**Chapter: 1.0**

**Organization Part**

* 1. **Organizational Overview**

Daraz.com.bd is a cutting edge e-commerce platform in Bangladesh. Daraz is funded by Rocket Internet which is one of the most successful Internet incubators globally responsible for starting 100+ market leading e-commerce companies. We employ the best people, processes and technology to provide all our customers with authentic products and first-class services.Daraz has its own style to bring theme and ideas to the lead generation. For your convenience we have several payment options including credit and debit cards.



Daraz is Bangladesh’s number one online shopping destination where you can shop the widest selection of electronics, fashion, home appliances, kid’s items and more in Bangladesh and have them shipped directly to your home or office at your convenience! Daraz offer nationwide delivery, free returns and have convenient payment options. Recently they think about an E-commerce website through which they can sell agricultural items. There is a huge scope because of Bangladesh is an agricultural country. On the other hand now a day we don’t get good quality food from the market. So Daraz think they will be successful to fulfill customer’s demand.

**Daraz provides:**

* Web Application Development
* Web Site Development
* Graphic Design
* Mobile Site Development
* Software Maintenance
* SEO Package
* Website Maintenance
* Facebook Marketing
* Email Marketing
  1. **The Vision**

Our vision is to seek competitive advantages for our clients through innovative use of modern technologies; and help the clients achieving long-term success and prosperity through integrated business application.

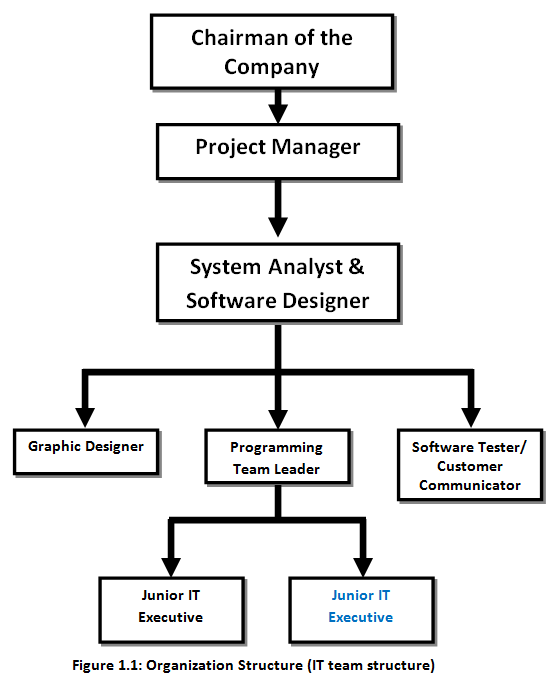
* 1. **The Mission**

To provide time tested, secure, reliable and cost effective enterprise level business solution to achieve business objectives and become a trusted growth partner for our customers, business associates and our people.

* 1. **My Position in this Company**

I joined in this company as a junior IT Executive.

* 1. **Organizational Structure**

****

**Chapter: 2.0**

**Project Introduction**

* 1. **Introduction**

Electronic Commerce (e-commerce) applications support the interaction between different parties participating in a commerce transaction via the network, as well as the management of the data involved in the process. E-commerce is fast gaining ground as an accepted and used business paradigm. 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. An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may 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. A good shopping cart design must be accompanied with user-friendly shopping cart application logic. It should be convenient for the customer to view the contents of their cart and to be able to remove or add items to their cart. The shopping cart application described in this project provides a number of features that are designed to make the customer more comfortable. This project helps in understanding the creation of an interactive web page and the technologies used to implement it. Online shopping allows you to browse through endless possibilities, and even offers merchandise that's unavailable in stores. If you're searching for a niche product that may not be distributed locally, you're sure to find what you're looking for on the internet. What's even more useful is the ability to compare items, similar or not, online. You can search through multiple stores at the same time, comparing material quality, sizes and pricing simultaneously.

* 1. **Background Of Study**

Due to the rapid technological growth and changes that has occurred in the recent years, electronic commerce is fast gaining ground as an accepted and used in business paradigm. More and more business houses are implementing online shopping system providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on online is becoming common. Since many people are now using this e-commerce technology there is need to come up with online shopping system that is easy to use (simple) and offer security to customer information.

* 1. **Objectives**

The objective of this project is to develop a general purpose e-commerce store where any agricultural product (such as fruit, vegetable, food items, small food processing tools) can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online shop. As an intern of Daraz I got the opportunity to develop an “Ecommerce Website for Daraz Bangladesh LTD”.

**2.3.1 Broad Objective**

To create an online shop management system that will provide secure services to customers, easy to use and provide management with features of store (products) management.

**2.3.2 Specific Objective**

1. To knowing when an item was saved or not saved in the shopping cart.

2. Returning to different parts of the site after adding an item to the shopping cart.

3. Easy scanning and selecting items in a list.

4. Effective categorical organization of products.

5. Simple navigation from home page to information and order links for specific products.

6. Obvious shopping links or buttons.

7. Minimal and effective security notifications or messages.

8. Consistent layout of product information.

9. User feedback system.

10. Quick online payment system.

* 1. **Proposed System**

Shop online is an online shopping application, which provides the online shopping facility available for everyone. Any types of the products will be available for the customer, and it can be easily purchased faster when compare with any other online shopping. Shop online application concentrates more on user friendly interfaces and promotes users to purchase. Shop online have registration facility. This will enable the new user to sign in and use the product much effectively. The registration process is faster and easier to any existing application.

In the user interface of shop online there will be accomplished with options to find new products available and most purchased and customer satisfied products. Customer can avail this facility and buy the product faster. All products in the website will be highlighted with the image of the product. By click on the image it will take you directly to the buy page, from where customer can purchase easily.

Shop online is provided with customer support page. Using this page any customer can get the assistance from the available customer support executive online. There we have support over phone, email and chat. Customer can use this facility any time.

As security issue pays major importance today, extra attention towards security is done in shop online application.

* 1. **Methodology**

The development process on “Development of ecommerce website System through Incremental Process Management Model” will complete following the structure described later on Software Analysis & Design.

This study on “Development of ecommerce website System through Incremental Process Management Model” is tentative in nature. It aims to development of ecommerce website System. The variables identified to manipulate through a handy inspection and from primary and secondary data.

* + 1. **Data Sources:**

For this project in data collection phase we collected two types of data i.e.

* Primary Data
* Secondary Data

**Primary data** are generated within the organization. The organizations practical experience, observation, and face-to-face interview with our own web administrators helped us generate the primary data.

**Secondary data** are generated by studying different articles, newspapers, research papers and of course information collected via Internet. Data, facts and statistics collected from different web sites and sources made us understand the project better.

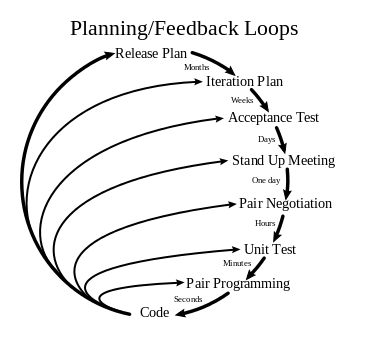
* 1. **Limitation of the Project**

One of the limitations of this project is constraints of time. The time that is assigned for this project is very short. After applying the software engineering procedures, it is very difficult to develop the complete website within this short time. For this reason, the scope of the project has become short. In this short time it is very difficult to understand the problem, collect information and construction the project. So far, I have learned seeing my senior classmates and friends, they had experienced the same during their time as well.

* 1. **Process model**

I have chosen “Extreme programming (XP)” model for developing ecommerce website System.

Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.



**Figure 2.1: Extreme programming (XP)**

Extreme Programming is successful because it stresses customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future this process delivers the software you need as you need it. Extreme Programming empowers your developers to confidently respond to changing customer requirements, even late in the life cycle.

Extreme Programming emphasizes teamwork. Managers, customers, and developers are all equal partners in a collaborative team. Extreme Programming implements a simple, yet effective environment enabling teams to become highly productive. The team self-organizes around the problem to solve it as efficiently as possible. Extreme Programming improves a software project in five essential ways; communication, simplicity, feedback, respect, and courage. Extreme Programmers constantly communicate with their customers and fellow programmers. They keep their design simple and clean. They get feedback by testing their software starting on day one.

They deliver the system to the customers as early as possible and implement changes as suggested. Every small success deepens their respect for the unique contributions of each and every team member. With this foundation Extreme Programmers are able to courageously respond to changing requirements and technology.

The most surprising aspect of Extreme Programming is its simple rules. Extreme Programming is a lot like a jig saw puzzle. There are many small pieces.

**2.7.1 Features and Advantages of Extreme programming model**

Extreme Programming is based on 12 principles:

* **The Planning Process --** The desired features of the software, which are communicated by the customer, are combined with cost estimates provided by the programmers to determine what the most important factors of the software are. This stage is sometimes called the Planning Game.
* **Small Releases --** The software is developed in small stages that are updated frequently, typically every two weeks.
* **Metaphor --** All members on an XP team use common names and descriptions to guide development and communicate on common terms.
* **Simple Design --** The software should include only the code that is necessary to achieve the desired results communicated by the customer at each stage in the process. The emphasis is not on building for future versions of the product.
* **Testing --**Testing is done consistently throughout the process. Programmers design the tests first and then write the software to fulfill the requirements of the test. The customer also provides acceptance tests at each stage to ensure the desired results are achieved.
* **Refactoring --** XP programmers improve the design of the software through every stage of development instead of waiting until the end of the development and going back to correct flaws.
* **Pair Programming --** All code is written by a pair of programmers working at the same machine.
* **Collective Ownership --** Every line of code belongs to every programmer working on the project, so there are no issues of proprietary authorship to slow the project down. Code is changed when it needs to be changed without delay.
* **Continuous Integration --** The XP team integrates and builds the software system multiple times per day to keep all the programmers at the same stage of the development process at once.
* **40-Hour Week --** The XP team does not work excessive overtime to ensure that the team remains well-rested, alert and effective.
* **On-Site Customer --** The XP project is directed by the customer who is available all the time to answer questions, set priorities and determine requirements of the project.
* **Coding Standard --** The programmers all write code in the same way. This allows them to work in pairs and to share ownership of the code.
  1. **Feasibility Study**

Feasibility study determines whether that solution is feasible or achievable for the organization. There are three major areas of feasibility study.

* Technical feasibility
* Economical feasibility
* Operational feasibility
  + 1. **Technical feasibility**

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system. The proposed system is compatible with a low qualification of computer with an internet connection only, as because it’s a web based software. As our client have well organized laptop and desktop. To maintain our system need a computer with a browser and internet connection which is already exists within the organization. In this way we can say that our software is technically feasible.

* + 1. **Economical feasibility**

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. Our system is economically feasible because by using the proposed system many works can be done within small time and which is not possible by man power within the same time. It also reduces the man power needed for providing the inventory report, sales report and stock report. So they have to pay less salary where the current system needs many employee and they are paying much salary. So we can say that, if they use proposed system they will be economically benefited.

* + 1. **Operational Feasibility**

User can easily operate the proposed system because the system is user friendly. Easy to insert inventory products, easy to create sales order. If the stuff of the organization has the basic to computer knowledge they could operate the software easily. Every features and the activity that are combined within the system is designed and developed belongs to previous format they had used with a more attractive user interface.

**Chapter: 3.0**

**Requirement Engineering**

* 1. **Methodology**

**1. Inception:**

Business plan:

Main goal of Daraz Bangladesh LTD is to make E-Commerce System which offers a complete set of inventory management, manufacturing and purchasing capabilities that improves supply chain management and delivers an end-to-end, procure-to-pay process.

**2. Elicitation:**

a. Time limit of the project:

Daraz Bangladesh LTD dedicated to develop the project within 30 October, 2015.

b. Scope:

Boundary: In this project my activities are limited to build user login, add user, cart system, payment getaway and add products etc. I am not going to develop the whole E-Commerce system.

c. Requirement:

We suggest our client about the project that the processing module will be

• Two types of user as Admin, Customer.

• Admin will insert products and maintain the whole system.

• User will buy products and create sales order.

• Reporting system.

**3. Elaboration:**

Basic requirements of the system are found:

i. Sales order printing facility.

ii. Admin will insert products easiest way.

iii. Stock Control facility.

iv. There should be an admin panel and admin will get the all privilege.

v. Large amount of data storage system.

vi. Strong data backup system.

vii. Most importantly a secured online system.

**4. Negotiation:**

Our project required time is 8 months and delivery time line is 1 November, 2015

**5. Specification:**

• Written document

• Set of graphical or mathematical model

• Collection of scenario

**6. Validation:**

• Black box testing

• White box texting

• QA Matrix

* 1. **Requirement analysis**

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed.

* + 1. **Hardware Requirement:**

The hardware listed by no means a minimum requirement to run the system, but rather a base limit for running the system smoothly and comfortably. This is also considering the potential amount of traffic that may go through the server.

- 1 x Intel Xeon E5504 2.0GHz Processor.

- 512 MB (DDR1) RAM.

- 80 GB 7200 rpm SATA Hard disk.

- 2 x 10/100/1000 Ethernet, 1 PCIe 2.0x16 slot.

