27/8/24 -: Assignment-01:- SET-1

1. You are tasked with creating a web page for a bike showroom. The showroom wants to clean and interactive design where customers can easily navigate through different bike brands and view the specific models and their specifications. The page should be divided into three frames using a frameset: a top row for displaying the showroom name, a left column for listing bike brands with hyperlinks and a right column for showing the corresponding bike models and specifications when a brand is selected.

```
<!DOCTYPE html>
<html>
<frameset rows="10%,*">
    <!-- Top row for showroom name -->
    <frame src="showroom_name.html" frameborder="0"
           scrolling="no">
    <frameset clos="20%,*">
        <frame src="bike-brands.html" frameborder="0"
               scrolling="auto" name="brands">
    <!-- Right column for bike models and specs -->
    <frame src="bike_models.html" frameborder="0"
           scrolling="auto" name="models">

    </frame set>
</frameset>
</html>
```

Showroom_name.html:-

```
<!DOCTYPE html>
<html>
<body>
```

```
    <h1> Bike showroom </h1>
  </body>
  </html>
```

bike_brands.html:-

```
<!DOCTYPE html>
<html>
<body>
  <ul>
    <li><a href="brand1.html" target="models">
                Brand1 </a></li>
    <li><a href="branda.html" target="models">
                Branda </a></li>
  </ul>
</body>
</html>
```

bike_models.html:-

```
<!DOCTYPE html>
<html>
<body>
  <h2> Brand1 Models </h2>
  <ul>
    <li> Model1: <span> specs: ---- </span></li>
    <li> Model2: <span> specs: ---- </span></li>
  </ul>
</body>
</html>
```

2. HTML Code for job application:-

```
<!DOCTYPE html>
<html>
<body>
```

```
<form>
    <label> full Name: </label>
    <input type :"text" required >
    < label> Highest Qualified Degree: </label>
    <select required>
        <option value: " "> Select Degree </option>
        <option value: "high_school "> Highschool
                                        </option>
        <option value :" bachelor "> Bachelor's </option>
        <option value = "Master "> Master's </option>
        <option value = "phd" > phD </option>
    <select >
    <label > Gender: </label>
    <select required >
        <option value = " " > Select Gender </option>
        <option value = "male"> Male </option>
        <option value = "female" > female </option>
        <option value = " other" > other </option>
    <select>
    <input type = "submit" value = " Submit Applications
    <input type = "reset" value = "Cancel" >
</form>
</body>
</html>
```

3 Questions:-

LAMP stack (Linux, Apache, MySQL/MariaDB, PHP (or) :-

- Linux :- The operating system that provides foundation for the stack

- Apache:- The webserver that handles requests and servers web content
- MySQL / Maria DB:- The database Management system for storing and managing data.
- PHP / Perl / Python:- Server-side languages for generating dynamic content.

WAMP stack (Windows, Apache, MySQL / MariaDB, PHP / Perl Python)

- Windows: The operating system for the stack
- Apache: Same role as in LAMP, serving web content
- MySQL / Maria DB:- Handles database operations, like in LAMP.
- PHP / pal python:- for server-site logic and interaction with databases.

Installation steps:-

LAMP stack on Linux (Ex:- Ubuntu)

1. Update System Packages
2. Install Apache
3. Install MySQL / Maria DB
4. Secure Installation
5. Install PHP
6. Restart Apache
7. Test PHP

WAMP stack on windows -

1. Download WAMP
2. Run the installer
3. Test Apache
4. Test PHP
5. visit http://localhost/test.php"
6. Manage MySQL: Use PHP admin at "http://localhost"

4 Question:-

HTTP Request and Response messages

structure and key concepts

1. Request line:
   - Method: Defines the action.
   - URI/URL: Specifies the resources.
   - HTTP version: protocol version.
2. Headers:
   - Provide metadata about the request
   - common headers:-
     - * Host
     - * User-Agent
     - * Accept
3. Black line:-
   - seperates headers from the body
4. Body:-
   - contains data sent to the server (eg. form data in a 'post' requests)

HTTP Response Message structure:-

1. status line:-
   - HTTP version
   - status code
   - Reason phrase

2. Headers:-
   - provides metadata about the Response
   - common headers
     * content type
     * content_length
     * set cookie

3. Blank line:-
   - separates headers from the body.

4. Body:-
   - contains the data sent to the client

Key HTTP Respons status codes-

1. 1xx (Informational)
2. 2xx (Success)
3. 3xx (Redireetion)
4. 4xx (client error)
5. 5xx (server errors)

Conclusion:- Understanding the structure of HTTP request and response message and managing HTTP status code is essential for building web applications.

Question-

Home page (index.html)

```
<!DOCTYPE html>
<html>
 <head>
    <title>small Business </title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
      <h1> Small Business </h1>
      <nav>
        <a href="index.html" >Home </a>
        <a href="about.html" >About </a>
        <a href="contact.html"> contact </a>
      </nav>
    </header>
    <div class="container">
        <div class="content">
            <h2> welcome </h2>
            <P> We are comitted to providing
                exceptional service and high
                quality products </P>
        </div>
      </div>
    <footer>
        <P> &copy; 2024 small business: All
            rights reserved </P>
    </footer>
</body>
</html>
```

About page (about.html):

```
<!DOCTYPE html>
<html>
<head>
    <title> Small Business </title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1> Small Business </h1>
        <nav>
            <a href="index.html"> Home </a>
            <a href="about.html"> About </a>
            <a href="contact.html"> contact </a>
        </nav>
    </header>
    <div class="container">
        <div class="content">
            <h2> About us </h2>
            <P> founded in 2024 </P>
        </div>
    </div>
    <footer>
        <P> &copy; 2024 Small Business </P>
    </footer>
</body>
</html>
```

Question -1 →

Wel co

Bike bra

Honda

Yamah

Questi

full

th

Question -1 → Output webpage.

