

1.ACKNOWLEDGEMENT

Veggies Box System

Acknowledgement:-

Fifth Semester Project is a major component of Academic Schedule of B.C.A. Hence I worked on **Veggie Box System**. The conceptual Knowledge acquired by Management/ Computer student is best manifested in the project they undergo. The present project gives a perfect vent to my understanding of the practicalities of information of different Educational areas. I express my whole hearted gratitude towards **Sherwood College Of Professional Management** for having given me the opportunity to undergo my project in the field of Software development of great report and allowing me to gain invaluable experience. I express my heart-felt gratitude to **Ms. Namrata Kumari** for supervising me during the project period. I also express my special thanks to all the staff member who gave me their precious time and help me whenever required. I am also grateful to my parents who have always been supportive in giving me correct decision and advice.

I also express my sincere thanks to all the Respondents, without whose help the completion of this project report was not possible

Arpit Shukla

5th Semester

2.Preface

Veggies Box System

Preface-

I am glad to make this project **Veggie Box System** and now I make this project report. In this project report, I include about my project, objective of project, lists of project activities etc.

Doing this project I have a great experience and I get knowledge about boutique system. These all experience and knowledge I am going to share by my project report.

During work on this project I met many technical problems. These technical problems are solved by our Teacher **Ms. Namrata Kumari Ma'am** He is very helpful for me in this project so I am very thankful to her.

Thanking you.

3. Project Report

3.1 INTRODUCTION

3.1.1 BACKGROUND :-

In the fast changing world, information technology and information management are going to play an important role. We are living in the computer age during past some year .The computer has gaining popularity. Computer revolution found its way into almost every aspect of human life and living. A computer is admirably suited to handle any information and hence is an information processor that is, it can receive data, perform some basic operations on that data and produces results according to a predetermined program.

This Software is used mainly for Vegetable stores to maintain the details of vegetables such as stock and account

The Veggie Box System software is so designed as to ease the workload of Vegetable shop professionals. The main feature includes Inventory, and stock control and customer management.

3.1.2 OBJECTIVE :-

Today's world is computer world because most of work is doing with the help of computer. Dependency on computer is behind the few reasons. We cannot easily manage to store large number of data or information single handle. If we will be need some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

As the generic software it can be used by a wide verity of outlets (Wholesalers) to automate the process of manually maintaining records related to the subject of maintain the Stock and Cash flow.

Veggies Box System

This software is basically updating the manual veggie box system to automated Inventory system. So that shop can manage their record in efficient way and organize them.

- The main objective is to automate computer environment
- To save manpower.
- It will speed the processing of data and transaction.
- It will provide best security features such as provisions of passwords

- **System Objective:-**

Today's world is computer world because most of work is doing with the help of computer. Dependency on computer is behind the few reasons. We cannot easily manage to store large number of data or information single handle. If we will be need some information or data in urgency then we cannot manage in manually these works are very difficult if we cannot use computer.

- **System Context:-**

This section clearly depicts the environment and boundaries of the veggie box Inventory System and the entities with which it interacts. It helps us see how the system fits into the existing scheme of things. What the system will do by itself.

Veggies Box System

- **Functional Requirement:-**

This Software must request Username and Password for access to data, after authentication will allow access to the system. The Software must allow input of products data from administrator and secured access.

- **Non-Functional Requirement:-**

In this Software Input error will be returned in red with appropriate message box. System should automatically update after every transaction.

3.1.3 PURPOSE AND SCOPE

- **PURPOSE :-**

The purpose of this document is to specify requirements and to give guidelines for the development of above said project. In particular it gives guidelines on how to prepare the above said project.

This document is intended to be a practical guide for people who developing this software.

- **SCOPE :-**

As this is generic software it can be used by a wide variety of vegetable shop to automate the process of manually maintaining the records related to the subject of maintaining the products details and customer data.

3.2 SURVEY OF

TECHNOLOGIES

3.2.1 JAVA PROGRAMMING LANGUAGE :-

Java is general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems(which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

3.2.2 MY SQL :-

My SQL 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. The My SQL development project has made its source code available under the terms the GNU General Public License, as well as under a variety of proprietary agreements. My SQL was owned and sponsored by a single for-profit firm,the Swedish company MySQLAB,no owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

3.3 REQUIREMENTS

& ANALYSIS

3.3.1 PROBLEM DEFINITION :-

Cinema-going is one of the most popular out-of-home cultural activities, affecting a serious of social, economic and cultural phenomena in modern societies. Cinemas are considered to be an integral part of cities and they contribute to the definition of a local geography and identity. They also contribute to the preservation of the collective memory, since they constitute a significant social and cultural practice linked to a specific place, which acts as a common reference or landmark for many individuals. Through this project we present a comprehensive solution for ticket booking in multiplexes. Theater management system, and ticket selling software that is easy to understand, easy to use and offers the simplicity of fast point-and-click service to the employee and admin. This powerful software program is specifically designed for theater owners, to sell tickets. This intuitive visual interface makes day-to-day aspects of selling, exchanging, refunding, and reporting fast and easy for both the user and administrators. Theater Management controls all back-end and front-end functionalities like, movie details, ticket rate and show time, customer information and sales history saved in a database, etc. Theater admin Manages the report details like counter wise report, daily, weekly, monthly report and movie report etc.

3.3.3 REQUIRMENTS SPECIFICATION :-

- **Software Requirements :-**

- 1 Java/JDK
- 2 NetBeans
- 3 MySql
- 4 SQL YOG

- **Hardware Requirements :-**

- 1 Pentium IV Processor
- 2 512 MB RAM
- 3 40 GB HDD
- 4 Color Monitor
- 5 Keyboard, Mouse

3.3.4 PRELIMINARY PRODUCT DESCRIPTION :-

MODULE DISCRIPTION :-

This system will cover mainly two Modules, i.e.

1. User (EMPLOYEE).
2. Theatre (Admin).

❖ **Theatre Module will cover FOUR Sub modules,**

a. Adding a Show/Deleting a Show/Updating a Show.

