**SYSTEMS ANALYSIS AND DESIGN**

**4.0 INTRODUCTION**

System analysis and design deal with planning the development of information systems through understanding and specifying in detail what a system should do and how the components of the system should be implemented and work together. System analysis and design solve business problems through analyzing the requirements of information systems and designing such systems by applying analysis and design techniques.

System analysis and design is the most essential phase in the development of a system since the logical system design arrived at as a result of systems analysis which is in turn converted into physical system design.

This chapter will outline the general system and application architecture, requirements specification, functional and non-functional requirements, use case modelling, the database design, Entity Relationship diagram, Normalization, and User interface design.

**4.1 GENERAL SYSTEM ARCHITECTURE**

The architecture of applications is usually broken into logical chunks called "tiers", where every tier is assigned a role. A “tier” can also be referred to as a “layer”. There are three layers involved in the application namely **Presentation Layer**, **Business Layer** and **Data Layer**. Each layer is explained in detailed below:

* **Presentation Layer:**

It is also known as Client layer. Top most layer of an application. This is the layer we see when we use a software. By using this layer, we can access the webpages. The main functionality of this layer is to communicate with Application layer. This layer passes the information which is given by the user in terms of keyboard actions, mouse clicks to the Application Layer. For example, login page of Gmail where an end user could see text boxes and buttons to enter user id, password and to click on sign-in.

In simple words, it is to view the application.

* **Application Layer:**

It is also known as Business Logic Layer which is also known as logical layer. As per the Gmail login page example, once user clicks on the login button, Application layer interacts with Database layer and sends required information to the Presentation layer. It controls an application’s functionality by performing detailed processing. This layer acts as a mediator between the Presentation and the Database layer. Complete business logic will be written in this layer.

In simple words, it is to perform operations on the application.

* **Data Layer:**

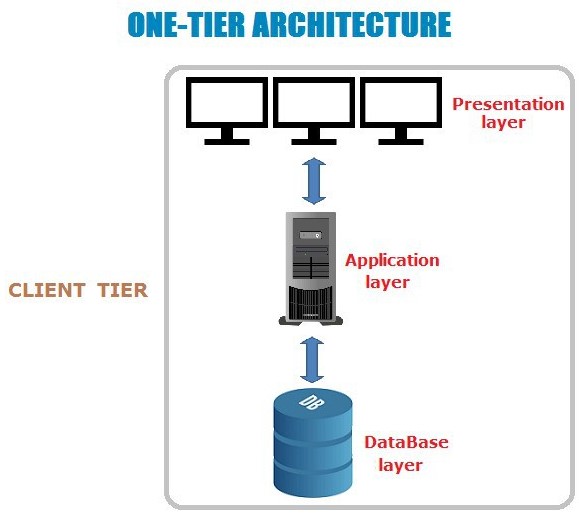
The data is stored in this layer. Application layer communicates with Database layer to retrieve the data. It contains methods that connects the database and performs required action e.g.: insert, update, delete etc.

In simple words, it is to share and retrieve the data.

There are four categories of application architectures which all have the four layers explained above. The categories of the application architecture are:

4.1.1 One Tier Architecture

One tier architecture has all the layers such as Presentation, Business, Data Access layers in a single software package. All traditional applications consist only of 1 tier, which resides on the client machine. Applications which handles all the three tiers such as MP3 player, MS Office are come under one tier application. The data is stored in the local system or a shared drive. This tier is also known as **Standalone application.**

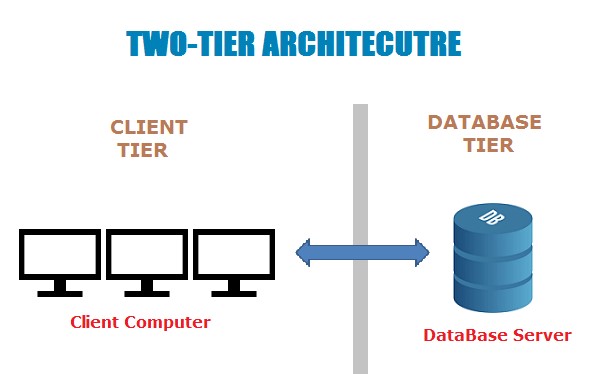


*Figure 4.0 One-Tier Architecture (Standalone Application)*

*Source:* (Gaur, 2018)

4.1.2 Two Tier Architecture

The Two-tier architecture is divided into two parts namely **Client Application (Client Tier)** and **Database (Data Tier).** Client system handles both Presentation and Application layers and Server system handles Database layer. The communication takes place between the Client and the Server. Client system sends the request to the Server system and the Server system processes the request and sends back the data to the Client System. It is also known as **client-server application**.

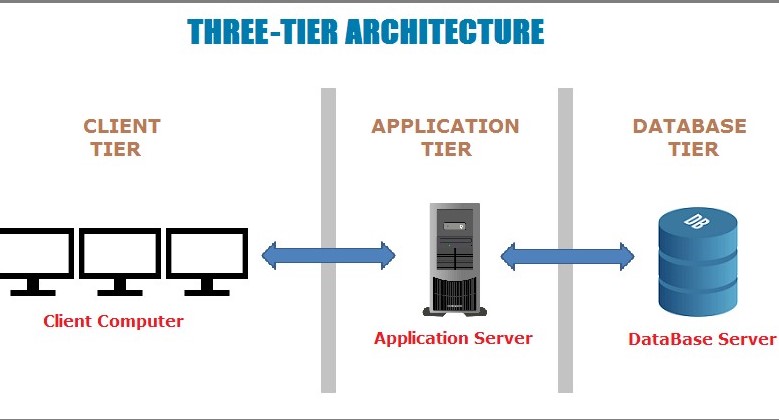


*Figure 4.1 Two-tier Architecture (Client-Server Application)*

*Source:* (Gaur, 2018)

4.1.3 Three Tier Architecture

The Three-tier architecture is divided into three parts namely: **Presentation layer (Client Tier),** **Application layer (Business Tier)** and **Database layer (Data Tier)**. Client system handles Presentation layer, Application server handles Application layer and Server system handles Database layer. A web browser is the first tier (presentation), an engine using some dynamic Web content technology (such as ASP, CGI, ColdFusion, Dart, JSP/Java, Node.js, PHP, Python or Ruby on Rails) is the middle tier (application logic), and a database is the third tier (storage). The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface. It is also known as **Web Based application.**

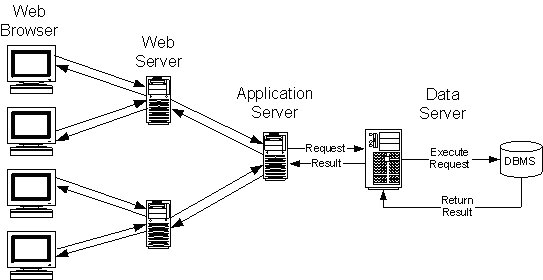


*Figure 4.2 Three-Tier Architecture (Web-Based Application)*

*Source:* (Gaur, 2018)

4.1.4 N-Tier Architecture

It is similar to three tier architectures but number of application servers are increased and represented in individual tiers in order to distributed the business logic so that the logic will be distributed. N-Tier application is also known as **Distributed application**.



*Figure 4.3 N-Tier Application (Distributed Application)*

*Source:* (Kramek, 2018)

Web applications lend themselves to an n-tiered approach by nature. Though many variations are possible, the most common structure is the three-tiered application. This project uses the three-tiered architecture because it is made up of three layers; the **presentation layer** would be the **web browser**; the **application layer** would be the **Django framework** and the **third layer** which is the **database layer** would be **Postgresql**.

**4.2 GENERAL APPLICATION ARCHITECTURE**

**4.2.1 The Model-View-Controller Design Pattern**

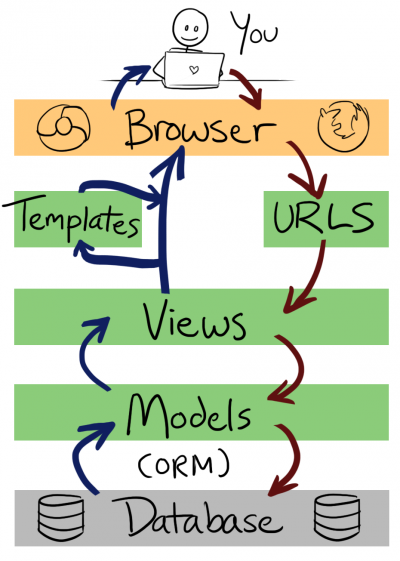
Although originally developed for desktop computing, MVC has been widely adopted as an architecture for World Wide Web applications in major programming languages. Several web frameworks have been created that enforce the pattern. These software frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server. MVC is short for **Model, View, and Controller.** MVC is a popular way of organizing code. The big idea behind MVC is that each section of code has a purpose, and those purposes are different. Some of the code holds the data of an app, some of the code makes the app look nice, and some of the code controls how the app functions. MVC is a way to organize code’s core functions into their own, neatly organized boxes. This makes thinking about the app, revisiting the app, and sharing the app with others much easier and cleaner.

