**THREE TIER ARCHITECTURE**

The three-tier architecture is the most popular implementation of a multi-tier architecture and consists of a single presentation tier, logic tier, and data tier. The following illustration shows an example of a simple, generic three-tier application**.**

****

**Presentation Tier:** The presentation tier is responsible for interacting with the logic tier through the API Gateway REST endpoints exposed over the internet. Any HTTPS capable client or device can communicate with these endpoints, giving your presentation tier the flexibility to take many forms (desktop applications, mobile apps, webpages, IoT devices, and so forth).

**Logic Tier:** The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information that is collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete, or modify data in the data tier.

The application tier is typically developed by using Python, Java, Perl, PHP or Ruby, and communicates with the data tier by using API calls.

**Data Tier:** The data tier, sometimes called database tier, data access tier or back-end, is where the information that is processed by the application is stored and managed.

In a three-tier application, all communication goes through the application tier. The presentation tier and the data tier cannot communicate directly with one another.

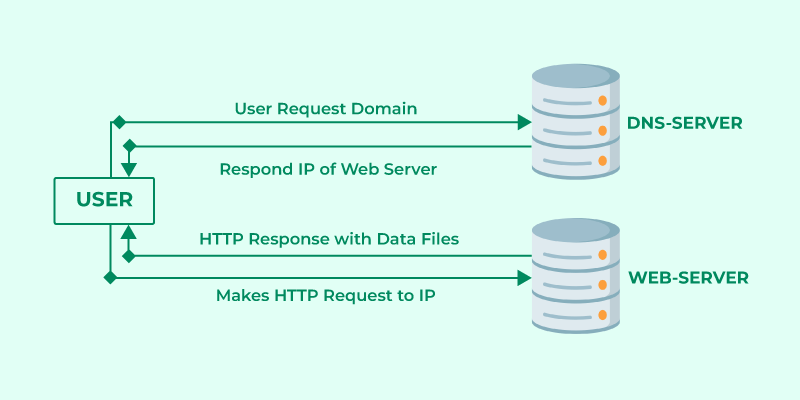
**Client**: Client is a computer (Host) i.e. capable of receiving information or using a particular service from the service providers (Servers).

**Server:** Server is a remote computer that provides information (data) or access to particular services.

* So, it is the Client requesting something and the Server serving it as long as it is in the database.

**Browser Interaction with the servers:**

1. User enters the URL (Uniform Resource Locator) of the website or file. The Browser then requests the [DNS (DOMAIN NAME SYSTEM)](https://www.geeksforgeeks.org/domain-name-system-dns-in-application-layer/) Server.
2. DNS Server lookup for the address of the WEB Server.
3. The DNS Server responds with the IP address of the WEB Server.
4. The Browser sends over an HTTP/HTTPS request to the WEB Server’s IP (provided by the DNS server).
5. The Server sends over the necessary files for the website.
6. The Browser then renders the files and the website is displayed. This rendering is done with the help of DOM (Document Object Model) interpreter, CSS interpreter, and JS Engine collectively known as the JIT or (Just in Time) Compilers.



**Application programming interface:** An API, or application programming interface, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features and functionality.

* APIs allow for the sharing of only the information necessary, keeping other internal system details hidden, which helps with system security. Servers or devices do not have to fully expose data—APIs enable the sharing of small packets of data, relevant to the specific request.

**FULL STACK DEVELOPMENT**

Full stack development is the process of designing, creating, testing, and deploying a complete web application from start to finish. It involves working with various technologies and tools, including front-end web development, back-end web development, and database development. And full stack development is a term used to describe a software engineer or developer who works with both the front and back end of a website or application.

**Front End:** Client software, also known as a front end, is a type of software that interacts with users. It is responsible for the graphical user interface (GUI) that users see and interact with. It allows users to access and use the features and functions of the underlying software or system.

**Front End Languages:** Several languages can be used for front-end development, including HTML, CSS, and JavaScript.

* HTML is the most basic of the three languages and is used to structure content on a webpage.
* CSS is used to style the content on a web page and can be used to create sophisticated layouts.
* JavaScript is used to add interactivity to a web page and can be used to create dynamic content.

**Back End:** Server software, also known as the back-end software, is responsible for managing and coordinating the activities of the server. It is responsible for ensuring that the server is up and running and that all the different server components are functioning correctly. Server software also provides an interface for users to interact with the server and for administrators to manage the server.

To run a website or application, you will need to have a server running server software. This software allows users to interact with your site or application. Without server software, there would be no way for users to request data or information from your site.

Server software is an integral part of any website or application. It is essential for anyone who wants to run a website or application to have a basic understanding of how it works.

**Back-end Languages:** There are a variety of back-end languages that can be used to develop a website or application. PHP, Java, Python, and Ruby are the most popular back-end languages. Each language has its strengths and weaknesses, so it's essential to choose the correct language for the project.

* PHP is a good choice for small projects that require simple functionality, and Java is a good choice for larger projects that require more complex functionality.
* Python is a good choice for projects requiring much data processing, and Ruby is a good choice for projects requiring a lot of user interaction.

**Database:** A database is an organized collection of data stored in a computer system and usually managed by a database management system (DBMS).

* The most commonly used types of databases are Relational and NoSQL.
* The data in relational databases is modeled in tables, making querying and processing efficient. Structured query language (SQL) is commonly used for data querying and writing.
* The data in NoSQL databases is modeled as key-value pairs.
* Examples of relational DBMS are MySQL, Oracle, Microdoft SQL Server and Postgre SQL.
* Examples of NoSQL DBMS are MongoDB, Cassandra and DynamoDB

**Software Development Lifecycle:** SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.



A typical Software Development Life Cycle consists of the following stages −

**Stage 1: Requirement Analysis:** Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

**Stage 2: Planning:** Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

**Stage 3: Designing the Product Architecture:** SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

**Stage 4: Building or Developing the Product:** In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

**Stage 5: Testing:** This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

**Stage 6: Deployment in the Market and Maintenance:** Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.