INDEX

Project Definition

Valentina Choco Project

Description

1) Following are the role of admin admin can manage the chocolate products according to category wise. it manage customers. it manage the order send by the customers. manage order status. manage cancel order by the customers. view feed back. manage the complaint send by customer

2) Following are the roles of customers view all the products according their requirements. add products into the carts. send order based on add to cart. view order status. cancel the order. admin can manage the Brosers means it uploaded by admin in which details about all product with description, price and images.

ORGANIZATION PROFILE

Front End	:	Admin side • PHP (version 7.1.11 or higher) Web services for android • PHP (version 7.1.11 or higher) Client side Android application using java programming.
Back End	:	Admin side • MYSQL, php(version 2.5) Client side • MYSQL, php(web service)
Reporting Tools	:	• PHP report maker (version 12.0.6 or higher)
Other Tools	:	• Presentation Microsoft power point 2019 • Diagrams Draw.Io • Documentation Microsoft word 2019

PROJECT PROFILE

A) EXISTING SYSTEM

There is no any existing system. All the task done manually and keep the record into the books. So, it is difficult to access and finding the record.

B) PROBLEM WITH EXISTING SYSTEM

C) PROPOSED SYSTEM

To solve the above problem we develop the online system. So, admin can take the orders anywhere any time. (24*7 hours) and admin can also manage the record easily and access anywhere any time.

UML (UNIFIED MODELING LANGUAGE)

DIAGRAM :-

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of **visually representing a system** along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at **Rational Software.** It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates

UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several **business processes or workflows.** For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts.

Data flow diagrams can be used in both Analysis and Design phase of Software Development Life Cycle (SDLC).
They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficacy.

There are several **types of UML diagrams** and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation).

Within the circular containers, we express the actions that the actors perform. Such actions are :Registration , login ,Categories ,product ,Add to cart ,Order ,Manage Employees ,Employee Assign installation, payment , cancel order , Order return ,Complain ,Feedback and Logout.

- Use case UML diagrams are good for showing dynamic behaviors between actors within a system, by simplifying the view of the system and not reflecting the details of implementation.

Components of UMl Diagram:-

- Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executable , libraries, etc.

- The purpose of this diagram is different. Component diagrams are used during the implementation phase of an application. However, it is prepared well in advance to visualize the implementation details.

- Initially, the system is designed using different UML diagrams and then when the artifacts are ready, component diagrams are used to get an idea of the implementation.

- This diagram is very important as without it the application cannot be implemented efficiently. A wellprepared component diagram is also important for other aspects such as application performance, maintenance, etc.

- Before drawing a component diagram, the following artifacts are to be identified clearly –

    - Files used in the system.
    - Libraries and other artifacts relevant to the application.
    - Relationships among the artifacts.

- After identifying the artifacts, the following points need to be kept in mind.

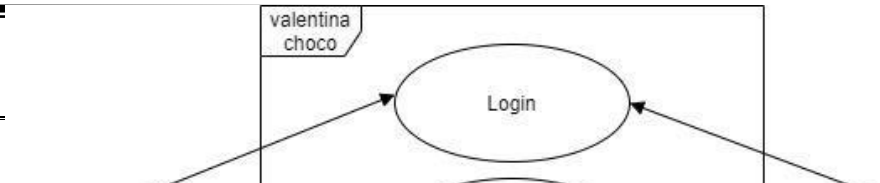
    - Use a meaningful name to identify the component for which the diagram is to be drawn.
    - Prepare a mental layout before producing the using tools.
    - Use notes for clarifying important points.

- Following is a component diagram for order management system. Here, the artifacts are files. The diagram shows the files in the application and their relationships. In actual, the component diagram also contains dlls, libraries, folders, etc.

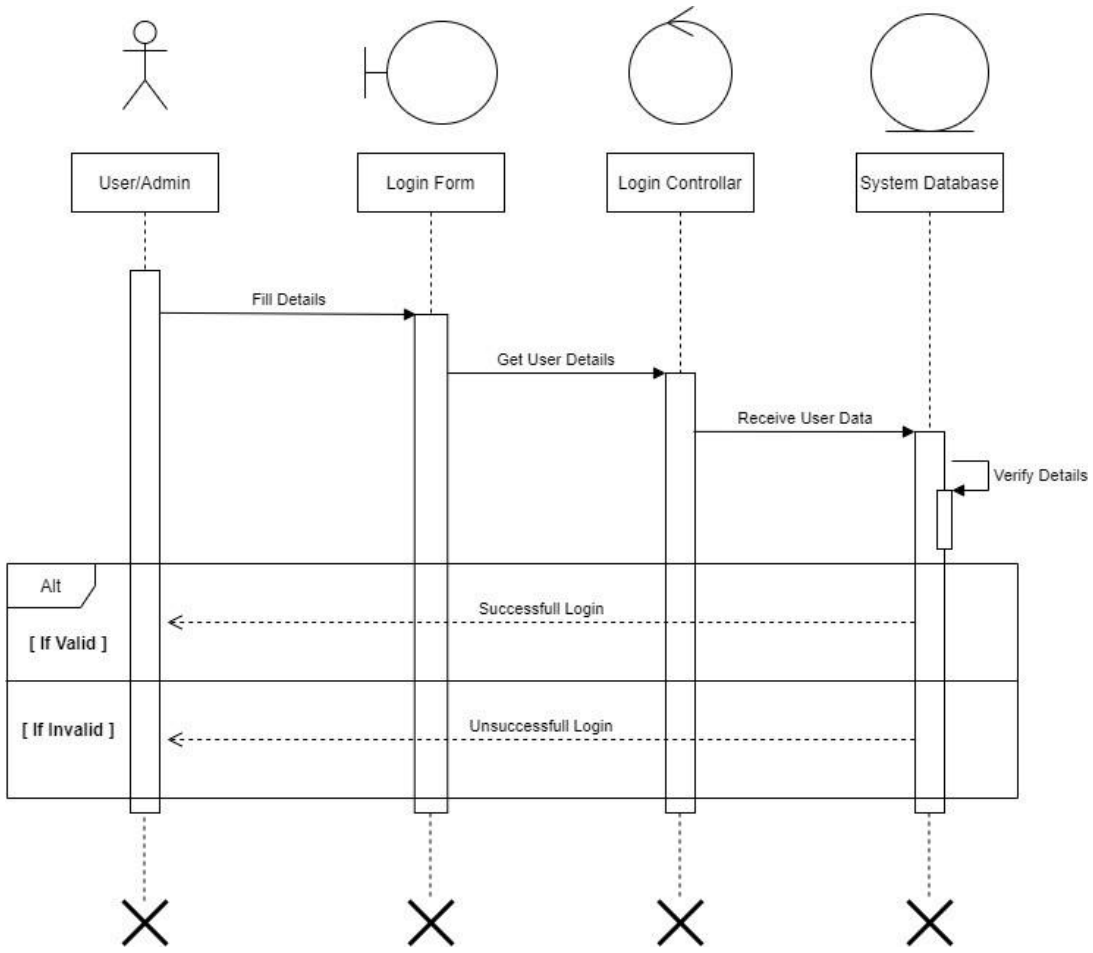
- Component diagrams are very important from implementation perspective. Thus, the implementation team of an application should have a proper knowledge of the component details

