Q1. Differentiate between XML and XSLT (4 mark)

i) XML (extensible markup language) is a markup language that is used to store & transport data, particularly structured data.

ii) It uses tags to define elements and attributes to define characteristics of those elements.

iii) XML provides a way to define and share data formats across different platforms, making it a popular format for data exchange between systems.

iv) XSLT (extensible stylesheet Language Transformations) is a language used to transform XML documents into other formats such as HTML, XHTML or even another XML format.

v) It uses an XML-based syntax to describe the transformation process, & it provides a way to selectively extract data from an XML document.

In summary, XML is used for data storage and exchange while XSLT is used to transform XML documents into other formats.

Q.2 Summarize XML schemas and how are they better than DTDs. Explain the difference between external and internal DTDs. (6 marks)

↳ Document Type Definition

i) XML Schemas

a) XML Schemas used to define the structure, content, and data types of XML documents.

b) They provide a way to define a set of rules that an XML document must follow, making it easier to validate and process the document.

c) It is use more powerful and expressive language than DTDs, allowing for more complex data types and structure.

d) XML Schemas are written in XML, making them easier to read, write and maintain.

e) Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Archi</to>
  <from>Parsha</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

ii) How XML Schemas are Better than DTDs:

a) XML Schemas provide better validation capabilities than DTDs.

b) It allow for the creation of user-defined data types.

c) XML Schemas are written in XML, making easier to integrate with other XML-based technologies

d) They are more extensible than DTDs, allow creation of more complex structure & data types.

iii) External and Internal DTDs.

a) DTD can be defined either internally or externaly to an XML documents.

b) A internal DTD is defined within the XML document itself, typically within the DOCTYPE declaration

c) External DTD is defined in a seperate file and referenced from within the XML document using SYSTEM or PUBLIC identifier.

d) External DTDs allow for the reose of DTDs across multiple documents, making it easier to maintain consistency and in data structure.

e) Internal DTDs are easier to distribute and manage, as they are ~~combined~~ contained within the XML document itself.

Q3 Design a form to accept workshop registeration details from participants and validate using Java Script? (4 mark)

```
<!DOCTYPE html>
<html>
<head>
    <title>Workshop Registration Form</title>
    <script>
        function validateForm() {
            var name = document.forms["form"]["name"].value;
            var email = document.forms["form"]["email"].value;
            var phone = document.forms["form"]["phone"].value;
            var workshop = document.forms["form"]["workshop"].value;

            if (name == "") {
                alert("Please enter your name.");
                return false;
            }

            .
            .
            .
            if(workshop == "") {
                alert("Please enter your a workshop.");
                return false;
            }
            return true;
        }
    </script>
</head>
```

```
<body>
    <h1> Workshop Registration form</h1>
    <form name="form" onsubmit="return validateform()"
          method="post">
        <label for="name"> Name: </label>
        <input type="text" id="name" name="name"><br>

        <label for="email"> Email: </label>
        <input type="email" id="email" name="email"><br>

        <label for="phone"> Phone: </label>
        <input type="text" id="phone" name="phone"><br>

        <label for="workshop"> Workshop: </label>
        <select id="workshop" name="workshop">
            <option value=""> Please select a workshop </option>
            <option value="Web Technology"> Web T. </option>
            <option value="Cloud Computing"> Cloud Computing </option>
            <option value="Compiler Design"> Compiler Design </option>
        </select> <br>

        <input type="submit" value="submit">
    </form>
</body>
</html>
```

Q.4 Show the document object properties with proper syntax with example.

i) document.getElementById()
ii) document.getElementByClass()
iii) document.getElementByName() (6 marks)

1) document.getElementById()

syntax: 'document.getElementById("elementId");'

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title> Get Element By Id Example </title>
</head>
<body>
    <h1 id="pageTitle"> Welcome to my Website </h1>
    <p id="description"> lorem ipsum. </p>

    <script>
        var pageTitle = document.getElementById("pageTitle");
        var des = document.getElementById("description");
        pageTitle.innerHTML = "Hello world!";
        des.style.color = "red";
    </script>
</body>
</html>
```

2) document.getElementByClass()

syntax: 'document.getElementByClasss("className");'

example:

