1. **Synopsis**

**1.1 Introduction of the System**

**1.1.1 Project title: Sell Gros**

**1.1.2 Category: Web Application**

**1.1.3 Overview:**

This project is a web-based application that provides an online platform for the grocery store or possible customers to order their basic, needed products.

The main purpose of this simple project is to let customers buy their groceries without going to the shop or store, with a fixed price in every region. A government tender will be called to provide these items. User orders for those items are based on their requirements. Admin will assign the dealers to supply these items to particular regions and also add bill generation. The system will list all products with a fixed price, and customers can save their desired product to their shopping cart and checkout when they are done.

**1.2. Background**

* + 1. **Introduction of the Company**

Not applicable

* + 1. **Brief note on Existing System**
* The present scenario for shopping is to visit the shops and market manually and then from the available product list one needs to choose the item he or she wants and then pay for the same item mainly in cash mode.
* Existing system is not much user-friendly as one needs to go to the market physically and then select items only from the available list So mostly it is difficult to get the product as per our desire.
* Till now we have these types of systems without having the fixed price for the items in different regions. Our system will provide the fixed price for the items in every region.

**1.3. Objective of the System**

* Objective of the system is something that you [plan](https://dictionary.cambridge.org/dictionary/english/plan) to do or [achieve](https://dictionary.cambridge.org/dictionary/english/achieve).
* The main objective of the sell gross is to manage the customers grocery orders and assign dealer to particular orders. It manages all the information about Shopping, products, region, dealer.
* It may be useful during the pandemic situations.

**1.4. Scope of the System**

* Scope is the limitation that a process faces from the beginning to the end.
* People can shop from home whenever they want. No worrying about store closing, holidays etc.
* Online shopping offers great safety in shopping from home, especially in times like the present – when the global pandemic is threatening the health of all those who mingle in public.
* We can also add some extra features like ordering food, electronic gadgets in one website.

**1.5. Structure of the System**

* + 1. **Login**

The Login Module is a portal module that allows customer/dealers/admin to type their user’s name and password to log in.

* + 1. **Registration module**
       1. **Customer registration**

This module is used to registercustomer by entering their necessary details.

* + - 1. **Dealer registration**

This module is used to registerdealer by entering their necessary details.

* + 1. **Admin module**
       1. **Product management**

This module is used to manage the products and their Information.

* + - 1. **Dealer authorization**

This module manages the authorized dealer.

**1.5.3.3 Dealer allotment**

This module is used to allot the dealer for the customer’s order.

**1.5.3.4 Region management**

This module is used to manage the region.

**1.5.3.5 View Feedback**

This module is used to manage the user’s feedback and report.

* + 1. **Dealers’ module**
       1. **Pending orders**

This module shows the active orders.

* + - 1. **Previous orders**

This module shows the previous orders.

* + - 1. **Bill generation**

This module generates the invoice/bill for active orders.

* + 1. **Customers module**

**1.5.4.1 Cart**

This module shows the products which is to be ordered.

**1.5.4.1.1 Modify cart**

This module is used add or remove quantity of the products in a cart.

**1.5.4.1.2 Place order**

This module allows the customers to order required items.

**1.5.4.2 Previous order**

This module shows previous order which is placed by the customer

**1.5.4.2.1 View bill**

Customer can view the bill based on orders.

**1.5.4.2.2** **Feedback and report**

Feedback module allows to get a review on how customers feel about dealer and products.

**1.5.4.3 Pending order**

This module shows the order that is pending.

* 1. **System Architecture**

Front End

Collect Data

**Users**

Display Data

**User Interface**

**HTML / CSS / JavaScript**

**Application Logic**

**PHP**

Request

**Database**

**MySQL**

Response

Back End

**Operating System**

**Processing Hardware**

* 1. **End User**
* All type of people with some knowledge of using computer.
  1. **Software/Hardware needs for the development.**
     1. **Hardware needs for the develop this product**
  + **CPU:** Intel or AMD processor with 64-bit support.
  + **RAM**: 4 GB or higher.
  + **Disk Storage:** 4 GB of free disk space or higher.
  + Keyboard, Mouse, Laptop or Computer
    1. **Software needs for the develop this product**
* **OS:** Windows 7 or higher.
* **Front End**: HTML/CSS
* **Back End**: PHP 5.0, phpMyAdmin.
* **Code Editor**: VS code
* **Web Server**: Apache Tomcat
* **Other Tools**: XAMPP.
  1. **Software and Hardware needs for the implementation.**
     1. **Hardware needs for implementing this product**
* **CPU:** Intel or AMD processor with 64-bit support.
* **RAM**: 4 GB or higher.
* **Disk Storage:** 4 GB of free disk space or higher.
* Keyboard, Mouse, Laptop or Computer
  + 1. **Software needs for implementing this product**
* Web browser
* Internet

1. **SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 Introduction**

Software requirement specification describes completely an external behavior of the proposed software. The software requirement specification is a document that completely describes what the proposed software should do without describing how the software will do it. It lays out functional and non-functional requirements.The basic purpose of SRS is to bridge the communication gap between the user and developer. Another important purpose is if developing the SRS is helping the clients to understand their own needs. Boundaries of software products are defined set of requirements. The software developer team designing, implements, tests, and delivers these requirements to client. This software requirement specification document will be the basic for final system. A high-quality software specification is a pre-requisite to high quality software and to reduce development cost.

**2.2 Overall Description**

This section describes the function of the project and their aim. It also includes the constraints and the requirements of the project.

**2.2.1 Product Perspective**

**2.2.1.1 System Interface**

This application runs in the latest version of Chrome or Firefox browser on windows, Linux, and mac.

* + - 1. **User Interface**
* **GUI (Graphical user interface)** is used to interact between user and system through different components.
* Each part of the user interface is designed to be as user-friendly.
  + - 1. **Hardware Interface**
* **Processor:** 1.9 gigahertz (GHz) x86- or x64-bit dual core processor. Recommended- 3.3 gigahertz (GHz) or faster 64-bit dual core processor.
* **RAM:** Minimum 4GB or higher
* **Storage:** Minimum 64GB is needed.
  + - 1. **Software Interface**
* Windows Xp and higher version and any compatible browser like chrome, Edge, Internet explorer.
* Xampp and MySQL

**2.2.1.5 Communication Interface**

Stable internet connection and browser.

**2.2.1.6 Interface with Server**

This application allows to interface with SQL Server, Xampp.

**2.2.2 Product Function**

**2.2.2.1 Admin:** Admin can add, remove, update the price, item, and dealer.

**2.2.2.2 Dealer:** Dealer can view the customer details and he can generate bill.

**2.2.2.3 Customer:** customer order the products and view the bill.

**2.2.3 User Characteristics**

**2.2.3.1 Admin:** He/she should have the knowledge of manipulating web applications.

**2.2.3.2 Dealer:** Basic knowledge of using computer and smartphone.

**2.2.3.3 Customer:** He/she should know how to use the browser and computer.

**2.2.4 General Constraints**

Not applicable

* + 1. **Assumption and Dependencies**
* For windows 11 1 gigahertz (GHz) or faster with 2 or more cores on a compatible 64-bit processor or System on a Chip (SoC). 4 gigabytes (GB). 64 GB or larger storage device.
* The system is dependent on the availability of an Apache Tomcat Server to run.

**2.3 Special requirements**

Not applicable

**2.4 Functional requirements**

**2.4.1** **Login module**

In this customer/dealer/admin can login to their system using username and password.

**Input:** username, password

**Process:** Check for username, password

**Output:** If username, password is valid customer/dealer/admin can login to their system.

**2.4.2 Registration module**

**2.4.2.1 Customer registration**

**Input:** Details of the customer

**Process**: Details are stored in database.

**Output**: Registration successful message will be displayed.

**2.4.2.2 Dealer registration**

**Input:** Details of the Dealer

**Process**: Details are stored in database.

**Output**: Registration successful message will be displayed.

**2.4.3 Admin module**

**2.4.3.1 Product management**

**2.4.3.1.1 Add**

**Input:** Enter the product name, price, image, description, stock, unit.

**Process:** It validates the product details. If it is valid then it stores into database otherwise shows error message.

