A Project report on

INVOICE GENERATOR

Submitted to

IIMT Engineering College, MEERUT



(BCA V Sem, 2017-18)

(BCA-865 Project Based On Software Engineering)

**Submitted To: -** **Submitted By:-**

Anjali sharma Brihmanand-08658033

Antriksh shekho-08658021

**Chaudhary charan singh university, Uttar Pradesh, MEERUT**

**Acknowledgement**

The satisfaction that accompanies the successful completion of any task would be incomplete without acknowledging those who have made it possible and to those who’s constant encouragement and guidelines has been a source of inspiration throughout the course if this project.

We are thank **Mr. S.L.Kapoor**(Professor and the HOD of M.C.A) for his valuable guidance, and his constant supervision without which carrying out this project would not have been possible.

We are very much thankful to our internal Guide **Dr**. **Pratima Singh** for providing his valuable suggestion and guidance for the successful completion of this Project.

**ABSTRACT**

**INTRODUCTION:-**

Revenue management is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time. In practice RM has meant determining pricing according to predicted demand levels so that price sensitive customers who are willing to purchase at off-peak times can do so at favorable prices while price insensitive customers who want to purchase at peak time will be able to do so.

Every companies need an effective and accurate billing system to be able to assure their revenue. The billing process involves receiving billing records from various networks, determining the billing rates associated with the billing records, calculating the cost for each billing record, aggregating these records periodically to generate invoices, sending invoices to the customer, and collecting payments received from the customer.

**EXISTING PROJECT:-**

* The most problem is data storage. Because the revenue management system have more data.
* Existing system speed is slow.
* Manual work is large.
* Security is less.
* Corruption of data is largely possible.

**PROPOSED PROJCT:-**

* Minimizing the manual work.
* Maximize the security.
* Maximize the speed of system.
* Minimizing the task of corruption of data.

**Front –End:-**

* We are going to use Java as our Front-end.

**Back –End:-**

* We are going to use Oracle as our Back-end

FACULTY NAME:- GROUP MEMBER NAME (ROLL NUMBER):-

**Dr. Pratima singh** Sachin Kumar Sharma(1502714945)

Kritika Sharma(1502714921)

Rishabh Agarwal(1502714939)

**Table of Contents**

1. Introduction………………………………………………….6
2. Problem statement and description…………………………6-8
3. SRS…………………………………………………………..9-15
4. Software Design Document…………………………………16-21
5. Coding………………………………………………………..22-56
6. Test Result……………………………………………………57
7. Snapshot of forms……………………………………………57-60
8. Snapshot of report……………………………………………61-62

`List of table name

1. Case\_details………..……………………………………………………61
2. Registration………...……………………………………………………62
3. Login…………….………………………………………………………62
4. CustomerData..….………………………………………………………62
5. SenderDetails……………………………………………………………62
6. RecieverDetails…….……………………………………………………62
7. Payabledetails……...……………………………………………………62

List of Forms

1. Registration form……………………………………………………….57
2. Login form……………………………..……………………………….58
3. Customer Bill …………………………………………………………..58

List of Abbreviations

* **DB-** Data Base.
* **SRS-** Software Requirement Specification.
* **DFD-** Data Flow Diagram.

**1. INTRODUCTION**

**Product Overview**

Revenue management is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time. In practice RM has meant determining pricing according to predicted demand levels so that price sensitive customers who are willing to purchase at off-peak times can do so at favorable prices while price insensitive customers who want to purchase at peak time will be able to do so.

Every companies need an effective and accurate billing system to be able to assure their revenue. The billing process involves receiving billing records from various networks, determining the billing rates associated with the billing records, calculating the cost for each billing record, aggregating these records periodically to generate invoices, sending invoices to the customer, and collecting payments received from the customer.

**2. Problem Statement**

* Existing system speed is slow.
* The most problem is data storage. Because the revenue management system have more data.
* Manual work is large.
* Security is less.
* Corruption of data is largely possible.

**PROPOSED PROJCT:-**

* Minimizing the manual work.
* Maximize the security.
* Maximize the speed of system.
* Minimizing the task of corruption of data.

**Objective of the System:-**

* This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching.
* The searching of record has been made quite simple as all the details of the Customer details can be obtained by simply keying in the identification of the Customer.
* Similarly record maintenance and updation can also be accomplished by using the identification of the student with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.
* The objective is to prepare the system:
* The software gives information about the feedback of Customer.
* It also provides various utilities that the user can use.

**1.1 Purpose of the Study:-**

For the purpose of Defaulter students in college, Invoice Generator have collect the information of customer and manage them and arranges them according to various constraints. If any modification is required that is to be also done.

* 1. **Scope:-**
* Admin will login and can add customer details and generate reports according to individual customer name and customer tax id. or by branch as per the requirements.
* Admin can login and view report and can modify the existing record and submit the action taken against customer according to the type of case. He can also search the record with the help of different attributes like customer name. or can also search by transaction details.
  1. **Definition, Acronym & Abbreviation:-**
* **Java:** Java is a programming language expressly designed for use in the [distributed](http://searchcio-midmarket.techtarget.com/definition/distributed) environment of the Internet. It was designed to have the "look and feel" of the [C++](http://searchsqlserver.techtarget.com/definition/C) language, but it is simpler to use than C++ and enforces an [object-oriented programming](http://searchsoa.techtarget.com/definition/object-oriented-programming) model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or [applet](http://searchsoa.techtarget.com/definition/applet) for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.
* **Oracle:** Oracle is an open source relational database management system. It is based on the structure query language ([SQL](http://techterms.com/definition/sql)), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with Oracle.
* **OS:** An operating system (sometimes abbreviated as “OS”) is the program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs.

**Abbreviations:-**

* **DB-** Data Base.
* **SRS-** Software Requirement Specification.
* **DFD-** Data Flow Diagram.
  1. **References:-**
* [www.google.co.in](http://www.google.co.in)
* [www.javaTpoint.com](http://www.javaTpoint.com)
* [www.Oracle.com](http://www.Oracle.com)
  1. **Overview:-**

Revenue management is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time.

1. **Admin –**
2. Can access the record of the customer.
3. Can modify records of the Customer.
4. Can add details of new customer in database.
5. Can generate reports.

SRS

**INTRODUCTION**

**Product Overview**

Invoice Generator is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time. In practice RM has meant determining pricing according to predicted demand levels so that price sensitive customers who are willing to purchase at off-peak times can do so at favorable prices while price insensitive customers who want to purchase at peak time will be able to do so.

**Problem statement**

**Existing Project:-**

* Existing system speed is slow.
* The most problem is data storage. Because the revenue management system have more data.
* Manual work is large.
* Security is less.
* Corruption of data is largely possible.

**PROPOSED PROJCT:-**

* Minimizing the manual work.
* Maximize the security.
* Maximize the speed of system.
* Minimizing the task of corruption of data.

**Objective of the System:-**

* This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching.
* The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification of the student.
* Similarly record maintenance and updation can also be accomplished by using the identification of the student with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.
* The objective is to prepare the system:
* The software gives information about the feedback of student, etc.
* We can get the information about the performance of the student.
* It also provides various utilities that the user can use.
  1. **Scope:-**
* Admin will login and can add student details and generate reports according to individual student no. or by branch as per the requirements.
* Admin can login and view report and can modify the existing record and submit the action taken against customer according to the type of case. He can also search the record with the help of different attributes like student name and transaction id.
  1. **Definition, Acronym & Abbreviation:-**

**Java:** Java is a programming language expressly designed for use in the [distributed](http://searchcio-midmarket.techtarget.com/definition/distributed) environment of the Internet. It was designed to have the "look and feel" of the [C++](http://searchsqlserver.techtarget.com/definition/C) language, but it is simpler to use than C++ and enforces an [object-oriented programming](http://searchsoa.techtarget.com/definition/object-oriented-programming) model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or [applet](http://searchsoa.techtarget.com/definition/applet) for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

**Oracle:** Oracle is an open source relational database management system. It is based on the structure query language ([SQL](http://techterms.com/definition/sql)), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with Oracle.

**OS:** An operating system (sometimes abbreviated as “OS”) is the program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs.

**Abbreviations:-**

* **DB-** Data Base.
* **SRS-** Software Requirement Specification.

**References:-**

* [www.google.co.in](http://www.google.co.in)
* [www.javaTpoint.com](http://www.javaTpoint.com)
* [www.Oracle.com](http://www.Oracle.com)

**Overview:-**

Revenue management is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time.

**Admin –**

1. Can access the record of the customer.

2. Can modify records of the Customer.

3. Can add details of new customer in database.

4. Can generate reports.

**2. Overall Description:-**

Revenue management is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time.

**2.1 Product perspective:-**

The perspective of this system simulator is to take **information** in an efficient manner and no wastage of paper and time for evaluating the result based on the type of case for particular customer. The application will be of web based.

**2.1.1 System Interface:-** None

**2.1.2 Hardware Interface:-**

* Screen resolution of at least 1260\*768 required for proper and complete viewing of screen. Higher resolution will not be problem.
* Printer for getting print of the reports.

**2.1.3 Software Interface**

The system shall interface with:

* OS of any version of windows.
* Notepad ++ for editor.
* Oracle For database.
  + 1. **Communication Protocols**:-
* If it is used as offline software then it will act as a standalone system and does not require any communication.

**2.1.5 Memory Constrains:-**

|  |  |
| --- | --- |
| RAM | 256 MB or More |
| HARD DISK | 20GB or More |

**2.1.6 Operations:-**

The product release will not cover any automated housekeeping aspects of the database. The DBA at the client site (i.e. Admin) will be responsible for manually deleting, adding data, Database backup and recovery will also have to handle by the Admin.

**2.1.7 Site Adaptation Requirements:-**

The terminals at client’s site will have to support the hardware and software interfaces specified in above section.

* 1. **User Characteristics:-**

User interact with the application while operating it must have knowledge about how to access the web based application and user interact while maintenance phase must have experience, technical expertise.

* 1. **Assumptions and Dependencies:-**

As Invoice Generator System is a web based project so it is completely dependent on web browsers and thus there is no operating system dependencies.

1. **Specific Requirements:-**

This section contains all the software requirements at a level of detail, that when combined with the system context diagram, use case, and use case descriptions, is sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements or not.

* 1. **External Interface Requirements:-**
* Keyboard.
* Mouse.
* Monitor.
* Printer.
  1. **Hardware Requirements**:-

|  |  |
| --- | --- |
| Hard disk | 20 GB or Above. |
| RAM | 256 MB or Above. |
| Processor | Pentium IV or Above. |
| Monitor | Resolution 1260\*768 or Above. |

* 1. **Software Requirements:-**
* Oracle for Database.
* OS any version of Windows.
* Web Browser (Chrome, Mozilla, and Edge of latest version for better performance).
  1. **Software System Attribute:-**

1. **Reliability –**

* We can easily get the customer info and detail.

1. **Availability –**

* Our System is available for any web browser and anywhere we have the without internet connection.

1. **Security –**

* This application is secure from the outsider due to authentication function (Login).

1. **Maintainability –**

* This application can be maintain or recover, if due to any interruption the data of application is corrupted.
* We have the database for maintaining the data in an organized manner.

1. **Portability –**

* Soon it can be easily transferred from one platform to another platform.

1. **Performance –**

* Performance is an indication of the responsiveness of a system to execute specific actions in a given time interval.
* The key issue for performance is increased client response time, reduced throughput and server resource over utilization.
  1. **Communication interface:-**
* If the application is treated as an offline application then there is no need of any communication interface.
  1. **Performance Requirements:-**

None

* 1. **Logical Database Requirements:-**

Oracle is used to maintain database that provide data integrity, data security, imposed integrity constraint on specific table to remove data redundancy and SQL queries are used to perform desired operation.

**Software Design Document**

**INTRODUCTION**

**Product Overview**

Invoice Generator is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time. In practice RM has meant determining pricing according to predicted demand levels so that price sensitive customers who are willing to purchase at off-peak times can do so at favorable prices while price insensitive customers who want to purchase at peak time will be able to do so.

**Problem statement**

**Existing Project:-**

* Existing system speed is slow.
* The most problem is data storage. Because the revenue management system have more data.
* Manual work is large.
* Security is less.
* Corruption of data is largely possible.

**PROPOSED PROJCT:-**

* Minimizing the manual work.
* Maximize the security.
* Maximize the speed of system.
* Minimizing the task of corruption of data.

