**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**BELAGAVI - 590 018**



**A MINI PROJECT REPORT**

**ON**

**VEGETABLE STOCK MANAGEMENT SYSTEM**

Submitted by

**ACHAL SREENIDHI**

4SF18CS004 4SF18CS158

In partial fulfillment of the requirements for the V semester B.E.(CSE)

### DBMS LABORATORY WITH MINI PROJECT

Under the Guidance of

**Ms. ALAKANANDA K**

Assistant Professor, Dept. of CSE

****

**Department of Computer Science & Engineering**

**SAHYADRI**

**College of Engineering & Management**

**Adyar, Mangaluru - 575 007**

**2020 - 2021**

**SAHYADRI**

**College of Engineering & Management**

**(**Affiliated to Visvesvaraya Technological University, BELAGAVI**)**

**Adyar, Mangaluru - 575 007**

**Department of Computer Science & Engineering**

**CERTIFICATE**

This is to certify that the project entitled “VEGETABLE STCOK MANAGEMENT SYSTEM”issubmitted in partial fulfillment for the requirement of V sem. B. E. (Computer Science & Engineering), “DBMS LABORATORY WITH MINI PROJECT” during the year 2020 – 21 is a result of bonafide work carried out by

ACHAL N 4SF18CS004

SREENIDHI 4SF18CS158

|  |  |  |
| --- | --- | --- |
| ———————— |  | ———————— |
| Ms. Alakananda K |  | Dr. Pushpalatha K |
| Assistant. Prof. Dept.of CS&E  SCEM, Mangaluru |  | HOD,Dept. of CS&E  SCEM, Mangaluru |
|  |  |  |

|  |  |
| --- | --- |
| **Signature of the Examiners** |  |
| 1 **…………………………..** |  |
| 2. **…………………………..** |  |

**ABSTRACT**

Vegetable Stock Management System is a project which is helpful for the businesses operate stores, where store owner keeps the records of purchase. And This project is also helpful for the customers for viewing the product details. This project eliminates the paperwork, human faults, manual delay and speed up the process.

Vegetable Stock Management System will have the ability to track overall worth and available stock. This is simple, fast and intelligent Stock management that can be used by anyone who has a smartphone. And it also helps the stores to calculate overall profit or loss.

Our website consists of login pages for customers and for the store owners. In that store owners can feed the product details like (it no, intake in KG, Rs etc.) and that can be updated regularly. And in the end, they can get the overall profit and the remaining stock information.

For the customers they can see the actual quantity of product stock that is available in each shop and they can see the details. This method will help the owners and customers to sell and buy the goods easily.

i

**ACKNOWLEDGEMENT**

It is with great satisfaction and euphoria that we are submitting the Mini Project Report on **“STOCK HUT”**. We have completed it as a part of the V semester DBMS Laboratory with Mini Project (15CSL58). It would be incomplete without thinking of the people who made it perfect with their constant guidance and encouragement.

We are profoundly indebted to our guides, **Ms. Alakananda K,** Assistant Professors, Department of Department of CSE for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Pushpalatha K**, Head & Associate Professor, Department of CSE for her invaluable support and guidance.

We sincerely thank **Dr. Rajesh S**, Principal, Sahyadri College of Engineering & Management, who have always been a great source of inspiration.

We also like to thank all Teaching and Non-teaching staff of Sahyadri College of Engineering and Management, Mangalore for their kind co-operation during the course of my work.

**Achal N(4SF18CS004)**

**Sreenidhi (4SF18CS158)**

ii

**PAGE INDEX**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Topic** | **Page No.** |
| 4.1  4.2  4.3      6. | **Introduction**  **Design**  E-R diagram  Relational schema  Schema diagram  **Normalization**  **Implementation**  System Requirements  Data Base and Table Structure  Functionalities  **Result**  **Conclusion**  **References** | 1  2  2  4  7  8  12  13  14  17  22  32 |

iii

**Chapter 1**

**INTRODUCTION**

A Database Management System (DBMS) refers to the technology for creating and managing databases, basically DBMS is a software tool to organize (create, retrieve, update and manage) data in a database.

Vegetable Stock Management System (STOCK HUT) is designed to help the store owner to easily maintain the available stocks present in his shop. The necessity of our project is that the customers can view the products that are available in the store and they can place their order from the home itself. It also helps for the store owners to update the item details that they have in their store and they can easily update it through our application. This also reduce the paper works all other work loads of store owner.

The implementation of this project involves the usage of MySQL for database, JAVA as programing language and NETBEANS IDE. MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many databases driven web applications, including Drupal, Joomla, php BB, and WordPress. MySQL is also used by many popular websites, including Facebook, Flickr, Media Wiki, Twitter, and YouTube.

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. The NetBeans Platform is a framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required.

**CHAPTER-2**

**DESIGN**

**2.1 ER DIAGRAM**

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information systems entity and the relationships between those entities.

Entity Class

Weak Entity Class

Relation type

Identifying Relation type

Attributes

Key Attributes

Multivalued Attributes

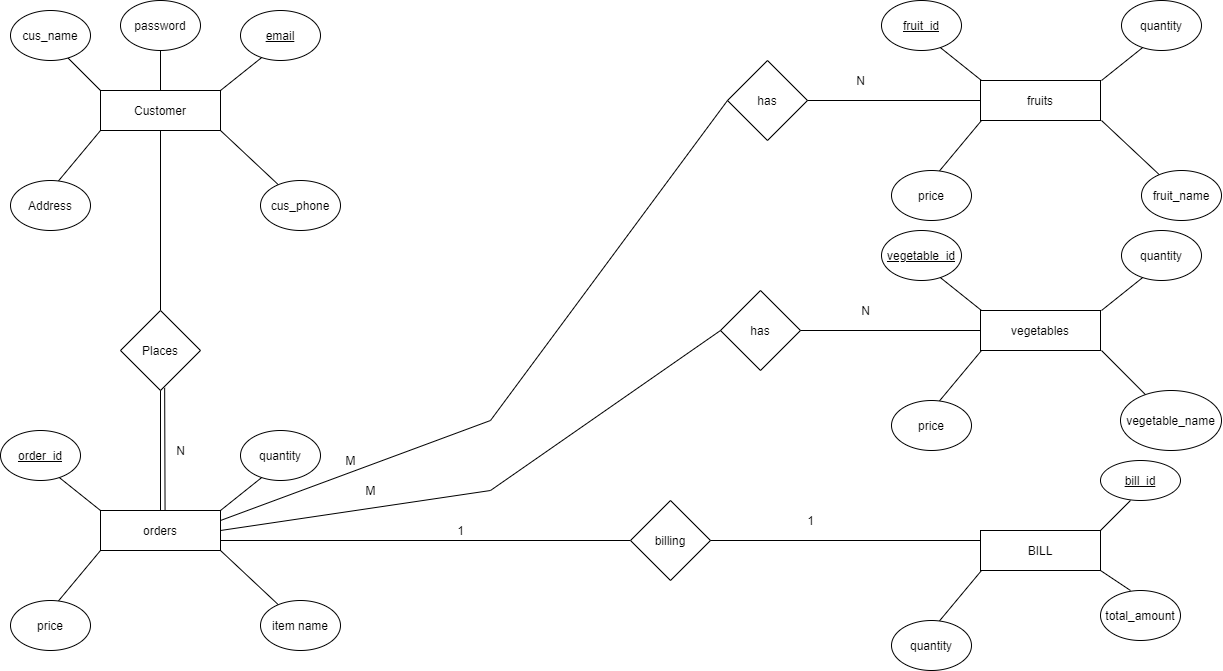
****

Figure 2.1: ER Diagram for STOCK HUT

**2.2 Relational Schema (ER to relational schema)**

**STEP 1: Mapping of Regular Entity Types**

Strong entities which have a unique attribute as primary key.

**Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cust\_email | Cust\_name | Address | Password | Cust\_phone |

**Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_id | Price | Item\_name | Quantity | Cust\_email |

**Fruits**

|  |  |  |  |
| --- | --- | --- | --- |
| Fruit\_id | Fruit\_name | Price | Quantity |

**Vegetables**

|  |  |  |  |
| --- | --- | --- | --- |
| Vegetable\_id | Vegetable\_name | Price | Quantity |

**Bill**

|  |  |  |
| --- | --- | --- |
| Bill\_id | Total\_amount | Quantity |

**STEP 2: Mapping of Weak Entity Types**

Since our ER diagram contains no weak entity, this step is ignored for the project.

**STEP 3: Mapping of Binary 1:1 Relationship Types**

There are three possibilities the relationship can be:

• mandatory at both ends

• mandatory at one end and optional at the other

• optional at both ends

**Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_id | Price | Item\_name | Quantity | Cust\_email |

**Bill**

|  |  |  |  |
| --- | --- | --- | --- |
| Bill\_id | Total\_amount | Quantity | Cust\_email |

**Order Bill**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_id | Bill\_id | Quantity | Cust\_email | Total\_amount |

**STEP 4: Mapping of Binary 1: N Relationship Types**

1: N refers to a one-to-many relationship (1 object on one side is related to many on

the other)

**Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cust\_email | Cust\_name | Address | Password | Cust\_phone |

**Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_id | Item\_name | Price | Quantity | Cust\_email |

**Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_id | Item\_name | Price | Quantity | Cust\_email |

**Fruits Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fruit\_id | Fruit\_name | Price | Quantity | Order\_id |

**Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order\_id | Item\_name | Price | Quantity | Cust\_email |

**Vegetable Order**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vegetable\_id | Vegetable\_name | Price | Quantity | Order\_id |

**STEP 5: Mapping of Binary M: N Relationship Types**

M: N refers to a many-to-many relationship (many objects on one side is related to many on the other).

The ERD of our project does not contain any M:N relation types.

