Chap-2 Programming Architecture

2.1 MVC

- MVC stands for Model, View and Controller.
- The **Model-View-Controller** (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.
- Each of these components are built to handle specific development aspects of an application.
- All most all the languages use MVC with slight variation, but conceptually it remains the same.
- One of the most Frequently Used Pattern
- Separates application functionality
- Promotes Organized Programming

Some Popular Framework that use MVC Concepts are:

- Ruby on Rails (Ruby)
- Laravel (PHP), Codeigniter(PHP), Zend(PHP)
- Angular(JS), BackBone(JS),Express(JS)
- Django(Python), Flask(Python)

Model:

- Model represents shape of the data and business logic.
- It maintains the data of the application.
- Model objects retrieve and store model state in a database.
- Model objects are the parts of the application that implement the logic for the application's data domain.
- Interactions with Database (CRUD)
- Communicates With Controller
- Can Sometimes Updates the View (depends on framework)

Model is Brain of the Application, Model is a data and business logic.

View:

- View is a user interface.

- View display data using model to the user and also enables them to modify the data.
- Views are the components that display the application's user interface (UI).
- Model is what the end user sees (UI)
- Usually Consist of HTML/CSS
- Communicates With Controller
- Speaks only with controller

View is a User Interface.

Controller:

- Controller handles the user request.
- Typically, user interact with View, which in-turn raises appropriate URL request, this request will be handled by a controller.
- The controller renders the appropriate view with the model data as a response.
- Receives Input from View / URLs
- Processes Requests (GET, POST, PUT, DELETE)
- Gets Data from the Model
- Passes Data to the View

Controller is a request handler.

Interaction between Model, View and Controller.

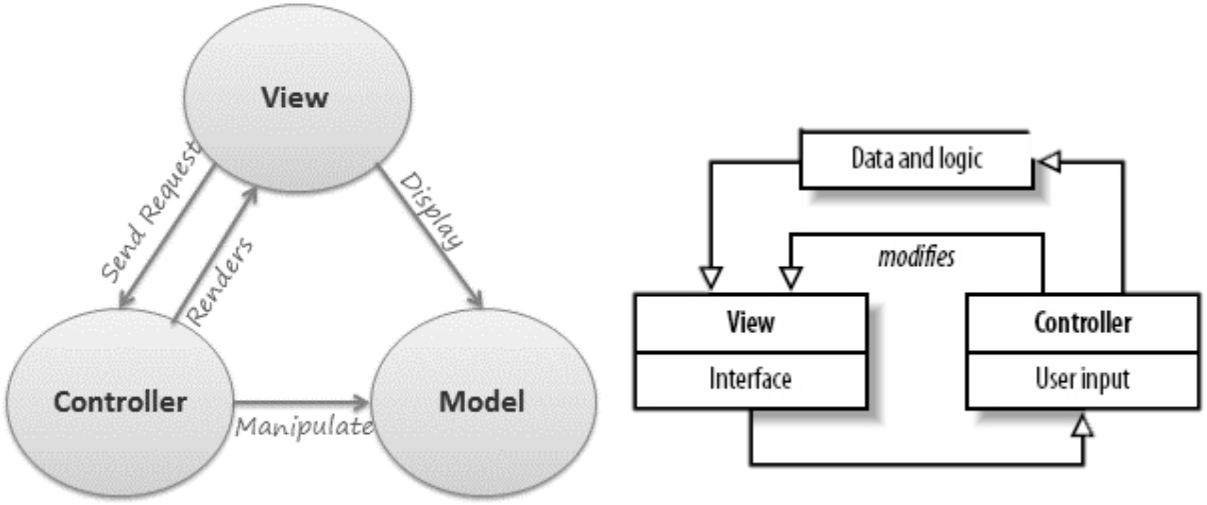


Fig: MVC Architecture

The following figure illustrates the flow of the **user's request in ASP.NET MVC**.

