



SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY::

(AUTONOMOUS)

Siddharth Nagar, Narayanavanam Road – 517583

QUESTION BANK (DESCRIPTIVE)

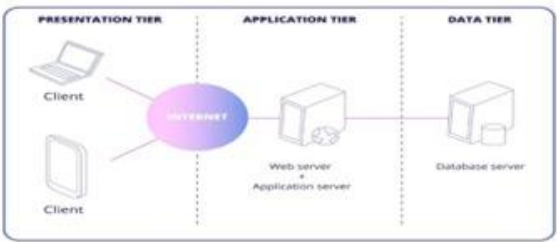
Subject & Code: Web Programming for Artificial Intelligence(20CS0907) Course & Branch:
B.Tech –CSM

Year & Sem : III-B.Tech & II-Sem

Regulation : R20

HTML5, CSS3, XML, JavaScript and JQuery

UNIT –I

1	<p>a What is Web Programming? Briefly explain the Architecture of WEB?</p> <p>Web programming is the process of creating web pages. During this process, the programmer is in charge of shaping the site according to the demands and needs of the company.</p> <p>Web architecture refers to the overall structure of a website or web application, including the way it is designed, implemented, and deployed. It involves the use of technologies and protocols such as HTML, CSS, JavaScript, and HTTP to build and deliver web pages and applications to users.</p> <p>Web architecture consists of several components, including the client, the server, the network, and the database. The client is the web browser or application that the user interacts with, and the server is the computer or group of computers that host the website or web application. The network is the infrastructure that connects the client and the server, such as the internet. The database is a collection of data that is used to store and retrieve information for the website or web application.</p> <p>Web architecture also includes the design and layout of the website or web application, as well as the way it is organized and the relationships between different pages and components. It also includes the way the website or web application is built and maintained, including the use of frameworks and libraries, and the deployment and hosting of the website or web application.</p> 	[L1] [CO1]	[6M]
	<p>b What is HTML? Briefly explain the tags in HTML?</p>	[L1] [CO1]	[6M]

	<p>HTML tags are like keywords which defines that how web browser will format and display the content. With the help of tags, a web browser can distinguish between an HTML content and a simple content. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags.</p> <ul style="list-style-type: none"> • The <!DOCTYPE html> declaration defines that this document is an HTML5 document • The <html> element is the root element of an HTML page • The <head> element contains meta information about the HTML page • The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab) • The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc. • The <h1> element defines a large heading • The <p> element defines a paragraph 		
2	<p>Create a Simple job Registration form using HTML</p> <pre> <!DOCTYPE html> <html> <head> <title> Job Registration form</title> </head> <body> <h2>Job application form</h2> <form> <h3>Personal Information</h3> First name: <input type="text" name="Fname" placeholder="first name"> Last name: <input type="text" name="Lname" placeholder="last name"> Middle name: <input type="text" name="Mname" placeholder="middle name">

 <table> <tr> <td>Current address</td> <td>Permanant address</td> </pre>	[L6] [CO1]	[12M]

```

</tr>

<tr>

<td><textarea rows="5" cols="15" name="Current address" placeholder="type your
address...."></textarea></td>

<td><textarea rows="5" cols="15" name="Permanant address" placeholder="type
your address...."></textarea></td>

</tr>

</table>

<p><strong>Phone:</strong></p>

Home Phone: <input type="number" name="pnumber" size="20" maxlength="5"
placeholder="Country code">

<br><br>

E-mail: <input type="E-mail" name="email" placeholder=" email address">

upload cv: <input type="file" name="cv"

<hr>

<br><br>

<hr>

<strong>Employment Details</strong>

<br>

<p><strong>position:</strong></p>

Professor: <input type="radio" name="position">

Assoc Prof: <input type="radio" name="Position">

Assit Prof: <input type="radio" name="position">

<br>

<p><strong> Are you Currently employed </strong></p>

yes: <input type="radio" name="employee">

No: <input type="radio" name="employee">

<hr>

<strong>Start date:</strong>

<input type="date" name="start date">

<hr>

<strong>Avalability</strong>

<br>

Monday: <input type="checkbox" name="day">

```

	<p>Tuesday: <input type="checkbox" name="day"></p> <p>Wednesday: <input type="checkbox" name="day"></p> <p>Thursday: <input type="checkbox" name="day"></p> <p>Friday: <input type="checkbox" name="day"></p> <p>Saturday: <input type="checkbox" name="day"></p> <p><hr></p> <p>Submit: <input type="Submit" name="submit"></p> <p>Reset: <input type="reset" name="reset"></p> <p></form></p> <p></body></p> <p><html></p>		
3	<p>Explain HTML 5.0 and mention the tag differences between HTML and HTML 5.0?</p> <p>HTML 5.0 Overview</p> <p>HTML 5.0 is the fifth major revision of the Hypertext Markup Language (HTML), which is the standard language for structuring and presenting content on the web. It was introduced to address the needs of modern web applications and provide better support for multimedia, interactivity, and cross-device compatibility.</p> <p>Key Features of HTML 5.0</p> <ol style="list-style-type: none"> 1. Simplified Syntax: Reduces the need for complex plugins by incorporating native support for multimedia (audio and video). 2. Semantics: Introduces new semantic elements (e.g., <header>, <footer>, <article>) to enhance document structure and improve accessibility. 3. Multimedia Support: Native support for <audio> and <video> tags eliminates the dependency on third-party plugins. 4. Graphics: Adds <canvas> and integrates SVG (Scalable Vector Graphics) to enable interactive and animated graphics. <p>Form Enhancements: New input types (e.g., date, email, url) and attributes (e.g., required, placeholder) simplify form validation and user interaction.</p>	[L5] [CO1]	[12M]

Differences in Tags Between HTML and HTML 5.0

Aspect	HTML (Pre-HTML 5.0)	HTML 5.0
Doctype	<code><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"></code>	<code><!DOCTYPE html></code> (simpler and easier to write)
Semantic Tags	No semantic tags; relied on <code><div></code> and <code></code> .	New semantic tags like <code><header></code> , <code><footer></code> , <code><article></code> , <code><section></code> .
Multimedia Tags	Required plugins (e.g., Flash) for multimedia.	Native support for <code><audio></code> and <code><video></code> .
Graphics Tags	Limited support; required external tools.	<code><canvas></code> and SVG support natively for drawing graphics.
Form Elements	Basic input types (<code>text</code> , <code>password</code> , <code>checkbox</code>).	Enhanced form input types (<code>email</code> , <code>url</code> , <code>number</code>) and attributes (<code>autofocus</code> , <code>required</code> , etc).
Scripting	No built-in support for APIs.	New APIs like Geolocation, Web Storage, and Drag-and-Drop.
Deprecated Tags	Tags like <code></code> , <code><center></code> , <code><frame></code> were used.	Deprecated tags are removed to streamline HTML and encourage best practices.

Example: HTML vs HTML 5.0

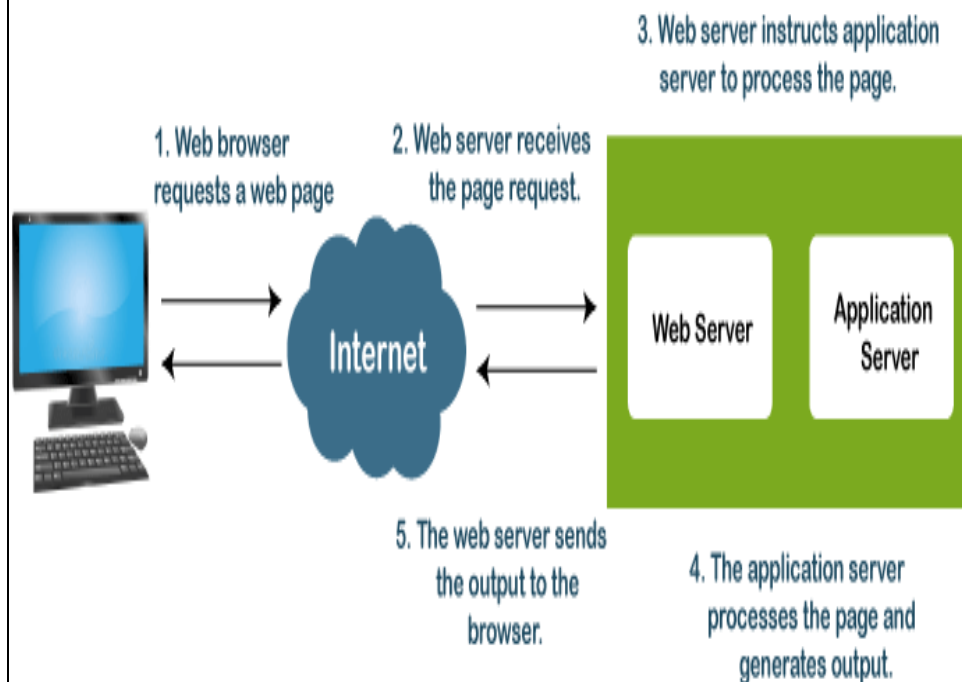
HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head>
    <title>Old HTML</title>
  </head>
  <body>
    <div>Header</div>
    <div>Main Content</div>
  </body>
</html>
```

HTML 5.0

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML 5.0 Example</title>
  </head>
  <body>
    <header>Header</header>
```

		<pre> <main>Main Content</main> <footer>Footer</footer> </body> </html> </pre>		
4		<p>Create a webpage using HTML and add CSS to the webpage.</p> <pre> <!DOCTYPE html> <html> <head> <style> body { background-color: linen; } h1 { color: maroon; margin-left: 40px; } </style> </head> <body> <h1>This is a heading</h1> <p>This is a paragraph.</p> </body> </html> </pre>	[L6] [CO1]	[12M]
5	a	<p>Describe about Internet Application.</p> <p>A web-application is an application program that is usually stored on a remote server, and users can access it through the use of Software known as web-browser.</p> <p>The Flow of the Web Application</p> <p>Let's understand how the flow of the typical web application looks like.</p>	[L2] [CO1]	[6M]



b Explain about JQuery UI and templates.

[L2] [CO1] [6M]

jQuery UI

jQuery UI is a curated set of user interface (UI) interactions, effects, widgets, and themes built on top of the jQuery JavaScript library. It simplifies the development of dynamic and interactive web applications by providing pre-built components and interactions.

Key Features of jQuery UI:

1. **Widgets:** Ready-to-use components like accordions, tabs, sliders, dialogs, date pickers, etc.
2. **Interactions:** Tools for building drag-and-drop, resizing, and sortable functionalities.
3. **Effects:** Predefined animation effects such as fading, bouncing, and sliding.
4. **Theming:** A customizable theming framework, including the ThemeRoller tool, allows developers to style components consistently.
5. **Cross-browser Compatibility:** Ensures a consistent experience across various browsers.
6. **Ease of Use:** Provides a simple API to implement advanced UI features with minimal effort.

jQuery Templates

jQuery Templates were a plugin designed to help developers manage dynamic HTML content more effectively by separating the HTML structure from JavaScript logic. Though it was deprecated, the concept of templates is widely

		<p>used in other modern frameworks like Handlebars and Mustache.</p> <p>Key Features of jQuery Templates:</p> <ol style="list-style-type: none"> 1. HTML Separation: Allows creating reusable HTML templates that can be dynamically populated with data. 2. Data Binding: Injects data into templates using placeholders or tags. 3. Dynamic Content Rendering: Simplifies rendering lists, tables, or any repetitive structures in web applications. 4. Event Handling: Templates can support event binding to generated HTML content. 		
6	a	<p>Discuss the CSS Text Properties</p> <p>To manipulate text using CSS properties. You can set following text properties of an element –</p> <ul style="list-style-type: none"> • The color property is used to set the color of a text. • The direction property is used to set the text direction. • The letter-spacing property is used to add or subtract space between the letters that make up a word. • The word-spacing property is used to add or subtract space between the words of a sentence. • The text-indent property is used to indent the text of a paragraph. • The text-align property is used to align the text of a document. • The text-decoration property is used to underline, overline, and strikethrough text. • The text-transform property is used to capitalize text or convert text to uppercase or lowercase letters. • The white-space property is used to control the flow and formatting of text. • The text-shadow property is used to set the text shadow around a text. 	[L6] [CO1]	[6M]
	b	Distinguish between CSS and CSS	[L4] [CO1]	[6M]

		<table><tr><th>CSS</th><th>CSS3</th></tr><tr><td>Capable of positioning texts and objects.</td><td>Capable of making web pages more attractive and takes less time to create. It is backward compatible with CSS.</td></tr><tr><td>Does not support responsive design.</td><td>Supports responsive design.</td></tr><tr><td>Cannot be split into modules.</td><td>Can be broken down into modules.</td></tr><tr><td>Cannot build 3D animation and transformation.</td><td>Supports animation and 3D transformations.</td></tr><tr><td>Slower compared to CSS3.</td><td>Faster than CSS.</td></tr><tr><td>Uses a set of standard colors and basic color schemes.</td><td>Has a good collection of HSL, RGBA, HSLA, and gradient colors.</td></tr><tr><td>Supports only single text blocks.</td><td>Supports multi-column text blocks.</td></tr><tr><td>Does not support media queries.</td><td>Supports media queries.</td></tr><tr><td></td><td></td></tr></table>	CSS	CSS3	Capable of positioning texts and objects.	Capable of making web pages more attractive and takes less time to create. It is backward compatible with CSS.	Does not support responsive design.	Supports responsive design.	Cannot be split into modules.	Can be broken down into modules.	Cannot build 3D animation and transformation.	Supports animation and 3D transformations.	Slower compared to CSS3.	Faster than CSS.	Uses a set of standard colors and basic color schemes.	Has a good collection of HSL, RGBA, HSLA, and gradient colors.	Supports only single text blocks.	Supports multi-column text blocks.	Does not support media queries.	Supports media queries.				
CSS	CSS3																							
Capable of positioning texts and objects.	Capable of making web pages more attractive and takes less time to create. It is backward compatible with CSS.																							
Does not support responsive design.	Supports responsive design.																							
Cannot be split into modules.	Can be broken down into modules.																							
Cannot build 3D animation and transformation.	Supports animation and 3D transformations.																							
Slower compared to CSS3.	Faster than CSS.																							
Uses a set of standard colors and basic color schemes.	Has a good collection of HSL, RGBA, HSLA, and gradient colors.																							
Supports only single text blocks.	Supports multi-column text blocks.																							
Does not support media queries.	Supports media queries.																							
7	a	<p>List and explain in detail the various selector strings with example.</p> <p>CSS selectors are patterns used to select and style HTML elements. They allow targeting specific elements or groups of elements for styling based on their type, attributes, class, ID, relationships, or state.</p> <ol style="list-style-type: none">1. Universal Selector (*)2. Type Selector (tagname)3. ID Selector (#id)4. Class Selector (.classname)5. Attribute Selectors6. Grouping Selectors (,) <p>1.Universal Selector (*)</p> <p><input type="checkbox"/> Description: Selects all elements in the document.</p>	[L1] [CO1]	[6M]																				

	<p>□ Example:</p> <pre>* { margin: 0; padding: 0; }</pre> <p>Removes margin and padding from all elements.</p> <p>2. Type Selector (tagname)</p> <ul style="list-style-type: none"> • Description: Selects all elements of a specific tag type. • Example <pre>p { color: blue; font-size: 16px; }</pre> <p>Applies blue text color and a font size of 16px to all <p> elements.</p> <p>3. ID Selector (#id)</p> <ul style="list-style-type: none"> • Description: Selects a single element with the specified id. • Example <pre>#main-header { background-color: green; color: white; }</pre> <p>Styles the element with id="main-header" with a green background and white text.</p>		
	<p>b Explain XHTML and Specify some new tags in XHTML?</p> <p>XHTML elements are referred to as "tags", though many prefer the term tag strictly in reference to the semantic structures delimiting the start and end of an element.</p>	[L5] [CO1]	[6M]

		<table><tr><th>Tag</th><th>Description</th></tr><tr><td><!--...--></td><td>Defines a comment</td></tr><tr><td><!DOCTYPE></td><td>Defines the document type</td></tr><tr><td><a></td><td>Defines an anchor</td></tr><tr><td><abbr></td><td>Defines an abbreviation</td></tr><tr><td><acronym></td><td>Defines an acronym</td></tr><tr><td><address></td><td>Defines an address element</td></tr><tr><td><applet></td><td>Deprecated. Defines an applet</td></tr><tr><td><area></td><td>Defines an area inside an image map</td></tr><tr><td></td><td>Defines bold text</td></tr><tr><td><base></td><td>Defines a base URL for all the links in a page</td></tr><tr><td><basefont></td><td>Deprecated. Defines a base font</td></tr><tr><td><bdo></td><td>Defines the direction of text display</td></tr><tr><td><big></td><td>Defines big text</td></tr></table>	Tag	Description	<!--...-->	Defines a comment	<!DOCTYPE>	Defines the document type	<a>	Defines an anchor	<abbr>	Defines an abbreviation	<acronym>	Defines an acronym	<address>	Defines an address element	<applet>	Deprecated. Defines an applet	<area>	Defines an area inside an image map		Defines bold text	<base>	Defines a base URL for all the links in a page	<basefont>	Deprecated. Defines a base font	<bdo>	Defines the direction of text display	<big>	Defines big text		
Tag	Description																															
<!--...-->	Defines a comment																															
<!DOCTYPE>	Defines the document type																															
<a>	Defines an anchor																															
<abbr>	Defines an abbreviation																															
<acronym>	Defines an acronym																															
<address>	Defines an address element																															
<applet>	Deprecated. Defines an applet																															
<area>	Defines an area inside an image map																															
	Defines bold text																															
<base>	Defines a base URL for all the links in a page																															
<basefont>	Deprecated. Defines a base font																															
<bdo>	Defines the direction of text display																															
<big>	Defines big text																															
8	a	<p>What is XML? Briefly explain the namespaces in XML?</p> <p>XML stands for Extensible Markup Language and is a text-based markup language derived from Standard Generalized Markup Language (SGML). XML was designed to store and transport data.</p> <p>XML was designed to be both human- and machine-readable.</p> <p>An XML document is a basic unit of XML information composed of elements and other markup in an orderly package.</p> <p>XML Namespaces provide a method to avoid element name conflicts.</p> <p>1.Name Conflicts</p> <p>In XML, element names are defined by the developer.</p> <p>This often results in a conflict when trying to mix XML documents from different XML applications.</p> <p>This XML carries HTML table information:</p> <pre><table> <tr> <td>Apples</td> <td>Bananas</td> </tr> </table></pre> <p>2.Solving the Name Conflict Using a Prefix</p> <p>Name conflicts in XML can easily be avoided using a name prefix.</p> <p>This XML carries information about an HTML table, and a piece of furniture:</p> <pre><h:table> <h:tr> <h:td>Apples</h:td> <h:td>Bananas</h:td> </h:tr> </h:table></pre>	[L1] [CO1]	[6M]																												

