



SAVEETHA SCHOOL OF ENGINEERING
SIMATS, CHENNAI

SET -2

| | | | |
|---|---|----|-----|
| 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 web development practices and enhance the functionality of web forms. | 10 | CO1 |
| 2 | Designing a user registration form for a new web application. The form needs to capture essential information such as name, email, and a message from users. It is crucial to implement client-side validation to ensure that the data entered is accurate and complete before the form is submitted. | 10 | CO1 |
| 3 | Designing a website that needs to function effectively across various devices and screen sizes. The design should include different types of layouts 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 maintains a consistent user experience. | 10 | CO1 |
| 4 | To create a webpage that offers a seamless user experience across devices, responsive design principles must be applied. This involves using fluid grids, flexible images, and media queries to ensure the layout adjusts gracefully to different screen sizes. For instance, on a desktop, the webpage might display multiple columns with detailed content, while on a smartphone, the same 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 fast loading times and a smooth experience on all devices. | 10 | CO1 |
| 5 | Given a scenario (e.g., creating a blog post, a product listing), 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 requests and responses that occur when a user accesses a webpage containing multiple resources (HTML, CSS, JavaScript, images). | 10 | CO1 |

Assignment - 1.

2.2.2024/6.23.

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 new version improve web development Practices and enhance the functionality of web forms.

HTML:-

```
<!DOCTYPE html>
```

```
<html xmlns = "http://www.w3.org/1999/xhtml">
```

```
<head>
```

```
<title> legacy web Application </title>
```

```
<meta http-equiv = "content-type" content = "text/html charset =  
"UTF-8"/>
```

```
<style type = "text/css">
```

```
• header & background - colour: #f8f8f8; padding: 10px; }
```

```
• content & padding: 10px; }
```

```
• footer & background - colour: #f8f8f8; padding: 10px; text-align:  
center; }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class = "header">
```

```
<h1> welcome to our website </h1>
```

```
</div>
```

```
</div>
```

```
<div class = "content">
```



```

<form action=" /submit" method="post">
  <label> name: </label>
  <input type="text" name="name" /> </input>
  <label> Email: </label>
  </header>
  <main>
    <form action=" /submit" method="post">
      <label for="name">name: </label>
      <label for="birthdate">Birthdate: </label>
      <input type="text" id="birthdate" name="birthdate">
      <input type="submit" value="submit">
    </form>
  </main>
  <footer>
  </footer>
</body>
</html>

```

2. Designing a user registration form for a new web application the form needs to capture essential information such as name email and a message from users. It is crucial to implement client side validation to ensure that the layout adapts well to complete before from the submit:-

HTML:-

```

<!DOCTYPE html>
<html lang="en">
  <head>

```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width
initial-scale=1.0">
<title>user Registration</title>
```

```
<style>
```

```
body {
```

```
font-family: Arial, sans-serif;
```

```
margin: 0;
```

```
padding: 20px;
```

```
background-color: #fufufy;
```

```
} form {
```

```
background-color: #fff
```

```
padding: 20px;
```

```
border-radius: 5px;
```

```
max-width: 400px;
```

```
margin: 0 auto;
```

```
box-shadow: 0 0 10px rgba(0,0,0,0.1);
```

```
{
```

```
label {
```

```
display: block;
```

```
margin-bottom: 5px;
```

```
font-weight: bold;
```

```
{
```

```
input { font-family:
```

```
width: 100%;
```

```
padding: 10px;
```

margin-bottom: 15px;

border-radius: 4px;

font-size: 16px;

}

in <button type="submit">

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

transition: background-color 350ms ease;

}

</style>

</head>

<body>

<form action="/submit" method="post" novalidate>

<h2>user Registration</h2>

<label for="name">name:</label>

<input type="email" id="email" name="Email" required

placeholder="example@example.com">

minlength="10" maxlength="500"

placeholder="Enter your message here...">

<input type="submit" value="Register">

</form>

</script>

</body>

</html>

Output:-

Registration form

Name : _____

Email : _____

Message : _____

- 3) Designing a website that needs to function effectively across various devices and screen sizes. The design should include different types of layouts 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 maintains a consistent user experience.

Implementation using CSS:-

1. Fixed Layout:-

- * Define a container with a fixed width, typically using 'px'.
- * Use CSS Properties like 'width', 'height', and 'margin' to align and size elements within the fixed width.

