**Clean Architecture Summary:**

Clean Architecture, also known as Onion, Hexagonal, or Ports and Adapters architecture, organizes dependencies using a Domain-centric approach. It focuses on solid architecture and healthy dependencies, emphasizing the domain over infrastructure.

Clean Architecture is suitable for projects where business logic is critical and high testability is desired. It differs from traditional layered architectures by placing the Domain at the center, with the web project communicating with various frameworks and the infrastructure project interacting with databases and services.

In Clean Architecture, the core project models all business rules and entities, serving as the dependency hub for other projects. Inner projects define interfaces, while outer projects implement them.

The infrastructure project includes repositories, database context classes, cached repositories, and API clients. The web project serves as the application entry point, containing API endpoints, controllers, views, view models, DTOs, and filters.

The shared kernel houses base entities, logging, and other common elements but should not have infrastructure dependencies.

Clean Architecture does not guarantee good software on its own, but it provides benefits such as separation of concerns, loose coupling, solid principle adherence, and high testability. However, it requires a learning curve and can be time-consuming to implement compared to simpler architectures like All-in-One or Layered architectures.