# Online GroceryShopping

**Introduction**

Online GroceryShoppingis developed in java, which can keep track of all your business activity in Online Groceryshop from small segments to large and very large segments.

Online GroceryShoppingsells various types of Online GroceryStore 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 theProject

Our proposed “Online GroceryShopping” is for those who run aOnline GroceryShoppingbusiness. Before doing anything we did a decent research on major difficulties for Online GroceryShopping. 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 GroceryShoppingrequires 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 GroceryShopping.

##### 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 GroceryShopping.
    - 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.

**Theoretical Background**

We have done a project on Online GroceryShoppingand 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 GroceryShoppingis 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 OnlineGroceryShoppingis 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 GroceryShoppingis 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 GroceryShoppingautomates 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 GroceryShoppingprovides 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 GroceryShoppinghas 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.

**ANALYSIS AND FEASIBILITYOFSTUDY**

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

##### Defining aproblem

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.

### IDENTIFICATION OFNEED

This is the most indispensable phase of the system which is to be developed, In this firstly we have mentioned our need which we want to develop. Here, the need and specification phase of system analysis is done to exactly find out the need and the requirements by the customers, and hence all the requirements is collected by the customers.

### PRELIMINARYINVESTIGATION

To evaluate and to define the problem in hand quickly, the preliminary investigation is carried out, to see if it is worthy of the following study and also it suggests some courses of actions if possible.

Following steps are involved in the preliminary investigation:

* + - The ProblemUnderstanding
    - Determining the project boundaries andconstraints
    - Feasibilitystudy
    - Estimation of the time andcost.
    - Documentation of PreliminaryReport.

### FEASIBILITY OFSTUDY

Feasibility study generally determines the need and solutions considered to accomplish the requirements are practically implementable in the software or not, information such as availability of the resource, estimation of cost for the development of the project and the cost which would be incurred on maintenance of the project is carried out in feasibility study.

There are different types of feasibility:

* + - Technical Feasibility
    - Operational Feasibility
    - Economic Feasibility

##### Operational Feasibility

* + - * This site is operational feasible because in this all users can easily operate access the facilities and module meant for according to the type of user.
      * The well-planned architecture would ensure the optimal utilization of the resources and will be secure for threats.
      * Thus provides easy access to all the users with their registered mail Id and password.

##### Technical Feasibility

Project is technical feasible due to following reasons:

* + - * This site is technical feasible because in this site, technology which is used to develop the site is efficient and is easily upgraded time to time and separated module makes it easy to implement andmaintenance.
      * Technical guarantees of accuracy, reliability, ease of access and the datasecurity.
      * The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles**.**

##### Economical Feasibility

Project is technical feasible due to following reasons:

* + - * The system is economically feasible and based on all freely licensed software. It does not require any additional hardware or software. There is nominal expenditure and economical feasibility forcertain.
      * This can be added to the official website of the college/institution as a module and does not require any separatespace.[8]

### PROJECT PLANNING

It is a process which includes the activities required for the successful completion of the project. Project planning generally prevents obstacles that arise in the project such as non-availability of the resources and it also determines project constraints.

Planning is generally done by the project and senior management team.

Senior Management is responsible for employing team members whereas the project management is responsible for making decisions and planning.

In this system also planning is executed for developing the whole project and meeting the requirements of the user.



**Feasibility:**

This project will be developed on computer, so first check whether the technology is technically available or not. Now a day’s computer interaction with any job becomes common for any kind of job or work.

And because of increasing usage of Computer, Computer is also available with a variety of hardware. Vendors can fulfill any type of hardware requirement. The whole project is developed by some special tools or by using languages and databases, which are also available in a variety.

Preliminary investigation of a system examines the feasibility of a system that is useful to an organization. It is the first phase of system development.

The main objective of this phase is to identify the current deficiencies in the user’s environment and to determine which existing problem are going to be solve in proposed system and also which new function needs to be added in proposed system.

An important outcome of such preliminary investigation is to determine whether the system that will meet all needed requirements.

Thus, three tests are carried out on the system namely operation, technical and economical.

Any project is beneficial if and only satisfies the organization requirement. For any new system setup, it only meets to be communicated and work the other supporting system.

The new system meets all existing operations since it provides right information at a right time to the right user. A Leigh man can easily operate with the system.

Technical Feasibility examines whether the technology needed is available and if it is available then it feasible to carry out all project activities.

The technical needs of a system include:

* The facility to produce outputs in a given time.
* Ability to process large number of transaction at a particular speed.
* Giving response to users under certain conditions.

The technology needed for our system is mainly:

* Latest version of browsers.
* Any operating system.

These technologies are available which helps to carry out the system efficiently.

Economical feasibility of a system examines whether the finance is available for implementing the new system and whether the money spent is recoverable the satisfaction.

The cost involves is in designing and developing a good investment for the organization.

Thus, hardware requirements used for proposed system are very standard. Moreover, by making use of proposed system to carry out the work speedily will increase and also saves the valuable time of an organization.

In the proposed system the finance is highly required for the installation of the software’s which can also be recovered by implementing a better system.

**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, JQuer

**DESIGNANDARCHITECTURE**

##### 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.

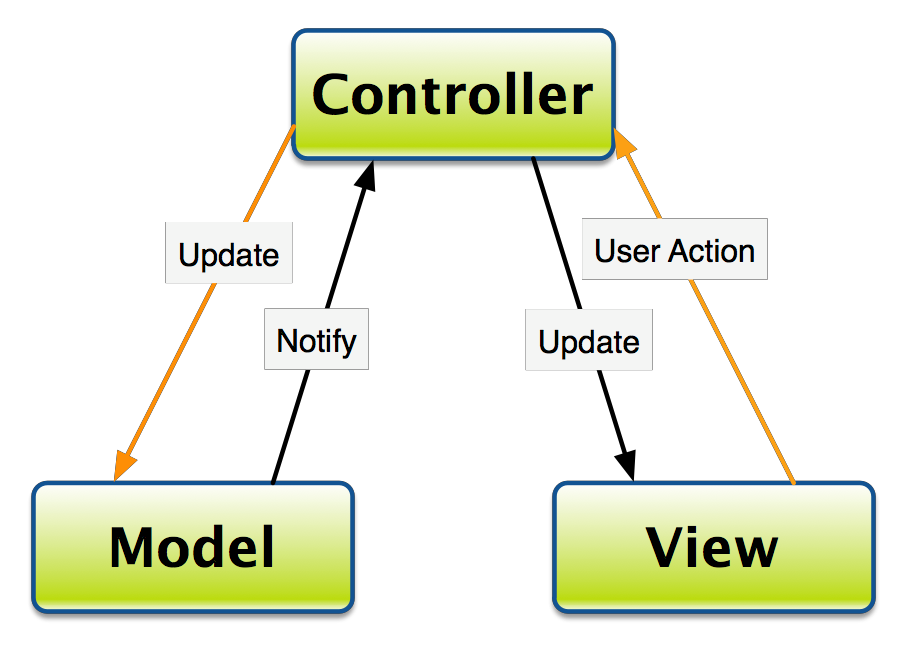


Diagram of A typical collaboration of the MVC components.

##### 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.

### INTRODUCTION

The purpose of System Design is to create a technical solution that satisfies the functional requirements for the system. At this point in the project lifecycle there should be a Functional Specification, written primarily in business terminology, containing a complete description of the operational needs of the various organizational entities that will use the new system.

Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. At Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective.

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements**.**

### DATA FLOWDIAGRAM

In an Information system, the flow of the data around the system is graphically represented by the data flow diagram.

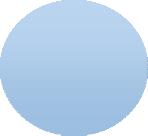
A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of the data and delays in the system. Data flow diagram the central tool and the basis from which other components are developed.

DFDs are the model of the proposed system. They clearly show the requirements on which the new system should be built. Later during the design activity this is taken as the basis for drawing the system’s Structure charts.

##### The various components of DFDs are:

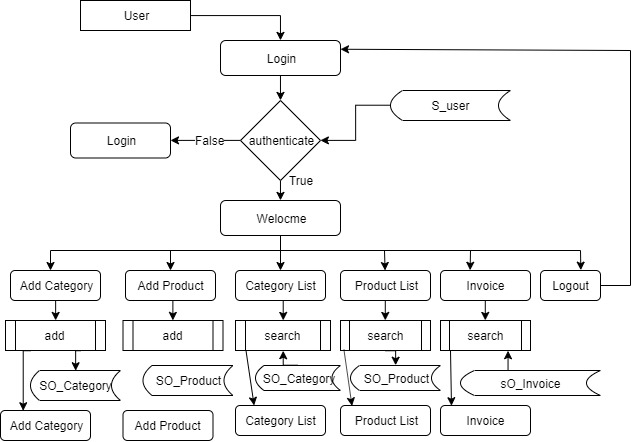
**Dataflow:** Data movement form the source to destination is shown by the arrows.

**Process:** The various activities and the actions performed on the data is represented through circle.

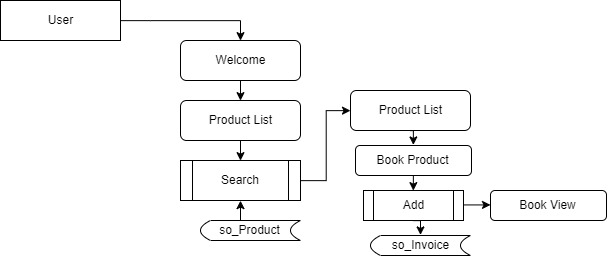


**Entities:** External sources or information of the data is represented by rectangle.

##### 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.

This project will be developed on computer, so first check whether the technology is technically available or not. Now a day’s computer interaction with any job becomes common for any kind of job or work.

And because of increasing usage of Computer, Computer is also available with a variety of hardware. Vendors can fulfill any type of hardware requirement. The whole project is developed by some special tools or by using languages and databases, which are also available in a variety.

Preliminary investigation of a system examines the feasibility of a system that is useful to an organization. It is the first phase of system development.

The main objective of this phase is to identify the current deficiencies in the user’s environment and to determine which existing problem are going to be solve in proposed system and also which new function needs to be added in proposed system.

An important outcome of such preliminary investigation is to determine whether the system that will meet all needed requirements.

Thus, three tests are carried out on the system namely operation, technical and economical.

Any project is beneficial if and only satisfies the organization requirement. For any new system setup, it only meets to be communicated and work the other supporting system.

