CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO SQL

Structure Query Language (SQL) is a programming language used for storing and managing data in Relational Database Management System (RDBMS). SQL was the first commercial language introduced for E.F Codd's Relational model. Today almost all RDBMS (MySQL, Oracle, Infomix, Sybase, MS Access) uses SQL as the standard database language. SQL is used to perform all type of data operations in RDBMS. Most of the actions you need to perform on a database are done with SQL statements. SQL defines following data languages to manipulate data of RDBMS:

1. DDL: Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Eg: create-To create new table or database, alter-For alteration, truncate-Delete data from table, drop-To drop a table

2. DML: Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Eg : insert-To insert a new row, update-To update existing row, delete-To delete a row, merge-merging two rows or two tables

3. TCL: Transaction Control Language

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.

Eg : commit-to permanently save, rollback-to undo change, save point-to save temporarily

4. DCL: Data Control Language

Data control language provides command to grant and take back authority.

Eg : grant-grant permission of right, revoke-take back permission

5. DQL-Data Query Language

DQL is used to operate on queries.

Eg : Select-retrieve records from one or more table

1.2 INTRODUCTION TO FRONT END SOFTWARE

The front end software used is PHP.PHP is an acronym for "PHP: Hypertext Preprocessor”. PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. It is a widely used, open source scripting language. It is free to download and use. PHP files can contain text, HTML, CSS, JavaScript, and PHP code. PHP code are executed on the server, and the result is returned to the browser as plain HTML.PHP files have extension ".php".

PHP code may be embedded into HTML or HTML5 makeup, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical application

1.3 PROJECT REPORT OUTLINE

The report is arranged in the following way:

Chapter 1: Gives the Introduction of the project
Chapter 2: Gives the Requirement Specification of the project
Chapter 3: Gives the Objective of the project
Chapter 4: Gives the Implementation of the project
Chapter 5: Gives the Front end design of the project
Chapter 6: Gives the Testing information of the project
Chapter 7: Gives the Results of the project

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

Operating System : Windows 10

Database : MYSQL

Tools : XAMPP SERVER 2.1

2.2 HARDWARE REQUIREMENTS

Processor : Any Processor above 500 MHz

RAM : 16GB

Hard Disk : 1TB+128 GB SSD

Compact Disk : CD-ROM, CD-R, CD-RW

Input device : Keyboard , Mouse

Output device : Monitor

CHAPTER 3

OBJECTIVE OF THE PROJECT

The project Grocery Database helps to store the Information about groceries and as well to search the details about the customer by having customer id. It helps to store or maintain all the information about all the information about the staff. It helps to Create Bill for the customer.

It is designed to achieve the following objectives:

1. To computerize all details regarding customer details and grocery details.
2. New customers can be registered so that they can purchase and be a part of grocery store and gain points according to their purchase.
3. This project also provides a complete set of solutions for some common and specific work in the grocery store.
4. A bill is generated during the time of customer check out.
5. Details can be inserted, retrieved, deleted.

CHAPTER 4

IMPLEMENTATION

4.1 ER DIAGRAM

An entity-relationship model (ER model) describes inter-related things of Interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model be implemented in a database, typically a relational database.

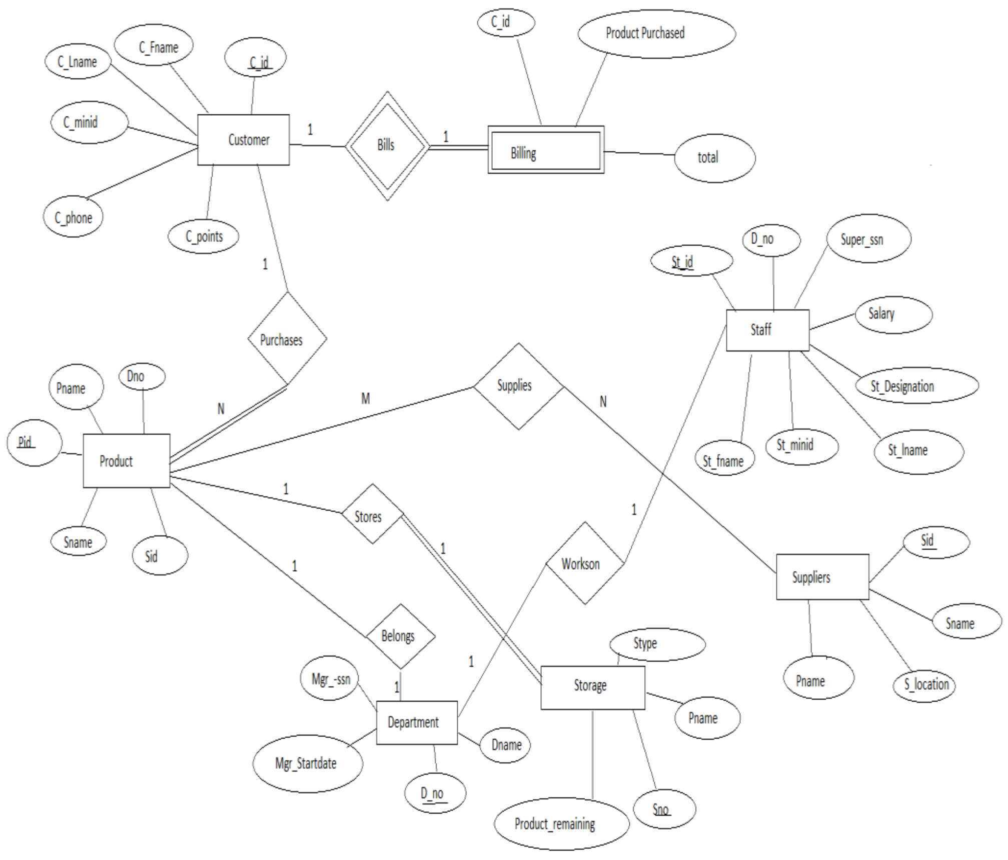


Figure 4.1:ER Diagram

The above ER diagram represents the Grocery database with entities like Customer, Product, Department, Suppliers, Staff, Storage. The above ER diagram shows the Relation between the entities. The Grocery database has many types of grocery Products which are purchased by Customers. The Products are supplied by Suppliers. The Products are stored in different types of Storage mediums depending on their life-cycle. Finally when the Customer checks out, the Billing system produces their respective bill as per their purchase of Products

4.2 MAPPING OF ER DIAGRAM TO SCHEMA DIAGRAM

An entity exits in the real world project with a relationship shows how two entities relate to each other. Both entities and relationships can have attributes. Anything that is useful to know about an entity or a relationship is an attribute. Since the database must be able to find an entity or a relationship, it must be able to identify it with a set of attributes that is unique.