**Objective of the System:-**

* This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching.
* The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification of the student.
* Similarly record maintenance and updation can also be accomplished by using the identification of the student with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.
* The objective is to prepare the system:
* The software gives information about the feedback of student, etc.
* We can get the information about the performance of the student.
* It also provides various utilities that the user can use.
  1. **Scope:-**
* Admin will login and can add student details and generate reports according to individual student no. or by branch as per the requirements.
* Admin can login and view report and can modify the existing record and submit the action taken against customer according to the type of case. He can also search the record with the help of different attributes like student name and transaction id.
  1. **Definition, Acronym & Abbreviation:-**

**Java:** Java is a programming language expressly designed for use in the [distributed](http://searchcio-midmarket.techtarget.com/definition/distributed) environment of the Internet. It was designed to have the "look and feel" of the [C++](http://searchsqlserver.techtarget.com/definition/C) language, but it is simpler to use than C++ and enforces an [object-oriented programming](http://searchsoa.techtarget.com/definition/object-oriented-programming) model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or [applet](http://searchsoa.techtarget.com/definition/applet) for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

**Oracle:** Oracle is an open source relational database management system. It is based on the structure query language ([SQL](http://techterms.com/definition/sql)), which is used for adding, removing, and modifying information in the database. Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with Oracle.

**OS:** An operating system (sometimes abbreviated as “OS”) is the program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs.

**Abbreviations:-**

* **DB-** Data Base.
* **SRS-** Software Requirement Specification.

**References:-**

* [www.google.co.in](http://www.google.co.in)
* [www.javaTpoint.com](http://www.javaTpoint.com)
* [www.Oracle.com](http://www.Oracle.com)

**Overview:-**

Revenue management is the application of information system and pricing strategies to allocate the right capacity to the right customer at the right place at the right time.

**Admin –**

1. Can access the record of the customer.

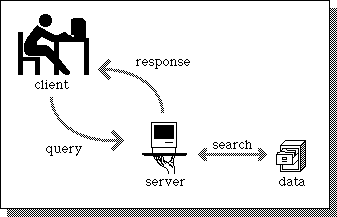
2. Can modify records of the Customer.

3. Can add details of new customer in database.

4. Can generate reports.

1. **System Architecture:-**
   1. **Client Server Architecture:-**

It is Client Server architecture. Where client request for the data through server it fulfill the requirement.



1. **Data Design:-**
   1. **Data Description:-**

|  |  |
| --- | --- |
| Module | Method |
| Registration | Update(), delete(), view() |
| Admin | Update(),delete(),add(),report() |
| Login | Login() |

**E-R DIAGRAM:-**

Login confirmation

Login table

Admin

Customerdetails

Can add, delete, modify

Generate

Reports

d

|  |  |
| --- | --- |
| **Screen Name** | **It’s function** |
| 1. Login Screen | This will be the entry point of the software where users like Invoice Generator or admin will enter their credentials and navigated to internal pages if the credentials will matched to the original credentials. |
| 1. Add details | On this page Admin will enter customer details along with his type of Invoice Generation case and its date. |
| 1. Update/delete | On this page Admin or Invoice can update or delete details of any particular customer with the help of his customer name. |
| 1. Customer Details | On this page Admin can find the name of customer and bill of customer. |

CODING

**Registration.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class Registration extends JFrame implements ActionListener

{

JLabel l1, l2, l3, l4, l5, l6, l7, l8;

JTextField tf1, tf2, tf5, tf6, tf7;

JButton btn1, btn2;

JPasswordField p1, p2;

Registration()

{

setVisible(true);

setSize(700, 700);

setLayout(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setTitle("Registration in Invoice Generator");

l1 = new JLabel("Registration in Invoice Generator:");

l1.setForeground(Color.blue);

l1.setFont(new Font("Serif", Font.BOLD, 20));

l2 = new JLabel("Name:");

l3 = new JLabel("Username");

l4 = new JLabel("Create Passowrd:");

l5 = new JLabel("Confirm Password:");

l6 = new JLabel("Country:");

l7 = new JLabel("State:");

l8 = new JLabel("Phone No:");

tf1 = new JTextField();

tf2 = new JTextField();

p1 = new JPasswordField();

p2 = new JPasswordField();

tf5 = new JTextField();

tf6 = new JTextField();

tf7 = new JTextField();

btn1 = new JButton("Submit");

btn2 = new JButton("Clear");

btn1.addActionListener(this);

btn2.addActionListener(this);

l1.setBounds(100, 30, 400, 30);

l2.setBounds(80, 70, 200, 30);

l3.setBounds(80, 110, 200, 30);

l4.setBounds(80, 150, 200, 30);

l5.setBounds(80, 190, 200, 30);

l6.setBounds(80, 230, 200, 30);

l7.setBounds(80, 270, 200, 30);

l8.setBounds(80, 310, 200, 30);

tf1.setBounds(300, 70, 200, 30);

tf2.setBounds(300, 110, 200, 30);

p1.setBounds(300, 150, 200, 30);

p2.setBounds(300, 190, 200, 30);

tf5.setBounds(300, 230, 200, 30);

tf6.setBounds(300, 270, 200, 30);

tf7.setBounds(300, 310, 200, 30);

btn1.setBounds(50, 350, 100, 30);

btn2.setBounds(170, 350, 100, 30);

add(l1);

add(l2);

add(tf1);

add(l3);

add(tf2);

add(l4);

add(p1);

add(l5);

add(p2);

add(l6);

add(tf5);

add(l7);

add(tf6);

add(l8);

add(tf7);

add(btn1);

add(btn2);

}

public void actionPerformed(ActionEvent e)

{

if (e.getSource() == btn1)

{

int x = 0;

String s1 = tf1.getText();

String s2 = tf2.getText();

char[] s3 = p1.getPassword();

char[] s4 = p2.getPassword();

String s8 = new String(s3);

String s9 = new String(s4);

String s5 = tf5.getText();

String s6 = tf6.getText();

String s7 = tf7.getText();

if (s8.equals(s9))

{

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "sachin");

PreparedStatement ps = con.prepareStatement("insert into emp values(?,?,?,?,?,?)");

ps.setString(1, s1);

ps.setString(2, s2);

ps.setString(3, s8);

ps.setString(4, s5);

ps.setString(5, s6);

ps.setString(6, s7);

ResultSet rs = ps.executeQuery();

x++;

if (x > 0)

{

JOptionPane.showMessageDialog(btn1, "Data Saved Successfully");

}

new Login();

}

catch (Exception ex)

{

System.out.println(ex);

}

}

else

{

JOptionPane.showMessageDialog(btn1, "Password Does Not Match");

}

}

else

{

tf1.setText("");

tf2.setText("");

p1.setText("");

p2.setText("");

tf5.setText("");

tf6.setText("");

tf7.setText("");

}

}

public static void main(String args[])

{

new Registration();

}

}

**Login.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class Login extends JFrame implements ActionListener

{

JLabel l1, l2, l3;

JTextField tf1;

JButton btn1;

JPasswordField p1;

Login()

{

setTitle("Login in invoice Generator");

setVisible(true);

setSize(800, 800);

setLayout(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

l1 = new JLabel("Login in invoice Generator");

l1.setForeground(Color.blue);

l1.setFont(new Font("Serif", Font.BOLD, 20));

l2 = new JLabel("UserName:");

l3 = new JLabel("Password:");

tf1 = new JTextField();

p1 = new JPasswordField();

btn1 = new JButton("Submit");

l1.setBounds(100, 30, 400, 30);

l2.setBounds(80, 70, 200, 30);

l3.setBounds(80, 110, 200, 30);

tf1.setBounds(300, 70, 200, 30);

p1.setBounds(300, 110, 200, 30);

btn1.setBounds(150, 160, 100, 30);

add(l1);

add(l2);

add(tf1);

add(l3);

add(p1);

add(btn1);

btn1.addActionListener(this);

}

public void actionPerformed(ActionEvent e)

{

if(showData())

{

new CustomerBill();

dispose();

}

}

public boolean showData()

{

JLabel l, l0;

String str1 = tf1.getText();

char[] p = p1.getPassword();

String str2 = new String(p);

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "sachin");

PreparedStatement ps = con.prepareStatement("select name from emp where username=? and pass=?");

ps.setString(1, str1);

ps.setString(2, str2);

ResultSet rs = ps.executeQuery();

if (rs.next())

{

setVisible(true);

setSize(600, 600);

setLayout(null);

l = new JLabel();

l0 = new JLabel("you are succefully logged in..");

l0.setForeground(Color.blue);

l0.setFont(new Font("Serif", Font.BOLD, 30));

l.setBounds(60, 50, 400, 30);

l0.setBounds(60, 100, 400, 40);

add(l);

add(l0);

l.setText("Welcome " + rs.getString(1));

l.setForeground(Color.red);

l.setFont(new Font("Serif", Font.BOLD, 30));

} else

{

JOptionPane.showMessageDialog(null,"Incorrect UserName or password..Try Again with correct detail");

return false;

}

}

catch (Exception ex)

{

System.out.println(ex);

}

return true;

}

public static void main(String arr[])

{

new Login();

}

}

**CustomerBill.java**

import java.awt.GridBagConstraints;

import java.awt.GridBagLayout;

import java.awt.Insets;

import javax.swing.BorderFactory;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.SwingUtilities;

import javax.swing.event.TableModelEvent;

import javax.swing.event.TableModelListener;

public class CustomerBill

{

private JFrame mainFrame;

public static final int PRICE\_INDEX = 1;

public static final int QUANTITY\_INDEX = 2;

public static final int SUBTOTAL\_INDEX = 3;

public static final int VAT\_INDEX = 4;

public static final int TOTAL\_INDEX = 5;

private static final double vatPercentage = 14.5;

/\*\*

\* Table Panel Data members

\*/

private JTable mainTable;

private JTextField subTotalText;

private static final Double zeroValue = new Double(0.00);

private static final Integer oneValue = new Integer(1);

private static final Object[][] columnData = new Object[][] { { "Item - 2", zeroValue, oneValue, zeroValue, zeroValue, zeroValue }, { "Item - 3", zeroValue, oneValue, zeroValue, zeroValue, zeroValue }, { "Item - 4", zeroValue, oneValue, zeroValue, zeroValue, zeroValue }, { "Item - 5", zeroValue, oneValue, zeroValue, zeroValue, zeroValue }, { "Item - 6", zeroValue, oneValue, zeroValue, zeroValue, zeroValue } };

/\*\*

\* Sender Data Members

\*/

private JTextField senderNameText, senderTaxId;

private JTextArea senderAddressText;

/\*\*

\* Receiver Data Members

\*/

private JTextField receiverNameText, receiverTaxIdText;

private JTextArea receiverAddressText;

/\*\*

\* Data Members for bank details

\*/

private JTextField bankNameText, bankIFSCText;

public CustomerBill()

{

initUI();

}

private void initUI()

{

JPanel billSenderPanel = createBillSenderPanel();

JPanel billReceiverPanel = createBillReceiverPanel();

JPanel tablePanel = createTablePane();

mainFrame = new JFrame();

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.pack();

mainFrame.setLocationRelativeTo(null);

mainFrame.setTitle("Customer Bill Management");

mainFrame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

mainFrame.setVisible(true);

mainFrame.setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.insets = new Insets(4, 4, 4, 4);

gbc.anchor = GridBagConstraints.WEST;

mainFrame.add(billSenderPanel, gbc);

gbc.gridx++;

mainFrame.add(billReceiverPanel, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 6;

mainFrame.add(tablePanel, gbc);

JPanel bottumPanel = createButtonPanel();

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 6;

mainFrame.add(bottumPanel, gbc);

}

/\*\*

\* Sender Panel

\*

\* @return

\*/

private JPanel createBillSenderPanel()

{

senderNameText = new JTextField(10);

senderTaxId = new JTextField(10);

senderAddressText = new JTextArea(5, 20);

JPanel billSenderPanel = new JPanel();

billSenderPanel.setBorder(BorderFactory.createEtchedBorder());

billSenderPanel.setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.insets = new Insets(4, 4, 4, 4);

gbc.anchor = GridBagConstraints.WEST;

billSenderPanel.add(new JLabel("Name : "), gbc);

gbc.gridx++;

billSenderPanel.add(senderNameText, gbc);

gbc.gridx++;

billSenderPanel.add(new JLabel("Tax Id: "), gbc);

gbc.gridx++;

billSenderPanel.add(senderTaxId, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.anchor = GridBagConstraints.NORTHWEST;

billSenderPanel.add(new JLabel("Address : "), gbc);

gbc.gridx++;

gbc.gridwidth = 3;

gbc.fill = GridBagConstraints.HORIZONTAL;

billSenderPanel.add(new JScrollPane(senderAddressText), gbc);

return billSenderPanel;

