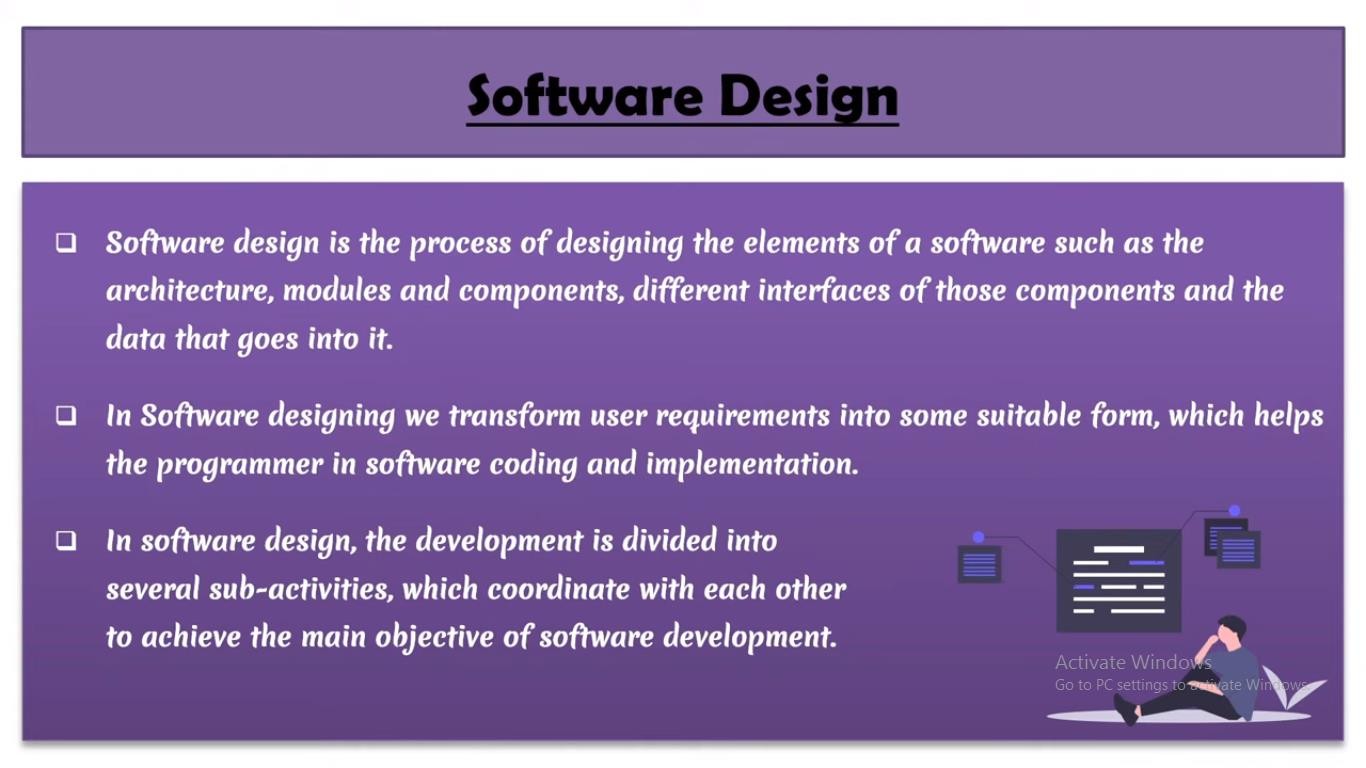
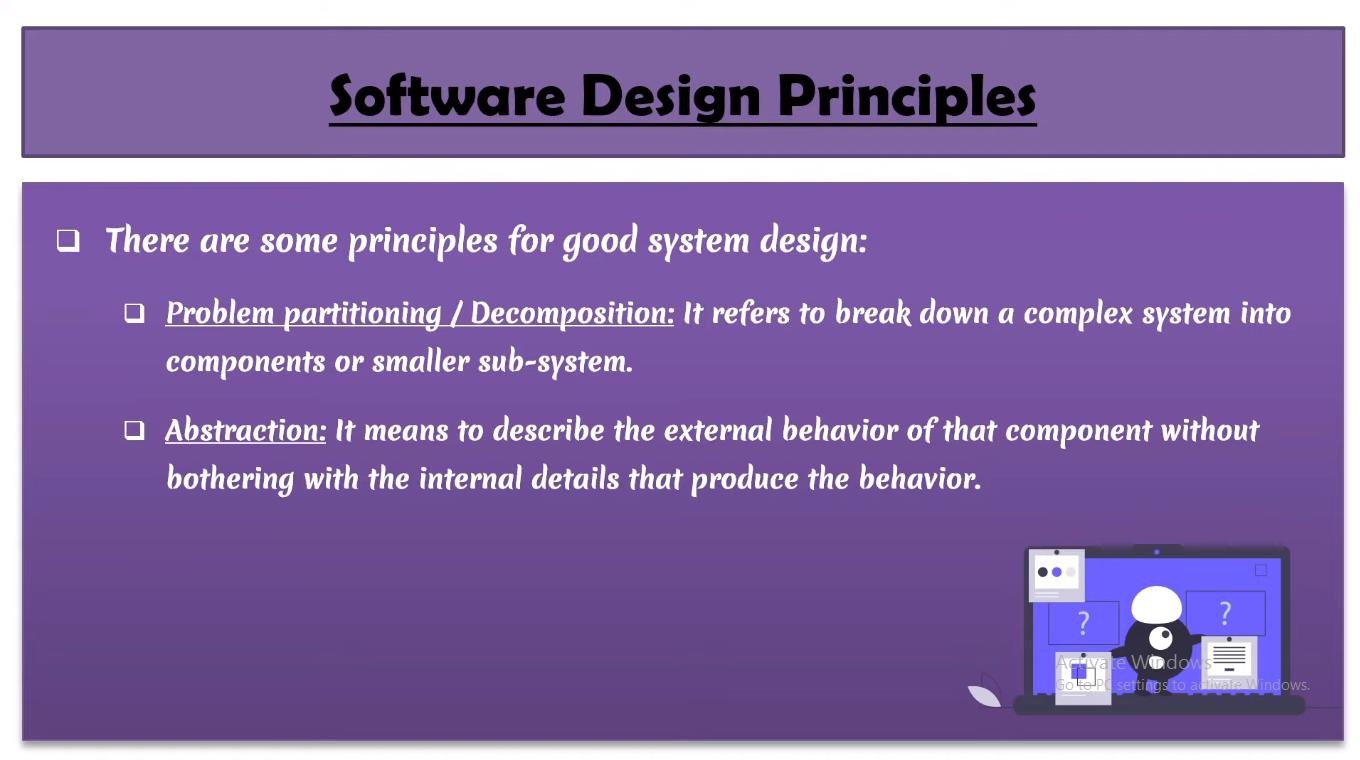