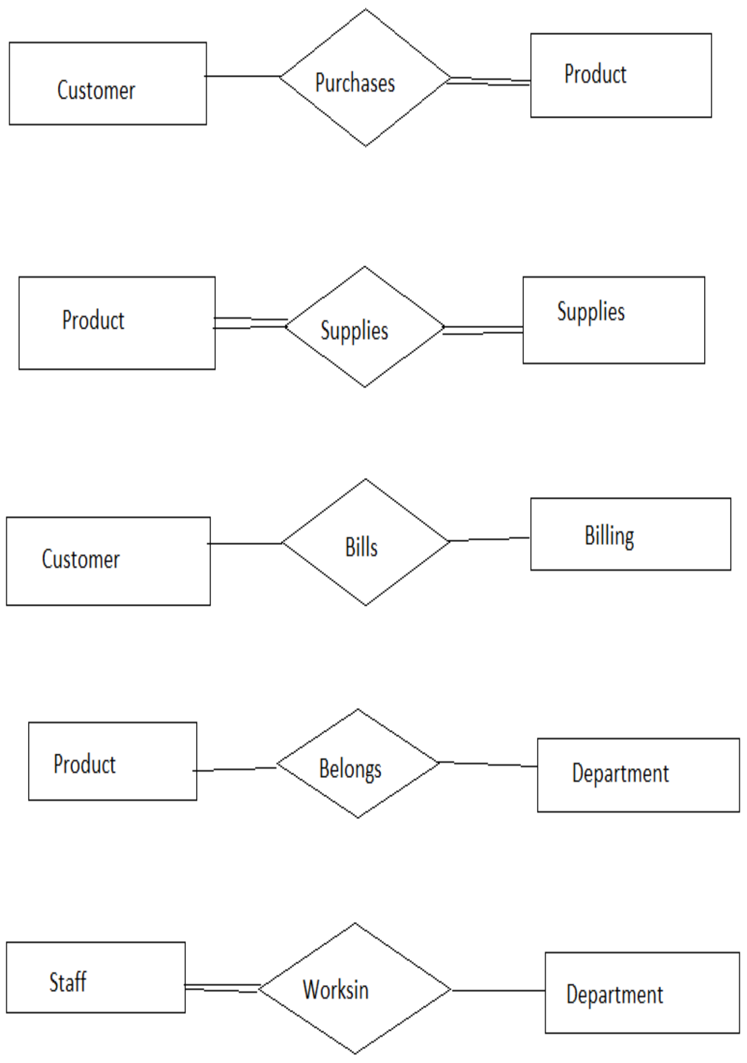


Figure 4.2: Mapping of ER Diagram to Schema

4.3 MAPPING OF THE ER SCHEMA TO RELATIONS

Step 1: Mapping of regular entity types

Identify the relation S and T that correspond to the entity types participating in

R. Choose one of the relations, say S, and include as foreign key in S the primary key of T. Include the simple attributes of the 1:1 relationship type R as attributes of S.

Customer

<u>Cid</u>	**C_Fname**	**C_minid**	**C_Lname**	**C_phone**	**C_points**

Product

<u>Pid</u>	**P_name**	**S_name**	**Sid**	**D_no**

Department

<u>D_no</u>	**D_name**	**Mgr_ssn**	**Mgr_startdate**

Storage

<u>S_no</u>	**S_type**	**Pid**	**Product_remaining**

Suppliers

<u>S_id</u>	**S_name**	**P_name**	**S_location**	**S_phone**

Staff

<u>St_id</u>	**St_name**	**St_designation**	**Salary**	**Super_Stid**	**St_lname**	**St_minid**	**Dno**

Step 2: Mapping of weak entity types

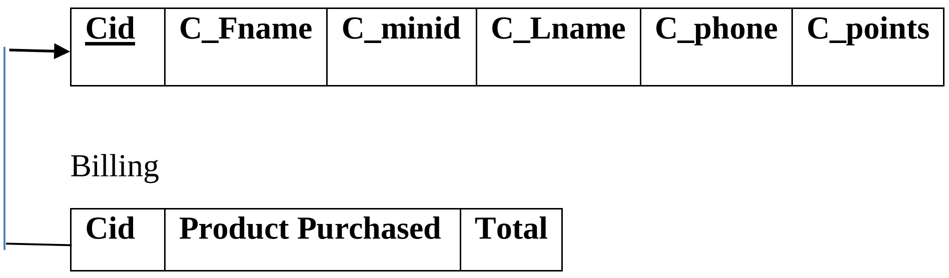
Identity the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relations T that represents the order entity type participating in R. Include any simple attributes of the 1: N relationship type as attributes of S.

Billing

Cid	**Product Purchased**	**Total**

Step 3: Mapping of binary 1:1 relation type

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R If any.

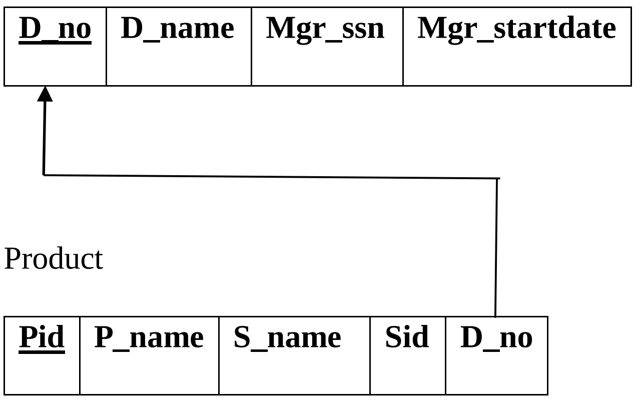


Customer

<u>Cid</u>	C_Fname	C_minid	C_Lname	C_phone	C_points

Billing

Cid	Product Purchased	Total

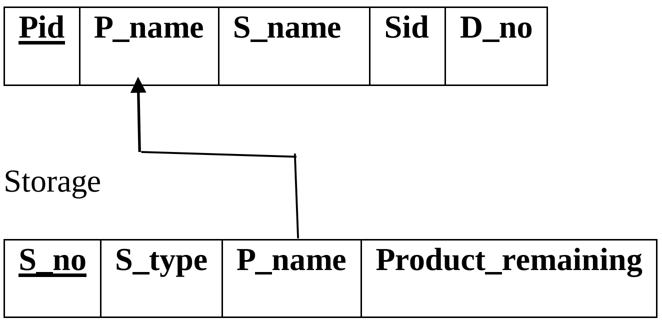


Department

<u>D_no</u>	D_name	Mgr_ssn	Mgr_startdate

Product

<u>Pid</u>	P_name	S_name	Sid	D_no



Product

<u>Pid</u>	P_name	S_name	Sid	D_no

Storage

<u>S_no</u>	S_type	P_name	Product_remaining

Step 4: Mapping of Binary 1: N Relation Types

For each regular binary 1: N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R. Include any simple attributes of the 1: N relation type as attributes of S.

Customer

Cid	C_Fname	C_minid	C_Lname	C_phone	C_points

Product

Pid	P_name	S_name	Sid	D_no	Cid

Department

Dno	D_name	Mgr_ssn	Mgr_startdate

Staff

St_id	St_name	St_designation	Salary	Super_Stid	St_lname	St_minid	Dno

Step 5: Mapping of Binary N: M Relation Types.

For each regular binary M: N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types, their combination will form the primary key of S. Also include any simple attributes of the M: N relationship type (or simple components of composite attributes) as attributes of S.