Theatre can add a show with timing and price of Ticket. This can be seen and booked by User. Theatre can delete show if it was performed. Theatre can Update show timing and Price.

b. Show Status.

Theatre can change the show status after every show.

c. Updating Screen/Price Status.

Theatre can Update show timing and Price

d. Each and every Customer Detailing.

Theatre can see each and every user details, which is registered (booked a ticket before)

❖ **EMPLOYEE Module will cover THREE Sub modules,**

1. REGISTER CUSTOMER
2. Booking a Ticket. (House full if there is no Ticket)
3. Booking Details. (Booking Status)

3.3.5 CONCEPTUAL MODELS :-

SYSTEM DEVELOPMENT LIFE CYCLE :-

The System development life cycle (SDLC), or Software development processing systems engineering, information systems and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems. In software engineering, the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process.

Broadly, following are the different activities to be considered while defining the system development life cycle for the said project:

- Problem Definition
- System Analysis
- Study of existing system
- Drawback of the existing system
- Proposed system
- System Requirement study
- Data flow analysis
- Feasibility study

Veggies Box System

- System design
- Input Design (Database & Forms)
- Updating
- Query /Report design
- Administration
- Testing
- Implementation
- Maintenance

1.SYSTEM ANALYSIS :-

Systems analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to requirements analysis or operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made.

System development can generally be thought of having two major components: systems analysis and systems design. In System Analysis more emphasis is given to understanding the details of an existing system or a proposed one and then deciding whether the proposed system is desirable or not and whether the existing system needs improvements. Thus, system analysis is the process of investigating a system, identifying problems, and using the information to recommend improvement to the system.

1.2 SYSTEM DESIGN :-

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to Product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user. Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

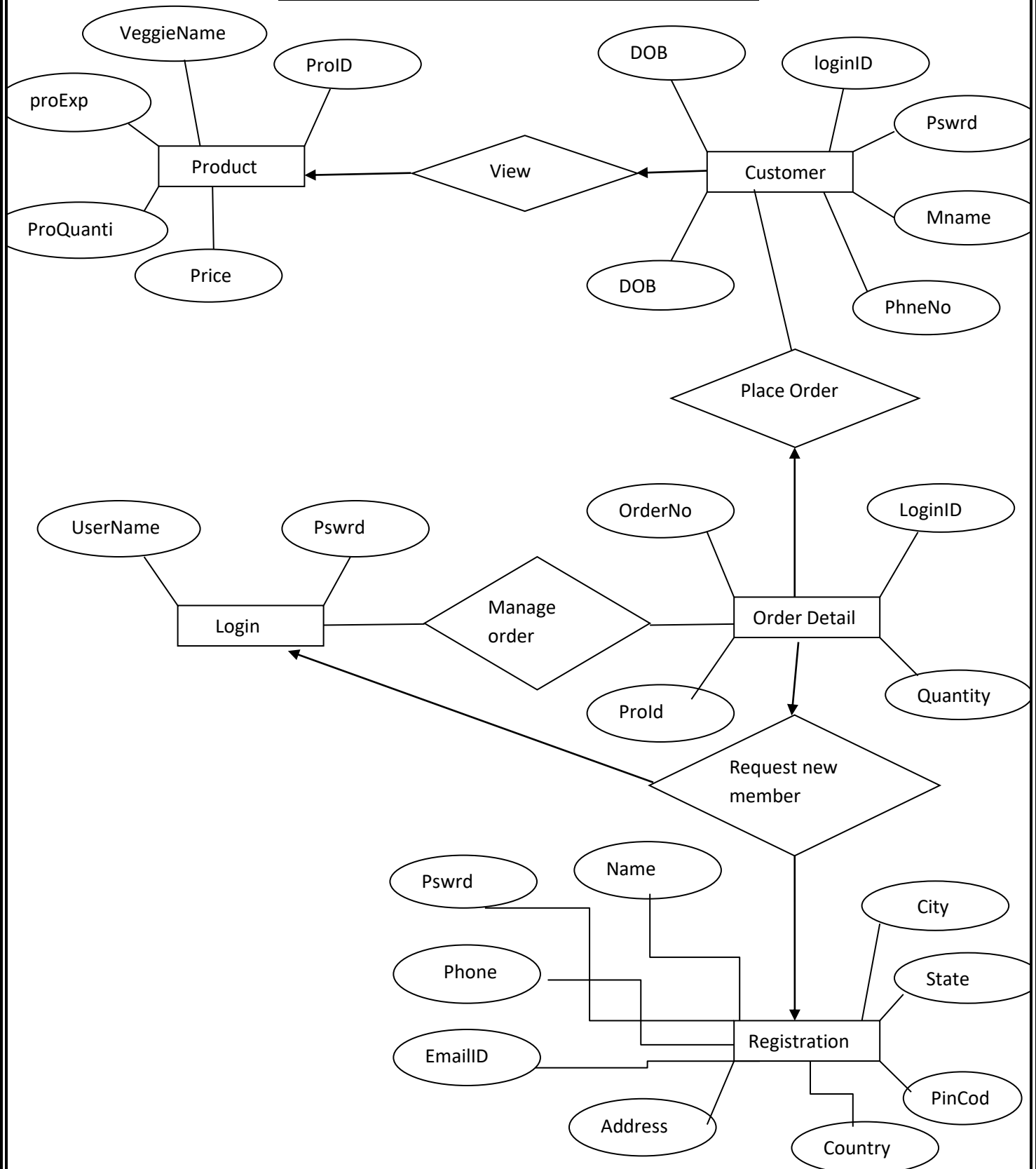
ENTITY RELATION DIAGRAMS :-

The Entity Relation Model or Entity Relation Diagram (ERD) is a data model or diagram for high-level description of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity relationship diagrams. Such models are typically used in the first stage of Management information system design; they are used for example, to describe information needs and/ or the type of information that is to be stored in the Database during the requirement analysis. The data modeling technique, however, can be used to describe any ontology (i.e. an overview and classification of used term and their relationships) for a certain universe of discourse (i.e. area of interest).