MVC has been around as a concept for a long time but has seen exponential growth since the advent of the Internet because it is the best way to design client-server applications.

* The **model(M)** is a model or representation of your data. It’s not the actual data, but an interface to the data. The model allows you to pull data from your database without knowing the intricacies of the underlying database. The model usually also provides an abstraction layer with your database, so that you can use the same model with multiple databases.
* The **view(V)** is what you see. It’s the presentation layer for your model. On your computer, the view is what you see in the browser for a Web app, or the UI for a desktop app. The view also provides an interface to collect user input.
* The **controller(C)** controls the flow of information between the model and the view. It uses programmed logic to decide what information is pulled from the database via the model and what information is passed to the view. It also gets information from the user via the view and implements business logic: either by changing the view, or modifying data through the model, or both.

This application follows the MVC pattern closely, however it does use its own logic in the implementation because the Django framework uses a different logic. Because the “C” is handled by the framework itself and also a lot happens in the Django models, templates and views, Django is often referred to as an **MTV framework.** In the MTV development pattern:

* **M stands for “Model,”** the data access layer. This layer contains anything and everything about the data: how to access it, how to validate it, which behaviors it has, and the relationships between the data.
* **T stands for “Template,”** the presentation layer. This layer contains presentation-related decisions: how something should be displayed on a Web page or other type of document.
* **V stands for “View,”** the business logic layer. This layer contains the logic that accesses the model and defers to the appropriate template(s). You can think of it as the bridge between models and templates.



*Figure 4.4 E-Shopper Application Architecture (MVT Application Design Patten)*

*Source:* (Openhatch, 2018)

**4.3 REQUIREMENT SPECIFICATION**

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behavior of a system or software application. It includes a variety of elements that attempts to define the intended functionality required by the customer to satisfy their different users. What the software will do and how it will be expected to function is fully described under requirements specification. In addition to specifying how the system should behave, the specification also defines at a high-level the main business processes that will be supported, what simplifying assumptions have been made and what key performance parameters will need to be met by the system. It outlines functional and non- functional requirements and may include a set of use cases that describe user interactions that the system must provide. Requirements specification allows a thorough analysis of requirements before design can start hence, reducing later redesign.

**4.3.1 FUNCTIONAL REQUIREMENTS**

Functional requirements describe in detail a system’s intended capabilities, appearance and interactions with users. It usually consists of a hierarchical organization of requirements, with the business/functional requirements at the highest-level and the detailed system requirements listed as their child items. The functional requirements for a system usually involves the user interface and describe each of the possible user input actions and the system’s response actions. Functional requirements therefore specify particular results of a system and it drives the application architecture of a system. Generally, the requirements are written as statements such as "System needs the ability to do x" with supporting detail and information included as necessary. It serves as a kind of guideline and continuing reference point as the system is being developed.

The system will have the following functional requirements:

* An administrator will be able to login and customize the system to the needs of the restaurant.
* Through the administrator’s panel, the system allows the manager to generate reports of orders, delete products and orders, and approve comments among others.
* The administrator can create multiple accounts for different users to perform specific actions like accepting orders, managing product stock etc.
* The administrator can check product inventory, accept orders and approve product reviews.
* Online Shoppers will be able to create an account so as to login to their portal and see their Order history etc.
* Online Shoppers will be able to place orders and receive notifications of their others in their emails.
* Online Shoppers can choose the language of their choice.
* Online Shoppers can review products with comments.
* Online Shoppers can also pay online before their orders are placed successfully

**4.3.2 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements essentially specify how a system should behave or work and also are a constraint upon the system’s behavior. They specify criteria that judge the operation of a system rather than specific behaviors. They also describe various attributes which affect the functionality’s effectiveness. It specifies the requirements that essentially embody the technical environment that the product needs to operate in and include the technical constraints that it needs to operate under. These technical requirements are critical in determining how the higher-level functional requirements will get decomposed into the more specific system requirements. Non-functional requirements do not alter a system’s functionality that is, the functional requirements remain the same regardless of the attributes attached to them. Non-functional requirements make up a significant part of the specification in that, users or clients may judge a system based on its non-functional requirements.

Below are the non-functional requirements our system will achieve:

* **Performance**

The system will be an interactive one hence the delays involved will be less in that there are no immediate delays in every action-response of the system.

* **Reliability**

As the system is meant for online shopping, the system will be reliable in that it will consistently perform according to its specification so as to enable users make place orders as and when they want to without any system failure.

* **Maintainability**

A system should be developed in such a way that it can evolve to meet the changing needs of a customer. The system will be developed in such a way that it can be customized to meet a particular vendor’s needs or preference.

* **Ease of use**

The system will ensure ease of use in that little training time will required to know how to use the system. The user interface will also be friendly so users will have ease when using the system.

* **Scalability**

It is the ability of a system to continue to function well when it is changed in size or volume to meet a user need. The system will be developed to fit onto any device such as a smartphone, laptop, tablet or desktop.

* **Security**

It is a system attribute that reflects the ability of a system to protect itself against external attacks, which may be deliberate or accidental. The system will be developed in such a way that proper login mechanism would be put in place to protect user accounts.

* **Safety**

The system will ensure safety so that information about orders placed is securely transmitted to the server without any changes in the information. PayPal Payment System is recognized internationally for it secure environment for online secured payments.

* **Hardware Requirements**

The hardware requirements are classified into server and client requirements which are detailed below:

* **Server Requirements**

Minimum CPU – Intel® processor Pentium IV minimum

Minimum Disk Space – 80GB

Minimum Memory – 6GB

* **Client Requirements**

Minimum CPU – Intel® processor Pentium IV 1GHz

Minimum Disk Space – 20GB

Minimum Memory – 2GB

* **Software Requirements**

The software requirement for the application runs best on a system with the following capabilities:

* Any operating system such as Windows, Linux, Macintosh, and Mozilla Firefox operating system
* A web browser such as Mozilla Firefox, internet explorer, opera, Google chrome etc.

**4.4 USE CASE MODELLING**

UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. A use-case model is a model of how different types of users interact with the system to solve a problem. It therefore describes the goals of the users, the interactions between the users and the system and the required behavior in satisfying these goals.

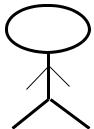
**4.4.1 USE CASE DIAGRAMS**

A use-case diagram describes a system's functional requirements in terms of use cases. It is a model of the system's intended functionality (use cases) and its environment (actors). Use cases enable you to relate what you need from a system to how the system delivers on those needs.

Think of a use-case model as a menu, much like the menu you'd find in a restaurant. By looking at the menu, you know what's available to you, the individual dishes as well as their prices. You also know what kind of cuisine the restaurant serves: Italian, Mexican, Chinese, and so on. By looking at the menu, you get an overall impression of the dining experience that awaits you in that restaurant. The menu, in effect, "models" the restaurant's behavior. It shows a subset of the model elements relevant for a particular purpose. Below is a table of use case symbols and their representations.

|  |  |
| --- | --- |
| SYMBOL | TERM AND REPRESENTATION |

**Actor:** An actor represents roles for users of a



system, including human users and other

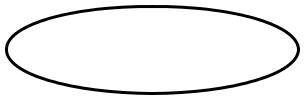
systems. An actor is external to a system.

**Association:** An association correspond to a

sequence of actions between the actors and use

case in achieving the use case.

**Use case:** A use case represents a user goal that



can be achieved by accessing the system.