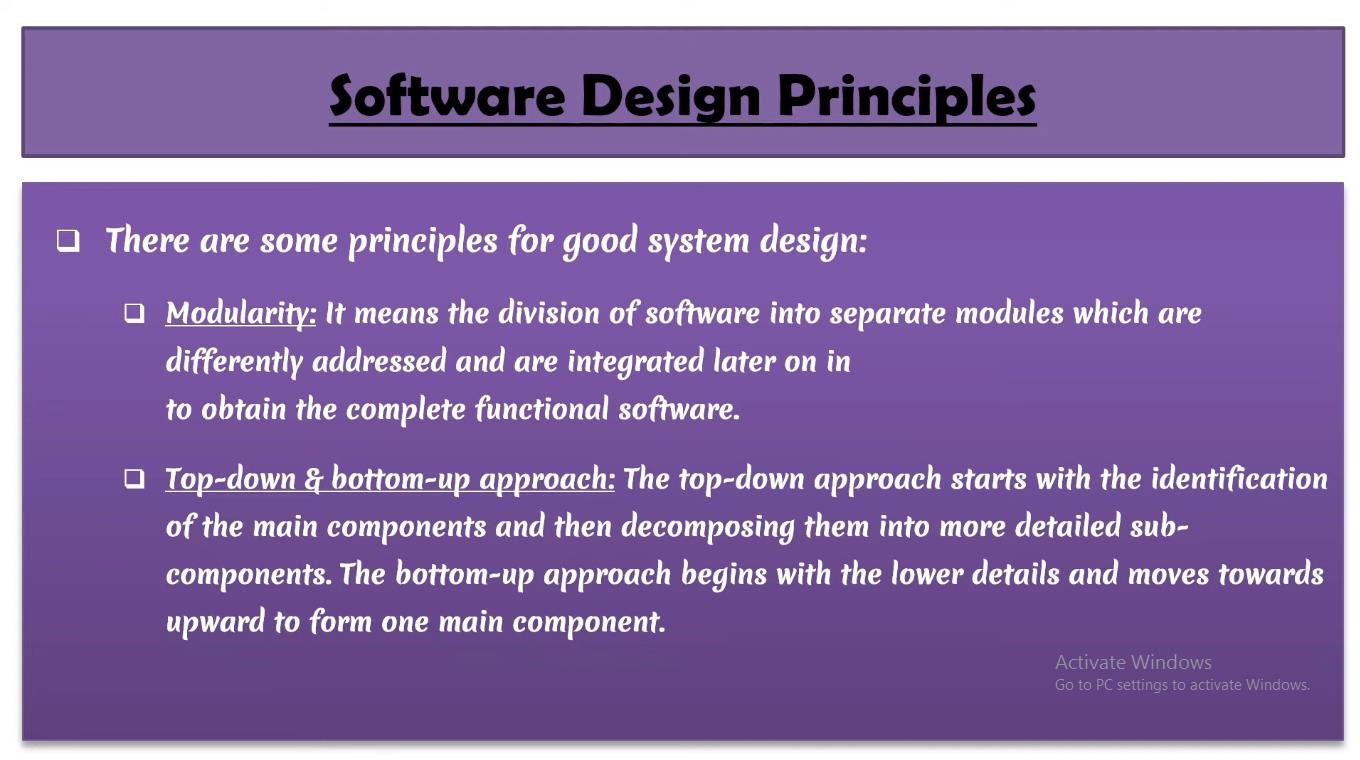
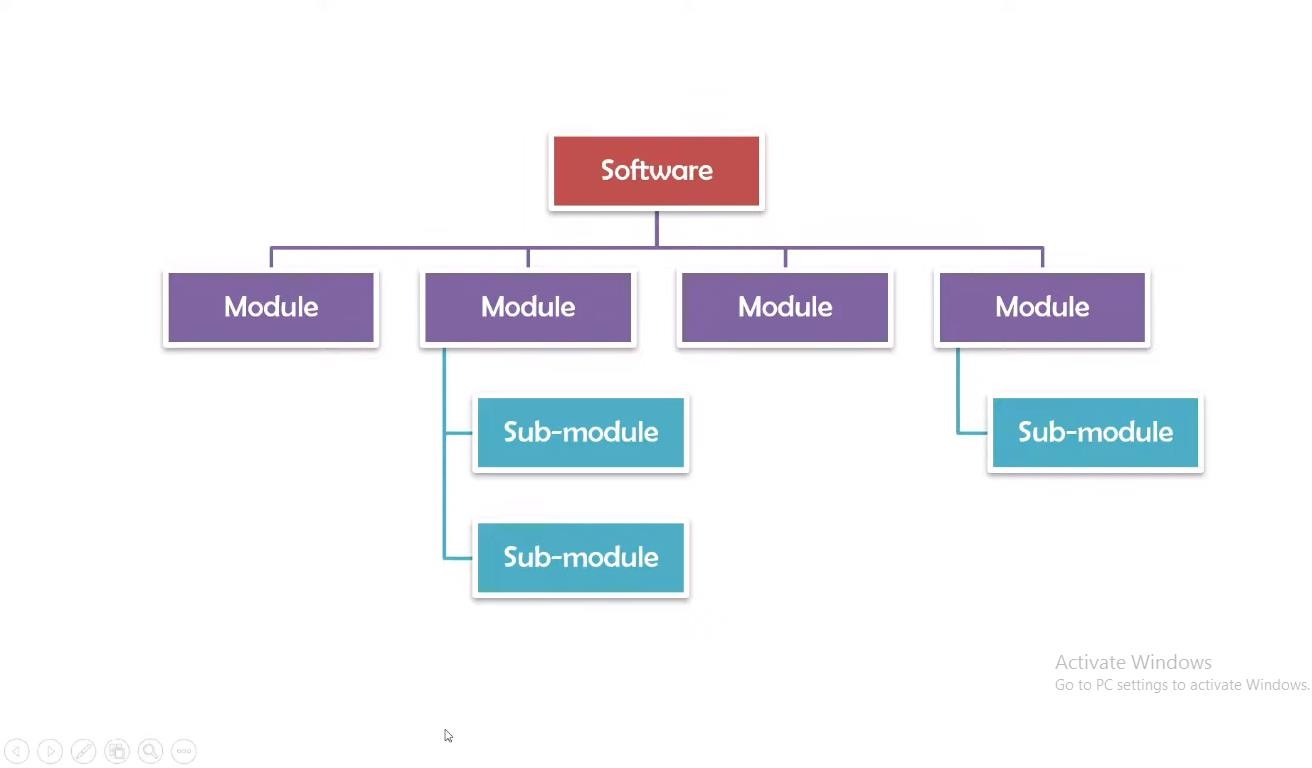
