INTRODUCTION/OBJECTIVES

At present E-Commerce is the most vast and popular business sector. Its an online buying and selling process so its needs are as simple as the business means. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace .

E-Commerce means business over internet. Internet access is becoming so mainstream now that your product/service can reach almost everyone on the planet with a internet-enabled device. Internet virtual shop remains open and operational 24x7 even if you/your staff are not working- this might not be wholly true if your product is a service-which requires immediate human-intervention.

Our project named **Farm To Home Organic Online** **Shop** is an online e-commerce web application developed for selling organic vegetables and fruits. As above mentioned it is a virtual store on the internet where customers can browse the catalog and purchase products of interest. Customers can pay via online seamlessly. The selected items will be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as credit card number. An e- mail notification is sent to the customer as soon as the order is placed .

Nowadays Kerala is moving towards Organic revolution due to the amount of poisonous chemicals in the form of pesticides and fertilizers in fruits and vegetables. In recent times, Government agencies, social organizations, political parties, women's organizations and farmers' self-help groups are all participating in what is becoming an organic farming revolution in the state of Kerala.

Our main objective is to find out a market for these homegrown organic products. Main beneficiaries of our venture is,

1. **Customer** ---- will get healthy organic food products in a very reasonable price.
2. **Seller** --------- Farmers will get their deserved share.

SYSTEM ANALYSIS

System analysis is the primary phase of the software development. System analysis is the study of existing operation to learn what they accomplish, why they work as they do and what role they may have in future processing activities. In analyzing the current system, it is necessary to identify the essential data and procedure required for a new approach. It is also necessary to pinpoint current weakness and problem so that they will not be carried over to the new system.

In simple words, System analysis is defines every fundamental elements of proposes system like,

* Identification of Need
* Preliminary Investigation
* Feasibility Study
* Project Planning
* Project Scheduling (PERT Chart and Gantt Chart both)
* Software requirement specifications (SRS)
* Software Engineering Paradigm applied
* Data models

We can check each of them with Farm To Home Organic Online Shop perspective.

**Identification of Need**

**Farm To Home Organic Online** **Shop** is a web application which sells organic vegetable, fruits and others. Consider a grocery shop which will have variety of goods including vegetable , fruits and so on. It maintained in through proper order or catalog. So our system need various catalog which can categorize different product like tomato comes under vegetable catalog and spinach comes under greens. Also each product details including unit price, unit weight and so on. We should have good sellers and their information to get better service. Need to acquire various payment options to make payment easily such as Credit/Debit card, Paypal, Master Pass and so on. More importantly how to fix price of commodity as equal for all seller.

**The Existing System**

There are many small scale organic vegetable/fruits supplies organizations that still rely upon the traditional techniques of the market, where customers either directly walk-in to the store, or place an order through a telephone call or a written letter. The customers in this case are retained solely on the basis of trust. This trust is largely vulnerable. A simple error – with either party – could lead to loss of the customer, and more than that, the trust vested upon the organization. These organizations might not be able to invest a huge amount of money on developing an e-commerce web site for itself. Instead, they still follow the traditional error-prone methods of receiving orders and selling a product, and handles cash in currency instead of electronic funds. Higher chances are that the customers or clients in this case, which could be high profile organizations, could turn to other vendors and suppliers who provide hassle free and easier trade platforms. A few important limitations of the existing system identified were:

On a customer’s perspective

* To know more about a specific product, the customers have to either walk-in to the store, or enquire over a telephone call or written letter.
* Little or limited information available. Probing for more information would be time consuming and difficult.
* Order placement through direct walk-in, or telephone call, or written letter
* No efficient feedback mechanisms. Lack of query / complaints addressing procedures

On a organization’s perspective

* Use of traditional way of record keeping – on papers and files
* Time consuming procedures
* Data redundancy
* Lack of security
* Lack of automated reports to understand trend and make decisions

Next step would be to gather customer’s and organization’s requirements, and propose solutions to the problems identified. Different proposals are then weighted, conducted a feasibility study upon, and the best one is selected. The proposal would be reviewed on user request and suitable changes made. For the problems as identified with the existing system above, a solution is to create easy-to-use web based software to maintain better relationship with customers, by adhering to all their needs and providing an easy, hassle free, error free and interactive market experience.

**Preliminary Investigation**

In this process, the development team visits the customer and studies their system. They investigate the need for possible software automation in the given system. By the end of Preliminary Investigation, the team furnishes a document that holds the different specific recommendations for the candidate system. It also includes the personnel assignments, costs, project schedule, and target dates.

In Our **Farm To Home Organic Online** **Shop**, we have to visit a real time organic vegetable store and conduct a preliminary investigation find out following things,

* + 1. Investigate the present system and identify the functions to be performed
    2. Identify the objectives of the new system. In general, an information system benefits a business by increasing efficiency, improving effectiveness, or providing a competitive advantage
    3. Identify constraints, i.e. the limitations placed on the project, usually relating to time, money and resources
    4. Evaluate feasibility - whether the proposed system promises sufficient benefit to invest the additional resources necessary to establish the user requirements in greater detail

**Feasibility Study**

A feasibility study is an evaluation and analysis of the potential of the proposed project which is based on extensive investigation and research to give full comfort to the decisions makers. Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

Feasibility analysis involves a few steps.

Enumerate a potential candidate systems

* + - * 1. Identify and describe characteristics of candidate systems
        2. Prepare system flow charts
        3. Determine and evaluate performance and cost effectiveness of each candidate system
        4. Weigh system performance and cost data
        5. Select the best candidate system
        6. Repair and report final project directive

The four aspects in the feasibility study are:

**Technical Feasibility**

Focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. In our case **Farm To Home Organic Online** **Shop** is possible or not with current technologies.

**Economic Feasibility**

Known as cost benefit analysis, this is to determine the benefits expected from a candidate system as compare to cost. Only if the benefits outweigh costs then the system is implemented, else the system is altered. In our case whether **Farm To Home Organic Online** **Shop** is economically feasible or not.

**Operational feasibility**

It is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition. Checking **Farm To Home Organic Online** **Shop** will fulfill the desired intentions or not.

**Behavioral feasibility**

This is a survey of the acceptability of the software within the organization, and 15 the attitude of people towards it.

**Requirement Gathering**

Requirements gathering is about creating a clear, concise and agreed set of customer requirements that allow you to provide what the customer wants. A requirement is a statement about an intended product that specifies what it should do or how to do it. For requirements to be effectively implemented and measured, they must be specific, unambiguous and clear.

We can gather requirement by many ways. Few of them are,

* + 1. Interviews
    2. Discussions,
    3. Questionnaires

we will focus on the two types of System Requirements:

1. Functional Requirements

Functional requirements specify the software functionality that the developers must build into the product to enable users to accomplish their tasks,thereby satisfying the business requirements. In simpler words, functional requirements state what the system must do.

* Business rules
* Transaction corrections, adjustments
* Administrative functions
* Authentication
* Audit tracking
* External interfaces
* Certification requirements
* Reporting requirements
* Historical data
* Legal / Regulatory requirements
  1. Non-Functional Requirements

Constraints or standards that the system must have or comply with. Non-functional requirements define the system’s quality characteristics.

* Scalability
* Capacity
* Availability
* Reliability
* Recover-ability
* Maintainability
* Serviceability
* Security
* Regulatory
* Manageability
* Environmental
* Data Integrity
* Usability
* Interoperability
* Performance

**Proposed System**

In our proposed system, Customer can buy the vegetables and fruits by their computer or smart phone. Customer can avoid lot of effort of buying a product. Customer can get detailed time to purchase the product also can maintain a virtual shopping cart. Our system proposed provides user friendly interface, and a hassle free shopping experience. Customer can able to place the order as well as cancel the order. The system also recommends a home delivery system for the purchased products. As seller, our proposed system allows to market their product though our proposed system. It makes wide visibility of seller’s product. Seller can update their product details such as product quantity and stock details. Seller can track down their sales by our proposed system. Our proposed system provides faster processing and ability to manage multiple customer order.

System Requirements Specifications

Software Specifications :

Operating System : Windows 7 and above

Web Server : Apache Tomcat 8.0

Environment : Java Enterprise Edition

Front End (User Interface) : HTML5,CSS3

Back End (Database interface) : MySQL 5.6

Server side scripting : JavaEE

Internet Browsers : Google Chrome/ Mozilla Firefox

Hardware Specifications :

When it comes to software development, careful selection of hardware configuration is very important. Insufficient memory may have adverse effect on speed and efficiency of the software. The processor should be powerful to handle all operations. The disk space should be large and sufficient enough to store large volumes of data once the software is operational.

Processor : Min. 1.6 GHz

RAM (Random Access memory) : Min. 1 GB

Disk Storage : Min. 40 GB

Display : SVGA COLOR

Keyboard : Standard Keyboard

Pointing Device : Any compatible mouse or pointing device

Printer : Inkjet (as required)

**Project Planning**

A project plan is a formal approved document which defines how the project will be executed, monitored and controlled. The project team should use this as a blueprint to broadly guide the project. Project plans are often a document created with a word processing tool. There may be several different companies and teams involved at various times throughout the project so it is important that all of them are aware of the project plan, dependencies with each other and how to communicate with each teams.

**Project Scheduling**

Project Scheduling in a project refers to road map of all activities to be done with specified order and within time slot allotted to each activity. Project managers tend to define various tasks, and project milestones and them arrange them keeping various factors in mind. They look for tasks lie in critical path in the schedule, which are necessary to complete in specific manner (because of task interdependency) and strictly within the time allocated. Arrangement of tasks which lies out of critical path are less likely to impact over all schedule of the project.

With so many different people and tasks which need to be managed it can be useful to use Pert chart or a Gantt chart to make sure everything progresses as it should.

PERT Chart

A PERT chart is a project management tool used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique. A PERT chart presents a graphic illustration of a project, consisting of numbered nodes representing events or milestones in the project linked by directional lines. The direction of the arrows on the lines indicates the sequence of tasks.

Using the PERT chart we were able to plan the execution of the project Farm To Home Organic Online Shop.

Consulting people

Identification of Needs people

Study the current system

New System Proposal

H/w &S/w requirements

Estab. Of project structure

System Design

Coding

Testing

Deliver New System

GANT Chart

A Gantt chart is a project technique that can be used for several purposes, including scheduling and resource planning. A Gantt chart is a bar chart, with each bar representing an activity. A Gantt chart provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in project.

A Gantt chart is constructed with a horizontal axis representing the total time span of the project, broken down into increments (for example, days, weeks or months) and a vertical axis representing the tasks that make up the project. Horizontal bars of varying lengths represent the sequences, timing, and time span for each task. The Gantt chart for Farm To Home Organic Online Shop is given below.

2

3

4

5

6

7

8

9

10

11

12

Jan 07 14 28 Feb 8 16 24 27 Mar 5 15 19 Apr 9 21 25 May 12 18 26 Jun 7 20

1. User Request and approval 7. Selection of a prototype

2. Requirement study 8. Physical Design

3. Initial Investigation 9. Logical Design (coding)

4. Feasibility Study 10. Testing

5. Requirement specification and approval 11. Implementation

6. Detailed investigation 12. User Training

**SOFTWARE REQUIREMENT SPECIFICATIONS**

In this section provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product. Also gives general description of the project , functional requirements, data requirements and constraints and assumptions made while designing the Farm To Home Organic Online Shop. It also gives the user viewpoint of product, specific requirements of the product, external interface requirements and gives detailed description of functional requirements.

# SPECIFIC REQUIREMENTS

The specific requirements are,

**Functionality**

1. Sell Configured to Ordered Products.

* The system shall display all the products that can be configured.
* The system shall allow user (admin) to select the product to configure.
* The system shall display all the available components of the product to configure
* The system shall allow user (admin) to update the configuration to resolve conflict in the current configuration.
* The system shall allow user (admin) to confirm the completion of current configuration

2. Provide comprehensive product details.

* The system shall display detailed information of the selected products.
* The system shall provide browsing options to see product details.

3. Detailed product Categorizations

* The system shall display detailed product categorization to the user.

4. Provide Search facility.

* The system shall enable user to select multiple options on the screen to search.
* The system shall display all the matching products based on the search
* The system shall enable user to navigate between the search results.
* The system shall notify the user when no matching product is found on the search.

5. Maintain customer profile.

* The system shall allow user to create profile and set his credential.
* The system shall authenticate user credentials to view the profile.
* The system shall allow user to update the profile information.

6. Provide personalized profile

* The system shall display both the active and completed order history in the customer profile.
* The system shall allow user to select the order from the order history.
* The system shall display the detailed information about the selected order.

7. Email confirmation.

* The system shall send an order confirmation to both seller and customer through email.
* The system shall maintain customer email information as a required part of customer profile.

8. Detailed invoice for customer.

* The system shall display detailed invoice for current order once it is confirmed.
* The system shall optionally allow user to print the invoice.

9. Provide shopping cart facility.

* The system shall provide shopping cart during online purchase.
* The system shall allow user to add/remove products in the shopping cart.

10. Online tracking of shipments

* The system shall display the current tracking information about the order.

11. Allow multiple payment methods.

* The system shall display available payment methods for payment.
* The system shall allow user to select the payment method for order.

12. Allow online change or cancellation of order.

* The system shall display the orders that are eligible to change.
* The system shall allow user to select the order to be changed.
* The system shall allow user to cancel the order

13. Provide detailed sitemap.

* The system shall allow user to view detailed sitemap.

14. Online Purchase of products.

* The system shall allow user to confirm the purchase.

## Usability

1. Graphical User Interface

* The system shall provide a uniform look and feel between all the web pages.
* The system shall provide a digital image for each product in the product catalog.
* The system shall provide responsive design for mobile, tablet and desktop.

## Performance

1. The product shall be based on web and has to be run from a web server.

The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run.

The performance shall depend upon hardware components of the client/customer

## Security

1. Data Transfer

* The system shall automatically log out all customers after a period of inactivity.
* The system shall confirm all transactions with the customer’s web browser.
* The system shall not leave any cookies on the customer’s computer containing any of the user’s confidential information such as password.
* The system shall use secure HTTP requests in all transactions that include any confidential customer information.

2. Data Storage

* The customer’s web browser shall never display a customer’s password. It shall always be echoed with special characters representing typed characters.
* The customer’s web browser shall never display a customer’s credit card number after retrieving from the database. It shall always be shown with just the last 4 digits of the credit card number.
* The system’s back-end servers shall never display a customer’s password. The customer’s password may be reset but never shown.
* The system’s back-end servers shall only be accessible to authenticated administrators.

## Design Constraints

1. Standard Development Tools

* The system shall be built using a standard web page development tool that conforms to recommendations published by the World Wide Web Consortium (W3C)

2. Web Based Product

* There are no memory requirements
* The computers must be equipped with web browsers such as Google Chrome.
* The product must be stored in such a way that allows the client easy access to it.
* Response time for loading the product should take no longer than five minutes.
* A general knowledge of basic computer skills is required to use the product

## Interfaces

There are many types of interfaces as such supported by the Farm To Home Organic Online Shop software system namely; User Interface, Software Interface and Hardware Interface.

The protocol used shall be HTTP.

The Port number used will be 8080.

