Coupling & Cohesion

Let’s assume the current authentication module has the following characteristics:

- High Coupling:
  - Direct dependencies on user management, session handling, and role management modules.
  - Tight integration with database queries within authentication logic.
  - Use of static methods that limit extensibility and testing.
- Low Cohesion:
  - Authentication logic scattered across multiple files.
  - Combined responsibilities such as token generation, validation, and user profile management in a single class.
- Proposed Modifications
  - To improve cohesion and reduce coupling, the module should be redesigned based on principles such as Single Responsibility Principle (SRP), Dependency Inversion, and Separation of Concerns.
- Decouple Authentication from Other Modules
  - Use interfaces for session management and database interaction.
  - Introduce dependency injection to remove direct dependencies.
- Improve Cohesion
  - Break down the authentication module into focused components:
  - AuthenticationService: Handles core authentication logic.
  - TokenService: Manages token generation and validation.

- Ensure Extensibility
    - Use strategy patterns for authentication (e.g., support for OAuth, JWT, etc.).
    - Modularize role management to allow independent evolution.
- Advantages of the New Design
    - Reduced Coupling:
        - Interfaces abstract the dependencies, making it easier to replace components without impacting other parts of the system.
    - Improved Cohesion:
        - Each class has a single responsibility, making the system more organized and easier to maintain.
    - Testability:
        - Mocking dependencies is straightforward, enabling unit testing of each component.
    - Extensibility:
        - Adding new token types or session mechanisms only requires implementing the respective interface.