**PROJECT REPORT**

**ON**

SHOP MANAGEMENT SYSTEM

(CS V Semester DBMS project)

**2015-16**



**Submitted to: Submitted by:**

Mr. B. P. Dubey Mr. KARAN BHUTYANI

Mr. DHRUV GOTHWAL

Ms. SHANIYA NISHA

Mr. VIKALP SAJWAN

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

**GRAPHIC ERA HILL UNVERSITY, DEHRADUN**

**CERTIFICATE**

### This is to certify that Mr. Karan Bhutyani (Roll No.- 1011281), Mr. Dhruv Gothwal (Roll No.- 1011269), Ms. Shaniya Nisha (Roll No.- 1011322) and Mr. Vikalp Sajwan (Roll No.- 1011342) have developed DBMS based project on “Shop Management System” for the CS V Semester DBMS Lab (PCS504) in Graphic Era Hill University, Dehradun under my supervision. The project carried out by Students is their own work as best of my knowledge.

Date:

(Mr. B. P. Dubey)

**SUPERVISOR**

Assistant Professor

Department of CS / IT GEHU, Dehradun

**ACKNOWLEDGMENT**

We wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

We would like to thank particularly our Supervisor Mr. B. P. Dubey for his patience, support and encouragement throughout the completion of this project and having faith in us.

We also acknowledge to all teachers who have taught us other subject like Java Programming by Mr. Mahesh Manchanda which is helpful in developing the project.

At last but not the least We greatly indebted to all other persons who directly or indirectly helped us during this work.

**Mr. KARAN BHUTYANI Mr. DHRUV GOTHWAL**

**(Roll No.- 1011281) (Roll No.- 1011269)**

**Ms. SHANIYA NISHA Mr. VIKALP SAJWAN**

**(Roll No.- 1011322) (Roll No.- 1011342)TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**LIST OF TABLES vi**

**LIST OF FIGURES vii**

**LIST OF SYMBOLS, ABBREVIATIONS viii**

**1. INTRODUCTION** 1

1.1 About Project 1

1.2 Java 2

1.2.1 Eclipse IDE 2

1.2.2 Swings 3

1.2.3 JDBC 3

1.3 DBMS 4

1.3.1 RDBMS 4

1.3.2 E-R Model 4

1.3.3 SQL 5

1.3.4 MySQL 5

**2. PROJECT 6**

2.1 Requirement Analysis 6

2.2 Software Specification 7

2.3 Database Description 8

2.3.1 Database Schema 8

2.3.2 Database Dictionary 9

2.4 ER Diagram 11

2.5 Modules 14

2.5.1 Welcome window 14

2.5.2 Shopping window 15

2.5.3 Customer Login Dialog box 15

2.5.4 Payment window 15

2.5.5 New Customer Registration 16

2.5.6 Previous Records 16

**3. SNAPSHOT OF PROJECT 18**

3.1 New Purchase 18

3.1.1 Registered Customer 22

3.1.2 Guest Customer 24

3.2 New Customer Registration 25

3.3. Purchase Records 27

**4. CONCLUSION 30**

4.1 SUMMARY 30

4.2 FUTURE WORKS 30

**APPENDIX:** CODE 31

**REFERENCE** 63

**LIST OF TABLES**

**TABLE Page No.**

Table 1 Software Specification 7

Table 2 customers table Data Dictionary 9

Table 3 item\_catagories table Data Dictionary 9

Table 4 payment\_type table Data Dictionary 9

Table 5 purchase table Data Dictionary 10

Table 6 items table Data Dictionary 10

**LIST OF FIGURES**

**FIGURE Page No.**

Figure 1: Cardinality notation used in ER diagram 14

Figure 2: ER diagram symbols 14

Figure 3: ER diagram of the Project Database 13

Figure 4: Flow of execution of software within graphical interface. 17

**LIST OF ABBREVIATIONS**

DBMS Data Base Management System

RDBMS Relational Data Base Management System

SQL Structure Query Language

E-R Entity Relationship

JDBC Java Database Connectivity

IDE Integrated Development Environment

JVM Java Virtual Machine

JRE Java Runtime Environment

GUI Graphical User Interface

**CHAPTER** **1**

**INTRODUCTION**

* 1. **ABOUT PROJECT**

In today’s world every business is being computerized. Giant enterprise companies are already totally computerized but with the computer technology getting cheaper with time, smaller businesses such as small scale shops are also computerizing their businesses. An inexpensive computer system can help these type of shops to increase productivity and prevent all the hassle in maintaining records which are faced while keeping paper records.

**The Shop Management System**

**“The Shop Management System”** is an almost required solution regarding the Shop. This software package provides guidance for all the shop purpose, as a perfect guide, the current demand for such software became needful. This project will provide for computerization of a small Shop whose main goal is to keep track on their inventory and billing process and wants to change from paper based transaction to computerized transaction.

**“The Shop Management System”** will make storing of the item stock records, purchase information, and customer information a lot easier. The stock information in the database will automatically generate bills when the customer items are scanned. The purchase history can be retrieved promptly.

The product could be implemented in a small sized organization, which will not be very keen on spending loads of money on ledgers. Whereas our product will greatly reduce the running by using common and cheap office items like database and desktop application. And also there is no requirement to store books or accounts. The data is directly stored in the database.

**1.2 JAVA**

In this project Java has been used as the Programming and Front-End tool.

**Java** is a general purpose programming language with a number of features that make the language well suited for use on this Project. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces an 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

**JAVA FEATURES**:

* Object Oriented Programming Language: Unlike C++, Java is purely OOP.
* Robust (Strong/ Powerful): Most programs in use today fail for one of the two reasons: memory management or exceptional conditions. Thus, the ability to create robust programs was given a high priority in the design of Java. Java forces the user to find mistake in the early stages of program development. Java checks code at compilation time. However, it also checks the code at run time also. Java programs will not crash because of its exception handling and its memory management features.
* Interpreted: Java programs are compiled to generate the byte code. This byte code can be downloaded and interpreted by the interpreter. .class file will have byte code instructions and JVM which contains an interpreter will execute the byte code.
* Portable: In Java, many types of computers and operating system are in use throughout the world and are connected to the Internet. For downloading programs through different plateforms connected to the Internet, some portable, executable code is needed. Java does not have implementation dependent aspects and it yields or gives same result on any machine.
* Architectural Neutral Language: Java development team work on the philosophy of write once; run anywhere, anytime, forever” and as a result the Java Virtual Machine (JVM) was developed. Only the JVM can execute the bytecode. Java byte code is not machine dependent, it can run on any machine with any processor and with any OS.
* High Performance: Along with interpreter there will be JIT (Just In Time) compiler which enhances the speed of execution.

1.2.1 **ECLIPSE Integrated Development Environment (IDE)**

ECLIPSE IDE : Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins.

1.2.2 **SWINGS**

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swings was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

1.2.3 **Java Database Connectivity (JDBC)**

JDBC is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases such as MySQL.

The JDBC library includes APIs for each of the tasks commonly associated with database usage:

* Making a connection to a database
* Creating SQL or MySQL statements
* Executing that SQL or MySQL queries in the database
* Viewing & Modifying the resulting records
  1. **Database Management System (DBMS)**

In this project MySQL RDBMS is used for creating and maintaining the shop database.

A **Database Management System** (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyse data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle etc. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS

* + 1. **RDBMS**

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd. A RDBMS lets you create, update, and administer a relational database. RDBMSs are a common choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data, and other applications since the 1980s. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use.

* + 1. **E-R Model**

An entity–relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

ER Model only visualize business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

* + 1. **SQL**

SQL (Structured Query Language) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and a data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

* + 1. **MySQL**

MySQL is an open-source relational database management system (RDBMS). The SQL acronym stands for Structured Query Language. Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years.

**CHAPTER** **2**

**PROJECT**

**2.1 REQUIREMENT ANALYSIS**

2.1.1 **Purpose**

The **Shop Management System** will make storing of the item stock records, purchase information, and customer information a lot easier. The supplier and sales information in the database it will automatically generate bills when the customer buys the items. The purchase history can be retrieved promptly. And records will be generated based on different criteria.

2.1.2 **Scope**

The project could be implemented in an average sized organization. An average company will not be very keen on spending loads of money on ledgers. Whereas our project will greatly reduce the costs which is using common and cheap office items like database and desktop application. And also there is no requirement to store books or accounts. The data is directly stored in the database in the hard disk of the PC.

2.1.3 **Overview**

This project “**Shop Management System**” is software part of the Shop Billing management system; on improving this software we can easily track all the sales, item detail and purchase detail, bill detail and also we can able generate report. The main goal of the software is build a good management tool. The shop management will make storing of the customer records, sales detail, purchase information in the database it will automatically generate bills when the customers purchase items. The customer purchase and item stock history can be retrieved very promptly. And reports should be generated based on different criteria.

**2.2 SOFTWARE AND HARDWARE SPECIFICATION USED FOR DEVELOPMENT**

This is the hardware and software specification of the system used for the development of this project and is recommended for a smooth development process of such projects.

2.2.1 **HARDWARE**

* Intel i3 processor
* Hard Disk capacity – 80 GB (with at least 20GB free space)
* 4 GB RAM

2.2.2 **SOFTWARE**

* JDK 1.7.0
* MySQL 5.6.17
* ECLIPSE LUNA IDE

**2.2 SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION FOR USER OF THIS SOFTWARE**

This is the recommended hardware and software specification of the system that is to be used for running this software.

2.2.1 **HARDWARE**

* Pentium 3 processor or higher
* Hard Disk capacity – at least 5 GB free space
* Minimum 1 GB RAM

2.2.2 **SOFTWARE**

* JRE version 1.7 or up
* MySQL 5.6.17 or up

**2.3 Database Description**

2.3.1 **DATABASE SCHEMA AND DESCRIPTION OF TABLES**

The database used in our application has following tables:

* items table
* customers table
* purchase table
* item\_catagories table
* payment\_type table
* The **items** table stores the details of items of the shop.

Fields: item\_id, item\_name, item\_details, item\_price, item\_catagory

Primary key: item\_id

Foreign key: item\_catagory

* The **customers** Table stores the information about registered customers.

Fields: cust\_id, cust\_name, cust\_mobile, cust\_points

Primary key: cust\_id

* The **purchase** Table stores the records of all the purchases made.

Fields: purchase\_id, cust\_id, items\_sold, total\_cost, payment\_type, time\_of\_purchase

Primary key : purchase\_id

Foreign keys: cust\_id, payment\_type

* The **item\_catagories** Table stores category names for the type of items that are sold in shop.

Fields: item\_cat\_id, item\_cat\_name

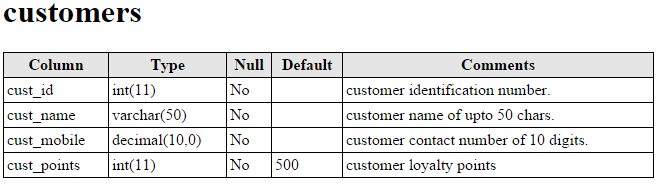
Primary key: item\_cat\_id

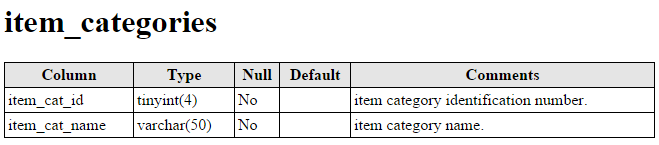
* The **payment\_type** Table stores the modes of payment that are available in the shop.

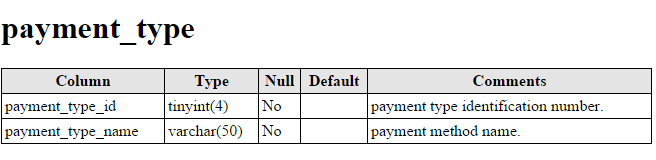
Fields: payment\_type\_id, payment\_type\_name

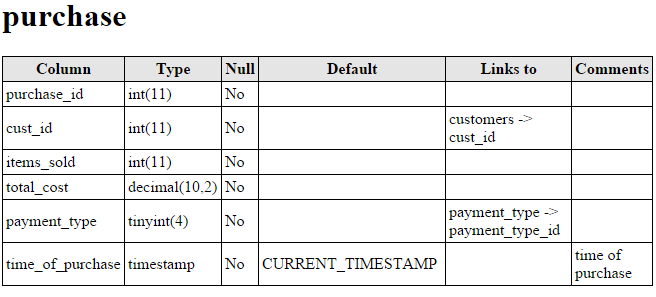
Primary key: payment\_type\_id

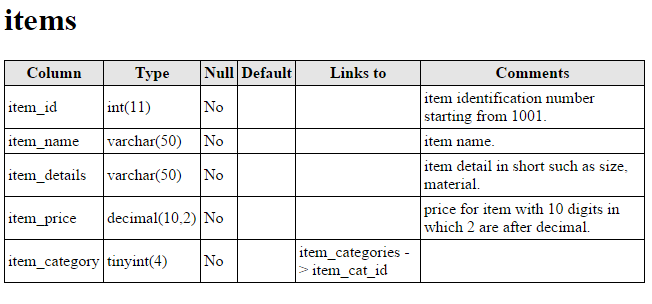
2.3.2 **DATABASE DICTIONARY**











**2.4 ER Diagram**

An Entity Relation(ER) Diagram is a specialized graphics that illustrates the interrelationship between entities in a database. ER diagrams often use symbols to represent 3 different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

An Entity Relationship Model (ERM), in software engineering is an abstract and conceptual representation of data. Entity Relationship modelling is a relational schema database modelling method, used to produce a type of conceptual schema or semantic data model of a system, often a relation database, and its requirements in a top-down fashion

**Entity:**

The thing which we want to store information. It is an elementary basic building block of storing information about business process. An entity represents an object defined within the information system about which you want to store information. Entities are distinct things in the enterprise.

**Relationships:**

A relationship is a named collection or association between entities or used to relate two or more entities with some common attributes or meaningful interaction between the objects.

**Attributes:**

Attributes are the properties of the entities and relationship, Descriptor of the entity. Attributes are elementary pieces of information attached to an entity.

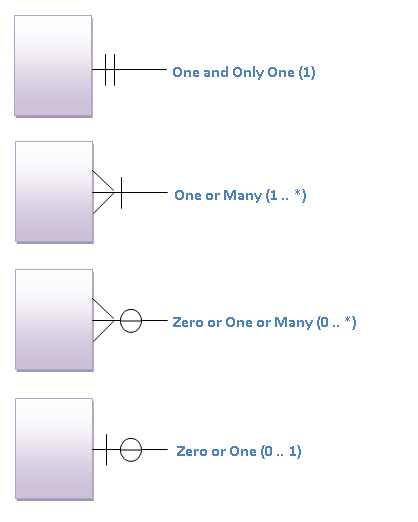


Figure 1 Cardinality notation used in ER diagram.

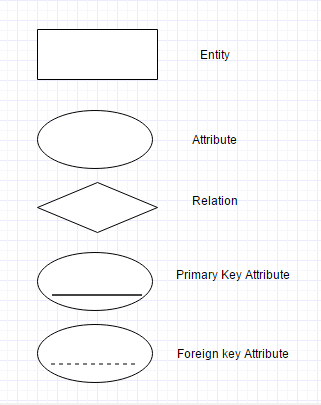


Figure 2 ER diagram symbols

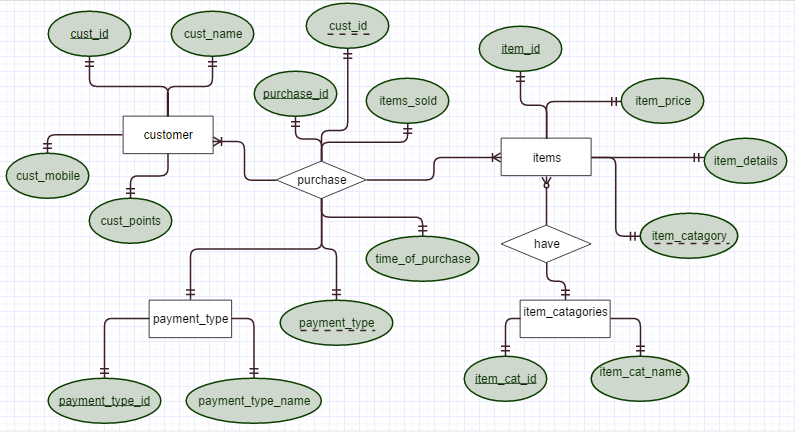


Figure 3 ER diagram of the Database

**2.5 MODULES**

The software is divided into 9 modules. Of which 6 modules are modules with graphical interface classes and 3 modules are background classes which have their own background function.

2.5.1 **GUI MODULES**

2.5.1.1 **WELCOME WINDOW**

This is the first graphical window that user sees on executing the software. In this window the user can further choose the operation he wants to perform.

SUB MODULES – New Purchase, Customer Registration, Previous Records

2.5.1.2 **NEW CUSTOMER REGISTRATION**

In this window user can register a new customer. The details required for new customer registration are NAME and MOBILE NUMBER. Upon successful registration, new customer information is inserted in the Customer table in the database and the CUST ID allotted to CUSTOMER is displayed in an information dialog box.

SUB MODULES – nil.

2.5.1.3 **PREVIOUS RECORDS**

In this window user can view SALES STATISTICS or PURCHASE RECORDS. The interface allows to input preferences for the stats or records such as time window, type of customer etc. And based on those preferences data is read from the Database and is displayed systematically.

SUB MODULES – nil.

2.5.1.4 **NEW PURCHASE (SHOPPING PAGE)**

In this window the items along with their details are fetched from the Items table in the database and are shown in a tabbed pane. Each tab represents a category of items and displays the items of that category in a table. User can add items from these table to a CUSTOMER SHOPPING TABLE. Upon adding items in shopping list the total is shown in a TOTAL text field.

SUB MODULES – Customer Login Dialog

2.5.1.5 **CUSTOMER LOGIN**

In this dialog box, the user can login the registered customer by entering the CUSTOMER ID (this is equivalent to the customer providing with his registered customer card which is then read by bar code reader to get customer id). If the customer id is found in the Customer table in the database then the customer is logged in. Otherwise the customer can choose to shop as a guest customer.

