10.executeQuery();
                                                        if (rs_8.next()) {
                                                                iprice_3 = rs_8.getFloat("Price");
                                                        }
                                                } catch ( Exception e ){ System.out.println(e);}

                                        pstmt_9.setFloat(4, iprice_3);
                                        pstmt_11.setFloat(4, iprice_3);

                                        iQua_3 = Float.parseFloat(textField_3.getText());

                                        pstmt_9.setFloat(5, iQua_3);
                                        pstmt_11.setFloat(5, iQua_3);
```

```
                                Float calPrice = iprice_3 * iQua_3 ;

                                pstmt_9.setFloat(6, calPrice);
                                pstmt_11.setFloat(6, calPrice);

                                int i = pstmt_9.executeUpdate();
                                int j = pstmt_11.executeUpdate();

                                System.out.println(i+j +" records inserted");
                                }
                                catch ( Exception e ){ System.out.println(e);}
                                try
                                {
                                        String sql = "SELECT * FROM billdesctemp";
                                        PreparedStatement pstmt = con.prepareStatement(sql);
                                        ResultSet rs = pstmt.executeQuery();
                                        table.setModel(DbUtils.resultSetToTableModel(rs));
                            } catch (Exception e) { System.out.println(e);   }
                            }
                        });

                btnAddItem_2.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent arg0) {
                                        try
                                {
                                        String others = null;

                                        PreparedStatement pstmt_12 = con.prepareStatement("insert into billdesctemp
values( ?, ?, ?, ?, ?, ?, NOW())");
                                        PreparedStatement pstmt_14 = con.prepareStatement("insert into totalbilldesc
values( ?, ?, ?, ?, ?, ?, NOW())");

                                        pstmt_12.setLong(1,timeI);  //1 specifies the first parameter in the query
                                        pstmt_14.setLong(1, timeI);

                                        pstmt_12.setLong(2, Long.parseLong(txtCusContactNo.getText()));
                                        pstmt_14.setLong(2, Long.parseLong(txtCusContactNo.getText()));

                                        if(comboBox_4.getSelectedItem().toString() != null)
                                        others = (String)comboBox_4.getItemAt(comboBox_4.getSelectedIndex());
                                else
                                        System.out.println("nulle");

                                pstmt_12.setString(3, others);
                                pstmt_14.setString(3, others);

                                PreparedStatement pstmt_13 = null;
                                        try {
                                                        String sql_1 = " SELECT Price FROM others WHERE Items = '" +
others + "' ";
                                                        pstmt_13 = con.prepareStatement(sql_1);
                                                        ResultSet rs_9 = pstmt_13.executeQuery();
                                                        if (rs_9.next()) {
                                                                iprice_4 = rs_9.getFloat("Price");
                                                        }
                                                } catch ( Exception e ){ System.out.println(e);}

                                        pstmt_12.setFloat(4, iprice_4);
                                        pstmt_14.setFloat(4, iprice_4);

                                        iQua_4 = Float.parseFloat(textField_4.getText());

                                        pstmt_12.setFloat(5, iQua_4);
                                        pstmt_14.setFloat(5, iQua_4);

                                        Float calPrice = iprice_4 * iQua_4 ;

                                        pstmt_12.setFloat(6, calPrice);
                                        pstmt_14.setFloat(6, calPrice);

                                        int i = pstmt_12.executeUpdate();
```

```
                        int j = pstmt_14.executeUpdate();

                        System.out.println(i+j +" records inserted");
                        }
                        catch ( Exception e ){ System.out.println(e);}
                        try
                        {
                                String sql = "SELECT * FROM billdesctemp";
                                PreparedStatement pstmt = con.prepareStatement(sql);
                                ResultSet rs = pstmt.executeQuery();
                                table.setModel(DbUtils.resultSetToTableModel(rs));
                    } catch (Exception e) { System.out.println(e);   }
                     }
            });
        btnGloceryTransactionDetails.addActionListener(new ActionListener() {
                     public void actionPerformed(ActionEvent arg0) {
                        try
                        {
                                String sql = "SELECT * FROM totalbilldesc";
                                PreparedStatement pstmt = con.prepareStatement(sql);
                                ResultSet rs = pstmt.executeQuery();
                                table_1.setModel(DbUtils.resultSetToTableModel(rs));
                    } catch (Exception e) { System.out.println(e);   }
                     }
            });

    }

//////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////
     ////////////////////////////////////  Method to Initialize components
//////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////

        @SuppressWarnings({ "rawtypes", "unchecked" })
        private void addComponents() throws Exception, ParseException
        {
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setBounds(0, 0, 1366, 740);
                contentPane = new JPanel();
                contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
                contentPane.setLayout(new BorderLayout(0, 0));
                setContentPane(contentPane);

                JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
                contentPane.add(tabbedPane, BorderLayout.CENTER);

                JPanel panel = new JPanel();
                tabbedPane.addTab("Customer", null, panel, null);

                JLabel lblCostumerName = new JLabel("Costumer Name :");
                lblCostumerName.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));

                txtFirstName = new JTextField();
                txtFirstName.setFont(new Font("Cambria", Font.ITALIC, 17));
                txtFirstName.setText("First Name");
                txtFirstName.setColumns(10);

                txtLastName = new JTextField();
                txtLastName.setFont(new Font("Cambria", Font.ITALIC, 17));
                txtLastName.setText("Last Name");
                txtLastName.setColumns(10);

                JLabel lblCostumerContactNo = new JLabel("Costumer Contact No :");
                lblCostumerContactNo.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));

                txtCusContactNo = new JTextField();
                txtCusContactNo.setFont(new Font("Cambria", Font.ITALIC, 17));
                txtCusContactNo.setColumns(10);
```

```
		btnSubmit = new JButton("Submit");

		btnSubmit.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));

		lblCustomerid = new JLabel("CustomerID");
		lblCustomerid.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));
		lblCustomerid.setVisible(false);

		btnReset = new JButton("Reset");

		btnReset.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));

		lblCusconformation = new JLabel("CusConformation");
		lblCusconformation.setHorizontalAlignment(SwingConstants.LEFT);
		lblCusconformation.setFont(new Font("Cambria", Font.ITALIC, 17));
		lblCusconformation.setVisible(false);

		GroupLayout gl_panel = new GroupLayout(panel);
		gl_panel.setHorizontalGroup(
			gl_panel.createParallelGroup(Alignment.LEADING)
				.addGroup(gl_panel.createSequentialGroup()
					.addGap(280)
					.addGroup(gl_panel.createParallelGroup(Alignment.TRAILING)
						.addGroup(gl_panel.createSequentialGroup()
							.addComponent(lblCustomerid)
							.addGap(541)
							.addComponent(btnReset, GroupLayout.PREFERRED_SIZE,
99, GroupLayout.PREFERRED_SIZE)
							.addGap(39)
							.addComponent(btnSubmit,
GroupLayout.PREFERRED_SIZE, 125, GroupLayout.PREFERRED_SIZE)
							.addContainerGap(160, Short.MAX_VALUE))
						.addGroup(gl_panel.createSequentialGroup()

	.addGroup(gl_panel.createParallelGroup(Alignment.LEADING)
								.addGroup(gl_panel.createSequentialGroup()
									.addComponent(lblCostumerContactNo,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
									.addGap(77)
									.addComponent(txtCusContactNo,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
									.addGap(51))
								.addGroup(gl_panel.createSequentialGroup()
									.addComponent(lblCostumerName,
GroupLayout.DEFAULT_SIZE, 132, Short.MAX_VALUE)
									.addGap(18)
									.addComponent(txtFirstName,
GroupLayout.DEFAULT_SIZE, 132, Short.MAX_VALUE)
									.addGap(18)
									.addComponent(txtLastName, 142, 142,
142)))
								.addGap(613))))
					.addGroup(gl_panel.createSequentialGroup()
						.addGap(416)
						.addComponent(lblCusconformation, GroupLayout.PREFERRED_SIZE, 501,
GroupLayout.PREFERRED_SIZE)
						.addContainerGap(418, Short.MAX_VALUE))
		);
		gl_panel.setVerticalGroup(
			gl_panel.createParallelGroup(Alignment.TRAILING)
				.addGroup(gl_panel.createSequentialGroup()
					.addGap(173)
					.addGroup(gl_panel.createParallelGroup(Alignment.BASELINE)
						.addComponent(lblCostumerName, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
						.addComponent(txtFirstName, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
						.addComponent(txtLastName, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
					.addGap(17)
					.addGroup(gl_panel.createParallelGroup(Alignment.BASELINE)
```

```
                                        .addComponent(lblCostumerContactNo,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(txtCusContactNo, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                .addGap(57)
                                .addComponent(lblCusconformation)
                                .addGap(251)
                                .addGroup(gl_panel.createParallelGroup(Alignment.BASELINE, false)
                                        .addComponent(btnReset)
                                        .addGroup(gl_panel.createSequentialGroup()
                                                .addGap(8)
                                                .addComponent(lblCustomerid))
                                        .addComponent(btnSubmit))
                                .addGap(61))
                );
                panel.setLayout(gl_panel);
                JPanel panel_1 = new JPanel();
                tabbedPane.addTab("Bill Calculator", null, panel_1, null);

                lblCustomerId_1 = new JLabel("Customer ID_1");
                lblCustomerId_1.setHorizontalAlignment(SwingConstants.CENTER);
                lblCustomerId_1.setVerticalAlignment(SwingConstants.TOP);
                lblCustomerId_1.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));
                lblCustomerId_1.setVisible(false);

                lblVegetable = new JLabel("Vegetables :");
                lblVegetable.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));

                Statement st = null;
                try
                {
                        st = con.createStatement();
                        String sql = "SELECT *FROM Vegetables";
                        ResultSet rs = st.executeQuery(sql);
                        String str=null;
                        ArrayList<String> arr =new ArrayList<>();
        while (rs.next()) {
                str = rs.getString("Vegetables");
                                arr.add(str);
                        }
        comboBox = new JComboBox<Object>(arr.toArray());
                comboBox.setFont(new Font("Cambria", Font.ITALIC, 15));
        }catch (Exception e) {
                        System.out.println(e);
                        }
                   finally {
                        st.close();
                }
                btnAddVegetable = new JButton("Add Vegetable");
                btnAddVegetable.setFont(new Font("Cambria", Font.BOLD, 15));
                textField = new JTextField();
                textField.setFont(new Font("Cambria", Font.BOLD, 15));
                textField.setColumns(10);
                lblFruits = new JLabel("Fruits : ");
                lblFruits.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));

