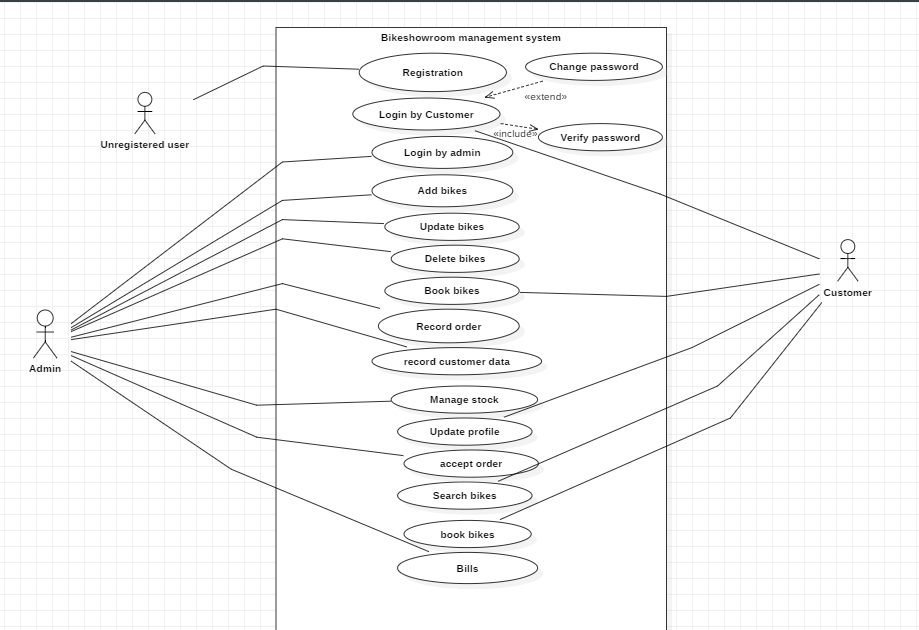
Computing Project

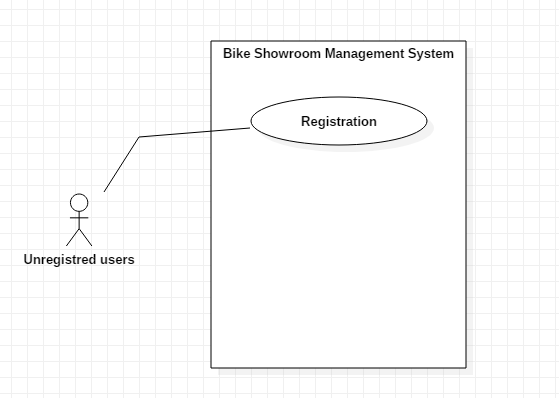
# Use Case Diagram.

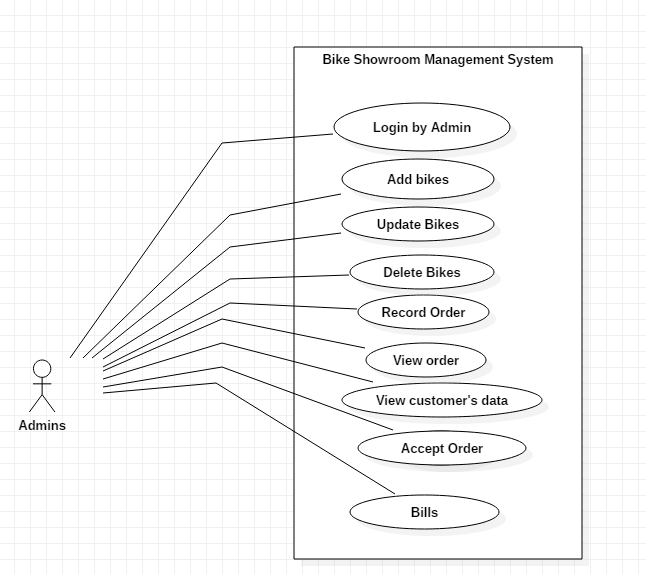
Use case Diagram shows an activity of users in the system. As shown below.



