**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

This project is a web based application specially for Ecommerce sector. To manage bakery with Ecommerce portal. Lezat having 5 types of users, vendors, affiliate marketer, customers, Admin, and delivery boy.

Lezat is a form of ecommerce portal where buyer is directly online to the seller’s computer usually via the internet. Lezat allows the customers to check and purchase various bakery products displayed in various categories and the user can browse through the items. Customer need to register on the site before add the selected item to the cart.He can then login using the same id ,password next time.Now he may pay through a credit card or cash on delivery.The customer receives a copy of the shopping receipt on his email id after the successful transaction. Location and information is provided in the web site to help deliveries run smoothly.

Lezat give an option for customers to become an affiliate marketer. Affiliate marketing is the process of earning a commission by promoting other peoples products. Customer buys a product, promote it to others and earn a piece of commission for each sale that you make.

Lezat provide vendor management. Every vendor have sells the products of Lezat.

**1.2 SCOPE OF THE PROJECT**

In this busy schedule LEZAT is convenience to purchase items at home. It saves time and efforts. Customers do not have to stand in queues in cash counters to pay for the products that have been purchased by them. They can purchase from their home or workplace and do not have to spend time travelling.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2. SYSTEM ANALYSIS**

The analysis is the process of understanding problem or requirements of the software in detail. These requirements will help to understand the existing system and determine the processes to be managed by the software. The “LEZAT” is an ecommerce web application.

**2.1 REQUIREMENT ANALYSIS**

Requirements analysis results in the specification of software’s operational characteristics, indicates software’s interface with other system elements, and establishes constraints that software must meet. Requirements analysis allows you to elaborate on basic requirements established during the inception, elicitation, and negotiation tasks that are part of Requirements engineering.

**REQUIREMENT GATHERING**

The requirement gathering can be done by following ways.

* Interview
* Questionnaire
* Site visit
* Website visit

LEZAT is a live project and I collect the requirements from the organization and proof attached.

**2.2 EXISTING SYSTEM**

In this busy schedule, It is very difficult to go to the bakery shop and face the hassle of packing and the crowd formed in front of the items displayed and the long queue at the billing counter.

Disadvantages of existing system:

* wastage of time
* travelling expense
* lack of variety items
* Inconvenience for purchase
* high price

**2.3 PROPOSED SYSTEM**

This project is a web based application especially for Ecommerce sector. To manage bakery with Ecommerce portal. Lezat having 5 types of users, vendors, affiliate marketer, customers, Admin, and delivery boy.

Lezat is a form of ecommerce portal where buyer is directly online to the seller’s computer usually via the internet.Lezat allows the customers to check and purchase various bakery products displayed in various categories and the user can browse through the items. Customer need to register on the site before add the selected item to the cart. Customer can login using the same id, password next time to purchase items. They can make payment through online or at the time of delivery (COD). The customer receives a copy of the shopping receipt on his email id after the successful transaction. The product delivered to the customers based on their registered address.

Lezat give an option for customers to become an affiliate marketer. Affiliate marketing is the process of earning a commission by promoting other peoples products. To become an Affiliate marketer the customer needs buy products for a fixed amount(decided by the company).Promote it to others and earn a piece of commission  by a product share links through social medias like -.Whatsapp, Face book … etc. If a customer buys a product through that link within 30 days, then he/she will get the commission, but if after 30 days, then the company will get the commission.

Lezat provide vendor management. Every vendor have sells the products of Lezat through online or direct. Lezat will assign a delivery boy to each vendors.

**Features of LEZAT**

We have added affiliate marketing system in this online bakery system so customers can market Lezat products to new customers and earn money. This feature will help Lezat to market their products without spending money. Their customers will be marketers. This single feature will bring new customers and increases the store sales.

Vendor can sell Lezat products in their store. Every vendor will have their own shop so that they will do marketing and Lezat will be promoted automatically.

* Secure registration and profile management facilities for customers.
* Saves time and efforts.
* The convenience of purchase at home.
* Wide variety/range of products is available.
* Good discounts / lower prices.
* Get detailed information about the product.
* Customers do not have to stand in queues in cash counters to pay for the products that have been purchased by them. They can purchase from their home or workplace and do not have to spend time travelling.
* Online consumers can track the order status and delivery status is also available.

**2.4 FEASIBILITY STUDY**

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that is spent on it. Feasibility study lets the developer foresee the future of the project and the usefulness. Feasibility study is a test of the system proposed regarding its Workability, impact on the organization, ability to meet the needs and effective use of resources. Thus, when a new project is proposed, it normally goes through a feasibility study before it’s approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as technical, legal, economic and behavioral feasibilities.

The feasibility of a project can be ascertained in terms of technical, economic, behavioral and legal factors. A feasibility study is documented with a report showing all the ramifications of the project.

The system proposed is tested whether it is feasible by conducting the following

* Technical feasibility
* Economic feasibility
* Behavioral feasibility

**2.4.1** **TECHNICAL FEASIBILITY**

“Lezat” is a web application. The application will provide user friendly environment also. The technical capability of the personnel as well as the capability of the available technology should be considered. Since using the front end angular and backend laravel with XAMPP server doesn’t find any sort of difficulties while using the same. The proposed system can run on any machines and works on the best software and hardware that had been used while designing the system so it would be feasible in all technical terms of feasibility.

**2.4.2** **ECONOMIC FEASIBILITY**

Economic analysis is frequently used to evaluate the effectiveness of the new system. It determines whether it is worthwhile to invest the money in the new project or not. The procedure is to determine the benefits and saving that are expected from a candidate system and compare with the existing system. If the benefits of the new system are out of weight the existing, the decision is made to design and implement. So there is no need for procuring additional hardware and software for the system. The proposed will replace the hectic cost and man power involved in the existing system. Thus the project is economically feasible.