SUB MODULES – Payment window

2.5.1.6 **PAYMENT PAGE**

The window has two interfaces which is based on the type of customer –

If the customer is registered, then the customer info is shown in the layout, and customer can choose to pay total bill or some part of bill amount from his loyalty points. Besides this the customer can choose the mode of payment that is CASH, CREDIT CARD or DEBIT CARD and complete the purchase.

If the customer is guest customer, then the customer can simply choose the mode of payment and complete the purchase.

On successful purchase, the record of purchase is stored in the Purchase table in the database and an information dialog box is shown. After clicking OK on the dialog box, the welcome window is shown.

SUB MODULES – nil

2.5.2 **NON GRAPHICAL MODULES**

2.5.2.1 **Controller Class**

* This is the main class that contains the main() method. So the execution of software begins from this class.
* This class consists of one object each of all other classes.
* This class contains the methods to traverse from one graphical interface to other. In this way it maintains the flow of execution between the elements of the software.
* All other graphical interface classes are unaware of each other and can only communicate with Controller class and DBConnection class. To achieve this a reference to an object of Controller class is passed to all the graphical classes.
* Whenever the user traverses from one graphical element to another, that graphical class calls the nextFrame() method of the Controller class and the appropriate code is run.

2.5.2.2 **DBConnection Class**

This is the class in which all the database operations are done. An object of this class is passed to the graphical classes that needs to do some database operations. And they can call the member methods of DBConnection class according to the need.

2.5.2.3 **Globals Class**

This class contains Global constants and Global variables to pass Information between the classes.

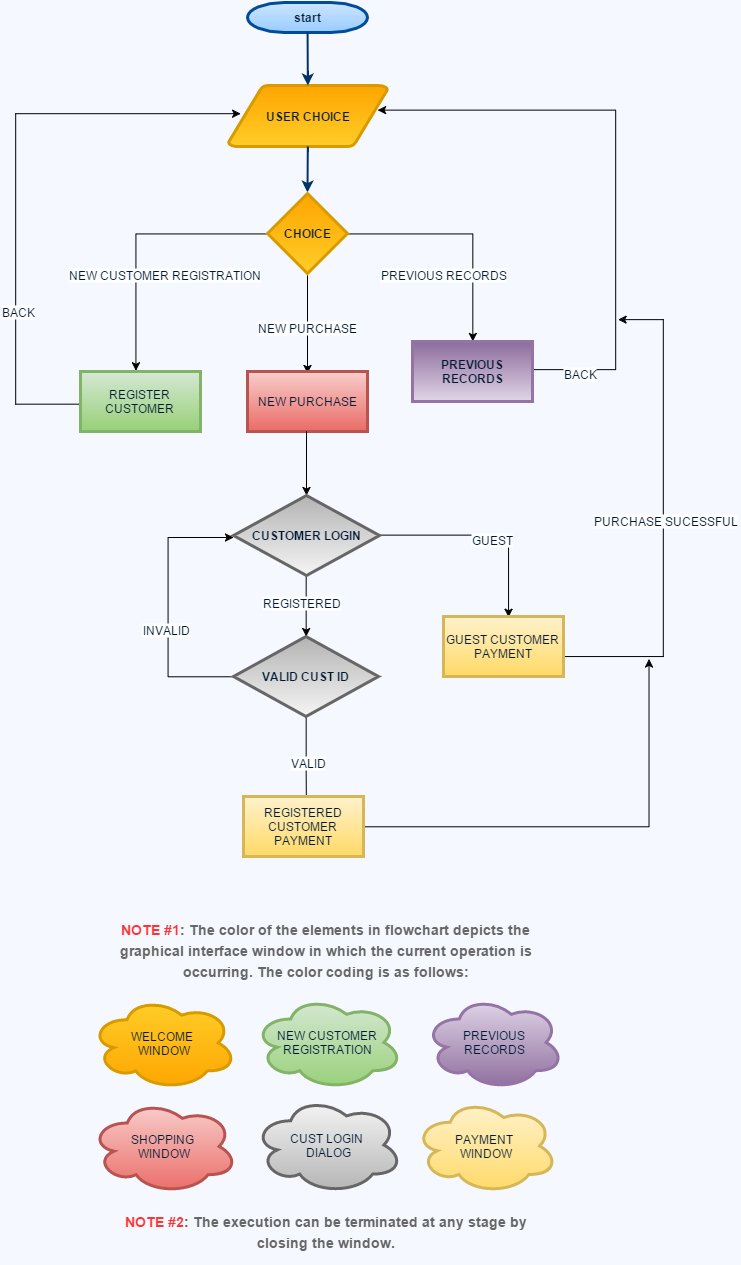
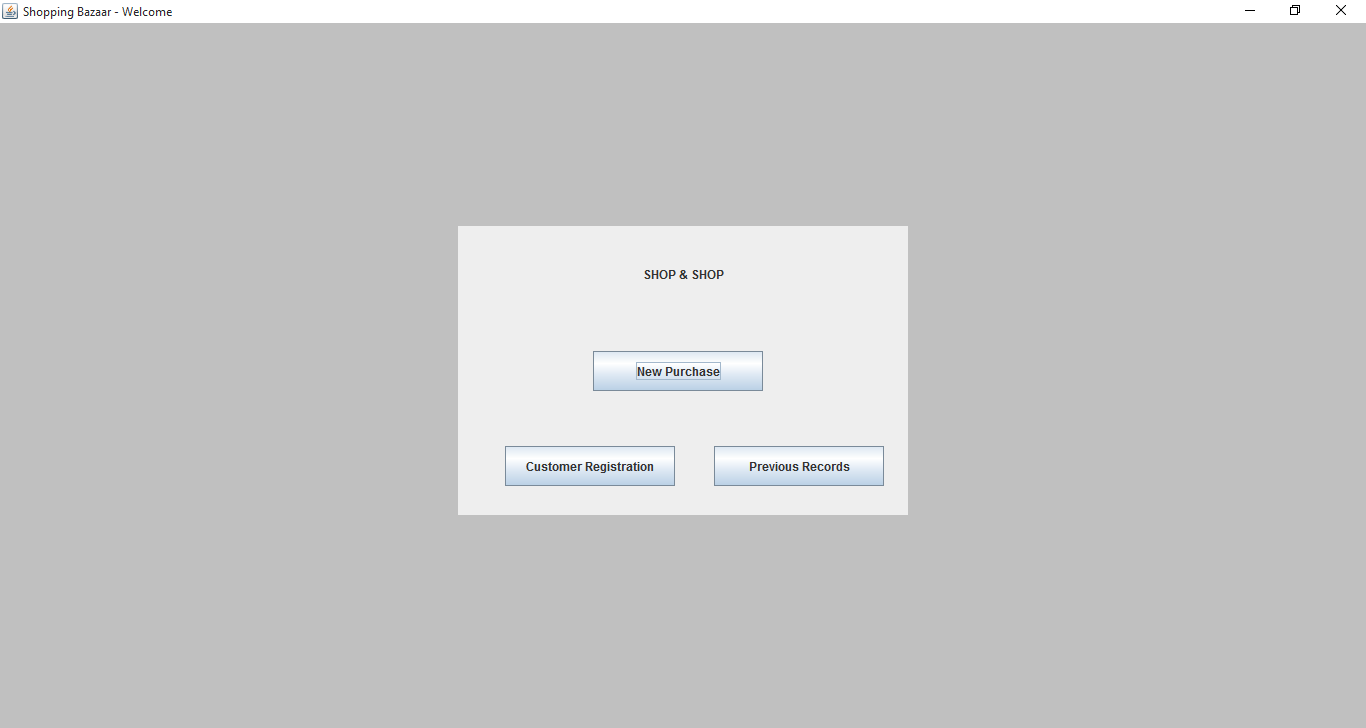


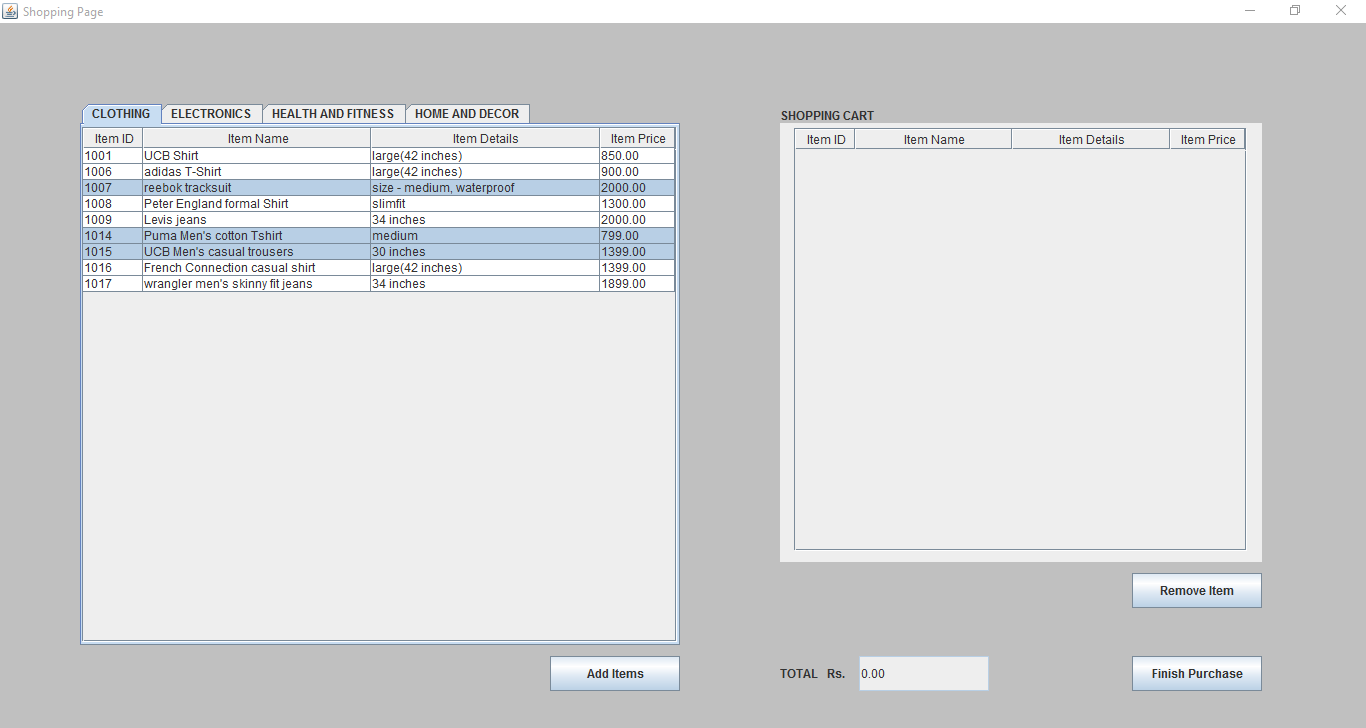
Figure 4: Flow of execution of software within graphical interface.