```
• fixed-container {  
    width: 1200px;  
    margin: 0 auto;  
}
```

Fluid layout:-

- * Use Percentage-based widths for elements, ensuring they expand or contract relative to their Parent Container.

```
• fluid-container {  
    width: 90%;  
    margin: 0 auto;  
}
```


3. Responsive layout with flexbox:-

* utilize flexbox to create a flexible grid system that adjust based on screen size.

- flex - container {

- display: flex;

- flex-wrap: wrap;

- flex - item {

- flex: 1 1 200px;

- @media (max-width: 768px) {

- flex - item {

- flex: 1 1 100%;

output:-

fixed layout

fluid layout

Responsive 1

Responsive 2

Responsive 3

Responsive 4

4. Responsive layout with Grid:-

* use CSS grid to create a more complex layout that can adjust to different screen sizes.

- grid - container {

- display: grid;

- grid-template-columns: repeat(3, 1fr);

- gap: 20px;

- @media (max-width: 768px) {

- grid - container {

- grid-template-columns: 1fr;

Assignment - 2.

4. To create a webpage that offers a seamless user experience across devices, responsive design Principles must be applied. For instance, on a desktop the webpage might display multiple columns with detailed content, while on a smartphone, the same content could stack vertically for easy scrolling. Additionally, touch-friendly elements and optimized images ensure fast loading times and a smooth experience on all devices.

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width,
```

```
initial-scale=1.0">
```

```
<title> Responsive webpage </title>
```

```
<style>
```

```
  .container {
```

```
    display: grid;
```

```
    grid-template-columns: repeat(3, 1fr);
```

```
    gap: 20px;
```

```
    max-width: 1200px;
```

```
    margin: 0 auto;
```

```
  }
```

```
  .item {
```

```
    padding: 20px;
```

```
    background-color: #f0f0f0;
```

```

img {
  max-width: 100%;
  height: auto;
}
@media (max-width: 768px) {
  .container {
    grid-template-columns: 1fr;
  }
  nav ul {
    display: none;
  }
  nav.hamburger > <div>
<ul>
  <li><a href="#" Home </a></li>
  <li><a href="#" About </a></li>
  <li><a href="#" Contact </a></li>
</ul>
</nav>
<div class="container">
  <div class="item">Content Block 1</div>
  <div class="item">Content Block 2</div>
  <div class="item">Content Block 3</div>
</div>
</body>
</html>

```

Output:-

Responsive webpage.

Home about service contact

Column 1
content goes here.

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 requests and responses that occur when a user accesses a webpage containing multiple resources (HTML, CSS, JavaScript, Images).

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device
```

```
width, initial-scale=1.0">
```

```
<title> Blog Post with Product Listing Title
```

```
<style>
```

```
body {
```

```
font-family: Arial, sans-serif;
```

```
line-height: 1.6;
```

```
{
```

```
h1 {
```

```
color: #333;
```

```
{
```

```
.content {
```

```
width: 90%;
```

```
max-width: 800px;
```

```
margin: 20px auto;
```

• Product-list {

list-style: none;

padding: 0;

}
• Product-list li {

border: 1px solid #ccc;

margin: 10px 0;

padding: 10px;

display: flex;

align-items: center;

}
• Product-list img {

max-width: 150px;

margin-right: 20px;

}

</style>

</head>

<body>

<div class="content">

<h1>my latest Blog Post</h1>

<p>welcome to my blog! Here are some of the

products I've been using lately:</p>

<ul class="Product-list">

Output:-

my Awesome blog.

- Home.
- About.
- Blog.
- Contact.


```

<h2> Product 1 </h2>
<p> A great Product that I highly recommend! </p>
</div>
<h2>
<h2>