**2.4.3** **BEHAVIOURAL FEASIBILITY**

The System is designed in user friendly manner and we need not to provide any special training for the persons using this software. It does not have any operational barriers. So no need to provide any special training for using this application software and hence it is behaviorally feasible.

**2.5 SYSTEM REQUIREMENT SPECIFICATION**

System requirements are expressed in a software requirement document. The Software requirement specification (SRS) is the official statement of what is required of the system developers. This requirement document includes the requirements definition and the requirement specification. The software requirement document is not a design document. It should set out what the system should do without specifying how it should be done. The requirement set out in this document is complete and consistent. The software specification document satisfies the following:-

* It specifies the external system behaviors.
* It specifies constraints on the implementation.
* It is easy to change.
* It serves as reference tool for system maintainers.
* It record forethought about the life cycle of the system.
* It characterizes acceptable response to undesired events**.**

**2.5.1** **ACTOR IDENTIFICATION**

An actor is someone or something that interacts with the system. An actor is he /she who use the system. An actor exchanges information with the system. Asking certain questions as detailed below can identify the actors of the system.

|  |  |  |
| --- | --- | --- |
| **1.** | Who will use the main functionality of the system? | Admin, |
|  |  | Customer |
|  |  | vendor, |
|  |  |  |
|  |  |  |
| **2.** | Who will lead support from the system and do their daily | Admin, |
|  | tasks? | vendor |
|  |  | Delivery boy |
|  |  |  |
|  |  |  |
| **3.** | Who will maintain and administrate the system? | Admin |
|  |  |  |
| **4.** | The external device needs to communicate? | Modem |
|  |  |  |
| **5.** | With which other systems, does this system need to | Database. |
|  | interact? |  |
|  |  |  |
| **6.** | Who was interest in the result produced by the system? | Admin |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

As per the above answers we can conclude the actor. They are

* Administrator.
* Customer.
* Vendor.
* Affiliate Marketer.
* Delivery boy.

**2.5.2** **USECASE IDENTIFICATION**

A use cases represents the functionality of an actor. It is defined as a set of actions performed by a system, which yields an observable result. An ellipse containing its name inside the ellipse or below it represents it. It is placed inside the system boundary and connected to an actor with an association. This shows how the use cases and the actor interact.

To find out the use cases, ask the following questions to each of the actors

* Which functions does the actor require from the system? What does the actor need to do?
* Does the actor need to read, create, destroy, modify or store some kind of information in the system?
* Could the actor’s daily work be simplified or made more efficient by adding new functions to the system?

**2.5.2.1 USE CASES**

**Use case for the actor Administrator**

|  |  |
| --- | --- |
| Which functions does the Administrator | Administrator requires the following |
| require from the system? What does the | functionalities from the system such as |
| Admin need to do? | login, add product, manage vendors , add |
|  | Staff ,Accept vendors request, Send products |
|  | View orders, cancel orders, manage delivery, |
|  | Add affiliate marketer, Add offers |
|  |  |
| Does the Administrator need to read, | Yes. Administrator need to create, view |
| create, destroy, modify or store some | and edit the data if require. |
| kind of information in the system? |  |
|  |  |
| Could the Administrator work be | Yes, the system can reduce his/her |
| simplified by adding new functions to | work. |
| the system? |  |
|  |  |

Above questions give the following use cases for the actor Admin.

* Login
* Add staff
* Add products.
* Accept vendors request.
* send products to vendors.
* manage delivery
* Add affiliate marketer.
* Add offers
* Manage customers.
* manage social network settings.
* view orders.

**Use case for Actor vendor**

|  |  |
| --- | --- |
| Which functions does the vendor require from | Vendor requires the following functionalities |
| the system? What does the vendor need to do? | from the system such as login, manage customers |
|  | view product, view stock, view reports, |
|  | update purchase limit, and logout |
|  |  |
| Does the vendor need to read, create, | Yes. vendor need to create, view and edit |
| destroy, modify or store some kind of | the data if require. |
| information in the system? |  |
|  |  |
| Could the vendor work be simplified by | Yes, the system can reduce his/her |
| adding new functions to the system? | work. |
|  |  |

Login

* view products
* Request for products
* Manage customers
* Payment
* view delivery report
* Change profile.
* Update purchase limit
* View top sold product
* Change password
* Logout

**Use case for Actor customer**

|  |  |
| --- | --- |
| Which functions does the customer require from | Customer requires the following functionalities |
| the system? What does the customer need to do? | from the system such as regisitration,login, |
|  | view product, add items to cart, purchase products,payment,view offers,view delivery details,add feedback, change profile. |
|  |  |
|  |  |
| Does the customer need to read, create, | Yes. customer need to create, view and edit |
| destroy, modify or store some kind of | the data if require. |
| information in the system? |  |
|  |  |
| Could the customer work be simplified by | Yes, the system can reduce his/her |
| adding new functions to the system? | work. |
|  |  |

Registration

* Login
* View Products
* Add items to cart
* purchase product
* Payment
* View offers
* View delivery details
* Change profile
* Add review.
* View order history
* Logout

**Use case for Actor Affiliate marketer**

|  |  |
| --- | --- |
| Which functions does the affiliate marketer require from | Affiliate marketer requires the following functionalities |
| the system? What does the affiliate marketer need to do? | from the system such as login, share login, |
|  | Change profile, credit amount, |
|  |  |
|  |  |
| Does the affiliate marketer need to read, create, | Yes. affiliate marketer need to create, view and edit |
| destroy, modify or store some kind of | the data if require. |
| information in the system? |  |
|  |  |
| Could the staff work be simplified by | Yes, the system can reduce his/her |
| adding new functions to the system? | work. |
|  |  |