**CHAPTER** **3**

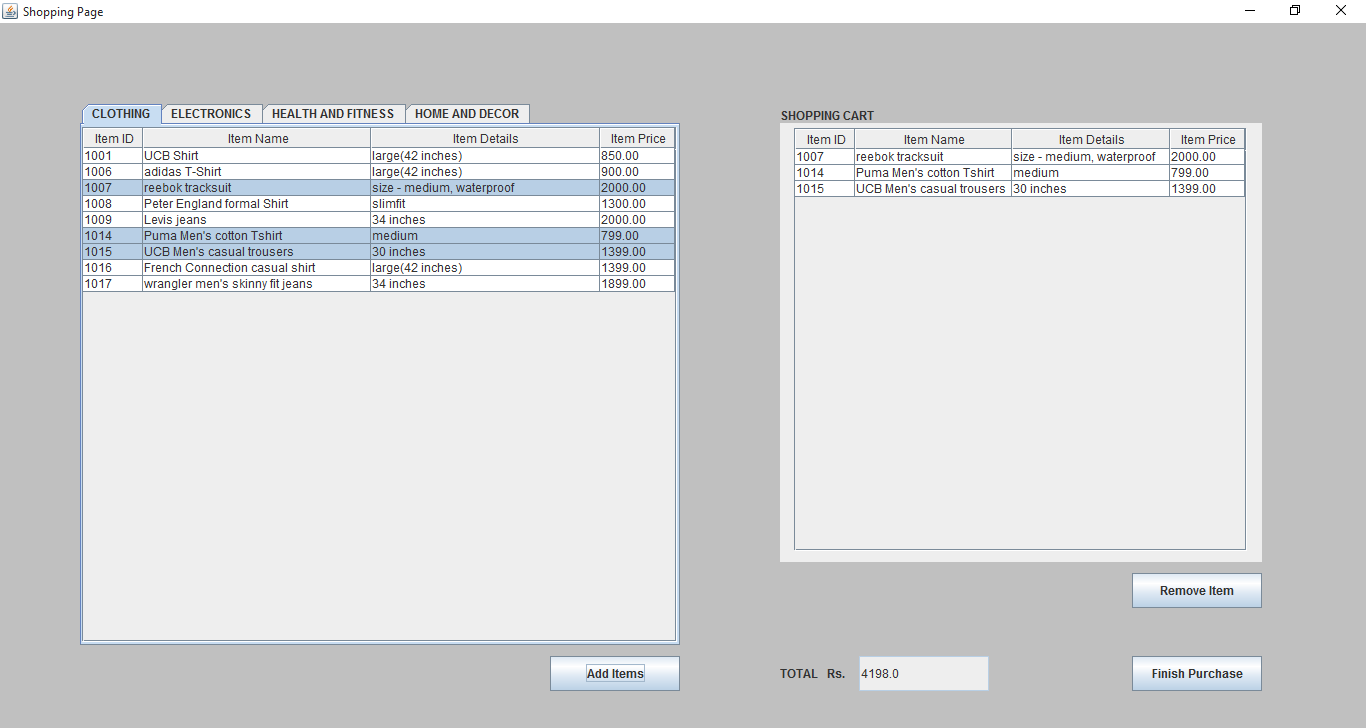
**SNAPSHOT OF PROJECT**

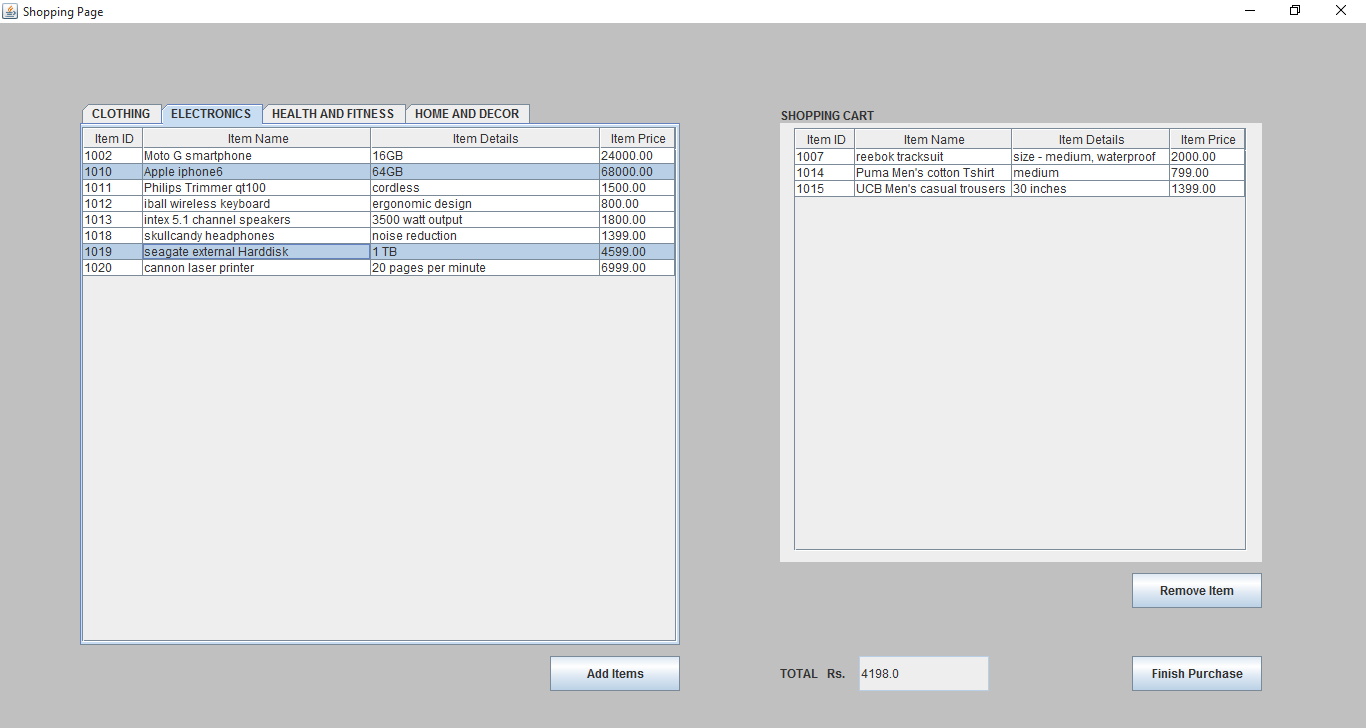
**3.1 New Purchase**



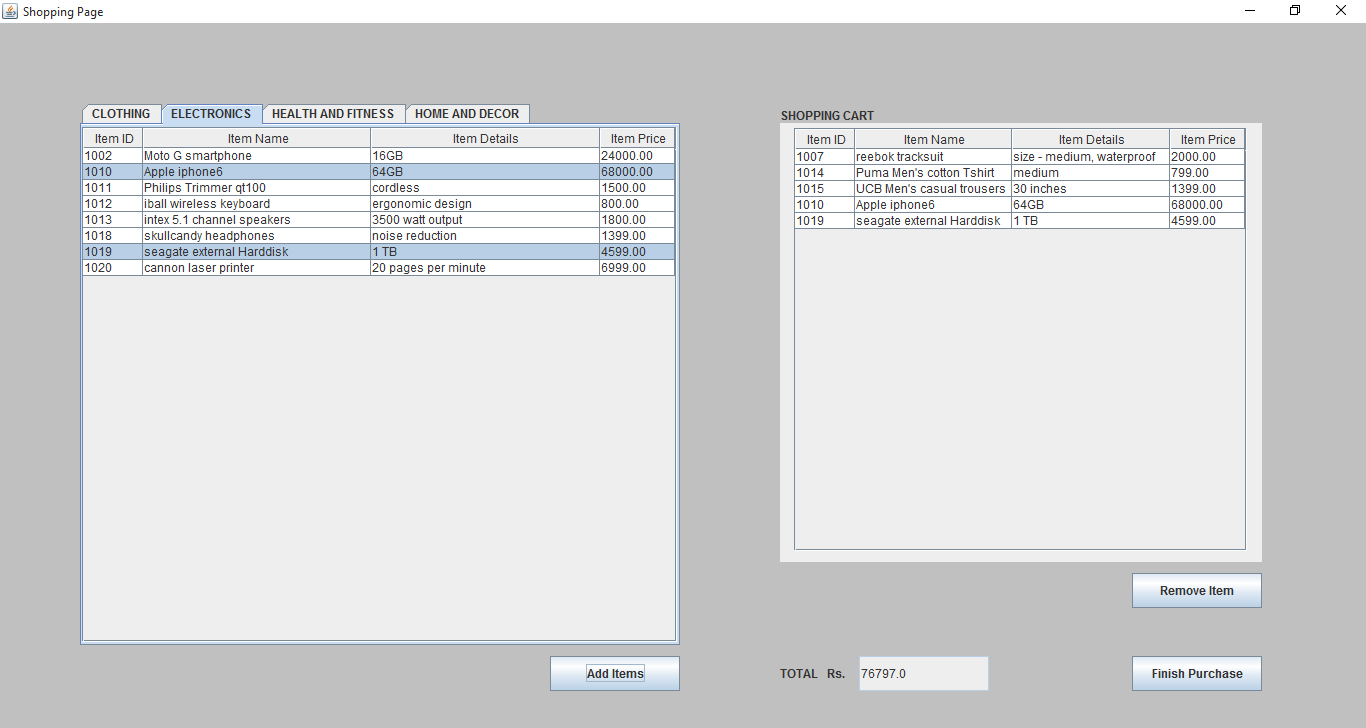


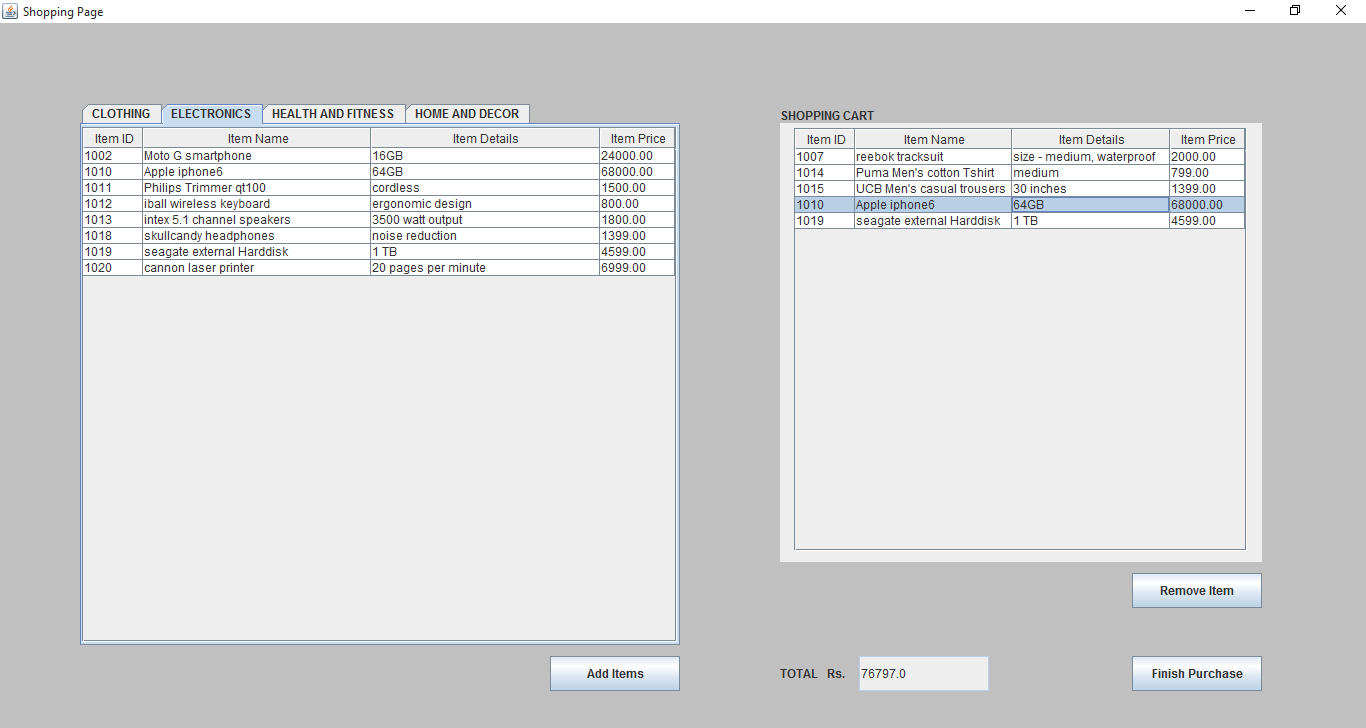
Adding clothing items to Shopping cart Table



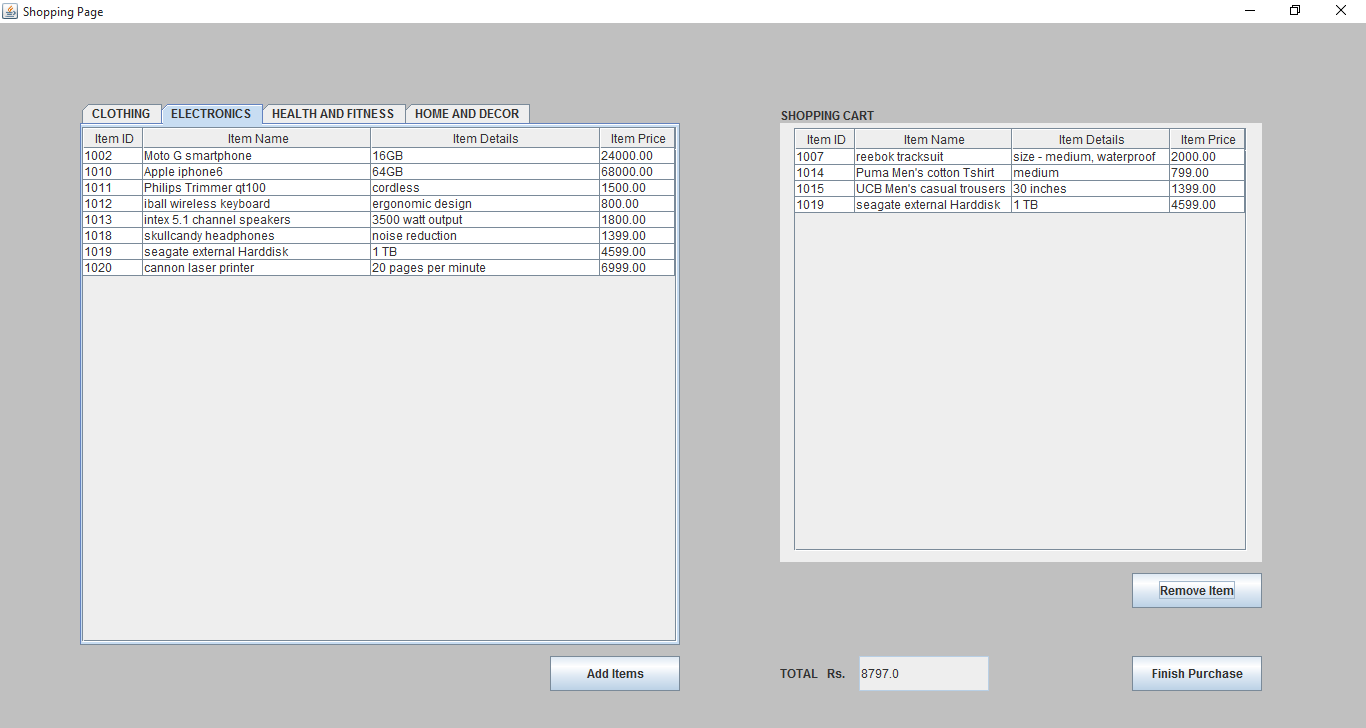


Adding electronic items to shopping cart table

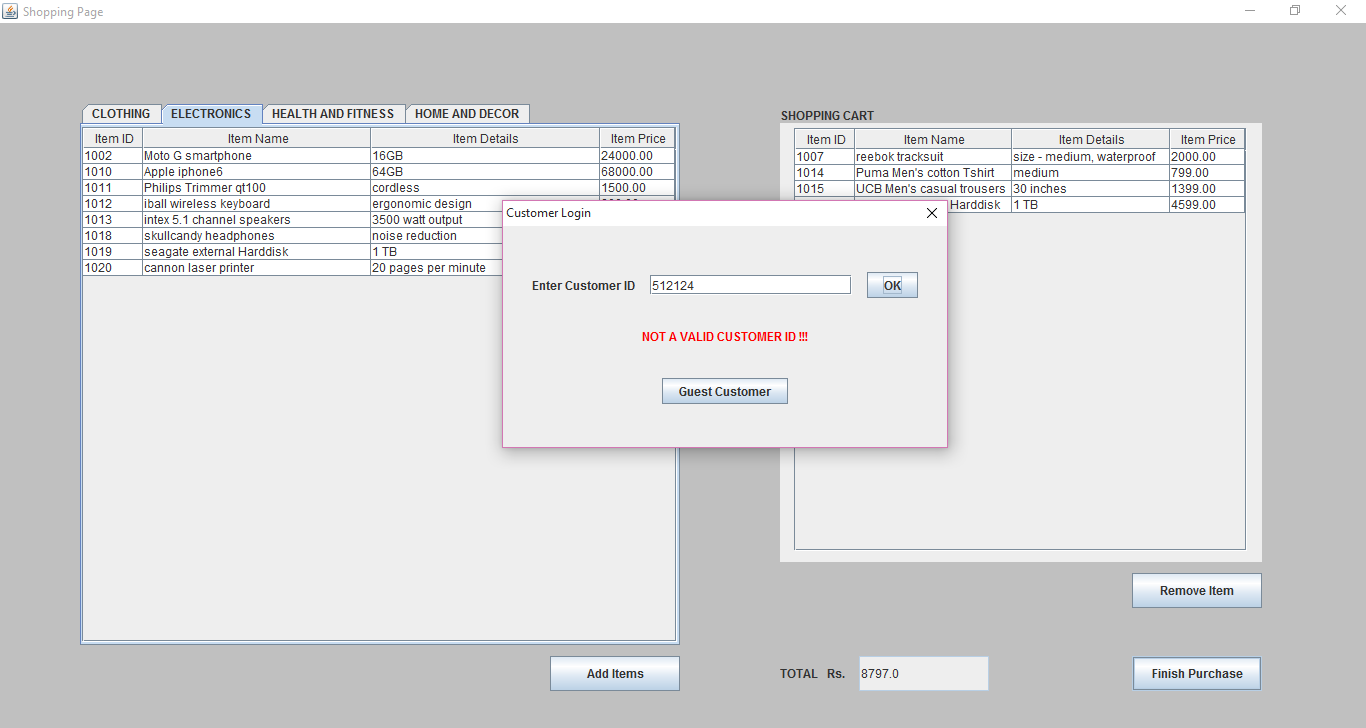




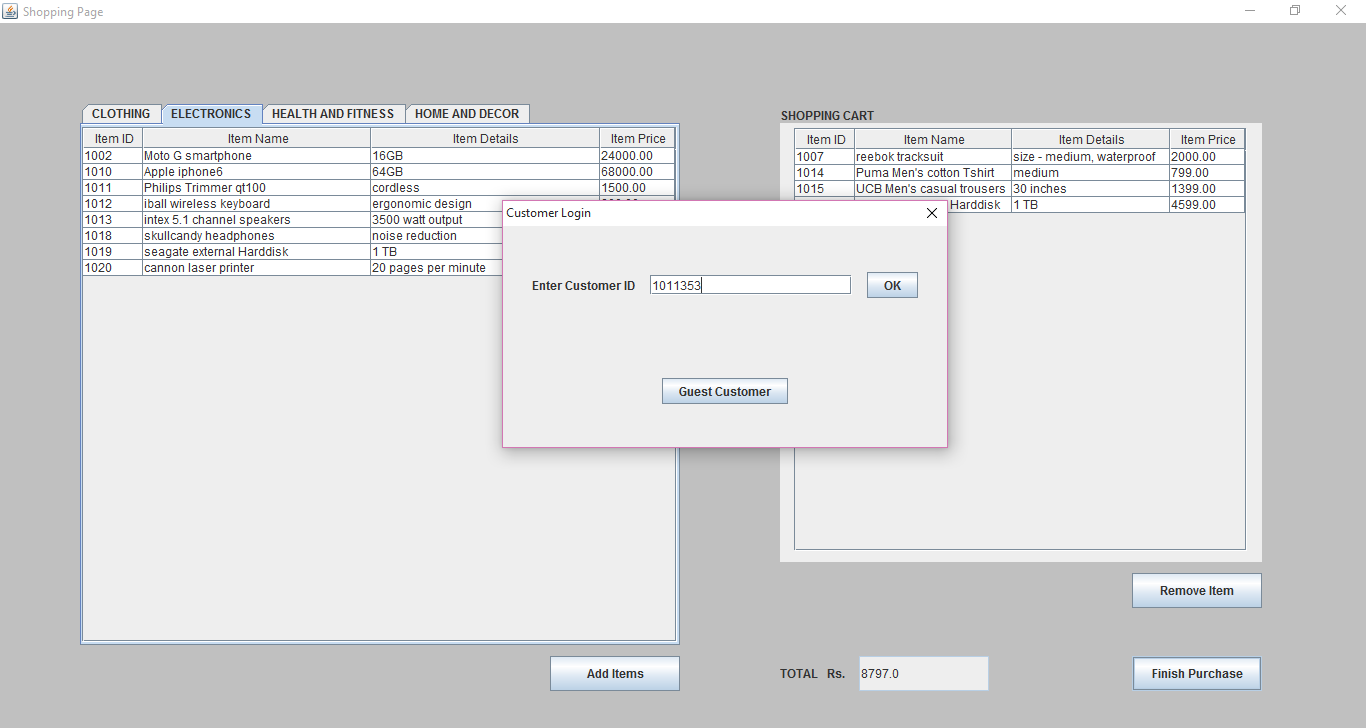
Removing item from shopping cart

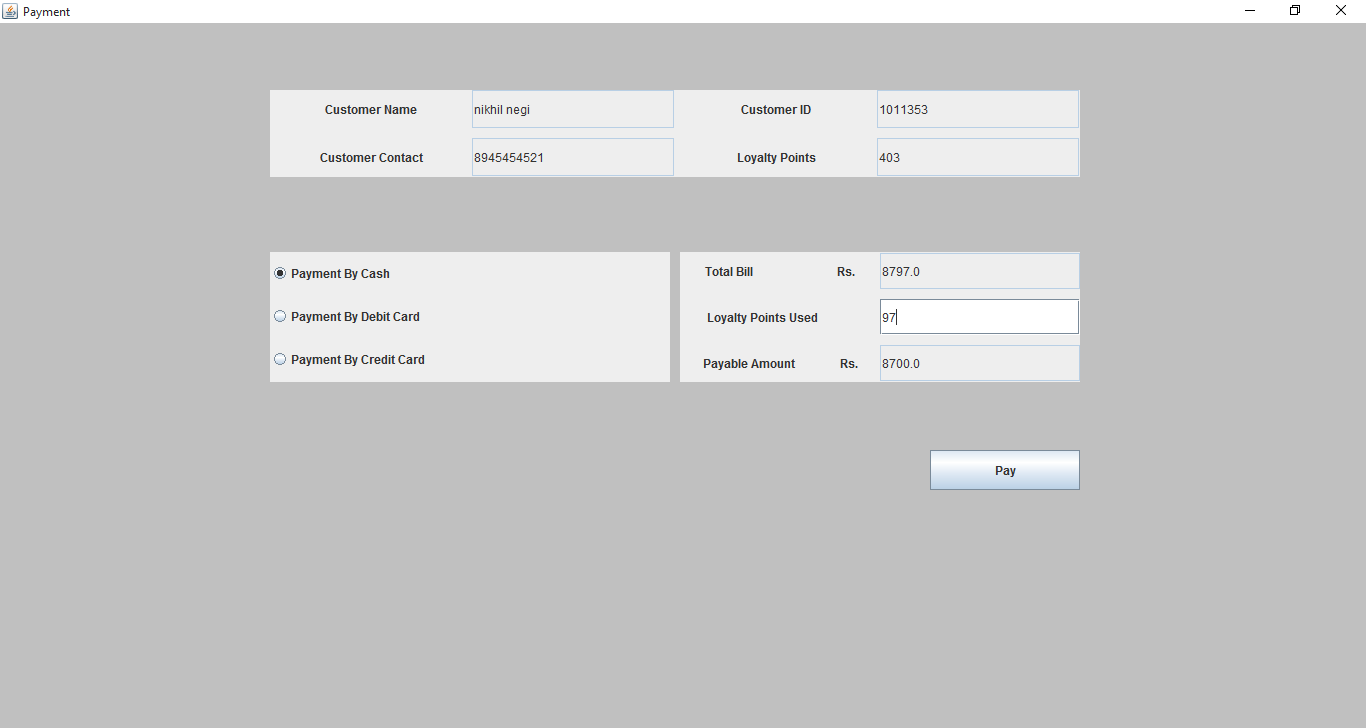


**3.1.1 Logging in Registered customer**

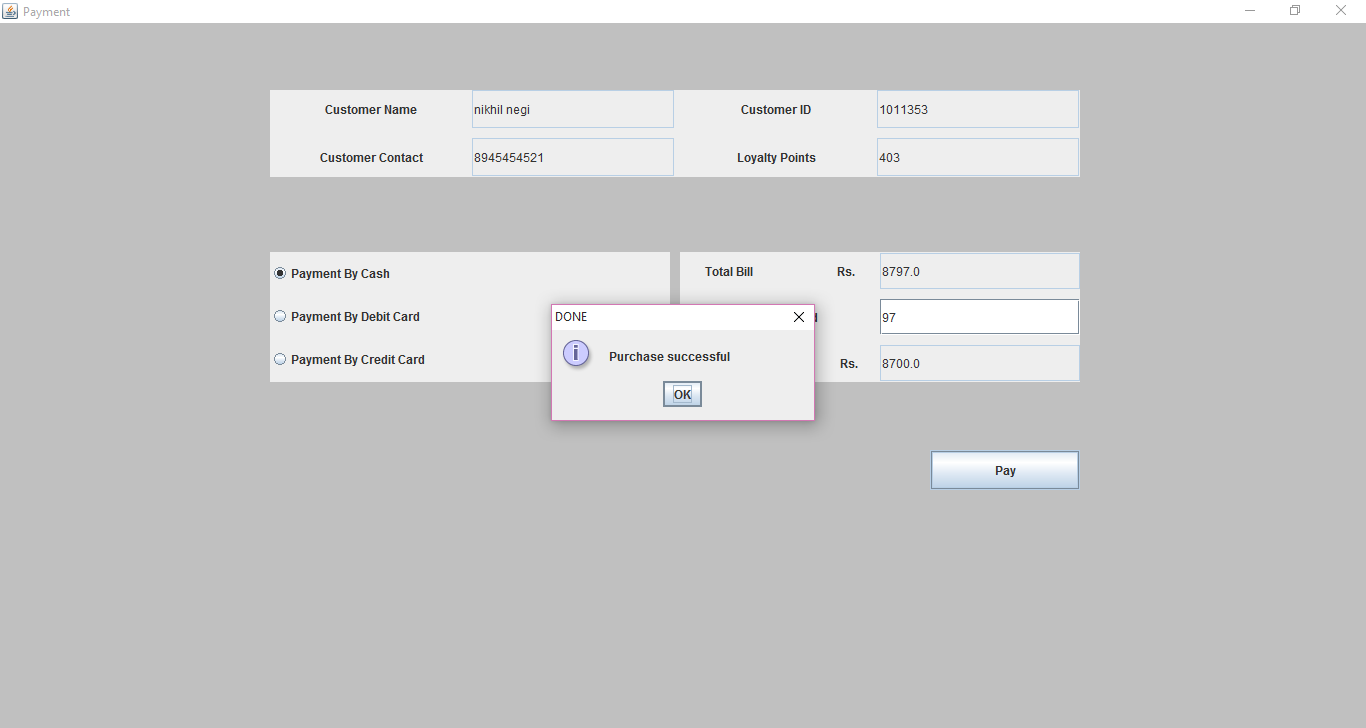


Trying to login with invalid id -> Error is displayed.

