# Online Grocery Shopping

**Chapter 1**

**Introduction**

Online Grocery Shopping is developed in java, which can keep track of all your business activity in Online Grocery shop from small segments to large and very large segments.

Online Grocery Shopping sells various types of Online Grocery Store items and it is very difficult to categorize these items on the basis of their manufacturing dates, type of gold used to manufacture it such as either using 24K or 22K. Which items comes under the category of ISI Gold Mark and which items are of local brand. It is also very difficult to analyze the overall transaction for a particular period of time and which ornaments are not available in the shop. To solve these problems this system has been developed. This system will follow the steps and rules to meet user requirements on demand and on time. Apart from this it will remove the difficulties faced by the admin to manage their shop and can be easily handled by the admin without any technical knowledge of the platform used and about the system. Through this system employees working under a particular shop can be easily managed and overall transactions of particular date and time can easily be viewed.

The current system does not provide the method to back up the database which is very important for any business transactions. Current system provides the details of overall business transactions but they do not provide information on sells/purchase in and sells/purchase out. By using this technique shop manager can use their business strategy such as cost cutting method for their business process to gain maximum profit.

##### Overview of the Project

Our proposed “Online Grocery Shopping ” is for those who run a Online Grocery Shopping business. Before doing anything we did a decent research on major difficulties for Online Grocery Shopping . We examined carefully about how to make a huge registering system without failure as well as different functions for different kind of user depending on theirprivilege.

The Online Grocery Shopping requires a system that will handle all the necessary and minute details easily and proper database security accordingly to the user. They requires software, which will store data about members, employees, products, payroll, receipts of members & all transactions that occur in Online Grocery Shopping .

##### Objectives

The objectives of this study are summarized below:

* + - The main objective of the project is to design and develop a user friendly efficient computerized Online Grocery Shopping .
    - An accurate system without any dataredundancy.
    - Secured data storage for Authorityend.
    - Secure the user ends data by providing each user’s own personalcredentials.
    - A flexible system which can maneuver the customer-staff relationship in an effectivemanner.
    - To provide better graphical userinterface.
    - Computerization can be helpful as means of saving time &money.

**Chapter 2**

**Theoretical Background**

We have done a project on Online Grocery Shopping and database management and transactions. This system is proposed to be an automate database management & transactions. This stores customer, member, payment, product, receipts, and products information. It also provides the facility of search & advanced search for searching the records efficiently & immediately. This system provides data storing & report generation with graphical user interface(GUI).

##### SystemStudy

It is always necessary to study and recognize the problems of existing system, which will help in finding out the requirements for the new system. System study helps in finding different alternatives for better solution.

###### The project study basically deals with different operations:

1: Data Gathering

2: Study of Existing System 3: Analyzing Problems

4: Studying various documents

5: Feasibility study for further improvements

###### Following are the steps taken during the initial study:

Initially, we collected all the information, which they wanted to store. Then we studied the working of the current system which is done manually. We noted the limitation of that system which motivated them to have new system. With the help of these documents we got basic ideas about the system as well as input output of the developedsystem.

The most important thing is to study system thoroughly. Here we are studying both existing system and proposed system so that advantages & disadvantages of both the

systems can be understood. The first task was identifying how system can be computerized. Some analysis and projections was done regarding changes to be made to the existing system. The new developed system for Online Fashion Store is simple without complexities.

##### Existing System

An Existing system refers to the system that is being followed till now. Online Grocery Shopping is working manually. The current system is time consuming and also it is very costly, because it involves a lot of paperwork. To manually handle the system was very difficult task. But now-a-days computerization made easy to work.

The following are the reasons why the current system should be computerized:

* + - To increase efficiency with reducedcost.
    - To reduce the burden of paperwork.
    - To save time management for recording details of each and every member and employee.
    - To generate required reportseasily.

##### ProposedSystem

The online Online Grocery Shopping is user-friendly application. This automated system makes all functionality easier for both owners and customers. It is very simple in design and to implement. The system requirements are very low. System resources and the system will work in almost all configurations.

###### It has the following objectives:

**Enhancement:**

The main objective of Smart Online Grocery Shopping is to enhance and upgrade the existing system by increasing its efficiency and effectiveness. The software improves the working methods by replacing the existing manual system with the computer- basedsystem.

###### Automation:

The Smart Online Grocery Shopping automates each and every activity of the manual system and increases its throughput. Thus the response time of the system is very less and it works veryfast.

###### Accuracy:

The Smart Online Grocery Shopping provides the uses a quick response with very accurate information regarding the users etc. Any details or system in an accurate manner, as and when required.

###### User-Friendly:

The software Smart Online Grocery Shopping has a very user-friendly interface. Thus the users will feel very easy to work on it. The software provides accuracy along with a pleasant interface. Make the present manual system more interactive, speedy and userfriendly.

###### Availability:

The transaction reports of the system can be retried as and when required. Thus, there is no delay in the availability of any information, whatever needed, can be captured very quickly and easily.

###### Maintenance Cost:

Reduce the cost of maintenance.

## Chapter 3

**System Analysis & Design**

The way that is followed while carrying on with the development application is as follows:

##### Defining a problem

Defining a problem is one of the important activities of the project. The objective is to define precisely the business problem to be solved & thereby determined the scope of the new system. This phase consist of 2 main tasks. The 1st task within this activity is to review the organization needs that originally initiated the project. The 2nd task is to identify, at an abstract or general level, the expected capabilities of the new system. Thus, it helps us to define the goal to be achieved & the boundary of the system. A clear understanding of the problem will help us in building a better system & reduce the risk of project failure. It also specifies the resources that have to be made available to the project. Three important factors project goal, project bounds & the resource limits are sometimes called the project’s term of reference.

##### Feasibilitystudy

The systems objectives outlined during the feasibility study serve as the basic from which the work of system design is initiated. Much of the activities involved at this stage is of technical nature requiring a certain degree of experience in designing systems, sound knowledge of computer related technology and through understanding of computers available in the market and the various facilities provided by the vendors. Nevertheless, a system cannot be designed in isolation without the active involvement of the user. The user has a vital role to play at this stage too. As we know that data collected during feasibility study wills we utilized systematically during the system design. It should, however be kept in mind that detailed study of the existing system is not necessarily over with the completion of the feasibility study. Depending on the plan of feasibility study, the level of detailed study will vary and the system designstagewillalsovaryintheamountofinvestigationthatstillneedstobedone.

This investigation is generally an urgent activity during the system. Sometimes, but rarely, this investigation may form a separate stage between feasibility study and computer system design. Designing a new system is a creative process, which calls for logical as well as lateral thinking. The logical approach involves systematic moves towards the end product keeping in mind the capabilities of the personnel and the equipment at each decision making step. Lateral thought implies encompassing of ideas beyond the usual functions and equipment. This is to ensure that no efforts are being made to fit previous solutions into newsituations.

The feasibility study proposes one or more conceptual solutions to the problem set for the project. The objective in assessing feasibility is to determine whether a development project has a reasonable chance of success. It helps us to determine the input & output of the system. The following are the criteria that are considered to confirm the project feasibility.

###### The following feasibility study was undertaken for the proposed system:

**Technical feasibility:**

At first it’s necessary to check that the proposed system is technically feasible or not & to determine the technology and skill necessary to carry out the project. If they are not available then find out the solution to obtain them. Hardware is already available in theUniversity.

###### Economic feasibility:

While considering economic feasibility, it is checked in points like performance, information and outputs from the system. The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project , which will give best , return at the earliest. One of the factors , which affect the development of a new system , is the cost it would require.

###### Social feasibility:

Although generally there is always resistance, initially to any change in the system is aimed at reliving the work load of the users to extent the system is going to facilitate user to perform

Operations like calculating salary amounts and deductions, generating reports with less possible errors. Thus there is no reason to make system sociallyunfeasible.