There shall be logical address of the system in IPv4 format.

1. User Interfaces

* The user interface for the software shall be compatible to any browser such as Google Chrome, Mozilla Firefox by which user can access to the system.
* The user interface shall be implemented using any tool or software package like JavaEE

2. Hardware Interfaces

* Since the application must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. As for e.g. Modem, WAN – LAN, Ethernet Cross-Cable.

3. Communications Interfaces

* The Farm To Home Organic Online Shop system shall use the HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.

## Licensing Requirements

Not Applicable

## Legal, Copyright, and Other Notices

Not Applicable

## Applicable Standards

It shall be as per the industry standard.

**SOFTWARE ENGINEERING PARADIGM APPLIED**

The software engineering paradigm which is also referred to as a software process model or Software Development Life Cycle (SDLC) model is the development strategy that encompasses the process, methods and tools. SDLC describes the period of time that starts with the software system being conceptualized and ends with the software system been discarded after usage.

The objectives of the use of software engineering paradigms include:

* The software development process becomes a structured process.
* Determine the order of states involved in software development and evolution, and to establish the transitions criteria for the next stage.
* The software engineering paradigm provides the guidance to the software engineer.
* A paradigm specifies the particular approach or philosophy for designing, building and maintaining software. Each paradigm has its own advantages and disadvantages which make some paradigm more suitable to be used in developing a given software system in a given situation than another.
* The techniques, tools procedures and methods in developing software systems are heavily dependent on the selected paradigm.
* There are common software process tasks, phases and activities that are modeled by software models. They are heavily affected by selected software paradigms. These tasks, phases and activities include:
* Requirements Engineering: Software specification and functional requirements obtained from the user.
* Requirements Analysis and Modeling
* Architectural Engineering, implementation and Design: Production of the software system as a product
* Software Testing and Validation: Activity that assures that customer specifications are met
* System Delivery.
* Software Evolution and Upgrading: System modification to meet continuing customer needs
* System Documenting
* Maintenance

Waterfall Model has applied in Farm To Home Organic Online Shop,

The waterfall model is also known as the linear sequential model and as classic life cycle model. This model was developed in 1970 to make the software development process more structured. It is the oldest and most commonly used software engineering.

It demands a systematic, sequential approach to software development that begins at the requirements analysis through planning, design, coding, testing, and delivery which are processed in a linear fashion. The steps in the waterfall mode are: requirements analysis, planning, design, coding, testing, and delivery. Brief description of the phases in the waterfall model is presented below.

## **Requirements Analysis**

In this phase the requirements of the software system are defined. This answers the ‘What is the software system to be developed’ question.

## **Design**

The design phase is an intermediate phase and bridge between ‘What’ the user/customer wants and the implemented system (code) that will be created to satisfy the requirements.

## Coding

This stage of the model involves the writing of the code. The actual design is turned into a set of programs.

## Testing

The code that is developed in the implementation phase is tested during the testing phase. This involves unit testing for the lowest level components, integration testing for groups of components and testing of the system as a whole.

## Delivery

In this phase the system is made operational. Main activities in this phase include training of the user’s, and installation of the system. Also, in this phase the system is maintained where bugs, errors, and defects are corrected. The system is made more efficient, new requirements are added and existing functionality and features modified to meet the changing customer/user/business needs.

SYSTEM DESIGN

This is the most creative and challenging face of the system development. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Design goes through the logical and physical stages of development. When designing a new system, the system analyst must have a clear understanding of the objectives of it. The first step is to determine how the output is to be produced and in what format. Second, input data and master files have to be designed to meet the requirements of the proposed output. The operational phases are handled through program construction and testing. For the development of this Farm To Home Organic Online Shop, modular approach would be used. Modularization is the process of partitioning the system into modular units of limited size to simplify enhancement and maintenance when necessary. Each and every module is selected according to the function it is expected to perform. Modularization makes system development simpler.

OUTPUT DESIGN

One of the most important features of the system for users is the output it produces. Output design should aim at producing quality products out of the system. System output could be displayed on the monitor for immediate need, or be created as a hard copy report or summary. The objective of output design is to define the formats of all documents and other displays that will be produced by the system. System output is the most important and direct source of information to the user. For many end users output is the main reason for developing the system and the basis on which they will evaluate the usefulness of the application.

INPUT DESIGN

In the input design, user oriented inputs are converted into a computer based system format. It also includes determining the record media, method of input, speed of capture and entry on to the screen. The major approach to input design is the menu and prompts design. The quality of system inputs determines the quality of system outputs. All data entry screens should be of interactive and user friendly. The input design determines how the user interacts with the system. Interactive input screens ensure the reliability and accuracy of the system. Input data is collected, and similar kind of data is grouped. Data can either be entered through the keyboard or pointed or selected with a mouse. Input design also aims to minimize the data entry errors for efficient system performance. The goal of designing input data is to make it free from logical errors.

In this proposed system, all inputs would be validated for integrity and correctness. In case of any deviation from the expected format, the user would be warned by a message and asked to retry. Data that passes all checks are transferred to the system for processing. Forms and UI design comprise another aspect of inputs design. Forms control information that is entered into the system by the user.

MODULARIZATION DETAILS

Modular design is the process of constructing a system into modular units of limited size to simplify maintenance when necessary. It is an attempt to minimize the complexity and make a problem manageable by subdividing it into smaller segments. The advantages are:

* It is easy to write, test and debug a module
* If a change is to be made, it is done in the particular module and entire program is not affected.
* Generally, modules of common nature are prepared, which can be used at many places.

Our application Farm To Home Organic Online Shop is a modular application. A number of patterns are driven by making systems modular .Modules are abstractions to make software easier to write and to maintain, and one of the most important pattern in application that faces users is called Model-View-Controller (MVC).

* Model: All data items are representations of the real things; they make up the Model.
* View: A View is component that transforms a model into something visible. A View is both input and output. A View is also a filter showing the interesting parts only [to a user].
* Controller: The Controller sets up and coordinates one or more Views.

**DATA INTEGRITY AND CONSTRAINTS**

It is important that data adhere to a predefined set of rules, as determined by the database administrator or application developer. As an example of data integrity, consider the tables **product** having **id** is unique and not null. We have some columns in each table have specific rules that constrain the data contained within them. We have incorporated few of them in Farm To Home Organic Online Shop are,

Null Rule :

A null rule is a rule defined on a single column that allows or disallows inserts or updates of rows containing a null (the absence of a value) in that column.

NOT NUL constraints for the rules associated with nulls in a column.

Unique Column Values :

A unique value rule defined on a column (or set of columns) allows the insert or update of a row only if it contains a unique value in that column (or set of columns).

UNIQUE key constraints for the rule associated with unique column values.

Primary Key Values :

A primary key value rule defined on a key (a column or set of columns) specifies that each row in the table can be uniquely identified by the values in the key.

PRIMARY KEY constraints for the rule associated with primary identification values.

Referential Integrity Rules :

A referential integrity rule is a rule defined on a key (a column or set of columns) in one table that guarantees that the values in that key match the values in a key in a related table (the referenced value).

FOREIGN KEYconstraints for the rules associated with referential integrity. Oracle supports the use of FOREIGN KEY integrity constraints to define the referential integrity actions, including:

* Update and delete No Action
* Delete CASCADE
* Delete SET NULL
* CHECK constraints for complex integrity rules

**DATABASE DESIGN**

A database is similar to a data file, which is the storage place for data Database systems are more powerful than data files and are highly organized. In a well-developed database, there are no duplicate pieces of data that the user or application must update at the same time. Related pieces of data are grouped together in a single structure or record and relationships can be defined between these structures and records. The database design is made-up of two levels.

* Conceptual Level
* Normalization

This level represents the major data objects and relationship between them. Conceptual level describes the essential feature of the system data. After the conceptual level, the next level of process of database design to organize the data structure into a good shape is called Normalization. The normalization simplifies the entries, removing the redundancies from the system data and finally builds a data structure, which is both flexible and adaptable to the system. The set of tables planned for the new proposed system are mentioned later in this report.

Table : PRODUCT

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| ProductID | INT | Primary Key |
| ProductName | VARCHAR(25) | Not Null |
| ProductUnitPrice | DECIMAL(4,2) | Not Null |
| ProductUnitWeight | INT | Not Null |
| ProductDescription | VARCHAR(50) | Not Null |
| ProductImage | VARCHAR(50) | Not Null |
| ProductThumbImage | VARCHAR(50) | Not Null |
| ProductUpdateDate | DATETIME | Not Null |
| Active | BOOLEAN | Not Null |
| Currency | varchar(4) | Not Null |
| CategoryID | INT | Foreign Key |

Table : CATEGORY

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| CategoryID | INT | Primary Key |
| CategoryName | VARCHAR(25) | Not Null |
| Description | VARCHAR(50) | Not Null |
| CategoryImage | VARCHAR(50) | Not Null |
| Active | BOOLEAN | Not Null |

Table : SELLERSPRODUCT

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| SellersProductID | INT | Primary Key |
| SellersStock | INT | Not Null |
| StockStatus | VARCHAR(10) | Not Null |
| SellersDiscount | DECIMAL(2,2) | Not Null |
| ProductID | INT | Foreign Key |
| SellerID | INT | Foreign Key |

Table : SELLER

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| SellerID | INT | Primary Key |
| About | VARCHAR(50) | Not Null |
| LogoImage | VARCHAR(50) | Not Null |
| RegistrationNumber | VARCHAR(25) | Not Null |
| CompanyName | VARCHAR(25) | Not Null |
| URL | VARCHAR(50) | Not Null |
| PaymentMethods | VARCHAR(50) | Not Null |
| ContactTitle | VARCHAR(50) | Not Null |
| UserID | INT | Foreign Key |

Table : DELIVERYADDRESS

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| id | INT | Primary Key |
| addressLine1 | VARCHAR(50) | Not Null |
| addressLin2 | VARCHAR(50) | Not Null |
| addressLine3 | VARCHAR(25) | Not Null |
| city | VARCHAR(25) | Not Null |
| state | VARCHAR(50) | Not Null |
| country | VARCHAR(50) | Not Null |
| pinCode | decimal(6,0) | Not Null |
| userId | INT | Foreign Key |

Table : ORDER

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| OrderID | INT | Primary Key |
| OrderDate | DATE | Not Null |
| OrderAmount | DECIMAL(6,2) | Not Null |
| Freight | DECIMAL(2,2) | Not Null |
| Tax | DECIMAL(2,2) | Not Null |
| OrderShipName | VARCHAR(50) | Not Null |
| OrderShipAddress1 | VARCHAR(50) | Not Null |
| OrderShipAddress2 | VARCHAR(50) | Not Null |
| OrderCity | VARCHAR(50) | Not Null |
| OrderState | VARCHAR(25) | Not Null |
| OrderPinCode | VARCHAR(6) | Not Null |
| OrderCountry | VARCHAR(25) | Not Null |
| OrderPhone | VARCHAR(10) | Not Null |
| OrderMobile | VARCHAR(10) | Not Null |
| OrderEmail | VARCHAR(25) | Not Null |
| Deleted | BOOLEAN | Not Null |
| TransactionStatus | VARCHAR(25) | Not Null |
| Paid | BOOLEAN | Not Null |
| ErrMsg | VARCHAR(50) | Not Null |
| ErrLoc | VARCHAR(50) | Not Null |
| Refund | BOOLEAN | Not Null |
| PaymentID | INT | Foreign Key |
| OrderUserID | INT | Foreign Key |

Table : ORDERDETAILS

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| DetailIOrderID | INT | Primary Key |
| DetailPrice | DECIMAL(6,2) | Not Null |
| DetailQuantity | INT | Not Null |
| Total | DECIMAL(6,2) | Not Null |
| ProductID | INT | Foreign Key |
| SellerID | INT | Foreign Key |
| OrderID | INT | Foreign Key |
| ReviewID | INT | Foreign Key |

Table : CART

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| CartItemID | INT | Primary Key |
| DateCreated | BOOLEAN | Not Null |
| Deleted | BOOLEAN | Not Null |
| Qty | INT | Not Null |
| ProductID | INT | Foreign Key |
| SellerID | INT | Foreign Key |
| UserID | INT | Foreign Key |

Table : PAYMENTDETAILS

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| PaymentID | INT | Primary Key |
| PaymentMode | VARCHAR(20) | Not Null |
| CreditCardTypeID | VARCHAR(50) | Not Null |
| CreditCardExpMonth | INT | Not Null |
| CreditCardExpYear | INT | Not Null |
| CreditCardNumber | VARCHAR(25) | Not Null |
| OrderID | INT | Foreign Key |

Table : REVIEW

|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| ReviewID | INT | Primary Key |
| ReviewDescription | VARCHAR(100) | Not Null |
| Rating | INT | Not Null |
| TimeStamp | DATETIME | Not Null |

Table : USER

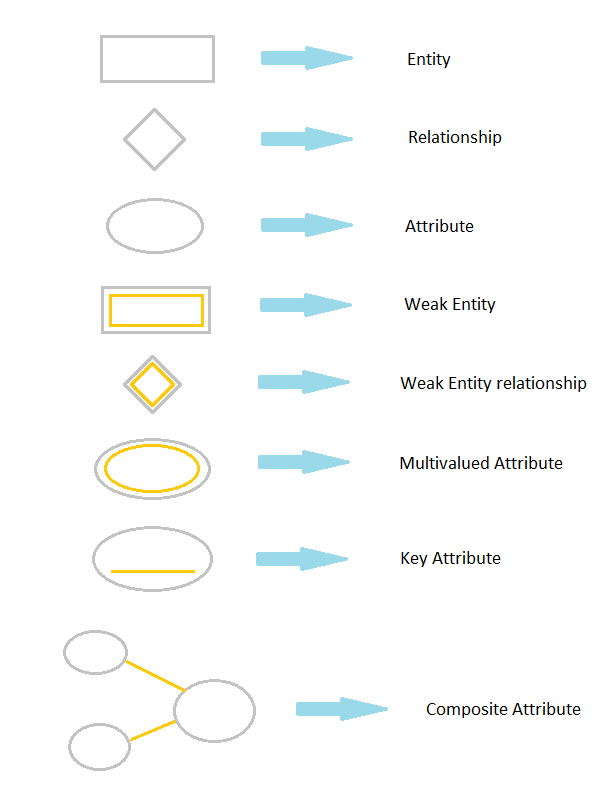
|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| UserID | INT | Primary Key |
| FirstName | VARCHAR(50) | Not Null |
| LastName | VARCHAR(50) | Not Null |
| Address1 | VARCHAR(50) | Not Null |
| Address2 | VARCHAR(50) | Not Null |
| City | VARCHAR(25) | Not Null |
| State | VARCHAR(25) | Not Null |
| Country | VARCHAR(25) | Not Null |
| PinCode | VARCHAR(6) | Not Null |
| Phone | VARCHAR(10) | Not Null |
| Mobile | VARCHAR(10) | Not Null |
| Email | VARCHAR(25) | Not Null |
| Password | VARCHAR(16) | Not Null |
| UserIP | VARCHAR(15) | Not Null |
| DateCreated | DATETIME | Not Null |
| Role | VARCHAR(10) | Foreign Key |

