Issue of Web Technology

Avinash Maskey

Introduction

- When building a web application, there are three main principles to bear in mind.
- From a **customer’s point of view,** the application should be simple, aesthetically pleasing, and address most of their problems.
- From the **business aspect,** a web application should stay aligned with its product/market fit.
- From a **software engineer’s perspective,** a web application should be scalable, functional, and able to withstand high traffic loads.
- All these issues are addressed in the web application’s architecture.

What is Web Application Architecture? (1)

So, what is a web application and how is it different from a website?

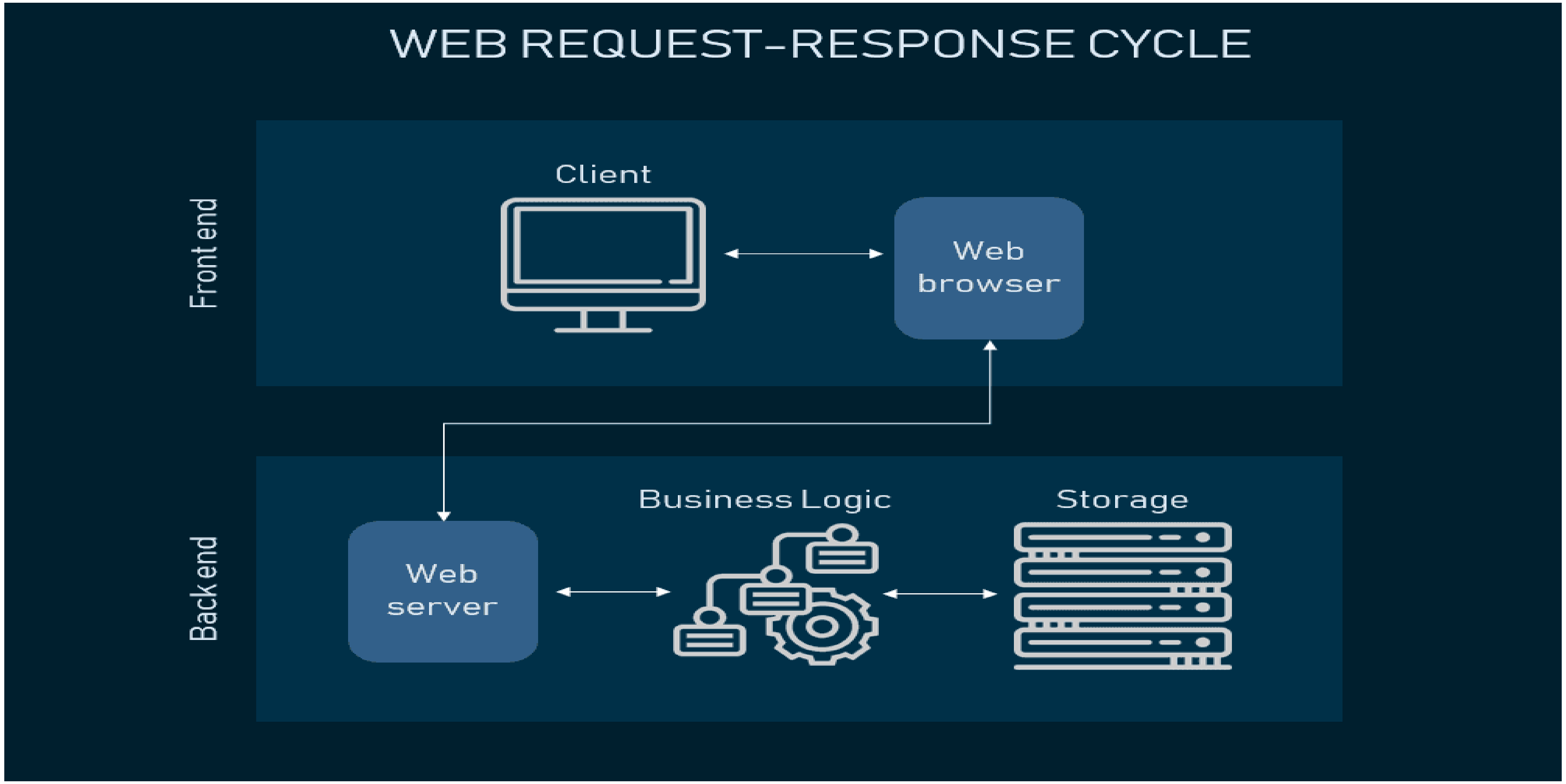
- The basic definition of a web application is a program that runs on a browser. It’s not a website, but the line between the two is ambiguous. To differentiate a web application from a website, remember these three formal characteristics.
- A web application:
  - addresses a particular problem, even if it’s simply finding some information
  - is as interactive as a desktop application
  - has a Content Management System
- A website is traditionally understood to simply be a combination of static pages. But today, most websites consist of both static and dynamic pages, which makes almost all modern websites - you guessed it! - web applications.

