**Project Proposal**

**On**

**E-commerce**

**Admond Tamang**

**00172885**

**Computing Project**

**Level 5 Diploma in Computing**

**Softwarica College of IT and E-Commerce**

**Kathmandu, Nepal**

**2018/12/28**

**Submitted to: Kiran Rana**

**Table of Contents**

[**1** **Introduction** 4](#_Toc534380675)

[**1.1** **Project Introduction** 4](#_Toc534380676)

[**1.2** **Justification for project** 4](#_Toc534380677)

[**1.2.1** **Background of the project** 4](#_Toc534380678)

[**1.2.2** **Problem Statement** 4](#_Toc534380679)

[**1.3** **Description of project** 4](#_Toc534380680)

[**1.3.1** **Features** 4](#_Toc534380681)

[**2** **Project Scope** 4](#_Toc534380682)

[**2.1** **Scope and limitation of project** 4](#_Toc534380683)

[**2.2** **Aims and Objectives** 4](#_Toc534380684)

[**3** **Development Methodology** 5](#_Toc534380685)

[**3.1** **Methodology used** 5](#_Toc534380686)

[**3.2** **Design Pattern** 5](#_Toc534380687)

[**3.3** **System Architecture** 6](#_Toc534380688)

[**4** **Work Breakdown Structure (WBS) / Scheduling** 6](#_Toc534380689)

[**4.1** **Work Breakdown Structure** 6](#_Toc534380690)

[**4.2** **Milestones** 6](#_Toc534380691)

[**4.3** **Scheduling / Gantt Chart** 7](#_Toc534380692)

[**5** **Risk Management** 8](#_Toc534380693)

[**6** **Configuration Management** 8](#_Toc534380694)

[**7** **Conclusion of the project** 8](#_Toc534380695)

[**8** **References** 8](#_Toc534380696)

[Figure 1 Phases of waterfall model 6](#_Toc534496190)

[Figure 2: MVC design pattern 7](#_Toc534496191)

[Figure 3: Three tier architecture 8](#_Toc534496192)

[Figure 4: Work break down structure 9](#_Toc534496193)

[Figure 5: Scheduling time for tasks 11](#_Toc534496194)

[Figure 6: Gantt chart 11](#_Toc534496195)

[Figure 7: Github root directory 13](#_Toc534496196)

[Figure 8: Github proposal directory 13](#_Toc534496197)

[Figure 9: Local directory of project 14](#_Toc534496198)

# **Introduction**

## **Project Introduction**

E-commerce website is a modern design website that provide user to search, buy, share products online.

Bigheart web company is a modern website design and development company. It focuses on user centric design. It helps client to achieve their desire website. It lets user to search, buy and share website online. Customer can shop products from trusted developers, compare prices, read reviews and share products with friends.

## **Justification for project**

### **Background of the project**

### **Problem Statement**

**This application will reduce the problems arise in previous types o**

## **Description of project**

### **Features**

The features of the project are listed below:

* **User can sign up and login**

User can easily create their account and login into the website.

* **User can upload their products**

Developers can upload their products in the application and make available for customers.

* **Add products to wish list**

User can add the product to wish list if he/she like the product.

* **Rate and comment products**

User can rate and comment about product.

* **Get help from developer via online chat**

If user get confused, then they can chat with product developer.

* **User friendly interface**

Anyone can use the application interface as it is easy to navigate, and simple design helps to understand easily.

* **Share products with friends**

It facilities to share products via social medias.

* **Filter and search products**

User can use filter products to find the exact product they are looking.

# **Project Scope**

## **Scope and limitation of project**

## **Aims and Objectives**

**Aims:** The aims that I want to achieve are listed below:

* Developer would post their website and provide services to the customer.
* Customer should easily interact with the website and buy websites according to their needs.

**Objectives:** Action that I will take to achieve my aim are:

* Customer will have the ability to sign-up their account and browse different products.
* Developers will have the facility to post their website and provide services to the customer.

# **Development Methodology**

## **Methodology used**

Waterfall methodology was used to develop the application. It is sequential life cycle model. It is sequence of process which we cannot overtake or skip until the previous phase has been completed.

The waterfall methodology was used because of following reasons:

1. It is simple and easy to understand.
2. Progress of my project can be tracked easily.
3. My project has clear requirement, so I think waterfall model is suitable for my project.