**System Boundary:** The system boundary

defines the confines or scope of a system

within which the use cases are placed

*Table 4.0 Use case symbols and their meanings.*

The actors involved with the system are:

**Customer:** This actor is one of the main beneficiary of the system. The customer can order productsusing the system.

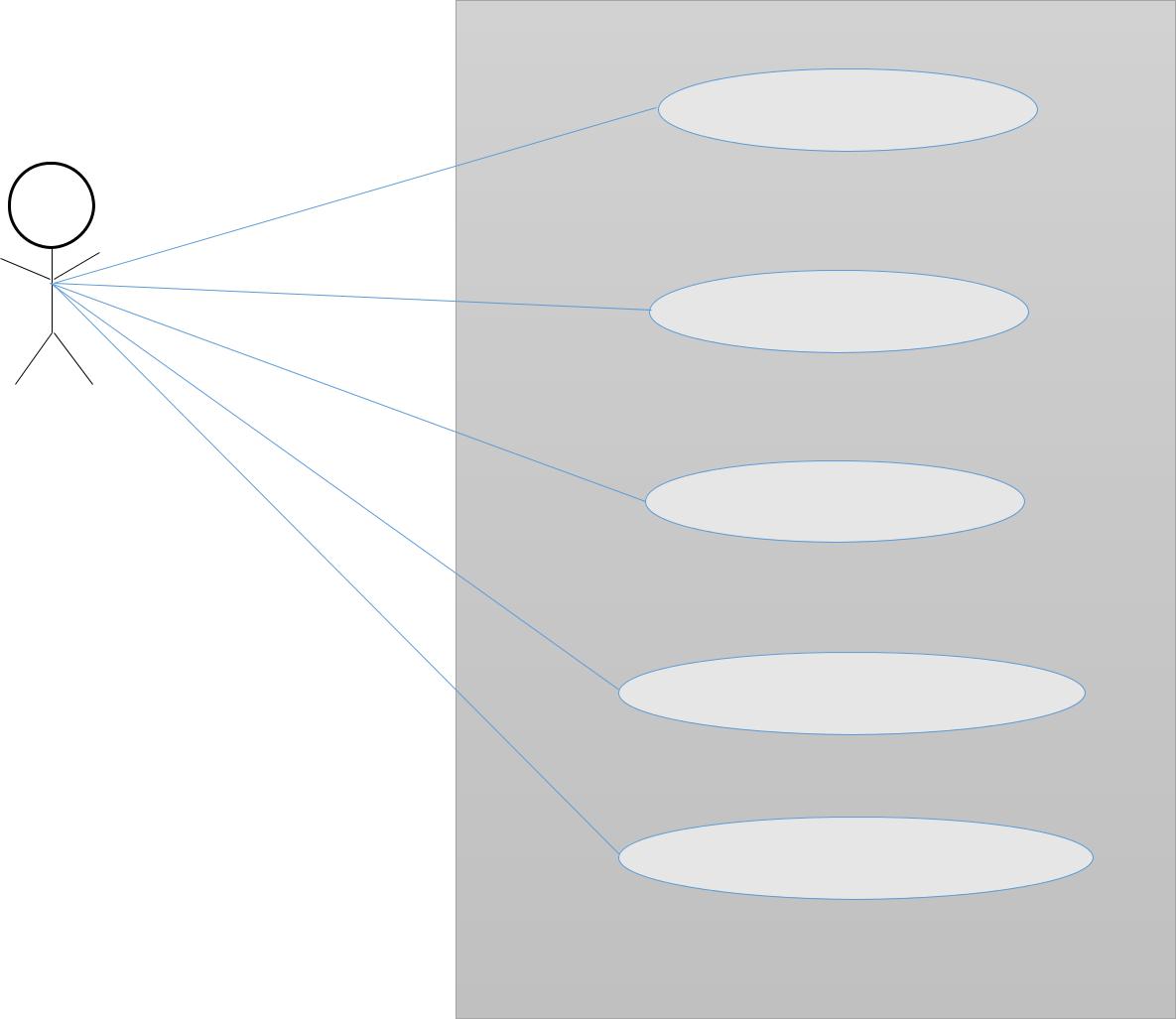
**Manager:** This actor is also another beneficiary of the system. Customizing the system to hispreference, editing products, approving orders and product reviews are some of the activities of this actor.

**System:** This actor is responsible for generating reports and sending confirmation messagesamong others. It however requires the administrator inputs to generate the reports.

Below are the use case diagrams for the system together with their descriptions:

****

*Figure 4.5 Use case diagram for customer*



**Login**

**Customize site**

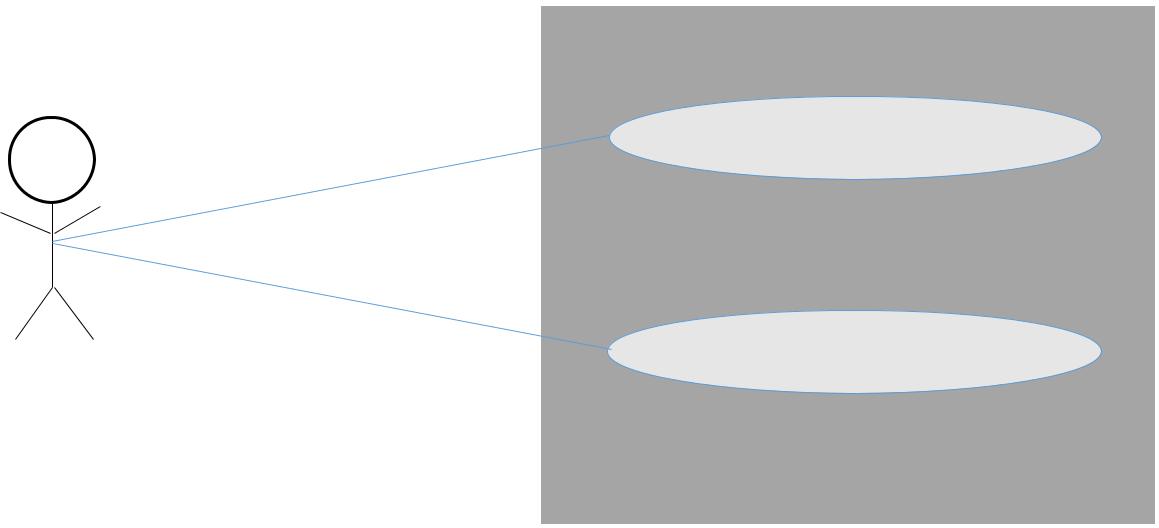
**Shop Owner**

**Edit Categories**

**Edit Products**

**Edit Delivery location**

*Figure 4.6 Use case diagram for Shop Owner*



**Generate Reports**

**Send Notifications (SMS/Email)**

**System about orders**

*Figure 4.7 Use case diagram for system*

**Application Use Case Scenarios**

*Table 4.1 Customer Registration*

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register** | |
|  |  | |
| Actor | Customer | |
|  |  | |
| Description | Enables new customers to create an account by | |
|  | entering their credentials. | |
|  |  |  |
| Steps Performed | 1. | Click on sign up |
|  | 2. | Enter personal details and password. |
|  | 3. | Click on sign up button. |
|  |  |  |

*Table 4.2 Customer login*

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Login** |  |
|  |  | |
| Actor | Customer | |
|  |  | |
| Precondition | Customer should have already registered. | |
|  |  | |
| Description | Enables existing customers to have access to | |
|  | the system’s functionality using valid | |
|  | credentials. | |
|  |  |  |
| Steps Performed | 1. | Click on login |
|  | 2. | Enter username and password. |
|  | 3. | Click on login. |
|  |  |  |

*Table 4.3 View Products*

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | **View Products** | | |
|  |  | |  |
| Actor | Customer | | |
|  |  | |  |
| Description | Enables a customer to view products available | | |
|  | together with their prices. | | |
|  |  | |  |
| Steps Performed | Go to the website of the using a web | | |
|  | browser. | | |
|  |  | | |
|  |  |  |  |
|  |  |  |  |
| *Table 4.4 Placing Orders* |  | |  |
| **Use Case Name** | **Placing an Order** | |  |
|  |  | |  |
| Actor | Customer | |  |
|  |  | |  |
| Precondition | Customer can login or choose not to login. | |  |
|  |  | |  |
| Description | Enables customers to order products which is | |  |
|  | successful after they have proceeded with the | |  |
|  | payment. | |  |
|  |  |  |  |
| Steps Performed | 1. | Enter quantity |  |
|  | 2. | Click on add to cart button to add |  |
|  |  | product to cart. |  |
|  | 3. | Click on Proceed to checkout. |  |
|  | 4. | Enter confirmation and payment details |  |
|  |  | and click on Place now. |  |
|  | 5. | Proceed with payment using either |  |
|  |  | debit card. |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| *Table 4.5 Manager Login* |  |  |
| **Use Case Name** | **Login** |  |
|  |  | |
| Actor | Manager | |
|  |  | |
| Description | Enables a manager to access the system’s | |
|  | functionality available to perform managerial | |
|  | duties. |  |
|  |  |  |
| Steps Performed | 1. | Proceed to admin Panel. |
|  | 2. | Enter username and password. |
|  | 3. | Click on login. |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Table 4.6 Editing Products* |  | |
| **Use Case Name** | **Edit Products** | |
|  |  | |
| Actor | Manager | |
|  |  | |
| Description | Enables a manager to make changes to available products such as adding new product, deleting product not available as well as upload images for the various products. | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
| Precondition | Manager should login. | |
|  |  |  |
| Steps Performed | 1. | Click on products. |
|  | 2. | Make changes to the product by adding, |
|  |  | editing or deleting. |
|  |  |  |