                Statement st_1 = null;
                try
                {
                        st_1 = con.createStatement();
                        String sql = "SELECT *FROM Fruits";
                        ResultSet rs_1 = st_1.executeQuery(sql);
                        String str_1 = null;
                        ArrayList<String> arr_1 =new ArrayList<>();
        while (rs_1.next()) {
                str_1 = rs_1.getString("Fruits");
                                arr_1.add(str_1);
                        }
        comboBox_1 = new JComboBox<Object>(arr_1.toArray());
                comboBox_1.setFont(new Font("Cambria", Font.ITALIC, 15));
        }catch (Exception e) {
                        System.out.println(e);
```

```
                }
            finally {
                st_1.close();
        }


        textField_1 = new JTextField();
        textField_1.setFont(new Font("Cambria", Font.BOLD, 15));
        textField_1.setColumns(10);

        btnAddFruit = new JButton("Add Fruit");
        btnAddFruit.setFont(new Font("Cambria", Font.BOLD, 15));

        Statement st_2 = null;
        try
        {
                st_2 = con.createStatement();
                String sql = "SELECT *FROM Kitchen";
                ResultSet rs_2 = st_2.executeQuery(sql);
                String str_2 = null;
                ArrayList<String> arr_2 =new ArrayList<>();
    while (rs_2.next()) {
        str_2 = rs_2.getString("Items");
                        arr_2.add(str_2);
                }
    comboBox_2 = new JComboBox<Object>(arr_2.toArray());
        comboBox_2.setFont(new Font("Cambria", Font.ITALIC, 15));
    }catch (Exception e) {
                System.out.println(e);
                }
            finally {
                st_2.close();
        }


        lblPulsesAndSpices = new JLabel("Pulses And Spices :");
        lblPulsesAndSpices.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));

        textField_2 = new JTextField();
        textField_2.setFont(new Font("Cambria", Font.BOLD, 15));
        textField_2.setColumns(10);

        btnAddItem = new JButton("Add Item");
        btnAddItem.setFont(new Font("Cambria", Font.BOLD, 15));

        lblStationary = new JLabel("Stationary : ");
        lblStationary.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));

        Statement st_3 = null;
        try
        {
                st_3 = con.createStatement();
                String sql = "SELECT *FROM Stationary";
                ResultSet rs_3 = st_3.executeQuery(sql);
                String str_3 = null;
                ArrayList<String> arr_3 =new ArrayList<>();
    while (rs_3.next()) {
        str_3 = rs_3.getString("Items");
                        arr_3.add(str_3);
                }
    comboBox_3 = new JComboBox(arr_3.toArray());
        comboBox_3.setFont(new Font("Cambria", Font.ITALIC, 15));
    }catch (Exception e) {
                System.out.println(e);
                }
            finally {
                st_3.close();
        }

        textField_3 = new JTextField();
        textField_3.setFont(new Font("Cambria", Font.BOLD, 15));
        textField_3.setColumns(10);
```

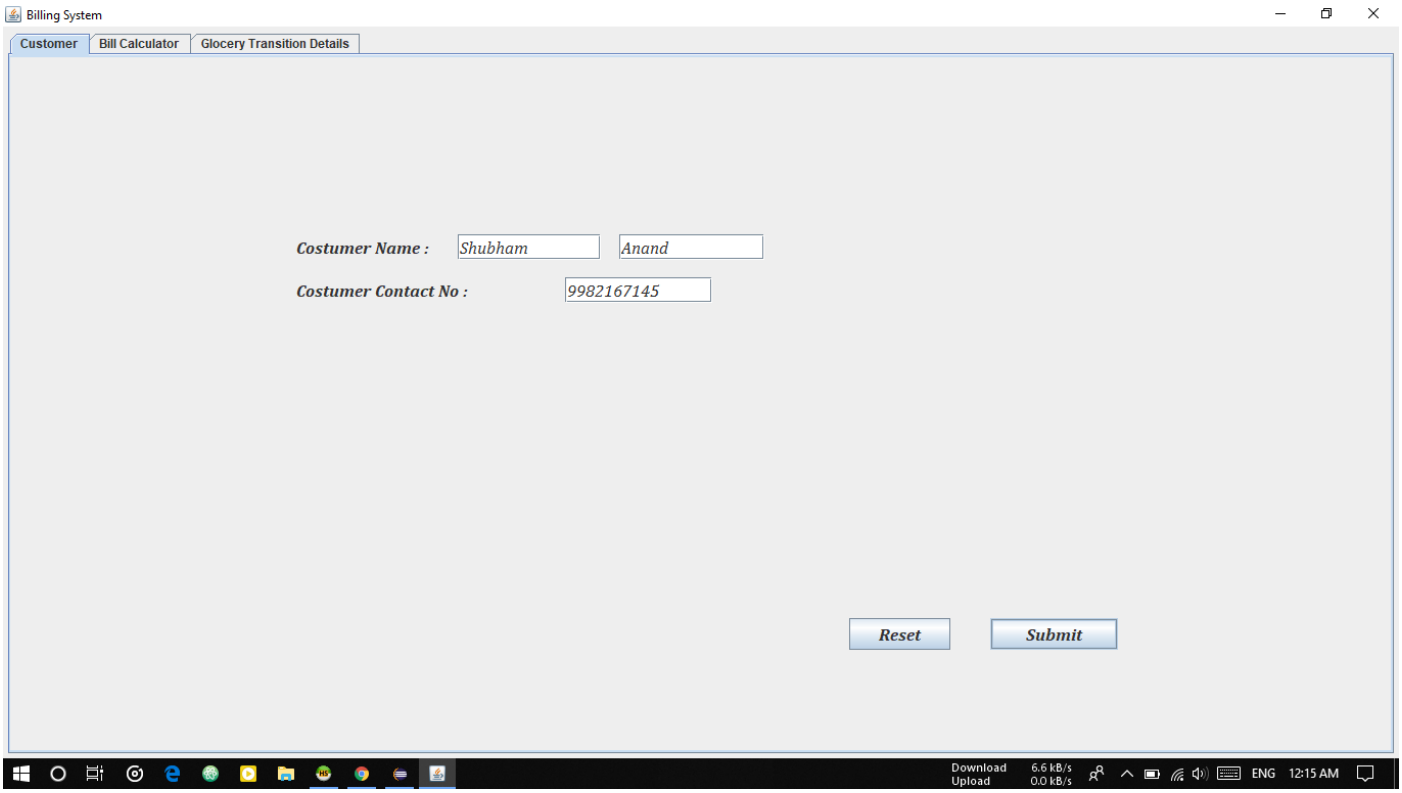
```
		btnAddItem_1 = new JButton("Add Item");
		btnAddItem_1.setFont(new Font("Cambria", Font.BOLD, 15));

		lblOthers = new JLabel("Others : ");
		lblOthers.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));

		textField_4 = new JTextField();
		textField_4.setFont(new Font("Cambria", Font.BOLD, 15));
		textField_4.setColumns(10);

		Statement st_4 = null;
		try
		{
			st_4 = con.createStatement();
			String sql = "SELECT *FROM Others";
			ResultSet rs_4 = st_4.executeQuery(sql);
			String str_4 = null;
			ArrayList<String> arr_4 =new ArrayList<>();
	while (rs_4.next()) {
		str_4 = rs_4.getString("Items");
				arr_4.add(str_4);
			}
	comboBox_4 = new JComboBox(arr_4.toArray());
		comboBox_4.setFont(new Font("Cambria", Font.ITALIC, 15));
	}catch (Exception e) {
			System.out.println(e);
			}
		  finally {
			st_4.close();
		}

		btnAddItem_2 = new JButton("Add Item");
		btnAddItem_2.setFont(new Font("Cambria", Font.BOLD, 15));

		btnSubmit_1 = new JButton("Submit");
		btnSubmit_1.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));

		lblBillID = new JLabel("BillID");
		lblBillID.setHorizontalAlignment(SwingConstants.CENTER);
		lblBillID.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));
		lblBillID.setVisible(false);

		btnTotal = new JButton("Total");
		btnTotal.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));

		lblGrandTotal = new JLabel("Grand Total");
		lblGrandTotal.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 15));
		lblGrandTotal.setVisible(false);

		scrollPane = new JScrollPane();

		lblSubmitedToBill = new JLabel("Submited to Bill Database");
		lblSubmitedToBill.setFont(new Font("Cambria", Font.ITALIC, 17));
		lblSubmitedToBill.setVisible(false);

		GroupLayout gl_panel_1 = new GroupLayout(panel_1);
		gl_panel_1.setHorizontalGroup(
			gl_panel_1.createParallelGroup(Alignment.LEADING)
				.addGroup(gl_panel_1.createSequentialGroup()
					.addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING)
						.addGroup(gl_panel_1.createSequentialGroup()
							.addGap(463)
							.addComponent(lblSubmitedToBill,
GroupLayout.PREFERRED_SIZE, 205, GroupLayout.PREFERRED_SIZE)
							.addGap(77)
							.addComponent(lblGrandTotal,
GroupLayout.PREFERRED_SIZE, 198, GroupLayout.PREFERRED_SIZE)
							.addGap(83)
							.addComponent(btnTotal)
							.addGap(62)
							.addComponent(btnSubmit_1))
```

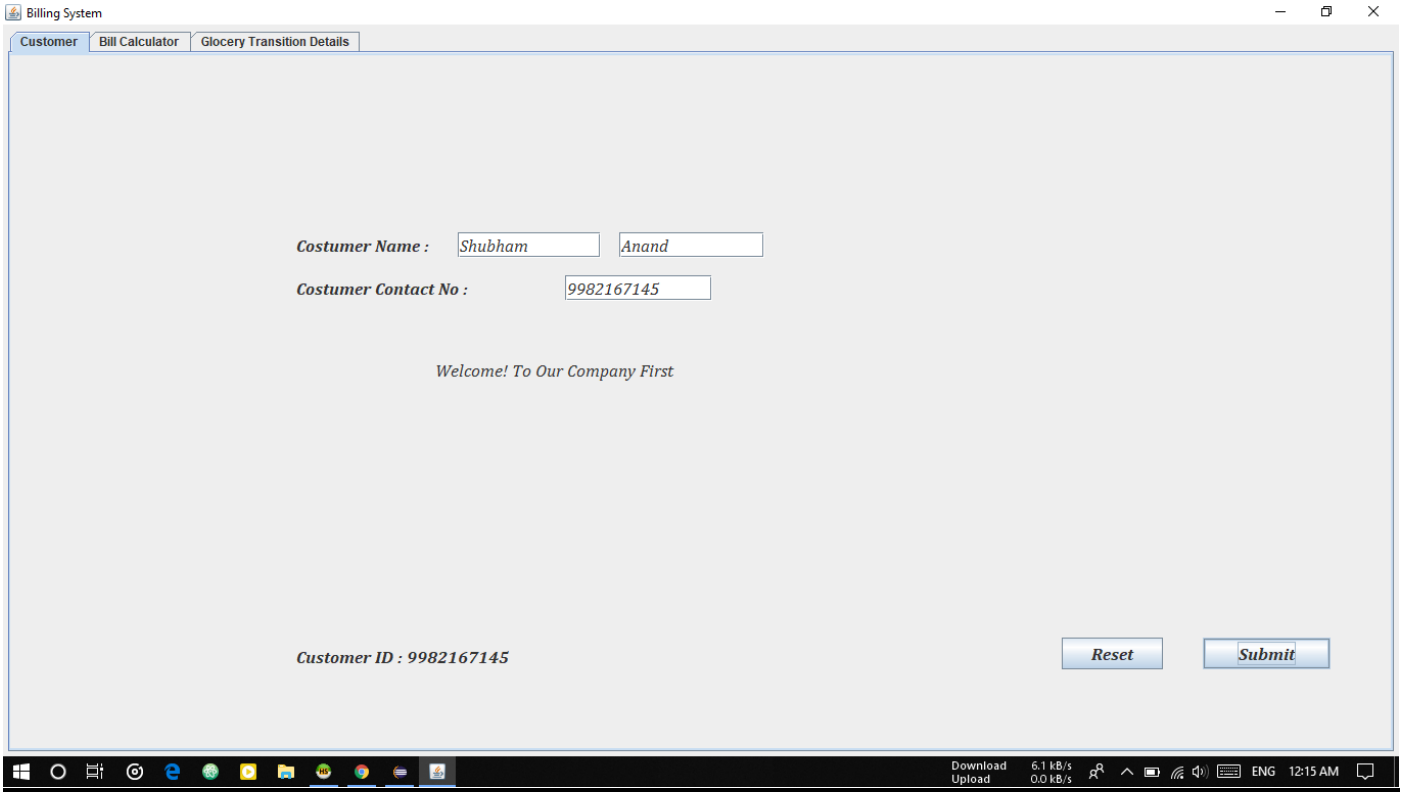
```
                                        .addGroup(gl_panel_1.createSequentialGroup()
                                                .addGap(270)