## Chapter 4

**Modules & Features**

##### Module

There are two basic modules in this system as describe briefly in below

* + - **Administrative module:** This user is an admin type who has full rights on the system.

###### AdministrativeModule

This module includes storing and retrieving the details of the data.

* + - * Create , Update, Manage, Delete User
      * Creating OfferPlan
      * ManageBilling
      * Manage User Enquiry throughEmail
      * Manage OwnerInfo

##### Features

There are many features in our system. Some salient and new features are:

* + - Login by FaceRecognition
    - Phone number verification through SMS on signup
    - Online PaymentGateway
    - WebcamIntegration
    - Activity Log ofUser’s

##### Application Requirements

###### UserInterface:

* + - * HTML has been used for developing the user layout for thesystem.
      * Java and JavaScript has been used for creating all the validations and client side scriptingfunctionality.
      * CSS has been used for designing the web page of thesystem.

###### Application:

* + - * Client On Internet : Web Browser , Operating System(Any)
      * Web Server :Apache
      * Database :MySQL
      * Markup Language: HTML, CSS
      * Scripting Language : Java, Javascript, JQuery

## Chapter 5

**System Implementation**

##### ImplementationMethodology

We follow the MVC design pattern for developing our system. Model–view– controller (MVC) is a software design pattern for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from theuser.

* + - **Model:** The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from thecontroller).
    - **View:** The view manages the display ofinformation.
    - **Controller:** The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change asappropriate.

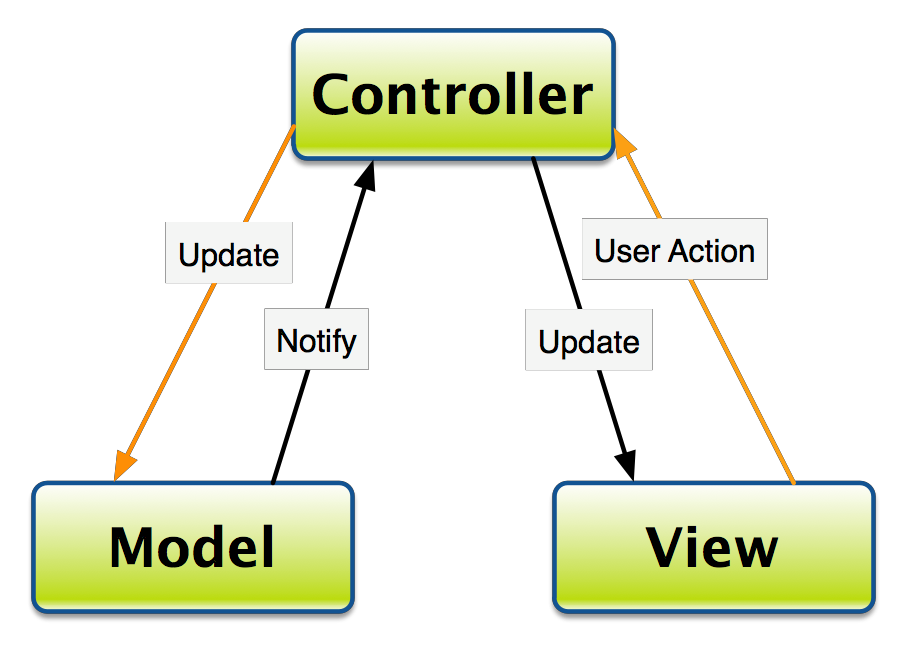


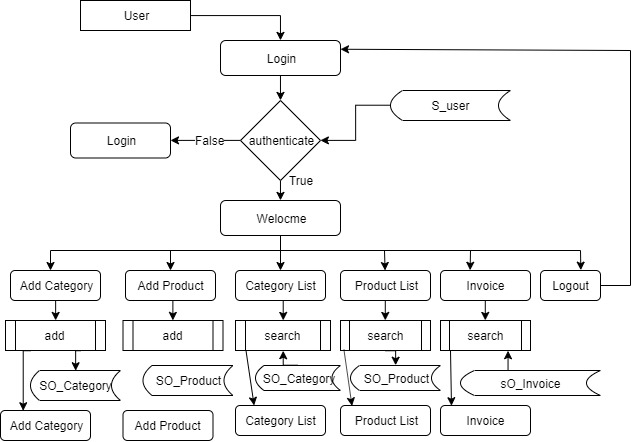
Fig. 5.1: Diagram of A typical collaboration of the MVC components.

##### Data FlowDiagram

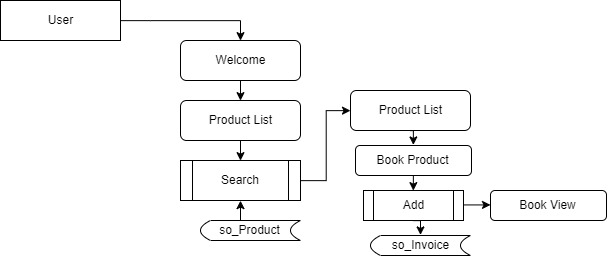
A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.[2] DFDs can also be used for the visualization of data processing.

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel.

##### Level 1 DFD (Administrative Panel) :



##### Level 1 DFD (User Panel) :

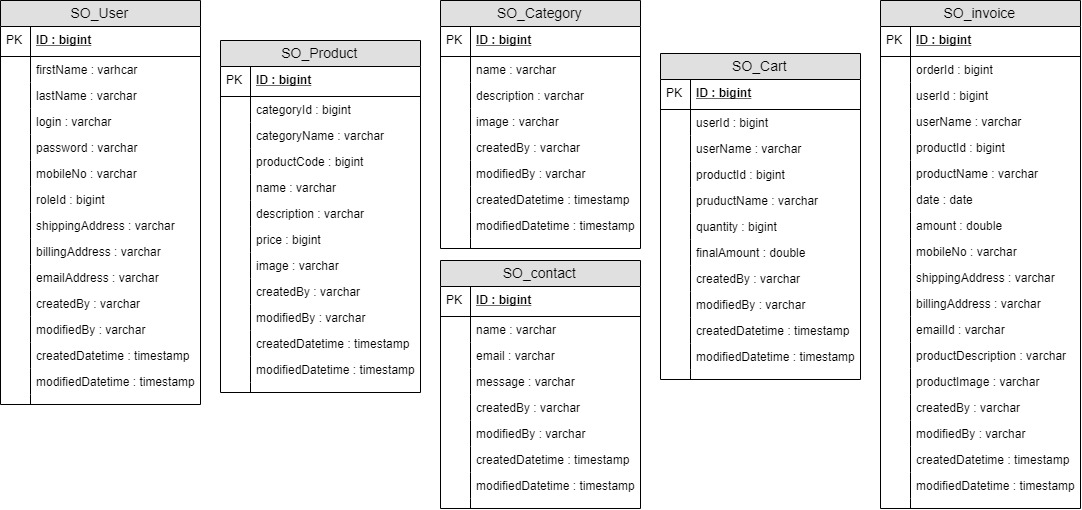


##### Entity Relationship Diagram

An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system. In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of anotherentity.

There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering.



##### Normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

* + - Normalization is used for mainly twopurpose,
    - Eliminating redundant (useless)data.

Ensuring data dependencies make sense i.e. data is logically stored.

Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Update and Deletion Anomalies are very frequent if Database is notnormalized.