		<pre><f:table> <f:name>African Coffee Table</f:name> <f:width>80</f:width> <f:length>120</f:length> </f:table></pre> <p>3.The xmlns Attribute When using prefixes in XML, a namespace for the prefix must be defined. The namespace can be defined by an xmlns attribute in the start tag of an element. The namespace declaration has the following syntax. xmlns:prefix="URI".</p> <p>4.Uniform Resource Identifier (URI) A Uniform Resource Identifier (URI) is a string of characters which identifies an Internet Resource. The most common URI is the Uniform Resource Locator (URL) which identifies an Internet domain address. Another, not so common type of URI is the Uniform Resource Name (URN).</p> <p>5. Default Namespaces Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax: xmlns="namespaceURI"</p>		
	b	<p>Explain Geolocation with an example?</p> <p>The HTML Geolocation API is used to locate a user's position. The HTML Geolocation API is used to get the geographical position of a user.</p> <pre><script> const x = document.getElementById("demo"); function getLocation() { if (navigator.geolocation) { navigator.geolocation.getCurrentPosition(showPosition); } else { x.innerHTML = "Geolocation is not supported by this browser."; } } function showPosition(position) { x.innerHTML = "Latitude: " + position.coords.latitude + "
Longitude: " + position.coords.longitude; ; } </script></pre>	[L2] [CO1]	[6M]
9	a	<p>What is client side and server-side programming?</p> <p>Client side programming:It is the program that runs on the client machine (browser) and deals with the user interface/display and any other processing that can happen on client machine. 1) Interact with temporary storage 2) Make interactive web pages</p>	[L1] [CO1]	[6M]

	<p>3) Interact with local storage 4) Sending request for data to server 5) Send request to server 6) work as an interface between server and user The Programming languages for client-side programming are :</p> <ol style="list-style-type: none"> 1) Javascript 2) VBScript 3) HTML 4) CSS 5) AJAX <p>Server Side Programming It is the program that runs on server dealing with the generation of content of web page.</p> <ol style="list-style-type: none"> 1) Querying the database 2) Operations over databases 3) Access/Write a file on server. 4) Interact with other servers. 5) Structure web applications. 6) Process user input. For example if user input is a text in search box, run a search algorithm on data stored on server and send the results. <p>Examples :The Programming languages for server-side programming are :</p> <ol style="list-style-type: none"> 1) PHP 2) C++ 3) Java and JSP 4) Python 5) Ruby 		
b	<p>Discuss the Levels of DOM</p> <p>The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects.</p> <p>Levels of DOM:</p> <p>Level 0: Provides low-level set of interfaces.</p> <p>Level 1: DOM level 1 can be described in two parts: CORE and HTML.</p> <p>CORE provides a low-level interfaces that can be used to represent any structured document. HTML provides high-level interfaces that can be used to represent HTML document.</p> <p>Level 2 : consists of six specifications: CORE2, VIEWS, EVENTS, STYLE, TRAVERSAL and RANGE. CORE2: extends functionality of CORE specified by DOM level 1. VIEWS: views allows programs to dynamically access and manipulate content of document. EVENTS: Events are scripts that is either executed by browser when user reacts to web page.</p>	[L6] [CO1]	[6M]

	<p>STYLE: allows programs to dynamically access and manipulate content of style sheets.</p> <p>TRAVERSAL: allows programs to dynamically traverse the document.</p> <p>RANGE: allows programs to dynamically identify a range of content in document.</p> <p>Level 3: consists of five different specifications:</p> <p>CORE3, LOAD and SAVE, VALIDATION, EVENTS, and XPATH.</p> <p>CORE3: extends functionality of CORE specified by DOM level 2.</p> <p>LOAD and SAVE: allows program to dynamically load the content of XML document into DOM document and save the DOM Document into XML document by serialization.</p> <p>VALIDATION: allows program to dynamically update the content and structure of document while ensuring document remains valid.</p> <p>EVENTS: extends functionality of Events specified by DOM Level 2.</p> <p>XPATH: XPATH is a path language that can be used to access DOM tree.</p>		
10	<p>Explain Briefly about JSON and JQuery</p> <p>JSON (JavaScript Object Notation)</p> <p>Definition: JSON is a lightweight data-interchange format that's easy for humans to read and write and easy for machines to parse and generate. It is commonly used to transmit data between a server and a web application.</p> <p>Key Features:</p> <ol style="list-style-type: none"> 1. Lightweight: Compact and efficient for data exchange. 2. Self-describing: Uses key-value pairs to structure data. 3. Language-independent: Although it derives from JavaScript, JSON can be used with most programming languages. 4. Easy to parse: Supported by modern programming languages with built-in parsers. <p>Structure: JSON uses two primary structures:</p> <ul style="list-style-type: none"> • Objects: Key-value pairs enclosed in {}. • Arrays: Ordered lists of values enclosed in []. <p>Example:</p> <pre>{ "name": "John Doe", "age": 30, "skills": ["JavaScript", "HTML", "CSS"], "isEmployed": true }</pre>	[L5] [CO1]	[12M]

```
}
```

Usage:

- Data exchange in APIs.
- Config files (e.g., package.json for Node.js).
- Storing lightweight structured data.

jQuery

Definition: jQuery is a fast, small, and feature-rich JavaScript library designed to simplify tasks like DOM manipulation, event handling, and AJAX operations.

Key Features:

1. **Simplified DOM Manipulation:** Use concise syntax to manipulate HTML and CSS.
2. **Event Handling:** Add event listeners with minimal code.
3. **AJAX Support:** Simplifies asynchronous HTTP requests for dynamic web applications.
4. **Cross-browser Compatibility:** Handles browser differences internally.
5. **Plugins:** Extend functionality with a vast library of plugins.

Basic Syntax:

```
$(selector).action();
```

- `$`: Represents jQuery.
- `selector`: Identifies HTML elements to interact with.
- `action`: Specifies the operation to be performed.

Example:**Select and Style Elements:**

```
$("#p").css("color", "blue");
```

Changes the text color of all `<p>` elements to blue.

Usage:

- Adding interactivity to web pages.
- Making asynchronous API calls.
- Simplifying complex JavaScript operations.

--	--	--	--

UNIT –II

Web Applications and services

1	<p>Discuss Briefly about AngularJS.</p> <p>AngularJS is a very powerful JavaScript Framework. It is used in Single Page Application (SPA) projects. It extends HTML DOM with additional attributes and makes it more responsive to user actions. AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.</p> <p>General Features</p> <p>The general features of AngularJS are as follows</p> <p>AngularJS is a efficient framework that can create Rich Internet Applications (RIA). AngularJS provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way.</p> <p>Applications written in AngularJS are cross-browser compliant.</p> <p>AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.</p> <p>Overall, AngularJS is a framework to build large scale, high-performance, and easy to-maintain web applications.</p> <p>Advantages of AngularJS</p> <p>It provides the capability to create Single Page Application in a very clean and maintainable way.</p> <p>It provides data binding capability to HTML. Thus, it gives user a rich and responsive experience.</p> <p>AngularJS code is unit testable.</p> <p>AngularJS provides reusable components.</p> <p>Disadvantages of AngularJS</p> <p>Though AngularJS comes with a lot of merits, here are some points of concern –</p> <p>Not Secure – Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.</p> <p>Not degradable – If the user of your application disables JavaScript, then nothing would be visible, except the basic page.</p> <p>AngularJS Directives</p> <p>The AngularJS framework can be divided into three major parts –</p> <p>ng-app – This directive defines and links an AngularJS application to HTML.</p> <p>ng-model – This directive binds the values of AngularJS application data to HTML input controls.</p> <p>ng-bind – This directive binds the AngularJS application data to HTML tags.</p> <p>AngularJS - Expressions</p> <p>Expressions are used to bind application data to HTML.</p> <p>Expressions are written inside double curly braces such as in {{ expression}}.</p> <p>Expressions behave similar to ngbind directives.</p> <p>AngularJS expressions are pure JavaScript expressions and output the data where they are used.</p> <p>AngularJS – Filters : Filters are used to modify the data. They can be clubbed in</p>	[L6][CO2]	[12M]
----------	--	------------------	--------------

	<p>expression or directives using pipe () character. The following list shows the commonly used filters.</p> <pre> <html> <head> <title>AngularJS First Application</title> </head> <body> <h1>Sample Application</h1> <div ng-app = ""> <p>Enter your Name: <input type = "text" ng-model = "name"></p> <p>Hello !</p> </div> <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.</script> </body> </html> </pre>		
2	<p>Explain briefly about MVC.</p> <p>The Model-View-Controller (MVC) framework is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs). Nowadays, MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also used for designing mobile apps. MVC was created by Trygve Reenskaug. The main goal of this design pattern was to solve the problem of users controlling a large and complex data set by splitting a large application into specific sections that all have their own purpose.</p> <p>Features of MVC :</p> <ul style="list-style-type: none"> • It provides a clear separation of business logic, UI logic, and input logic. • It offers full control over your HTML and URLs which makes it easy to design web application architecture. • It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs. • It supports Test Driven Development (TDD). <p>Components of MVC :</p> <p>The MVC framework includes the following 3 components:</p> <ul style="list-style-type: none"> • Controller 	[L2][CO2]	[12M]

- Model
- View

Controller:

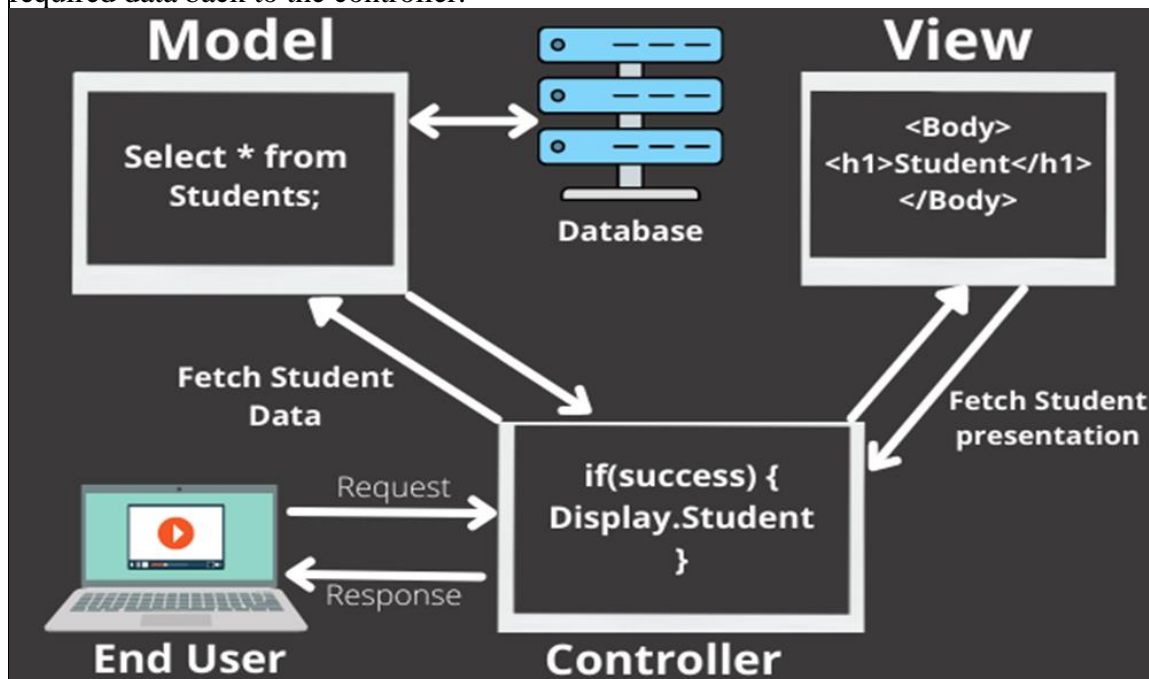
The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn't have to worry about handling data logic, it just tells the model what to do. It process all the business logic and incoming requests, manipulate data using the Model component and interact with the View to render the final output.

View:

The View component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren't taken directly but through the controller. It only interacts with the controller.

Model:

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller's request because the controller can't interact with the database by itself. The model interacts with the database and gives the required data back to the controller.



3 Explain about LARAVEL framework.

[L5][CO2]

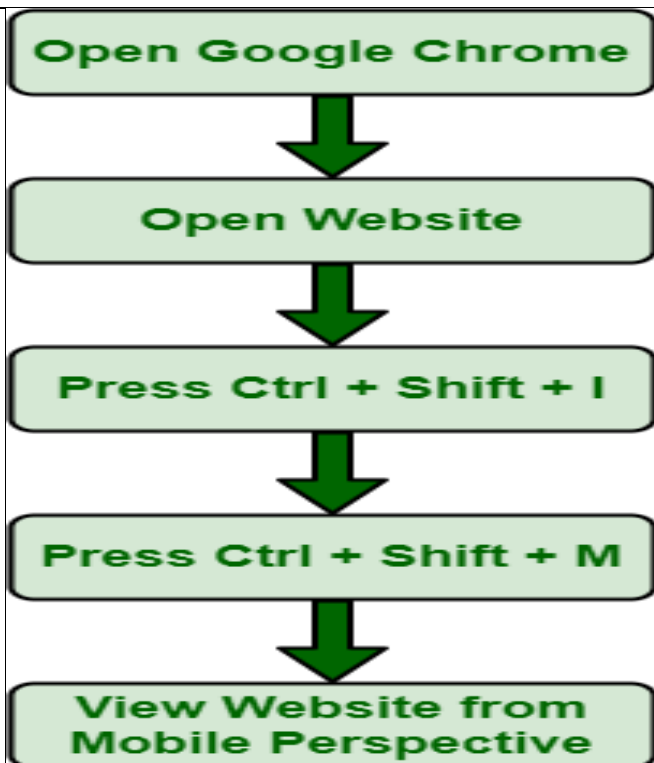
[12M]

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.

Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like CodeIgniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development. If you are familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It saves a lot of time if you are planning to develop a website from scratch. Moreover, a website built in Laravel is secure and prevents several web attacks.

Features of Laravel

	<p>Laravel offers the following key features which makes it an ideal choice for designing web applications –</p> <p>Modularity Laravel provides 20 built in libraries and modules which helps in enhancement of the application. Every module is integrated with Composer dependency manager which eases updates.</p> <p>Testability Laravel includes features and helpers which helps in testing through various test cases. This feature helps in maintaining the code as per the requirements.</p> <p>Routing Laravel provides a flexible approach to the user to define routes in the web application. Routing helps to scale the application in a better way and increases its performance.</p> <p>Configuration Management A web application designed in Laravel will be running on different environments, which means that there will be a constant change in its configuration. Laravel provides a consistent approach to handle the configuration in an efficient way.</p> <p>Query Builder and ORM Laravel incorporates a query builder which helps in querying databases using various simple chain methods. It provides ORM (Object Relational Mapper) and ActiveRecord implementation called Eloquent.</p> <p>Schema Builder Schema Builder maintains the database definitions and schema in PHP code. It also maintains a track of changes with respect to database migrations.</p> <p>Template Engine Laravel uses the Blade Template engine, a lightweight template language used to design hierarchical blocks and layouts with predefined blocks that include dynamic content.</p> <p>E-mail Laravel includes a mail class which helps in sending mail with rich content and attachments from the web application.</p> <p>Authentication User authentication is a common feature in web applications. Laravel eases designing authentication as it includes features such as register, forgot password and send password reminders.</p> <p>Redis Laravel uses Redis to connect to an existing session and general-purpose cache. Redis interacts with session directly.</p> <p>Queues Laravel includes queue services like emailing large number of users or a specified Cron job. These queues help in completing tasks in an easier manner without waiting for the previous task to be completed.</p>		
4	<p>What is RWD and explain it briefly.</p> <p>Responsive web design (RWD) is a web development approach that creates dynamic changes to the appearance of a website, depending on the screen size and orientation of the device being used to view it.</p> <p>RWD can be obtained by using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones).</p> <p>How to check whether the Website is Responsive?</p> <p>One should know that not all Websites are responsive. Non-responsive websites display perfectly on desktop, laptop but not on mobile or tablets. Below are some steps given that one can follow to determine whether website is responsive or not.</p>	[L1][CO2]	[12M]



Step 1: Go to Google Chrome and Open it.

Step 2: Go to website that you want to check for responsive design.