The new system meets all existing operations since it provides right information at a right time to the right user. A Leigh man can easily operate with the system. Technical Feasibility examines whether the technology needed is available and if it is available then it feasible to carry out all project activities.

The technical needs of a system include:

* The facility to produce outputs in a given time.
* Ability to process large number of transactions at a particular speed.
* Giving response to users under certain conditions.

The technology needed for our system is mainly:

* Latest version of browsers.
* Any operating system.

These technologies are available which helps to carry out the system efficiently. Economic feasibility of a system examines whether the finance is available for implementing the new system and whether the money spent is recoverable the satisfaction. The cost involves is in designing and developing a good investment for the organization. Thus, hardware requirements used for proposed system are very standard. Moreover, by making use of proposed system to carry out the work speedily will increase and also saves the valuable time of an organization.

In the proposed system the finance is highly required for the installation of the software’s which can also be recovered by implementing a better system.

**E R DIAGRAM:**

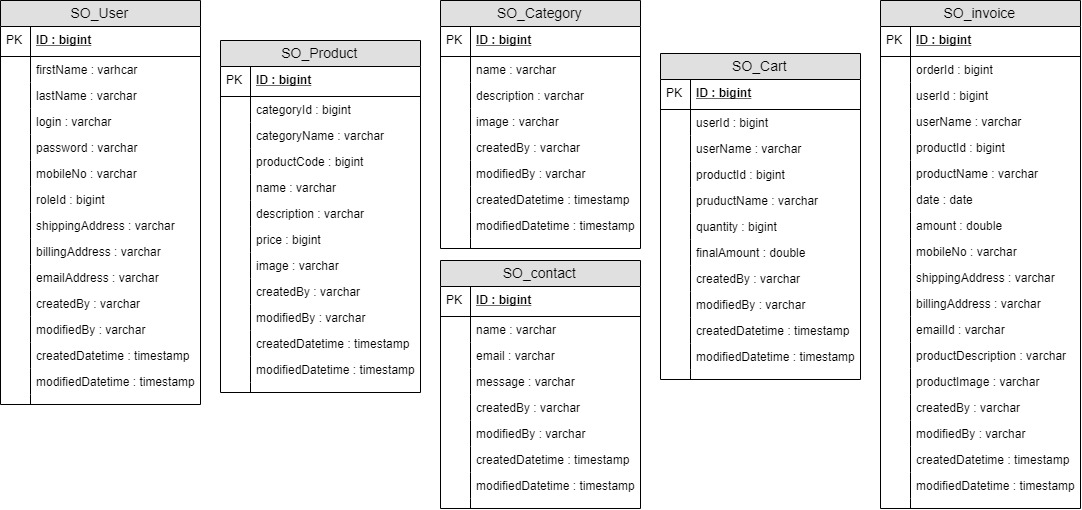
**E-RDIAGRAM**

An E-R model is an abstract way to describe a database. Describing a database usually starts with a relational database, which stores data in tables. Some of the data in these tables point to data in other tables - for instance, your entry in the database could point to several entries for each of the phone numbers that are yours. The ER model would say that you are an entity, and each phone number is an entity, and the relationship between you and the phone numbers is 'has a phone number'. Diagrams created to design these entities and relationships are called entity–relationship diagrams or ER diagrams.

Entity Relationships are three kinds:

1. One-One
2. One-Many
3. Many-Many
4. **One-One :** One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number(B).
5. **One-Many:** One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entityA.
6. **Many-Many:** One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated within

**ER-Diagram for the system**



##### 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:

**Data dictionary**

**Data validation:**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been integrated in the system in almost every area where there is a possibility for the user to commit errors. The system will not recognize invalid data.

Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been integrated where necessary.

The system is designed to be a user friendly one. In other words, the system has been designed to communicate effectively with the user. The system has been designed with pop-up menus.

**Different Type Of validation:**

* Data type validation;
* Range and constraint validation;
* Code and Cross-reference validation; and

Structured validation

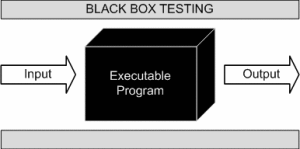
**Implementation and Testing:**

**Black-Box Testing**:

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

This can be following way:

* Input interfacing
* Processing
* Output interfacing



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors.

**White-Box Testing:**

White Box Testing, also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential.

White box testing is testing beyond the user interface and into the nitty-gritty of a system. This method is named so because the software program, in the eyes of the tester, is like a white/ transparent box; inside which one clearly sees.

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.