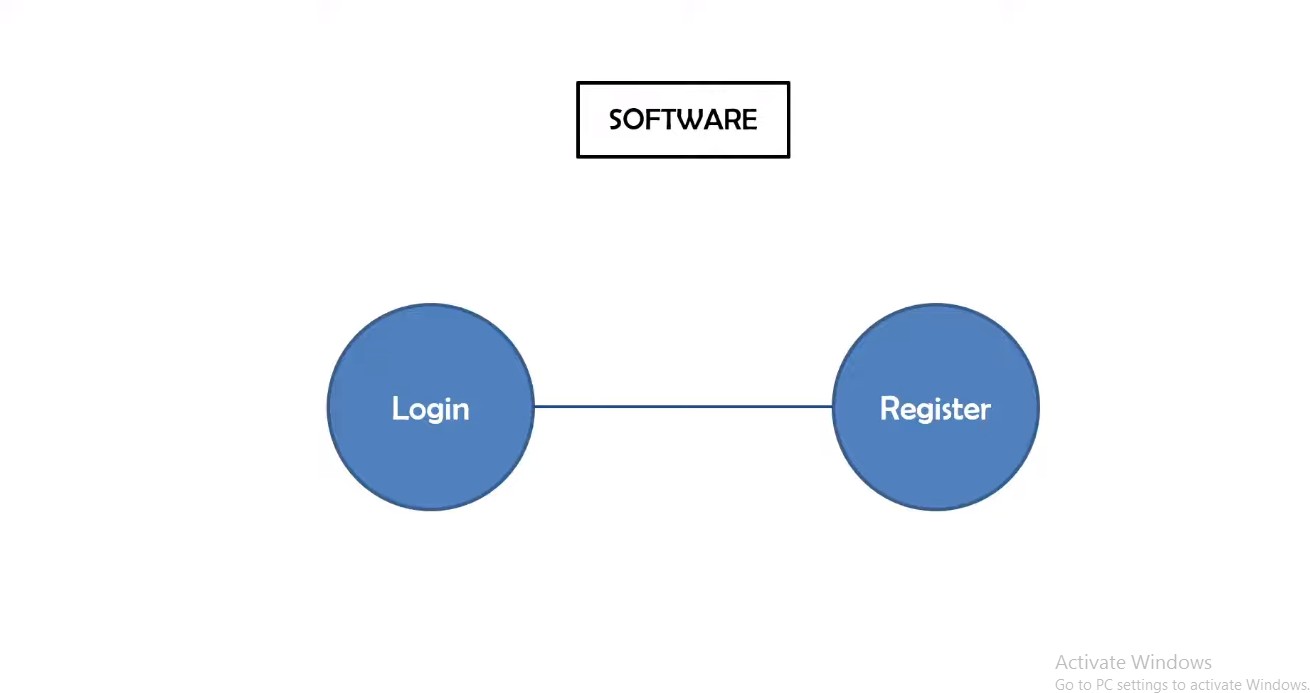
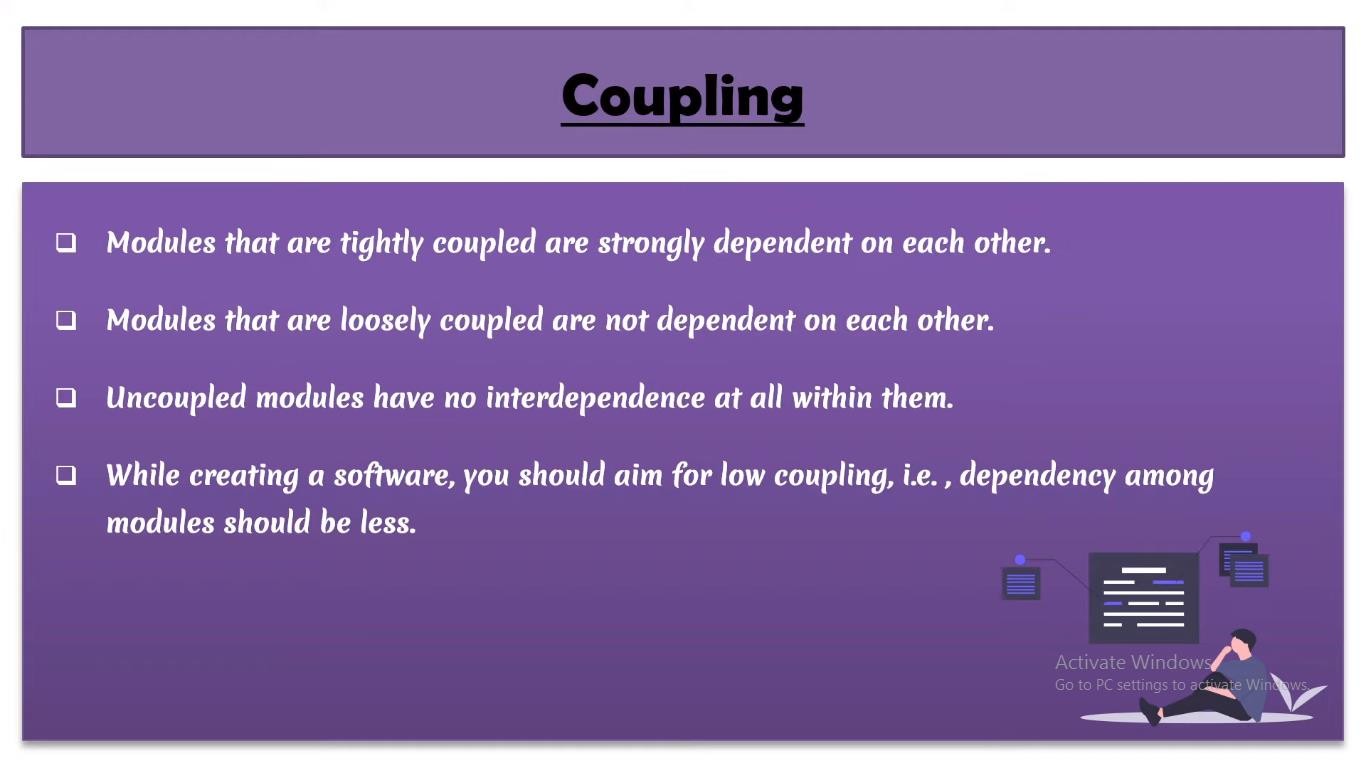
` 