        .addGroup(gl_panel_1.createParallelGroup(Alignment.TRAILING)
                                                .addComponent(lblCustomerId_1,
GroupLayout.PREFERRED_SIZE, 301, GroupLayout.PREFERRED_SIZE)
                                                .addGroup(gl_panel_1.createSequentialGroup()

        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING)
                                                        .addComponent(lblVegetable,
GroupLayout.PREFERRED_SIZE, 88, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(lblFruits,
GroupLayout.PREFERRED_SIZE, 62, GroupLayout.PREFERRED_SIZE)

        .addComponent(lblPulsesAndSpices)
                                                        .addComponent(lblStationary)
                                                        .addComponent(lblOthers))
                                                .addGap(53)

        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING, false)
                                                        .addComponent(comboBox_4, 0,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(comboBox_3, 0,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(comboBox_2,
Alignment.TRAILING, 0, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(comboBox_1, 0,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(comboBox,
GroupLayout.PREFERRED_SIZE, 169, GroupLayout.PREFERRED_SIZE))))

        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING)
                                                .addGroup(gl_panel_1.createSequentialGroup()
                                                        .addGap(54)

        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING, false)
                                                        .addComponent(textField_4, 0, 0,
Short.MAX_VALUE)
                                                        .addComponent(textField_3, 0, 0,
Short.MAX_VALUE)
                                                        .addComponent(textField_2, 0, 0,
Short.MAX_VALUE)
                                                        .addComponent(textField_1, 0, 0,
Short.MAX_VALUE)
                                                        .addComponent(textField,
GroupLayout.PREFERRED_SIZE, 76, GroupLayout.PREFERRED_SIZE))
                                                .addGap(88)

        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING, false)
                                                        .addComponent(btnAddItem_2,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(btnAddItem_1,
Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(btnAddItem,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(btnAddFruit,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(btnAddVegetable)))
                                                .addGroup(gl_panel_1.createSequentialGroup()
                                                        .addGap(36)
                                                        .addComponent(lblBillID,
GroupLayout.PREFERRED_SIZE, 281, GroupLayout.PREFERRED_SIZE))))
                                        .addGroup(gl_panel_1.createSequentialGroup()
                                                .addGap(28)
                                                .addComponent(scrollPane,
GroupLayout.PREFERRED_SIZE, 1270, GroupLayout.PREFERRED_SIZE)))
                                .addContainerGap(37, Short.MAX_VALUE))
                );
                gl_panel_1.setVerticalGroup(
                        gl_panel_1.createParallelGroup(Alignment.LEADING)
                                .addGroup(gl_panel_1.createSequentialGroup()
```

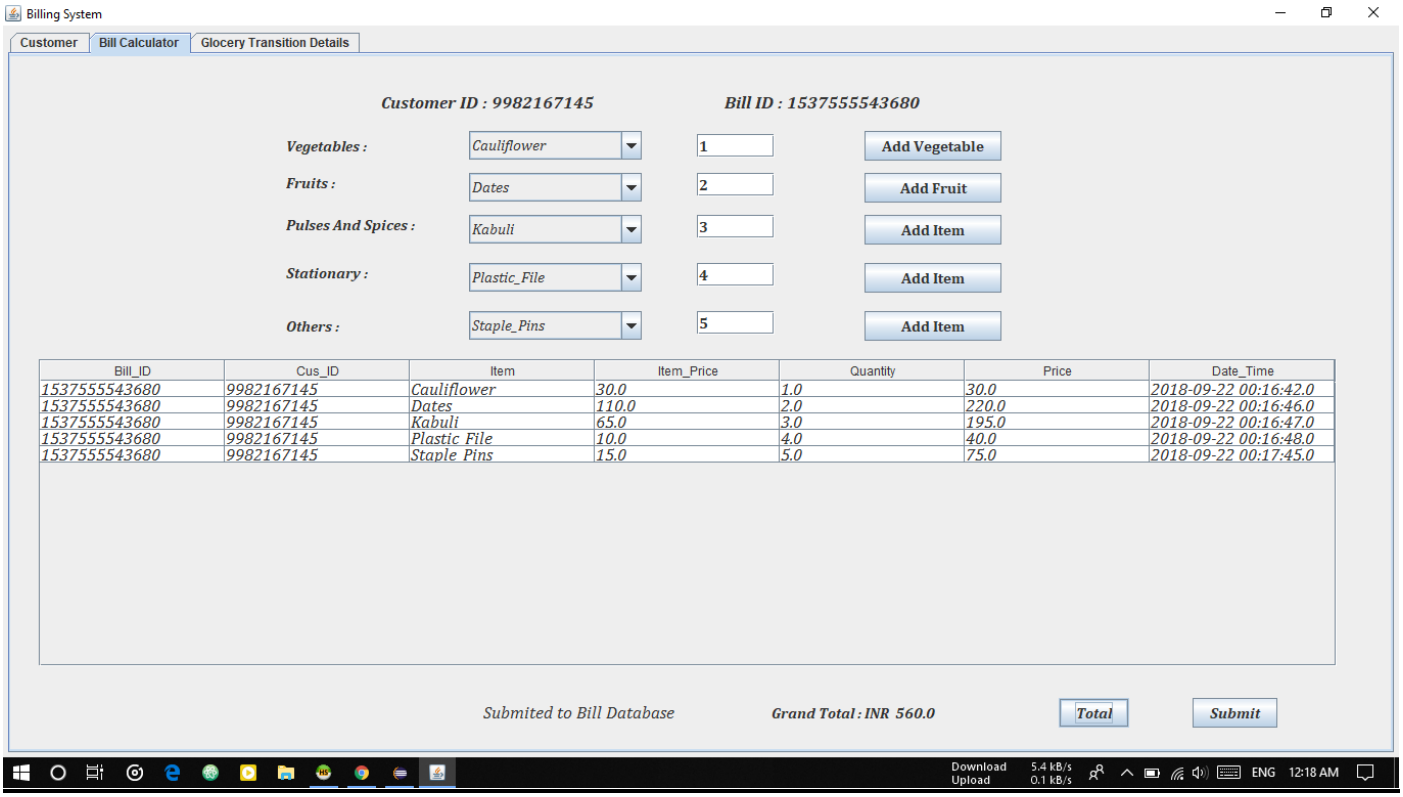
```
                                        .addGap(34)
                                        .addGroup(gl_panel_1.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(lblCustomerId_1)
                                                .addComponent(lblBillID))
                                        .addGap(18)
                                        .addGroup(gl_panel_1.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(lblVegetable)
                                                .addComponent(btnAddVegetable)
                                                .addComponent(textField, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addComponent(comboBox, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                        .addPreferredGap(ComponentPlacement.UNRELATED)
                                        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING)
                                                .addGroup(gl_panel_1.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(comboBox_1,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(textField_1,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(btnAddFruit))
                                                .addComponent(lblFruits))
                                        .addPreferredGap(ComponentPlacement.UNRELATED)
                                        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING)
                                                .addComponent(lblPulsesAndSpices)
                                                .addComponent(comboBox_2, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addGroup(gl_panel_1.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(textField_2,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(btnAddItem)))
                                        .addGap(18)
                                        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING)
                                                .addComponent(lblStationary)
                                                .addComponent(textField_3, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addComponent(comboBox_3, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addComponent(btnAddItem_1))
                                        .addGap(18)
                                        .addGroup(gl_panel_1.createParallelGroup(Alignment.LEADING)
                                                .addGroup(gl_panel_1.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(lblOthers)
                                                        .addComponent(textField_4,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(comboBox_4,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                                .addComponent(btnAddItem_2))
                                        .addGap(18)
                                        .addComponent(scrollPane, GroupLayout.PREFERRED_SIZE, 300,
GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(ComponentPlacement.RELATED, 31, Short.MAX_VALUE)
                                        .addGroup(gl_panel_1.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(lblSubmitedToBill)
                                                .addComponent(lblGrandTotal)
                                                .addComponent(btnTotal)
                                                .addComponent(btnSubmit_1))
                                        .addGap(21))
                );

                table = new JTable();
                table.setFont(new Font("Cambria", Font.ITALIC, 17));
                scrollPane.setViewportView(table);

                panel_1.setLayout(gl_panel_1);

                JPanel panel_2 = new JPanel();
                tabbedPane.addTab("Glocery Transition Details", null, panel_2, null);