Figure 1 Phases of waterfall model

## **Design Pattern**

I followed MVC design pattern which stands for Model-View-Controller.

The reason of choosing MVC design pattern are: (interserver, 2018)

* **Faster development process**

It supports parallel development which means that one programmer can work on application design and other can work on business logic simultaneously.

* **Modification does not affect entire model**

Model part does not depend upon view part, so any modification doesn’t entirely affect the architecture.

* **High cohesion and low coupling**

It makes low coupling among models and enables grouping of related logical action on a controller.

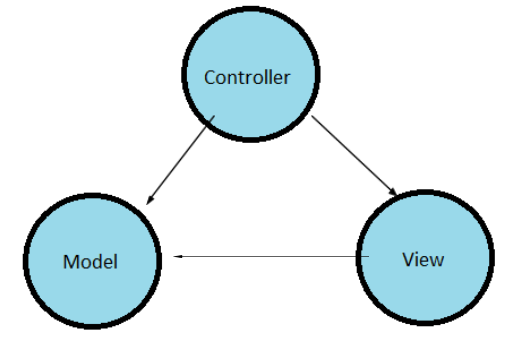


Figure 2: MVC design pattern

## **System Architecture**

3-tier architecture was used in the development of the project. It is divided into three layers. They are:

1. **Presentation**

Handles the interaction between user and the client business. For example, HTML5, CSS.

1. **Application**

Takes request from presentation tier and returns the output to presentation tier. For example, Java, .NET.

1. **Data tier**

Responsible for storing data and sending it to business tier. For example, MySQL, Oracle.

I have used 3 tier architectures because of following reasons:

1. **Scalability**

Middle tier can be added to make system run smoother. The system can work even hardware needs to be added and updated to load load-balance the presentation tier.

1. **Security**

Client is not directly access to the database. Middle layer protects the database tier ensuring strong security.

1. **Performance**

Presentation tier can cache requests, so network utilization is minimized and run smoothly.

1. **Maintainability**

It manages presentation code and business logic separately so change in business logic does not affect presentation layer.

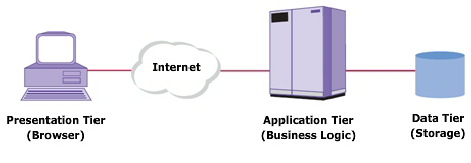


Figure 3: Three tier architecture

# **Work Breakdown Structure (WBS) / Scheduling**

## **Work Breakdown Structure**

All the things a project needs to accomplish in the project is displays graphically which helps to understand the projects activities clearly. The purpose of WBS is to break down complex activities into simple form.

Figure 4: Work break down structure

## **Milestones**

|  |  |
| --- | --- |
| **Milestones** | **Date** |
| **Project Management**  Risk Management  Work Breakdown Structure  Configuration Management  Proposal Submission | **12/21/2018 -1/3/2019**  12/21/2018 - 12/24/2018  12/25/2018 - 12/27/2018  12/28/2018 - 12/30/2018  12/31/2018 - 1/1/2019 |
| **Analysis**  Requirement Analysis  Use Case  Architecture (Initial Class Diagram)  Analysis Specification | **1/2/2019 - 1/25/2019**  1/2/2019 - 1/7/2019  1/8/2019 - 1/11/2019  1/12/2019 - 1/17/2019  1/18/2019 - 1/25/2019 |
| **Design**  Structural Diagram  Behavioral Diagram  UI Design  Database Design | **1/26/2019 - 2/24/2019**  1/26/2019 - 2/3/2019  2/4/2019 - 2/13/2019  2/14/2019 - 2/20/2019  2/21/2019 - 2/24/2019 |
| **Implementation**  Building Database  Coding | **2/25/2019 - 3/28/2019**  2/25/2019 - 3/1/2019  3/2/2019 - 3/28/2019 |
| **Testing**  Unit Testing  Integration Testing  Blackbox Testing  Whitebox Testing | **3/29/2019 - 4/8/2019**  3/29/2019 - 3/31/2019  4/1/2019 - 4/3/2019  4/4/2019 - 4/6/2019  4/7/2019 - 4/8/2019 |
| **Deployment**  User Training  Final Report | **4/9/2019 - 4/18/2019**  4/9/2019 - 4/13/2019  4/14/2019 - 4/18/2019 |