</div>
<h2> Product 2 </h2>
<p> This Product has changed my life! </p>
</div>
<h2>
<h2>
<h2>
</div>
<body>
</html>

```

The title of my blog Post.

Published on Aug 27, 2024 by Author Name a
 disruptive ~~image~~ related to the blog post this
 is the introduction to my blog Post.

Subheading 1:-

Here is some detailed content.

subheading 2.

More detailed about content.

comparison table:

| feature | option A | option B. |
|-------------|----------|---------------|
| Price | \$100 | \$150 |
| Performance | Good | excellent |
| Support | 24/7 | Businesshours |

subheading 3:

- Key Point 1
- Key Point 2.
- Key Point 3.

Conclusion: summarize the Blog Post.

© 2024 my awesome Blog. All rights reserved.

| Rubrics | split up | Marks obtained | Total Marks |
|------------------------|----------|----------------|-------------|
| Form Design. | | | |
| user interface design. | | | |
| features of HTML | | | |
| form design | | | |
| validation | | | |
| layout look | | | |
| layout design | | | |
| CSS background | | | |
| form design. | | | |
| layout design | | | |
| navigation | | | |
| form design. | | | |
| CSS | | | |
| layout design | | | |
| Java script. | | | |



Course Code /Title: CSA4399 – Internet Programming
Programme : Computer Science and Engineering

ASSIGNMENT 3 QUESTIONS

SET -5

| S.No | Questions | Marks | CO | BTL |
|------|---|-------|-----|-----|
| 1 | Implementing a feature in a web application that tracks the number of accesses by a client within a single session. You need to use Java Servlets to manage and monitor session data. The application should count how many times the client accesses the application during their session and retrieve information about the session, such as the session ID, creation time, and last accessed time. | 20 | CO4 | 3 |
| 2 | Write a scenario where you had to use JSTL to solve a complex problem and how you went about it. Also, elaborate the function library in JSTL and how to create custom functions. | 20 | CO4 | 2 |
| 3 | A page of stock market quotes uses script to refresh the page every five minutes in order to ensure the latest statistics remain available. 20 seconds before the five minute period expires, a confirm dialog appears asking if the user needs more time before the page refreshes. This allows the user to be aware of the impending refresh and to avoid it if desired. | 20 | CO5 | 3 |
| 4 | You are developing an e-commerce application that needs to integrate with an external payment gateway service. This service is described using a WSDL file. How would you use the WSDL file to integrate the payment gateway service into your e-commerce application? Describe the steps involved in generating the client code, invoking the service, and handling any potential errors | 20 | CO6 | 3 |

Assignment-3.

- ① Implementing a feature in a web application that tracks the number of accesses by a client within a single session using Java servlets using HTTP session object to manage and monitor session data.

servlet code:-

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
@WebServlet("/sessionTracker")
```

```
public class sessionTracker extends HttpServlet {
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
```

```
        ServletException, IOException {
```

```
        response.setContentType("text/html");
```

```
        PrintWriter out = response.getWriter();
```

```
        HttpSession session = request.getSession(true);
```

```
        String sessionId = session.getId();
```

```
        long creationTime = session.getCreationTime();
```

```
        long lastAccessedTime = session.getLast
```

```
            AccessedTime();
```

```
        Integer visitCount = (Integer) session.getAttribute(
            "visitCount");
```

```
        if (visitCount == null) {
```

```

    visitCount++;
    session.setAttribute("visitCount", visitCount);
    out.println("<html><body>");
    out.println("    <html>session Tracking Example</html>");
    out.println("    <p>sessionID: " + sessionId + "</p>");
    out.println("    <p>session created: " + new Date().toString() + "</p>");
    out.println("    <p>last accessed: " + new Date().toString() + "</p>");
    out.println("    <p>no. of accesses in this session: visitCount</p>");
    out.println("</body></html>");
}
}

```

Output:-

Session Tracking Example.

Session ID: 12345ABCD

session created: Mon Sep 09 12:00:00 IST 2024

last accessed: Mon Sep 09 12:01:05 IST 2024

No. of accesses in this session: 1

- Q) Write a scenario where you had to use JSTL to solve a complex problem and how you went about it. Also elaborate the function library in JSTL and how to create custom functions.

JSP code using JSTL.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
```

```
<html>
```

```
<head>
```

```
<title>order management</title>
```

```
</head>
```

```
<body>
```

```
<h2>orderlist</h2>
```

```
<form method="GET" action="order.jsp">
```

```
<label for="status" id="status">
```

```
<select name="status">filter by status: <labels>
```

```
<option value="All">All</option>
```

```
<option value="Delivered">Delivered</option>
```

```
<option value="pending">pending</option>
```

```
</select>
```

```
<input type="submit" value="Filter">
```

```
</form>
```

```
<table border="1">
```

```
<thead>
```

```
<tr>
```

```
<th>orderID</th>
```

```
<th>date</th>
```

```
<th>status</th>
```

```
<th>Amount</th>
```

```
</tr>
```

```
</thead>
```

```
</table>
```

```

<c:forEach var="order" items="${orders}">
  <c:choose>
    <c:when test="${param.status} == 'all'">
      // order = status = param.status;
    <tr>
      <td>${order.id}</td>
      <td>${order.date}</td>
      <td>${order.status}</td>
      <td>${order.amount}</td>
    </tr>
  </c:choose>
</c:forEach>
</c:when>
</c:choose>
</c:forEach>
</tbody>
</table>
</body>
</html>

```

Output:-

Order list.

| Order ID | Date | Status | Amount |
|----------|------------|---------|--------|
| 1002 | 2024-09-08 | Pending | 150.00 |
| 1003 | 2024-09-09 | Pending | 300.00 |

JSTL Function Library (fn)

④ Creating Custom Functions in JSTL

