**UML:**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

* UML stands for **U**nified **M**odeling **L**anguage.
* UML is a pictorial language used to make software blue prints.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design

**Goals of UML:**

There are a number of goals for developing UML but the most important is to define some general purpose modeling language which all modelers can use and also it needs to be made simple to understand and use.

At the conclusion the goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today’s complex environment.

**Things** are the most important building blocks of UML. Things can be:

* Structural
* Behavioral
* Grouping
* Annotational

# 1. Structural things:

The **Structural things** define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

## Class:

Class represents set of objects having similar responsibilities.

class

## Interface:

Interface defines a set of operations which specify the responsibility of a class.

Interface

## Collaboration:

Collaboration defines interaction between elements.

Collaboration

## Use case:

Use case represents a set of actions performed by a system for a specific goal.

Use case

## Component:

Component describes physical part of a system.

Component

## Node:

A node can be defined as a physical element that exists at run time.



# 2. Behavioral things:

**A behavioral thing** consists of the dynamic parts of UML models. Following are the behavioral things:

## Interaction:

Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

Interaction

## State machine:

State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.



# 3. Grouping things:

**Grouping things** can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available:

## Package:

Package is the only one grouping thing available for gathering structural and behavioral things.



# 4.Annotational things:

**Annotational things** can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. **Note** is the only one Annotational thing available.

## Note:

A note is used to render comments, constraints etc of an UML element.

Note

# Relationship :

Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

## 1. Dependency:

Dependency is a relationship between two things in which change in one element also affects the other one.

Dependency

## 2. Association:

Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.

Association

## 3. Generalization:

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.

Generalization

## 4. Realization:

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in case of interfaces.

Realization

# UML Diagrams:

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

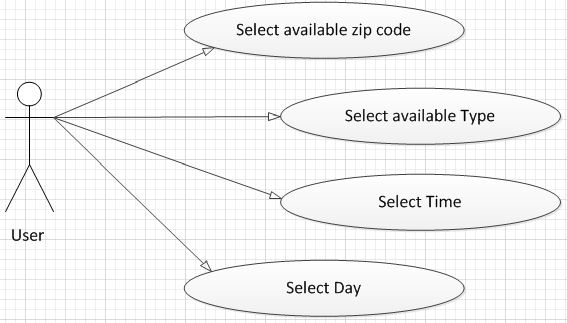
The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it a complete one.

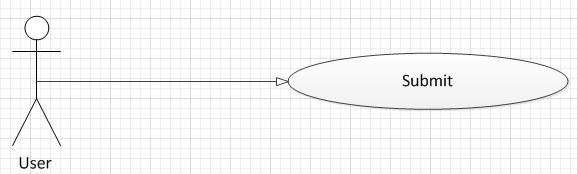
UML includes the following diagrams and the details are described below:

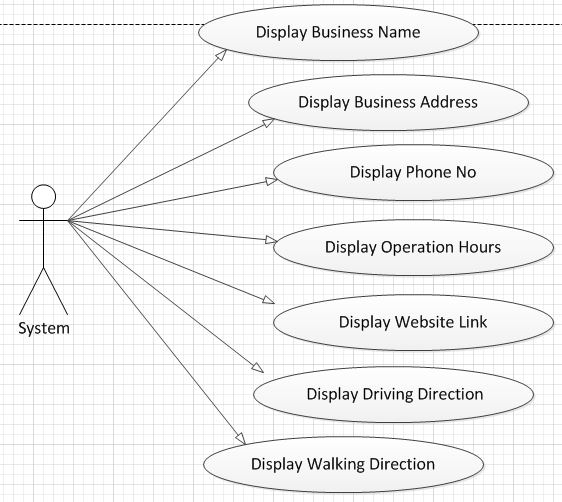
**1. Use-case diagram**

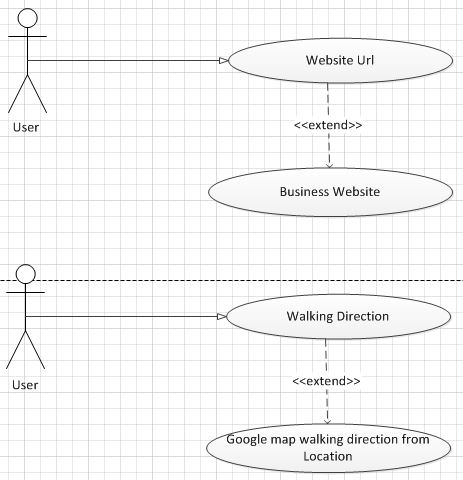
A use case illustrates a unit of functionality provided by the system. The main purpose of the use-case diagram is to help development teams visualize the functional requirements of a system, including the relationship of "actors" (human beings who will interact with the system) to essential processes, as well as the relationships among different use cases. Use-case diagrams generally show groups of use cases — either all use cases for the complete system, or a breakout of a particular group of use cases with related functionality (e.g., all security administration-related use cases). To show a use case on a use-case diagram, you draw an oval in the middle of the diagram and put the name of the use case in the center of, or below, the oval. To draw an actor (indicating a system user) on a use-case diagram, you draw a stick person to the left or right of your diagram (and just in case you're wondering, some people draw prettier stick people than others). Use simple lines to depict relationships between actors and use cases.A use-case diagram is typically used to communicate the high-level functions of the system and the system's scope.