What is Web Application Architecture? (2)

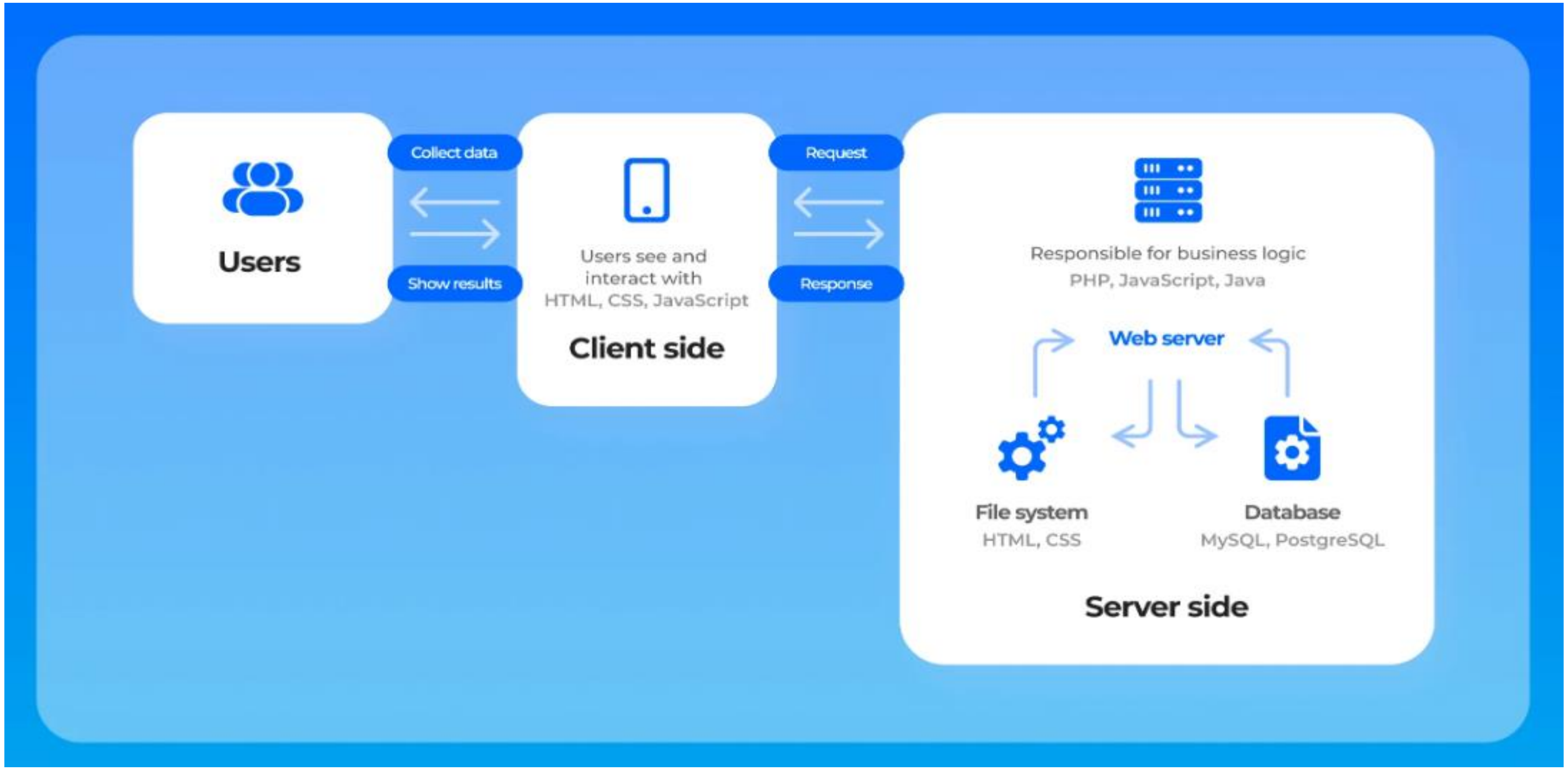
- Your computer, or smartphone, or any other device you’re browsing with is called a client. The other half of the web equation is called a server because it serves you the data you request. Their communication is called a client-server model, whose main concern is receiving your request and delivering the response back.
- **Web application architecture** is a mechanism that determines how application components communicate with each other. Or, in other words, the way the client and the server are connected is established by web application architecture.

