SOFTWARE DESIGN & ARCHITECTURE

Name: BABAR KAMAL

ID: 5507

Question No: 01

Answer:

Selected Architecture Style: Hierarchical Style Architecture

Description:

This style is based on hierarchies of different levels of logical modules or submodules of the software structure.

Modules of different levels are basically interlinked or connected by internal and external connective layers such as upper, middle- and lower-layer architecture etc.

We have two sub styles model in it as below:

1-Layered Style:

It is simply organized as layers or hierarchies' style in which each layer gives service to the responding above layer and reformed given to them as client to layer below.

Here, the connectors are basically defined by the major protocols that how the layers can contact or interlinked with each other.

Simple layer architecture is like:

In the front-end: User Interface mgt. authentication and authorization formed

Core business logic and system services System support like databases and OS etc

Each layer exposes an interface (API) to be used by the layer above it

Each layer acts as a

− Server: service provider to layer "above"

− Client: service consumer of the layer "below".

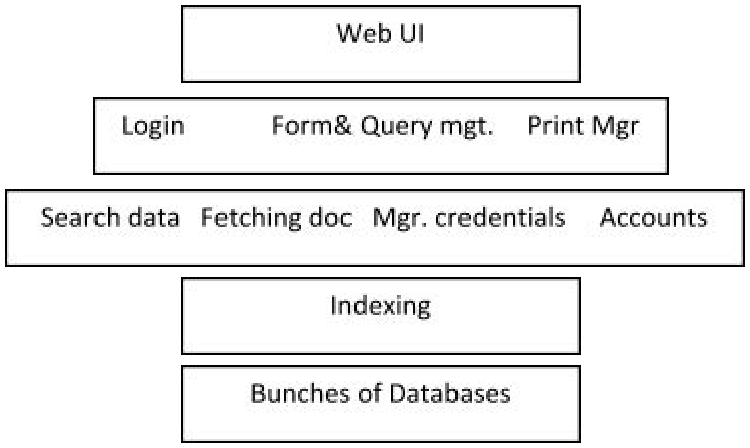
Specialization:

Different exceptions are made to allow the non-linked layers to communicate straight-forwardly which is usually done for the efficiency purpose.

Sample:

We took a simple Admin system of any college including the library as an example.

- It allows full control and electronic access to patent material from leading colleges and libraries.
- The lower level layer or the bottom one is the individual databases in each college library.
  Diagrammatic Explanation is as follows:



Another one example explanation is:

- operating systems – 2INC0 "Operating systems" SfS: Y3Q1
- network and protocol stacks – 2IC60 "Computer networks and security" SfS, WbS: Y2Q4

Components:

- UI
- Authentication and forms
- Search engine
- Document retrieval
- Rights manager
- Accounts management
- databases

Applicable domains of layered architecture(constraints):

Every system is divided between the application-specific portions and platform-specific portions which provide generic services to the application of the system.

- Offers that have clean partitions between core facilities, dangerous amenities, user interface services, etc.
- Applications that have a number of classes that are closely related to each other so that they can be grouped together into a package to provide the services to others.
-

Advantages:

- Applications that have a number of classes that are closely related to each other so that they can be grouped together into a package to provide the services to others.
- Clear dependence structure benefits evolution − Lower layers are independent from the upper layers − Upper layers can evolve independently from the lower layers as long as the interface semantics is unchanged − Strict layering: limits propagation of change.
- Reuse − e.g., standardized layer interfaces for libraries/frameworks
- Incremental software development based on increasing levels of abstraction.
- Enhanced independence of upper layer to lower layer since there is no impact from the changes of lower layer services as long as their interfaces remain unchanged.
- Enhanced flexibility: interchangeability and reusability are enhanced due to the separation of the standard interface and its implementation.
- Component-based technology is a suitable technology to implement layered architecture; this makes it much easier for the system to allow for plug-and-play of new components.
- Elevation of compactness: each layer can be an intangible mechanism installed independently.

Disadvantages:

- Lower runtime performance since a user's request or a response to a client must go through potentially several layers.
- There are also performance concerns of overhead on the data processing and buffering by each layer.
- Many applications cannot fit this architecture design.
- Cracks of interlayer statements may cause gridlocks, and “linking” may cause tight connections.

- Omissions and error management are issues in the layered architecture, since liabilities in one layer must spread upward to all work layers.
- Not generally applicable
- Performance (mostly for strict layering and many layers)