**What is actually a design document!**

The design document gets developed by designer’s who design this in order to   
give description of the product which the developers in the software development team use to develop the product. In short the design document gives in a nutshell the main idea and structure of the product that would be developed by developers.   
  
The design document comes following the design of architecture diagram. This is the diagram which explains the architecture which would be used for developing the software product. The architecture diagram contains one or more of the following namely the technology that would be used to develop the software product, description of the product that would be developed, description of the interfaces that would be used in thedevelopment process for communication, the main components that would be developed for the product, the interfaces that would be developed between different components, the hardware requirement for developing the software product and so on. The above all would be analyzed and designed in architecture diagram. Following this architecture diagram design the design of design document phase starts.

**Kinds of design documents:**

The design document is mainly classified as two types namely

High Level Design Document also called as HLD   
Low Level Design Document also called as LLD

**High Level Design Document:**

High Level Design Document is the acronym for HLD. The high level design   
document must be designed by taking into account one or more of the following namely

* Careful analysis and study of the requirement document must be made to prepare the design document. This is because it is vital that all the items addressed in requirements document must be brought under design document.   
  .
* High level design gives the overview of the development of product. In other words how the program is going to be divided into functions, modules, subdivision etc.   
  .
* The design document prepared must be prepared taking into account things like how to develop the product in other words say for instance how the program is going to be divided into functions, modules, subdivision etc. so that they are easily maintainable in later stage.   
  .
* Design document must be designed also taking into account things like easy portability of the software product to different systems

**Low Level Design Document:**

Low Level Design Document is the acronym for LLD. The Low Level Design Document gives the design of the actual program code which is designed based on the High Level Design Document. A good Low Level Design Document developed will make the program very easy to be developed by developers because if proper analysis is made and the LowLevel Design Document is prepared then the code can be developed by developers directly from Low Level Design Document with minimal effort of debugging and testing.

In general the design document designed have the following namely

* The design document starts by giving a description about the purpose of the design document   
  .
* Has sections in design document which explain what all area would be covered in the design document in others words has description about scope of the design document   
  .
* The design document has sections which gives explanations about the   
  ............Document which was used as input for this document for instance   
  ............one could be say user requirements document and so on   
    
  ............Document which would be resulted as output from the design   
  ............document for instance say a test plan and so on   
  .
* Design document has an abstract section. This section gives a summary about the design document designed.   
  .
* Design document has a diagram showing the system architecture   
  .
* Design document also has details about functions of the software product.

Apart from the above contents present in design document varies depending upon the type of design document as explained above.

**Uses of Design Document**

A Good design document so analyzed and prepared helps in the following ways

* Good design document helps the development team to give quality product   
  .
* Design document analyses the timeline for each component development. So over all rough estimated time of completion can be calculated in the design documents stage itself. This can be used to compare with the target date set for completion of the project. If the time is greater than target date then unwanted and less important components can be left and may be taken off for next phase of development. Thus the design documents helps to divide the development stage into several phases and set target dates for each to achieve the total product delivered to customer on time. Thus on whole good design document helps to deliver quality software product within the timeframe.

# **Galla – the Small Shop ERP Product**

**Design Document**

Date: October 24, 2005

**Contact information**

Palwencha (palwencha@it.iitb.ac.in) (project manager, documentation)

Anirudha Joshi (anirudha@iitb.ac.in) (domain specialist, product and human-interaction design)

Gulavani Bhargav Subhash (bhargav@cse.iitb.ac.in) (software architect, system interface design and development)

Avinash Gupta (avinash08@iitb.ac.in) (quality assurance)

Nilesh Nalnikar (nileshnalnikar@iitb.ac.in) (tooling, system interface design and development)

**Index**

**1 Introduction**

**1.1 Background**

**1.2 Design Goals**

**2 Architecture**

**2.1 Introduction**

**2.2 Data**

**2.3 User Interface**

**2.3.1 Vendor Session Screens**

**2.3.2 Customer Session Screens**

**2.4 Biblography**

# **Introduction**

## ***Background***

Millions of small shops in urban India are threatened by the changing business environment. As large shopping malls, departmental stores and super markets have emerged to dot the cities, traditional small shops have had to fight back to retain market share.

