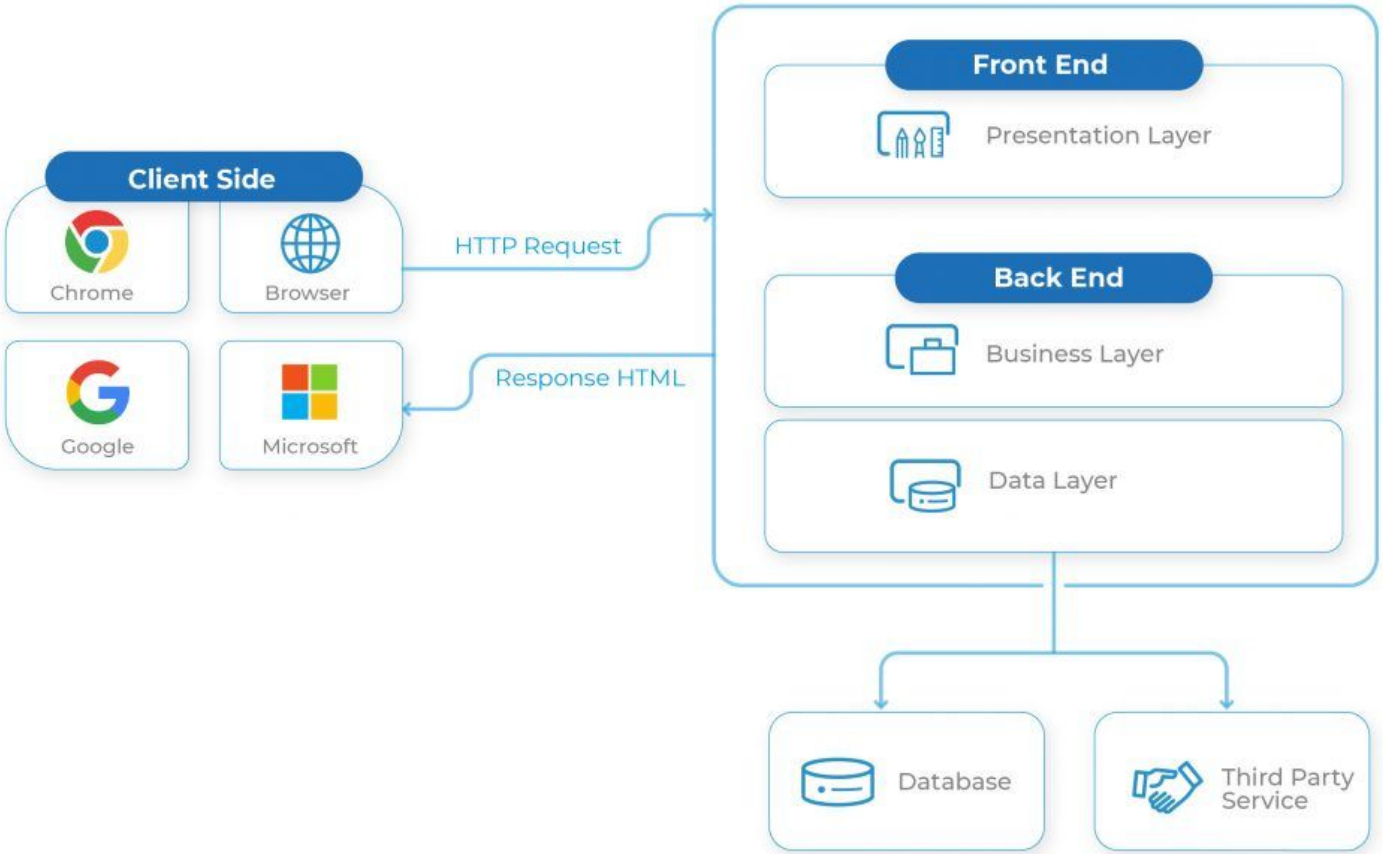
Web Application Architecture

- A web app architecture presents a layout with all the software components (such as databases, applications and middleware) and how they interact with each other.
- It defines how the data is delivered through HTTP and ensures that the client-side server and the backend server can understand.