##### 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.

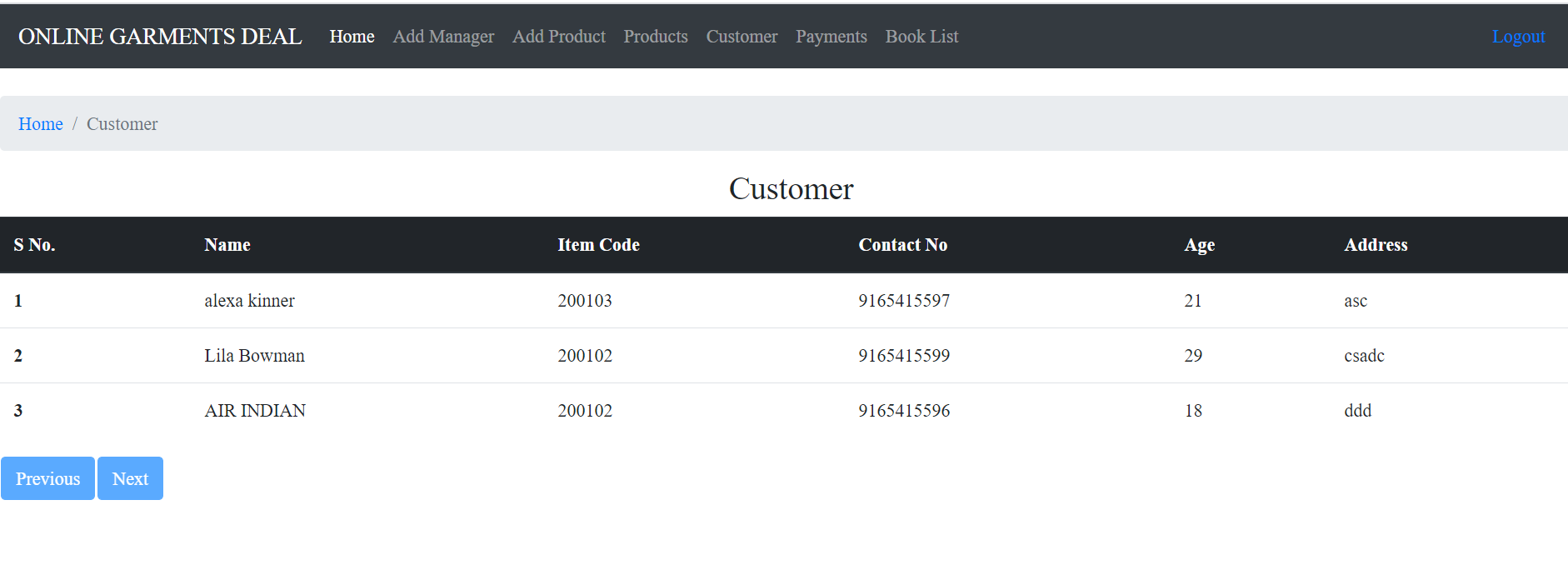
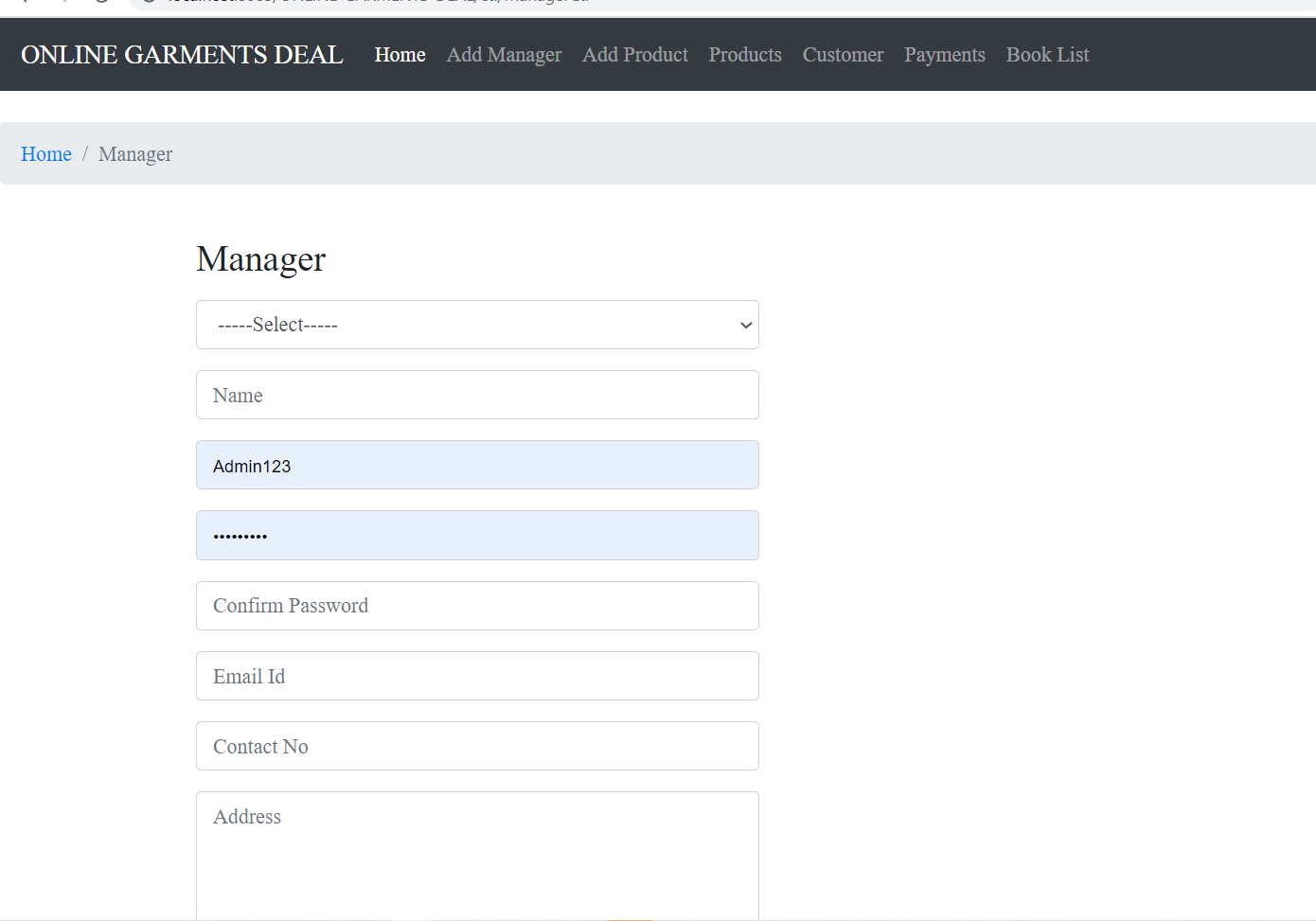
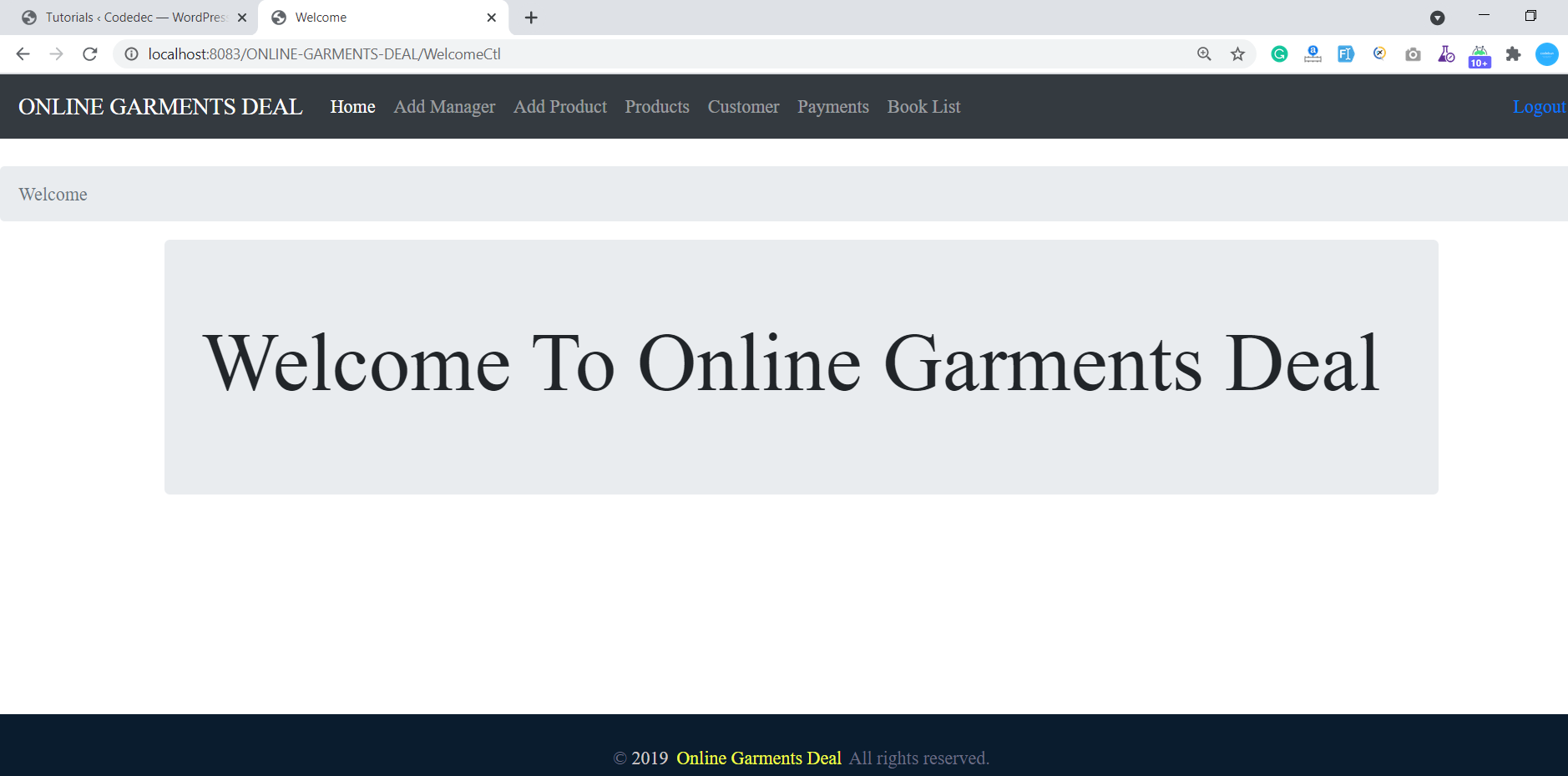
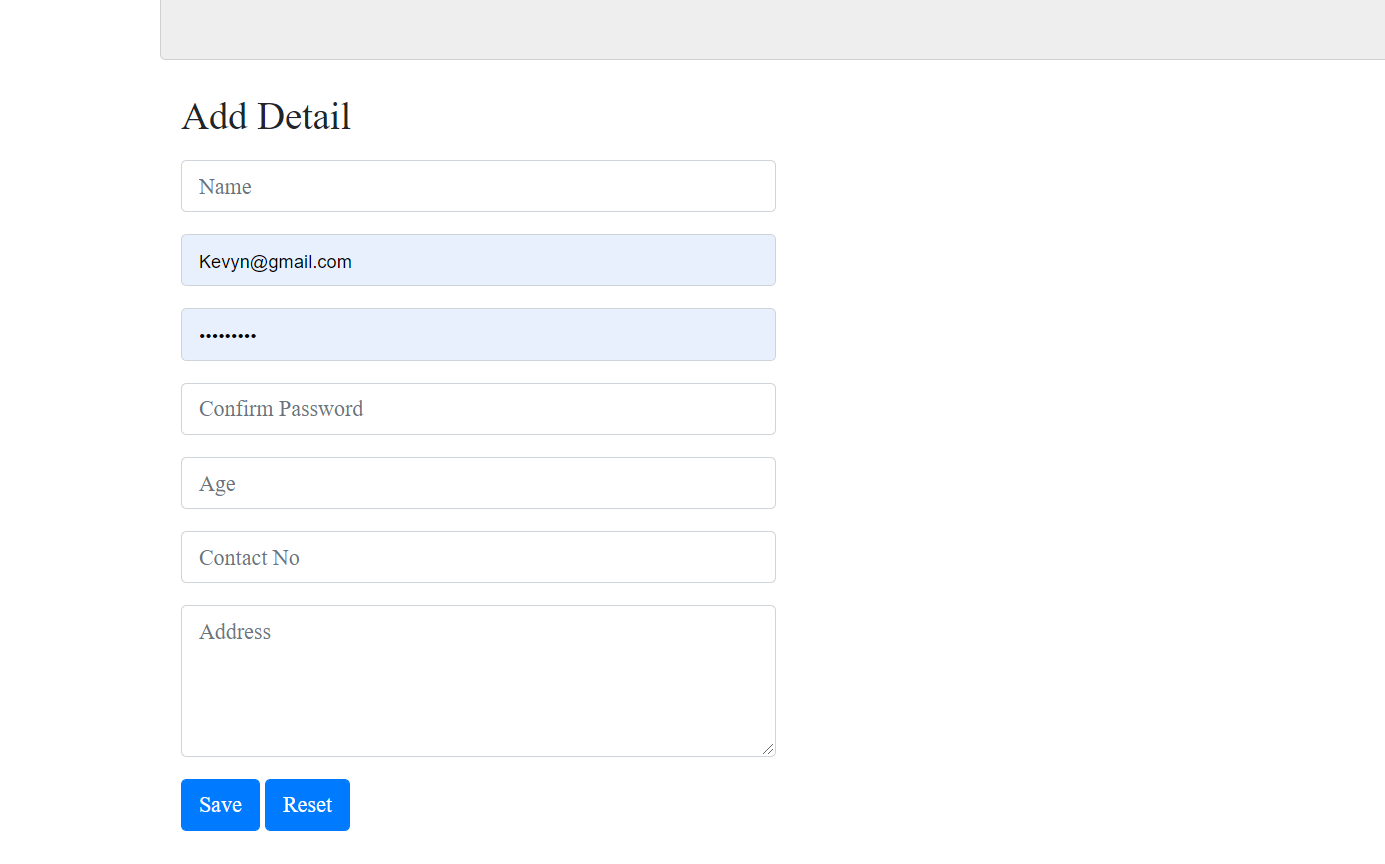
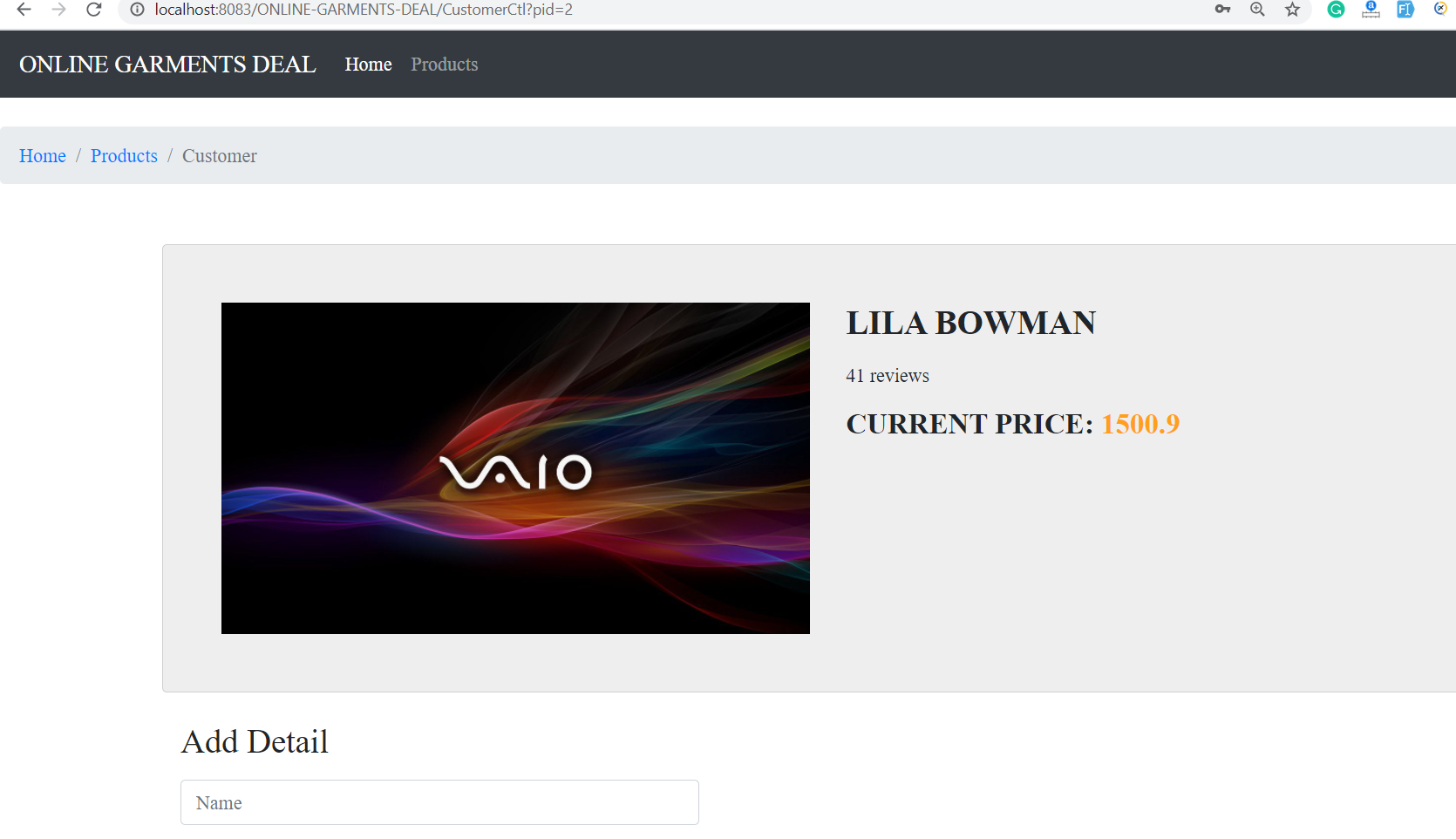
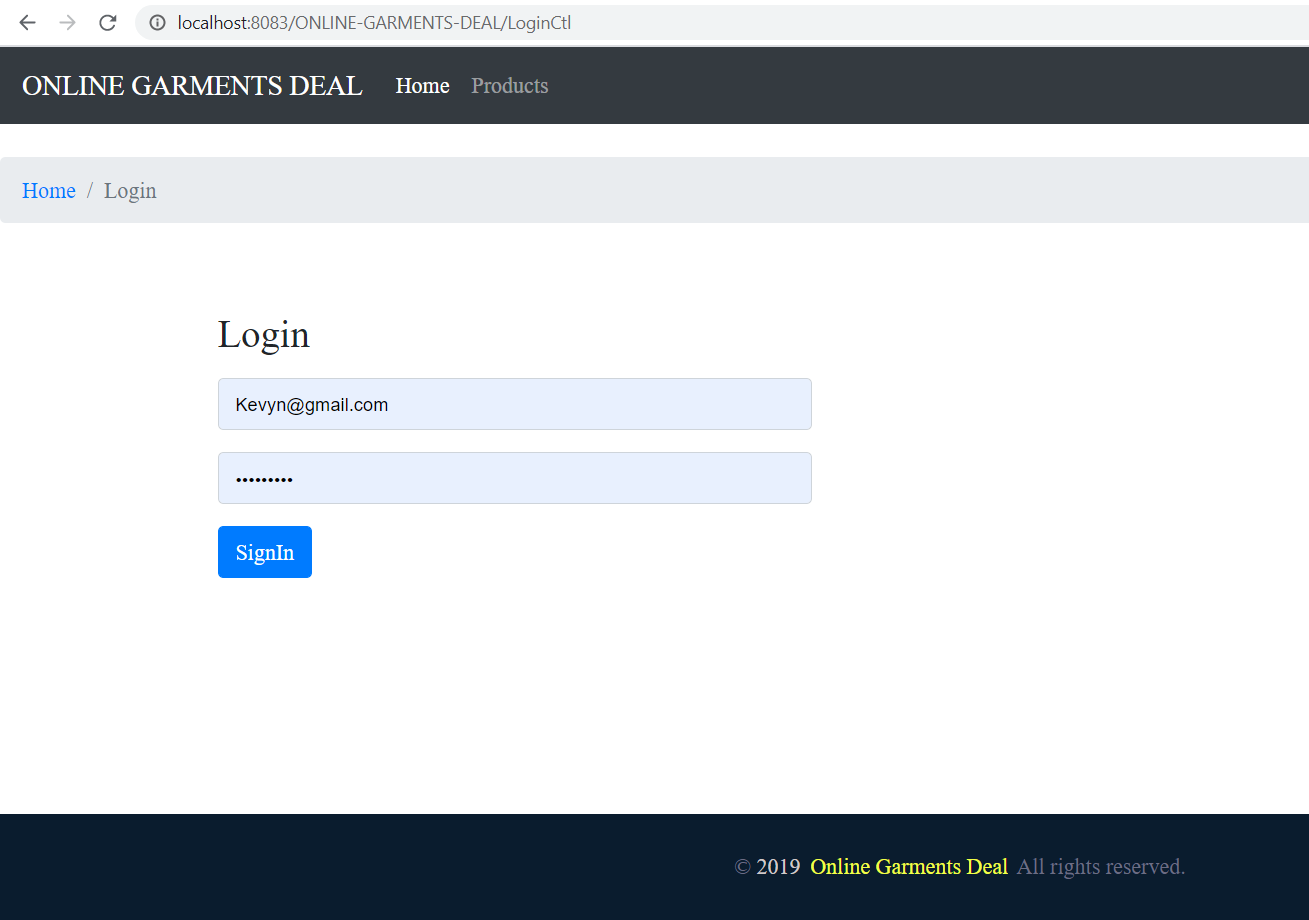