The onslaught of organized retail through large shops is evident in metros and is spreading fast to smaller cities. Large shopping malls provide enhanced shopping experience to the young, upward mobile population that often purchases high-margin, high-value items. Large shops are marketed better, often as multi-city chains such as Crosswords, D-Mart, Big Bazaar, Foodworld, Lifestyle and Shoppers Stop. Their large scale enables them to operate at lower margins – many of the larger grocery stores give discounts over the printed price. Some have even started marketing themselves as the cheapest – “Is se sastaa aur accha kahin nahin.” (No where else will you find it cheaper and better than here.) They use automated systems for procurement, demand forecasting, printing price labels, billing and inventory tracking, ensuring high efficiency at lower costs.

However, large shops have some problems. Several of the large shops are very crowded on weekends. Because customers tend to buy for the whole week at a time, their average shopping is higher, leading to long queues at the checkout cashiers. To manage crowds, these shops have had to deploy better security arrangements, thus taking away some of the shopping experience. Since a large shop caters to a larger geographical area, customers have to either bring their vehicles (cars, scooters) leading to traffic congestion near the large shop, or need to rely on public transport (bus, rickshaw, taxi) to reach the shop and particularly to carry their purchase back home. Moreover, and perhaps because of these reasons, large shops have not yet managed to attract shoppers from lower-middle and lower classes in the cities.

This poses an opportunity to the small shops. Small shops have responded by improving quality of service, in particular by introducing free home delivery. Some small shops also provide credit to customers and retain long-term relationship with them. Customers of small shops rely on personal recommendations by the shop keeper. In case of many of the non-branded items such as cereals, shops serve as a brand.

However going forward, this may not prove to be sufficient. If small shops have to sustain competition with the organized retail chains, they will have to constantly improve efficiency and use the power of information technology to improve service. Small shops compete not only with large shops, but also with each other. In thse process, no doubt some of the small shops will be shaken out of the business. Those that remain competitive must ensure that they are equipped to do business in the environment of the future.

## ***Design Goals***

Powai Technologies Pvt. Ltd. is a start-up company that proposes to bring out a device (Galla) for implementing ERP systems in small shops in India and a service (galla.com) that will serve these small shops through this device.

Galla is an Enterprise Resource Planning (ERP) tool for small shops. It is a scalable piece of hardware. The shop may start with only one device, but may scale up to connect multiple devices together as the business grows.

galla.com, the service from Powai Technologies Pvt. Ltd. provides shop keepers with these kinds of services:

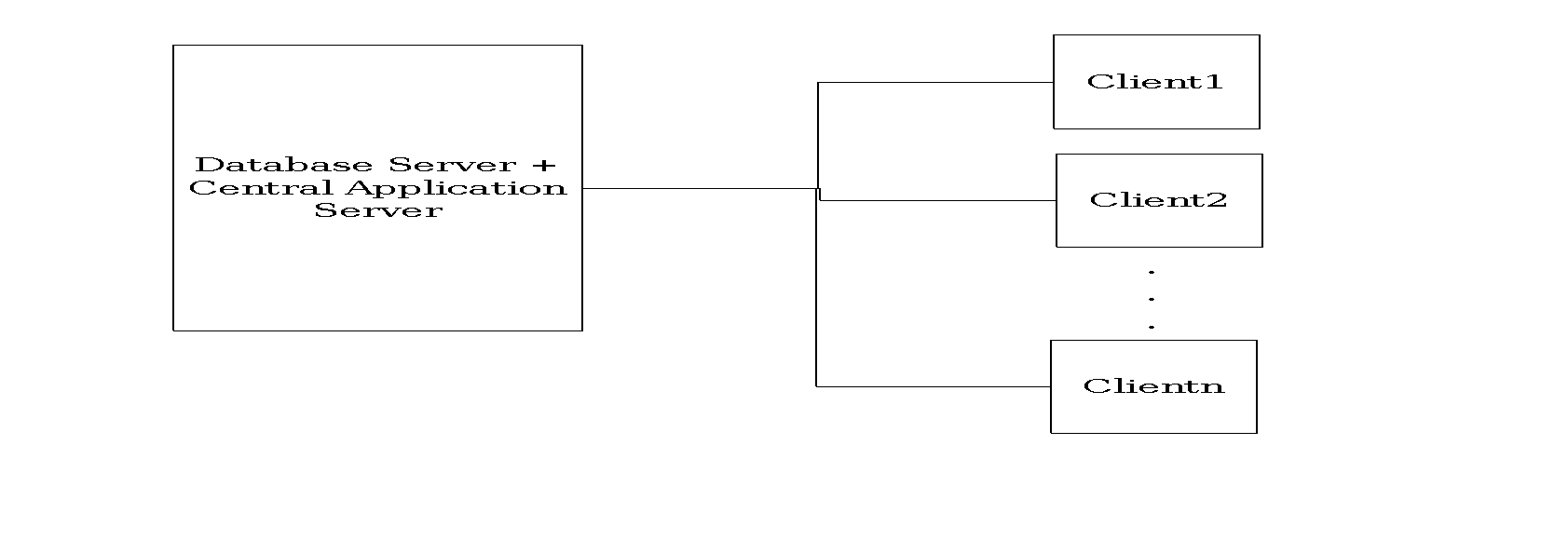
* Services to administer and maintain Galla
* Services of accounting, demand forecasting, cash-flow management etc.
* A web-based shopping interface for customers to locate best deals which hook up through the device as additional orders