**STEP 6: Mapping of Multivalued Attributes**

For each multivalued attribute A

The ERD of our project does not contain any Multivalued attributes.

• Create a new relation R

• Primary key of R is the combination of A and K

• If the multivalued attribute is composite, include its simple components

The ERD of our project does not contain any Multivalued attributes.

**STEP 7: Mapping of N-ary Relationship Types**

The ERD of our project does not contain any N-ary relation types..

**2.3 SCHEMA DIAGRAM**

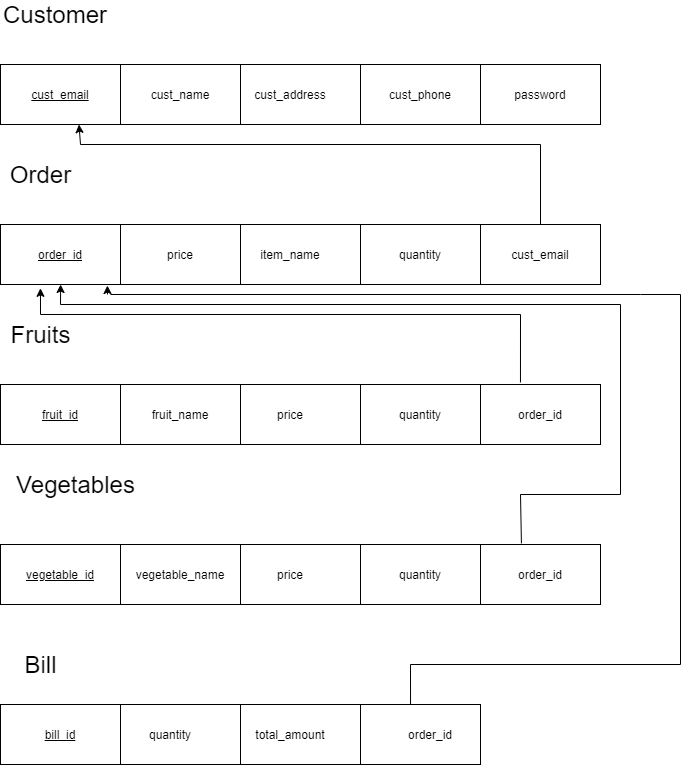


Figure 2.3.1 : Schema Diagram of STOCK HUT

**Chapter 3**

**NORMALIZATION**

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

As per the rule of 1NF, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

A table is said to be in 2NF if both the following conditions hold:

• Table is in 1NF (First normal form)

• No non-prime attribute is dependent on the proper subset of any candidate key of table.

A table design is said to be in 3NF if both the following conditions hold:

• Table must be in 2NF

• Transitive functional dependency of non-prime attribute on any super key

should be removed.

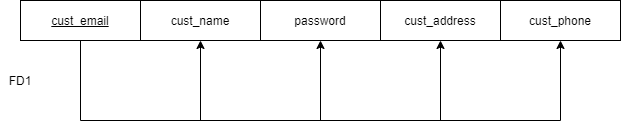
**CUSTOMER**

Figure 4.1: Normalization of Customer Table

**FD1** : {cust\_email} {cust\_name, password, cust\_address,cust\_phone}

The relations are in 1NF since there are no multivalued attributes or

nested relations.

The relations are in 2NF since no non-key attributes are functionally dependent on part of the primary key.

The relations are in 3NF since no non-key attributes are functionally determined by another non- key attribute.

The relation is in 1st,2nd and 3rd Normal Form.

**ORDER**

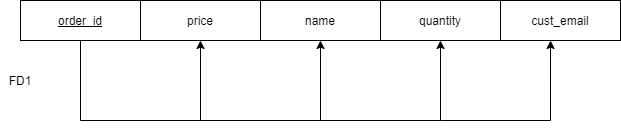
****

Figure 4.2: Normalization of Orders Table

**FD1**: {order\_id} {price, name,quantity,cust\_email}

The above relation is in 1NF because there are no multivalued attributes in the

relational schema.

The above relation is 2NF because all the attributes in the relational are fully

functionally dependent on primary key.

The above relation is 3NF because there is no transitive dependency on primary key.

The relation is in 1st,2nd and 3rd Normal Form.

**FRUITS**

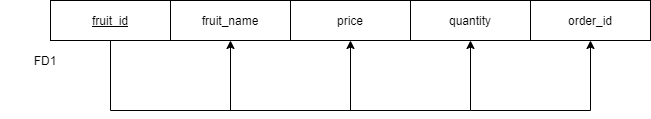


Figure 4.3: Normalization of Fruits Table

**FD1**:{fruit\_id} {fruit\_name,price,quantity,order\_id}

The above relation is in 1NF because there are no multivalued attributes in the relational

Schema.

The above relation is 2NF because all the attributes in the relational are fully functionally dependent on primary key.

The above relation is 3NF because there is no transitive dependency on primary key.

The relation is in 1st,2nd and 3rd Normal Form.

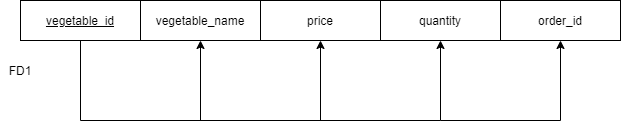
**VEGETABLES**

Figure 4.4: Normalization of Vegetables Table

**FD1**:{vegetable\_id} {vegetable\_name,price,quantity,order\_id}

The above relation is in 1NF because there are no multivalued attributes in the

relational Schema.

The above relation is 2NF because all the attributes in the relational are fully

functionally dependent on primary key.

The above relation is 3NF because there is no transitive dependency on

primary key.

The relation is in 1st,2nd and 3rd Normal Form

**BILLING**

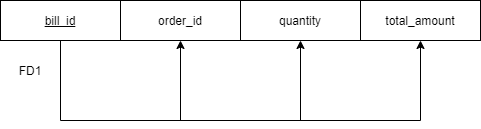
****

Figure 4.5: Normalization of Billing Table

**FD1**: {bill\_id} {order\_id, quantity, total\_amount}

The above relation is in 1NF because there are no multivalued attributes in the

relational Schema.

The above relation is 2NF because all the attributes in the relational are fully

functionally dependent on primary key.

The above relation is 3NF because there is no transitive dependency on

primary key.

The relation is in 1st,2nd and 3rd Normal Form.

**Chapter 4**

**IMPLEMENTATION**

**MySQL Version 8**

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons –

* MySQL is released under an open-source license.
* MySQL is a very powerful program in its own right. It handles a large subset

of the functionality of the most expensive and powerful database packages.

* MySQL uses a standard form of the well-known SQL data language.
* MySQL is customizable. The open-source GPL license allows programmers to

modify the MySQL software to fit their own specific environments.

**JAVA**

Java is the general-purpose computer programming language that is concurrent, class based, object oriented and specifically designed to have a few implementations dependencies as possible. Java Swings is a GUI widget toolkit for Java. It is a part of Oracle Java Foundation classes- an API for providing a graphical interface for the Java Programs.

**NETBEANS**

NetBeans is an integrated development environment for Java. NetBeans allows application to be developed from modular software components called modules. NetBeans run on Microsoft windows, Mac OS, Linux and Solaris.

The Alumni Management System (ALUMS SQUADRA) can be executed only on a single system that has MySQL Server and Java environment installed. The application can simply be executed by starting the server and Java Applications.

**4.1 SYSTEM REQUIREMENTS**

**4.1.1. Hardware Requirements:**

Processor: Any processor above 500 MHz

RAM: 4GB • Hard Disk: 500GB

Input Device: Standard keyboard and Mouse

Output Device: Monitor

**4.1.2. Software Requirements:**

Database : MySQL

Programming Language : Java

IDE : NetBeans 8.0.2

**4.2 DATABASE AND TABLE STRUCTURE**

**4.2.1 Create Tables and its Structures**

**Register**

CREATE TABLE REGISTER (C\_NAME VARCHAR (50), C\_EMAIL VARCHAR (50) PRIMARY KEY, C\_PHONE VARCHAR (30), ADDRESS VARCHAR (50), C\_PASSWORD VARCHAR (20),);

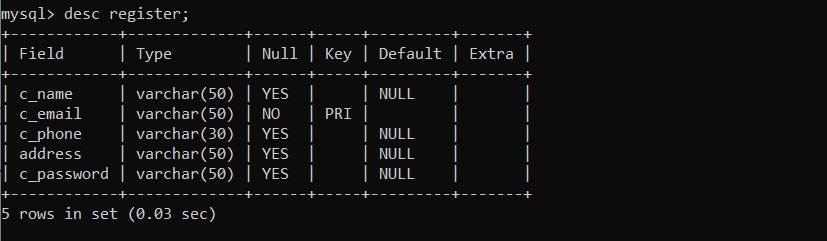


Fig 4.2.1.1 Register Table

**FRUITS**

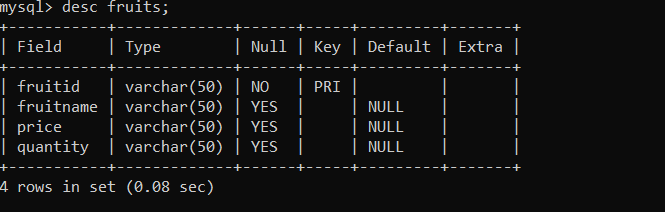
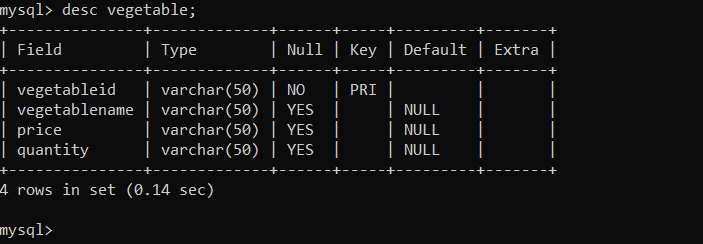
****CREATE TABLE FRUITS (FRUITID VARCHAR (50) PRIMARY KEY, FRUITNAME VARCHAR (50), PRICE VARCHAR (50), QUANTITY VARCHAR (50));

