***What is Prototyping?***

Prototyping is the process of quickly putting together a working model in order to test various aspects of a design, illustrate ideas or features & gather early user feedback. It is believed to reduce project risks and cost.

***What is Test-driven Development (TDD)?***

TDD is a software development methodology involving repeatedly writing test cases for classes and then building the corresponding classes to implement only the code necessary to successful pass the tests.

***What is Domain driven design(DDD)?***

Domain driven design (DDD) is an approach to the design of software, based on the two premises that complex domain designs should be based on a model, and that, for most software projects, the primary focus should be on the domain and domain logic (as opposed to being the particular technology used to implement the system).

***What is SRP?***

* SRP stands for Single Responsibility Principle. This states that a class should have only one reason to change. If a class is doing too much (i.e. hitting the DB, writing files, doing business logic, calling a web service, etc,) it’s violating SRP. It will be very difficult to change later should we need to change it, and it will likely be prone to have more defects.
* SRP applies to the methods or in a class also. The two are inseparable. A class, or object, is the combination of its data and its behavior.

***The IHttpHandler and IHttpHandlerFactory interfaces?***

* The IHttpHandler interface is implemented by all the handlers. The interface consists of one property called IsReusable. The IsReusable property gets a value indicating whether another request can use the IHttpHandler instance.
* The method ProcessRequest() allows you to process the current request. This is the core place where all your code goes. This method receives a parameter of type HttpContext using which you can access the intrinsic objects such as Request and Response.
* The IHttpHandlerFactory interface consists of two methods ­ GetHandler and ReleaseHandler. The GetHandler() method instantiates the required HTTP handler based on some condition and returns it back to ASP.NET. The ReleaseHandler() method allows the factory to reuse an existing handler.

***Difference between a Layer and Tier?***

* Layer: It refers to the separation of the logic that is developed for an application in different files.
* Tier :It refers to the physical location of the Layer files.
* example: In an ASP.NET web site, we create GUI web forms, business logic , data access logic, database all in one computer. In this case, we have 4 layers and 1 Tier. if the GUI Web Forms, business logic, data access logic and database are all in different computers, we have 4 layers and 4 Tiers.

***Why should we go ahead with a Layered architecture model?***

* Layered Architecture helps to decompose an otherwise monolithic system into groups of components each placed at a certain layer.
* Layers also standardize the format, the content and the meaning of the messages that cross the layer boundary.
* Conceptualizes the whole system as stack of layers
* Each layer focuses on one specific responsibility
* Each layer consumes the services of the layers below it and offers services to the layers above it.
* Data flow is always linear, both upstream and downstream

***List some of the Advantages of Layered Architecture.***

* ***Ex changeability:*** Allows swapping with an alternate implementation since communication between layers is through well defined protocols/interfaces
* ***Standardization:*** Enables development of standardized tasks and interfaces since the levels of abstraction are clearly defined and commonly accepted
* ***Localization of Dependencies:*** Standardized interfaces between layers confine the effect of change implementation to the layer that is changed

***List some of the Disadvantages of Layered Architecture.***

* ***Low Efficiency:*** Data transfer occurs through a number of intermediate layers. This is less efficient than a group of objects communicating directly with each other
* ***Granularity of Layers:*** The decision about the granularity of layers and the assignments of tasks to the layers is difficult. Too few layers do not fully exploit the pattern and too many layers add overheard of data transfer.
* ***Duplication:*** Work Services performed by lower layers may not be required by higher layers and those performed by higher layers may not be needed by the lower layers. The duplication has a negative impact on the performance of the system

***In which situation we should go ahead for a MVC pattern?***

* The pattern is applicable only in the context of interactive applications with human–computer interface.
* MVC divides an interactive application into three areas Processing, Output, Model
* The Model is independent of specific input or output representations.
* The View component has more than one representation of a single model
* The Controller translates events to service requests for the model or display requests for the view

***List some of the Advantages of MVC Architecture.***

* ***Multiple views:*** MVC separates the model views and allows multiple views to represent the same model simultaneously
* ***Pluggable views and controllers***: The conceptual separation of MVC allows exchange of view and controller objects of a model

***List some of the Disadvantages of MVC Architecture.***

* ***View and the Controller code breaks:*** Both view and controller components make direct calls to the model. If a change occurs in the model’s interface, this coupling increases the possibility of code breaks in the view and the controller code
* ***Inefficiency of data access in View:*** The view at times might have to make multiple calls to obtain its display data and thereby weakens the performance of the system

***Which situation we should dive for Event Driven Architecture?***

* Deals with how the change in the state of one system is propagated to other systems in the application
* The subject has neither the knowledge the interested parties nor of the event’s subsequent processing
* Also known as Publish Subscribe pattern