- Component diagrams can be used to

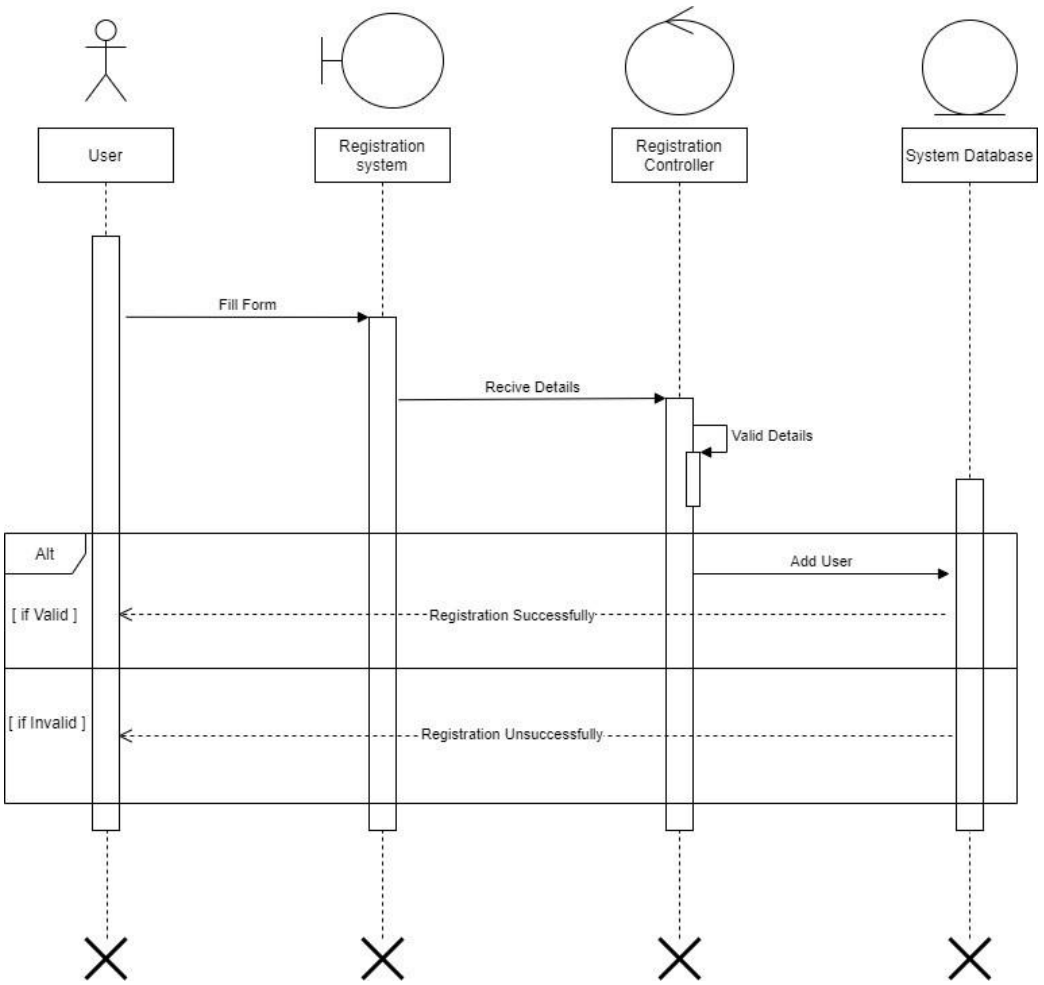
    - Model the components of a system.
    - Model the database schema.
    - Model the executables of an application.
    - Model the system's source code.


