ASSIGNMENT-1

① You are updating a legacy web application to take advantage of modern web standards. The project involves transitioning from an earlier HTML version to HTML5. The goal is to leverage HTML5's new features to improve code clarity, adhere to current practices, and enhance the functionality of web forms.

A: ① Doctype Declaration:

- Use '<!DOCTYPE html>' for HTML5 to ensure the document is rendered in standards mode.

② New Semantic Elements:

- Replace generic ' ' tags with semantic elements like '<header>', '<footer>', '<article>', '<section>', and '<nav>' for better structure and accessibility.

③ Form Enhancements:

- New Input Types: HTML5 introduces new input types like 'email', 'tel', 'url', 'number', 'date', and 'range', which enhance form validation and user experience.
- Placeholder: Use the "placeholder" attribute to provide hints within form fields.
- Required fields: Use the "required" attribute to enforce field completion.
- Pattern fields: Use the "required" attribute to ensure field completion.

④ Client-Side Storage:
• LocalStorage and SessionStorage offer ways to store data on the client side, which can be useful for persisting user preferences or form data.

⑤ HTML5 APIs:
• Geolocation API: To get the user's location.
• Canvas API: For drawing graphics and animations.
• Web Storage API: For storing data on the browser.

⑥ Multimedia Support:
• '<audio>' and '<video>' elements: These provide native support for embedding and controlling multimedia content without relying on third-party plugins.

⑦ Accessibility Improvements:
• HTML5's semantic elements improve accessibility by making the page structure clearer to assistive technologies.

⑧ Responsive Design:
• Use the '<meta name="viewport" content="width=device-width, initial-scale=1">' tag to ensure your website is responsive and will work well on various devices.

⑨ Modern JavaScript Integration:
• Consider leveraging modern JavaScript features and frameworks alongside HTML5 to further enhance the functionality and interactivity of your application.

(1) Designing a User registration form for a new application, the form need to capture essential information such as name, email, and a manag a form users. It is crucial to implement client-side validation to ensure that the data entered is accurate and complete before the form is submitted.

```
<! DOCTYPE html>
<html lang = "en">
<head>
    <meta charset = 'UTF-8'>
    <meta name = "viewport" content = "width = device-width,
                                       initial-scale = 1.0">
<title> User Registration Form </title>
<style>
    body {
        font-family : Arial , San-Serif;
        margin : 20px;
    }
    form {
        max-width : 400px;
        margin : 0 auto;
        Padding : 20 px ;
        border : 1px solid #ccc;
        border-radius : 5px; }

label {
        display : block;
        margin-bottom : 10px; }
input, textarea {
        width : 100%;
        Padding : 10px;
        margin-bottom : 15px;
        border : 1px solid #ccc;
        border-radius : 3px;
    }
input [type = "submit"] {
        background-color : #4CAF50;
        color : white;
        border : none;
        cursor : pointer;
}
```

```
input[type="submit"]:hover{
    background-color: #4caf50;
}
</style>
</head>
<body>
<form id="RegistrationForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <label for="message">Message:</label>
    <textarea id="message" name="message" rows="4" required></textarea>
    <input type="submit" value="Register">
</form>
<script>
document.getElementById("RegistrationForm").addEventListener("submit",
    function(event){
let name = document.getElementById("name").value;
let email = document.getElementById("email").value;
let message = document.getElementById("message").value;
if(name.trim() === ""){
    alert("Please Enter your name:");
    event.preventDefault();
}else if(email.trim() === ""){
    alert("Please enter your Email.");
    event.preventDefault();
}else if(message.trim() === ""){
    alert("Please enter your message:");
    event.preventDefault(); }});
function validationEmail(Email){
const re = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.
                [a-zA-Z]{2,6}$/;
```

return re.test(String(email).to Lower Case());

}

</script>

</body> </html>

(3) Designing a website that needs to function effectively across various devices and screen sizes. The design should include different types of layout such as fixed, fluid, and responsive. To achieve this, you need to use CSS techniques like flexbox or Grid to ensure that the layout adapts well to different screen sizes and maintain a consistent user experience.

Ans:

* Fixed layout;

```
<!DocType html>
<html lang="en">
<head>
<meta Charset="UTF-8">
<meta Name="viewport"
Content="width=device-width, initial scale=
<style>
body {
margin:0;
Padding:0;
<div class="container">
<div class="header" > fixed layout
Header </div>
<div class="main" > fixed width
Content </div>
</body>
</html>
```

