# **3-Tier Architecture: A Complete Overview**

## What is a 3-Tier Architecture?

A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts. This added flexibility can improve overall time-to-market and decrease development cycle times by giving development teams the ability to replace or upgrade independent tiers without affecting the other parts of the system.

For example, the user interface of a web application could be redeveloped or modernized without affecting the underlying functional business and data access logic underneath. This architectural system is often ideal for embedding and integrating 3rd party software into an existing application. This integration flexibility also makes it ideal for embedding analytics software into pre-existing applications and is often used by [embedded analytics](https://www.logianalytics.com/resources/bi-encyclopedia/embedded-business-intelligence/) vendors for this reason. 3-tier architectures are often used in cloud or on-premises based applications as well as in software-as-a-service (SaaS) applications.

What Do the 3 Tiers Mean?

* **Presentation Tier-**The presentation tier is the front end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, CSS, or through other popular web development frameworks, and communicates with others layers through API calls.
* **Application Tier-**The application tier contains the functional business logic which drives an application’s core capabilities. It’s often written in Java, .NET, C#, Python, C++, etc.
* **Data Tier-**The data tier comprises of the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls.
* The typical structure for a 3-tier architecture deployment would have the presentation tier deployed to a desktop, laptop, tablet or mobile device either via a web browser or a web-based application utilizing a web server. The underlying application tier is usually hosted on one or more application servers, but can also be hosted in the cloud, or on a dedicated workstation depending on the complexity and processing power needed by the application. And the data layer would normally comprise of one or more relational databases, big data sources, or other types of database systems hosted either on-premises or in the cloud.
* A simple example of a 3-tier architecture in action would be logging into a media account such as Netflix and watching a video. You start by logging in either via the web or via a mobile application. Once you’ve logged in you might access a specific video through the Netflix interface which is the presentation tier used by you as an end user. Once you’ve selected a video that information is passed on to the application tier which will query the data tier to call the information or in this case a video back up to the presentation tier. This happens every time you access a video from most media sites.

## What Are the Benefits of Using a 3-Layer Architecture?

There are many benefits to using a 3-layer architecture including speed of development, scalability, performance, and availability.  As mentioned, modularizing different tiers of an application gives development teams the ability to develop and enhance a product with greater speed than developing a singular code base because a specific layer can be upgraded with minimal impact on the other layers.  It can also help improve development efficiency by allowing teams to focus on their core competencies. Many development teams have separate developers who specialize in front- end, server back-end, and data back-end development, by modularizing these parts of an application you no longer have to rely on full stack developers and can better utilize the specialties of each team.

Scalability is another great advantage of a 3-layer architecture. By separating out the different layers you can scale each independently depending on the need at any given time. For example, if you are receiving many web requests but not many requests which affect your application layer, you can scale your web servers without touching your application servers. Similarly, if you are receiving many large application requests from only a handful of web users, you can scale out your application and data layers to meet those requests without touch your web servers. This allows you to load balance each layer independently, improving overall performance with minimal resources. Additionally, the independence created from modularizing the different tiers gives you many deployment options. For example, you may choose to have your web servers hosted in a public or private cloud while you’re application and data layers may be hosted onsite. Or you may have your application and data layers hosted in the cloud while your web servers may be locally hosted, or any combination thereof.

By having disparate layers you can also increase reliability and availability by hosting different parts of your application on different servers and utilizing cached results. With a full stack system you have to worry about a server going down and greatly affecting performance throughout your entire system, but with a 3-layer application, the increased independence created when physically separating different parts of an application minimizes performance issues when a server goes down.