* Login
* share links
* Get credit amount

**Use case for Actor Delivery Boy**

|  |  |
| --- | --- |
| Which functions does the delivery require from | Delivery boy requires the following functionalities |
| the system? What does the delivery need to do? | from the system such as login, view delivery details, pick item from shop,update location, |
|  | Send confirmation,update status |
|  |  |
|  |  |
| Does the delivery boy need to read, create, | Yes. delivery boy need to create, view and edit |
| destroy, modify or store some kind of | the data if require. |
| information in the system? |  |
|  |  |
| Could the delivery boy work be simplified by | Yes, the system can reduce his/her |
| adding new functions to the system? | work. |
|  |  |

* Login
* view delivery details
* update location
* pick item from shop
* update delivery status
* Send confirmation
* change profile
* Logout

**Use case diagram for Admin**

**Admin**

**Use case diagram for customer**

**customer**

**Use case diagram for vendor**

**vendor**

**Use case diagram for Affiliate marketer**

**Affiliate marketer**

**Use case diagram for Delivery Boy**

**Delivery Boy**

**2.5.3** **ACTIVITY DIAGRAM**

The activity diagram supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario. It uses rounded rectangles to imply a specific system function, arrows to represent flow through the system, decision diamonds to depict a branching decision, and solid horizontal lines to indicate that parallel activities are occurring.

The basic purposes of activity diagrams are similar to other diagrams. It captures the dynamic behavior of the system. Other diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. So the purposes can be described as:

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.

**Activity diagram for user**

Login

Authentication

Check

Search product

view credit points

Edit profile

Logout

get discount

change password

view delivery details

Confirm order

make payment

Add to cart

Account

Cancel order

**Activity diagram for Admin**

Login

Authentication

Check

Add product

View users order

Add vendors

Add Category

change price/Qnt

Logout

modify deatils

Add delivery boy

Add affiliate marketer

Manage delivery

edit item

Accept product request

send product

**Actvity diagram for vendor**

Login

Authentication

Check

View user’s order

Search product

Edit profile

Logout

Confirm order

Change password

Edit product

Add product

Account

Cancel order

View delivery status

Request for product

View confirmation

Manage customers

**Activity diagram for affiliate marketer**

Login

Authentication

Check

Share link

Edit profile

Logout

Change password

View account

Amount credit

Get discount

**Activity diagram for delivery boy**

Login

Authentication

Check

view delivery details

d

Edit profile

Logout

Change password

view Account

pick from shop

send confirmation

Update location

update status

**2.5.4** **SEQUENCE DIAGRAM**

Sequence diagrams are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and its environment. A sequence diagram shows an interaction arranged in a time sequence. It shows the objects participating in the interaction by their life lines and the messages they exchange, arranged in a time sequence.

A sequence diagram has two dimensions: a vertical dimension represents time, horizontal dimension represents different objects. The vertical line is called the object’s lifeline. The lifeline represents the object’s existence during the interaction. This form was first popularized by Jacobson. An object is shown as a box at top of a dashed vertical line. A role is slot for an object within a collaboration that describes the type of object that may play the role and its relationships to other roles. However, a sequence diagram does not show the relationships among the roles or the association among the objects. An object role is shown as a vertical dashed line, the life line.

Each message is represented by an arrow between the life lines of two objects. The order in which these messages occur shown top to bottom on the page. Each message is labeled with the message name. The label also can include the argument and some control information and show self-delegation, a message that an object sends to itself, by sending the message arrow back to the same lifeline. The horizontal ordering of the lifelines is arbitrary. Often, all arrows are arranged to proceed in one direction across the page, but this is not always possible and the order conveys no information.

The sequence diagram is very simple and has immediate visual appeal-this is its greatest strength. A sequence diagram is an alternative way to understand the overall flow of the control of a program. Instead of looking at the code and trying to find out the overall sequence of behavior, we can use the sequence diagram to quickly understand that sequence.

**Database**

**Admin customer vendor deliveryboy**

Login Login

Login Login

Add product view products

View profile

View product Request

View product

View profile

Add item to cart

Send product view stock

Purchase items payment

Manage payment confirmation

Payment manage customers

Confirmation

Allocate delivery get notification

Add aff\_marketerView delivery details

Share link update location

Amount credit update status

Status of delivery delivery status

**2.6 SYSTEM REQUIREMENTS**

**2.6.1** **HARDWARE AND SOFTWARE REQUIREMENTS**

Hardware and software requirements for the installation and smooth functioning of this product could be configured based on the requirements needed by the component of the operating environment that works as front-end system here we suggest minimum configuration for the both hardware and software components.

Working off with this software is requirements concrete on system environments.

It includes two phases.

* Hardware Requirements
* Software Requirements

**Hardware Requirements**

The hardware required for the application depends on the following:

* Determining size and capacity requirements
* Computer evaluation and measurements
* Financial factors
* Maintenance and support.

Processor : Pentium IV or higher version (32bit/64bit) Processor

RAM : 1GB and Above

Hard Disk : 40GB and above

Keyboard : Standard Keyboard with 104 keys

Mouse : Standard Mouse with 3 buttons

Monitor : Display Panel (1366\*768)

**Software Requirements**

The software required for the application depends on the following factors:

* The flexibility of the software
* Software contracts
* Limitation of the software

Programming Language: PHP

DBMS: Mysql

Development tool: Front end-Angular

Back end –Laravel

Development Platform: Windows 10

**CHAPTER-3**

**SYSTEM DESIGN**

**3. SYSTEM DESIGN**

Design is a meaningful engineering representation of something that is to be built. It is an iterative process through which requirements are translated in to a blueprint for constructing the software. The goal of the design phase is to plan a solution of the problem specified by the requirements document.

Major activities during the design phase are:

* Data Base Design
* Architectural Design
* Interface Design
* Modular Design

**3.1. DATABASE DESIGN**

A database is collections of inter related data stored with minimum redundancy to serve many users quickly and efficiently. In database design data independence, accuracy, privacy, and security are given higher priority. Database design is an integrated approach to file design. This activity deals with the design of the physical database. All entries and attributes have been identified while creating the database. The database design deals with the grouping of data into number of tables so as to reduce the duplication of data, minimize storage space, and retrieve the data efficiently.