```
<body>
    <p class = "highlight"> This paragraph will be highlighted </P>
    <P> This line will not highlighted </P>

    <script>
        var highpara = document.getElementsByClassName("highligh
                                                        );

        for(var i=0; i< highpara.length; i++){
            highpara[i].style.fontWeight = "Bold";
        }
    </script>
</body>
```

3) document.getElementByName()

syntax: 'document.getElementByName("name");'

example:

```
<body>
    <form>
        <label for = "Name"> Name: </label>
        <input type = "text" name = "name" id = "name"><br>
        <input type = "submit" value = "submit">
    </form>

    <script>
        var name = document.getElementByName("name")[0];
        name.value = "John";
    </script>
</body>
```

Note: document.getElementByName() returns an array-like object, so we need to use '[0]' to access the first element.

Q.5 Propose the steps involved in connecting to MySQL with PHP? (4 mark)

i) Install MySQL on server or local machine
ii) Install PHP on server or local machine
iii) Create a database on MySQL
iv) Create a User with appropriate privileges to access the database.
v) Connect to MySQL from PHP with 'mysql_connect()' function.

```
<?php

$conn = mysql_connect("localhost", "username", "password",
                      "databasename");

if(!$conn) {
    die("Connection failed", mysql_connect_error());
}
echo "Connected Successfully";
?>
```

vi) Execute queries:

```
<?php
$sql = "SELECT * FROM table_name";
$result = mysql_query($conn, $sql);

if (!$result) {
    die ("Query failed:" . mysql_error($conn));
}
while ($row = mysqli_fetch_assoc($result)){
    echo "id: ", $row["id"], " - Name". $row["name"]. "<br>";
}
mysqli_close($conn);
?>
```

Q.6 Provide solution realisation to include PHP server with sample code. (6 mark)

i) Install XAMPP server that include Apache, PHP and MySQL in one package.

ii) Create PHP file with '.php' extension.

iii) Start the server from XAMPP

iv) place the PHP file in the server's document root directory.

if you are using Apache server on windows, the default document root directory is "C:\xampp\htdocs".

v) Access the PHP file from web browser by entering the URL of the file

for example, if the file named "myfile.php" is located in root directory "C:\xampp\htdocs", you can access it by entering "http://localhost/myfile.php".

vi) Sample code:

```
<!DOCTYPE html>
<html>
<body>
<?php
    echo "Hello, world!";
?>
</body>
</html>
```

Q.7 Design and develop a program to demonstrate button styles (warning, danger, Default, Primary, links, success) using Bootstrap. (6 mark)