Fig 4.2.1.2 Fruits Table

**VEGETABLE**

CREATE TABLE VEGETABLE (VEGETABLEID VARCHAR (50) PRIMARY KEY, VEGETABLENAME VARCHAR (50), PRICE VARCHAR (20), QUANTITY VARCHAR (20));

Fig 4.2.1.3 Vegetable Table

**FRUITSORDER**

CREATE TABLE FRUITSORDER (ORDER\_ID INT (11) NOT NULL AUTO\_INCREMENT, C\_EMAIL VARCHAR (50), FRUITNAME VARCHAR (50), PRICE VARCHAR (20), QUANTITY VARCHAR (20), FOREIGN KEY (C\_EMAIL) REFERENCES REGISTER(C\_EMAIL));

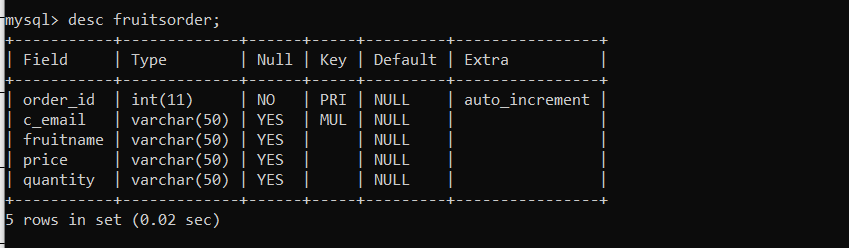
****

Fig 4.2.1.4 Fruitsorder Table

**VEGETABLEORDER**

CREATE TABLE VEGETABLEORDER (ORDER\_ID INT (11) NOT NULL AUTO\_INCREMENT, C\_EMAIL VARCHAR (50), VEGETABLENAME VARCHAR (50), PRICE VARCHAR (20), QUANTITY VARCHAR (20), FOREIGN KEY (C\_EMAIL) REFERENCES REGISTER(C\_EMAIL));

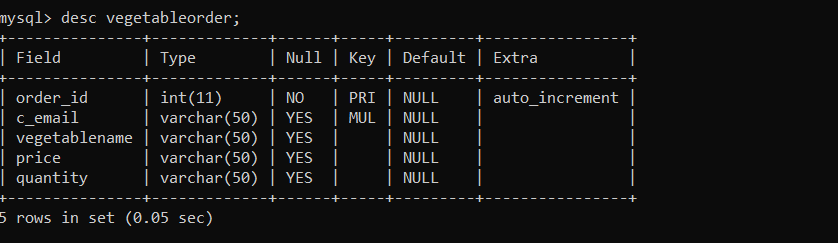


Fig 4.2.1.5 Vegetableorder Table

**BILLING**

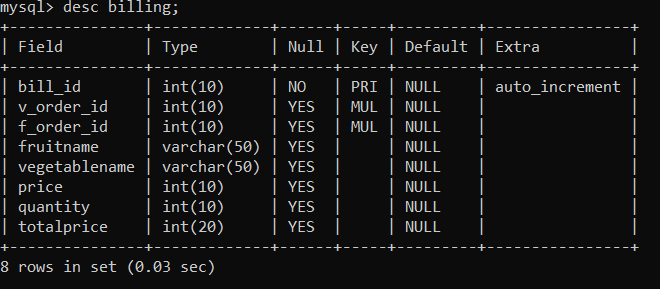
****CREATE TABLE BILLING (BILL\_ID INT (10) PRIMARY KEY, V\_ORDER\_ID INT (10), F\_ORDER\_ID INT (10), CUST\_ADDRESS VARCHAR (50), FRUITNAME VARCHAR (50), VEGETABLENAME VARCHAR (50), PRICE INT (10), QUANTITY INT (10), TOTALPRICE INT (20), FOREIGN KEY (V\_ORDER\_ID) REFERENCES VEGETABLEORDER(ORDER\_ID), FOREIGN KEY (F\_ORDER\_ID) REFERENCES FRUITSORDER(ORDER\_ID));

Fig 4.2.1.6 BillingTable

**4.3 FUNCTIONALITIES**

**4.3.1 CONNECTION**

The Frontend is done using Java and the Backend is done using MySQL. Java is connected to MySQL using JDK. Connection between Java and MySQL is done using following syntax:

try

{

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

Connection con=DriverManager.getConnection ("jdbc:mysql://localhost:3306/vegetables","

root","Shangershang");

return con;

}

Catch (Exception e){

System.out.println(e);

return null;

}

**4.3.2 INSERT**

Insert function is used to add user details, event details, job details, donation details into the database.

try{

String m=n.getText().trim();

String t=us.getText().trim();

String o=em.getText().trim();

String p=dob.getText().trim();

String q=ph.getText().trim();

String r=ut.getText().trim();

String s=pw.getText().trim();

Connection con = DriverManager.getConnection

("jdbc:mysql://localhost:3306/dbms?zeroDateTimeBehavior=

CONVERT\_TO\_NULL","root","Shangershang");

Statement st=con.createStatement();

st.execute("insert into register

values('"+m+"','"+t+"','"+o+"','"+p+"','"+q+"','"+r+"','"+s+"')");

JOptionPane.showMessageDialog(this, "Record Inserted");

new Login().setVisible(true);

this.dispose();

}

Catch (SQLException e){

JOptionPane.showMessageDialog(null, e);

}

**4.3.4 UPDATE**

String vegetableid=jTextField1.getText();

String vegetablename=jTextField4.getText();

String price=jTextField2.getText();

String quantity=jTextField3.getText();

try

{

Connection con=ConnectionProvider.getCon();

Statement st=con.createStatement();

st.executeUpdate("insert into vegetable values('"+vegetableid+"','"+vegetablename+"','"+price+"','"+quantity+"')");

JOptionPane.showMessageDialog(null,"Sucessfully updated");

jTextField1.setText("");

jTextField4.setText("");

jTextField2.setText("");

jTextField3.setText("");

}

catch (Exception e)

{

JOptionPane.showMessageDialog(null, “vegetable Id already exists");

setVisible(false);

new vegetables().setVisible(true);

}

**4.3.5 DISPLAY**

try

{

Connection con=ConnectionProvider.getCon();

Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select \* from vegetable");

jTable1.setModel(DbUtils.resultSetToTableModel(rs));

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,"CONNECTION NOT

ESATABLISHED");

}

**4.3.6 USAGE OF TRIGGERS**

Trigger operation is used to raise an error message if the Phone number entered in the user info is lesser or greater than 10 digits.

DELIMITER $$

CREATE TRIGGER customer\_phone

BEFORE INSERT

ON register FOR EACH ROW

BEGIN

if length(new.c\_phone)>10 or length(new.c\_phone)<10

then

signal sqlstate '45000'

set message\_text='phone number is not valid';

end if;

END$$

DELIMITER ;