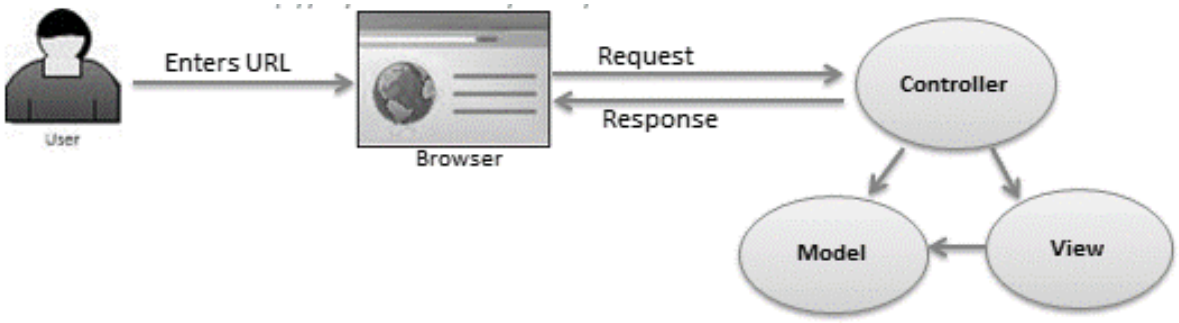


Fig: Request/Response in MVC Architecture

As per the above figure, when the user enters a URL in the browser, it goes to the server and calls appropriate controller. Then, the Controller uses the appropriate View and Model and creates the response and sends it back to the user.

Advantages of MVC

1) **Faster development process:** MVC supports rapid and parallel development. With MVC, one programmer can work on the view while other can work on the controller to create business logic of the web application. The application developed using MVC can be three times faster than application developed using other development patterns.

2) **Ability to provide multiple views:** In the MVC Model, you can create multiple views for a model. Code duplication is very limited in MVC because it separates data and business logic from the display.

3) **Support for asynchronous technique:** MVC also supports asynchronous technique, which helps developers to develop an application that loads very fast.

4) **Modification does not affect the entire model:** Modification does not affect the entire model because model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire architecture.

5) **MVC model returns the data without formatting:** MVC pattern returns data without applying any formatting so the same components can be used and called for use with any interface.

6) **SEO friendly Development platform**: Using this platform, it is very easy to develop SEO-friendly URLs to generate more visits from a specific application.

7**) Ideal for Developing Large Size Web Application.**

Disadvantages of MVC

1) Increased complexity

For a simple interface, strictly abide by the MVC, to make the model, view and controller separation, will increase the complexity of the structure, and may produce too much of the update operation, reduce the operation efficiency.

2) Inefficiency of data access in view

3) Difficulty of using MVC with modern user interface.

4) Need multiple programmers

5) Knowledge on multiple technologies is required.

6) Developer have knowledge of client side code and html code.

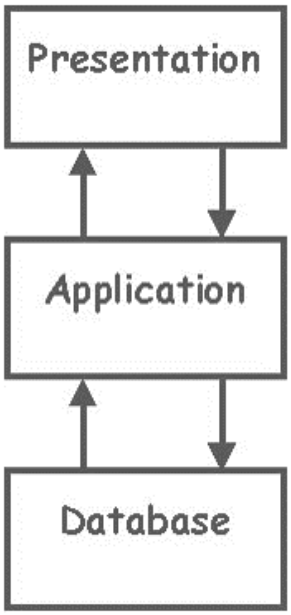
2.2 N-tier Architecture

- Layered / multi-tier architecture
- **The n-tier architecture** is an industry-proven software architecture model.
- It is suitable to support enterprise level client-server applications by providing solutions to **scalability, security, fault tolerance, reusability, and maintainability.**
- It helps developers to create flexible and reusable applications.
- This architecture model provides Software Developers to create Reusable application/systems with maximum flexibility.

N-Tier and **N-Layer** are entirely different concepts.