Suppliers

<u>S_id</u>	S_name	P_name	S_location	S_phone

Product

<u>Pid</u>	P_name	S_name	Sid	D_no

Step 6: Mapping of Multi valued Attributes

Mapping of Multi valued attributes: For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

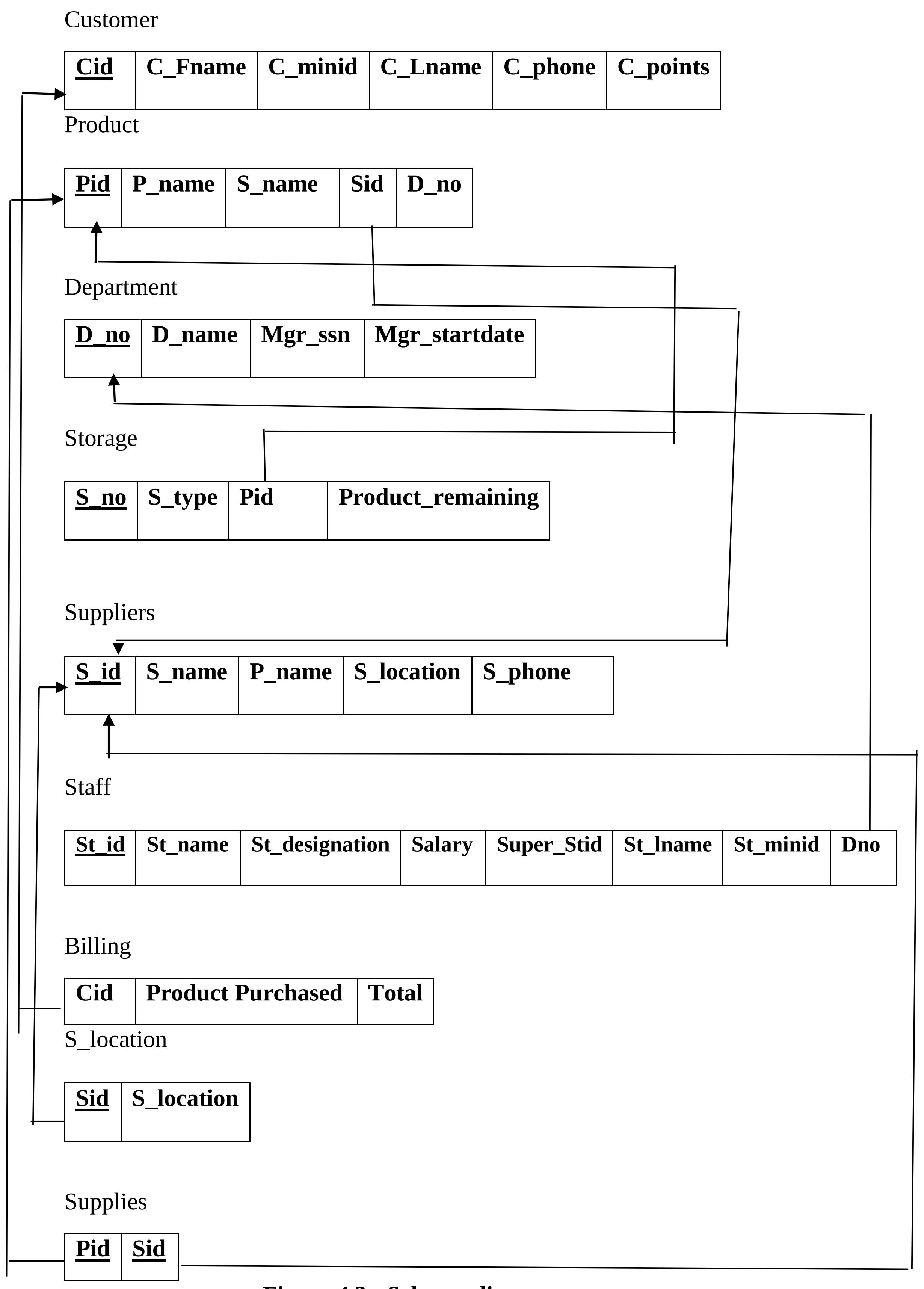
S_location

<u>Sid</u>	S_location

Step 7: Mapping of N-ary Relationship Type

For each n-ary relationship type R, where n>2, create a new relationship S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. Also include any simple attributes of the n –ary relationship type (or simple components of composite attributes) as attributes of S

4.3 MAPPING OF THE ER SCHEMA TO RELATIONS



Customer

Cid	**C_Fname**	**C_minid**	**C_Lname**	**C_phone**	**C_points**

Product

Pid	**P_name**	**S_name**	**Sid**	**D_no**

Department

D_no	**D_name**	**Mgr_ssn**	**Mgr_startdate**

Storage

S_no	**S_type**	**Pid**	**Product_remaining**

Suppliers

S_id	**S_name**	**P_name**	**S_location**	**S_phone**

Staff

St_id	**St_name**	**St_designation**	**Salary**	**Super_Stid**	**St_lname**	**St_minid**	**Dno**

Billing

Cid	**Product Purchased**	**Total**

S_location

Sid	**S_location**

Supplies

Pid	**Sid**

Figure 4.3: Schema diagram

Figure 4.3 is the Schema diagram of the project which gives the detail information of the entities. It gives the details of each table along with the primary key and foreign key of the tables. Figure 4.3 consists of Customer, Product, Department, Suppliers, Staff, Storage, Billing, Address Table. It also gives the relationship between each table.

4.4 NORMALIZE THE RELATIONS

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

FIRST NORMAL FORM(1NF)

A Data is said to be in first normal form if:

1. There are no duplicate rows in the table.
2. Each cell is single valued or atomic

SECOND NORMAL FORM (2NF)

A Database is said to be in second normal form if

1.If the tables are in 1NF and if all non-key attributes of the tables are fully functionally dependent on all the key attributes.

THIRD NORMAL FORM (3NF)

A Database is said to be in second normal form if

1.If all the tables in it are in 2NF and without any transitive dependencies, i.e X>Y, Y>Z, X>Z.

2. According to CODD’s definition a relation schema R is in 3NF if it satisfies 2NF and all non-prime attributes are transitively dependent on the primary key**.**

4.5 CREATION OF TABLES

4.4.1 CREATING CUSTOMER TABLE

```
 CREATE TABLE CUSTOMER
(CID VARCHAR(15) PRIMARY KEY,
C_FNAME VARCHAR(20),
C_MINID VARCHAR (20),
C_LNAME VARCHAR (20),
C_PHONE NUMBER (10),
C_POINTS NUMBER (5));
```

4.4.2 CREATING DEPARTMENT TABLE

```
CREATE TABLE DEPARTMENT
(DNO VARCHAR (15) PRIMARY KEY,
D_NAME VARCHAR (20),
MGR_SSN VARCHAR (15),
MGR_START_DATE DATE (10));
```

4.4.3 CREATING SUPPLIERS TABLE

```
CREATE TABLE SUPPLIERS
(SID VARCHAR (15) PRIMARY KEY,
S_NAME VARCHAR (20),
P_NAME VARCHAR (20),
S_LOCATION VARCHAR (20),
S_PHONE NUMBER (10));
```

4.4.4 CREATING PRODUCT TABLE

```
CREATE TABLE PRODUCT
(PID VARCHAR (15) PRIMARY KEY,
P_NAME VARCHAR (20),
S_NAME VARCHAR (20),
SID REFERENCES SUPPLIERS (SID),
DNO REFERENCES DEPARTMENT (DNO));
```

4.4.5 CREATING STORAGE TABLE

```
CREATE TABLE STORAGE
(SNO VARCHAR (15) PRIMARY KEY,
S_TYPE VARCHAR (15),
P_NAME VARCHAR (15),
PRODUCT_REMAINING (15));
```

4.4.6 CREATING STAFF TABLE

```
CREATE TABLE STAFF
(ST_ID VARCHAR (15) PRIMARY KEY,
ST_FNAME VARCHAR (15),
ST_MINID VARCHAR (15),
ST_LNAME VARCHAR (15),
ST_DESIGNATION VARCHAR (15),
SUPER_STID VARCHAR (15),
DNO REFERENCES DEPARTMENT (DNO));
SALARY NUMBER (10));
```

4.4.7 CREATING BILLING TABLE

```
CREATE TABLE BILLING
(CID VARCHAR (15) PRIMARY KEY,
PRODUCT_PURCHASED VARCHAR (50),
TOTAL NUMBER (5));
```