In the case of design a Management Information System that is based on a database, the conceptual data model is, a later stage(usually called logical design), mapped to a logical data model such as, relational data model; this is turn in mapped to a physical model during physical design. Note that sometimes, both of the phases are referred a “physical design”. There are number of convention for entity-relation diagrams (ERDs). The classical notation is describe in the remainder of this article, and mainly related to the conceptual modeling. There is a range of notation more typically employed in physical and logical database design.

Veggies Box System

ENTITY RELATIONSHIP DIAGRAM

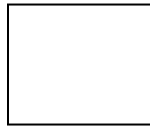


DATA FLOW DIAGRAM :-

The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It's also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD.

In the DFD, four symbols are used and they are as follows.

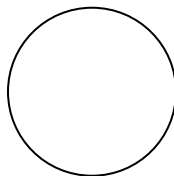
1. A square defines a source (originator) or destination of system data.



2. An arrow identifies data flow-data in motion. It is a pipeline through which information flows.

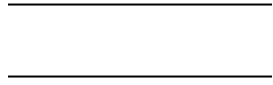


3. A circle or a "bubble" (Some people use an oval bubble) represents a process that transfers incoming data flows into outgoing data flows.



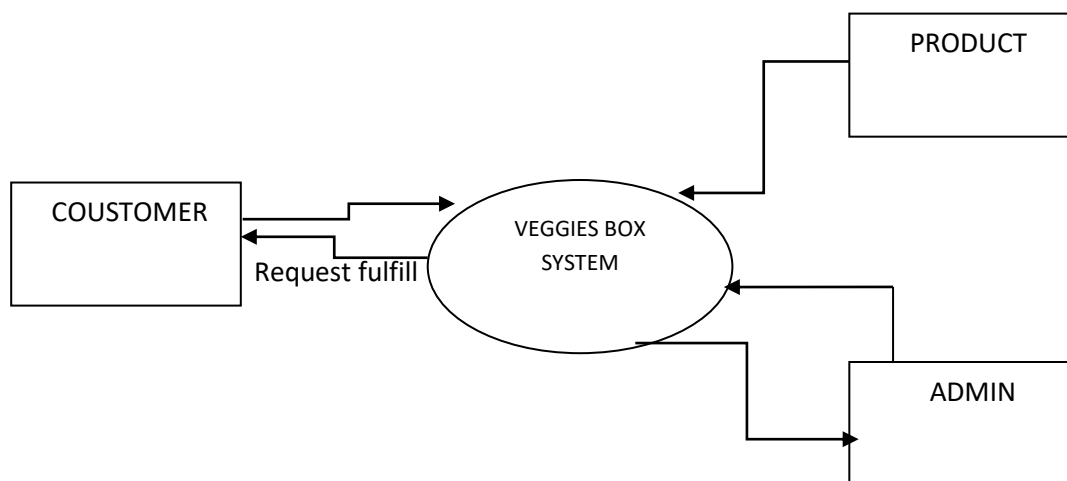
Veggies Box System

4. An open rectangle is a data store-data at rest, or a temporary Repository of data.



Context Level Data Flow Diagram :-

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Online book store is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.