galla.com may be only one of the services. Through Galla, the shopkeeper may interface with other such services from other organizations. Figure 1 shows a schematic of the proposed vision for Galla, galla.com and other services.

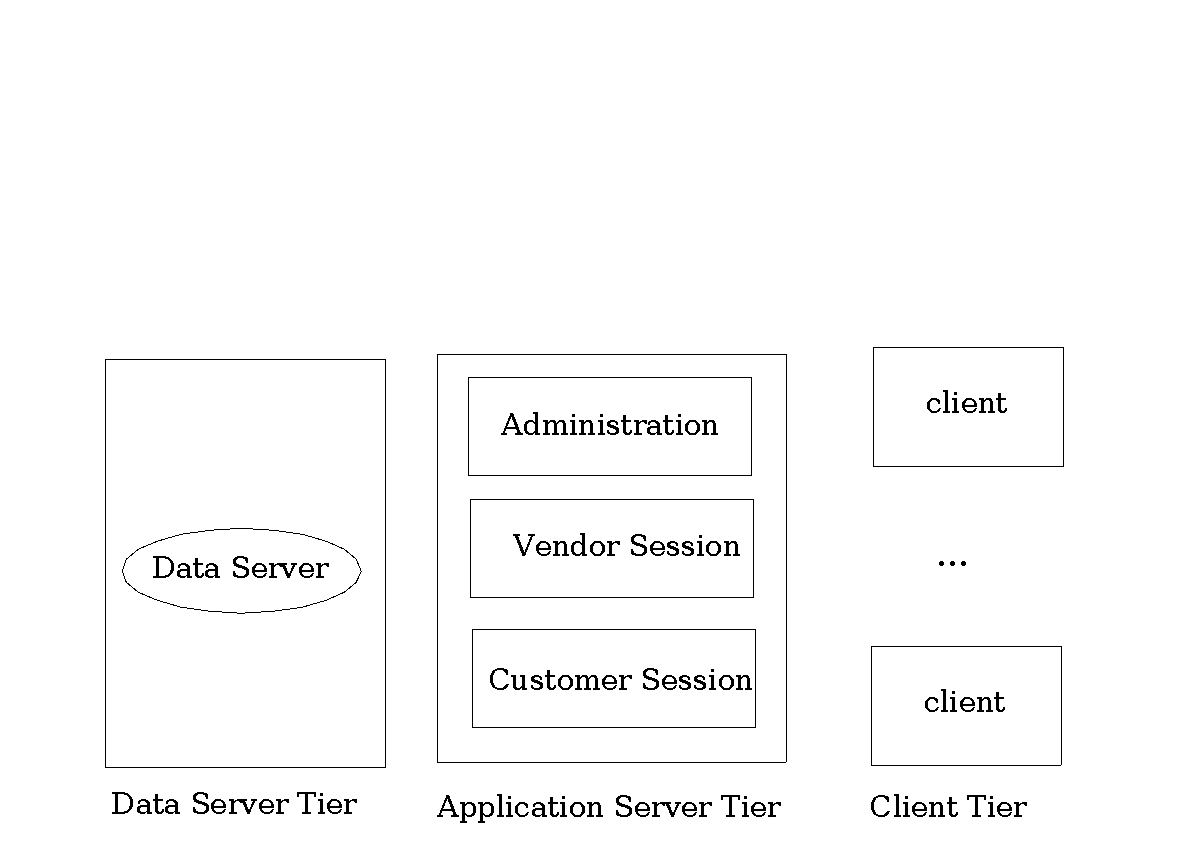
**Architecture**

**2.1 Introduction**

**Distributed, client-server:**

* + Clients can be either thin clients or other full-fledged clients having Windows/Linux/Mac installed.
  + 'n' is the maximum number of clients allowed
  + The central server and the database server can be deployed on the same machine. The central server is the application server. This enables us to use the thin clients.
  + The proximity of the application server and the database server will boost performance
  + The thin clients can be connected to the central server by a LAN, since the deployment is for the small shops
  + Each client may have a display, an input device like keyboard (probably customized/ scaled down version), bar-code reader, and printer.