Table : ROLE

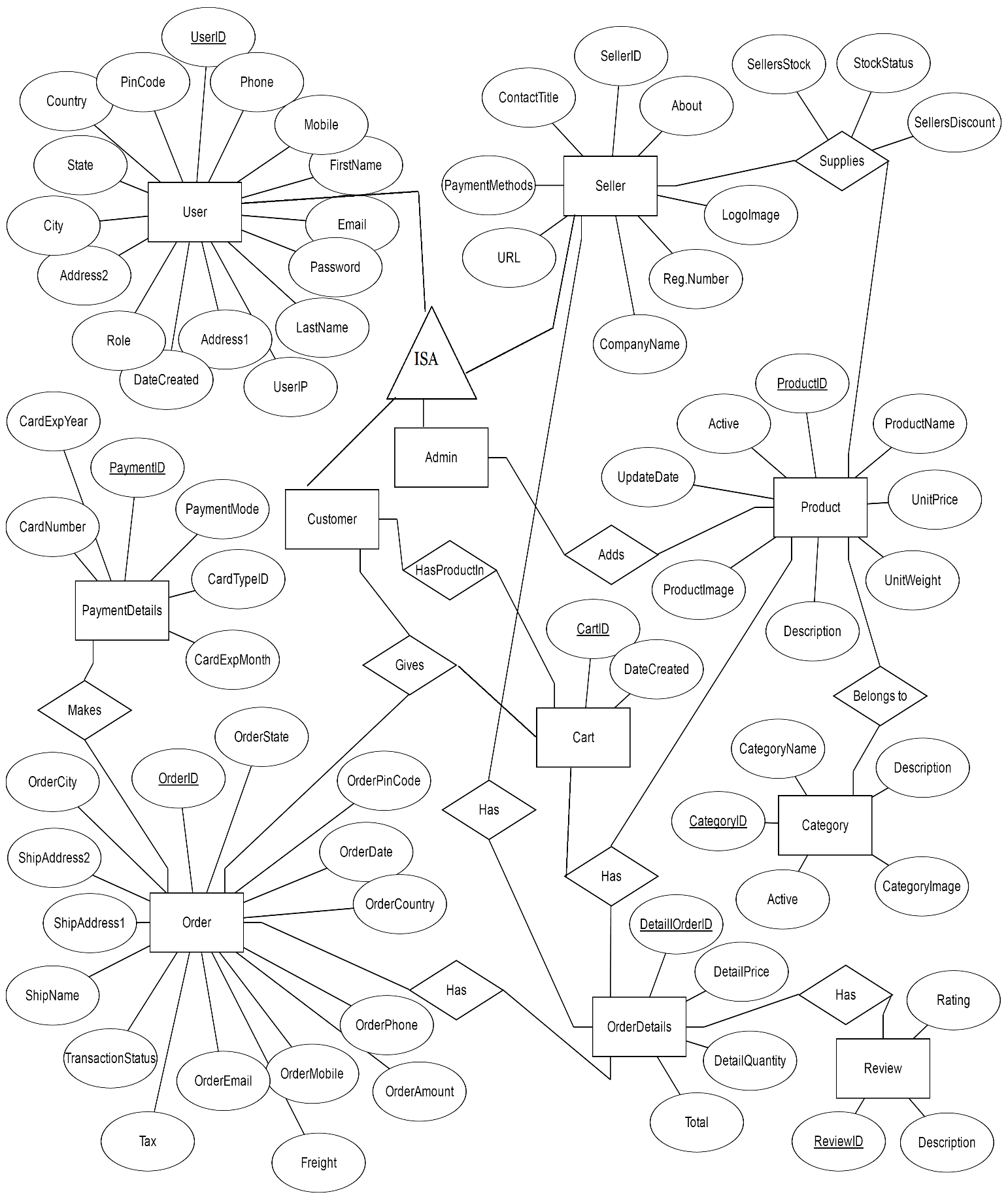
|  |  |  |
| --- | --- | --- |
| NAME | DATATYPE | CONSTRAINT |
| id | INT | Primary Key |
| RoleName | VARCHAR(45 ) | Not Null |
| RoleDesc | VARCHAR(45 ) | Not Null |

# ENTITY RELATIONSHIP DIAGRAM

# An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. It works around real-world entities and the associations among them. In other words, ER diagrams illustrate the logical structure of databases. Some of the symbols and notations to representing ERD shown below,

****

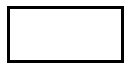
# ERD for **Farm To Home Organic Online Shop**



**DATA FLOW DIAGRAMS**

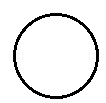
# Data flow diagrams (DFD) are graphical representation of the flow of data through an information system. The Data Flow Diagram shows the logical flow of a system and defines the boundaries of the system. For a candidate system it describes the inputs, outputs, databases and procedures. A Data Flow Diagram, also known as Bubble Chart is the best and easiest tool to represent the flow of the data in the project. It has the purpose of clarifying system requirements and identifying major transformations that will become programs during actual system design. It is the major starting point in the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. A DFD consists of a series of bubbles joined by lines. The bubble represents data transformation and lines represent data flow in the system.

# To better understand a DFD, a legend of symbols used in its construction is given below:



A **rectangle or square** is used to represent source or destination of data.

An **arrow** represents flow of data.



A **circle or ellipse** represents a process that transforms data.



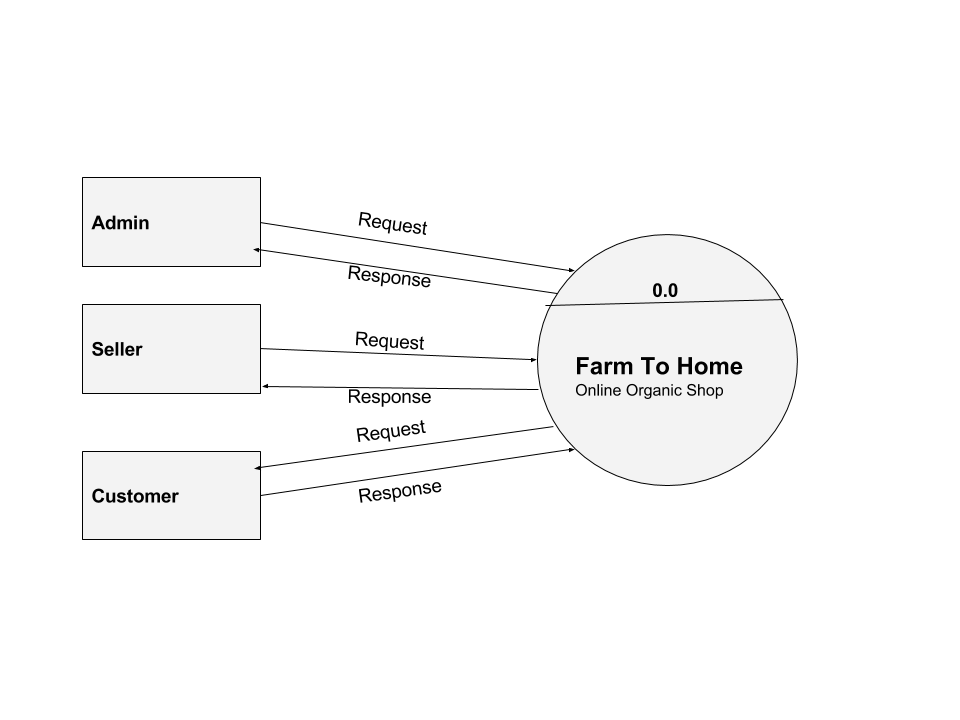
An **open rectangle** represents a data store.

The DFD at the simplest level is referred to as the zero level DFD or in simple words a Context Analysis Diagram, These are expanded level by level each explaining its process in detail. Processes are numbered for easy identification and are normally labelled in block letters. Each data flow is labelled for easy understanding.

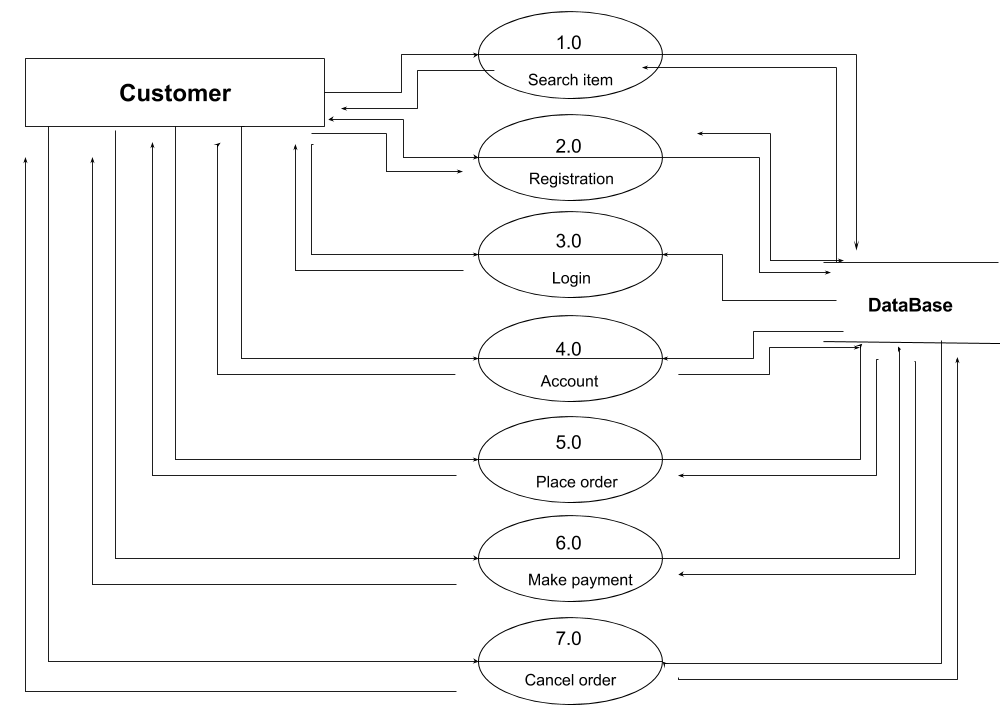
## PROPOSED SYSTEM DFDs

The DFDs for the proposed system are depicted as various levels to show the functionalities in detail. Please note that only the most important modules have been represented below as of now.