* Fluid Layout:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport">
</head>
<main>
Padding: 20px;
</main>
<IBody cass </head>
</body>
<div class="container">
<div class="head">* Fluid Layout </div>
</body>
</html>
```

by incorporating these CSS techniques. We can create a website with fixed, fluid, and responsive layouts that adapt to various device to screen sizes users-friendly experience.

ASSIGNMENT-II

④. To create a webpage that offers a seamless user experience across devices, responsive design principles must be applied. This involves using fluid grid, flexible images, and media queries to ensure the layout adjusts gracefully to different screen sizes. For instance, the website might display many time with detailed content, the font content could stack vertically for easy scrolling. Navigation menus should transform into a hamburger icon on smaller screens, ensuring they remain accessible without taking up too much space. Additionally, touch-friendly elements and optimized images ensure that users enjoy a smooth experience of all devices.

* Fluid Grids: Percentage based width.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width
initial-scale=1.0">
<title> Responsive Layout with Fluid Grids </title>

<style>
    body{
      margin:0;
      font-family: arial;
box-sizing: border-box;
    }
@media (max-width: 768px){
      .container{
grid-template-columns: 1fr;
    }}
<div class="container">
  <div class="item"> Content </div>
  <div class="item"> Content </div>
</div>
</body>
</html>
```

Flexible Images;

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width initial-scale=1.
</style>
</head>
```

</body>
<img src = "example.jpg"
alt = "Example Img">
</body>
</html>

Use flexible grids for flexibility, media source for responsive design, and techniques like flexible images and touch-friendly elements to enhance usability and performance across devices.

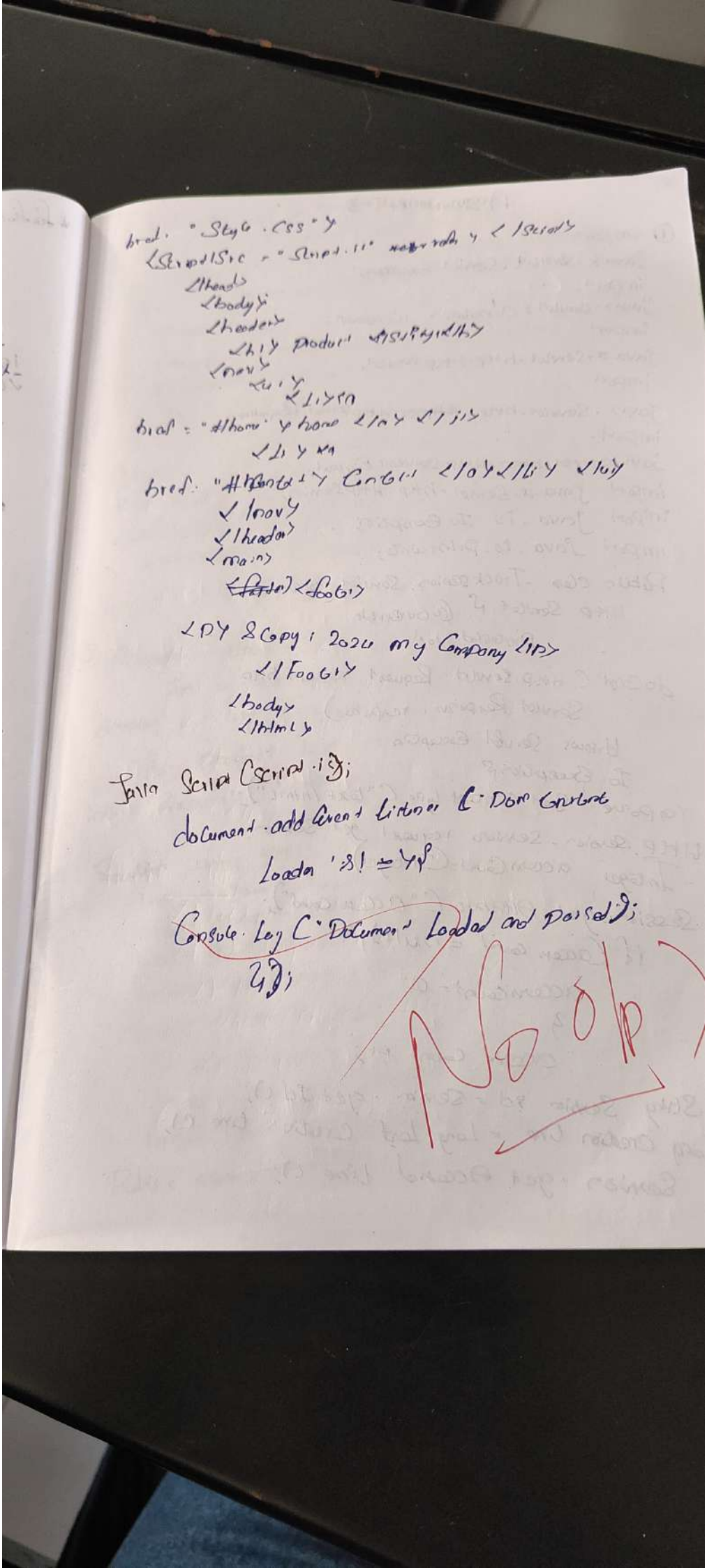
⑤ Given a scenario, design a webpage using appropriate elements, tables, lists and images for optimal readability and user experience. Also describe step by step the sequence of HTTP Request and response that occur when a user accesses a webpage containing multiple resources.

Ans:
Webpage Design: Product Listing

Sample HTML Layout that includes product images, descriptions, and prices, laid out with tables and lists where appropriate.

HTML Structure:

<!DOCTYPE Html>
<html Lang = "en">
<head>
<meta charset = "UTF-8">
<meta Name = "Viewport"
Content = "width device-width ; initial Scale
<title> Product Listing </title>



ASSIGNMENT-3

① import
javax.servlet.ServletException;
import
javax.servlet.annotation.WebServlet;
import
Javax.servlet.http.HttpServlet;
import
Javax.servlet.http.~~HttpSe~~ HttpServletResponse;
import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.io.IoException;
import java.io.PrintWriter;
Public class TrackSession Servlet extends
HttpServlet {@Override
Protected void
doGet (HttpServletRequest request, HttpServletResponse response)
throws ServletException
IoException {
response.setContentType("text/html");
HttpSession session = request.getSession();
Integer accessCount = (Integer)
session.getAttribute("accessCount");
if (accessCount == null) {
accessCount = 0;
}
accessCount++;
String SessionId = session.getId();
Long creationTime = Long.getCreationTime();
session.getLastAccessedTime();