**3.2.2 Software Requirement:**

- Web Server

Wampp Server

* Server Side Scripting

PHP 1.3

* Database Engine

MySQL 5.1.34

* Database Tools

MySQL Administrator

MySQL Query Browser

* Designing Tools

Adobe PhotoshopCS6

* Text Editor

PhpStorm-6.0.2

WordPressVersion4.1.1

* Word Processor

Microsoft Word 2010

* + 1. **Software Requirement for client:**

Browser

-Windows Internet Explorer 6.0 or above

-Mozilla Firefox version 17.0.1

-Google Chrome 20.0.1132.57

* + 1. **Functional Requirements:**

1. If the domain name is valid, then the user can navigate through system and can perform various functions provided as an output

2. If the domain name is invalid, then it gives more possible sites, respectively.

3. All information should be recorded in database.

4. Every user must be provided with unique id.

5. Date validation should be done according to the requirement.

6. Fields which is mandatory should not be left blank.

7. Compatibility of entered data with data types must be providing.

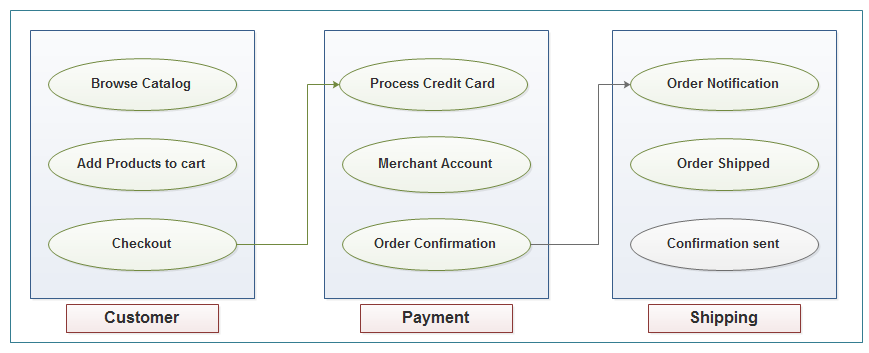
* 1. **Architectural Design:**

The design process for identifying the sub-systems making up a system and the framework for sub-system control and communication is architectural design.

The output of this design process is a description of the software architecture.

Now that I have been given a brief taste of the elements required for an E-Commerce site, let's take a quick look at how each of the elements work within the context of an E-Commerce transaction using the diagram below. A customer (1) browses catalog, (2) adds products to a cart, (3) decides to check out and make a credit card payment. The payment is (4) processed, (5) deposited into merchant account, and (6) confirmed. Finally the transaction is complete. An (7) order notification is sent to your shipping fulfillment department where the (8) order is shipped, and a (9) confirmation is sent to the user.

E-Commerce Site Process Model-

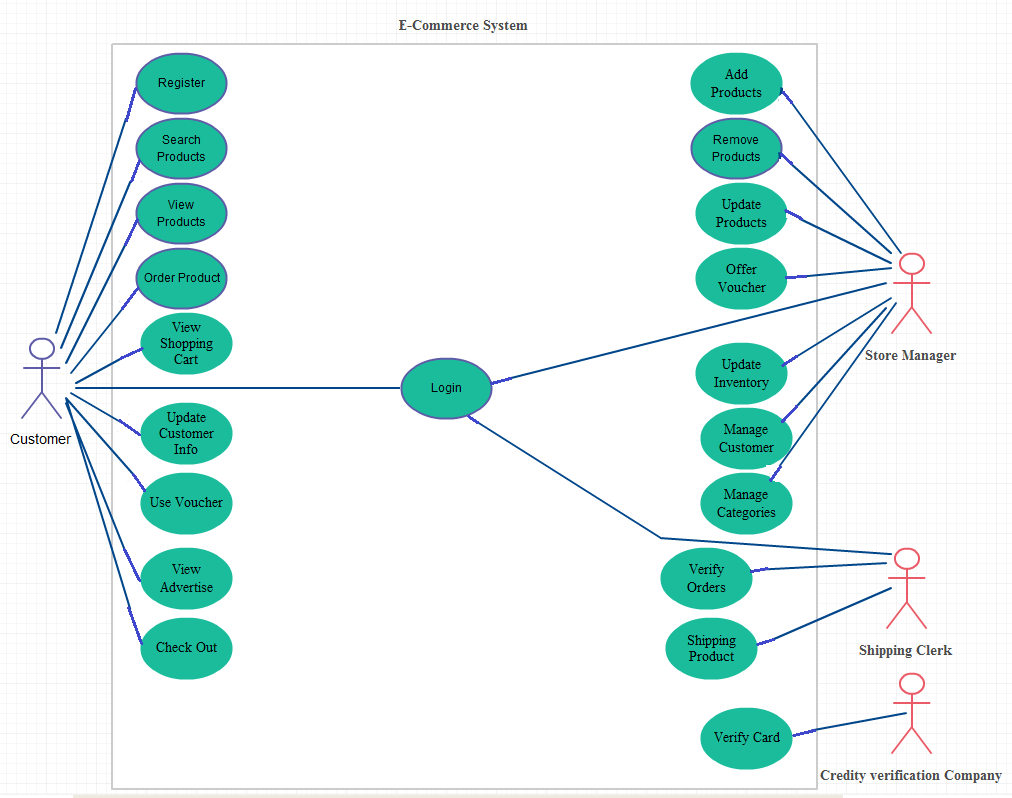
****

**Figure 3.1: E-Commerce Site Architectural Design**

* 1. **Interactive Diagram**

From the name Interaction it is clear that the diagram is used to describe some type of interactions among the different elements in the model. So this interaction or use case is a part of dynamic behavior of the system.

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. In software and systems engineering, a use case is a list of steps, typically defining interactions between a role and a system, to achieve a goal. The actor can be a human or an external system.



**Figure 3.2: Use Case Diagram**

* 1. **Scopes**
* [F1] Login
* [F2] Customer Registration
* [F3] Update User
* [F4] Delete User
* [F5] Add products
* [F6] Update products
* [F7] Delete products
* [F8] Add Sales Orders
* [F9] Checkout Orders
* [F10] Update Sales Orders
* [F11] Customer Feedback
* [F12] Online Payment
  1. **Function Point Estimation**

**FP count for function F1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 2 | 3 | 4 | 3 | 4 | 12 |
| Number of External Output | 2 | 3 | 4 | 3 | 5 | 15 |
| Number of External Inquiries | 2 | 2 | 2 | 2 | 4 | 8 |
| Number of Internal Logical Files | 1 | 1 | 1 | 1 | 10 | 10 |
| Number of External Interface | 1 | 1 | 1 | 1 | 7 | 7 |
|  |  |  |  |  | Count Total | 52 |

**Complexity adjustment value for function F1**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 5 |
| 2 | Are specialized data communications required? | 3 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 1 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 3 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 0 |
| 10 | Is the internal processing complex? | 0 |
| 11 | Is the code designed to be reusable? | 5 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 5 |
|  | Σ (Fi) | 24 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP =46.28

**FP count for function F2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 1 | 1 | 1 | 1 | 5 | 5 |
| Number of External Inquiries | 2 | 2 | 2 | 2 | 4 | 8 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 1 | 1 | 1 | 1 | 7 | 7 |
|  |  |  |  |  | Count Total | 64 |

**Complexity adjustment value for function F2**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 0 |
| 2 | Are specialized data communications required? | 3 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 5 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 3 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 0 |
| 10 | Is the internal processing complex? | 3 |
| 11 | Is the code designed to be reusable? | 3 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 1 |
|  | Σ (Fi) | 20 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 54.4

**FP count for function F3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 2 | 2 | 2 | 2 | 4 | 8 |
| Number of External Output | 1 | 1 | 1 | 1 | 5 | 5 |
| Number of External Inquiries | 2 | 2 | 2 | 2 | 4 | 8 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 1 | 1 | 1 | 1 | 7 | 7 |
|  |  |  |  |  | Count Total | 68 |

**Complexity adjustment value for function F3**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 2 |
| 2 | Are specialized data communications required? | 3 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 1 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 3 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 0 |
| 10 | Is the internal processing complex? | 0 |
| 11 | Is the code designed to be reusable? | 5 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 5 |
|  | Σ (Fi) | 21 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP =58.48

**FP count for function F4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 2 | 2 | 2 | 2 | 5 | 10 |
| Number of External Inquiries | 4 | 4 | 4 | 4 | 4 | 16 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 2 | 2 | 2 | 2 | 7 | 14 |
|  |  |  |  |  | Count Total | 84 |

**Complexity adjustment value for function F4**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 2 |
| 2 | Are specialized data communications required? | 2 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 2 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 4 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 3 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 22 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 73.08

**FP count for function F5**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 2 | 2 | 2 | 2 | 5 | 10 |
| Number of External Inquiries | 4 | 4 | 4 | 4 | 4 | 16 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 2 | 2 | 2 | 2 | 7 | 14 |
|  |  |  |  |  | Count Total | 84 |

**Complexity adjustment value for function F5**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 2 |
| 2 | Are specialized data communications required? | 2 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 2 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 4 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 3 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 22 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 73.08

**FP count for function F6**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 2 | 2 | 2 | 2 | 5 | 10 |
| Number of External Inquiries | 4 | 4 | 4 | 4 | 4 | 16 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 2 | 2 | 2 | 2 | 7 | 14 |
|  |  |  |  |  | Count Total | 84 |

**Complexity adjustment value for function F6**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 2 |
| 2 | Are specialized data communications required? | 2 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 2 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 4 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 3 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 22 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 73.08

**FP count for function F7**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 2 | 2 | 2 | 2 | 5 | 10 |
| Number of External Inquiries | 4 | 4 | 4 | 4 | 4 | 16 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 2 | 2 | 2 | 2 | 7 | 14 |
|  |  |  |  |  | Count Total | 84 |

**Complexity adjustment value for function F7**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 2 |
| 2 | Are specialized data communications required? | 2 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 2 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 4 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 3 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 22 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 73.08

**FP count for function F8**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 2 | 2 | 2 | 2 | 5 | 10 |
| Number of External Inquiries | 4 | 4 | 4 | 4 | 4 | 16 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 2 | 2 | 2 | 2 | 7 | 14 |
|  |  |  |  |  | Count Total | 84 |

**Complexity adjustment value for function F8**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 2 |
| 2 | Are specialized data communications required? | 2 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 2 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 4 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 2 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 3 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 22 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 73.08

**FP count for function F9**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 1 | 1 | 1 | 1 | 5 | 5 |
| Number of External Inquiries | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 1 | 1 | 1 | 1 | 7 | 7 |
|  |  |  |  |  | Count Total | 60 |

**Complexity adjustment value for function F9**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 1 |
| 2 | Are specialized data communications required? | 1 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 1 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 2 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 0 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 5 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 17 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 49.2

**FP count for function F10**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 1 | 1 | 1 | 1 | 5 | 5 |
| Number of External Inquiries | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 1 | 1 | 1 | 1 | 7 | 7 |
|  |  |  |  |  | Count Total | 60 |

**Complexity adjustment value for function F10**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 1 |
| 2 | Are specialized data communications required? | 1 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 1 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 2 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 0 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 5 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 17 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 49.2

**FP count for function F11**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 1 | 1 | 1 | 1 | 5 | 5 |
| Number of External Inquiries | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 1 | 1 | 1 | 1 | 7 | 7 |
|  |  |  |  |  | Count Total | 60 |

**Complexity adjustment value for function F11**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 1 |
| 2 | Are specialized data communications required? | 1 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 1 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 2 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 0 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 5 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 17 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 49.2

**FP count for function F12**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Information Domain Value | Optimistic | Most Likely | Pessimistic | Est. Count | Weight | FP Count |
| Number of External Input | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of External Output | 1 | 1 | 1 | 1 | 5 | 5 |
| Number of External Inquiries | 1 | 1 | 1 | 1 | 4 | 4 |
| Number of Internal Logical Files | 4 | 4 | 4 | 4 | 10 | 40 |
| Number of External Interface | 1 | 1 | 1 | 1 | 7 | 7 |
|  |  |  |  |  | Count Total | 60 |

**Complexity adjustment value for function F12**

|  |  |  |
| --- | --- | --- |
| Number | Factor | Value |
| 1 | Does the system require reliable backup and recovery? | 1 |
| 2 | Are specialized data communications required? | 1 |
| 3 | Are there any distributed processing functions? | 0 |
| 4 | Is performance critical? | 1 |
| 5 | Does the system run in existing operational environment? | 0 |
| 6 | Does the system require on-line data entry? | 2 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Are the ILFs updated on-line? | 0 |
| 9 | Are the input, output, files or inquiries complex? | 2 |
| 10 | Is the internal processing complex? | 2 |
| 11 | Is the code designed to be reusable? | 5 |
| 12 | Are conversation and installation included in the design? | 0 |
| 13 | Is the system designed for multiple installations? | 0 |
| 14 | Is the system designed to facilitate change and ease of use? | 3 |
|  | Σ (Fi) | 17 |

FP-Estimated = Count Total \* [0.65 + 0.01 \* Σ (Fi)]

FP = 49.2

**Effort** = Total FP / Total no. of function

= 721.36/12

= 60.11

= 61 man month (Approximate)

**Time Frame Calculation**

= 61/ 8

=8 months

**3.6 Process Based Estimation**

Total system development is a combination of set of tasks. These set of tasks should done sequentially and timely. Project schedule works as the guideline of the system developer.