}

/\*\*

\* Receiver Panel

\*

\* @return

\*/

private JPanel createBillReceiverPanel()

{

receiverNameText = new JTextField(10);

receiverTaxIdText = new JTextField(10);

receiverAddressText = new JTextArea(5, 20);

JPanel billReceiverPanel = new JPanel();

billReceiverPanel.setBorder(BorderFactory.createEtchedBorder());

billReceiverPanel.setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.insets = new Insets(4, 4, 4, 4);

gbc.anchor = GridBagConstraints.WEST;

billReceiverPanel.add(new JLabel("Name : "), gbc);

gbc.gridx++;

billReceiverPanel.add(receiverNameText, gbc);

gbc.gridx++;

billReceiverPanel.add(new JLabel("Tax Id : "), gbc);

gbc.gridx++;

billReceiverPanel.add(receiverTaxIdText, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.anchor = GridBagConstraints.NORTHWEST;

billReceiverPanel.add(new JLabel("Address : "), gbc);

gbc.gridx++;

gbc.gridwidth = 3;

gbc.fill = GridBagConstraints.HORIZONTAL;

billReceiverPanel.add(new JScrollPane(receiverAddressText), gbc);

return billReceiverPanel;

}

private JPanel createTablePane()

{

JPanel tablePanel = new JPanel();

tablePanel.setBorder(BorderFactory.createEtchedBorder());

tablePanel.setLayout(new GridBagLayout());

mainTable = new JTable();

mainTable.setModel(new CustomTableModel(columnData));

JScrollPane tableScrollPane = new JScrollPane();

tableScrollPane.setViewportView(mainTable);

mainTable.getModel().addTableModelListener(new TableModelListener()

{

public void tableChanged(TableModelEvent evt)

{

CustomTableModel modal = (CustomTableModel) evt.getSource();

int selectedRowIndex = mainTable.getSelectedRow();

double price = Double.parseDouble(modal.getValueAt(selectedRowIndex, PRICE\_INDEX).toString().trim());

int quantity = Integer.parseInt(modal.getValueAt(selectedRowIndex, QUANTITY\_INDEX).toString().trim());

double subTotalAmount = price \* quantity;

double vatAmount = (vatPercentage \* subTotalAmount) / 100;

double total = subTotalAmount + vatAmount;

columnData[selectedRowIndex][SUBTOTAL\_INDEX] = subTotalAmount;

columnData[selectedRowIndex][VAT\_INDEX] = vatAmount;

columnData[selectedRowIndex][TOTAL\_INDEX] = total;

double totalAmountToPay = 0;

for(int i = 0; i < columnData.length; i++)

{

totalAmountToPay += Double.parseDouble(columnData[i][TOTAL\_INDEX].toString());

}

updateSubTotal(totalAmountToPay);

}

});

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.insets = new Insets(4, 4, 4, 4);

gbc.anchor = GridBagConstraints.WEST;

gbc.gridwidth = 2;

tablePanel.add(tableScrollPane, gbc);

JLabel subTotal = new JLabel("Sub Total :");

subTotalText = new JTextField(10);

gbc.gridy++;

gbc.gridwidth = 1;

tablePanel.add(subTotal, gbc);

gbc.gridx++;

tablePanel.add(subTotalText, gbc);

return tablePanel;

}

private void updateSubTotal(double amount)

{

subTotalText.setText("" + amount);

subTotalText.revalidate();

}

/\*\*

\* This method creates the button panel

\*

\* @return

\*/

private JPanel createButtonPanel()

{

JPanel buttonPanel = new JPanel();

JButton okButton = new JButton("OK");

okButton.addActionListener(new java.awt.event.ActionListener()

{

public void actionPerformed(java.awt.event.ActionEvent evt)

{

okButtonActionPerformed(evt);

}

});

JButton cancelButton = new JButton("Cancel");

cancelButton.setText("CANCEL");

cancelButton.addActionListener(new java.awt.event.ActionListener()

{

public void actionPerformed(java.awt.event.ActionEvent evt)

{

cancelButtonActionPerformed(evt);

}

});

JLabel bankNameLbl = new JLabel("Bank Name : ");

JLabel bankIFSCLbl = new JLabel("Bank IFSC : ");

bankNameText = new JTextField(10);

bankIFSCText = new JTextField(10);

buttonPanel.setBorder(BorderFactory.createEtchedBorder());

buttonPanel.setLayout(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.insets = new Insets(4, 4, 4, 4);

gbc.anchor = GridBagConstraints.WEST;

buttonPanel.add(bankNameLbl, gbc);

gbc.gridx++;

buttonPanel.add(bankNameText, gbc);

gbc.gridx++;

buttonPanel.add(bankIFSCLbl, gbc);

gbc.gridx++;

buttonPanel.add(bankIFSCText, gbc);

gbc.gridx++;

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

buttonPanel.add(okButton, gbc);

gbc.gridx++;

gbc.gridwidth = 2;

buttonPanel.add(cancelButton, gbc);

return buttonPanel;

}

private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt)

{

mainFrame.dispose();

}

private void okButtonActionPerformed(java.awt.event.ActionEvent evt)

{

try {

DBManager dBManager = new DBManager();

for (int i = 0; i < columnData.length; i++) {

Object[] singleRowData = columnData[i];

dBManager.insertRecordInCustomerData(singleRowData);

}

String[] payableData = new String[3];

payableData[0] = bankNameText.getText().toString().trim();

payableData[1] = bankIFSCText.getText().toString().trim();

payableData[2] = subTotalText.getText().toString().trim();

dBManager.insertPayableDetails(payableData);

String[] receiverData = new String[3];

receiverData[0] = receiverNameText.getText().toString().trim();

receiverData[1] = receiverAddressText.getText().toString().trim();

receiverData[2] = receiverTaxIdText.getText().toString().trim();

dBManager.insertReceiverDetails(receiverData);

String[] senderData = new String[3];

senderData[0] = senderNameText.getText().toString().trim();

senderData[1] = senderAddressText.getText().toString().trim();

senderData[2] = senderTaxId.getText().toString().trim();

dBManager.insertSenderDetails(senderData);

dBManager.closeConnection();

JOptionPane.showMessageDialog(mainFrame, "Thank you for shopping !!! Please visit again !!!", "Invoice Genaration Successful", JOptionPane.INFORMATION\_MESSAGE);

mainFrame.dispose();

}

catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args)

{

SwingUtilities.invokeLater(new Runnable()

{

public void run()

{

new CustomerBill();

}

});

}

}

**DBManager.java**

import java.sql.\*;

public class DBManager

{

private Connection connection;

public DBManager()

{

try

{

connection = openConnection();

}

catch(Exception ex)

{

ex.printStackTrace();

}

}

private Connection openConnection()

{

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

return DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "sachin");

}

catch(Exception ex)

{

ex.printStackTrace();

return null;

}

}

public void insertSenderDetails(String[] senderData)

{

try

{

PreparedStatement ps = connection.prepareStatement("insert into senderdetails values(?,?,?)");

ps.setString(1, senderData[0]);

ps.setString(2, senderData[1]);

ps.setString(3, senderData[2]);

ps.executeQuery();

}

catch(Exception ex)

{

ex.printStackTrace();

}

}

public void insertReceiverDetails(String[] receiverData)

{

try

{

PreparedStatement ps = connection.prepareStatement("insert into receiverdetails values(?,?,?)");

ps.setString(1, receiverData[0]);

ps.setString(2, receiverData[1]);

ps.setString(3, receiverData[2]);

ps.executeQuery();

}

catch(Exception ex)

{

ex.printStackTrace();

}

}

public void insertPayableDetails(String[] payableData)

{

try

{

PreparedStatement ps = connection.prepareStatement("insert into payabledata values(?,?,?)");

ps.setString(1, payableData[0]);

ps.setString(2, payableData[1]);

ps.setString(3, payableData[2]);

ps.executeQuery();

}

catch(Exception ex)

{

ex.printStackTrace();

}

}

public void insertRecordInCustomerData(Object[] singleRowData)

{

try

{

String itemName = singleRowData[0].toString();

double price = Double.parseDouble(singleRowData[CustomerBill.PRICE\_INDEX].toString());

int quantity = Integer.parseInt(singleRowData[CustomerBill.QUANTITY\_INDEX].toString());

double subTotalAmount = Double.parseDouble(singleRowData[CustomerBill.SUBTOTAL\_INDEX].toString());

double vatAmount = Double.parseDouble(singleRowData[CustomerBill.VAT\_INDEX].toString());

double totalAmount = Double.parseDouble(singleRowData[CustomerBill.TOTAL\_INDEX].toString());

PreparedStatement ps = connection.prepareStatement("insert into customerdata values(?,?,?,?,?,?)");

ps.setString(1, itemName);

ps.setString(2, "" + price);

ps.setString(3, "" + quantity);

ps.setString(4, "" + subTotalAmount);

ps.setString(5, "" + vatAmount);

ps.setString(6, "" + totalAmount);

ps.executeQuery();

}

catch(Exception ex)

{

ex.printStackTrace();

}

}

public void closeConnection()

{

try

{

if(null != connection)

{

connection.close();

}

}

catch(Exception ex)

{

ex.printStackTrace();