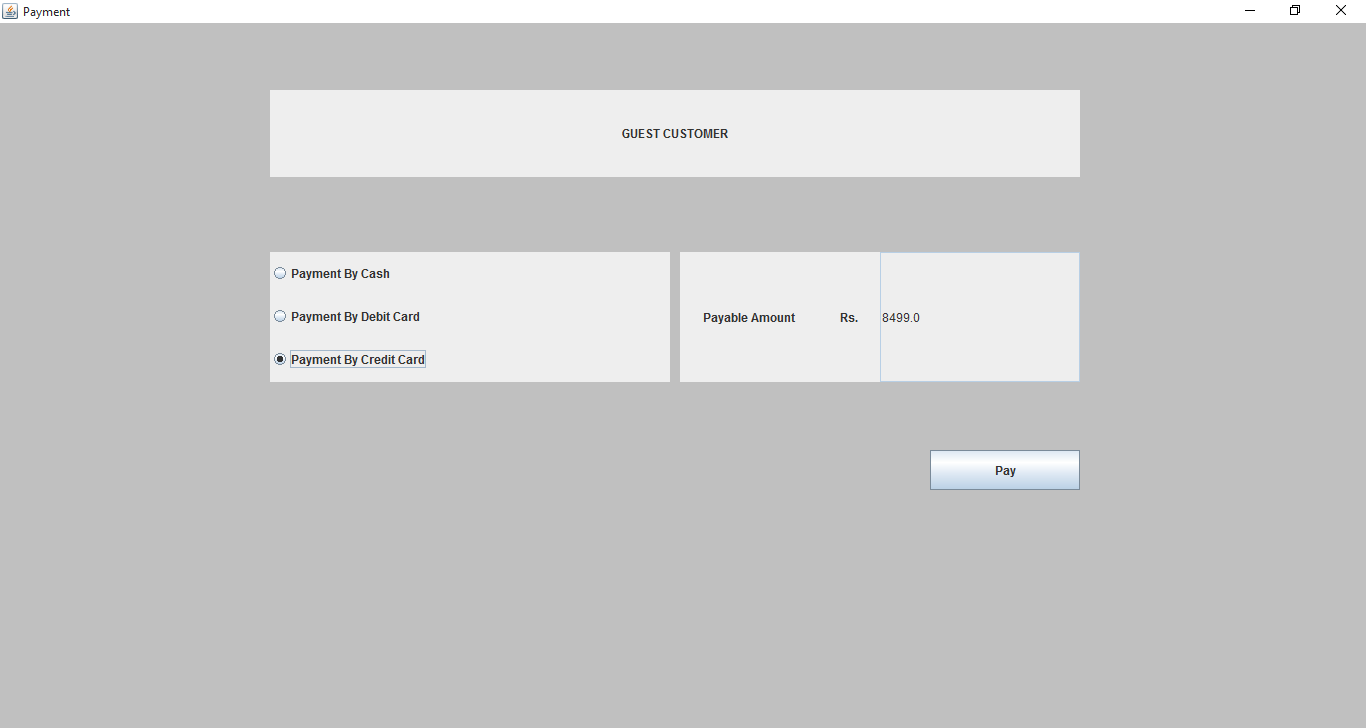


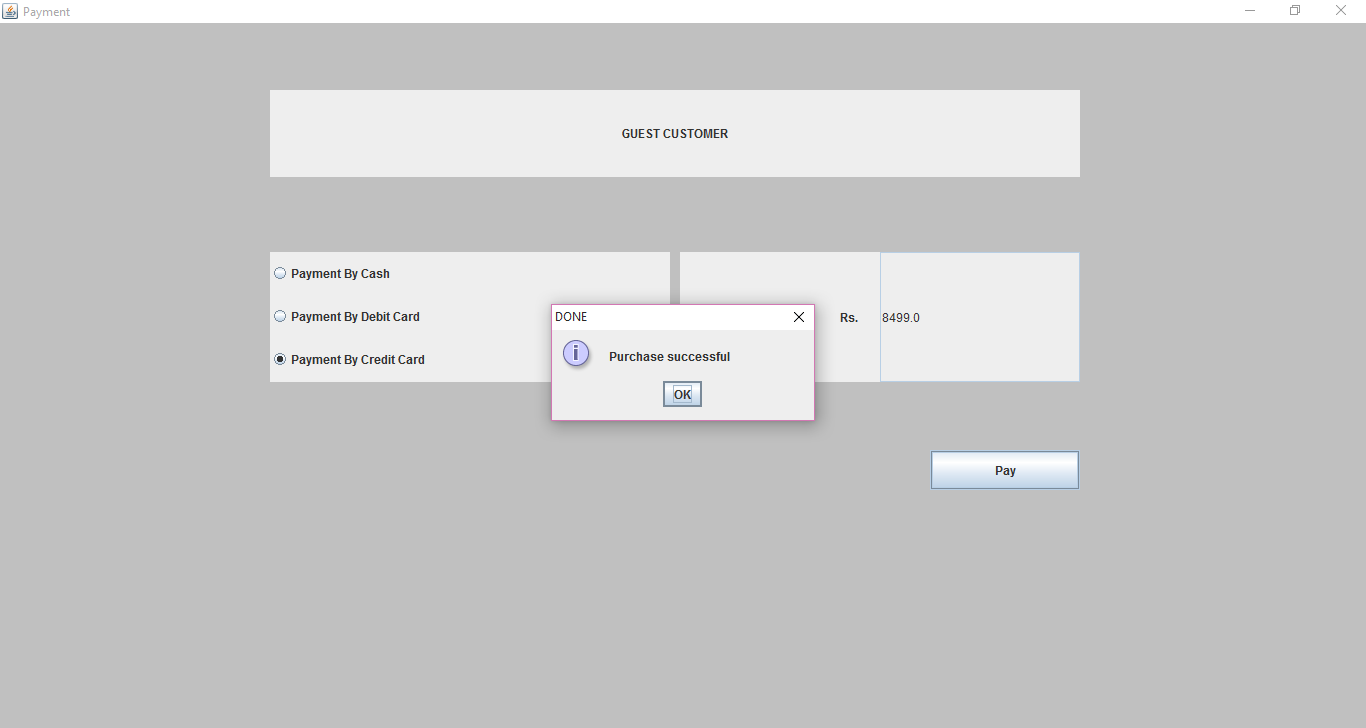


Using 97 points from loyalty points of customer to reduce bill amount.