Client Side
Chrome
Browser
Google
Microsoft
HTTP Request
Response HTML
Front End
Presentation Layer
Back End
Business Layer
Data Layer
Database
Third Party Service

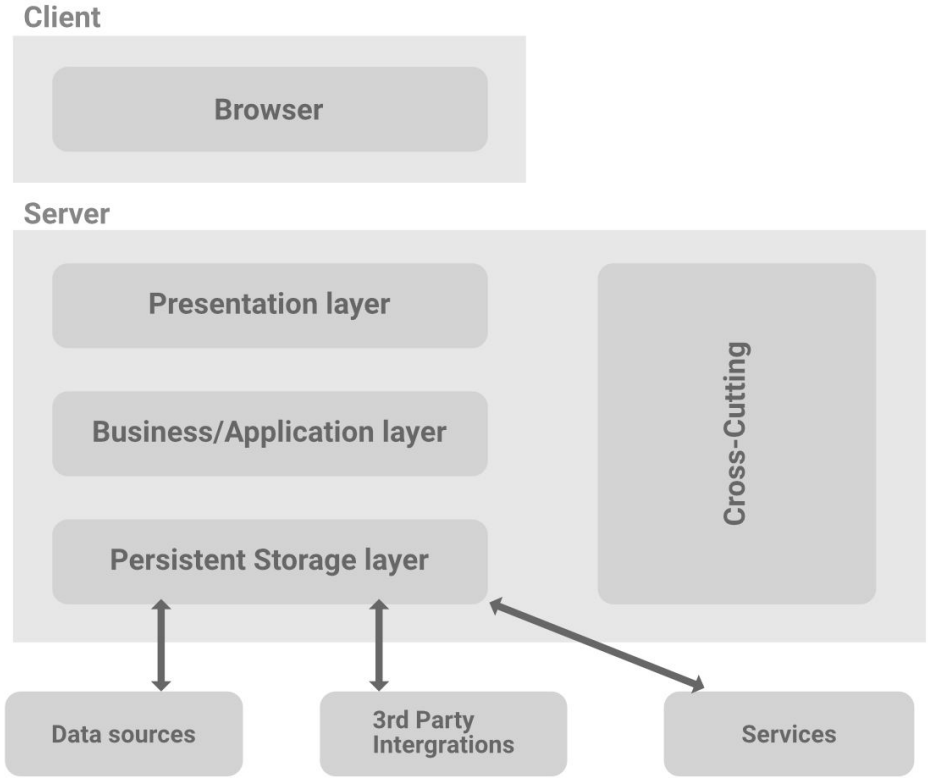
Components

1) Web Browser: the client-side component or the front-end component is the key component that interacts with the user, receives the input and manages the presentation logic while controlling user interactions with the application.

2) Web Server: also known as the backend component or the server-side component handles the business logic and processes the user requests by routing the requests to the right component and managing the entire application operations.

3) Database Server: provides the required data for the application. It handles data-related tasks.

Web Application Three Tier Architecture Layers



- Presentation Layer: This layer is accessible to the client via a browser and it includes user interface components
- Business Layer: It is also referred to as a Business Logic or Domain Logic or Application Layer. It accepts the user's request from the browser, processes it, and regulates the routes through which the data will be accessed.
- Persistence Layer: It is also referred to as a storage or data access layer. This layer collects all the data calls and provides access to the persistent storage of an application.

Front-end Technologies

HTML

HTML or Hypertext Markup Language is a popular standard markup language that enables developers to structure web page contents using a series of page elements.

CSS

CSS or Cascading Style Sheets is a popular style sheet language. Using CSS, you can define a style for elements and reuse them multiple times.

JavaScript

JavaScript or JS is the most popular client-side programming language which is used by more than 90% of websites in recent times.

React

- React is an open-source JavaScript.
- React benefits enables developers to easily create high-quality dynamic web applications with minimal code and effort.
- ReactJS is easy to learn and use.
- The code is reusable.
- ReactJS uses a virtual DOM which means concerned elements are updated when a change is made instead of the entire DOM tree being rewritten.

Vue.js

- Vue.js is an open-source JavaScript framework
- This framework enables developers to easily build UI interfaces for web, desktop and mobile devices.
- Vue.js is largely adapted by Chinese companies such as Alibaba and Xiomi.
- GitLab, Adobe are some of the popular companies that use Vue.js.