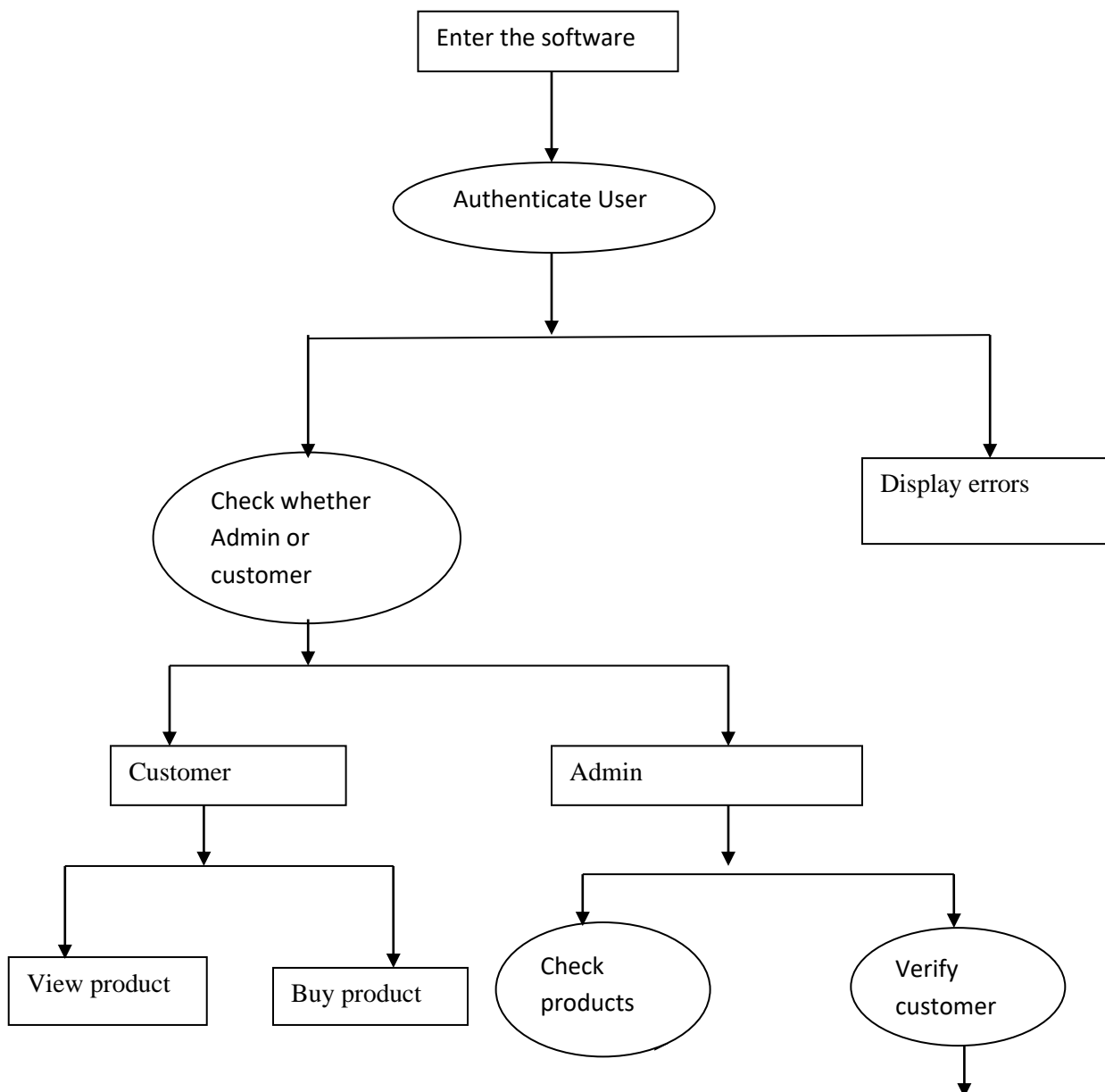


Zero Level Data Flow Diagram

Veggies Box System

First Level DFD :-

This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major high-level processes of the system and their interrelation. A process model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level 1



Veggies Box System

ScreenShot Of Data Table:-

LOGIN:-

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	ZeroFill?	Comment
* type	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
id	varchar	30		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
name	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
contact	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
address	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
age	decimal	30,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
adhaarid	decimal	15,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

CUSTOMER:-

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	ZeroFill?	Comment
* cust_name	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
cust_mail	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
cust_cont	decimal	20,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
cust_addr	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_name	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_qua	decimal	30,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
price	decimal	30,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
total_pri	decimal	30,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

SALE:-

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	ZeroFill?	Comment
* cust_name	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
cust_cont	decimal	20,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
total_pri	decimal	20,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
total_qua	decimal	20,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
date	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Veggies Box System

VEGETABLES:-

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
* pro_id	decimal	20,0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_name	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_qua	decimal	30,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_price	decimal	30,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

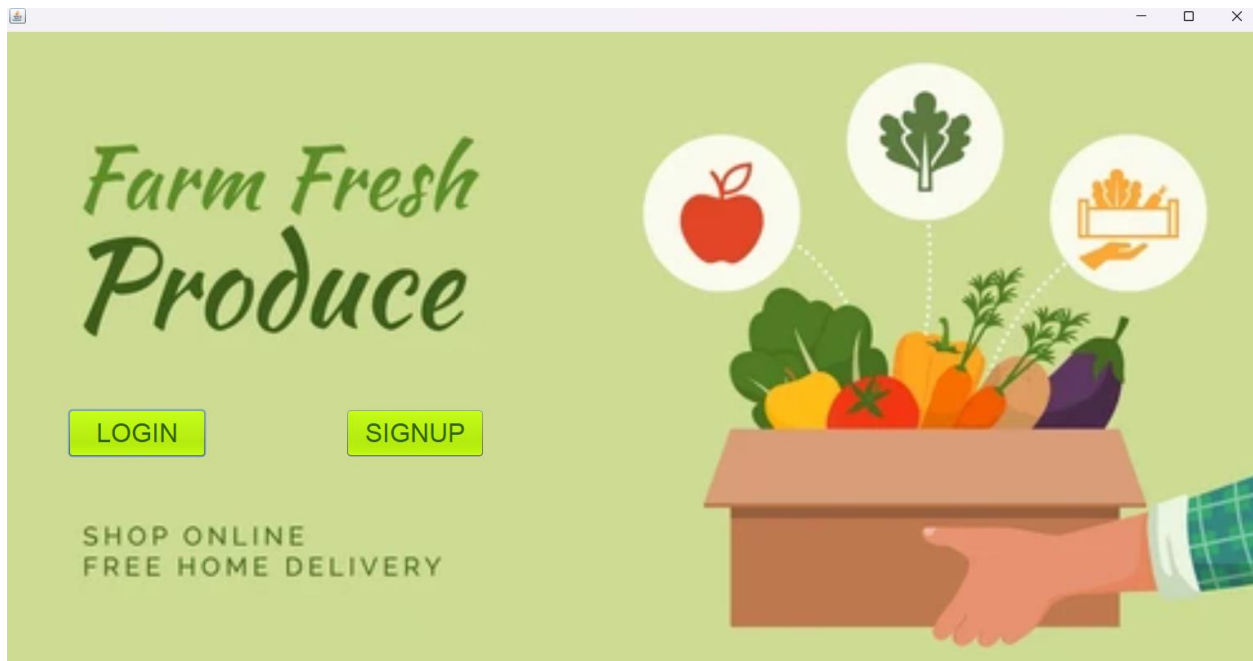
SELERS:-

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
* id	decimal	20,0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
sname	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
addr	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
cont	decimal	20,0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_date	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_name	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_qua	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pro_price	varchar	20		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