Guidelines for designing a database:

* Design a relational schema so that it is easy to explain its meaning. Do not combine attributed from multiple entity and relationships typeintosingle relation.
* Design the database schema so that no insertion, deletion or modification anomalies are present in the relation.
* As far as possible, avoid placing attributes in a base relation whose values may frequently be null.
* Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys in away that no spurious tuples are generated.

**Advantage**

* Ease of use
* Data independence
* Accuracy and integrity
* Avoiding inordinate delays
* Recovery from failure
* Privacy and security.

**3.1.1 E-R DIAGRAM**

An entity-relationship diagram is a data modeling technique that creates a graphical representation of the entities, and relationship between entities, within an information system.

**There are three basic elements in ER models:**

* **Entities** are the “things” about which we seek information
* **Attributes** are the data we collect about entities.
* **Relationships** provided the structure needed to drawinformation from multiple entities.

**E-R Diagram Symbols:**

Entity

Attributes

Relation

users

view

add

categories

admin

does

has

delivery boy

add

product

requestt

payment

order

vendor

add to

view

view

done for

does

cart

delivery

payment

delivery\_status

**3.1.2 TABLE DESIGN**

In the database all the information are stored in the form of tables.A table is simply a way storing data in rows and columns. In the system data is stored in many tables. The table structures are shown below with sample data.

**TABLES**

**Table 1: users**

This table stores the details of customer.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Types** | **Constraints** | **Description** |
| id | int | Primary Key | To strore the primary key |
| username | Varchar(200) | Not null | To store customer name |
| Password | Varchar(200) | Not null | To store password |
| Role\_id | Int(11) | Foreign key | To store role |
| Address | Text | Not null | To store street name |
| Email | Varchar(200) | Not null | To store email |
| Pincode | Varchar(200) | Not null | To store pincode |
| mobile | Varchar(15) | Not null | To store phone number |
| gender | enum | Not null | To store gender |
| Refer\_id | Int(11) | Not null | To store reference id |
| Affiliate\_marketer | Int (11) | Not null | To store affiliate marketer |
| Total\_credit | Int (11) | Not null | To store credit points |
| Landmark | Varchar(200) | Not null | To store landmark |

**Table 2:user\_details**

This table stores the details of vendor shop and vendor’s details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Types** | **constraints** | **Description** |
| Id | Bigint(255) | Primary key | To store the primary key |
| User\_id | Bigint(255) | Foreign key | To store the user id |
| Gender | Varchar(200) | Not null | To store the gender |
| Date\_of\_birth | Date | Not null | To store the date of birth |
| Address | Text | Not null | To store the address |
| Shop\_name | Varchar(200) | Not null | To store the shop name |
| Pan\_card\_no | Varchar(200) | Not null | To store the pan card number |
| District\_state | Varchar(200) | Not null | To store the state |
| Country | Varchar(200) | Not null | To store the country |
| Location | Varchar(200) | Not null | To store the location |
| Pincode | Varchar(200) | Not null | To store the shop pincode |

**Table 3:Products**

This table stores the details of product.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data type** | **Constraints** | **Description** |
| Id | Int(10) | Primary key | To store primary key |
| r\_point | Varchar(200) | Not null | To store referral point |
| n\_point | Varchar(200) | Not null | To store normal point |
| name | Varchar(200) | Not null | To store product name |
| Slug | Varchar(200) | Not null | To store produt slug |
| Product\_order | Varchar(200) | Not null | To store product order |
| Category\_id | Int(10) | Foreign key | To store category id |
| Sku | Varchar(200) | Not null | To store sku |
| Short\_description | Text | Not null | To store description |
| Description | Text | Not null | To store description |
| Product\_code | Varchar(200) | Not null | To store product code |
| Status | Tinyint(4) | Not null | To store status |
| In\_stock | Tinyint(4) | Not null | To store stock |
| Unit\_type | Enum | Not null | To store unit type |
| Unit | Int(255) | Not null | To store product unit |
| Is\_taxable | Tinyint(4) | Not null | To store tax |
| Tax\_amount | Int(11) | Not null | To store tax amount |
| Price | Decimal(10,2) | Not null | To store product price |
| Product\_cost | Int(255) | Not null | To store product cost |
| Meta\_title | Varchar(200) | Not null | To store meta title |
| Meta\_key | Varchar(200) | Not null | To store meta key |

**Table 4:Orders**

This table stores the order of the product.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| Id | Int(11) | Primary key | To store primary key |
| Order\_status | Int(11) | Not null | To store order status |
| User\_id | Int(11) | Not null | To store user id |
| Vendor\_id | Int(11) | Not null | To store vendor id |
| Order\_price | Decimal(10,2) | Not null | To store product price |
| Payment\_method | Varchar(200) | Not null | To store order status |
| Purchase\_price | Decimal(10,2) | Not null | To store purchase date |
| If\_cancelled | Int(11) | Not null | To store cancelled order |
| If\_deliverd | Int(11) | Not null | To store deliverd order |
| Deliverd\_date | Date | Not null | To store deliverd date |

**Table 5:referral\_address**

This table stores the details of affiliate marketer.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **constraints** | **Description** |
| Id | Int | Primary key | To store primary key |
| Name | Varchar(200) | Not null | To store the name |
| User\_id | Int(11) | Not null | To store user id |
| Email | Varchar(200) | Not null | To store email |
| Comment | Varchar(200) | Not null | To store message |

**Table 6: reference\_point**

This table stores the details of reference point.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **constraints** | **Description** |
| Id | Int(11) | Primary key | To store pimarykey |
| Refer\_user\_id | Int(11) | Not null | To refer user id |
| User\_id | Int(11) | Not null | To store the user id |
| Point | Varchar(200) | Not null | To store the credit point |
| Order\_id | Int(11) | Not null | To store the order id |