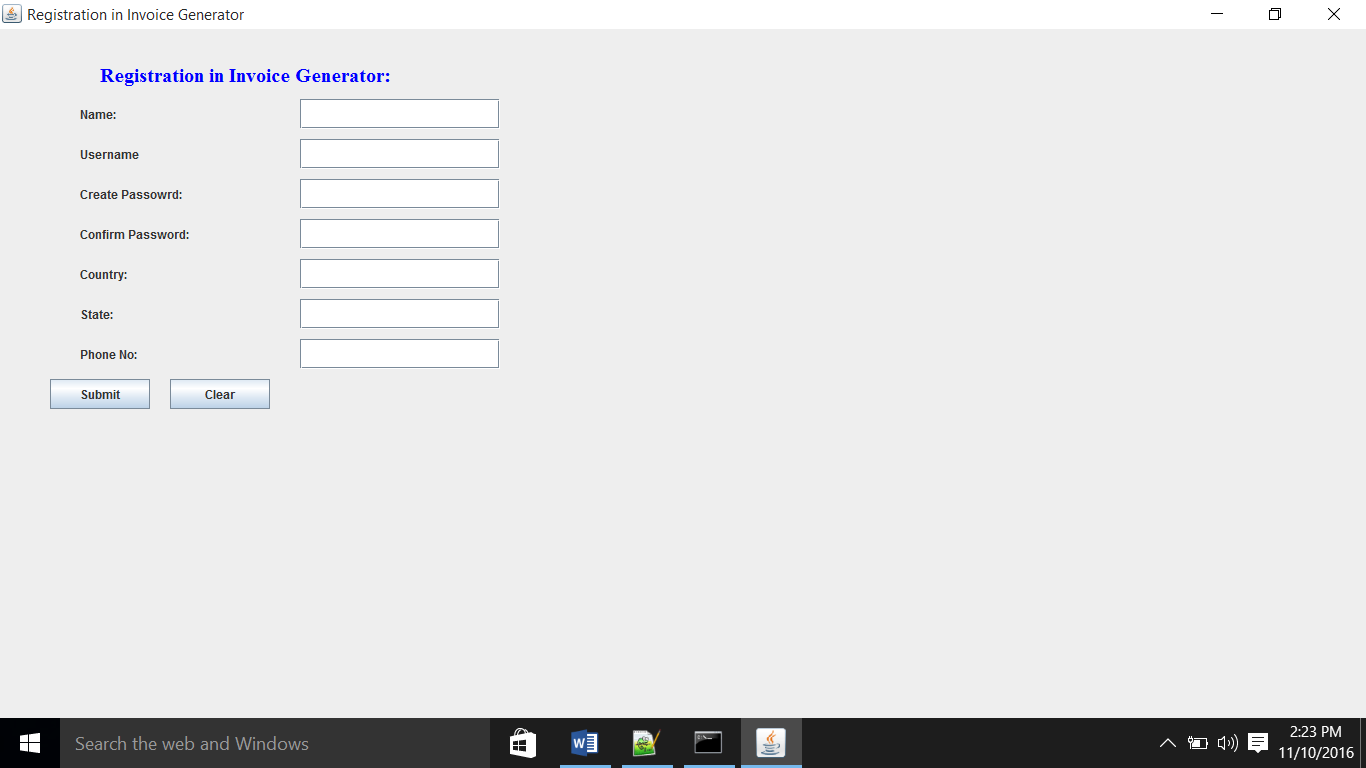
}

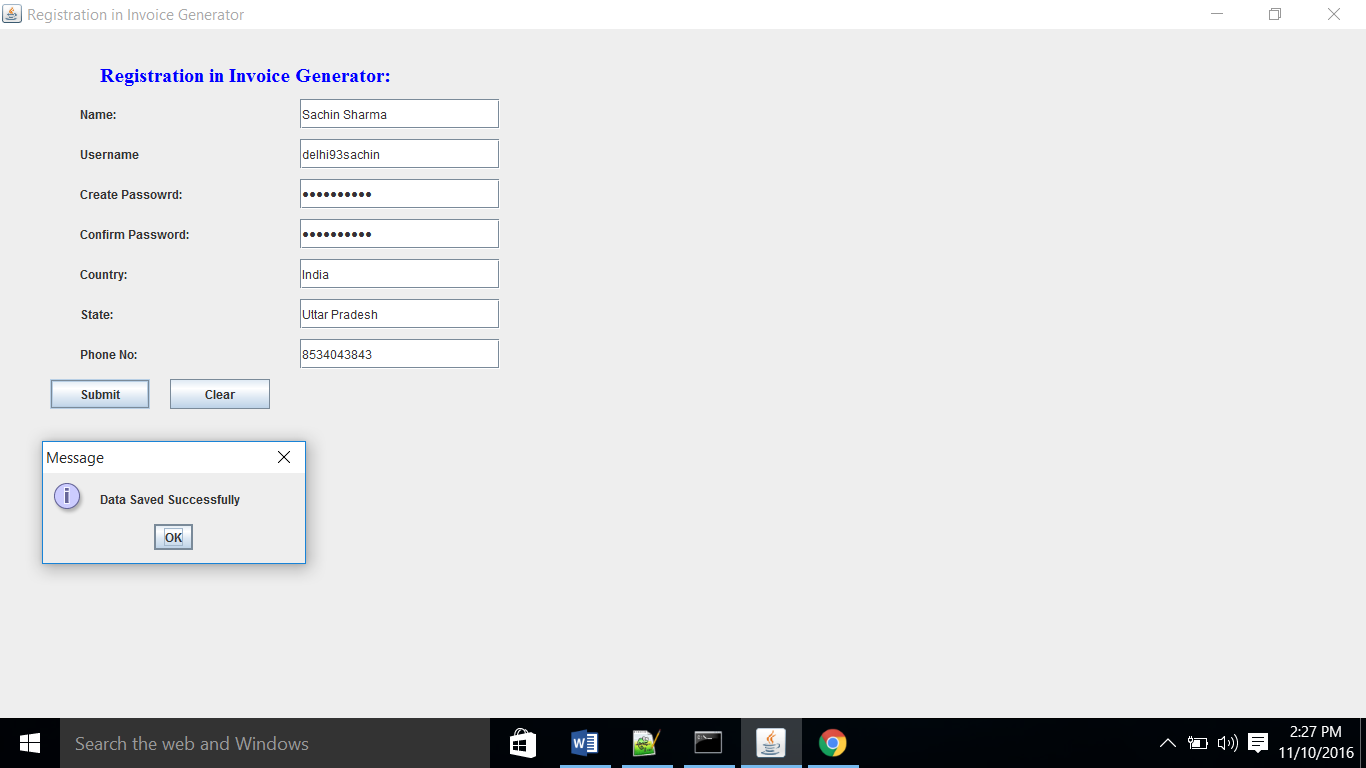
}

}

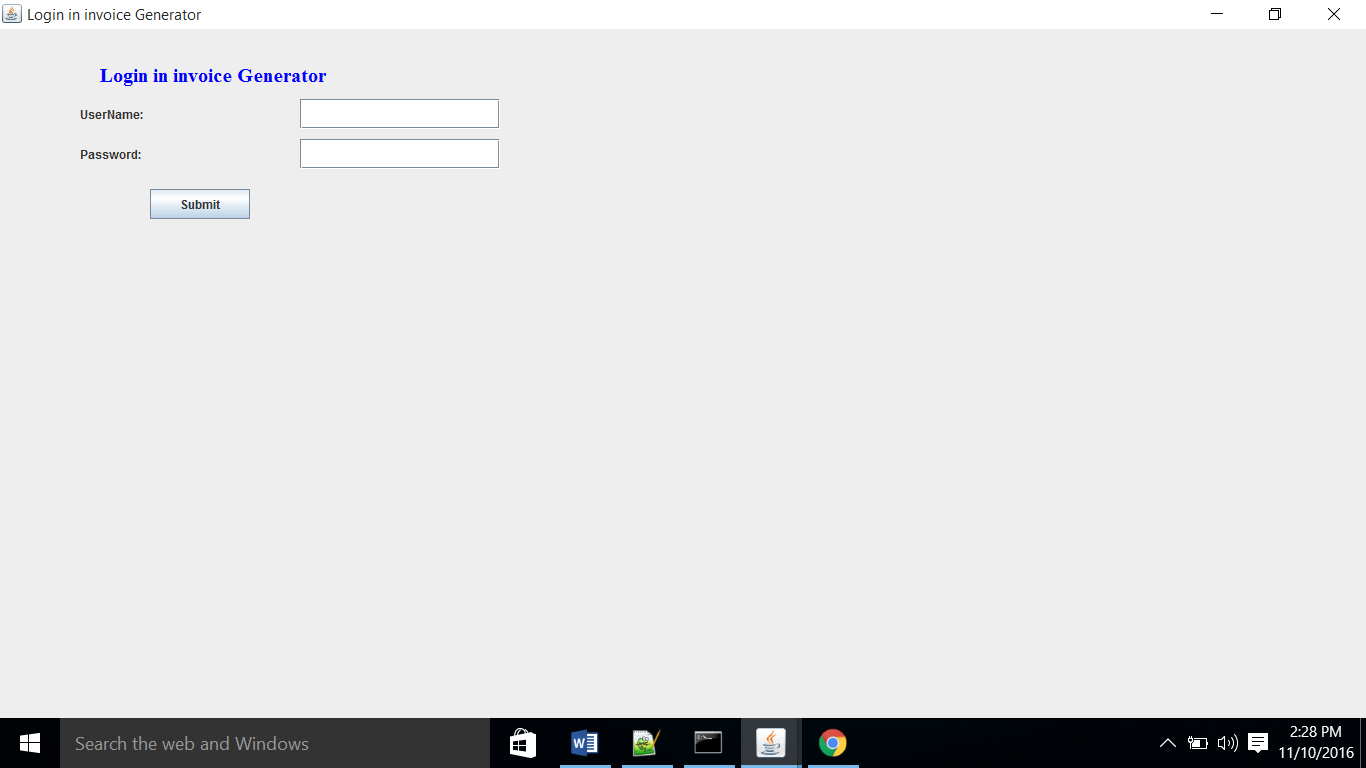
**Snapshots Of Forms**

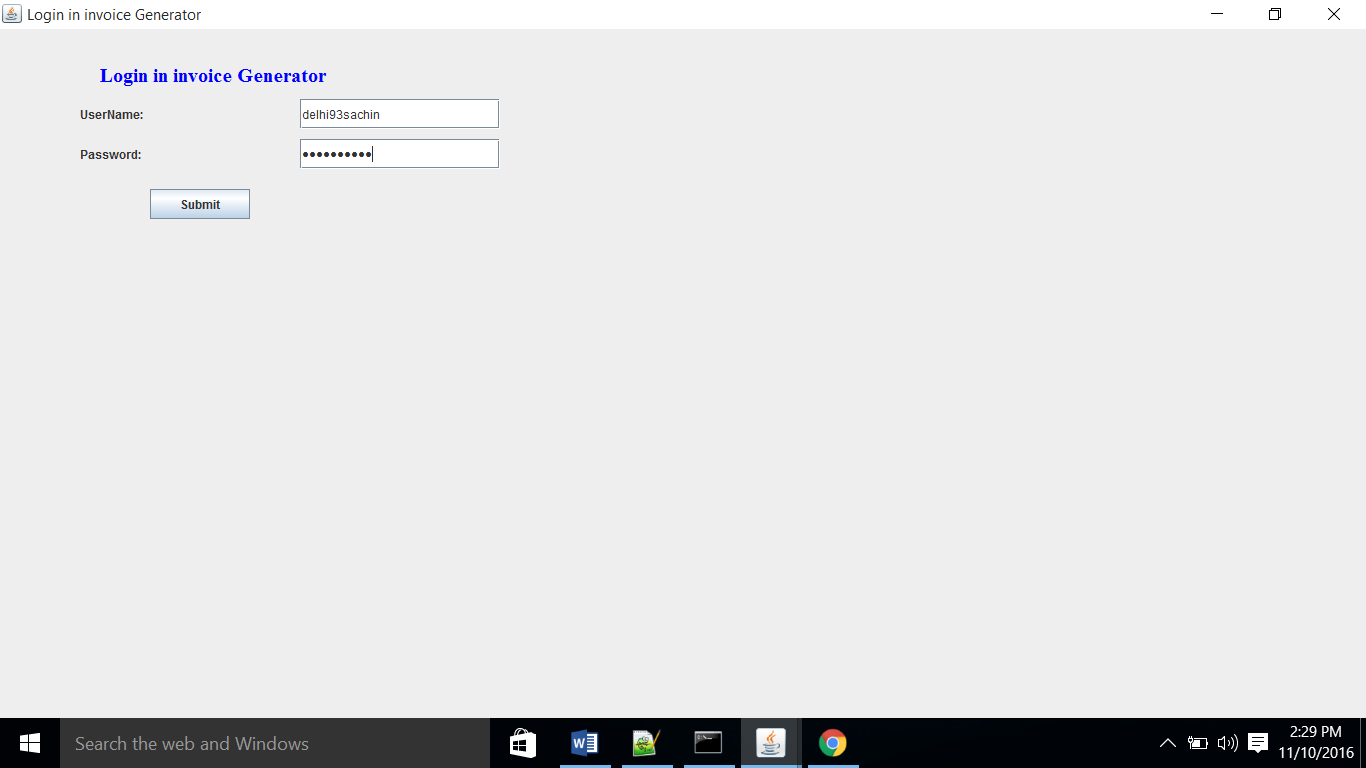
**Registration form**



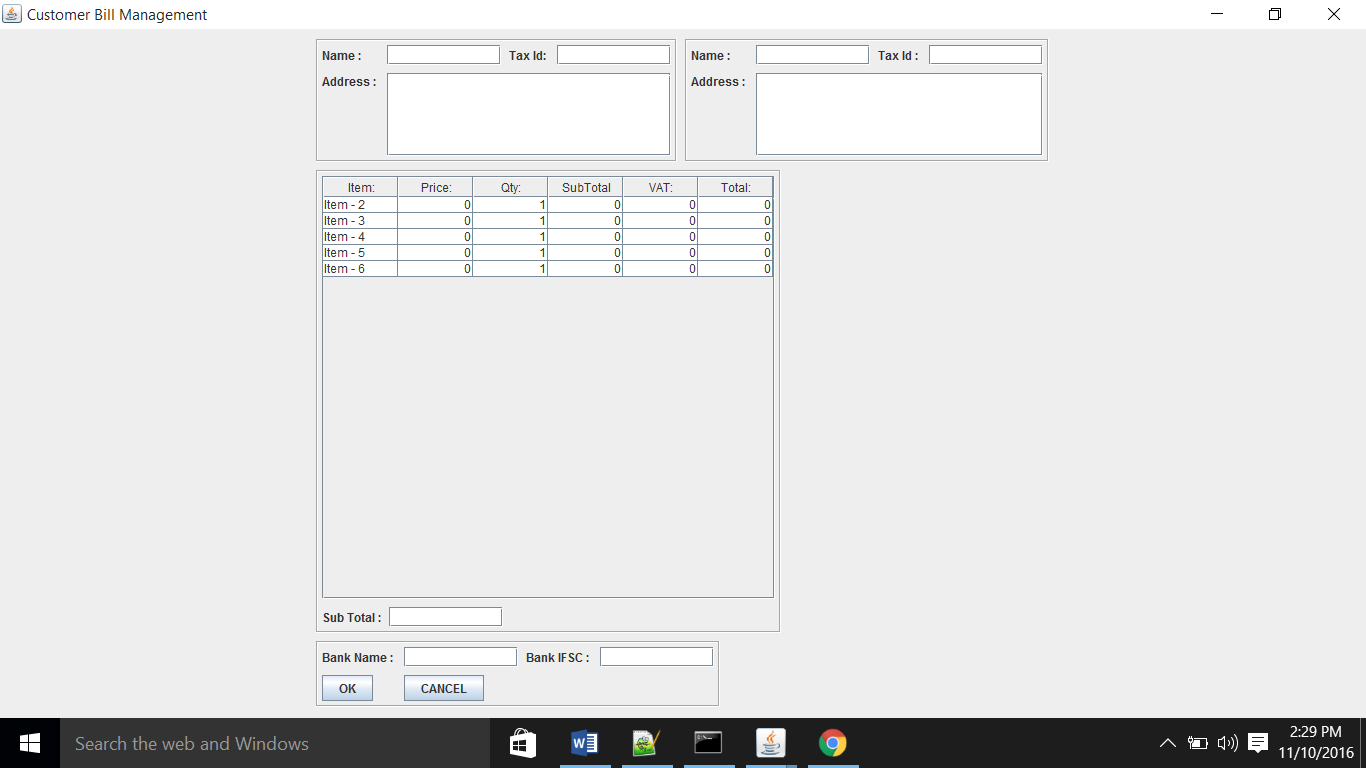