1. **Top Level Architecture**

We will implement the 3-tier architecture. The tiers will be as shown in the diagram. The functionality of each of the tiers is as follows:

* 1. Client tier: Is responsible for the presentation of data, receiving user events and controlling the user interface. The actual business logic (e.g. calculating total cost and tax) has been moved to an application-server.
  2. Application Server tier: Business-objects that implement the business rules "live" here, and are available to the client-tier. This tier protects the data from direct access by the clients. Our object oriented analysis "OOA" will be aimed at the development of this layer.
  3. Data Server tier: This tier is responsible for data storage. It is important to note that boundaries between tiers are logical. As shown in the deployment diagram above, the Data tier and the Application Server tier will reside on the same machine.

This architecture enables us to achieve the required Quality Attributes as follows:

* 1. Performance and Cost: The decoupling of the clients from the application server enables us to use high performance machine for only the application server, the clients can be thin clients. Thus reducing the overall cost. Since the data server and application server reside on the same machine, it would be possible to tune the data server and the application programs to achieve the desired performance, network overload is avoided. Dynamic load balancing can be done, i.e., if bottlenecks in terms of performance occur, the server process can be moved to other servers at runtime.
  2. Flexibility: Since we have a different client-tier, clients can be thin clients, or full-fledged clients running any of the operating systems. It is easily possible to increase the number of terminals as required by the manager.
  3. Security: The security features can be implemented at the application server layer. Security is bolstered by the decoupling of the application server and the clients. As a rule servers are "trusted" systems. Their authorization is simpler than that of thousands of "untrusted" client-PCs. Data protection and security is simpler to obtain. Therefore it makes sense to run critical business processes, that work with security sensitive data, on the server.
  4. Modifiability: Re-definition of the storage strategy won’t influence the clients. RDBMS’ offer a certain independence from storage details for the clients. In the future, even radical changes, like let’s say switching form an RDBMS to an OODBS (may be for performance reasons), won’t influence the client. In well designed systems, the client still accesses data over a stable and well designed interface which encapsulates all the storage details. The application server can be built using the OOD paradigm and can provide for the incremental changes. Also addition and deletion of patches from the application server can be done on the fly without affecting any of the clients.

**Comparison with other Architecture:**

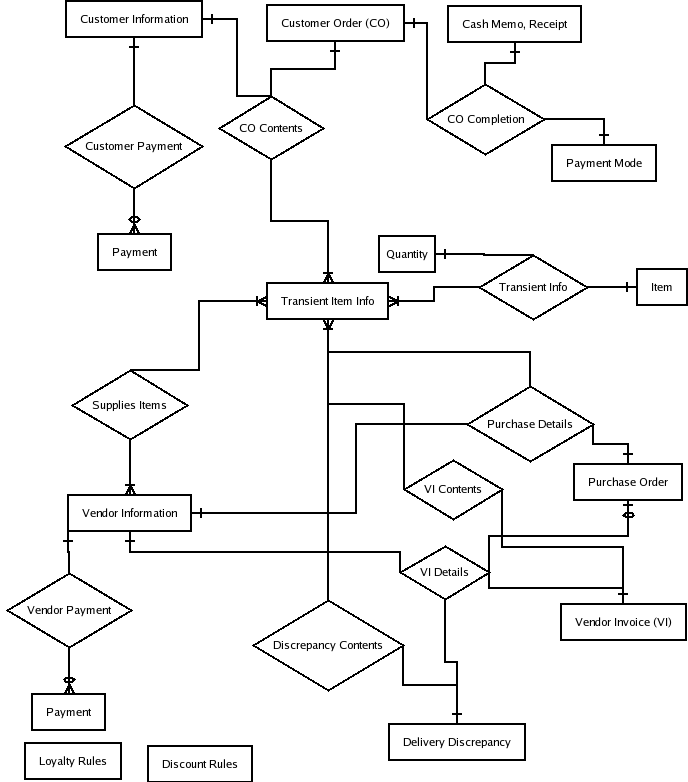
Our 3 tier architecture is similar to the layered architecture. Where the database layer is at the core and application server layer and the client layer work on top of it in that order.