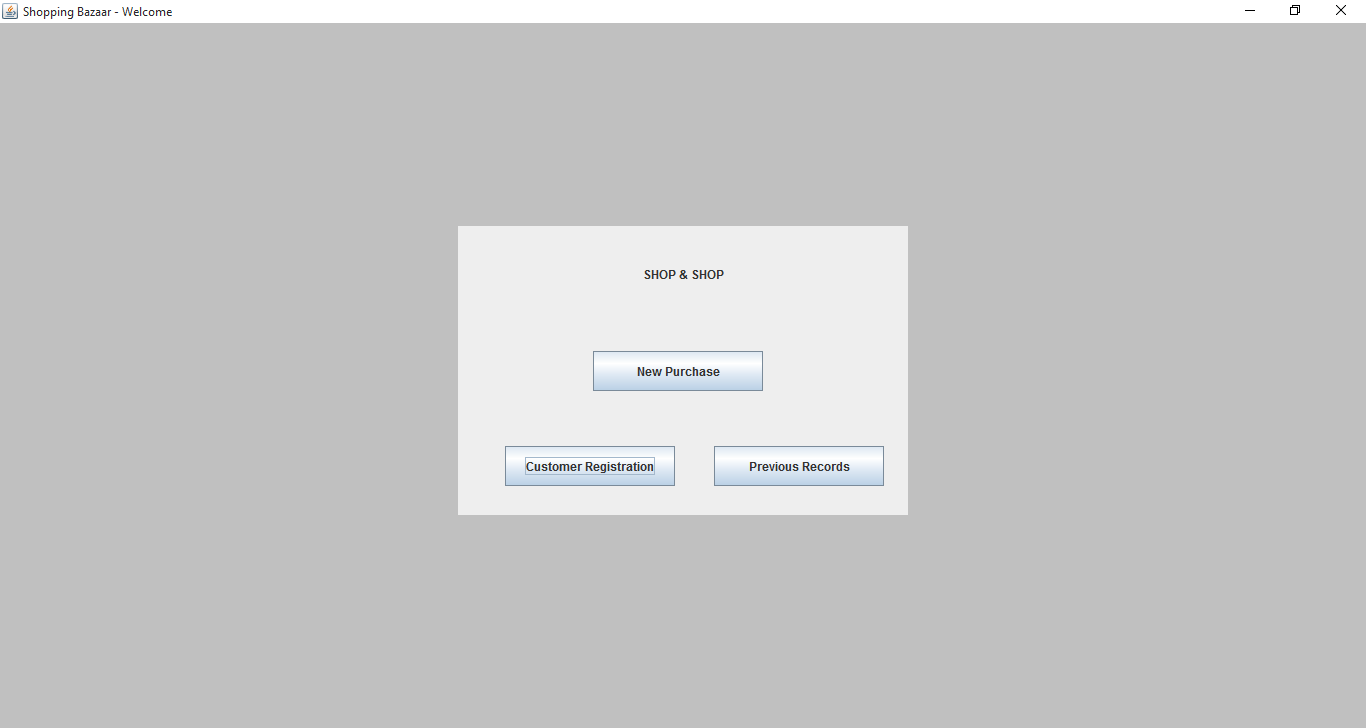


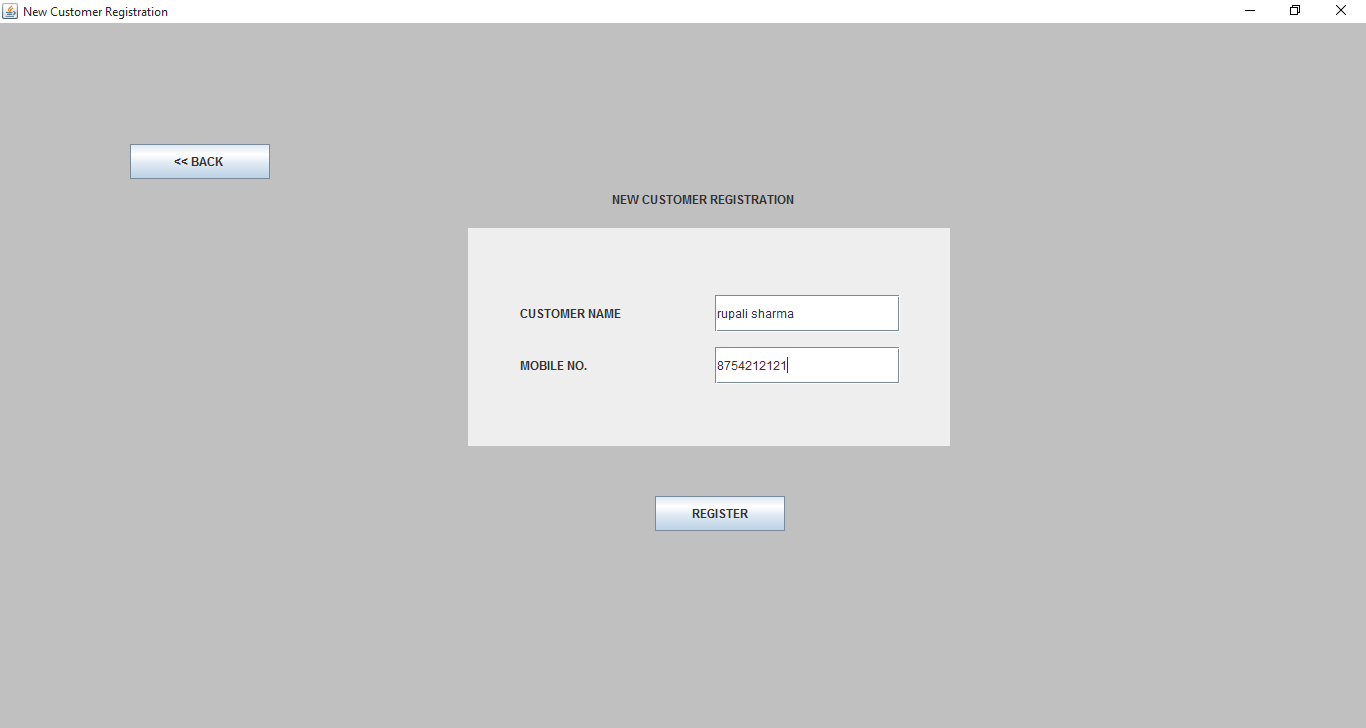
**3.1.2 Purchasing as guest Customer**

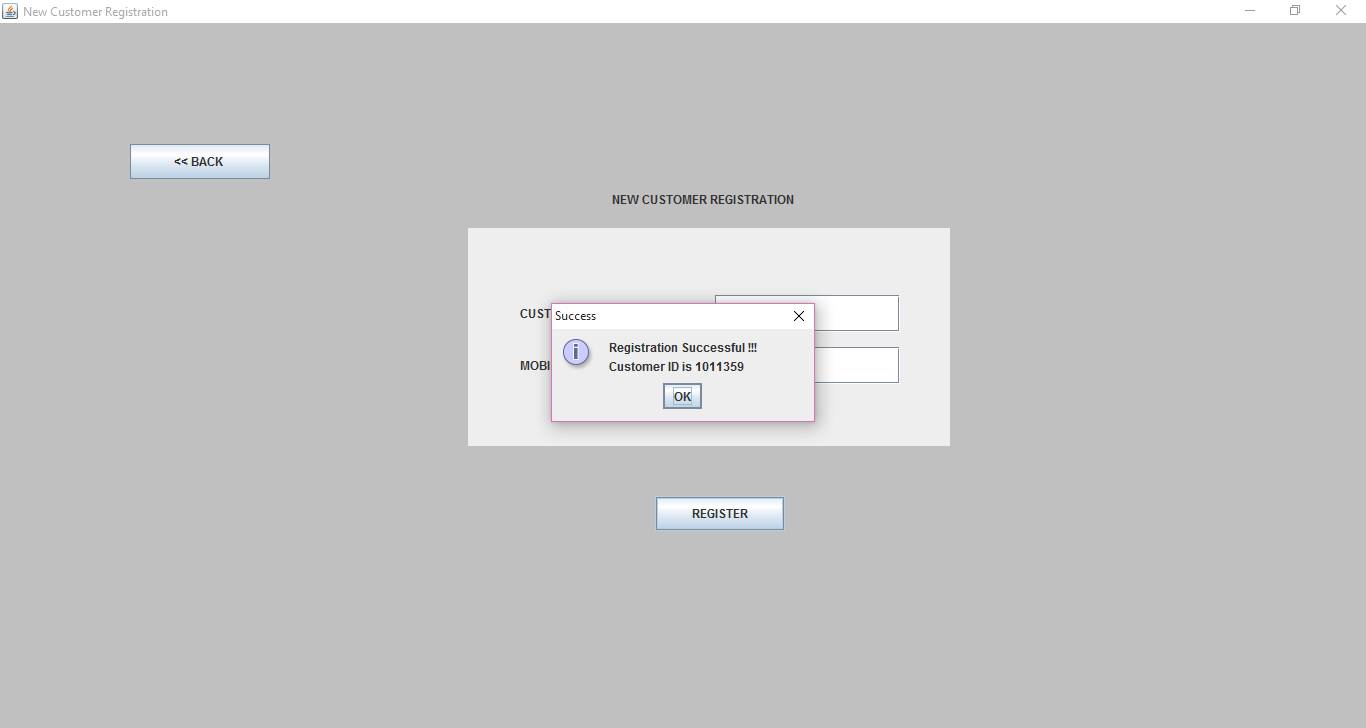




**3.2 New Customer Registration**

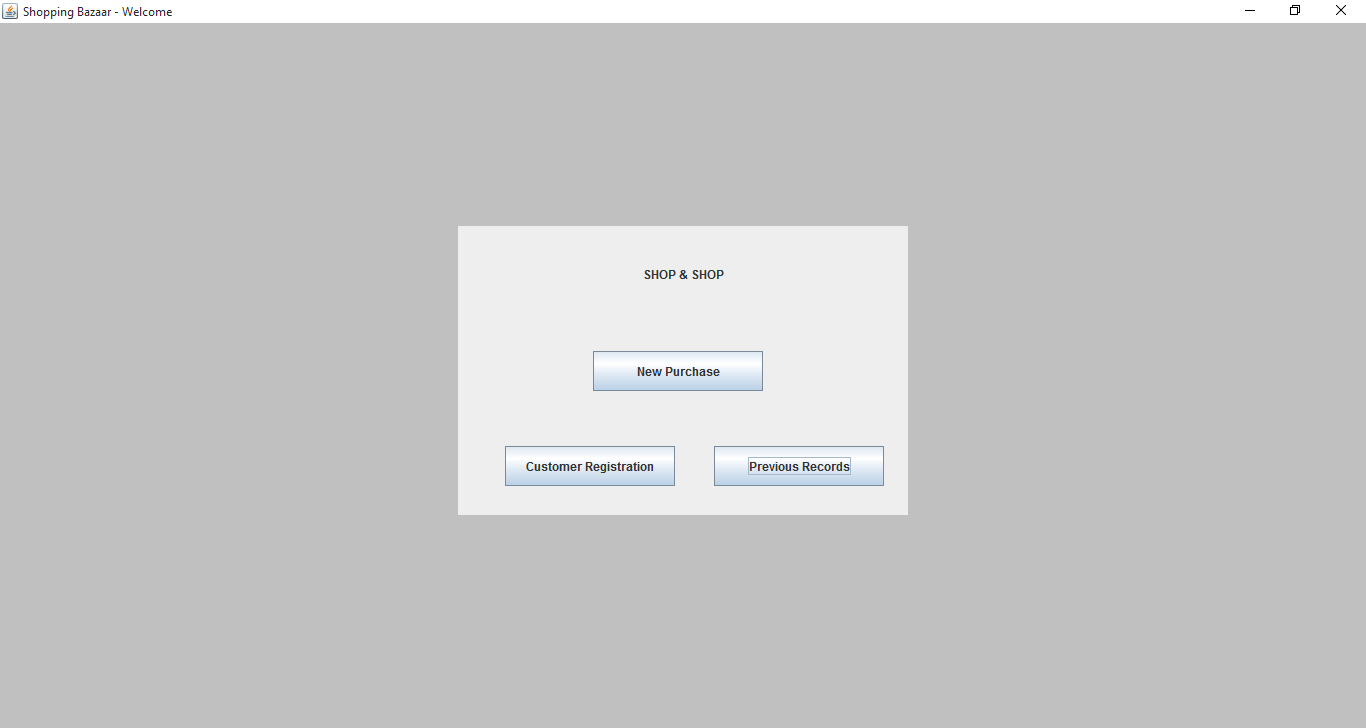


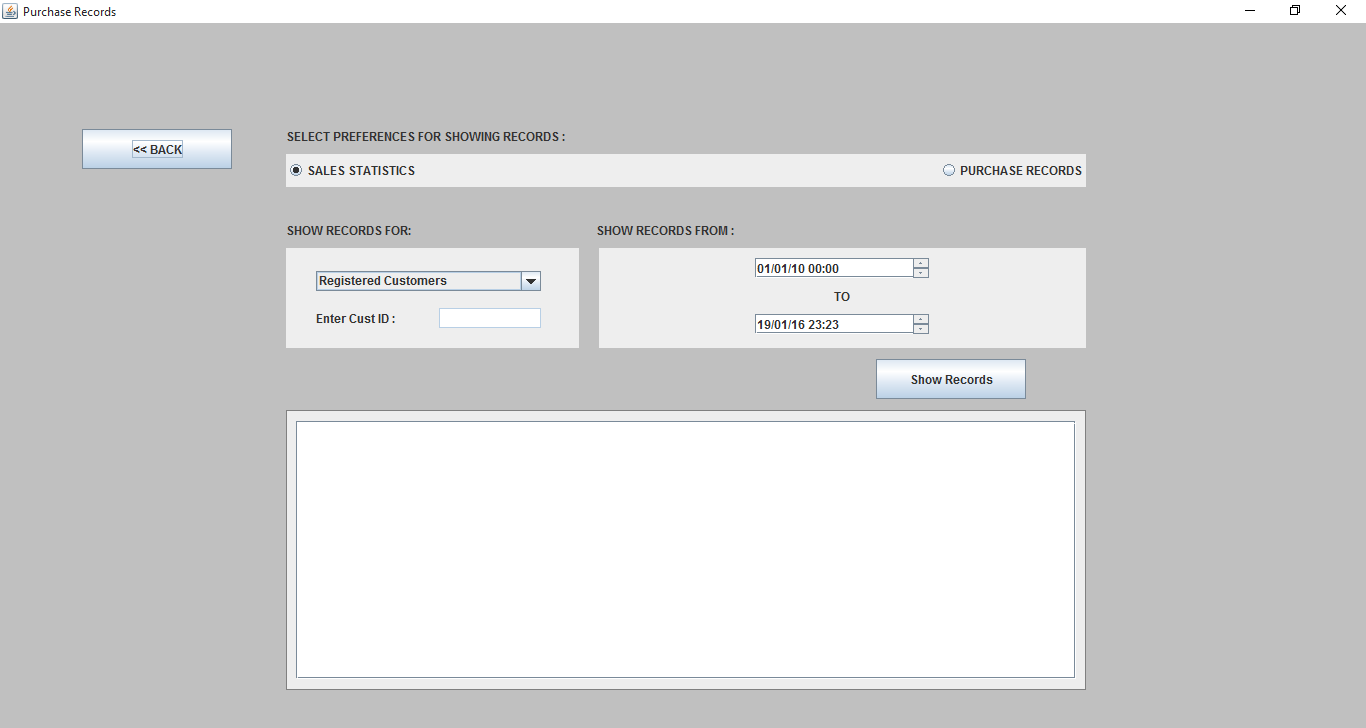


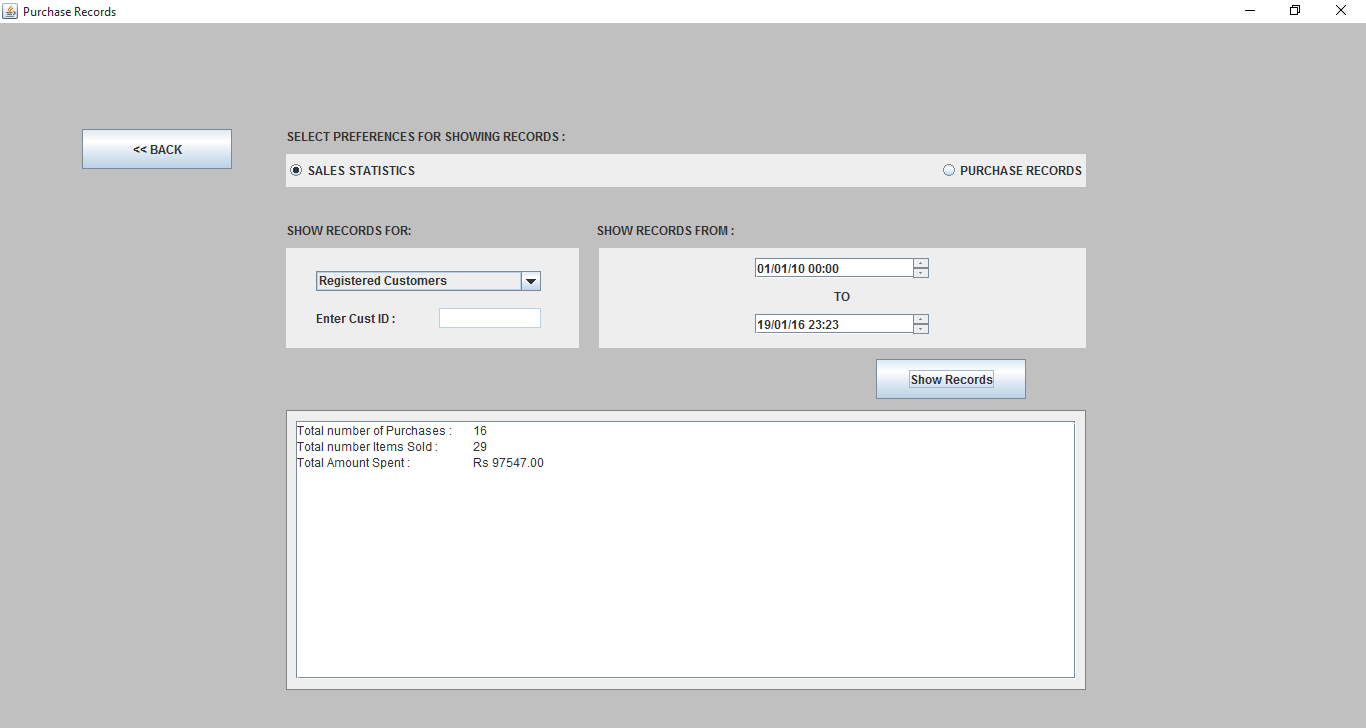


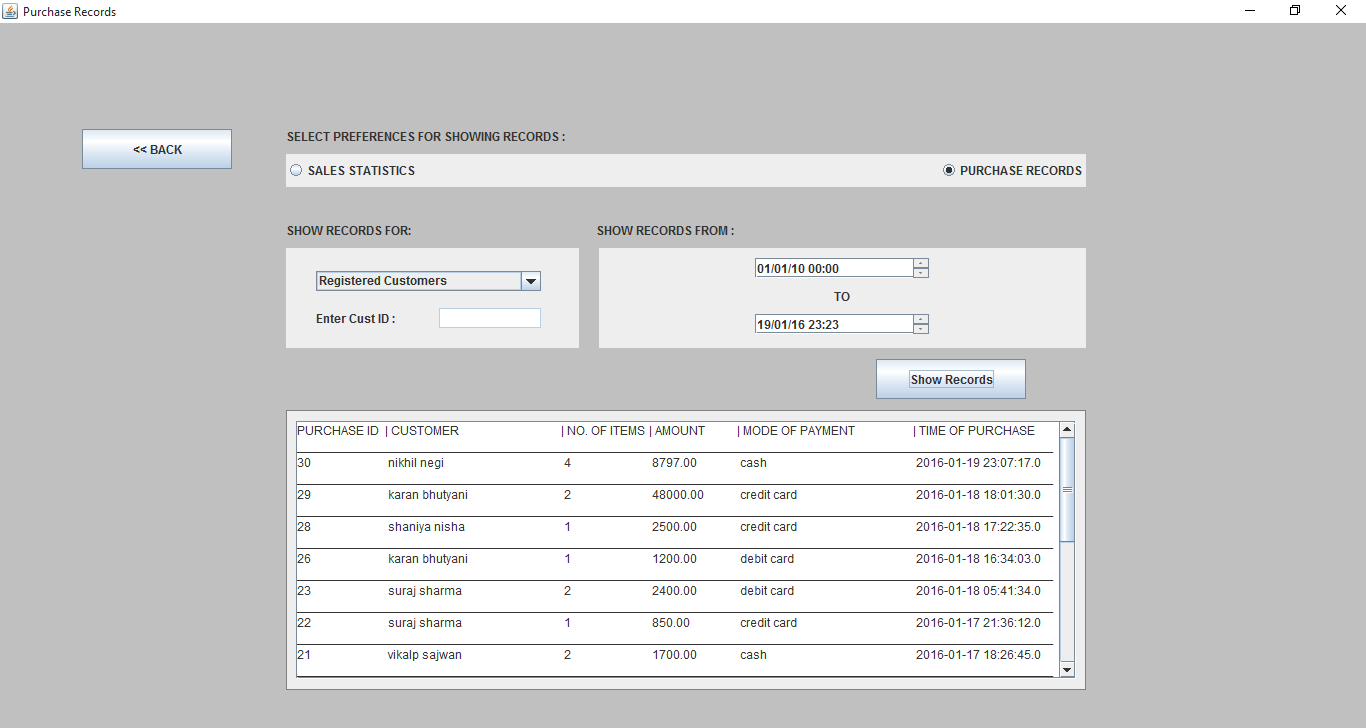
The customer id of customer is returned so that he can use this id in future purchases to log in.

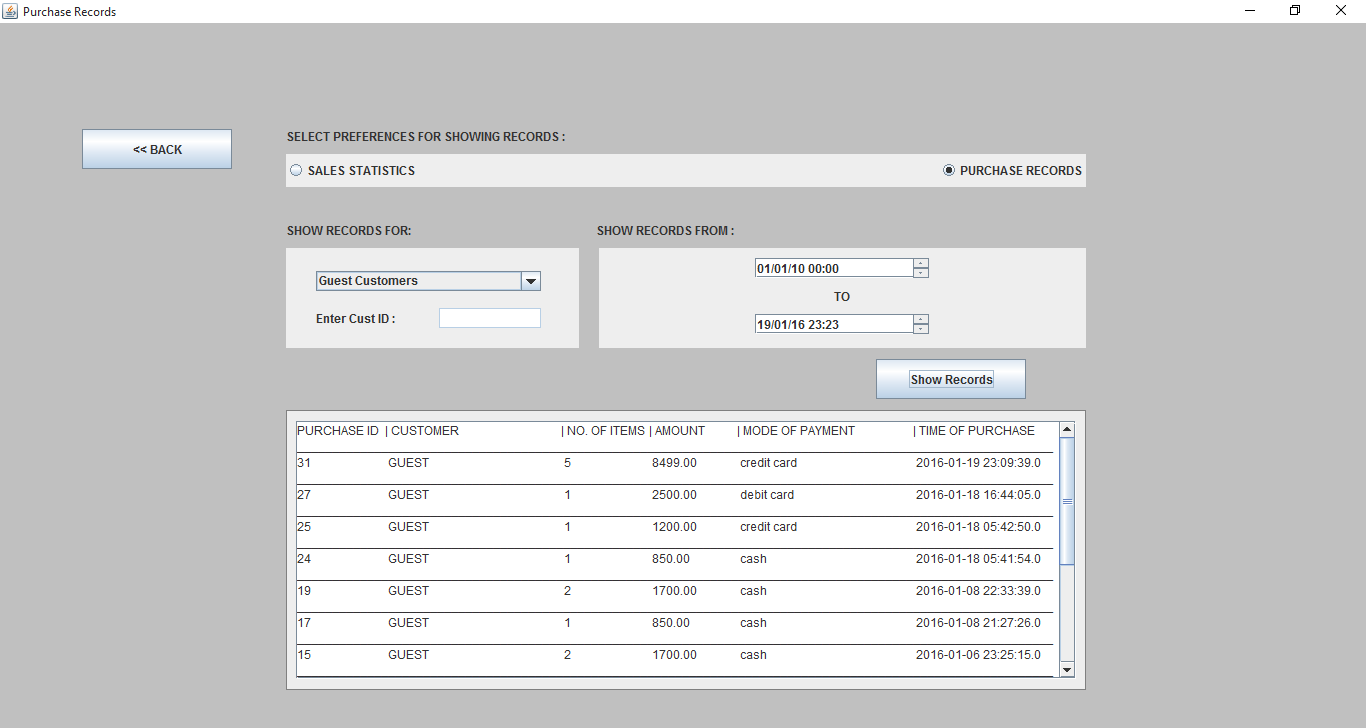
**3.3 Purchasing History Statistics and Records**

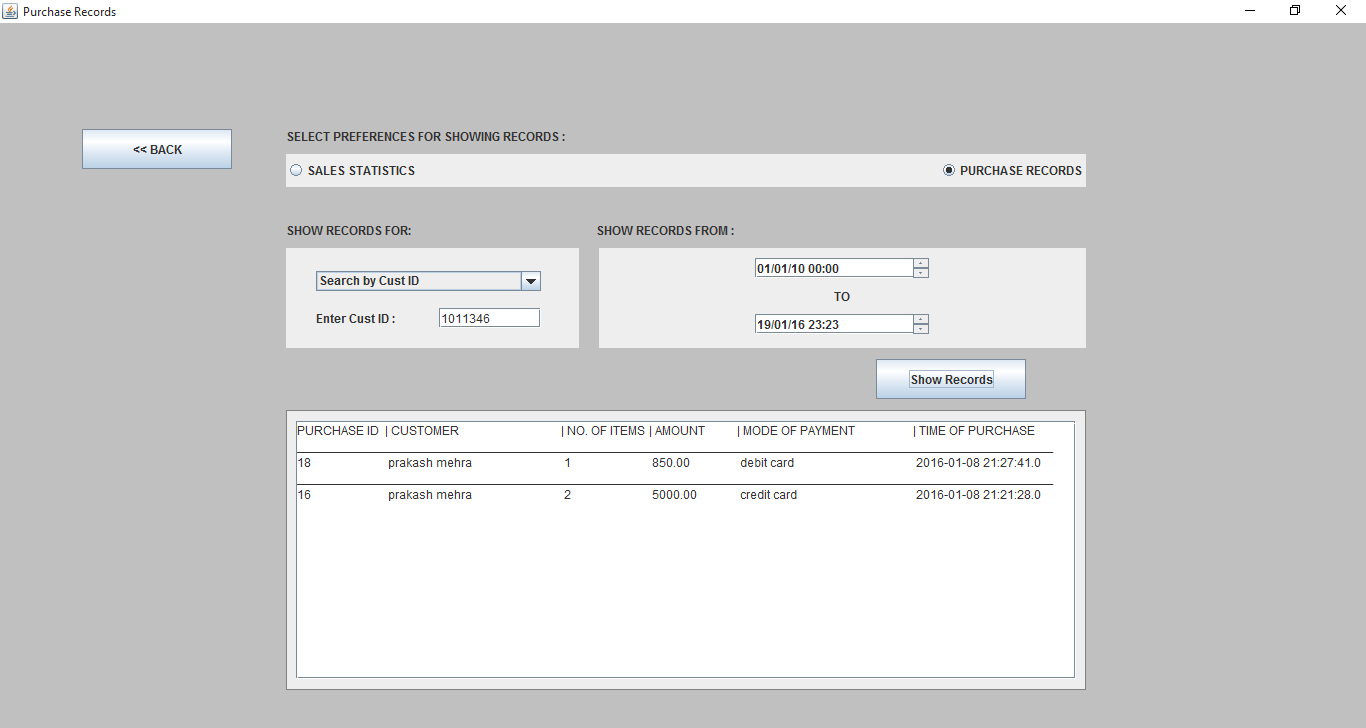












**CHAPTER 4**

**CONCLUSION**

The project ***“*Shop Management System*”*** has been created with the intention of providing the user with application which will suffice all needs for the items and other updates.