**Output:** product details are stored in database and Successful message will be displayed.

**2.4.3.1.2 Update**

**Input:** Enter the product name, price, image, description, stock, unit which is to be updated.

**Process:** It validates the product details.

**Output:** Product will be updated, and successful message will be displayed.

**2.4.3.1.3 Active/Inactive**

**Input:** Select the toggle button to active or inactive the products

**Process:** Enables/disables the products which is added.

**Output:** Display the respective message.

**2.4.3.2 Dealer Authorization**

**Input:** Select accept or reject button for authenticating dealer.

**Process:** operation is used accept/reject dealer.

**Output:** Dealer status will be updated in database and gets respective message.

**2.4.3.3 Dealer Allotment**

**Input:** click on allot button.

**Process:** Assigning dealer based on order to supply orders.

**Output:** Dealer Id will be assigned to every order in database.

**2.4.3.4 Region management**

**2.4.3.4.1 Add**

**Input**: Enter Region name, pin code.

**Process:** It validates the region details. If it is valid then it stores into database otherwise shows error message.

**Output:** Display the successful message.

**2.4.3.4.2 Update**

**Input:** click on update and change region name, pin code

**Process:** It validates the updated region details. If it is valid then it stores into database otherwise shows error message.

**Output:** Region details will be updated in database anddisplay the successful message

**2.4.3.4.3 Active/Inactive**

**Input:** Select the toggle button to active or inactive the region

**Process:** Enables/disables the region which is added.

**Output:** Display the respective message.

**2.4.3.5 View Feedback**

**Input:** click on view Feedback

**Process:** Feedback will be retrieved from database.

**Output**: Feedback will be displayed.

**2.4.4 Dealer**

**2.4.4.1 Pending orders**

**Input:** Click on pending orders.

**Process:** Orders will be retrieved from the database.

**Output:** Active order will be displayed.

**2.4.4.2 Previous order**

**Input:** Click on previous orders.

**Process:** Dealer assigned order will be retrieved from database.

**Output:** Previous orders will be retrieved from the database.

**2.4.4.2 Bill Generation**

**Input:** Click on bill generate.

**Process:** The bill will be generated stored to database.

**Output:** Bill will be displayed.

**2.4.5 Customer**

**2.4.5.1 Cart**

**2.4.5.1.2 Modify cart**

**Input:** Products will be loaded and select increase/decrease button.

**Process:** Product quantity can be increased or decreased.

**Output:** Changes are updated in database.

**2.4.5.1.2 place order**

**Input:** click on place order

**Process:** Order will be placed and updated in database.

**Output:** Display the successful message

**2.4.5.2 Previous order**

**2.4.5.2.1 view bill**

**Input:** Select orders.

**Process:** Calculation of amount and bill generation.

**Output:** Bill will be displayed.

**2.4.5.2.2 Feedback**

**Input:** click on feedback and report.

**Process:** Process the id and feedback or report and stores it.

**Output:** Notification message will be shown regarding feedback or report.

**2.4.5.3 Pending orders.**

**Input:** Click on Pending orders.

**Process:** Orders will be retrieved from the database.

**Output:** Active order will be displayed.

**2.5 Design Constraints**

**2.5.1 Hardware Constraints**

**RAM:**4 GB or higher

**Storage:**64 GB or higher

**2.5.2 Software Constraints**

HTML, CSS, PHP, MySQL, Internet, Browser.

**2.5.3 Fault Tolerance**

At the time of verification and validation invalid information will be removed.

Only valid data will be stored in the database.

**2.5.4 Security**

Only authorized user can access the application by using their username and password.

**2.5.5 Standard Compliances**

Not applicable

**2.6 System Attributes**

* **Availability**

Available for 24x7

* **Portability**

This application is machine independent and can be used in any systems.

* **Reliability**

This application is reliable and works efficiently. All input data will be verified and validated to avoid system failure.

* **Maintainability**

The application is maintained in a better way by providing updates. If any requirement needed it should be updated immediately.

* **Scalability**

The application’s functionalities can be changed as per the user’s demand. Software will remain stable and works constantly while making changes or updating or upgrading software.

**2.7 Other Requirements**

Not applicable

**3 System Design**

* 1. **Introduction**
* System design is the process of defining the architecture, module interfaces and data for a system to satisfy specified requirements.
* The purpose of the design phase is to plan the solution of the problem specified by the requirement documents.
* This is the first step that moving from problem domain to the solution domain.
* The design of the system is essentially a blueprint or a plan for a solution for the system.

**3.2 Assumption and constraints**

* For windows 11 1 gigahertz (GHz) or faster with 2 or more cores on a compatible 64-bit processor or System on a Chip (SoC). 4 gigabytes (GB). 64 GB or larger storage device.
* The system is dependent on the availability of an Apache Tomcat Server to run.
* 3 months of time for completion.

**3.3 Functional Decomposition**

Functional decomposition is the process of taking a complex process and breaking it down into its smaller, simpler parts. Using functional decomposition large or complex functionalities are more easily understood. It is mainly used during project analysis phase, so each phase can be viewed as software. So, this has modular with some sub modules.

* + 1. **System Software Architecture**

Dealer authorization

Login

Customer Registration

Registration

Dealer Registration

Product management

Admin

Add

Update

Active/Inactive

SellGross

Dealer allotment

Add

Region management

Update

Active/Inactive

View Feedback

Pending order

Dealer

Previous order

Bill generation

Modify cart

Customer

Cart

Place order

Previous order

View bill

Feedback

Pending order

**3.3.2 System technical architecture**

Internet

Chrome client

Xampp server

MySQL Database server

Mozilla client

Presentation Tier

Application Tier

Data Management Tier

Printer

Computer

Hard disk or storage device

Keyboard and mouse

**3.3.3 System hardware architecture**

**3.3.4 External Interface**

Not applicable

* 1. **Description of programs**
     1. **Context Flow Diagram**

In CFD entire system is considered as a single process. Context flow diagram shows input and outputs of the system. It shows all the external entities that interact with the system and how the data flows between these external entities and system.

Order placement

Registration

Customer

Registration

Feedback

Bill view

Admin

Bill generates

Dealer

Products, Regions, Dealers

Messages

previous orders

Previous orders

* + 1. **Data Flow Diagram**

Data flow diagram shows the flow of data through system. Data flow diagrams also called the data flow graphs. It views a system as a function that transforms the inputs into desired outputs. It aims to capture the transformation that taken place within a system to the input data so that eventually the output data is produced.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Name** | **Description** |
|  | Process | It performs transformation of data from one state to another. |
|  | Sink/Source | It represents the external entity that may be either source or sink. |
|  | Flow of data | It represents the flow of data from source to destination. |
|  | Data Source/Data storage | It is the place where data is stored. |

**Top level DFD**

Users

* 1. **Description of the components**
     1. **Login module**
        1. **Input**

User name and password

* + - 1. **Process**

Read details and validation

* + - 1. **Output**

Logged in to their respective pages.

* + - 1. **Interface with another functional components**

Independent

* + - 1. **Resource allocation**

Admin/Dealer/Customer table

* + - 1. **User interface**

Textboxes are provided to enter the username and password. Login button is provided to move to next page

* 1. **Description of the components**
     1. **Login module**
        1. **Input**

User name and password

* + - 1. **Process**

Read details and validation

* + - 1. **Output**

Logged in to their respective pages.

* + - 1. **Interface with another functional components**

Independent

* + - 1. **Resource allocation**

Admin/Dealer/Customer table

* + - 1. **User interface**

Textboxes are provided to enter the username and password. Login button is provided to move to next page.