***What are the participating components for Event Driven Architecture?***

Generators / Subject / Publisher – The source of the event

Subscribers / Observer – Reaction in response to events

Dispatcher – Dispatches events using pattern matching, and correlation techniques

Channel – Component responsible for activities such as translation, filtering of event messages between event generators and event subscribers

***List some of the Advantages of Event Driven Architecture***

* Extremely loosely coupled architecture permits a lower level subject to communicate with a higher level observer without impacting the system’s layering
* Supports broadcast communication in a highly distributed architecture style
* New business services as event listeners can be invoked dynamically and transparently without making any changes to the event source itself.
* Event interceptors help make informed business decisions by providing insights into the system

***List some of the Disadvantages of Event Driven Architecture***

* Event traceability through the system is difficult as the event can traverse in different directions
* Without complex update semantics, all changes in the subject are pushed to all observers regardless of their need, resulting in unnecessary broadcast traffic
* The observers have no knowledge of each other’s presence, so the final outcome of an event response can be hard to track down
* In distributed event architecture, the event context gets copied for each observer and reduces memory scalability

***What are the drivers for Broker Architecture?***

* Broker is a central piece of software that is used by several distributed components to communicate with each other.
* Individual components need not be equipped with different communication mechanism; the components only need the means to talk to the Broker.
* Broker also provides registry services for adding, removing, exchanging, activating and locating components.
* Client side proxies represent a layer between clients and the Broker. This additional layer provides transparency, in that a remote object appears to the client as a local one.
* A Broker is analogous to a messenger that is responsible for the transmission of requests from clients to servers, as well as the transmission of responses and exceptions back to the client.

***Who are the participating components in Broker Architecture?***

* ***Server:*** Objects that expose their functionality through interfaces conforming to Interface Definition Language (IDL) or through a binary standard specific for the Broker
* ***Clients:*** Applications that access the services of at least one server by forwarding requests to the Broker.
* ***Proxies***: Components that hide the communication details by exposing the same object oriented interface as client and server.
* ***Bridge***: This is an optional component used for integrating two brokers across different heterogeneous network

***List some advantages of Broker Architecture***

* ***Distribution transparency:*** A client can treat remote objects like a local object by using a client¬side proxy as a layer between itself and the Broker (CORBA).
* ***Centralization:*** Additional responsibilities such as Instance management, Security, Transaction Management etc can be centralized Ex: EJB Container

***List some Disadvantages of Broker Architecture***

* ***Complex Memory Management:*** Services referenced by external clients are managed by the Broker process and executed remotely in a separate process. The lifecycle of such service instances referenced by external clients makes memory management complex
* ***Increased Overhead:*** Client side proxies expose the same interface as the server object. Invocations that appear to be local, might actually be remote and carry overheads with regard to marshalling, network and so on
* ***Portability Concerns:*** Broker architectures utilize nonstandard generation of stub and proxies to implement the core connection and marshalling mechanism

***List some of the characteristics of Pipes and Filter architecture***

* The pattern is envisaged as a stream of data flowing through a set of processing stages.
* Data can be processed sequentially or in parallel, based on the context
* Pipes and Filters can be combined in different ways to create a family of systems
* Many integration architectural patterns are variants of the Pipes and Filter pattern.
* The key benefit of this architectural style is ¬ The processing steps can be changed and composed in different ways as all the filters and pipes expose the same external interface.
* Many Enterprise Integration Patterns are based on Pipes and Filters Architecture Style
* Content¬Based Router
* Splitter
* Aggregator
* ReSequencer

***What are the main participating components of Pipes and Filter architecture***

* ***Filter:*** Enriches, refines or transforms input data. Each processing step is implemented by a filter component
* ***Pipes:*** Implement the data flow between adjacent processing steps
* ***Pipeline:*** A sequence of filters that are connected using pipes
* ***Port*** The connection between the filter and the pipe. In the basic form, each filter component has one input port and one output port

***List some of the Advantages of Pipes and Filters.***

* Allows complex processing on a message while maintaining independence and flexibility
* Filters that use the same external interface can be composed into different solutions, by connecting the components to different pipes, without having to change the filters themselves
* Filters are easy to reuse in different contexts than large components
* Multiprocessing of filters, for example running them in parallel or quasi¬parallel, is possible

***List some of the Disadvantages of Pipes and Filters.***

* Increased overhead due to
* The small granularity of each processing step
* Number of context switch to be made between filters
* Performance is affected by the need to copy data between address spaces of filters
* Constraints on the allowed data size in the pipes (typically queues) affects the scalability of data transfers between filters