                JLabel lblSearchcus = new JLabel("Customer ID :");
                lblSearchcus.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));

                textField_5 = new JTextField();
```

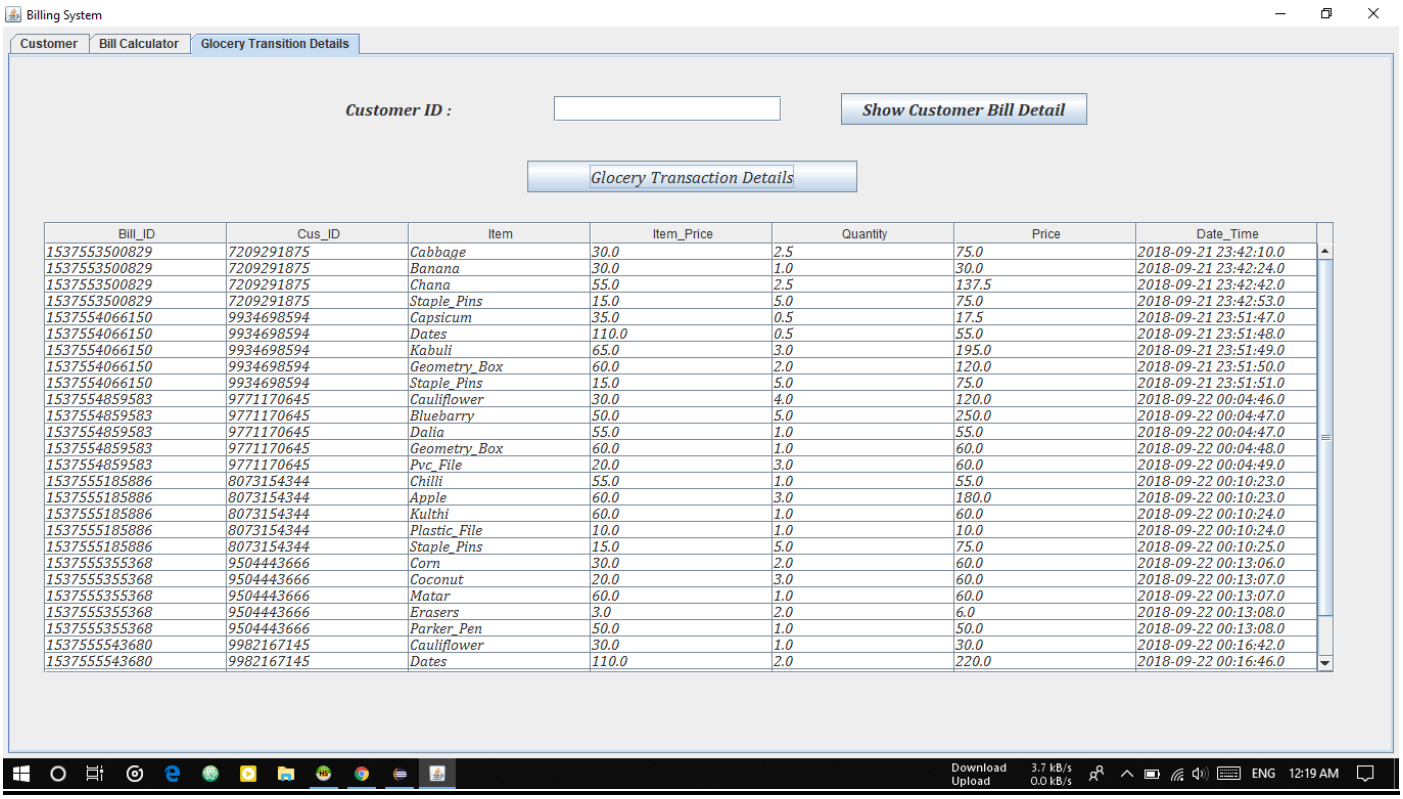
```
                textField_5.setFont(new Font("Cambria", Font.ITALIC, 17));
                textField_5.setColumns(10);

                btnSearchCustomer = new JButton("Show Customer Bill Detail");

                btnSearchCustomer.setFont(new Font("Cambria", Font.BOLD | Font.ITALIC, 17));

                btnGloceryTransactionDetails = new JButton("Glocery Transaction Details");

                btnGloceryTransactionDetails.setFont(new Font("Cambria", Font.ITALIC, 17));

                JScrollPane scrollPane_1 = new JScrollPane();

                GroupLayout gl_panel_2 = new GroupLayout(panel_2);
                gl_panel_2.setHorizontalGroup(
                        gl_panel_2.createParallelGroup(Alignment.LEADING)
                                .addGroup(gl_panel_2.createSequentialGroup()
                                        .addGroup(gl_panel_2.createParallelGroup(Alignment.LEADING)
                                                .addGroup(gl_panel_2.createSequentialGroup()
                                                        .addGap(33)
                                                        .addComponent(scrollPane_1,
GroupLayout.PREFERRED_SIZE, 1263, GroupLayout.PREFERRED_SIZE))
                                                .addGroup(gl_panel_2.createSequentialGroup()
                                                        .addGap(328)
                                                        .addComponent(lblSearchcus,
GroupLayout.PREFERRED_SIZE, 127, GroupLayout.PREFERRED_SIZE)
                                                        .addGap(77)
                                                        .addComponent(textField_5,
GroupLayout.PREFERRED_SIZE, 223, GroupLayout.PREFERRED_SIZE)
                                                        .addGap(58)
                                                        .addComponent(btnSearchCustomer,
GroupLayout.PREFERRED_SIZE, 241, GroupLayout.PREFERRED_SIZE))
                                                .addGroup(gl_panel_2.createSequentialGroup()
                                                        .addGap(506)
                                                        .addComponent(btnGloceryTransactionDetails,
GroupLayout.PREFERRED_SIZE, 323, GroupLayout.PREFERRED_SIZE)))
                                        .addContainerGap(39, Short.MAX_VALUE))
                );
                gl_panel_2.setVerticalGroup(
                        gl_panel_2.createParallelGroup(Alignment.LEADING)
                                .addGroup(gl_panel_2.createSequentialGroup()
                                        .addGap(35)
                                        .addGroup(gl_panel_2.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(textField_5, GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addComponent(btnSearchCustomer)
                                                .addComponent(lblSearchcus))
                                        .addGap(35)
                                        .addComponent(btnGloceryTransactionDetails)
                                        .addGap(29)
                                        .addComponent(scrollPane_1, GroupLayout.PREFERRED_SIZE, 441,
GroupLayout.PREFERRED_SIZE)
                                        .addContainerGap(67, Short.MAX_VALUE))
                );