Angular.js

- Angular is an open-source web app framework developed by Google

Back-end Technologies

Technologies

- Node.Js
- Java
- Python
- PHP Laravel
- Go
- Ruby
- .Net

Node.js

- Node.js is an open-source cross-platform runtime environment
- Developers use JavaScript to build node.js applications

Java

- Java is an object-oriented and class-based programming language that enables developers to write code and run it on any platform using Java virtual machine (JVM) environment.

Python

- Python is an open-source high-level programming language
- Google, Spotify, Instagram and Facebook are some of the popular IT giants that use Python.

PHP Laravel

- PHP Laravel is a PHP framework that helps developers to build custom web apps with ease. It is an open-source framework

Go

- Go Programming language comes from the IT giant Google

.NET

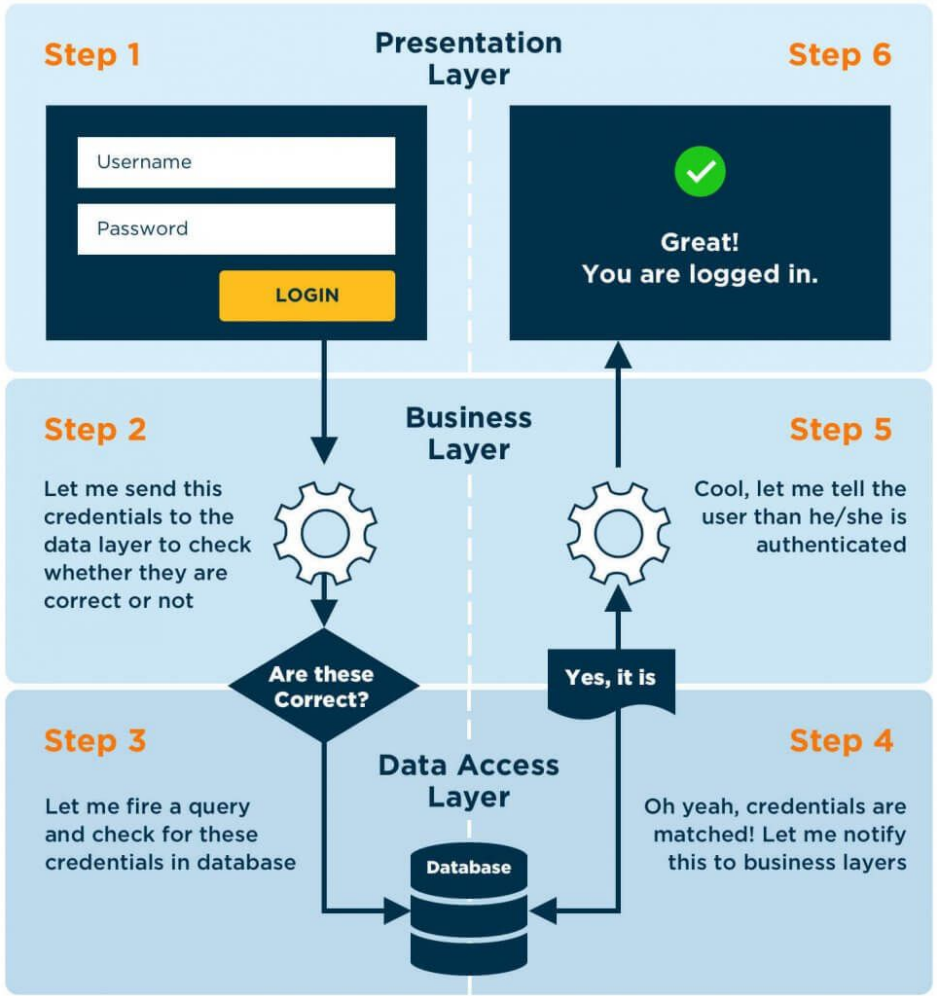
- .NET is a software development framework developed by Microsoft

Ruby

- Ruby is a popular programming language
- Combined with Rails framework, it allows developers to quickly build and deploy apps.
- Airbnb, GitHub, Bloomberg and Etsy are a few popular companies that use Ruby.