**Table 7: categories**

This table stores the product category.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| id | Int(11) | Primary key | To store primary key |
| Vender\_id | Int(11) | Not null | To store the vender id |
| Sub\_title | Varchar(200) | Not null | To store the description |
| name | Varchar(200) | Not null | To store the product name |
| slug | Varchar(200) | Not null | To store the slug |
| image | Varchar(200) | Not null | To store the product image |
| Category\_order | Int(11) | Not null | To store category order |

**Table 8: payment**

This table stores the online payment

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| Id | Int(11) | Primary key | To store primary key |
| Order\_id | Int(11) | Foreign key | To store order id |
| Payment\_status | Varchar(200) | Not null | To store payment status |
| Paid\_amount | Decimal(10,4) | Not null | To store product price |
| Mode | Varchar(200) | Not null | To store payment mode |
| Txnid | Varchar(200) | Not null | To store transaction id |
| Payment\_source | Varchar(200) | Not null | To store payment source |
| Pg\_type | Varchar(200) | Not null | To store payment type |
| Bank\_ref\_number | Varchar(200) | Not null | To store reference number |
| Bankcode | Varchar(200) | Not null | To store bank code |
| Error | Varchar(200) | Not null | To store error |
| Error message | Varchar(200) | Not null | To store error message |
| Name\_on\_card | Varchar(200) | Not null | To store name on card |
| Issuing\_bank | Varchar(200) | Not null | To store bamk name |
| Card\_type | Varchar(200) | Not null | To store card type |

**Table: 9 order\_payment**

This table stores the information of cash on delivery.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| Id | Int(11) | Primary key | To store primary key |
| Order\_id | Int(11) | Foreign key | To store order id |
| Total\_amount | Decimal(10,4) | Not null | To store amount |
| Status | Varchar(200) | Not null | To store status |
| Payment\_type | Varchar(200) | Not null | To store payment type |
| Payment\_id | Int(11) | Not null | To store payment id |
| Id\_who\_collected-billing | Int(11) | Not null | To store delivery details |
| Name\_who\_collected\_billing | Varchar(200) | Not null | To store name |
| Amount\_billing | Decimal(10,4) | Not null | To store amount |
| number\_billing | Varchar(200) | Not null | To store number of bill |
| Date\_billing | Date | Not null | To store date |

**Table: 10 product\_medias**

This table stores the product image

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| Id | Int(11) | Prinmary key | To store the primary key |
| Product\_id | Int(11) | Not null | To store product id |
| Type | Enum | Not null | To store the type of image |
| File\_path | Varchar(200) | Not null | To store file path |

**Table :11 Stock**

This table stores the stock information of product

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| id | Int(10) | Primary key | To store primary key |
| Product\_id | Int(11) | Foreign key | To store product id |
| Batch\_name | Varchar(200) | Not null | To store batch name |
| Quantity | Int(11) | Not null | To store quantity |
| Unique\_code | Varchar(200) | Not null | To store product code |
| Manufacturing\_date | date | Not null | To store manufacturing date |
| Expiry\_date | date | Not null | To store expiry date |
| In\_stock | Varchar(200) | Not null | To store in stock or not |
| status | Varchar(200) | Not null | To store stock status |

**Table:12 usercart**

This table stores the cart information.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| id | Int(11) | Primary key | To store primary kry |
| user\_id | Int(11) | Foreign key | To store user id |
| Product\_id | Int(11) | Not null | To store product id |
| Quantity | Int(11) | Not null | To store product quantity |

**Table 13: order\_status\_log**

This table stores the delivery status

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| id | Int (11) | Primary key | To store primary key |
| Order\_id | Int (11) | Foreign key | To store order id |
| Purchase\_Status | Int(11) | Not null | Tostore purchase status |
| Changed\_by | Int (11) | Not null | To store whom change |
| Comment | Varchar(200) | Not null | To store message |

**Table 14:vendorsproduct**

This table store the product of vendors.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data types** | **Constraints** | **Description** |
| id | Int(11) | Primary key | To store primary key |
| Vendor\_id | Int(11) | Not null | To store vendor id |
| productname | Varchar(200) | Not null | To store product name |
| productcategory | Varchar(200) | Not null | To store category |
| productcode | Varchar(200) | Not null | To store product code |
| quantity | Varchar(200) | Not null | To store quantity |
| mrp | Varchar(200) | Not null | To store MRP |
| productprice | Varchar(200) | Not null | To store price |
| description | Varchar(200) | Not null | To store description |
| instock | Varchar(200) | Not null | To store stock |
| Is\_taxable | Varchar(200) | Not null | To store tax |
| Taxtype\_id | Int(11) | Not null | To store tax type |
| Tax\_amount | Varchar(200) | Not null | To store tax amount |

**3.2 ARCHITECTURAL DESIGN**

The architectural design develops a modular program structure and represents the control relationships between modules. It also defines interfaces that enable data to flow throughout the program.

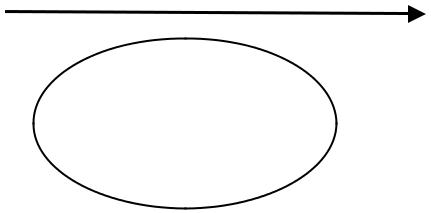
**3.2.1** **DATA FLOW DIAGRAM**

A data flow diagram is a graphical technique that depicts data flow and transforms that are applied as data move from input to output. The DFD is used to represent increasing information flow and functional details. A Level 0 DFD also called a fundamental system model or context model represents the entire software elements as a single bubble with input and output indicated by incoming and outgoing arrows respectively. Additional process and information flow parts are represented in next level i.e., Level 1 DFD. Each of the processes represented at level 1 are sub functions of overall system depicted in the context model.