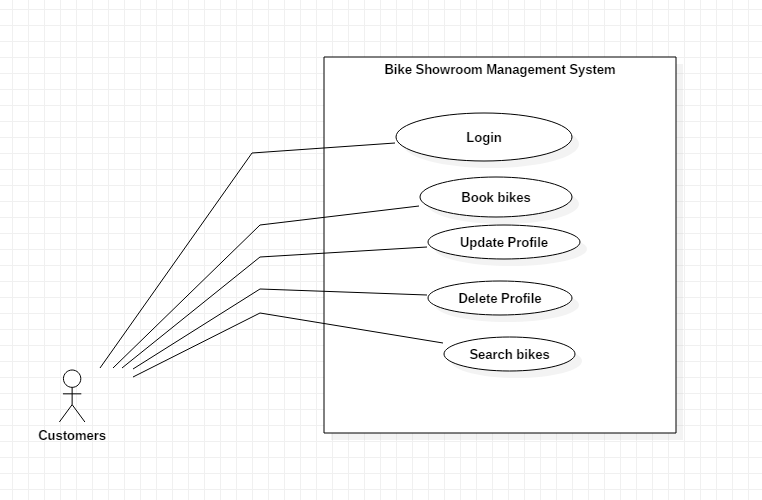
# Users in use case

**Unregistered users.**

  
2. Use Case by Admins.



3. Use case by customers.

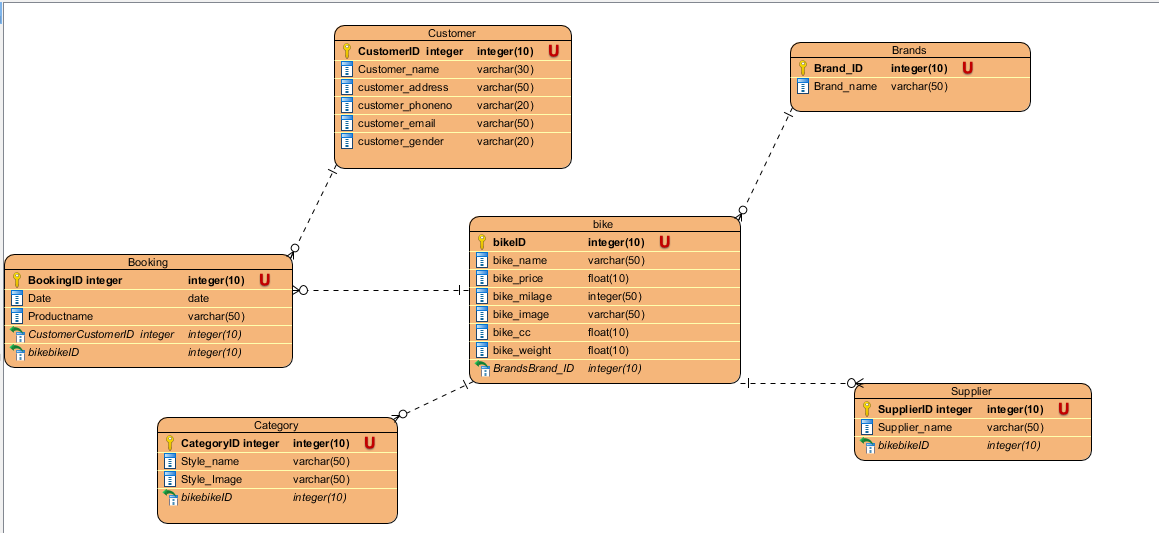


# Database Design

A database design shows data models in a system. It shows how data are handled in a system. A poor database design may fail the system for future development and improvements because as the system grows, new kind of data may come which may have relations with previous type of data, but the previous design is so bad that the relation may not be maintained which makes it very hard to implement it. Therefore, a database design must be very systematic and carefully designed.

## ER Diagram

Entity Relationship Diagram is a database design tool which contains different entities and their relationship with each other. An entity contains certain similar component of data.



ER diagram

Justification

* To visualize database design ideas, we have a chance to identify the mistakes and design flaws, and to make correction before executing the changes in database.
* By visualizing a database schema with an ERD, we have a full picture of the entire database schema. You can easily locate entities, view their attributes and to identify the relationships they have with others.

Advantage

* It is very simple if we know relationship between entities and attributes.
* It is better visual representation.
* It is an effective communication tool for database designer.

Disadvantage

* It has limited constraints and specification.
* Information can be hidden in ER model.
* It is difficult to show data manipulation in ER model.