3-tier architecture is also similar to the client serve architecture, or it is a specialization of client server architecture.

We have not preferred the pipe-filter architecture as the top level architecture since, in the broader scope, our requirements do not hint at only the data processing/manipulating aspects of ERP. There are many quality requirements which are crucial for the success of the project. We may have pipe-filter like architecture for some of the embedded modules in the application server layer.

## ***Data Model***

The following databases will be required for the system to carry out the requirements detailed in the software requirements specification. In the Entity Relation Diagram, CO refers to Customer Order, and VI refers to Vendor Invoice.



In the following we list the attributes of each of the tables, their primary keys (underlined) and the foreign keys (if any)

* **Item**
  + Item-Code
  + Description
  + IsReturnable
  + Type (loose/packed – if loose, it will have rate/unit, else it will have price)
* **Transient Item Info**
  + Item-Code (foreign key)
  + Category-Code
  + Cost (per unit item)
  + Expiry-date
  + Vendor-id
  + Commission
* **Transient Info**
  + Item-Code
  + Category-Code
  + Quantity (the remaining quantity)
  + Safe-Quantity (this is the derived attribute and the system might modify this attribute periodically. This is used to indicate whether the item is nearly out of stock)
* **Vendor Information**
  + Vendor-Id
  + Name
  + Address
  + Phone
* **Supplies Items**
  + Vendor-Id (foreign key)
  + Item-Code (foreign key)
  + Category-Code (foreign key)
* **Vendor Payment**
  + Vendor-Id (foreign key)
  + Payment-Number
  + Date
  + Amount
  + Payment-Mode (this will be a finite domain attribute – {cash, credit-card, credit})
* **Purchase Order**
  + Order-Id
  + Vendor-Id (foreign key)
  + Date
* **Purchase Details**
  + Order-Id (foreign key)
  + Item-Code (foreign key)
  + Category-Code (foreign key)
  + Quantity (for each of the ordered item)
* **Vendor Invoice (VI)**
  + Invoice-Id
  + Vendor-Id (foreign key)
  + Discrepancy-Id (foreign key)
  + Order-Id (foreign key -- could be null)
  + Date
* **VI Contents**
  + Invoice-Id (foreign key)
  + Item-Code (foreign key)
  + Category-Code (foreign key)
  + Quantity
* **Delivery Discrepancy**
  + Discrepancy-Id
  + Invoice-Id (foreign key)
  + Item-Code (foreign key)
  + Category-Code (foreign key)
  + Quantity
* **Customer Information**
  + Customer-Id
  + Name
  + Address
  + Phone
* **Customer Order (CO)**
  + Order-Id
  + Customer-Id (foreign key)
  + Date
* **CO Contents**
  + Order-Id (foreign key)
  + Item-Code (foreign key)
  + Category-Code (foreign key)
* **Cash Memo, Receipt**
  + Receipt-Id
  + Cost
  + Tax
  + Payment-Mode (this will be a finite domain attribute – {cash, credit-card, credit})
* **Customer Payment**
  + Payment-Id
  + Customer-Id (foreign key)
  + Date

## ***User Interface***

### **Admin Screens**

* + Admin screen
* Start-up
  + Please Authenticate Yourself screen
* Stock taking
  + Stock Summary screen / print
    - Stock Item screen
* Setting up customer loyalty benefits
  + Loyalty Program Scheme Settings screen
  + Special Customer Settings screen
  + Rewards Points Rules screen
    - New Reward Points Rule
  + Reward Points Redemption Rules screen
    - New Reward Points Redemption rule screen
  + Discounts screen
    - New Discount screen
* Preparing the orders and demand forecasting + Cash flow forecasting + Setting up discounts and sales
  + Sales Trends screen / print
    - Select Options screen
  + Demand Forecast screen / print
    - Set Forecast Parameters screen
  + Cash Flow Forecast screen / print
    - (same Set Forecast Parameters screen as demand forecast)
  + Discounts Schemes Summary screen / print
    - New Discount Scheme screen / print
  + Pending Orders Summary screen / print
    - Order screen / print (has options for each item: Order quantity, frequency, Delivery challan quantity, Verified filed quantity, Damage / discrepancy quantity, Sold, Unsold, Returned, Bill quantity, Bill amount, Contested amount) tracks to DC no/date, bill no/date, allows multiple DCs, bills, POs to be linked to each other. Think about this, it is not easy to visualize as one view!!!!!!!!
* Customer credit tracking
  + Customer Group Credit Summary screen
    - Create New Group screen
    - Customer Group Trends screen
    - Customer Credit Summary screen
      * Customer Bill screen
      * Customer Trend screen
* Filing stocks
  + Filing Stock Item screen
    - Bar Code print
* Expenses
  + Enter Expense Voucher screen
    - Set Expense as Recurring screen
* Shut-down
  + Shut-down Confirmation screen