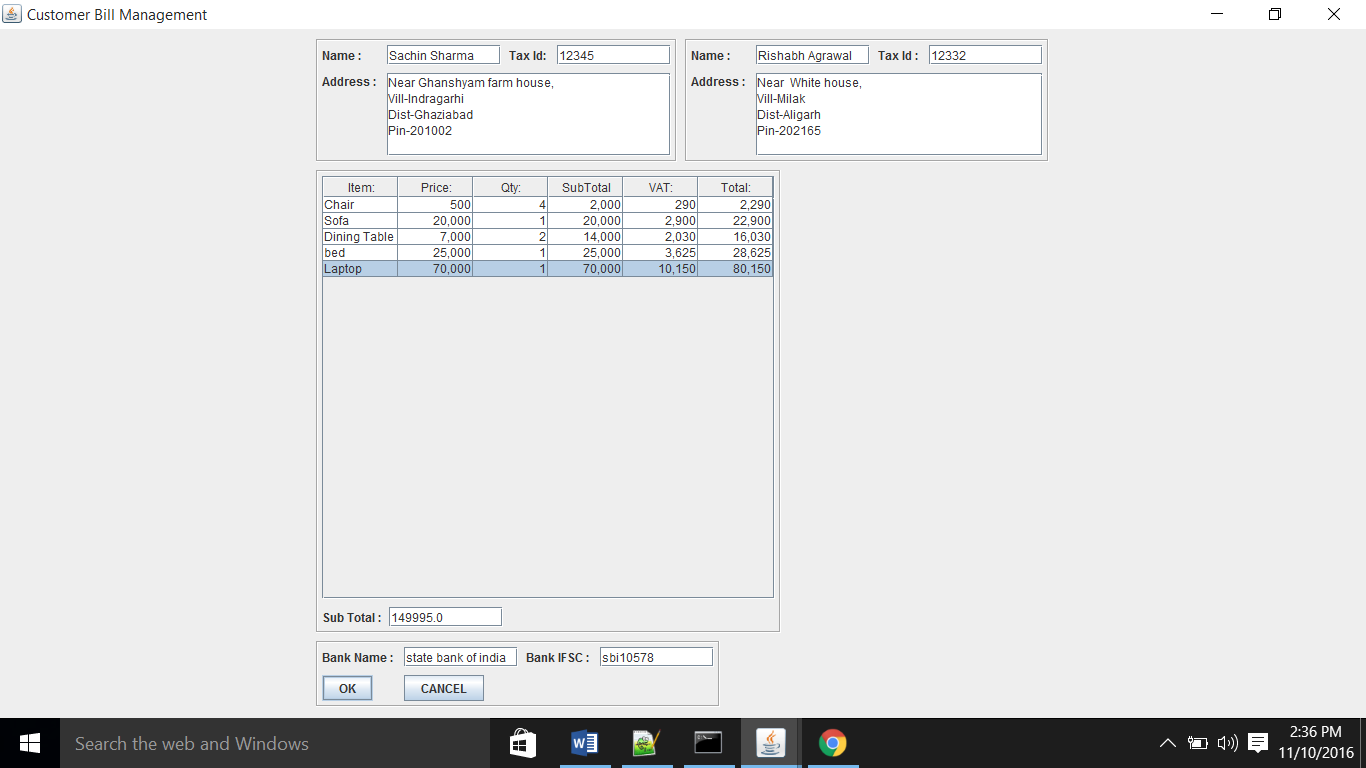
**Login form:-**

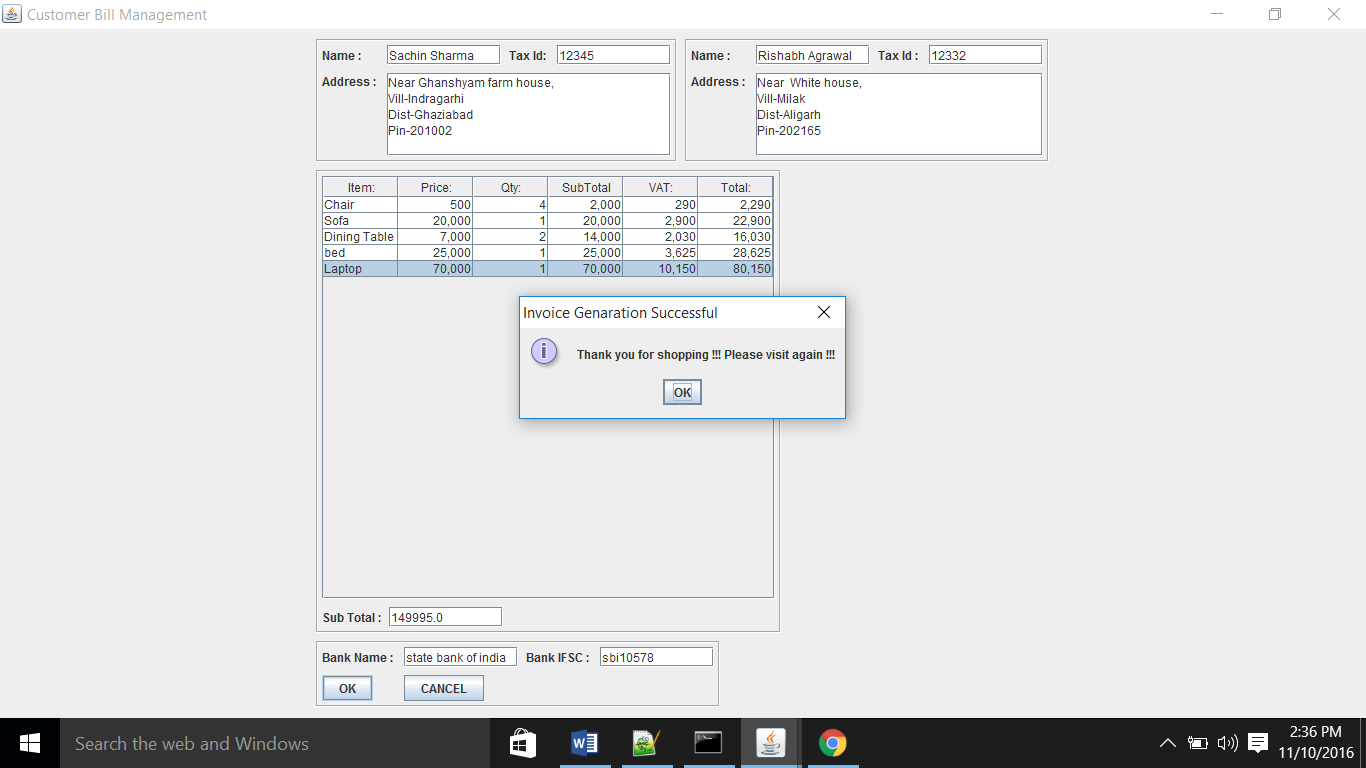


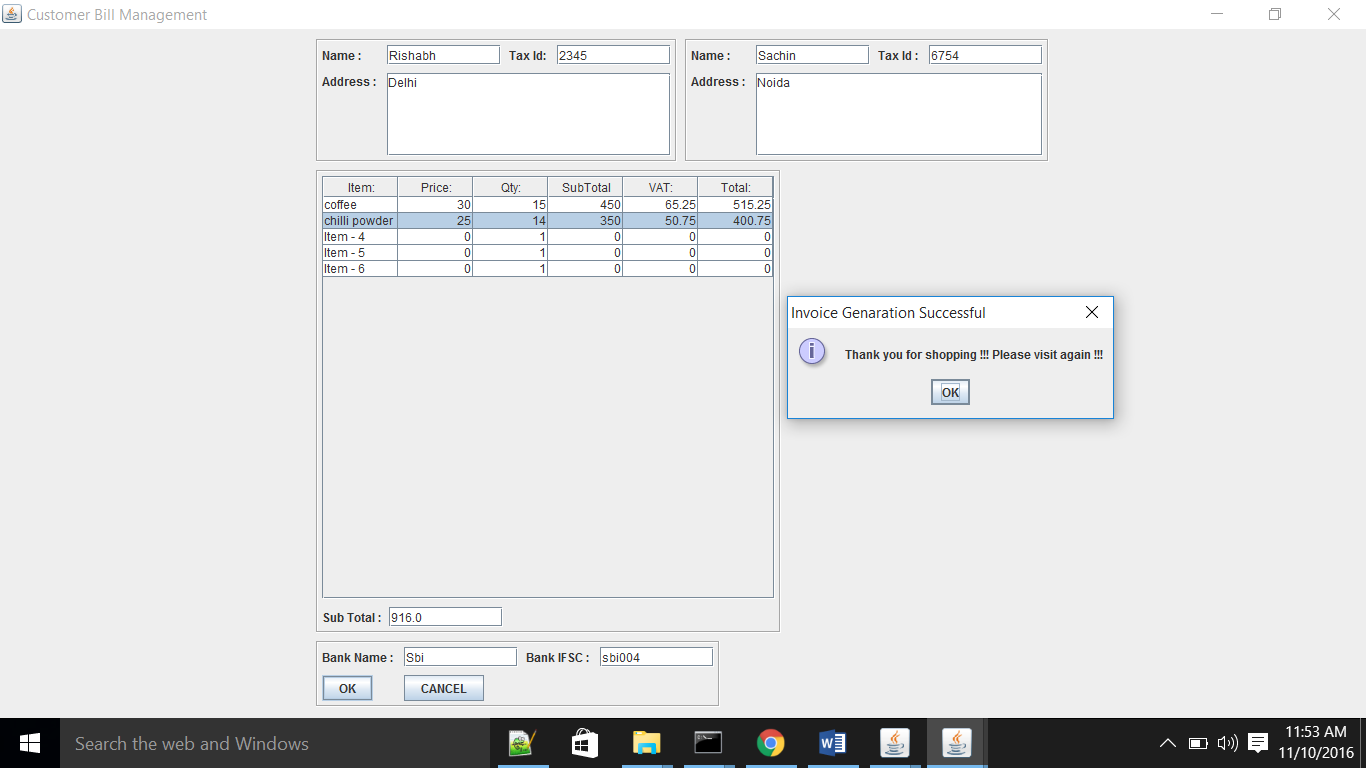


**Customer\_Bill:-**

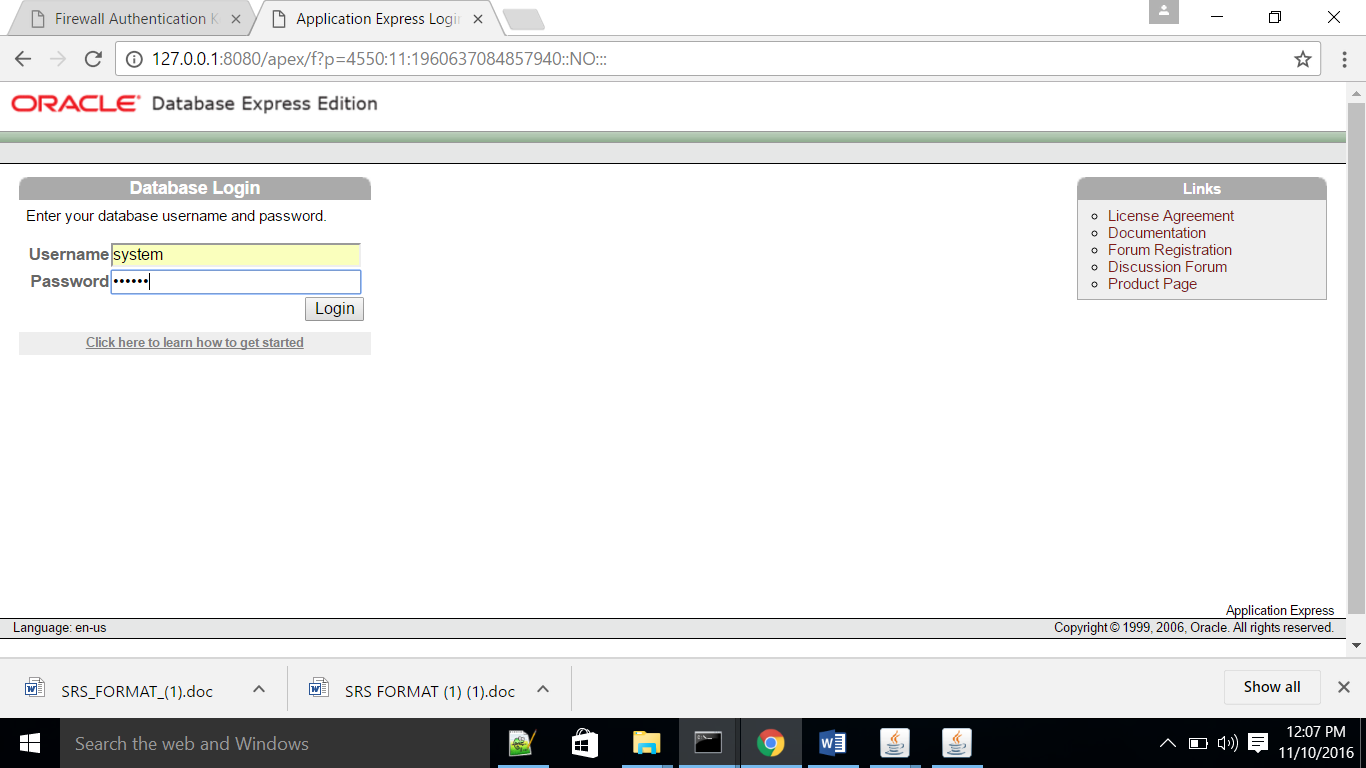