```

<taglib xmlns="http://java.sun.com/xml/ns/javac"
  version="2.1">
  <taglib-version>1.0</taglib-version>

```



```

<short-name>custom</short-name>
<uri>http://example.com/custom</uri>
<function>
  <name>reverseString</name>
  <function-class>com.example.customFunction1Fun-
    ction-class>
  <function-signature>java.lang.String
    reverseString(java.lang.String)
  </function-signature>
</function>
</taglib>

```

Output:-

Custom-function Example

Original: HelloWorld

Reversed: dlrow olleH

- ③ To implement the described functionality for refreshing a stock market quotes page every five minutes, with a confirmation dialog appearing 20 seconds before the refresh

Javascript code snippet:

```

<!DOCTYPE html> -
<html lang="en"> -
<head>
  <meta charset="UTF-8"> -
  <title>stock market quotes</title> -
  <script>
    function refreshPage() {
      location.reload();
    }
  
```

Page Refresh.

(The Page reloads, displaying updated content).

Stack Mailed Quotes.

- Apple Inc. (APPL) : \$152.00
- Microsoft Corp. (MSET) : \$250.00 -
- Alphabet Inc. (GOOGL) : \$2850.00

4. To integrate an external Payment gateway service into your e-commerce application using a WSDL file.

① Generate client code from WSDL.

wsimport -keep-s src-d bin-p com.Example.Payment-
verbose http://

Example.com/paymentgateway?wsdl).

② Integrate Generate code into Application.

- Include generated code.
- Configure service Endpoint.

③ Invoke the Payment Service.

• create service instance -
PaymentService service = new PaymentService();
PaymentPortType port = service.getPaymentPort();

• Invoke Methods:

PaymentResponse response = port.ProcessPayment
(paymentRequest);


```

set Timeout(c) = 20;
const confirmRefresh = confirm("The Page will refresh  
in 20 seconds.");
if (!confirmRefresh) {
    refreshPage();
} else {
    alert("Page refresh canceled");
}
$, 20, 0000);
</script>
<body>
    <h>stock Market Quotes</h>
</body>
</html>

```

Output:-

Page Display:-

stock Market Quotes

- Apple Inc (AAPL): \$1500.00
- Microsoft Corp. (MSFT): \$250.00
- Alphabet Inc (GOOGL): \$2800.00

Confirmation Dialog

The Page will refresh in 20 seconds

ok cancel

Alert:

Page refresh canceled

ok

④ Handle Response and Errors

- check Response.

```
if (response.is successful) {  
    // Handle successful Payment  
}
```

```
} else {  
    // Handle Payment failure  
}
```

- Exception Handling.

```
try {  
    PaymentResponse response = port.ProcessPayment  
        (PaymentRequest);  
}
```

```
} catch (SOAPFaultException e) {  
    // Handle SOAP faults (eg, invalid request).  
}
```