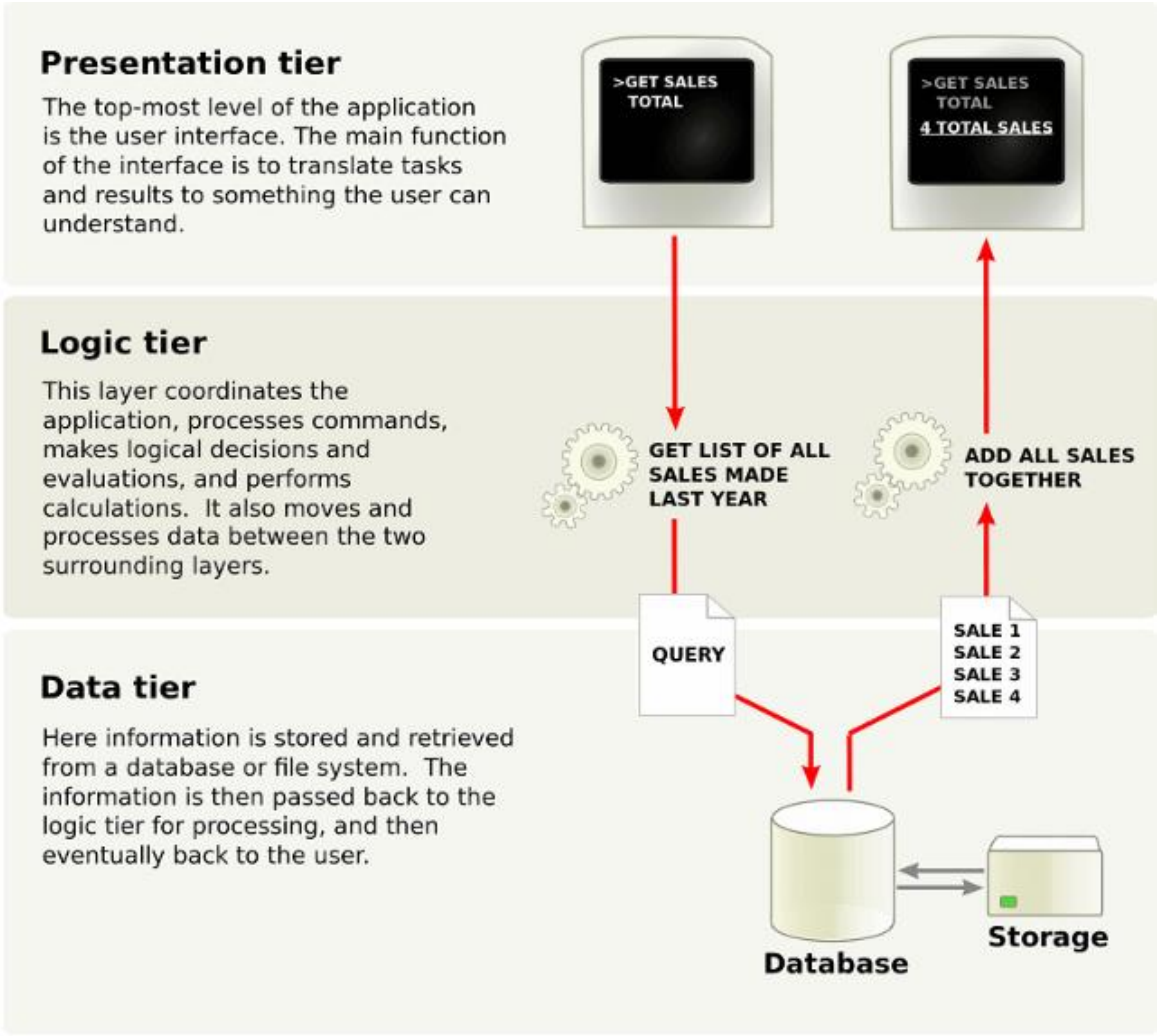
- **N-Tier** refers to the actual n system components of your application.
- **N-Layers** refer to the internal architecture of your component.

- **N-Tier** architecture usually has at least three separate logical parts, each located on separate physical server. Each tier is responsible for a specific functionality.
- **N-Layers** of application may reside on the same physical computer (same tier) and the components in each layer communicates with the components of other layer by well-defined interfaces.



N Tier Architecture Diagram

- The separate physical location of these tiers is what differentiates n-tier architecture from the model-view-controller framework that only separates presentation, logic, and data tiers in concept.
- N-tier architecture also differs from MVC framework in that the former has a middle layer or a logic tier, which facilitates all communications between the different tiers.
- When you use the MVC framework, the interaction that happens is **triangular;** instead of going through the logic tier, it is the control layer that accesses the model and view layers, while the model layer accesses the view layer. Additionally, the control layer makes a model using the requirements and then pushes that model into the view layer.
- This is not to say that you can only use either the MVC framework or the n-tier architecture. There are a lot of software that brings together these two frameworks. For instance, you can use the n-tier architecture as the overall architecture, or use the MVC framework in the presentation tier.