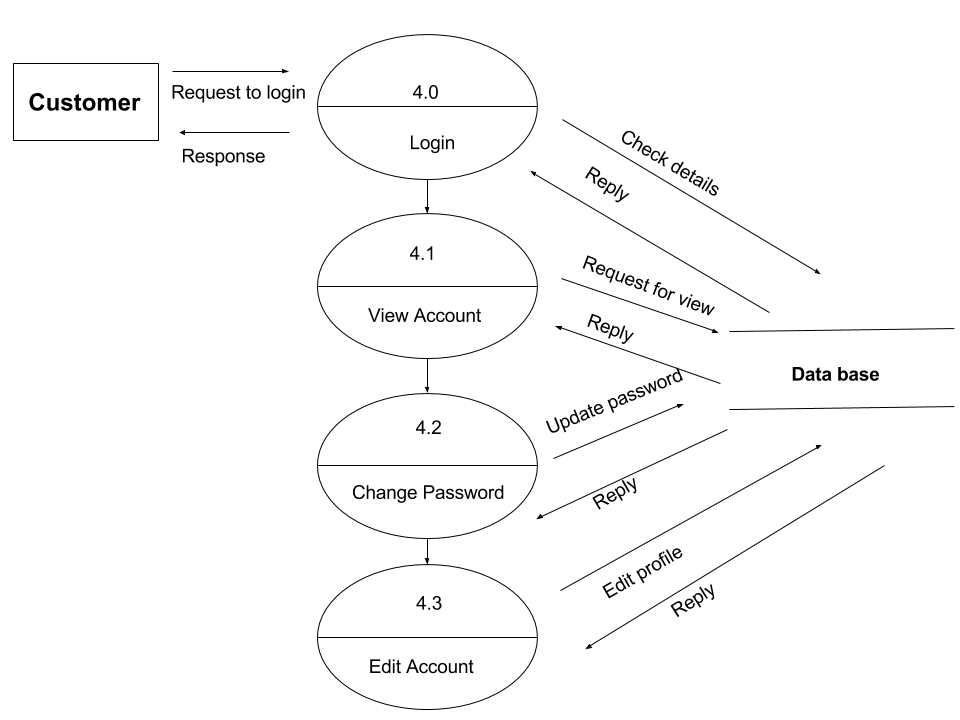
**Zero Level DFD**



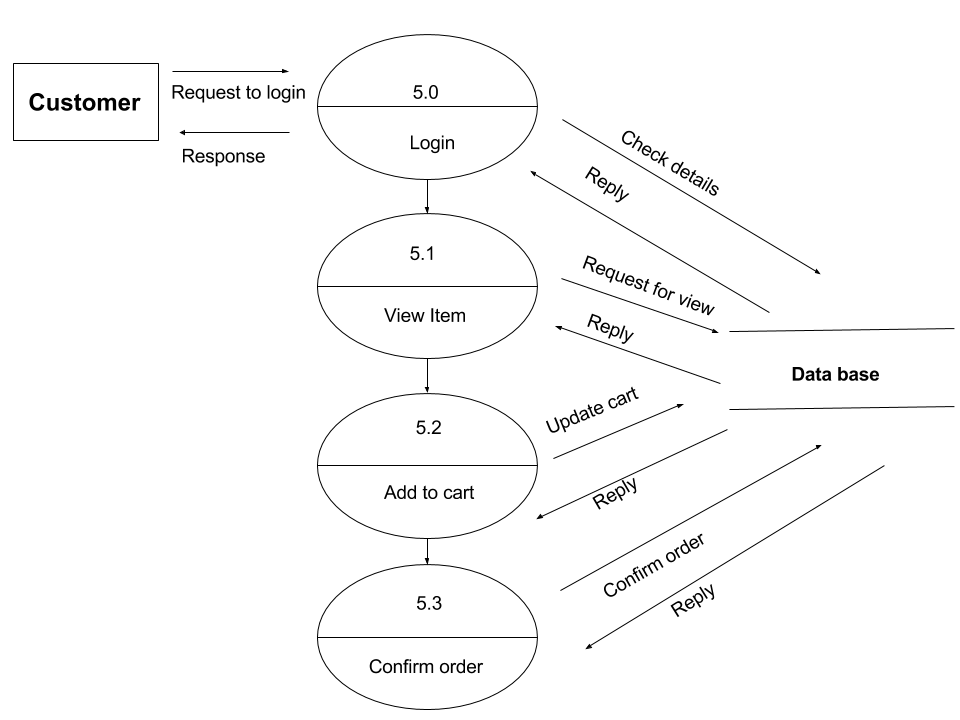
**Customer Module 1st Level DFD**



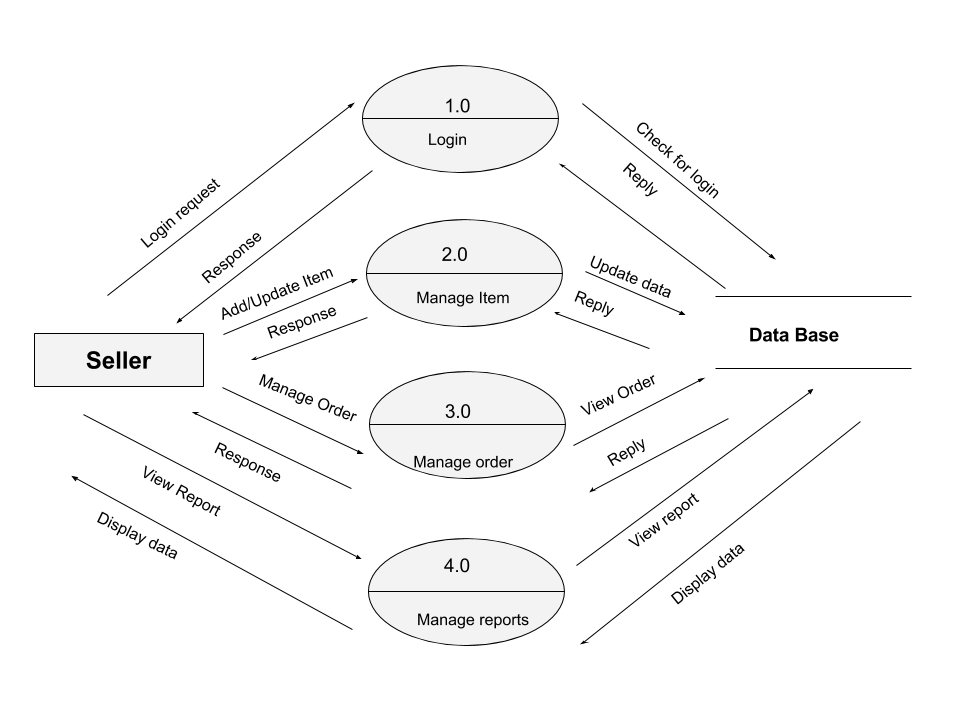
**Customer Module 2nd Level DFD 4**

****

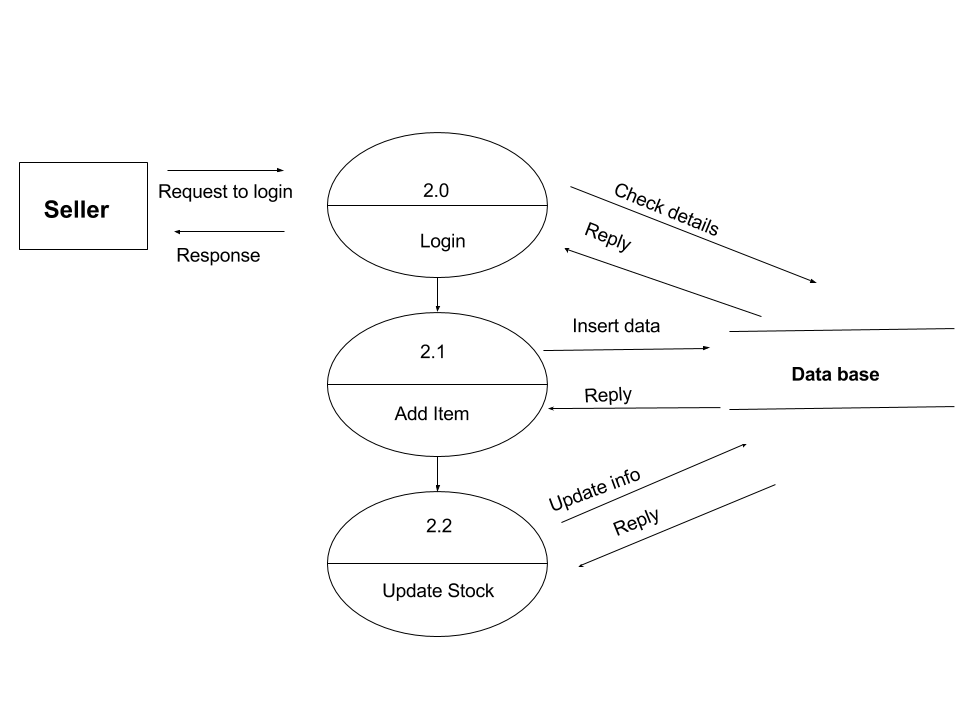
**Customer Module 2nd Level DFD 5**

****

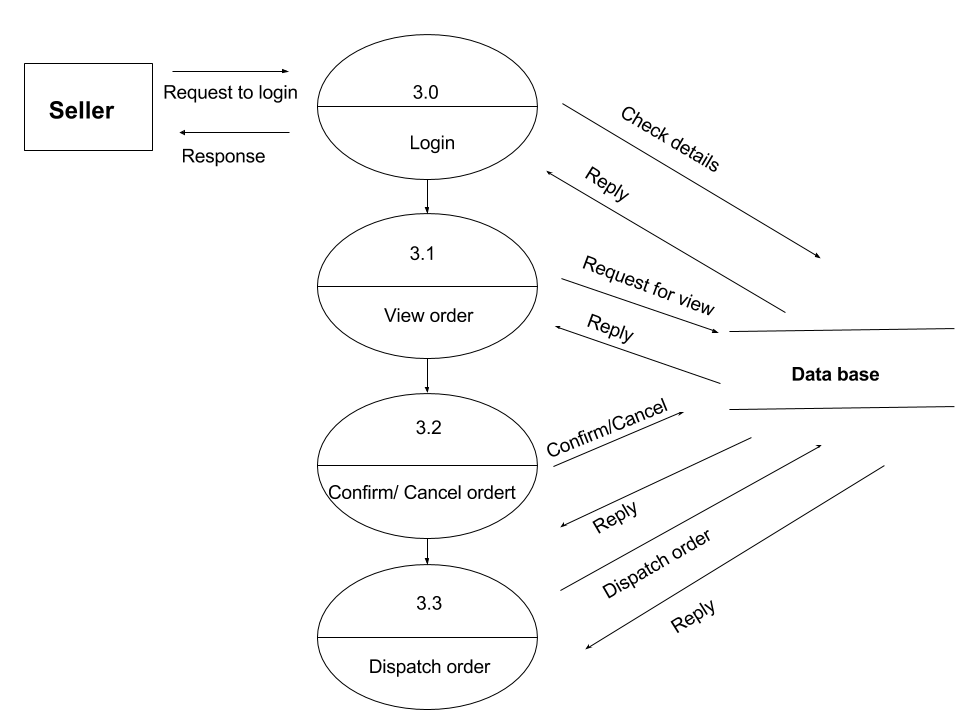
**Seller Module 1st Level DFD**

****

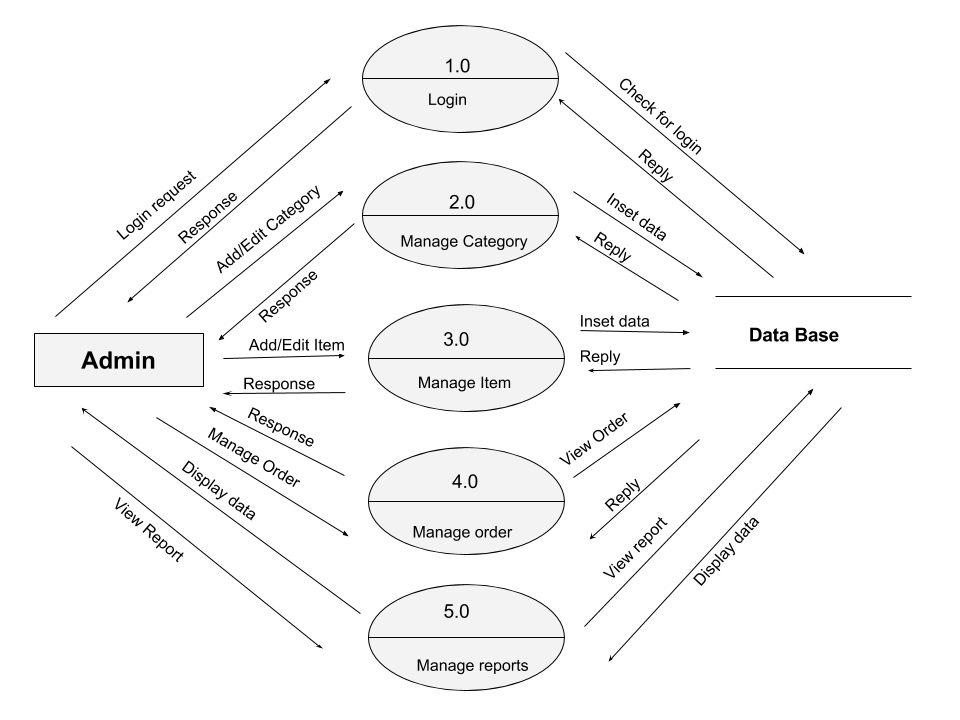
**Seller Module 2nd Level DFD 2**

****

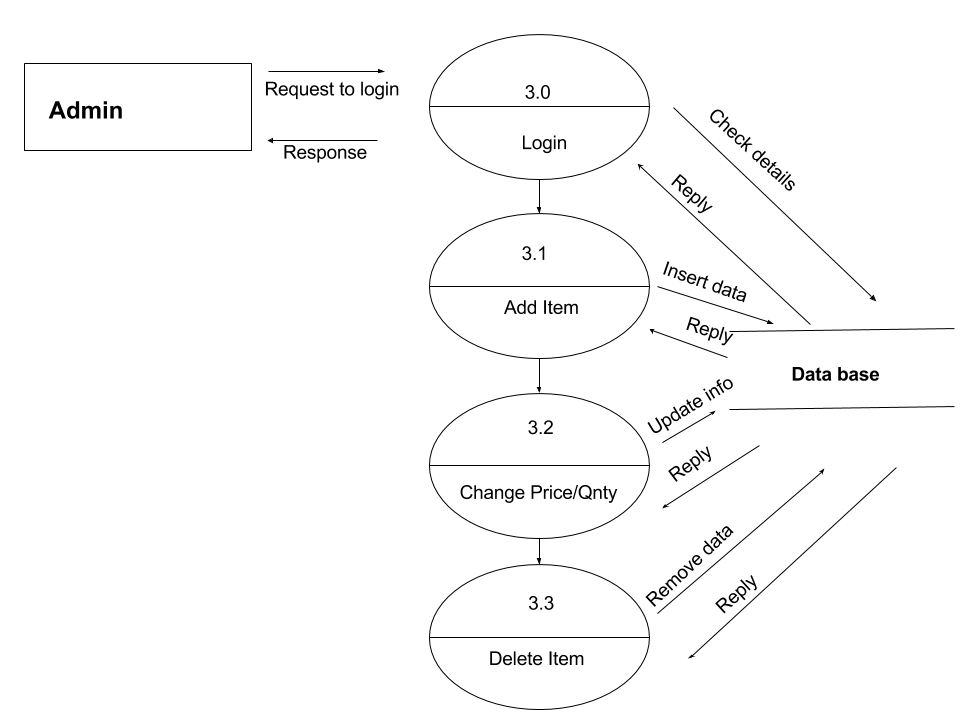
**Seller Module 2nd Level DFD 3**

****

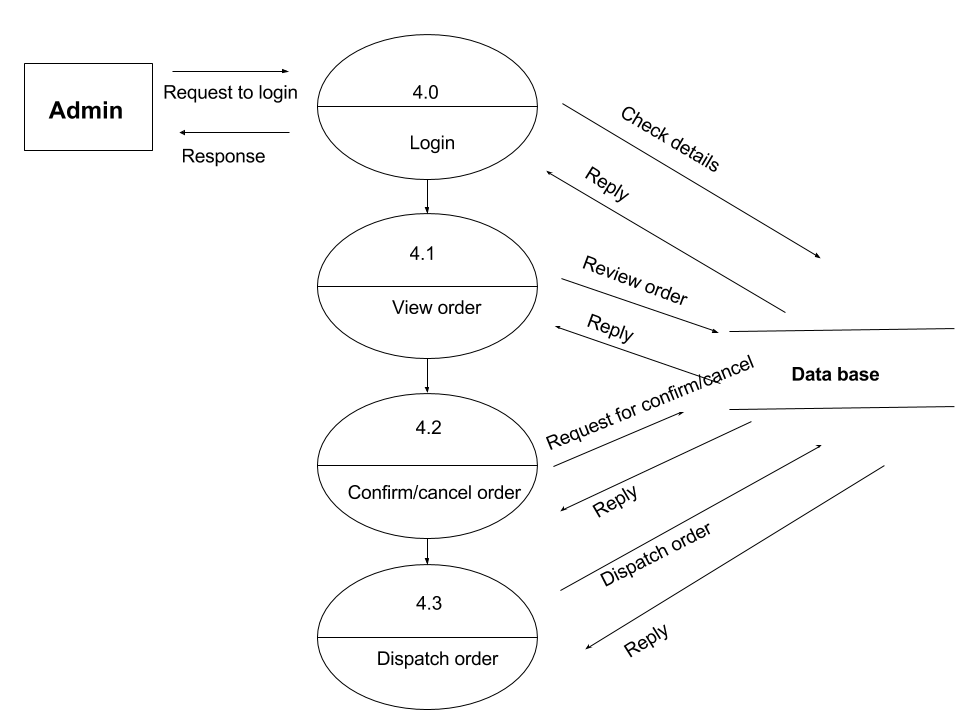
**Admin Module 1st Level DFD**

****

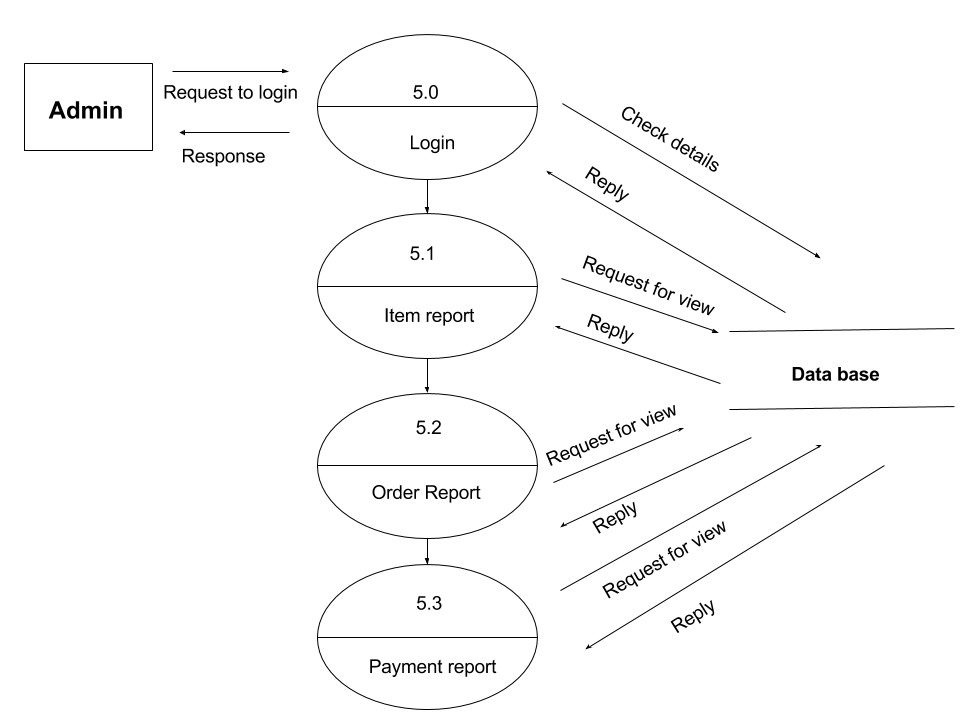
**Admin Module 2nd Level DFD 3**

****

**Admin Module 2nd Level DFD 4**

****

**Admin Module 2nd Level DFD 5**

****

**USER INTERFACE DESIGN**

User interface is the front-end application view to which user interacts in order to use the software. User can manipulate and control the software as well as hardware by means of user interface. Today, user interface is found at almost every place where digital technology exists, right from computers, mobile phones, cars, music players, airplanes, ships etc.