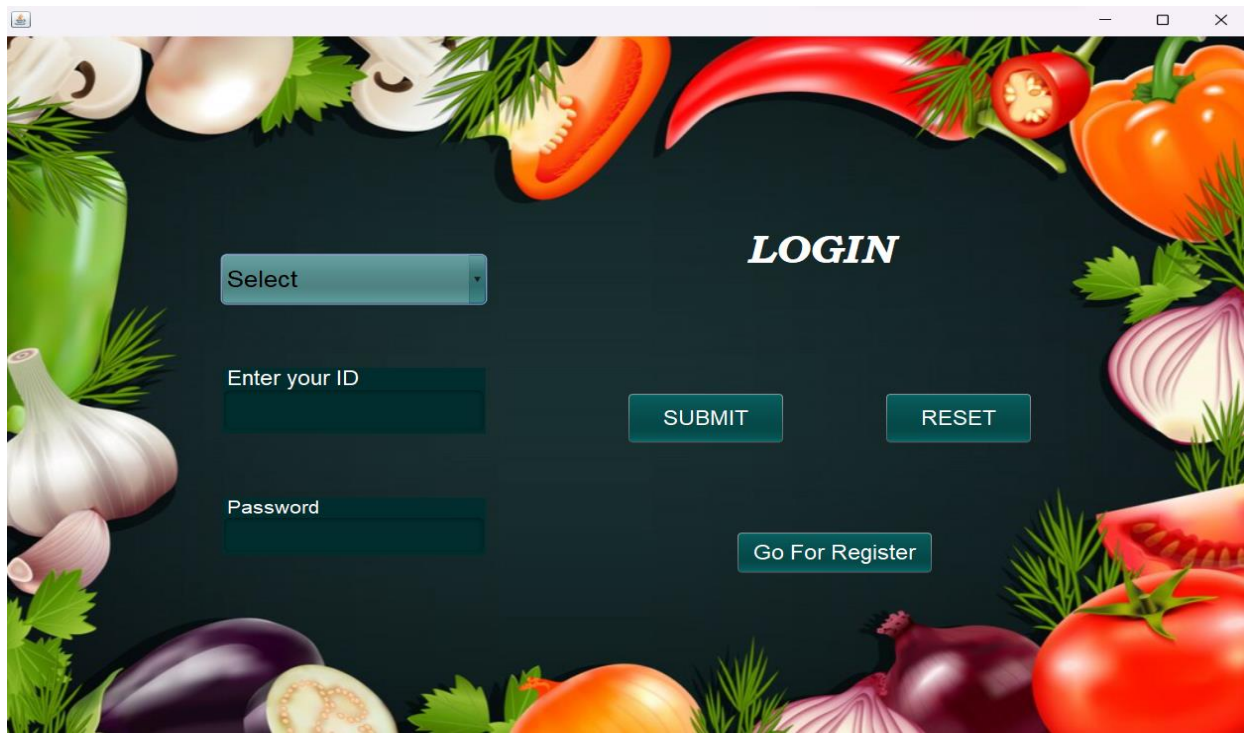
3.5 PROJECT **SCREENSHOT**

Veggies Box System

Main Window:-



Login Window:-



Veggies Box System

Register Window:-

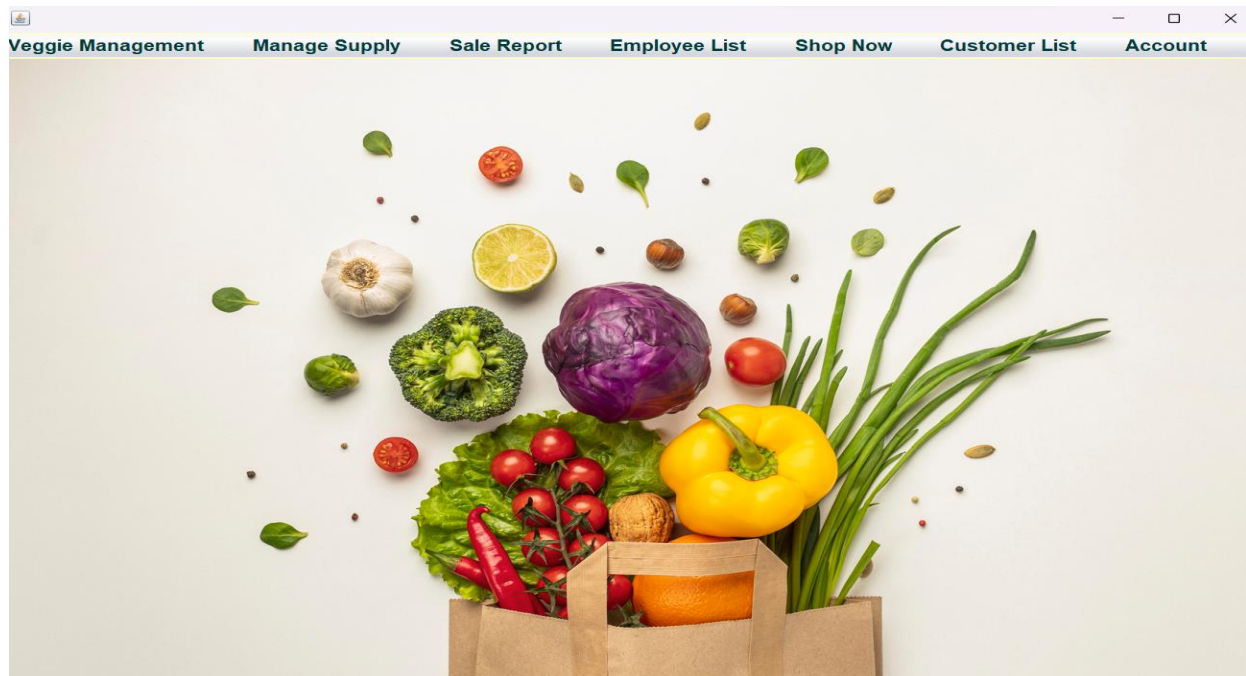


The screenshot shows a web browser window titled "SIGN-UP". On the left is a large purple rectangle with white geometric line patterns. On the right is a registration form with the following fields and labels:

- Enter your ID* (text input)
- Enter your name* (text input)
- Enter your contact* (text input)
- Enter your address* (text input)
- Enter your age* (text input)
- Enter your id proof* (text input)
- Enter Password* (password input)


Below the form are three buttons: "REGISTER", "Go For Login", and "RESET".

Home Window:-



Veggies Box System

Buy Veggie:-



Enter Customer Details

Customer Name

Customer E-mail


Customer Contact

Customer Address

[GO TO SHOP](#)

Add Cart:-

Product ID	Product Name	Product Price	Product Quantity
2	Tomato	100	50
3	Onion	70	80
4	Ginger	70	90
5	Capsicum	40	100
7	carrot	70	110



Product Name

Product Price

Quantity in KG

Total Price

[ADD TO CART](#) [GO TO CART](#)

Veggies Box System

Payment Gateway:-

Select Method Of Payment

☐ QR

☐ CARD

☒ UPI

CONFIRM

UPI
UNIFIED PAYMENTS INTERFACE

Movies Details:-

Total Sale

Customer Name	Customer Contact	Price	Quantity	Date
Yonik	6789765662	4650	52	10-01-2024
Yonik	6789765662	4650	52	10-01-2024
yonik	6789765662	4650	52	10-01-2024
Yonik	6789765662	4650	52	10-01-2024
Yonik	6789765662	4650	52	10-01-2024
arpit	787878877878	630	17	10-01-2024
adarsh	76767676	1880	30	13-02-2024
ram	7897989	980	14	13-02-2024