Step 3: Press Ctrl + Shift + I. This will open Chrome DevTools. These are web developer and debugging tools built into browser and help one to edit page and identify problem easily.

Step 4: Press Ctrl + Shift + M. This will toggle device toolbar i.e.; toolbar will appear at top of page and enable one to see appearance of website on mobile device.

Step 5: View website from mobile, tablet perspective.

Features of Responsive Design

There are basically 4 aspects of responsive design as given

Features of Responsive Web Design

Reflowing Content

Relative Sizing

Breakpoints

Adaptation to all Devices

below:

Reflowing Content

Reflowing content simply refers to content that adjusts its width to fit width of window. Every day, new devices are being developed with new screen sizes.

Relative Sizing

Relative sizing simply means adjusting size of element according to width of browser or screen.

Breakpoints

Breakpoints, also known as media queries, are the simplest filters applied to CSS styles that can be used for different style rules for various devices such as laptops, mobiles, tablets, etc.

Adaptation to all Devices

Flexibility is main component of responsive website. Their flexibility to adapt to different size screen provide better user experience and makes it more appealing.

5	a	<p>List out some popular MVC frameworks</p> <p>Some of the most popular and extensively used MVC frameworks are listed below.</p> <ul style="list-style-type: none"> • Ruby on Rails • Django • CherryPy • Spring MVC • Catalyst • Rails 	[L1][CO2]	[6M]
---	---	---	-----------	------

	<ul style="list-style-type: none"> • Zend Framework • Fuel PHP • Laravel <p>Symphony</p>		
b	<p>What are the core Features of AngularJS</p> <p>Core Features The core features of AngularJS are as follows – Data-binding – It is the automatic synchronization of data between model and view components. Scope – These are objects that refer to the model. They act as a glue between controller and view. Controller – These are JavaScript functions bound to a particular scope. Services – AngularJS comes with several built-in services such as \$http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app. Filters – These select a subset of items from an array and returns a new array. Directives – Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc. Templates – These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using partials. Routing – It is concept of switching views. Model View Whatever – MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.</p>	[L1][CO2]	[6M]
6	<p>What is Single Page web application with example?</p> <p>Single page application (SPA) is a web application that fits on a single page. All your code (JS, HTML, CSS) is retrieved with a single page load. And navigation between pages performed without refreshing the whole page. Step 1: Create a Module. ... Step 2: Define Controller. ... Step 3: Deploy the AngularJS script in HTML Code. ... Step 5: Configure Routes. ... Step 6: Time to Build Controllers. ... Step 7: Configuring the HTML Pages. ... Step 8: Add Links to Those HTML Pages.</p> <p>ADVANTAGES No page refresh When you are using SPA, you don't need to refresh the whole page, just load the part of the page which needs to be changed. Angular allows you to pre-load and cache all your pages, so you don't need extra requests to download them. Better user experience</p>	[L1][CO2]	[12M]

		<p>SPA feels like a native application: fast and responsive.</p> <p>Ability to work offline</p> <p>Even if user loses internet connection, SPA can still work because all the pages are already loaded.</p> <p>DISADVANTAGES</p> <p>More complex to build</p> <p>You need to write pretty much javascript, handle shared state between pages, manage permissions, etc.</p> <p>Initial load is slow</p> <p>SPA needs to download more resources when you open it.</p> <p>Client should have javascript enabled</p> <p>Of course, SPA requires javascript. But fortunately, almost everyone has javascript enabled.</p> <pre> <!DOCTYPE html> <!--ng-app directive tells AngularJS that myApp is the root element of the application --> <html ng-app="myApp"> <head> <!--import the angularjs libraries--> <script src= "https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.4.7/angular.min.js"> </script> <script src= "https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.4.7/angular-route.min.js"> </script> <style> body { text-align: center; font-family: Arial, Helvetica, sans-serif; } h1 { color: green; } </style> </head> </pre>		
7	a	<p>What are the MVC architectural patterns?</p> <p>The MVC architectural pattern allows us to adhere to the following design principles:</p> <ol style="list-style-type: none"> 1. Divide and conquer. The three components can be somewhat independently designed. 2. Increase cohesion. The components have stronger layer cohesion than if the view and controller were together in a single UI layer. 3. Reduce coupling. The communication channels between the three components are minimal and easy to find. 4. Increase reuse. The view and controller normally make extensive use of reusable components for various kinds of UI controls. The UI, however will become application specific, therefore it will not be easily reusable. 5. Design for flexibility. It is usually quite easy to change the UI by changing the view, the controller, or both. 	[L1][CO2]	[6M]
	b	<p>Discuss about Angular JS Objects?</p> <p>AngularJS Directives</p>	[L2][CO2]	[6M]

		<p>The AngularJS framework can be divided into three major parts –</p> <p>ng-app – This directive defines and links an AngularJS application to HTML.</p> <p>ng-model – This directive binds the values of AngularJS application data to HTML input controls.</p> <p>ng-bind – This directive binds the AngularJS application data to HTML tags.</p> <p>AngularJS - Expressions</p> <p>Expressions are used to bind application data to HTML.</p> <p>Expressions are written inside double curly braces such as in {{ expression }}.</p> <p>Expressions behave similar to ngbind directives.</p> <p>AngularJS expressions are pure JavaScript expressions and output the data where they are used.</p> <p>AngularJS – Filters : Filters are used to modify the data. They can be clubbed in expression or directives using pipe () character. The following list shows the commonly used filters.</p> <p>AngularJS Module</p> <p>AngularJS supports modular approach. Modules are used to separate logic such as services, controllers, application etc. from the code and maintain the code clean. We define modules in separate js files and name them as per the module.js file.</p> <p>In the following example, we are going to create two modules –</p> <p>Application Module – used to initialize an application with controller(s).</p> <p>Controller Module – used to define the controller.</p>		
8	a	<p>Describe the features of Laravel framework?</p> <p>Features of Laravel</p> <p>Laravel offers the following key features which makes it an ideal choice for designing web applications –</p> <p>Modularity</p> <p>Laravel provides 20 built in libraries and modules which helps in enhancement of the application. Every module is integrated with Composer dependency manager which eases updates.</p> <p>Testability</p> <p>Laravel includes features and helpers which helps in testing through various test cases. This feature helps in maintaining the code as per the requirements.</p> <p>Routing</p> <p>Laravel provides a flexible approach to the user to define routes in the web application. Routing helps to scale the application in a better way and increases its performance.</p> <p>Configuration Management</p> <p>A web application designed in Laravel will be running on different environments, which means that there will be a constant change in its configuration. Laravel provides a consistent approach to handle the configuration in an efficient way.</p> <p>Query Builder and ORM</p> <p>Laravel incorporates a query builder which helps in querying databases using various simple chain methods. It provides ORM (Object Relational Mapper) and ActiveRecord implementation called Eloquent.</p> <p>Schema Builder</p> <p>Schema Builder maintains the database definitions and schema in PHP code. It also maintains a track of changes with respect to database migrations.</p>	[L2][CO2]	[6M]

		<p>Template Engine Laravel uses the Blade Template engine, a lightweight template language used to design hierarchical blocks and layouts with predefined blocks that include dynamic content.</p> <p>E-mail Laravel includes a mail class which helps in sending mail with rich content and attachments from the web application.</p>		
	b	<p>Discuss about Angular JS Arrays?</p> <p>AngularJS Arrays In AngularJS, arrays play a crucial role in managing collections of data. Arrays are used within AngularJS applications to store and manipulate lists of items, such as data for iteration in views, maintaining dynamic data structures, or passing multiple parameters. They are typically combined with directives like ng-repeat for efficient rendering and interaction.</p>	[L2][CO2]	[6M]
		<p>Key Features of AngularJS Arrays:</p> <ol style="list-style-type: none"> 1. Dynamic Nature: AngularJS arrays can store elements of various types (numbers, strings, objects, etc.) and adjust their size dynamically. 2. Two-way Binding: When used with the \$scope object, arrays are bound to the view, ensuring that any updates to the array automatically reflect in the UI. 3. Integration with Directives: Arrays work seamlessly with AngularJS directives like ng-repeat, ng-options, and filters. 4. Native JavaScript Array Methods: AngularJS arrays support native JavaScript methods like push(), splice(), filter(), and more, enabling powerful data manipulation. <p>Examples of Using Arrays in AngularJS:</p> <p>1. Storing Simple Data:</p> <pre>app.controller("arrayController", function(\$scope) { \$scope.numbers = [1, 2, 3, 4, 5]; });</pre> <p>2. Using ng-repeat to Display Arrays:</p> <pre><div ng-app="myApp" ng-controller="arrayController"> <li ng-repeat="number in numbers">{{ number }} </div></pre>		
9	a	Describe web applications with an example?	[L2][CO2]	[6M]

	<p>Web Applications</p> <p>A web application is a software program that runs on a web server and is accessed through a web browser over the internet or an intranet. Unlike desktop applications, web applications do not require installation on a user's device. They interact with users via web pages, using technologies like HTML, CSS, JavaScript, and server-side scripting languages.</p> <p>Characteristics of Web Applications</p> <ol style="list-style-type: none"> 1. Platform Independence: Accessible from any device with a web browser. 2. No Installation Needed: Updates and functionality are managed on the server side. 3. Dynamic Interaction: Web applications can provide real-time responses and interactivity using technologies like AJAX and WebSockets. 4. Centralized Data Storage: Data is stored on servers, ensuring availability and security. <p>Components of a Web Application</p> <ol style="list-style-type: none"> 1. Front-End (Client-Side): <ul style="list-style-type: none"> o Built with HTML, CSS, and JavaScript. o Focuses on the user interface (UI) and interaction. 2. Back-End (Server-Side): <ul style="list-style-type: none"> o Handles business logic, database interaction, and server-side processing. o Built using server-side languages like PHP, Python, Ruby, or Node.js. 3. Database: <ul style="list-style-type: none"> o Stores application data and user information. o Examples: MySQL, MongoDB, PostgreSQL. <p>Benefits of Web Applications</p> <ol style="list-style-type: none"> 1. Cross-Platform Compatibility: Work on any device with a browser. 2. Scalability: Easily updated and maintained on the server. 3. Cost-Effective: Reduces distribution and installation overhead. 		
b	<p>What is responsive image and explain with example?</p> <p>Responsive Image</p> <p>A responsive image is an image that adapts to different screen sizes, resolutions, and devices. It ensures that the image looks good on all devices, such as desktops, tablets, and smartphones, without compromising performance (e.g., loading large images on small screens). The goal is to deliver the best image quality while optimizing page load times.</p> <p>Why Use Responsive Images?</p> <ol style="list-style-type: none"> 1. Better User Experience: Images scale properly on different screen sizes. 2. Improved Performance: Load smaller images for mobile devices, saving bandwidth and improving loading times. 3. SEO Benefits: Google rewards fast-loading pages, which can be achieved 	[L1][CO2]	[6M]

with responsive images.

How to Make an Image Responsive?

1. **Using CSS (with max-width and height):** You can make an image responsive by setting its width to 100% and height to auto, allowing it to scale relative to its container.

Example:

```

```

2. **Using the <picture> Element:** The <picture> element allows you to define different sources for images based on various conditions like screen size or resolution.

Example:

```
<picture>
  <source srcset="small.jpg" media="(max-width: 600px)">
  <source srcset="medium.jpg" media="(max-width: 1200px)">
  
</picture>
```

3. **Using srcset and sizes Attributes (for the tag):** You can specify different image sizes for different screen resolutions and sizes using the srcset and sizes attributes of the tag.

Example:

```

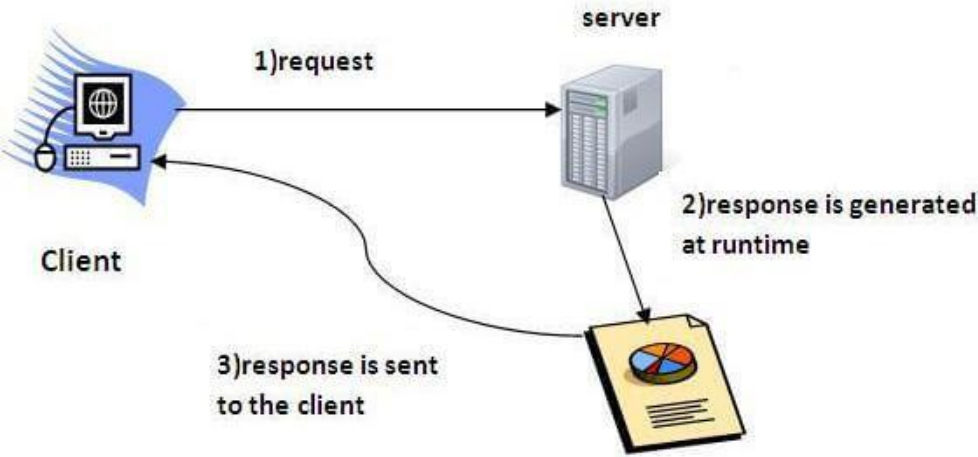
```

10	a	<p>Explain the features of Responsive web Design</p> <p>Features of Responsive Design</p> <p>There are basically 4 aspects of responsive design as given below:</p> <div data-bbox="188 1288 1214 2078"> <pre> graph TD A[Features of Responsive Web Design] --> B[Reflowing Content] A --> C[Relative Sizing] A --> D[Breakpoints] A --> E[Adaptation to all Devices] </pre> </div> <p>Reflowing Content</p> <p>Reflowing content simply refers to content that adjusts its width to fit width of</p>	[L5][CO2]	[6M]
----	---	---	-----------	------

	<p>window. Every day, new devices are being developed with new screen sizes.</p> <p>Relative Sizing</p> <p>Relative sizing simply means adjusting size of element according to width of browser or screen.</p> <p>Breakpoints</p> <p>Breakpoints, also known as media queries, are the simplest filters applied to CSS styles that can be used for different style rules for various devices such as laptops, mobiles, tablets, etc.</p> <p>Adaptation to all Devices</p> <p>Flexibility is main component of responsive website. Their flexibility to adapt to different size screen provide better user experience and makes it more appealing.</p>		
b	<p>What are the Advantages and Disadvantages of Single Page web application</p> <p>ADVANTAGES</p> <p>No page refresh</p> <p>When you are using SPA, you don't need to refresh the whole page, just load the part of the page which needs to be changed. Angular allows you to pre-load and cache all your pages, so you don't need extra requests to download them.</p> <p>Better user experience</p> <p>SPA feels like a native application: fast and responsive.</p> <p>Ability to work offline</p> <p>Even if user loses internet connection, SPA can still work because all the pages are already loaded.</p> <p>DISADVANTAGES</p> <p>More complex to build</p> <p>You need to write pretty much javascript, handle shared state between pages, manage permissions, etc.</p> <p>Initial load is slow</p> <p>SPA needs to download more resources when you open it.</p> <p>Client should have javascript enabled</p> <p>Of course, SPA requires javascript. But fortunately, almost everyone has javascript enabled.</p>	[L1][CO2]	[6M]

UNIT –III

Web Communication Processes and Technologies

1	<p>a What is servlet? What are the advantages of servlet?</p> <p>Servlet is a technology i.e. used to create web application. Servlet is an API that provides many interfaces and classes including documentations. Servlet is an interface that must be implemented for creating any servlet. Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any type of requests. Servlet is a web component that is deployed on the server to create dynamic web page.</p>  <p>Advantages of servlet</p> <ul style="list-style-type: none"> • Efficient and scalable • Reduces the execution time • Avoids loading program for every request. • Portability • Server and platform independent • Able to deploy in any server and worked on various operating system • Robustness • Managed by JVM 	[L1][CO3]	[6M]
	<p>b Illustrate the Life cycle of servlets.</p> <p>Init():</p> <ul style="list-style-type: none"> • This method define the initialization information • The init() method is called only once throughout the life cycle of the servlet • It must be called by the servlet container before the servlet can service any request. 	[L3][CO3]	[6M]

Service()

- The servlet container calls the service() method for servicing any request.
- The service() method determines the kind of request and calls the appropriate method (doGet() or doPost()) for handling the request and sends response to the client using the methods of the response object.
-

Destroy():

- This method removes the servlet from the server
- Define any clean up codes that releases resources, such as closing a database connection etc before removing the servlet from memory
- It is called only once throughout the life cycle of the servlet

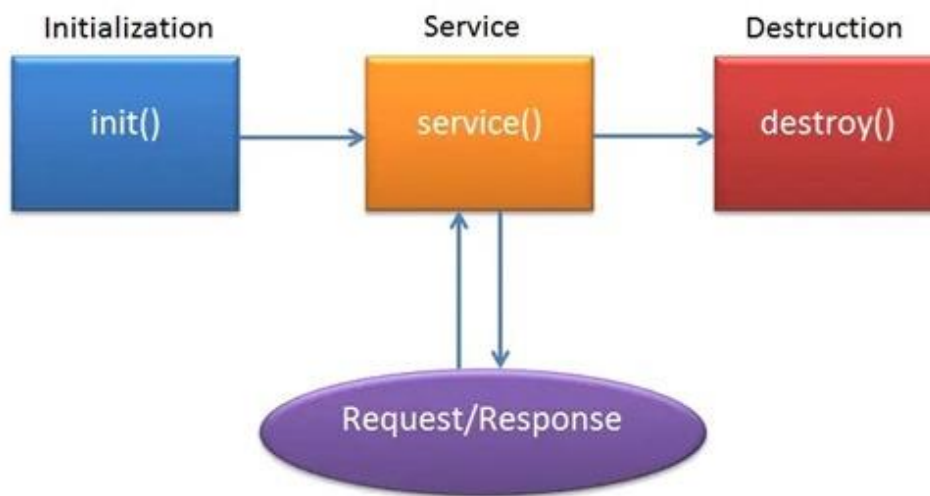


Fig: Life Cycle of a Servlet

2 Create a servlet code to get parameters from HTML document.

[L6][CO3]