User interface is part of software and is designed such a way that it is expected to provide the user insight of the software. UI provides fundamental platform for human-computer interaction.

UI can be graphical, text-based, audio-video based, depending upon the underlying hardware and software combination. UI can be hardware or software or a combination of both.

The software becomes more popular if its user interface is:

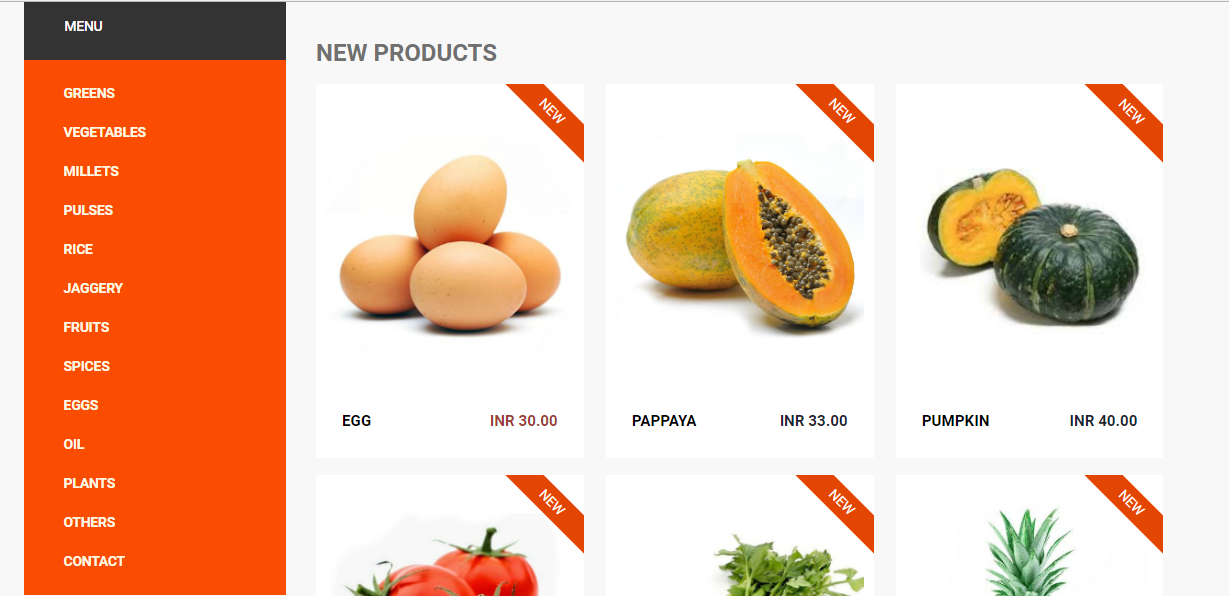
* Attractive
* Simple to use
* Responsive in short time
* Clear to understand
* Consistent on all interfacing screens

In **Farm To Home Organic Online** **Shop,** UI is designed by HTML, JavaScript and Cascading Style sheets (CSS).CSS are commonly used for design elements but can incur difficulties with browser compatibilities, therefore usually good for positioning and font styles.

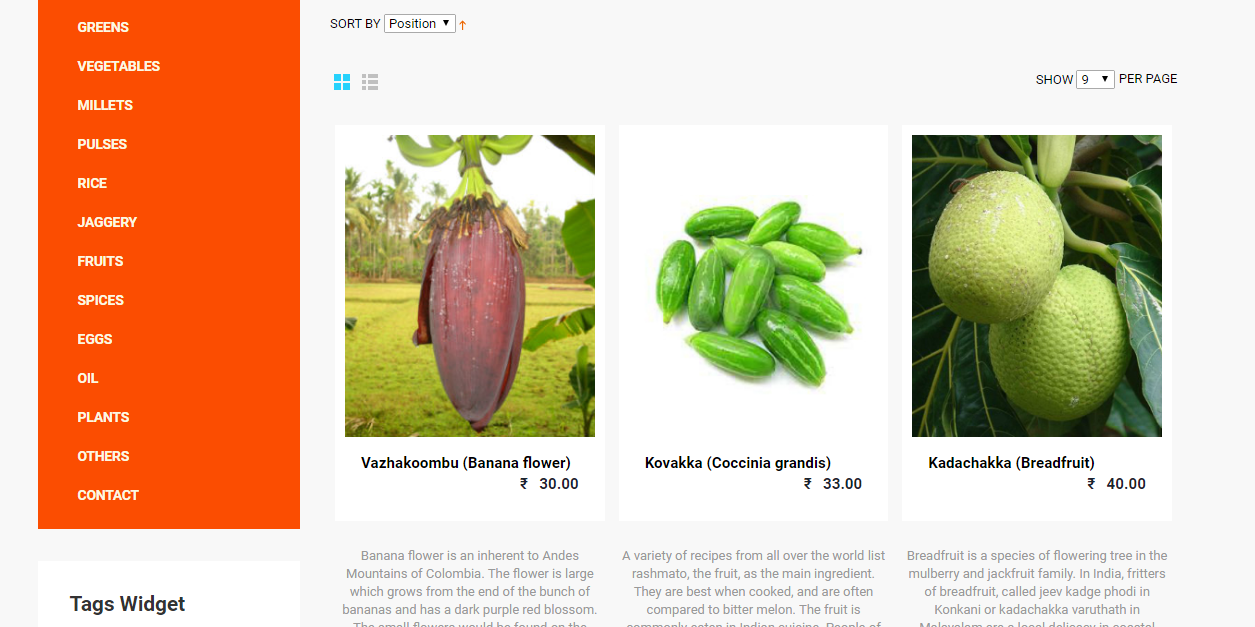
Few of the Graphical design of **Farm To Home Organic Online** **Shop shown** below**,**



