| Assignment Rubics | Marks split up | Marks Score | Total Marks |
|---|---|---|---|
| Question-1 | Code Implementation 8M | | |
| | Session Data Accuracy 5M | | |
| | Efficiency & Clarity 3M | | |
| | Explanation 4M | | |
| Question-2 | Scenario Explanation 6M | | |
| | Function Library explanation 5M | | |
| | Custom functions 5M | | |
| | Clarity & organization 4M | | |
| Question-3 | Script functi- 8M | | |
| | User Interaction design 5M | | |
| | Code Efficiency 4M | | |
| | Explanation 3M | | |
| Question-4 | understanding of WSDL 6M | | |
| | Client code Generation 6M | | |
| | Error Handling 4M | | |
| | Clarity & Depth 4M | | |

① Implementi

-tracks the
a style
http sess
data.

Servlet

import j
import
import
import
import
import
import
import
import
import
@ web
publi

# 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 HttpSession 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.get
                             Attribute("visitCount");
```

```
if (visitCount == null){
    visitCount = 0;
}
visitCount++;
session.setAttribute("visitCount", visitCount);
out.println("<html><body>");
out.println("<h1>Session Tracking Example</h1>");
out.println("<p>Session ID: " + session.getId() + "</p>");
out.println("<p>Session Created: " + new Date
                (session.getCreationTime()) + "</p>");
out.println("<p>Last Accessed: " + new Date
                (session.getLastAccessedTime()) + "</p>");
out.println("<p>No. of accesses in this session:
                visitCount + "</p>");
out.println("</body></html>");
}
```

Output:

Session Tracking Example:

Session ID: 12345ABCDE

Session Created: Mon Sep 09 12:00:00 IST 2024.

Last Accessed: Mon Sep 09 12:01:05 IST 2024.

No. of access in this session: 1

(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.

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> order List </h2>
    <form method = "GET" action = "order.jsp">
        <label for = "status"> filter by status: </la
        <select name = "status" id = "status">
            <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">
        <thread>
            <tr>
                <th> order ID </th>
                <th> Date </th>
                <th> Status </th>
                <th> Amount </th>
            </tr>
        </thread>
```

```
<body>
    <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:when>
        </c:choose>
    </c:forEach>
</tbody>
</table>
</body>
</html>
```

Output:

order List

| Order ID | Date | Status | Amount |
|---|---|---|---|
| 1003 | 2024-08-09 | Pending | 150.00 |
| 1004 | 2024-09-09 | Pending | 300.00 |

JSTL Function Library (fn)

* Creating custom functions in JSTL

```
<taglib xmlns = "http://java.sun.com/xml/ns/javaee"
        version = "0.1">
    <tlib-version>1.0</tlib-version>
    <short-name>custom</short-name>
    <uri>http://example.com/custom</uri>
    <function>
        <name>reverseString</name>
        <function-class>com.example.customFunctions
        </function-class>
        <function-signature>java.lang.String
        reverseString(java.lang.String)
        </function-signature>
    </function>
</taglib>
```