How does the web request work? (1)

- To understand the components of web application architecture, we need to understand how they are used in performing the most basic action - receiving and responding to a web request.



How does the web request work? (2)



How does the web request work? (3)

- Let’s look at Amazon.com to illustrate our explanation.
  - First, *you visit amazon.com.* You type in the URL and as you hit Enter, your browser prepares to recognize this URL, because it needs to know the address of the server where the page is located. So it sends your request to the Domain Name Center (DNS), a repository of domain names and their IP addresses. If you've already visited Amazon from the same browser, it will pull the address from the cache. Then, a browser sends the request to the found IP address using the HTTPS protocol.
  - Second, the *web server processes the request.* The web server where Amazon.com is located catches the request and sends it to the storage area to locate the page and all data that follows with it. But its route is held via Business Logic (also called Domain Logic and Application Logic). BL manages how each piece of data is accessed and determines this workflow specifically for each application. As BL processes the request, it sends it to storage to locate the looked-for data.
  - Third, *you receive your data.* Your response travels back to you and you see the content of the web page on your display. The graphical interface you see when scrolling Amazon's or any other website is called the front end of an application - it depicts all UX and UI components so that a user can access the information they came looking for.

Layers of Web Application Architecture (1)

- **Web application architecture** is structured into multiple layers, each with its distinct responsibilities.
- This layered approach promotes separation of concerns, making the development, maintenance, and scaling of web applications more manageable and efficient.
- The three primary layers are — Presentation Layer, Business Logic Layer (Application Layer), and Data Service Layer (Persistence Layer) serve as the foundation for most web applications today.
- **Presentation Layer**
  - The Presentation Layer is the user interface of the web application. It's what users interact with directly. This layer is responsible for displaying data to the user and collecting user input to be processed or sent to the server. It is built using technologies like HTML, CSS, and JavaScript, and it may use frameworks and libraries such as React, Angular, or Vue.js to enhance interactivity and user experience.

Layers of Web Application Architecture (2)

- **Responsibilities of Presentation Layer**
  - Rendering user interface components.
  - Capturing user actions (clicks, inputs, navigation).
  - Sending and receiving data from the Business Logic Layer.
  - Ensuring responsive and accessible design across various devices and screen sizes.

- **Business Logic Layer (Application Layer)**
  - The Business Logic Layer, or Application Layer, is the core of the application, containing the business rules and logic specific to the application. This layer processes user inputs, interacts with the Data Service Layer to retrieve or update data, and determines the flow of the application based on business objectives and rules.

Layers of Web Application Architecture (3)

- **Responsibilities of Business Logic Layer (Application Layer)**
  - Processing data input from the Presentation Layer.
  - Implementing application-specific rules and logic (e.g., calculations, data transformations).
  - Managing user sessions and authentication.
  - Making decisions and performing operations based on user inputs and data from the Data Service Layer.
  - Serving as the intermediary between the Presentation Layer and the Data Service Layer.

Layers of Web Application Architecture (4)

- **Data Service Layer (Persistence Layer)**
  - The Data Service Layer, also known as the Persistence Layer, handles data storage and retrieval operations. It interacts with databases or other storage systems, ensuring that data is accurately saved and efficiently retrieved as needed by the Business Logic Layer. This layer abstracts the complexity of data access from the rest of the application, providing a simplified interface for data operations.

- **Responsibilities of Data Service Layer:**
  - Interacting with databases or data storage systems to retrieve, update, insert, and delete data.
  - Implementing data access patterns and strategies (e.g., Repository pattern).
  - Managing transactions, data integrity, and security.
  - Caching data to improve performance.
  - Ensuring that the application's data storage needs are met efficiently and securely.