All the requirements specifications was followed as for as possible and few additional features were added that can make the application more user friendly and less complicated.

The project ***“*Shop Management System*”*** has been developed with the proper planning and keeping in mind the needs of such software.

This report provides with the fully fledged description of the software design capabilities and implementation details.

We hope that the ***“*Shop Management System*”*** fulfils all the needs in possible manner.

Limitations:

* Does not keep track of the quantity of items in stock.
* Does not keeps the record of which items were bought by which customer.
* This front end( Java SWINGS) is very difficult to upgrade and improve with the increase in the scale of business.
* End user should have the basic knowledge of the computers.
* No recovery for database if the current machine crashes.

Future enhancement:

The future enhancements may include new future such as

* Keeping track of number of items in stock.
* Module for adding new items information to the database.
* Multiple concurrent access system to increase scale of the business.
* Employees detail module.

**APPENDIX**

**Code**

**Class - Controller.java**

**package** shop\_management;

**import** java.sql.SQLException;

**public** **class** **Controller** {

**DBConnection** dBConnection = **new** DBConnection();

**Welcome** welcomeF = **new** Welcome(**this**);

**Shopping** shoppingF = **new** Shopping(**this**, dBConnection);

**PreviousRecords** prevRecordsF = **new** PreviousRecords(**this**, dBConnection);

**CustomerLogin** custLoginF = **new** CustomerLogin(**this**, dBConnection);

**Payment** paymentF = **new** Payment(**this**, dBConnection);

**CustomerRegistration** custRegF = **new** CustomerRegistration(**this**, dBConnection);

**Controller**(){

welcomeF.setVisible(**true**);

}

**void** **nextFrame**(**Welcome** welcomeF, **int** pageChoice){

welcomeF.setVisible(**false**);

**if**(pageChoice == **Globals**.***SHOPPING\_PAGE***){

shoppingF.setVisible(**true**);

shoppingF.setTables();

}

**else** **if**(pageChoice == **Globals**.***PREV\_RECORDS\_PAGE***){

prevRecordsF.reset();

prevRecordsF.setVisible(**true**);

}

**else** **if**(pageChoice == **Globals**.***CUST\_REG\_PAGE***){

custRegF.setVisible(**true**);

}

}

**void** **nextFrame**(**Shopping** shoppingF){

custLoginF.reset();

custLoginF.setVisible(**true**);

}

**void** **nextFrame**(**CustomerLogin** custLoginF, **boolean** loginType) **throws** **SQLException**, **ClassNotFoundException** {

shoppingF.setVisible(**false**);

custLoginF.setVisible(**false**);

paymentF.reset();

**if**(loginType == **Globals**.***CUST\_LOGIN***){

paymentF.setLoginType(**Globals**.***CUST\_LOGIN***);

}

**else** **if**(loginType == **Globals**.***GUEST\_LOGIN***){

paymentF.setLoginType(**Globals**.***GUEST\_LOGIN***);

}

paymentF.setVisible(**true**);

}

**void** **nextFrame**(**Payment** paymentF){

shoppingF.reset();

custLoginF.reset();

paymentF.reset();

prevRecordsF.reset();

**Globals**.*reset*();

paymentF.setVisible(**false**);

welcomeF.setVisible(**true**);

}

**void** **nextFrame**(**PreviousRecords** prevRecordsF){

prevRecordsF.setVisible(**false**);

welcomeF.setVisible(**true**);

}

**void** **nextFrame**(**CustomerRegistration** custRegF){

custRegF.setVisible(**false**);

welcomeF.setVisible(**true**);

}

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

**new** Controller();

}

}

**Class - DBConnection.java**

**package** shop\_management;

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

**import** java.text.SimpleDateFormat;

**public** **class** **DBConnection** {

Connection con = **null**;

Statement statement = **null**;

ResultSet resultSet = **null**;

**public** **void** **connect**() **throws** **ClassNotFoundException**, **SQLException** {

**Class**.*forName*("com.mysql.jdbc.Driver");

con = **DriverManager**.*getConnection*("jdbc:mysql://localhost:3306/shop\_management","root","root");

}

**public** ResultSet **getItems**(**int** tableCat) **throws** **SQLException**, **ClassNotFoundException**{

connect();

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT item\_id, item\_name, item\_details, item\_price FROM items"

+ " WHERE item\_category="+

tableCat);

**return** resultSet;

}

**public** ResultSet **custLogin**(**int** custID) **throws** **SQLException**, **ClassNotFoundException**{

connect();

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT \* FROM customers WHERE cust\_id = "+custID);

**return** resultSet;

}

**public** **void** **insertPurchaseRecord**(**int** paymentMethod) **throws** **SQLException**, **ClassNotFoundException** {

connect();

**String** **sql** = "INSERT INTO purchase (purchase\_id, cust\_id, items\_sold, total\_cost, payment\_type, time\_of\_purchase) " +

"Values (NULL" +

"," +

**Globals**.*CUSTOMER\_ID*+

"," +

**Globals**.*NUMBER\_OF\_ITEMS*+

"," +

**Globals**.*BILL\_AMOUNT*+

"," +

paymentMethod+

",CURRENT\_TIMESTAMP)";

statement = con.createStatement();

statement.executeUpdate(sql);

}

**public** **Long** **insertNewCustomer**(**String** name, **Long** contact) **throws** **SQLException**, **ClassNotFoundException** {

connect();

**String** **sql** = "INSERT INTO customers (cust\_id, cust\_name, cust\_mobile) " +

"Values (NULL" +

",'" +

name +

"'," +

contact +

")";

**String** **sqlForId** = "SELECT LAST\_INSERT\_ID()";

statement = con.createStatement();

statement.executeUpdate(sql);

resultSet = statement.executeQuery(sqlForId);

resultSet.next();

**long** **newCustId** =resultSet.getLong(1);

**return** newCustId;

}

**public** **void** **updatePoints**(**int** points) **throws** **SQLException**, **ClassNotFoundException** {

connect();

**String** **sql** = "Update customers " +

"SET cust\_points = "+

points +

" WHERE cust\_id = "+

**Globals**.*CUSTOMER\_ID*

;

statement = con.createStatement();

statement.executeUpdate(sql);

}

**public** **String**[] **getStats**(**int** custType)**throws** **SQLException**, **ClassNotFoundException**{

connect();

**SimpleDateFormat** **sdf** = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

**String** **timePeriod** = " AND time\_of\_purchase >= '"

+ sdf.format(**Globals**.*START\_DATE*)

+"' AND time\_of\_purchase <= '"

+ sdf.format(**Globals**.*END\_DATE*)

+"'";

**String** **timePeriodForAll** = " WHERE time\_of\_purchase >= '"

+ sdf.format(**Globals**.*START\_DATE*)

+"' AND time\_of\_purchase <= '"

+ sdf.format(**Globals**.*END\_DATE*)

+"'";

**String**[] **data** = **new** **String**[10];

**if**(custType == **Globals**.***REGISTERED\_CUST***){

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT count(\*) FROM purchase WHERE cust\_id != 0" + timePeriod);

resultSet.next();

data[3] = "Total number of Purchases : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(items\_sold) FROM purchase WHERE cust\_id != 0" + timePeriod);

resultSet.next();

data[4] = "Total number Items Sold : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(total\_cost) FROM purchase WHERE cust\_id != 0" + timePeriod);

resultSet.next();

data[5] = "Total Amount Spent : \tRs "+ resultSet.getString(1);

}

**else** **if** (custType == **Globals**.***GUEST\_CUST***){

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT count(\*) FROM purchase WHERE cust\_id = 0" + timePeriod);

resultSet.next();

data[3] = "Total number of Purchases : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(items\_sold) FROM purchase WHERE cust\_id = 0" + timePeriod);

resultSet.next();

data[4] = "Total number Items Sold : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(total\_cost) FROM purchase WHERE cust\_id = 0" + timePeriod);

resultSet.next();

data[5] = "Total Amount Spent : \tRs "+ resultSet.getString(1);

}

**else** **if** (custType == **Globals**.***ALL\_CUST***){

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT count(\*) FROM purchase" + timePeriodForAll);

resultSet.next();

data[3] = "Total number of Purchases : \t\t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(items\_sold) FROM purchase" + timePeriodForAll);

resultSet.next();

data[4] = "Total number Items Sold : \t\t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT count(\*) FROM purchase WHERE cust\_id = 0" + timePeriod);

resultSet.next();

data[5] = "Purchases done by Guest Customers : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT count(\*) FROM purchase WHERE cust\_id != 0" + timePeriod);

resultSet.next();

data[6] = "Purchases done by Registered Customers : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(total\_cost) FROM purchase" + timePeriodForAll);

resultSet.next();

data[7] = "Total Amount Spent : \t\tRs "+ resultSet.getString(1);

}

**else** {

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT cust\_name, cust\_mobile, cust\_points FROM customers"

+ " WHERE cust\_id = "+

custType);

resultSet.next();

data[0] = "Customer Name : \t" + resultSet.getString(1);

data[1] = "Customer Contact : \t" + resultSet.getString(2);

data[2] = "Loyalty Points : \t" + resultSet.getString(3);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT count(\*) FROM purchase WHERE cust\_id = "+

custType + timePeriod );

resultSet.next();

data[3] = "Total number of Purchases : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(items\_sold) FROM purchase WHERE cust\_id = "+

custType + timePeriod );

resultSet.next();

data[4] = "Total number Items Sold : \t"+ resultSet.getString(1);

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT sum(total\_cost) FROM purchase WHERE cust\_id = "+

custType + timePeriod );

resultSet.next();

data[5] = "Total Amount Spent : \tRs "+ resultSet.getString(1);

}

**return** data;

}

**public** ResultSet **getRecords**(**int** custType) **throws** **SQLException**, **ClassNotFoundException**{

connect();

**SimpleDateFormat** **sdf** = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

**String** **timePeriod** = " AND time\_of\_purchase >= '"

+ sdf.format(**Globals**.*START\_DATE*)

+"' AND time\_of\_purchase <= '"

+ sdf.format(**Globals**.*END\_DATE*)

+"' ORDER BY time\_of\_purchase DESC";

ResultSet **resultSet**;

**String** **query** = **new** String();

**if**(custType == **Globals**.***REGISTERED\_CUST***)

query = " AND purchase.cust\_id != 0";

**else** **if** (custType == **Globals**.***GUEST\_CUST***)

query = " AND purchase.cust\_id = 0";

**else** **if** (custType == **Globals**.***ALL\_CUST***)

query = "";

**else**

query = " AND purchase.cust\_id =" + custType;

statement = con.createStatement();

resultSet = statement.executeQuery("SELECT purchase\_id, cust\_name, items\_sold, total\_cost, payment\_type\_name, time\_of\_purchase"

+ " FROM customers, purchase, payment\_type"

+ " WHERE customers.cust\_id = purchase.cust\_id"

+ " AND purchase.payment\_type = payment\_type.payment\_type\_id"

+ query + timePeriod);

**return** resultSet;

}

**public** **void** **disconnect**() {

**try** {

**if** (resultSet != **null**) {

resultSet.close();

}

**if** (statement != **null**) {

statement.close();

}

**if** (con != **null**) {

con.close();

}

} **catch** (**SQLException** **e**) {

**System**.***out***.println("SUPER EXCEPTION");

e.printStackTrace();

}

}

}

**Class - Globals.java**

**package** shop\_management;

**import** java.util.Date;

**public** **class** **Globals** {

**public** **static** **final** **boolean** ***CUST\_LOGIN*** = **true**;

**public** **static** **final** **boolean** ***GUEST\_LOGIN*** = **false**;

**public** **static** **final** **int** ***SHOPPING\_PAGE*** = 1;

**public** **static** **final** **int** ***PREV\_RECORDS\_PAGE*** = 2;

**public** **static** **final** **int** ***CUST\_REG\_PAGE*** = 3;

**public** **static** **final** **int** ***CLOTHING*** = 1;

**public** **static** **final** **int** ***ELECTRONICS*** = 2;

**public** **static** **final** **int** ***HEALTH\_AND\_FITNESS*** = 3;

**public** **static** **final** **int** ***HOME\_AND\_DECOR*** = 4;

**public** **static** **final** **int** ***REGISTERED\_CUST*** = 0;

**public** **static** **final** **int** ***GUEST\_CUST*** = 1;

**public** **static** **final** **int** ***ALL\_CUST*** = 2;

**public** **static** **final** **int** ***SINGLE\_CUST*** = 3;

**public** **static** **final** **int** ***CREDIT*** = 1;

**public** **static** **final** **int** ***DEBIT*** = 2;

**public** **static** **final** **int** ***CASH*** = 3;

**public** **static** **int** *CUSTOMER\_ID*;

**public** **static** **float** *BILL\_AMOUNT*;

**public** **static** **int** *POINTS*;

**public** **static** **int** *NUMBER\_OF\_ITEMS*;

**public** **static** **Date** *START\_DATE*;

**public** **static** **Date** *END\_DATE*;

**public** **static** **void** **reset**(){

*CUSTOMER\_ID* = 0;

*BILL\_AMOUNT* = (**float**) 0.0;

*POINTS* = 0;

*NUMBER\_OF\_ITEMS* = 0;

}

}

**Class - Welcome.java**

**package** shop\_management;

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

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

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

***@SuppressWarnings***("serial")

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

**JPanel** mainP, subP;

**JButton** newPurchaseB, prevRecordsB, custRegB ;

**JLabel** shopNameL;

**Controller** controller;

**private** **JPanel** sub2P;

**public** **Welcome**(**Controller** controller) {

**this**.controller = controller;

getContentPane().setBackground(**Color**.***LIGHT\_GRAY***);

setResizable(**true**);

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

setTitle("Shopping Bazaar - Welcome");

**Toolkit** **tk** = **Toolkit**.*getDefaultToolkit*();

**Dimension** **dim** = tk.getScreenSize();

setSize(dim.width, dim.height);

setBackground(**Color**.***LIGHT\_GRAY***);

setExtendedState(***MAXIMIZED\_BOTH*** );

getContentPane().setLayout(**null**);

mainP = **new** JPanel();

mainP.setSize(450, 289);

mainP.setLocation(458, 203);

mainP.setLayout(**new** GridLayout(3, 1, 0, 0));

getContentPane().add(mainP);

shopNameL = **new** JLabel("SHOP & SHOP");

shopNameL.setHorizontalAlignment(SwingConstants.***CENTER***);

mainP.add(shopNameL);

sub2P = **new** JPanel();

mainP.add(sub2P);

sub2P.setLayout(**null**);

newPurchaseB = **new** JButton("New Purchase");

newPurchaseB.setBounds(135, 29, 170, 40);

sub2P.add(newPurchaseB);

newPurchaseB.addActionListener(**this**);

subP = **new** JPanel();

subP.setLayout(**null**);

mainP.add(subP);

prevRecordsB = **new** JButton("Previous Records");

prevRecordsB.setBounds(256, 28, 170, 40);

prevRecordsB.addActionListener(**this**);

subP.add(prevRecordsB);

custRegB = **new** JButton("Customer Registration");

custRegB.setBounds(47, 28, 170, 40);

custRegB.addActionListener(**this**);

subP.add(custRegB);

}

***@Override***

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

**if**(e.getSource() == newPurchaseB)

controller.nextFrame(**this**, **Globals**.***SHOPPING\_PAGE***);

**else** **if**(e.getSource() == prevRecordsB)

controller.nextFrame(**this**, **Globals**.***PREV\_RECORDS\_PAGE***);

**else** **if**(e.getSource() == custRegB)

controller.nextFrame(**this**, **Globals**.***CUST\_REG\_PAGE***);

}

}

**Class - Shopping.java**

**package** shop\_management;

**import** java.awt.Color;

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

**import** java.util.Arrays;

**import** java.util.Vector;

**import** java.awt.Dimension;

**import** java.awt.Toolkit;

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

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

**import** java.awt.event.ActionListener;

**import** java.awt.event.ActionEvent;

***@SuppressWarnings***("serial")

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

**DBConnection** dBConnection;

**private** **JTable** clothingTable = **new** JTable();

**private** **JTable** electronicsTable = **new** JTable();

**private** **JTable** healthTable = **new** JTable();

**private** **JTable** decorTable = **new** JTable();

**private** **JTable** shopListTable = **new** JTable();

**JTabbedPane** catagoryTP;

**JButton** finishPurchaseB, addItemB, removeItemB;

**JTextField** totalTF;

**Controller** controller;

**JPanel** shopListP ;

**JLabel** totalL, shopCartL;

ResultSet resultSet;

**public** **Shopping**(**Controller** controller, **DBConnection** dBConnection) {

**this**.dBConnection = dBConnection;

**this**.controller = controller;

getContentPane().setBackground(**Color**.***LIGHT\_GRAY***);

setResizable(**true**);

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

setTitle("Shopping Page");