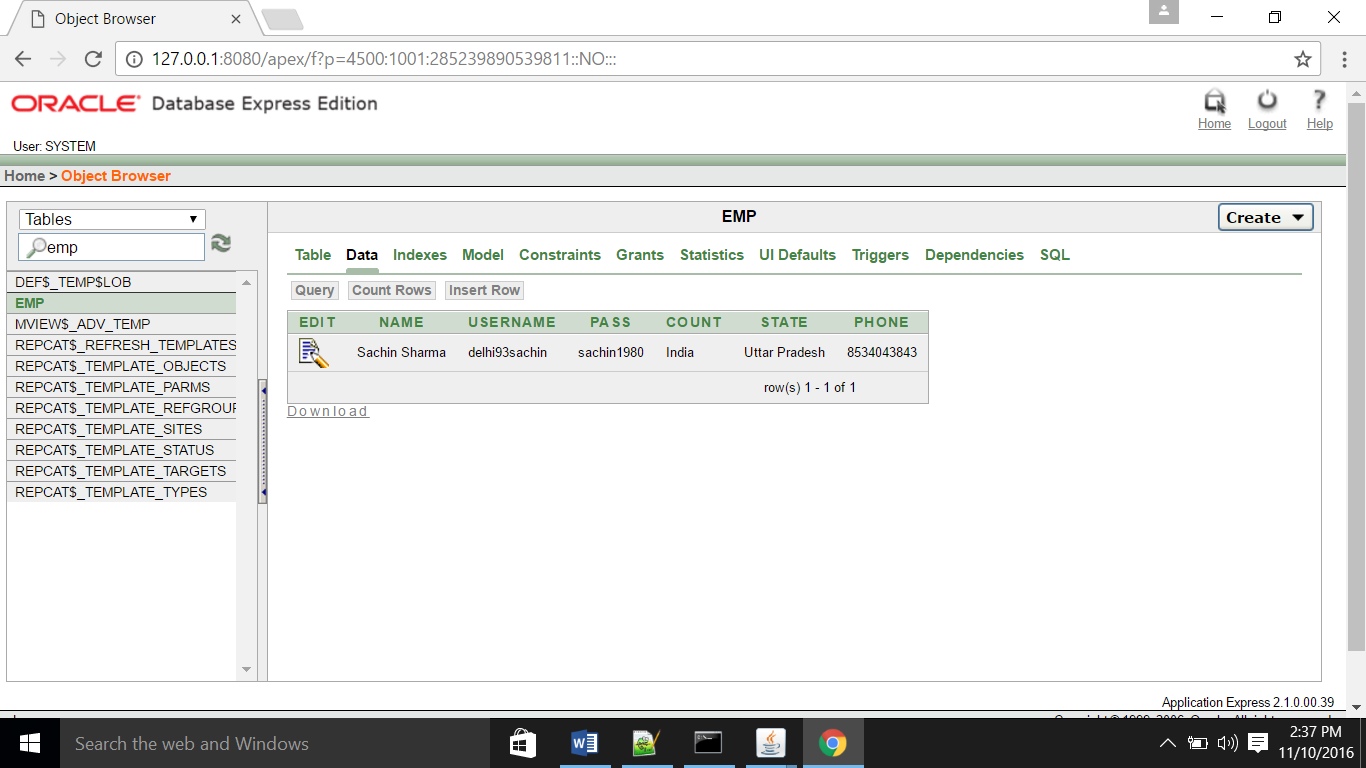




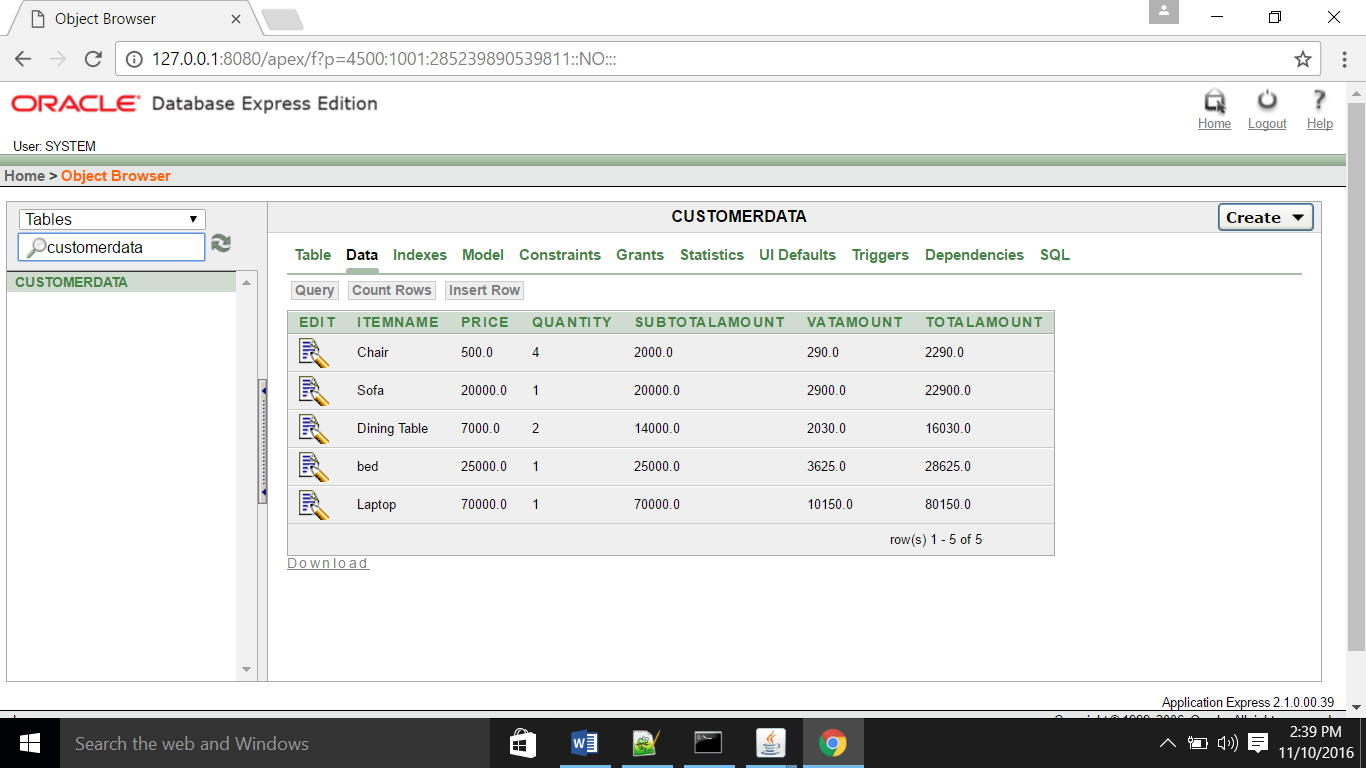
**Database Snapshot:-**



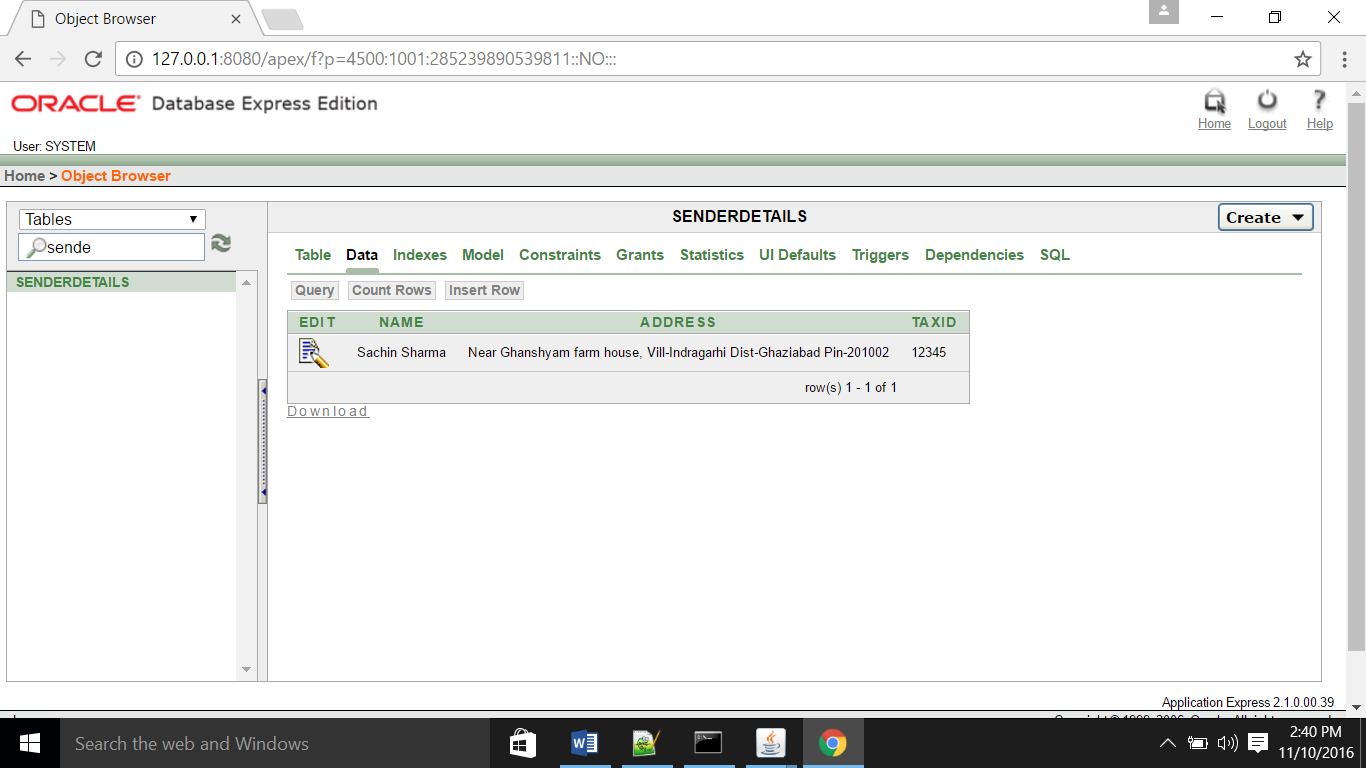
**EMP:-**



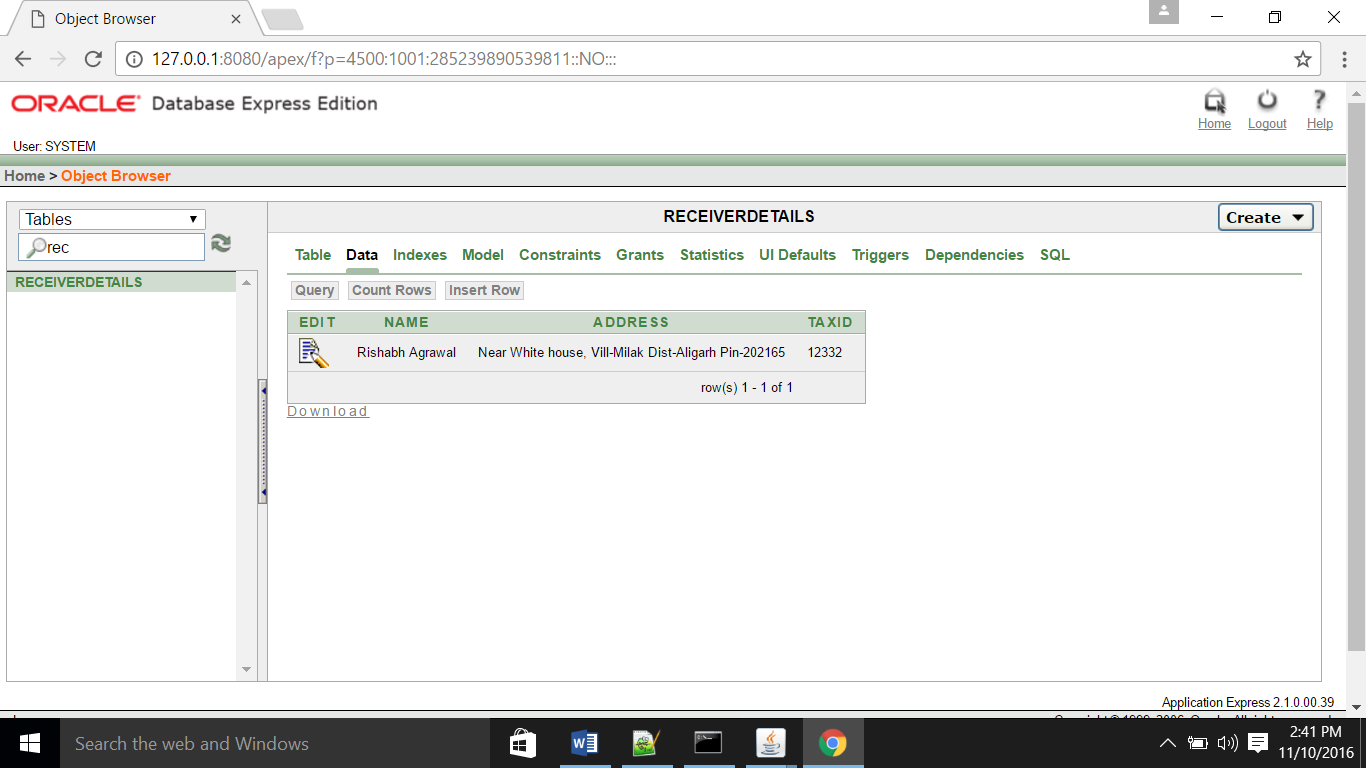