[12M]

NewServlet.java

```

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class NewServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest rq,
  
```

```

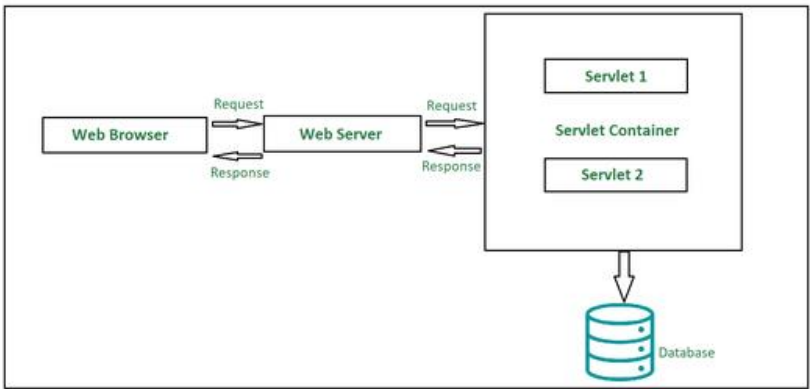
HttpServletResponse rs)
    throws ServletException, IOException {
    rs.setContentType("text/html");
    PrintWriter out=rs.getWriter();
    HttpSession session=rq.getSession(true);
    Integer
count=(Integer)session.getAttribute("NewServlet.HitCount");
    if(count==null)
    {
        count=new Integer(1);
    }
    else
    {

count=new Integer(count.intValue()+1);
    }

    session.setAttribute("NewServlet.HitCount",count);
    out.println("<html><html><title>Session example
</title></head>");
    out.println("<body><h4>Session server to
"+"demonstrate session Tracking and its life cycle </h4>");
    out.println("<br> session status : ");
    if(session.isNew())
    {
        out.println("New session <br>");}
    out.println("HitCount for your current session is :
"+"count);
    out.println("<br> <h2>some basic session
information</h2>");
    out.println("session ID : "+session.getId()+"<br>");
    out.println("it is a new session: "+session.isNew());
    out.println("<br>session creation time :
"+"session.getCreationTime());
    out.println("("+new
Date(session.getCreationTime())+"<br>");
    out.println("Last accessed Time :
"+"session.getLastAccessedTime());
    out.println("("+ new
Date(session.getLastAccessedTime())+"<br>");
    out.println("Max Inactive Time :

```

		<pre>" +session.getMaxInactiveInterval()+"</br>"); out.println("session info as in cookie : "+rq.isRequestedSessionIdFromCookie()+"</br>"); out.println("</body></html>"); } } }</pre>																														
3	a	<p>Distinguish between Generic Servlet and HttpServlet.</p> <table><tr><th></th><th>Servlet</th><th>GenericServlet</th><th>HttpServlet</th></tr><tr><td>What it is?</td><td>Interface</td><td>Abstract Class</td><td>Abstract Class</td></tr><tr><td>Package</td><td>javax.servlet</td><td>javax.servlet</td><td>javax.servlet.http</td></tr><tr><td>Hierarchy</td><td>Top level interface</td><td>Implements Servlet interface</td><td>Extends GenericServlet</td></tr><tr><td>Methods</td><td>init(), service(), destroy(), getServletConfig(), getServletInfo()</td><td>init(), service(), destroy(), getServletConfig(), getServletInfo(), log(), getInitParameter(), getInitParameterNames(), getServletContext(), getServletName()</td><td>doGet(), doPost(), doPut(), doDelete(), doHead(), doOptions(), doTrace(), getLastModified(), service()</td></tr><tr><td>Abstract Methods</td><td>All methods are abstract.</td><td>Only service() method is abstract.</td><td>No abstract methods.</td></tr><tr><td>When to use?</td><td>Use it when you want to develop your own Servlet container.</td><td>Use to write protocol independent servlets.</td><td>Use to write HTTP-specific servlets.</td></tr></table>		Servlet	GenericServlet	HttpServlet	What it is?	Interface	Abstract Class	Abstract Class	Package	javax.servlet	javax.servlet	javax.servlet.http	Hierarchy	Top level interface	Implements Servlet interface	Extends GenericServlet	Methods	init(), service(), destroy(), getServletConfig(), getServletInfo()	init(), service(), destroy(), getServletConfig(), getServletInfo(), log(), getInitParameter(), getInitParameterNames(), getServletContext(), getServletName()	doGet(), doPost(), doPut(), doDelete(), doHead(), doOptions(), doTrace(), getLastModified(), service()	Abstract Methods	All methods are abstract.	Only service() method is abstract.	No abstract methods.	When to use?	Use it when you want to develop your own Servlet container.	Use to write protocol independent servlets.	Use to write HTTP-specific servlets.	[L5][CO3]	[6M]
	Servlet	GenericServlet	HttpServlet																													
What it is?	Interface	Abstract Class	Abstract Class																													
Package	javax.servlet	javax.servlet	javax.servlet.http																													
Hierarchy	Top level interface	Implements Servlet interface	Extends GenericServlet																													
Methods	init(), service(), destroy(), getServletConfig(), getServletInfo()	init(), service(), destroy(), getServletConfig(), getServletInfo(), log(), getInitParameter(), getInitParameterNames(), getServletContext(), getServletName()	doGet(), doPost(), doPut(), doDelete(), doHead(), doOptions(), doTrace(), getLastModified(), service()																													
Abstract Methods	All methods are abstract.	Only service() method is abstract.	No abstract methods.																													
When to use?	Use it when you want to develop your own Servlet container.	Use to write protocol independent servlets.	Use to write HTTP-specific servlets.																													
	b	<p>What is Servlet Context Interface</p> <p>An object of ServletContext is created by the web container at time of deploying the project.</p> <ul style="list-style-type: none">➤ This object can be used to get configuration information from web.xml file.➤ There is only one ServletContext object per web application.➤ If any information is shared to many servlet, it is better to provide it from the web.xml file using the <context-param> element. <p>There is given some commonly used methods of ServletContext interface.</p> <p>public String getInitParameter(String name):Returns the parameter value for the specified parameter name.</p> <p>public Enumeration getInitParameterNames():Returns the names of the</p>	[L1][CO3]	[6M]																												

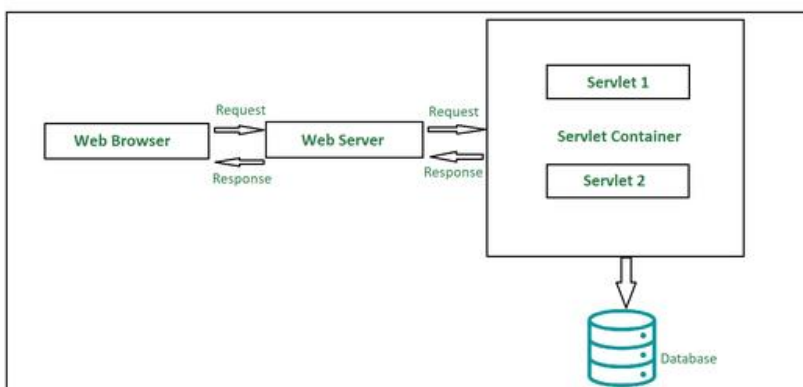
		<p>context's initialization parameters.</p> <p>public void setAttribute(String name, Object object): sets the given object in the application scope.</p> <p>public Object getAttribute(String name): Returns the attribute for the specified name.</p> <p>public Enumeration getInitParameterNames(): Returns the names of the context's initialization parameters as an Enumeration of String objects.</p> <p>public void removeAttribute(String name): Removes the attribute with the given name from the servlet context.</p> <p>getServletContext() method of ServletConfig interface returns the object of ServletContext.</p> <p>getServletContext() method of GenericServlet class returns the object of ServletContext.</p> <p>Syntax of getServletContext() method</p> <pre>public ServletContext getServletContext()</pre>		
4	a	<p>Discuss about the architecture of servlet</p> <p>Servlet is a technology i.e. used to create web application. Servlet is an API that provides many interfaces and classes including documentations. Servlet is an interface that must be implemented for creating any servlet. Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any type of requests. Servlet is a web component that is deployed on the server to create dynamic web page.</p>  <pre> graph LR WB[Web Browser] -- Request --> WS[Web Server] WS -- Response --> WB WS -- Request --> SC[Servlet Container] SC -- Response --> WS subgraph SC [Servlet Container] S1[Servlet 1] S2[Servlet 2] end SC --> DB[(Database)] </pre> <p>1. Client</p> <p>The client shown in the architecture above is primarily working as a medium who is sending out HTTP requests over to the web server and again processing the response it gets back from the server. As we can see in the diagram, our client here is the web browser.</p> <p>2. Web Server</p>	[L2][CO3]	[6M]

		<p>Primary job of a web server is to process the requests and responses that a user sends over time and maintain how a web user.</p> <p>3. Web Container</p> <p>Web container is another typical component in servlet architecture which is responsible for communicating with the servlets.</p>		
	b	<p>Explain about HTTP Request Response model.</p> <p>HTTP Request/Response Model: HTTP Request/Response is a client-server protocols where the client sends a request to the server, and the server responds with the requested information.</p> <p>A client (a browser) sends an HTTP request to the web.</p> <p>A web server receives the request.</p> <p>The server runs an application to process the request.</p> <p>The server returns an HTTP response (output) to the browser.</p> <p>The client (the browser) receives the response.</p> <p>HTTP stands for Hyper Text Transfer Protocol</p> <p>WWW is about communication between web clients and servers</p> <p>Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses</p> <p>Block diagram of HTTP Request/Response:</p>	[L2][CO3]	[6M]

	<pre> graph LR subgraph Browser1 [Browser] B1["An event occurs... • Create an XMLHttpRequest object • Send HttpRequest"] end subgraph Internet1 [Internet] I1((Internet)) end subgraph Server [Server] S["• Process HTTPRequest • Create a response and send data back to the browser"] end subgraph Browser2 [Browser] B2["• Process the returned data using JavaScript • Update page content"] end subgraph Internet2 [Internet] I2((Internet)) end B1 --> I1 I1 --> S S --> I2 I2 --> B2 </pre> <p>The diagram illustrates the flow of an HTTP request and response. It starts with a Browser on the left where an event occurs, leading to the creation of an XMLHttpRequest object and the sending of an HttpRequest to the Internet. The Internet then routes the request to a Server on the right, which processes the request and creates a response to send back to the browser. The response travels back through the Internet to the Browser, where the data is processed using JavaScript and the page content is updated.</p>		
5	<p>Create a Java servlet program to display current Date and Time.</p> <p><u>NewServlet.java</u></p> <pre> import java.io.IOException; import java.io.PrintWriter; import java.util.Date; import javax.servlet.ServletException; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import javax.servlet.http.HttpSession; public class NewServlet extends HttpServlet { @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { response.setContentType("text/html"); PrintWriter out=response.getWriter(); HttpSession session=request.getSession(true); Integer count=(Integer)session.getAttribute("NewServlet.HitCount"); if(count==null) { count=new Integer(1); } else { count=new Integer(count.intValue()+1); } session.setAttribute("NewServlet.HitCount",count); out.println("<html><html><title>Session example </title></head>"); out.println("<body><h4>Session server to "+"demonstrate session Tracking and its life cycle </h4>"); } } </pre>	[L6][CO3]	[12M]

		<pre> out.println("</br> session status : "); if(session.isNew()) { out.println("New session </br>");} out.println("HitCount for your current session is : "+count); out.println("</br> <h2>some basic session information</h2>"); out.println("session ID : "+session.getId()+"</br>"); out.println("it is a new session: "+session.isNew()); out.println("</br>session creation time : "+session.getCreationTime()); out.println("(" + new Date(session.getCreationTime())+"</br>"); out.println("Last accessed Time : "+session.getLastAccessedTime()); out.println("(" + new Date(session.getLastAccessedTime())+"</br>"); out.println("Max Inactive Time : "+session.getMaxInactiveInterval()+"</br>"); out.println("session info as in cookie : "+rq.isRequestedSessionIdFromCookie()+"</br>"); out.println("</body></html>"); } } } </pre>		
6	a	<p>Explain about HTTP servlet Request.</p> <p>HTTP Servlet Request HTTP Servlet Request is an interface in the Java Servlet API that allows a servlet to receive HTTP requests sent by a client (usually a web browser). Key Features of HttpServletRequest:</p> <ol style="list-style-type: none"> 1. Retrieve Request Parameters: You can retrieve parameters sent with the HTTP request, such as form data or URL query parameters. 2. Access Request Headers: It allows access to the headers sent by the client in the request. 3. Handle Request Attributes: You can store and retrieve attributes that are specific to the current request. 4. Session Management: It allows retrieval of session data related to the user's session (e.g., HttpSession). 5. Request Methods: It helps to determine the method of the HTTP request (GET, POST, etc.). <p>Common Methods of HttpServletRequest:</p> <ul style="list-style-type: none"> getParameter(String name) <p>Syntax: String value = request.getParameter("username"); getHeader(String name) Syntax: String userAgent = request.getHeader("User-Agent"); getSession() Syntax: HttpSession session = request.getSession(); getMethod()</p>	[L2][CO3]	[6M]

		<p>Syntax: <code>String method = request.getMethod();</code> <code>getRequestURI()</code> Syntax: <code>String uri = request.getRequestURI();</code></p>		
	b	<p>Discuss about HTTP servlet Response with syntax</p> <p>HTTP Servlet Response is an interface in the Java Servlet API that allows a servlet to send an HTTP response back to the client (e.g., a browser).</p> <p>Key Features of HttpServletResponse:</p> <ol style="list-style-type: none"> 1. Set Response Status: You can set the status code for the response (e.g., 200 OK, 404 Not Found). 2. Set Response Headers: You can modify or add headers to the response (e.g., content type, caching information). 3. Send Output Data: The servlet can send output data such as HTML, JSON, or plain text to the client using the response output stream. 4. Handle Redirects: You can redirect the client to another URL using <code>sendRedirect()</code>. 5. Manage Cookies: You can send cookies to the client using <code>addCookie()</code>. <p>Common Methods of HttpServletResponse:</p> <ul style="list-style-type: none"> • <code>setContentType(String type)</code> <p>Syntax: <code>response.setContentType("text/html");</code> <code>getWriter()</code> Syntax: <code>PrintWriter out = response.getWriter();</code> <code>setStatus(int statusCode)</code> Syntax: <code>response.setStatus(HttpServletResponse.SC_OK); // 200 OK</code> <code>sendRedirect(String location)</code> Syntax: <code>response.sendRedirect("http://www.example.com");</code></p>	[L2][CO3]	[6M]
7	a	<p>Describe about overview of servlet.</p> <p>Servlet is a technology i.e. used to create web application. Servlet is an API that provides many interfaces and classes including documentations. Servlet is an interface that must be implemented for creating any servlet. Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any type of requests. Servlet is a web component that is deployed on the server to create dynamic web page.</p> <p>Architecture</p>	[L2][CO3]	[6M]



1. Client

The client shown in the architecture above is primarily working as a medium who is sending out HTTP requests over to the web server and again processing the response it gets back from the server. As we can see in the diagram, our client here is the web browser.

2. Web Server

Primary job of a web server is to process the requests and responses that a user sends over time and maintain how a web user.

3. Web Container

Web container is another typical component in servlet architecture which is responsible for communicating with the servlets.

b Discuss about POST method with an example.

[L2][CO3]

[6M]

POST Method

The POST method is a commonly used HTTP method that sends data to the Web server in the request body of HTTP. The various characteristics of the POST method are:

The POST requests cannot be bookmarked as they do not appear in the URL.

The POST requests do not get cached.

The POST requests are not saved as history by the web browsers.

There is no restriction on the amount of data to be sent in a POST request

The POST method can be used to send ASCII as well as binary data.

When communicating sensitive data, such as when submitting an HTML form, a POST request must be used.

The data sent by the POST method goes through the HTTP header, so security depends on the HTTP protocol. By using secure HTTP (HTTPS), you can ensure that your information is protected.