**Description of Milestones:**

* **Project Management (12 days)**
  + //In **each point** explain how many days you are allocating to each task and sub-task.
* **Analysis (24 days)**
* **Design (30 days)**
* **Implementation (32 days)**
* **Testing (11 days)**
* **Deployment (10 days)**

## **Scheduling / Gantt Chart**

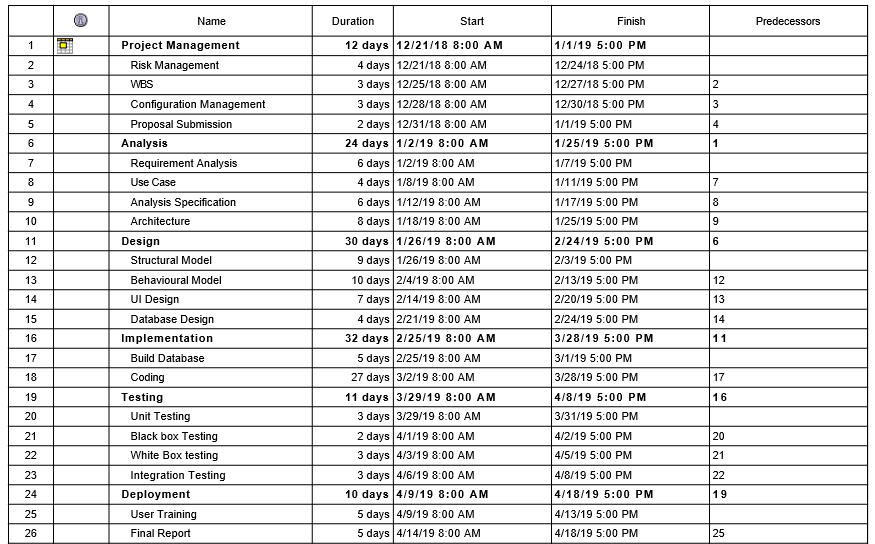
****

Figure 5: Scheduling time for tasks

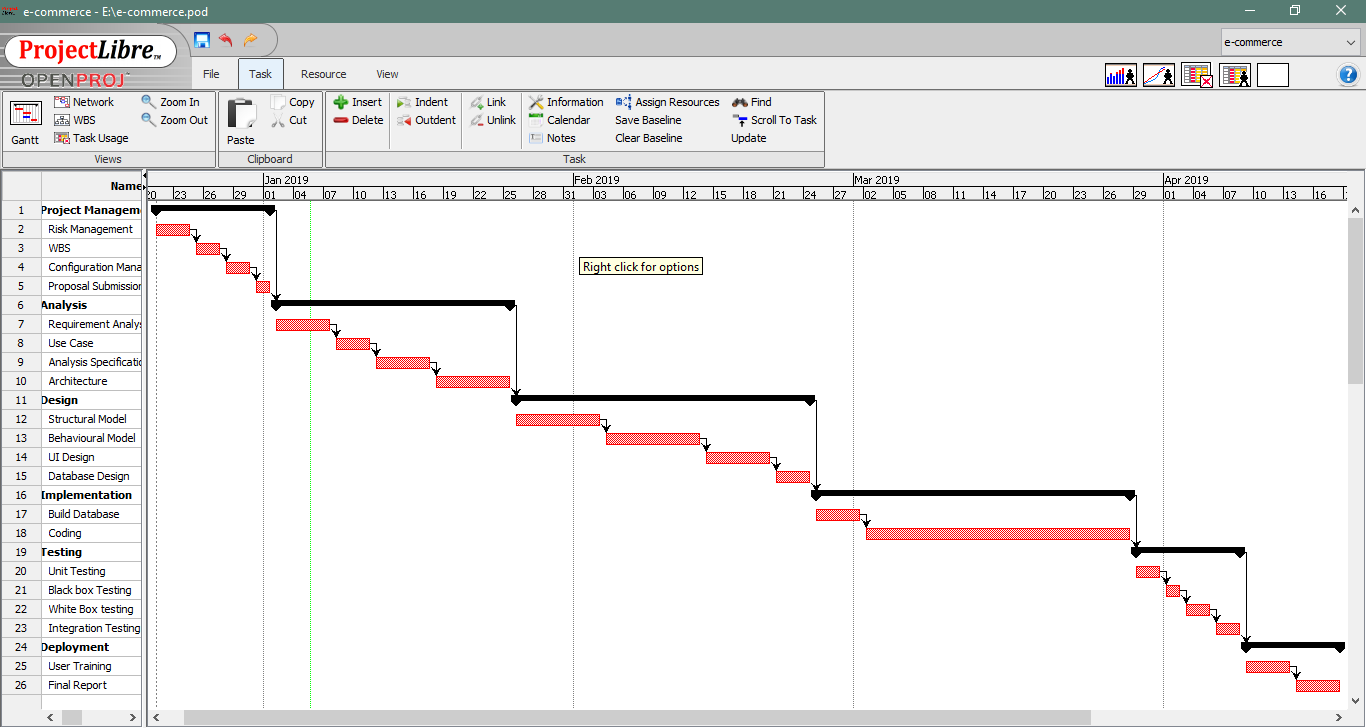
****

Figure 6: Gantt chart

# **Risk Management**

It is the process of identifying and prioritizing possible threats to an organization or system. Harmful risk is analyzed and solution to the

threats are prepared.

**Impact=likelihood \*Consequences**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **Risks** | **Likelihood** | **Consequences** | **Impact** | **Solution** |
|  | Lack of staff training | 2 | 3 | 6 | At the stage of deployment, staff should be trained properly to use the system. |
|  | Data theft | 3 | 4 | 12 | Strong company policy should be established. |
|  | Failure to meet requirement | 2 | 4 | 12 | Proper planning is to be done in every phase. |
|  | DDOS attack | 1 | 4 | 4 | Placing the publicly exposed server in DMZ. |
|  | Database error | 1 | 5 | 5 | Analyze the data traffic and upgrade the hardware if necessary. |
|  | Server failure | 1 | 5 | 5 | IT security should be hired to look after the server. |
|  | Change in government policy | 1 | 5 | 5 | Should be aware about the situation of the country and make changes accordingly. |

# **Configuration Management**

Configuration management is used to keep track of application and related information which includes software version and updates. It keeps track of modification, changes and update of any projects.