**Data flow diagram symbol:**

Source/ Destination of data

Data flow



Process



Storage

**Level 0: Context Level**

EATSWEETS

0

Request

Requestt

Responset

Responset

database

Users

**Level 1**

admin process

0.1

Login

Admin

username &password

customer process0.2

0.2

Login

Customer

Vendor

Delivery boy

Affiliate marketer

vendor process

0.3

Data base

Login

affiliate marketer process 0.4

0 00ksw

0

0.0

00

Login

username &password

delivery process 0.5

Login

**level 2:Admin**

Add vendors 0.1.1

vendor\_id

vendors

Admin

users

vendor\_id

view vendors 0.1.2

vendors

delivery boy

add delivery boy0.1.3

delivery status

view delivery statuus0.1.4

status\_id

dynamicstatus delivey\_status

vendors order

view vendors request0.1.5

vorder\_id

request\_form

affiliate marketer

add affiliate marketer0.1.6

aid

users

product details

add product0.1.7

prd\_id

view product

products stock

prd-id

confirm orders

view customer0.1.10

confirm order 0.1.9

view product0.1.8

view customers

users

**level 2:customer**

customer details

Registration 0.2.1

customer

cid

users

product details

search product 0.2.2

prd\_id

products

add to cart

cart\_id

select item 0.2.3

usercart

payment details

payment\_id

payment

0.2.4

payment dynamic status

delivery status

review\_id

view delivery status 0.2.5

**level 2:vendor**

prd\_id

view product 0.3.1

vendor

products

order details

order\_id

product order 0.3.2

request\_form

customers

cid

view customers 0.3.3

users

payment details

pid

payment 0.3.4

payment dynamic status

view order status 0.3.5

status\_id

order status

product details

**level 2:affiliate marketer**

view product

0.4.1

affiliate marketer

prd\_id

products

link details

aid

share link

0.4.2

referral\_addres

credit details

aid

view credit amount 0.4.3

0.4.3

referral\_point

delivery details

**level 2:delivery boy**

view

delivery details 0.5.1

0.5.

delivery boy

delivery\_id

orders

status\_id

delivery status

Add delivery status 0.5.2

dynamic status

view status

status\_id

view

status 0.5.3

**3.2.2** **HIERARCHICAL DIAGRAM**

The hierarchical diagram is a technique for representing the modules of a system as a hierarchy and for documenting each module. It was used to develop requirements, construct the design, and support implementation of an expert system to demonstrate and verify the system. Structure charts can be used to display several types of information.

LEZAT

customer

affiliate marketer

delivery boy

vendor

Admin

view user order

register

view products

share link

view orders

pick from shop

add aaffiliate marketer

view points

confirm order

request product

login

Add delivery boy

Add product category

view users

Add vendors

1

role &permission

view approval

purchase

update status

Add product

view users order

role &permission

role &permission

view delivery status

view vendor’s order

add vendor product

add to cart

view stock

confirm order

checkout

view delivery details

edit stock

**3.3 INTERFACE DESIGN**

An interface design elements for the software tell how information flows into and out of the system and how it is communicated among the components as part of the architecture.

**3.3.1** **INPUT DESIGN**

Input design is the link between the information system and users and those steps that are necessary to put transaction data into a usable form for processing data entry. Instructing the computer to read data from a written printed document can active the activity of putting data into the computer for processing or it can occur by keying data directly into the system. The design of input focusing on controlling the errors, avoid delay, and keeping the process simple. System analyst decides the following input design details.

* What data to input?
* What medium to use?
* How the data is arranged and coded?

In my project named Lezats, I tried to include the following design constrains provided in the software engineering.

**1: Avoid scattering of fields in the forms**

In all forms of the software the textboxes (which provided to input some data), label (which label the text boxes), combo box (list a set of values) etc all are arranged in a neat and well format. It provides a simple look to the pages. The buttons are placed at the bottom of the page and easily accessible to the user. The menus are arranged below the heading and at a minimum level of menus are arranged with pages. Menu provides the continuity to the pages.

**2: User only needs to enter a minimum amount of data**

All forms contain a minimum amount data, but most essentials. No page provides or wanted bulky of data. It provides more easiness to the user. It creates more the software to the end user. Also the operation continues by single click.

**3: Avoid confusion in the forms**

All forms have a well defined menus and each menu name indicate its purpose. So the user can easily access various forms without confusion. Each form and its sub forms are well labeled. So the user can easily identify the forms and work on that.

**The following are the input forms present in this project:**

* Login form
* Add vendor details form
* Add delivery boy details form
* Add product details form

**3.3.2** **OUTPUT DESIGN**

Designing computer should proceed in well thought out manner. The term output means any information produced by the information system weather printed or displayed. Output design is a process that involves designing necessary output that have to be used by various users according to requirement. The efficient intelligent output design should remove the system relationship with the users and help in decision making.

When designing the output, system analyst must accomplish the following:

* Determine the information present
* Decide whether to print, display the information and select output medium
* Arrange information in acceptable format.

In my project, the outputs are in the form of reports. They are well format and it provides the output in a correct and neat format.

**The following are the output forms present in this project:**

* Form for viewing customers details
* Form for viewing product details
* Form for viewing stock report details
* Form for viewing payment report details
* Form for viewing customer log

**3.4 PROCEDURAL DESIGN**

The procedural design determines the modules included in the whole project which help us to identify the major functions.

**MODULE SPECIFICATIONS**

The following are the modules in this application.

* Admin
* Vendor
* Delivery boy
* Customer
* Affiliate marketer

**Admin Module**

Admin is one who manages the entire working of a system. Following are the main functions that can be performed by admin:

* Add details : add product, vendor and delivery details, add affiliate marketer
* View details: view customer details, view vendor details, view delivery boy details, view product details, view product order details, view delivery details, view review.
* Update details: update vendor details, delivery boy details, product details, and product order details .
* Delete details: delete vendor details, product details, and product order details.