Display message

Admin/Dealer/Customer

User name

Password

Admin/Dealer/Customer

Valid

Invalid

**3.5.2 Registration module**

Registration

Customer registration

Dealer registration

* + - 1. **Customer Registration**
         1. **Input**

Registration details of the customer.

**3.5.2.1.2 Process**

Read customer registration details

* + - * 1. **Output**

Registration successful message view be display and stored in database.

* + - * 1. **Interface with another functional components**

Independent

* + - * 1. **Resource allocation**

Customer table

* + - * 1. **User interface**

Textboxes are provided to enter the details. Register button is provided to store the details.

Valid data

Customer

Customer

Invalid

Customer details

Show message

* + - 1. **Dealer Registration**
         1. **Input**

Registration details of the Dealer.

**3.5.2.1.2 Process**

Read Dealer registration details

* + - * 1. **Output**

Registration successful message view be display and stored in database.

* + - * 1. **Interface with another functional components**

Independent

* + - * 1. **Resource allocation**

Dealer table

* + - * 1. **User interface**

Textboxes are provided to enter the details. Register button is provided to store the details.

Valid data

Dealer

Dealer

Invalid

Customer details

Show message

**3.5.3 Admin module**

Admin

* + - 1. **Product Management**

Product Management

* + - * 1. **Add**

**Input**

Product Details- name, image, price, description, stock, Unit.

**Process**

Validates the products and Store to database

**Output**

Show successful message

**Interface with another component**

Independent

**Resource allocation**

Product table

**User Interface**

Textboxes are provided for entering product details and add products button is provided to add product.

Invalid

Read details

Admin

Valid

Products

* + - * 1. **Update**

**Input**

Enter Product Details- name, image, price, description, stock, Unit to be updated.

**Process**

Validate input and update database

**Output**

Changes are updated in database and successful message will be displayed.

**Interface with another components**

Independent

**Resource allocation**

Product table

**User interface**

Update button will be provided for update details. Textboxes are provided for updating details. Update button will help to save details to database.

Message

Updated data

Admin

Product

Product selected

Display

Load

* + - * 1. **Active/Inactive**

**Input**

Product ID

**Process**

Product will not be available to use.

**Output**

product will be disabled and product status will be updated

**3.4.3.1.3.4 Interface with other components**

Independent

**Resource allocation**

Product table

**User Interface**

Item list will be displayed in the form of list. To active/inactive the product, set as active/inactive button is be provided.

Generate message

Click event

Admin

Product

Load

Product\_status

Display

* + - 1. **Dealer Authorization**
         1. **Input**

Dealer details.

* + - * 1. **Process**

Selection process and update table

* + - * 1. **Output**

Successful message.

* + - * 1. **Interface with another functional components**

Independent

* + - * 1. **Resource allocation**

Dealer table

* + - * 1. **User interface**

Approve and reject button is provided to authenticate.

Dealer details

Admin

Dealer details

* + - 1. **Dealer Allotment**
         1. **Input**

Dealer details and item order details.

* + - * 1. **Process**

Dealer is allotted to orders and updated in database.

* + - * 1. **Output**

Successful message

* + - * 1. **Interface with another functional components**

Independent

* + - * 1. **Resource allocation**

Dealer table and order table.

* + - * 1. **User interface**

Dropdown list for selecting dealer and allot button to allot dealer.

Admin

Item order

Dealer

Dealer id

Item order details

Item order details

Dealer id

Message

* + - 1. **Region Management**

Region management

* + - * 1. **Add**

**Input**

Region details- name, pin code

**3.5.3.4.1.2 Process**

Validation for existence

**Output**

Region will be added to database and success message will be shown.

**Interface with other components**

Independent

**3.5.3.4.1.5 Resource allocation**

Region table

**3.5.3.4.1.6** **User interface**

Textboxes are provided for adding region details. By clicking add button region will be added to table.

Invalid

Product

Admin

Details

Valid data

Region

* + - * 1. **Update**

**Input**

Region details- name, pin code

**Process**

Update operation and store to database

**Output**

Successful message

**Interface with other components**

Independent

**Resource allocation**

Region table

**User interface**

Textboxes are provided for entering details. update button will be provided to update database.

Updated

Database

Admin

Region table

Load regions

Modified

Show message

Region selected

* + - * 1. **Active/Inactive**

**Input**

Region id

**Process**

Region\_status update operation and update database

**Output**

Successful message

**Interface with another functional components**

Independent

**Resource allocation**

Region table

**User interface**

Region details will be displayed on the screen. and set as active/inactive button will be provided for active/inactive operation.

Updated

Database

Display

Update

Operation

Details

Load

Admin

Region table

Success message

* + - 1. **View feedback**

* + - * 1. **Input**

Feedback\_id

* + - * 1. **Process**

Feedback of the order will be displayed which is retrieved from feedback table.

* + - * 1. **Output**

Successful message

* + - * 1. **Interface with another functional components**

Independent

**3.4.2.4.5 Resource allocation**

Feedback table

**3.4.2.4.6 User interface**

Feedback from the user will be display for respective orders

Admin

Feedback

Feedback details

* + 1. **Dealer**

Dealer

* + - 1. **Pending orders**
         1. **Input**

Order id

* + - * 1. **Process**

Retrieve pending order details

* + - * 1. **Output**

Pending orders will be displayed

* + - * 1. **Interface with another functional components**

Independent

* + - * 1. **Resource allocation**

Item order table

* + - * 1. **User interface**

Shows all pending order.

Pending orders

Button click

Dealer

Order

Load

* + - 1. **Previous orders**
         1. **Input**

click

* + - * 1. **Process**

Retrieving order details

* + - * 1. **Output**

Orders will be displayed

* + - * 1. **Interface with another functional components**

Independent

* + - * 1. **Resource allocation**

Item order table

* + - * 1. **User interface**

Shows all proceeded order.

Button click

Dealer

Order

Load

* + - 1. **Bill generation**
         1. **Input**

Order details

* + - * 1. **Process**

Bill generation

* + - * 1. **Output**

Bill will be displayed

* + - * 1. **Interface with another functional components**

Independent

* + - * 1. **Resource allocation**

Cart table, Item order table, Bill table.

* + - * 1. **User interface**

After clicking on generate bill button the bill will be generated and displayed on the screen.

Display

Generated bill

Dealer

Item order

Bill

Cart

* + 1. **Customer**

Customer

Pending order

Previous order

Cart

* + - 1. **Cart**

Cart

* + - * 1. **Modify cart**

**Input**

Button clicks

**Process**

Store details to cart

**Output**

Successful message

**Interface with another functional component**

Independent

**User interface**

It shows all products which are in cart.

Product id

Customer

Cart

Product

quantity

* + - * 1. **Place order**

**Input**

Order details

**Process**

Generate order\_id and store it to order table

**Output**

Successful message

**Interface with another functional component**

Independent

**User interface**

Button is used to place order.

Show message

Order id

Cart details

Load

Customer

Cart

Order details

Order

* + - 1. **Previous order**

Previous order

* + - * 1. **View bill**

**Input**

Button clicks

**Process**

Retrieve bill from table

**Output**

Display bill

**Interface with another functional component**

Independent

**User Interface**

View button is given to view bill for previous order.

Button clicks

Customer

Bill

Bill details

* + - * 1. **Feedback**

**Input**

Entered feedback

**Process**

Store to feedback table

**Output**

Show successful message

**Interface with another functional component**

Independent

**User Interface**

Feedback button is given to write feedback.

Button clicks

Button clicks

Store

Customer

Feedback

Show message

**s**

* + - 1. **Pending order**
         1. **Input**

Button clicks

* + - * 1. **Process**

Retrieve previous orders

* + - * 1. **Output**

Display active orders

* + - * 1. **Interface with another functional component**Independent
        2. **User Interface**

Button is given to view pending order.