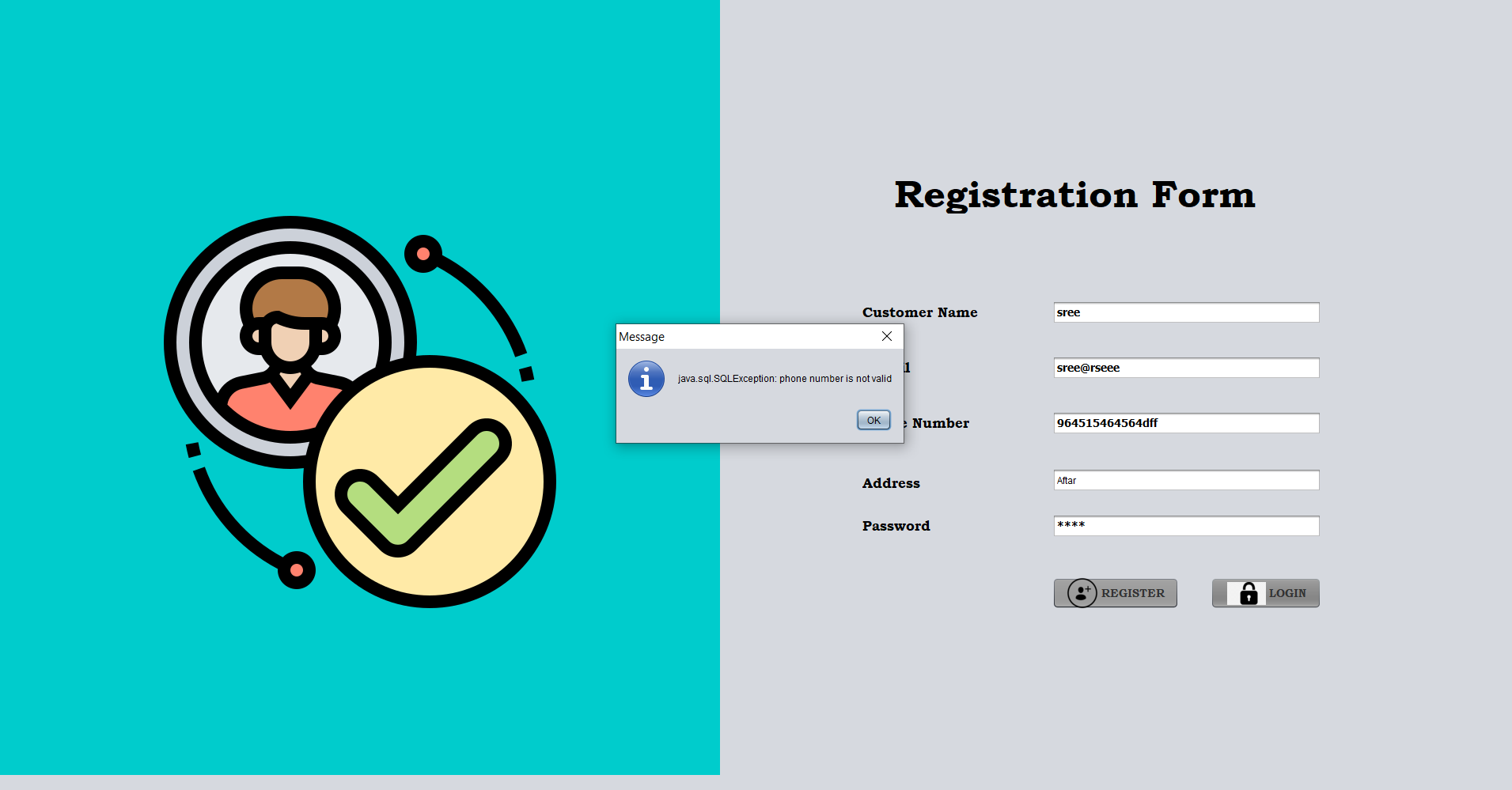


Fig 4.3.6 Trigger Function

**CHAPTER-5**

**RESULTS**

**5.1 SNAPSHOTS**

**5.1.1 FIRST PAGE**

The Fig 5.1.1 represents a simple user interface. This is the first page to either login or to register.

Fig 5.1.1 User Login

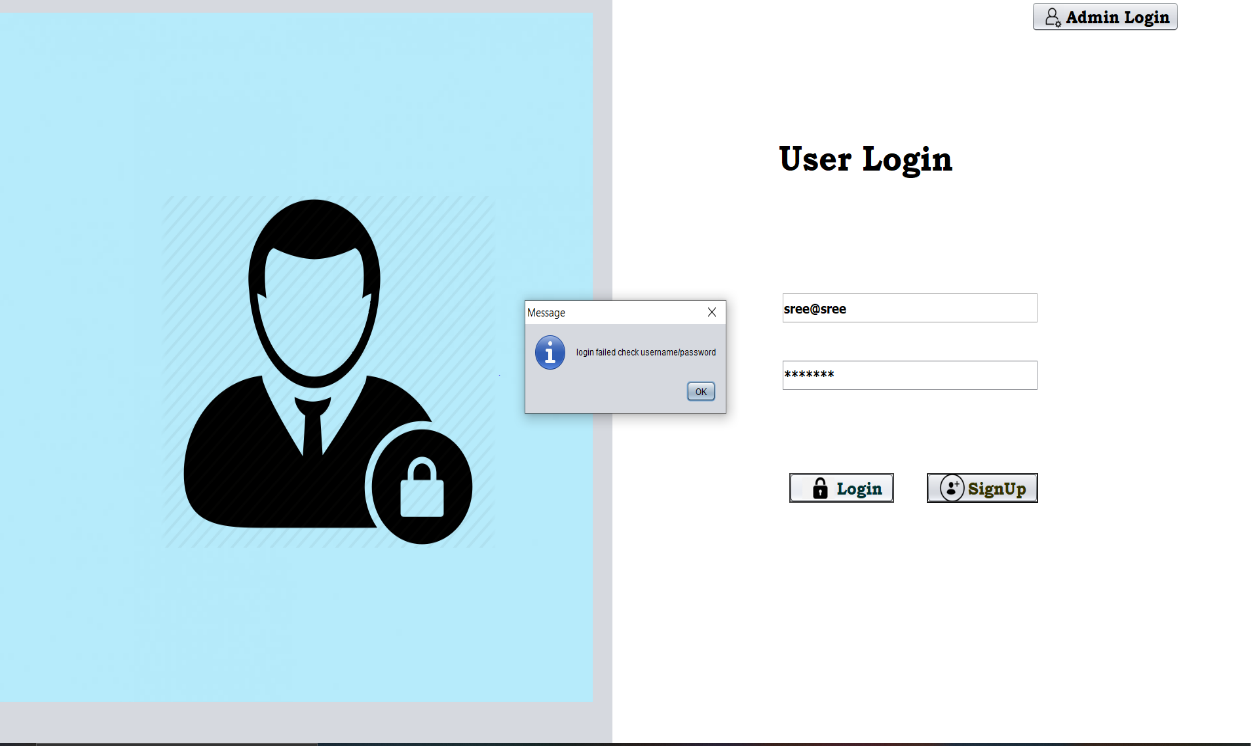


Fig 5.1.1.1 Successful User Login

**5.1.2 CUSTOMER REGISTRATION**

The Fig 5.1.2 represents the customer has to register him/herself with provided details;

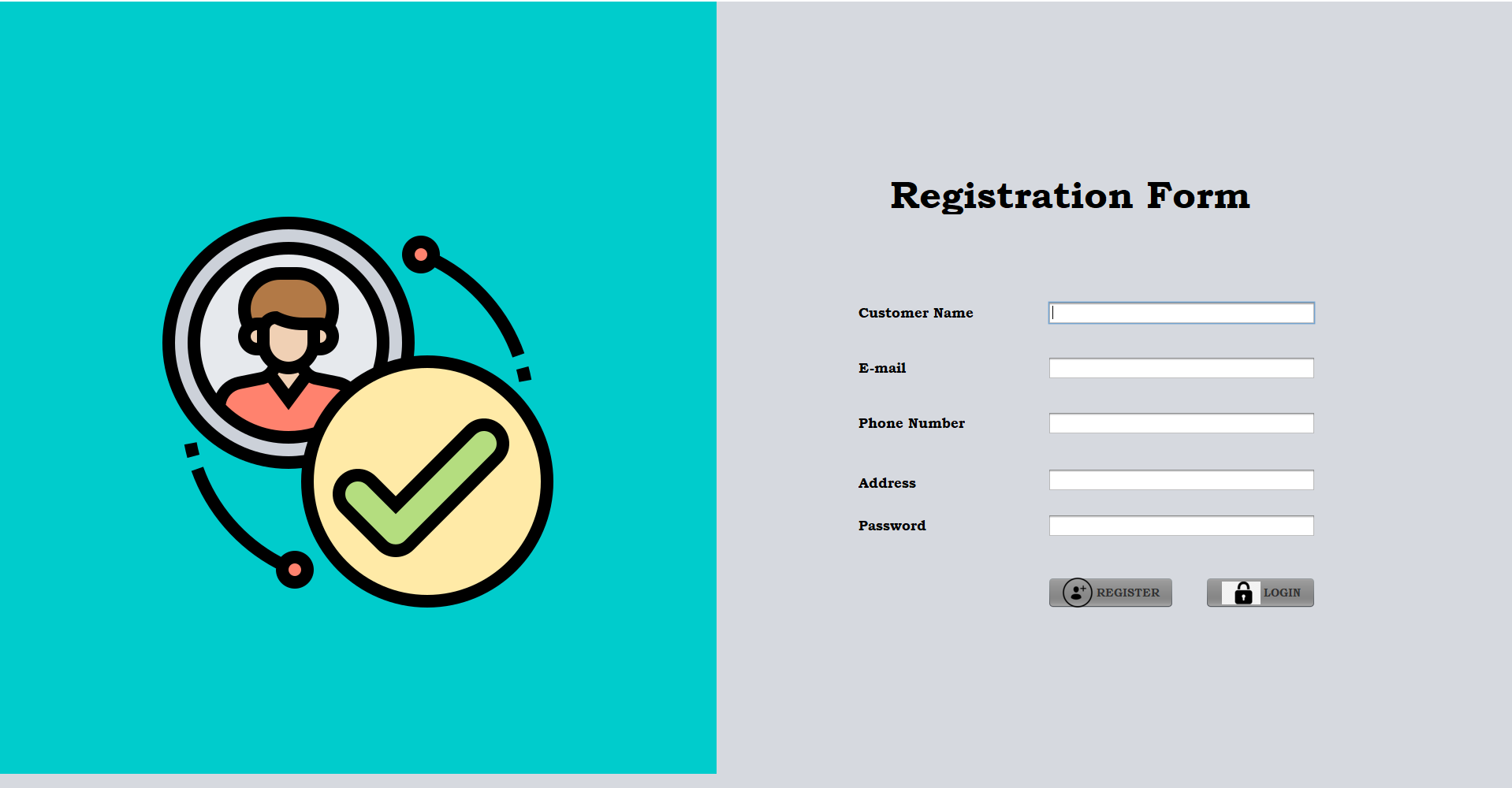
****

Fig 5.1.2 User Registration



Fig 5.1.2.1 User Registration Successful

**5.1.3 LOGIN AS HOST**

Fig 5.1.3 represents the owner login as host to enter the item details.