*Table 4.7 Edit Product Category*

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Edit Product category** | |
|  |  | |
| Actor | Manager | |
|  |  | |
| Description | Enables the manager to group products based on its category. | |
|  |  | |
|  |  | |
| Precondition | Manager should login. | |
|  |  |  |
| Steps Performed | 1. | Click on categories |
|  | 2. | View all categories. |
|  |  | 3. Make the necessary changes by adding, editing or deleting categories. |
|  |  |  |
|  |  |  |

*Table 4.8 Report Generation*

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Generate Report** | |
|  |  |  |
| Actor | System |  |
|  |  | |
| Description | Generates reports on periodic basis. | |
|  |  |  |
|  | 1. | Click on report on the manager’s |
| Steps Performed |  | interface. |
|  | 2. | Enter the period, that is start and end |
|  |  | date desired for report. |
|  |  |  |

*Table 4.9 Send SMS/Email*

|  |  |
| --- | --- |
| **Use Case Name** | **Send SMS/Email** |
|  |  |
| Actor | System |
|  |  |
| Description | Sends order confirmation to the customer and |
|  | order details to the vendor via Email. |
|  |  |
|  | When an order is successful after payment, the |
| Steps Performed | system automatically sends an email |
|  | to the customer. |
|  |  |

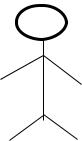
**4.4.2 SEQUENCE DIAGRAM**

Sequence diagrams are the most common kind of interaction diagrams that shows how actors and objects interact to realize a use case scenario. It focuses on the message interchange between a number of lifelines. The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. We normally draw a sequence diagram if we have a use case, to describe how the main components of the system interact. Then again sequence diagram helps us identify messages arriving at an interface of a component, to describe how the internal parts of the component interact.

*Table 4.10 Sequence diagram symbols and their meanings*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| SYMBOL | | | | | TERM & MEANING |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  | **An actor**: Is an entity or system that derives benefit from and is external to the system. Participates in a sequence by sending and/or receiving messages. |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  | **An object**: Participates in a sequence by sending and/or receiving messages. |
|  | An Object: a class | | |  |
|  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  | **A lifeline**: Represents the life of an object during a sequence. |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  | **A focus of control**: Is a long narrow rectangle |
|  |  |  |  |  |
|  |  |  |  |  | placed on or above a lifeline. Denotes when an |
|  |  |  |  |  | object is sending or receiving messages. |
|  |  |  |  |  |  |
|  |  | | | | **A message**: Conveys information from one |
|  |  |  |  |  | object to another. |
|  | **A Message ()** |  |  |  |  |
|  |  | | | | **Object destruction**: An X is placed at the end |
|  | X |  |  |  | of an object’s lifeline to show that it is going |
|  |  |  |  |  | out of existence |
|  |  |  |  |  |  |

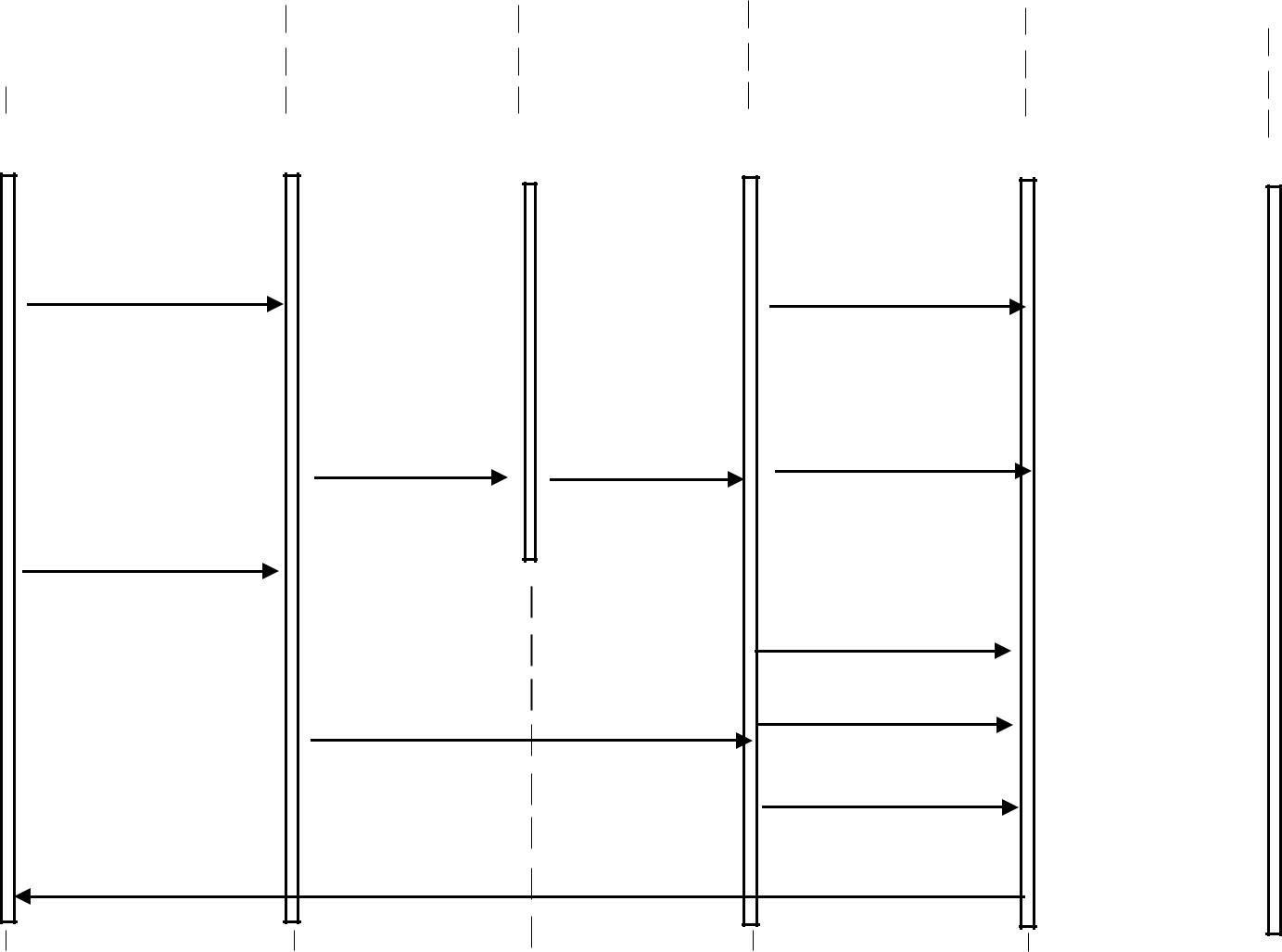


**Sequence Diagram for Customer Use Case**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| :Home |  | :Login |  | : Login |  | :Dashboard |  | :Logout |
|  |  | Page |  | Page |  |  |  |  |
|  |  |  |  |  |  |  |  |  |



Customer



|  |  |  |
| --- | --- | --- |
| Visit Shop() |  | View Product() |
| New Customer |  |  |
| signup() | logon() | Send feedback() |

View Product()

Place order()

Existing Customer logon() Make payment() logout()

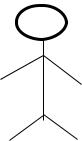


Success messages()