Architectural Issues in Web Technology (1)

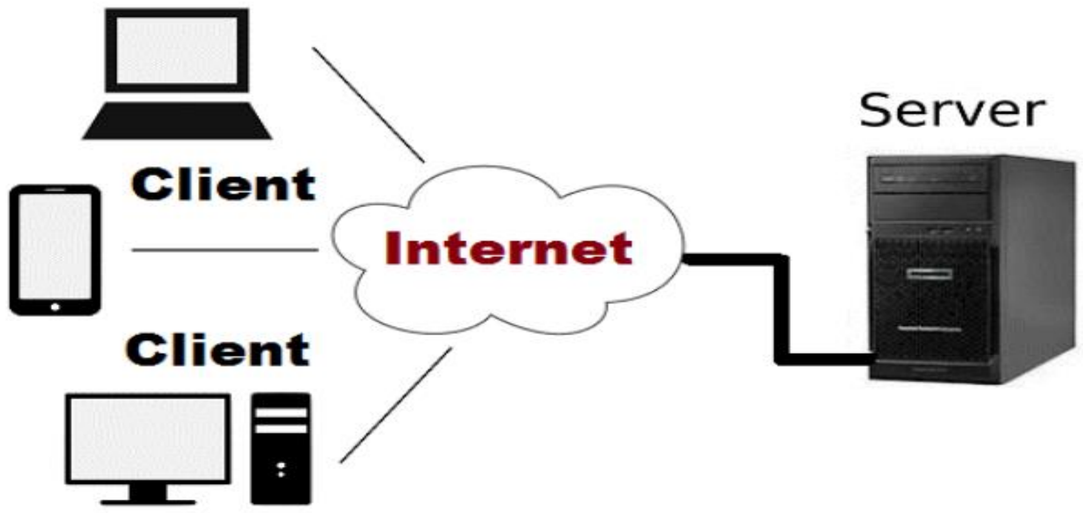
- When discussing the architecture of web technology, it's essential to recognize that web applications have evolved from simple, static web pages to complex, interactive applications offering rich user experiences similar to desktop applications.
- This evolution has **introduced several architectural challenges:**
  - **Scalability:** As the number of users grows, web applications must efficiently manage increased loads. Scalability can be horizontal (adding more machines) or vertical (adding more power to existing machines), and choosing the right strategy depends on the application's specific needs.
  - **Performance:** Web applications must minimize latency to provide a smooth user experience. This involves optimizing the front-end (e.g., through efficient JavaScript, minimizing HTTP requests) and the back-end (e.g., database optimization, efficient algorithms).
  - **Security:** Web applications are exposed to a wide range of security threats, from data breaches to denial of service attacks. Architects must ensure secure data handling, implement proper authentication and authorization, and protect against common vulnerabilities like SQL injection and cross-site scripting (XSS).

Architectural Issues in Web Technology (2)

- **Maintainability:** As web applications grow, maintaining codebase and infrastructure becomes challenging. A clean, modular architecture is essential for facilitating updates, bug fixes, and adding new features without impacting existing functionality.
- **Cross-Platform Compatibility:** Web applications must work seamlessly across different devices, browsers, and operating systems. This requires responsive design, browser compatibility testing, and potentially the use of web standards like HTML5 and CSS3.

Client/Server Architecture

- The ***client/server architecture*** is based on the hardware and software components that interact to form a system.
- The system includes three main components:
  - **Client:** The client is any computer process that requests service from the server.
  - **Server:** The server is any computer process providing services to the clients.
  - **Communication Middleware:** The communication middleware is any computer process through which clients and servers communicate and is also known as ***communication layer***.



Client/Server Architecture (Contd.)

- **Advantages:**
  - All the data and resources are controlled by server in this way all data and resources are very consistent.
  - You can easily increase the number of client in this architecture at any time. This all increases the scalability of the network.
  - This is very easy to maintain you can easily repair, replace or add clients in this network. The independence of the changes also known as encapsulation.
  - This network is very easy to use and it is not complicated.
- **Disadvantages:**
  - Traffic is a big problem in this network.
  - When you add large numbers of the client with server this network will be more complicated.
  - When the server goes down all the clients are not able to send their request. The whole work will be stopped.
  - The hardware and software are very expensive.
  - The client does not have resources for each resource they need to request the server because of all resources exist on server.

Client/Server Architecture (Contd.)

