Part 1:

1.

Client-side web development:

In web development, 'client side' refers to everything in a web application that is displayed or takes place on the client (end user device). This includes what the user sees, such as text, images, and the rest of the UI, along with any actions that an application performs within the user's browser.

Server-side in web development:

Server-side development, sometimes called back-end development , is a type of development that involves programs that run on a server. This type of programming is important because web browsers, or clients, interact with web servers to retrieve information.

Difference between client-side and server-side web development:

A.One major difference between client-side and server-side development is where code runs. In client-side development, the code runs on the client's or user's device. In server-side development, the code runs through a server.

B.The way scripts run is another difference between client-side and server-side development. In client-side scripting, scripts simply run on a device. Often, client-side scripts run in a browser. In contrast, server-side scripts run on a web server.

C. client-side and server-side development also have different purposes. The primary goal of client-side development is to create visual effects and elements for websites, including layouts and user interfaces. Server-side development focuses on the actual content of a webpage, involving tasks like interacting with databases and retrieving information from a web server.

2.

HTTP request:

An HTTP request is made by a client, to a named host, which is located on a server. The aim of the request is to access a resource on the server. To make the request, the client uses components of a URL , which includes the information needed to access the resource.

Types of http request:

A.Post
B.Patch
C.Put
D.Delete
E.Get

3.**JSON :**

JSON is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications

4.Middleware :

Middleware is software and cloud services that provide common services and capabilities to applications and help developers and operators build and deploy applications more efficiently. Middleware acts like the connective tissue between applications, data, and users.

Example:

```
const express = require('express');
const app = express();

// Middleware function
const loggerMiddleware = (req, res, next) => {
console.log(`Received request: ${req.method} ${req.url}`);
next();
};


app.use(loggerMiddleware);


app.get('/', (req, res) => {
res.send('Hello, World!');
});


app.listen(3000, () => {
console.log('Server is running on port 3000');
});
text
```

5.Controller:

In web development, a controller is a component or a class that plays a crucial role in the Model-View-Controller (MVC) architecture. The primary responsibility of a controller is to handle the flow of data and control the interactions between the model (data layer) and the view (presentation layer).

The controller's role in the MVC architecture involves the following tasks:

Here are the key responsibilities of the controller in the MVC architecture:

1. Handling user input: The controller captures user input, such as button clicks or form submissions, and initiates the corresponding actions in response.
2. Updating the model: The controller interacts with the model component to update the data or state based on the user's input or actions.
3. Managing the flow of control: The controller determines the appropriate views to display based on the current state of the model. It decides which views should be shown to the user and handles the transitions between different views.