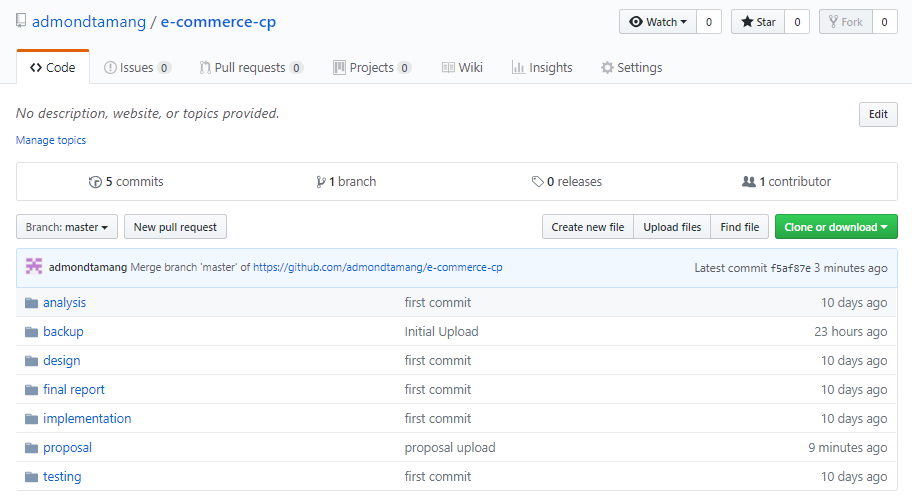


Figure 7: Github root directory

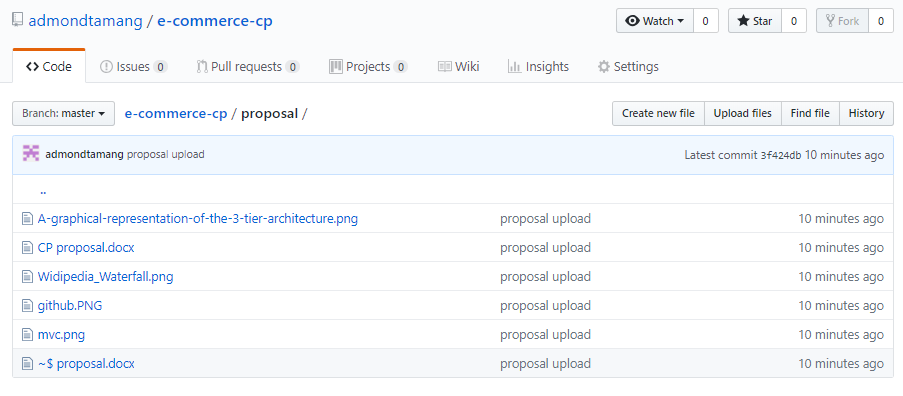


Figure 8: Github proposal directory

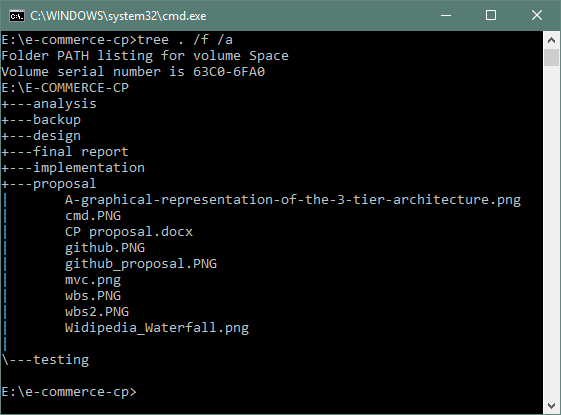


Figure 9: Local directory of project

## **Requirements**

**Functional requirement**

Those requirements which are related to behavioral aspect of the system. It specifies what the system should do. It includes business rule, administrative function and transaction correction.

**Non-Functional requirement**

These requirements specify how the system should behave and the limits on requirements.

## **MoSCoW prioritization**

I have prioritized my features using MoSCoW prioritization technique.

**Note**

**M**=must have

**S**=should have

**C**=could have

**W**=won’t have

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **Requirement** | **MoSCoW** | **Functionality** |
|  | Login | M | F |
|  | Register | M | F |
|  | Post product | M | F |
|  | Add to Wishlist | C | F |
|  | Search product | M | F |
|  | Filter product | S | F |
|  | View user profile | M | F |
|  | Edit profile | M | F |
|  | Chat with owner | S | F |
|  | Buy product | M | F |
|  | Rate product | C | F |
|  | View product | M | F |
|  | Comment product | C | F |
|  | Generate bills | M | F |
|  | Verify store owner/ seller | M | F |
|  | Delete user | S | F |

**Non-functional requirement**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Requirement** | **Description** |
|  | **Verfication** | **To verify customer and store.** |
|  | **Security** | **Protect user and system from hacking.** |
|  | **User friendly** | **To provide smooth experience.** |
|  | **Efiicient** | **Make system fast and reliable.** |