**Toolkit** **tk** = **Toolkit**.*getDefaultToolkit*();

**Dimension** **dim** = tk.getScreenSize();

setSize(dim.width, dim.height);

setBackground(**Color**.***LIGHT\_GRAY***);

setExtendedState(***MAXIMIZED\_BOTH*** );

getContentPane().setLayout(**null**);

catagoryTP = **new** JTabbedPane(**JTabbedPane**.***TOP***);

catagoryTP.setBounds(80, 79, 600, 543);

catagoryTP.addTab("CLOTHING", **null**, **new** JScrollPane(clothingTable), **null**);

catagoryTP.addTab("ELECTRONICS", **null**, **new** JScrollPane(electronicsTable), **null**);

catagoryTP.addTab("HEALTH AND FITNESS", **null**, **new** JScrollPane(healthTable), **null**);

catagoryTP.addTab("HOME AND DECOR", **null**, **new** JScrollPane(decorTable), **null**);

getContentPane().add(catagoryTP);

shopListP = **new** JPanel();

shopListP.setBounds(780, 100, 482, 439);

shopListP.add(**new** JScrollPane(shopListTable));

getContentPane().add(shopListP);

addItemB = **new** JButton("Add Items");

addItemB.addActionListener(**this**);

addItemB.setBounds(550, 633, 130, 35);

getContentPane().add(addItemB);

removeItemB = **new** JButton("Remove Item");

removeItemB.setBounds(1132, 550, 130, 35);

removeItemB.addActionListener(**this**);

getContentPane().add(removeItemB);

totalL = **new** JLabel("TOTAL Rs.");

totalL.setBounds(780, 633, 130, 35);

getContentPane().add(totalL);

totalTF = **new** JTextField("0.00",10);

totalTF.setEditable(**false**);

totalTF.setBounds(859, 633, 130, 35);

getContentPane().add(totalTF);

finishPurchaseB = **new** JButton("Finish Purchase");

finishPurchaseB.setBounds(1132, 633, 130, 35);

finishPurchaseB.addActionListener(**this**);

getContentPane().add(finishPurchaseB);

shopCartL = **new** JLabel("SHOPPING CART",SwingConstants.***CENTER***);

shopCartL.setBounds(762, 75, 130, 35);

getContentPane().add(shopCartL);

}

**void** **setTables**(){

**try** {

resultSet = dBConnection.getItems(**Globals**.***CLOTHING***);

**DefaultTableModel** **clothingModel** = (**DefaultTableModel**)buildTableModel(resultSet);

clothingTable.setModel(clothingModel);

**TableColumn** **idCol** = clothingTable.getColumnModel().getColumn(0);

idCol.setMaxWidth(60);

idCol = clothingTable.getColumnModel().getColumn(3);

idCol.setMaxWidth(120);

resultSet = dBConnection.getItems(**Globals**.***ELECTRONICS***);

**DefaultTableModel** **electronicsModel** = (**DefaultTableModel**)buildTableModel(resultSet);

electronicsTable.setModel(electronicsModel);

idCol = electronicsTable.getColumnModel().getColumn(0);

idCol.setMaxWidth(60);

idCol = electronicsTable.getColumnModel().getColumn(3);

idCol.setMaxWidth(120);

resultSet = dBConnection.getItems(**Globals**.***HEALTH\_AND\_FITNESS***);

**DefaultTableModel** **healthModel** = (**DefaultTableModel**)buildTableModel(resultSet);

healthTable.setModel(healthModel);

idCol = healthTable.getColumnModel().getColumn(0);

idCol.setMaxWidth(60);

idCol = healthTable.getColumnModel().getColumn(3);

idCol.setMaxWidth(120);

resultSet = dBConnection.getItems(**Globals**.***HOME\_AND\_DECOR***);

**DefaultTableModel** **decorModel** = (**DefaultTableModel**)buildTableModel(resultSet);

decorTable.setModel(decorModel);

idCol = decorTable.getColumnModel().getColumn(0);

idCol.setMaxWidth(60);

idCol = decorTable.getColumnModel().getColumn(3);

idCol.setMaxWidth(120);

**DefaultTableModel** **shopListModel** = (**DefaultTableModel**)*buildTableModel*();

shopListTable.setModel(shopListModel);

idCol = shopListTable.getColumnModel().getColumn(0);

idCol.setMaxWidth(60);

idCol = shopListTable.getColumnModel().getColumn(3);

idCol.setMaxWidth(120);

} **catch** (**SQLException** **e**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

} **catch** (**ClassNotFoundException** **e1**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database Driver !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**finally**{

dBConnection.disconnect();

}

}

**void** **reset**(){

totalTF.setText("0.00");

catagoryTP.setSelectedIndex(0);

**DefaultTableModel** **shopListModel** = (**DefaultTableModel**)shopListTable.getModel();

shopListModel.setRowCount(0);

}

**public** **void** **moveItemsToCart**(**JTable** table){

**int** **selectedItems**[] = table.getSelectedRows();

**DefaultTableModel** **itemsModel** = (**DefaultTableModel**)table.getModel();

**DefaultTableModel** **shopListModel** = (**DefaultTableModel**)shopListTable.getModel();

**for** (**int** **i** : selectedItems) {

**Vector**<Object> **rowData** = **new** Vector<Object>();

**for**(**int** **j**=0 ;j <=3; j++){

rowData.add(itemsModel.getValueAt(i, j));

}

**float** **prevTotal** = **Float**.*parseFloat*(totalTF.getText());

**String** **itemPrice** = itemsModel.getValueAt(i, 3).toString();

**float** **newTotal** = prevTotal + **Float**.*parseFloat*(itemPrice);

totalTF.setText(""+newTotal);

shopListModel.addRow(rowData);

}

}

**public** **void** **removeItemsFromCart**(){

**DefaultTableModel** **shopListModel** = (**DefaultTableModel**)shopListTable.getModel();

**int** []**selectedItems** = shopListTable.getSelectedRows();

**Arrays**.*sort*(selectedItems);

**for** (**int** **left** = 0, **right** = selectedItems.length - 1; left < right; left++, right--) {

**int** **temp** = selectedItems[left];

selectedItems[left] = selectedItems[right];

selectedItems[right] = temp;

}

**for** (**int** **i** : selectedItems) {

**float** **prevTotal** = **Float**.*parseFloat*(totalTF.getText());

**String** **itemPrice** = shopListModel.getValueAt(i, 3).toString();

**float** **newTotal** = prevTotal - **Float**.*parseFloat*(itemPrice);

totalTF.setText(""+newTotal);

shopListModel.removeRow(i);

}

}

**public** **DefaultTableModel** **buildTableModel**(ResultSet resultSet)

**throws** **SQLException** {

ResultSetMetaData **metaData** = resultSet.getMetaData();

**Vector**<String> **columnNames** = **new** Vector<String>();

**int** **columnCount** = metaData.getColumnCount();

columnNames.add("Item ID");

columnNames.add("Item Name");

columnNames.add("Item Details");

columnNames.add("Item Price");

**Vector**<**Vector**<Object>> **data** = **new** Vector<**Vector**<Object>>();

**while** (resultSet.next()) {

**Vector**<Object> **vector** = **new** Vector<Object>();

**for** (**int** **columnIndex** = 1; columnIndex <= columnCount; columnIndex++) {

vector.add(resultSet.getObject(columnIndex));

}

data.add(vector);

}

**return** **new** DefaultTableModel(data, columnNames){

***@Override***

**public** **boolean** **isCellEditable**(**int** row, **int** column){

**return** **false**;

}

};

}

**public** **static** **DefaultTableModel** **buildTableModel**()

**throws** **SQLException** {

**Vector**<String> **columnNames** = **new** Vector<String>();

columnNames.add("Item ID");

columnNames.add("Item Name");

columnNames.add("Item Details");

columnNames.add("Item Price");

**return** **new** DefaultTableModel(columnNames, 0){

***@Override***

**public** **boolean** **isCellEditable**(**int** row, **int** column){

**return** **false**;

}

};

}

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

**if**(e.getSource() == finishPurchaseB){

**Globals**.*NUMBER\_OF\_ITEMS* = shopListTable.getRowCount();

**Globals**.*BILL\_AMOUNT* = **Float**.*parseFloat*(totalTF.getText());

**if**(**Float**.*parseFloat*(totalTF.getText()) == 0.0)

**JOptionPane**.*showMessageDialog*(**this**, "Please add atleast one item to cart !!!", "ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

**else**

controller.nextFrame(**this**);

}

**else** **if**(e.getSource() == addItemB){

**int** **selectedTab** = catagoryTP.getSelectedIndex();

**if**(selectedTab == 0)

moveItemsToCart(clothingTable);

**else** **if**(selectedTab == 1)

moveItemsToCart(electronicsTable);

**else** **if**(selectedTab == 2)

moveItemsToCart(healthTable);

**else**

moveItemsToCart(decorTable);

}

**else** **if**(e.getSource() == removeItemB){

removeItemsFromCart();

}

}

}

**Class - CustomerLogin.java**

**package** shop\_management;

**import** java.awt.FlowLayout;

**import** java.awt.GridLayout;

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

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

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.Color;

***@SuppressWarnings***("serial")

**public** **class** **CustomerLogin** **extends** **JDialog** **implements** ActionListener{

**private** **JTextField** custIdTF;

**JButton** guestCustB, okB;

**Controller** controller;

**DBConnection** dBConnection;

**JLabel** warningL ;

ResultSet resultSet;

**public** **CustomerLogin**(**Controller** controller, **DBConnection** dBConnection) {

**this**.controller = controller;

**this**.dBConnection = dBConnection;

setBounds(500, 200, 450, 250);

setResizable(**false**);

setTitle("Customer Login");

setDefaultCloseOperation(**JDialog**.***DISPOSE\_ON\_CLOSE***);

setAlwaysOnTop(**true**);

setModalityType(*ModalityType*.***APPLICATION\_MODAL***);

getContentPane().setLayout(**new** GridLayout(3, 1));

**JPanel** **custIDP** = **new** JPanel(**new** FlowLayout(**FlowLayout**.***CENTER***, 15, 45));

getContentPane().add(custIDP);

**JLabel** **custIdL** = **new** JLabel("Enter Customer ID");

custIDP.add(custIdL);

custIdTF = **new** JTextField(18);

custIDP.add(custIdTF);

custIdTF.addActionListener(**new** ActionListener(){

***@Override***

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

okB.doClick();

}

});

okB = **new** JButton("OK");

okB.addActionListener(**this**);

custIDP.add(okB);

warningL = **new** JLabel("", SwingConstants.***CENTER***);

warningL.setForeground(**Color**.***RED***);

getContentPane().add(warningL);

**JPanel** **buttonP** = **new** JPanel(**new** FlowLayout());

getContentPane().add(buttonP);

guestCustB = **new** JButton("Guest Customer");

guestCustB.addActionListener(**this**);

guestCustB.setSize(130, 35);

buttonP.add(guestCustB);

}

**void** **reset**(){

custIdTF.setText("");

warningL.setText("");

}

***@Override***

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

**if**(e.getSource() == okB){

**try** {

**int** **id** = **Integer**.*parseInt*(custIdTF.getText());

**try** {

resultSet = dBConnection.custLogin(id);

**if**(resultSet.isBeforeFirst()){

**Globals**.*CUSTOMER\_ID* = id;

controller.nextFrame(**this**, **Globals**.***CUST\_LOGIN***);

}

**else**{

warningL.setText("NOT A VALID CUSTOMER ID !!!");

}

} **catch** (**SQLException** **e1**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

} **catch** (**ClassNotFoundException** **e2**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database Driver !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**finally**{

dBConnection.disconnect();

}

dBConnection.disconnect();

}

**catch**(**NumberFormatException** **e3**){

warningL.setText("ENTER NUMERIC CUSTOMER ID !!!");

}

}

**else** **if**(e.getSource() == guestCustB){

**try** {

controller.nextFrame(**this**, **Globals**.***GUEST\_LOGIN***);

} **catch** (**SQLException** **e4**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

} **catch** (**ClassNotFoundException** **e5**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database Driver !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**finally**{

dBConnection.disconnect();

}

}

}

}

**Class - Payment.java**

**package** shop\_management;

**import** java.awt.Dimension;

**import** java.awt.Toolkit;

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

**import** javax.swing.event.DocumentEvent;

**import** javax.swing.event.DocumentListener;

**import** java.awt.GridLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.Color;

**import** java.awt.FlowLayout;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

***@SuppressWarnings***("serial")

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

**DBConnection** dBConnection;

**private** **JTextField** custNameTF;

**private** **JTextField** custIdTF;

**private** **JTextField** custContactTF;

**private** **JTextField** pointsTF;

**private** **JTextField** totalBillTF;

**private** **JTextField** pointsUsedTF;

**private** **JTextField** payAmountTF;

**JRadioButton** payCashRB, payDebitRB, payCreditRB;

**JLabel** pointsUsedL;

**JLabel** totalBillL;

**JLabel** warningL;

**Controller** controller;

**ButtonGroup** payMethodBG;

ResultSet resultSet;

**JPanel** custInfoP, guestInfoP, billP, custInfoP1, custInfoP2, payMethodP;

**JButton** payB;

**public** **Payment**(**Controller** controller, **DBConnection** dBConnection) {

**this**.controller = controller;

**this**.dBConnection = dBConnection;

getContentPane().setBackground(**Color**.***LIGHT\_GRAY***);

setResizable(**true**);

setTitle("Payment");

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

**Toolkit** **tk** = **Toolkit**.*getDefaultToolkit*();

**Dimension** **dim** = tk.getScreenSize();

setSize(dim.width, dim.height);

setExtendedState(***MAXIMIZED\_BOTH*** );

getContentPane().setLayout(**null**);

custInfoP = **new** JPanel();

custInfoP.setBounds(270, 67, 810, 87);

custInfoP.setLayout(**new** GridLayout(0, 2, 0, 0));

guestInfoP = **new** JPanel();

**FlowLayout** **flowLayout** = (**FlowLayout**) guestInfoP.getLayout();

flowLayout.setVgap(35);

guestInfoP.setBounds(270, 67, 810, 87);

getContentPane().add(guestInfoP);

**JLabel** **lblNewLabel** = **new** JLabel("GUEST CUSTOMER");

guestInfoP.add(lblNewLabel);

custInfoP1 = **new** JPanel();

custInfoP.add(custInfoP1);

custInfoP1.setLayout(**new** GridLayout(2, 2, 0, 10));

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

custNameL.setHorizontalAlignment(SwingConstants.***CENTER***);

custInfoP1.add(custNameL);

custNameTF = **new** JTextField();

custNameTF.setEditable(**false**);

custInfoP1.add(custNameTF);

custNameTF.setColumns(10);

**JLabel** **custContactL** = **new** JLabel("Customer Contact");

custContactL.setHorizontalAlignment(SwingConstants.***CENTER***);

custInfoP1.add(custContactL);

custContactTF = **new** JTextField();

custContactTF.setEditable(**false**);

custInfoP1.add(custContactTF);

custContactTF.setColumns(10);

custInfoP2 = **new** JPanel();

custInfoP.add(custInfoP2);

custInfoP2.setLayout(**new** GridLayout(2, 2, 0, 10));

**JLabel** **custIdL** = **new** JLabel("Customer ID");

custIdL.setHorizontalAlignment(SwingConstants.***CENTER***);

custInfoP2.add(custIdL);

custIdTF = **new** JTextField();

custIdTF.setEditable(**false**);

custInfoP2.add(custIdTF);

custIdTF.setColumns(10);

**JLabel** **pointsL** = **new** JLabel("Loyalty Points");

pointsL.setHorizontalAlignment(SwingConstants.***CENTER***);

custInfoP2.add(pointsL);

pointsTF = **new** JTextField();

pointsTF.setEditable(**false**);

custInfoP2.add(pointsTF);

pointsTF.setColumns(10);

getContentPane().add(custInfoP);

billP = **new** JPanel();

billP.setBounds(680, 229, 400, 130);

getContentPane().add(billP);

billP.setLayout(**new** GridLayout(0, 2, 0, 10));

totalBillL = **new** JLabel("Total Bill Rs.");

totalBillL.setHorizontalAlignment(SwingConstants.***CENTER***);

billP.add(totalBillL);

totalBillTF = **new** JTextField();

totalBillTF.setEditable(**false**);

billP.add(totalBillTF);

totalBillTF.setColumns(10);

pointsUsedL = **new** JLabel(" Loyalty Points Used");

pointsUsedL.setHorizontalAlignment(SwingConstants.***LEFT***);

billP.add(pointsUsedL);

pointsUsedTF = **new** JTextField();

billP.add(pointsUsedTF);

pointsUsedTF.setColumns(10);

pointsUsedTF.getDocument().addDocumentListener(**new** DocumentListener(){

***@Override***

**public** **void** **insertUpdate**(DocumentEvent e) {

updateFields();

}

***@Override***

**public** **void** **removeUpdate**(DocumentEvent e) {

updateFields();

}

***@Override***

**public** **void** **changedUpdate**(DocumentEvent e) {

updateFields();

}

}

);

**JLabel** **payAmountL** = **new** JLabel("Payable Amount Rs.");

payAmountL.setHorizontalAlignment(SwingConstants.***CENTER***);

billP.add(payAmountL);

payAmountTF = **new** JTextField();

payAmountTF.setEditable(**false**);

payAmountTF.setColumns(10);

billP.add(payAmountTF);

payMethodP = **new** JPanel();

payMethodP.setBounds(270, 229, 400, 130);

getContentPane().add(payMethodP);

payMethodP.setLayout(**new** GridLayout(3, 1, 0, 0));

payMethodBG = **new** ButtonGroup();

payCashRB = **new** JRadioButton("Payment By Cash");

payCashRB.setHorizontalAlignment(SwingConstants.***LEFT***);

payMethodP.add(payCashRB);

payDebitRB = **new** JRadioButton("Payment By Debit Card");

payDebitRB.setHorizontalAlignment(SwingConstants.***LEFT***);

payMethodP.add(payDebitRB);

payCreditRB = **new** JRadioButton("Payment By Credit Card");

payCreditRB.setHorizontalAlignment(SwingConstants.***LEFT***);

payMethodP.add(payCreditRB);

payMethodBG.add(payCashRB);

payMethodBG.add(payDebitRB);

payMethodBG.add(payCreditRB);

payB = **new** JButton("Pay");

payB.setBounds(930, 427, 150, 40);

payB.addActionListener(**this**);

getContentPane().add(payB);

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

warningL.setForeground(**Color**.***RED***);

warningL.setBounds(270, 427, 400, 14);

getContentPane().add(warningL);