**customer**

Following are the main functions that can be performed by customer:

* Add details: add item to cart, payment.
* View details: view product details, view product category, view delivery details, and view order details.
* Update details: update profile.
* Delete details: delete item from cart.

**vendor**

Following are the main functions that can be performed by stock\_in\_charge

* View details: view product details, view product order details, view delivery details.
* Update details: update stock details.

**Affiliate marketer**

Following are the main functions that can be performed by affiliate marketer:

* View details: view status, view credit amount.
* Update details: share link

**Delivery Boy**

Following are the main functions that can be performed by delivery boy:

* View details: view delivery details.
* Update details: update delivery status.

**CHAPTER 4**

**CODING**

****

**4** **CODING**

**4.1 ABOUT THE SOFTWARE TOOLS USED**

**4.1.1 ANGULAR**

Angular is a JavaScript framework which makes you able to create reactive **Single Page Applications** (SPAs). This is a leading front-end development framework which is regularly updated by Angular team of Google. Angular 7 is completely based on components. It consists of several components forming a tree structure with parent and child components.

**4.1.2 LARAVEL**

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.

Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like CodeIgniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development.

If you are familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It saves a lot time if you are planning to develop a website from scratch. Moreover, a website built in Laravel is secure and prevents several web attacks.

**Advantages of Laravel**

Laravel offers you the following advantages, when you are designing a web application based on it −

* The web application becomes more scalable, owing to the Laravel framework.
* Considerable time is saved in designing the web application, since Laravel reuses the components from other framework in developing web application.
* It includes namespaces and interfaces, thus helps to organize and manage resources.

## Features of Laravel

Laravel offers the following key features which makes it an ideal choice for designing web applications −

### Modularity

Laravel provides 20 built in libraries and modules which helps in enhancement of the application. Every module is integrated with Composer dependency manager which eases updates.

### Testability

Laravel includes features and helpers which helps in testing through various test cases. This feature helps in maintaining the code as per the requirements.

### Routing

Laravel provides a flexible approach to the user to define routes in the web application. Routing helps to scale the application in a better way and increases its performance.

### Configuration Management

A web application designed in Laravel will be running on different environments, which means that there will be a constant change in its configuration. Laravel provides a consistent approach to handle the configuration in an efficient way.

### Query Builder and ORM

Laravel incorporates a query builder which helps in querying databases using various simple chain methods. It provides **ORM** (Object Relational Mapper) and **ActiveRecord** implementation called Eloquent.

### Schema Builder

Schema Builder maintains the database definitions and schema in PHP code. It also maintains a track of changes with respect to database migrations.

### Template Engine

Laravel uses the **Blade Template** engine, a lightweight template language used to design hierarchical blocks and layouts with predefined blocks that include dynamic content.

### E-mail

Laravel includes a **mail** class which helps in sending mail with rich content and attachments from the web application.

### Authentication

User authentication is a common feature in web applications. Laravel eases designing authentication as it includes features such as **register, forgot password** and **send password reminders**.

### Redis

Laravel uses **Redis** to connect to an existing session and general-purpose cache. Redis interacts with session directly.

### Queues

Laravel includes queue services like emailing large number of users or a specified **Cron** job. These queues help in completing tasks in an easier manner without waiting for the previous task to be completed.

### Event and Command Bus

Laravel 5.1 includes **Command Bus** which helps in executing commands and dispatch events in a simple way. The commands in Laravel act as per the application’s lifecycle.

**4.1.3** **MySQL**

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

**MySQL is a database management system.**

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

**MySQL databases are relational.**

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist

1. MySQL software is Open Source.

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations.

2. The MySQL Database Server is very fast, reliable, scalable, and easy to use.

MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, network together.

**4.2 CODING PRINCIPLES**

The input to the coding phase is the design document. During coding phase, modules identified in the design document are coded according to the module specification. Objectives of coding phase are, to transform design into code and unit test the code.

**4.2.1** **Coding Guidelines**

* Code should be easy to understand.
* Don’t take pride in cryptic code.
* Code should be well documented.
* Comments should be present.
* Functions should be small.
* Do not use Go-to statement.
* Do not use the same variable for multiple purposes.

**4.3 SAMPLE CODE**

**CHAPTER 5**

**TESTING**

**5** **SYSTEM TESTING**

For software that is newly developed, primary importance is given to testing the system. It is the last opportunity for the developer to detect the possible errors in the software before handing over it to the customer. Testing is the processes by which the developer will generate a set of data, which gives the maximum probability of finding all types of errors that can occur in the software.

The various steps in testing the system can be listed as below:

1. Running the program to identify any errors that might have occurred while feeding the program into the system.
2. Applying the screen formats to regulate users to extend, so that the screens are comprehensible to the user.
3. Presenting the formats to the administration for the purpose of obtaining approval and checking if any modification has to be done. Obtaining feedbacks from users and analyzing the scope for improvement.
4. Checking the data accessibility from the data server and whether any I mprovement is needed or not.

Testing is a methodology for evaluating the project. The good test has a high probability of finding an error. Testing is generally two types- Black box testing and White box testing.

* Unit Testing
* Integration Testing
* System Testing
* Validation Testing

**5.1 UNIT TESTING**

Unit testing is carried out to screen wise, each screen being identified as an object.

Attention is diverted to individual modules, independently to one another to locate in

coding and logic.

In unit testing,

* Module interface is tested to ensure that information properly flows into and out of the program under test.
* Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in algorithm execution.
* Boundary condition is tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
* All independent paths through the control structures are executed to ensure that all statements in the module have been executed at least once.
* Error handling paths are also tested.