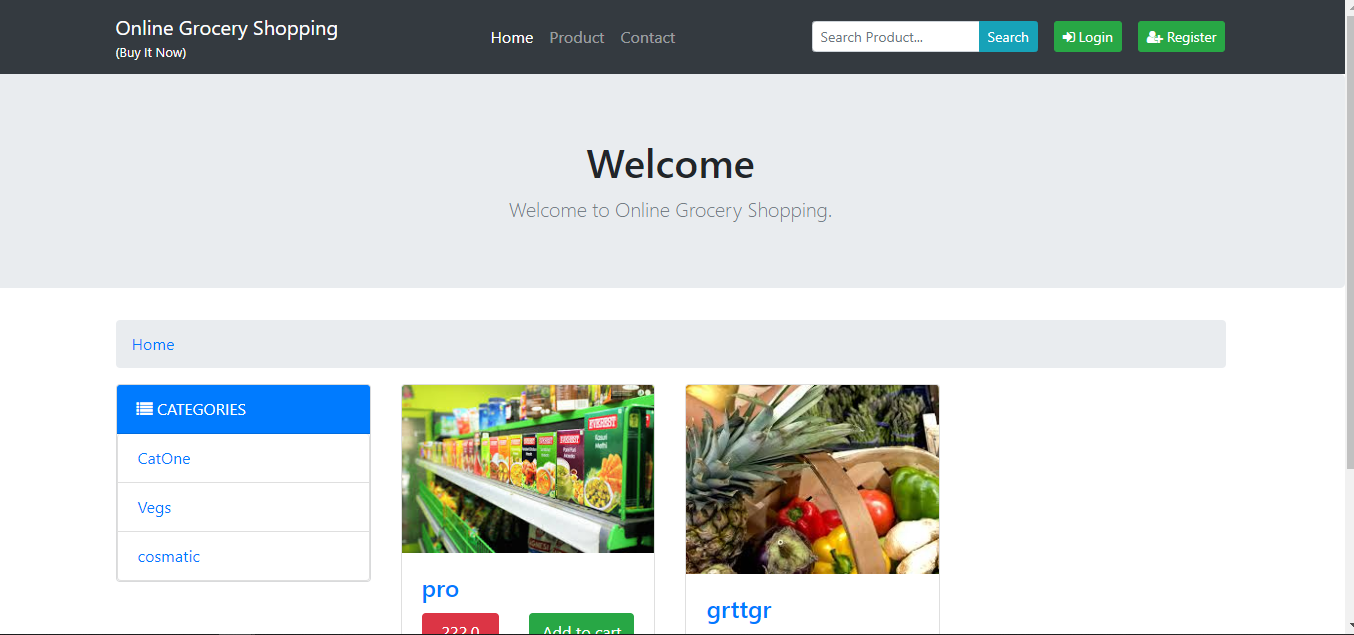
Normalization rule are divided into following normal form.

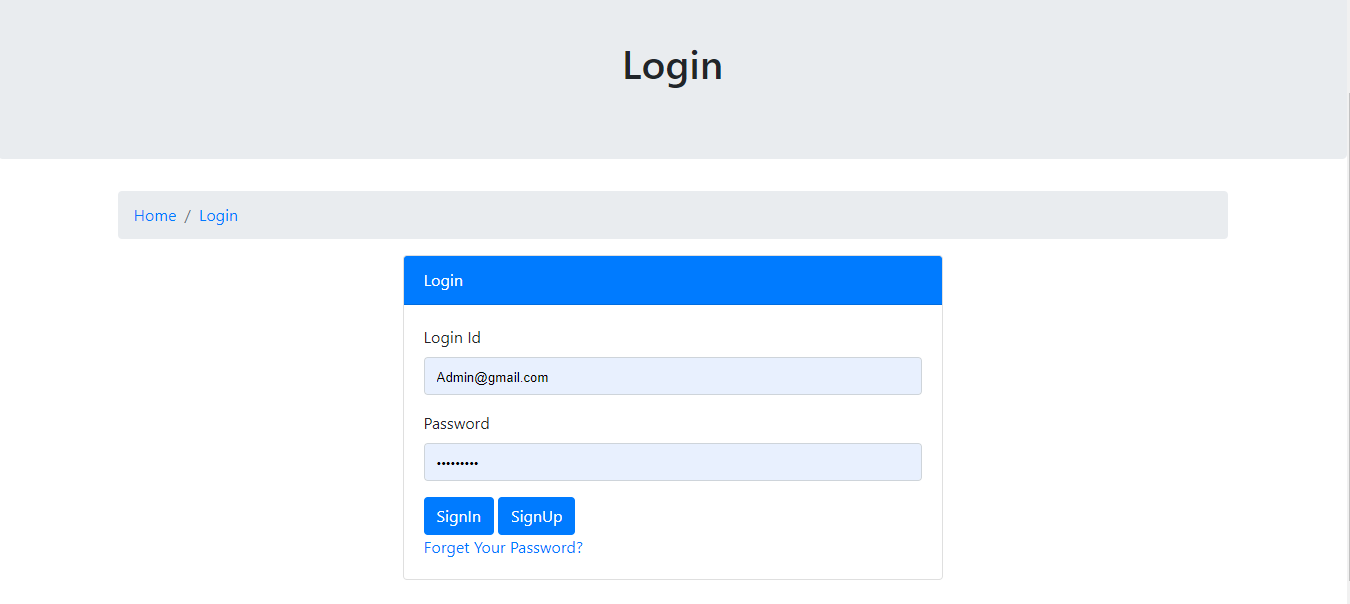
* + - First NormalForm
    - Second NormalForm
    - Third NormalForm
    - BCNF