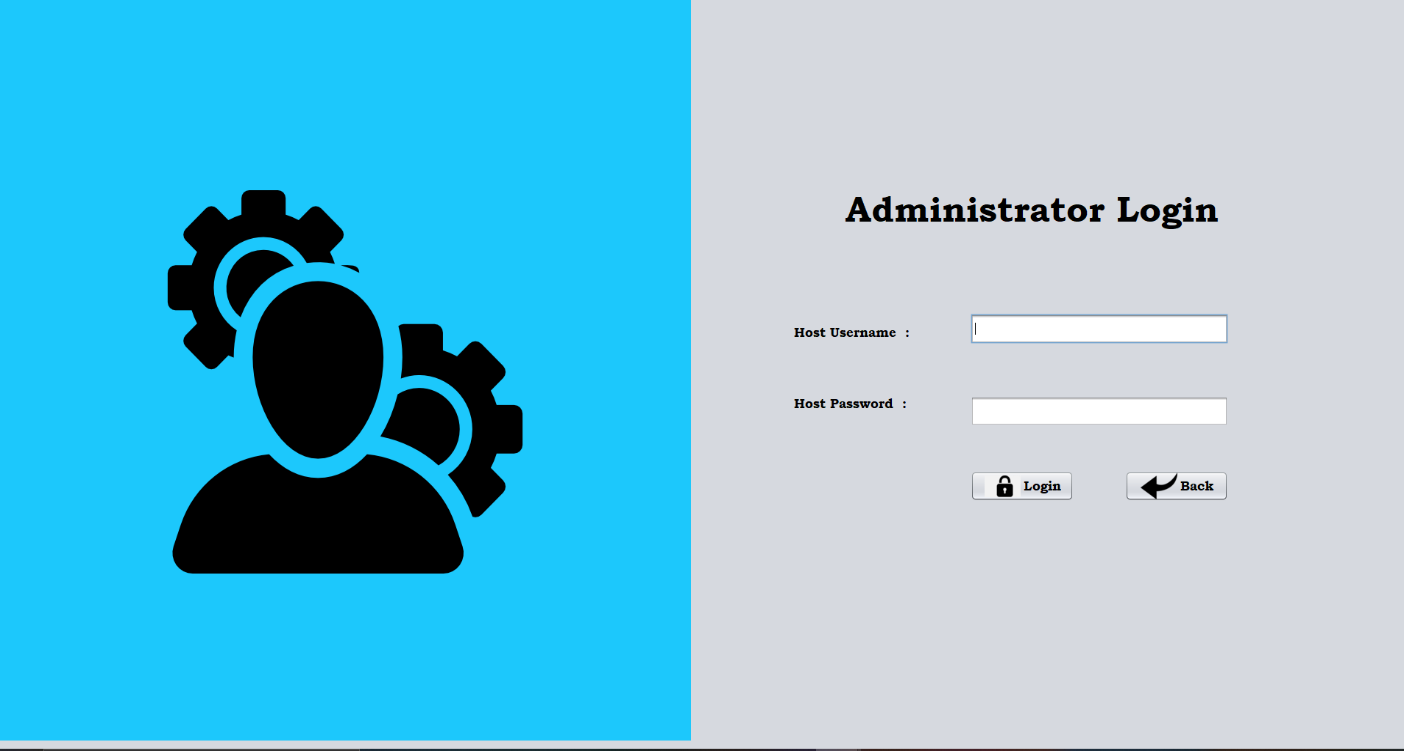


Fig 5.1.3 Administrator Login

**5.1.4 HOME PAGE FOR HOST**

The Fig 5.1.4 represents the sections were the store owner can add fruits and vegetables in a separate column.

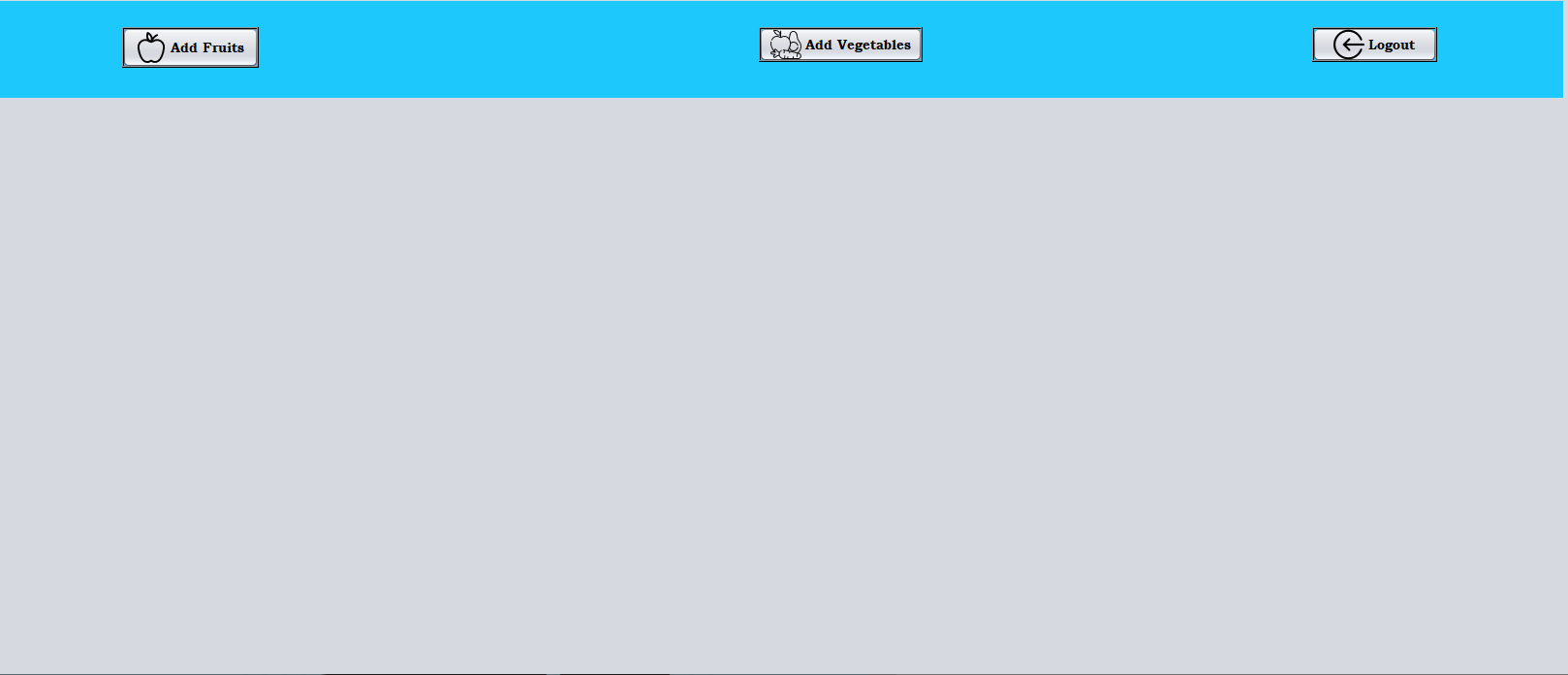


Fig 5.1.4 Home Page for Host

**5.1.6 UPDATE, INSERT AND DELETE ITEMS**

The Fig 5.1.6 represents the details of the items available in the shop. Here the store owner can insert or can even update the items that is present in his shop based on the availability. If an item is not available then he can delete the item.

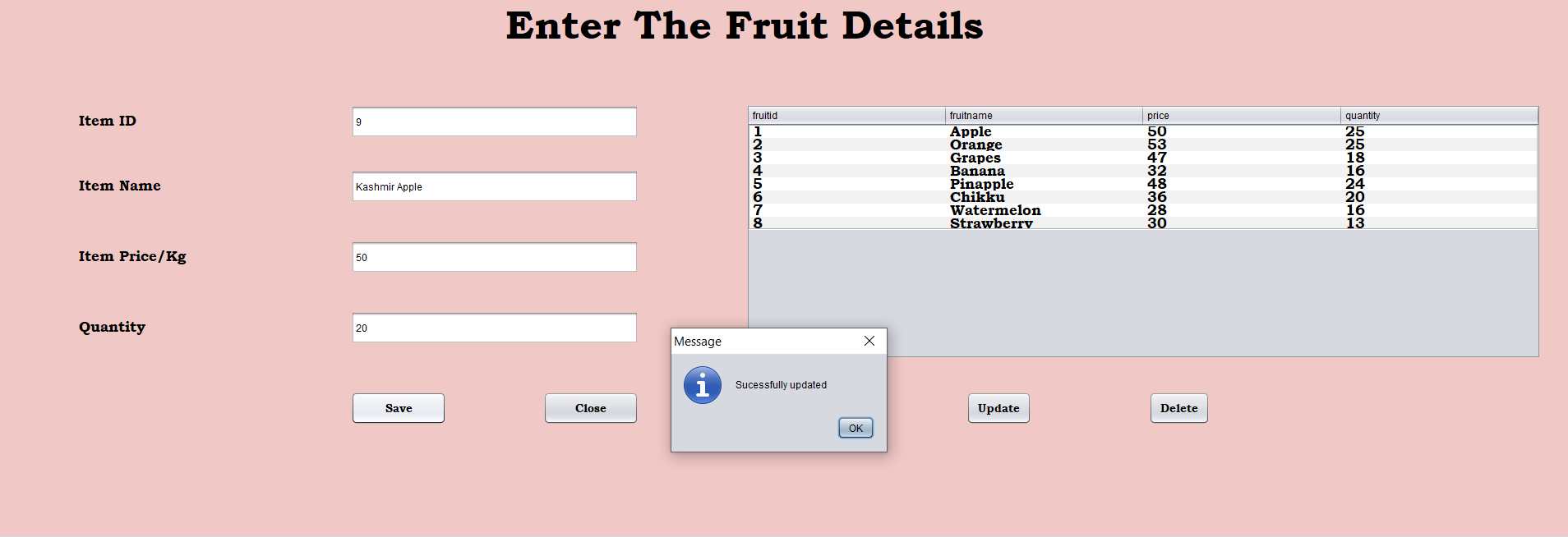
****

Fig 5.1.6 Inserting Items.