Presentation Layer:

- The presentation tier is the user interface.
- This is what the software user sees and interacts with.
- This is where they enter the needed information. This tier also acts as a go-between for the data tier and the user, passing on the user's different actions to the logic tier.
- Layer that users can access directly such as desktop UI, web page and etc.
- Also called client presentation layer

Application Layer / Business Logic Layer / Logic Layer:

- Also called middle layer
- Accepts the data from the application layer and passes it to the data layer.
- Business logic acts as an interface between Client layer and Data Access Layer

- All business logic – like validation of data, calculations, data insertion/modification are written under business logic layer.
- It makes communication faster and easier between the client and data layer
- Defines a proper workflow activity that is necessary to complete a task.

Data layer:

- The data tier is where all the data used in your application are stored.
- You can securely store data on this tier, do transaction, and even search through volumes and volumes of data in a matter of seconds.
- The external data source to store the application data

Such as database server, CRM System, ERP System, mainframe or other legacy system.

- The one we meet often is database server.
- For N –tier architecture we need to use the non-embedded database server, which can be run in an individual computer.
- Such as SQL Server, ORACLE, DB2, Postgre, MYSQL etc.

Some of the popular sites who have applied this architecture are

- MakeMyTrip.com
- Sales Force enterprise application
- Indian Railways – IRCTC
- Amazon.com, etc.

What are the Benefits of N-Tier Architecture?

There are several benefits to using n-tier architecture for your software. These are scalability, ease of management, flexibility, and security.

- **Secure:** You can secure each of the three tiers separately using different methods.
- **Easy to manage:** You can manage each tier separately, adding or modifying each tier without affecting the other tiers.
- **Scalable:** If you need to add more resources, you can do it per tier, without affecting the other tiers.
- **Flexible:** Apart from isolated scalability, you can also expand each tier in any manner that your requirements dictate.

- **More efficient development.** N-tier architecture is very friendly for development, as different teams may work on each tier. This way, you can be sure the design and presentation professionals work on the presentation tier and the database experts work on the data tier.
- **Easy to add new features.** If you want to introduce a new feature, you can add it to the appropriate tier without affecting the other tiers.
- **Easy to reuse.** Because the application is divided into independent tiers, you can easily reuse each tier for other software projects. For instance, if you want to use the same program, but for a different data set, you can just replicate the logic and presentation tiers and then create a new data tier.

2.3 Client-Server Traditional Model

- **Client-Server Architecture:** It is an architecture model where the client (one program) requests a service from a server (another program)
- **I.e.** it is a request-response service provided over the internet or through an intranet.

In this model, **Client** will serve as one set of program/code which executes a set of actions over the network. While **Server**, on the other hand, is a set of another program, which sends the result sets to the client system as requested.

- In this, client computer provides an interface to an end user to request a service or a resource from a server and on the other hand server then processes the request and displays the result to the end user.

- **An example of Client-Server Model**– an ATM machine. A bank is the server for processing the application within the large customer databases and ATM machine is the client having a user interface with some simple application processing.