# **References**

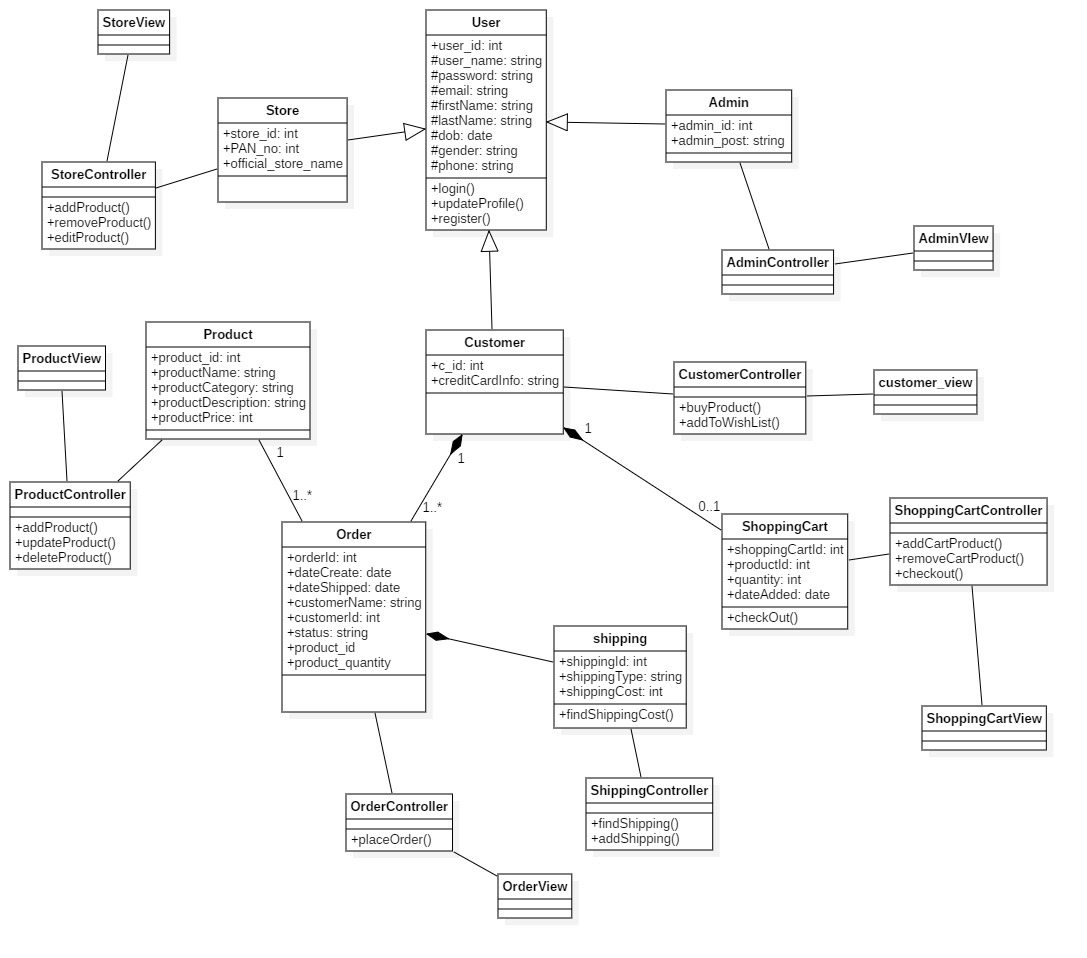
**Class Diagram**

Class diagram describes the attributes and operations of a system.

**Class Notation**

A class notation consists of three parts:

1. Class Name
   * Determines the name of class and shown at top.
2. Class Attributes
   * Attributes are variables of any class (data members) in code
   * Data type of variable are shown after colon.
3. Class Operations
   * Operations are shown in the third partition. It defines behavior of an class.
   * After colon return type of method parameters are shown following the parameter name.



I designed class diagram for my project because it helps to describes the responsibilities of the system which allows me to understand working mechanism of the system. As it holds class, attribute and operations which is required for the development of application. (htt)

**Use case diagram**

It shows the functionality of a system. It describes how a user's react with a system.