                table_1 = new JTable();
                table_1.setFont(new Font("Cambria", Font.ITALIC, 15));
                scrollPane_1.setViewportView(table_1);
                panel_2.setLayout(gl_panel_2);
        }
    }
```

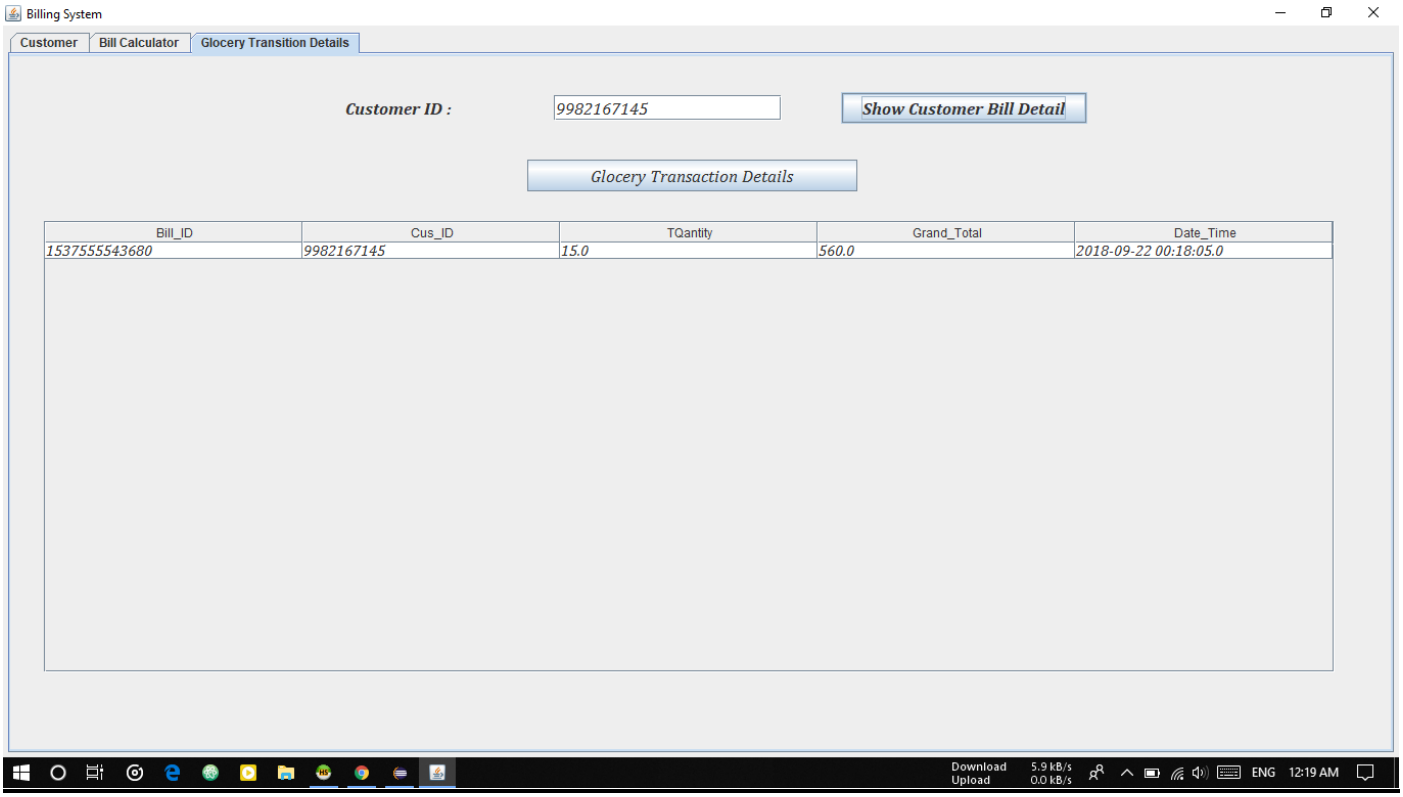
OUTPUT

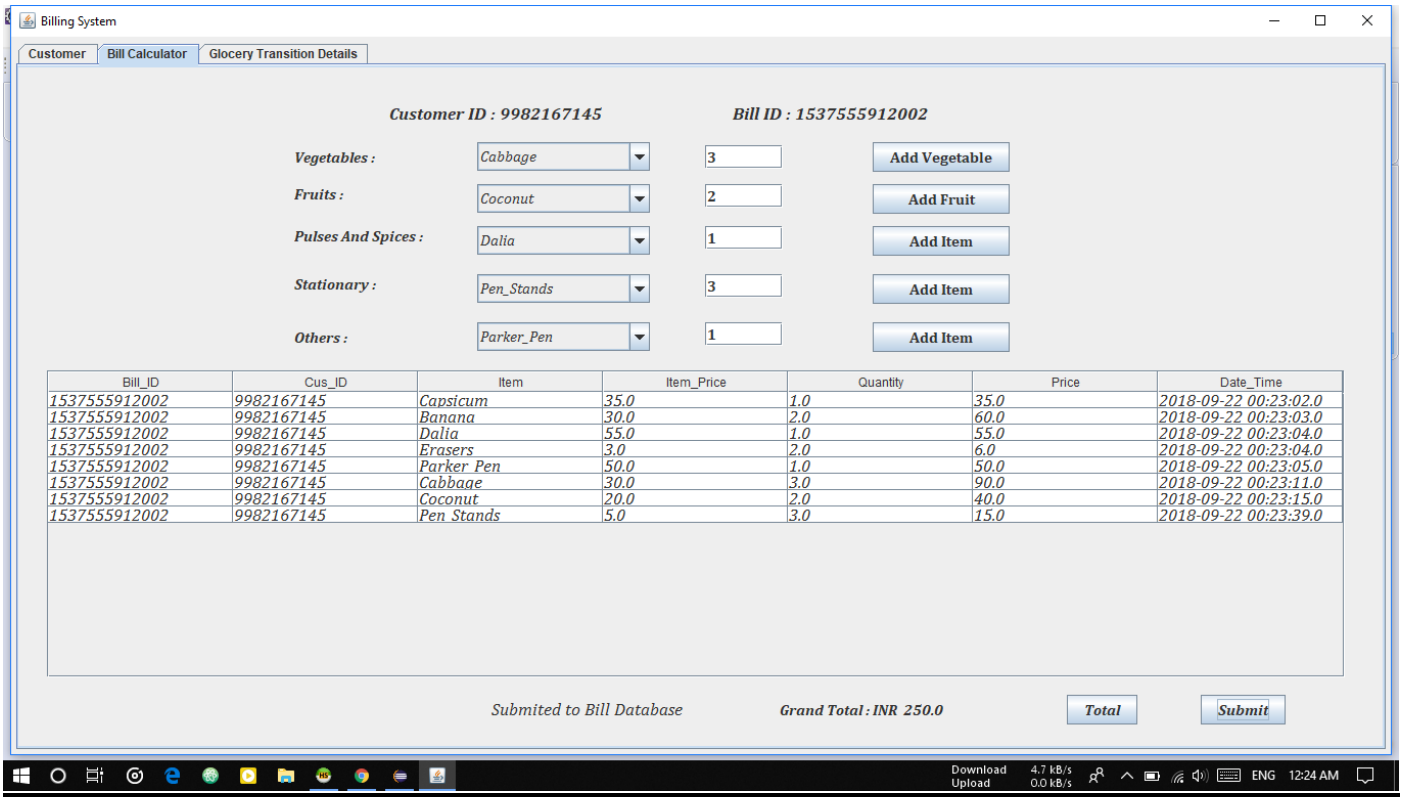

Billing System
Customer
Bill Calculator
Glocery Transition Details
Costumer Name :
Shubham
Anand
Costumer Contact No :
9982167145
Reset
Submit
Download
Upload
6.6 kB/s
0.0 kB/s
ENG
12:15 AM

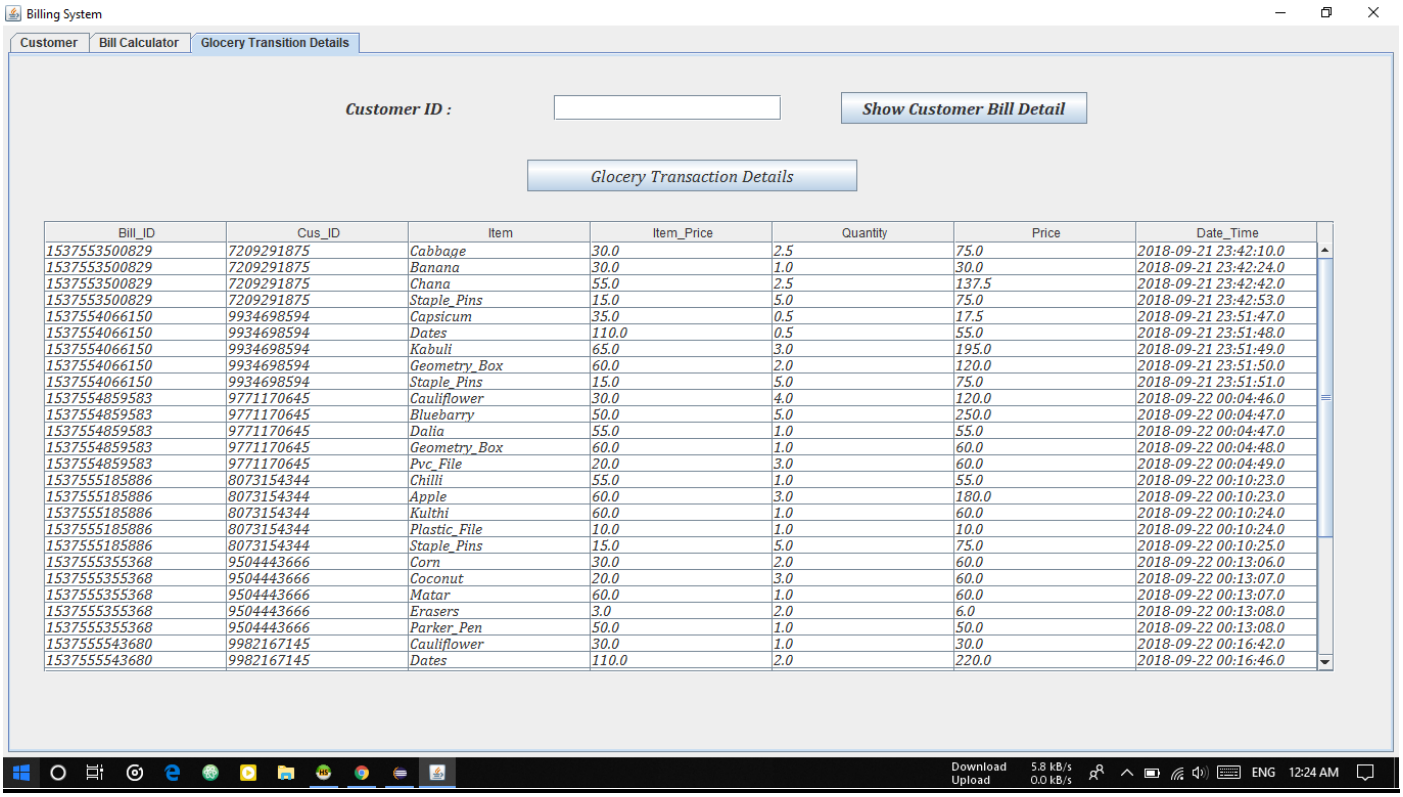

Billing System
Customer
Bill Calculator
Glocery Transition Details
Costumer Name :
Shubham
Anand
Costumer Contact No :
9982167145
Welcome! To Our Company First
Customer ID : 9982167145
Reset
Submit
Download
Upload
6.1 kB/s
0.0 kB/s
ENG
12:15 AM


Billing System
Customer
Bill Calculator
Glocery Transition Details
Customer ID : 9982167145
Bill ID : 1537555543680
Vegetables :
Cauliflower
1
Add Vegetable
Fruits :
Dates
2
Add Fruit
Pulses And Spices :
Kabuli
3
Add Item
Stationary :
Plastic_File
4
Add Item
Others :
Staple_Pins
5
Add Item
Bill_ID Cus_ID Item Item_Price Quantity Price Date_Time
1537555543680 9982167145 Cauliflower 30.0 1.0 30.0 2018-09-22 00:16:42.0
1537555543680 9982167145 Dates 110.0 2.0 220.0 2018-09-22 00:16:46.0
1537555543680 9982167145 Kabuli 65.0 3.0 195.0 2018-09-22 00:16:47.0
1537555543680 9982167145 Plastic_File 10.0 4.0 40.0 2018-09-22 00:16:48.0
1537555543680 9982167145 Staple_Pins 15.0 5.0 75.0 2018-09-22 00:17:45.0
Submited to Bill Database
Grand Total : INR 560.0
Total
Submit
Download 5.4 kB/s
Upload 0.1 kB/s
ENG 12:18 AM

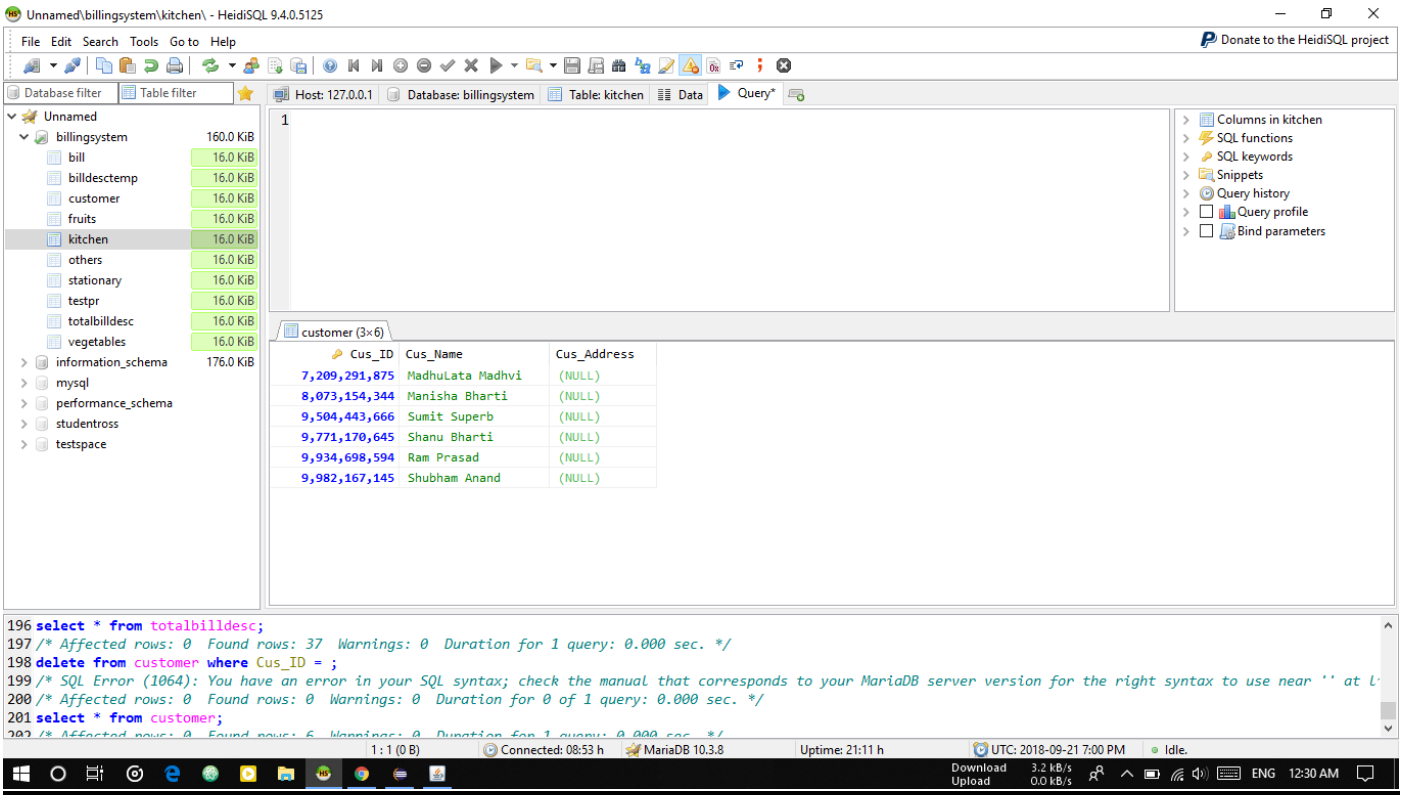

Billing System
Customer
Bill Calculator
Glocery Transition Details
Customer ID :
Show Customer Bill Detail
Glocery Transaction Details
Bill_ID Cus_ID Item Item_Price Quantity Price Date_Time
1537553500829 7209291875 Cabbage 30.0 2.5 75.0 2018-09-21 23:42:10.0
1537553500829 7209291875 Banana 30.0 1.0 30.0 2018-09-21 23:42:24.0
1537553500829 7209291875 Chana 55.0 2.5 137.5 2018-09-21 23:42:42.0
1537553500829 7209291875 Staple_Pins 15.0 5.0 75.0 2018-09-21 23:42:53.0
1537554066150 9934698594 Capsicum 35.0 0.5 17.5 2018-09-21 23:51:47.0
1537554066150 9934698594 Dates 110.0 0.5 55.0 2018-09-21 23:51:48.0
1537554066150 9934698594 Kabuli 65.0 3.0 195.0 2018-09-21 23:51:49.0
1537554066150 9934698594 Geometry_Box 60.0 2.0 120.0 2018-09-21 23:51:50.0
1537554066150 9934698594 Staple_Pins 15.0 5.0 75.0 2018-09-21 23:51:51.0
1537554859583 9771170645 Cauliflower 30.0 4.0 120.0 2018-09-22 00:04:46.0
1537554859583 9771170645 Bluebarry 50.0 5.0 250.0 2018-09-22 00:04:47.0
1537554859583 9771170645 Dalia 55.0 1.0 55.0 2018-09-22 00:04:47.0
1537554859583 9771170645 Geometry_Box 60.0 1.0 60.0 2018-09-22 00:04:48.0
1537554859583 9771170645 Pvc_File 20.0 3.0 60.0 2018-09-22 00:04:49.0
1537555185886 8073154344 Chilli 55.0 1.0 55.0 2018-09-22 00:10:23.0
1537555185886 8073154344 Apple 60.0 3.0 180.0 2018-09-22 00:10:23.0
1537555185886 8073154344 Kulthi 60.0 1.0 60.0 2018-09-22 00:10:24.0
1537555185886 8073154344 Plastic_File 10.0 1.0 10.0 2018-09-22 00:10:24.0
1537555185886 8073154344 Staple_Pins 15.0 5.0 75.0 2018-09-22 00:10:25.0
1537555355368 9504443666 Corn 30.0 2.0 60.0 2018-09-22 00:13:06.0
1537555355368 9504443666 Coconut 20.0 3.0 60.0 2018-09-22 00:13:07.0