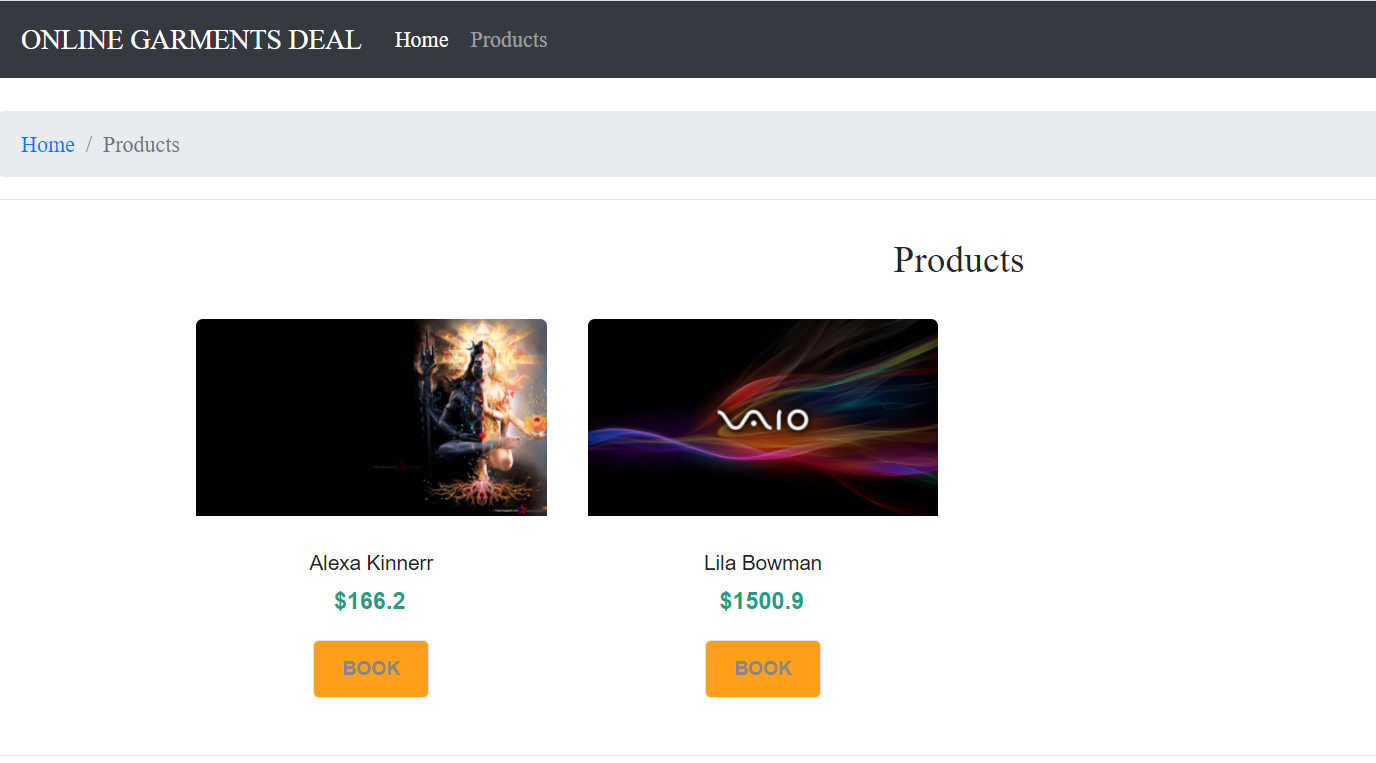
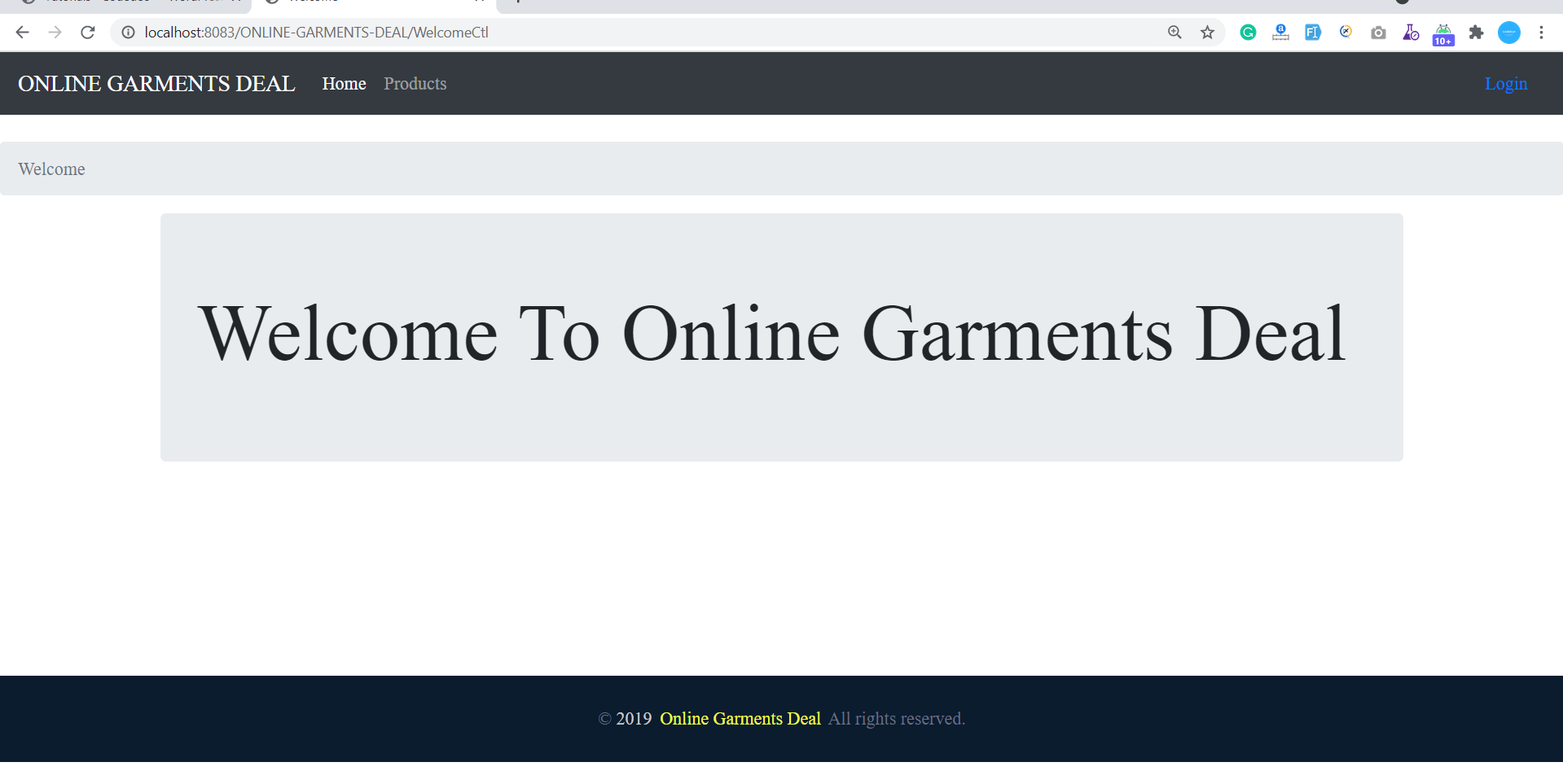
Application of the project can be done more attractively. Database management and all maintenance module can be updated which helps the administrator. More security measures can betaken.