4.5 INSERTION OF TUPLES

4.5.1 INSERTING VALUES INTO CUSTOMER TABLE

```
INSERT INTO customer VALUES ('C001','KUSHAL','KUMAR','K A', 987123, 75);
INSERT INTO customer VALUES ('C0010','Sharan',",", 336788, 65);
INSERT INTO customer VALUES ('C0011','Prakruthi','G K',", 997756, 80);
INSERT INTO customer VALUES ('C0012','Kumuda','N R',", 456789, 40);
INSERT INTO customer VALUES ('C002','Poojitha',",'Keshav', 876234, 95);
INSERT INTO customer VALUES ('C003','Rohan','C',", 123098, 85);
INSERT INTO customer VALUES ('C004','Ravishankar','C',", 654235, 75);
INSERT INTO customer VALUES ('C005','Roopesh','G B',", 456321, 45);
INSERT INTO customer VALUES ('C006','Pramod','R','Gowda', 456789, 55);
INSERT INTO customer VALUES ('C007','K',",'Shreevershith', 654678, 66);
INSERT INTO customer VALUES ('C008','Ramesh','K',", 987345, 76);
INSERT INTO customer VALUES ('C009','Thanushree','K A',", 554433, 78);
```

4.5.2 INSERTING VALUES INTO DEPARTMENT TABLE

```
INSERT INTO department VALUES ('d001','Administration','mg001', 2018-01-18);
INSERT INTO department VALUES ('d002','Administration','mg002', 2017-03-15);
INSERT INTO department VALUES ('d003','Cleaning','mg002', 2018-02-20);
INSERT INTO department VALUES ('d004','Billing','mg001', 2018-08-15);
INSERT INTO department VALUES ('d005','Storage','mg002', 2018-01-18);
```

4.5.3 INSERTING VALUES INTO SUPPLIERS TABLE

```
INSERT INTO suppliers VALUES
('s002','green_distributors','vegetables','mandya',747437);
INSERT INTO suppliers VALUES ('s003','nescafe','beverages','bengaluru',874536);
INSERT INTO suppliers VALUES
('s004','shiva_distributors','grains&cerals','bengaluru',234678);
INSERT INTO suppliers VALUES
('s005','ari_distributors','biscuts&chocolates','bengaluru',345321);
INSERT INTO suppliers VALUES ('S006','shree distributors', 'milk & milk
products','bengaluru ', 747437);
```

4.5.4 INSERTING VALUES INTO PRODUCT TABLE

```
INSERT INTO product VALUES('p001','cool_drinks','ali_distributors',01,'d001');
INSERT INTO product VALUES('p002','biscuts','hari_distributors',03,'d005');
INSERT INTO product VALUES('p003','chocolates','hari_distributors',02,'d005');
INSERT INTO product VALUES('p004','milk & milk prod','nandhini',02,'d005');
```

4.5.5 INSERTING VALUES INTO STAFF TABLE

```
INSERT INTO staff VALUES('st001','pavan','r','kumar','manager',25000,'d001',");
INSERT INTO staff VALUES('st002','koushik',",",'billing',20000,'d005','st001');
INSERT INTO staff VALUES('st003','shivappa',",",'watchman',7000,'d003','st001');
INSERT INTO staff VALUES('st004','kiran','kumar','k','manager',30000,'d005',");
INSERT INTO staff VALUES('st005','shivu',",",'cleaner',9000,'d005',");
```

4.5.6 INSERTING VALUES INTO STORAGE TABLE

```
INSERT INTO storage VALUES(1,'cold storage','vegetables',74);
INSERT INTO storage VALUES(2,'cold storage','cool drinks',55);
INSERT INTO storage VALUES(3,'dry storage','beverages',65);
INSERT INTO storage VALUES(4,'normal storage','grains & cerals',200);
INSERT INTO storage VALUES(5,'cold storage','milk & milk prod',120);
```

4.5.7 INSERTING VALUES INTO BILLING TABLE

```
INSERT INTO billing VALUES('C001','milk & milk prod',150);
INSERT INTO billing VALUES('C002','biscuts milk & chocolates',200);
INSERT INTO billing VALUES('C003','vegetables fruits & chocolates',175);
INSERT INTO billing VALUES('C005','chocolates milk and beverages',200);
INSERT INTO billing VALUES('C006','vegetables',75);
```

4.6 CREATION OF TRIGGERS

A trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database. The trigger stores the customer cid ,cphone and cpoints. The trigger acts after the insertion of the doctor record.

```
DROP TRIGGER IF EXISTS `TRACK`;

CREATE DEFINER=`root`@`localhost` TRIGGER `TRACK` AFTER
INSERT ON`customer`

 FOR EACH ROW INSERT INTO CTRACK VALUES(NEW.CID,NEW.C_PHO
NE,NEW.C_POINTS)
```

4.7 CREATION OF STORED PROCEDURES

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary.

Uses for stored procedures include data-validation or access-control mechanisms. Furthermore, stored procedures can consolidate and centralize logic that was originally implemented in applications. Stored procedures can access or modify data in a database, but it is not tied to a specific database or object, which offers a number of advantages.

The stored procedure displays Dno and Pid of all the department and product in the database

```
DROP PROCEDURE `proc`;
CREATE DEFINER=`root`@`localhost`
PROCEDURE`proc`() NOT DETERMINISTIC NO SQL SQL SECUR
ITY DEFINER SELECT D.DNO,P.PID FROM department D,pro
duct P WHERE D.DNO=P.DNO
```

CHAPTER 5

FRONT END DESIGN

5.1 CONNECTIVITY TO DATABASE

To Connect to MySQL Using PHP:

There are several methods for connecting to a MySQL database using PHP:

- MySQL Improved (*mysql*) extension
- PDO (PHP Data Objects)
- Legacy MySQL (*mysql_*) functions
- Connecting to a remote MySQL database using PHP

Connecting to MySQL using the MySQL Improved extension:

The MySQL Improved extension uses the *mysql*class, which replaces the set of legacy MySQL functions.

To connect to MySQL using the MySQL Improved extension, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
<?php

$mysqli = new mysqli("localhost", "username", "password", "dbname");

?>
```

2. After the code connects to MySQL and selects the database, you can run SQL queries and perform other operations. For example, the following PHP code runs a SQL query that extracts the *Did* from the *DOCTOR* table, and stores the result in the *$result* variable:

```
<?php

$result = $mysqli->query("SELECT Did FROM DOCTOR");

?>
```

3. Finally, we close the connection. Although this isn't strictly speaking necessary, PHP will automatically close the connection when the script ends.

```
<?php

//close the connection

mysqli_close($mysqli);

?>
```

5.2 FRONT END CODE

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design 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.[citation needed] The UML has become the standard language in object-oriented analysis and design.[citation needed] It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.[citation needed]

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words the first step in the solution to the problem is the design of the project.

CONNECTIVITY BASIC CODE USED:

dbh.inc.php:

```
<?php
$dbServername = "localhost";
$dbUsername = "root";
$dbPassword = " ";
$dbName = "railway";
$conn = mysqli_connect ($dbServername, $dbUsername, $dbPassword, $dbName);
?>
```

CREATING FRONTEND PAGE

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Home Page</title>
    <link rel="stylesheet" href="home.css">
  </head>
  <body>
    <header>
      <div class="main">
        <div class="logo">
          <img src="">
        </div>
        <br>
        <ul>
            <li><a href="about.html">About</a></li>
              <li><a href="services.html">Services</a></li>
              <li><a href="contacts.php">Contact Us</a></li>
                <li><a href="login1.php">Login</a></li>
        </ul>
      </div>
      <div class="title">
        <h1>KUSHAL GROCERY</h1>
        <center><h3 style="color:white;">Quality at it's best</h3></center>
      </div>
    </header>
  </body>
</html>
```