- There are basically three-types of client/server architectures:
  - **One-tier architecture:**

    In this architecture, it keeps all of the elements of an application, including the interface, middleware and back-end data, in one place.
  - **Two-tier Architecture:**

    In this architecture, the ***user interface*** and ***application programs*** are placed on the ***client side*** and ***database system*** on the ***server side***. The application programs that resides at the client side invoke the DBMS at the server side.
  - **Three-tier Architecture:**

    This architecture adds ***application server*** between the ***client*** and ***database server***. The client communicates with the application server, which in turn communicates with the database server. The application server stores the business rule (procedures and constraints) used for accessing data from database server.

Fig: One Tier Client Server Architecture

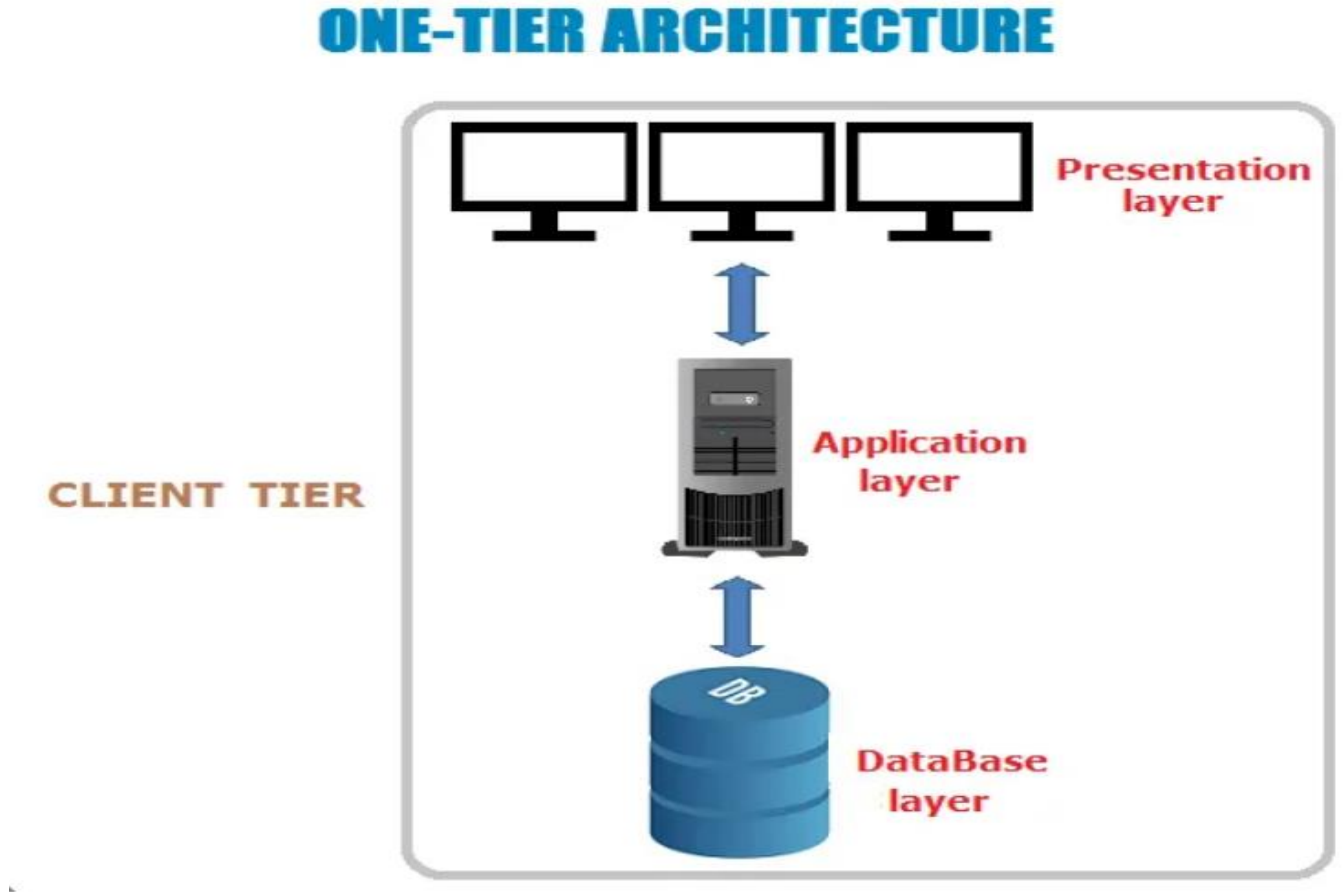


Fig: Two Tier Client Server Architecture

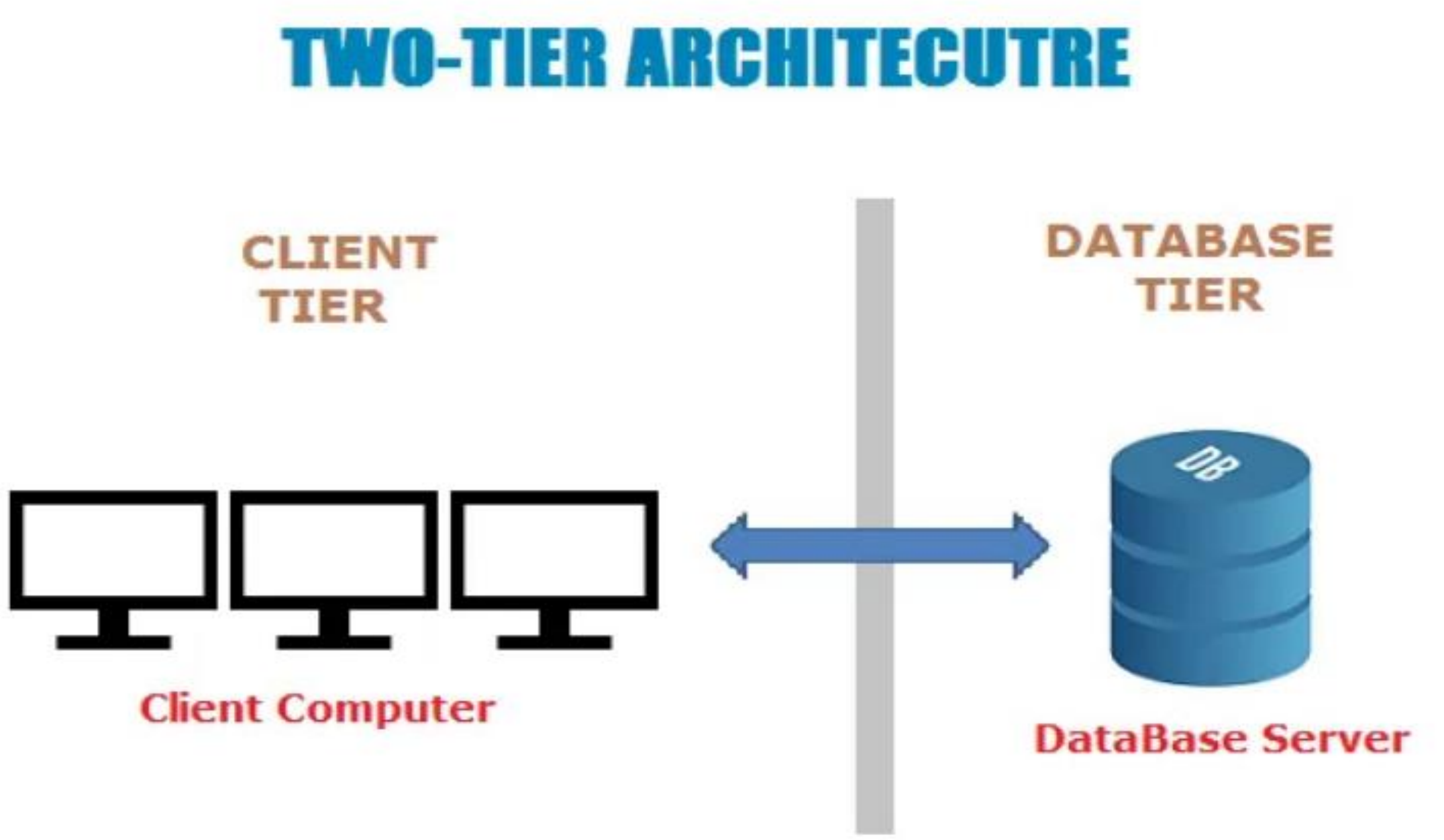
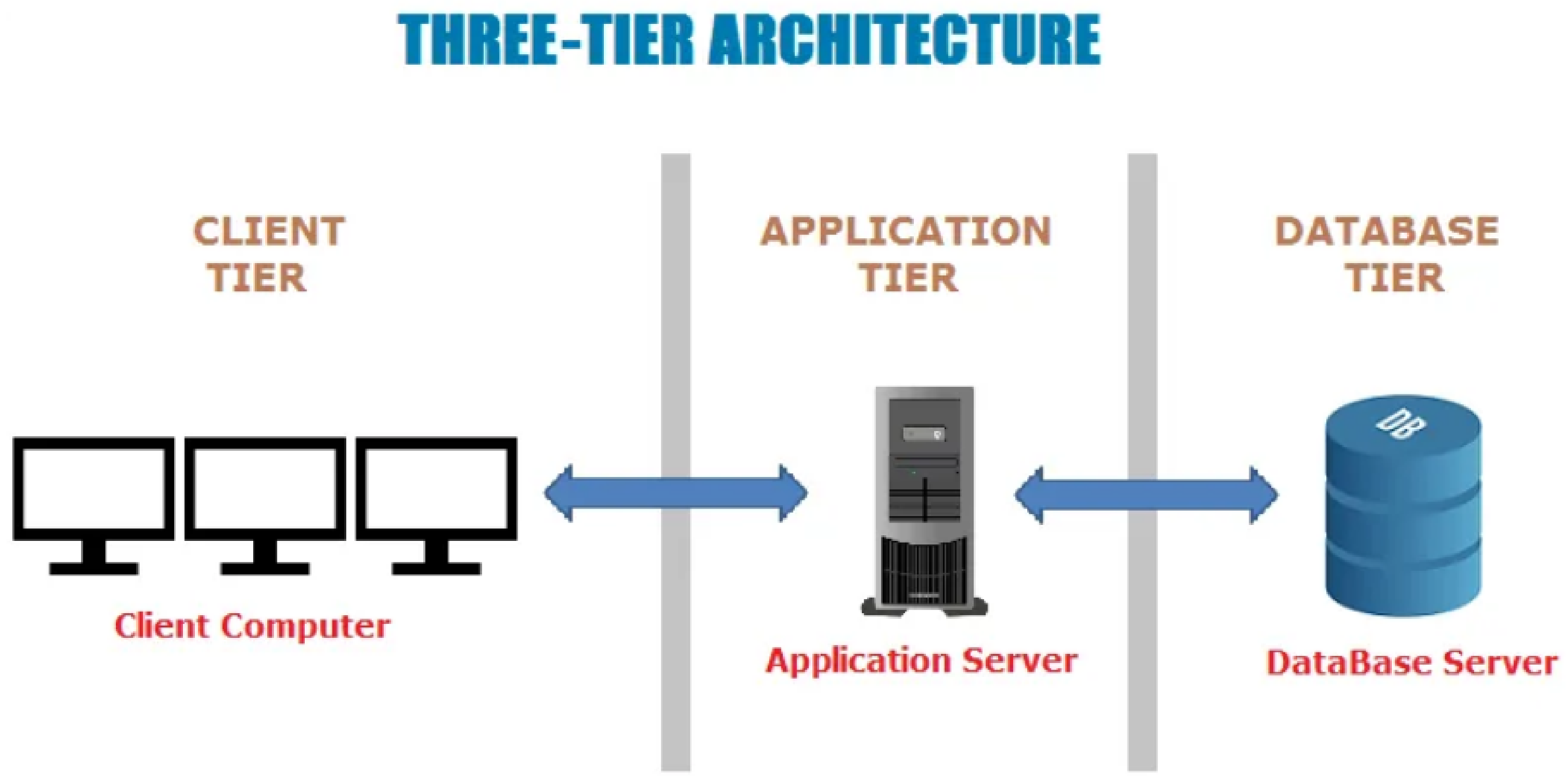