valentina
choco
Login

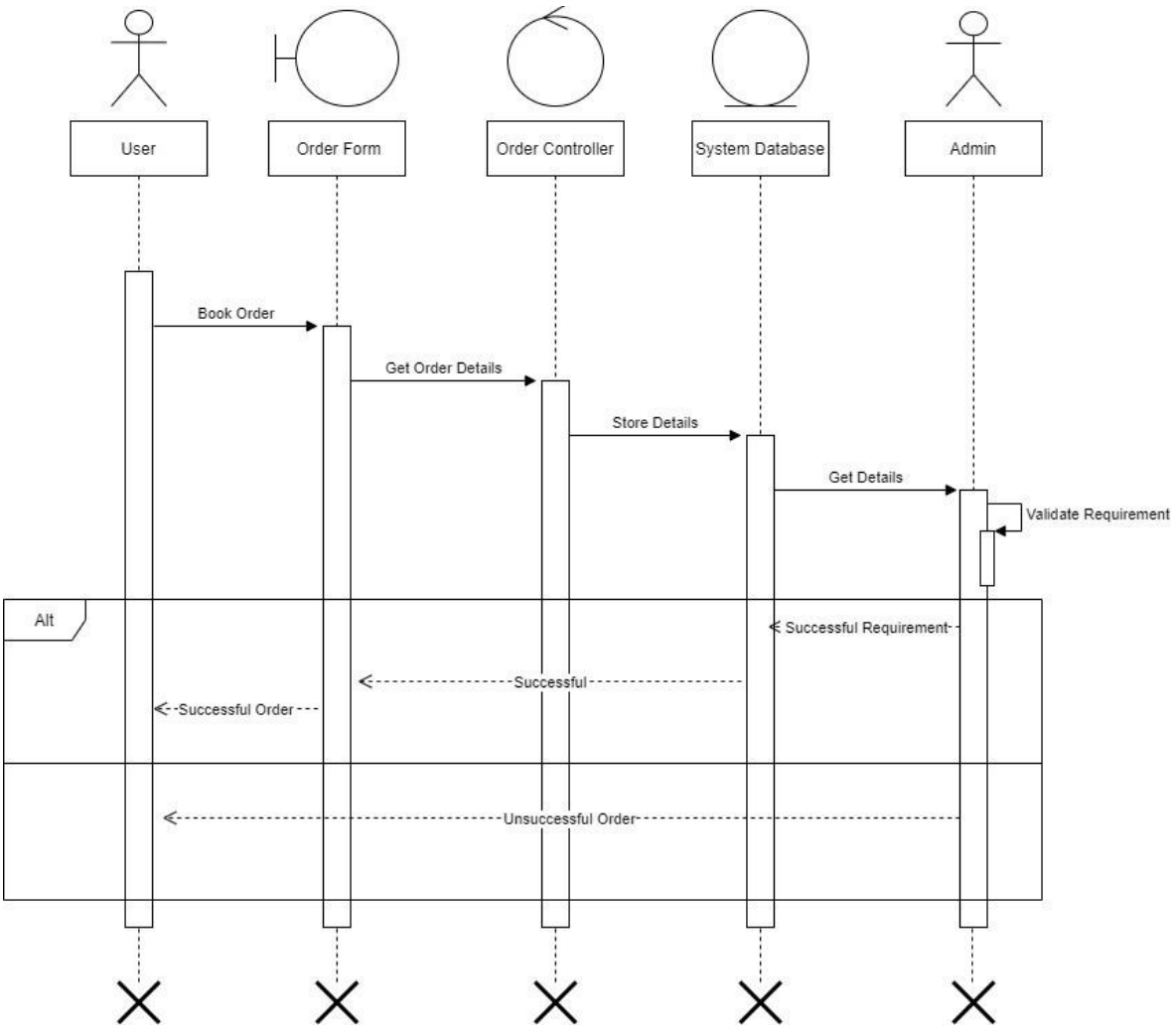
Sequence Diagram :-



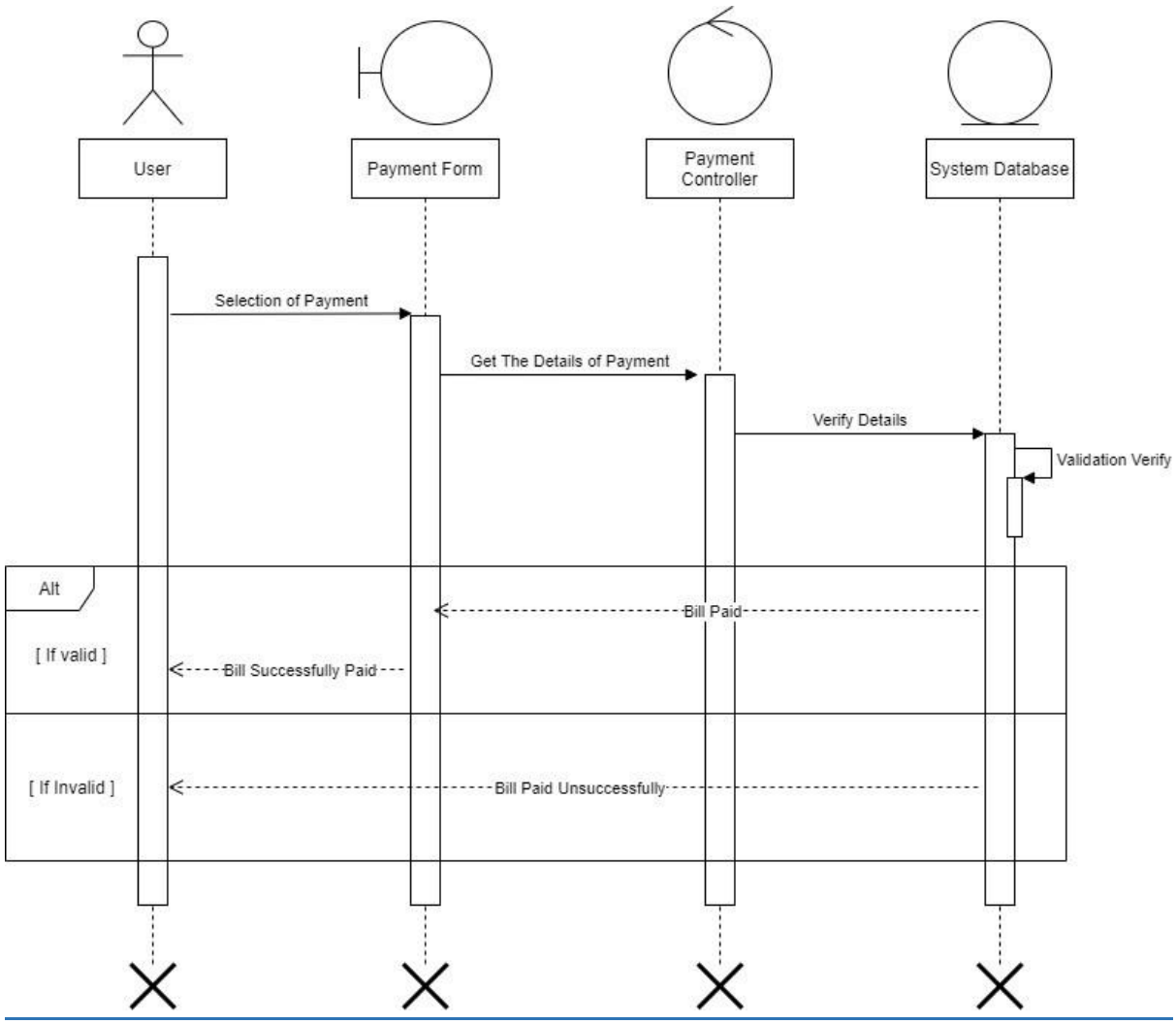
1. Login



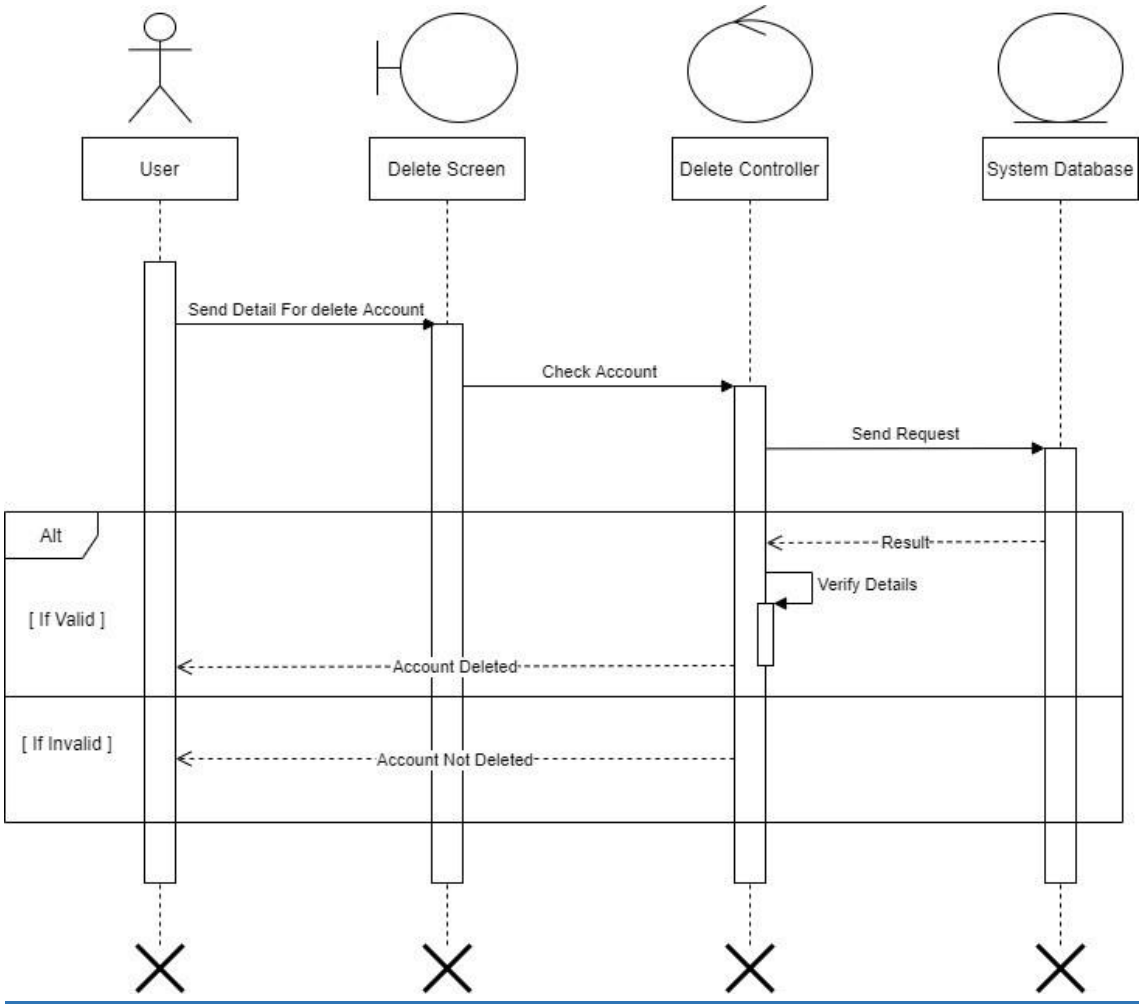
2.Registration :-


User
Order Form
Order Controller
System Database
Admin
Book Order
Get Order Details
Store Details
Get Details
Validate Requirement
Alt
Successful Requirement
Successful
Successful Order
Unsuccessful Order