Some other parts of the web application which is separated from the main layers that exist in the architecture are

- Cross-cutting code: This part handles communications, operational management, and security. It affects all parts of the system but should never mix with them.
- Third-party integrations: Using third-party APIs we can integrate payment gateways, social logins, GDSs in travel websites, etc.


Step 1
Presentation Layer
Step 6
Username
Password
LOGIN
Great!
You are logged in.
Step 2
Business Layer
Step 5
Let me send this credentials to the data layer to check whether they are correct or not
Cool, let me tell the user than he/she is authenticated
Are these Correct?
Yes, it is
Step 3
Data Access Layer
Step 4
Let me fire a query and check for these credentials in database
Oh yeah, credentials are matched! Let me notify this to business layers
Database

Types of Web Application Architecture

Monolithic Architecture

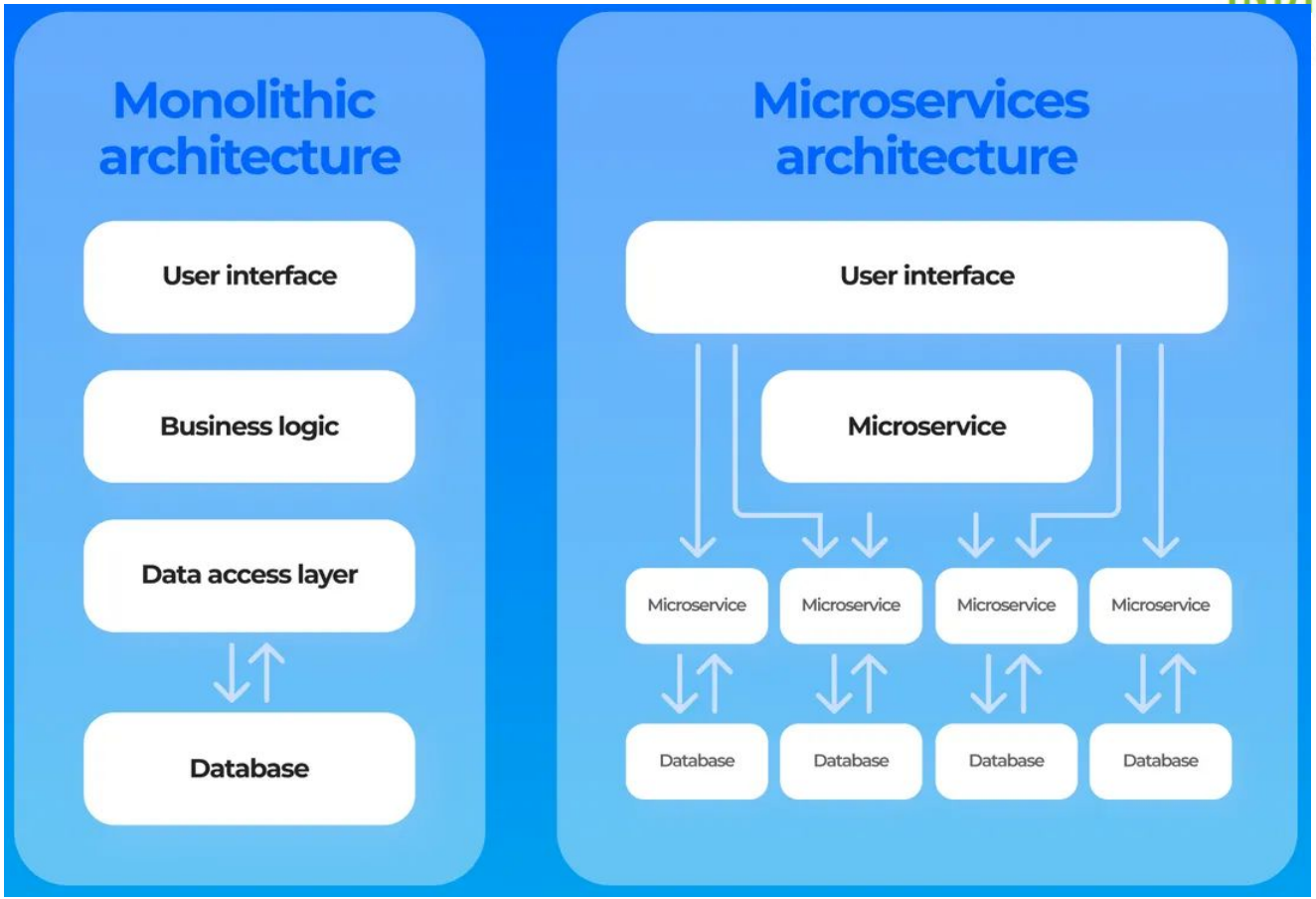
- In monolithic architecture, all the components coexist as a single module managed by a single team—everything is bundled together.
- Its centralized codebase and repository make testing and debugging simpler.
- For smaller applications, monolithic architecture is often the best solution.