Output:

```
Custom Function Example
Original: Hello World
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.

Java code JavaScript code Snippet:

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

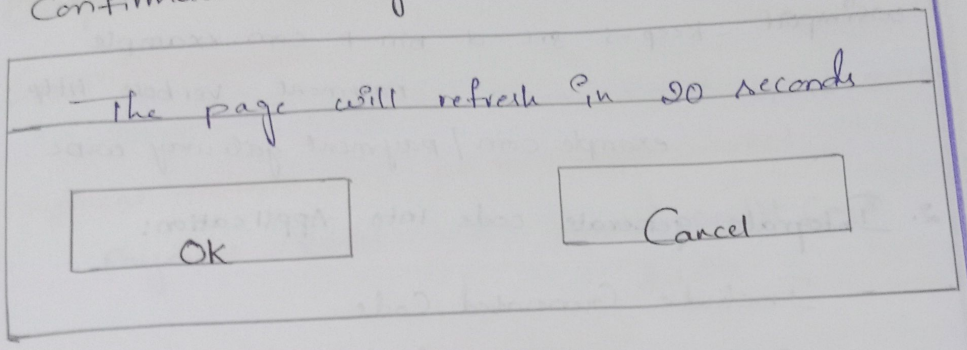
```
<meta charset = "UTF-8">
<title> stock Market Quotes </title>
<script>
    function refresh page () {
        location. reload ();
    }
    set timeout (() => {
        const confirmrefresh = confirm ("The page
            will refresh in 20 seconds");
        if (!confirm refresh) {
            refresh page ();
        } else {
            alert ("page refresh canceled");
        }
    }, 28 0000);
</script>
<body>
    <h1> Stock Market Quotes </h1>
</body>
</html>
```

**Output :**

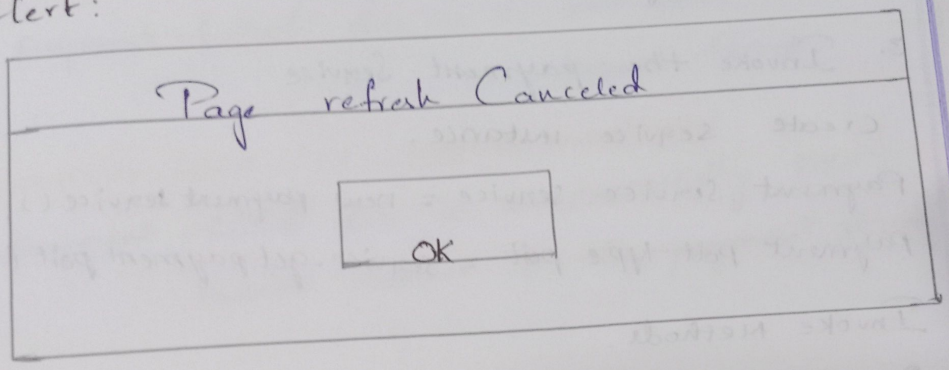
Page Display :

| Stock Market Quotes |
| --- |
| - Apple Inc. (AAPL) : $ 150.00 |
| - Microsoft Corp (MSFT) : $ 250.00 |
| - Alphabel . Inc (GOOGL) : $ 2800.00 |

Confirmation Dialog

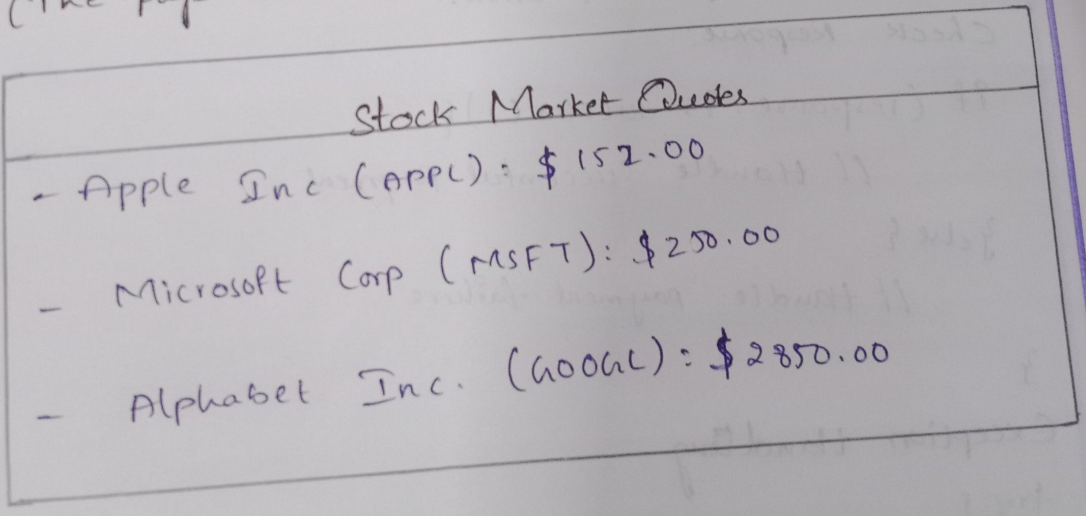



Alert:




Page Refresh

(The page reloads, displaying updated content)




(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);
```

4. Handle Response & Errors:

Check Response:

```
if (response.isSuccess()){
    // Handle Successful Payment
}else{
    // Handle payment failure
}
```

Exception Handling

```
try {
    PaymentResponse response = port.processPayment(paymentRequest);
} catch (SOAPFault exception) {
    // Handle SOAP Faults (eg: invalid request)
} catch (WebServiceException e) {
    // Handle connectivity & Configuration errors
}
```

Output :

Successful payment:

Payment successful. Transaction ID: 1987633410

Payment failure (eg: Invalid card Details):

Payment failed. Error: 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.