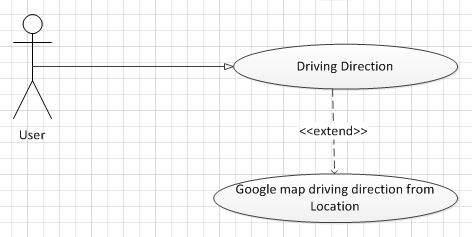
**USE CASE DIAGRAMS:**

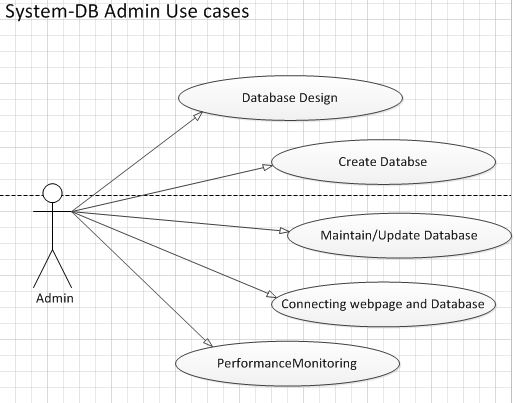










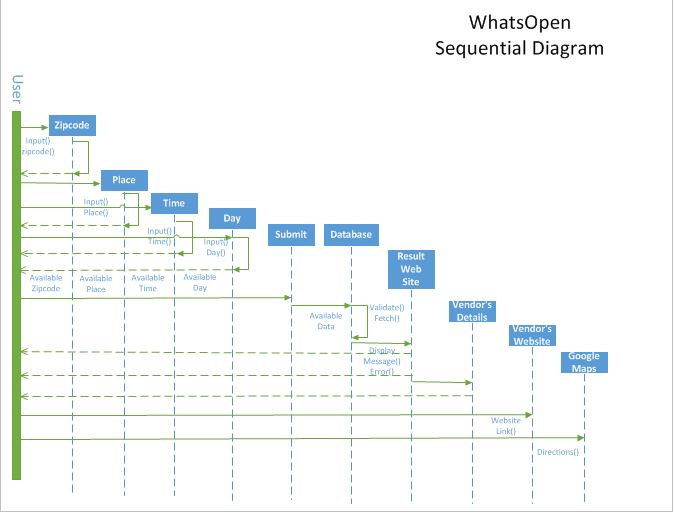


**2. Sequence diagram**

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They are almost self explanatory; they show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects.

A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent.

**SEQUENCE DIAGRAM:**

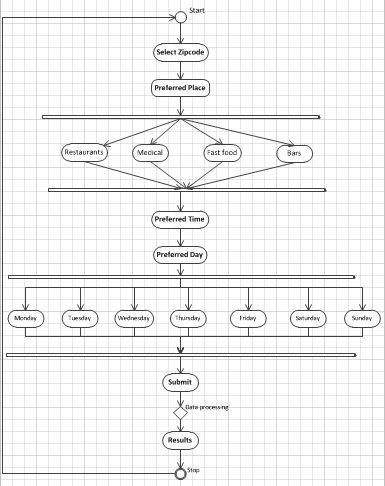
****

**3. Activity diagram**

Activity diagrams show the procedural flow of control between two or more class objects while processing an activity. Activity diagrams can be used to model higher-level business process at the business unit level, or to model low-level internal class actions. In my experience, activity diagrams are best used to model higher-level processes, such as how the company is currently doing business, or how it would like to do business. This is because activity diagrams are "less technical" in appearance, compared to sequence diagrams, and business-minded people tend to understand them more quickly.

An activity diagram's notation set is similar to that used in a state-chart diagram. Like a state-chart diagram, the activity diagram starts with a solid circle connected to the initial activity. The activity is modeled by drawing a rectangle with rounded edges, enclosing the activity's name. Activities can be connected to other activities through transition lines, or to decision points that connect to different activities guarded by conditions of the decision point. Activities that terminate the modeled process are connected to a termination point (just as in a state-chart diagram). Optionally, the activities can be grouped into swim lanes, which are used to indicate the object that actually performs the activity.

**ACTIVITY DIAGRAM:**



**Class Diagram**: 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. The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling.