Advantages of monolithic applications:

- Simple to develop.
- Easier to deploy as only a single jar/war file is deployed.
- The problems of network latency and security are relatively less in comparison to other architectures.
- Developers need not learn different applications, they can keep their focus on one application.

Disadvantages of Monolithic applications:

- It becomes too large with time and hence, difficult to manage.
- We need to redeploy the whole application, even for a small change.
- For any new developer joining the project, it is very difficult to understand the logic of a large Monolithic application even if his responsibility is related to a single functionality.
- Even if a single part of the application is facing a large load/traffic, we need to deploy the instances of the entire application in multiple servers. It is very inefficient and takes up more resources unnecessarily. Hence, horizontal scaling is not feasible in monolithic applications.
- It is very difficult to adopt any new technology which is well suited for a particular functionality as it affects the entire application, both in terms of time and cost.
- It is not very reliable, as a single bug in any module can bring down the entire monolithic application.


Monolithic architecture
User interface
Business logic
Data access layer
Database
Microservices architecture
User interface
Microservice
Microservice
Microservice
Microservice
Microservice
Database
Database
Database
Database

Microservices Architecture

- The idea behind microservices is breaking down applications into smaller, more digestible pieces so that they’re easier to build and maintain.
- These small, separately maintained pieces work together to run the application as a whole.
- Netflix uses this architecture.

Advantages of microservices:

- It is easy to manage as it is relatively smaller.
- If there's any update in one of the microservices, then we need to redeploy only that microservice.
- Microservices are self-contained and, hence, deployed independently. Their start-up and deployment times are relatively less.
- It is very easy for a new developer to onboard the project as he needs to understand only a particular microservice providing the functionality he will be working on and not the whole system.
- If a particular microservice is facing a large load because of the users using that functionality in excess, then we need to scale out that microservice only. Hence, the microservices architecture supports horizontal scaling.
- Each microservice can use different technology based on the business requirements.
- If a particular microservice goes down due to some bug, then it doesn't affect other microservices and the whole system remains intact and continues providing other functionalities to the users.

Disadvantages of microservices:

- Its complexity increases with the increase in a number of microservices.
- Skilled developers are required to work with microservices architecture, which can identify the microservices and manage their inter-communications.
- Microservices are costly in terms of network usage as they need to interact with each other and all these remote calls result in network latency.
- Microservices are less secure relative to monolithic applications due to the inter-services communication over the network.

Serverless Architecture

- In this type of web application architecture, an application developer consults a third-party cloud infrastructure services provider for outsourcing server as well as infrastructure management.
- It helps the developers to write and deploy the code without worrying about the infrastructure.
- The servers are provided by third-party service providers like Microsoft Azure, Amazon Web Services (AWS), or Google Cloud.

Advantages of Serverless Architecture

- The cost incurred by a serverless application is based on the number of function executions, measured in milliseconds instead of hours.
- Process agility: Smaller deployable units result in faster delivery of features to the market, increasing the ability to adapt to change.
- Reduced operational costs
- Zero system administration.
- Easier operational management.
- Faster set up.
- Scalable, no need to worry about the number of concurrent requests.

Disadvantages of Serverless Architecture

- Reduced overall control.
- Vendor lock-in requires more trust for a third-party provider.
- Cost is unpredictable because the number of executions is not predefined
- Multi-tenancy means it’s technically possible that neighbour functions could hog the system resources behind the scenes.
- Testing locally becomes tricky.