*Figure 4.8 shows the sequence diagram for the customer use case.*

**Sequence Diagram for Manager Use Case**

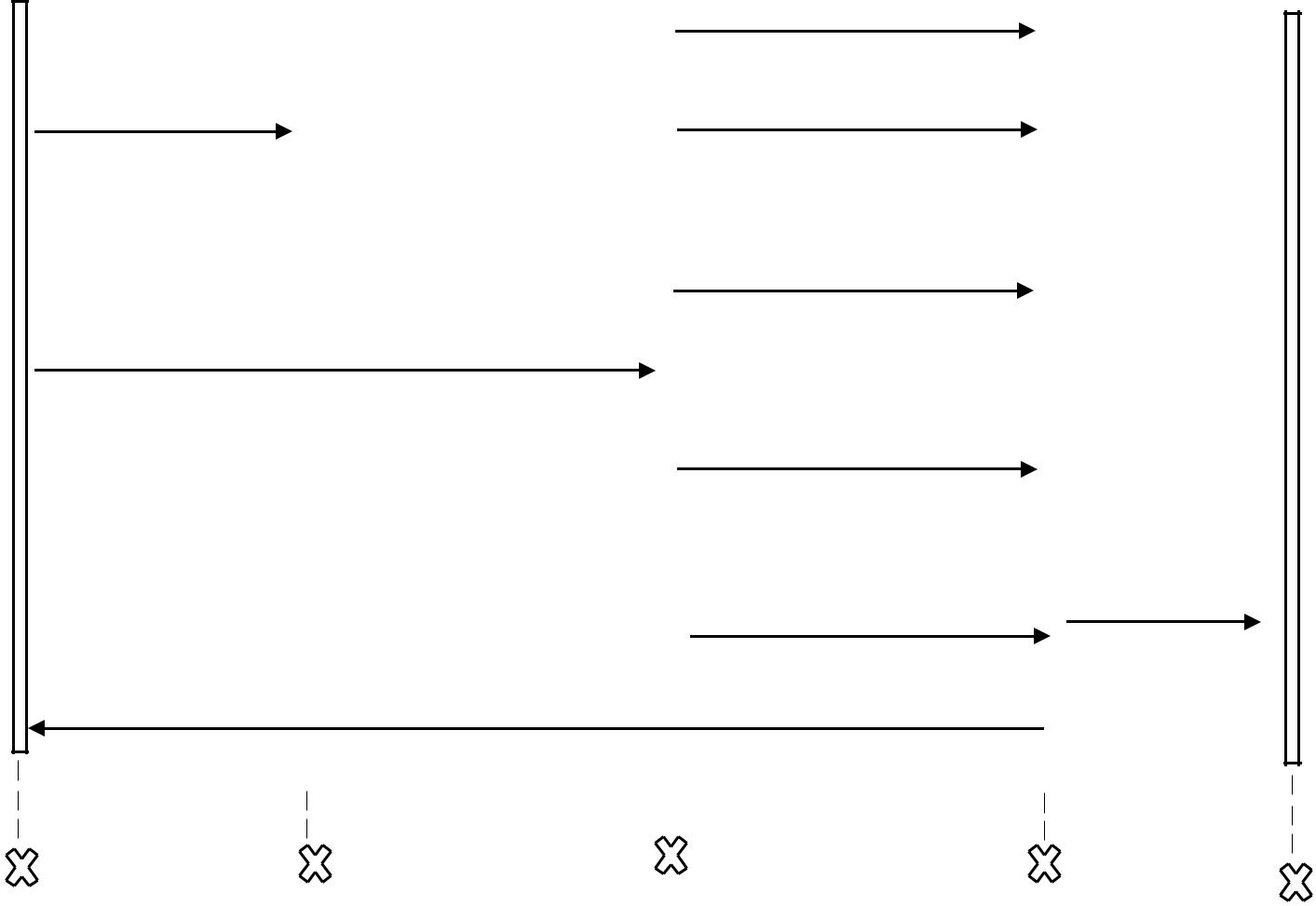
****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| :Manager |  | :Manager |  | :Other |  | :Logout |
| Login Page |  | Dashboard |  | Activities |  |  |
|  |  |  |  |  |  |  |



Manager

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | Customize website() |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Manager logon() |  |  |  |  |  |  | Edit Product category() |  |  |  |
|  |  |  |  |  |  |  | Edit menu() |  |  |  |
| Move |  |  | manager to dashboard() |  |  |  | Edit Delivery locations() |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | Approve Orders() |  |  | logon() |
|  |  |  |  |  |  |  |  |  |
|  |  |  | Success messages() | |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |



*Figure 4.9 shows the sequence diagram for the Manager use case.*

**4.4.3 ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. It describes the flow of control of the target system, such as the exploring complex business rules and operations, describing the use case also the business process. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows).

**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  | Handles web Form |
| Homepage | |  | Logon System |  |
| Click logon | Submit form | data and Validations |
|  |  |  |

Get customer details

Re-enter details

No

|  |  |
| --- | --- |
| Display Error | Details |
|  |

found?

Yes

Display dashboard to

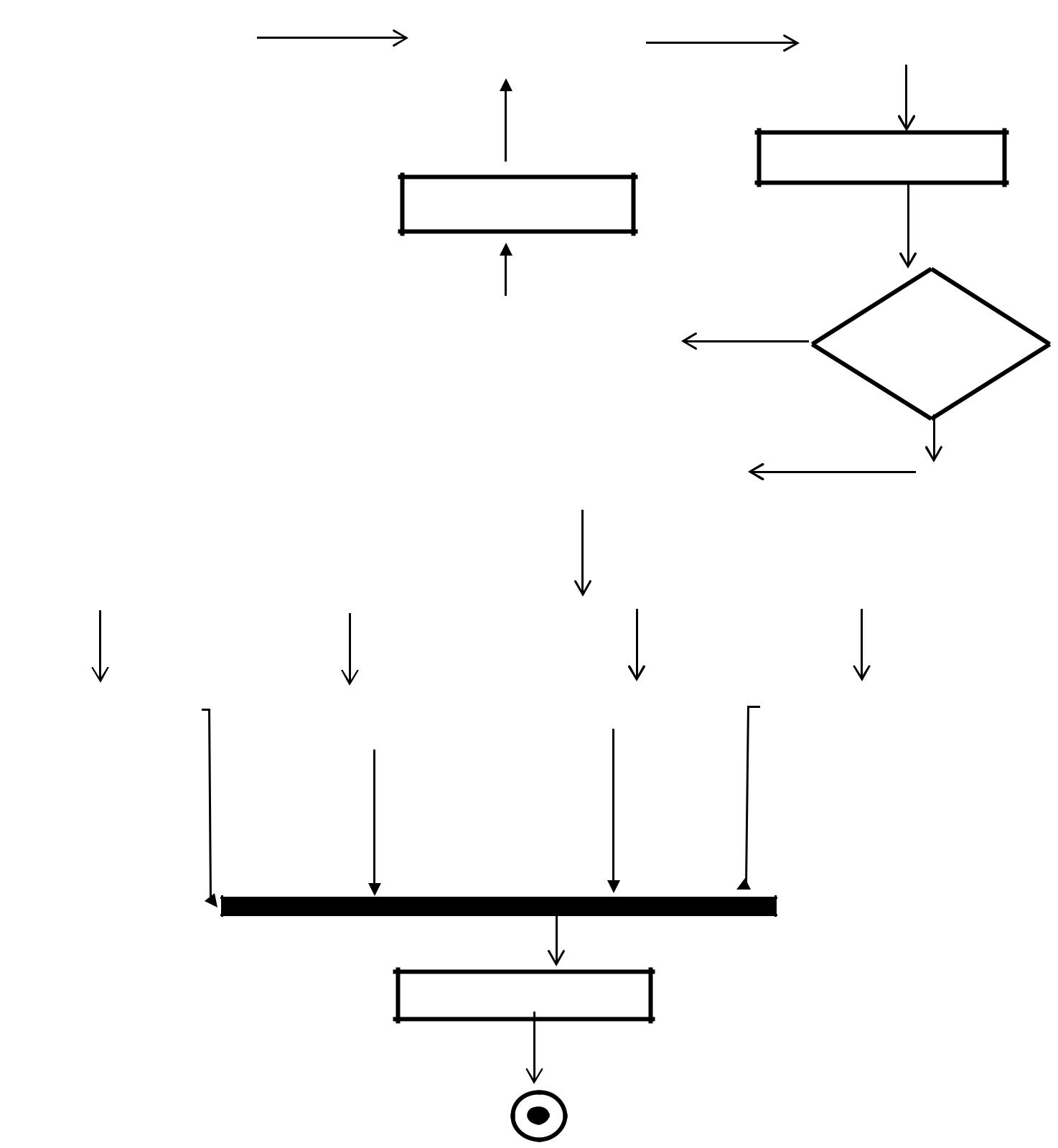
Customer

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| Check Products |  |  | Place Order |  | Make Payment |  | Send Feedback |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

*Figure 4.10 shows the activity diagram for the customer’s use case*

**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  | Manager Logon |  | Handles web Form |
| Homepage | |  |  |
| click admin login | System | Submit form | data and Validations |
|  |  |
|  |  |  |
|  |  |  |  |