Load

Active order details

Customer

order

4. **Database Design**

**4.1** **Introduction**

Database is a collection of related data. Relational database stores data in a table or relations. The data stored in a relation are arranged in records. Each record consists of set of attributes. Fields can be referred to characteristics of records. This document describes the table that is used to design software, its attributes, data types, constraints and relationship among those tables.

The design process consists of the following steps:

* Determine the purpose of your database. Find and organize the information required.
* Divide the information into tables. Turn information items into columns...
* Specify primary keys.
* Set up the table relationships.
* Refine your design.
* Apply the normalization rules.

**4.2 Purpose and Scope**

**Purpose**

* Avoid Redundant Data

The table in the database should be constructed following standards and with utmost dedication. It should have different fields and minimize redundant data. The table should always have a Primary Key that would be a unique id.

* Faultless Information

The database should follow the standards and conventions and provide meaningful information useful to the organization. (Constraint)

* Data Integrity

Integrity assists in guaranteeing that the values are valid and faultless. Data Integrity is set to tables, relationships, etc.

* Modify

The database developed should be worked upon with the conventions and standards, so that it can be easily modified whenever the need arises.

**Scope**

* Normalization of Database.
* Imposing Integrity Constraint.
* Establishing the Relation between the tables.
* Accessing the data from multiple tables. (Usage of join and sub query….)

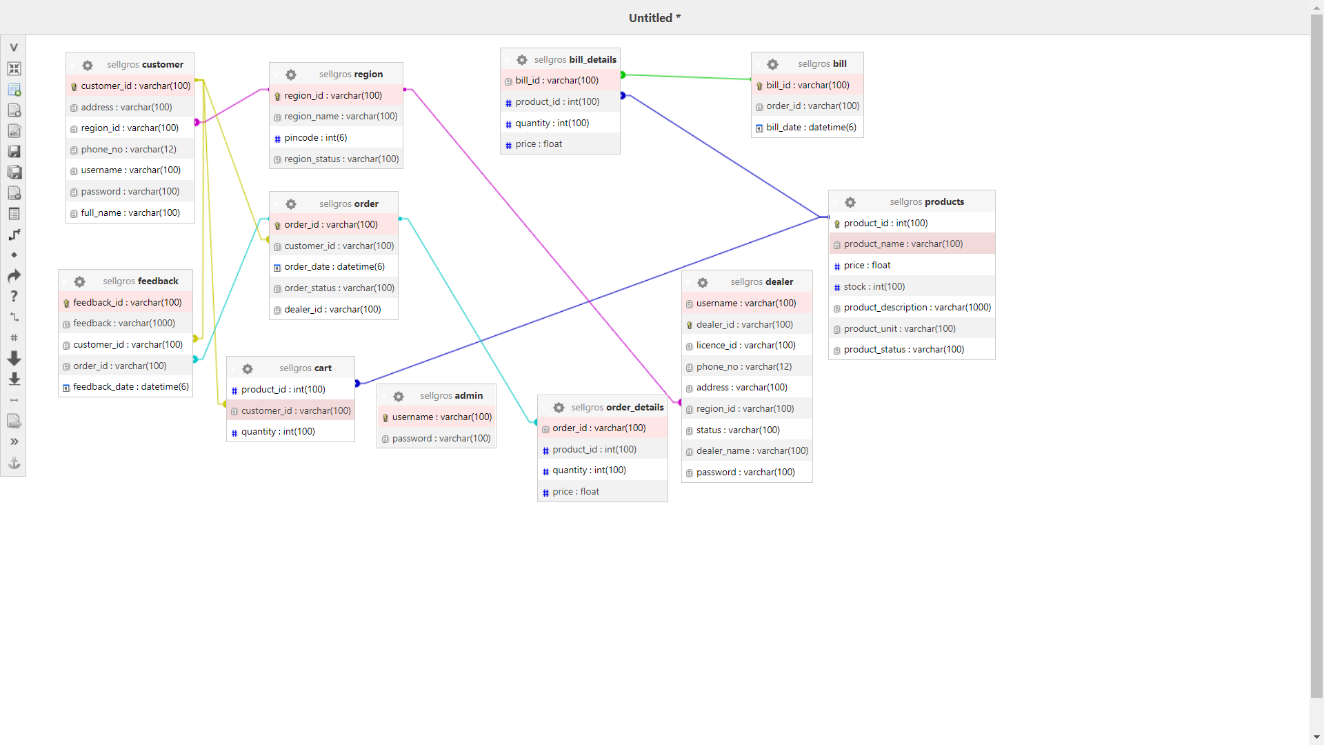
**4.3 Database Identification**

The identification of database by unique name given to the various database objects. The identifier is the name of database object. The following are the various database objects.

**4.4 Schema Information**

Database schema its structure described in a formal language supported by the Database Management System (DBMS). The term “schema” refers to the organization of the data as a blueprint of how database is constructed (divide into database tables in the case of relational databases)

In relational database, the schema defines the tables, fields, relationships, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views.



**4.5 Table Definition**

**4.5.1 Dealer table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| username | varchar | 100 | Not null | User name of the dealer |
| dealer\_id | varchar | 100 | Not null (primary key) | Id of the dealer |
| licence\_id | varchar | 100 | Not null | License id of dealer |
| phone\_no | varchar | 12 | Not null | Phone no. of the dealer |
| address | varchar | 100 | Not null | Address of the dealer |
| region\_id | varchar | 100 | Not null (foreign key) | Region\_id of the dealer |
| status | varchar | 100 | Not null | Status of the dealer |
| dealer\_name | varchar | 100 | Not null | Name of the dealer |
| Password | varchar | 100 | Not null | Password of the dealer |

**4.5.2 Customer table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| customer\_id | Varchar | 100 | Not null (primary key) | Id of the customer |
| address | Varchar | 100 | Not null | Address of the customer |
| region\_id | Varchar | 100 | Not null (foreign key) | Region\_id of the customer |
| phone\_no | Varchar | 12 | Not null | Phone no. of the customer |
| username | Varchar | 100 | Not null | Username of the customer |
| password | Varchar | 100 | Not null | Password of the customer |
| full\_name | Varchar | 100 | Not null | Fullname of the customer |

**4.5.3 Admin table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| username | Varchar | 100 | Not null | Username of the admin |
| password | Varchar | 100 | Not null | Password of the admin |

**4.5.4 Region table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| region\_id | varchar (primary key) | 100 | Not null | Id of the region |
| region\_name | Varchar | 100 | Not null | Name of the region |
| pincode | Int | 6 | Not null | Pin code for the region |
| Region\_status | varchar | 100 | Not null | Status of the Region |

**4.5.5 product table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| product\_id | Int | 100 | Not null (primary key) | Id of the product |
| product\_name | Varchar | 100 | Not null | Name of the product |
| Price | Float |  | Not null | Price of the product |
| Stock | Int | 100 | Not null | Stock of the project |
| product\_description | Varchar | 100 | Not null | Description of the product |
| product\_unit | Varchar | 100 | Not null | Unit of the product |
| Product \_status | varchar | 100 | Not null | Status of the product |

**4.5.6 Order table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| order\_id | Varchar | 100 | Not null (primary key) | Id of the order |
| customer\_id | Varchar | 100 | Not null (foreign key) | Id of the customer |
| order\_date | Datetime | 6 | Not null | Date of the order |
| order\_status | Varchar | 100 | Not null | Status of the order |
| dealer\_id | Varchar | 100 | null | Id of the dealer |

**4.5.7 Order\_details table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| order\_id | Varchar | 100 | Not null (foreign key) | Id of the order |
| product\_id | Int | 100 | Not null (foreign key) | Id of the products |
| quantity | Int | 100 | Not null | Quantity of the product |
| price | float |  | Not null | Price of the product |

**4.5.8 Cart table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| product\_id | Int | 100 | Not null (foreign key) | Id of the products |
| customer\_id | Varchar | 100 | Not null (foreign key) | Id of the customer |
| quantity | Int | 100 | Not null | Quantity of the product |

**4.5.9 Bill table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| Bill\_id | Varchar | 100 | Not null(primary key) | Id of the bill |
| Order\_id | Varchar | 100 | Not null | Id of the order |
| Bill\_date | Datetime | 6 | Not null | Date of the bill |

**4.5.10 Bill\_details table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| Bill\_id | Varchar | 100 | Not null (foreign key) | Id of the bill |
| Product\_id | Varchar | 100 | Not null (foreign key) | Id of the products |
| quantity | Int | 100 | Not null | Quantity of the products |
| price | Float |  | Not null | Price of the products |

**4.5.11 Feedback table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** | **Description** |
| feedback\_id | Varchar | 100 | Not null (primary key) | Id of the feedback |
| feedback | Varchar | 1000 | Not null | Feedback about order and dealer |
| customer\_id | Varchar | 100 | Not null (foreign key) | Id of the customer |
| order\_id | Varchar | 100 | Not null (foreign key) | Id of the order |
| feedback\_date | Datetime | 6 | Not null | Date of the feedback |

**4.6 Physical Design**

Physical design is where you translate schemas into actual database structures.