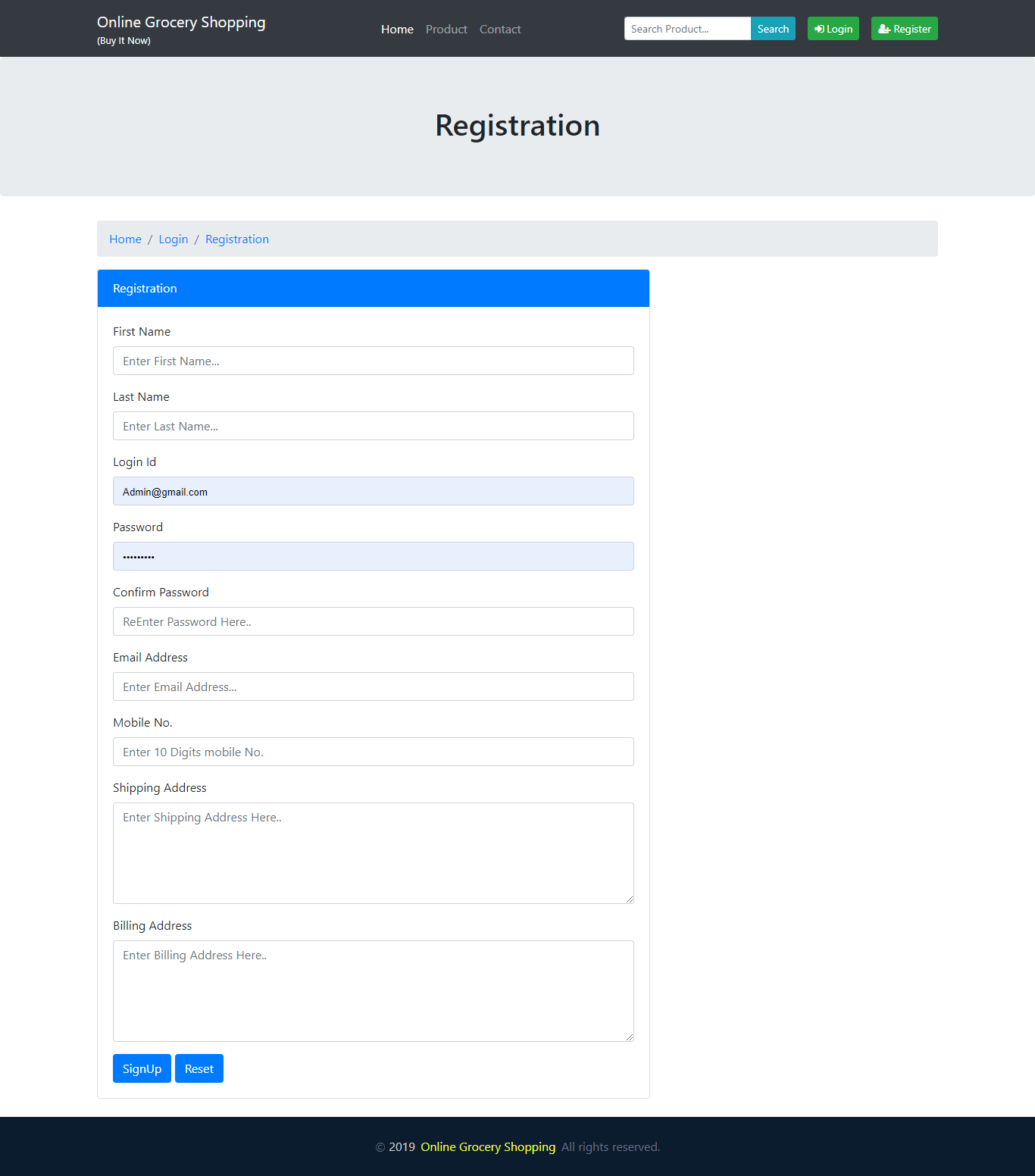
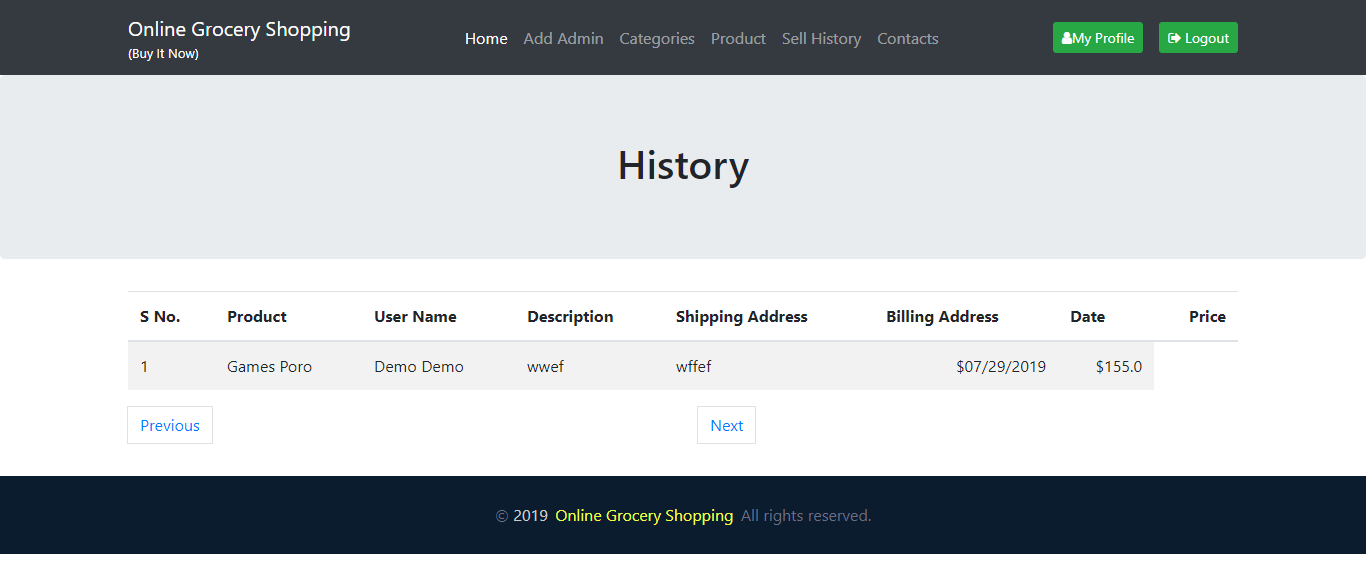
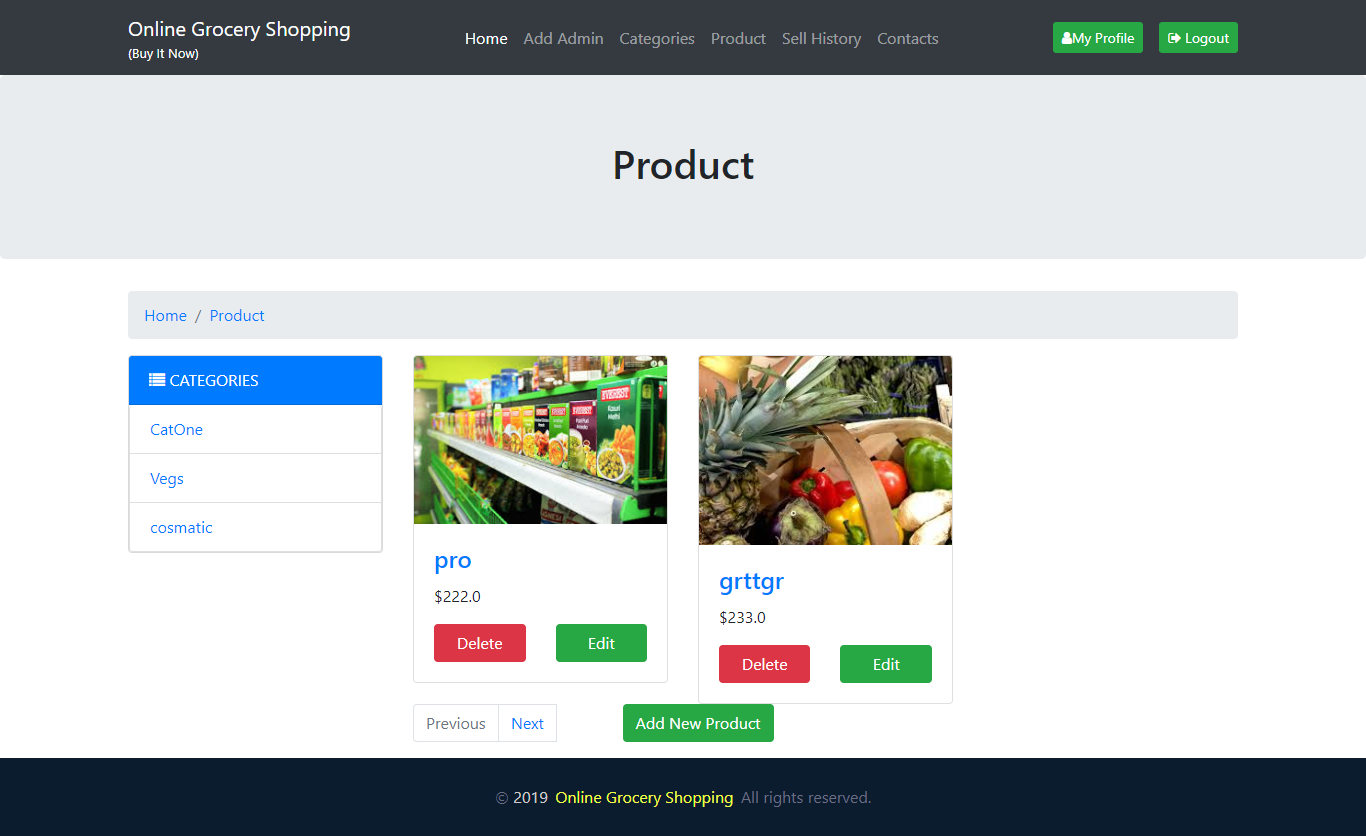
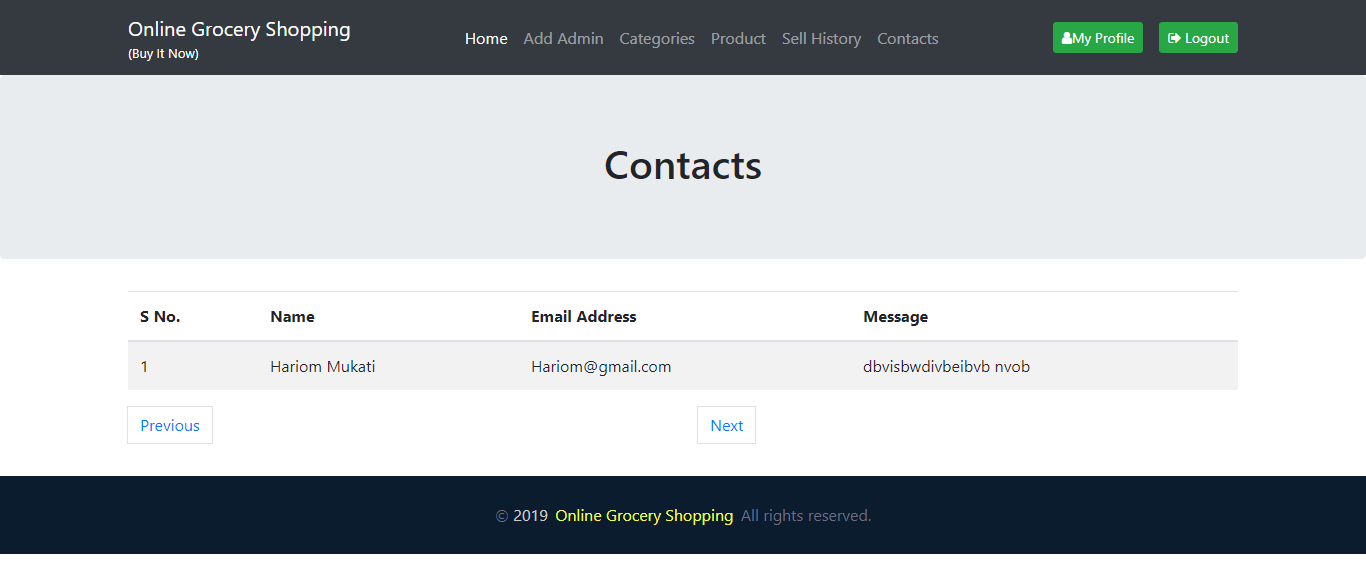
