2. Issues in Web Technology

2.1 Architectural issues of Web Layer

The Web Layer, also referred to as the UI layer in software architecture, is responsible for handling user interactions and presenting content via the internet. It focuses on the user interface and the behavior of the application, managing tasks like handling user interactions and events. While the web layer may contain some logic, core application logic is typically located in the services layer. The architecture of the Web Layer addresses many issues that impact both functionality and performance.

The three Layers within the Web Layer (UI layer) are:

1. **HTML-The Content Layer**: The content layer is where you store all the content that your customers want to read or look at. This includes text and images as well as multimedia. It's also important to make sure that every aspect of your site is represented in the content layer. That way, your customers who have JavaScript turned off or can't view CSS will still have access to the entire site, if not all the functionality.
2. **CSS - the Styles Layer**: Store all your styles for your Web site in an external style sheet. This defines the way the pages should look, and you can have separate style sheets for various media types. Store your CSS in an external style sheet so that you can get the benefits of the style layer across the site.
3. **JavaScript - the Behavior Layer**: JavaScript is the most commonly used language for writing the behavior layer; ASP, CGI (Common Gateway Interface) and PHP can also generate Web page behaviors. However, when most developers refer to the behavior layer, they mean that layer that is activated directly in the Web browser - so JavaScript is nearly always the language of choice. You use this layer to interact directly with the DOM or Document Object Model.

Basic Protocols

Short Overview of basic Protocols

Network protocols are a set of rules outlining how connected devices communicate across a network to exchange information easily and safely. They constitute a set of rules for routing and addressing packets of data, allowing them to travel across networks and reach their correct destinations. Protocols serve as a common language for devices, enabling communication regardless of differences in software, hardware, or internal processes.

The Internet Protocol (IP) is the method or protocol by which data is sent from one computer to another on the internet. Each computer (known as a host) on the internet has at least one IP address that uniquely identifies it from all other computers on the internet.

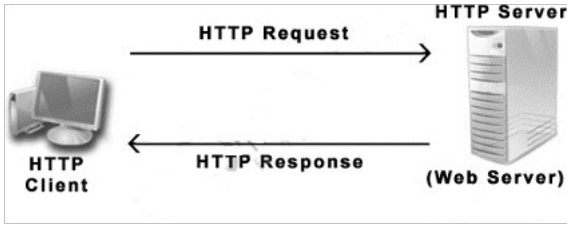
Data traversing the Internet is divided into smaller pieces called packets. IP information is attached to each packet, helping routers send packets to the correct destination. Every device or domain connecting to the Internet is assigned an IP address, and as packets are directed to the IP address attached to them, data arrives where it is needed.

Once the packets reach their destination, they are handled differently depending on the transport protocol used in combination with IP. The most common transport protocols are TCP and UDP.

IP is the defining set of protocols that enable the modern internet. It was initially defined in May 1974 in a paper titled "A Protocol for Packet Network Intercommunication," published by the Institute of Electrical and Electronics Engineers and authored by Vinton Cerf and Robert Kahn.

Some protocols used in internet are described below

1. **Transmission Control Protocol (TCP):** TCP, or Transmission Control Protocol, is one of the core protocols of the Internet Protocol (IP) suite, providing reliable, connection-oriented communication between devices on a network. TCP ensures the successful and ordered delivery of data packets between applications running on devices across a network. It breaks data into packets, sends them, and then ensures they are received correctly.
2. **Internet Protocol (IP):** The IP protocol, or Internet Protocol, is a fundamental communication protocol that facilitates the transmission of data between devices on a network. It is a key component of the Internet Protocol Suite, commonly referred to as TCP/IP. The IP protocol provides a standardized set of rules for addressing and routing data packets across networks.
3. **Hypertext Transfer Protocol (HTTP)**: HTTP, or Hypertext Transfer Protocol, is the foundation of data communication on the World Wide Web. It is an application layer protocol that facilitates the transfer of information between a client (such as a web browser) and a server. HTTP is a stateless protocol, meaning that each request from a client to a server is independent and doesn't retain any information about previous requests. HTTP typically uses port 80 for communication



The client, in this case the browser communicates with the server, requesting a web page. The server processes that request and responds by sending the web page contents to the browser.

A browser works as an HTTP client because it sends requests to an HTTP server which is called Web server. The Web Server then sends responses back to the client. The standard and default port for HTTP servers to listen on is 80 but it can be changed to any other port like 8080 etc.

HTTP is connectionless: After a request is made, the client disconnects from the server and waits for a response. The server must re-establish the connection after it processes the request.