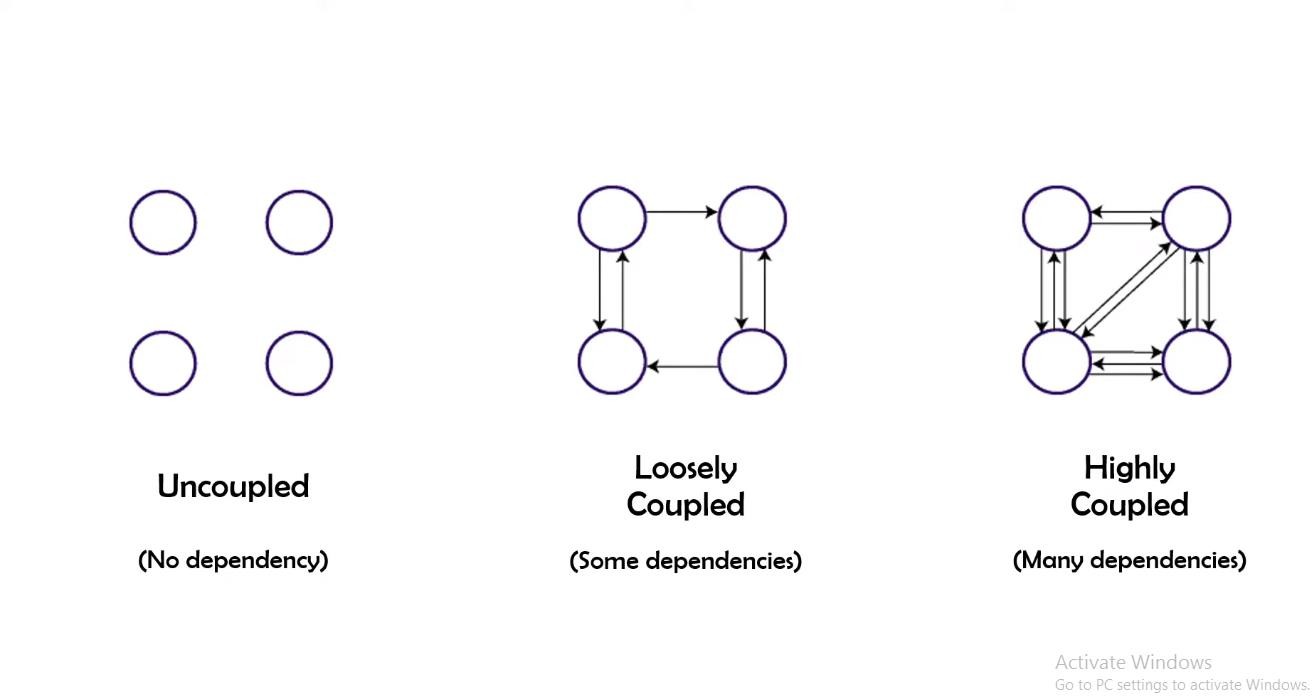


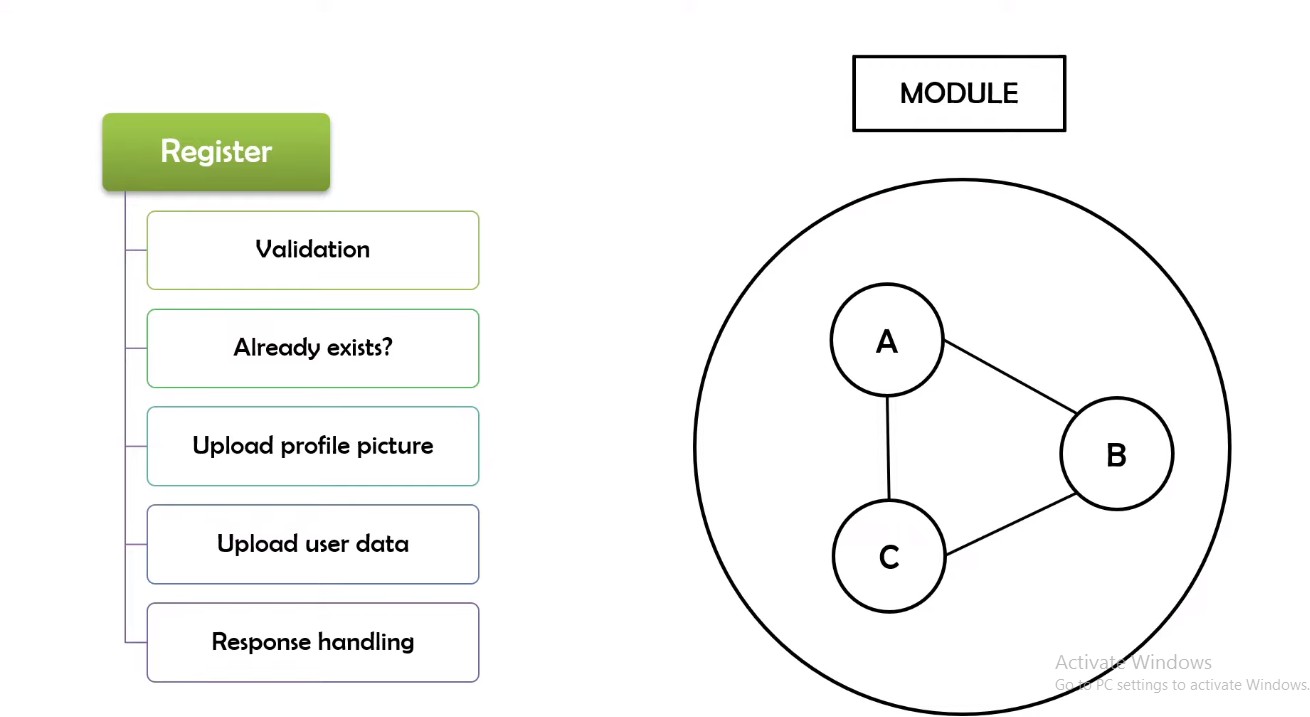


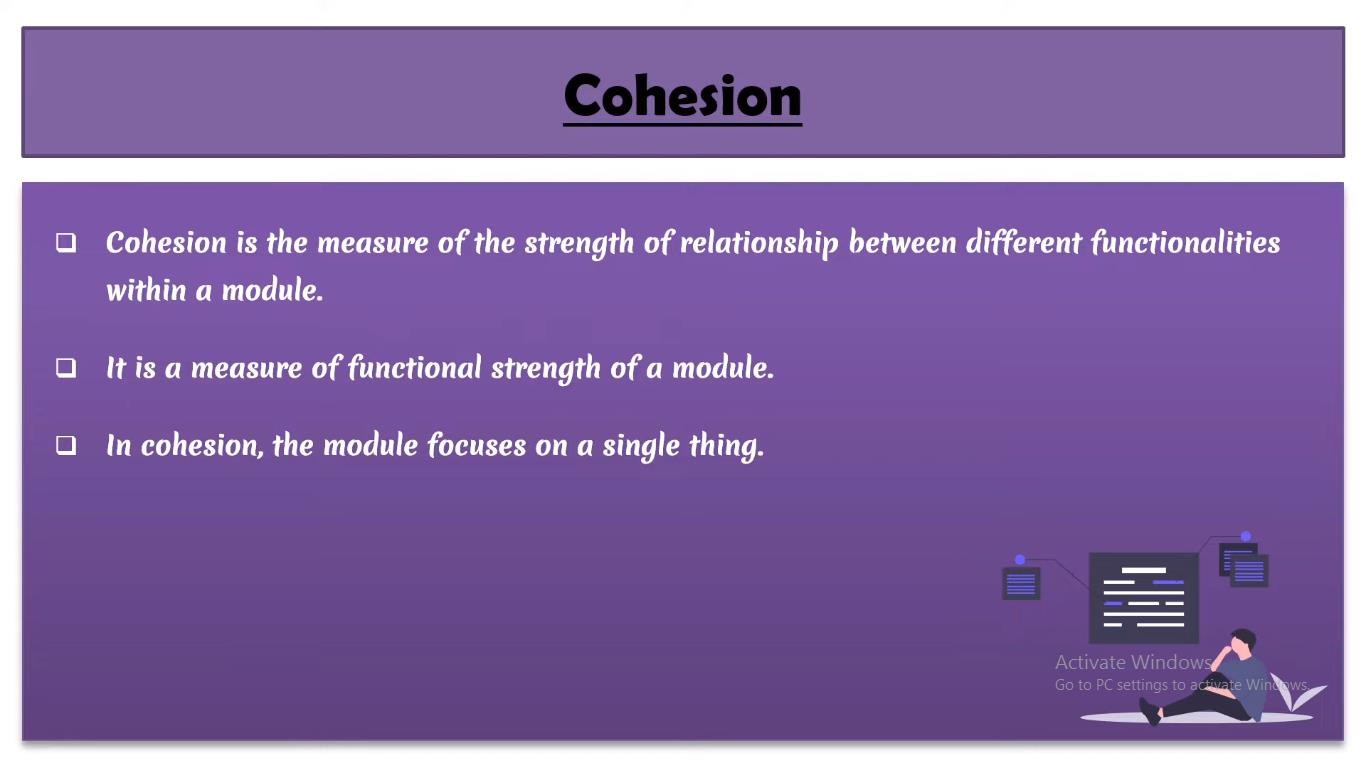


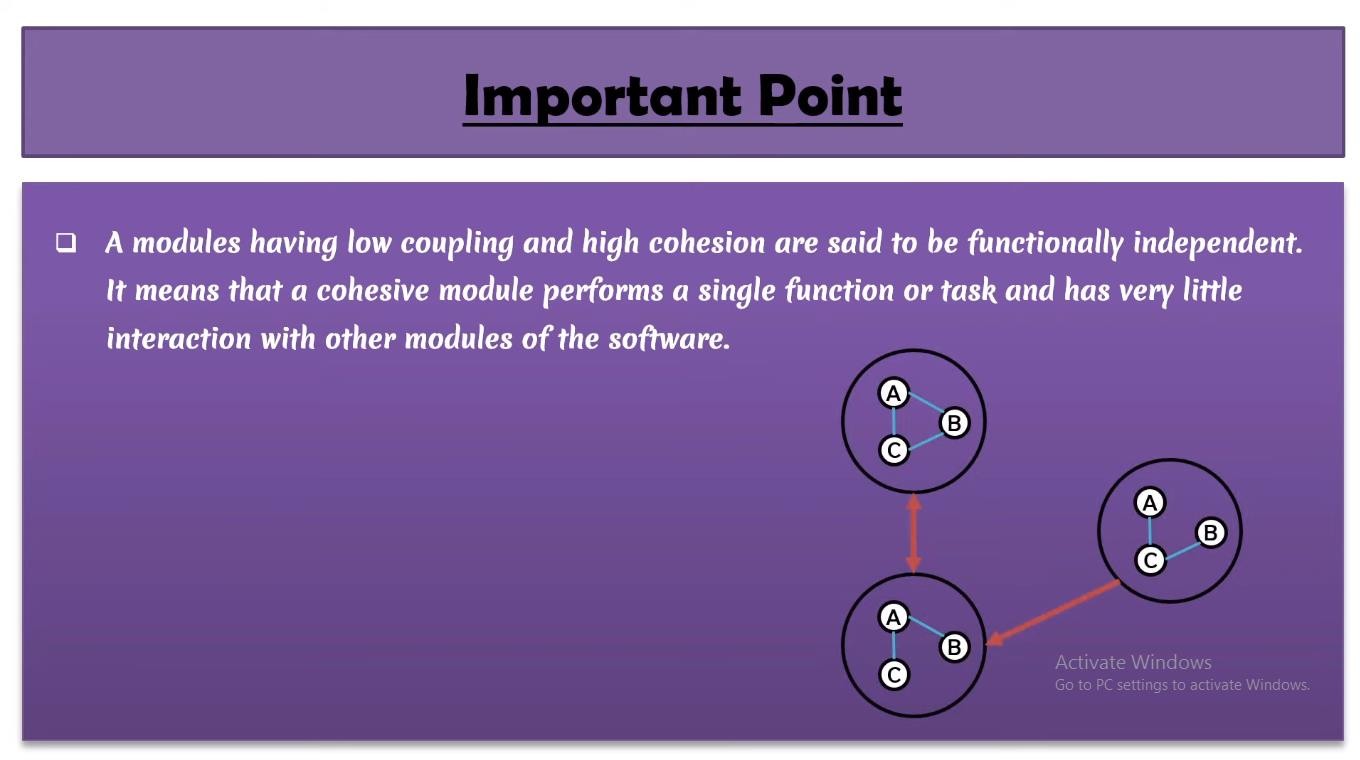








COHESION

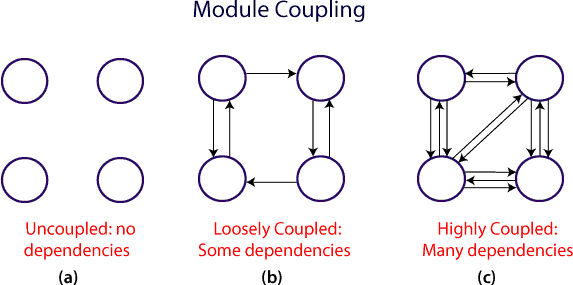


Coupling and Cohesion Module Coupling

In software engineering, the coupling is the degree of interdependence between software

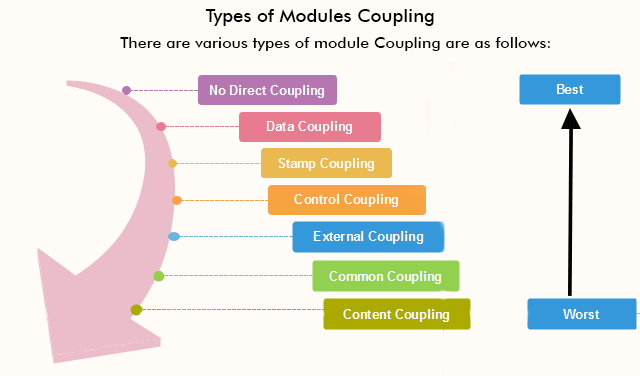
modules. Two modules that are tightly coupled are strongly dependent on each other. However, two modules that are loosely coupled are not dependent on each other. **Uncoupled modules** have no interdependence at all within them.

### The various types of coupling techniques are shown in fig:

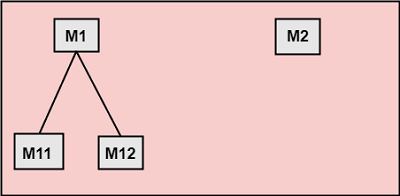


A good design is the one that has low coupling. Coupling is measured by the number of relations between the modules. That is, the coupling increases as the number of calls between modules increase or the amount of shared data is large. Thus, it can be said that a design with high coupling will have more errors.

## Types of Module Coupling

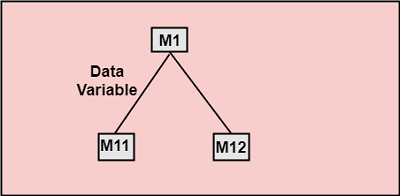


1. **No Direct Coupling:** there is a direct coupling between M1 and M2.