**TEST CASES**

**Login form**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
|  |  |  |  |  |
| **1** | Enter wrong user | Display login form again | Message | Pass |
|  | name and pass word | with a warning message | displayed. |  |
|  |  |  |  |  |
| **2** | Enter correct user | Display login form again | Message | Pass |
|  | name and wrong | with a warning message | displayed. |  |
|  | password |  |  |  |
|  |  |  |  |  |
| **3** | Enter correct user | Users can login into the | Appropriate home | Pass |
|  | name and password | system | page is displayed |  |
|  |  |  |  |  |
| **4** | Press login button | Display a warning | Warning message | Pass |
|  | without filling the | message to fill the fields. | is displayed. |  |
|  | user name and |  |  |  |
|  | password. |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Form displayed. | Display the | Form loaded | Pass |
|  |  | registration form. |  |  |
|  |  |  |  |  |
| 2 | Enter the name in | Display an invalid | Invalid message | Pass |
|  | integers. | message. | displayed |  |
|  |  |  |  |  |
| 3 | Enter the mobile | Display an invalid | Invalid message | Pass |
|  | number in characters. | message. | displayed. |  |
|  |  |  |  |  |
| 4 | Enter the mobile | Display an invalid | Invalid message |  |
|  | number more than | message. | displayed. | Pass |
|  | and less than 10 |  |  |  |
|  | integers. |  |  |  |
|  |  |  |  |  |
| 5 | Click the save button | Display a warning | Warning message | Pass |
|  | without filling the | message to fill the | displayed. |  |
|  | details | details. |  |  |
|  |  |  |  |  |
| 6 | Click on save button | Accept the details. | Registration | Pass |
|  | with filled fields. |  | successfully done. |  |
|  |  |  |  |  |
| 7 | Click cancel button | Clear all fields to | All fields cleared. | Pass |
|  |  | blank |  |  |
|  |  |  |  |  |

**Add product details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
|  |
|  |  |  |  |  |
| 1 | Form displayed. | Display the | Form loaded | Pass |
|  |  | registration form. |  |  |
|  |  |  |  |  |
| 2 | Click the save button | Display a warning | Warning message | Pass |
|  | without filling the | message to fill the | displayed. |  |
|  | details | details. |  |  |
|  |  |  |  |  |
| 3 | Click on save button | Accept the details. | Registration | Pass |
|  | with filled fields. |  | successfully done. |  |
|  |  |  |  |  |

**View product details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
|  |
|  |  |  |  |  |
| 1 | Form displayed. | Display form with | Form loaded with | Pass |
|  |  | all controls. | all controls. |  |
|  |  |  |  |  |
| 2 | Click on the button. | View form displayed | Result displayed | Pass. |
|  |  | with valuable details | correctly. |  |
|  |  |  |  |  |

**5.2 INTEGRATION TESTING**

Integration testing is a symmetric technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. Unit tested module were taken and a single program structure was built that has been dictated by and tested in small segments, where errors were easy to locate and rectify. Each database or table manipulation operation was written as single program was tested again with numerous test data to check for its functionality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Input/procedure** | **Expected Result** | **Actual** | **Pass/Fail** |
|  |  |  | **Result** |  |
|  |  |  |  |  |
| 1. | Check the value pass between | Appropriate | Same as | Pass |
|  | different forms are | operations of | expected. |  |
|  | appropriate format | different forms. |  |  |
|  |  |  |  |  |

**5.3 SYSTEM TESTING**

System testing is used test the entire system (Integration of the all modules). It also tests to find the discrepancies between the system and the original objective, current specification and system documentation. The entire system is checked to correct deviation to achieve correctness

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Input/procedure** | **Expected** | **Actual** | **Pass/Fail** |
|  |  | **Result** | **Result** |  |
|  |  |  |  |  |
| 1. | Check whether indented output | All operations | Same as | Pass |
|  | is obtained. | are carried out | expected. |  |
|  |  | properly. |  |  |
|  |  |  |  |  |



**5.4 VALIDATION TESTING**

At the conclusion of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins validation test has been conducted one of the two possible conditions exists. One is the function or performance characteristics confirm to specification and are accepted and the other is deviation from specification is uncovered and a deficiency list is created.

**CHAPTER 6**

**IMPLEMENTATION**

****

**6 SYSTEM IMPLEMENTATION**

System implementation is the stage where the theoretical design is turned into a working system. The system can be implemented only after through testing is done and if it if found to work according to specifications. The following methods were undergone.

* Testing developed programs with updating.
* Correction of errors identified.
* Creating the tables of the system with actual data.
* Making necessary changes with actual data.
* Doing a parallel run of the system to find out any errors identified and to correct them.
* Training of user personnel’s.

The implementation method used to implement Queue Management is Parallel Run. That is, the new system will work parallel to the existing system. The new system will replace the existing system completely.

The implementation stage involves following tasks.

* Careful planning.
* Investigation of the current system and constraints.
* Design of methods to achieve the changeover.
* Training of the staff in the changeover phase.
* Evaluation of the changeover method

Technologies used in the development of the software are:

* Development tool : notepad++, XAMPP control panel
* Language : PHP
* Database : MySQL
* Web Server : Mozilla Firefox
* Scripting : HTML, JavaScript, CSS
* Operating System : Windows 7

**CHAPTER 7**

**CONCLUSION**

****

**7 CONCLUSION**

To conclude the description about the project the project developed using PHP with and MySQL is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement.

**CHAPTER 8**

**REFERENCES**

1. **REFERENCES 8.1 WEBSITES**

[http://www.w3schools.com](http://www.w3schools.com/)

[http://www.stackoverflow.com](http://www.stackoverflow.com/)

[http://www.codeproject.com](http://www.codeproject.com/)

[https://youtube.com](https://youtube.com/)

**8.2 REFERENCE BOOKS**

* Software Engineering, ROGER. S. PRESSMAN, Tata McGraw Hill, Fifth Edition, Year 2004.
* Database System Concepts, Abraham Silberschatz, Henry F Korth, S Sudarshan, Sixth edition, Year 2011

****

**APPENDIX**