PRINT

Toatal Sale
26740

Veggies Box System

Add seler:-

WholeSeler detail

Seler ID: 78
Seler Name: eknfw
Seler Address: kwnwkn
Seler Contact: 7979797

WholeSeler detail

Product Date: 12/12/2024
Product Name: ginger
Product Quantity in KG: 100
Product Price per KG: 60

ADD

Manage Customer:-

Customer Name	Customer Email	Customer Contact	Customer Address	product Name	Total Quantity	Product Price
Sam	sam@gmail.com	7878787878	US	Tomato	5	500
Sam	sam@gmail.com	7878787878	US	Onion	7	630
William	will@gmail.com	2345678234	UK	Onion	10	900
William	will@gmail.com	2345678234	UK	Potato	8	560
William	will@gmail.com	2345678234	UK	Tomato	7	700
Ram	rr@gmail.com	1111111111	India	Onion	10	900
Ram	rr@gmail.com	1111111111	India	Tomato	9	900
Idid	diwdi@diw	5678767	iwniwcnlw	Tomato	10	1000
ndwuw	uebcu	878878787	ueuqeu	Onion	10	900
gyft	dfguhugyt	765456787	ijhugyttyg	Tomato	10	1000
lucnwin	jievninc	6776776767	fefec	Onion	10	900
hguyfjn	ftyguhjh	876567876	hjihgg	Tomato	10	1000
hguyfjn	ftyguhjh	876567876	hjihgg	Potato	20	1400
hguyfjn	ftyguhjh	876567876	hjihgg	Onion	7	630
sjiccn	jelfwic	87898789	wifhwifj	Onion	10	900
sjiccn	jelfwic	87898789	wifhwifj	Potato	8	560
arplt	arplt@gmail.com	787878877878	lucknow	Onion	9	630
adarsh	adarsh12	76767676	Lucknow	Onion	9	630
adarsh	adarsh12	76767676	Lucknow	Capsicum	5	200
adarsh	adarsh12	76767676	Lucknow	carrot	15	1050
ram	ram12@3	7897989	UK	carrot	10	700
ram	ram12@3	7897989	UK	Onion	4	280

Customer Name: ndwuw
Customer E-mail: uebcu
Customer Contact: 878878787
Customer Address: ueuqeu

DELETE **RESET**

3.4 PROJECT CODE

Login:-

Veggies Box System

```
try{

    String type = jComboBox1.getSelectedItem().toString();

    DB.DBcon db = new DB.DBcon();

    db.pstmt = db.con.prepareStatement("select * from login where type=? AND id=? AND
password=?");

    db.pstmt.setString(1, type);

    db.pstmt.setString(2, jTextField1.getText());

    db.pstmt.setString(3, jPasswordField1.getText());

    db.rst=db.pstmt.executeQuery();

    if(db.rst.next())

    {

        switch (type) {

            case "Admin":

                this.setVisible(false);

                new HomePage().setVisible(true);

                break;

            case "Employee":

                this.setVisible(false);

                new HomePage().setVisible(true);

                break;

            default:
```


Veggies Box System

```
        JOptionPane.showMessageDialog(this,"Incorrect item selected");

        break;

    }

}

else

{

    JOptionPane.showMessageDialog(this,"Incorrect User Name or Password");

}

}

catch(Exception e)

{

    e.printStackTrace();

}
```

Registration:-

```
try{

    jLabel3.setText("Employee");

    DB.DBcon db = new DB.DBcon();

    db.pstmt          =          db.con.prepareStatement("insert          into
login(id,name,contact,address,age,adhaarid,password,type)values(?,?,?,?,?,?,?,?)");

    db.pstmt.setString(1, jTextField2.getText());

    db.pstmt.setString(2, jTextField3.getText());

    db.pstmt.setString(3, jTextField4.getText());

    db.pstmt.setString(4, jTextField5.getText());

    db.pstmt.setString(5, jTextField6.getText());
```

Veggies Box System

```
db.pstmt.setString(6, jTextField7.getText());

db.pstmt.setString(7, jPasswordField1.getText());

db.pstmt.setString(8, jLabel3.getText());


int i = db.pstmt.executeUpdate();

if(i>0)

{

    JOptionPane.showMessageDialog(this,"Data Registered Successfully");

    new Login().setVisible(true);

    this.setVisible(false);

}

}

catch(Exception e)

{

    e.printStackTrace();

}

}
```

Add Veggie:-

```
try{

    DB.DBcon db = new DB.DBcon();

    db.pstmt = db.con.prepareStatement("insert into veggie(pro_id,pro_name,pro_qua,pro_price)
values(?,?,?,?)");

    db.pstmt.setString(1, jTextField1.getText());
```

Veggies Box System

```
db.pstmt.setString(2, jTextField2.getText());

db.pstmt.setString(3, jTextField3.getText());

db.pstmt.setString(4, jTextField4.getText());

int i = db.pstmt.executeUpdate();

if(i>0)

{

    JOptionPane.showMessageDialog(this,"Vegetabel Added Succesfully");

    jTextField1.setText("");

    jTextField2.setText("");

    jTextField3.setText("");

    jTextField4.setText("");

}

}

catch(Exception e)

{

    e.printStackTrace();

}
```

UpdateVeggie:-

// Update veggi code:

```
try{

    DB.DBcon db = new DB.DBcon();

    db.pstmt = db.con.prepareStatement("update veggie set pro_name=?,pro_qua=?,pro_price=?
where pro_id=?");
```