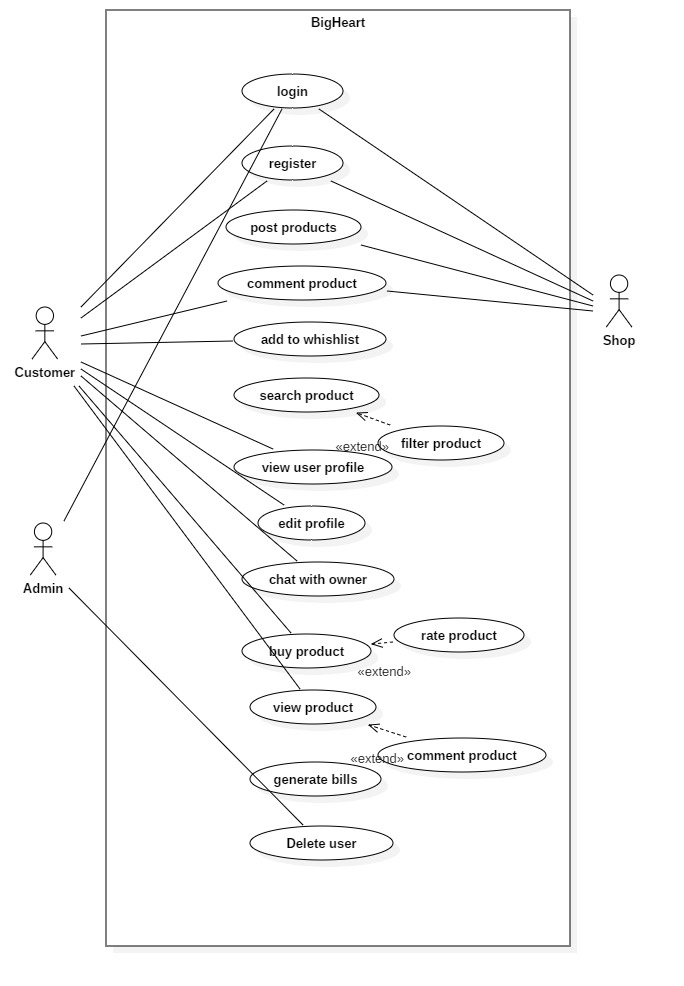
**Use case notation**

Actors: Users that interacts with the system.

System: That consist different use cases.

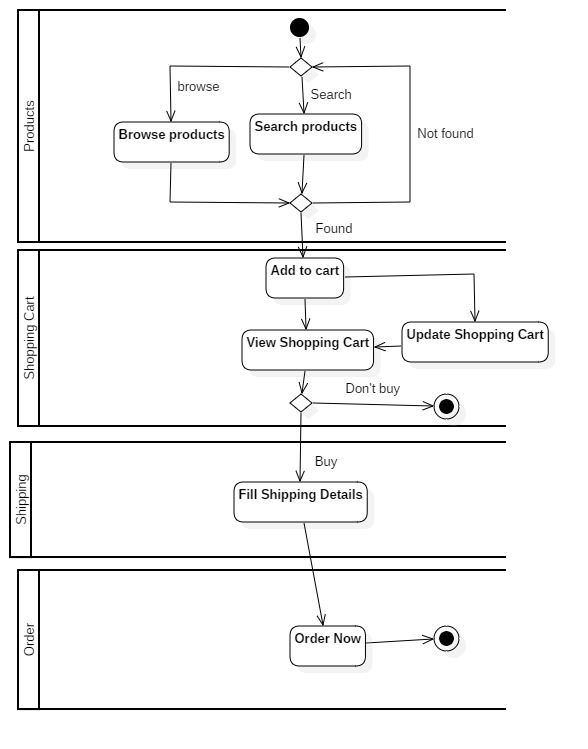
Use case: action that is performed by actor.

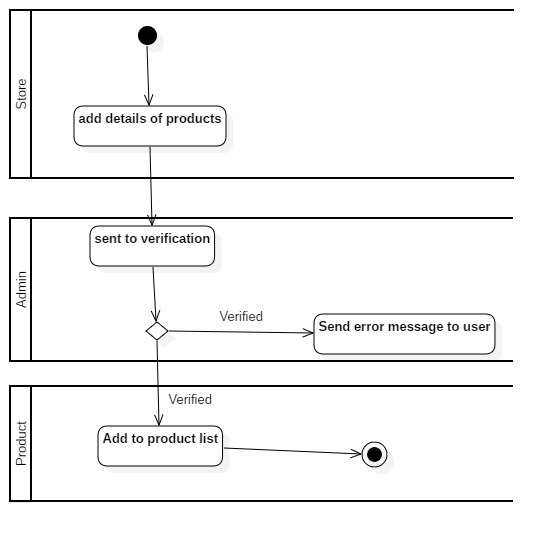
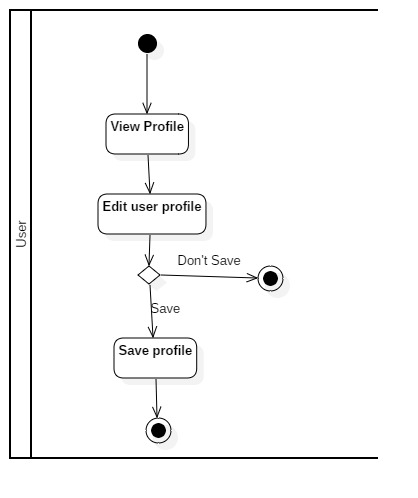
I have used use case because it helps to represent action performed by actor via use-case.



**Activity diagram**

I had used activity diagram to illustrate the flow of system. An activity diagram focuses on condition of flow and the sequence in which it happens. It helps to understand the flow of programs on a high level. It also enables them to figure out constraints and conditions that cause events.





I created activity diagram because it helps to show the flow operation in a system.