* + 1. **Process based estimation Table:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Activity** | **Customer**  **Communication** | **Planning** | **Engineering** | | **Construction** | | **Implementation** | **Total** |
| **Function** |  |  | **Analysis** | **Design** | **Code** | **Test** |  |  |
| F1 |  |  | 0.25 | 0.25 | 0.50 |  |  | 1.00 |
| F2 |  |  | 0.25 | 0.25 | 0.25 |  |  | 0.75 |
| F3 |  |  | 0.25 | 0.25 | 0.50 |  |  | 1.00 |
| F4 |  |  | 0.15 | 0.25 | 0.15 |  |  | 0.55 |
| F5 |  |  | 0.50 | 0.50 | 1.00 |  |  | 2.00 |
| F6 |  |  | 0.50 | 0.25 | 0.25 |  |  | 1.00 |
| F7 |  |  | 0.50 | 0.25 | 0.50 |  |  | 1.25 |
| F8 |  |  | 0.50 | 0.25 | 0.50 |  |  | 1.25 |
| F9 |  |  | 0.50 | 0.25 | 0.50 |  |  | 1.25 |
| F10 |  |  | 0.25 | 0.50 | 0.50 |  |  | 1.25 |
| F11 |  |  | 0.50 | 0.50 | 0.50 |  |  | 1.50 |
| F12 |  |  | 0.50 | 0.50 | 0.50 |  |  | 1.50 |
| **Total** | **1** | **1.5** | **4.65** | **4.00** | **5.65** | **1.5** | **0.75** | **19.5** |
|  |  |  |  |  |  |  |  |  |
| **Effort** | **5%** | **8%** | **24%** | **21%** | **30%** | **8%** | **4%** |  |

**3.6.2 Process based estimation pie chart:**

Figure3.3: Process based estimation pie chart

**3.6.3 Project Schedule chart:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Category** | **1st M** | **2nd M** | **3rd M** | **4th M** | **5th M** | **6th M** | **7th M** | **8th M** |
| Customer Communication |  |  |  |  |  |  |  |  |
| Planning |  |  |  |  |  |  |  |  |
| Analysis |  |  |  |  |  |  |  |  |
| Design |  |  |  |  |  |  |  |  |
| Code |  |  |  |  |  |  |  |  |
| Test |  |  |  |  |  |  |  |  |
| Implementation |  |  |  |  |  |  |  |  |

* 1. **Cost Estimation**

**3.7.1 Personnel cost (All in BDT):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **No. of Members** | **Months** | **Salary** | **Total** |
| System Analyst | 1 | 2 | 30,000 | 60,000/= |
| Senior Developer | 2 | 2 | 25,000 | 1,00,000/= |
| Junior Developer | 2 | 2 | 15,000 | 60,000/= |
| Web/Graphics Designer | 2 | 1 | 15,000 | 30,000/= |
| Customer Communicator/Tester | 1 | 1 | 15,000 | 15,000/= |
| **Total** |  |  |  | **2,65,000/=** |

**3.7.2 Hardware cost (All in BDT):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type of Product** | **No. of Product** | **Depreciation Price per Product(BDT)** | **Total Amount(BDT)** |
| Laptop | 6 | 32,500/= | 1,95,000/= |
| Desktop | 2 | 25,000/= | 50,000/= |
| Maintenance | | | 500/= |
| **Total** |  |  | **2,45,500/=** |

**3.7.3 Software cost (All in BDT):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type of Product** | **No. of Product** | **Depreciation Price per Product(BDT)** | **Total Amount(BDT)** |
| Windows 8 | 2 | 100/= | 200/= |
| Antivirus | 8 | 1000/= | 8000/= |
| Microsoft Office 2010 | 8 | 100/= | 800/= |
| **Total** |  |  | **9000/=** |

**3.7.4 Other cost (All in BDT):**

|  |  |  |
| --- | --- | --- |
| **Type of Product** | Monthly Bill**(BDT)** | **Total Amount(BDT)** |
| Electricity Bill | 2,500/= | 30,800/= |
| House Rent | 25,000/= | 1,50,000/= |
| **Total** |  | **1,80,800/=** |

Total Cost = 2, 65,000 + 2, 45,500 + 9,000 + 1, 80,800

**Total Cost (BDT): 7,00,300/=**

**In word:** Seven lac three hundred taka only.

* 1. **Activity Diagram:**

**3.8.1 Activity Diagram: Login**

Enter username, password

Correct

No

Yes

Access level define

Figure 3.4: Activity Diagram: Login

**3.8.2 Activity Diagram: Customer Registration**

Invalid information

No

Customer gives basic info for registration

Correct info

Yes

Registration Successful

Display conformation massage

type

Figure 3.5: Activity Diagram: Add User

**3.8.3 Activity Diagram: Update User**

Correct info

Customer enters user name/pass word

No

Invalid information

Yes

Update user info

Display conformation massage

Figure 3.6: Activity Diagram: Update User

**3.8.4 Activity Diagram: Delete User**

Correct info

No

Invalid information

Admin enters user name/pass word

Yes

Delete user info

Display conformation massage

Figure3.7: Activity Diagram: Delete User

**3.8.5 Activity Diagram: Add products**

Correct info

info

No

Invalid information

Admin enters user name/pass word

Yes

Products add

Display conformation massage

Figure 3.8: Activity Diagram: Add products

**3.8.6 Activity Diagram: Update products**

Correct info

info

No

Invalid information

Admin enters user name/pass word

Yes

Update products

Display conformation massage

Figure 3.9: Activity Diagram: Update products

**3.8.7 Activity Diagram: Create Sales Order**

Available

info

No

Cart empty

Customer choose product

Yes

Create Sales order

Display conformation massage

Figure 3.10: Activity Diagram: Create Sales Order

**3.8.8 Activity Diagram: Update Sales Order**

Correct info

info

No

Invalid information

Admin/User enters user name/pass word

Yes

Update Sales Order

Display conformation massage

Figure 3.11: Activity Diagram: Update Sales Order

**3.8.9 Activity Diagram: Online Payment**

Correct info

No

Invalid information

Customer enter credit card password

Yes

Payment Successfully

Display conformation massage

Figure 3.12: Activity Diagram: Online Payment

**Chapter: 4.0**

**Analysis and Design**

* 1. **Class Diagrams**

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

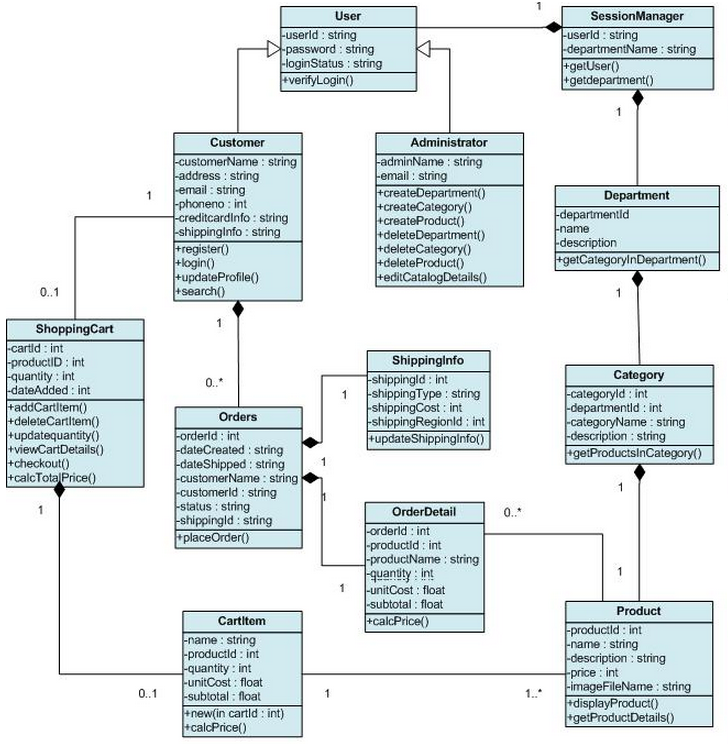
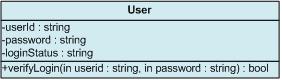


Figure 4.1: Class Diagrams

**4.1.1 Class Descriptions**

1. **User**



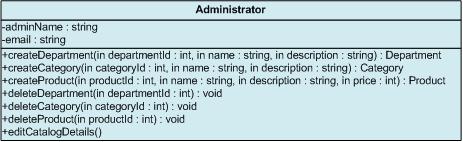
This class handles all user actions. The User class is the super class of Customer and Administrator. It includes the private methods to verify the login of the user. The verifyLogin() method is called when the user clicks the "Sign In" button. It returns true if the login is successful, false if it is not.

1. **SessionManager**

****

The SessionManager class supports the User’s required operations like getUser() and getdepartment(). The getUser method is called when the user information needs to be displayed. The getdepartment() method is used to get the exisiting departments in the catalog. When the user needs to get a product, the system has to start from the Session Manager and navigate down to the Product class.

1. **Administrator**

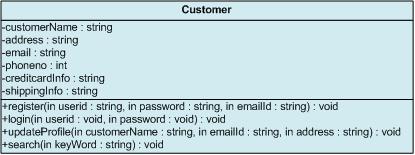
****

This class handles the Administrator actions. It inherits all the User class responsibilities and its functions. It has a functions createDepartment(), createCatogory(), createProduct() are called when the administrator has to create/add a new department or category or product respectively to the catalog. Similarly, operations like deleteDepartment(), deleteCategory() and deleteProduct() are called when deleting a department/category/product respectively from catalog. For these delete operations to be valid there should be atleast one department, category and product already existing in the Catalog. Also, the editCatalogDetails() method enables administrator to edit and update the product details.

**Conditions for createProduct Operation**

* Purpose: To enable Administrator to create and add new products to the catalog.
* Pre Condition: Administrator must be logged in to be able to create a new product. Also, the department and/or category to which the new new product is to be associated should exist in catalog.
* Post Condition: Product will be added to the corresponding department and/or category.
* Input: Administrator will enter the name and necessary details to create a new product and click "Add" button to complete the action.
* Output: After the action, the changes to the catalog will be updated and saved and a message will be displayed accordingly.

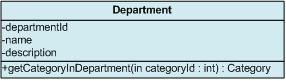
1. **Customer**

****

This class handles the customer actions. It inherits all the User class responsibilities and its functions. The updateProfile() method is called when customer wants to edit, update and save his/her personal information on the website for future use. But to update profile, customer must register and login through the Register.aspx and Login.aspx Webpage which uses the register() and login() methods. Also, the search() function is called when customer searches for some product and enters some Key-Words in the text-box on Search.aspx page.   
  
**Conditions for Login Operation**

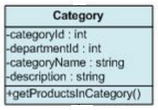
* Purpose: This is implemented to enable user authentication. A valid user account must be used for an existing customer.
* Pre Condition: The customer should be registered with website and should have a valid userid and password.
* Post Condition: Customer will be able to browse through the website.
* Input: The customer can login to the e-Commerce shopping system by entering his user name and password.
* Output: The system will verify that the login name matches the login password. If the user name or password is invalid, the appropriate error message will be indicated and the user will be requested to re-enter user name and password. If the user inputs are valid, the main page will be displayed.

1. **Department**



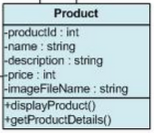
This class has all different departments line Jazz, Pop etc. This class has a private method called getCategoryInDepartment which is used get all the categories in a particular department. A department may contain zero or more categories and the corresponding products.

1. **Category**



This class has all different categories for the corresponding departments. This class has a private method called getProductsInCategory which is used get all the products bellonging to a particular category and department. A category can belong to one or more departments. Also, a category can contain zero or more products.

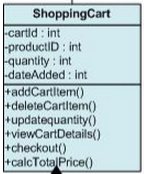
1. **Product**



This class represents collection of products of a particular category and/or department. The displayProduct() and getProductDetails() method are called when the user clicks on a product or on the "Search" button on the Search.aspx Web form. The displayProduct()

method is used to retrive image of the product and getProductDetails() method retrives its details as a data set whenever a customer clicks on the product. A product can belong to one or more departments and/or categories.

1. **ShoppingCart**



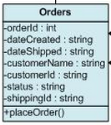
ShoppingCart class has all the products that are added by a customer to buy. The addCartItem() and deleteCartItem() methods are called when customer performs actions like "Add to Cart" and "Delete" of products on the ShoppingCart.aspx Webpage. The updatequantity() method is called when Customer clicks "Update" button on ShoppingCart.aspx Webpage to increase or decrease the number of products in the cart. Also, viewCartDetails() method is called when customer clicks on the "View Details" button to see a summary of the cart. It will display the summary only when cart is not empty, else it will display "Your Cart is empty."   
  
**Conditions for addCartItem Operation**

* Purpose: This is implemented so the customer can add products to shopping cart while searching or browsing catalog.
* Pre Condition: The customer must be logged in to add a product to the cart.
* Post Condition: Product will be added to the shopping cart.
* Input: When customer finds the products he wants, he adds them to the shopping cart by clicking on the "Add to Cart" button.
* Output: Product will be added to the shopping cart and the system will store and keep track of the information of products that have been added into shopping cart.

**Conditions for Checkout Operation**

* Purpose: To allow customer to buy the products added to the shopping cart.
* Pre Condition: Customer must be logged in and must have atleast one item in shopping Cart to be able to checkout and place the order.
* Post Condition: Customer will be succesfully checked out and will be able to edit his profile information and place order.
* Input: When the customer finishes shopping, he requests to checkout by clicking "checkout" button on Cart.aspx page.
* Output: If the payment information of this customer already exists, the system prompts the customer to review or input a new one. If the credit card is valid, the order form will be processed by the system and checkout is complete.