Get Manager details

Re-enter details

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | No | Details | |
|  | Display Error | |  |
|  |  |  |  |  |
|  |  |  |  |  | found? | |
|  |  |  | |  |  |  |
|  |  | Display dashboard to | |  | Yes | |
|  |  | Manger | |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Customize Site |  | Approve Orders |  | Edit Product Details |  | View Reports |
|  |  |  |  |  |  |  |

Activity Confirmations

*Figure 4.11 shows the activity diagram for the Manager use case*

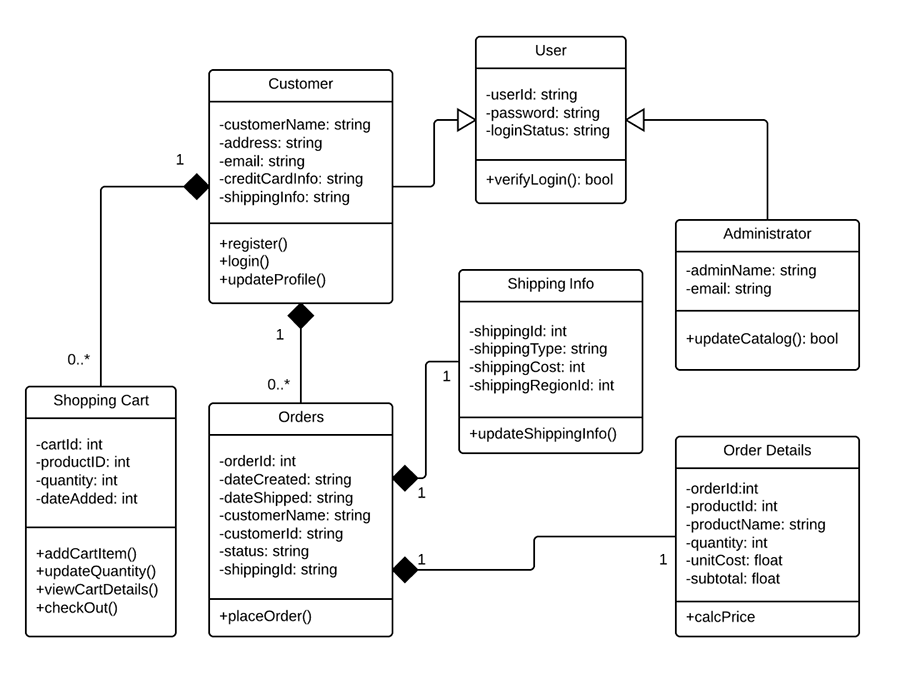
**4.4.4 CLASS DIAGRAM**

The class diagram is a central modeling technique that runs through nearly all object-oriented methods. This diagram describes the types of objects in the system and various kinds of static relationships which exist between them.

* **Relationships**

There are three principal kinds of relationships which are important:

* **Association** - represent relationships between instances of types (a person works for a company a company has a number of offices.
* **Inheritance** - the most obvious addition to ER diagrams for use in OO. It has an immediate correspondence to inheritance in OO design.
* **Aggregation** - Aggregation, a form of object composition in object-oriented design.

****

*Figure 4.12 shows the class diagram for the ecommerce system*

**4.5 DATABASE DESIGN**

Database is any collection of data, or information, that is specially organized for rapid search and retrieval by a computer. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations. A database is stored as a file or a set of files on magnetic disk or tape, optical disk, or some other secondary storage device. The information in these files may be broken down into records, each of which consists of one or more fields. Fields are the basic units of data storage, and each field typically contains information pertaining to one aspect or attribute of the entity described by the database. Records are also organized into tables that include information about relationships between its various fields.

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

A well-designed database gives access to up-to-date and accurate information. The Entity-Relationship (ER) model, Unified Modeling Language (UML), Relational Model (RM) among others are some of the models used by database designers. The Entity-Relationship (ER) model was used for our database design.

**4.5.1 Entity relationship model**

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database.

An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

* **Entities**, which are represented by rectangles. An entity is an object or concept about which you want to store information.



*Figure 4.13 Entity diagram*

A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone. 

*Figure 4.14 A weak entity diagram*

* **Actions**, which are represented by diamond shapes, show how two entities share information in the database.



*Figure 4.15 Actions diagram*

In some cases, entities can be self-linked. For example, employees can supervise other employees.  
 

*Figure 4.16 Self-linked entity*

* **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.



*Figure 4.17 Attribute diagram*

A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.



*Figure 4.18 Multivalued attribute diagram*

A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.

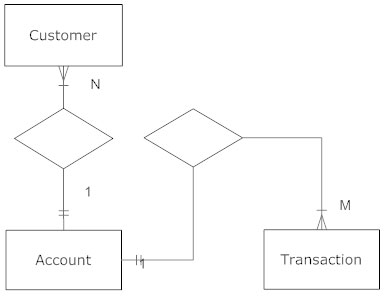


*Figure 4.19 Derived attribute diagram*

* **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.
* **Cardinality** specifies how many instances of an entity relate to one instance of another entity or it defines the numerical attributes of the relationship between two entities or entity sets. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.

The three main cardinal relationships are **one-to-one**, **one-to-many**, and **many-many**.

A one-to-one example would be one student associated with one mailing address. A one-to-many example (or many-to-one, depending on the relationship direction): One student registers for multiple courses, but all those courses have a single line back to that one student. Many-to-many example: Students as a group are associated with multiple faculty members, and faculty members in turn are associated with multiple students.

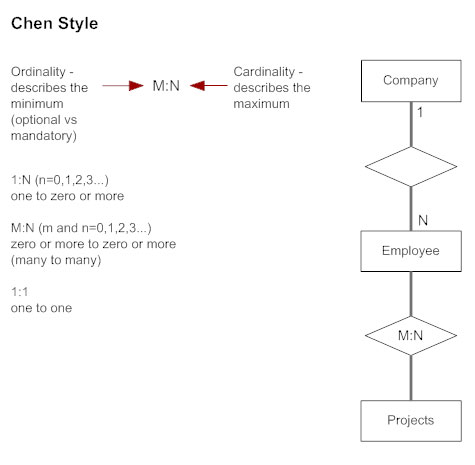


*Figure 4.20 Customer Cardinality diagram*

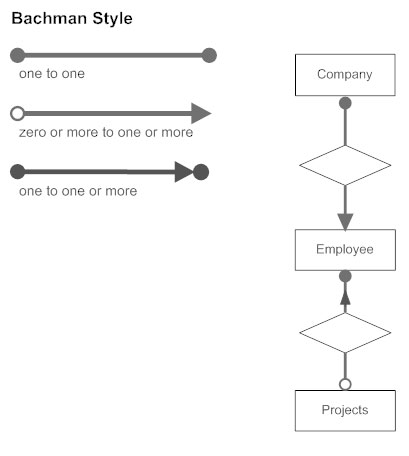
There are many notation styles that express cardinality **Information Engineering Style, Martin Style, Chen Style, Bachman Style** etc**.**



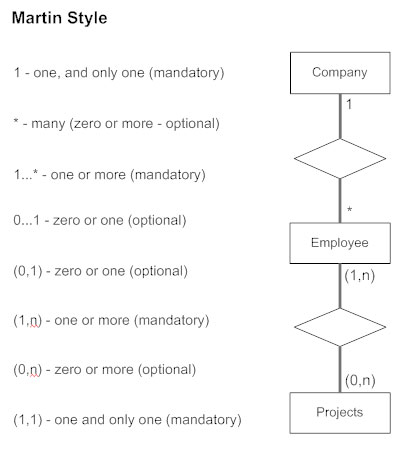
*Figure 4.21 Information Engineering Cardinality Style*



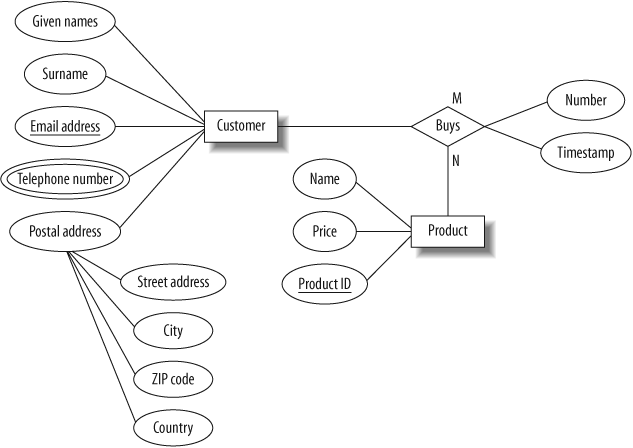
*Figure 4.22 Chen Cardinality Style*



*Figure 4.23 Bachman Cardinality Style*



*Figure 4.24 Martin Cardinality Style*

****

*4.25 ER diagram for the customer model*

**4.6 NORMALIZATION**

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes: **Eliminating redundant(useless) data** and **ensuring data dependencies make sense i.e. data is logically stored.**

* **Insertion Anomaly:**

It occurs when certain attributes cannot be inserted or entered into the database without the presence of other attributes.