Veggies Box System

```
db.pstmt.setString(1, jTextField6.getText());

db.pstmt.setString(2, jTextField7.getText());

db.pstmt.setString(3, jTextField8.getText());

db.pstmt.setString(4, jTextField5.getText());


int i = db.pstmt.executeUpdate();

if(i>0)
{
    JOptionPane.showMessageDialog(this,"Veggie Updated Succesfully");

    jTextField5.setText("");

    jTextField6.setText("");

    jTextField7.setText("");

    jTextField8.setText("");


    Vector<Object>header=new Vector<Object>();

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

    header.add("Product ID");

    header.add("Product Name");

    header.add("Product Quantity");

    header.add("Product Price");


    db.pstmt=db.con.prepareStatement("select * from veggie");

    db.rst=db.pstmt.executeQuery();
```

Veggies Box System

```
while(db.rst.next())
{
    Vector<Object>temp=new Vector<Object>();
    temp.add(db.rst.getString(1));
    temp.add(db.rst.getString(2));
    temp.add(db.rst.getString(3));
    temp.add(db.rst.getString(4));
    data.add(temp);
}
jTable1.setModel(new DefaultTableModel(data,header));
}
}
catch(Exception e)
{
    e.printStackTrace();
}
```

DeleteVeggie:-

```
try{
    DB.DBcon db = new DB.DBcon();
    db.pstmt = db.con.prepareStatement("delete from veggie where pro_id=?");
    db.pstmt.setString(1, jLabel5.getText());
    int i = db.pstmt.executeUpdate();
    if(i>0)
    {
        JOptionPane.showMessageDialog(this,"product deleted successfully");
        jTextField5.setText("");
    }
}
```

Veggies Box System

```
        jTextField6.setText("");  
  
        jTextField7.setText("");  
  
        jTextField8.setText("");  
  
    }  
}  
catch(Exception e)  
{  
    e.printStackTrace();  
  
}
```

SearchVeggie:-

```
try{  
  
    Vector<Object>header=new Vector<Object>();  
  
    Vector<Vector<Object>>data=new Vector<Vector<Object>>();  
  
    header.add("Product ID");  
  
    header.add("Product Name");  
  
    header.add("Product Quantity");  
  
    header.add("Product Price");  
  
  
    DB.DBcon db = new DB.DBcon();  
  
    db.pstmt = db.con.prepareStatement("select * from veggie where pro_id=? OR pro_name=? OR  
pro_qua=? OR pro_price=?");  
  
    db.pstmt.setString(1, jTextField10.getText());  
  
    db.pstmt.setString(2, jTextField10.getText());  
  
    db.pstmt.setString(3, jTextField10.getText());  
  
    db.pstmt.setString(4, jTextField10.getText());
```

Veggies Box System

```
db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

    Vector<Object>temp=new Vector<Object>();

    temp.add(db.rst.getString(1));

    temp.add(db.rst.getString(2));

    temp.add(db.rst.getString(3));

    temp.add(db.rst.getString(4));

    data.add(temp);

}

jTable2.setModel(new DefaultTableModel(data,header))

}

catch(Exception e)

{

    e.printStackTrace();

}
```

View Veggie:-

```
try{

    Vector<Object>header=new Vector<Object>();

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

    header.add("Product ID");

    header.add("Product Name");

    header.add("Product Quantity");

    header.add("Product Price");
```

Veggies Box System

```
DB.DBcon db=new DB.DBcon();

db.pstmt=db.con.prepareStatement("select * from veggie");

db.rst=db.pstmt.executeQuery();

while(db.rst.next())

{

    Vector<Object>temp=new Vector<Object>();

    temp.add(db.rst.getString(1));

    temp.add(db.rst.getString(2));

    temp.add(db.rst.getString(3));

    temp.add(db.rst.getString(4));

    data.add(temp);

}

jTable1.setModel(new DefaultTableModel(data,header));

}

catch(Exception e)

{

    e.printStackTrace();

}
```

Buy Veggie:-

```
try{
```

```
    DB.DBcon db = new DB.DBcon();

    db.pstmt          =          db.con.prepareStatement("insert          into
customer(cust_name,cust_mail,cust_cont,cust_addr,pro_name,pro_qua,price) values(?,?,?,?,?,?,?)");
```


Veggies Box System

```
db.pstmt.setString(1, jTextField27.getText());
db.pstmt.setString(2, jTextField28.getText());
db.pstmt.setString(3, jTextField29.getText());
db.pstmt.setString(4, jTextField30.getText());
db.pstmt.setString(5, jLabel17.getText());
db.pstmt.setString(6, jSpinner1.getValue().toString());
db.pstmt.setString(7, jLabel19.getText());

int i = db.pstmt.executeUpdate();

if(i>0)
{
    JOptionPane.showMessageDialog(this," Added to Cart");

    jLabel17.setText("");
    jLabel18.setText("");
    jLabel19.setText("");
}
}

catch(Exception e)
{
    e.printStackTrace();
}
```

3.6 TESTING

Testing:-

System Testing: -

Black box testing method was used for system testing. The black box testing usually demonstrates that software functions are operational; that the input is properly accepted and the output is correctly produced; and that integrity of external information (databases) is maintained.

Why testing is done:-

- Testing is the process of running a system with the intention of finding errors.
- Testing enhances the integrity of a system by detecting deviations in design and errors in the system.
- Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system.
- Testing also add value to the product by confirming to the user requirements.

Causes of Errors:-

The most common causes of errors in a software system are:

- **Communication gap between the developer and the business decisionmaker:** A communication gap between the developer and the business decision maker is normally due to subtle differences between them. The differences can be classified into five broad areas: Thought process, Background and Experience, Interest, Priorities, Language.

Veggies Box System

- **Time provided to a developer to complete the project:** A common source of errors in projects comes from time constraints in delivering a product. To keep to the schedule, features can be cut. To keep the features, the schedule can be slipped. Failing to adjust the feature set or schedule when problems are discovered can lead to rushed work and flawed systems.
- **Over Commitment by the developer:** High enthusiasm can lead to over commitment by the developer. In these situations, developers are usually unable to adhere to deadlines or quality due to lack of resources or required skills on the team.
- **Insufficient testing and quality control:** Insufficient testing is also a major source of breakdown of e-commerce systems during operations, as testing must be done during all phases of development.
- **Inadequate requirements gathering:** A short time to market results in developers starting work on the Web site development without truly understanding the business and technical requirements. Also, developers may create client-side scripts using language that may not work on some client browsers.
- **Keeping pace with the fast changing Technology:** New technologies are constantly introduced. There may not be adequate time to develop expertise in the new technologies. This is a problem for two reasons. First, the technology may not be properly implemented. Second, the technology may not integrate well with the existing environment.