5.2.2 FRONT END CODE FOR INSERTING VALUES

```
<?php
$host = "localhost";
$user = "root";
$password="";
$db = "grocery";
$conn = new mysqli($host, $user, $password, $db);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if (isset($_POST['submit']))
{
    $no = $_POST['id'];
    $name1 = $_POST['name1'];
    $name2 = $_POST['name2'];
    $name3 = $_POST['name3'];
    $Cphone = $_POST['phone'];
  $Cpoints = $_POST['points'];
$query = "select count(*) from customer where CID = '$no'";
$execute = mysqli_query($conn,$query);
$count = mysqli_fetch_row($execute);
if($count[0] == 1)
{
  ?><script type="text/javascript">alert("Data Already Exists")</script><?php
}

    $sql = "INSERT INTO customer(CID,C_Fname,C_minid,C_Lname,C_Phone,C_Points )
VALUES ('$no', '$name1', '$name2','$name3',$Cphone,$Cpoints)";
    if ($conn->query($sql) === TRUE) {
    ?><script type="text/javascript">alert("Record inserted Successfully")</script><?php
}
}
?>

<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title></title>
</head>
<style>

div{
    background-color:;
    width: 200px;
    border: 1px;
    padding: 25px;
    position: center;
    margin: 0px;
}
body{
  margin: 0;
  padding: 0;
  background: url(grocery14.jpeg);
  background-size: cover;
  background-position: center;
  font-family: sans-serif;
}
</style>
<body>
  <center><div align = "center">
    <form action="custins.php" method="post">
      Customer<br><br>
    Customer_id: <input name="id" type="text" required><br><br>
    Fname: <input name="name1" type="text" required><br><br>
    Cminid: <input name="name2" type="text"><br><br>
    Clname:<input name="name3" type="text"><br><br>
    Cphone:<input name="phone" type="text" required><br><br>
    Cpoints:<input name="points" type="text"><br><br>
    <input type="submit" name="submit">
        </form>
    </div>
  </center>
<center><h3> <a href="customer.php">HOME</a></h3></center>
</body>
</html>
```

5.2.3 FRONT END CODE FOR UPDATING VALUES

```
<?php
$host = "localhost";
```

```
$user = "root";
$password="";
$db = "grocery";
$conn = new mysqli($host, $user, $password, $db);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if (isset($_POST['insert']))
{
    $no = $_POST['no'];
    $name1 = $_POST['name1'];
    $name2 = $_POST['name2'];
    $name3 = $_POST['name3'];
    $phone = $_POST['phone'];
    $points = $_POST['points'];
    $sql = "UPDATE customer SET
C_fname='$name1',C_minid='$name2',C_lname='$name3',c_phone=$phone,c_points
='$points' WHERE cid='$no'";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

}

?>
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title></title>
</head>
<style>

div{
    background-color:;
    border: 1px ;
    padding: 25px;
    margin: 0px;
}
body{
  margin: 0;
  padding: 0;
  background: url(grocery14.jpeg);
  background-size: cover;
  background-position: center;
  font-family: sans-serif;
}
</style>
<body>
  <div align = "center">
    <form action="custupd.php" method="post">
       Update<br><br>
    Cid: <input name="no" type="text" required><br><br>
    Customer fname: <input name="name1" type="text"><br><br>
    Customer m_init: <input name="name2" type="text"><br><br>
    Customer lname: <input name="name3" type="text"><br><br>
    Customer phone: <input name="phone" type="text"><br><br>
    Customer points: <input name="points" type="text"><br><br>

    <input type="submit" name="insert">
        </form>
    </div>
<center><h3> <a href="customer.php">HOME</a></h3></center>
</body>
</html>
```

5.2.6 FRONT END CODE FOR DELETING VALUES

```
<?php
$host = "localhost";
$user = "root";
$password="";
$db = "grocery";
$conn = new mysqli($host, $user, $password, $db);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if (isset($_POST['insert']))
{
    $no = $_POST['no'];

    $sql =  "DELETE FROM customer WHERE CID='$no'";
    if (mysqli_query($conn, $sql)) {
    ?><script type="text/javascript">alert("Record Deleted
Successfully")</script><?php
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}
}
?>
<!DOCTYPE html>
<html lang="">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title></title>
</head>
<style>

div{


    background-color:;
    border: 1px;
    padding: 25px;
    margin: 0px;
}
body{
  margin: 0;
  padding: 0;
  background: url(grocery24.jpeg);
  background-size: cover;
  background-position: center;
  font-family: sans-serif;
}
b {
  color: #8b0000;
}


}
</style>
<body>
  <div align = "center">
    <form action="custdel.php" method="post">
      DELETE<br><br>
    <b>Customer_No</b> : <input name="no" type="text" required><br><br>
    <input type="submit" name="insert">
        </form>
    </div>
```

```
<center><h3> <a href="customer.php">HOME</a></h3></center>
</body>
</html>
```

5.2.8 FRONT END CODE FOR DISPLAYING VALUES

```
<!DOCTYPE html>
<html lang="">
<head>
    <title>DISPLAY CUSTOMER INFO</title>
</head>
  <style>
    html, body {
  margin: 0;
  padding: 0;
  width: 100%;
}
body{
  width: 100%;
  height: 100vh;
  background: url();
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  display: table;
  font-family: sans-serif;
}
    table {
      padding: 5px;
      text-align: center;
      font-size: 20px;
    }
    h3 {
      margin-left: 75%;
    }
```

```
          body{
           margin: 0;
           padding: 0;
           background: url(grocery32.lpeg.jpg);
           background-size: cover;
           background-position: center;
           font-family: sans-serif;
          }

        </style>
    <body>

        <table width=850; border="5" align="center">
        <caption><h1                style="color:white;">CUSTOMER
DETAILS</h1></caption>
          <tr>
            <th style="color:white;">CID</th>
            <th style="color:white;">C_FNAME</th>
            <th style="color:white;">C_MINID</th>
            <th style="color:white;">C_LNAME</th>
            <th style="color:white;">C_PHONE</th>
            <th style="color:white;">C_POINTS</th>



          </tr>
          <?php
            global $connection;
            $connection = mysqli_connect('localhost','root','','grocery');
            $Query = "SELECT * FROM CUSTOMER";
            $Execute = mysqli_query($connection, $Query);
            while($Datarows = mysqli_fetch_array($Execute))
            {
              $CID = $Datarows['CID'];
```

```
                $C_Fname = $Datarows['C_Fname'];
                $C_minid = $Datarows['C_minid'];
                $C_Lname = $Datarows['C_Lname'];
                $C_Phone = $Datarows['C_Phone'];
                $C_Points = $Datarows['C_Points'];




              ?>


            <tr>
            <td style="color:white;"><?php echo $CID; ?></td>
            <td style="color:white;"><?php echo $C_Fname; ?></td>
            <td style="color:white;"><?php echo $C_minid; ?></td>
            <td style="color:white;"><?php echo $C_Lname; ?></td>
            <td style="color:white;"><?php echo $C_Phone; ?></td>
            <td style="color:white;"><?php echo $C_Points; ?></td>
                </tr>


        <?php   } ?>
        </table>
        <h3> <a href="customer.php">HOME</a></h3>
    </body>
    </html>
```

CHAPTER 6

TESTING

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components sub assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TESTING PROCESS

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

6.2 TESTING OBJECTIVES

The main objectives of testing process are as follows.

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

6.3 TEST CASES

The test cases provided here test the most important features of the project.