```
} catch (WebServiceException e) {  
    // Handle connectivity or configuration errors  
}
```

Output:-

successful Payment:-

Payment successful. Transaction ID: 1987653210.

Payment failure (eg: Invalid Card Details):

Payment failed due to a SOAP fault: Invalid credit Card Details.

SOAP Fault (eg: Invalid Request)

Payment failed due to a SOAP fault: Invalid Request format.

connectivity Issue (eg: service unavailable)

Payment failed due to a connectivity issue.
cannot connect to the Payment gateway.

| Assignment Rubrics | splitup | Marks obtained | Total marks |
|--------------------|--|----------------|-------------|
| Question-1 | Code implementation 8M Session Data 5M Accuracy Efficiency and clarity 3M Explanation 4M | | |
| Question-2 | Scenario Explanation 6M Function Library Explanation 5M custom Functions 5M Clarity and organization 4M | | |
| Question-3 | Script functionality dialog 8M user Interaction Design refresh 5M code efficiency 4M Explanation 3M | | |
| Question-4 | understand script Function: understanding of WSDL 6M client code Generation 6M error Handling 4M Clarity and Depth 4M | | |



Course Code /Title: CSA4399 – Internet Programming
Programme : Computer Science and Engineering

ASSIGNMENT 4 QUESTIONS

| S.No | Questions | Marks | CO | BTL |
|------|--|-------|-----|-----|
| 1 | From a developer's perspective, discuss why JDBC is essential in building database-driven applications. How to achieve JDBC CONNECTION pooling using JDBC Data Source and JNDI in Apache Tomcat Server. Provide examples of executing SQL queries using JDBC statements. Discuss the differences between Statement, Prepared Statement, and Callable Statement. | 20 | CO5 | 3 |
| 2 | Describe the lifecycle phases of a JSP page. Explain the significance of each phase in the JSP execution process. Discuss the different ways to embed Java code within a JSP page with examples. Explain the advantages and disadvantages of using scriptlets, declarations, and expressions in JSP. | 20 | CO4 | 2 |
| 3 | You need to develop a PHP program that generates a chessboard using HTML tables. The table should have a total width of 400px, and each cell should have a height and width of 30px. The chessboard should alternate colors between black and white for each cell to represent a typical chessboard layout. How would you write a PHP program using nested for loops to create a chessboard? The chessboard should be displayed using an HTML table with a total width of 400px, and each cell should have a height and width of 30px. Explain how you would use the nested for loops to alternate the cell colors and ensure the chessboard pattern is correctly displayed. Provide the code for this program | 20 | CO4 | 2 |
| | You are developing a PHP application that reads content from a text file and uses regular expressions to extract specific patterns, such as email addresses and phone numbers. After extracting the data, the application should store the results in a new XML file following a defined schema for the data. Additionally, you need to compare and contrast DTD (Document Type Definition) and XML Schema for defining the XML structure. How would you create a PHP application that reads content from a text file and uses regular expressions to extract specific patterns (e.g., | 20 | CO5 | 2 |



Assignment-4.

Why JDBC is Essential in Building Database-Driven Applications.

JDBC is Essential because it provides a standard API for Java applications to interact with databases.

Achieving JDBC Connection Pooling using JDBC Data source and JNDI in Apache Tomcat.

Configure DataSource

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

Look up DataSource in Java code using Java.

```
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;
import java.sql.Connection;

public class DatabaseUtil {
    public static Connection getConnection()
        throws Exception {
```

```

content initContent = new InitialContent();
DataSource ds = (DataSource) initContent;
lookup("java:comp/env/jdbc/myDB");
return ds.getConnection();
}
}

```

Executing SQL Queries using JDBC statement.

① Using a statement.

```

try (Connection conn = DatabaseUtil.getConnection();
     Statement stmt = conn.createStatement()) {
    String query = "SELECT * FROM users";
    ResultSet rs = stmt.executeQuery(query);
    while (rs.next()) {
        System.out.println("user ID: " + rs.getInt("id") + " +
        name: " + rs.getString("name"));
    }
}

```

② using a Prepared statement.

```

try (Connection conn = DatabaseUtil.getConnection();
     PreparedStatement pstmt = conn.prepareStatement(
        "SELECT * FROM users WHERE id = ?")) {
    pstmt.setInt(1, 1);
    ResultSet rs = pstmt.executeQuery();
    while (rs.next()) {
        System.out.println("user ID: " + rs.getInt(
            "id") + ", name: " + rs.getString("name"));
    }
}

```


3) using a Callable statement for stored Procedures.

```
try (Connection conn = DatabaseUtil.getConnection();  
    CallableStatement cstmt = conn.prepareCall("{  
    • Call getUsersById(2)}")) {
```

```
    cstmt.setInt(1, 1);
```

```
    ResultSet rs = cstmt.executeQuery();
```

```
    while (rs.next()) {
```

```
        System.out.println("user ID: " + rs.getInt("user ID")  
        + ", name: " + rs.getString("name"));
```

```
    }
```

Output:-

Statement Example output:-

user ID: 1, name: Vishnu

user ID: 2, name: Priya

Prepared statement.

user ID: -1, name, Vishnu.

Callable statement.

user ID: -1, name: Vishnu.

2) Lifecycle Phases of a JSP Page.

① Translation Phase

② Compilation Phase

③ Initialization Phase

④ Request Processing Phase

⑤ Destruction Phase

Embedding Java code in JSP.

① Scriptlets.