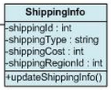
1. **Orders**



This class will store all information regarding the orders made by each customer. The placeOrder() method is called when customer clicks on the "Place Order" button on the Order.aspx Web form. It returns true if the order is placed successfully, false if it is not. The "Place Order" button will be enabled only when customer has a valid shopping cart and has entered valid personal, billing and shipping details.   
  
**Conditions for placeOrder Operation**

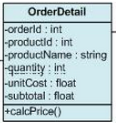
* Purpose: To allow customer to place an Order for buying the products added to the shopping cart.
* Pre Condition: Customer must be logged in and must have atleast one item in shopping Cart to be able to place the order Also, customer should have correct profile information and valid Credit Card details entered into the system.
* Post Condition: Customer will be succesfully able to place the order.
* Input: When the customer finishes shopping, he requests to place an order by clicking "Place Order" button on Order.aspx page.
* Output: If the profile information, payment information the credit card of this customer is valid, the order form will be processed by the system and order is placed.

1. **ShippingInfo**



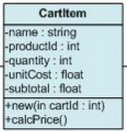
This class handles the shipping information regarding every customer and their orders. The updateShippingInfo() method is called when customer edits his/her shipping Information and clicks the "Update" button on Profile.aspx Webpage.

1. **OrderDetail**



This Class handles details regarding every order that the customer makes. The calcPrice() method is called to calculate the total amount of the order placed. The method also calculates the total amount of the order including the shipping charges once the customer has selected the shipping region.

1. **CartItem**



A CartItem object indicates the quantity, unitcost, subtotal amount of the product selected by a customer. When a customer performs a product selection and clicks "Add to Cart" button, a new cartItem object is created and added to the cart. The calcPrice() method is called to calculate the total amount of the items added to cart.

* 1. **Entity Relationship Diagram(ERD)**

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system’s entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

**The elements of an ERD are:**

* Entities
* Relationships
* Attributes

**Steps involved in creating an ERD include:**

1. Identifying and defining the entities
2. Determining all interactions between the entities
3. Analyzing the nature of interactions/determining the cardinality of the relationships
4. Creating the ERD

**Entity**

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.



An entity represented by a rectangle.

**Attributes**

Attributes are properties of entities. Attributes are represented by means of eclipses. Every eclipse represents one attribute and is directly connected to its entity (rectangle).



A relationship described by aneclipse.

**Relationship**

Relationships are represented by diamond shaped box. Name of the relationship is written in the diamond-box. All entities (rectangles), participating in relationship, are connected to it by a line.



A relationship described by a diamond.

**BINARY RELATIONSHIP AND CARDINALITY**

A relationship where two entities are participating, is called a binary relationship. Cardinality is the number of instance of an entity from a relation that can be associated with the relation

**One-to-one**

When only one instance of entity is associated with the relationship, it is marked as '1'. This image below reflects that only 1 instance of each entity should be associated with the relationship. It depicts one-to-one relationship



**One-to-many**

When more than one instance of entity is associated with the relationship, it is marked as 'N'. This image below reflects that only 1 instance of entity on the left and more than one instance of entity on the right can be associated with the relationship. It depicts one-to-many relationship



**Many-to-one**

When more than one instance of entity is associated with the relationship, it is marked as 'N'. This image below reflects that more than one instance of entity on the left and only one instance of entity on the right can be associated with the relationship. It depicts many-to-one relationship.



**Many-to-many**

This image below reflects that more than one instance of entity on the left and more than one instance of entity on the right can be associated with the relationship. It depicts many-to-many relationship



**PARTICIPATION CONSTRAINTS**

**Total Participation:** Each entity in the entity is involved in the relationship. Total participation is represented by double lines.

**Partial participation:** Not all entities are involved in the relationship. Partial participation is represented by single line.



**Primary Key:** A primary key is an attribute or collection of attributes that allow us to identify an entity uniquely.

**Foreign key:** A foreign key is an attribute of a relation, which refers to an existing attribute of another relationship.

**ER Diagram**

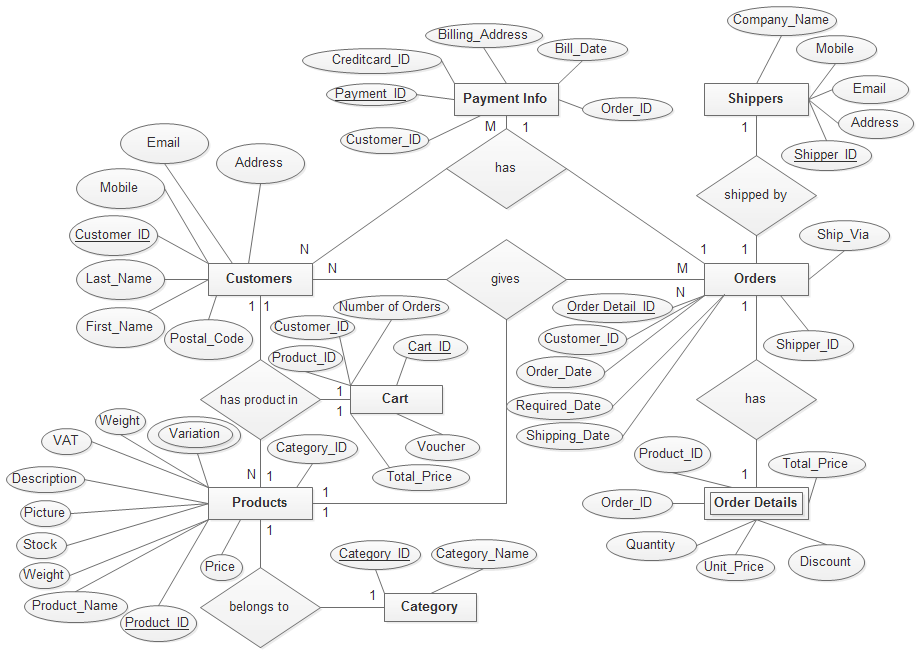


Figure 4.1: Entry Relationship Diagram

* 1. **Data Flow Diagram**

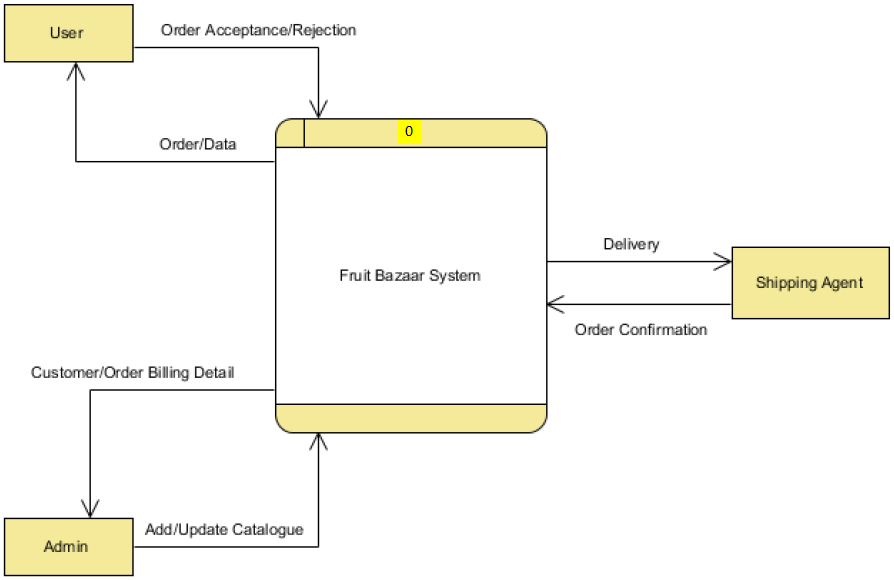
A data flow diagram is a graphical representation that depicts information flow and the transforms that are applied as data move from input to output.

The DFD may use to represent a system or software at any level of abstraction. DFD may partition into levels that represent increasing information flow and functional detail. Therefore, the DFD provides a mechanism for functional modeling as well as information flow modeling.

A level 0 DFD, which is also known as fundamental system model or a context model, represents the entire software or system element into as a single bubble with input and output data indicated by incoming and outgoing arrows respectively. Then bubble of context model should decompose into several levels.

**Data Flow Diagram of E-Commerce System:**

**Context Level Diagram**



**E-Commerce System**

Figure 4.2: Context Level Diagrams

The System is recognized as a complete system “E-commerce System. Admin, Users, Shipping  
Agent are the main actors for this system.

**Level 1 Diagram:**

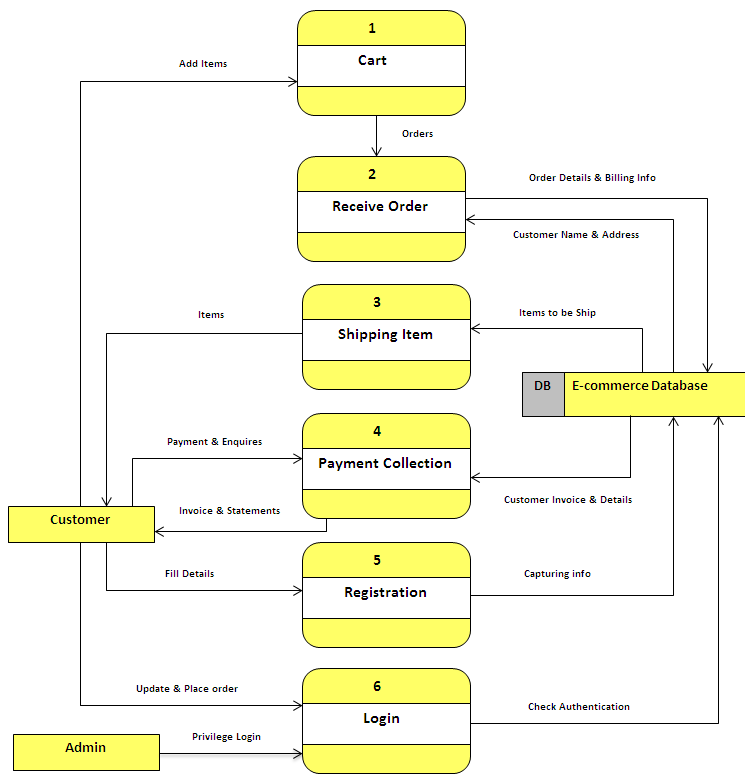
****

Figure 4.3: Level 1 Diagrams

User can Log-in, add, remove items to cart, register, pay. Admin can have priviledgedlogin,they can change, modify catalog, maintain user data. After adding items to cart they can make order .The orders are stored in order database. The orders are processed and items from the warehouse are delivered to customer by shipping agent.

**Level 2 Process 1 Diagram (Cart):**

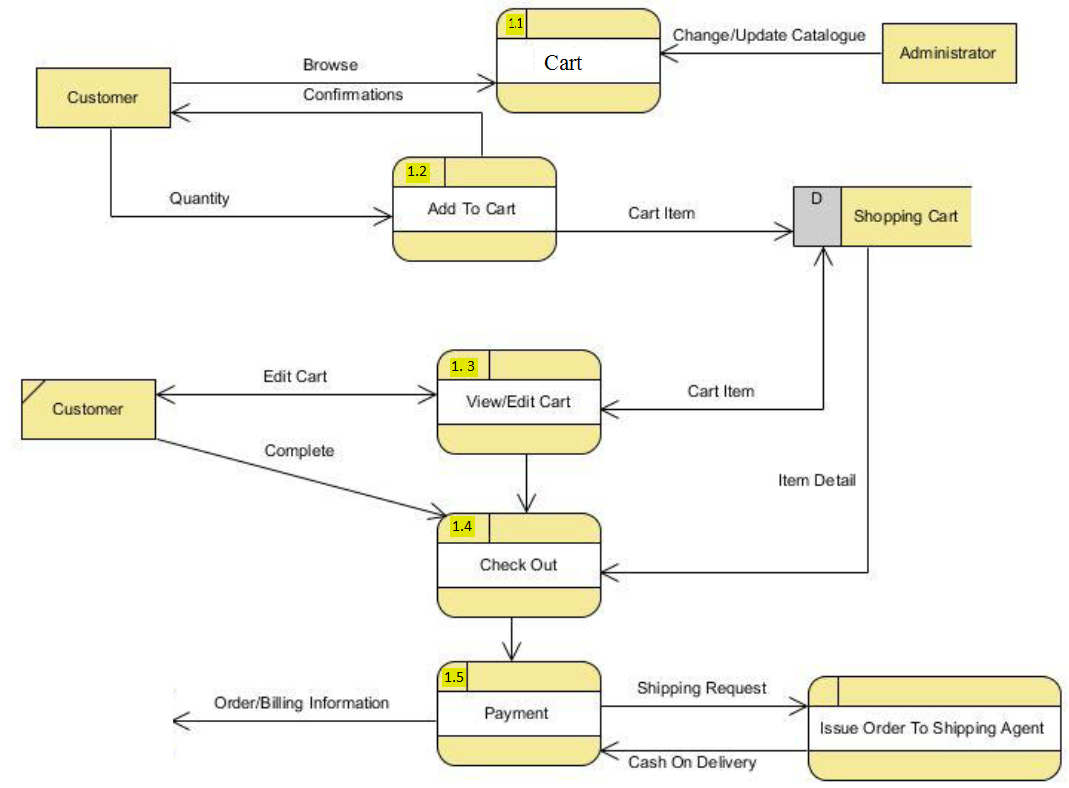
****

Figure 4.4: Level 2 of Process 1 Diagrams

Customer/User can browse the catalog and can add items to their cart for ordering. Moreover Customer can remove items from their cart. And after adding items to their cart they can checkout and move to Ordering. For Order a customer need to be log-in.