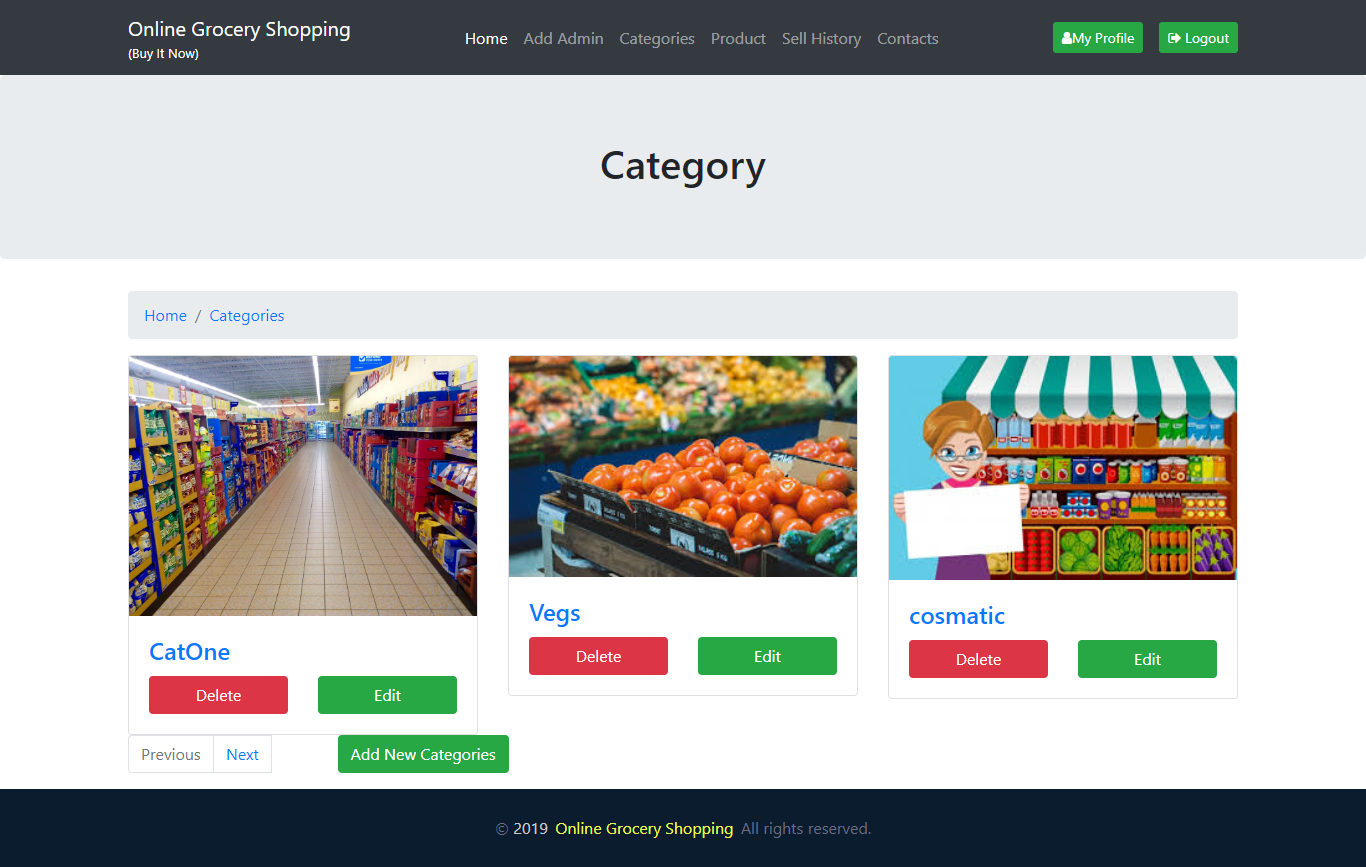
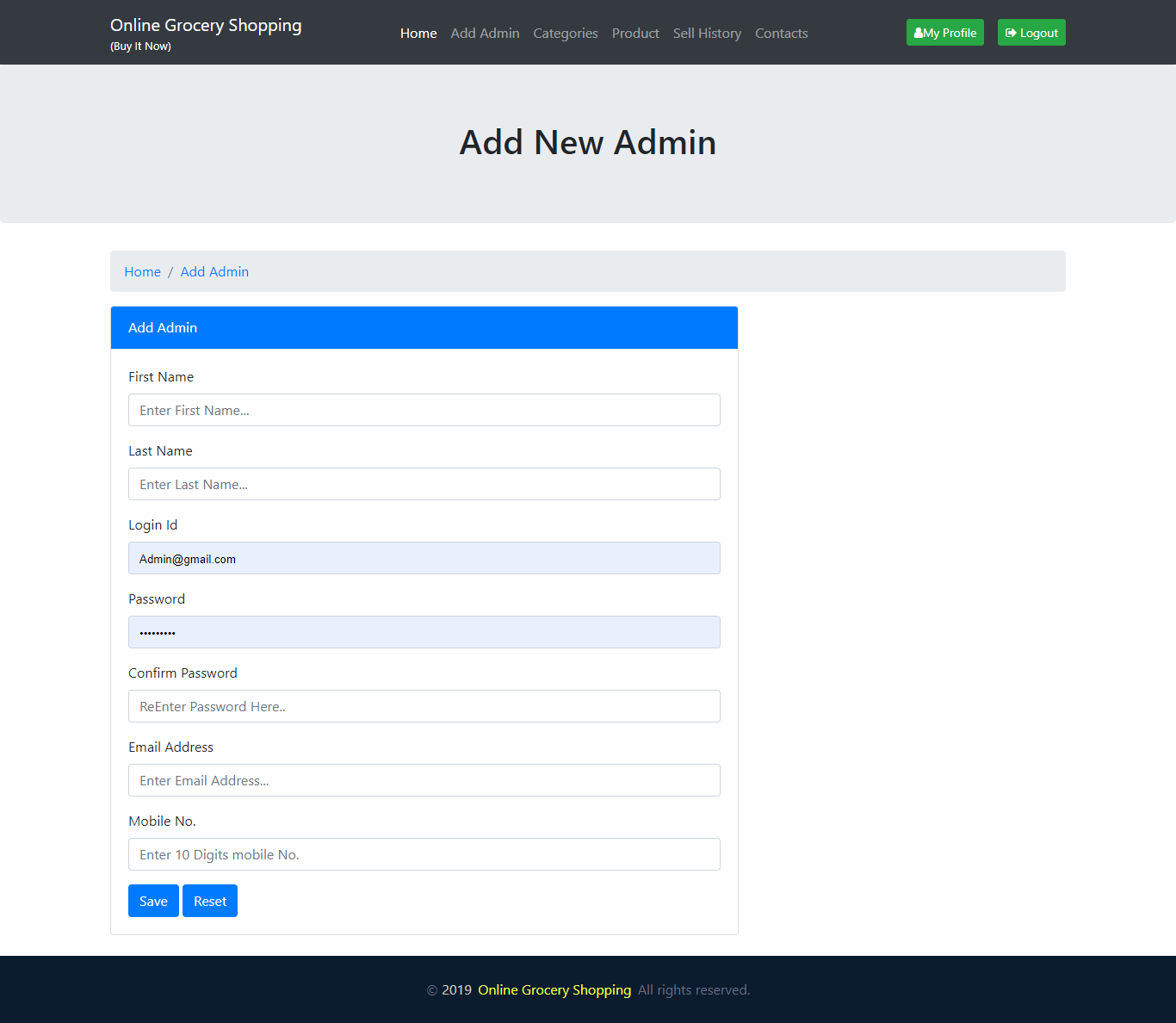
As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