6.3.1 Test cases for the project

Table 6.3 Test Case for the project

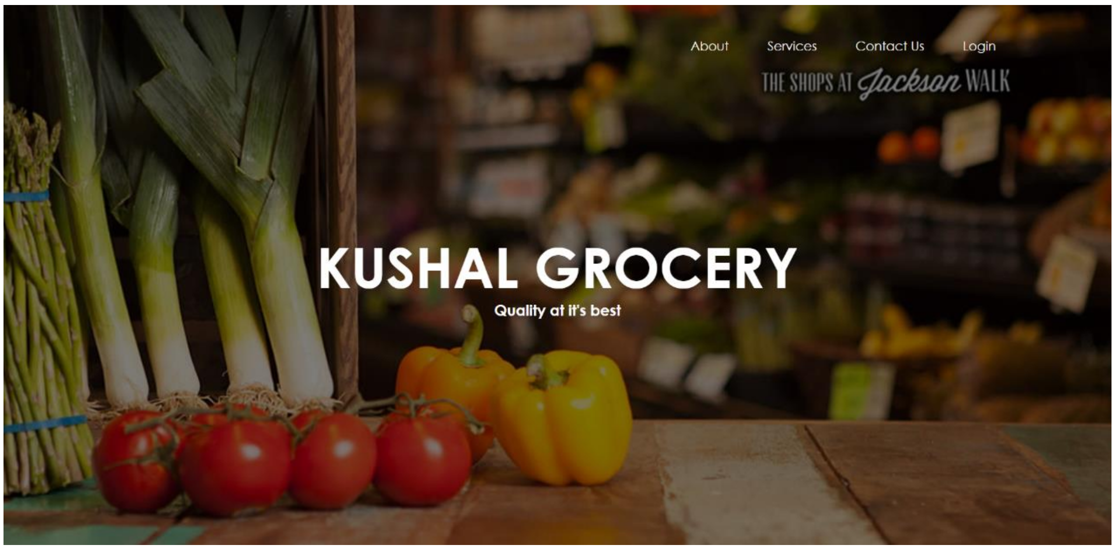
Sl No	**Test Input**	**Expected Results**	**Observed Results**	**Remarks**
1	Insert a Record	New tuple should be inserted	Query Ok 1 row affected or inserted	PASS
2	Insert a Record	New tuple should be inserted	ERROR	FAIL
3	Search a Record	Display the Record	Required Record Found	TRUE
4	Search a Record	Display the Record	No Record found	FAIL
5	Delete a record	Delete the Record	Query Ok 1 row affected or record Deleted	PASS

CHAPTER 7

RESULTS

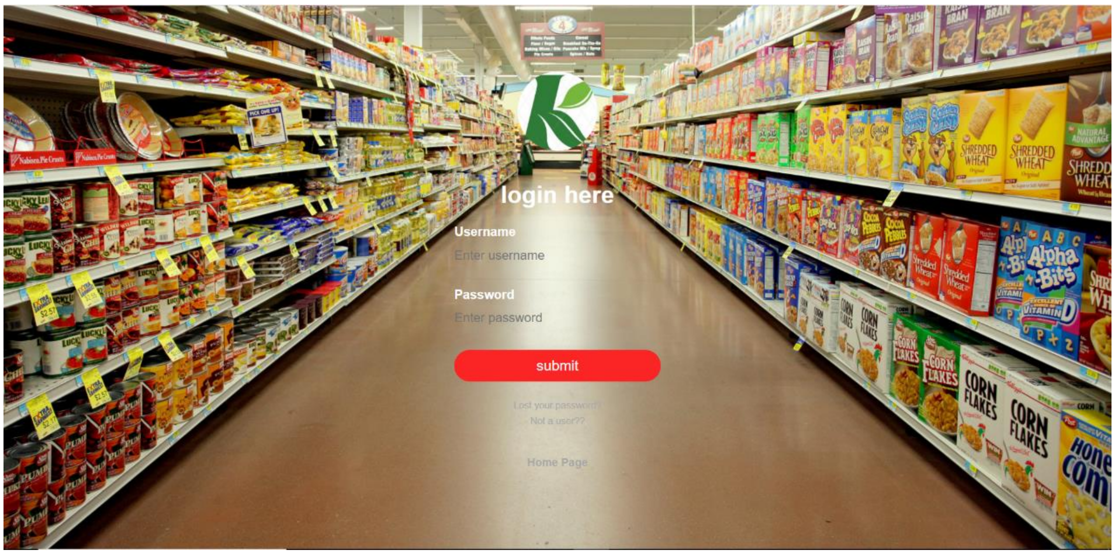
This section describes the screens of the "Project title". The snapshots are shown below for each module.

7.1 SNAPSHOTS

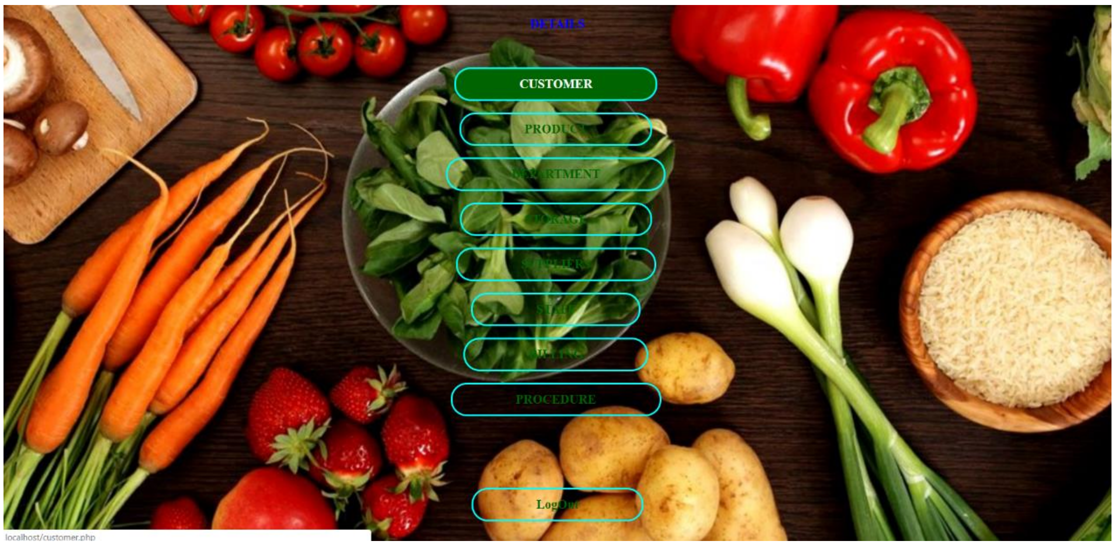


Snapshot 7.1.1 Front Page

The page layout shows the front page which contains the details of the grocery. On clicking on Login, Contact Us, Services, About will give the details of the links.