1537555355368 9504443666 Matar 60.0 1.0 60.0 2018-09-22 00:13:07.0
1537555355368 9504443666 Erasers 3.0 2.0 6.0 2018-09-22 00:13:08.0
1537555355368 9504443666 Parker_Pen 50.0 1.0 50.0 2018-09-22 00:13:08.0
1537555543680 9982167145 Cauliflower 30.0 1.0 30.0 2018-09-22 00:16:42.0
1537555543680 9982167145 Dates 110.0 2.0 220.0 2018-09-22 00:16:46.0
Download 3.7 kB/s
Upload 0.0 kB/s
ENG 12:19 AM

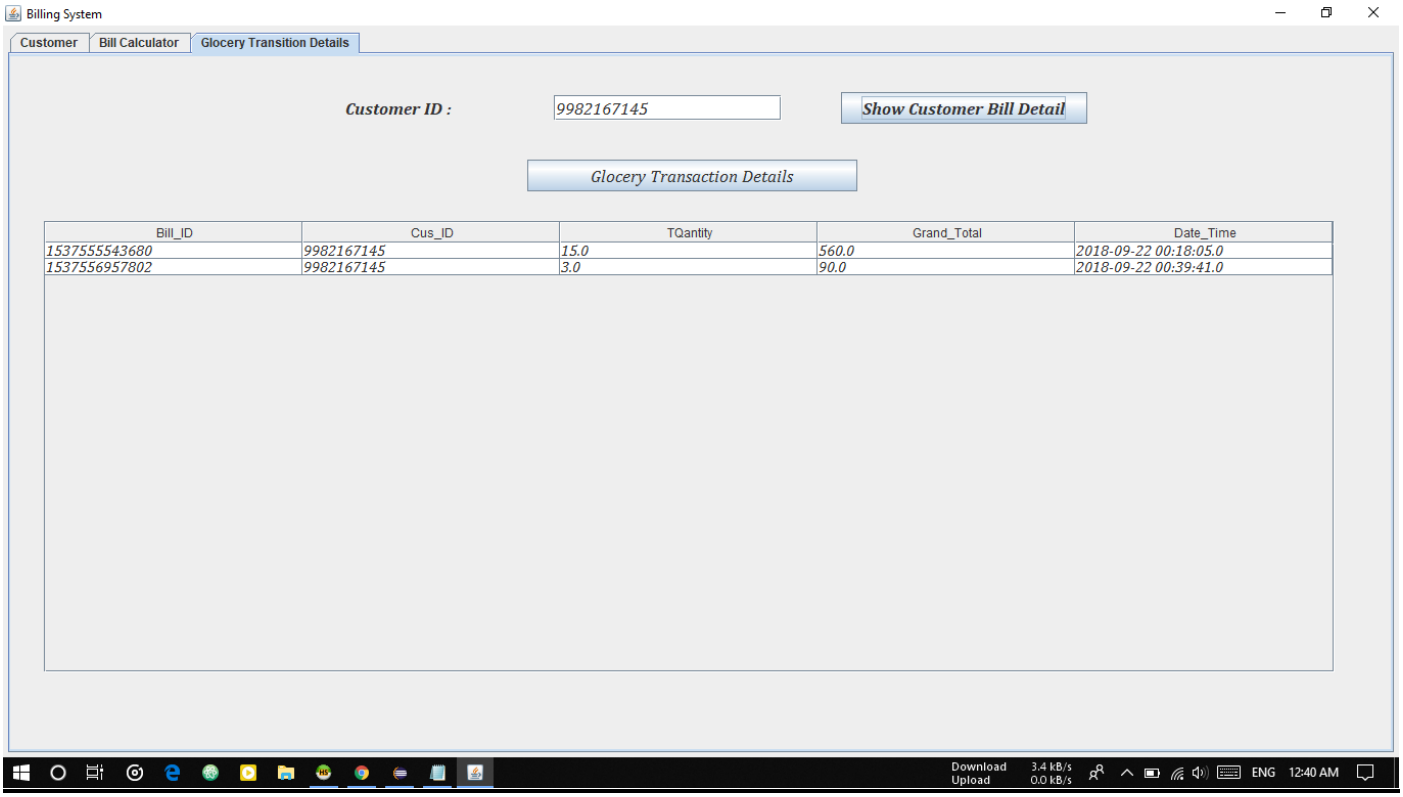

Billing System
Customer
Bill Calculator
Glocery Transition Details
Customer ID :
9982167145
Show Customer Bill Detail
Glocery Transaction Details
Bill_ID
Cus_ID
TQantity
Grand_Total
Date_Time
1537555543680
9982167145
15.0
560.0
2018-09-22 00:18:05.0
Download
Upload
5.9 kB/s
0.0 kB/s
ENG
12:19 AM

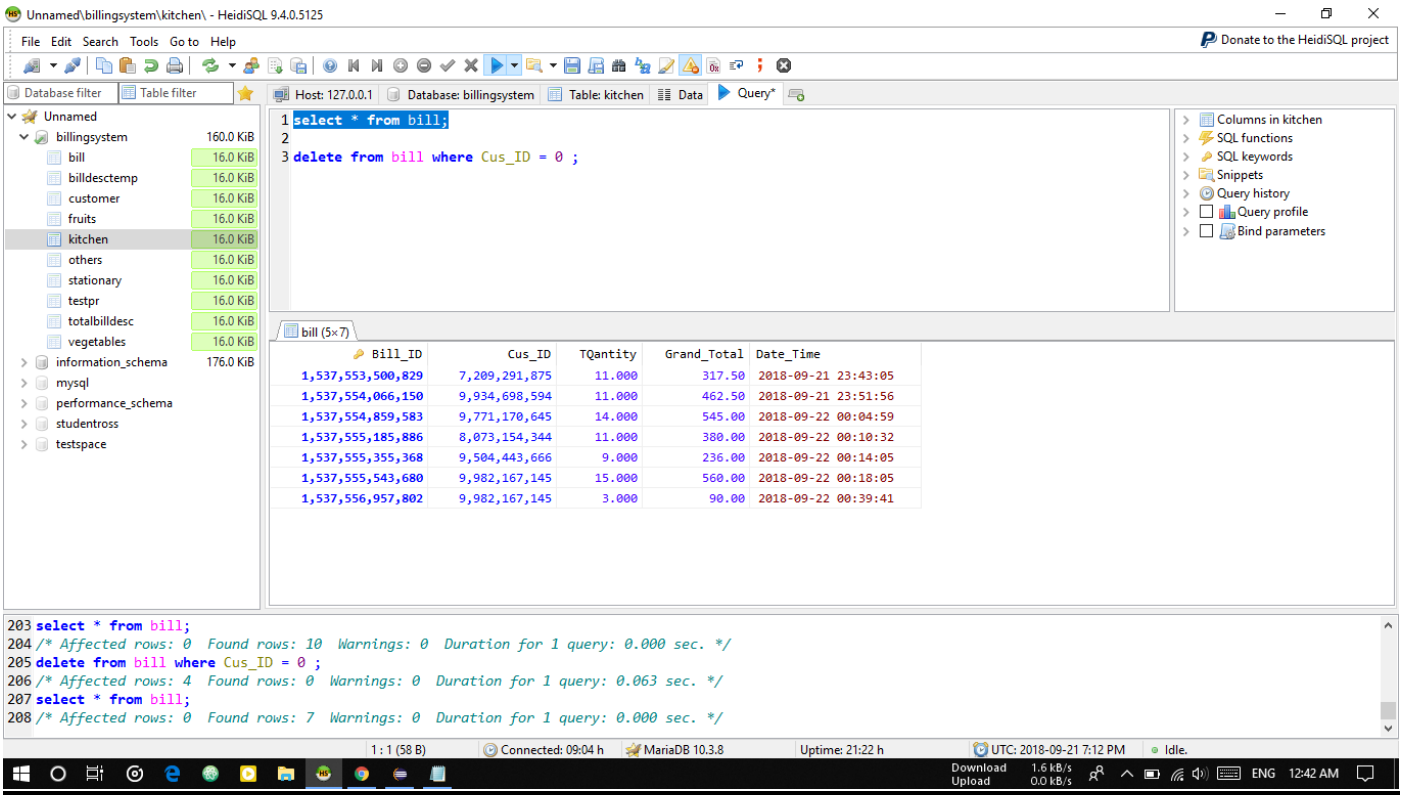

Billing System
Customer
Bill Calculator
Glocery Transition Details
Customer ID : 9982167145
Bill ID : 1537555912002
Vegetables :
Cabbage
3
Add Vegetable
Fruits :
Coconut
2
Add Fruit
Pulses And Spices :
Dalia
1
Add Item
Stationary :
Pen_Stands
3
Add Item
Others :
Parker_Pen
1
Add Item
Bill_ID
Cus_ID
Item
Item_Price
Quantity
Price
Date_Time
1537555912002 9982167145 Capsicum 35.0 1.0 35.0 2018-09-22 00:23:02.0
1537555912002 9982167145 Banana 30.0 2.0 60.0 2018-09-22 00:23:03.0
1537555912002 9982167145 Dalia 55.0 1.0 55.0 2018-09-22 00:23:04.0
1537555912002 9982167145 Erasers 3.0 2.0 6.0 2018-09-22 00:23:04.0
1537555912002 9982167145 Parker_Pen 50.0 1.0 50.0 2018-09-22 00:23:05.0
1537555912002 9982167145 Cabbage 30.0 3.0 90.0 2018-09-22 00:23:11.0
1537555912002 9982167145 Coconut 20.0 2.0 40.0 2018-09-22 00:23:15.0
1537555912002 9982167145 Pen_Stands 5.0 3.0 15.0 2018-09-22 00:23:39.0
Submited to Bill Database
Grand Total : INR 250.0
Total
Submit
Download
Upload
4.7 kB/s
0.0 kB/s
ENG
12:24 AM

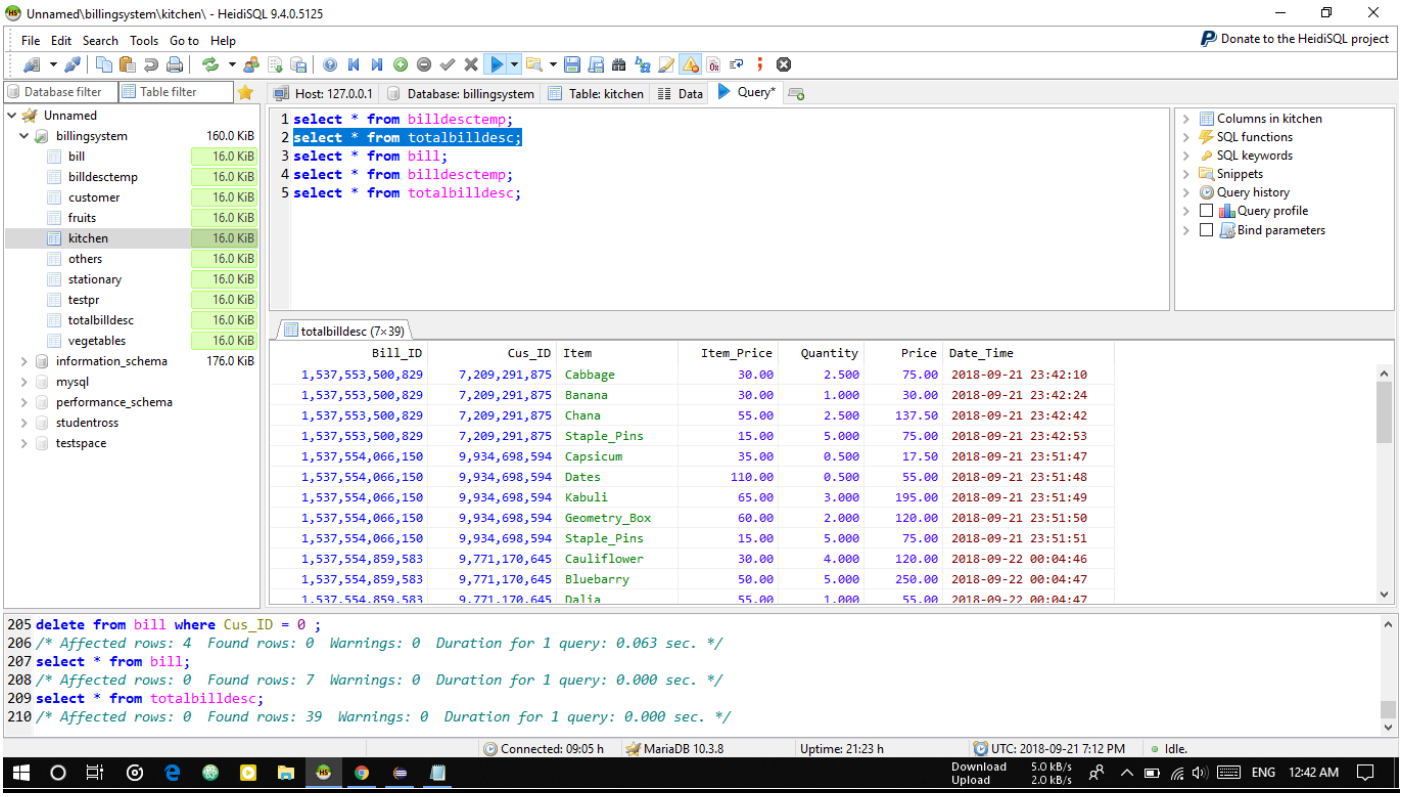


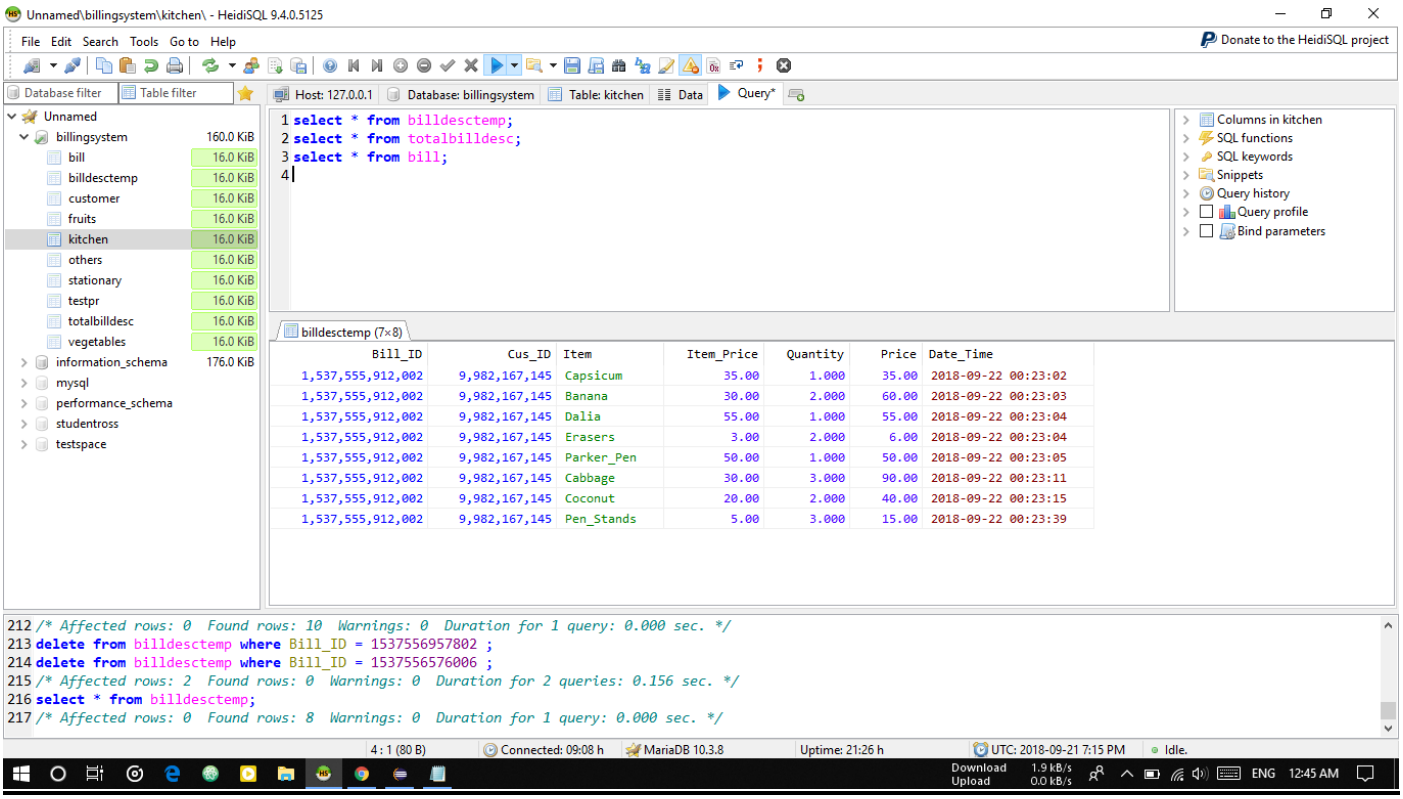
Bill_ID	Cus_ID	Item	Item_Price	Quantity	Price	Date_Time
1537553500829	7209291875	Cabbage	30.0	2.5	75.0	2018-09-21 23:42:10.0
1537553500829	7209291875	Banana	30.0	1.0	30.0	2018-09-21 23:42:24.0
1537553500829	7209291875	Chana	55.0	2.5	137.5	2018-09-21 23:42:42.0
1537553500829	7209291875	Staple_Pins	15.0	5.0	75.0	2018-09-21 23:42:53.0
1537554066150	9934698594	Capsicum	35.0	0.5	17.5	2018-09-21 23:51:47.0
1537554066150	9934698594	Dates	110.0	0.5	55.0	2018-09-21 23:51:48.0
1537554066150	9934698594	Kabuli	65.0	3.0	195.0	2018-09-21 23:51:49.0
1537554066150	9934698594	Geometry_Box	60.0	2.0	120.0	2018-09-21 23:51:50.0
1537554066150	9934698594	Staple_Pins	15.0	5.0	75.0	2018-09-21 23:51:51.0
1537554859583	9771170645	Cauliflower	30.0	4.0	120.0	2018-09-22 00:04:46.0
1537554859583	9771170645	Bluebarry	50.0	5.0	250.0	2018-09-22 00:04:47.0
1537554859583	9771170645	Dalia	55.0	1.0	55.0	2018-09-22 00:04:47.0
1537554859583	9771170645	Geometry_Box	60.0	1.0	60.0	2018-09-22 00:04:48.0
1537554859583	9771170645	Pvc_File	20.0	3.0	60.0	2018-09-22 00:04:49.0
1537555185886	8073154344	Chilli	55.0	1.0	55.0	2018-09-22 00:10:23.0
1537555185886	8073154344	Apple	60.0	3.0	180.0	2018-09-22 00:10:23.0
1537555185886	8073154344	Kulthi	60.0	1.0	60.0	2018-09-22 00:10:24.0
1537555185886	8073154344	Plastic_File	10.0	1.0	10.0	2018-09-22 00:10:24.0
1537555185886	8073154344	Staple_Pins	15.0	5.0	75.0	2018-09-22 00:10:25.0
1537555355368	9504443666	Corn	30.0	2.0	60.0	2018-09-22 00:13:06.0