As per **First Normal Form**, there are no repeating or duplicate fields in our database and each cell contains only a single value. For example:

**Screen shot**







**Source code**

package onlinegroceryshopping.bean;

public class ProductBean extends BaseBean {

private long categoryId;

private String categoryName;

private long productCode;

private String name;

private String description;

private double price;

private String image;

public long getCategoryId() {

return categoryId;

}

public void setCategoryId(long categoryId) {

this.categoryId = categoryId;

}

public String getCategoryName() {

return categoryName;

}

public void setCategoryName(String categoryName) {

this.categoryName = categoryName;

}

public long getProductCode() {

return productCode;

}

public void setProductCode(long productCode) {

this.productCode = productCode;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public String getImage() {

return image;

}

public void setImage(String image) {

this.image = image;

}

@Override

public String getKey() {

// TODO Auto-generated method stub

return id+"";

}

@Override

public String getValue() {

// TODO Auto-generated method stub

return name;

}

}

package onlinegroceryshopping.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import org.apache.log4j.Logger;

import onlinegroceryshopping.bean.CategoryBean;

import onlinegroceryshopping.bean.ProductBean;

import onlinegroceryshopping.exception.ApplicationException;

import onlinegroceryshopping.exception.DatabaseException;

import onlinegroceryshopping.exception.DuplicateRecordException;

import onlinegroceryshopping.util.JDBCDataSource;

public class ProductModel {

private static Logger log = Logger.getLogger(ProductModel.class);

public Integer nextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM SO\_PRODUCT");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting Next PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public Integer nextProductCode() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(productCode) FROM SO\_PRODUCT");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting Next Product Code");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

if (pk > 0) {

return pk + 1;

} else {

return 100101;

}

}

public ProductBean findByName(String name) throws ApplicationException {

log.debug("Model findBy Name Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM SO\_PRODUCT WHERE NAME=?");

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, name);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ProductBean();

bean.setId(rs.getLong(1));

bean.setCategoryId(rs.getLong(2));

bean.setCategoryName(rs.getString(3));

bean.setProductCode(rs.getLong(4));

bean.setName(rs.getString(5));

bean.setDescription(rs.getString(6));

bean.setPrice(rs.getDouble(7));

bean.setImage(rs.getString(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting Product by Fined By Name");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findBy Name End");

return bean;

}

public ProductBean findByCategoryAndName(long catId, String name) throws ApplicationException {

log.debug("Model findBy CategoryIdAndName Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM SO\_PRODUCT WHERE categoryId=? And Name=?");

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1,catId);

pstmt.setString(2, name);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ProductBean();

bean.setId(rs.getLong(1));

bean.setCategoryId(rs.getLong(2));

bean.setCategoryName(rs.getString(3));

bean.setProductCode(rs.getLong(4));

bean.setName(rs.getString(5));

bean.setDescription(rs.getString(6));

bean.setPrice(rs.getDouble(7));

bean.setImage(rs.getString(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting Product by Fined By CategoryId And Name");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findBy CategoryIdAndName End");

return bean;

}

public ProductBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM SO\_PRODUCT WHERE ID=?");

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ProductBean();

bean.setId(rs.getLong(1));

bean.setCategoryId(rs.getLong(2));

bean.setCategoryName(rs.getString(3));

bean.setProductCode(rs.getLong(4));

bean.setName(rs.getString(5));

bean.setDescription(rs.getString(6));

bean.setPrice(rs.getDouble(7));

bean.setImage(rs.getString(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public ProductBean findByProductCode(long pk) throws ApplicationException {

log.debug("Model findByProductCode Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM SO\_PRODUCT WHERE ProductCode=?");

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ProductBean();

bean.setId(rs.getLong(1));

bean.setCategoryId(rs.getLong(2));

bean.setCategoryName(rs.getString(3));

bean.setProductCode(rs.getLong(4));

bean.setName(rs.getString(5));

bean.setDescription(rs.getString(6));

bean.setPrice(rs.getDouble(7));

bean.setImage(rs.getString(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting Product by Product Code");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByProductCode End");

return bean;

}

public long add(ProductBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model add Started");

Connection conn = null;

int pk = 0;

int NextProductCode = 0;

ProductBean duplicataProduct = findByCategoryAndName(bean.getCategoryId(),bean.getName());

// Check if create Product already exist

if (duplicataProduct != null) {

throw new DuplicateRecordException("Product Is Already Exist This Category");

}

CategoryModel cModel=new CategoryModel();

CategoryBean cBean= cModel.findByPK(bean.getCategoryId());

bean.setCategoryName(cBean.getName());

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

NextProductCode=nextProductCode();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO SO\_PRODUCT VALUES(?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setLong(2,bean.getCategoryId());

pstmt.setString(3,bean.getCategoryName());

pstmt.setLong(4,NextProductCode);

pstmt.setString(5, bean.getName());

pstmt.setString(6, bean.getDescription());

pstmt.setDouble(7,bean.getPrice());

pstmt.setString(8,bean.getImage());

pstmt.setString(9, bean.getCreatedBy());

pstmt.setString(10, bean.getModifiedBy());

pstmt.setTimestamp(11, bean.getCreatedDatetime());

pstmt.setTimestamp(12, bean.getModifiedDatetime());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add Product");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model add End");

return pk;

}

public void delete(ProductBean bean) throws ApplicationException {

log.debug("Model delete Started");

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn

.prepareStatement("DELETE FROM SO\_PRODUCT WHERE ID=?");

pstmt.setLong(1, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

// log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException(

"Exception : Delete rollback exception "

+ ex.getMessage());

}

throw new ApplicationException(

"Exception : Exception in delete Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model delete Started");

}

public List search(ProductBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

/\*\*

\* Search PRODUCT with pagination

\*

\* @return list : List of Product

\* @param bean

\* : Search Parameters

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\*

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List search(ProductBean bean, int pageNo, int pageSize)

throws ApplicationException {

log.debug("Model search Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM SO\_PRODUCT WHERE 1=1");

if (bean != null) {

if (bean.getId() > 0) {

sql.append(" AND id = " + bean.getId());