Testing Principles:-

- To discover as yet undiscovered errors.
- All tests should be traceable to customer's requirement.
- Tests should be planned long before the testing actually begins.
- Testing should begin "in the small" & progress towards "testing in the large".
- Exhaustive Testing is not possible.
- To be most effective training should be conducted by an Independent Third Party

Testing Objectives:-

- Testing is a process of executing a program with the intent of finding errors.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

Kinds of Testing:-

Black Box Testing- Not based on any knowledge of internal designs or code. Tests are based on requirements and functionality.

White Box Testing- Based on the knowledge of the internal logic of an application's code. Tests are based on coverage of code statements, branches, paths and statements.

Unit Testing- The most 'micro' scale of testing; to test particular functions and code modules. Typically done by the programmer and not by the testers, as it

Veggies Box System

requires detailed knowledge of the internal program design and code. Not always easily done unless the application has a well-designed architecture with tight code; may require developing test driver modules or test harnesses.

Integration Testing- Testing of combined parts of an application to determine if they function together correctly. The ‘parts’ can be code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/ server and distributed systems.

Functional Testing- Black-box type testing geared to functional requirements of an application; testers should do this type of testing. This doesn’t mean that the programmers shouldn’t check that their code works before releasing it.

Regression Testing- Re-testing after fixes or modifications of the software or its environment. It is difficult to determine how much re testing is needed, especially near the end of the development cycle. Automated testing tools can be especially useful for this type of testing.

Acceptance Testing- Final testing based on the specifications of the end user or customer or based on use by end-users/ customers over some limited period of time.

Veggies Box System

User Acceptance Testing- Determining if software is satisfactory to an end user customer

Testing Technique Used:-

We will continuously test our project to insure that it is fully functional. In order to perform testing test cases are designed with the intent of finding the errors in the project and help in removing those errors. Testing begins at the module level and is conducted systematically. It is generally conducted by independent test groups or third party.

Testing is done in our project Multiplex Ticket Booking System with help of black box testing that exercise all the functional requirement of the project test cases are designed using this approaches by providing set of input conditions to get the expected output.

Test Cases Designed

TEST CASES FOR CLIENT:-

TEST CASE ID	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT
1	Password and key combination	Start	Operation started
2	No Input	Alert Message	Select a valid configuration setting

TEST CASES FOR SERVER:-

TEST CASE ID	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT
1	127.0.0.1	Connection establish	Client Connected
2	Get Status	Alert Box	If no client selected "alert box" is displayed
3	Choose Path	Transfer files to valid path	Files transferred

3.7 IMPLEMENTATION

Implementation:-

System Implementation:-

During the implementation stage the system is physically created. Necessary programs are coded, debugged and documented. A new hardware is selected, ordered and installed.

System Specification:-

Every computer system consists of three major elements.

1. The Hardware
2. Application software such as visual studio
3. Operating system

For successful operation of the package following must be kept in mind:

Too many packages should not be used, as very few systems may have all those packages installed due to memory problem. Thus, the compatibility of the system development will get reduced.

3.8 FUTURE SCOPE **OF PROJECT**

Future Scop Of Project:-

The proposed system helps them in many ways. It helps them do billing very easily. Account maintenance also becomes easier. They can keep track of their all movie details and customer account details. The software is provided with all the master entries to enter any new movies or to add or modify and delete the movie shows and timings.

As this is generic software it can be used by a wide variety of multiplex cinema halls to automate the process of manually maintaining the records related to the subject of maintaining the movie details and customer data.

In future it can be modify, so that it can be done online.

3.9 CONCLUSION

Conclusion:-

This section discusses the result of work done in this project and also mentions the future scope improvement.

The software will be developed by implementing the concept of modularity which in turn reduces the complexity involved in maintaining it. The administrator should have a sound technical knowledge about maintaining the software and further enhancements will be undertaken by the developer.

The application is portable which ensure its adaptability for use on different computer terminals with different operating system and standards.

The factors guarantee the software's availability includes proper termination and correct input details. Also the resources used for the project development are Microsoft certified which speaks of its high quality standards.

Hence we may conclude that the application system being developed helps a great deal in modifying the computerized VEGGIES BOX SYSTEM.

4. BIBLIOGRAPHY

Book References:-

The following books were referred during the analysis and execution phase of the project

Common Language Runtime

-By Steven Pratschner

SOFTWARE ENGINEERING

-By Roger S. Pressman

UNIFIED MODELING LANGUAGE

-By Gradi Booch, Ivar Jacobson,
James Rambaugh

COMPLETE REFERENCE .NET

-By David S Platt

MSDN 2003

-By Microsoft

IMAGES

Bing Search

Msn Search

Google Search

HTML PUBLISHING BIBL

- Alan Simpson.

C# 2008

-Andrew Troelson

Designing and implementation phase: -

Veggies Box System

1. Software engineering: a practitioners approach by roger s pressman.
2. System analysis and design by Elias m. Ewad.
3. DBMS : Bipin C Desai
4. Introduction to dbms – shailendrasharma
5. Modern database management -drkumar
6. Fundamentals of database systems- ramezelmsari
7. An intro to dbms- christhopher j date
8. Database system concepts-abrahamsilbersch
9. Database management systems-raghuramkrishnan
10. Principles of database systems- jd Ullman
11. Foundation of databases- sergebabiteboul
- 12.Database systems-connolly

WEBSITES:-

- www.google.com
- www.netbeans.com
- www.sql.com
- www.wikipedia.com
- www.ask.com
- www.sitebay.com
- <https://www.freeprojectz.com/dfd/online-movie-ticket-booking-system-dataflow-diagram>
- <https://www.slideshare.net/mrinovater007/online-movie-ticket-booking-11583340>
- www.w3school.com
- www.projectcode.com
- www.projectwork.com