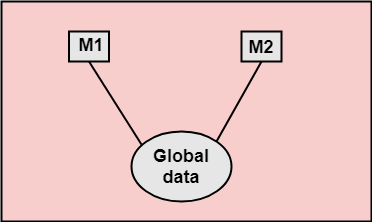


In this case, modules are subordinates to different modules. Therefore, no direct coupling.

1. Data Coupling: When data of one module is passed to another module, this is called data coupling.



1. **Stamp Coupling:** Two modules are stamp coupled if they communicate using composite data items such as structure, objects, etc. When the module passes non-global data structure or entire structure to another module, they are said to be stamp coupled. For example, passing structure variable in C or object in C++ language to a module.
2. **Control Coupling:** Control Coupling exists among two modules if data from one module is used to direct the structure of instruction execution in another.
3. **External Coupling:** External Coupling arises when two modules share an externally imposed data format, communication protocols, or device interface. This is related to communication to external tools and devices.
4. **Common Coupling:** Two modules are common coupled if they share information through some global data items.

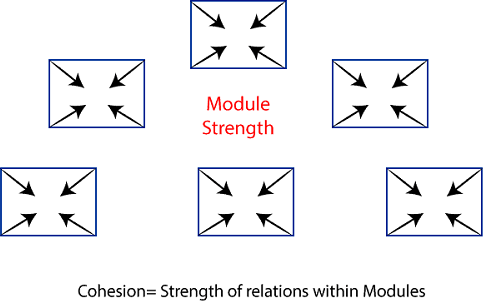


1. **Content Coupling:** Content Coupling exists among two modules if they share code, e.g., a branch from one module into another module.

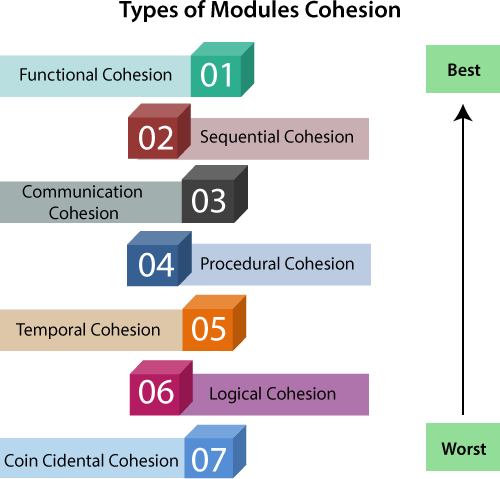
**Module Cohesion**

In computer programming, cohesion defines to the degree to which the elements of a module belong together. Thus, cohesion measures the strength of relationships between pieces of functionality within a given module. For example, in highly cohesive systems, functionality is strongly related.

Cohesion is an ordinal type of measurement and is generally described as "high cohesion" or "low cohesion."



### Types of Modules Cohesion



* 1. **Functional Cohesion**: Functional Cohesion is said to exist if the different elements of a module, cooperate to achieve a single function.
  2. **Sequential Cohesion:** A module is said to possess sequential cohesion if the element of a module form the components of the sequence, where the output from one component of the sequence is input to the next.
  3. **Communicational Cohesion:** A module is said to have communicational cohesion, if all tasks of the module refer to or update the same data structure, e.g., the set of functions defined on an array or a stack.
  4. **Procedural Cohesion:** A module is said to be procedural cohesion if the set of purpose of the module are all parts of a procedure in which particular sequence of steps has to be carried out for achieving a goal, e.g., the algorithm for decoding a message.
  5. **Temporal Cohesion:** When a module includes functions that are associated by the fact that all the methods must be executed in the same time, the module is said to exhibit temporal cohesion.
  6. **Logical Cohesion:** A module is said to be logically cohesive if all the elements of the module perform a similar operation. For example Error handling, data input and data output, etc.

|  |  |
| --- | --- |
| **Coupling** | **Cohesion** |
| Coupling is also called Inter-Module Binding. | Cohesion is also called Intra-Module Binding. |
| Coupling shows the relationships between modules. | Cohesion shows the relationship within the module. |
| Coupling shows the relative **independence** between the modules. | Cohesion shows the module's relative **functional** strength. |
| While creating, you should aim for low coupling, i.e., dependency among modules should be less. | While creating you should aim for high cohesion, i.e., a cohesive component/ module focuses on a single function (i.e., single-mindedness) with little interaction with other modules of the system. |

* 1. **Coincidental Cohesion:** A module is said to have coincidental cohesion if it performs a set of tasks that are associated with each other very loosely, if at all.

**Introduction to Software Design Process**

Software Design is the process to transform the user requirements into some suitable form, which helps the programmer in software coding and implementation. During the software design phase, the design document is produced, based on the customer requirements as documented in the SRS document. Hence the aim of this phase is to transform the SRS document into the design document.

The following items are designed and documented during the design phase:

* Different modules required.
* Control relationships among modules.
* Interface among different modules.
* Data structure among the different modules.
* Algorithms required implementing among the individual modules.

### Objectives of Software Design:

1. **Correctness:**

A good design should be correct i.e. it should correctly implement all the functionalities of the system.

### Efficiency:

A good software design should address the resources, time, and cost optimization issues.

### Understandability:

A good design should be easily understandable, for which it should be modular and all the modules are arranged in layers.

### Completeness:

The design should have all the components like data structures, modules, and external interfaces, etc.

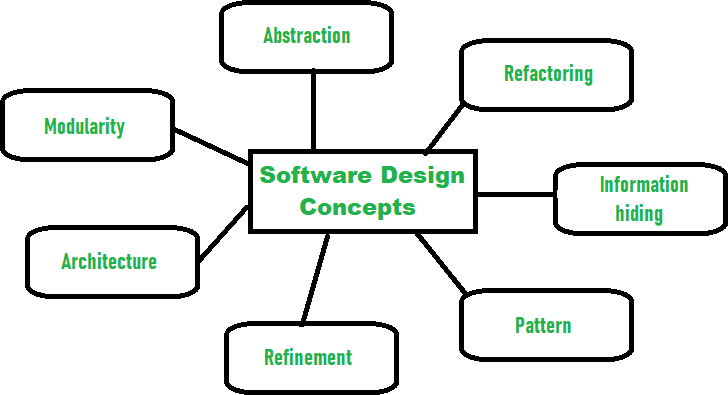
### Maintainability:

A good software design should be easily amenable to change whenever a change request is made from the customer side.