}

if (bean.getProductCode() > 0) {

sql.append(" AND ProductCode = " + bean.getProductCode());

}

if (bean.getCategoryId() > 0) {

sql.append(" AND CategoryId = " + bean.getCategoryId());

}

if (bean.getName() != null && bean.getName().length() > 0) {

sql.append(" AND NAME LIKE '" + bean.getName() + "%'");

}

if (bean.getCategoryName() != null && bean.getCategoryName().length() > 0) {

sql.append(" AND CategoryName LIKE '" + bean.getCategoryName() + "%'");

}

if (bean.getDescription() != null

&& bean.getDescription().length() > 0) {

sql.append(" AND DESCRIPTION LIKE '" + bean.getDescription()

+ "%'");

}

}

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" Limit " + pageNo + ", " + pageSize);

// sql.append(" limit " + pageNo + "," + pageSize);

}

ArrayList list = new ArrayList();

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new ProductBean();

bean.setId(rs.getLong(1));

bean.setCategoryId(rs.getLong(2));

bean.setCategoryName(rs.getString(3));

bean.setProductCode(rs.getLong(4));

bean.setName(rs.getString(5));

bean.setDescription(rs.getString(6));

bean.setPrice(rs.getDouble(7));

bean.setImage(rs.getString(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

list.add(bean);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException(

"Exception : Exception in search Product");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

public List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of Product with pagination

\*

\* @return list : List of Product

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List list(int pageNo, int pageSize) throws ApplicationException {

log.debug("Model list Started");

ArrayList list = new ArrayList();

StringBuffer sql = new StringBuffer("select \* from SO\_PRODUCT");

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" limit " + pageNo + "," + pageSize);

}

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

ProductBean bean = new ProductBean();

bean.setId(rs.getLong(1));

bean.setCategoryId(rs.getLong(2));

bean.setCategoryName(rs.getString(3));

bean.setProductCode(rs.getLong(4));

bean.setName(rs.getString(5));

bean.setDescription(rs.getString(6));

bean.setPrice(rs.getDouble(7));

bean.setImage(rs.getString(8));

bean.setCreatedBy(rs.getString(9));

bean.setModifiedBy(rs.getString(10));

bean.setCreatedDatetime(rs.getTimestamp(11));

bean.setModifiedDatetime(rs.getTimestamp(12));

list.add(bean);

}

rs.close();

} catch (Exception e) {

// log.error("Database Exception..", e);

throw new ApplicationException(

"Exception : Exception in getting list of Product");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

/\*\*

\* Update a Category

\*

\* @param bean

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public void update(ProductBean bean) throws ApplicationException,

DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

ProductBean duplicataProduct = findByCategoryAndName(bean.getCategoryId(),bean.getName());

// Check if updated Role already exist

if (duplicataProduct != null && duplicataProduct.getId() != bean.getId()) {

throw new DuplicateRecordException("Product Is Already Exist This Category");

}

CategoryModel cModel=new CategoryModel();

CategoryBean cBean= cModel.findByPK(bean.getCategoryId());

bean.setCategoryName(cBean.getName());

ProductBean pBean=findByPK(bean.getId());

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn

.prepareStatement("UPDATE SO\_PRODUCT SET CATEGORYID=?,CATEGORYNAME=?,PRODUCTCODE=?, NAME=?,DESCRIPTION=?,PRICE=?,IMAGE=?,CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");

pstmt.setLong(1,bean.getCategoryId());

pstmt.setString(2,bean.getCategoryName());

pstmt.setLong(3,pBean.getProductCode());

pstmt.setString(4, bean.getName());

pstmt.setString(5, bean.getDescription());

pstmt.setDouble(6,bean.getPrice());

pstmt.setString(7,bean.getImage());

pstmt.setString(8, bean.getCreatedBy());

pstmt.setString(9, bean.getModifiedBy());

pstmt.setTimestamp(10, bean.getCreatedDatetime());

pstmt.setTimestamp(11, bean.getModifiedDatetime());

pstmt.setLong(12, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException(

"Exception : Delete rollback exception "

+ ex.getMessage());

}

throw new ApplicationException("Exception in updating Product ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

}

package onlinegroceryshopping.controller;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;

import onlinegroceryshopping.bean.BaseBean;

import onlinegroceryshopping.bean.CategoryBean;

import onlinegroceryshopping.bean.ProductBean;

import onlinegroceryshopping.exception.ApplicationException;

import onlinegroceryshopping.model.CategoryModel;

import onlinegroceryshopping.model.ProductModel;

import onlinegroceryshopping.util.DataUtility;

import onlinegroceryshopping.util.PropertyReader;

import onlinegroceryshopping.util.ServletUtility;

/\*\*

\* Servlet implementation class ProductListCtl

\*/

@WebServlet(name = "ProductListCtl", urlPatterns = { "/ctl/adminPortal/prod/prodList" })

public class ProductListCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

private static Logger log = Logger.getLogger(ProductListCtl.class);

/\*\*

\* Populates bean object from request parameters

\*

\* @param request

\* @return

\*/

@Override

protected BaseBean populateBean(HttpServletRequest request) {

log.debug("ProductListCtl populateBean method start");

ProductBean bean = new ProductBean();

bean.setName(DataUtility.getString(request.getParameter("name")));

log.debug("ProductListCtl populateBean method end");

return bean;

}

/\*\*

\* Contains Display logics

\*/

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("ProductListCtl doGet method start");

List list = null;

int pageNo = 1;

int pageSize = DataUtility.getInt(PropertyReader.getValue("page.size"));

long serProId=DataUtility.getLong(request.getParameter("proId"));

ProductModel model = new ProductModel();

ProductBean bean = (ProductBean) populateBean(request);

long prId=DataUtility.getLong(request.getParameter("prdId"));

try {

if(prId>0) {

ProductBean pBean=new ProductBean();

pBean.setId(prId);

model.delete(pBean);

}

if(serProId>0) {

bean.setCategoryId(serProId);

}

list = model.search(bean, pageNo, pageSize);

if (list == null || list.size() == 0) {

ServletUtility.setErrorMessage("No Record Found", request);

}

ServletUtility.setList(list, request);

request.setAttribute("size",model.search(bean).size());

ServletUtility.setPageNo(pageNo, request);

ServletUtility.setPageSize(pageSize, request);

ServletUtility.forward(getView(), request, response);

} catch (ApplicationException e) {

ServletUtility.handleException(e, request, response);

e.printStackTrace();

return;

}

log.debug("ProductListCtl doGet method end");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("ProductListCtl doPost method start");

List list = null;

int pageNo = DataUtility.getInt(request.getParameter("pageNo"));

int pageSize = DataUtility.getInt(request.getParameter("pageSize"));

pageNo = (pageNo == 0) ? 1 : pageNo;

pageSize = (pageSize == 0) ? DataUtility.getInt(PropertyReader.getValue("page.size")) : pageSize;

ProductBean bean = (ProductBean) populateBean(request);

ProductModel model = new ProductModel();

String[] ids = request.getParameterValues("ids");

String op = DataUtility.getString(request.getParameter("operation"));

if (OP\_SEARCH.equalsIgnoreCase(op) || OP\_NEXT.equalsIgnoreCase(op) || OP\_PREVIOUS.equalsIgnoreCase(op)) {

if (OP\_SEARCH.equalsIgnoreCase(op)) {

pageNo = 1;

} else if (OP\_NEXT.equalsIgnoreCase(op)) {

pageNo++;

} else if (OP\_PREVIOUS.equalsIgnoreCase(op) && pageNo > 1) {

pageNo--;

}

} else if (OP\_NEW.equalsIgnoreCase(op)) {

ServletUtility.redirect(SOTGView.PRODUCT\_CTL, request, response);

return;

} else if (OP\_DELETE.equalsIgnoreCase(op)) {

pageNo = 1;

if (ids != null && ids.length > 0) {

ProductBean deletebean = new ProductBean();

for (String id : ids) {

deletebean.setId(DataUtility.getInt(id));

try {

model.delete(deletebean);