		<p>Here is an example code snippet that submits an HTML form using the POST method:</p> <p>Example:</p> <p>Copy Code<html></p> <pre> <body> <form action="registration.php" method="post"> Name: <input type="text" name="name"> Email: <input type="text" name="email"> <input type="submit"> </form> </body> </html> </pre>		
8	a	<p>Explain in detail about GET method.</p> <p>GET Method</p> <p>The GET method is one of the most commonly used methods of HTTP. It is usually implemented to request a particular resource data from the Web server by specifying the parameters as a query string (name and value pairs) in the URL part of the request.</p> <p>Example:</p> <p>http://example.com/category/index.php?category_name=HTML&lesson=introduction</p> <p>The various characteristics of the GET method are:</p> <p>The GET requests can be bookmarked as they appear in the URL.</p> <p>The GET request can be cached.</p> <p>The GET request is saved in the browser history if it is executed using a web browser.</p> <p>There are character length restrictions (2048 characters maximum) for this method as they appear in the URL.</p> <p>The GET method cannot be used to send binary data such as images and Word</p>	[L2][CO3]	[6M]

	<p>documents.</p> <p>The data can only be retrieved from requests that use the GET method and have no other effect.</p> <p>When communicating sensitive data such as login credentials, a GET request should not be used as it appears in the URL, making it less secure.</p> <p>Since the GET request only requests data and does not modify any resources, it is considered a safe and ideal method to request data only.</p>		
b	<p>Explain the block diagram of HTTP Request Response model?</p> <p>HTTP Request/Response Model: HTTP Request/Response is a client-server protocols where the client sends a request to the server, and the server responds with the requested information. A client (a browser) sends an HTTP request to the web. A web server receives the request. The server runs an application to process the request. The server returns an HTTP response (output) to the browser. The client (the browser) receives the response. HTTP stands for Hyper Text Transfer Protocol WWW is about communication between web clients and servers Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses</p> <p>Block diagram of HTTP Request/Response:</p> <pre> graph LR subgraph Browser1 [Browser] B1[An event occurs... • Create an XMLHttpRequest object • Send HttpRequest] end subgraph Internet1 [Internet] end subgraph Server [Server] S[• Process HTTPRequest • Create a response and send data back to the browser] end subgraph Browser2 [Browser] B2[• Process the returned data using JavaScript • Update page content] end subgraph Internet2 [Internet] end Browser1 --> Internet1 Internet1 --> Server Server --> Internet2 Internet2 --> Browser2 </pre>	[L2][CO3]	[6M]
9	<p>Discuss in detail about AJAX architecture</p> <p>AJAX is an abbreviation for Asynchronous JavaScript and XML. AJAX is a technique not a programming language which is used by the developers to make</p>	[L2][CO3]	[12M]

the websites behaving like desktop applications.

It operates on the client-side for creating asynchronous web applications. AJAX is a group of technologies that uses number of web technologies for creating a set of web development techniques.

Technologies involved in AJAX are:

HTML – It is used at the client side.

JavaScript – It is used to make the request.

CSS – It is also used at the client side.

XML – It is just a request formats.

JSON – It is also a request formats.(java script object notation)

php – It is used at the server side.

There are too many web applications running on the web that are using AJAX Technology.

Some are : 1. Gmail

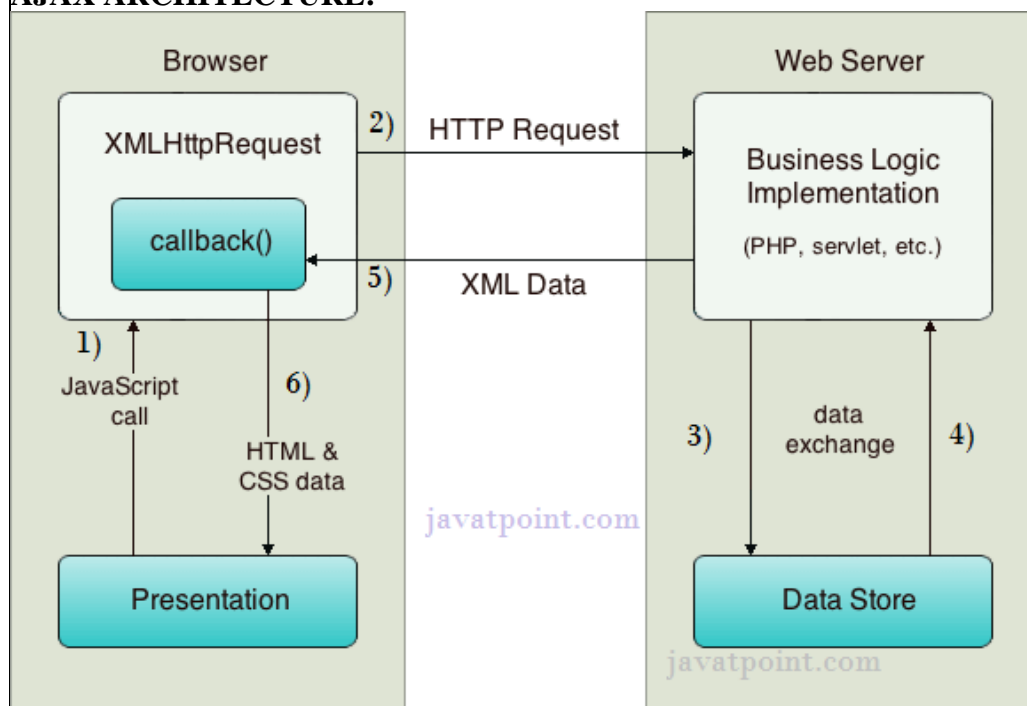
2. Face book

3. Twitter

4. Google maps

5. YouTube etc

AJAX ARCHITECTURE:



1. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.

2. HTTP Request is sent to the server by XMLHttpRequest object.

3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.

4. Data is retrieved.

5. Server sends XML data or JSON data to the XMLHttpRequest callback function.

		<p>6. HTML and CSS data is displayed on the browser.</p> <p>Advantages of Ajax :</p> <p>Response time of AJAX is high henceforth it increases speed and performance.</p> <p>AJAX supports a lot of browsers.</p> <p>Some of the complex AJAX applications gives feeling that we are using it on Desktop.</p> <p>There is lesser requirement of time between server and client.</p> <p>AJAX allows multiple tasks to be performed at same time.</p> <p>Disadvantages of Ajax :</p> <p>We face browser compatibility issues in AJAX.</p> <p>If JavaScript is disabled by user then those users cannot use AJAX as AJAX needs JavaScript to be enabled.</p> <p>Various search engine like Google can't index AJAX pages.</p> <p>We cannot bookmark AJAX updated page content.</p> <p>Since data is downloadable from client side henceforth AJAX is less secure.</p>		
10	a	<p>Illustrate AJAX framework and its security</p> <p>Security issues of AJAX AJAX Security: Server Side</p> <p>AJAX-based Web applications use the same server-side security schemes of regular Web applications.</p> <p>You specify authentication, authorization, and data protection requirements in your web.xml file (declarative) or in your program (programmatic).</p> <p>AJAX-based Web applications are subject to the same security threats as regular Web applications.</p> <p>AJAX Security: Client Side</p> <p>JavaScript code is visible to a user/hacker. Hacker can use JavaScript code for inferring server-side weaknesses.</p> <p>JavaScript code is downloaded from the server and executed ("eval") at the client and can compromise the client by mal-intended code.</p>	[L2][CO3]	[6M]

	b	Describe about generating dynamic content Generating Dynamic Content: Generating dynamic content typically involves generating or modifying content on the fly based on various factors, such as user input, database queries, or real-time events. To generate dynamic content need to follow these steps: Define your content generation goals: Determine the purpose and context of the dynamic content. Are you generating personalized recommendations, dynamic product listings, or real-time updates? Identify the data sources: Determine the data sources you'll be working with, such as databases, APIs, or user input. These sources will provide the information required to generate the dynamic content. Design the content generation logic: Create algorithms, rules, or logic that will process the data and generate the dynamic content. Consider factors such as user preferences, relevance, or real-time events that should influence the content generation process. Implement the content generation logic: Write code or use tools that allow you to implement the content generation logic. Depending on your programming language or framework, you may have specific libraries or APIs that can assist with dynamic content generation. Test and iterate: Test your dynamic content generation system thoroughly. Validate that it produces the desired results and meets your goals. If necessary, iterate on the logic, algorithms, or data sources to improve the quality or relevance of the generated content. Monitor and update: Continuously monitor the performance of your dynamic content generation system. Collect feedback from users and make updates as needed. Content trends and user preferences may change over time, so it's important to adapt and optimize your system accordingly.	[L2][CO3]	[6M]

UNIT –IV

Web Servers

1	Differentiate between Session and Cookie	[L4][CO4]	[12M]
---	--	-----------	-------

	<table><tr><th><u>Cookies</u></th><th><u>Sessions</u></th></tr><tr><td>Cookies are stored on the user's computer.</td><td>Sessions are stored on the server.</td></tr><tr><td>A cookie can store a limited amount of data, a maximum of 4KB.</td><td>The session can store an unlimited amount of data.</td></tr><tr><td>The cookie does not depend on the Session.</td><td>The session depends on the cookie.</td></tr><tr><td>The cookie expires according to the time of expiry set for it</td><td>The session expires after the user closes the web browser.</td></tr><tr><td>Cookie has many security issues as it can be accessed by anyone easily.</td><td>The session is secure as it cannot be accessed by anyone easily.</td></tr><tr><td>There is no function available to disable a Cookie.</td><td>A Session can be disabled by using the <u>session destroy()</u> function.</td></tr><tr><td>The <u>setcookie()</u> function should always be used prior to tag.</td><td>The <u>session start()</u> function should always appear prior to tag.</td></tr></table>	<u>Cookies</u>	<u>Sessions</u>	Cookies are stored on the user's computer.	Sessions are stored on the server.	A cookie can store a limited amount of data, a maximum of 4KB.	The session can store an unlimited amount of data.	The cookie does not depend on the Session.	The session depends on the cookie.	The cookie expires according to the time of expiry set for it	The session expires after the user closes the web browser.	Cookie has many security issues as it can be accessed by anyone easily.	The session is secure as it cannot be accessed by anyone easily.	There is no function available to disable a Cookie.	A Session can be disabled by using the <u>session destroy()</u> function.	The <u>setcookie()</u> function should always be used prior to tag.	The <u>session start()</u> function should always appear prior to tag.		
<u>Cookies</u>	<u>Sessions</u>																		
Cookies are stored on the user's computer.	Sessions are stored on the server.																		
A cookie can store a limited amount of data, a maximum of 4KB.	The session can store an unlimited amount of data.																		
The cookie does not depend on the Session.	The session depends on the cookie.																		
The cookie expires according to the time of expiry set for it	The session expires after the user closes the web browser.																		
Cookie has many security issues as it can be accessed by anyone easily.	The session is secure as it cannot be accessed by anyone easily.																		
There is no function available to disable a Cookie.	A Session can be disabled by using the <u>session destroy()</u> function.																		
The <u>setcookie()</u> function should always be used prior to tag.	The <u>session start()</u> function should always appear prior to tag.																		
2	<p>Explain node.js with example</p> <p>Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more.</p> <p>Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser.</p> <p>Features of Node.js</p> <p>Following are some of the important features that make Node.js the first choice of software architects.</p> <ul style="list-style-type: none">• Asynchronous and Event Driven – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.• Very Fast – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.• Single Threaded but Highly Scalable – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.• No Buffering – Node.js applications never buffer any data. These applications simply output the data in chunks.• License – Node.js is released under the MIT license <p>Following are the areas where Node.js is proving itself as a perfect technology partner.</p> <ul style="list-style-type: none">• I/O bound Applications	[L2][CO4]	[12M]																

- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

```
/* Hello, World! program in node.js */
console.log("Hello, World!")
```

Now execute main.js file using Node.js interpreter to see the result –

```
$ node main.js
```

If everything is fine with your installation, this should produce the following result –

```
Hello, World!
```

Example:

```
// Node.js program to display some
// text on console screen

// Accessing console module
const console = require('console');

// Calling console.info() method
console.info("Welcome to GeeksforGeeks");
```

Steps to Run: First, Download and Install in your system and then use the following command to run your code.

```
node filename.js
```

Output: The output will display on console screen.

```
Welcome to GeeksforGeeks
```

3 Design an application in node.js for student management

[L6][CO4]

[12M]

```
import javax.swing.*;
import java.awt.*;
import java.awt.Image;
import java.awt.event.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.print.*;
import javax.swing.print.Printer;
import java.io.*;
import java.io.IOException;
```

```
// Creating the fee class
```

```

public class fee extends Frame {

    JLabel l1, l2, l3, l4,
        l5, l6, l7, l8,
        l9, l10, l12, l13,
        l14, l11, l15;

    JTextField tf1, tf2, tf3,
        tf4, tf5, tf6,
        tf7, tf8, tf9,
        tf10;

    JTextArea area2, area1;

    JRadioButton rb1, rb2, rb3,
        rb4, rb5, rb6,
        rb7;

    JFileChooser f1;

    // Default constructor to
    // initialize the parameters
    fee()
    {

        l1 = new JLabel("Fee Report");
        l1.setBounds(550, 100, 250, 20);

        l2 = new JLabel(
            "Name of the Student:");
        l2.setBounds(50, 150, 250, 20);

        tf1 = new JTextField();
        tf1.setBounds(250, 150, 250, 20);

        l3 = new JLabel(
            "Name of the Father:");
        l3.setBounds(50, 200, 250, 20);

        tf2 = new JTextField();
        tf2.setBounds(250, 200, 250, 20);

        l4 = new JLabel("Roll Number:");
        l4.setBounds(50, 250, 250, 20);

        tf3 = new JTextField();
        tf3.setBounds(250, 250, 250, 20);

        l5 = new JLabel("Email ID:");
        l5.setBounds(50, 300, 250, 20);

        tf4 = new JTextField();
        tf4.setBounds(250, 300, 250, 20);

        l6 = new JLabel("Contact Number:");

```

```

16.setBounds(50, 350, 250, 20);

tf5 = new JTextField();
tf5.setBounds(250, 350, 250, 20);

17 = new JLabel("Address:");
17.setBounds(50, 400, 250, 20);

area1 = new JTextArea();
area1.setBounds(250, 400, 250, 90);

19 = new JLabel("Gender:");
19.setBounds(50, 500, 250, 20);

JRadioButton r5
    = new JRadioButton(" Male");
JRadioButton r6
    = new JRadioButton(" Female");

r5.setBounds(250, 500, 100, 30);
r6.setBounds(350, 500, 100, 30);

ButtonGroup bg = new ButtonGroup();
bg.add(r5);
bg.add(r6);

110 = new JLabel("Nationality:");
110.setBounds(50, 550, 250, 20);

tf6 = new JTextField();
tf6.setBounds(250, 550, 250, 20);

111 = new JLabel(
    "Year of passing 10th");
111.setBounds(50, 600, 250, 20);

String language[]
    = { "2016", "2015", "2014" };

final JComboBox cb1
    = new JComboBox(language);

cb1.setBounds(250, 600, 90, 20);

112 = new JLabel(
    "Year of passing 12th");
112.setBounds(50, 650, 250, 20);

String languagess[]
    = { "2019", "2018", "2017" };

113 = new JLabel(
    "Points Secured in 10th:");
113.setBounds(50, 700, 250, 20);

```

```

tf7 = new JTextField();
tf7.setBounds(250, 700, 250, 20);

l14 = new JLabel("Percentage in 12th:");
l14.setBounds(50, 750, 250, 20);

tf8 = new JTextField();
tf8.setBounds(250, 750, 250, 20);

ImageIcon i2 = new ImageIcon("2.png");
JLabel l15
    = new JLabel("", i2, JLabel.CENTER);

l15.setBounds(900, 50, 600, 200);

final JComboBox cb2
    = new JComboBox(languagess);

cb2.setBounds(250, 650, 90, 20);
l8 = new JLabel(
    "Groups Offered here are:");
l8.setBounds(800, 150, 250, 20);

rb1 = new JRadioButton("SEAS");
rb1.setBounds(550, 150, 100, 30);

rb2 = new JRadioButton("SLABS");
rb2.setBounds(650, 150, 100, 30);

ButtonGroup bg1 = new ButtonGroup();

bg1.add(rb1);
bg1.add(rb2);

rb3 = new JRadioButton("HOSTELLER");
rb3.setBounds(550, 200, 100, 30);

rb4 = new JRadioButton("DAY SCHOLAR");
rb4.setBounds(650, 200, 120, 30);

ButtonGroup bg2 = new ButtonGroup();
bg2.add(rb3);
bg2.add(rb4);

String languages[]
    = { "CSE", "ECE", "EEE",
        "CIVIL", "MECH" };
final JComboBox cb
    = new JComboBox(languages);
cb.setBounds(800, 200, 90, 20);

final JLabel label
    = new JLabel();
label.setBounds(600, 430, 500, 30);
JButton b = new JButton("Show");

```



```

b.setBounds(1000, 300, 80, 30);

final DefaultListModel<String> li1
    = new DefaultListModel<>();

li1.addElement("CSE(2, 50, 000)");
li1.addElement("ECE(2, 50, 000)");
li1.addElement("EEE(2, 50, 000)");
li1.addElement("MECH(2, 50, 000)");
li1.addElement("CIVIL(2, 50, 000)");

final JList<String> list1
    = new JList<>(li1);

list1.setBounds(600, 300, 125, 125);

DefaultListModel<String> li2
    = new DefaultListModel<>();

li2.addElement(
    "2 SHARE(1, 50, 000)");
li2.addElement(
    "3 SHARE(1, 40, 000)");
li2.addElement(
    "5 SHARE(1, 20, 000)");
li2.addElement(
    "8 SHARE(1, 10, 000)");
li2.addElement(
    "bus(40, 000)");

final JList<String> list2
    = new JList<>(li2);
list2.setBounds(
    800, 300, 125, 125);

JButton Receipt
    = new JButton("Generate Receipt");
Receipt.setBounds(600, 490, 150, 30);
JButton b2 = new JButton("Reset");
b2.setBounds(750, 490, 150, 30);
JButton Print = new JButton("Print");
Print.setBounds(900, 490, 150, 30);

area2 = new JTextArea();
area2.setBounds(600, 540, 450, 240);

add(l1);
add(l2);
add(l3);
add(l4);
add(l5);
add(l6);
add(l7);
add(l8);
add(l9);

```

```

add(l10);
add(l11);
add(l12);
add(l13);
add(l14);
add(tf1);
add(tf2);
add(tf3);
add(tf4);
add(tf5);
add(tf6);
add(tf7);
add(tf8);
add(area1);
add(area2);
add(l15);
add(rb1);
add(rb2);
add(rb3);
add(rb4);
add(r5);
add(r6);
add(cb);
add(cb1);
add(cb2);
add(list1);
add(list2);
add(b);
add(label);
add(Receipt);
add(b2);
add(Print);