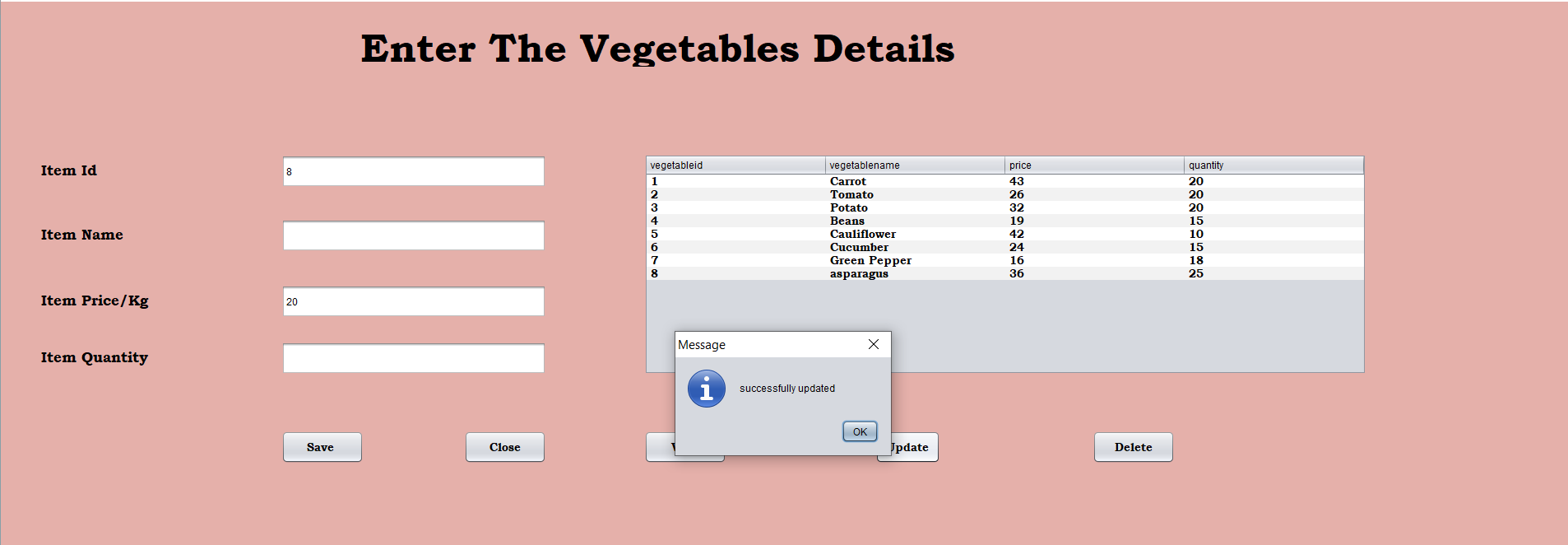


Fig 5.1.6.1 Updating Items

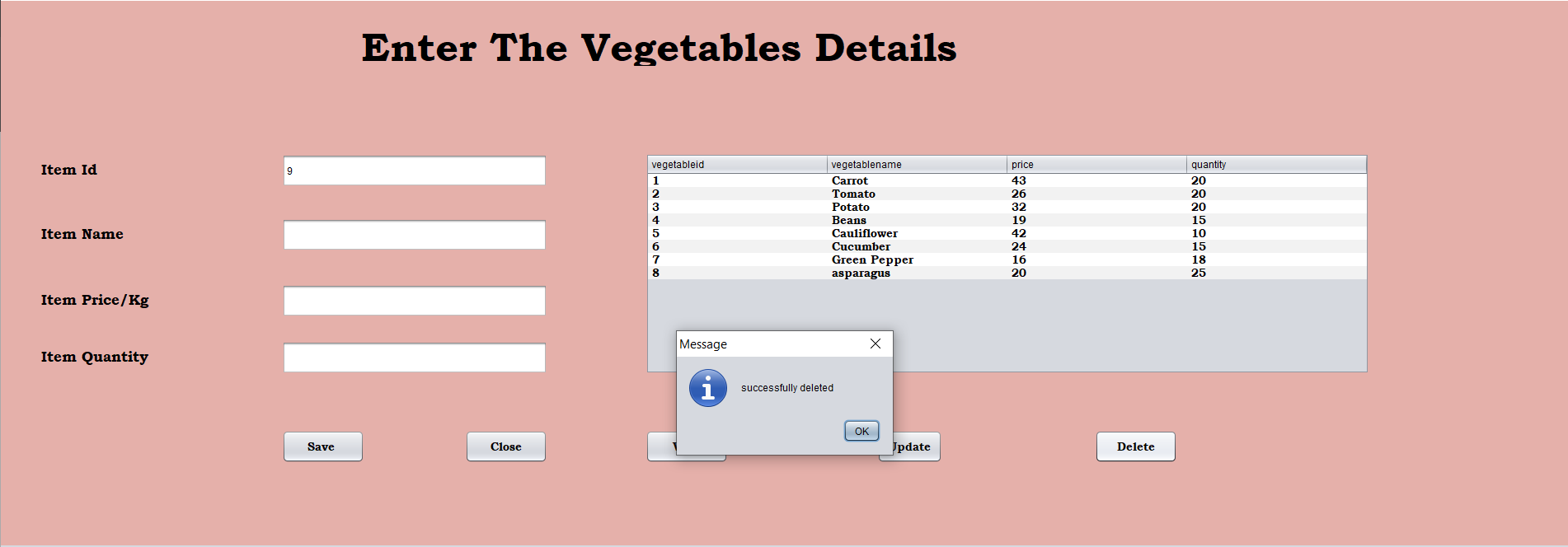


Fig 5.1.6 Details of The Items

**5.1.7 DASHBOARD FOR CUSTOMER**

The Fig 5.1.7 represents the Customer dashboard. Here the customer can choose vegetable or fruits based on his/her preferences.

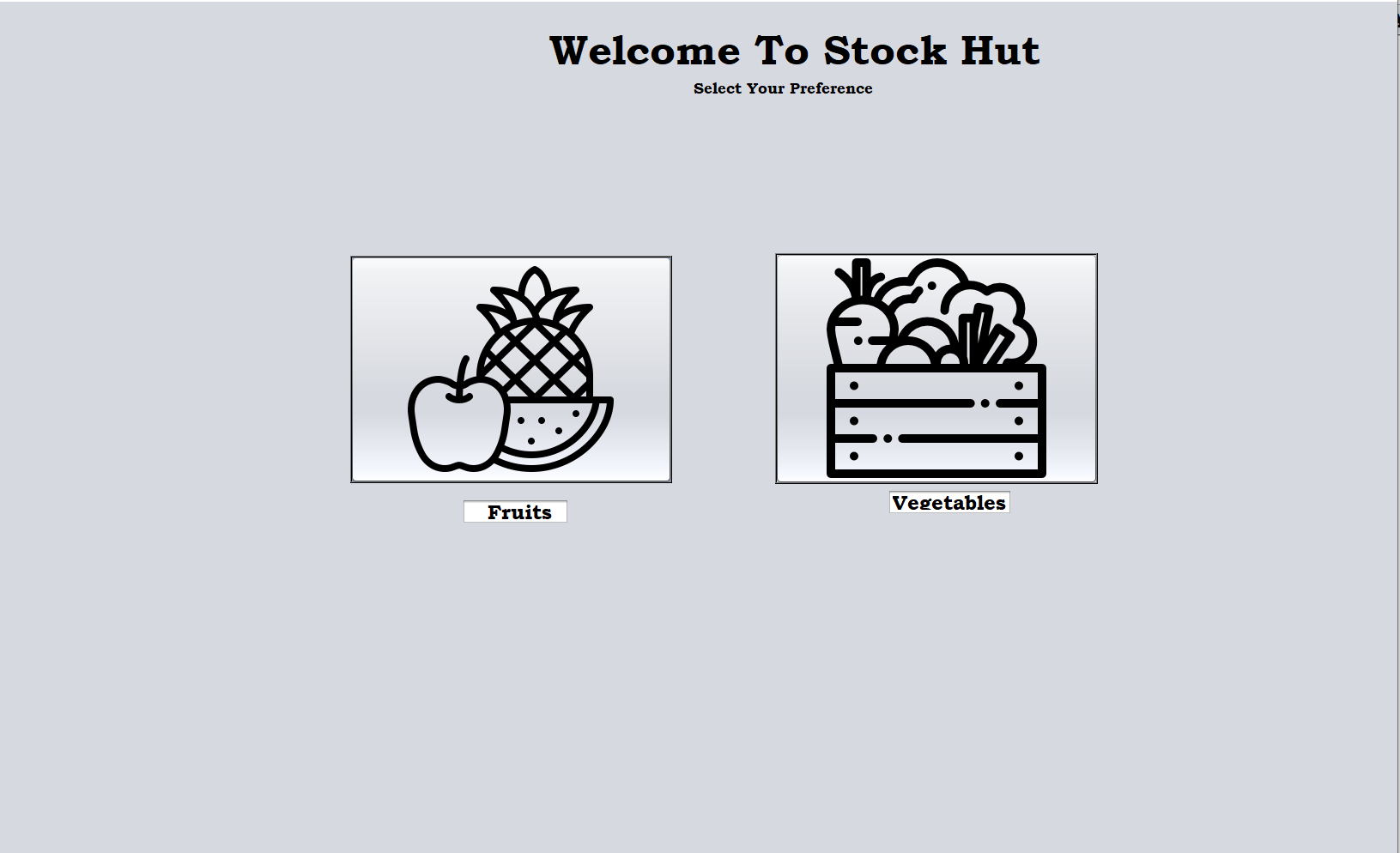
****

Fig 5.1.7 Customer Dashboard

**5.1.8 VIEW FRUITS AND VEGETABLE LIST**

The Fig 5.1.8 indicates that the customer can view the details of the items (fruits and vegetables) present in the owner’s shop.