### **Vendor Session Screens**

* + Vendors Summary screen
    - Add / Edit Vendor screen
      * Add / Edit Vendor Item screen
    - Vendor Statement screen
      * Vendor Payment screen
    - (same Order screen / print in preparing orders use case)
* Create new vendor account
  + (same Add / Edit Vendor screen as above)
* Placing orders
  + - (same Order screen / print in preparing orders use case)
* Accepting delivery and billing
  + - (same Order screen / print in preparing orders use case)
* Vendor returns
  + - (same Order screen / print in preparing orders use case)
* Vendor settlement
  + - (same as Vendor Payment screen)

### **Customer Session Screens**

* Ordering
  + Order / Bill screen
    - Up-sale Items screen
    - Recurring Order screen
    - Credit Card Transaction screen
    - Credit Transaction screen
    - Add / Edit Customer screen
* Order Settlement
  + - (same Order / Bill screen as above)
* Creating New Account
  + - (same Add / Edit Customer screen as above)
* Account Settlement
  + Customer Statement screen
    - (same Order / Bill screen as above)
* Goods Returns
  + Search Order / Bill screen
    - (same Order / Bill screen as above)
  + Admin screen
  + Please Authenticate Yourself screen
  + Stock Summary screen / print
    - Stock Item screen
  + Loyalty Program Scheme Settings screen
  + Special Customer Settings screen
  + Rewards Points Rules screen
    - New Reward Points Rule
  + Reward Points Redemption Rules screen
    - New Reward Points Redemption rule screen
  + Discounts screen
    - New Discount screen
  + Sales Trends screen / print
    - Select Options screen
  + Demand Forecast screen / print
    - Set Forecast Parameters screen
  + Cash Flow Forecast screen / print
    - (same Set Forecast Parameters screen as demand forecast)
  + Discounts Schemes Summary screen / print
    - New Discount Scheme screen / print
  + Pending Orders Summary screen / print
    - Order screen / print (has options for each item: Order quantity, frequency, Delivery challan quantity, Verified filed quantity, Damage / discrepancy quantity, Sold, Unsold, Returned, Bill quantity, Bill amount, Contested amount) tracks to DC no/date, bill no/date, allows multiple DCs, bills, POs to be linked to each other. Think about this, it is not easy to visualize as one view!!!!!!!!
  + Customer Group Credit Summary screen
    - Create New Group screen
    - Customer Group Trends screen
    - Customer Credit Summary screen
      * Customer Bill screen
      * Customer Trend screen
  + Filing Stock Item screen
    - Bar Code print
  + Enter Expense Voucher screen
    - Set Expense as Recurring screen
  + Shut-down Confirmation screen
  + Vendors Summary screen
    - Add / Edit Vendor screen
      * Add / Edit Vendor Item screen
    - Vendor Statement screen
      * Vendor Payment screen
    - (same Order screen / print in preparing orders use case)
  + (same Add / Edit Vendor screen as above)
    - (same Order screen / print in preparing orders use case)
    - (same Order screen / print in preparing orders use case)
    - (same Order screen / print in preparing orders use case)
    - (same as Vendor Payment screen)
  + Order / Bill screen
    - Up-sale Items screen
    - Recurring Order screen
    - Credit Card Transaction screen
    - Credit Transaction screen
    - Add / Edit Customer screen
    - (same Order / Bill screen as above)
    - (same Add / Edit Customer screen as above)
  + Customer Statement screen
    - (same Order / Bill screen as above)
  + Search Order / Bill screen
    - (same Order / Bill screen as above)
  + Enter Expense Voucher screen
    - Set Expense as Recurring screen
  + Shut-down Confirmation screen



## ***Bibliography***

1. Software Engineering- A Practioner's Approach (6th edition) -

2. Roger S. Pressman, McGraw Hill International Edition, 2005.

3. http://www.it.iitb.ac.in/~it607/resources/DID.doc