```

```

b.addActionListener(new ActionListener() {

    // Method to display the data
    // entered in the text fields
    public void actionPerformed(ActionEvent e)
    {
        String data = "";
        if (list1.getSelectedIndex() != -1) {
            data = "You had selected the Group:"
                + list1.getSelectedValue();
            label.setText(data);
        }
        if (list2.getSelectedIndex() != -1) {
            data += " and Hostel with the "
                + "facility of: ";

            for (Object frame :
                list2.getSelectedValues()) {
                data += frame + " ";
            }
        }
        label.setText(data);
    }
});

```

```

    }
});

// Reset the text fields
b2.addActionListener(
    new ActionListener() {
        public void actionPerformed(
            ActionEvent e)
        {
            area2.setText("");
            area1.setText(" ");
            tf1.setText("");
            tf2.setText("");
            tf3.setText("");
            tf4.setText("");
            tf5.setText("");
            tf6.setText(" ");
        }
    });

// Implementing the Print action
Print.addActionListener(
    new ActionListener() {
        public void actionPerformed(
            ActionEvent e)
        {
            try {
                area2.print();
            }
            catch (java.awt.print
                .PrinterException a) {
                System.err.format(
                    "NoPrinter Found",
                    a.getMessage());
            }
        }
    });

// Generating the receipt
Receipt.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        area2.setText(
            "-----"
            + "-----FEE RECEIPT----"
            + "-----"
            + "-----"
            + "-----\n");

        area2.setText(area2.getText()
            + "Student Name: "
            + tf1.getText()
            + "\n");
        area2.setText(area2.getText()

```

```

        + "Father's Name: "
        + tf2.getText()
        + "\n");
area2.setText(area2.getText()
        + "RollNumber: "
        + tf3.getText()
        + "\n");
area2.setText(area2.getText()
        + "Email ID: "
        + tf4.getText()
        + "\n");
area2.setText(area2.getText()
        + "Contact Number: "
        + tf5.getText()
        + "\n");
area2.setText(area2.getText()
        + "Wants to take: "
        + cb.getSelectedItem()
        .toString()
        + "\n");

if (rb1.isSelected()) {
    area2.setText(area2.getText()
        + "Wants to Join in "
        + "School of Engineering "
        + "and Applied Sciences\n");
}
if (rb2.isSelected()) {
    area2.setText(area2.getText()
        + "Wants to Join in "
        + "School of Liberal "
        + "Arts and Sciences\n");
}
if (rb3.isSelected()) {
    area2.setText(area2.getText()
        + "Wants to be a "
        + "Hosteller \n");
}
if (rb4.isSelected()) {
    area2.setText(area2.getText()
        + "Wants to be a "
        + "Day Scholar \n");
}
area2.setText(area2.getText()
        + "Had chosen: "
        + list1.getSelectedValue()
        .toString()
        + "\n");
area2.setText(area2.getText()
        + "Had chosen: "
        + list2.getSelectedValue()
        .toString()
        + "\n");

int index2 = list2.getSelectedIndex();

```

```

        if (index2 == 0) {
            area2.setText(area2.getText()
                + "
                + "Total amount to be "
                + "paid is 4 Lakhs  \n");
        }

        if (index2 == 1) {
            area2.setText(area2.getText()
                + "
                + "Total amount to be paid "
                + "is 3.9 Lakhs  \n");
        }

        if (index2 == 2) {
            area2.setText(area2.getText()
                + "
                + "Total amount to be paid "
                + "is 3.8 Lakhs  \n");
        }

        if (index2 == 3) {
            area2.setText(area2.getText()
                + "
                + "Total amount to be paid "
                + "is 3.7 Lakhs  \n");
        }

        if (index2 == 4) {
            area2.setText(area2.getText()
                + "
                + "Total amount to be paid "
                + "is 2.9 Lakhs  \n");
        }

        if (e.getSource() == Receipt) {
            try {
                FileWriter fw
                    = new FileWriter(
                        "java.txt", true);
                fw.write(area2.getText());
                fw.close();
            }
            catch (Exception ae) {
                System.out.println(ae);
            }
        }


        JOptionPane.showMessageDialog(
            area2, "DATA SAVED SUCCESSFULLY");
    };
});
addWindowListener(
    new WindowAdapter() {
        public void windowClosing(

```

		<pre> WindowEvent we) { System.exit(0); } }); setSize(800, 800); setLayout(null); setVisible(true); setBackground(Color.cyan); } public static void main(String[] args) { new fee(); } } </pre>		
4	a	<p>Illustrate Express Framework in detail?</p> <p>Express framework Express is a powerful tool for building robust and scalable web applications. Express is a fast, flexible, and minimalist web framework designed for Node.js. It simplifies the process of building web applications and APIs . Developed and maintained by the Node.js foundation, Express.js offers a robust set of features that enhance productivity.</p> <p>Here are some key features of Express:</p> <p>Middleware and Routing: Middleware functions allow you to handle tasks like authentication, logging, and error handling. Routing ensures that incoming requests are directed to the appropriate handlers.</p> <p>Minimalistic Design: Express follows a simple and minimalistic design philosophy. This simplicity allows you to quickly set up a server, define routes, and handle HTTP requests efficiently.</p> <p>Flexibility and Customization: Express doesn't impose a strict application architecture. You can structure your code according to your preferences, whether you're building a RESTful API or a full-fledged web app.</p> <p>Scalability: Designed to be lightweight and scalable, Express handles a large number of requests asynchronously. Its event-driven architecture ensures responsiveness even under heavy loads.</p> <p>Active Community Support: With a community, Express.js receives regular updates and improvements. You'll find documentation, tutorials, and plugins to enhance your development experience.</p> <p>Basic Example of an Express App:</p> <pre> var express = require('express'); var app = express(); app.get('/', function (req, res) { res.send('Welcome to JavaTpoint!'); }); var server = app.listen(8000, function () { var host = server.address().address; var port = server.address().port; </pre>	[L2][CO4]	[6M]

		console.log('Example app listening at http://%s:%s', host, port); });		
	b	<p>Discuss about different type of data and sessions.</p> <p>DATA The information exchanged between a user's web browser and the web server during their browsing session is called data. This data can include various types of information: Session Data: This includes information about the user's current browsing session, such as the pages they have visited, actions they have taken (like submitting forms or clicking links), and any temporary state information needed to maintain the session. Cookie Data: Cookies are small pieces of data sent by the web server to the user's browser and stored on their device. This data can include user-specific information, such as login credentials, preferences, shopping cart contents, or tracking identifiers. Server-side Data: In addition to session data and cookies, the web server may also store information on its side related to user interactions. Different Types of Sessions in Web Servers:</p> <ol style="list-style-type: none"> 1. Session Management: A session represents a user's interaction with a web server over a specific time. Sessions allow tracking and maintaining user-specific data across multiple HTTP requests. 	[L2][CO4]	[6M]
		<p>Types of Sessions:</p> <ol style="list-style-type: none"> 1. Client-Side Sessions: <ul style="list-style-type: none"> ○ Data is stored on the user's browser using cookies, localStorage, or sessionStorage. ○ Advantages: Reduces server load. ○ Disadvantages: Limited storage capacity, security concerns. 2. Example: // Storing data in sessionStorage sessionStorage.setItem('username', 'JohnDoe'); <p>Server-Side Sessions:</p> <ul style="list-style-type: none"> • Session data is stored on the server and is associated with a unique session ID sent to the client as a cookie. • Advantages: More secure, supports complex data structures. • Disadvantages: Requires server resources. • Example: Using HttpSession in Java <pre>HttpSession session = request.getSession(); session.setAttribute("username", "JohnDoe"); ○ .</pre>		
5	a	<p>What is an Event and explain different types of events?</p> <p>What is an Event? An event in programming refers to any action or occurrence recognized by a program, typically originating from the user, the system, or other sources. Events trigger a response or a series of responses, which are handled using event listeners or handlers. Events are widely used in web development, GUI applications, and systems programming to make applications interactive and dynamic.</p>	[L1][CO4]	[8M]

	<p>Different Types of Events:</p> <p>1. User Interface Events</p> <p>These events are triggered by user interactions with the user interface (UI).</p> <ul style="list-style-type: none"> • Click Events: Triggered when the user clicks on an element. <ul style="list-style-type: none"> ◦ Example: Clicking a button. ◦ Code Example: <pre>document.getElementById("btn").addEventListener("click", () => { alert("Button clicked!"); });</pre> <p>Mouse Events: Triggered by mouse actions like moving, clicking, or hovering.</p> <ul style="list-style-type: none"> • Examples: mouseover, mouseout, mousedown, mouseup. • <p>Code Example</p> <pre>document.getElementById("div").addEventListener("mouseover", () => { console.log("Mouse over the div!"); });</pre> <p>Keyboard Events: Triggered when the user interacts with the keyboard.</p> <ul style="list-style-type: none"> • Examples: keydown, keypress, keyup. • Code Example: <pre>document.addEventListener("keydown", (event) => { console.log(`Key pressed: \${event.key}`); });</pre> <p>Focus Events: Triggered when an element gains or loses focus.</p> <ul style="list-style-type: none"> • Examples: focus, blur. • Code Example <p>2. Document and Window Events</p> <p>These events relate to the document or browser window.</p> <ul style="list-style-type: none"> • Load Events: Triggered when the document or specific resources are fully loaded. <ul style="list-style-type: none"> ◦ Example: load. ◦ Code Example: <pre>window.addEventListener("load", () => { console.log("Page fully loaded!"); });</pre> <p>Scroll Events: Triggered when the user scrolls the document or an element.</p> <ul style="list-style-type: none"> • Example: scroll. • Code Example: <pre>javascript Copy code window.addEventListener("scroll", () => { console.log("Page is being scrolled!"); });</pre>		
b	<p>Explain the Callback Event.</p> <p>What is a Callback Event?</p> <p>A callback event refers to the use of a callback function to handle an event when it</p>	[L2][CO4]	[4M]

		<p>occurs. In programming, a callback function is a function that is passed as an argument to another function and is executed after the completion of an event or operation. This is commonly used in asynchronous programming to perform tasks after an event, such as a user action, network request, or timer completion.</p> <p>Key Characteristics of Callback Events:</p> <ol style="list-style-type: none"> 1. Asynchronous Nature: Callback events often work in an asynchronous way, where the code execution continues without waiting for the event to complete. 2. Event Binding: Callback functions are bound to specific events using methods like <code>addEventListener</code> (in JavaScript). 3. Flexibility: You can define custom behavior for each event by passing different callback functions. 		
6	a	<p>Illustrate the Terminal REPL</p> <p>Read-Eval-Print Loop(REPL)</p> <p>A Read-Eval-Print Loop, or REPL, is a computer environment where user inputs are read and evaluated, and then the results are returned to the user.</p> <p>It performs the following tasks –</p> <p>Read – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.</p> <p>Eval – Takes and evaluates the data structure.</p> <p>Print – Prints the result.</p> <p>Loop – Loops the above command until the user presses ctrl-c twice.</p> <p>Online REPL Terminal</p> <p>To simplify your learning, we have set up an easy to use Node.js REPL environment online, where you can practice Node.js syntax – Launch Node.js REPL Terminal </p> <p>Starting REPL</p> <p>REPL can be started by simply running node on shell/console without any arguments as follows.</p> <pre>\$ node</pre> <p>You will see the REPL Command prompt > where you can type any Node.js command –</p> <pre>\$ node ></pre>	[L3][CO4]	[8M]

Simple Expression

Let's try a simple mathematics at the Node.js REPL command prompt –

```
$ node
> 1 + 3
4
> 1 + ( 2 * 3 ) - 4
3
>
```

Use Variables

You can make use variables to store values and print later like any conventional script. If **var** keyword is not used, then the value is stored in the variable and printed. Whereas if **var** keyword is used, then the value is stored but not printed. You can print variables using **console.log()**.

```
$ node
> x = 10
10
> var y = 10
undefined
> x + y
20
> console.log("Hello World")
Hello World
undefined
```

Underscore Variable

You can use underscore (**_**) to get the last result –


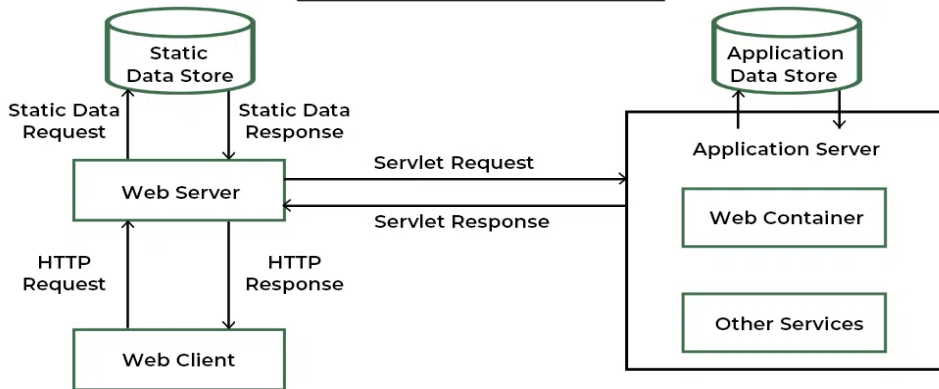
```
$ node
> var x = 10
undefined
> var y = 20
undefined
> x + y
30
> var sum = _
undefined
> console.log(sum)
30
undefined
>
```

Stopping REPL

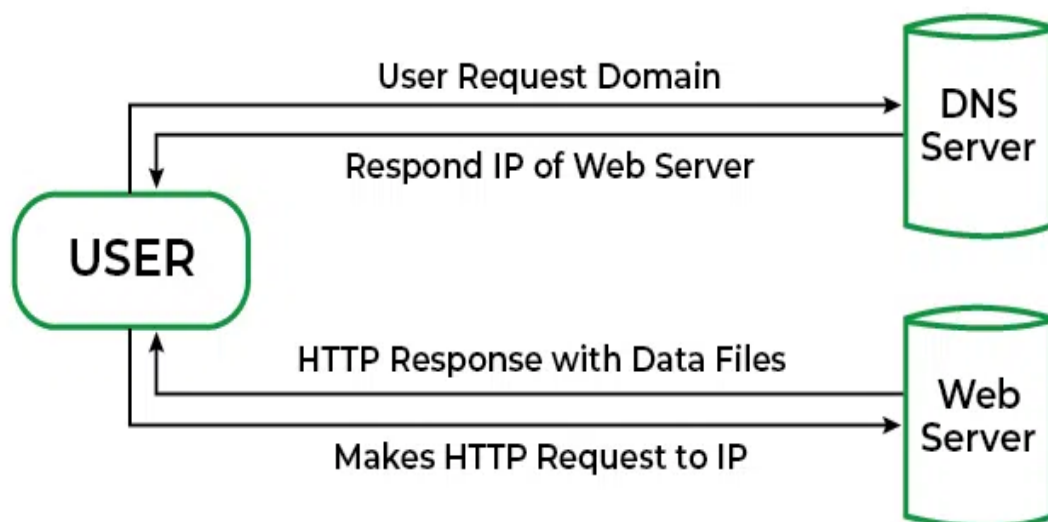
As mentioned above, you will need to use **ctrl-c twice** to come out of Node.js REPL.

```
$ node
>
(^C again to quit)
>
```

	<p>b What is URL Rewriting explain with an example?</p> <p>URL rewriting in Java Servlets is a technique used to modify the URL of a web page before it is sent to the client's browser.</p> <p>Here's how you can perform URL rewriting in Java Servlets:</p> <p>Obtaining the session ID:</p> <p>Use the `HttpServletRequest` object's `getSession()` method to retrieve the current session.</p> <p>Appending the session ID to URLs:</p> <p>Modify the URLs that need session tracking by appending the session ID as a query parameter.</p> <p>Handling the rewritten URLs on subsequent requests:</p> <p>Parse the query parameters in subsequent requests to retrieve the session ID.</p>	[L1][CO4]	[4M]
7	<p>a Explain Briefly Cookies and its types?</p> <p>Cookies are text files with small pieces of data — like a username and password — that are used to identify your computer as you use a computer network.</p> <p>Specific cookies known as HTTP cookies are used to identify specific users and improve your web browsing experience.</p> <p>Data stored in a cookie is created by the server upon your connection. This data is labeled with an ID unique to you and your computer.</p> <p>When the cookie is exchanged between your computer and the network server, the server reads the ID and knows what information to specifically serve to you.</p> <p>Session cookies. Session cookies, also known as 'temporary cookies', help websites recognise users and the information provided when they navigate through a website.</p> <p>Persistent cookies or permanent cookies are stored on users' hard drive until it expires or until the user deletes the cookie. These cookies remain on a user's device even after they close a web browser.</p> <p>First-Party Cookies</p> <p>These cookies allow website owners to collect analytics data, remember language settings, and perform other useful functions that provide a good user experience.</p> <p>An example of a first-party cookie is when a user signs into an ecommerce website,</p>	[L2][CO4]	[6M]