**CustomerData:-**



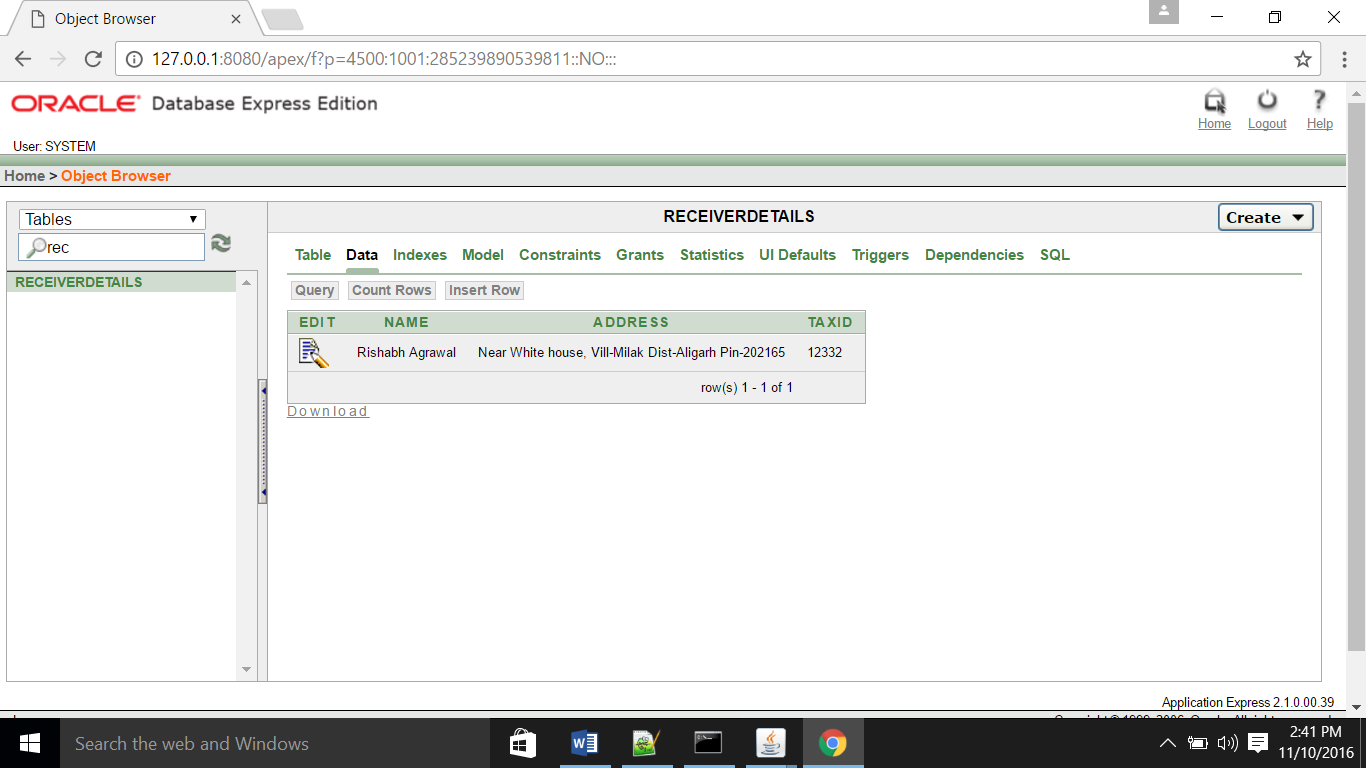
**SenderDetails:-**



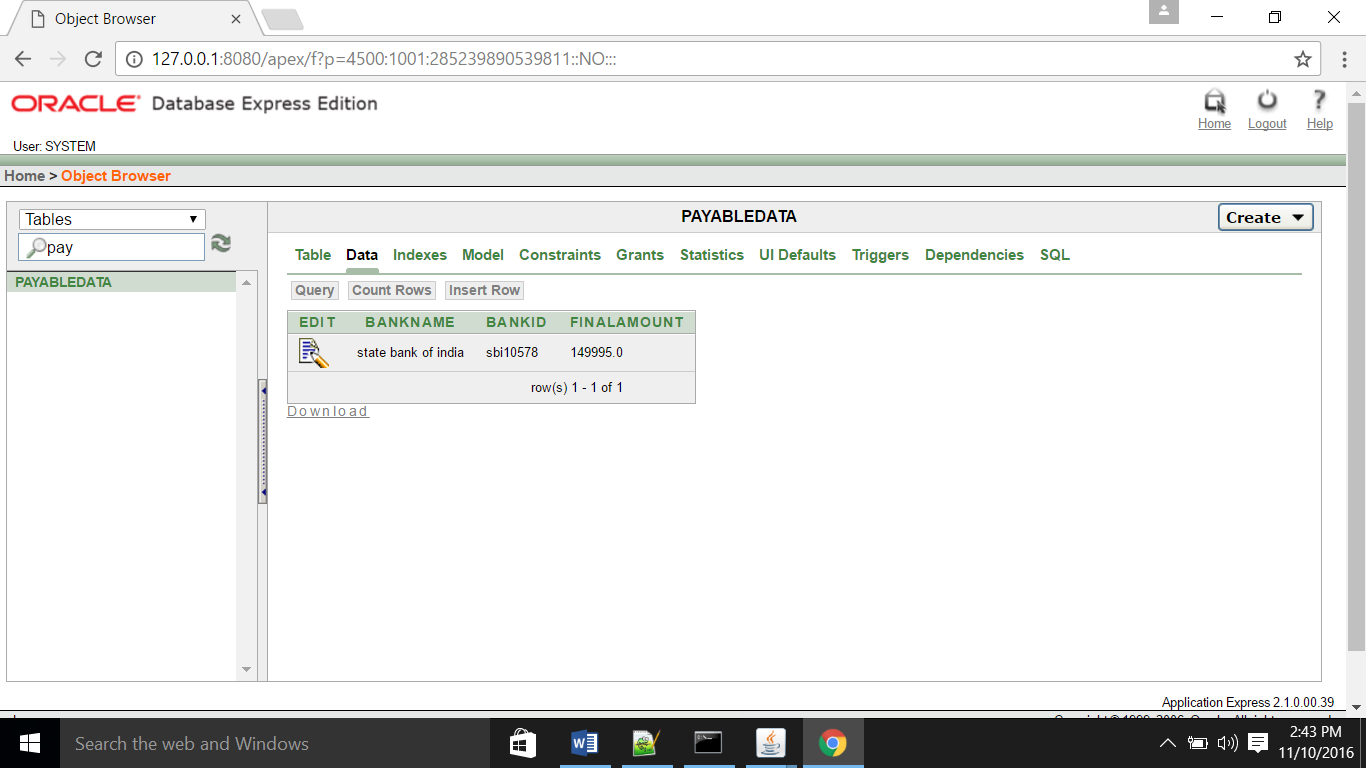
**ReceiverDetails:-**



**ReceiverDetails:-**

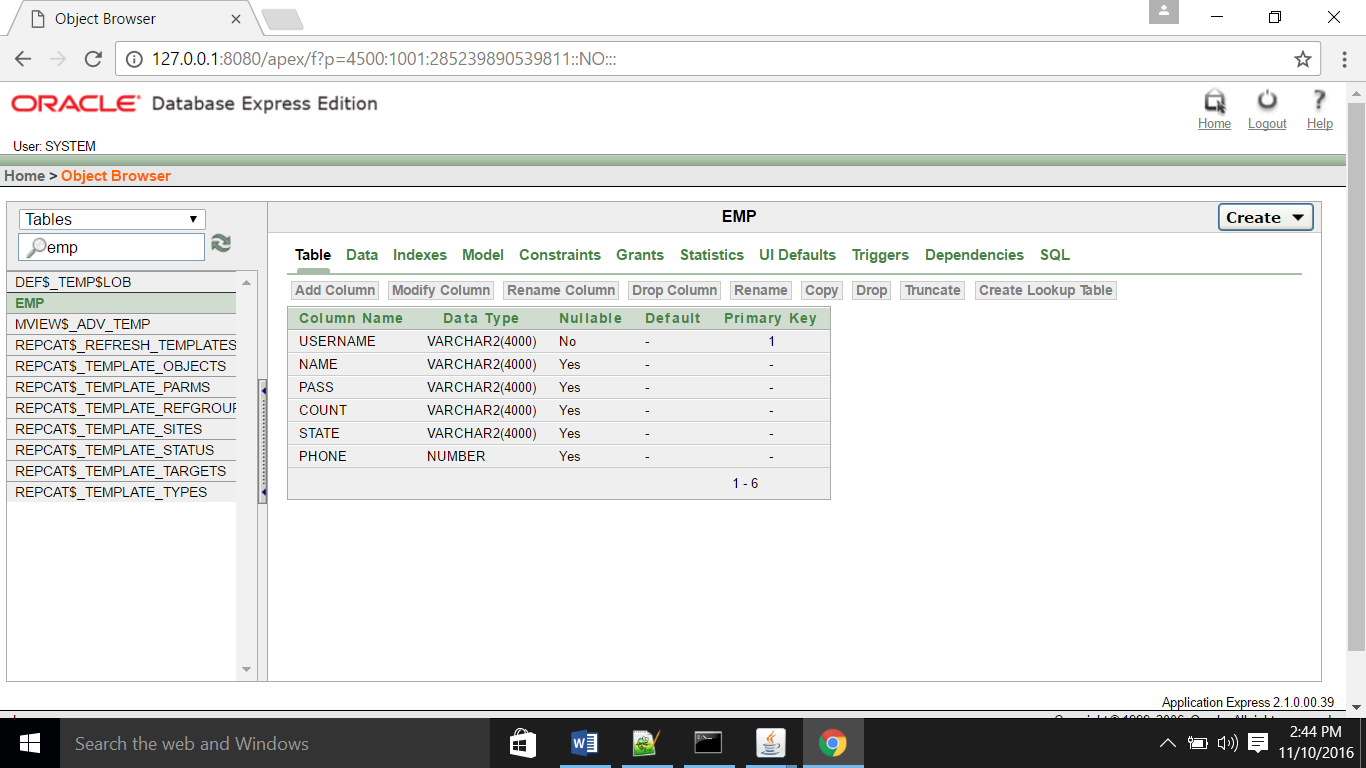


**PayableData:-**

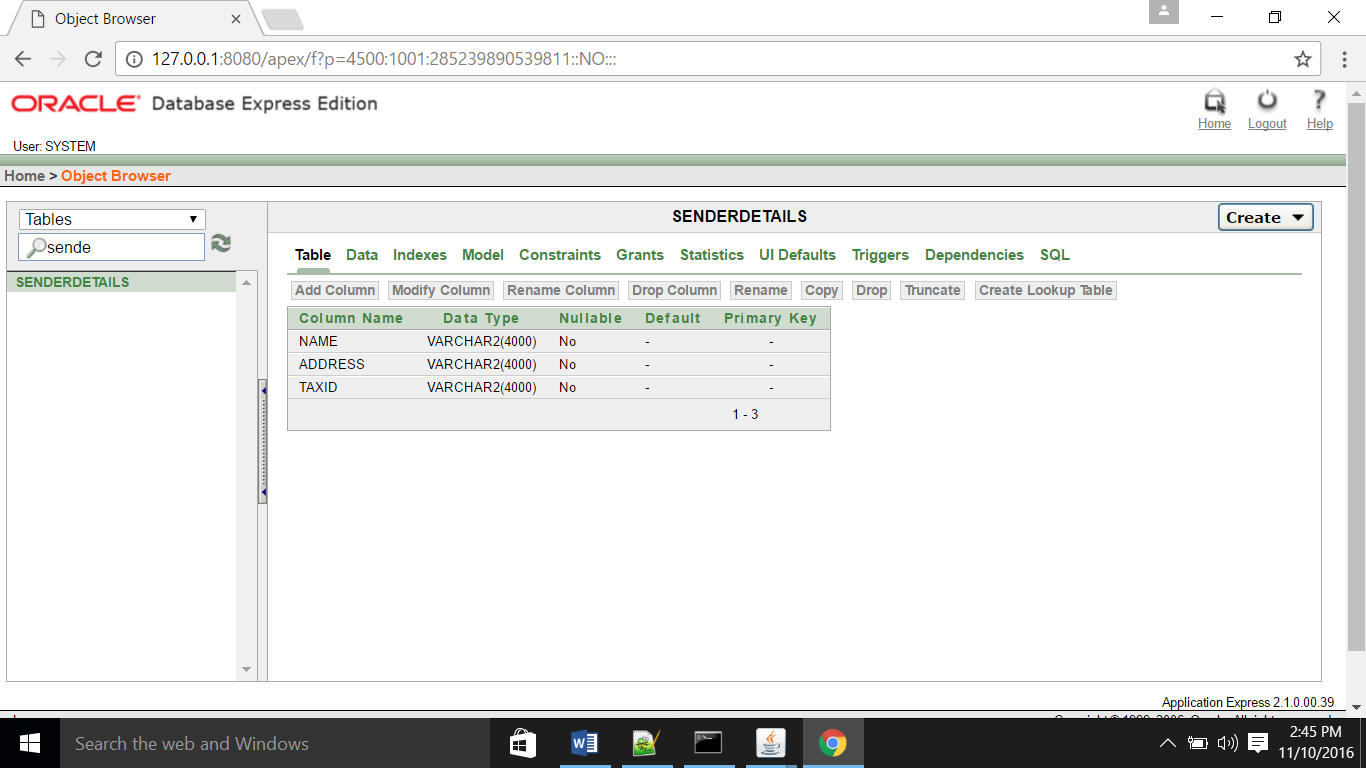