Fig: Three Tier Client Server Architecture



What is N-Tier?

- The **N-Tier architecture** is an extension of the 3-tier architecture that allows developers to further separate the different components or concerns of an application into distinct, interconnected layers.
- Each layer in an N-Tier architecture has a specific responsibility and operates independently of the others, communicating through well-defined interfaces.
- This separation enhances the application's scalability, maintainability, and flexibility, allowing for the independent updating, deployment, and scaling of each layer.
- In N-Tier architecture, "N" refers to any number of separate tiers (or layers) that make up the entire application stack.
- Beyond the basic three layers (presentation, business logic, and data service), you can introduce additional layers like caching, application services, API layers, and more, depending on the application's complexity and requirements.
- N-Tier architecture is also known as ***Multi-tier architecture.***

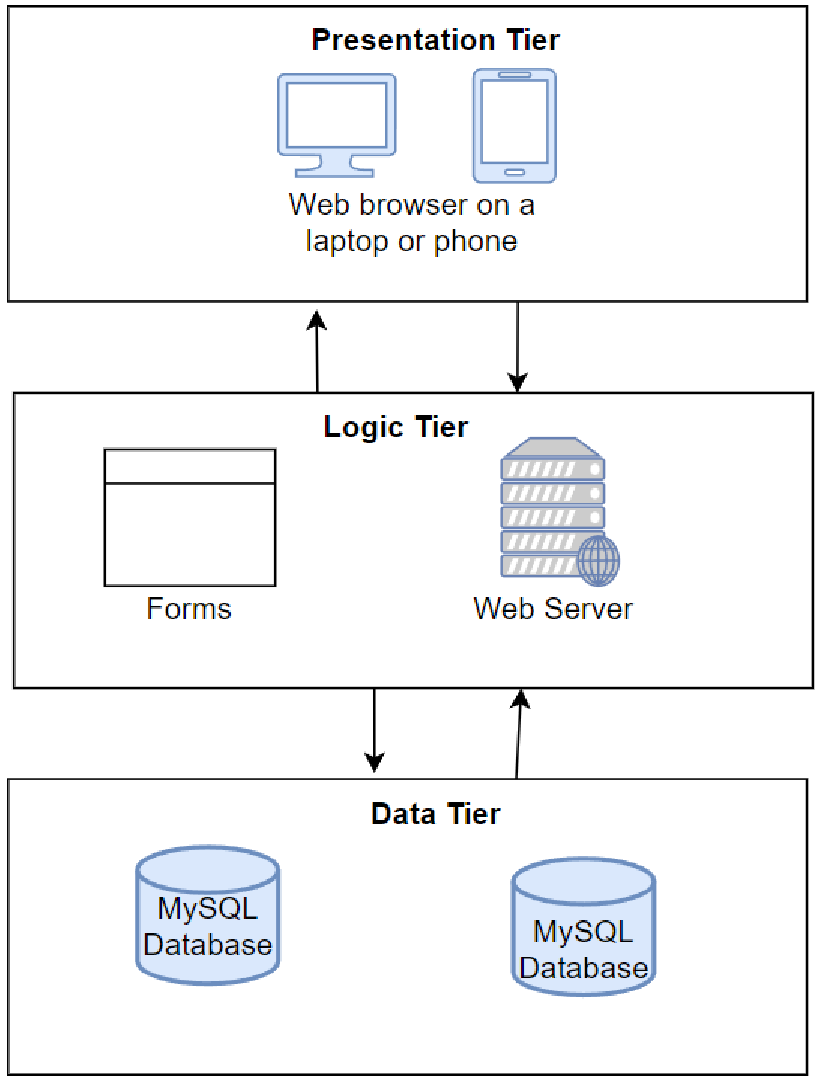
Common Layers in N-Tier Architecture (1)

- **Presentation Layer (Client Layer):** This is the topmost layer of the application, responsible for presenting the user interface and facilitating user interaction. It is developed using technologies such as HTML, CSS, JavaScript, and frameworks like React or Angular.
- **Application Layer (Business Logic Layer):** It contains the business logic and rules of the application. This layer processes user requests, performs operations based on business logic, and makes calls to lower layers for data.
- **Service Layer:** Optional layer that acts as an intermediary between the business logic and data access layers, providing a set of reusable services or APIs. This layer abstracts the business logic operations and makes them available as services.
- **Data Access Layer (DAL):** This layer abstracts the access to the data storage, providing an interface to perform CRUD (Create, Read, Update, Delete) operations on the database without exposing the details of the database to the other layers.

Common Layers in N-Tier Architecture (2)

- **Persistence Layer:** Sometimes considered part of the DAL or as a separate layer, it is responsible for managing the database connections and transactions, ensuring data integrity and persistence.
- **Caching Layer:** An optional layer aimed at improving the performance of the application by temporarily storing frequently accessed data in memory.
- **API Layer:** For applications that interact with other applications or services, an API layer provides a set of endpoints for accessing the application's functionality programmatically.
- **Integration Layer:** This layer handles communication with external services, third-party APIs, and integrates different parts of the application or different applications within the ecosystem.

Figure: N-tier Architecture



THANK YOU!