**Level 2 Process 2 Diagrams (Registration):**

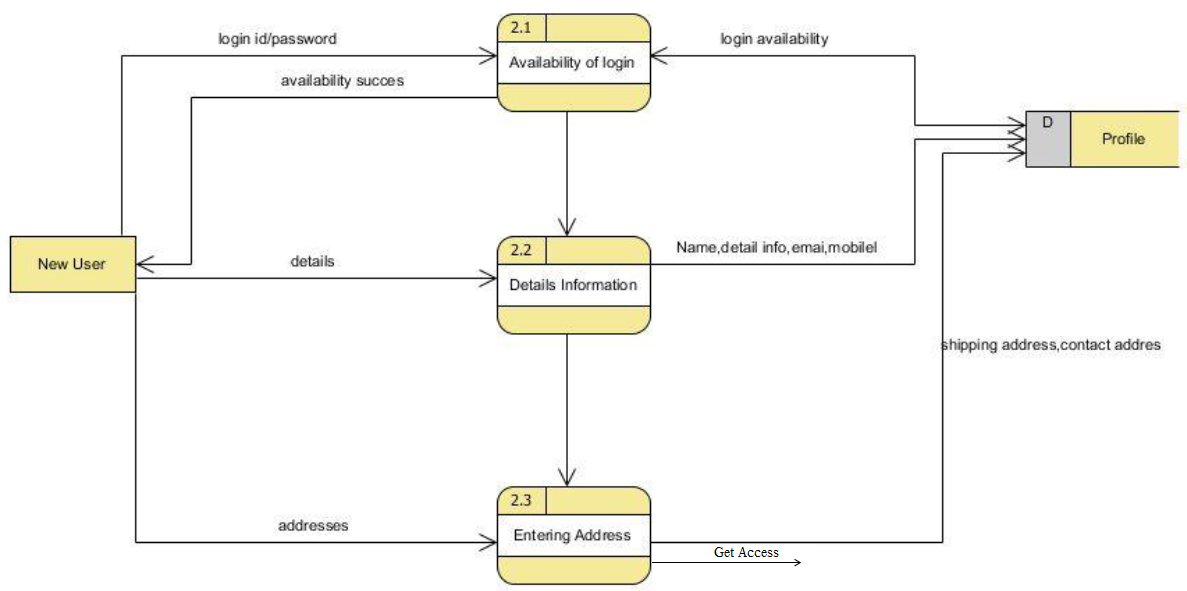
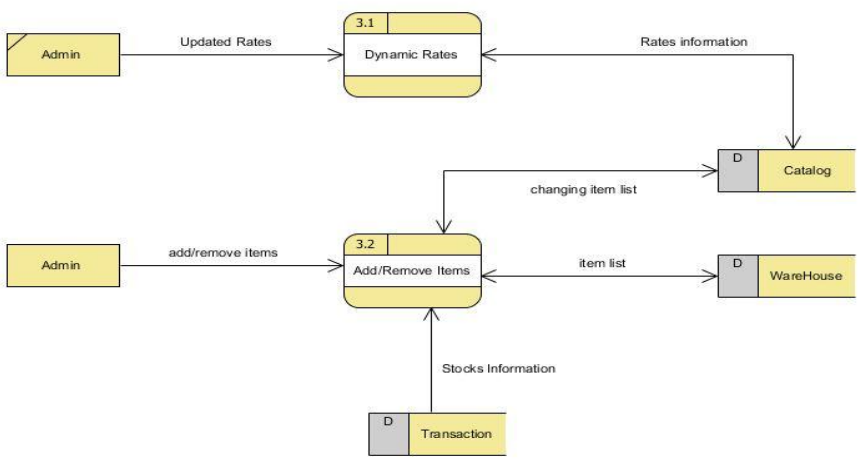
****

Figure 4.5: Level 2 Process 2 Diagrams

**Level 2 Process 3 Diagrams(Maintenance):**

****

Display

Figure 4.6: Level 2 Process 3 Diagrams

Admin can maintain the Catalog. The Catalog contains all items the are in stock with their rates. Admin can also change the catalog information like Dynamic rates according to market value. He can add remove items from catalog and stocks.

**Level 2 Process 4 Diagrams (Profile Handling):**

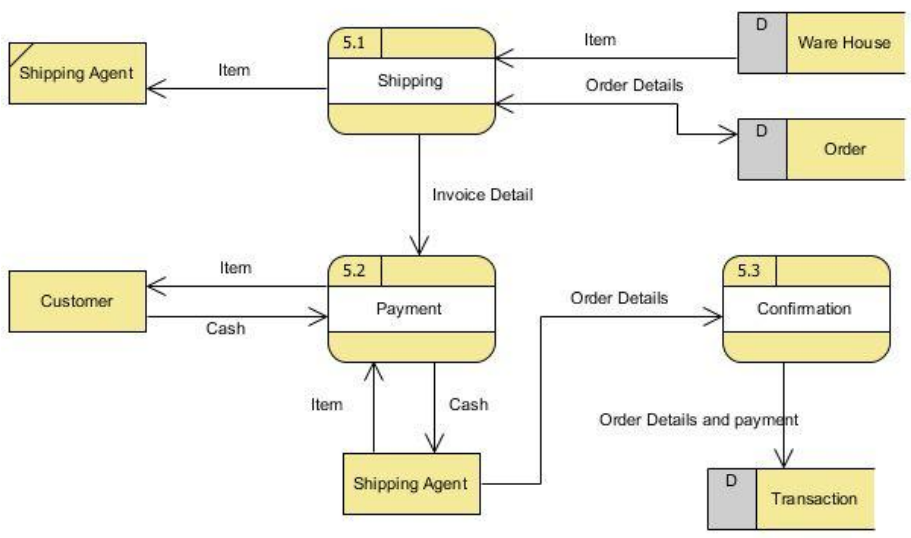
****

Use Site

Figure 4.7: Level 2 Process 4 Diagrams

User can manage their profile. Profile handling includes changing their account information like email, shipping address, passwords. If the user has created a invalid account then Admin can remove the account or can intimate User about any fault in their profile.

**Level 2 Process 5 Diagrams (Physical Transaction):**

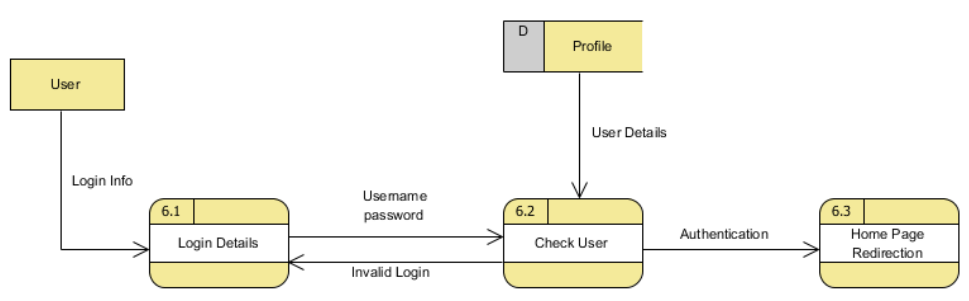
****

Sell

Figure 4.8: Level 2 Process 5 Diagrams

At the time of delivery A shipping Agent which will take the items from warehouse along with the bill and deliver the item to customer. Customer will pay the respective bill and the bill will be stored in transaction database .For keeping record of cash transaction we have a transaction database which will store bill amount and bill id.

**Level 2 Process6 Diagrams (LOGIN):**

****

LOGIN

Figure 4.9: Level 2 Process 6 Diagrams

User enters their login-id along with password. If the entered data is valid combination then home page will be displayed otherwise a message intimating user about their in-appropriate login details.

**Chapter: 5.0**

**Risk Engineering**

* 1. **Risk Engineering**

Risk analysis and management are a series of works that help a system development team to understand and manage uncertainty. Many problems can arise while developing a system. A risk is a potential problem – it may happen may not. There are several steps to analyze and manage risks. The first step is risk identification. Next each risk is analyzed to determine the likelihood that it will occur and the damage that it will do if it does occur. Once this information is established risks are remarked. Finally, a plan is developed to manage those risks with high probability and impact.

**There are different Stages of risks. They area:**

1. **Risk identification:** Risk identification is the process of detecting potential risks or hazards through data collection. A range of data collection and manipulation tools and techniques exists. The team is using both automated and manual techniques to collect data and begin to characterize potential risks to Web resources. Web crawling is one effective way to collect information about the state of Web pages and sites.
2. **Risk classification:** Risk classification is the process of developing a structured model to categorize risk and fitting observable risk attributes and events into the model. The team combines quantitative and qualitative methods to characterize.
3. **Risk assessment:** Risk assessment is the process of defining relevant risk scenarios or sequences of events that could result in damage or loss and the probability of these events. Many sources focus on risk assessment. Rosenthal describes the characteristics of a generic standard for risk assessment as "transparent, coherent, consistent, complete, comprehensive, impartial, uniform, balanced, defensible, sustainable, flexible, and accompanied by suitable and sufficient guidance.
4. **Risk analysis:** Risk analysis determines the potential impact of risk patterns or scenarios, the possible extent of loss, and the direct and indirect costs of recovery. This step identifies vulnerabilities, considers the willingness of the organization to accept risk given potential consequences, and develops mitigation responses.
5. **Risk management implementation:** defines policies, procedures, and mechanisms to manage and respond to identifiable risks. The implemented program should balance the value of assets and the direct and indirect costs of preventing or recovering from damage or loss.
   1. **The RMMM Plan:**

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: BR-01** | |  |
| Description | Installment risk for the project | Business Risk |
| Impact | It will make misunderstanding between client and organization | Disaster |
| Prevention | We will try to sell the project with full down payment. | |
| Cure | License code implementation | |
| Status | Working on process |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: BR-02** | |  |
| Description | The project obsolescence to the clients | Business Risk |
| Impact | It will effect to clients when they organized it. | Disaster |
| Prevention | Some extra features include within the projects. It will help us. | |
| Cure | Reduce the project system for clients and develop much updated way. | |
| Status | Done |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: BR-03** | |  |
| Description | Competition of different market competitors. | Business Risk |
| Impact | It will increase the competition among organizations. | Disaster |
| Prevention | Include some extra features within the projects. It will help us to convince the clients. | |
| Cure | Reduce the project cost for clients and focus on increase the clients. | |
| Status | Working on process |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: BR-04** | |  |
| Description | Privacy and security risk of project | Business Risk |
| Impact | Client will hamper and hopeless about the system. | Disaster |
| Prevention | Included extra privacy and security system. | |
| Cure | Resell the project very strongly by password system | |
| Status | We will work this way in future |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: TR-01** | |  |
| Description | Lack of adept technical persons for organization | Technical Risk |
| Impact | If it is happen then the client organization will be unable to operate the system. | Marginal |
| Prevention | Arrange the training session for operators. | |
| Cure | To send an expert process to the client organization for solve this problem | |
| Status | Done |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: TR-02** | |  |
| Description | Lack of implementation for organization | Technical Risk |
| Impact | The client fall in danger situation. | Marginal |
| Prevention | It will manage for the client to operate the system. | |
| Cure | For solving this problem, modify this system with more implement. | |
| Status | Done |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: TR-03** | |  |
| Description | Responsive for different devices | Technical Risk |
| Impact | If the system is not responsive then it will not possible to browse the site by different devices like tabs, mobiles etc. | Disaster |
| Prevention | Develop the system as responsive. | |
| Cure | If problem arise then need to update the system for responsive. | |
| Status | Working on process. |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: PR-01** | |  |
| Description | Lack of required knowledge or skill | Project Risk |
| Impact | The development will hamper. | Marginal |
| Prevention | Make detail conversation with the clients about their requirement before. | |
| Cure | To needed update the system according the client’s requirements. | |
| Status | Done |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: PR-02** | |  |
| Description | Expire the project accomplishment time. | Project Risk |
| Impact | The agreement can be dismissed by the client | Disaster |
| Prevention | Try to complete the project within time limit. | |
| Cure | Add extra team member with the development team. | |
| Status | Considered this term. |  |

|  |  |  |
| --- | --- | --- |
| **RMMM Plane No: PR-03** | |  |
| Description | Unclear or misunderstood scope/objectives | Project Risk |
| Impact | It will effete to the client organization and that will be unable to operate the system. | Marginal |
| Prevention | Clean and clearly declared for operators. | |
| Cure | Change some scope/objectives for solve this problem | |
| Status | We will work this way in future. |  |

**Chapter: 6.0**

**Coding**

* 1. **Flowchart**

A flowchart is a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure. The purpose of a flow chart is to provide people with a common language or reference point when dealing with a project or process.