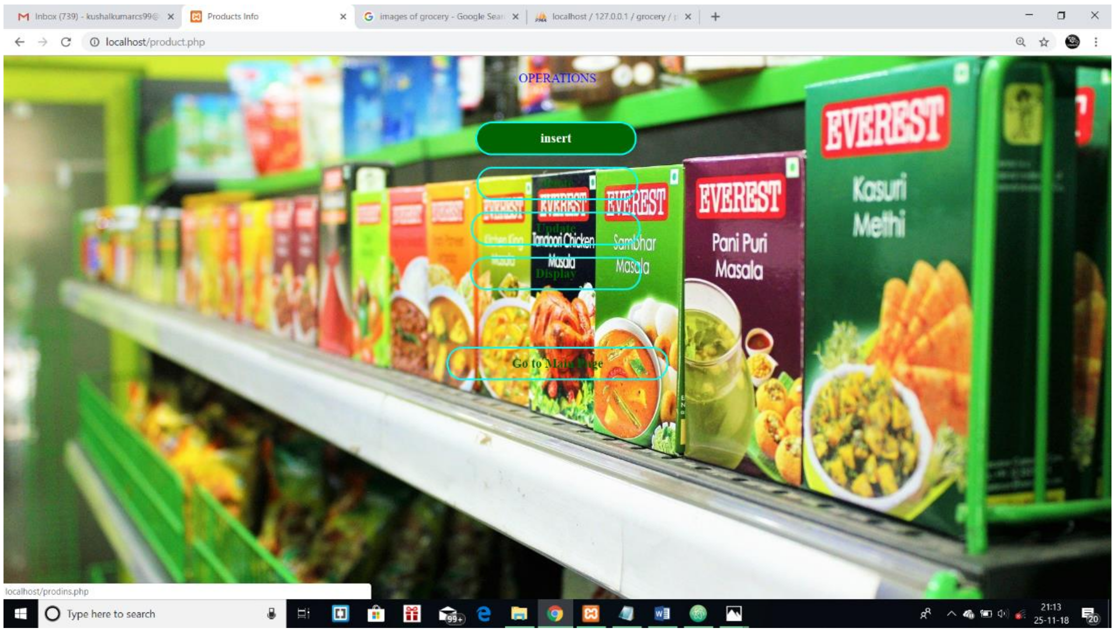


Snapshot 7.1.2 Login Page



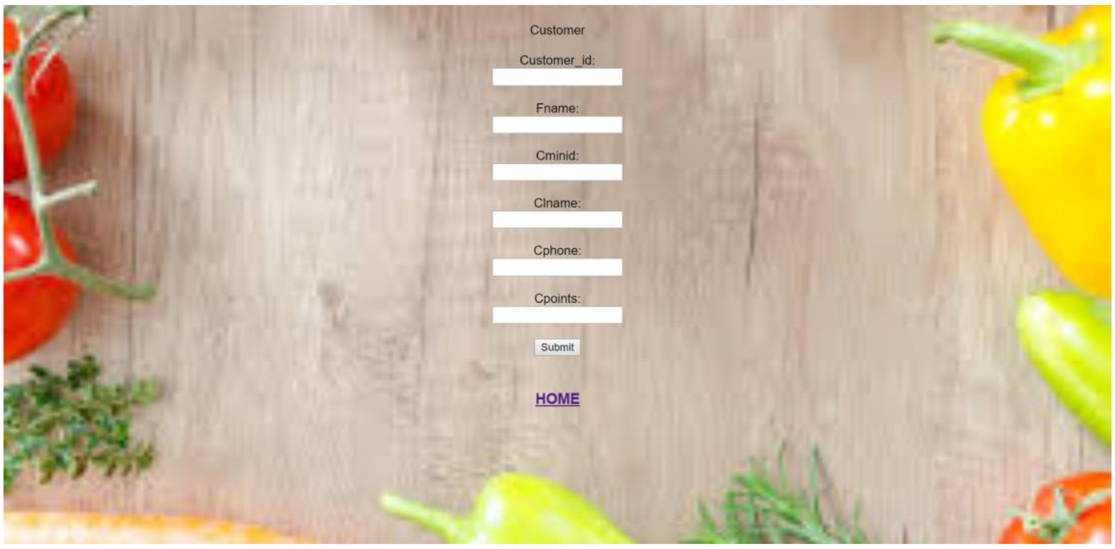
Snapshot 7.1.3 Tables Page

The above snapshot displays all the table names in a order



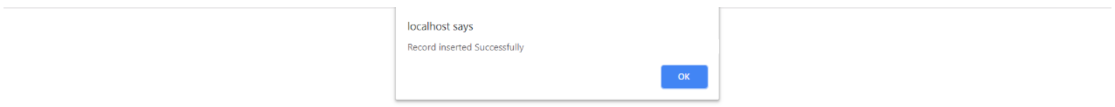
Snapshot 7.1.4 Operations Page

The above snapshot describes about the operations done by the each tables. It has insert, delete, update, and display operations.



Snapshot 7.1.5 Insertion of Records

This page layout is for the insertion of the new doctor who joins the hospital. It tells to fill the details of the doctor like Cid, CFname, Cminid, CLname, CPhone, and the CPoints.

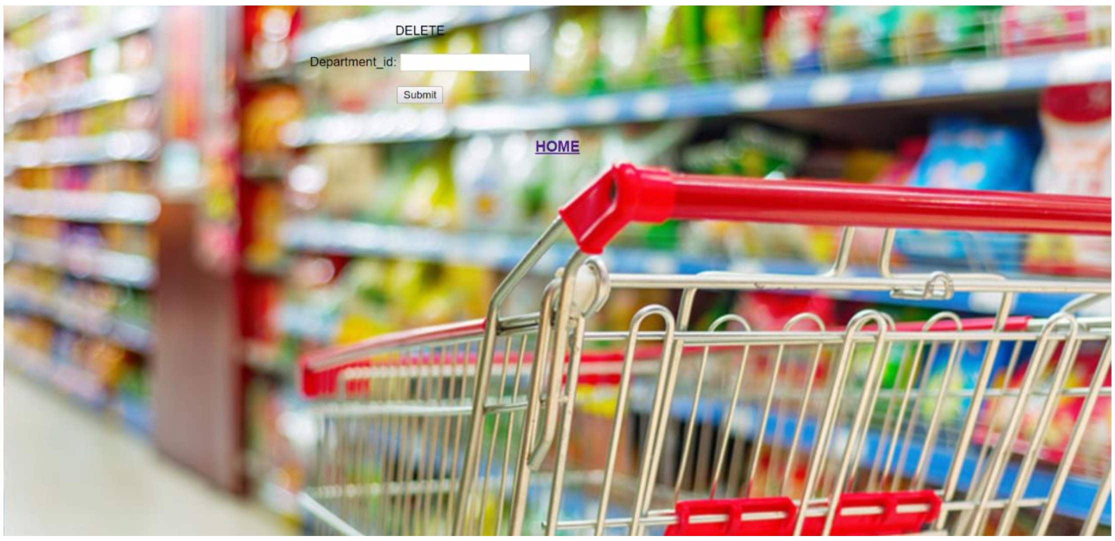


localhost says

Record inserted Successfully

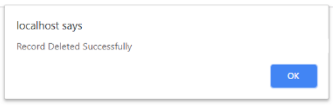
Snapshot 7.1.6 Insert Operation

The snapshot shows the acknowledgement for insert operation



Snapshot 7.1.7 Deletion Of Records

The above snapshot describes about delete operation of customer table. It takes CID has primary attribute to delete a record in the customer table.