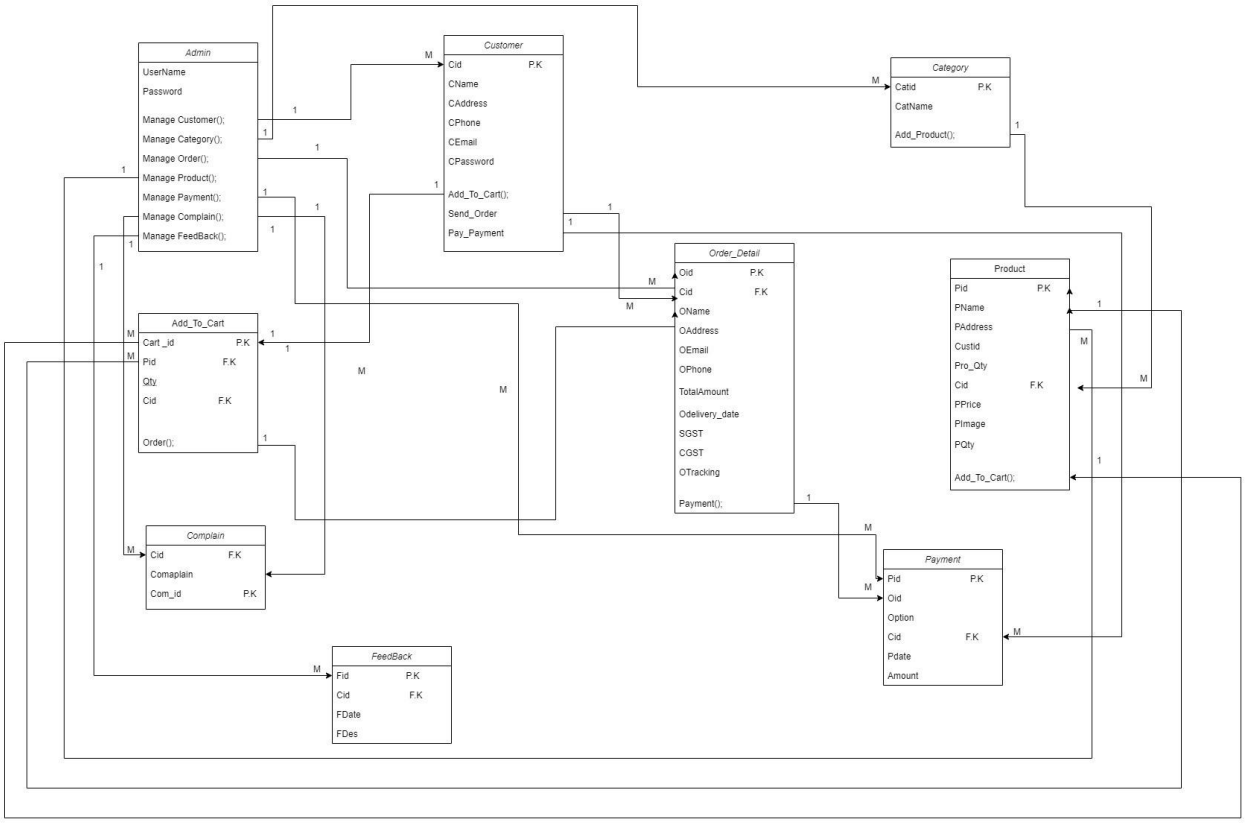
3.Product :-


User
Payment Form
Payment Controller
System Database
Selection of Payment
Get The Details of Payment
Verify Details
Validation Verify
Alt
[If valid]
Bill Paid
Bill Successfully Paid
[If Invalid]
Bill Paid Unsuccessfully