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 trainersandmembers.
    - Online payment through facerecognition.
    - Barcode generation for membership card and using this, members can take entry toOnline Fashion Store.
    - Finger print matching for taking entry to Grocery Online GroceryShopping.

.

**Screen shot**



**Source code**

packageonlinegroceryshopping.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() {

returncategoryId;

}

public void setCategoryId(long categoryId) {

this.categoryId = categoryId;

}

public String getCategoryName() {

returncategoryName;

}

public void setCategoryName(String categoryName) {

this.categoryName = categoryName;

}

public long getProductCode() {

returnproductCode;

}

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;

}

}

packageonlinegroceryshopping.model;

importjava.sql.Connection;

importjava.sql.PreparedStatement;

importjava.sql.ResultSet;

importjava.util.ArrayList;

importjava.util.List;

import org.apache.log4j.Logger;

importonlinegroceryshopping.bean.CategoryBean;

importonlinegroceryshopping.bean.ProductBean;

importonlinegroceryshopping.exception.ApplicationException;

importonlinegroceryshopping.exception.DatabaseException;

importonlinegroceryshopping.exception.DuplicateRecordException;

importonlinegroceryshopping.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;

intpk = 0;

try {

conn = JDBCDataSource.getConnection();

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

ResultSetrs = 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");

returnpk + 1;

}

public Integer nextProductCode() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

intpk = 0;

try {

conn = JDBCDataSource.getConnection();

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

ResultSetrs = 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) {

returnpk + 1;

} else {

return 100101;

}

}

publicProductBeanfindByName(String name) throws ApplicationException {

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

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

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatementpstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, name);

ResultSetrs = 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;

}

publicProductBeanfindByCategoryAndName(long catId, String name) throws ApplicationException {

log.debug("Model findByCategoryIdAndName Started");

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

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatementpstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1,catId);

pstmt.setString(2, name);

ResultSetrs = 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 findByCategoryIdAndName End");

return bean;

}

publicProductBeanfindByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

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

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatementpstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSetrs = 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;

}

publicProductBeanfindByProductCode(long pk) throws ApplicationException {

log.debug("Model findByProductCode Started");

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

ProductBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatementpstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSetrs = 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;

intpk = 0;

intNextProductCode = 0;

ProductBeanduplicataProduct = findByCategoryAndName(bean.getCategoryId(),bean.getName());

// Check if create Product already exist

if (duplicataProduct != null) {

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

}

CategoryModelcModel=new CategoryModel();

CategoryBeancBean= 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

PreparedStatementpstmt = 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");

returnpk;

}

public void delete(ProductBean bean) throws ApplicationException {

log.debug("Model delete Started");

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatementpstmt = 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

\* @parampageNo

\* : Current Page No.

\* @parampageSize

\* : Size of Page

\*

\* @throws DatabaseException

\* @throwsApplicationException

\*/

public List search(ProductBean bean, intpageNo, intpageSize)

throwsApplicationException {

log.debug("Model search Started");

StringBuffersql = 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();

PreparedStatementpstmt = conn.prepareStatement(sql.toString());

ResultSetrs = 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

\* @parampageNo

\* : Current Page No.

\* @parampageSize

\* : Size of Page

\* @throws DatabaseException

\* @throwsApplicationException

\*/

public List list(intpageNo, intpageSize) throws ApplicationException {

log.debug("Model list Started");

ArrayList list = new ArrayList();

StringBuffersql = 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();

PreparedStatementpstmt = conn.prepareStatement(sql.toString());

ResultSetrs = 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

\* @throwsApplicationException

\*/

public void update(ProductBean bean) throws ApplicationException,

DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

ProductBeanduplicataProduct = 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");

}

CategoryModelcModel=new CategoryModel();

CategoryBeancBean= cModel.findByPK(bean.getCategoryId());

bean.setCategoryName(cBean.getName());

ProductBeanpBean=findByPK(bean.getId());

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatementpstmt = 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");

}

}

packageonlinegroceryshopping.controller;

importjava.io.IOException;

importjava.util.List;

importjavax.servlet.ServletException;

importjavax.servlet.annotation.WebServlet;

importjavax.servlet.http.HttpServlet;

importjavax.servlet.http.HttpServletRequest;

importjavax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;

importonlinegroceryshopping.bean.BaseBean;

importonlinegroceryshopping.bean.CategoryBean;

importonlinegroceryshopping.bean.ProductBean;

importonlinegroceryshopping.exception.ApplicationException;

importonlinegroceryshopping.model.CategoryModel;

importonlinegroceryshopping.model.ProductModel;

importonlinegroceryshopping.util.DataUtility;

importonlinegroceryshopping.util.PropertyReader;

importonlinegroceryshopping.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

protectedBaseBeanpopulateBean(HttpServletRequest request) {

log.debug("ProductListCtlpopulateBean method start");

ProductBean bean = new ProductBean();

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

log.debug("ProductListCtlpopulateBean 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("ProductListCtldoGet method start");

List list = null;

intpageNo = 1;

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

longserProId=DataUtility.getLong(request.getParameter("proId"));

ProductModel model = new ProductModel();

ProductBean bean = (ProductBean) populateBean(request);

longprId=DataUtility.getLong(request.getParameter("prdId"));

try {

if(prId>0) {

ProductBeanpBean=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("ProductListCtldoGet method end");

}

/\*\*

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

\*/

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

log.debug("ProductListCtldoPost method start");

List list = null;

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

intpageSize = 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) {

ProductBeandeletebean = 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("WelcomeListCtldoPost method end");

}

@Override

protected String getView() {

// TODO Auto-generated method stub

returnSOTGView.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="fafa-list"></i> Categories</div>

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

<% CategoryModelcModel=new CategoryModel();

CategoryBeancBean=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">

<%

intpageNo = ServletUtility.getPageNo(request);

intpageSize = 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="btnbtn-danger btn-block">Delete</a>

</div>

<div class="col">

<a href="<%=SOTGView.PRODUCT\_CTL%>?id=<%=bean.getId()%>" class="btnbtn-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="btnbtn-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>

Beans:

**package** com.online.garments.deal.bean;

**import** java.sql.Timestamp;

**public** **abstract** **class** BaseBean **implements** DropdownListBean, Comparable<BaseBean> {

/\*\*

\* Generic Attribute Id For All Child

\*/

**protected** **long** id;

/\*\*

\* Generic Attribute Created BY For All Child

\*/

**protected** String createdBy;

/\*\*

\* Generic Attribute Modified BY For All Child

\*/

**protected** String modifiedBy;

/\*\*

\* Generic Attribute Created Date and TIme For All Child

\*/

**protected** Timestamp createdDatetime;

/\*\*

\* Generic Attribute Modified Date And TIme For All Child

\*/

**protected** Timestamp modifiedDatetime;

/\*\*

\* accessor

\*/

/\*\*

\* **@return** ID of Child

\*/

**public** **long** getId() {

**return** id;

}

/\*\*

\* **@param** Id

\* To set Child ID

\*/

**public** **void** setId(**long** id) {

**this**.id = id;

}

/\*\*

\* **@return** Created By Child

\*/

**public** String getCreatedBy() {

**return** createdBy;

}

/\*\*

\* **@param** CreatedBy

\* To set Child Creatd By

\*/

**public** **void** setCreatedBy(String createdBy) {

**this**.createdBy = createdBy;

}

/\*\*

\* **@return** Modified By Child

\*/

**public** String getModifiedBy() {

**return** modifiedBy;

}

/\*\*

\* **@param** Modified

\* by To set Child ModifiedBY

\*/

**public** **void** setModifiedBy(String modifiedBy) {

**this**.modifiedBy = modifiedBy;

}

/\*\*

\* **@return** Created Date And TIme of Child

\*/

**public** Timestamp getCreatedDatetime() {

**return** createdDatetime;

}

/\*\*

\* **@param** Created

\* Date and Time To set Child Created dATE aND tiME

\*/

**public** **void** setCreatedDatetime(Timestamp createdDatetime) {

**this**.createdDatetime = createdDatetime;

}

/\*\*

\* **@return** Modified Date And Time of Child

\*/

**public** Timestamp getModifiedDatetime() {

**return** modifiedDatetime;

}

/\*\*

\* **@param** Modified

\* By To set Child ModiFied By

\*/

**public** **void** setModifiedDatetime(Timestamp modifiedDatetime) {

**this**.modifiedDatetime = modifiedDatetime;

}

**public** **int** compareTo(BaseBean next) {

// **TODO** Auto-generated method stub

**return** getValue().compareTo(next.getValue());

}

}

**package** com.online.garments.deal.bean;

**public** **class** CustomerBean **extends** BaseBean {

**private** String Name;

**private** String age;

**private** String address;

**private** String contectNo;

**private** String productChoice;

**private** String itemCode;

**private** String MultipleItems;

**private** String login;

**private** String password;

**private** String confirmPassword;

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getConfirmPassword() {

**return** confirmPassword;

}

**public** **void** setConfirmPassword(String confirmPassword) {

**this**.confirmPassword = confirmPassword;

}

**public** String getLogin() {

**return** login;

}

**public** **void** setLogin(String login) {

**this**.login = login;

}

**public** String getName() {

**return** Name;

}

**public** **void** setName(String name) {

Name = name;