* Entity to table
* Tuples to rows
* Attribute to columns
* Primary Key and Alternate Key to Unique Index
* Domain into Constraints

**4.7 Data Dictionary**

A data dictionary is a file or set of files that include metadata. The data dictionary holds records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

The data dictionary, in general, includes information about the following:

* Name of the data item
* Aliases
* Description/purpose
* Related data items
* Range of values
* Data structure definition

**4.8 ER Diagram**

ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagram created using ER-modeling method are called Entry-Relationship diagram or ER-diagram or ERDs.

|  |  |
| --- | --- |
| Symbol | Conversion |
|  | Entity |
|  | Weak entity |
|  | Relationship |
|  | Identity relation |
|  | Attribute |
|  | Derived attribute |
| E  R  E  1  R | Cardinality ratio 1: N to E1:E2 to R |

**Components of an ER-diagram**

1. **Entity**

An entity can be a real word object, either animate or inanimate, that can be merely identifiable.

An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.

Entity set

An entity set is a collection of related types of entities.

Strong entity

An entity with uniquely identified by its attribute.

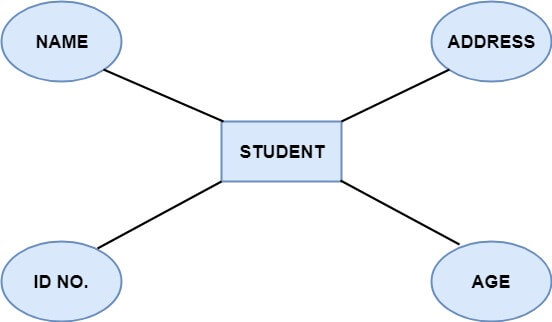
Weak Entity

In a relational database, a week entity is an entity that cannot be uniquely identified by its attributes alone.

1. **Attributes**

Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

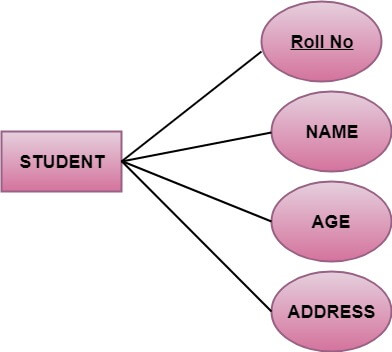
There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.



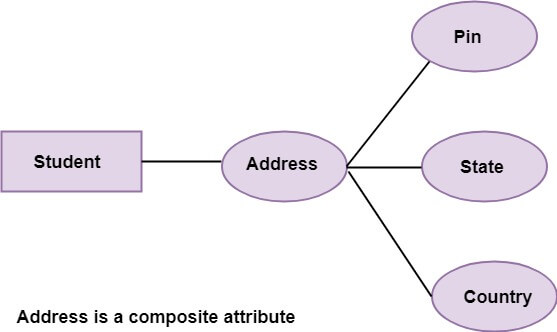
**There are four types of Attributes:**

* + - 1. **Key attribute**
      2. **Composite attribute**
      3. **Single-valued attribute**
      4. **Multi-valued attribute**
      5. **Derived attribute**

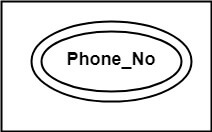
1. **Key attribute**: Key is an attribute or collection of attributes that uniquely identifies an entity among the entity set. For example, the roll number of a student makes him identifiable among students.



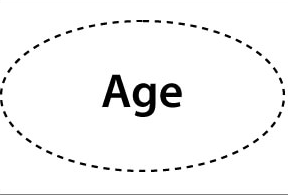
1. **Composite attribute:** An attribute that is a combination of other attributes is called a composite attribute. For example, in student entity, the student address is a composite attribute as an address is composed of other characteristics such as pin code, state, country.

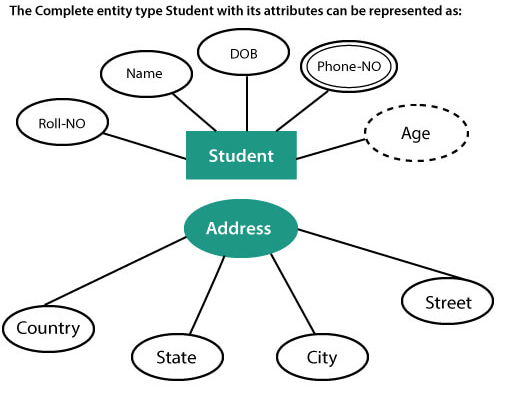


1. **Single-valued attribute**: Single-valued attribute contain a single value. For example, Social\_Security\_Number.
2. **Multi-valued Attribute**: If an attribute can have more than one value, it is known as a multi-valued attribute. Multi-valued attributes are depicted by the double ellipse. For example, a person can have more than one phone number, email-address, etc.



1. **Derived attribute:** Derived attributes are the attribute that does not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from date\_of\_birth. In the ER diagram, Derived attributes are depicted by the dashed ellipse.





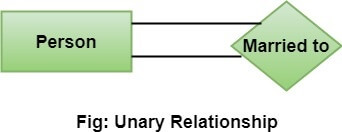
1. **Relationships**

The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works at a department, a student enrolls in a course. Here, Works at and enrolls are called relationships.

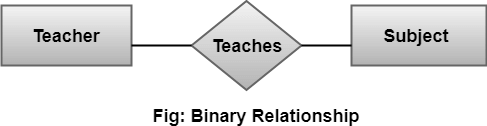
Degree of a relationship set

The number of participating entities in a relationship describes the degree of the relationship. The three most common relationships in E-R models are:

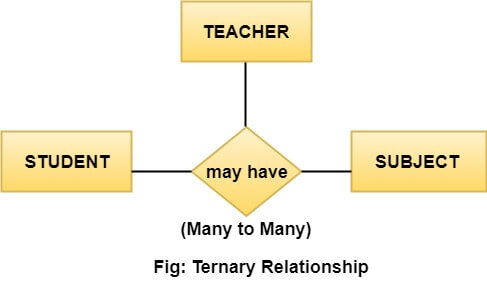
1. Unary (degree1)
2. Binary (degree2)
3. Ternary (degree3)
   1. **Unary relationship**: This is also called recursive relationships. It is a relationship between the instances of one entity type. For example, one person is married to only one person.



* 1. **Binary relationship:** It is a relationship between the instances of two entity types. For example, the Teacher teaches the subject.



* 1. **Ternary relationship:** It is a relationship amongst instances of three entity types. In fig, the relationships "**may have**" provide the association of three entities, i.e., TEACHER, STUDENT, and SUBJECT. All three entities are many-to-many participants. There may be one or many participants in a ternary relationship.