# Meta Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User table | | | | | |
| Column Name | Type | Length | Key | Null | Description |
| Customer ID | Int | 10 | PK | No | Unique identification of customer |
| Customer Name | Varchar | 30 |  | No | Customer’s Name |
| Customer’s address | Varchar | 50 |  | No | Customer’s address |
| Customer’s email | Varchar | 50 |  | No | Customer’s email |
| Customer’s phone | Varchar | 20 |  | No | Customer’s phone |
| Customer’s gender | Varchar | 20 |  | No | Customer’s gender |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product table | | | | | |
| Column Name | Type | Length | Key | Null | Description |
| Bike ID | Int | 10 | PK | No | Unique identification of Bike |
| Bike Name | Varchar | 30 |  | No | Bike Name |
| Bike price | Float | 10 |  | No | Bike price |
| Bike CC | Float | 10 |  | No | Bike CC |
| Bike mileage | Float | 10 |  | No | Bike mileage |
| Bike weight | Float | 10 |  | No | Bike weight |
| Bike Image | Varchar | 20 |  | No | Bike’s Image |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Booking table | | | | | |
| Column Name | Type | Length | Key | Null | Description |
| Booking ID | Int | 10 | PK | No | Unique identification of Booking |
| Date | Date |  |  | No | Booking Date |
| Product name | Varchar | 30 |  | No | Product Name |
| Customer ID | Int | 10 | FK | No | Customer’s ID |
| Bike ID | Int | 10 | FK | No | Bike ID |
| Bike Image | Varchar | 50 |  | No | Bike Image |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category table | | | | | |
| Column Name | Type | Length | Key | Null | Description |
| Category ID | Int | 10 | PK | No | Unique identification of Category |
| Style Name | Varchar | 50 |  | No | Bike Style name |
| Bike ID | Int | 10 | FK | No | Bike ID |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Brand table | | | | | |
| Column Name | Type | Length | Key | Null | Description |
| Brand ID | Int | 10 | PK | No | Brand ID |
| Brand Name | Varchar | 50 |  | No | Bike Brand name |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Supplier table | | | | | |
| Column Name | Type | Length | Key | Null | Description |
| Supplier ID | Int | 10 | PK | No | Supplier ID |
| Supplier Name | Varchar | 50 |  | No | Bike Supplier name |
| Brand ID | Int | 10 | FK | No | Brand ID |

# Activity Diagram

Activity diagram is a behavioural design tool which shows the flow of activity in an application. An activity can be drawn using activities, swim lanes, initial and final point, conditions, signals, join and fork nodes, merge events, action flow, object flow, time events, interruptive edge etc.

I am using this design because it helps me, or any other designer understand how different functionality in my system will be working. Since I will be implementing using these designs, the implementation process will be a lot faster and easier.

Justification

* To draw the activity flow of a system.
* To describe the sequence from one activity to another.
* To describe the parallel, branched and concurrent flow of the system

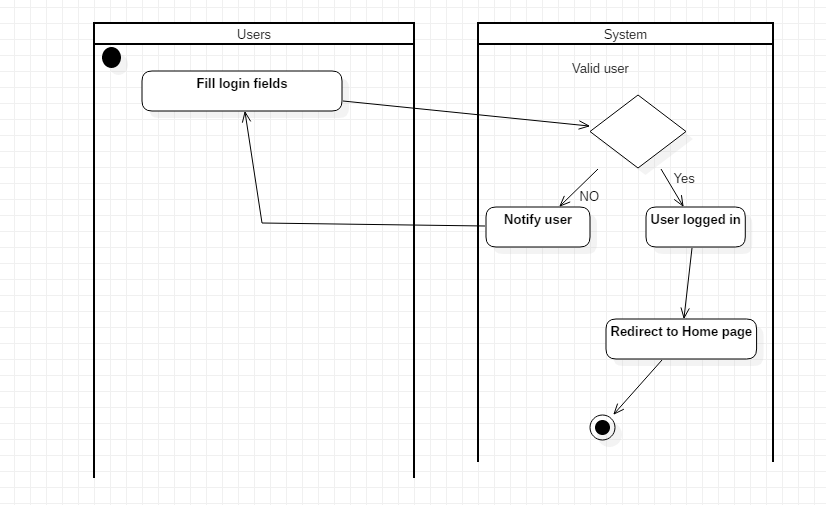
Advantage

* Since it is the most user-friendly diagram. So, generally regarded as an essential tool.
* It helps to display multiple conditions and actors within a work flow using swim lanes.
* These diagrams are normally easily comprehensive for both analysts and stakeholder.