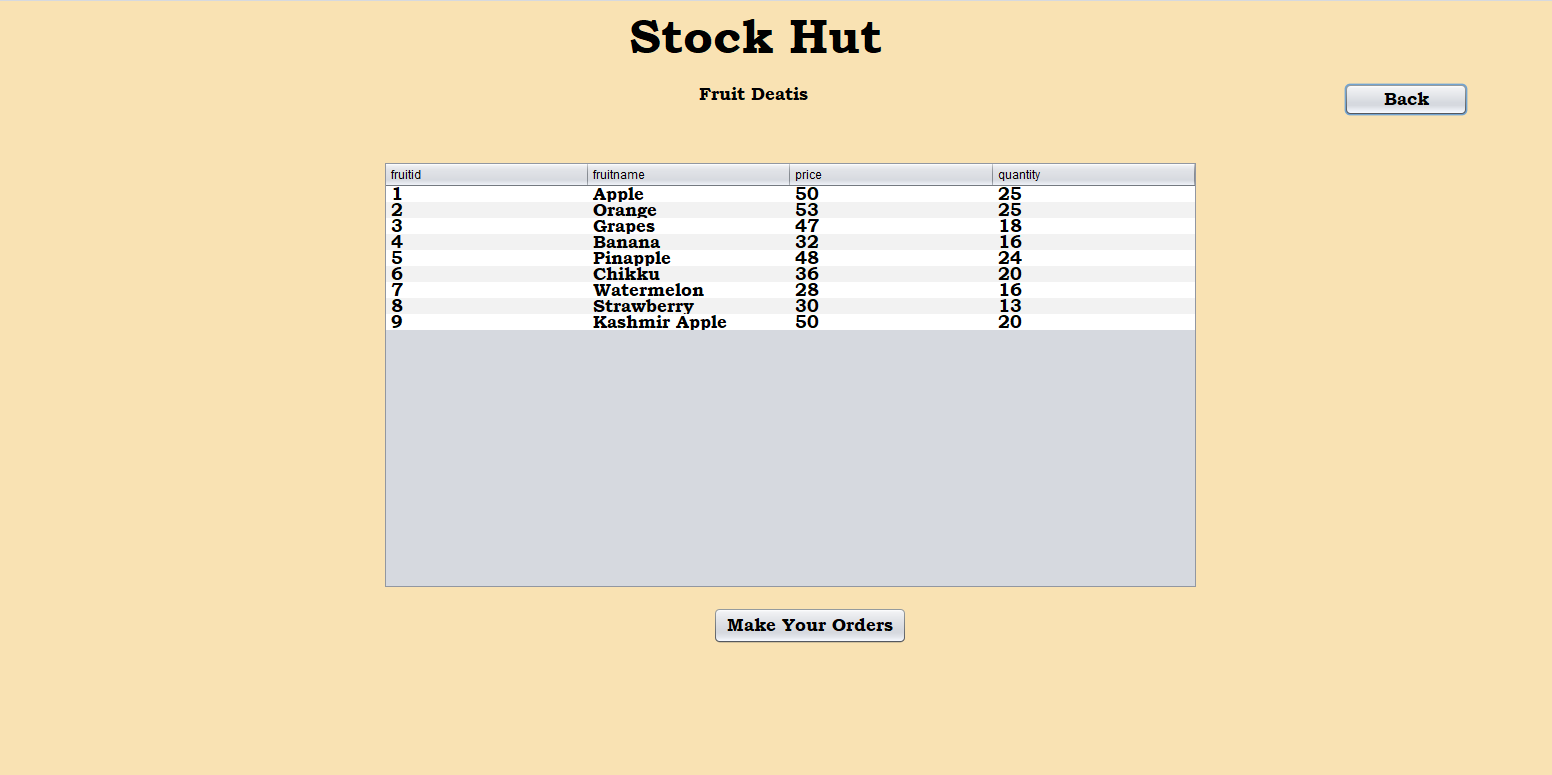


Fig 5.1.8 View Fruits

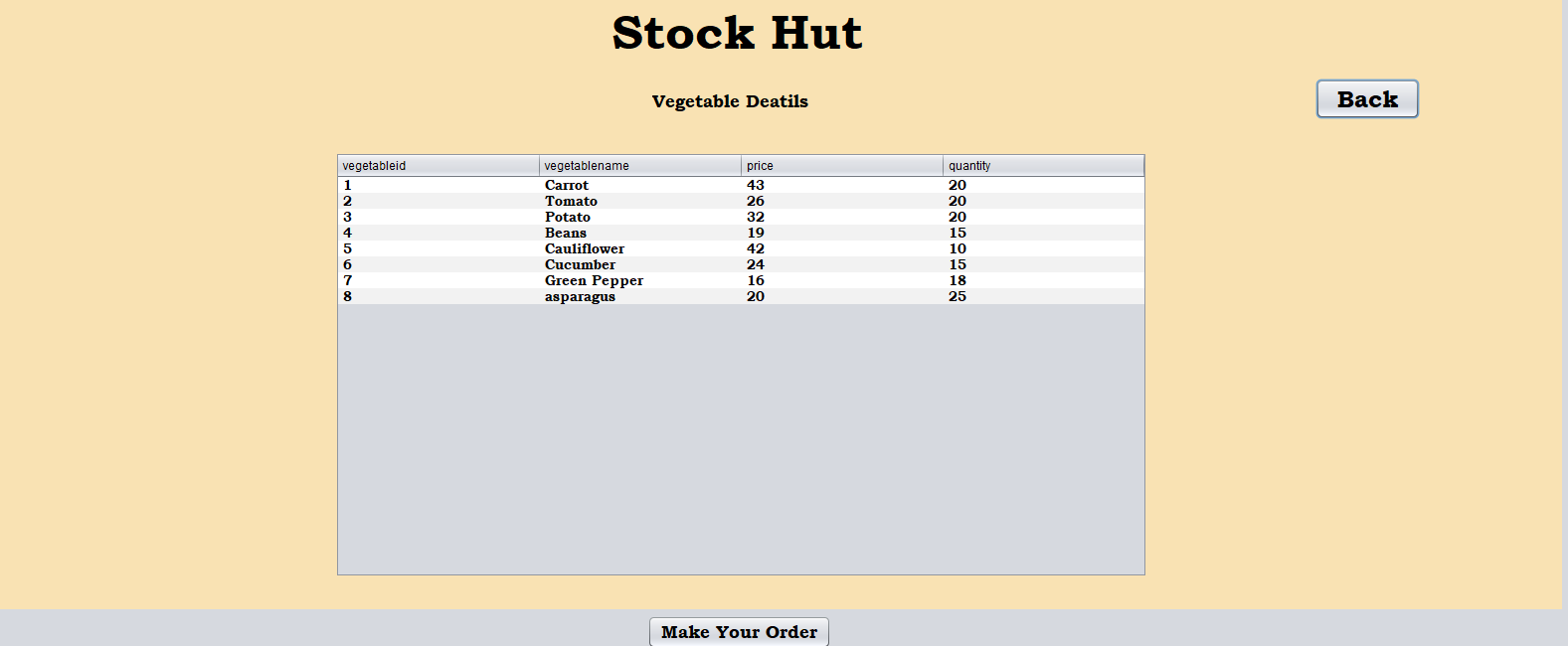
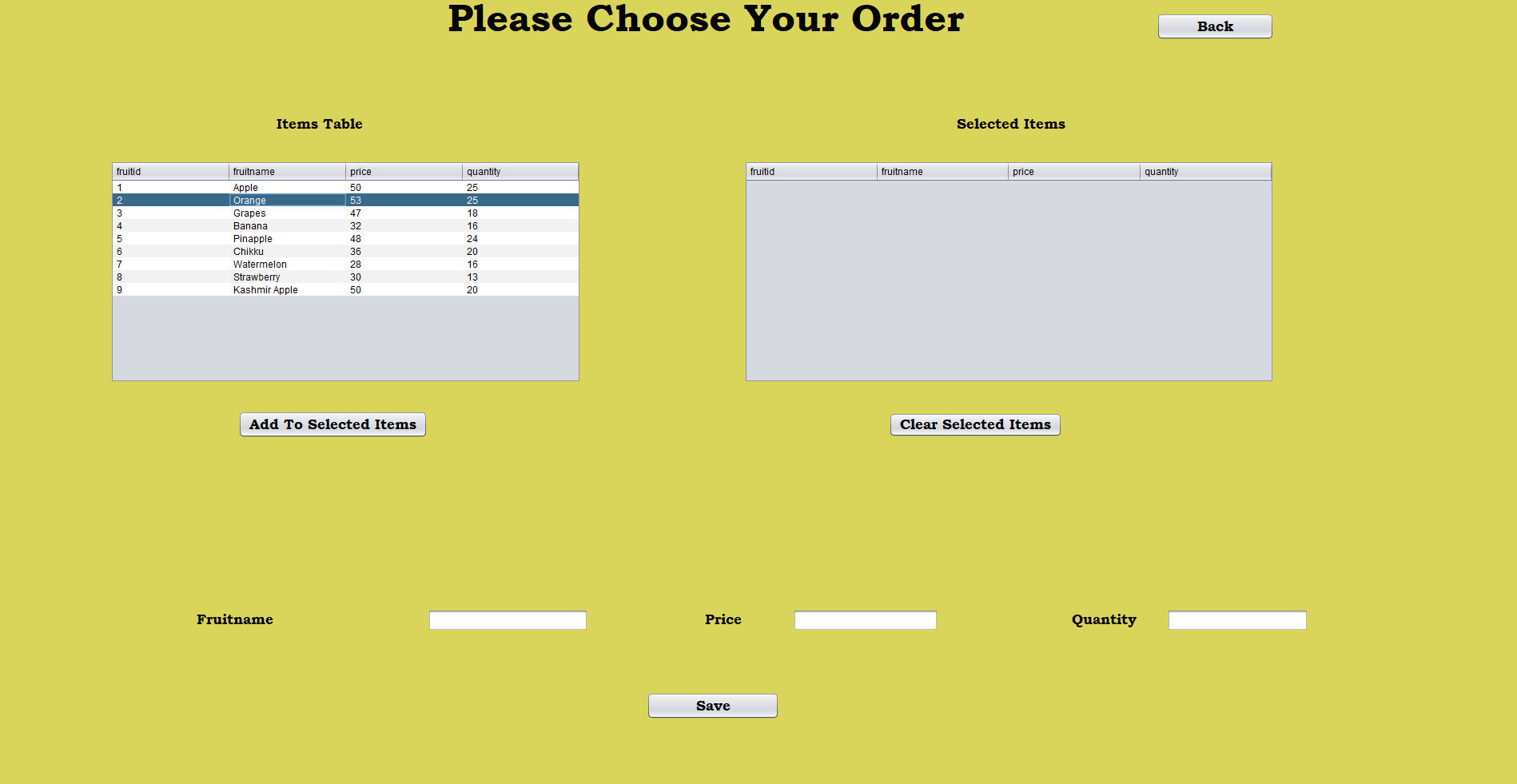


Fig 5.1.8.1 View Vegetables

**5.1.9 CUSTOMER PLACES ORDERS**

The Fig 5.1.9 indicates that the customer can select the available fruits/vegetables and can place his/her order.