1. **Cardinality Ratio**

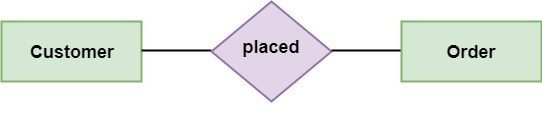
Cardinality describes the number of entities in one entity set, which can be associated with the number of entities of other sets via relationship set.

**Types of Cardinalities**

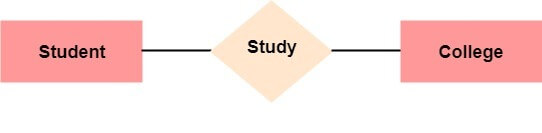
* 1. **One to One**: One entity from entity set A can be contained with at most one entity of entity set B and vice versa. Let us assume that each student has only one student ID, and each student ID is assigned to only one person. So, the relationship will be one to one.



* 1. **One to many:** When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationships. For example, a client can place many orders; a order cannot be placed by many customers.



* 1. **Many to One:** More than one entity from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A. For example - many students can study in a single college, but a student cannot study in many colleges at the same time.

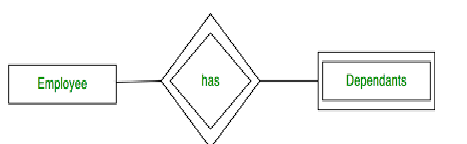


* 1. **Many to Many:** One entity from A can be associated with more than one entity from B and vice-versa. For example, the student can be assigned to many projects, and a project can be assigned to many students.



1. **Identifying relationship**

An identiffying relationship is a relationship between two entities in which an instance of a child entity is identified through its association with a parent entity, which means the child entity is dependent on the parent entity for its identity and cannot exits without it.

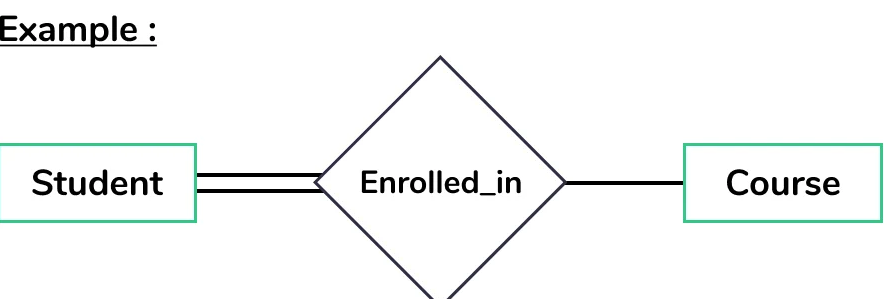


1. **Participation Contraints**

The participation constraint specifies the number of instance of an entity can participate in a relationship set.

Total participation – Each entity is involved in the relatioship. Total participation is represented by double lines.

Partial participation – Not all entities are involved in the relatioship. Partial participation is represented by single lines.



has

Region

Customer

Dealer

has

1

place

1

n

1

n

give

1

n

feedback

1

n

n

Order

has

has

Cart

products

1

n

n

1

1

n

Order\_details

n

has

1

have

1

1

bill

n

1

adds

Bill\_details

has

has

**4.9 Database Administration**

**4.9.1 DBMS system Information**

A database is an organized collection of structured information, or data typically stored electronically in a computer system. A data base is usually controlled by a database management system (DBMS).

In our project we are using MySQL database.

* + 1. **DBMS Configuration**

Steps for configure Apache and MySQL in XAMPP

* 1. In phpMyAdmin, click the Users tab at the top.
  2. Find the row that has User root and Host 127.0.0.1.
  3. Click Edit Privileges.
  4. Click Change password.
  5. Enter the password twice (write it down somewhere if you're sure, you can remember it)
  6. Click the Go button

**4.9.3 Software Support Required**

MySQL Required XAMPP

## Software Requirements

The following operating systems are officially supported:

* Windows xp or higher version (64-bit, Professional level or higher)
* Mac OS X 10.6.1 or higher
* Ubuntu 9.10 (64bit) or higher version

**4.9.4 Hardware (storage) requirements**

* Hard Disk:1 TB Required 500GB(Recommended)
  + - CPU: Intel Core 3GHz (or Dual Core 2GHz) or equal AMD CPU
* Cores: Dual (Quad Core is recommended)
* RAM: 4 GB (6 GB recommended)
  + - Graphic Accelerators: NVidia or ATI with support of OpenGL 1.5 or higher
    - Display Resolution: 1280×1024 is recommended, 1024×768 is minimum.

**4.9.5 Backup and Recover**

Recovery is the process of restoring a database to the correct state in the event of a failure

Database backup is a way to protect and restore a database. It is performed through database replication and can be done for a database or a database server.

Using phpMyAdmin to Back Up or Restore MySQL

If you’re running phpMyAdmin backing up and restoring your MySQL database is simple.

The export function is used as a backup, and the import function is used to restore.

Step 1: Create a MySQL Database Backup

1. Open phpMyAdmin. On the directory tree on the left, click the database you want to back up.

This should open the directory structure in the right-hand window. You’ll also notice that, in the directory tree on the left, all the assets under the main database are highlighted.

2. Click Export on the menu across the top of the display.

You’ll see a section called “Export Method.” Use Quick to save a copy of the whole database. Choose Custom to select individual tables or other special options.

Leave the Format field set to SQL, unless you have a good reason to change it.

3. Click Go. If you select Quick, your web browser will download a copy of the database into your specified downloads folder. You can copy that to a safe location.

Step 2: Clear the Old Database Information

It’s important to clear out old data before restoring a backup. If there’s any old data, it isn’t overwritten when you restore. This can create duplicate tables, causing errors and conflicts.

1. Open phpMyAdmin, on the navigation pane on the left, choose the database you want to restore.

2. Click the check all box near the bottom. Then, use the drop-down menu labeled with selected to select Drop.

3. The tool should prompt you to confirm that you want to go forward. Click yes.

This will get rid of all the existing data, clearing the way for your restoration.

Step 3: Restore Your Backed up MySQL Database

In phpMyAdmin, the Import tool is used to restore a database.

1. On the menu across the top, click Import.

2. The first section is labeled File to import. A couple of lines down, there’s a line that starts with “Browse your computer,” with a button labeled Choose File. Click that button.

3. Use the dialog box to navigate to the location where you’ve saved the export file that you want to restore. Leave all the options set to default. (If you created your backup with different options, you can select those here.)

4. Click Go.

**5. Detail Design**

**5.1 Introduction**

During detailed design, the internal logic of each module specified in system design is decided. During this phase further details of the modules are decided. Design of each of the modules usually specified in a high-level description language which is independent of the language in which software eventually be implemented.

**5.2 Structure of the system**

Login

Customer Registration

Registration

Dealer Registration

Product management

Admin

Add

Update

Active/Inactive

SellGross

Dealer authorization

Dealer allotment

Add

Region management

Update

Active/Inactive

View Feedback

Pending order

Dealer

Previous order

Bill generation

Modify cart

Customer

Cart

Place order

Previous order

View bill

Feedback

Pending order

**5.3 Module Description**

**Structure Chart**

Structure chart is a top-down modular design, consist of squares representing different models in a systems and lines**.** Structure chart shows how program has been partitioned into manageable modules hierarchy and organization of those modules and communicational interface.

|  |  |  |
| --- | --- | --- |
| Symbol | Name | Process |
|  | Data Flow | Shows the direction flow of data |
|  | Control Flow | Shows the direction flow of control |
|  | Processing | Shows manipulation, calculation and processing |
|  | Module invocation | It represents subordinate module invoked by super ordinate module |
| Main  A  B  C | Conditional invocation | It indicates the invocation of sub-ordinates. Module depends on the evaluation of condition |
| Main  A  B | Invocation | It represents the repetition |

**Flow Chart**

A flowchart is a graphically representation of the structure of process or system, algorithm or the step-by-step solution of the problem. Flowchart describes the flow of data through an information processing system and the parts of the flows. The flow is a set of the logic operations that meet the certain requirements.