```
PrintWriter out = response.getWriter();
out.println("<html><body>");
out.println("<h2>Session Tracking</h2>");
out.println("Session id " + session.getId() + "<br>");
out.println("Time: " + new Date() + "<br>");
out.println("Access count: " + count + "<br>");
out.println("</body></html>");
}
}
```

Output:

 Session ID : 12345678910abc

 Creation time : Mon Sep 10 15:20:30 2020

 Number of access : 5

② Scenario using JSTL:

In a web application, you have a requirement to display a list of product with varying categories. The product list needs to be dynamically categorised into separate section on a web page.

Solution using JSTL:

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
//import java.util.List;
import java.util.Map;
import java.util.HashMap;

Public class
```

```
Product List Servlet extends HttpServlet {
    Protected void
doGet (HttpServletRequest request, HttpServletResponse
    Product C"Laptop", "Electronics", 1000, true);
New
    Product C"Shirt", "Clothing", 30, true);
New
Product C"Washing machine", "Home appliances", False);
Product C"Clothing", "clothing", 30, true);
    }
Product "List.JSP"): forward request, response
    }
}
```

Output,

<Storage Laptop Strong> - $1000 - Available.
<Clothing> Shirt - $30 - Available.
<Home Appliance> Washing machine - 0 + out of stock

(3)

To achieve this scroll the Html Page, add Javascript for Automatic refresh and Confirm Dialog.

Html and Java Script Code:

```
<!DocType html>
<html lang="en">
<head>
    <meta charset: "UTF-8">
    <Meta name = "view Port"
Content = "width = device-width, initial-
                Scale = 1.0">
<title> Stock market Quotes </title>
```

```
<script>
    var use response confirm("the page will refresh
in 20 seconds. Do you need more time?");
    if (!user Response) {
      window.location.reload();
    } else {
      setInterval(refresh page, 30000);
    }, 20000);
  window.location.reload();
  }, refresh Interval);
</h1> Stock market Quotes </h1>
<p> Here you can display real-time stock market quotes </p>
</body>
</Html>
```

Output;

Page Display:

The page will refresh in 20 seconds.

Do you need more time?

[Cancel] [OK]

(4) Steps to Integrate a Payment Gateway Using WSDL

For Java;

```
Import. your. Package. Name. Payment Service.;
import. Package. Name. Payment Service Port type;
Public class Payment client {
  Public static void main (String [] args []) {
Payment Service = Service = New Payment Service ();
String response = Port. Process Payment
("amount", Currency, "Payment Details");
```

System.out.println("Response: " + response);
}
}

Output:
Payment Response: Payment Successful for amount 1000.0

ASSIGNMENT - 4

① <Context>
<Resource name = "jdbc/mydatasource"
auth="Container"
type = "javax.sql.DataSource"
maxTotal = "20"
maxIdle = "10"
maxWaitMillis = "10000"
username = "dbuser"
password = "dbpassword"
driverClassName = "com.mysql.cj.jdbc.Driver"
url = "jdbc:mysql://localhost:3306/mydatabase"/>
</Context>

Callable Statement:
Import java.sql.CallableStatement;
Import java.sql.Connection;
Import java.sql.SQLException;
Import java.sql.Types;
Public class CallableStatementExample5
Public void CallStoredProcedure() throws Employees {
String sql = "{call getEmployeesName
(?,?)}";

```
try(Connection conn = DriverManager.getConnection(
        Connection());
    Employee name = (String) in.getString();
System.out.println("Employee Name: " + EmployeeName);
} catch (SQLException e){
    e.printStackTrace();
}}}
```

Output:

Employee ID = 1, Name: John
Employee ID = 2. Name: Jane
Employee ID: 3: Name: Emily
Rows Updated: 1
Employee name: John

(2) Life cycle of phases of a JSP Page:

(1) Translation phase:

Description: The JSP page is translated into a Java Servlet by the JSP Engine.