**** Fig 5.1.9. Order Window

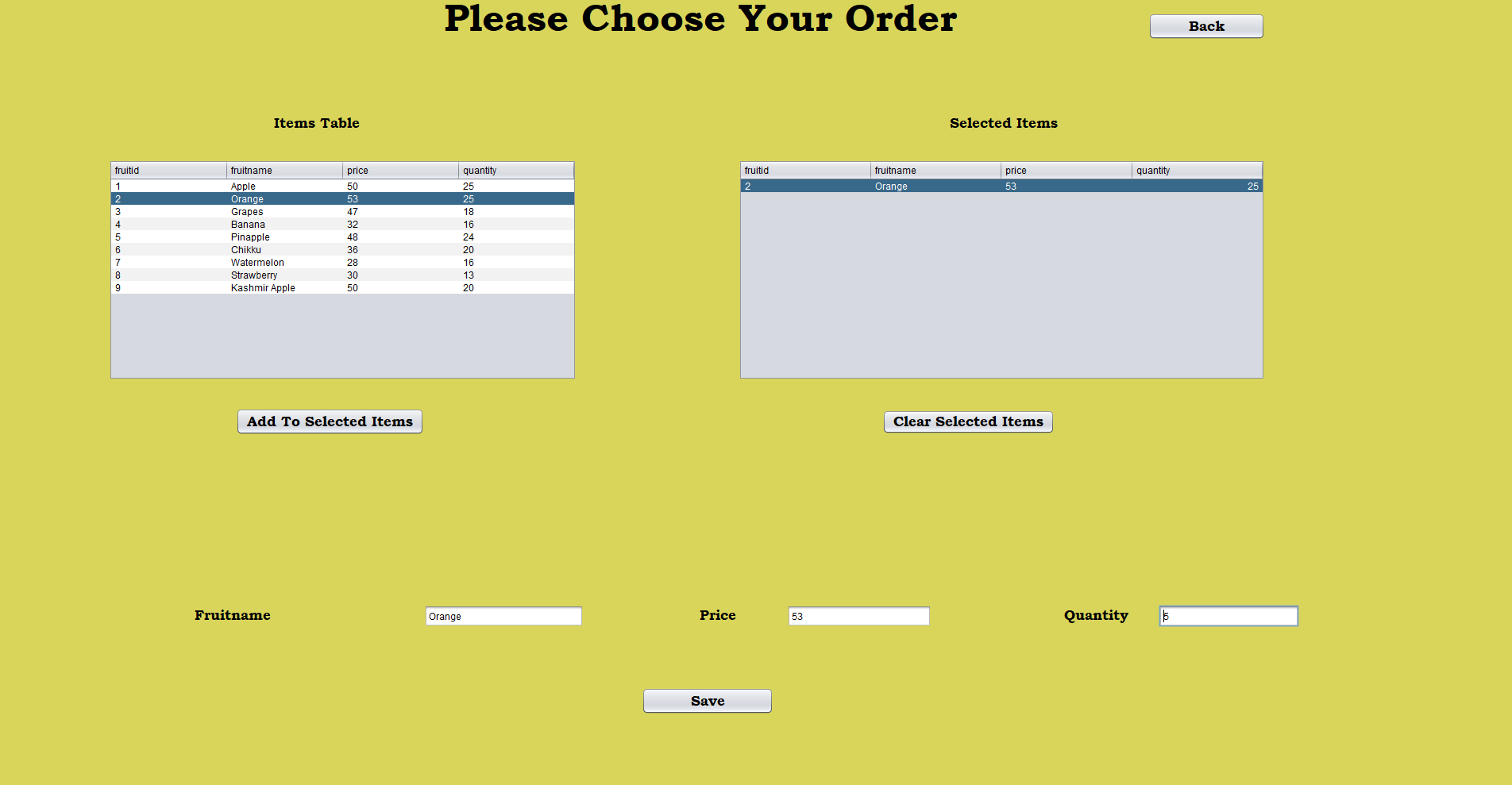


Fig 5.1.9.1 Add to cart



Fig 5.1.9.2 Ordered Item Successfully Placed

**5.1.10 BILLING INFORMATION**

The Fig 5.1.10 represents the bill generated for his/her selected orders.

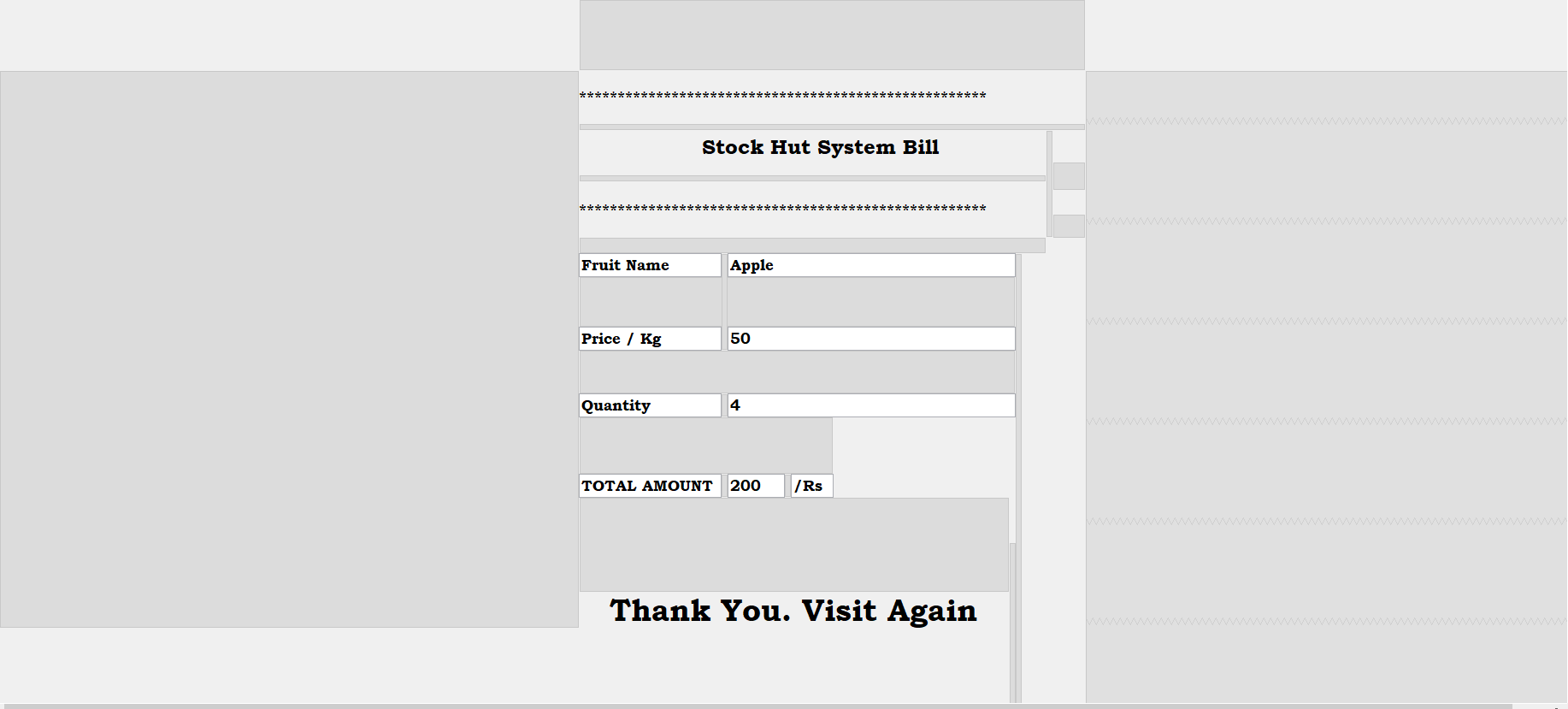


Fig 5.1.10 Customer Bill Generator

**CHAPTER-6**

**CONCLUSION**

The project named as STOCKHUT is an attempt to develop a database system by using java and SQL. We started with modest aim with no prior experience in any programming projects as this, but ended up learning many things, fine tuning the programming skills and getting into the real world of software development. It is extremely user friendly and is convenient for both customers and store owners to

buy fruits and vegetables.

Any future modifications to the project can be made for instance by the during the process of designing and implementing was immense and appealing.

**REFERENCES**

[1] Fundamentals of Database Systems, Rameez Elmar and Shamkant B. Navathe

[2] Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014

[3] Herbert Schildt, JAVA the complete Reference, Tata McGraw Hill,7th edition ,2007

[4] Stack Overflow(<https://stackoverflow.com/>)

[5] YouTube(<https://www.youtube.com/>)

[6] Google(<https://www.google.com/>)