|  |  |  |
| --- | --- | --- |
| Symbol | Name | Purpose |
|  | Terminator | It indicates the start and end of the process |
|  | Input/Output | Input/Output data |
|  | Decision | It represents the comparison or  question that determines an alternate path to be followed |
|  | Flow direction | Shows the direction of data flow |
|  | Processing | It represents manipulation, calculation or information processing |
|  | Direction action storage | File storage |
|  | Preparation (Looping) | An instruction or looping |
|  | In page | Connects different modules in same page |
|  | Off page | Connects different modules in another page |
|  | Delay | The Delay flowchart symbol depicts any waiting period that is part of a process |

**5.3.1 Login**

**5.3.1.1 Input**

Username and password

**5.3.1.2 Procedural Details**

Login

Display respective page

Input details

Authenticate

Read username and password

Display error message

Validate

Load page

Admin/Dealer

Order table

Customer

Load page

Product table

Valid

**5.3.1.3 File I/O Interfaces**

Admin/Dealer/Customer table

**5.3.1.4 Output**

Login and display respective pages

**5.3.1.5 Implementation aspects (if any)**

Textboxes, Buttons and Tabs

**5.3.2 Registration**

**5.3.2.1 Customer**

**5.3.2.1.1 Input**

Customer registration details

**5.3.2.1.2 Procedural Details**

Customer registration

Input customer details

Validation

Display successful message and load next page

Store to customer table

Display Error message

**5.3.2.1.3 File I/O Interfaces**

Customer table

**5.3.2.1.4 Output**

Add details to customer table and show message

**5.3.2.1.5 Implementation aspects (if any)**

**5.3.2.2 Dealer**

**5.3.2.1.1 Input**

Dealer registration details

**5.3.2.1.2 Procedural Details**

Dealer registration

Input dealer details

Validation

Display successful message and load next page

Store to dealer table

Display Error message

**5.3.2.1.3 File I/O Interfaces**

Dealer table

**5.3.2.1.4 Output**

Details will be stored and successful message will be displayed

**5.3.2.1.5 Implementation aspects (if any)**

Textboxes, Buttons and Dropdown list

**5.3.3 Admin**

**5.3.3.1 Product Management**

**5.3.3.1.1 Add**

**5.3.3.1.1.1 Input**

Product details

**5.3.3.1.1.2 Procedural Details**

Start

Product Details

Validation

Product

Display error message

If exists

Store to database

Product added successfully message

Stop

True

False

**5.3.2.1.1.3 File I/O Interfaces**

Product table

**5.3.2.1.1.4 Output**

Store the product details and show successful message

**5.3.2.1.1.5 Implementation aspects (if any)**

Textboxes, and Buttons

**5.3.3.1.2 Update**

**5.3.3.1.2.1 Input**

Updated product details

**5.3.3.1.2.2 Procedural Details**

Start

Make changes

Product

Click on update product

Update database

Stop

Show successful message

Retrieve Products

**5.3.3.1.2.3 File I/O Interfaces**

Product table

**5.3.3.1.2.4 Output**

Changes will be updated

**5.3.3.1.2.5 Implementation aspects (if any)**

Textboxes, and Buttons

**5.3.3.1.3 Active/Inactive**

**5.3.3.1.3.1 Input**

Active/Inactive text from buttons

**5.3.3.1.3.2 Procedural Details**

Step:1 Start

Step:2 CLICK on set as

active/inactive button

Step3: IF active THEN

UPDATE product table SET status as active

ELSE IF inactive THEN

UPDATE product table SET status as inactive

Step4: DISPLAY message as product

status has been changed

Step5: END

**5.3.3.1.3.3 File I/O Interfaces**

Product table

**5.3.3.1.3.4 Output**

Successful message

**5.3.3.1.3.5 Implementation aspects (if any)**

Buttons

**5.3.3.2 Dealer Authorization**

**5.3.3.2.1 Input**

Dealer details

**5.3.3.2.2 Procedural Details**

Step1: Start

Step2: retrieve dealer details where status is pending

Step3: CLICK on accept or reject

Step4: IF accept THEN

SET status as approve

END IF

IF reject THEN

SET status as reject

END IF

Step5: END

**5.3.3.2.3 File I/O Interfaces**

Dealer table

**5.3.3.2.4 Output**

Update dealer status and show message

**5.3.3.2.5 Implementation aspects (if any)**

Buttons

**5.3.3.3 Dealer Allotment**

**5.3.3.3.1 Input**

Dealer details and item order details

**5.3.3.3.2 Procedural Details**

Step1: Start

Step2: CLICK on DEALER ALLOTMENT

Step3: retrieve orders from order table where dealer\_id is NULL

Step4: retrieve customer order details from order details table

Step5: retrieve region from dealer table where region\_id in dealer table

Step6: SHOW orders and dealer\_id

Step7: SELECT dealer\_id from dropdown list

Step8: CLICK on allot button

Step9: assign dealer\_id to orders and UPDATE

dealer\_id in order table

Step10: END

**5.3.3.3.3 File I/O Interfaces**

Dealer table and item order table

**5.3.3.3.4 Output**

Successful message

**5.3.3.3.5 Implementation aspects (if any)**

Dropdown list, buttons

**5.3.3.4 Region Management**

**5.3.3.4.1 Add**

**5.3.3.4.1.1 Input**

Region details

**5.3.3.4.1.2 Procedural Details**

Start

Region Details

Validation

Region

Display error message

If exists

Store to database

Region added successfully message

Stop

True

False

**5.3.3.4.1.3 File I/O Interfaces**

Region table

**5.3.3.4.1.4 Output**

Region will be stored and successful message will be displayed

**5.3.3.4.1.5 Implementation aspects (if any)**

Textboxes and Buttons

**5.3.3.4.2 Update**

**5.3.3.4.2.1 Input**

Updated region details

**5.3.3.4.2.2 Procedural Details**

Start

Region

Make changes

Click on update region

Update database

Stop

Show successful message

Retrieve Regions

**5.3.3.4.2.3 File I/O Interfaces**

Region table

**5.3.3.4.2.4 Output**

Region updated message

**5.3.3.4.2.5 Implementation aspects (if any)**

Textboxes and Buttons

**5.3.3.4.3 Active/Inactive**

**5.3.3.4.3.1 Input**

Region id

**5.3.3.4.3.2 Procedural Details**

Step1: Start

Step2: CLICK on active/inactive button

Step3: IF active THEN

UPDATE region table SET status as active

ELSE IF inactive THEN

UPDATE region table SET status as inactive

Step4: DISPLAY message as region status has been changed

Step5: END

**5.3.3.4.3.3 File I/O Interfaces**

Region table

**5.3.3.4.3.4 Output**

Successful message

**5.3.3.4.3.5 Implementation aspects (if any)**

Buttons

**5.3.3.5 View feedback**

**5.3.3.5.1 Input**

Button clicks

**5.3.3.5.2 Procedural Details**

View feedback

Retrieve feedback from database

Load feedback page

Display feedback

**5.3.3.5.3 File I/O Interfaces**

Feedback table

**5.3.3.5.4 Output**

Feedback will be displayed

**5.3.3.5.5 Implementation aspects (if any)**

Buttons

**5.3.4 Dealer**

**5.3.4.1 Pending orders**

**5.3.4.1.1 Input**

Button clicks

**5.3.4.1.2 Procedural Details**

Step1: start

Step2: CLICK on pending order

Step3: retrieve orders where dealer\_id in order table where order\_status is pending.

Step4: DISPLAY orders with generate bill button

**5.3.4.1.3 File I/O Interfaces**

Order table

**5.3.4.1.4 Output**

Pending orders will be displayed

**5.3.4.1.5 Implementation aspects (if any)**

Buttons

**5.3.4.2 Previous orders**

**5.3.4.2.1 Input**

Button clicks

**5.3.4.2.2 Procedural Details**

Step1: Start

Step2: CLICK on previous order

Step3: retrieve order where order\_status is processed in order table

Step4: retrieve order\_id where order\_id in order table

Step5: DISPLAY orders.