} catch (ApplicationException e) {

ServletUtility.handleException(e, request, response);

e.printStackTrace();

return;

}

}

ServletUtility.setSuccessMessage("Data Deleted Successfully", request);

} else {

ServletUtility.setErrorMessage("Select at least one record", request);

}

} else if (OP\_RESET.equalsIgnoreCase(op)) {

ServletUtility.redirect(SOTGView.PRODUCT\_LIST\_CTL, request, response);

return;

}

try {

list = model.search(bean, pageNo, pageSize);

if (list == null || list.size() == 0) {

ServletUtility.setErrorMessage("NO Record Found", request);

}

ServletUtility.setList(list, request);

request.setAttribute("size",model.search(bean).size());

ServletUtility.setPageNo(pageNo, request);

ServletUtility.setPageSize(pageSize, request);

ServletUtility.forward(getView(), request, response);

} catch (ApplicationException e) {

ServletUtility.handleException(e, request, response);

e.printStackTrace();

return;

}

log.debug("WelcomeListCtl doPost method end");

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return SOTGView.PRODUCT\_LIST\_VIEW;

}

}

<%@page import="onlinegroceryshopping.controller.ProductListCtl"%>

<%@page import="onlinegroceryshopping.bean.ProductBean"%>

<%@page import="onlinegroceryshopping.util.ServletUtility"%>

<%@page import="java.util.Iterator"%>

<%@page import="java.util.List"%>

<%@page import="onlinegroceryshopping.bean.CategoryBean"%>

<%@page import="onlinegroceryshopping.model.CategoryModel"%>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Product</title>

</head>

<body>

<%@ include file="Header.jsp" %>

<section class="jumbotron text-center">

<div class="container">

<h1 class="jumbotron-heading">Product</h1>

</div>

</section>

<div class="container">

<div class="row">

<div class="col">

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="<%=SOTGView.HOME\_CTL%>">Home</a></li>

<li class="breadcrumb-item active"><a href="<%=SOTGView.PRODUCT\_LIST\_CTL%>">Product</a></li>

</ol>

</nav>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-12 col-sm-3">

<div class="card bg-light mb-3">

<div class="card-header bg-primary text-white text-uppercase"><i class="fa fa-list"></i> Categories</div>

<ul class="list-group category\_block">

<% CategoryModel cModel=new CategoryModel();

CategoryBean cBean=null;

List cList=cModel.list();

Iterator<CategoryBean> cit=cList.iterator();

while(cit.hasNext()){

cBean=cit.next();

%>

<li class="list-group-item"><a href="<%=SOTGView.PRODUCT\_LIST\_CTL%>?Category=<%=cBean.getName()%>&cId=<%=cBean.getId()%>"><%=cBean.getName()%></a></li>

<%} %>

</ul>

</div>

</div>

<div class="col">

<div class="row">

<%

int pageNo = ServletUtility.getPageNo(request);

int pageSize = ServletUtility.getPageSize(request);

int size=(int)request.getAttribute("size");

int index = ((pageNo - 1) \* pageSize) + 1;

ProductBean bean=null;

List list = ServletUtility.getList(request);

Iterator<ProductBean> it = list.iterator();

while (it.hasNext()) {

bean = it.next();

%>

<div class="col-12 col-md-6 col-lg-4">

<div class="card">

<a href="<%=SOTGView.PRODUCT\_DETAIL\_CTL%>?product=<%=bean.getName()%>&proId=<%=bean.getId()%>" ><img class="card-img-top" src="<%=SOTGView.APP\_CONTEXT%>/images/<%=bean.getImage()%>" alt="Card image cap"></a>

<div class="card-body">

<h4 class="card-title"><a href="<%=SOTGView.PRODUCT\_DETAIL\_CTL%>?product=<%=bean.getName()%>&proId=<%=bean.getId()%>" title="View Product"><%=bean.getName()%></a></h4>

<p class="bloc\_left\_price">$<%=bean.getPrice()%></p>

<div class="row">

<div class="col">

<a href="<%=SOTGView.PRODUCT\_LIST\_CTL%>?prdId=<%=bean.getId()%>" class="btn btn-danger btn-block">Delete</a>

</div>

<div class="col">

<a href="<%=SOTGView.PRODUCT\_CTL%>?id=<%=bean.getId()%>" class="btn btn-success btn-block">Edit</a>

</div>

</div>

</div>

</div>

</div>

<%} %>

<div class="col-12">

<nav aria-label="...">

<ul class="pagination">

<li class="page-item disabled">

<input type="submit" name="operation" class="page-link"

value="<%=ProductListCtl.OP\_PREVIOUS%>"

<%=(pageNo == 1) ? "disabled" : ""%>>

</li>

<li class="page-item">

<input type="submit" name="operation" class="page-link"

value="<%=ProductListCtl.OP\_NEXT%>"

<%=((list.size() < pageSize) || size==pageNo\*pageSize) ? "disabled" : ""%>>

</li>

<li class="page-item">

</li>

&nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;

<li class="page-item">

<a class="btn btn-success btn-block" href="<%=SOTGView.PRODUCT\_CTL%>">Add New Product</a>

</li>

</ul>

</nav>

</div>

</div>

</div>

</div>

</div>

<%@ include file="Footer.jsp" %>

</body>

</html>

## Testing

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

###### Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

###### Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Testing goes side by side with the implementation that is aimed at ensuring that the system works accurately and efficiently before the live operation is performed .The common view of testing held by the user is process of executing a program with explicit intention of handling errors. The application which has been developed has to be tested to prove its validity. Testing is considered to be the least creative phase of the whole cycle of system design. In the real sense it is the phase, which helps to bring out the creativity of the other phases, and makes itshine.

The Smart Online Fashion Store was tested using the following two techniques of application testing:

###### Unit Testing:

* In the line of strategy the entire individuals function and modules were put to test independently
* By following this strategy all the errors in coding were identified and corrected.
* This method was applied in combination with the White Box and Black Box testing
* Technique to find errors in eachmodule.
* The effort of specific combination of data on system operation wastested.
* The following were the testes carried out for Graphical User Interface(GUI).
* It was seen that the pages opens properly based on related menu based commands.
* It was tested whether all relevant menus, buttons, icons and other controls are available and properlydisplayed.

###### System Testing

We use this testing method. System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

###### Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

###### Multi-user System Testing

Database Locking Schemes: Whenever more than one person is accessing a record/s some type of process must be used to prevent the outer users from attempting to update the same record at the same time. This process is a locking scheme. In its simplest form, a locking scheme allows only one user at a time to update information in the database.

## Chapter 7

##### FutureWork

The project has been developed in a very short period of time and all efforts have been taken so that this project is very efficient in its execution there still exists some scope of improvement in our project. The following lists some of the enhancement that can be added incorporate into theproject.

Application of the project can be done more attractively. Database management and all maintenance module can be updated which helps the administrator. More security measures can betaken.

There are also few features which can be integrated with this system to make it more flexible. Below list shows the future points to be consider:

* + - Real-time Chat BOT option for members and trainer, so that members can directly enquiry theirs trainer on any time through the ChatBOT.
    - Automated Fitness suggestion by enquiring the condition of thehealth.
    - Real time Claim Processing Bot.
    - Video conversation option for trainers andmembers.
    - Online payment through facerecognition.
    - Barcode generation for membership card and using this, members can take entry to Online Fashion Store.
    - Finger print matching for taking entry to Grocery Online Grocery Shopping .

.

##### Conclusions

The “Online Grocery Shopping” is successfully designed and developed to fulfilling the necessary requirements, as identified in the requirements analysis phase, such as the system is very much user friendly, form level validation and field level validation are performing very efficiently. The old manual system was suffering from a series of drawbacks. The present project has been developed to meet the aspirations indicated in the modern age.