```
<!DOCTYPE html>
<html>
  <head>
    <title> Bootstrap Button Styles</title>
    <link rel="stylesheet" href="https:---"> // Bootstrap CSS file
  </head>
  <body>
    <div class="container mt-5">
    <h1> Bootstrap Button styles </h1>
    <p> Click on the button below to see different styles: </p>
    <button class="btn btn-warning"> Warning Button </button>
    <button class="btn btn-danger"> Danger Button </button>
    <button class="btn btn-default"> Default Button </button>
    <button class="btn btn-primary"> Primary Button </button>
    <button class="btn btn-link"> Link Button </button>
    <button class="btn btn-success"> Success Button </button>
    </div>
    <!-- Bootstrap JS file -->
    <script src=" "></script>
  </body>
</html>
```

Q.8 Exemplify the most common Spring Boot CLI commands? (4 mark)

i) 'spring init' - initializes a new Spring Boot Project

example - 'spring init --dependencies=web,data-jpa myproject'

ii) 'spring run' - this commands runs a 'spring Boot application.

example - 'spring run myapp.groovy'

iii) 'spring test' - this command runs tests for a spring Boot application

example - 'spring test --watch MyTest'

iv) 'spring jar' - builds a standalone executable JAR file for Spring Boot application.

example - 'spring jar myapp.jar myapp.groovy.

Q9 Reveal how does a spring boot application gets started? Interpret starter dependencies in Spring boot? (6 marks)

~~i) When a spring boot application started,~~

i) A Spring Boot application is started by executing the 'main()' method of the 'Application' class.

ii) The '@SpringBootApplication' annotation is commonly used to annotate the 'Application' class.

iii) The '@SpringBootApplication' combines three annotations: '@Configuration', '@EnableAutoConfiguration', '@ComponentScan'.

iv) The first annotation indicates that the class contains Spring Configuration Information.

v) Second annotation enables auto-configuration features, which configures based on dependencies that are on classpath.

vi) third annotation scans application's packages, such as controllers, services and repositories.

vii) Once the application context is started, it initialize all the spring beans including controllers, services, repositories and other components.

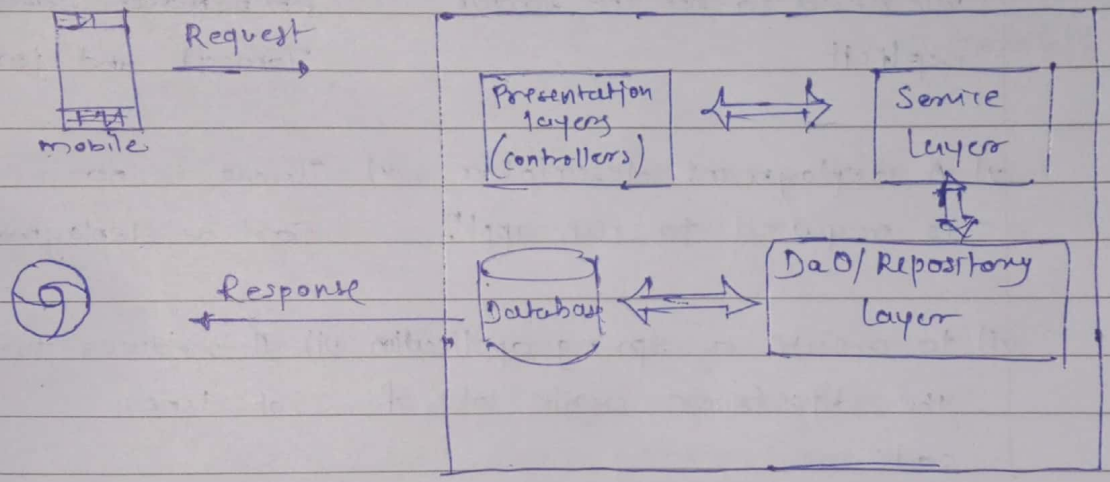
viii) After initialization is complete, application is ready to handle requests and respond.

Starter Dependencies -

i) Spring Boot applications can be quickly and easily configured with the required dependencies for specific functionality, without having to manually add each dependency to the project.

ii) This greatly simplifies the process of setting up a Spring Boot application & reduces the chances of version conflicts between dependencies.

Client		Server				Database
Browser - Chrome, Mozilla, Mobile Phone, Laptop, Desktop, POSTMAN	→ Request ← Reply	Presentation layer Controller uses To accept Request (what client wants)	Service layer Business logic (Classes, Method)	DAO layer Database Connectivity	→ Request ← Reply	Repository layer Database Oracle, MySQL



Request → Controller → Controller uses services from Service Layer

Database ← DAO

Spring	SpringBoot
i) It is an open source lightweight framework widely used to develop enterprise applications.	i) It is built on top of the conventional spring framework, widely used to develop REST APIs
ii) Most important feature is dependency injection	ii) most important feature is auto-configuration.
iii) It helps to create a loosely coupled appl^n.	iii) It helps to create a stand-alone application.
iv) to run Spring application, we need to set the server explicit	iv) Spring Boot provides embedded servers such as Tomcat and jerry etc.
v) A deployment descriptor is reqwired to run appl^n.	v) There is no requirement for a deployment descriptor.
vi) to create a spring application the developers write lots of code.	vi) It reduces the lines of lines
vii) It doesn't support for the in-memory database.	vii) It provides support for the in-memory database such as H2.

Q.10 How to connect Mongo DB with React JS? (4 mark)

i) Install the 'mongoose' package in React.js application

```
npm install mongoose
```

ii) In React.js component where you want to use database connection, import the mongoose package and use it to connect to your MongoDB database.

```
import mongoose from 'mongoose';

const uri = 'mongodb://localhost:27017/mydatabase';

mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology:
                                                    true });
```

iii) Define a schema for your data and create model using the 'mongoose.model()' method.