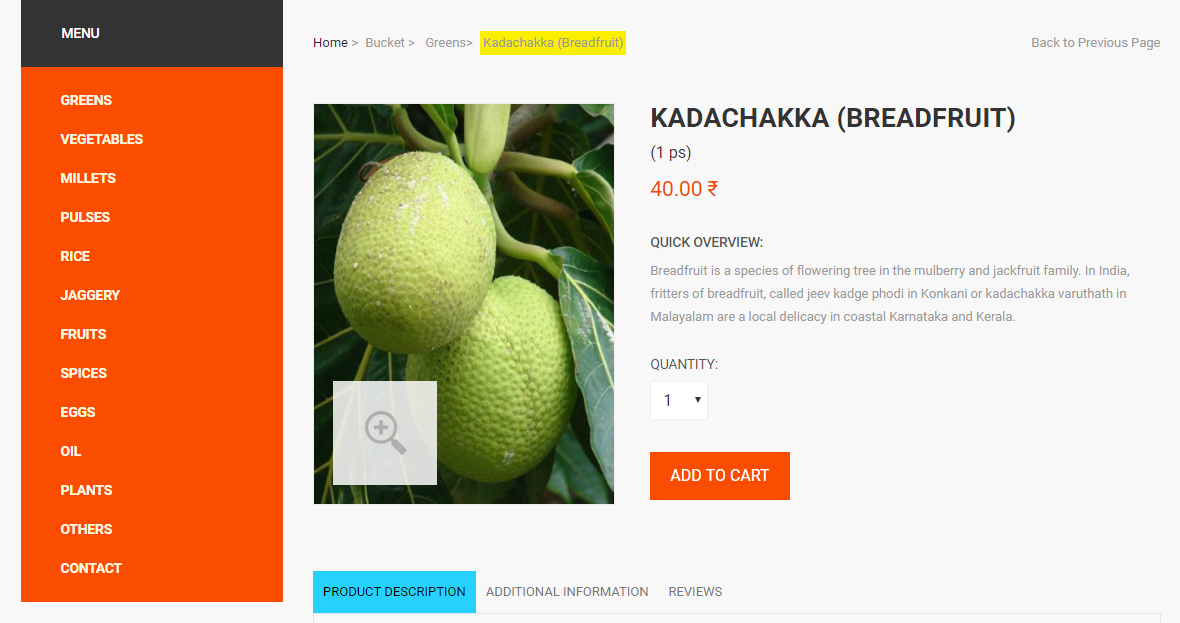
**Home Page**



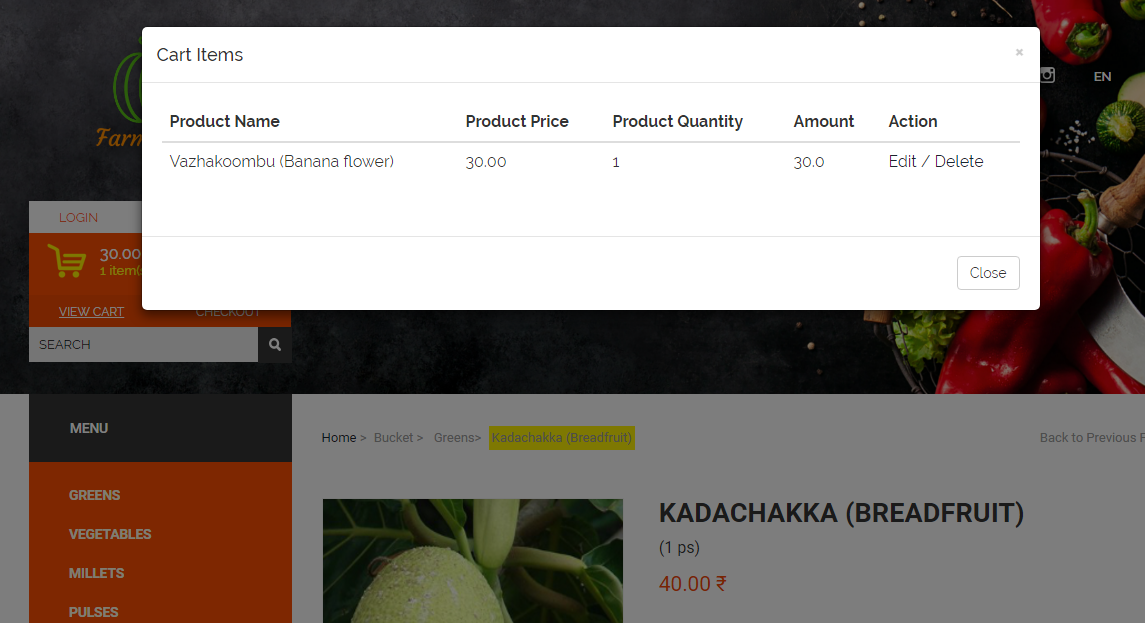
Product page…



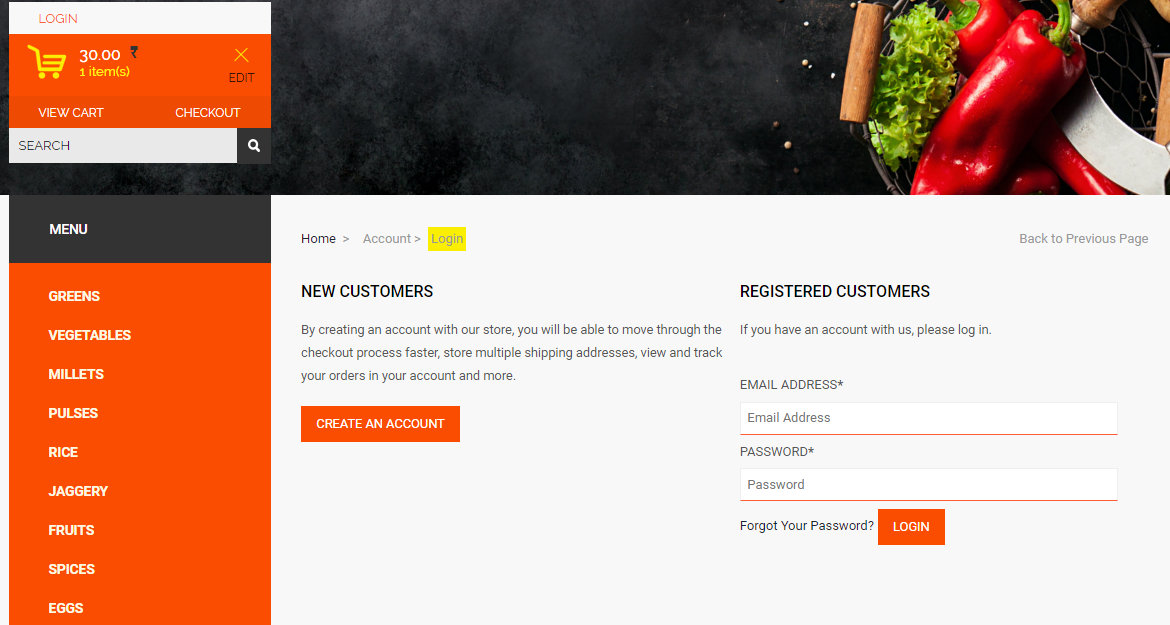
Product review page…



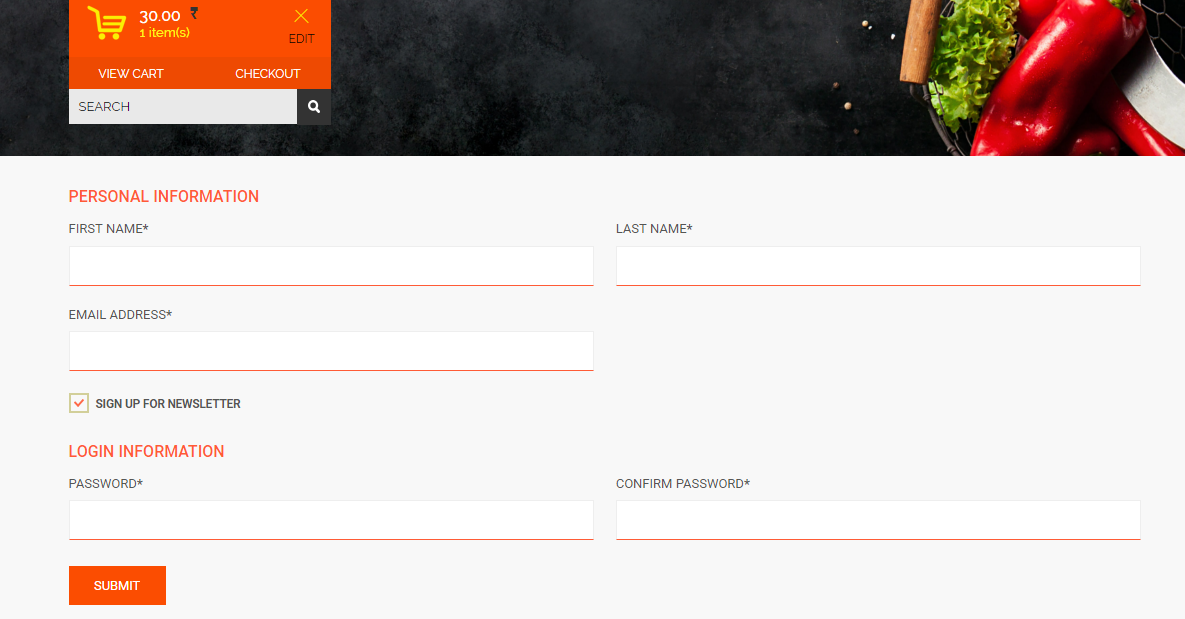
View cart items…



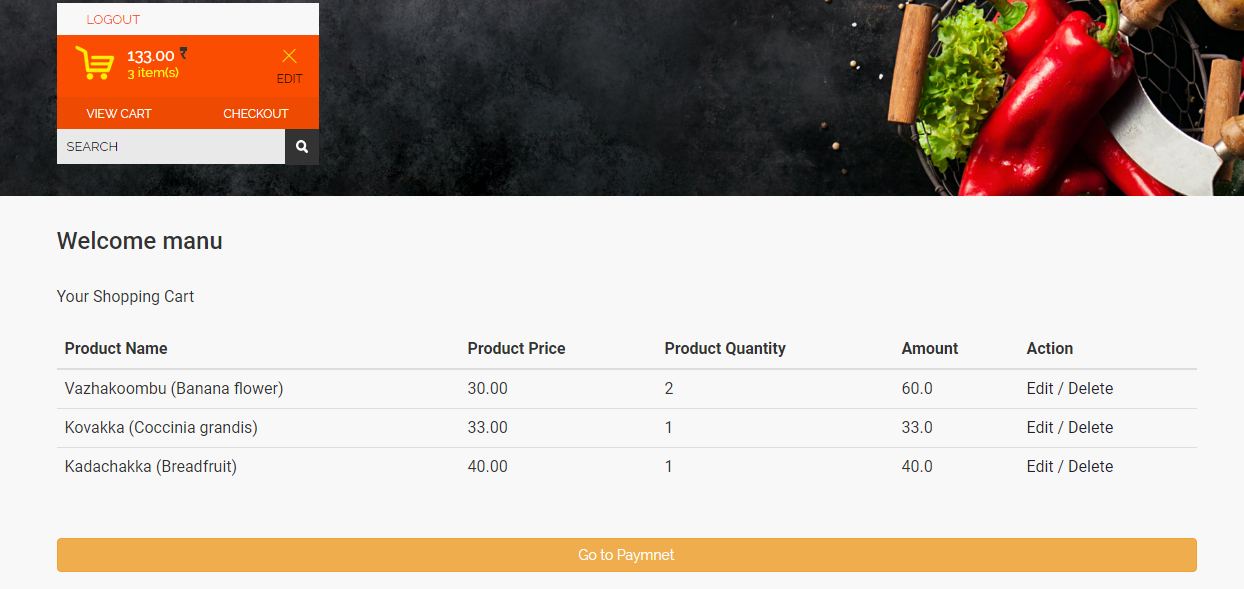
Login page…



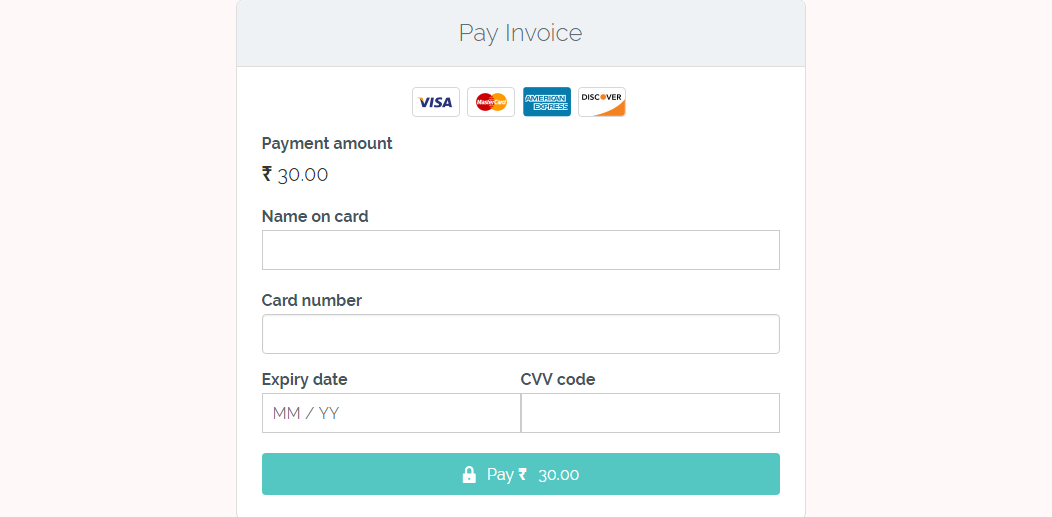
Logout page...



Check-out page…



Payment page…



**Test Cases**

Test case in simple terms refers to a documentation which specifies input, pre-conditions, set of execution steps and expected result. A good test case is the one which is effective at finding defects and also covers most of the scenarios/combinations on the system under test.

**Unit Testing**

* Unit testing is a type of testing to check if the small piece of code is doing what it is supposed to do.
* Unit testing checks a single component of an application.
* The scope of Unit testing is narrow, it covers the Unit or small piece of code under test. Therefore while writing a unit test shorter codes are used that target just a single class.
* Unit tests should have no dependencies on code outside the unit tested.
* This is first type of testing is to be carried out in Software testing life cycle and generally executed by developer.
* Unit testing is not further sub divided into different types.
* Unit testing is starts with the module specification.
* The detailed visibility of the code is comes under Unit testing.
* Unit testing mainly focus on the testing the functionality of individual units only and does not uncover the issues arises when different modules are interacting with each other.
* The goal of Unit testing is to test the each unit separately and ensure that each unit is working as expected.
* Unit testing comes under White box testing type.

**System Test**

System testing is aimed at ensuring the system works accurately before live operation commences. Testing is vital to the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subjected to a verity of tests: Online Response, Volume, Stress Recovery & Security and Usable tests. A series of testing are performed for the proposed system before the system is ready for user acceptance testing. Nothing is complete without testing, as it is vital success of the system.

Main test cases for Farm To Home Organic Online Shop,

We have few pages have to tested,

* Home page
* Products page
* Product review page
* Check-out page
* Payment page
* Confirmation page
* Login page
* Register page
* Admin pages and so on.

Below, we have listed important segments and test cases for each page

**Home page**

* Home page fonts and layout should proper.
* All links should present such as “login”,”view cart”,”checkout”,”edit” etc
* Check category menu
* Check “New product”, “Top product” and “Recommended product ”
* Check tag widget
* Check product images are proper
* Check whether page is redirecting to “Products” page while clicking on any category menu item.
* Check whether page is redirecting to “Product review” page while clicking on any products.
* Check “login”,”view cart”,”checkout”,”edit” links works as intended.

**Products page**

* Checkpage fonts and layout should proper.
* Check All links should present such as “login”,”view cart”,”checkout”,”edit” etc.
* Check category menu.
* Check tag widget.
* Check sorting of products.
* Check list view and grid view of products.
* Check product image, name, description and amount along with rupee symbol.
* Check bred crump is available and correct.
* Check whether page is redirecting to “Product review” page while clicking on any products.
* Check “login”,”view cart”,”checkout”,”edit” links works as intended.

**Product review**

* Checkpage fonts and layout should proper.
* Check All links should present such as “login”,”view cart”,”checkout”,”edit” etc.
* Check category menu.
* Check tag widget.
* Check bred crump is available and correct.
* Checkpage fonts and layout should proper.
* Check All links should present such as “login”,”view cart”,”checkout”,”edit” etc.
* Check product image, name, description and amount along with rupee symbol.
* Check quantity select box is present.
* Check “login”,”view cart”,”checkout”,”edit” links works as intended.
* Check quantity “add to cart” button is present.
* Check “YOU MIGHT ALSO LIKE” section is present with image, name and price.
* Check pin code popup appeared on click of “add to cart” button.
* Check whether the item is added into shopping cart.
* Check if the user adds the same item to the cart while continuing to shop, the item count in the shopping cart should get incremented.
* All items and their totals should be displayed in the cart.
* A user can add more items to the cart- total should reflect the same.
* Update the contents added to the cart- total should reflect that too.
* Remove items from the cart.
* Alter item quantity from the cart.
* Proceed to checkout.

**Check-out page**

* Checkpage fonts and layout should proper.
* To access this page user should log in.
* So, login page should appear.
* After successful log in page should redirect to check-out page
* Check All links should present such as “logout”, “view cart”,”checkout”,”edit” etc.
* Able see selected items in a tabular view having name, price, quantity, amount and action can be performed (delete/edit).
* And ”Go to payment page” button for navigate to payment page.

**Payment page**

* Checkpage fonts and layout should proper.
* Check All links should present such as “logout”, “view cart”,”checkout”,”edit” etc.
* Check “login”,”view cart”,”checkout”,”edit” links works as intended.
* Check amount to be paid is present.
* Check all fields are present such as name on card, card number, expiry date, CVV code.

**Confirmation page**

* Checkpage fonts and layout should proper.
* Check order id and details are present.
* Check invoice can be downloaded.
* Check invoice is sent to user’s e-mail id.

**Login page**

* Checkpage fonts and layout should proper.
* Check All links should present such as “login”,”view cart”,”checkout”,”edit” etc.
* Check fields are presents such as “Email address”, “Password” and login button.
* Check “CREATE AN ACCOUNT” button is present.
* Check bred crump is available and correct.
* Check login is working properly.

**Register page**

* Checkpage fonts and layout should proper.
* Check All links should present such as “login”,”view cart”,”checkout”,”edit” etc.
* Check fields are presents such as “First name”, “Last name”, “Email address”, “Password”, “Confirm password” and submit button.
* Check registration is working properly.

CODING

In designing phase we have already taken all the major decision regarding the system, now it’s time to develop the physical system. We will consider designing phase output as input for coding phase. The basic role of this phase is to convert designing into code using the programming language decided in designing phase. The well-developed code in this phase can help to reduce the efforts required in testing and maintenance. But even we make any silly mistake; it may lead us to put extra efforts in testing and maintenance.

If we see it in a business perspective, the cost for testing efforts and maintenance is much higher than coding. So it always makes sense to spend time on coding phase. Here all developers write their own code and merged with other developer’s code to make sure that all modules developed by different developers interact with each other as per expectations. This is one of the longest phases in software development life cycle.

**Technologies used in Coding**

* Apache Maven 3 as Build tool

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

* Apache Tomcat 8 as Application server

The Apache Tomcat® software is an open source implementation of several JavaEE specifications including Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies and provides a "pure [Java](https://en.wikipedia.org/wiki/Java_(programming_language))" [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) [web server](https://en.wikipedia.org/wiki/Web_server) environment in which [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) code can run.

* Eclipse Mars as JavaEE IDE

 Is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), and is the most widely used Java IDE.[[6]](https://en.wikipedia.org/wiki/Eclipse_(software)#cite_note-6) It contains a base [workspace](https://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](https://en.wikipedia.org/wiki/Plug-in_(computing)) system for customizing the environment. Eclipse is written mostly in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) and its primary use is for developing Java applications.

* GitHub as version control repository

Is a web-based [Git](https://en.wikipedia.org/wiki/Git" \o "Git) or [version control repository](https://en.wikipedia.org/wiki/Repository_(version_control)) and [Internet hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service). It offers all of the [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality of Git as well as adding its own features. It provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project.

* Spring Framework 4 as JavaEE framework

The Spring Framework is an [application framework](https://en.wikipedia.org/wiki/Application_framework) and [inversion of control](https://en.wikipedia.org/wiki/Inversion_of_control) [container](https://en.wikipedia.org/wiki/Servlet_container) for the [Java platform](https://en.wikipedia.org/wiki/Java_platform). The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the [Java EE](https://en.wikipedia.org/wiki/Java_EE) platform. Although the framework does not impose any specific [programming model](https://en.wikipedia.org/wiki/Programming_model), it has become popular in the Java community as an alternative to, replacement for, or even addition to the [Enterprise JavaBeans](https://en.wikipedia.org/wiki/Enterprise_JavaBeans) (EJB) model. The Spring Framework is [open source](https://en.wikipedia.org/wiki/Open_source).

**SQL commands for**

Database creation (along with constraints):

CREATE DATABASE IF NOT EXISTS `farmtohome`

-------------------------------------------------------

CREATE TABLE `category` (

`CategoryID` int(11) NOT NULL AUTO\_INCREMENT,

`CategoryName` varchar(45) NOT NULL,

`Description` varchar(45) NOT NULL,

`CategoryImage` varchar(45) NOT NULL,

`Active` tinyint(1) NOT NULL DEFAULT '1',

PRIMARY KEY (`CategoryID`)

) ENGINE=InnoDB AUTO\_INCREMENT=113 DEFAULT CHARSET=utf8;

----------------------------------------------------------------------------

CREATE TABLE `order` (

`OrderID` int(11) NOT NULL,

`OrderDate` datetime NOT NULL,

`OrderAmount` decimal(6,2) NOT NULL,

`Freight` decimal(2,2) NOT NULL,

`Tax` decimal(2,2) NOT NULL,

`OrderShipName` varchar(45) NOT NULL,

`OrderShipAddress1` varchar(45) NOT NULL,

`OrderShipAddress2` varchar(45) DEFAULT NULL,

`OrderCity` varchar(45) NOT NULL,

`OrderState` varchar(45) NOT NULL,

`OrderPinCode` varchar(45) NOT NULL,

`OrderCountry` varchar(45) NOT NULL,

`OrderPhone` varchar(45) NOT NULL,

`OrderMobile` varchar(45) NOT NULL,

`OrderEmail` varchar(45) NOT NULL,

`Deleted` tinyint(1) NOT NULL,

`TransactionStatus` varchar(45) NOT NULL,

`Paid` tinyint(1) DEFAULT NULL,

`ErrMsg` varchar(45) DEFAULT NULL,

`ErrLoc` varchar(45) DEFAULT NULL,

`Refund` tinyint(1) DEFAULT NULL,

`PaymentID` int(11) DEFAULT NULL,

`OrderUserID` int(11) DEFAULT NULL,

PRIMARY KEY (`OrderID`),

KEY `FK\_ORDER\_UESR\_idx` (`OrderUserID`),

CONSTRAINT `FK\_ORDER\_UESR` FOREIGN KEY (`OrderUserID`) REFERENCES `user` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----------------------------------------------------------------------------

CREATE TABLE `deliveryaddress` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`addressLine1` varchar(45) NOT NULL DEFAULT '',

`addressLin2` varchar(45) DEFAULT NULL,

`addressLine3` varchar(45) DEFAULT NULL,

`city` varchar(45) NOT NULL DEFAULT '',

`state` varchar(45) NOT NULL DEFAULT '',

`country` varchar(45) NOT NULL DEFAULT '',

`pinCode` decimal(10,0) NOT NULL DEFAULT '0',

`userId` int(11) NOT NULL DEFAULT '0',

PRIMARY KEY (`id`),

KEY `FK\_DeliveryAddress\_userid` (`userId`),

CONSTRAINT `FK\_DeliveryAddress\_userid` FOREIGN KEY (`userId`) REFERENCES `user` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