### Software Design Concepts:

Concepts are defined as a principal idea or invention that comes into our mind or in thought to understand something. The **software design concept** simply means the idea or principle behind the design. It describes how you plan to solve the problem of designing software, the logic, or thinking behind how you will design software. It allows the software engineer to create the model of the system or software or product that is to be developed or built.

The software design concept provides a supporting and essential structure or model for developing the right software. There are many concepts of software design and some of them are given below:



The following **points should be considered while designing Software:**

### Abstraction- hide irrelevant data

Abstraction simply means to hide the details to reduce complexity and increases efficiency or quality. Different levels of Abstraction are necessary and must be applied at each stage of the design process so that any error that is present can be removed to increase the efficiency of the software solution and to refine the software solution. The solution should be described in broad ways that cover a wide range of different things at a higher level of abstraction and a more detailed description of a solution of software should be given at the lower level of abstraction.

### Modularity- subdivide the system

Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project. In the same way, modularity in design means subdividing a system into smaller parts so that these parts can be created independently and then use these parts in different systems to perform different functions. It is necessary to divide the software into components known as modules because nowadays there are different software available like Monolithic software that is hard to grasp for software engineers. So, modularity in design has now become a trend and is also important. If the system contains fewer components then it would mean the system is complex which requires a lot of effort (cost) but if we are able to divide the system into components then the cost would be small.

### Architecture- design a structure of something

Architecture simply means a technique to design a structure of something. Architecture in designing software is a concept that focuses on various elements and the data of the structure. These components interact with each other and use the data of the structure in architecture.

### Refinement- removes impurities

Refinement simply means to refine something to remove any impurities if present and increase the quality. The refinement concept of software design is actually a process of developing or presenting the software or system in a detailed manner that means to elaborate a system or software. Refinement is very necessary to find out any error if present and then to reduce it

.

### Pattern- a repeated form

The pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern. The pattern in the design process means the repetition of a solution to a common recurring problem within a certain context.

### Information Hiding- hide the information

Information hiding simply means to hide the information so that it cannot be accessed by an unwanted party. In software design, information hiding is achieved by designing the modules in a manner that the information gathered or contained in one module is hidden and can’t be accessed by any other modules.

### Refactoring- reconstruct something

Refactoring simply means reconstructing something in such a way that it does not affect the behavior of any other features. Refactoring in software design means reconstructing the design to reduce complexity and simplify it without affecting the behavior or its functions. Fowler has defined refactoring as “the process of changing a software system in a way that it won’t affect the behavior of the design and improves the internal structure”.

### Different levels of Software Design:

There are three different levels of software design. They are:

### Architectural Design:

The architecture of a system can be viewed as the overall structure of the system & the way in which structure provides conceptual integrity of the system. The architectural design identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of the proposed solution domain.

### Preliminary or high-level design:

Here the problem is decomposed into a set of modules, the control relationship among various modules identified, and also the interfaces among various modules are identified. The outcome of this stage is called the program architecture. Design representation techniques used in this stage are structure chart and UML.

### Detailed design:

Once the high-level design is complete, a detailed design is undertaken. In detailed design, each module is examined carefully to design the data structure and algorithms. The stage outcome is documented in the form of a module specification document.

* + Software design is more creative than analysis.
  + It is helpful in problem solving activity.

Initial Requirements

Gather Data on User Requiremnets

Analyse Requirement Data

Obtain Answer to all Requirement question

Validate the Design Against the requirements

High level Design

Refine and document the design

### What is Software Design?

Software design is a process to convert the user requirements into client requirements user interface form, that helps the software developer in coding and implementation. The software design deals with portraying the client’s requirement which is described in [Software Requirement Specification (SRS) document](https://www.geeksforgeeks.org/software-engineering-parts-of-a-srs-document/) into a user interface form.

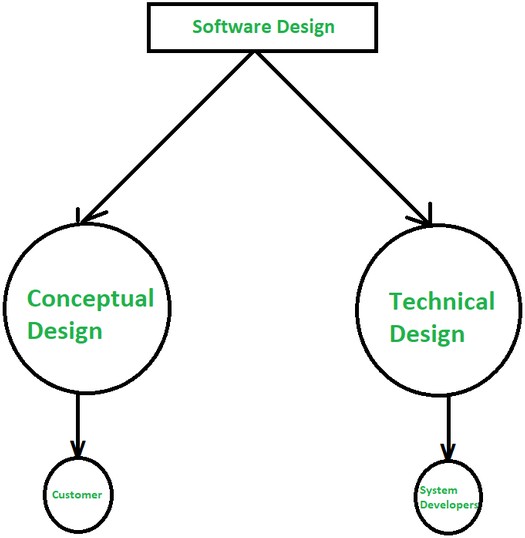
To transform requirements into working system designers must satisfy both customers and therefore the system builders. The customers should understand what the system is to try to do. At an equivalent time, the system builders must understand how to try to. To accomplish the design is split into two parts as shown in the figure below and is named as the 2 parts iterative process. A software design has two parts:

### Conceptual Design :

Conceptual design is an initial/starting phase in the process of planning, during which the broad outlines of function and sort of something are coupled. It tells the customers that what actually the system will do.

1. Technical Design:

A Technical design is a phase in which the event team writes the code and describes the minute detail of either the whole design or some parts of it. It tells the designers that what actually the system will do.



**DIFFERENCE BETWEEN CONCEPTUAL DESIGN AND TECHNICAL DESIGN**

# 