**6.1.1 Login function:**

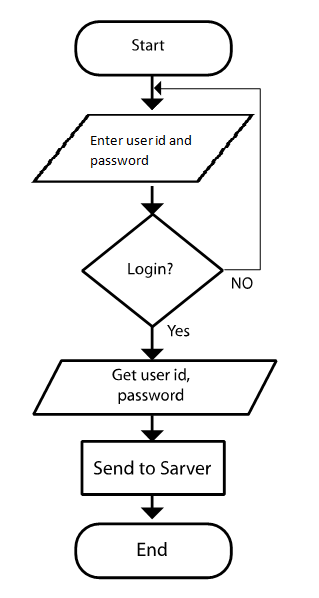
****

Figure 6.1: Client Side

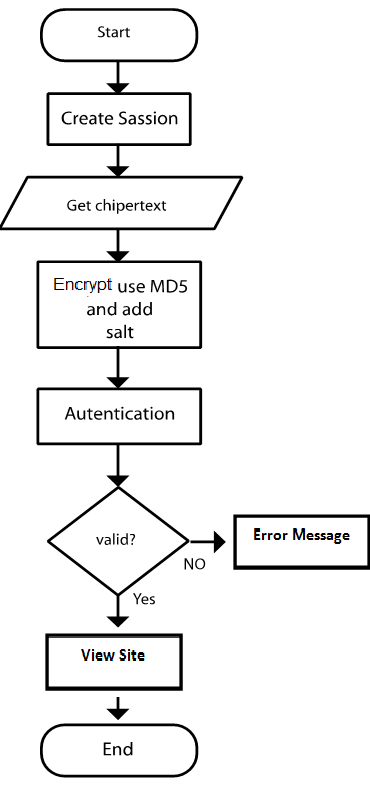


Figure 6.2: Sarver Side

* 1. **Database Diagram**

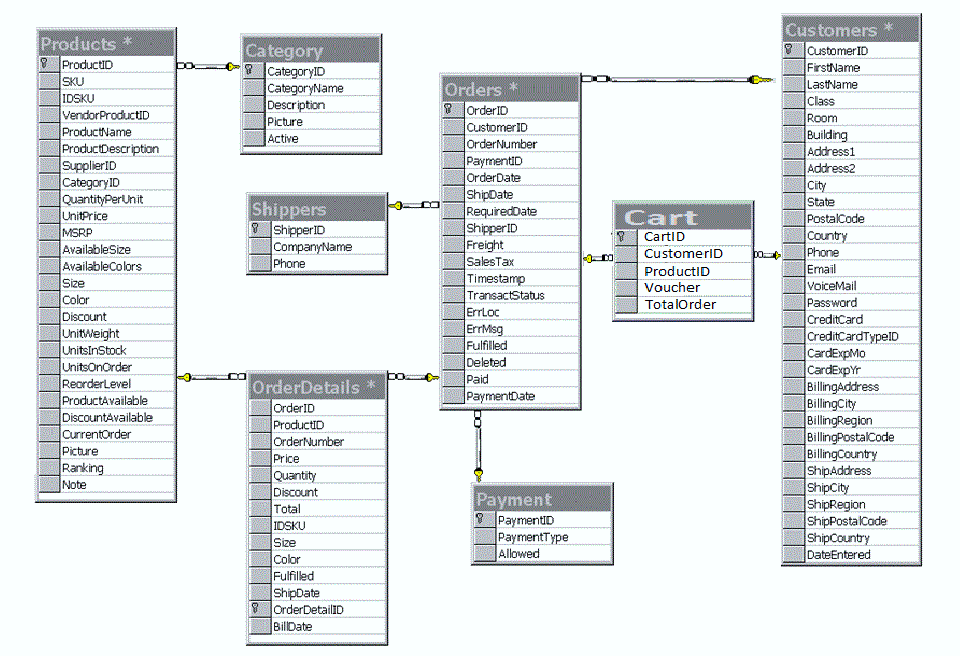
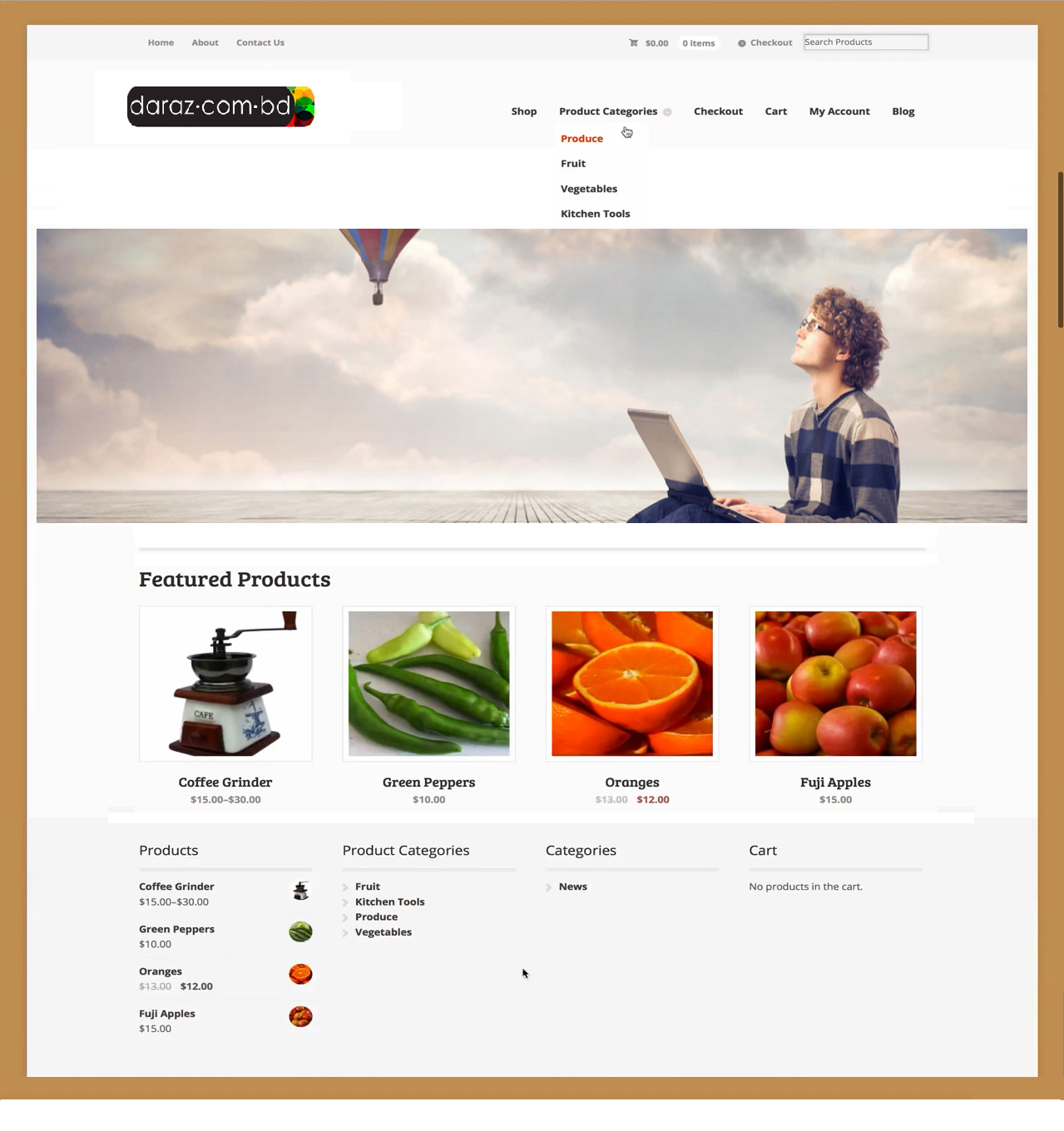