		<p>like Amazon.</p> <p>A third-party cookie is a cookie that's placed on a user's device -- computer, cellphone or tablet -- by a website from a domain other than the one the user is visiting.</p> <p>A Flash cookie, also known as a local shared object, is a text file that is sent by a Web server to a client when the browser requests content supported by Adobe Flash, a popular browser plug-in.</p> <p>A zombie cookie is a piece of data that could be stored in multiple locations -- since failure of removing all copies of the zombie cookie will make the removal reversible, zombie cookies can be difficult to remove.</p>		
	b	<p>Describe about web server with an example.</p> <p>A web server is a software application or hardware device that stores, processes, and serves web content to users over the internet. It plays a critical role in the client-server model of the World Wide Web, where clients (typically web browsers) request web pages and resources, and servers respond to these requests by delivering the requested content.</p> <p>Web servers operate on the Hypertext Transfer Protocol (HTTP), which is the foundation of data communication on the World Wide Web.</p> <p>When you enter a website's URL into your browser, it sends an HTTP request to the web server hosting that website, which then sends back the web page you requested, allowing you to view it in your browser.</p> <p> Working of Web Server</p> 	[L2][CO4]	[6M]
8	a	<p>Distinguish between different types of webserver architectures.</p> <p>Web Server Architecture Web server architecture refers to the structure and design of web servers, outlining how they handle incoming requests and deliver web content. There are two main approaches to web server architecture:</p> <p>Single-Tier (Single Server) Architecture: In a single-tier architecture, a single server is responsible for both processing requests and serving web content. This is suitable for small websites or applications with low traffic. However, it has limitations in terms of scalability and fault tolerance. If the server goes down, the entire service becomes unavailable.</p>	[L4][CO4]	[6M]

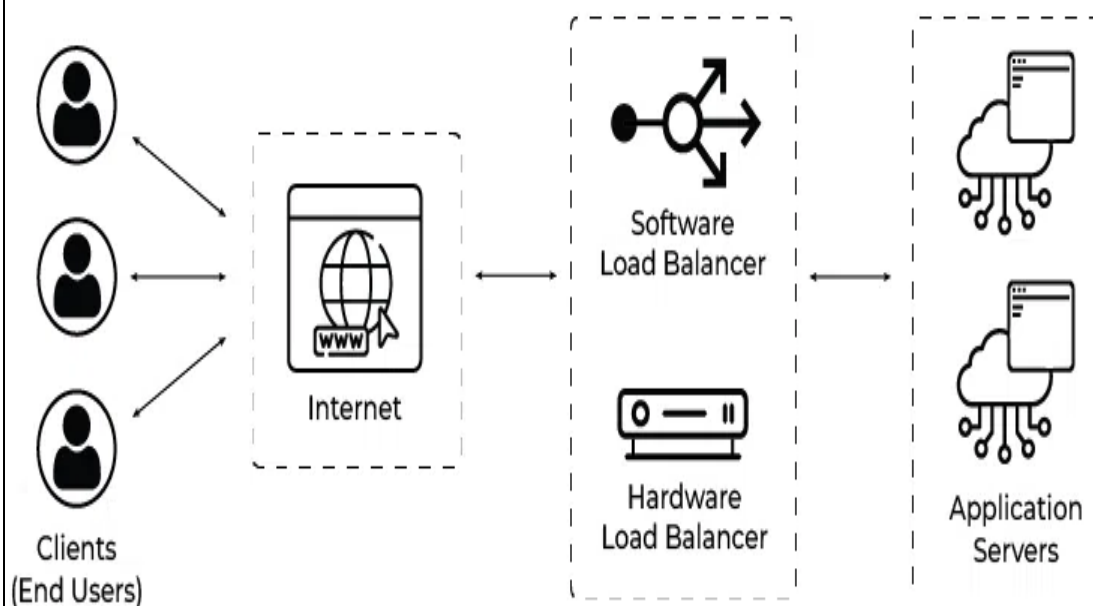
Single Server Architecture of Web Server



Multi-Tier (Load-Balanced) Architecture:

In a multi-tier architecture, multiple servers are used to distribute the workload and ensure high availability. This approach often involves load balancers that evenly distribute incoming requests across a cluster of web servers. Each server can serve web content independently, and if one server fails, the load balancer redirects traffic to healthy servers, ensuring uninterrupted service.

Load Balance web server architecture



b What are the benefits and uses of web servers?

Benefits of Web Servers:

Using web servers offers several advantages, including:

[L1][CO4]

[6M]

		<p>Scalability : Web servers can handle a large number of simultaneous connections, making them suitable for high-traffic websites.</p> <p>Reliability : They are designed for continuous operation and can recover from failures gracefully.</p> <p>Security : Web servers include security features to protect against common web threats like DDoS attacks and SQL injection.</p> <p>Customization : Web server configurations can be tailored to specific application requirements.</p> <p>Uses of Web Servers</p> <ul style="list-style-type: none"> Hosting Websites Running Web Applications Serving APIs (Application Programming Interfaces) File Hosting and Distribution Managing Emails 		
9	a	<p>Discuss Rendering JSON Data</p> <p>Rendering JSON Data:</p> <p>Rendering JSON data involves presenting it in a human-readable format, typically within a user interface, web page, or application. JSON (JavaScript Object Notation) is a lightweight data interchange format commonly used for transmitting data between a server and a client, or between different parts of an application.</p> <p>Here's a detailed explanation of rendering JSON data:</p> <p>Understanding JSON: JSON is a text-based data format that represents data as key-value pairs, arrays, and nested objects.</p> <p>Data Retrieval: Before rendering JSON data, it needs to be retrieved from a data source.</p> <p>Parsing JSON: Once the JSON data is retrieved, it needs to be parsed to convert it from a string format into a JavaScript object. Most modern programming languages and frameworks provide built-in functions or libraries to parse JSON data.</p> <p>Rendering in UI: After parsing, the JSON data can be rendered in the user interface (UI) of a web page or application.</p> <p>Updating Data: In interactive applications, JSON data may need to be dynamically updated or refreshed based on user interactions or external events.</p> <p>Error Handling: When rendering JSON data, it's important to handle errors gracefully.</p>	[L2][CO4]	[6M]
	b	<p>Describe about Node package manager.</p> <p>Node Package Manager (NPM):</p> <p>Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node.js packages in your application. It is also an online repository for open-source Node.js packages.</p> <p>It has now become a popular package manager for other open-source JavaScript frameworks like AngularJS, jQuery, Gulp, Bower etc.</p> <p>Official website: https://www.npmjs.com</p> <p>NPM is included with Node.js installation. After you install Node.js, verify NPM installation by writing the following command in terminal or command prompt.</p> <p>C:\> npm -v</p>	[L2][CO4]	[6M]

2.11.3

NPM performs the operation in two modes: global and local. In the global mode, NPM performs operations which affect all the Node.js applications on the computer whereas in the local mode, NPM performs operations for the particular local directory which affects an application in that directory only.

Install Package Locally:

Use the following command to install any third party module in your local Node.js project folder.

```
C:\>npm install <package name>
```

For example, the following command will install ExpressJS into MyNodeProj folder.

```
C:\MyNodeProj> npm install express
```

Add Dependency into package.json:

Use --save at the end of the install command to add dependency entry into package.json of your application.

For example, the following command will install ExpressJS in your application and also adds dependency entry into the package.json.

```
C:\MyNodeProj> npm install express --save
```

Install Package Globally:

NPM can also install packages globally so that all the node.js application on that computer can import and use the installed packages.

Apply -g in the install command to install package globally. For example, the following command will install ExpressJS globally.

```
C:\MyNodeProj> npm install -g express
```

Update Package:

To update the package installed locally in your Node.js project, navigate the command prompt or terminal window path to the project folder and write the following update command.

```
C:\MyNodeProj> npm update <package name>
```

The following command will update the existing ExpressJS module to the latest version.

```
C:\MyNodeProj> npm update express
```

	<p>Uninstall Packages:</p> <p>Use the following command to remove a local package from your project.</p> <p>C:\>npm uninstall <package name></p> <p>The following command will uninstall ExpressJS from the application.</p> <p>C:\MyNodeProj> npm uninstall express</p>		
10	<p>Create a shopping cart application</p> <p><u>NewServlet1.java</u></p> <pre> import java.io.IOException; import java.io.PrintWriter; import java.util.ArrayList; import javax.servlet.ServletException; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import javax.servlet.http.HttpSession; public class NewServlet1 extends HttpServlet { @Override protected void doGet(HttpServletRequest rq, HttpServletResponse rs) throws ServletException, IOException { HttpSession s=rq.getSession();int itemcount=0; ArrayList cart; cart = (ArrayList)s.getAttribute("cart"); if(cart!=null) { itemcount=cart.size(); } rs.setContentType("text/html"); PrintWriter p=rs.getWriter(); p.println("<html><head><title> shopping cart 1 </title></head><body>"); p.println("<h1>Welcome to the shopping cart</h1>"); p.println("<p>You've "+itemcount+" items in your cart.</p>"); p.println("<form action='"); p.println(rs.encodeURL("http://localhost:8084/simple1/NewServlet")); p.println("<method='post'>"); p.println("<p><input type=checkbox name=item value=java>"); p.println("Item1:JAVA </p>"); p.println("<p><input type=checkbox name=item value=c>"); p.println("Item2:c </p>"); p.println("<p><input type=checkbox name=item value=c++>"); p.println("Item3:C++ </p>"); p.println("<p><input type=checkbox name=item value=C#>"); p.println("Item4:C# </p>"); p.println("<p><input type=submit value=Add to cart>"); p.println("</form></body><html>"); p.close(); } </pre>	[L6][CO4]	[12M]

NewServlet.java

```

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.util.*;
public class NewServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)

throws ServletException, IOException {
    HttpSession ses=request.getSession(true);
    ArrayList cart=(ArrayList)ses.getAttribute("cart");
if(cart==null)
{
cart=new ArrayList();
ses.setAttribute("cart",cart);
}
PrintWriter out=response.getWriter();
response.setContentType("text/html");
String[] itemselected;
String itemname;
itemselected=request.getParameterValues("item");
if(itemselected!=null)
{
for(int i=0;i<itemselected.length;i++)
{
itemname=itemselected[i];

    cart.add(itemname);
    }

}

//print the contents of the cart
out.println("<html><head><title>Shopping cart contents</title></head>");
out.println("<body>");
out.println("<form action=\"");

out.println(response.encodeURL("/simple1/NewServlet1"));

out.println("\"method=\"post\">");
out.println("<h1>Items currently in your cart</h1>");
out.println("<hr>");

Iterator iterator=cart.iterator();
while(iterator.hasNext())
{
out.println("<p>" + iterator.next() + "</p>");
}

```

<pre>out.println("<hr>"); out.println("Back"); out.println("</form>"); out.println("<hr>"); out.println("</body></html>"); out.close(); } }</pre>		
---	--	--

UNIT –V

Storage, Web toolkits - Backend and Frontend Web frameworks

1	a	<p>Explain Briefly MongoDB Framework</p> <p>MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.</p> <p>In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.</p> <p>MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.</p> <p>The manufacturing company 10gen has defined MongoDB as:</p> <p>"MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen</p> <p>MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.</p> <p>The primary purpose of building MongoDB is:</p> <ul style="list-style-type: none"> • Scalability • Performance • High Availability • Scaling from single server deployments to large, complex multi-site architectures. • Key points of MongoDB • Develop Faster • Deploy Easier • Scale Bigger <p>MongoDB's Aggregation Framework enable aggregation language, to the database to execute the workload against the data it holds enables users to send an analytics or data processing workload, written using queries.</p> <p>You can think of the Aggregation Framework as having two parts:</p> <ol style="list-style-type: none"> 1. The Aggregations API provided by the MongoDB Driver embedded in each application to enable the application to define an aggregation task called a pipeline and send it to the database for the database to process 2. The Aggregation Runtime running in the database to receive the pipeline request from 	[L2][CO5]	[6M]
----------	----------	---	------------------	-------------

	b	<p>Discuss the features of MongoDB.</p> <p>Features of MongoDB</p> <p>These are some important features of MongoDB:</p> <ol style="list-style-type: none"> 1. Support ad hoc queries In MongoDB, you can search by field, range query and it also supports regular expression searches. 2. Indexing You can index any field in a document. 3. Replication MongoDB supports Master Slave replication. A master can perform Reads and Writes and a Slave copies data from the master and can only be used for reads or back up (not writes) 4. Duplication of data MongoDB can run over multiple servers. The data is duplicated to keep the system up and also keep its running condition in case of hardware failure. 5. Load balancing It has an automatic load balancing configuration because of data placed in shards. 6. Supports map reduce and aggregation tools. 7. Uses JavaScript instead of Procedures. 8. It is a schema-less database written in C++. 9. Provides high performance. 10. Stores files of any size easily without complicating your stack. 11. Easy to administer in the case of failures. 12. It also supports: <ul style="list-style-type: none"> ◦ JSON data model with dynamic schemas ◦ Auto-sharding for horizontal scalability ◦ Built in replication for high availability ◦ Now a day many companies using MongoDB to create new types of applications, improve performance and availability. 	[L2][CO5]	[6M]
2		<p>Explain Accessing of MongoDB Documents from Node.js</p> <p>Access MongoDB With Node. JS</p>	[L2][CO5]	[12M]

Install MongoDB.
Create Project.
Create Database.
Create Collection.
Insert Data.
Find Data.
Update Data.
Delete Data.

Access MongoDB in Node.js

Learn how to access document-based database MongoDB using Node.js in this section.

In order to access MongoDB database, we need to install MongoDB drivers. To install native [mongodb](#) drivers using NPM, open command prompt and write the following command to install MongoDB driver in your application.

```
npm install mongodb --save
```

This will include mongodb folder inside node_modules folder. Now, start the MongoDB server using the following command. (Assuming that your MongoDB database is at C:\MyNodeJSConsoleApp\MyMongoDB folder.)

```
mongod -dbpath C:\MyNodeJSConsoleApp\MyMongoDB
```

Connecting MongoDB

The following example demonstrates connecting to the local MongoDB database.

app.js

```
var MongoClient = require('mongodb').MongoClient;

// Connect to the db
MongoClient.connect("mongodb://localhost:27017/MyDb", function (err, db) {

    if(err) throw err;

    //Write database Insert/Update/Query code here..

});
```

Insert Documents

The following example demonstrates inserting documents into MongoDB database.

```
app.js

var MongoClient = require('mongodb').MongoClient;

// Connect to the db
MongoClient.connect("mongodb://localhost:27017/MyDb", function (err, db) {

    db.collection('Persons', function (err, collection) {


        collection.insert({ id: 1, firstName: 'Steve', lastName: 'Jobs' });
        collection.insert({ id: 2, firstName: 'Bill', lastName: 'Gates' });
        collection.insert({ id: 3, firstName: 'James', lastName: 'Bond' });

        db.collection('Persons').count(function (err, count) {
            if (err) throw err;

            console.log('Total Rows: ' + count);
        });
    });
});
```

Update/Delete Documents

The following example demonstrates updating or deleting an existing documents(records).

```
app.js  Copy

var MongoClient = require('mongodb').MongoClient;

// Connect to the db
MongoClient.connect("mongodb://localhost:27017/MyDb", function (err, db) {

    db.collection('Persons', function (err, collection) {

        collection.update({id: 1}, { $set: { firstName: 'James', lastName: 'Gosling' } }, {w:1},
            function(err, result){
                if(err) throw err;
                console.log('Document Updated Su

    });

    collection.remove({id:2}, {w:1}, function(err, result) {

        if(err) throw err;

        console.log('Document Removed Successfully');
    });

});

});
```

	<h2>Query Database</h2> <p>The following example demonstrates executing a query in the MongoDB database.</p> <pre> app.js var MongoClient = require('mongodb').MongoClient; // Connect to the db MongoClient.connect("mongodb://localhost:27017/MyDb", function (err, db) { db.collection('Persons', function (err, collection) { collection.find().toArray(function(err, items) { if(err) throw err; console.log(items); }); }); }); </pre> <p>So, in this way you can connect and access MongoDB database.</p>		
3	<p>Create a MongoDB collection of “Research articles “with required details</p> <p>Create a MongoDB collection of “Research articles “with required details</p> <pre> package net.javaguides.mongodb.collection; import java.util.ArrayList; import org.bson.Document; import com.mongodb.MongoCommandException; import com.mongodb.client.MongoClients; import com.mongodb.client.MongoCollection; /** * Java MongoDB Create Collection Example * @author Ramesh Fadatare */ public class MongoCreateCollection { public static void main(String[] args) { try (var mongoClient = MongoClients.create("mongodb://localhost:27017")) { var database = mongoClient.getDatabase("javaguides"); try { database.createCollection("users"); System.out.println("Collection created successfully"); } catch (MongoCommandException e) { database.getCollection("users").drop(); } var docs = new ArrayList < Document > (); </pre>	[L6][CO5]	[12M]

		<pre> MongoCollection < Document > collection = database.getCollection("users"); var d1 = new Document("_id", 1); d1.append("_firstName", "Ramesh"); d1.append("_lastName", "Fadatare"); docs.add(d1); var d2 = new Document("_id", 2); d2.append("_firstName", "Tony"); d2.append("_lastName", "Stark"); docs.add(d2); var d3 = new Document("_id", 3); d3.append("_firstName", "Tom"); d3.append("_lastName", "Cruise"); docs.add(d3); var d4 = new Document("_id", 4); d4.append("_firstName", "Amir"); d4.append("_lastName", "Khan"); docs.add(d4); var d5 = new Document("_id", 5); d5.append("_firstName", "Umesh"); d5.append("_lastName", "Fadatare"); docs.add(d5); collection.insertMany(docs); } } } </pre>		
4	a	<p>List out Various Backend Web frameworks and explain briefly.</p> <ol style="list-style-type: none"> 1. Django 2. Laravel 3. Express 4. Spring Boot 5. Ruby on Rails 6. Flask 7. ASP.NET 	[L1][CO5]	[6M]
	b	List out Various Frontend Web frameworks and explain in detail.	[L1][CO5]	[6M]