```
<% int sum = 5 + 10; %>
```

```
<%> The sum is: <% = sum; %>
```

Output:-

The sum is : 15.

② Declarations:-

```
<%: int add (int a, int b) { return a+b; } %>
```

```
<%> The result is : <% = add (3, 7) %>
```

Output:-

The result is : 10.

③ Expressions

```
<%> current time : <% = new java.util.Date() %>
```

Output:-

current time: Mon Sep 09 09:30:00 PDT 2024

Scriptlets:-

Advantages:- Easy to use for embedding simple

Java logic.

Disadvantages:- leads to messy code, difficult to maintain.

Declarations:-

Advantages:- useful for declaring reusable methods and variables across multiple requests.

① Disadvantages:- Can clutter JSP with Java code, leading to poor separation of concerns.
Expressions:-

Advantages. Simplifies outputting dynamic content directly in JSP.

Disadvantages: limited to Expressions.

③ Generates a chessboard using HTML tables with width of 400px (total) and each cell has a height and width of 30px.

PHP code:-

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>chessboard </title>
```

```
<style>
```

```
table {
```

```
width: 400px;
```

```
border-collapse: collapse;
```

```
}
```

```
td {
```

```
width: 30px;
```

```
height: 30px;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```

<table>
<tr>
  // Loop for 8 rows
  for ($row=0; $row<8; $row++){
    echo "<tr>";
    for ($col=0; $col<8; $col++){
      if (($row+$col)%2 == 0){
        echo "<td style='background-color: white;'><
          1+1>";
      }
    }
  }

```

```

false;

```

```

echo "<td style='background-color: black;'><td>";

```

```

echo "</tr>";

```

```

2>

```

```

</table>

```

```

</body>

```

```

</html>

```

Output:-

```

[ ] [#] [ ] [#] [#] [ ] [#]
[#] [ ] [#] [ ] [#] [ ] [#] [ ]
[ ] [#] [ ] [#] [ ] [#] [ ] [#]
[#] [ ] [#] [ ] [#] [ ] [#] [ ]
[ ] [#] [ ] [#] [ ] [#] [ ] [#]
[#] [ ] [#] [ ] [#] [ ] [#] [ ]

```


PHP Application to Extract Data and store in XML.

Steps:-

- ① Read Content from a Text file
- ② Extract Patterns using Regular.
- ③ create and store Results in an XML file
- ④ Define XML Schema.

PHP code:-

<?php

```
$filename = "input.txt";
```

```
$content = file_get_contents($filename);
```

```
$preg = match_all (". [a-zA-Z0-9-_.+]+@[a-zA-Z0-9-]+
```

```
1. [a-zA-Z]{2}"; $content, $email(s));
```

```
$preg = match_all (". !+2[0-9] {1,3} [-. \ ] [0-9]
```

```
{1,4} [-. \ ] ? [0-9] {3,4} / 4"; $content,  
$phones;
```

```
$xml = new simpleXMLElement ('<data>|data>');
```

```
$emailElement = $xml->addChild ('emails');
```

```
foreach ($email(s) as $email) {
```

```
$emailElement->addChild ("email, $email);
```

```
$phoneElement = $xml->addChild ('phones');
```

```
for each ($phones(s) as $phone) {
```

```
$phoneElement->addChild ('phone, $phone);
```

```
}
```

```
$xml->asXML ('output.xml');
```

```
echo "Data extracted and saved to output.xml";
```

Output:-

<data>

<emails>

<email> example@example.com </email>

<email> example@example.com </email>

</emails>

<phones>

<phone>+123-456-7890 </phone>

<phone>987-654-3210 </phone>

</phones>

</data>

| Assigned Rubrics | split up | marks obtained | Total Marks |
|------------------|--|----------------|-------------|
| Question 1. | <p>JDBC: Explanation of JDBC 5M</p> <p>Connection Pooling 6M</p> <p>SQL Queries 5M</p> <p>Statement Types. 4M</p> | | |
| Question 2. | <p>Life Cycle Phases Explanation 6M</p> <p>Embedding Java code. 5M</p> <p>Advantages & Disadvantages 5M</p> <p>Clarity & Depth. 4M</p> | | |
| Question 3. | <p>Code Implementation 8M</p> <p>HTML Table Structure 5M</p> <p>Alternating colors logic. 4M</p> <p>Explanation. 5M</p> | | |
| Question 4. | <p>Code Implementation 8M</p> <p>Pattern Extraction 5M</p> <p>XML File Generation 4M</p> <p>DTD vs XML Schema comparison. 8M</p> | | |