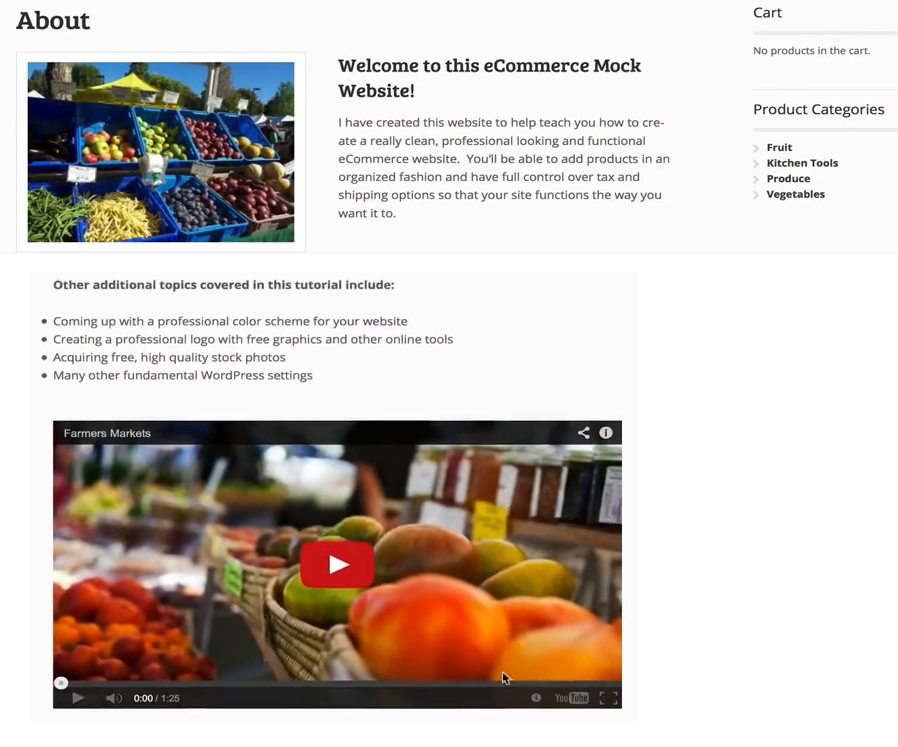
Figure 6.3: Database Diagram

* 1. **User Interface**

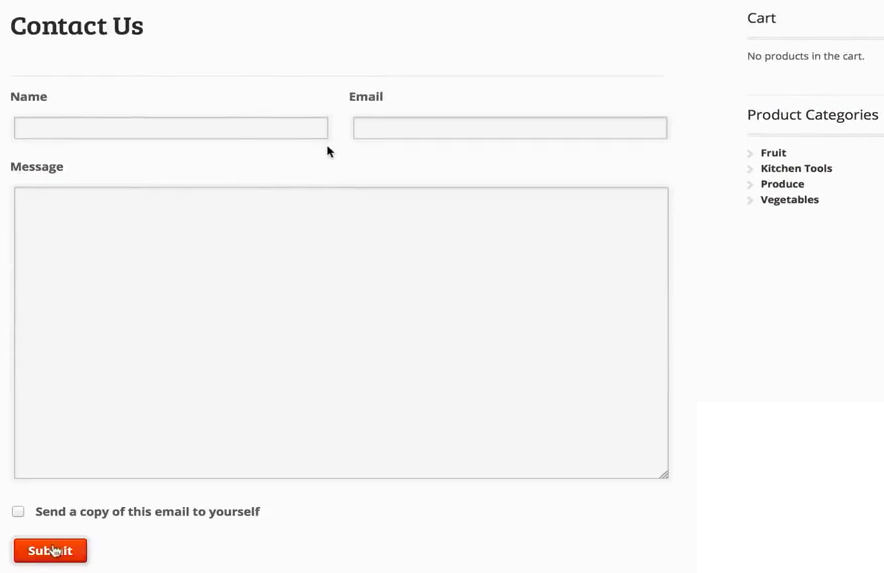
**6.3.1 Home Page**



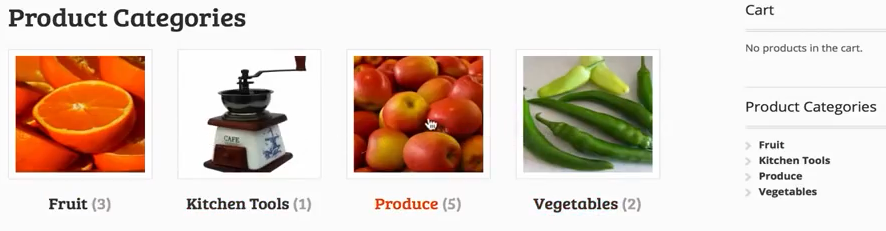
**6.3.2 About Page**

****

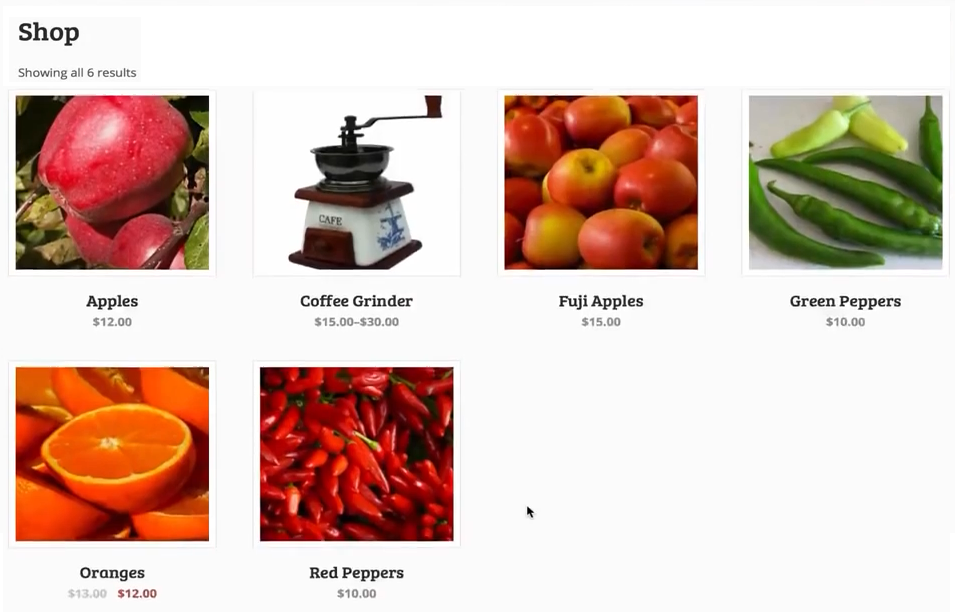
**6.3.3 Contact Page**

****

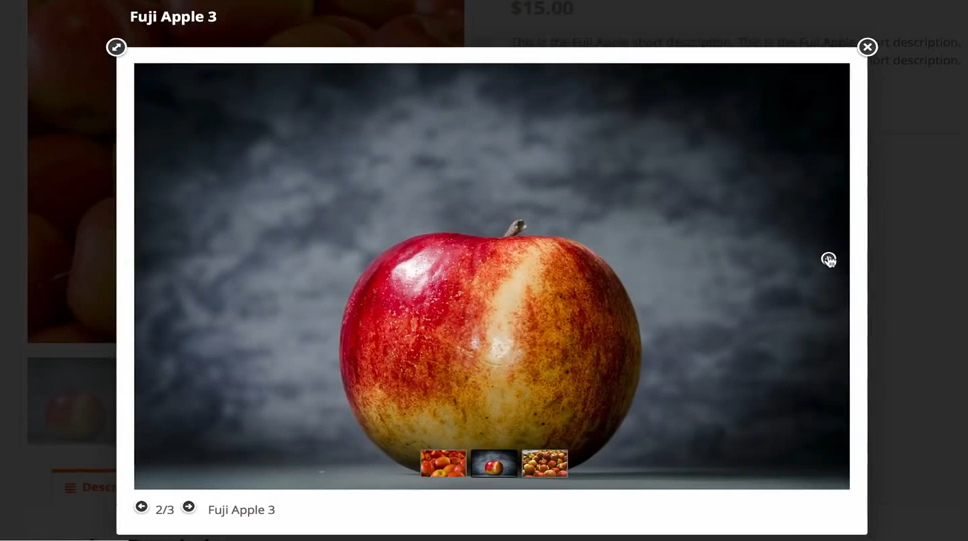
**6.3.4 Product Categories Page**



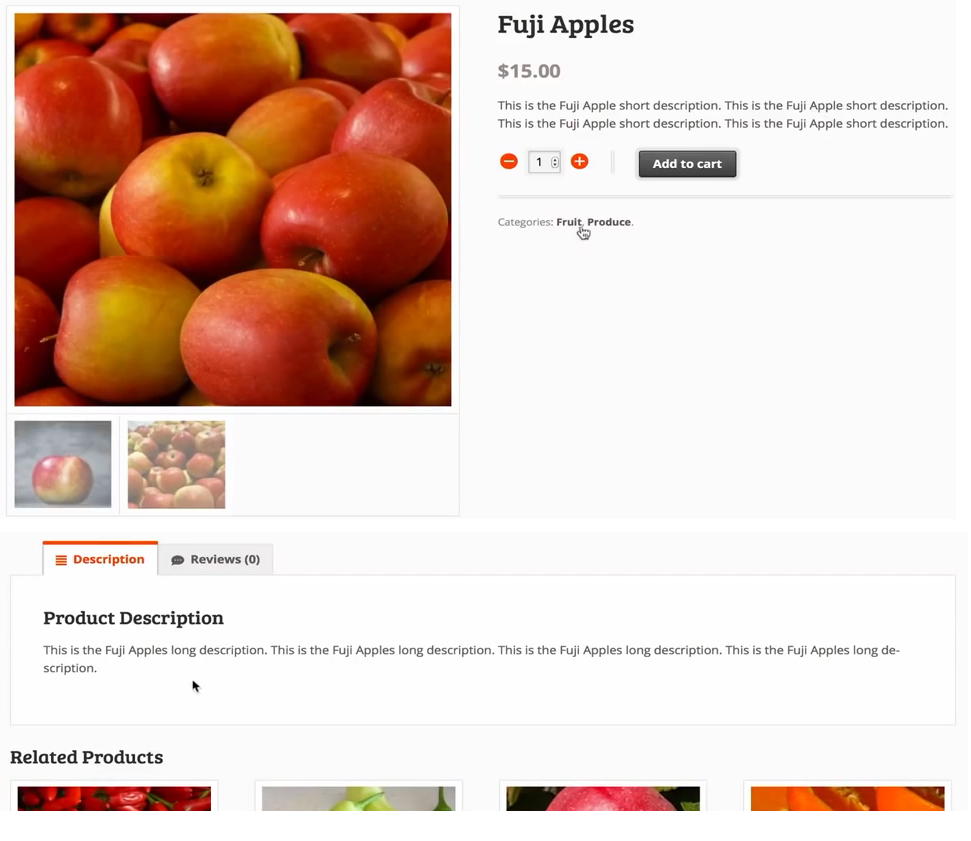
**6.3.5 Shop Page**

****

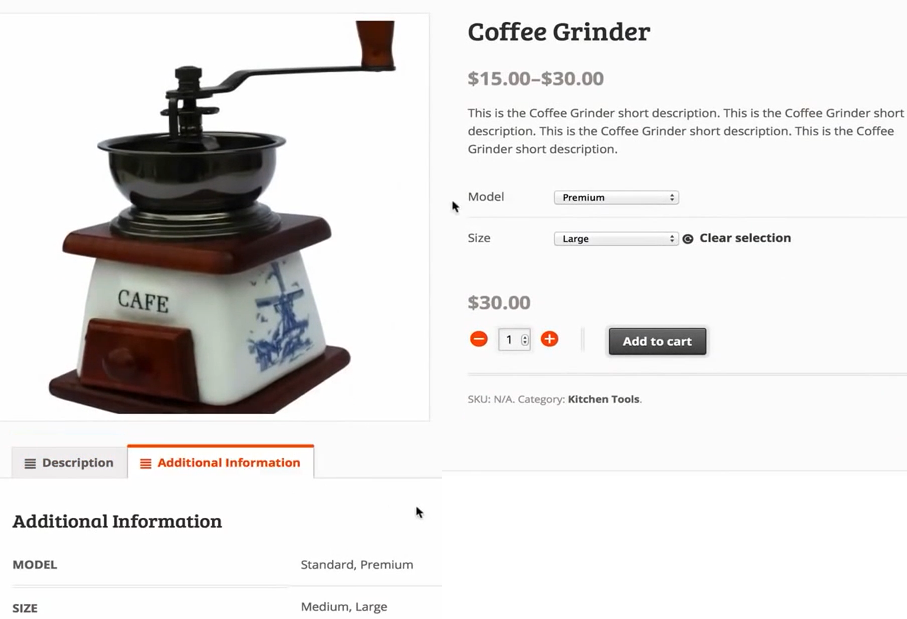
**6.3.6 Image Slider**

****

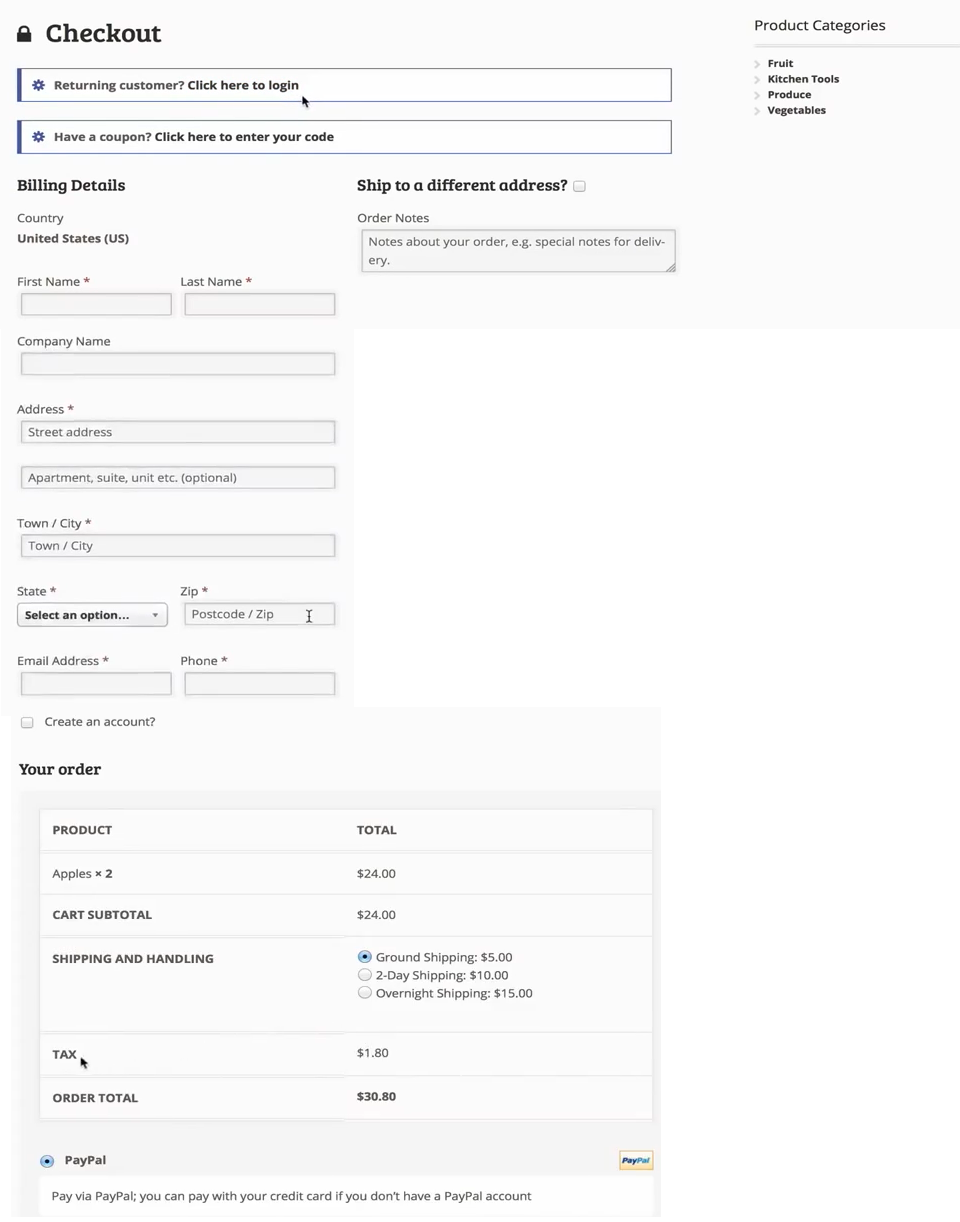
**6.3.7 Product description Page 1**

****

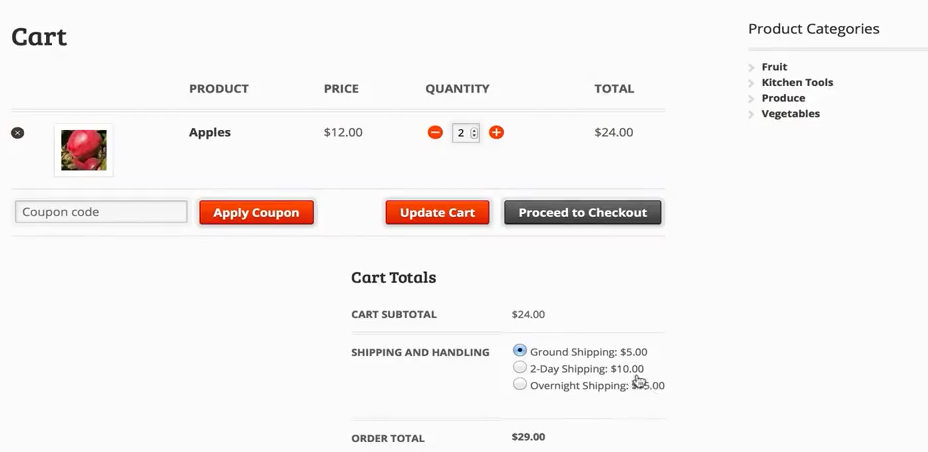
**6.3.8 Product description Page 2**

****

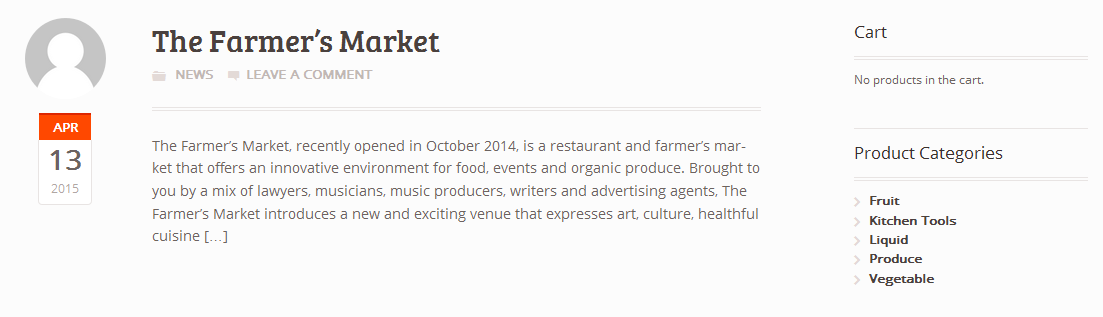
**6.3.9 Checkout Page**

****

**6.3.10 Cart Page**

****

**6.3.11 Blog Page**

****

**Chapter: 7.0**

**Quality Assurance**

* 1. **System testing**

According to the common process framework (CPF), the software testing is the final activity that has to initiate after testing. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

The objectives of software testing are:

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an as-yet-undiscovered error.
* A successful test is one that uncovers an as-yet-undiscovered error.