}

**public** String getAge() {

**return** age;

}

**public** **void** setAge(String age) {

**this**.age = age;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getContectNo() {

**return** contectNo;

}

**public** **void** setContectNo(String contectNo) {

**this**.contectNo = contectNo;

}

**public** String getProductChoice() {

**return** productChoice;

}

**public** **void** setProductChoice(String productChoice) {

**this**.productChoice = productChoice;

}

**public** String getItemCode() {

**return** itemCode;

}

**public** **void** setItemCode(String itemCode) {

**this**.itemCode = itemCode;

}

**public** String getMultipleItems() {

**return** MultipleItems;

}

**public** **void** setMultipleItems(String multipleItems) {

MultipleItems = multipleItems;

}

@Override

**public** String getKey() {

// **TODO** Auto-generated method stub

**return** id+"";

}

@Override

**public** String getValue() {

// **TODO** Auto-generated method stub

**return** Name;

}

}

**Controller:**

package com.online.garments.deal.controller;

import java.io.IOException;

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 javax.servlet.http.HttpSession;

import org.apache.log4j.Logger;

import com.online.garments.deal.bean.BaseBean;

import com.online.garments.deal.bean.CustomerBean;

import com.online.garments.deal.bean.LoginBean;

import com.online.garments.deal.bean.ManagerBean;

import com.online.garments.deal.bean.PaymentBean;

import com.online.garments.deal.bean.ProductBean;

import com.online.garments.deal.exception.ApplicationException;

import com.online.garments.deal.exception.DuplicateRecordException;

import com.online.garments.deal.model.CustomerModel;

import com.online.garments.deal.model.LoginModel;

import com.online.garments.deal.model.ManagerModel;

import com.online.garments.deal.model.PaymentModel;

import com.online.garments.deal.model.ProductModel;

import com.online.garments.deal.util.DataUtility;

import com.online.garments.deal.util.DataValidator;

import com.online.garments.deal.util.PropertyReader;

import com.online.garments.deal.util.ServletUtility;

/\*\*

\* Servlet implementation class CustomerCtl

\*/

@WebServlet(name = "CustomerCtl", urlPatterns = { "/CustomerCtl" })

public class CustomerCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

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

@Override

protected boolean validate(HttpServletRequest request) {

log.debug("CustomerCtl validate method start");

boolean pass = true;

if (DataValidator.isNull(request.getParameter("name"))) {

request.setAttribute("name",PropertyReader.getValue("error.require", "Name"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("contact")))

{

request.setAttribute("contact",PropertyReader.getValue("error.require", "Contact No"));

pass = false;

}else if(!DataValidator.isPhoneNo(request.getParameter("contact")))

{

request.setAttribute("contact", PropertyReader.getValue("error.invalid","Contact No"));

pass=false;

}

if (DataValidator.isNull(request.getParameter("age")))

{

request.setAttribute("age",PropertyReader.getValue("error.require", "Age"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("address")))

{

request.setAttribute("address",PropertyReader.getValue("error.require", "Address"));

pass = false;

}

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

HttpSession session=request.getSession();

LoginBean lBean=(LoginBean)session.getAttribute("user");

if(lBean==null) {

String login = request.getParameter("login");

if (DataValidator.isNull(login)) {

request.setAttribute("login", PropertyReader.getValue("error.require", "Login Id"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("password"))) {

request.setAttribute("password", PropertyReader.getValue("error.require", "Password"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("confirmPassword"))) {

request.setAttribute("confirmPassword", PropertyReader.getValue("error.require", "Confirm Password"));

pass = false;

}

if(!OP\_PAYMENT.equalsIgnoreCase(op)) {

if (!request.getParameter("password").equals(

request.getParameter("confirmPassword"))

&& !"".equals(request.getParameter("confirmPassword"))) {

/\*ServletUtility.setErrorMessage("Confirm Password did not match",

request);\*/

request.setAttribute("confirmPassword", PropertyReader.getValue("error.confirmPassword","Confirm Password"));

pass = false;

}

}

}

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

pass=true;

}

log.debug("CustomerCtl validate method end");

return pass;

}

@Override

protected BaseBean populateBean(HttpServletRequest request) {

log.debug("CustomerCtl Method populatebean Started");

CustomerBean bean = new CustomerBean();

bean.setId(DataUtility.getLong(request.getParameter("id")));

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

bean.setAge(DataUtility.getString(request.getParameter("age")));

bean.setContectNo(DataUtility.getString(request.getParameter("contact")));

bean.setAddress(DataUtility.getString(request.getParameter("address")));

HttpSession session=request.getSession();

LoginBean lBean=(LoginBean)session.getAttribute("user");

if(lBean==null) {

bean.setLogin(DataUtility.getString(request.getParameter("login")));

bean.setPassword(DataUtility.getString(request.getParameter("password")));

bean.setConfirmPassword(DataUtility.getString(request

.getParameter("confirmPassword")));

}else {

bean.setLogin(lBean.getLogin());

bean.setPassword(lBean.getPassword());

}

populateDTO(bean, request);

log.debug("CustomerCtl Method populatebean Ended");

return bean;

}

/\*\*

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

\*/

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

log.debug("CustomerCtl Method doGet Started");

System.out.println("in do get");

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

long id = DataUtility.getLong(request.getParameter("id"));

HttpSession session=request.getSession(true);

long pid=DataUtility.getLong(request.getParameter("pid"));

if(pid==0) {

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

}else {

session.setAttribute("pid", pid);

ServletUtility.setOpration("Add", request);

CustomerModel model = new CustomerModel();

if (id > 0 || op != null) {

CustomerBean bean;

try {

bean = model.findByPk(id);

ServletUtility.setOpration("Edit", request);

ServletUtility.setBean(bean, request);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

}

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

}

log.debug("CustomerCtl Method doGett Ended");

}

/\*\*

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

\*/

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

log.debug("CustomerCtl Method doPost Started");

System.out.println("in do post");

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

// get model

HttpSession session=request.getSession();

CustomerModel model = new CustomerModel();

long id = DataUtility.getLong(request.getParameter("id"));

ServletUtility.setOpration("Add", request);

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

CustomerBean bean = (CustomerBean) populateBean(request);

try {

if (id > 0) {

model.update(bean);

ServletUtility.setSuccessMessage("Data is successfully Updated",request);

ServletUtility.setOpration("Edit", request);

ServletUtility.setBean(bean, request);

} else {

session.setAttribute("cusBean", bean);

ServletUtility.forward(OGDView.PAYMENT\_VIEW, request, response);

}

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

} catch (DuplicateRecordException e) {

ServletUtility.setBean(bean, request);

ServletUtility.setErrorMessage(e.getMessage(), request);

}

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

}

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

CustomerBean bean = (CustomerBean) populateBean(request);

try {

model.delete(bean);

ServletUtility.redirect(OGDView.CUSTOMER\_LIST\_CTL, request,response);

return;

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

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

ServletUtility.redirect(OGDView.CUSTOMER\_LIST\_CTL, request, response);

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

ServletUtility.redirect(OGDView.CUSTOMER\_CTL, request, response);

return;

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

CustomerBean cBean=(CustomerBean)session.getAttribute("cusBean");

ProductModel pModel=new ProductModel();

long pId=(long)session.getAttribute("pid");

try {

LoginBean lBean=(LoginBean)session.getAttribute("user");

if(lBean==null) {

LoginModel lModel=new LoginModel();

LoginBean lbBean=new LoginBean();

lbBean.setLogin(cBean.getLogin());

lbBean.setPassword(cBean.getPassword());

lbBean.setRole(3L);

lModel.add(lbBean);

}

ProductBean pBean=pModel.findByPk(pId);

cBean.setItemCode(pBean.getItemCode());

if(lBean!=null) {

cBean.setLogin(lBean.getLogin());

}

long pk=model.add(cBean);

PaymentModel pyModel=new PaymentModel();

PaymentBean pyBean=new PaymentBean();

pyBean.setCustomerId(pk);

pyBean.setProductId(pId);

pyBean.setAmount(pBean.getPrice());

pyBean.setLogin(cBean.getLogin());

if(lBean!=null) {

pyBean.setLogin(lBean.getLogin());

}

pyModel.add(pyBean);

ServletUtility.setSuccessMessage("Payment Done Successfully", request);

ServletUtility.forward(OGDView.SUCCESS\_VIEW, request, response);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

} catch (DuplicateRecordException e) {

ServletUtility.setErrorMessage(e.getMessage(), request);

}

}

log.debug("CustomerCtl Method doPost Ended");

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return OGDView.CUSTOMER\_VIEW;

}

}

**Model:**

package com.online.garments.deal.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 com.online.garments.deal.bean.CustomerBean;

import com.online.garments.deal.bean.LoginBean;

import com.online.garments.deal.exception.ApplicationException;

import com.online.garments.deal.exception.DatabaseException;

import com.online.garments.deal.exception.DuplicateRecordException;

import com.online.garments.deal.util.JDBCDataSource;

public class CustomerModel {

private static Logger log = Logger.getLogger(CustomerModel.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 Customer");