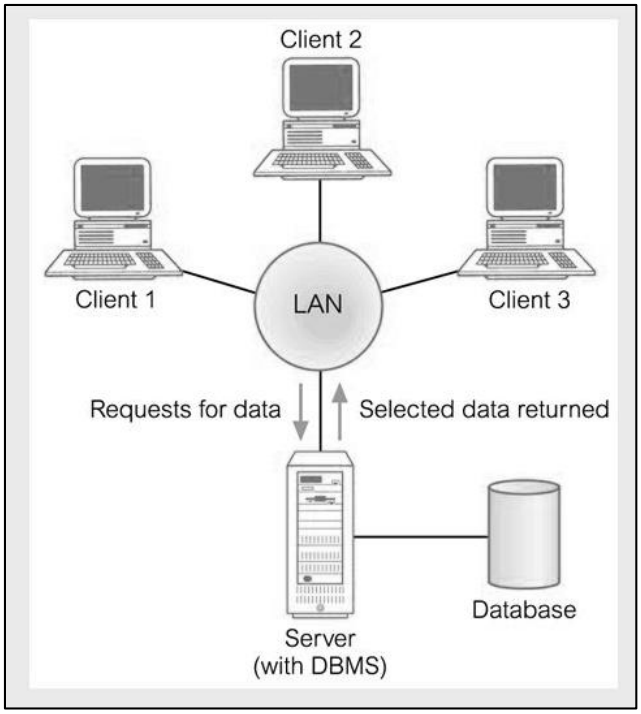


Figure: Two Tier Client server Traditional model

Characteristics of Client-Server Architecture

1. Client and server machines need different amount of hardware and software resources.
2. Client and server machine may belong to different vendors
3. Increase of the client machines → migration to a more powerful server or to multi-server solution
4. A client server application interacts directly with a transport layer protocol to establish communication and to send or receive information
5. The transport protocol then uses lower layer protocols to send or receive individual message. Thus, a computer needs a complete stack of protocols to run either a client or a server.

2.4 Comparison amongst 2-Tire, 3-Tier and N-Tier Architecture

I) 1-Tier Architecture:

- All the layers (Presentation, logical and Data) can only run in one computer
- In order to achieve 1 tier, we need to use the embedded database system , which cannot run in an individual process

- Otherwise, there will be at least 2-tier because non-embedded database usually can run in an individual computer(tier)

ii) 2-Tier Architecture:

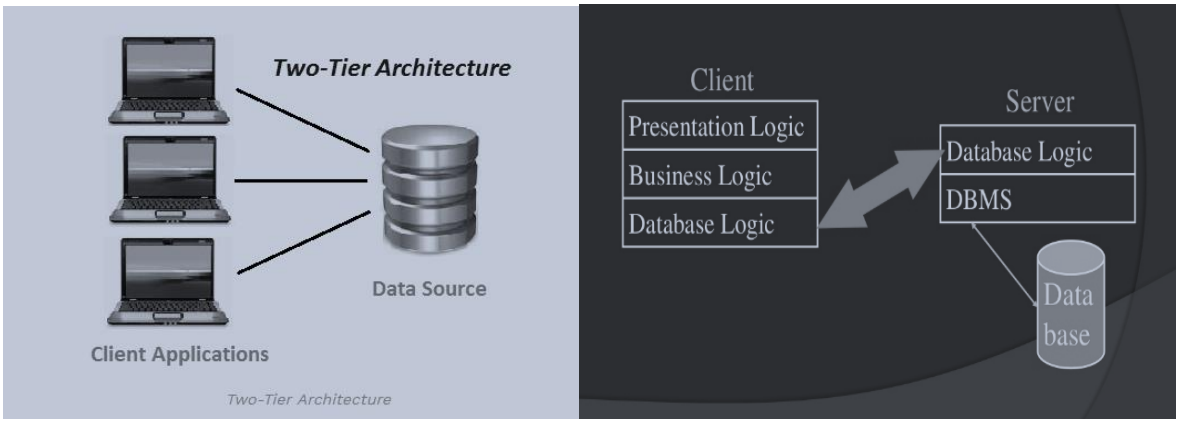


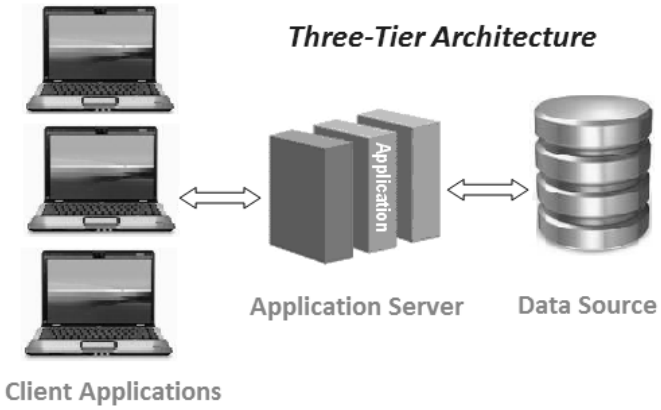
Fig: 2-Tier Architecture

The Two-tier architecture is divided into two parts:

1) Client Application (Client Tier)
2) Database (Data Tier)

- ❖ On client application side the code is written for saving the data in the SQL server database.
- ❖ Client sends the request to server and it process the request & send back with data.
- ❖ The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.
- ❖ Either presentation layer or application layer can only run in one computer, **or** application layer and data layer can only run in one computer.
- ❖ The whole application cannot run in more than 2 computers

iii) 3- Tier Architecture:



Three-tier Architecture

- **Three-tier architecture** typically consist of a presentation tier, a business or data access tier, and a data tier.
- Three layers in the three tier architecture are as follows:

1) Client layer
2) Business layer
3) Data layer

1) Client layer:

- Also called as *Presentation layer* which contains UI part of our application.
- Used for the design purpose where data is presented to the user or input is taken from the user.
- For example designing registration form which contains text box, label, button etc.

2) Business layer:

- In this layer all business logic written like validation of data, calculations, data insertion etc.
- This acts as an interface between Client layer and Data Access Layer.
- Also called the intermediary layer, helps to make communication faster between client and data layer.

3) Data layer:

- In this layer actual database is comes in the picture.

- ❖ Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.

3-Tier Summary:

- The simplest case of N-Tier Architecture
- All above three layers are able to run in three separate computers
- Practically, these three layers can also be deployed in one computer (3- tier architecture, but deployed as 1- tier)

Advantages and Disadvantages of 1 or 2-tier Architecture

Advantages

- Simple and faster for a lower number of users due to fewer process and fewer tiers
- Low cost for hardware, network, maintenance and deployment due to less hardware and network bandwidth needed
- Easy to maintain and modification is bit easy
- Communication is faster

Disadvantages:

- Will have issues when the number of user gets big
- Has limitation to solve issues like security, Scalability, fault tolerance and etc. because it can be deployed in only 1 or 2 computers.
- In two tier architecture application performance will be degrade upon increasing the users.
- Cost-ineffective

Comparison between 2-Tier, 3-Tier and N-Tier Architecture:

2-Tier Architecture	3-Tier Architecture	N-Tier Architecture
✓ Two tire client server architecture consists of two logical layer: client layer and server layer.	✓ Three tire client server architecture consists of three logical layer: client layer, application layer and server layer.	✓ High performance, lightweight persistent objects ✓ Scalability – Each tier can scale horizontally

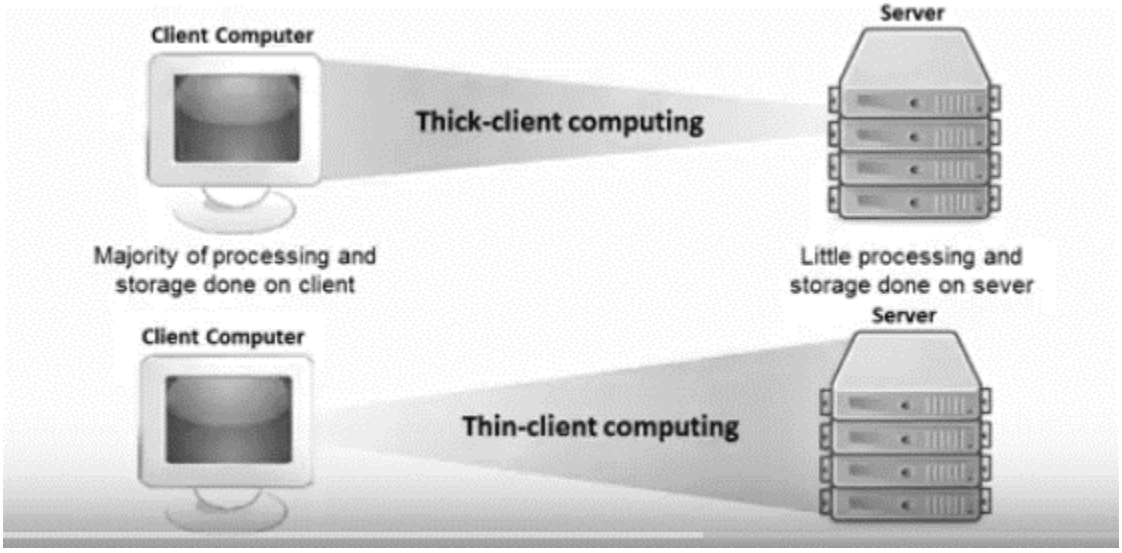
✓ There is a communication happens between client and server without any middle ware. ✓ Direct communication occur in this architecture ✓ Run faster ✓ Less flexible because it is tightly coupled. ✓ Easy to build and maintain ✓ Provides poor security because of direct access ✓ Performance loss whenever the users increases rapidly.	✓ There is a communication happens between client and server with middle ware and application layer acts as middle ware ✓ Indirect communication occur in this architecture ✓ Run slower ✓ More flexible because it is loosely coupled. ✓ Complex to build and maintain ✓ Provides extra security because of indirect access ✓ Performance loss whenever the system run on internet but it gives more performance than two tier.	✓ Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers. ✓ High degree of flexibility in deployment platform and configuration ✓ Better Re-use ✓ Improve Data Integrity ✓ Improved Security – Client is not direct access to database. ✓ Easy to maintain and modification is bit easy, won't affect other modules ✓ In N-tier architecture application performance is good.