* **Update Anomaly:**

An update anomaly exists when one or more instances of duplicated data is updated but not all.

* **Deletion Anomaly:**

This is when certain attributes are lost because of the deletion of other attributes.

**4.6.1 NORMALIZATION RULE**

Normalization rules are divided into the following normal forms:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. Boyce and Codd Normal Form (BCNF)

* First Normal Form (1NF):

The rule for First Normal Form states that no two rows of data must contain repeating group of information, therefore each set of columns must have a unique value. To ensure tables in the project are in First Normal Form, it should follow the following 4 rules:

1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

* Second Normal Form (2NF):

A table is said to be in 2NF if it is in 1NF and there is no partial dependency of any column on the primary key. To ensure tables in the project are in the Second Normal Form:

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

* Third normal Form (3NF):

It applies that a table must be in 2NF and every non-prime attribute of the table must be dependent on the primary key and hence cannot be determined by another non-prime attribute. To ensure tables in the project are in the Third Normal Form:

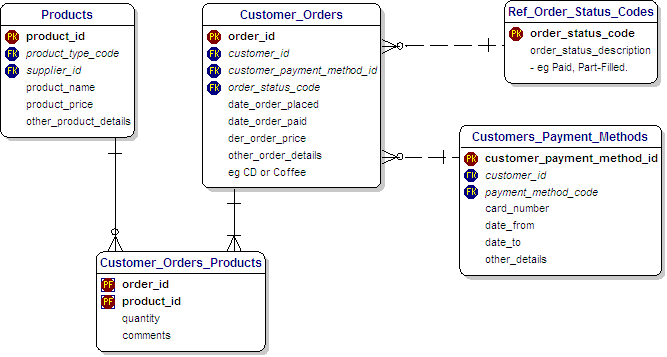
1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

* Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. To ensure tables in the project are in the BCNF, following conditions must be satisfied:

1. R must be in 3rd Normal Form
2. and, for each functional dependency (X → Y), X should be a super Key.

The database for this project is normalized in the **Third Normal Form (3NF)** to ensure data redundant data is eliminated that is, all data is stored in only one place and also to ensure data dependencies are logical which means all related data items are stored together.



*Figure 4.26 Normalized database for all tables*

**4.7 USER INTERFACE DESIGN**

User interface design or UI design generally refers to the visual layout of the elements that a user might interact with in a website, or technological product. This could be the control buttons of a radio, or the visual layout of a webpage. User interface designs must not only be attractive to potential users but must also be functional and created with users in mind. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design). Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself.

During the system development phase, several UI design features like input controls which includes textboxes, radio buttons, checkboxes, date fields, dropdown lists etc. For example, when a customer wants to place an order, recommendations will be made on the products they select to help the customer order with ease. The use of navigation components such as pagination, search field, slider, icons, tags etc. will be used to make user interactions with the website easier. Information components like notifications, messages boxes, contact forms etc. will also be used to help the user get quick feedbacks from the system.

The following best User interface design practices used in Software and website Development will be used in the user interface (UI) design for this project:

* **Clarity:**

The interface will be designed with the customer who willmake the orders in mind. The information content is conveyed quickly and accurately. The interface will be simple for the customer to navigate easily.

* **Consistency:**

A unique design, conformity with user's expectation in mind. The UI design elements that would beused in the system implementation will be consistent throughout the site. This will help the user to be familiar with certain UI elements so that they will not be using different elements for similar operations.

* **Conciseness**:

Users will not be overloaded with extraneous information**.** Consideration would be made for the spatial relationshipsbetween items on the page and structure the page based on importance. Careful placement of items would help draw attention to the most important pieces of information and can aid scanning and readability.

* **Strategically use color and texture:**

Direct attention would be drawn towards or redirected away from items using color, light, contrast, etc. The intention is to blend colors that will make the website nice and simple.

* **Detectability:**

Carefully consideration will be made on how to usetypeface to help draw the user's attention towards information required. Different sizes, fonts, and arrangement of the text to help increase scan ability, legibility and readability of the user.

* **Comprehensibility:**

It is imperative toinform users of location, actions, changes in state, or errors. The use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for users and that is exactly what will be done during the implementation of the system. This makes the meaning clearly understandable, unambiguous, interpretable, and recognizable.

* **Thinking about the defaults:**

By carefully thinking about and anticipating the goalspeople bring to our site, we can create defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where you might have an opportunity to have some fields pre-chosen or filled out.

**SUMMARY**

This chapter focused on the analysis and design of the system which were the general system and application architecture, requirements specification, functional and non-functional requirements, use case modelling, the database design, Entity Relationship diagram, Normalization, and User interface design. Outline for the requirement specification of the system which included both functional and non-functional requirements and the hardware requirements were explained. Further explanation on the various Unified Modelling languages used in modelling the system were discussed. This included use case diagrams, activity diagrams, class diagrams and sequence diagrams. These diagrams provided a pictorial view of the relationships between the various entities or users of our system. Also, the design of the system’s database including the Entity-Relationship (ER) diagram and the importance of database normalization were elaborated including the database being normalized in the third normal form (3NF). Finally, the user interface was looked at including how the user interface will look and why it was necessary for the interface to be user friendly.

**CHAPTER FIVE**

**SYSTEM IMPLEMENTATION**

**5.0 INTRODUCTION**

This chapter details out the implementation of the system. This include mapping logical design onto physical platform, the construction which consist of screen shots of forms, databases and reports among others. The chapter will also outline the various testing that were performed on the system, that is unit and system testing and the results obtained.

**5.1 MAPPING LOGICAL DESIGN ONTO PHYSICAL PLATFORM**

Logical design involves arranging data into a series of logical relationships called entities and attributes. Here, we defined the various fields for the database schema. Customer first names and other names were converted into first\_name and other\_name respectively. We used the larval framework and the other development tools to develop the various model of the project and linked them together. The next sub-topic gives a graphical view of the various outputs after we transformed the logical design into executable codes

**5.2 CONSTRUCTION**

This section focuses on the pictorial view of the user interface including the various forms, databases and reports. Below are various screen shots showing how the system was constructed.

The screenshots of the various tables constructed are shown below:

**CHAPTER SIX**

**FINDINGS AND CONCLUSION**

**6.0 INTRODUCTION**

Findings and conclusion concerns itself with all the information gathered including facts and figures compiled to fulfil the objectives of a research or the development of a project as well as the illations drawn from the findings. Also included is the principal results of a research project as well as what the project suggested or indicated and the interpretations of the relevance of the findings of a project.

This final chapter will consist of findings and conclusion. This will include summary of various problems faced in the development of the system, achievements and challenges, recommendations as well as enhancements that can be made to the system in future.

**6.1 SUMMARY OF PROBLEMS**

Every research or project has challenges and this project is no exception. A number of problems were encountered during the development of the project. Mobile Money payment is a major payment platform in Africa and my goal was to integrate it into this project to easy payment for users. These major mobile money networks don’t provide an API for developers to interface other apps to use these services. There are third party companies that provide API but in only few programming languages like PHP and Java. I used the Python programming language to develop this project but all the effort to use this API’s was not successful because there was no API for the language. I made an effort to contact them but all of them said there was no API for the Python programming language. Because of this, I had to use PayPal as the only payment gateway to accept payments for both debit and credit cards.

The above were basically the problems encountered while working on this project.

**6.2 ACHIEVEMENTS AND CHALLENGES**

**6.2.1 ACHIEVEMENTS**

Through dedication, determination, effort, skill and hard work a number of successes and achievements were achieved these are stated below:

* Working alone has helped me to develop new skills like completing a project within a specific time. I used to do a lot of side projects but never finished any of them but this project has taught me to break-up projects into smaller pieces and accomplished them within a specific time.
* The development of the project required the use of a Python framework known as Django and also a database known as Postgresql which my first time of was using them but I was able to learn and use it for the development of the project. This have therefore helped me to enhance my knowledge and programming skills.
* Implementing the online shop in twenty different languages was the ultimate goal for this project. This is an achievement because the main motivations for undertaking this project included making online shopping easy for shoppers and shop owners. By means of internationalization and localization that problem is tackled.

**6.2.2 CHALLENGES**

Although the development of the project was a success, a number of challenges were encountered as well because every research or project has challenges and this project is no exception.