	<p>Top Most Popular Frontend Frameworks 2023</p> <ol style="list-style-type: none"> 1. React. 2. Angular. 3. JQuery. 4. Vue.js. 5. Backbone.js. 6. Ember.js. 7. Semantic-UI. 8. SveltE 		
5	<p>Explain Manipulating of MongoDB Documents from Node.js</p> <pre>// Importing mongoose module const mongoose = require("mongoose"); // Database Address const url = "mongodb://localhost:27017/GFG"; // Connecting to database mongoose .connect(url) .then((ans) => { console.log("Connected Successful"); }) .catch((err) => { console.log("Error in the Connection"); }); // Calling Schema class const Schema = mongoose.Schema; // Creating Structure of the collection const collection_structure = new Schema({ name: { type: String, required: true, }, marks: { type: Number, }, }); // Creating collection const collections = mongoose.model("GFG2", collection_structure); // Inserting one document collections .create({</pre>	[L2][CO5]	[12M]

		<pre> // Inserting value of only one key name: "GFG", marks: "10q0", // Inserting wrong value }) .then((ans) => { console.log(ans); }) .catch((err) => { console.log(err.message); }); </pre>		
6	a	<p>Discuss about Django framework in detail?</p> <p>It is developed in the context of several architecture ideologies.</p> <p>Every stack part is designed to render it separate and, thus, loosely connected.</p> <p>The developer will write fewer code while encouraging the fast creation of applications.</p> <p>When completed, the model does not replicate itself, maybe easily replicated at several intersections in application formation.</p> <p>Consequently, it encourages best practices in technology by keeping a clean architecture in its own application and thus a super-fast growth.</p> <p>It is a high-level web framework from Python that allows for the rapid implementation of security and sustainable websites.</p> <p>Designed by seasoned programmers, it takes care of most of the web creation headache and you can focus on developing the software without the need to start from scratch.</p> <p>It's free and open-source, it has a vibrant and involved community, excellent resources, and lots of free and paid services.</p> <p>It has a customized caching system.</p> <p>Django REST framework is one of the best tools for building API's.</p> <p>Applications:</p> <p>Data-analysis tools.</p> <p>Photo-based verification systems.</p> <p>E-mail systems.</p> <p>Famous companies that use Django framework:</p> <p>Instagram</p> <p>NASA</p> <p>Dropbox</p> <p>Spotify etc.</p>	[L2][CO5]	[6M]
	b	<p>Illustrate Ruby on Rails framework?</p> <p>Ruby on Rails (RoR) is a server-side web application framework written in the Ruby programming language. It was developed by David Heinemeier Hansson and</p>	[L3][CO5]	[6M]

		<p>released in 2004. Rails follows the Model-View-Controller (MVC) architecture pattern.</p> <p>Key Features of Rails as a Backend Framework</p> <p>Active Record: The built-in object-relational mapping (ORM) system that simplifies database interactions. Allows developers to perform CRUD operations (Create, Read, Update, Delete) on database objects using Ruby code. Migrations facilitate easy database schema changes over time.</p> <p>Routing: Rails' routing system maps incoming HTTP requests to controller actions. RESTful routing follows standard practices for creating web services, making it easy to build APIs.</p> <p>Controllers and Actions: Controllers handle web requests and coordinate responses. Each action in a controller corresponds to a specific functionality, such as viewing or editing data.</p> <p>Middleware and Background Jobs</p> <p>Middleware: Acts as a bridge between incoming HTTP requests and the application, used for tasks such as authentication or logging.</p> <p>Background Jobs: Handle tasks outside the request-response cycle, like sending emails or processing images.</p> <p>Pros and Cons</p> <p>Pros: Rapid development cycle due to a strong convention over configuration. Large and active community provides a wealth of resources, documentation, and libraries. Mature ecosystem with many gems that simplify common tasks.</p> <p>Cons: Performance may degrade with very large applications unless optimized. Opinionated framework: Rails' conventions may not suit all projects.</p>		
7	a	<p>Compare the main differences between a MongoDB and a traditional SQL database.</p> <p>Data Model</p> <ul style="list-style-type: none"> • MongoDB: Uses a document-oriented model, where data is stored in flexible, JSON-like BSON documents. Each document can have its own unique structure. • SQL Databases: Follow a relational model, where data is organized into predefined tables with rows and columns. Each row must adhere to a fixed schema. 	[L6][CO5]	[6M]
		2. Schema		

		<ul style="list-style-type: none"> • MongoDB: Schema-less or schema-flexible. It allows for dynamic fields and structures, making it suitable for unstructured or semi-structured data. • SQL Databases: Enforce a strict schema. The table structure must be defined beforehand, and any changes to it can be complex. 		
		3. Scalability <ul style="list-style-type: none"> • MongoDB: Built for horizontal scaling using sharding, distributing data across multiple servers for scalability and high availability. • SQL Databases: Primarily use vertical scaling, adding more power (CPU, RAM) to a single server. Some relational databases support sharding but with more complexity. 		
		4. Query Language <ul style="list-style-type: none"> • MongoDB: Uses a non-SQL query language, with a rich set of operators for filtering and aggregating data, often resembling JavaScript syntax. • SQL Databases: Use SQL (Structured Query Language), a standard query language for defining, manipulating, and querying data. 		
		5. Transactions <ul style="list-style-type: none"> • MongoDB: Supports multi-document ACID transactions (introduced in version 4.0), but it is relatively newer and less robust compared to SQL databases. • SQL Databases: Provide strong ACID compliance and have well-established support for complex multi-statement transactions. 		
		6. Use Cases <ul style="list-style-type: none"> • MongoDB: Best for applications needing flexibility, handling unstructured data, or requiring rapid development (e.g., content management systems, IoT applications). • SQL Databases: Ideal for applications with structured data and complex relationships (e.g., banking, enterprise resource planning). 		
	b	What are some security considerations when developing backend web applications? 1. Input Validation and Sanitization <ul style="list-style-type: none"> • Why: Prevent malicious input, such as SQL injection, cross-site scripting 	[L1][CO5]	[6M]

		<p>(XSS), or buffer overflows.</p> <ul style="list-style-type: none"> • How: <ul style="list-style-type: none"> ○ Validate user input on both client and server sides. ○ Sanitize inputs by escaping special characters. ○ Use prepared statements or parameterized queries for database interactions. 		
		<p>2. Authentication and Authorization</p> <ul style="list-style-type: none"> • Why: Ensure only legitimate users can access the system and resources. • How: <ul style="list-style-type: none"> ○ Use secure authentication methods (e.g., OAuth, JWT, password hashing). ○ Implement multi-factor authentication (MFA). ○ Follow the principle of least privilege (grant minimal permissions). 		
		<p>3. Secure Data Storage</p> <ul style="list-style-type: none"> • Why: Protect sensitive data, like user credentials and personal information. • How: <ul style="list-style-type: none"> ○ Encrypt sensitive data at rest and in transit using strong encryption algorithms (e.g., AES-256, TLS). ○ Avoid storing plain-text passwords; hash them using secure hashing algorithms (e.g., bcrypt, Argon2). 		
		<p>4. Protection Against Common Attacks</p> <ul style="list-style-type: none"> • Why: Defend against well-known threats like: <ul style="list-style-type: none"> ○ SQL Injection: Exploit vulnerabilities in SQL queries. ○ Cross-Site Scripting (XSS): Inject malicious scripts into web pages. ○ Cross-Site Request Forgery (CSRF): Trick users into performing unwanted actions. • How: <ul style="list-style-type: none"> ○ Use frameworks and libraries that help prevent these attacks. ○ Implement Content Security Policies (CSP) for XSS. ○ Use CSRF tokens to validate requests. 		
		<p>5. Secure APIs and Endpoints</p> <ul style="list-style-type: none"> • Why: Prevent unauthorized access to backend services. • How: <ul style="list-style-type: none"> ○ Authenticate and authorize all API requests. ○ Rate-limit APIs to prevent abuse. 		

		<ul style="list-style-type: none"> ○ Use HTTPS for secure communication. 		
		<p>6. Error Handling and Logging</p> <ul style="list-style-type: none"> • Why: Avoid revealing sensitive information through error messages or logs. • How: <ul style="list-style-type: none"> ○ Customize error messages to hide stack traces or sensitive data. ○ Use secure logging practices (e.g., avoid logging sensitive user data). 		
8	a	<p>What is Meteor JS framework and explain in detail with an example?</p> <p>What is Meteor.js?</p> <p>Meteor.js is a full-stack, open-source JavaScript framework designed for building modern web and mobile applications. It is built on top of Node.js and provides an integrated environment for both frontend and backend development. Meteor simplifies the development process by enabling developers to use JavaScript across the stack and seamlessly synchronize data between the client and server.</p> <p>Key Features of Meteor.js</p> <ol style="list-style-type: none"> 1. Isomorphic JavaScript: Use JavaScript for both client-side and server-side code. 2. Real-Time Updates: Built-in support for real-time data synchronization between the server and client using WebSockets. 3. Unified Environment: Combines frontend (e.g., React, Angular, Blaze), backend (Node.js), and database (MongoDB). 4. Reactive Programming: Automatically updates UI when the underlying data changes. 5. Built-in Packages: Comes with tools for authentication, routing, and bundling. 6. Cross-Platform Development: Allows developers to create web, iOS, and Android apps from the same codebase. <p>How Meteor Works</p> <ul style="list-style-type: none"> • Database: Typically uses MongoDB. Meteor's <code>MiniMongo</code> (client-side in-memory database) mirrors the server-side database for real-time data synchronization. • Server: Runs on Node.js and handles business logic, database interactions, and publishes data. • Client: The frontend subscribes to data from the server and updates dynamically in real-time using reactivity. 	[L1][CO5]	[6M]
	b	Explain features of Meteor JS framework.	[L2][CO5]	[6M]

		<p>Features of Meteor.js Framework</p> <p>Meteor.js is a full-stack JavaScript framework designed to simplify the development of modern web and mobile applications. Below are its key features:</p>		
		<p>1. Full-Stack Solution</p> <ul style="list-style-type: none"> Meteor.js provides an integrated environment that covers both the frontend and backend, as well as the database layer. 		
		<p>2. Real-Time Data Synchronization</p> <ul style="list-style-type: none"> One of Meteor's standout features is its ability to enable real-time updates. This is achieved through Meteor's data-on-the-wire architecture using WebSockets. 		
		<p>3. Reactive Programming</p> <ul style="list-style-type: none"> Meteor provides reactive data sources, enabling applications to update the user interface dynamically when underlying data changes. 		
		<p>4. Isomorphic JavaScript</p> <ul style="list-style-type: none"> Meteor allows developers to write both server-side and client-side code in JavaScript, unifying the development process. 		
		<p>5. Database Integration (MongoDB and MiniMongo)</p> <ul style="list-style-type: none"> Meteor is tightly integrated with MongoDB. This ensures that client and server data remain synchronized in real time. 		
		<p>6. Built-in Package Management</p> <ul style="list-style-type: none"> Meteor comes with its own package management system, powered by Atmosphere.js, which allows developers to easily add pre-built functionality (e.g., authentication, routing). 		
		<p>7. Cross-Platform Development</p>		

		<ul style="list-style-type: none"> Meteor enables the development of both web and mobile applications from the same codebase. 		
		8. Ease of Deployment <ul style="list-style-type: none"> Meteor provides tools for seamless deployment of applications to production. 		
9	a	What are the Features of Django framework and explain in detail? Features of Django Framework Django is a high-level, open-source web framework for Python that promotes rapid development and clean, pragmatic design. It is widely used for building robust, scalable, and secure web applications. Below are its key features explained in detail:	[L1][CO5]	[6M]
		1. MVT Architecture (Model-View-Template) <ul style="list-style-type: none"> Django follows the Model-View-Template (MVT) architectural pattern: <ul style="list-style-type: none"> Model: Handles database interactions and defines the data structure. View: Contains business logic and processes requests and responses. Template: Renders dynamic HTML pages for the user interface. 		
		2. Built-in Admin Interface <ul style="list-style-type: none"> Django provides an automatic admin interface that is generated from the models. It includes features like filtering, searching, and user authentication out-of-the-box. 		
		3. ORM (Object-Relational Mapping) <ul style="list-style-type: none"> Django includes a powerful Object-Relational Mapper that enables developers to interact with databases using Python code instead of writing raw SQL queries. It supports multiple databases (e.g., PostgreSQL, MySQL, SQLite). 		
		4. Scalability and Reusability <ul style="list-style-type: none"> Django is designed to build both small-scale and large-scale applications. Its modular approach allows developers to reuse components like apps, 		

		middleware, and templates across projects.		
		<p>5. Security</p> <ul style="list-style-type: none"> Django provides robust security features to protect web applications from common vulnerabilities: <ul style="list-style-type: none"> Cross-Site Scripting (XSS) prevention. Cross-Site Request Forgery (CSRF) protection. SQL Injection prevention via query parameterization. Clickjacking protection with X-Frame-Options. 		
		<p>6. URL Routing</p> <ul style="list-style-type: none"> Django uses a clean and flexible URL dispatcher for mapping URLs to views. It simplifies creating human-readable and SEO-friendly URLs. 		
	b	<p>What are the Features of Meteor framework and explain with example?</p> <p>1. Full-Stack JavaScript Framework</p> <ul style="list-style-type: none"> Feature: Meteor allows developers to use JavaScript for both client-side and server-side programming, creating a unified development environment. <p>2. Real-Time Data Synchronization</p> <ul style="list-style-type: none"> Feature: Meteor enables real-time synchronization of data between the client and the server using WebSockets, without the need for manual AJAX calls. <p>3. Isomorphic Code</p> <ul style="list-style-type: none"> Feature: Code can be shared between the client and the server. For example, validation logic can be reused, reducing redundancy. <p>4. Real-Time Frontend</p> <ul style="list-style-type: none"> Feature: Meteor integrates seamlessly with frontend frameworks like React, Angular, Blaze, and Vue.js to provide a responsive and real-time user interface. <p>5. Built-In Package Management</p> <ul style="list-style-type: none"> <p>Feature: Meteor has a built-in package manager, Atmosphere.js, for adding functionality (e.g., authentication, routing). It also supports NPM for Node.js</p>	[L1][CO5]	[6M]

		<p>packages.</p> <ul style="list-style-type: none"> • <p>6. Cross-Platform Development</p> <ul style="list-style-type: none"> • Feature: Meteor allows developers to build web, iOS, and Android applications from the same codebase. 		
10	a	<p>Illustrate Angular framework with example.</p> <p>Angular Framework Overview</p> <p>Angular is a popular open-source web application framework developed and maintained by Google. It is built with TypeScript and is designed for building dynamic, single-page web applications (SPAs). Angular supports features like two-way data binding, dependency injection, modular development, and a rich component-based architecture.</p> <p>Illustrating Angular with an Example</p> <p>Here are the key features of the Angular framework:</p>	[L3][CO5]	[6M]
		<p>1. Component-Based Architecture</p> <ul style="list-style-type: none"> • Applications are built as a collection of reusable components, each encapsulating its logic, templates, and styles. • Promotes modular development, making code more maintainable and scalable. 		
		<p>2. Two-Way Data Binding</p> <ul style="list-style-type: none"> • Synchronizes the data between the model (logic) and the view (UI) in real time. • Reduces the need for manual DOM manipulation. 		
		<p>3. Dependency Injection (DI)</p> <ul style="list-style-type: none"> • Built-in DI mechanism helps manage services and dependencies efficiently. • Encourages modularity and improves testability. 		

		4. Directives <ul style="list-style-type: none"> Angular allows extending HTML functionality using: <ul style="list-style-type: none"> Structural Directives (e.g., <code>*ngIf</code>, <code>*ngFor</code>) to manipulate DOM elements. Attribute Directives (e.g., <code>[class]</code>, <code>[style]</code>) to modify behavior or appearance. 		
		5. Powerful Template System <ul style="list-style-type: none"> Combines HTML with Angular's template syntax to display dynamic data. Includes features like interpolation (<code>{{ }}</code>), property binding, and event binding. 		
		6. Reactive Programming with RxJS <ul style="list-style-type: none"> Angular uses RxJS (Reactive Extensions for JavaScript) for handling asynchronous operations. Provides Observables for event handling, HTTP requests, and more. 		
	b Describe Meteor framework in detail. Meteor Framework: Meteor is an open-source, full-stack JavaScript framework designed for building modern web, mobile, and desktop applications. It enables rapid development by providing an integrated environment with tools and libraries for both the client and server sides.	1. Full-Stack JavaScript Framework <ul style="list-style-type: none"> Feature: Meteor allows developers to use JavaScript for both client-side and server-side programming, creating a unified development environment. 2. Real-Time Data Synchronization <ul style="list-style-type: none"> Feature: Meteor enables real-time synchronization of data between the client and the server using WebSockets, without the need for manual AJAX calls. 3. Isomorphic Code <ul style="list-style-type: none"> Feature: Code can be shared between the client and the server. For example, validation logic can be reused, reducing redundancy. 4. Real-Time Frontend	[L2][CO5]	[6M]

	<ul style="list-style-type: none">• Feature: Meteor integrates seamlessly with frontend frameworks like React, Angular, Blaze, and Vue.js to provide a responsive and real-time user interface. <p>5. Built-In Package Management</p> <ul style="list-style-type: none">• <p>Feature: Meteor has a built-in package manager, Atmosphere.js, for adding functionality (e.g., authentication, routing). It also supports NPM for Node.js packages.</p> <ul style="list-style-type: none">• <p>6. Cross-Platform Development</p> <ul style="list-style-type: none">• Feature: Meteor allows developers to build web, iOS, and Android applications from the same codebase.		
--	---	--	--