HTTP is media independent: Any type of data can be sent by HTTP as long as both the client and server know how to handle the data content. How content is handled is determined by the MIME (Multipurpose Internet Mail Extensions- a specification for formatting non-ASCII messages so that they can be sent over the Internet. Many e-mail clients now support MIME, which enables them to send and receive graphics, audio, and video files via the Internet) specification.

HTTP is stateless: This is a direct result of HTTP's being connectionless. The server and client are aware of each other only during a request. Afterwards, each forgets the other. For this reason, neither the client nor the browser can retain information between different requests across the web pages.

4. **Hypertext Transfer Protocol Secure (HTTPS)**: HTTPS is a secure version of HTTP. It adds a layer of encryption through SSL/TLS protocols to ensure the confidentiality and integrity of data during transmission. HTTPS URLs begin with "https://" and uses port 443 for communication.

Security Features:

- Encryption: HTTPS encrypts the data being transmitted, making it more difficult for unauthorized parties to intercept and decipher.
- Data Integrity: It ensures that the data remains unchanged during transmission, preventing tampering or modification by attackers.
- Authentication: HTTPS provides a way for the user's browser to verify the identity of the website's server, helping to prevent man-in-the-middle attacks.

5. **File Transfer Protocol (FTP)**: FTP stands for File Transfer Protocol, and it is a standard network protocol used to transfer files from one host to another over a TCP-based network, such as the internet. It is commonly used for uploading and downloading files between a client and a server.
6. **Simple Mail Transfer Protocol (SMTP):** SMTP stands for Simple Mail Transfer Protocol. It is a communication protocol widely used for electronic mail (email) transmission over the internet. SMTP is the set of rules governing the interaction between the mail servers to send and receive emails.
7. **User Datagram Protocol** : UDP, or User Datagram Protocol, is a connectionless transport protocol that operates at the transport layer of the Internet Protocol (IP) suite. Unlike TCP (Transmission Control Protocol), UDP does not establish a reliable, connection-oriented communication channel before data transfer. Instead, it provides a lightweight, best-effort delivery mechanism for sending data between devices on a network.
8. **Post Office Protocol (POP)** and **Internet Message Access Protocol (IMAP):** These protocols are used by email clients to retrieve messages from a mail server. POP is known for downloading emails to the local device, while IMAP allows users to manage emails directly on the server.
9. **Domain Name System (DNS)**: Domain Name System (DNS) is a protocol that allows us to use human readable names to communicate over networks, rather than having to manage and memorize IP addresses. DNS translates human-readable domain names into IP addresses. It is crucial for locating resources on the Internet by associating domain names with numeric IP addresses.
10. **Dynamic Host Configuration Protocol (DHCP)**: DHCP is used to automatically assign IP addresses and other network configuration information to devices on a network.
11. **Border Gateway Protocol (BGP)**: BGP is a routing protocol used to exchange routing and reachability information between different autonomous systems on the Internet.
12. **WebSocket**: WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection. It is commonly used in web applications for real-time communication.

2.2 Tier Technology

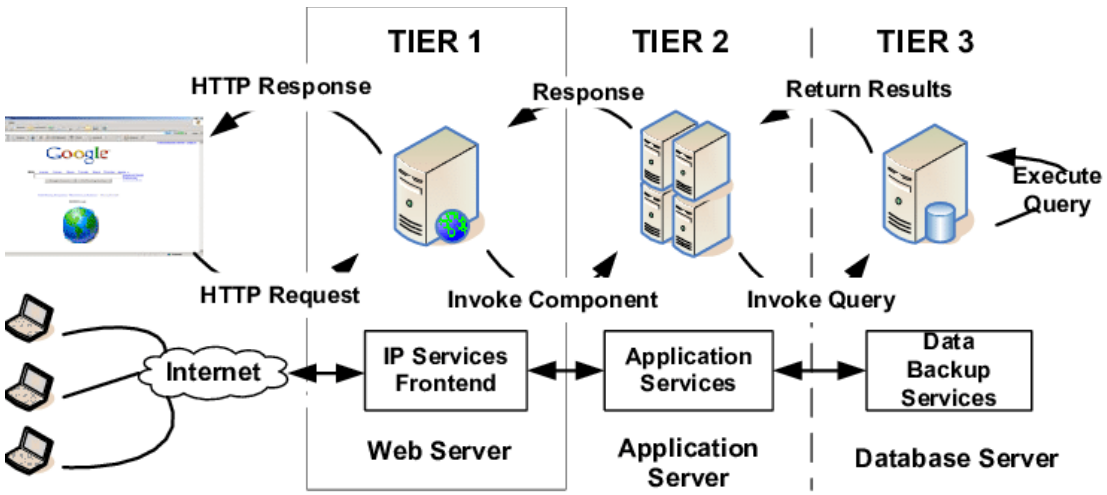
Tier technology in web technology refers to the architectural approach of dividing a web application into multiple logical or physical layers (tiers) to separate concerns, improve scalability, and enhance maintainability. It generally involves structuring a web application or system into different tiers or layers, where each tier is responsible for specific functionality and interacts with other tiers in a structured manner. The concept of tiers is closely related to multitier architecture, which divides the application into distinct layers to achieve scalability, maintainability, and separation of concerns. Each tier handles different aspects of the application, and they communicate with one another to deliver the desired functionality.