1537555355368	9504443666	Coconut	20.0	3.0	60.0	2018-09-22 00:13:07.0
1537555355368	9504443666	Matar	60.0	1.0	60.0	2018-09-22 00:13:07.0
1537555355368	9504443666	Erasers	3.0	2.0	6.0	2018-09-22 00:13:08.0
1537555355368	9504443666	Parker_Pen	50.0	1.0	50.0	2018-09-22 00:13:08.0
1537555543680	9982167145	Cauliflower	30.0	1.0	30.0	2018-09-22 00:16:42.0
1537555543680	9982167145	Dates	110.0	2.0	220.0	2018-09-22 00:16:46.0




Billing System
Customer
Bill Calculator
Glocery Transition Details
Customer ID :
9982167145
Show Customer Bill Detail
Glocery Transaction Details
Bill_ID
Cus_ID
TQantity
Grand_Total
Date_Time
1537555543680
9982167145
15.0
560.0
2018-09-22 00:18:05.0
1537556957802
9982167145
3.0
90.0
2018-09-22 00:39:41.0
Download
Upload
3.4 kB/s
0.0 kB/s
ENG
12:40 AM


Unnamed\billingsystem\kitchen\ - HeidiSQL 9.4.0.5125
File Edit Search Tools Go to Help
Donate to the HeidiSQL project
Host: 127.0.0.1
Database: billingsystem
Table: kitchen
Data
Query*
Unnamed
billingsystem 160.0 KiB
bill 16.0 KiB
billdesctemp 16.0 KiB
customer 16.0 KiB
fruits 16.0 KiB
kitchen 16.0 KiB
others 16.0 KiB
stationary 16.0 KiB
testpr 16.0 KiB
totalbilldesc 16.0 KiB
vegetables 16.0 KiB
information_schema 176.0 KiB
mysql
performance_schema
studentross
testspace
1 select * from bill;
2
3 delete from bill where Cus_ID = 0 ;
Columns in kitchen
SQL functions
SQL keywords
Snippets
Query history
Query profile
Bind parameters
bill (5×7)
Bill_ID Cus_ID TQantity Grand_Total Date_Time
1,537,553,500,829 7,209,291,875 11.000 317.50 2018-09-21 23:43:05
1,537,554,066,150 9,934,698,594 11.000 462.50 2018-09-21 23:51:56
1,537,554,859,583 9,771,170,645 14.000 545.00 2018-09-22 00:04:59
1,537,555,185,886 8,073,154,344 11.000 380.00 2018-09-22 00:10:32
1,537,555,355,368 9,504,443,666 9.000 236.00 2018-09-22 00:14:05
1,537,555,543,680 9,982,167,145 15.000 560.00 2018-09-22 00:18:05
1,537,556,957,802 9,982,167,145 3.000 90.00 2018-09-22 00:39:41
203 select * from bill;
204 /* Affected rows: 0 Found rows: 10 Warnings: 0 Duration for 1 query: 0.000 sec. */
205 delete from bill where Cus_ID = 0 ;
206 /* Affected rows: 4 Found rows: 0 Warnings: 0 Duration for 1 query: 0.063 sec. */
207 select * from bill;
208 /* Affected rows: 0 Found rows: 7 Warnings: 0 Duration for 1 query: 0.000 sec. */
1 : 1 (58 B)
Connected: 09:04 h
MariaDB 10.3.8
Uptime: 21:22 h
UTC: 2018-09-21 7:12 PM
Idle.
Download
Upload
1.6 kB/s
0.0 kB/s
ENG
12:42 AM


Unnamed\billingsystem\kitchen\ - HeidiSQL 9.4.0.5125
File Edit Search Tools Go to Help
Donate to the HeidiSQL project
Database filter
Table filter
Host: 127.0.0.1
Database: billingsystem
Table: kitchen
Data
Query*
Unnamed
billingsystem 160.0 KiB
bill 16.0 KiB
billdesctemp 16.0 KiB
customer 16.0 KiB
fruits 16.0 KiB
kitchen 16.0 KiB
others 16.0 KiB
stationary 16.0 KiB
testpr 16.0 KiB
totalbilldesc 16.0 KiB
vegetables 16.0 KiB
information_schema 176.0 KiB
mysql
performance_schema
studentross
testspace
1 select * from billdesctemp;
2 select * from totalbilldesc;
3 select * from bill;
4 select * from billdesctemp;
5 select * from totalbilldesc;
Columns in kitchen
SQL functions
SQL keywords
Snippets
Query history
Query profile
Bind parameters
totalbilldesc (7×39)
Bill_ID Cus_ID Item Item_Price Quantity Price Date_Time
1,537,553,500,829 7,209,291,875 Cabbage 30.00 2.500 75.00 2018-09-21 23:42:10