```
const dataSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true }
});

const Data = mongoose.model('Data', dataSchema);
```

Q11 Why to pass data from one component to other component using props in ReactJS? Write code for ParentComponent.js and ChildComponent.js. (6 mark)

i) Props in ReactJS:

a) Reusability - we can ~~use~~ reuse the same component in different parts of application with different data. Makes it more modular and easier.

b) Data flow control - the data flow is unidirectional meaning data can only flow downwards from parent components to child components, Using props we can control the flow of data in appl", and can ensure that components remain independant to each other.

c) Code organization :- we can keep related data and functionality in same place using props. This makes code more organized & easier to understand.

d) Permance Optimization - Using props we can avoid unnessary re-rendering of components. ReactJs is designed to only update the components that need to be updated when data changes.

ii) Code :

a) ParentComponent.js

```
import React from 'react';
import ChildComponent from './ChildComponent';

function ParentComponent() {
  const name = 'John';
  const age = 30;
  return (
    <div>
      <h1> Parent Component </h1>
      <ChildComponent name={name} age={age} />
    </div>
  );
}
export default ParentComponent;
```

b) ChildComponent.js

```
import React from 'react';
function ChildComponent(props) {
  const {name, age} = props;
  return (
    <div>
      <h2> Child Component </h2>
      <p> Name: {name} </p>
      <p> Age: {age} </p>
    </div>
  );
}
export default ChildComponent;
```

Q.12 Create a workflow to represent working of the webserver using Node.js (4 mark)

i) Request handling - The web server receives the HTTP request from a client, such as web browser.

ii) Routing - The web server uses the routing mechanism to determine which code to execute based on URL of the request.

iii) Data processing - The web server may need to process data associated with request, such as user input or data from database.

iv) Response generation - The web server generates HTTP response that includes the requested data, along with necessary HTTP headers.

v) Sending Response - The web server sends the response back to client.

vi) Error handling - The web server handles any errors that occurs during the request handling, routing, data processing, response generation stages.

vii) Example :

```
const http = require('http')
const server = http.createServer((req, res)=> {
    // Request Handling
    if (req.url === '/'){
        //routing
        if (req.method === 'GET'){
            // data processing
            const data = { message: 'Hello, world'};
            // Response generation
            res.writeHead(200, {'Content-Type': 'application/
                json'});
```

```
        res.write(JSON.stringify(data));
        // Sending response
        res.end();
    } else {
        // Error Handling
        res.writeHead(405, {'Content-Type': 'text/plain'});
        res.write('Method not allowed');
        res.end();
    }
} else {
    // Error handling
    res.writeHead(404, {'Content-Type': 'text/plain'});
    res.write('Not found');
    res.end();
}
});

const port = 3000;
server.listen(port, () => {
    console.log('Server running on port ${port}');
});
```

Q.13 Write the prefered method of resolving unhandled exceptions in Node.js? (6 mark)

i) The preferred method of resolving unhandled exception in Node.js is to use the 'process' object's 'uncaughtException' event.

ii) This event emitted when an unhandled exception occurs in Node.js process.

iii) To use this event, you need to register a listener function using 'process.on' method.

iv) The listener function takes an error object as it's argument. (contains info about unhandled exception)

v) Finally, we can call the 'process.exit' method to exit the Node.js process with error status code (1) to indicate that error occured.

vi) Example:

```
process.on('uncaughtException', (err) => {
    console.error('Uncaught exception:', err);
    //perform any necessary cleanup or error
      handling here*/
    process.exit(1);
});
```

Q.14 Write a program to handle file handling (read and write) using various functionalities that Node.js offers, and implement code to read and write to files. (6 marks)

```
// Import the fs module
const fs = require('fs');

// Define the path of the file to read and write to
const filePath = 'example.txt';

// write data to the file
fs.writeFile(filePath, 'Hello, world!', (err) => {
    if (err) throw err;
    console.log('Data written to file.');

    // Read data from the file.
    fs.readFile(filePath, 'utf8', (err, data) => {
        if (err) throw err;
        console.log('Data read from file: ${data}');

        // Append data to the file
        fs.appendFile(filePath, '\n This is a new line', (err) => {
            if (err) throw err;
            console.log('Data appended to file.');
        });
    });
});
```