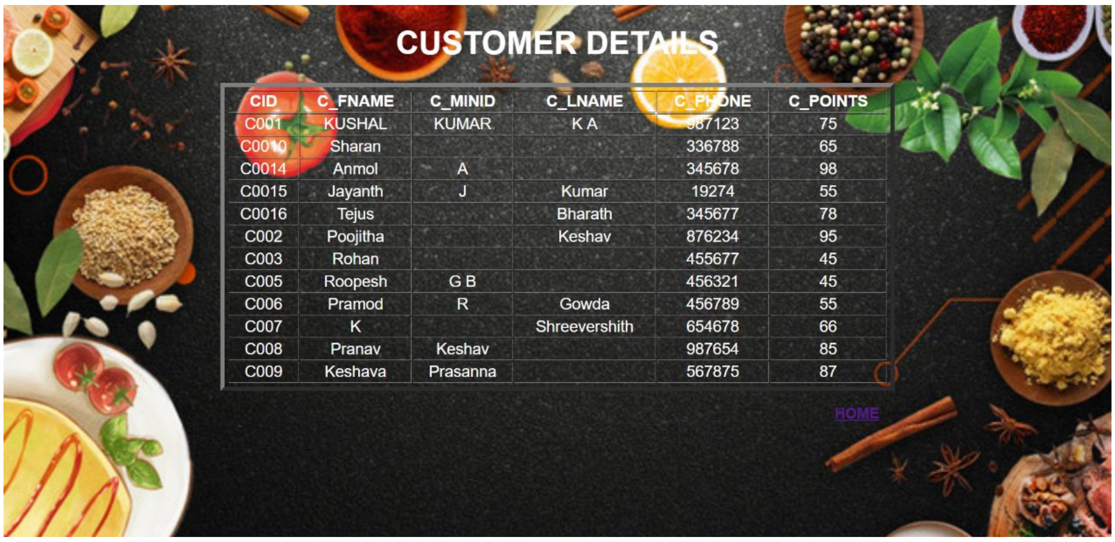


localhost says

Record Deleted Successfully

Snapshot 7.1.8 Delete Operation

The above snapshot shows the acknowledgement for deleting a record from the customer table.

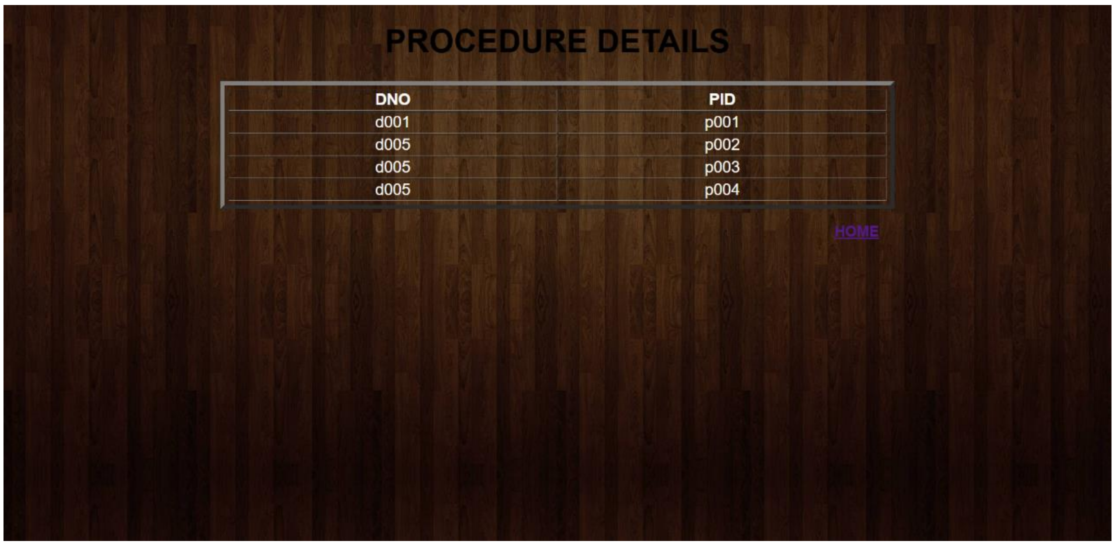


CUSTOMER DETAILS

CID	C_FNAME	C_MINID	C_LNAME	C_PHONE	C_POINTS
C001	KUSHAL	KUMAR	K A	987123	75
C0010	Sharan			336788	65
C0014	Anmol	A		345678	98
C0015	Jayanth	J	Kumar	19274	55
C0016	Tejus		Bharath	345677	78
C002	Poojitha		Keshav	876234	95
C003	Rohan			455677	45
C005	Roopesh	G B		456321	45
C006	Pramod	R	Gowda	456789	55
C007	K		Shreevershith	654678	66
C008	Pranav	Keshav		987654	85
C009	Keshava	Prasanna		567875	87

Snapshot 7.1.9 Display Opeartion

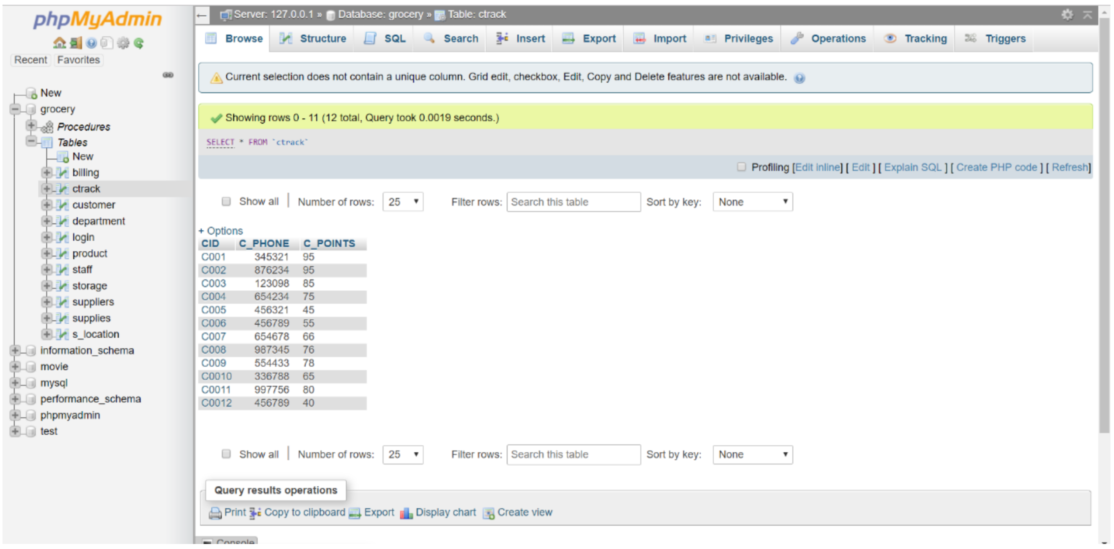
The above table displays all the details of currently inserted records into the customer table.


PROCEDURE DETAILS

DNO	PID
d001	p001
d005	p002
d005	p003
d005	p004

Snapshot 7.1.10 Stored Procedure

This page layout gives the details of the department id and product id which is displayed by calling the stored procedure.



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 11 (12 total, Query took 0.0019 seconds.)

```
SELECT * FROM `ctrack`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

CID	C_PHONE	C_POINTS
C001	345321	95
C002	876234	95
C003	123098	85
C004	654234	75
C005	456321	45
C006	456789	55
C007	654678	66
C008	987345	76
C009	554433	78
C0010	336788	65
C0011	997756	80
C0012	456789	40

Snapshot 7.1.11 Trigger

This page layout gives the details about the trigger. This displays the Customer Cid and their cpoints.

Snapshot1 : -------- Layout

CONCLUSION

The project "Grocery Database" is for computerizing the working in a grocery. The system is used to store the details of customers and staff in the computer. It provides the facility for updating the details of customer and staff by using their unique id. The database is a more efficient way to store the data. It also allows to delete the details of customers and staff. The project displays the type and status of the products. It also generates the bills of the customer. It provides facility to store the products reports of the grocery store.

REFERENCES

[1] Raghu Ramakrishan and Johannes Gehrke , “Database Management Systems” , McGRAW HILL , 3rd Edition.

[2] Ramez Elmasri and Shamkant B. Navathe , “Fundamentals of Database Systems”, Pearson , 7th Edition.

[3] Herbert Schildt , Java: “The Complete References” , McGRAW HILL , 7th Edition.

[4] https://www.tutorialspoint.com

[5] https://www.w3schools.com

[6] https://www.youtube.com