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 PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public CustomerBean findByName(String name) throws ApplicationException {

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

StringBuffer sql = new StringBuffer("SELECT \* FROM Customer WHERE name=?");

CustomerBean 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 CustomerBean();

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

bean.setName(rs.getString(2));

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

bean.setAddress(rs.getString(4));

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

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

bean.setItemCode(rs.getString(7));

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

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

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

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

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

bean.setLogin(rs.getString(13));

}

rs.close();

} catch (Exception e) {

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

throw new ApplicationException("Exception : Exception in getting Customer by Name");

} finally {

JDBCDataSource.closeConnection(conn);

}

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

return bean;

}

public CustomerBean findByContactNo(String contact) throws ApplicationException {

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

StringBuffer sql = new StringBuffer("SELECT \* FROM Customer WHERE contactNo=?");

CustomerBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setString(1, contact);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new CustomerBean();

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

bean.setName(rs.getString(2));

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

bean.setAddress(rs.getString(4));

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

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

bean.setItemCode(rs.getString(7));

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

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

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

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

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

bean.setLogin(rs.getString(13));

}

rs.close();

} catch (Exception e) {

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

throw new ApplicationException("Exception : Exception in getting Customer by contact");

} finally {

JDBCDataSource.closeConnection(conn);

}

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

return bean;

}

public CustomerBean findByPk(long id) throws ApplicationException {

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

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

CustomerBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setLong(1, id);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new CustomerBean();

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

bean.setName(rs.getString(2));

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

bean.setAddress(rs.getString(4));

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

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

bean.setItemCode(rs.getString(7));

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

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

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

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

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

bean.setLogin(rs.getString(13));

}

rs.close();

} catch (Exception e) {

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

throw new ApplicationException("Exception : Exception in getting Customer by Pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

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

return bean;

}

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

log.debug("Model add Started");

Connection conn = null;

int pk = 0;

CustomerBean duplicataLogin = findByContactNo(bean.getContectNo());

if (duplicataLogin != null) {

throw new DuplicateRecordException("Contact No Id already exists");

}

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

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

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO Customer VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

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

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

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

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

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

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

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

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

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

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

pstmt.setString(13,bean.getLogin());

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 Customer");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model add End");

return pk;

}

public void delete(CustomerBean 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 Customer 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 Customer");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model delete Started");

}

public void update(CustomerBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

CustomerBean duplicataCustomer = findByContactNo(bean.getContectNo());

// Check if updated Role already exist

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

throw new DuplicateRecordException("Contact No already exists");

}

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE Customer SET Name=?,age=?,Address=?,ContectNo=?,ProductChoice=?,ItemCode=?,MultipleItems=?,CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=?,Login=? WHERE ID=?");

pstmt.setString(1, bean.getName());

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

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

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

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

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

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

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

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

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

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

pstmt.setString(12,bean.getLogin());

pstmt.setLong(13, 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 Customer ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

public List search(CustomerBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

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

throws ApplicationException {

log.debug("Model search Started");

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

if (bean != null) {

if (bean.getId() > 0) {

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

}

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

sql.append(" AND ContectNo LIKE '" + bean.getContectNo() + "%'");

}

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

sql.append(" AND ItemCode LIKE '" + bean.getItemCode() + "%'");

}

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

sql.append(" AND Login LIKE '" + bean.getLogin() + "%'");

}

}

// 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 CustomerBean();

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

bean.setName(rs.getString(2));

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

bean.setAddress(rs.getString(4));

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

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

bean.setItemCode(rs.getString(7));

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

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

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

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

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

bean.setLogin(rs.getString(13));

list.add(bean);

}

rs.close();

} catch (Exception e) {

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

throw new ApplicationException(

"Exception : Exception in search Customer");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

public List list() throws ApplicationException {

return list(0, 0);

}

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

log.debug("Model list Started");

ArrayList list = new ArrayList();

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

// 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()) {

CustomerBean bean = new CustomerBean();

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

bean.setName(rs.getString(2));

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

bean.setAddress(rs.getString(4));

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

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

bean.setItemCode(rs.getString(7));

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

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

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

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

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

bean.setLogin(rs.getString(13));

list.add(bean);

}

rs.close();

} catch (Exception e) {

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

throw new ApplicationException(

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

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

}

package com.online.garments.deal.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 com.online.garments.deal.bean.LoginBean;

import com.online.garments.deal.exception.ApplicationException;

import com.online.garments.deal.exception.DatabaseException;

import com.online.garments.deal.exception.DuplicateRecordException;

import com.online.garments.deal.util.DataUtility;

import com.online.garments.deal.util.JDBCDataSource;

public class LoginModel {

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

/\*\*

\* Find next PK of Role

\*

\* @throws DatabaseException

\*/

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 Login");

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 PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public LoginBean findByLogin(String login) throws ApplicationException {

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

StringBuffer sql = new StringBuffer("SELECT \* FROM Login WHERE Login=?");

LoginBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setString(1, login);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new LoginBean();

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

bean.setLogin(rs.getString(2));

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

bean.setCreatedBy(rs.getString(4));

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

bean.setCreatedDatetime(rs.getTimestamp(6));

bean.setModifiedDatetime(rs.getTimestamp(7));

bean.setRole(rs.getLong(8));

}

rs.close();

} catch (Exception e) {

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

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

} finally {

JDBCDataSource.closeConnection(conn);

}

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

return bean;

}

public LoginBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

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

LoginBean 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 LoginBean();

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

bean.setLogin(rs.getString(2));

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

bean.setCreatedBy(rs.getString(4));

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

bean.setCreatedDatetime(rs.getTimestamp(6));

bean.setModifiedDatetime(rs.getTimestamp(7));

bean.setRole(rs.getLong(8));

}

rs.close();

} catch (Exception e) {

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

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

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

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

log.debug("Model add Started");

Connection conn = null;

int pk = 0;

LoginBean duplicataLogin = findByLogin(bean.getLogin());

if (duplicataLogin != null) {

throw new DuplicateRecordException("Login Id already exists");

}

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

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

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO Login VALUES(?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

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

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

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

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

pstmt.setTimestamp(6, DataUtility.getCurrentTimestamp());

pstmt.setTimestamp(7, DataUtility.getCurrentTimestamp());

pstmt.setLong(8,bean.getRole());

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 Login");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model add End");

return pk;

}

public void delete(LoginBean 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 Login 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 void update(LoginBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

LoginBean duplicataRole = findByLogin(bean.getLogin());

// Check if updated Role already exist

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

throw new DuplicateRecordException("Login Id already exists");

}

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE login SET Login=?,Password=?,CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=?,Role=? WHERE ID=?");

pstmt.setString(1, bean.getLogin());

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

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

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

pstmt.setTimestamp(5, bean.getCreatedDatetime());

pstmt.setTimestamp(6, bean.getModifiedDatetime());

pstmt.setLong(7,bean.getRole());

pstmt.setLong(8, 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 Login ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

public List search(LoginBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

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

throws ApplicationException {

log.debug("Model search Started");

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

if (bean != null) {

if (bean.getId() > 0) {

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

}

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

sql.append(" AND LOGIN LIKE '" + bean.getLogin() + "%'");

}

}

// 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 LoginBean();

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

bean.setLogin(rs.getString(2));

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

bean.setCreatedBy(rs.getString(4));

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

bean.setCreatedDatetime(rs.getTimestamp(6));

bean.setModifiedDatetime(rs.getTimestamp(7));

bean.setRole(rs.getLong(8));

list.add(bean);

}

rs.close();

} catch (Exception e) {

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

throw new ApplicationException(

"Exception : Exception in search Login");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

public List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of Role with pagination

\*

\* @return list : List of Role

\* @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 Login");

// 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()) {

LoginBean bean = new LoginBean();

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

bean.setLogin(rs.getString(2));

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

bean.setCreatedBy(rs.getString(4));

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

bean.setCreatedDatetime(rs.getTimestamp(6));

bean.setModifiedDatetime(rs.getTimestamp(7));

bean.setRole(rs.getLong(8));

list.add(bean);

}

rs.close();

} catch (Exception e) {

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

throw new ApplicationException(

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

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

public LoginBean authenticate(String login, String password) throws ApplicationException {

log.debug("Model authenticate Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM Login WHERE LOGIN = ? AND PASSWORD = ?");

LoginBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

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

pstmt.setString(1, login);

pstmt.setString(2, password);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new LoginBean();

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

bean.setLogin(rs.getString(2));

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

bean.setCreatedBy(rs.getString(4));

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

bean.setCreatedDatetime(rs.getTimestamp(6));

bean.setModifiedDatetime(rs.getTimestamp(7));

bean.setRole(rs.getLong(8));

System.out.println("Usermodel here");

}

} catch (Exception e) {

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

throw new ApplicationException("Exception : Exception in get Authenticate Login");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model authenticate End");

return bean;

}

}

package com.online.garments.deal.util;

import java.sql.Connection;

import java.sql.SQLException;

import java.util.ResourceBundle;

import com.mchange.v2.c3p0.ComboPooledDataSource;

import com.online.garments.deal.exception.ApplicationException;

/\*\*

\* JDBC DataSource is a Data Connection Pool

\*

\* @author Navigable Set

\* @version 1.0

\* @Copyright (c) Navigable Set

\*

\*/

public class JDBCDataSource

{

/\*\*

\* JDBC Database connection pool ( DCP )

\*/

private static JDBCDataSource datasource;

private JDBCDataSource() {

}

private ComboPooledDataSource cpds = null;

/\*\*

\* Create instance of Connection Pool

\*

\* @return

\*/

public static JDBCDataSource getInstance() {

if (datasource == null) {

ResourceBundle rb = ResourceBundle.getBundle("com.online.garments.deal.bundle.system");

datasource = new JDBCDataSource();

datasource.cpds = new ComboPooledDataSource();

try {

datasource.cpds.setDriverClass(rb.getString("driver"));

} catch (Exception e) {

e.printStackTrace();

}

datasource.cpds.setJdbcUrl(rb.getString("url"));

datasource.cpds.setUser(rb.getString("username"));

datasource.cpds.setPassword(rb.getString("password"));

datasource.cpds.setInitialPoolSize(new Integer((String) rb .getString("initialPoolSize")));

datasource.cpds.setAcquireIncrement(new Integer((String) rb.getString("acquireIncrement")));

datasource.cpds.setMaxPoolSize(new Integer((String) rb.getString("maxPoolSize")));

datasource.cpds.setMaxIdleTime(DataUtility.getInt(rb.getString("timeout")));

datasource.cpds.setMinPoolSize(new Integer((String) rb.getString("minPoolSize")));