Significance: This phase ensures that the JSP content is converted into a form that the Java Servlet container can execute.

(2) Compilation phase:

Description: The Java source code generated from the translation phase.

Significance: Compilation ensures that the JSP is converted into executable.

③ Initialization Phase:
Description: The Servlet Container initializes the Servlet instance.
Significance: Initialization sets up any resource the JSP might need such db.

④ Requesting Processing Phase:
Description: The Servlet Process incoming client request by calling the Service()
Significance: This phase is where the dynamic content generation occurs.

⑤ Destroy Phase:
Description: The Servlet Container destroys the Servlet instance the destroy().
Significance: The phase ensure that resource are Properly released.

Qn. no: 3:
PHP Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="View Port"
Content: "width=device-width-initial-Scale
<title>Chessboard</title>
<style>
```

```
table {
    border-collapse: collapse;
    width: 240px;
    height: 240px;
}
td {
    width: 30px;
    height: 30px;
}
for (row=0; $row<8; $row++) {
    echo "<tr>";
    for (col=0; $col<8; $col++) {
        echo "</tr>";
    }
</table>
</body>
</html>
```

Output:

[W] [B] [W] [B] [W] [B] [W] [B]
[B] [W] [B] [W] [B] [W] [B] [W]
[W] [B] [W] [B] [W] [B] [W] [B]
[B] [W] [B] [W] [B] [W] [B] [W]
[W] [B] [W] [B] [W] [B] [W] [B]
[B] [W] [B] [W] [B] [W] [B] [W]
[W] [B] [W] [B] [W] [B] [W] [B]
[B] [W] [B] [W] [B] [W] [B] [W]

* W Represent a White Cell.
* B Represent a Black Cell.

Qn. no 4. PHP Code for the Application.

```
<?php
$textFilePath = "input.txt";
$textFilePath = "output.xml";
$textContent = file_get_contents($textFilePath);
$emailPattern = '/[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}/i';
$phonePattern = '/\b\d{3}[-.]?\d{3}[-.]?\d{4}\b/';
preg_match_all($emailPattern, $textContent, $emails);
preg_match_all($phonePattern, $textContent, $phones);
$xml = new
SimpleXMLElement("<Data/>");
$emailsElement = $xml->addChild("Emails");
Sample xmlElement("<DATA/>");
$phoneElement = $xml->addChild("phone number");
Echo "Data extracted and Saved to xml
File Successfully";
?>
```

Output:

Email: Support@example.com.

Phone Number: 1234567890.

ASSIGNMENT-3

1. Tracking session Data with Java servlet

Code Implementation (8 marks)	
Session Data Accuracy (5 marks)	
Efficiency and Clarity (3 marks)	
Explanation (4 marks)	

(2) Using JSTL for Complex problem Solving (20 marks)

Scenario Explanation (6 marks):	
Function Library Explanation (5 marks):	
Custom functions (5 marks)	
Clarity and organization (4 marks)	

(3) Stock market Page Auto-refresh with JavaScript (20 marks)

Script functioning (8 marks)	
User Instruction Design (3 marks):	
Code Efficiency (4 marks)	
Explanation (3 marks)	

(4) Integrating a Payment Gateway Using WSDL (20 marks)

Understanding of WSDL (6 marks)	
Client Code Generation (6 marks)	
Error handling (4 marks)	
Clarity and Depth (4 marks)	

Assignment - 4

① JDBC in Database - Driven Application (20 marks)

Explanation of JDBC (5 marks)	
Connection Pooling (5 marks)	
Sql Queries (5 marks)	
Statement types (5 marks).	

② JSP Page Lifecycle phases (20 marks)

Life cycle Phases Explanation (5 marks):	
Embedding Java Code (5 marks):	
Advantages and Disadvantages (5 marks):	
Clarity and Depth (5 marks).	

③ PHP Chessboard Program (30 marks):

Code Implementation (8 marks)	
HTML Table Structures (5 marks)	
Alternating colors Logic. (5 marks)	
Explanation (3 marks).	

④ PHP Application with XML and Regex Exp (20 marks)

Code Implementation (8 marks)	
Pattern Extraction (5 marks)	
XML file Generation (5 marks)	
DTD VS XML schema Comparison (3 marks)	