1,537,553,500,829 7,209,291,875 Banana 30.00 1.000 30.00 2018-09-21 23:42:24
1,537,553,500,829 7,209,291,875 Chana 55.00 2.500 137.50 2018-09-21 23:42:42
1,537,553,500,829 7,209,291,875 Staple_Pins 15.00 5.000 75.00 2018-09-21 23:42:53
1,537,554,066,150 9,934,698,594 Capsicum 35.00 0.500 17.50 2018-09-21 23:51:47
1,537,554,066,150 9,934,698,594 Dates 110.00 0.500 55.00 2018-09-21 23:51:48
1,537,554,066,150 9,934,698,594 Kabuli 65.00 3.000 195.00 2018-09-21 23:51:49
1,537,554,066,150 9,934,698,594 Geometry_Box 60.00 2.000 120.00 2018-09-21 23:51:50
1,537,554,066,150 9,934,698,594 Staple_Pins 15.00 5.000 75.00 2018-09-21 23:51:51
1,537,554,859,583 9,771,170,645 Cauliflower 30.00 4.000 120.00 2018-09-22 00:04:46
1,537,554,859,583 9,771,170,645 Bluebarry 50.00 5.000 250.00 2018-09-22 00:04:47
1,537,554,859,583 9,771,170,645 Dalia 55.00 1.000 55.00 2018-09-22 00:04:47
205 delete from bill where Cus_ID = 0 ;
206 /* Affected rows: 4 Found rows: 0 Warnings: 0 Duration for 1 query: 0.063 sec. */
207 select * from bill;
208 /* Affected rows: 0 Found rows: 7 Warnings: 0 Duration for 1 query: 0.000 sec. */
209 select * from totalbilldesc;
210 /* Affected rows: 0 Found rows: 39 Warnings: 0 Duration for 1 query: 0.000 sec. */
Connected: 09:05 h
MariaDB 10.3.8
Uptime: 21:23 h
UTC: 2018-09-21 7:12 PM
Idle.
Download 5.0 kB/s
Upload 2.0 kB/s
ENG 12:42 AM


Unnamed\billingsystem\kitchen\ - HeidiSQL 9.4.0.5125
File Edit Search Tools Go to Help
Donate to the HeidiSQL project
Database filter
Table filter
Host: 127.0.0.1
Database: billingsystem
Table: kitchen
Data
Query*
Unnamed
billingsystem 160.0 KiB
bill 16.0 KiB
billdesctemp 16.0 KiB
customer 16.0 KiB
fruits 16.0 KiB
kitchen 16.0 KiB
others 16.0 KiB
stationary 16.0 KiB
testpr 16.0 KiB
totalbilldesc 16.0 KiB
vegetables 16.0 KiB
information_schema 176.0 KiB
mysql
performance_schema
studentross
testspace
1 select * from billdesctemp;
2 select * from totalbilldesc;
3 select * from bill;
4
Columns in kitchen
SQL functions
SQL keywords
Snippets
Query history
Query profile
Bind parameters
billdesctemp (7×8)
Bill_ID Cus_ID Item Item_Price Quantity Price Date_Time
1,537,555,912,002 9,982,167,145 Capsicum 35.00 1.000 35.00 2018-09-22 00:23:02
1,537,555,912,002 9,982,167,145 Banana 30.00 2.000 60.00 2018-09-22 00:23:03
1,537,555,912,002 9,982,167,145 Dalia 55.00 1.000 55.00 2018-09-22 00:23:04
1,537,555,912,002 9,982,167,145 Erasers 3.00 2.000 6.00 2018-09-22 00:23:04
1,537,555,912,002 9,982,167,145 Parker_Pen 50.00 1.000 50.00 2018-09-22 00:23:05
1,537,555,912,002 9,982,167,145 Cabbage 30.00 3.000 90.00 2018-09-22 00:23:11
1,537,555,912,002 9,982,167,145 Coconut 20.00 2.000 40.00 2018-09-22 00:23:15
1,537,555,912,002 9,982,167,145 Pen_Stands 5.00 3.000 15.00 2018-09-22 00:23:39
212 /* Affected rows: 0 Found rows: 10 Warnings: 0 Duration for 1 query: 0.000 sec. */
213 delete from billdesctemp where Bill_ID = 1537556957802 ;
214 delete from billdesctemp where Bill_ID = 1537556576006 ;
215 /* Affected rows: 2 Found rows: 0 Warnings: 0 Duration for 2 queries: 0.156 sec. */
216 select * from billdesctemp;
217 /* Affected rows: 0 Found rows: 8 Warnings: 0 Duration for 1 query: 0.000 sec. */
4 : 1 (80 B)
Connected: 09:08 h
MariaDB 10.3.8
Uptime: 21:26 h
UTC: 2018-09-21 7:15 PM
Idle.
Download 1.9 kB/s
Upload 0.0 kB/s
ENG 12:45 AM