Step6: END

**5.3.4.2.3 File I/O Interfaces**

Order table

**5.3.4.2.4 Output**

Previous order will be displayed

**5.3.4.2.5 Implementation aspects (if any)**

Buttons

**5.3.4.3 Bill Generation**

**5.3.4.3.1 Input**

Order details

**5.3.4.3.2 Procedural Details**

Bill generation

Input order details from database

Generate bill id

Display bill in pdf format

**5.3.4.3.3 File I/O Interfaces**

Cart table, Order table and Bill table

**5.3.4.3.4 Output**

Display bill

**5.3.4.3.5 Implementation aspects (if any)**

Buttons

**5.3.5 Customer**

**5.3.5.1 Cart**

**5.3.5.1.1 Modify cart**

**5.3.5.1.1.1 Input**

Items in the cart

**5.3.5.1.1.2 Procedural Details**

Step1: start

Step2: CLICK on add to cart button

Step3: retrieve product\_id from

product table

Step4: store to cart table

Step5: CLICK on cart

Step6: retrieve details from cart table

Step7: SET quantity by increasing or\

decreasing

Step8: store quantity to cart table

**5.3.5.1.1.3 File I/O Interfaces**

Cart table

**5.3.5.1.1.4 Output**

Items will be added to cart with quantity

**5.3.5.1.1.5 Implementation aspects (if any)**

Buttons

**5.3.5.1.2 Place order**

**5.3.5.1.2.1 Input**

Details from cart

**5.3.5.1.2.2 Procedural Details**

Start

Click on Place order

Cart

Order

Order details

Retrieve details from cart

Generate order id

Store to database

Order placed successfully message

Stop

**5.3.5.1.2.3 File I/O Interfaces**

Order table, Cart table

**5.3.5.1.2.4 Output**

Order placed message

**5.3.5.1.2.5 Implementation aspects (if any)**

Buttons

**5.3.5.2 Previous order**

**5.3.5.2.1 View bill**

**5.3.5.2.1.1 Input**

Button clicks

**5.3.5.2.1.2 Procedural Details**

Start

Click on view bill

Retrieve bill from database

Bill

Show bill

Stop

**5.3.5.2.1.3 File I/O Interfaces**

Bill table

**5.3.5.2.1.4 Output**

Bill will be displayed

**5.3.5.2.1.5 Implementation aspects (if any)**

Buttons

**5.3.5.2.2 Feedback**

**5.3.5.2.2.1 Input**

Feedback details

**5.3.5.2.2.2 Procedural Details**

Feedback

Feedback details

Generate feedback id for feedback

Store to database

Display message

**5.3.5.2.2.3 File I/O Interfaces**

Feedback table

**5.3.5.2.2.4 Output**

Successful message

**5.3.5.2.2.5 Implementation aspects (if any)**

Textboxes, Buttons, Hyperlinks

**5.3.5.3 Pending orders**

**5.3.5.3.1 Input**

Click

**5.3.5.3.2 Procedural Details**

Step1: start

Step 2: CLICK on pending orders

Step 3: Retrieve dealer\_id from order table where dealer\_id is NULL

Step 4: DISPLAY orders which are pending.

**5.3.5.3.3 File I/O Interfaces**

Order table

**5.3.5.3.4 Output**

Pending orders will be displayed

**5.3.5.3.5 Implementation aspects (if any)**

Buttons

**8 TESTING**

**8.1 Introduction**

Testing is the major quality control measures and during the software development it is used to detect errors that could have occurred during any of the phase like requirement analysis, design, coding. The goal of the testing is to uncover errors in the program.

**8.2 Levels of Testing**

Testing is done in different levels which includes the following.

* Unit Testing
* Integration Testing
* System testing
* Acceptance testing
* **Unit Testing**

In Unit testing each module gets tested during the coding phase itself. The purpose is to exercise the different parts of the module code to detect the coding errors.

* **Integration Testing**

After new testing the modules are gradually integrated into sub systems. It is performed to detect design errors by focusing on testing the interconnection between modules.

* **System Testing**

System is tested against the system requirement if all the requirements are met and if the system performs as specified by the requirement.

* **Acceptance Testing**

It is performed to demonstrate to the client on real life data of the client, the operation of the system.

**8.3 Test Case**

It is the input that tests the genuineness of the program and successful execution of the test case revels. that there are no errors in the program that are under testing. it is a set of conditions or variables under which tester will determine whether an application or software is working currently

|  |  |
| --- | --- |
| Test case-ID | 01 |
| Test Case Title | Login |
| Purpose of testing | Testing the login |
| Test data | Username and password |
| Steps | Step:1 IF username and Password is valid THEN  Step:2 DISPLAY Login successful message.  Step:3 ELSE DISPLAY Login Unsuccessful message. |
| Expected Output | Invalid output:      Valid output: |

|  |  |
| --- | --- |
| Test case-ID | 02 |
| Test Case Title | Registration |
| Purpose of testing | To register customer for the web-site. |
| Test data | Full name, Username, Password, Address, Region, Phone number |
| Steps | 1. Click on customer registration.  2. Enter the Full name, Username, Password, Address, Region, Phone number  3. Click on register button  4. If record is valid display registration successful message.  5. Else display warning message with description. |
| Expected Output | Invalid output:      Valid output: |

|  |  |
| --- | --- |
| Test case-ID | 03 |
| Test Case Title | Registration |
| Purpose of testing | To register dealer for the web-site. |
| Test data | Full name, Username, Password, Address, Region, Phone number, License ID |
| Steps | 1. Click on customer registration.  2. Enter the Full name, Username, Password, Address, Region, Phone number, License ID  3. Click on register button  4. If record is valid display registration successful message.  5. Else display warning message with description. |
| Expected Output | Invalid output:        Valid output: |

|  |  |
| --- | --- |
| Test case-ID | 04 |
| Test Case Title | Feedback |
| Purpose of testing | To give feedback for the dealer or order. |
| Test data | Feedback |
| Steps | 1. Click on feedback  2. Enter the feedback  3. Click on submit button  4. If feedback is given display successful message.  5. Else display warning message with description. |
| Expected Output | Invalid output:    Valid output: |

|  |  |
| --- | --- |
| Test case-ID | 05 |
| Test Case Title | Add product |
| Purpose of testing | To add product for website. |
| Test data | Image, Product name, price, quantity, stock, unit, description. |
| Steps | 1. Click on add product  2. Enter the Image, Product name, price, quantity, stock, unit, description.  3. Click on add product button  4. If product details are added display successful message.  5. Else display warning message with description. |
| Expected Output | Invalid output:      Valid output: |

|  |  |
| --- | --- |
| Test case-ID | 06 |
| Test Case Title | Update product |
| Purpose of testing | To change the added product for website. |
| Test data | Product name/price/quantity/stock/unit/description. |
| Steps | 1. Click on update product  2. Enter the details of the product you want to update.  3. Click on update product button  4. If product details are added display successful message.  5. Else display warning message with description. |
| Expected Output | Invalid output:      Valid output: |

|  |  |
| --- | --- |
| Test case-ID | 07 |
| Test Case Title | Add region |
| Purpose of testing | To add region for website. |
| Test data | Region name and pin code. |
| Steps | 1. Click on add region  2. Enter the region name and pin code.  3. Click on add region button  4. If region details are added display successful message.  5. Else display warning message with description. |
| Expected Output | Invalid output:        Valid output: |

|  |  |
| --- | --- |
| Test case-ID | 08 |
| Test Case Title | Add update region |
| Purpose of testing | To update region for website. |
| Test data | Region name/pin code. |
| Steps | 1. Click on update region  2. Enter the region name/pin code.  3. Click on update region button  4. If region details are added display successful message.  5. Else display warning message with description. |
| Expected Output | Invalid output:      Valid output: |