4.Payment:-

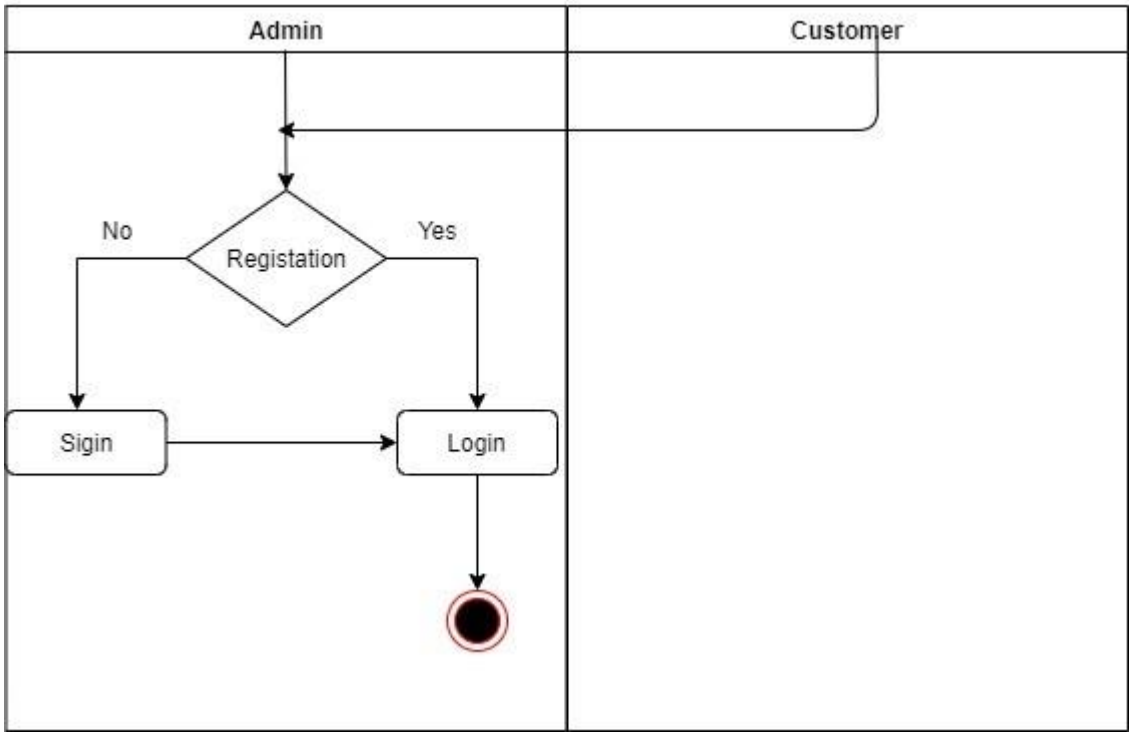


5. Delete Account:- Class diagram

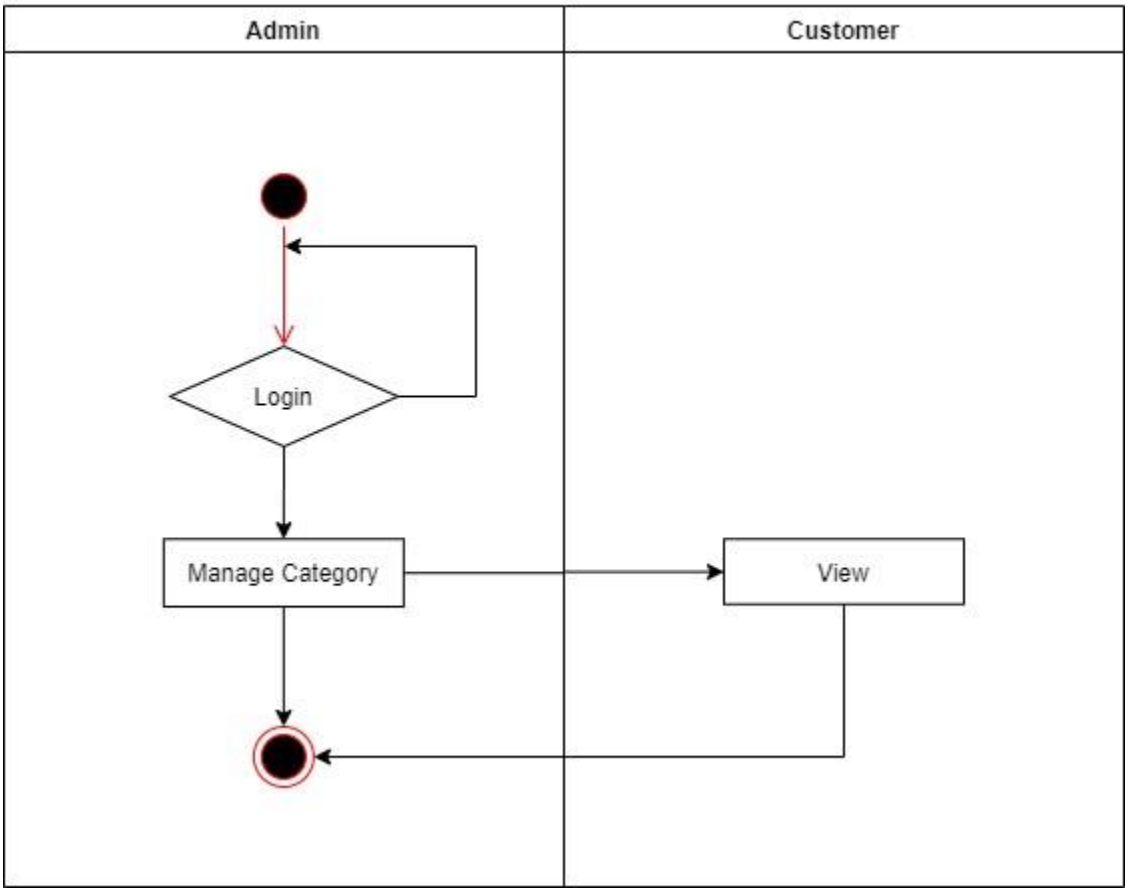


Activity Diagram:-

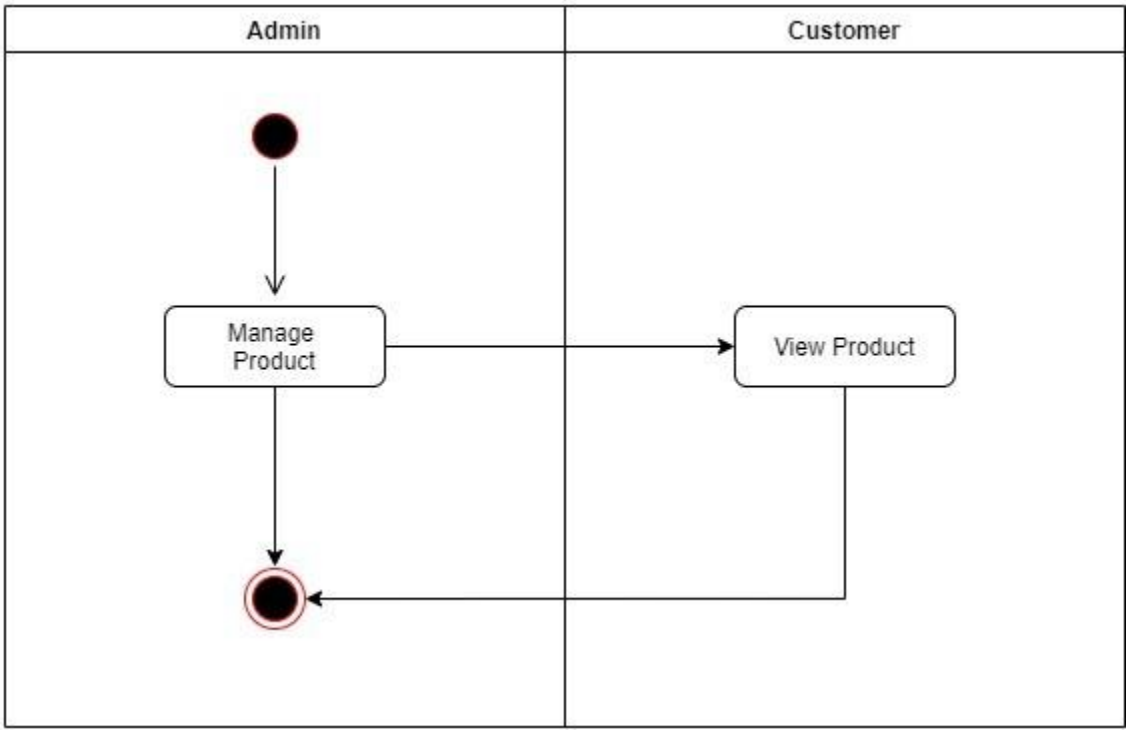
1. Admin



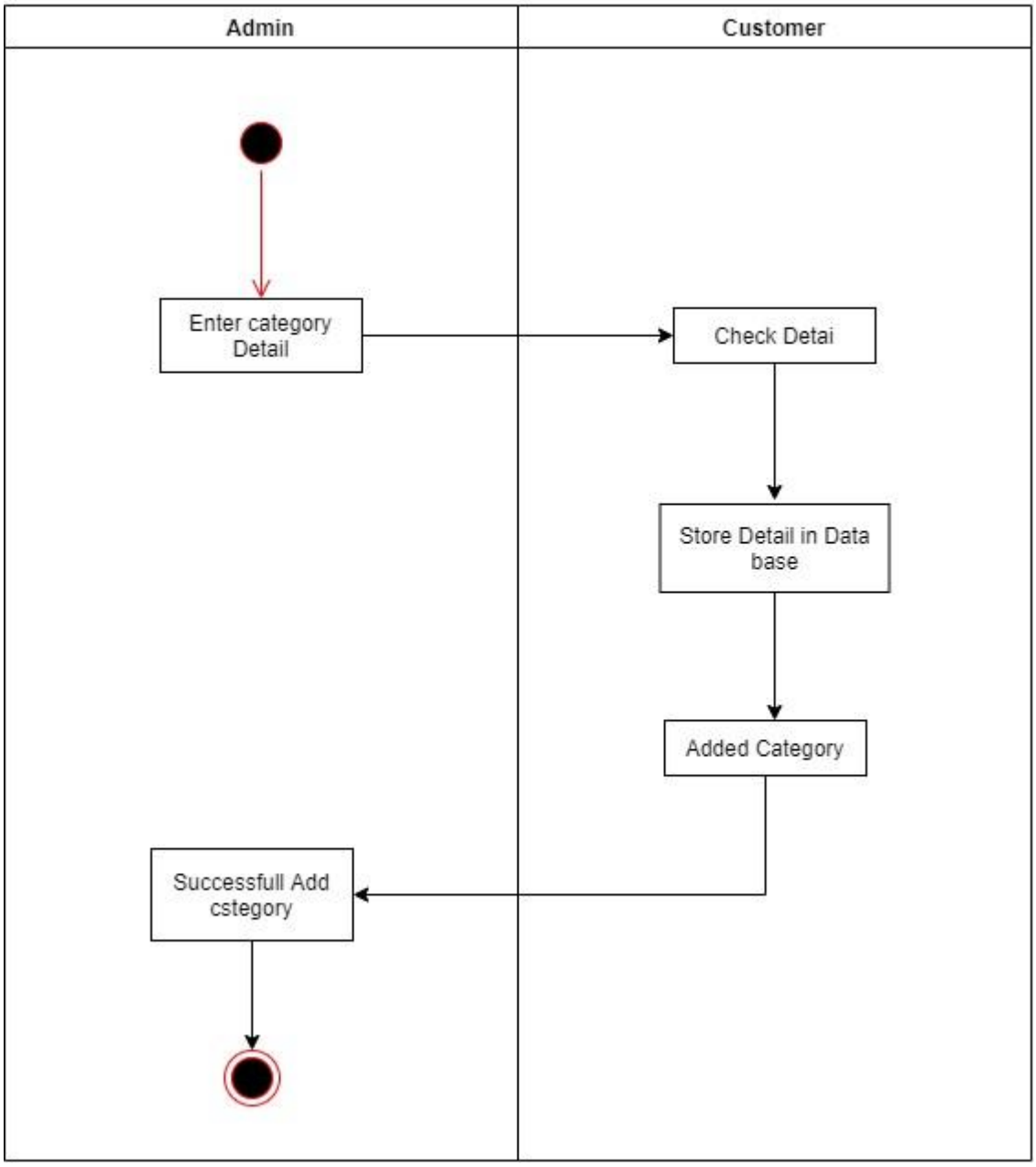
2 . Category



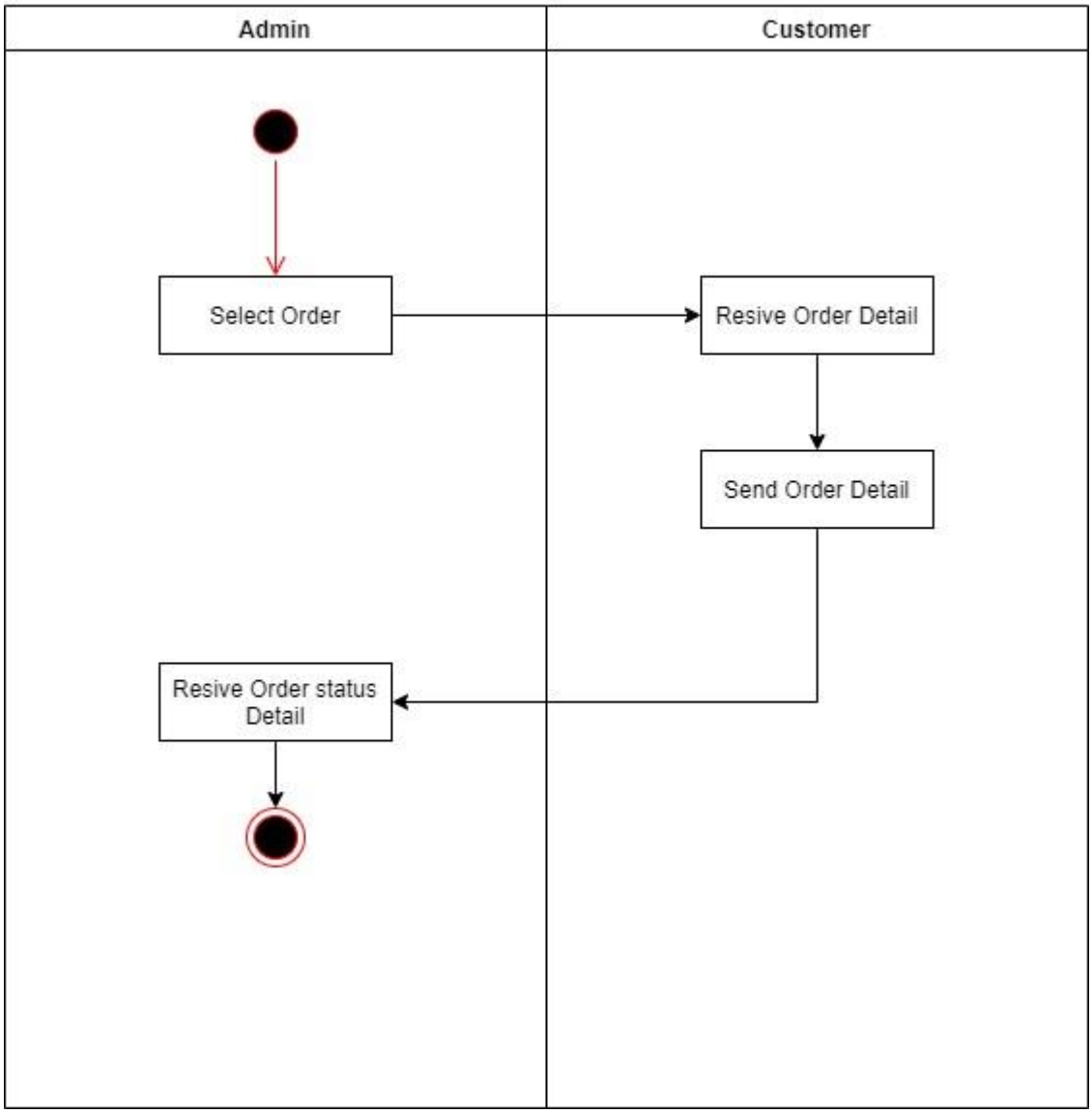
3. Product



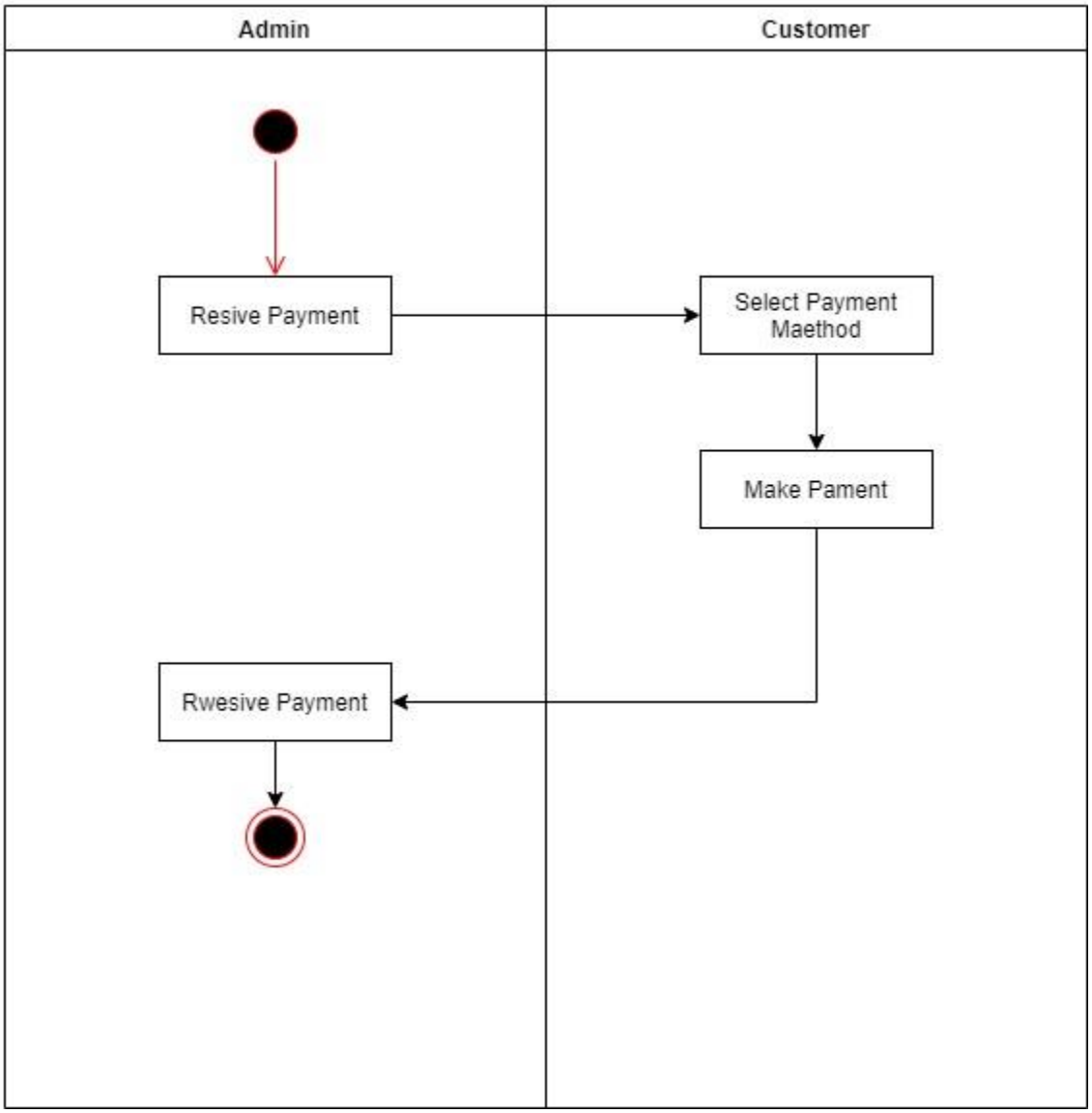
4 . Add to Cart


Admin
Customer
Enter category Detail
Check Detai
Store Detail in Data base
Added Category
Successfull Add cstegory

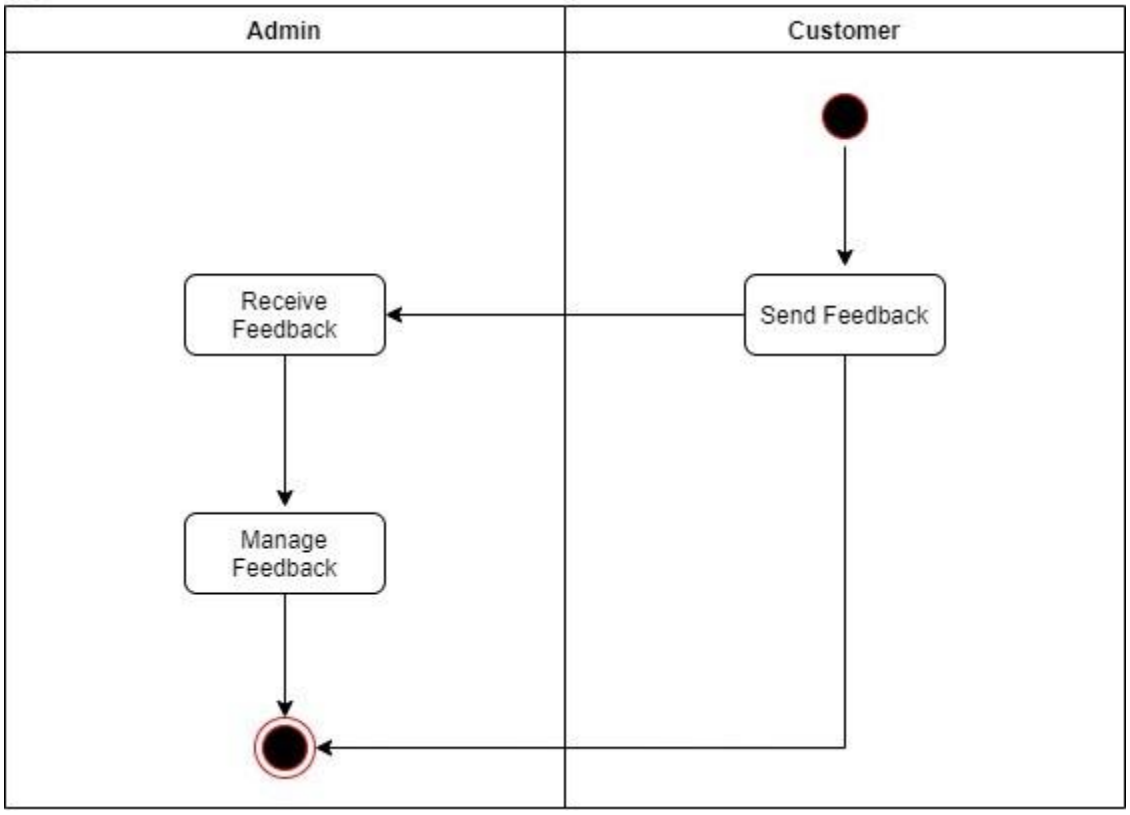
5 . Order


Admin
Customer
Select Order
Resive Order Detail
Send Order Detail
Resive Order status Detail

6. Payment


Admin
Customer
Resive Payment
Select Payment Maethod
Make Pament
Rwesive Payment

7, Feedback



Components Of UML Diagram Types:-

- Lifeline :-

  Lifelines in a UML Structure diagram are used to represent each instance in interaction.

- Actor :-

  Actor is used in UML to specify a role that is either played by a user or any system that is going to interact...

- Activity :-

  In Unified Modelling Language, Activity is a shape that is Used to Show a major task must be completed...

- State :-

  The shape of a State is used to denote the condition of an event or activity in the system. We also use it to...

Class diagram :-

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

What Is Class :-

A description of a group of objects all with similar roles in the system, which consists of:

- **Structural features** (attributes) define what objects of the class "know"

- Represent the state of an object of the class.
- Are descriptions of the structural or static features of a class.

- **Behavioral features** (operations) define what objects of the class "can do"

- Define the way in which objects may interact.
- Operations are descriptions of behavioral or dynamic features of a class.

Class Notation

A class notation consists of **three** parts:

1. **Class Name**

- The name of the class appears in the first partition.