These include:

* One of the challenges faced was to complete the project on time. This is as a result of the fact that I had to make time for classes, mid-semester exams, final examinations and also for developing the project.
* Implementing the Mobile Money Payment system was the biggest challenge of this project. This was very difficult from the onset because there was no API’s for the Python programming language.

**6.3 RECOMMENDATIONS**

Recommendations involve additions, suggestions or courses of actions that could be added to the project in the future.

Some recommendations for the system include:

* The project is purely web-based hence we recommend that a mobile version could be developed to make it more convenient for users to use especially now that mobile devices are abundant.
* Another recommendation is that more payment options should be added especially for the mobile money section. We have options only for either debit or credit cards hence limiting those who can use the application.
* Adding multiple currencies for user to choose their currency of choice before they start shopping is one recommendation for future works.

**6.4 FUTURE WORK**

In future, I hope to delve more into the integration of Mobile Money payments. This is because the payment aspect of the system does not offer this to the user. The hope is to expand the choices available to the shoppers. Also, I hope to add multiple currency to make users choose the currency of their choice before they start shopping.

**6.5 CONCLUSION**

The Internet has become a major resource in modern business, thus electronic shopping has gained significance not only from the entrepreneur’s but also from the customer’s point of view. For the entrepreneur, electronic shopping generates new business opportunities and for the customer, it makes comparative shopping possible.

As per a survey, most consumers of online stores are impulsive and usually make a decision to stay on a site within the first few seconds. “Website design is like a shop interior. If the shop looks poor or like hundreds of other shops the customer is most likely to skip to the other site.

Hence, the project has been designed to provide the user with easy navigation, retrieval of data and necessary feedback as much as possible. In this project, the user is provided with a multi-lingual ecommerce web site that can be used to buy books online. To implement this as a web application Django was used as the Technology. Django has several advantages such as enhanced performance, scalability, built-in security and simplicity.

To build any web application using the Django framework, the programming language needed is Python. Python was the language used to build this application. For the client browser to connect to the Django framework, Web Server Interface Gateway (WSGI) was used as the Web Server.

Django uses Object-Relational Mapping (ORM) to interact with the database as it provides in-memory caching that eliminates the need to contact the database server frequently and helps protect against SQL injection attack. PostgreSQL was used as back-end database since it is one of the most popular and open source databases, and it provides fast data access, easy installation and simplicity.

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. The design of the project which includes Data Model and Process Model illustrates how the database is built with different tables, how the data is accessed and processed from the tables.

The building of the project has given me a precise knowledge about how Django is used to develop a website, how it connects to the database to access the data and how the data and web pages are modified to provide the user with a shopping cart application.

REFERENCES

10ecommercetrends. (2017, November 20). *10 Ecommerce Trends 2017*. Retrieved from 10 Ecommerce Trends : http://10ecommercetrends.com/

Ahiabenu, K. (2017, November 15). *How can e-commerce transform your business?* Retrieved from Graphic Online: https://www.graphic.com.gh/features/opinion/how-can-e-commerce-transform-your-business.html

ARM Worldwide . (2017, November 17). *What are the E-Commerce Tech Trends in 2018*. Retrieved from ARM Worldwide: https://armworldwide.com/e-commerce-tech-trends/

Boampong, P. B. (2017, November 13). *Breakthrough of E-commerce in Ghana*. Retrieved from Ghana Web: https://www.ghanaweb.com/GhanaHomePage/NewsArchive/Breakthrough-of-E-commerce-in-Ghana-347247

Codecademy. (2018, April 05). *MVC: Model, View, Controller*. Retrieved from Codecademy: https://www.codecademy.com/articles/mvc

Django Stars. (2018, January 15). *Why We Use Django Framework*. Retrieved from Django Stars: https://djangostars.com/blog/why-we-use-django-framework/

*eCommerce Report: Ghana’s Top 20 eCommerce Websites*. (2017, November 22). Retrieved from Modern Ghana: https://www.modernghana.com/news/640481/ecommerce-report-ghanas-top-20-ecommerce-websites.html

Frederick, J. (2017, November 22). *4 Emerging Ecommerce Technology Innovations*. Retrieved from PFS Web: http://www.pfsweb.com/blog/4-emerging-ecommerce-technology-innovations/

Gary, H. (2017, November 22). *15 Must-Have Features for E-commerce Sites*. Retrieved from Search Engine Journal: https://www.searchenginejournal.com/15-must-have-features-for-e-commerce-sites/181974/

Gaur, R. (2018, April 05). *Software Architecture: One-Tier, Two-Tier, Three Tier, N Tier*. Retrieved from TECHYNEWS4U: https://techynews4u.com/software-architecture-one-tier-two-tier-three-tier-n-tier/

Inflectra. (2018, April 06). *What are System Requirements Specifications/Software (SRS)?* Retrieved from Inflectra: https://www.inflectra.com/ideas/topic/requirements-definition.aspx

Investopedia. (2017, November 20). *Business To Consumer - B To C*. Retrieved from Investopedia: https://www.investopedia.com/terms/b/btoc.asp

Kramek, A. (2018, April 05). *Introduction to Client Server Architecture*. Retrieved from Foxite: http://weblogs.foxite.com/andykramek/2008/09/29/introduction-to-client-server-architecture/

Miva. (2017, November 26). *The History Of Ecommerce: How Did It All Begin?* Retrieved from Miva: https://www.miva.com/blog/the-history-of-ecommerce-how-did-it-all-begin/

Nicole, F. (2016, April 5). *4 Issues Your E-Commerce Business Needs to Address*. Retrieved from Business News Daily: https://www.businessnewsdaily.com/8947-research-roundup-ecommerce-issues.html

Nige, B. (2018, January 15). *Why Django?* Retrieved from Django Book: https://djangobook.com/tutorials/why-django/

Openhatch. (2018, April 05). *Django for Designers/Basic views*. Retrieved from Openhatch: http://wiki.openhatch.org/Django\_for\_Designers/Basic\_views

Opoku, J. (2017, November 25). *Setting up an e-commerce business in Ghana*. Retrieved from Sidekick Gh: https://www.sidekickgh.com/setting-up-an-e-commerce-business-in-ghana/

Opoku, J. (2017, October 17). *Why Responsive Websites are Awesome*. Retrieved from Sidekick Gh: https://www.sidekickgh.com/why-responsive-websites-are-awesome/

Post, J. (2017, December 19). *How to Beat 4 Big Challenges for Small E-Commerce Retailers*. Retrieved December 19, 2017, from Business News Daily: https://www.businessnewsdaily.com/6028-small-ecommerce-challenges.html

Rajkumar. (2018, April 05). *Software Architecture: One-Tier, Two-Tier, Three Tier, N Tier*. Retrieved from SoftwareTestingMaterial: https://www.softwaretestingmaterial.com/software-architecture/

Sheppy. (2018, January 17). *CSS: Cascading Style Sheets*. Retrieved from MDN web docs: https://developer.mozilla.org/en-US/docs/Web/CSS

SmartDraw. (2018, April 09). *Entity Relationship Diagram*. Retrieved from SmartDraw: https://www.smartdraw.com/entity-relationship-diagram/

Smith, L. (2018, January 25). *What PostgreSQL has over other open source SQL databases: Part I*. Retrieved from Compose: https://www.compose.com/articles/what-postgresql-has-over-other-open-source-sql-databases/

studytonight. (2018, April 09). *Normalization of Database*. Retrieved from studytonight: https://www.studytonight.com/dbms/database-normalization.php

Tawiah, A. (2017, November 22). *Ghana’s Top 12 eCommerce Websites*. Retrieved from IT News Africa : http://www.itnewsafrica.com/2015/09/ghanas-top-12-ecommerce-websites/

Team, P. C. (2017, November 23). *7 Key Features Online Shoppers Demand From an Online Store*. Retrieved from PinnacleCart: https://www.pinnaclecart.com/blog/7-key-features-online-shoppers-demand-from-an-online-store/

The Editors of Encyclopaedia Britannica. (2018, April 06). *Database*. Retrieved from Britannica: https://www.britannica.com/technology/database

Visual Paradigm. (2018, April 06). *What is Unified Modeling Language (UML)?* Retrieved from Visual Paradigm: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

Wikibooks. (2017, December 21). *Introduction to Computer Information Systems/E-Commerce*. Retrieved from Wikibooks: https://en.wikibooks.org/wiki/Introduction\_to\_Computer\_Information\_Systems/E-Commerce#Manufacturer\_and\_E-Tailer\_Sites

Wikipedia. (2018, April 09). *User interface design*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/User\_interface\_design