}

return datasource;

}

/\*\*

\* Gets the connection from ComboPooledDataSource

\*

\* @return connection

\*/

public static Connection getConnection() throws Exception {

return getInstance().cpds.getConnection();

}

/\*\*

\* Closes a connection4

\*

\* @param connection

\* @throws Exception

\*/

public static void closeConnection(Connection connection) {

if (connection != null) {

try {

connection.close();

} catch (Exception e) {

}

}

}

public static void trnRollback(Connection connection)

throws ApplicationException {

if (connection != null) {

try {

connection.rollback();

} catch (SQLException ex) {

throw new ApplicationException(ex.toString());

}

}

}

}

package com.online.garments.deal.util;

import java.io.IOException;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.online.garments.deal.bean.BaseBean;

import com.online.garments.deal.controller.BaseCtl;

import com.online.garments.deal.controller.OGDView;

/\*\*

\* This class provides utility operation for Servlet container like forward,

\* redirect, handle generic exception=same time exception, manage success and

\* error message, manage default Bean and List, manage pagination parameters

\*

\* @author Navigable Set

\* @version 1.0

\* @Copyright (c) Navigable Set

\*/

public class ServletUtility {

public static void forward(String page, HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

RequestDispatcher rd = request.getRequestDispatcher(page);

System.out.println(page);

rd.forward(request, response);

}

public static void redirect(String page, HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

response.sendRedirect(page);

}

public static void handleException(Exception e, HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

request.setAttribute("exception", e);

ServletUtility.forward(OGDView.ERROR\_CTL, request, response);

e.printStackTrace();

}

public static String getErrorMessage(String property, HttpServletRequest request) {

String val = (String) request.getAttribute(property);

if (val == null) {

return "";

} else {

return val;

}

}

public static String getMessage(String property, HttpServletRequest request) {

String val = (String) request.getAttribute(property);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setErrorMessage(String msg, HttpServletRequest request) {

request.setAttribute(BaseCtl.MSG\_ERROR, msg);

}

public static String getErrorMessage(HttpServletRequest request) {

String val = (String) request.getAttribute(BaseCtl.MSG\_ERROR);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setSuccessMessage(String msg, HttpServletRequest request) {

request.setAttribute(BaseCtl.MSG\_SUCCESS, msg);

}

public static String getSuccessMessage(HttpServletRequest request) {

String val = (String) request.getAttribute(BaseCtl.MSG\_SUCCESS);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setBean(BaseBean bean, HttpServletRequest request) {

request.setAttribute("bean", bean);

}

public static BaseBean getBean(HttpServletRequest request) {

return (BaseBean) request.getAttribute("bean");

}

public static String getParameter(String property, HttpServletRequest request) {

String val = (String) request.getParameter(property);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setList(List list, HttpServletRequest request) {

request.setAttribute("list", list);

}

public static List getList(HttpServletRequest request) {

return (List) request.getAttribute("list");

}

/\*\*

\* Sets Page Number for List pages

\*

\* @param pageNo

\* @param request

\*/

public static void setPageNo(int pageNo, HttpServletRequest request) {

request.setAttribute("pageNo", pageNo);

}

public static int getPageNo(HttpServletRequest request) {

return (Integer) request.getAttribute("pageNo");

}

public static void setPageSize(int pageSize, HttpServletRequest request) {

request.setAttribute("pageSize", pageSize);

}

public static int getPageSize(HttpServletRequest request) {

return (Integer) request.getAttribute("pageSize");

}

public static void setOpration(String msg, HttpServletRequest request) {

request.setAttribute("Opration", msg);

}

public static String getOpration(HttpServletRequest request) {

String val = (String) request.getAttribute("Opration");

if (val == null) {

return "";

} else {

return val;

}

}

}

<%@page import=*"com.online.garments.deal.bean.ProductBean"*%>

<%@page import=*"com.online.garments.deal.model.ProductModel"*%>

<%@page import=*"com.online.garments.deal.controller.CustomerCtl"*%>

<%@page import=*"com.online.garments.deal.util.ServletUtility"*%>

<%@page import=*"com.online.garments.deal.util.DataUtility"*%>

<%@ 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>Customer</title>

</head>

<body>

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

<br>

<nav aria-label=*"breadcrumb"* role=*"navigation"*> <nav

aria-label=*"breadcrumb"* role=*"navigation"*>

<ol class=*"breadcrumb"*>

<li class=*"breadcrumb-item"*><a href=*"*<%=OGDView.WELCOME\_CTL%>*"*>Home</a></li>

<li class=*"breadcrumb-item"*><a href=*"*<%=OGDView.PRODUCT\_LIST\_CTL%>*"*>Products</a></li>

<li class=*"breadcrumb-item active"* aria-current=*"page"*>Customer</li>

</ol>

</nav>

<div class=*"container"*>

<div class=*"card"*>

<div class=*"container-fliud"*>

<div class=*"wrapper row"*>

<div class=*"preview col-md-6"*>

<% Long pid=(Long)session.getAttribute("pid");

ProductModel pModel=**new** ProductModel();

ProductBean pBean=pModel.findByPk(pid);

%>

<div class=*"preview-pic tab-content"*>

<div class=*"tab-pane active"* id=*"pic-1"*><img src=*"*<%=OGDView.APP\_CONTEXT%>*/image/*<%=pBean.getImage()%>*"* /></div>

</div>

</div>

<div class=*"details col-md-6"*>

<h3 class=*"product-title"*><%=pBean.getProductName() %></h3>

<div class=*"rating"*>

<div class=*"stars"*>

<span class=*"fa fa-star checked"*></span>

<span class=*"fa fa-star checked"*></span>

<span class=*"fa fa-star checked"*></span>

<span class=*"fa fa-star"*></span>

<span class=*"fa fa-star"*></span>

</div>

<span class=*"review-no"*>41 reviews</span>

</div>

<h4 class=*"price"*>current price: <span><%=pBean.getPrice()%></span></h4>

</div>

</div>

</div>

</div>

</div>

<div col-md-5 img-thumbnail">

<div id=*"feedback"*> <div class=*"container"*>

<div class=*"col-md-5"*>

<div class=*"form-area"*>

<form role=*"form"* action=*"*<%=OGDView.CUSTOMER\_CTL%>*"* method=*"post"* >

<jsp:useBean id=*"bean"* class=*"com.online.garments.deal.bean.CustomerBean"*

scope=*"request"*></jsp:useBean>

<input type=*"hidden"* name=*"id"* value=*"*<%=bean.getId()%>*"*>

<input type=*"hidden"* name=*"createdBy"* value=*"*<%=bean.getCreatedBy()%>*"*>

<input type=*"hidden"* name=*"modifiedBy"* value=*"*<%=bean.getModifiedBy()%>*"*>

<input type=*"hidden"* name=*"createdDatetime"* value=*"*<%=DataUtility.getTimestamp(bean.getCreatedDatetime())%>*"*>

<input type=*"hidden"* name=*"modifiedDatetime"* value=*"*<%=DataUtility.getTimestamp(bean.getModifiedDatetime())%>*"*>

<br style="clear:*both*">

<h3 style="margin-bottom: *15px*; text-align: *left*: ;">Add Detail</h3>

<b><font color=*"red"*> <%=ServletUtility.getErrorMessage(request)%>

</font></b>

<b><font color=*"Green"*> <%=ServletUtility.getSuccessMessage(request)%>

</font></b>

<div class=*"form-group"*>

<input type=*"text"* class=*"form-control"* name=*"name"*

placeholder=*"Name"* value=*"*<%=DataUtility.getStringData(bean.getName())%>*"* >

<font color=*"red"*><%=ServletUtility.getErrorMessage("name", request)%></font>

</div>

<%**if**(!userLoggedIn){%>

<div class=*"form-group"*>

<input type=*"text"* class=*"form-control"* name=*"login"*

placeholder=*"Login Id"* value=*"*<%=DataUtility.getStringData(bean.getLogin())%>*"* >

<font color=*"red"*><%=ServletUtility.getErrorMessage("name", request)%></font>

</div>

<div class=*"form-group"*>

<input type=*"password"* class=*"form-control"* name=*"password"*

placeholder=*"Password"* value=*"*<%=DataUtility.getStringData(bean.getPassword())%>*"* >

<font color=*"red"*><%=ServletUtility.getErrorMessage("password", request)%></font>

</div>

<div class=*"form-group"*>

<input type=*"password"* class=*"form-control"* name=*"confirmPassword"*

placeholder=*"Confirm Password"* value=*"*<%=DataUtility.getStringData(bean.getConfirmPassword())%>*"* >

<font color=*"red"*><%=ServletUtility.getErrorMessage("confirmPassword", request)%></font>

</div>

<%} %>

<div class=*"form-group"*>

<input type=*"text"* class=*"form-control"* name=*"age"*

placeholder=*"Age"* value=*"*<%=DataUtility.getStringData(bean.getAge())%>*"* >

<font color=*"red"*><%=ServletUtility.getErrorMessage("age", request)%></font>

</div>

<div class=*"form-group"*>

<input type=*"text"* class=*"form-control"* name=*"contact"*

placeholder=*"Contact No"* value=*"*<%=DataUtility.getStringData(bean.getContectNo())%>*"* >

<font color=*"red"*><%=ServletUtility.getErrorMessage("contact", request)%></font>

</div>

<div class=*"form-group"*>

<textarea class=*"form-control"* name=*"address"* placeholder=*"Address"* rows=*"4"*><%=DataUtility.getStringData(bean.getAddress()) %></textarea>

<font color=*"red"*><%=ServletUtility.getErrorMessage("address", request)%></font>

</div>

<input type=*"submit"* name=*"operation"*

class=*"btn btn-primary pull-right"* value=*"*<%=CustomerCtl.OP\_SAVE%>*"*>

<input type=*"submit"* name=*"operation"*

class=*"btn btn-primary pull-right"* value=*"*<%=CustomerCtl.OP\_RESET%>*"*>

</form>

</div>

</div>

</div> </div>

<!--feedback-->

<br>

<div style="margin-top: *83px*">

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

</div>

</body>

</html>

##### 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.