Comprehension-Based Questions

1. What problem did Sunrise Family Clinic encounter as it grew and added more features to its appointment system?
2. Explain, using the receptionist analogy, how dependency injection improves flexibility in a software system.
3. What is the main difference between constructor injection and setter injection in dependency injection?
4. List two benefits of using dependency injection in a Node.js/Express application.
5. What is the role of an IoC container like TypeDI in a Node.js application?
6. How does TypeDI enable swapping between SMS and Email notification services without changing business logic?
7. Why is constructor injection generally preferred over property or setter injection in TypeScript/Node.js?
8. What is the purpose of using interfaces for services in a modular system?
9. How can you use TypeDI to inject a mock service for testing purposes?
10. Why is it important to reset the DI container between tests?
11. In the MVC analogy, what real-world role does the Controller layer represent in a library system?
12. What is the main responsibility of the Repository layer in the MVC pattern?
13. How does the Repository Pattern help in swapping storage backends without changing business logic?
14. Why is it considered a bad practice to mix data access code with business logic?
15. How does dependency injection contribute to modularity and testability in an MVC-based system?
16. What is the benefit of defining repository interfaces separate from their implementations?
17. In the provided example, what would happen if you wanted to switch from in-memory to database storage for books?
18. What is the primary purpose of the Repository Pattern in software design?
19. Describe the analogy of the university records office as it relates to the Repository Pattern.
20. Why should business logic never directly access storage-specific types or queries, according to best practices for the Repository Pattern?