Thin and Thick Client

- In the world of client/server architecture, you need to determine if it will be the client or the server that handles the bulk of the workload.
- By client, we mean the application that runs on a personal computer or workstation and relies on a server to perform some operations.
- Thick or thin client architecture is actually quite similar. In both cases, you can consider it as being the client application running on a PC whose function is to send and receive data over the network to the server program.
- The server would normally communicate that information to the middle-tier software (the backend), which retrieves and stores that information from a database.

Diagrammatic illustration of Thin and thick client:

- Client "**Thickness**" is a term which refers to the amount of processing and storage that it does in comparison to the server it is connected to.



Very Little Processing and Storage Done on Client

Majority of Processing and Storage Done on Server

Figure: Think and Thick client Illustration

Thin Clients

Advantages:

- A thin client is designed to be especially small so that the bulk of the data processing occurs on the server.
- Software updates can be done once on server and automatically end up on every client.
- Easy to set up and configure e.g. add new terminals to a network
- Much more secure as all data is stored in central place, the server.
- A thin client is a network computer without a hard disk drive. They act as a simple terminal to the server and require constant communication with the server as well.
- Thin clients provide a desktop experience in environments where the end user has a well-defined and regular number of tasks for which the system is used.

Disadvantages:

- Very reliant on the central server. If it goes down most functionality is lost.
- This method is dependent on a very powerful and reliable central server which will be expensive
- With this method there is a much higher demand on bandwidth.

Thick Clients

Advantages:

- A thick client (also called a fat client) is one that will perform the bulk of the processing in client/server applications. With thick clients, there is no need for continuous server communications as it is mainly communicating archival storage information to the server.
- More robust and more reliable. This results in greater “Up-Time” and availability of services.
- The preferred solution when you want to run heavy duty / resource hungry software applications.
- Can operated without relying on a constant connection to a central server

Disadvantages:

- Often more expensive and higher specification client computers required.
- Every client needs its own software installed, this increase the effort and time on the part of the network administration team
- Due to the non-centralized, distributed nature this method is more prone to integrity issues.

Thick vs. Thin – A Quick Comparison

Thin Clients	**Thick Clients**
1. Easy to deploy as they require no extra or specialized software installation 2. Needs to validate with the server after data capture 3. If the server goes down, data collection is halted as the client needs constant communication with the server 4. Cannot be interfaced with other equipment (in plants or factory settings for example) 5. Clients run only and exactly as specified by the server 6. More downtime 7. Portability in that all applications are on the server so any workstation can access 8. Opportunity to use older, outdated PCs as clients	1. Reduced security threat 2. More expensive to deploy and more work for IT to deploy 3. Data verified by client not server (immediate validation) 4. Robust technology provides better uptime 5. Only needs intermittent communication with server 6. More expensive to deploy and more work for IT to deploy 7. Require more resources but less servers 8. Can store local files and applications 9. Reduced server demands 10. Increased security issues