The design of tests for software can be challenging as the initial design of the product itself. Software can be tested in one of two ways:

* Knowing the specified function that the software has been designed to perform, tests can be conducted that demonstrate each function fully while at the same time searching for errors in each function. This approach is known as black-box testing.
* Knowing the internal workings of software, tests can be conducted to ensure that internal operations are performed according to specifications and all internal components have been adequately exercised. This approach is known as white-box testing.
  1. **Software Testing Strategy**

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of a software. The strategy provides a road map that describes the steps to be conducted as part of testing.

Testing strategy that will be followed in this software project –

* Unit testing
* Integration testing
* Validation testing

The first step in software testing is unit testing. Unit testing concentrates on each unit of the software as implemented in source code. Unit testing focuses on each component individually. The unit test is white-box oriented. Thus, unit testing of this library software will be done after completion of every module or component.

The next step is integration testing. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective of integration testing is to take unit tested components and build a program structure that has been dictated by design.

The integration testing strategy that has been chosen for this project is top down testing. Black-box testing method is the most prevalent for integration testing. Top down integration strategy will be used to perform integration testing. Top down integration will be done by breadth-first manner. Breadth-first integration incorporates all components directly subordinate at each level, moving across the structure horizontally.

After the software has been integrated, a set of high order tests are conducted. Hence, the validation criteria that have been mentioned in requirements engineering should be tested. Validation testing provides final assurance that software meets all functional, behavioral and performance requirements. The black-box testing method is exclusively used in validation.

* 1. **System Testing Methodology**
* **Black-box Testing**

Black-box testing which is also known as behavioral testing focuses on the functional requirements of the software. It enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black-box testing method will be applied to test the modules of LMS.

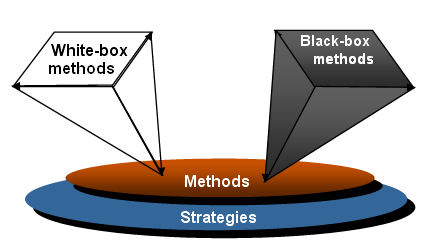


Figure 7.1: Black box & White box testing

* **White-box Testing**

White-box testing, which also known as *glass-box testing*,is a test case design method that uses the control structure of the procedural design to derived test cases. Using white-box testing methods, software engineer can derive test cases that,

1. guarantee that all independent paths within a module have been exercised at least once
2. exercise all logical decisions on their true and false sides
3. execute all loops at their boundaries and within their operational bounds
4. Exercise internal data structures to ensure their validity.

The modules that contain some complex calculations or decision making code such as check the availability of the library item will be tested using white-box method.

* 1. **Testing Design**

|  |  |
| --- | --- |
| Testing scenario No:1 | |
| Scenario | User Login testing scenario of our system |
| Input’s | Username, password of admin for Login |
| Desired Output’s | When enter username, password then get access level define. |
| Actual Output’s | For login our system work correctly |
| Verdict | Getting result from Desired Output’s and Actual Output’s decided this system is successful for login. |

|  |  |
| --- | --- |
| Testing scenario No:2 | |
| Scenario | User registration testing scenario of our system |
| Input’s | User’s basic info for registration |
| Desired Output’s | When enter all basic info correctly, user will be registered in the system . |
| Actual Output’s | For User registration our system work correctly |
| Verdict | Getting result from Desired Output’s and Actual Output’s decided this system is successful for User registration. |

|  |  |
| --- | --- |
| Testing scenario No:3 | |
| Scenario | Admin Login testing scenario of our system |
| Input’s | Username, password of user for Login |
| Desired Output’s | When enter username, password then get access level define. |
| Actual Output’s | For login our system work correctly |
| Verdict | Getting result from Desired Output’s and Actual Output’s decided this system is successful for login. |

|  |  |
| --- | --- |
| Testing scenario No:4 | |
| Scenario | Products insert testing scenario of our system |
| Input’s | Admin insert products details and quantity. |
| Desired Output’s | Products will show for Sales |
| Actual Output’s | I check this process and get actual outputs |
| Verdict | My system is worked correctly and successfully. |

|  |  |
| --- | --- |
| Testing scenario No:5 | |
| Scenario | Sales order creating scenario of our system |
| Input’s | Users select products for buy and insert quantity as well as use voucher. |
| Desired Output’s | Products will add into cart and create sales order. |
| Actual Output’s | I check this process and get actual outputs |
| Verdict | My system is worked correctly and successfully. |

|  |  |
| --- | --- |
| Testing scenario No:6 | |
| Scenario | Online payment testing scenario of our system |
| Input’s | User enters Credit card number for online payment. |
| Desired Output’s | Authority will check all necessary info for payment. |
| Actual Output’s | I check this process and get actual outputs |
| Verdict | My system is worked correctly and successfully. |

|  |  |
| --- | --- |
| Testing scenario No:7 | |
| Scenario | Update user testing scenario of our system |
| Input’s | User enters username, password and user information for user updating. |
| Desired Output’s | User will update user information then display information and conformation message. |
| Actual Output’s | I check this process and get actual outputs |
| Verdict | My system is worked correctly and successfully. |

|  |  |
| --- | --- |
| Testing scenario No:8 | |
| Scenario | Delete user testing scenario of our system |
| Input’s | Admin enters username, password and user type for user deleting. |
| Desired Output’s | Admin will delete user information then display conformation message. |
| Actual Output’s | I check this process and get actual outputs |
| Verdict | My system is worked correctly and successfully. |

|  |  |
| --- | --- |
| Testing scenario No:9 | |
| Scenario | Update stock testing scenario of our system |
| Input’s | Admin insert new stock information. |
| Desired Output’s | Update stock will show. |
| Actual Output’s | My desired output access to actual and practical output. So this is successful. |
| Verdict | My system is worked correctly and successfully. |

|  |  |
| --- | --- |
| Testing scenario No:10 | |
| Scenario | Customer feedback testing scenario of our system |
| Input’s | Customer gives product review or service feedback. |
| Desired Output’s | Review and feedback will show in website. |
| Actual Output’s | My desired output access to actual and practical output. So this is successful. |
| Verdict | My system is worked correctly and successfully. |

* 1. **Quality Assurance Matrix**

|  |  |
| --- | --- |
| Quality Assurance scenario no: 1 | |
| Requirement | The system worked according to requirement. |
| Provided outputs | My requirement was Customer enters basic info for registration. After registration they get access in the system. |
| Decision | This system working correctly. We hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 2 | |
| Requirement | The system worked according to requirement. |
| Provided outputs | My requirement was Customer enters username and password for login. After login they get access level for define. |
| Decision | This system working correctly. We hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 3 | |
| Requirement | The system worked according to requirement. |
| Provided outputs | Admin insert new stock information. Update stock will show. |
| Decision | This system working correctly. We hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 4 | |
| Requirement | The system worked according to requirement |
| Provided outputs | The requirement was user insert products and gets ready the product for sale. |
| Decision | This system working correctly. We hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 5 | |
| Requirement | The system worked according to requirement |
| Provided outputs | Customer gives product review or service feedback. Review and feedback will show in website. |
| Decision | I hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 6 | |
| Requirement | The system worked according to requirement |
| Provided outputs | User will update user information. After that display user info and conformation massage. |
| Decision | I hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 7 | |
| Requirement | The system worked according to requirement |
| Provided outputs | Admin will delete user information of user. After that display conformation massage. |
| Decision | I hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 8 | |
| Requirement | The system worked according to requirement |
| Provided outputs | The requirement was user will add product into cart and sales order generated. |
| Decision | I hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 9 | |
| Requirement | The system worked according to requirement |
| Provided outputs | User will insert user name and password then see their profile. |
| Decision | I hope our system will work in future. |

|  |  |
| --- | --- |
| Quality Assurance scenario no: 10 | |
| Requirement | The system worked according to requirement |
| Provided outputs | User enters Credit card number for online payment. Authority will check all necessary info for payment. |
| Decision | I hope our system will work in future. |

* 1. **Things for Implementation**

**7.6.1 Process:** It is critical that the organization defines a process that is robust and certified by experts in order to initiate the software assurance quality culture. The process will serve as a guideline that may evolve over time. Most importantly, it should be made official and should be followed through. Improvements will be made until a mature process is established.

**7.6.2 Managerial Commitment:** Managerial commitment should stem from the CIO to ensure alignment from each of the development managers, as well as from the development areas of each country. Everyone must be aware of the value that is added by testing & QA to the business. The process, therefore, must account for the value of the solutions that it offers to the organization.

**7.6.3 Personal Experience:** Hiring someone as a tester that lacks necessary experience is a common mistake. It is vital to acknowledge that the position requires experience in both the business and in software development in general.

**7.6.4 Deliverables:** As part of the software development and testing processes, it is necessary to define deliverables, such as requirements, a testing plan, and testing cases. These will guarantee that testers can effectively follow-up throughout the project from the software quality perspective.

**7.6.5 Tool Usage:** Both the use of tools for tracking and managing defects, as well as the creation of test cases and execution, are essential for increasing the maturity of the testing & QA process. The process may begin without tools, but they are a requisite for increasing execution maturity.

**7.6.6 Metrics:** Developing and creating metrics to track the software quality in its current state, as well as to compare the improvement with previous versions, will help increase the value and maturity of the testing process.

**7.6.7 Testing Environment:** Implementation of appropriate testing environments that allow developers to reproduce the system execution in production environments is crucial to the creation and execution of the corresponding test cases.

**7.6.8 Test Data:** The testing environment required for day-to-day operation should provide or ensure availability of the necessary data to enable the corresponding test execution

**7.6.9 Change Management:** Like any other production environment, the testing environment should properly track changes in configuration, ensuring not only controlled results, but that the tests are run in environments that closely resemble those of the real production environment.

**7.6.10 Developer Awareness:** It is critical to have an awareness process that includes management commitment at each and every business unit and for associated developers. The goal is to demonstrate that testing activities add value to their daily work.

**Chapter: 8.0**

**Conclusion**

* 1. **Conclusion**

My project is only a humble venture to satisfy the needs in an online shop. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the organization.

The objective of software planning is to provide a frame work that enables the manager to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

Although I could not include all the functionality that I thought to include in this software, I worked hard to make it fully functional in this small amount of time. As our knowledge of programming grows by time, I shall look to make it a better one in every possible way. It is my heartiest request to the teachers, supervisor and all the user who will evaluate this project should bear in mind that I am still learning and in a beginner level at this moment.

Honesty, I worked very hard to make this project full-fill but as I am human being there always be some errors. I tried to make it errorless as far as possible and successful in every step. I hope this software project serve well to its benefactor.

***Bibliography***

**Books and Internet Data:**

[1] Chen, L. (2000). Enticing Online Consumers: A Technology Acceptance Perspective Research- in-Progress.ACM Proceedings, SIGCPR.

[2] Silberschatz, Abraham, Korth, Henry F, and Sudarshan S. *Database System Concepts.* 4th ed. Boston: McGraw Hill, 2002.

[3] Kendall, E. Kendall. *System Analysis and Design.*4th edition. New Delhi: Prentice Hall, 1999.

[4] Pressman, Roger S. *Software Engineering:A Practitioner’s Approach.* 5th edition. Boston: McGraw Hill, 2004.

[5] Roger S. Pressman “Software Engineering- A practitioner’s approach”, 7th Ed. (BS).

[6] O’Brien, James and M. Markas, George. Management Information System.8th International ed.2008.

[7] Miller, Dave. Data Communication and Networks.1st ed.2006.

[8]Kubilus, N. J. (2000). Designing an e-commerce site for users. September 2000, Crossroads, Volume 7 Issue 1.

[9]Wordpress(March2015) [Online].

Available:https://codex.wordpress.org/Theme\_Development

[10] Github (March 2015) [Online].

Available: <https://github.com/panique/php-mvc>

[11] php.net (March 2015) [Online].

Available:http://php.net/manual/en/pdo.prepared-statements.php

[12] Codeigniter (March 2015) [Online].

Available: http://www.codeigniter.com/docs

[13] Bootstrap (March 2015) [Online].

Available:http://www.tutorialspoint.com/bootstrap/bootstrap\_code.htm

[14] Amazon (March 2015) [Online].

Available: http://www.amazon.com/gp/product/

[15] Ebay (March 2015) [Online].

Available:<http://www.ebay.com/cln/inspiredbycharm/Unique-ContainerGardens/194775859018>

[16] Kaymu (March 2015) [Online].

Available:<http://www.kaymu.com.bd/?gclid=COm3r-OQ7MQCFU8mjgodj0YAkA>