Assignment. No.08

Semester	T.E. Semester V – Computer Engineering
Subject	Software Engineering
Subject Professor In-charge	Dr. Sachin Bojewar
Assisting Teachers	Dr. Sachin Bojewar

Student Name	Deep Salunkhe	
Roll Number	21102A0014	
Grade and Subject Teacher's Signature		

Assignment Number	08
Assignment Title	Why high cohesion and low coupling is desirable in design?.

Why high cohesion and low coupling is desirable in design?

1. **Cohesion:**

Cohesion refers to the measure of how closely related the responsibilities and functions within a module or component are. In other words, it measures the degree to which the elements within a module work together to achieve a common purpose. Cohesion aims to ensure that a module or class has a well-defined, single purpose or responsibility.
There are several types of cohesion, ranked from weakest to strongest:

- **Coincidental Cohesion:** The module contains unrelated and arbitrary functions.
- **Logical Cohesion:** Functions are related by a vague, high-level commonality, such as all involving file operations.
- **Temporal Cohesion:** Functions are grouped because they are executed at the same time, but have no inherent relationship.
- **Procedural Cohesion:** Functions are grouped because they belong to a common procedure or process.
- **Communicational Cohesion:** Functions operate on the same data or share results but have no direct relationship.
- **Sequential Cohesion:** Functions are related because they must be executed in a specific order.
- **Functional Cohesion:** Functions are highly related and perform a single, well-defined task.
- **Informational Cohesion:** Functions are related by the need to share and operate on the same data.
- **Coincidental Cohesion:** Functions are grouped arbitrarily and have no logical relationship.

2. **Coupling:**

Coupling, on the other hand, refers to the degree of interdependence between modules or components in a software system. It measures how much one module relies on the functionality provided by another module. Low coupling implies that modules are loosely connected, while high coupling indicates strong interdependence.
There are various types of coupling, ranging from loose to tight:

Assignment. No.08

- **Content Coupling:** One module directly accesses or modifies the content (variables or data) of another module, which is the strongest form of coupling and is highly discouraged.
- **Common Coupling:** Modules share global data or variables.
- **External Coupling:** Modules depend on external data formats or communication interfaces.
- **Control Coupling:** One module controls the flow of another module by passing control information (e.g., function calls).
- **Stamp Coupling:** Modules share a data structure, such as a record or struct, but do not use all its elements.
- **Data Coupling:** Modules share only data and no control information.
- **No Coupling (or No Coupling):** Modules are completely independent and do not rely on each other.

Why High Cohesion and Low Coupling are Desirable:

The goal in software design is to achieve high cohesion and low coupling because these principles lead to more maintainable, flexible, and robust software systems:

1. **Ease of Maintenance:** High cohesion means that each module has a single, well-defined purpose, making it easier to understand and maintain. Low coupling ensures that changes in one module have minimal impact on others, reducing the risk of unintended side effects during maintenance.
2. **Reusability:** High cohesion often results in more reusable modules. When a module has a specific, clear purpose, it can be easily extracted and reused in other parts of the software or in different projects.
3. **Flexibility and Extensibility:** Low coupling allows for greater flexibility in modifying or extending the software system. Modules can be replaced or enhanced without affecting the overall system's stability.
4. **Testability:** Modules with high cohesion are typically easier to test in isolation, which improves the testing process and helps identify defects more efficiently.
5. **Scalability:** Low coupling allows for easier scalability of the system. New modules can be added or existing ones modified with minimal impact on other parts of the system.
6. **Maintainability:** Software systems with high cohesion and low coupling are easier to understand, leading to better maintainability. Developers can work on individual modules without needing to understand the entire system.

In conclusion, high cohesion and low coupling are desirable in software design because they lead to more maintainable, flexible, and robust systems. These principles enhance the software's ability to evolve, adapt to changing requirements, and minimize the risk of errors during development and maintenance.