------------------------------------------------------------------------

CREATE TABLE `cart` (

`CartItemID` int(11) NOT NULL AUTO\_INCREMENT,

`DateCreated` tinyint(1) NOT NULL,

`Deleted` tinyint(1) NOT NULL,

`Qty` int(11) NOT NULL,

`ProductID` int(11) NOT NULL,

`SellerID` int(11) NOT NULL,

`UserID` int(11) NOT NULL,

PRIMARY KEY (`CartItemID`),

KEY `FK\_PRODUCT\_idx` (`ProductID`),

KEY `FK\_SELLER\_idx` (`SellerID`),

KEY `FK\_USER\_idx` (`UserID`),

CONSTRAINT `FK\_PRODUCT\_CART` FOREIGN KEY (`ProductID`) REFERENCES `product` (`ProductID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_SELLER\_CART` FOREIGN KEY (`SellerID`) REFERENCES `seller` (`SellerID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_USER\_CART` FOREIGN KEY (`UserID`) REFERENCES `user` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

------------------------------------------------------------------------

CREATE TABLE `orderdetails` (

`DetailIOrderID` int(11) NOT NULL AUTO\_INCREMENT,

`DetailPrice` decimal(6,2) NOT NULL,

`DetailQuantity` int(11) NOT NULL,

`Total` decimal(6,2) NOT NULL,

`ProductID` int(11) NOT NULL,

`SellerID` int(11) NOT NULL,

`OrderID` int(11) NOT NULL,

`ReviewID` int(11) NOT NULL,

PRIMARY KEY (`DetailIOrderID`),

KEY `FK\_PRODUCT\_idx` (`ProductID`),

KEY `FK\_SELLER\_idx` (`SellerID`),

KEY `FK\_ORDER\_idx` (`OrderID`),

KEY `FK\_REVIEW\_idx` (`ReviewID`),

CONSTRAINT `FK\_ORDER\_ORDER` FOREIGN KEY (`OrderID`) REFERENCES `order` (`OrderID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_PRODUCT\_ORDER` FOREIGN KEY (`ProductID`) REFERENCES `product` (`ProductID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_REVIEW\_ORDER` FOREIGN KEY (`ReviewID`) REFERENCES `review` (`ReviewID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_SELLER\_ORDER` FOREIGN KEY (`SellerID`) REFERENCES `seller` (`SellerID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

----------------------------------------------------------------------

CREATE TABLE `paymentdetails` (

`PaymentID` int(11) NOT NULL AUTO\_INCREMENT,

`PaymentMode` varchar(45) NOT NULL,

`CreditCardTypeID` varchar(45) NOT NULL,

`CreditCardExpMonth` int(11) NOT NULL,

`CreditCardExpYear` int(11) NOT NULL,

`CreditCardNumber` varchar(45) NOT NULL,

`OrderID` int(11) NOT NULL,

PRIMARY KEY (`PaymentID`),

KEY `FK\_ORDER\_idx` (`OrderID`),

CONSTRAINT `FK\_ORDER` FOREIGN KEY (`OrderID`) REFERENCES `order` (`OrderID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

------------------------------------------------------------------------------

CREATE TABLE `product` (

`ProductID` int(11) NOT NULL AUTO\_INCREMENT,

`ProductName` varchar(45) NOT NULL,

`ProductUnitPrice` decimal(4,2) NOT NULL,

`ProductUnitWeight` varchar(20) NOT NULL,

`ProductDescription` varchar(500) NOT NULL DEFAULT '',

`ProductImage` varchar(50) DEFAULT NULL,

`ProductThumbImage` varchar(50) DEFAULT NULL,

`ProductUpdateDate` datetime DEFAULT NULL,

`Active` tinyint(1) DEFAULT NULL,

`CategoryID` int(11) NOT NULL,

`Currency` varchar(4) NOT NULL DEFAULT '',

PRIMARY KEY (`ProductID`),

KEY `FK\_CATEGORY\_idx` (`CategoryID`),

CONSTRAINT `FK\_CATEGORY` FOREIGN KEY (`CategoryID`) REFERENCES `category` (`CategoryID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8;

-------------------------------------------------------------------

CREATE TABLE `review` (

`ReviewID` int(11) NOT NULL AUTO\_INCREMENT,

`ReviewDescription` varchar(45) NOT NULL,

`Rating` int(11) NOT NULL,

`TimeStamp` datetime NOT NULL,

PRIMARY KEY (`ReviewID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-------------------------------------------------------------------------

CREATE TABLE `role` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`RoleName` varchar(45) NOT NULL,

`RoleDesc` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

-------------------------------------------------------------------------

CREATE TABLE `seller` (

`SellerID` int(11) NOT NULL AUTO\_INCREMENT,

`About` varchar(45) NOT NULL,

`LogoImage` varchar(45) DEFAULT NULL,

`RegistrationNumber` varchar(45) NOT NULL,

`CompanyName` varchar(45) NOT NULL,

`URL` varchar(45) DEFAULT NULL,

`PaymentMethods` varchar(45) NOT NULL,

`ContactTitle` varchar(45) NOT NULL,

`UserID` int(11) DEFAULT NULL,

PRIMARY KEY (`SellerID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-------------------------------------------------------------------------------

CREATE TABLE `sellersproduct` (

`SellersProductID` int(11) NOT NULL AUTO\_INCREMENT,

`SellersStock` bigint(20) NOT NULL,

`StockStatus` varchar(45) NOT NULL,

`SellersDiscount` decimal(2,2) DEFAULT NULL,

`ProductID` int(11) NOT NULL,

`SellerID` int(11) NOT NULL,

PRIMARY KEY (`SellersProductID`),

KEY `FK\_PRODUCT\_idx` (`ProductID`),

KEY `FK\_SELLER\_idx` (`SellerID`),

CONSTRAINT `FK\_PRODUCT\_SP` FOREIGN KEY (`ProductID`) REFERENCES `product` (`ProductID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_SELLER\_SP` FOREIGN KEY (`SellerID`) REFERENCES `seller` (`SellerID`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

----------------------------------------------------------------

CREATE TABLE `user` (

`UserID` int(11) NOT NULL AUTO\_INCREMENT,

`FirstName` varchar(45) NOT NULL DEFAULT '',

`LastName` varchar(45) NOT NULL,

`Address1` varchar(45) DEFAULT NULL,

`Address2` varchar(45) DEFAULT NULL,

`City` varchar(45) DEFAULT NULL,

`State` varchar(45) DEFAULT NULL,

`Country` varchar(45) DEFAULT NULL,

`PinCode` varchar(45) DEFAULT NULL,

`Phone` varchar(45) DEFAULT NULL,

`Mobile` varchar(45) DEFAULT NULL,

`Email` varchar(45) NOT NULL DEFAULT '',

`Password` varchar(45) NOT NULL DEFAULT '',

`UserIP` varchar(45) DEFAULT NULL,

`DateCreated` datetime NOT NULL,

`Role` int(11) NOT NULL,

PRIMARY KEY (`UserID`),

KEY `FK\_ROLE\_idx` (`Role`),

CONSTRAINT `FK\_ROLE` FOREIGN KEY (`Role`) REFERENCES `role` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=504 DEFAULT CHARSET=utf8;

---------------------------------------------------------------------------

Data insertion in tables :

INSERT INTO `farmtohome`.`cart`

(`CartItemID`,`DateCreated`,`Deleted`,`Qty`,`ProductID`,`SellerID`,`UserID`)

VALUES(<{CartItemID: }>,<{DateCreated: }>,<{Deleted: }>,<{Qty: }>,<{ProductID: }>,<{SellerID: }>,<{UserID: }>);

----------------------------------------------------------------------------

INSERT INTO `farmtohome`.`category`

(`CategoryID`,`CategoryName`,`Description`,`CategoryImage`,`Active`)

VALUES(<{CategoryID: }>,<{CategoryName: }>,<{Description: }>,<{CategoryImage: }>,<{Active: 1}>);

----------------------------------------------------------------------------

INSERT INTO `farmtohome`.`deliveryaddress`

(`id`,`addressLine1`,`addressLin2`,`addressLine3`,`city`,`state`,`country`,`pinCode`,`userId`)

VALUES

(<{id: }>,<{addressLine1: }>,<{addressLin2: }>,<{addressLine3: }>,<{city: }>,<{state: }>,<{country: }>,<{pinCode: 0}>,<{userId: 0}>);

-----------------------------------------------------------------------------

INSERT INTO `farmtohome`.`order`

(`OrderID`,`OrderDate`,`OrderAmount`,`Freight`,`Tax`,`OrderShipName`,`OrderShipAddress1`,

`OrderShipAddress2`,`OrderCity`,`OrderState`,`OrderPinCode`,`OrderCountry`,`OrderPhone`,

`OrderMobile`,`OrderEmail`,`Deleted`,`TransactionStatus`,`Paid`,`ErrMsg`,`ErrLoc`,`Refund`,

`PaymentID`,`OrderUserID`)

VALUES

(<{OrderID: }>,<{OrderDate: }>,<{OrderAmount: }>,<{Freight: }>,<{Tax: }>,<{OrderShipName: }>,

<{OrderShipAddress1: }>,<{OrderShipAddress2: }>,<{OrderCity: }>,<{OrderState: }>,<{OrderPinCode: }>,

<{OrderCountry: }>,<{OrderPhone: }>,<{OrderMobile: }>,<{OrderEmail: }>,<{Deleted: }>,

<{TransactionStatus: }>,<{Paid: }>,<{ErrMsg: }>,<{ErrLoc: }>,<{Refund: }>,<{PaymentID: }>,

<{OrderUserID: }>);

----------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`orderdetails`

(`DetailIOrderID`,`DetailPrice`,`DetailQuantity`,`Total`,`ProductID`,`SellerID`,`OrderID`,`ReviewID`)

VALUES

(<{DetailIOrderID: }>,<{DetailPrice: }>,<{DetailQuantity: }>,<{Total: }>,<{ProductID: }>,<{SellerID: }>,

<{OrderID: }>,<{ReviewID: }>);

-----------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`paymentdetails`

(`PaymentID`,`PaymentMode`,`CreditCardExpMonth`,`CreditCardExpYear`,`CreditCardNumber`,`OrderID`,

`NameOnCard`)

VALUES(<{PaymentID: }>,<{PaymentMode: }>,<{CreditCardExpMonth: 0}>,<{CreditCardExpYear: }>,

<{CreditCardNumber: }>,<{OrderID: }>,<{NameOnCard: }>);

-----------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`product`

(`ProductID`,`ProductName`,`ProductUnitPrice`,`ProductUnitWeight`,`ProductDescription`,`ProductImage`,

`ProductThumbImage`,`ProductUpdateDate`,`Active`,`CategoryID`,`Currency`)

VALUES

(<{ProductID: }>,<{ProductName: }>,<{ProductUnitPrice: }>,<{ProductUnitWeight: }>,<{ProductDescription: }>,<{ProductImage: }>,<{ProductThumbImage: }>,<{ProductUpdateDate: }>,<{Active: }>,

<{CategoryID: }>,<{Currency: }>);

-----------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`review`

(`ReviewID`,`ReviewDescription`,`Rating`,`TimeStamp`)

VALUES

(<{ReviewID: }>,<{ReviewDescription: }>,<{Rating: }>,<{TimeStamp: }>);

------------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`role`

(`id`,`RoleName`,`RoleDesc`)

VALUES

(<{id: }>,<{RoleName: }>,<{RoleDesc: }>);

--------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`seller`

(`SellerID`,`About`,`LogoImage`,`RegistrationNumber`,`CompanyName`,`URL`,`PaymentMethods`,

`ContactTitle`,`UserID`)

VALUES

(<{SellerID: }>,<{About: }>,<{LogoImage: }>,<{RegistrationNumber: }>,<{CompanyName: }>,<{URL: }>,

<{PaymentMethods: }>,<{ContactTitle: }>,<{UserID: }>);

-----------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`sellersproduct`

(`SellersProductID`,`SellersStock`,`StockStatus`,`SellersDiscount`,`ProductID`,`SellerID`)

VALUES(<{SellersProductID: }>,<{SellersStock: }>,<{StockStatus: }>,<{SellersDiscount: }>,<{ProductID: }>,<{SellerID: }>);

------------------------------------------------------------------------------------

INSERT INTO `farmtohome`.`user`

(`UserID`,`FirstName`,`LastName`,`Address1`,`Address2`,`City`,`State`,`Country`,`PinCode`,`Phone`,`Mobile`,

`Email`,`Password`,`UserIP`,`DateCreated`,`Role`)

VALUES

(<{UserID: }>,<{FirstName: }>,<{LastName: }>,<{Address1: }>,<{Address2: }>,<{City: }>,<{State: }>,

<{Country: }>,<{PinCode: }>,<{Phone: }>,<{Mobile: }>,<{Email: }>,<{Password: }>,<{UserIP: }>,

<{DateCreated: }>,<{Role: }>);

------------------------------------------------------------------------------------

**Comments and Description of Coding segments**

Our **Farm To Home Organic Online Shop** is a java based web application. The java comments are statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code for specific time.

## Types of Java Comments:

There are 3 types of comments in java.

1. Single Line Comment

// This is a single line comments

1. Multi Line Comment

/\*

This is

multi line

comment

\*/

1. Documentation Comment

/\*\*

This

is

documentation

comment

\*/

**Standardization of the coding**

It tells developers how they must write their code. Instead of each developer coding in their own preferred style, they will write all code to the standards outlined in the document. This makes sure that a large project is coded in a consistent style parts are not written differently by different programmers. Not only does this solution make the code easier to understand, it also ensures that any developer who looks at the code will know what to expect throughout the entire application.

Why we need coding conventions?

* 80% of the lifetime cost of a piece of software goes to maintenance.
* Hardly any software is maintained for its whole life by the original author.
* Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
* If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.

For **Farm To Home Organic Online Shop** , we have chosen Oracle’s coding conventions.

**Code Efficiency**

Code efficiency is a broad term used to depict the reliability, speed and programming methodology used in developing codes for an application. Code efficiency is directly linked with algorithmic efficiency and the speed of runtime execution for software. It is the key element in ensuring high performance. The goal of code efficiency is to reduce resource consumption and completion time as much as possible with minimum risk to the business or operating environment. The software product quality can be accessed and evaluated with the help of the efficiency of the code used.

Recommendations for code efficiency include:

* To remove unnecessary code or code that goes to redundant processing
* To make use of optimal memory and nonvolatile storage
* To ensure the best speed or run time for completing the algorithm
* To make use of reusable components wherever possible
* To make use of error and exception handling at all layers of software, such as the user interface, logic and data flow
* To create programming code that ensures data integrity and consistency
* To develop programming code that's compliant with the design logic and flow
* To make use of coding practices applicable to the related software
* To optimize the use of data access and data management practices
* To use the best keywords, data types and variables, and other available programming concepts to implement the related algorithm

**Error handling**

In **Farm To Home Organic Online Shop** has implemented Java’s exception handling mechanism to handle runtime errors such as ClassNotFound, IO, SQL, Remote etc. The core advantage of exception handling is to maintain the normal flow of the application. Exception normally disrupts the normal flow of the application that is why we use exception handling.