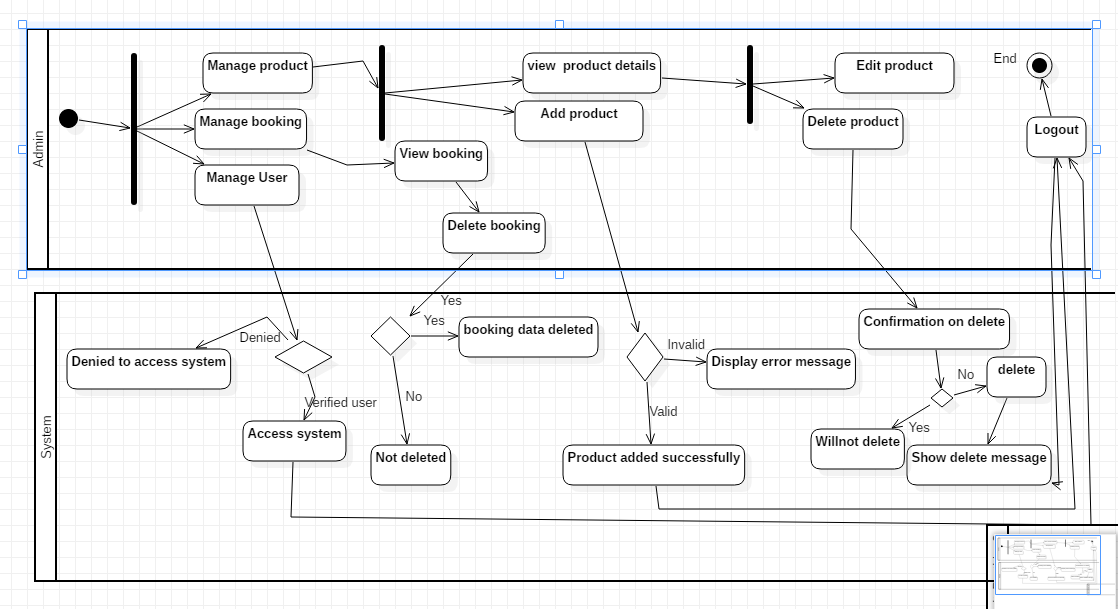
Disadvantage

* These diagrams can lead the over complex which might affect the user-friendly nature.
* These diagrams do not give the detail about how object behave or collaborate.

## Login Activity



In login activity, user and the system interact with each other to perform login. A user submits his/her login to system and the system checks, if they are valid then the validity will decide the success of login.



# Requirements

|  |  |  |
| --- | --- | --- |
| Functional and non-functional determination | Requirements | MoSCoW prioritization |
| F(R1) | Registration for customer | M |
| F(R2) | Login for admin | M |
| F(R3) | Login for customer | M |
| F(R4) | Add to stock | M |
| F(R5) | Update stock | M |
| F(R6) | Delete stock | M |
| F(R7) | View bookings | M |
| F(R8) | View members | M |
| F(R9) | Search for bikes | M |
| F(R10) | View new launches bikes | M |
| F(R11) | View upcoming models | S |
| F(R12) | View popular models | S |
| F(R13) | View according to brands | C |
| F(R14) | Show price of bikes | M |
| NF(R15) | User friendly interface | M |
| F(R16) | Generate bills | S |
| F(R17) | Update profile of customer | M |
| NF(R18) | Website security | M |
| NF(R19) | Reliable | M |
| F(R20) | Supplier’s information | M |
| F(R21) | Second hand bikes | C |
| F(R22) | Bike’s style | C |
| F(R23) | Feedback from clients | M |
| F(R24) | Book bike | M |

## Prioritization

Prioritizing requirements is a very important step in a software application. It must be done during the requirement analysis phase because it helps us manage requirements and resources. Priorities can help responding to new changes that may arise later. Prioritization is also helpful when releasing software in phases. (Wheatcraft, 2012)

I am going to use MoSCoW prioritization technique for prioritizing requirements in my project. The reason I am going to use MoSCoW is that the problem with simply saying that requirements are of High, Medium or Low importance is that the definitions of these priorities are missing. The term MoSCoW stands for:

* **Must Have**

These provide the Minimum Usable Subset (MUS) of requirements which the project guarantees to deliver

* **Should Have**

Important but not vital.

* **Could Have**

Wanted or desirable but less important.

* **Won’t Have this time**

These are requirements which the project team has agreed it will not deliver or perhaps not appropriate at that time.