NFR: Non function Requirements

1. Availability – Application should be up 24X7
2. Concurrency
3. Maintainability
4. Scalability – Serve million of users
5. Reliability - Application should be up 24X7
6. Usability – Less than 3 seconds to response

Functional Requirements:

1. List product
2. Filter products by brand and categories
3. Add to cart
4. Apply discount coupons
5. Checkout
6. List order and order item history
7. Login as user and remember my items.

**Monolithic**:

1. Easy to develop, debug and deploy.
2. UI, Business and DB in same code base.
3. Difficult to manage, hard to implement new features.
4. Unable to implement new technologies.
5. Difficult to scale an individual module.

Vertical Scaling- Avoid problem of network latency and security

DRY – Don’t Repeat Yourself.

KISS- Keep it Simple Silly.

YAGNI – You ain’t gonna need it.

**Layered (N- LAyer) Architecture**:

1. Break down application into logical layers
2. Oranising code for **separation of concerns**.
3. Layers can be modified without affecting other layers.
4. Presentation Layer, Business Layer, Database Layer.
5. .Low coupling , High Cohesion.

Problems:

1. Highly couples and dependent on each other
2. Code organization hard toto maintain.
3. Frameworks hard to change.
4. Dependent layers causing complexity and hard to change

SOLID:

1. Single Responsibility – Each module should have one functionality.
2. Open Closed – Open to extension , closed for modification.
3. Liskov Substitution – Abstract the underlying service bus implementation so that it can be replaced from Rabbit MQ to Kafka in future.
4. Interface segregation – Interfaces should be broken into smaller and specific ones.
5. Dependency Inversion – High level modules should not depend on low level modules, both should depend on abstractions.

**Clean Architecture:**  (Maintainability , Flexibility and Testability)

1. Separate the elements of design into circle levels
2. Keep the core business logic and application domain at the center of solution.