The most common tiers in web architecture are described below:

1. **Presentation Tier (Front-end)**:
   - This is the client-side or user interface (UI) part of the application, often built with HTML, CSS, and JavaScript.
   - The presentation tier is responsible for interacting with the user and presenting data in a meaningful way.
   - Examples: Web browsers, mobile apps, or any platform where the end user interacts with the application.

2. **Logic Tier (Middle-tier):**
   - This is the business logic layer of the application.
   - It handles processing the data, applying business rules, and making decisions based on inputs.
   - The middle tier is often developed using server-side programming languages like Node.js, Python (Django, Flask), Java (Spring), Ruby (Rails), etc.
   - It is responsible for tasks like user authentication, data processing, and API interactions.

3. **Data Tier (Back-end):**
   - This is the data layer, where all the data storage and retrieval happen.
   - It typically involves databases like MySQL, PostgreSQL, MongoDB, etc.
   - The data tier is responsible for managing, storing, and querying data in response to requests from the logic tier.
   - It may also include a server-side API for interacting with the front-end.



Additional Tiers (Optional):

In complex systems, we might encounter additional tiers:

1. **Caching Tier**: Often used for optimizing performance, a caching tier stores frequently accessed data in memory (e.g., Redis, Memcached) to reduce the load on the data tier.
2. **Network/Infrastructure Tier**: This tier deals with things like load balancers, content delivery networks (CDNs), and networking concerns for scaling applications and ensuring performance.

Levels of Architectural Structures

Tier technology in web technology refers to the architectural approach of dividing a web app into multiple logical or physical layers (tiers) to separate concerns, improve scalability, and enhance maintainability. Each tier is responsible for specific functionality and interacts with other tiers in a structured manner. Here's a brief explanation of the common tiers:

1. **Single-Tier Architecture (Monolithic)**:

Description: All components (UI, business logic, and data storage) are tightly coupled and run on a single system.

Use Case: Simple applications or prototypes.

Pros: Easy to develop and deploy.

Cons: Difficult to scale, maintain, or update.

Example: A Microsoft Word or Notepad where all functionality (UI, logic, data) is within the application.

2. **Two-Tier Architecture (Client-Server):**

Description:

- **Client Tier**: Handles the user interface and presentation logic.
- **Server Tier**: Manages business logic and data storage.

Use Case: Small to medium-sized applications.

Pros: Better separation of concerns compared to single-tier.

Cons: Limited scalability and tight coupling between business logic and data storage.

Example : ***Client-Server applications***: Applications where the client makes requests to the server, which processes request and returns the results.

3. **Three-Tier Architecture:**

Description:

- **Presentation Tier** (Client Layer): Handles the user interface (e.g. HTML, CSS, JS).
- **Application Tier (Business Logic Layer)**: Manages business rules, workflows, and processing (e.g., server-side code like Node.js, Django).
- **Data Tier (Database Layer)**: Stores and retrieves data (e.g., MySQL, Oracle).

Use Case: Most modern web applications.

Pros:

- Clear separation of concerns.
- Improved scalability, maintainability, and flexibility.

Cons: Increased complexity in deployment and communication between tiers.

4. **N-Tier Architecture:**

Description: Extends the three-tier model by adding more layers for specific functionalities, such as:

- **Caching Tier**: For improving performance (e.g., Redis, Memcached).

- **API Tier**: For exposing services to external systems.
- **Middleware Tier**: For handling communication between layers.

Use Case: Large-scale, complex applications.

Pros:

- Highly modular and scalable.
- Better fault isolation and performance optimization.

Cons:

- Increased complexity in design and maintenance.
- Higher cost of infrastructure and development.

Key Benefits of Tier Technology:

1. **Modularity**: Each tier can be developed, tested, and maintained independently.
2. **Scalability**: Tiers can be scaled individually based on demand (e.g., scaling the database tier separately from the application tier).
3. **Security**: Separation of layers reduces the risk of vulnerabilities spreading across the system.
4. **Reusability**: Business logic and services can be reused across multiple applications.
5. **Flexibility**: Easier to adopt new technologies or frameworks for specific tiers.

By using tier technology, web applications become more organized, scalable, and easier to manage as they grow in complexity.