2. **Class Attributes**
   - Attributes are shown in the second partition.
   - The attribute type is shown after the colon.
   - Attributes map onto member variables (data members) in code.
3. **Class Operations** (Methods)
   - Operations are shown in the third partition. They are services the class provides.
   - The return type of a method is shown after the colon at the end of the method signature.
   - The return type of method parameters is shown after the colon following the parameter name.
   - Operations map onto class methods in code.

Class Relationship :-

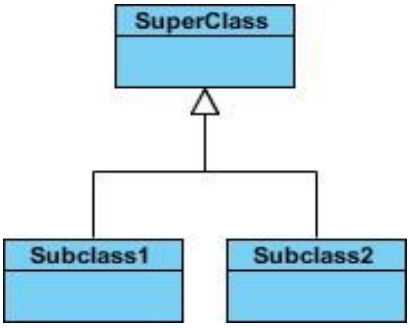
A class may be involved in one or more relationships with other classes. A relationship can be one of the following types.

Inheritance

- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class

:-

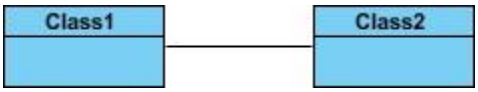
Graphical Representation:-



Simple Association:

- A structural link between two peer classes.
- There is an association between Class1 and Class2.
- A solid line connecting two classes.

Graphical Representation:-

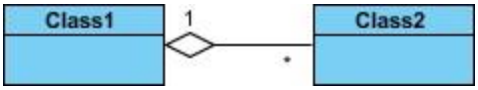


Aggregation :-

A special type of association. It represents a "part of" relationship.

- Class2 is part of Class1.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.
- A solid line with an unfilled diamond at the association end connected to the class of composite.

Graphical Representation:-

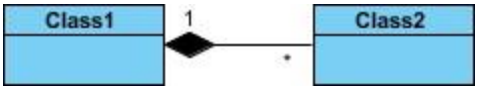


Composition :-

A special type of aggregation where parts are destroyed when the whole is destroyed.

- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite.

Graphical Representation:-



Dependency :-

- Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).

- Class1 depends on Class2
- A dashed line with an open arrow

Graphical Representation:-

Graphical Representation:-

Data Dictionary :-

- A data dictionary is a collection of descriptions of the data objects or items in a data model for the benefit of programmers and others who need to refer to them.

- A first step in analyzing a system of objects with which users interacts is to identify each object and its relationship to other

objects. This process is called data modeling and results in a picture of object relationships.

- After each data object or item is given a descriptive name, its relationship is described (or it becomes part of some structure that implicitly describes relationship), the type of data (such as text or image or binary value) is described, possible predefined values are listed, and a brief textual description is provided. This collection can be organized for reference into a book called a data dictionary.

- When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms.

A Data Dictionary Contains:-

- A data dictionary is a collection of descriptions of the data objects or items in a data model for the benefit of programmers and others who need to refer to them.

- A first step in analyzing a system of objects with which users interacts is to identify each object and its relationship to other objects. This process is called data modeling and results in a picture of object relationships.

- After each data object or item is given a descriptive name, its relationship is described (or it becomes part of some structure that implicitly describes relationship), the type of data (such as text or image or binary value) is described, possible predefined values are listed, and a brief textual description is provided. This collection can be organized for reference into a book called a data dictionary.

- When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms.

Data Dictionary:

Data Dictionary:

Table Name :	**ADMIN**
Table Description :	**This Table Contains Details About Admin**
Constraint :	**Not Null**

1.Admin

Username	It will store user name.
Password	It will store admin password

Data Dictionary:

Table Name :	**Customer Table**
Table Description :	**This Table Contains Details About Customer Table**
Constraint :	**Not Null**

2. Customer Table

Field Name	Data Type	Size	Constriction	Description
CId	Int	3	Primary key	It Will Store Customer Id
CName	Text	25	Not null	It Will Store Customer Cname
CAddress	Varchar 2	30	Not null	It Will Store Customer Caddress
CEmail	Varchar 2	15	Not null	It Will Store Customer CEmail
CPhone	Number	10	Not null	It Will Store Customer CPhone
CPassword	Int	10	Not null	It Will Store Customer CPassword

Data Dictionary:

Table Name :	**Category Table**
Table Description :	**This Table Contains Details About Category Table**
Constraint :	**Not Null**

3.Category Table

Field Name	Data Type	Size	Constriction	Description
CatId	Int	3	Primary Key /	It Will Store Category Id
CatName	Text	25		It Will Store Category name

Data Dictionary:

Table Name :	Product
Table Description :	This Table Contains Details About Product
Constraint :	Not Null

4.Product

Field Name	Data Type	Size	Constriction	Description
PId	Int	3	Primary Key	It Will Store Product Id
PName	Text	25	Not nol	It Will Store Product PName
PAddress	Varchar 2	30	Not nol	It Will Store Product PAddress
PImage	Photo		Not nol	It Will Store Product PImage
PPrice	Int	10	Not nol	It Will Store Product PPrice
PQty	Int	20	Not nol	It Will Store Product PQty
CatId	Int	3	Foreign Key	It Will Store Product CatId

Data Dictionary:

Table Name :	Add To Cart
Table Description :	This Table Contains Details About Add To Cart
Constraint :	Not Null

5.Add To Cart

Field Name	Data Type	Size	Constriction	Description
Cart Id	Int	3	Foreign Key	It Will Store Cart Id
Pid	Int	3	Foreign Key	It Will Store Pid
Cid	Int	3	Foreign Key	It Will Store Cid
Qty	Int	3	Not null	It Will Store Qty

Data Dictionary:

Table Name :	**Order Detail**
Table Description :	**This Table Contains Details About Order Detail**
Constraint :	**Not Null**

6.OrderDeatil

Field Name	Data Type	Size	Constriction	Description
OId	Int	3	Primary Key	It Will Store Order
Oname	Text	25	Not null	It Will Store Order Oname
Oaddress	Varchar 2	30	Not null	It Will Store Order Oaddress
OPhone	Number	10	Not null	It Will Store Order Ophone
Oemail	Int	20	Not null	It Will Store Order Oemail
OCId	Int	3	Not null	It Will Store Order OCid
OcatId	Int	3	Not null	It Will Store Order OcatId

Oamount	Int	20	Not null	It Will Store Order Oamount
SGst	Varchar	20	Not null	It Will Store Order SGst
CGst	Varchar	20	Not null	It Will Store Order CGst
TotalAmount	Int	20	Not null	It Will Store Order TotalAmount
ODate	Date	8	Not null	It Will Store Order ODate
IsAccept	Boolean	20	Not null	It Will Store IsAccept

Data Dictionary:

Table Name :	**Payment**
Table Description :	**This Table Contains Details About Payment**
Constraint :	**Not Null**

7.Payment

Field name	Data type	Size	Constriction	Description
PId	Int	3	Primary Key	It Will Store PId
OId	Int	3	Foreign Key	It Will Store OId
Amount	Int	10		It Will Store Amount
PType	-------			It Will Store PType
PDate	Date	8	Not null	It Will Store PDate

Data Dictionary:

Table Name :	**Feedback**
Table Description :	**This Table Contains Details About Feedback**
Constraint :	**Not Null**

8.Feedback

Field name	Data Type	Size	Constriction	Description
Fid	Int	3	Primary Key	It Will Store Fid
FDes	Varchar	50		It Will Store FDes
FDate	Date	8		It Will Store FDate
Cid	Int	3	Foreign Key	It Will Store Cid