**Tables:**

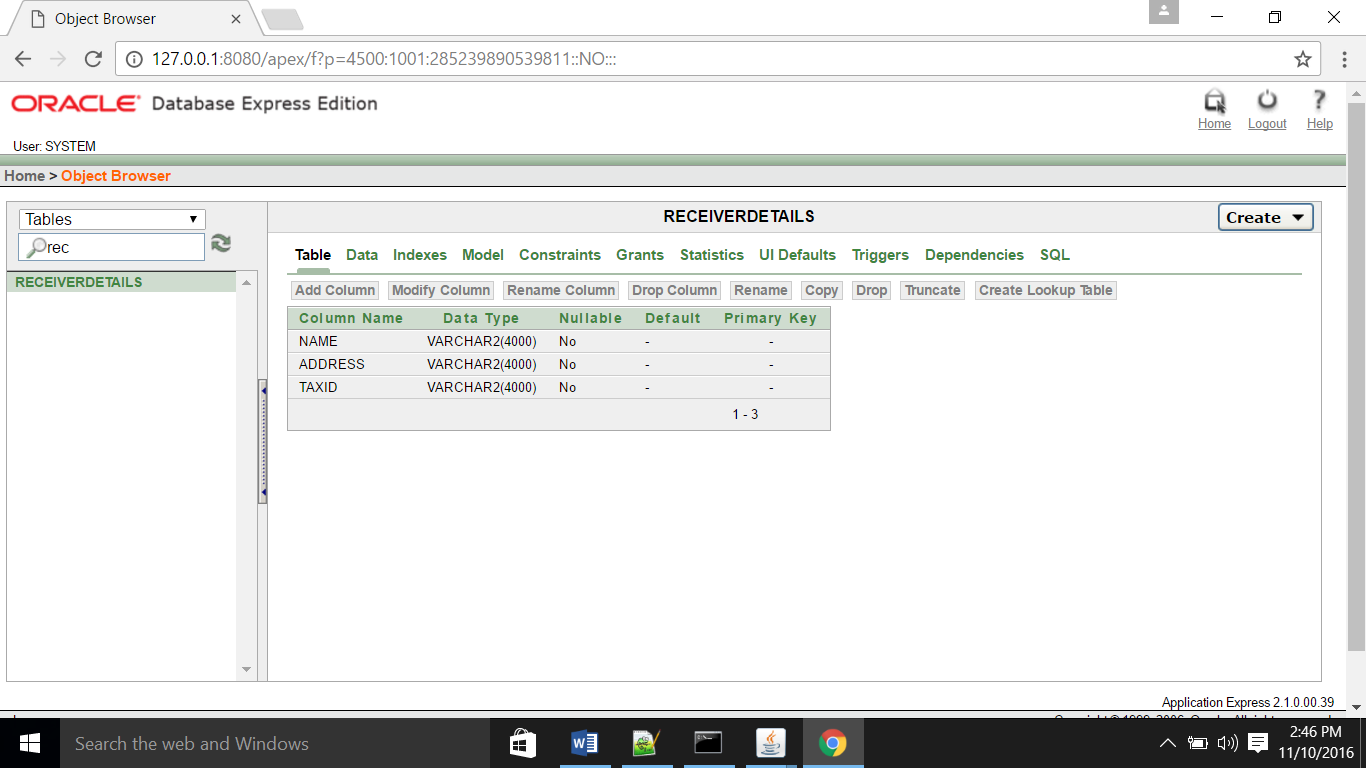
Emp Table:-



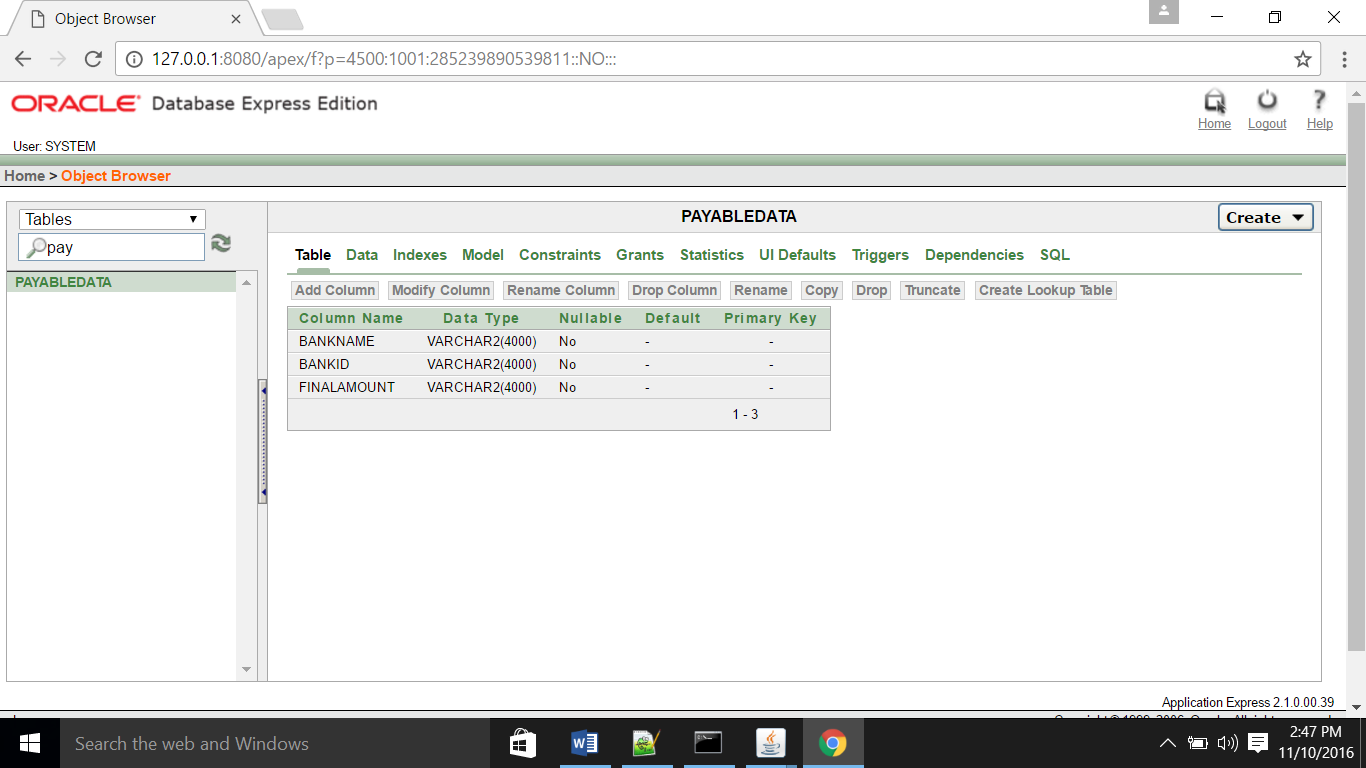
SenderDetails Table:-



ReceiverDetails Table:-



Payabledata Table:-



CustomerData Table:-