There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun microsystem says there are three types of exceptions:

1. Checked Exception
2. Unchecked Exception
3. Error

Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException, SQLException etc. Checked exceptions are checked at compile-time.

Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.

### Error

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

Try..Catch..Block

Java try block is used to enclose the code that might throw an exception. It must be used within the method. Java try block must be followed by either catch or finally block.

try {

//code that may throw exception

} catch (Exception\_class\_Name ref){

// code to handle exception

} finally {

// finally statement

}

**Validation checks**

In In **Farm To Home Organic Online Shop,** we have done both front end and back end validations by using JQuery validation plugin and Spring MVC validator respectively.

Consider a form submitting from front end via Ajax or HTML form, JQuery validation plugin validate the entire form and if success it allows to submit to back end. In java side Spring MVC validator will handle on submitted form.

# jQuery Validation Plugin

This jQuery plugin makes simple clientside form validation easy, whilst still offering plenty of customization options. It makes a good choice if you’re building something new from scratch, but also when you’re trying to integrate something into an existing application with lots of existing markup. The plugin comes bundled with a useful set of validation methods, including URL and email validation, while providing an API to write your own methods. All bundled methods come with default error messages in English and translations into 37 other languages.

Spring MVC custom validator

Spring MVC supports validation by means of a validator object that implements the Validator interface. You can write the following validator to check if the required form fields are filled. Also it provides lot of utility methods to validate the required form fields.

TESTING

**Testing Objective**

Testing is the penultimate step of software development. An elaborate testing of data is prepared and the system is using test data. While doing testing, errors are noted and correction is made. The users are trained to operate the developed system. Both hardware and software securities are made to run the developed system successfully.

System testing is aimed at ensuring the system works accurately before live operation commences. Testing is vital to the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subjected to a verity of tests: Online Response, Volume, Stress Recovery & Security and Usable tests. A series of testing are performed for the proposed system before the system is ready for user acceptance testing. Nothing is complete without testing, as it is vital success of the system.

**System Testing**

When a system is developed, it is hoped that it performs properly. In practice, however, some errors always occur. The main purpose of testing an information system is to find the errors and correct them. A successful test is one, which finds an error.

The main objectives of system testing are:

* To ensure during operation the system will perform as per specification.
* To make sure that the system meets user’s requirements during operation.
* To verify that the controls incorporated in the system function as intended.
* To see that when correct inputs are fed to the system the outputs are correct.
* To make sure that during operation, incorrect input and output will be deleted.

The scope of a system test should include both manual operations and computerized operations. System testing is a comprehensive evaluation of the programs, manual procedures, computer operations and controls. System testing is the process of checking if the developed system is working according to the original objectives and requirements. All testing needs to be conducted in accordance to the test conditions specified earlier.

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of software designs the module. To check whether each module in the software works properly so that it gives desired outputs to the given inputs. All Validations and conditions are tested in the module level in the unit test. Control paths are tested to ensure the information properly flows into, and out of the program unit and out of the program unit under test. Boundary condition is tested to ensure that the modules operate at boundaries.

The module interface id tested to ensure that information properly flows into and out of the program unit under test. The local data structures are examined to ensure that the data stored temporarily maintains it integrity during all steps in an algorithm execution. The module interface id tested to ensure that information properly flows into and out of the program unit under test. The local data structures are examined to ensure that the data stored temporarily maintains it integrity during all steps in an algorithm execution

**Integration testing**

The major concerns of integration testing are developing an incremental strategy that will limit the complexity of entire actions among components as they are added to the system. Developing a component as they are added to the system, developing an implementation & integration schedules that will make the modules available when needed, and designing test cases that will demonstrate the viability of the evolving system. Though each program works individually, they should work after linking them together. This is also referred to as Interfacing.

Data may be lost across interface and one module can have an adverse effect on another. Subroutines, after linking, may not do the desired function expected by the main routine. Integration testing is a systematic technique for constructing program structure while at same time, conducting test to uncover errors associated with the interface. In the testing, the programs are constructed and tested in small segments.

**Validation Testing**

Validation succeeds when the software function in a manner that is reasonably expected by the customer. The proposed system has been tested by using validation testing and found to be working satisfactorily. If a user wants to use the proposed system, the user has to specify username and password. In validation testing, it was found that the user is allowed to login only if the correct password is given.

System Security measures

Security has been implemented in our **Farm To Home Organic Online Shop** using Spring security. Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements.

Features,

* Comprehensive and extensible support for both Authentication and Authorization
* Protection against attacks like session fixation, clickjacking, cross site request forgery, etc
* Servlet API integration
* Optional integration with Spring Web MVC
* Much more…

Future scope

As and when this software is asked for by an organization, it can be customized based on their preferences. For this, another round of SDLC process has to be initiated to understand user requirements and feasibility of changes. However, as the system is built on a basic framework and a set of functional.

* Multiple language support
* Handicapped support
* Precise order status update
* Sub admin module can be added
* Based on the future security issues, security can be improved using emerging technologies.
* As the technology emerges, it is possible to upgrade the system and be adaptable to desired environment.

Bibliography

* SYSTEM ANALYSIS AND DESIGN, Award E.M.
* SOFTWARE ENGINEERING, K.K. Aggarwal and Yogesh Singh.
* OCP/OCA , Cathy Sierra

Appendices

* JSP – Java Server Pages
* JAVAEE – Java Enterprise Edition
* HTML – Hyper Text Markup Language
* CSS – Cascading Style Sheet
* MVC – Model View Controller
* HTTP – Hyper Text Transfer Protocol

CODE SNIPPETS

public class WebDotXml extends AbstractAnnotationConfigDispatcherServletInitializer {

@Override

protected Class<?>[] getRootConfigClasses() {

return new Class[] { SpringWebConfig.class };

}

@Override

protected Class<?>[] getServletConfigClasses() {

return null;

}

@Override

protected String[] getServletMappings() {

return new String[] { "/" };

}

}

--------------------------------------------------------------------------------------------------------------------

@Configuration

@EnableWebSecurity

public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

@Autowired

BasicDataSource dataSource;

@Autowired

public void configAuthentication(AuthenticationManagerBuilder auth) throws Exception {

auth.jdbcAuthentication().dataSource(dataSource)

.usersByUsernameQuery(

"select Email as username,Password as password, true as enabled from user where Email=?")

.authoritiesByUsernameQuery(

"select Email as username, RoleName as role from user,role where user.Role = role.id and user.Email=?");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests()

.antMatchers("/", "/home","/Review","/List","/Contact","/Join","/register").permitAll()

.antMatchers("/Admin").access("hasRole('ROLE\_ADMIN')")

.antMatchers("/Checkout","/Payment").access("hasRole('ROLE\_USER')")

.and()

.csrf()

.and()

.exceptionHandling().accessDeniedPage("/AccessDenied");

http.formLogin()

.loginPage("/Login")

.failureUrl("/Login?error")

.usernameParameter("ssoId")

.passwordParameter("password");

http.logout()

.logoutUrl("/Logout")

.invalidateHttpSession(true)

.logoutSuccessUrl("/Login?logout")

.deleteCookies("JSESSIONID");

http.sessionManagement()

.sessionFixation().migrateSession()

.sessionCreationPolicy(SessionCreationPolicy.IF\_REQUIRED)

//.invalidSessionUrl("/Login?invalidSession")

.maximumSessions(1)

.expiredUrl("/Login?sessionExpired");

}

@Bean

public HttpSessionEventPublisher httpSessionEventPublisher() {

return new HttpSessionEventPublisher();

}

}

-------------------------------------------------------------------------------------------------------------------

@EnableWebMvc

@Configuration

@ComponentScan("com.farmtohome.\*")

@PropertySource("classpath:application.properties")

public class SpringWebConfig extends WebMvcConfigurerAdapter {

@Autowired

private Environment env;

@Override

public void addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/resources/\*\*")

.addResourceLocations("/resources/");

}

@Bean

public InternalResourceViewResolver viewResolver() {

InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();

viewResolver.setViewClass(JstlView.class);

viewResolver.setPrefix("/WEB-INF/views/jsp/");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

@Bean(destroyMethod = "close")

public BasicDataSource dataSource(){

BasicDataSource dataSource = new BasicDataSource();

dataSource.setDriverClassName(env.getProperty("jdbc.driverClassName"));

dataSource.setUrl(env.getProperty("jdbc.url"));

dataSource.setUsername(env.getProperty("jdbc.username"));

dataSource.setPassword(env.getProperty("jdbc.password"));

dataSource.setInitialSize(3);

return dataSource;

}

@Bean

public JdbcTemplate jdbcTemplate(BasicDataSource dataSource) {

JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);

jdbcTemplate.setResultsMapCaseInsensitive(true);

return jdbcTemplate;

}

@Bean

public JavaMailSender eMailSender(){

JavaMailSenderImpl mailSender = new JavaMailSenderImpl();

mailSender.setHost("smtp.gmail.com");

mailSender.setPort(587);

mailSender.setUsername("Your-gmail-id"); // need to set

mailSender.setPassword("Your-gmail-password"); // need to set

Properties javaMailProperties = new Properties();

javaMailProperties.put("mail.smtp.starttls.enable", "true");

javaMailProperties.put("mail.smtp.auth", "true");

javaMailProperties.put("mail.transport.protocol", "smtp");

javaMailProperties.put("mail.debug", "true");//Prints out everything on screen

mailSender.setJavaMailProperties(javaMailProperties);

return mailSender;

}

}

------------------------------------------------------------------------------------------------------------------

@Controller

public class WelcomeController {

private final Logger logger = LoggerFactory.getLogger(WelcomeController.class);

@Autowired

private UserService userService;

@Autowired

ServletContext servletContext;

@Autowired

HttpServletRequest httpServletRequest;

@RequestMapping(value = "/")

public String welcomePage(){

logger.info("Rendering Welcome Page");

return "index";

}

@RequestMapping(value = "/Review")

public String productReviewPage(){

logger.info("Rendering Product Review Page");

return "single";

}

@RequestMapping(value = "/List")

public String products(){

logger.info("Rendering Product Products Page");

return "products";

}

@RequestMapping(value = "/Checkout")

public String checkoutProducts(ModelMap model, Principal principal,HttpServletRequest request){

logger.info("Rendering Product checkout Products Page");

String userId = principal.getName();

User user = userService.getUser(userId);

request.getSession().setAttribute("userDetails", user);

ShoppingCart shoppingCart = (ShoppingCart) servletContext.getAttribute("shoppingCart");

request.getSession().setAttribute("sessionShoppingCart", shoppingCart);

model.addAttribute("username", userId);

return "checkout";

}

@RequestMapping(value = "/Contact")

public String contact(){

logger.info("Rendering Product checkout contact Page");

return "contact";

}

@RequestMapping(value = "/Login")

public String loginPage(ModelMap model){

logger.info("Rendering Product checkout contact Page");

if(null != httpServletRequest.getParameter("error")){

model.put("loginMsg", "Invalid username or password");

}

if(null != httpServletRequest.getParameter("logout")){

model.put("loginMsg", "Successfully logged out");

}

if(null != httpServletRequest.getParameter("sessionExpired")){

model.put("loginMsg", "Session has expired");

}

return "login";

}

@RequestMapping(value = "/Join")

public String registerPage(){

logger.info("Rendering Product checkout Register Page");

return "register";

}

@RequestMapping(value = "/Admin")

public String adminPage(){

logger.info("Rendering Product checkout Admin Page");

return "admin";

}

@RequestMapping(value = "/Payment")

public String paymentPage(){

logger.info("Rendering Product checkout Payment Page");

return "payment";

}

@RequestMapping(value = "/AccessDenied")

public String test(){

logger.info("Rendering Product Review Page");

return "accessDenied";

}

@RequestMapping(value = "/jack")

@ResponseBody

public TestVo testV(){

TestVo t = new TestVo("Rinson", "George", "Kulangara");

return t;

}

}

-----------------------------------------------------------------------------------------------------------------------

@Controller

public class ServiceController {

/\*@Autowired

private JdbcTemplate jdbcTemplate;\*/

@Autowired

ProductService productService;

@Autowired

UserService userService;

@Autowired

ServletContext servletContext;

private final Gson gson = new Gson();

@RequestMapping(value = "/availability")

@ResponseBody

public String availabilityHomePage(){

List<Product> newProducts = productService.getNewProducts();

List<Product> topProducts = productService.getTopProducts();

List<Product> recommendedProducts = productService.getRecommendedProducts();

List<Category> categories = productService.getCategory();

servletContext.setAttribute("categories", categories);

Availability availability = new Availability();

availability.setCategory(categories);

availability.setNewProducts(newProducts);

availability.setRecommendedProducts(recommendedProducts);

availability.setTopProducts(topProducts);

return gson.toJson(availability);

}

@RequestMapping(value = "/products")

@ResponseBody

public String getProducts(@RequestParam String categoryId){

if(!StringUtils.isEmpty(categoryId)){

ProductsVO productsVO = new ProductsVO();

List<Product> products = productService.getProducts(categoryId);

@SuppressWarnings("unchecked")

List<Category> categories = (List<Category>) servletContext.getAttribute("categories");

productsVO.setCategory(categories);

productsVO.setProducts(products);

return gson.toJson(productsVO);

}

return null;

}

@RequestMapping(value = "/productDetails")

@ResponseBody

public String getProductDetails(@RequestParam String productId){

if(!StringUtils.isEmpty(productId)){

Product product = productService.getProduct(productId);

List<Category> categories = (List<Category>) servletContext.getAttribute("categories");

List<Product> recommendedProducts = productService.getRecommendedProducts();

ProductDetailsVO productDetailsVO = new ProductDetailsVO();

productDetailsVO.setCategory(categories);

productDetailsVO.setSelectedProduct(product);

productDetailsVO.setSuggestion(recommendedProducts);

return gson.toJson(productDetailsVO);

}

return null;

}

@RequestMapping(value = "/addToCart", method = RequestMethod.POST)

@ResponseBody

public String addToCart(@RequestBody CartItem cartItem){

boolean isSellerAvailable = productService.checkSellerAvailability(cartItem.getPinCode());

if(isSellerAvailable){

ShoppingCart shoppingCart = productService.addToShoppingCart(cartItem, servletContext);

return new Gson().toJson(shoppingCart);

}else{

JsonObject o = new JsonObject();

o.addProperty("msg", "Seller is not available for this product. Please choose different PinCode");

return o.toString();

}

}

@RequestMapping(value = "/doRegister", method = RequestMethod.POST)

public ModelAndView register(@ModelAttribute RegistrationForm registrationForm){

String status = userService.addUser(registrationForm);

if(status.equalsIgnoreCase("added")){

return new ModelAndView("login", "status", status);

}else {

return new ModelAndView("register", "status", status);

}

}

@RequestMapping(value = "/doPayment", method = RequestMethod.POST)

public ModelAndView payment(@ModelAttribute PaymentForm paymentForm){

return new ModelAndView("confirmBooking");

}

}