}

**void** **setLoginType**(**boolean** loginType) **throws** **SQLException**, **ClassNotFoundException** {

totalBillTF.setText(""+ **Globals**.*BILL\_AMOUNT*);

payAmountTF.setText(""+ **Globals**.*BILL\_AMOUNT*);

pointsUsedTF.setText("");

**if**(loginType == **Globals**.***CUST\_LOGIN***){

resultSet = dBConnection.custLogin(**Globals**.*CUSTOMER\_ID*);

setCustInfo();

guestInfoP.setVisible(**false**);

custInfoP.setVisible(**true**);

billP.add(totalBillL,0);

billP.add(totalBillTF,1);

billP.add(pointsUsedL, 2);

billP.add(pointsUsedTF, 3);

payCashRB.setSelected(**true**);

dBConnection.disconnect();

}

**else** **if**(loginType == **Globals**.***GUEST\_LOGIN***){

billP.remove(totalBillL);

billP.remove(totalBillTF);

billP.remove(pointsUsedTF);

billP.remove(pointsUsedL);

custInfoP.setVisible(**false**);

guestInfoP.setVisible(**true**);

payCashRB.setSelected(**true**);

}

}

**void** **reset**(){

custNameTF.setText("");

custIdTF.setText("");

custContactTF.setText("");

pointsTF.setText("");

pointsUsedTF.setText("");

totalBillTF.setText("");

payAmountTF.setText("");

payCashRB.setSelected(**true**);

warningL.setText("");

}

**public** **void** **setCustInfo**() **throws** **SQLException**{

resultSet.next();

custNameTF.setText(resultSet.getString("cust\_name"));

custIdTF.setText(resultSet.getString("cust\_id"));

custContactTF.setText(resultSet.getString("cust\_mobile"));

pointsTF.setText(resultSet.getString("cust\_points"));

**Globals**.*POINTS* = resultSet.getInt("cust\_points");

}

**public** **void** **updateFields**(){

**int** **pointsUsed** = 0;

**try**{

pointsUsed = **Integer**.*parseInt*(pointsUsedTF.getText());

**if**(pointsUsed > **Globals**.*POINTS*){

warningL.setText("NOT ENOUGH POINTS !!!");

pointsTF.setText("" + **Globals**.*POINTS*);

payAmountTF.setText("" + **Globals**.*BILL\_AMOUNT*);

payB.setEnabled(**false**);

}

**else** **if**(**Globals**.*BILL\_AMOUNT* < pointsUsed){

warningL.setText("CAN NOT USE POINTS MORE THAN BILL AMOUNT !!!");

pointsTF.setText("" + **Globals**.*POINTS*);

payAmountTF.setText("" + **Globals**.*BILL\_AMOUNT*);

payB.setEnabled(**false**);

}

**else**{

pointsTF.setText("" + (**Globals**.*POINTS* - pointsUsed));

payAmountTF.setText("" + (**Globals**.*BILL\_AMOUNT* - pointsUsed));

warningL.setText("");

payB.setEnabled(**true**);

}

}

**catch**(**NumberFormatException** **e2**){

warningL.setText("ENTER INTEGER VALUE ONLY !!!");

pointsTF.setText("" + **Globals**.*POINTS*);

payAmountTF.setText("" + **Globals**.*BILL\_AMOUNT*);

}

}

***@Override***

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

**if**(e.getSource() == payB){

**int** **pointsUsed** = 0;

**try** {

**try** {

pointsUsed = **Integer**.*parseInt*(pointsUsedTF.getText());

} **catch** (**NumberFormatException** **e1**) {

pointsUsed = 0;

}

**if**(pointsUsed != 0){

dBConnection.updatePoints(**Globals**.*POINTS* - pointsUsed);

}

**if**(payCashRB.isSelected())

dBConnection.insertPurchaseRecord(**Globals**.***CASH***);

**else** **if**(payCreditRB.isSelected())

dBConnection.insertPurchaseRecord(**Globals**.***CREDIT***);

**else**

dBConnection.insertPurchaseRecord(**Globals**.***DEBIT***);

} **catch** (**SQLException** **e1**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**catch** (**ClassNotFoundException** **e2**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database Driver !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**finally**{

dBConnection.disconnect();

}

**JOptionPane**.*showMessageDialog*(**this**, "Purchase successful", "DONE", **JOptionPane**.***INFORMATION\_MESSAGE***);

controller.nextFrame(**this**);

}

}

}

**Class - CustomerRegistration.java**

**package** shop\_management;

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Toolkit;

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

**import** java.awt.event.ActionListener;

**import** java.sql.SQLException;

**import** java.awt.event.ActionEvent;

**import** java.awt.GridLayout;

***@SuppressWarnings***("serial")

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

**DBConnection** dBConnection;

**JButton** registerB, backB;

**Controller** controller;

**private** **JTextField** nameTF;

**private** **JTextField** contactTF;

**public** **CustomerRegistration**(**Controller** controller, **DBConnection** dBConnection) {

**this**.dBConnection = dBConnection;

**this**.controller = controller;

getContentPane().setBackground(**Color**.***LIGHT\_GRAY***);

setResizable(**true**);

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

setTitle("New Customer Registration");

**Toolkit** **tk** = **Toolkit**.*getDefaultToolkit*();

**Dimension** **dim** = tk.getScreenSize();

setSize(dim.width, dim.height);

setBackground(**Color**.***LIGHT\_GRAY***);

setExtendedState(***MAXIMIZED\_BOTH*** );

getContentPane().setLayout(**null**);

**JPanel** **outerP** = **new** JPanel();

outerP.setBounds(468, 205, 482, 218);

getContentPane().add(outerP);

outerP.setLayout(**null**);

**JPanel** **innerP** = **new** JPanel();

innerP.setBounds(52, 67, 381, 89);

outerP.add(innerP);

innerP.setLayout(**new** GridLayout(2, 2, 10, 15));

**JLabel** **lblNewLabel\_1** = **new** JLabel("CUSTOMER NAME");

innerP.add(lblNewLabel\_1);

nameTF = **new** JTextField();

innerP.add(nameTF);

nameTF.setColumns(10);

**JLabel** **lblNewLabel** = **new** JLabel("MOBILE NO.");

innerP.add(lblNewLabel);

contactTF = **new** JTextField();

innerP.add(contactTF);

contactTF.setColumns(10);

registerB = **new** JButton("REGISTER");

registerB.setBounds(655, 473, 130, 35);

registerB.addActionListener(**this**);

getContentPane().add(registerB);

**JLabel** **infoL** = **new** JLabel("NEW CUSTOMER REGISTRATION",SwingConstants.***CENTER***);

infoL.setBounds(584, 159, 238, 35);

getContentPane().add(infoL);

backB = **new** JButton("<< BACK ");

backB.setBounds(130, 121, 140, 35);

backB.addActionListener(**this**);

getContentPane().add(backB);

}

**void** **reset**(){

nameTF.setText("");

contactTF.setText("");

}

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

**if**(e.getSource() == registerB){

**try**{

**Long** **contact** = **Long**.*parseLong*(contactTF.getText());

**String** **name** = nameTF.getText();

**if**(name.trim().equals(""))

**throw** **new** Exception();

**Long** **id** = dBConnection.insertNewCustomer(name, contact);

**JOptionPane**.*showMessageDialog*(**this**, "Registration Successful !!!\nCustomer ID is " + id, "Success", **JOptionPane**.***INFORMATION\_MESSAGE***);

reset();

controller.nextFrame(**this**);

}

**catch**(**NumberFormatException** **e1**){

**JOptionPane**.*showMessageDialog*(**this**, "Please enter a numeric contact number !!!", "ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**catch** (**SQLException** **e2**) {

**JOptionPane**.*showMessageDialog*(**this**, "Mobile number cannot have more than 10 digits !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**catch** (**Exception** **e3**) {

**JOptionPane**.*showMessageDialog*(**this**, "Please enter a name !!!", "ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**finally** {

dBConnection.disconnect();

}

}

**else** **if**(e.getSource() == backB){

reset();

controller.nextFrame(**this**);

}

}

}

**Class - PreviousRecords.java**

**package** shop\_management;

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Toolkit;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.ItemEvent;

**import** java.awt.event.ItemListener;

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

**import** java.awt.FlowLayout;

**import** java.awt.GridLayout;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.Calendar;

**import** java.util.Date;

**import** javax.swing.border.MatteBorder;;

***@SuppressWarnings***("serial")

**public** **class** **PreviousRecords** **extends** **JFrame** **implements** ActionListener, ItemListener{

**DBConnection** dBConnection;

**Controller** controller;

**private** **JTextField** custIdTF;

**ButtonGroup** choiceBG = **new** ButtonGroup();

**JComboBox**<String> custTypeCB;

**JSpinner** endDateS, startDateS;

**JPanel** panel;

**JTextArea** outputTA;

**JRadioButton** recordsRB, statsRB;

**JButton** backB, showRecordsB;

**public** **PreviousRecords**(**Controller** controller, **DBConnection** dBConnection) {

**this**.controller = controller;

**this**.dBConnection = dBConnection;

setResizable(**true**);

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

setTitle("Purchase Records");

**Toolkit** **tk** = **Toolkit**.*getDefaultToolkit*();

**Dimension** **dim** = tk.getScreenSize();

setSize(dim.width, dim.height);

getContentPane().setBackground(**Color**.***LIGHT\_GRAY***);

setExtendedState(***MAXIMIZED\_BOTH*** );

getContentPane().setLayout(**null**);

**JLabel** **lbl1** = **new** JLabel("SELECT PREFERENCES FOR SHOWING RECORDS :");

lbl1.setBounds(286, 106, 358, 14);

getContentPane().add(lbl1);

**JPanel** **choiceP** = **new** JPanel();

choiceP.setBounds(286, 131, 800, 33);

getContentPane().add(choiceP);

choiceP.setLayout(**new** GridLayout(0, 2, 0, 0));

statsRB = **new** JRadioButton("SALES STATISTICS");

choiceP.add(statsRB);

recordsRB = **new** JRadioButton("PURCHASE RECORDS");

recordsRB.setHorizontalAlignment(SwingConstants.***TRAILING***);

choiceP.add(recordsRB);

choiceBG.add(statsRB);

choiceBG.add(recordsRB);

showRecordsB = **new** JButton("Show Records");

showRecordsB.setBounds(876, 336, 150, 40);

showRecordsB.addActionListener(**this**);

getContentPane().add(showRecordsB);

**JLabel** **lbl2** = **new** JLabel("SHOW RECORDS FOR:");

lbl2.setBounds(286, 200, 233, 14);

getContentPane().add(lbl2);

**JPanel** **chooseTimePeriodP** = **new** JPanel();

chooseTimePeriodP.setBounds(599, 225, 487, 100);

getContentPane().add(chooseTimePeriodP);

chooseTimePeriodP.setLayout(**new** FlowLayout(**FlowLayout**.***CENTER***, 100, 10));

startDateS = **new** JSpinner();

chooseTimePeriodP.add(startDateS);

**JLabel** **lbl5** = **new** JLabel(" TO ");

chooseTimePeriodP.add(lbl5);

endDateS = **new** JSpinner();

chooseTimePeriodP.add(endDateS);

**JPanel** **chooseCustTypeP** = **new** JPanel();

chooseCustTypeP.setBounds(286, 225, 293, 100);

getContentPane().add(chooseCustTypeP);

custIdTF = **new** JTextField();

custIdTF.setEnabled(**false**);

custIdTF.setBounds(153, 60, 102, 20);

custIdTF.setColumns(12);

chooseCustTypeP.setLayout(**null**);

**String** **custType**[] = {"Registered Customers","Guest Customers","All customers","Search by Cust ID"};

custTypeCB = **new** JComboBox<String>(custType);

custTypeCB.setBounds(30, 23, 225, 20);

chooseCustTypeP.add(custTypeCB);

custTypeCB.addItemListener(**this**);

**JLabel** **lbl4** = **new** JLabel("Enter Cust ID : ");

lbl4.setBounds(30, 63, 107, 14);

chooseCustTypeP.add(lbl4);

chooseCustTypeP.add(custIdTF);

**JLabel** **lbl3** = **new** JLabel("SHOW RECORDS FROM :");

lbl3.setBounds(596, 200, 200, 14);

getContentPane().add(lbl3);

backB = **new** JButton("<< BACK");

backB.setBounds(82, 106, 150, 40);

backB.addActionListener(**this**);

getContentPane().add(backB);

panel = **new** JPanel();

panel.setBorder(**new** MatteBorder(1, 1, 1, 1, (**Color**) **Color**.***GRAY***));

panel.setBounds(286, 387, 800, 280);

getContentPane().add(panel);

panel.setLayout(**null**);

outputTA = **new** JTextArea();

outputTA.setEditable(**false**);

**JScrollPane** **scrollPane** = **new** JScrollPane(outputTA);

scrollPane.setBounds(10, 11, 780, 258);

panel.add(scrollPane);

}

**void** **reset**(){

statsRB.setSelected(**true**);

custTypeCB.setSelectedIndex(0);

custIdTF.setText("");

outputTA.setText("");

**Long** **currentTime** = **System**.*currentTimeMillis*();

startDateS.setModel(**new** SpinnerDateModel(**new** Date(1262284200000L), **new** Date(1262284200000L), **new** Date(currentTime), **Calendar**.***DAY\_OF\_MONTH***));

endDateS.setModel(**new** SpinnerDateModel(**new** Date(currentTime), **new** Date(1262284200000L), **new** Date(currentTime), **Calendar**.***DAY\_OF\_MONTH***));

}

***@Override***

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

**if**(custTypeCB.getSelectedIndex() == **Globals**.***SINGLE\_CUST***){

**try** {

**int** **id** = **Integer**.*parseInt*(custIdTF.getText());

**if**(id == 0){

**JOptionPane**.*showMessageDialog*(**null**, " NOT A VALID CUSTOMER ID !!!", **null**, **JOptionPane**.***ERROR\_MESSAGE***);

**return**;

}

**try** {

ResultSet **resultSet** = dBConnection.custLogin(id);

**if**(!resultSet.isBeforeFirst()){

**JOptionPane**.*showMessageDialog*(**null**, " NOT A VALID CUSTOMER ID !!!", **null**, **JOptionPane**.***ERROR\_MESSAGE***);

**return**;

}

} **catch** (**SQLException** **e1**) {

dBConnection.disconnect();

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

**return**;

} **catch** (**ClassNotFoundException** **e2**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database Driver !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**finally**{

dBConnection.disconnect();

}

}

**catch**(**NumberFormatException** **e3**){

**JOptionPane**.*showMessageDialog*(**null**, "ENTER NUMERIC CUSTOMER ID !!!", **null**, **JOptionPane**.***ERROR\_MESSAGE***);

**return**;

}

}

**if**(e.getSource() == showRecordsB){

**Globals**.*START\_DATE* = (**Date**)startDateS.getValue();

**Globals**.*END\_DATE* = (**Date**)endDateS.getValue();

**try** {

**if**(statsRB.isSelected())

callForStats();

**else**

callForRecords();

} **catch** (**SQLException** **e1**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

} **catch** (**ClassNotFoundException** **e2**) {

**JOptionPane**.*showMessageDialog*(**this**, "Something is wrong with the database Driver !!!", "DATABASE ERROR", **JOptionPane**.***ERROR\_MESSAGE***);

}

**finally**{

dBConnection.disconnect();

}

}

**else** **if**(e.getSource() == backB){

controller.nextFrame(**this**);

}

}

**private** **void** **callForStats**() **throws** **SQLException**, **ClassNotFoundException**{

**String**[] **stats**;

outputTA.setText("");

**if**(custTypeCB.getSelectedIndex() == 3)

stats = dBConnection.getStats(**Integer**.*parseInt*(custIdTF.getText()));

**else**

stats = dBConnection.getStats(custTypeCB.getSelectedIndex());

**for**(**String** **str** : stats){

**if**(str == **null**)

**continue**;

outputTA.append(str+"\n");

}

}

**private** **void** **callForRecords**() **throws** **SQLException**, **NumberFormatException**, **ClassNotFoundException**{

ResultSet **rs**;

outputTA.setText("PURCHASE ID"

+"\t"+"| CUSTOMER"

+"\t\t"+"| NO. OF ITEMS"

+"\t"+"| AMOUNT"

+"\t"+"| MODE OF PAYMENT"

+"\t"+"| TIME OF PURCHASE"

+"\n");

**if**(custTypeCB.getSelectedIndex() == 3)

rs = dBConnection.getRecords(**Integer**.*parseInt*(custIdTF.getText()));

**else**

rs = dBConnection.getRecords(custTypeCB.getSelectedIndex());

**while**(rs.next()){

**String** **name** = rs.getString(2);

//trick to make all name length same so that they line up correctly in output textarea.

**if**(name.length()<20){

**int** **spacesRequired** = 20 - name.length();

**for**(**int** **i** = 0; i< spacesRequired; i++)

name.concat(" ");

}

outputTA.append("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

+ "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

outputTA.append(rs.getString(1)

+"\t "+name

+"\t\t "+rs.getString(3)

+"\t "+rs.getString(4)

+"\t "+rs.getString(5)

+"\t\t "+rs.getString(6)+"\n");

}

dBConnection.disconnect();

}

***@Override***

**public** **void** **itemStateChanged**(**ItemEvent** e) {

**if**(custTypeCB.getSelectedIndex() == 3)

custIdTF.setEnabled(**true**);

**else**

custIdTF.setEnabled(**false**);

}

}

**REFERENCES**

1. Elmasri and Navathe: Fundamentals of Database Systems, 5th Edition, Pearson Education.
2. Raghu Ramakrishnan and Johannes Gehrke: Database Management Systems, 3rd Edition, McGraw-Hill.
3. Silberschatz, Korth and Sudharshan: Data base System Concepts, 6th Edition, Mc-GrawHill, 2010.
4. C.J. Date, A. Kannan, S. Swamynatham: A Introduction to Database Systems, 8th Edition, Pearson education.
5. <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>
6. <https://www.siteground.com/tutorials/php-mysql/mysql.htm>
7. <https://en.wikipedia.org/wiki/Java(programming_language).htm>
8. <https://www.draw.io>
9. <https://www.gliffy.com>
10. <https://www.stackoverflow.com>
11. <http://www.tutorialspoint.com/dbms/>
12. <http://www.javatpoint.com/>