**Design Patterns**

1. **Q:** How would you implement a thread-safe singleton pattern without using locks?  
   **A:** Use static initialization or Lazy<T> for thread-safe singleton creation.
2. **Q:** What is the difference between the Proxy and Decorator patterns? Provide a real-world example of each.  
   **A:** Proxy controls access to an object, while Decorator adds behavior. Example: Proxy for lazy loading, Decorator for adding logging.
3. **Q:** How would you implement a chain of responsibility pattern for handling multiple levels of user authentication?  
   **A:** Create a chain of handlers where each handler processes the request or passes it to the next.
4. **Q:** What is the difference between the Observer and Pub/Sub patterns?  
   **A:** Observer is a direct relationship between subject and observer, while Pub/Sub uses a message broker for decoupling.
5. **Q:** How would you implement a caching mechanism using the Proxy pattern?  
   **A:** Create a proxy that checks the cache before delegating to the real object.
6. **Q:** What is the difference between the Strategy and State patterns?  
   **A:** Strategy encapsulates interchangeable algorithms, while State changes behavior based on internal state.
7. **Q:** How would you implement a distributed logging system using the Observer pattern?  
   **A:** Use observers to send logs to multiple destinations (e.g., file, database, cloud).
8. **Q:** What is the difference between the Adapter and Facade patterns?  
   **A:** Adapter makes incompatible interfaces work together, while Facade simplifies a complex subsystem.
9. **Q:** How would you implement a retry mechanism using the Command pattern?  
   **A:** Encapsulate the operation in a command object and retry it on failure.
10. **Q:** What is the difference between the Template Method and Strategy patterns?  
    **A:** Template Method defines a skeleton in a base class, while Strategy encapsulates interchangeable algorithms.
11. **Q:** How would you implement a load balancer using the Proxy pattern?  
    **A:** Create a proxy that distributes requests across multiple servers.
12. **Q:** What is the difference between the Composite and Decorator patterns?  
    **A:** Composite treats individual and group objects uniformly, while Decorator adds behavior dynamically.
13. **Q:** How would you implement a distributed transaction system using the Saga pattern?  
    **A:** Break the transaction into smaller, compensatable steps and handle failures with compensating actions.
14. **Q:** What is the difference between the Factory and Abstract Factory patterns?  
    **A:** Factory creates a single type of object, while Abstract Factory creates families of related objects.
15. **Q:** How would you implement a rate-limiting mechanism using the Proxy pattern?  
    **A:** Create a proxy that tracks request counts and throttles excessive requests.
16. **Q:** What is the difference between the Flyweight and Singleton patterns?  
    **A:** Flyweight shares state to reduce memory usage, while Singleton ensures a single instance.
17. **Q:** How would you implement a distributed locking mechanism using the Proxy pattern?  
    **A:** Create a proxy that coordinates locks across distributed systems.
18. **Q:** What is the difference between the Builder and Factory patterns?  
    **A:** Builder constructs complex objects step-by-step, while Factory creates objects in one go.
19. **Q:** How would you implement a circuit breaker pattern in a microservices architecture?  
    **A:** Use a circuit breaker to stop requests to a failing service and retry after a timeout.
20. **Q:** What is the difference between the Mediator and Observer patterns?  
    **A:** Mediator centralizes communication between objects, while Observer decouples subjects and observers.