Welcome to showroom

Bike brands	welcome to the bike show
Honda Yamaha.	Please select a brand to view the available brands and their models and specifications

Question -2:-

Job Application Form

Full name

Highest degree

select your degree

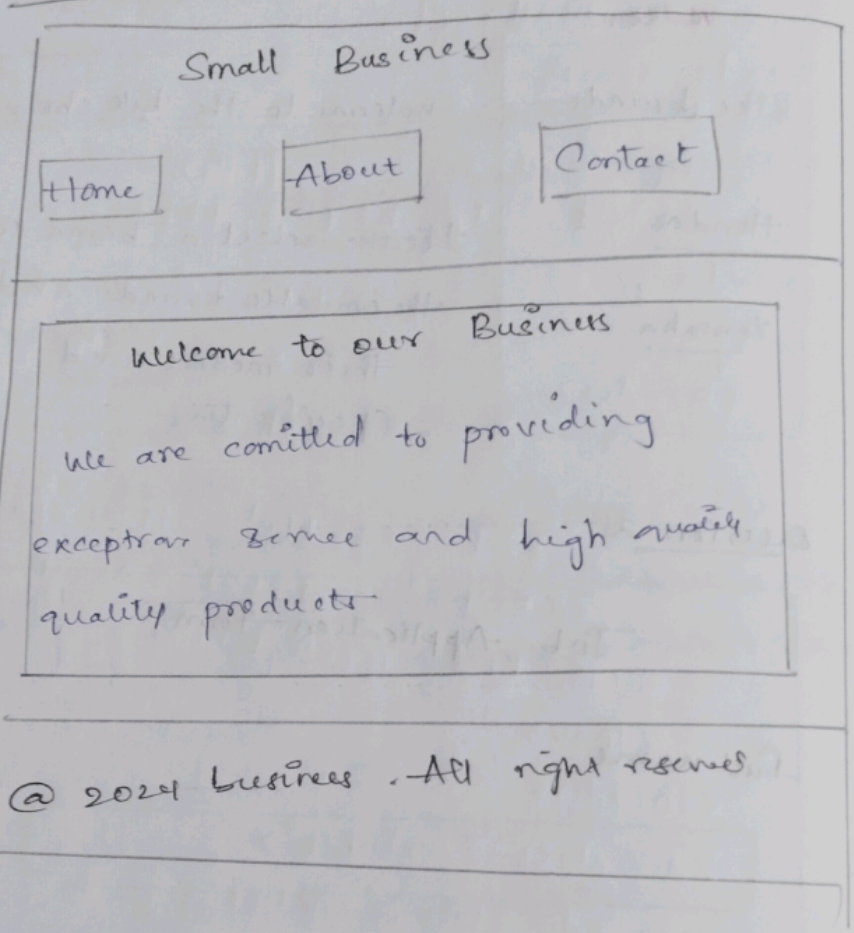
Gender

- Male
- female

Submit

Cancel

Question -5:


Small Business
Home
About
Contact
Welcome to our Business
We are comitted to providing exceptional service and high quality products
@ 2024 Business . All right reserved

Assignment Rubrics	Marks split up	Mark save	Total marks
Assignment1 Question1	form design	4	
	button design		
	layout look and field		
Question -2	form design		
	button design		
	layout look & field		
Question 3	MySQL database		
	PHP script		
	output		
Assignment 4 Question -4	HTML & PHP code		
	output		
Question 5	html & structure		
	output		

① Implement a feature in a web application that tracks the number of access 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.HTTPServletResponse;
import javax.servlet.http.HTTPSession;
@ webservlet ("/sessionTracker")
public class SessionTracker extends HTTPServlet{
    HTtp servletResponse) throws servlet exception
                    IO exception{
  response.setContentType("text/html");
  printwriter out = response.getwriter();
HttpSession session = request.getsession(true);
long creationTime = session.get(creationTime();)
long lastAccessedTime = session.getLastAccessd
                        time();
Integer visitcount = (Integer) session.get
                Attribute("visit count");
```

visit cou
session.se
out prin
out.printl
out prin
out prin

out prin

Output:-
Session
Session
sessio
Last
No. of

② Wri
To
abo
in

JSP

```
visit count++;
session.setAttribute("visit count", visit count)
out.println("<html><body>");
out.println("<h1> session Tracking Example </h1>
out.println("<p> sessionID :" + sessionId + "</p>");
out.println("<p> Last-Accessed :" + new Date
                (Last-Accessed Time) + " </p>");
out.println("</body></html>");
```

Output:-

Session Tracking Example

Session ID: 12345-ABCD

session created: Mon sep 10 10:00:00 IST 2024

Last-Accessed: Mon sep 09 10:01:07 IST 2024

No. of accesses 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>
    <h1> Order List </h1>
    <form method = "GET" action = "order.jsp">
        <label for = "status"> filter by status: </label>
        <select name = "status" id = "status">
            <option value = "Delivered"> Delivered
                </option>
            <option value = "Pending"> Pending
                </option>
        </select>
        <input page = "submit type value = "filter">
    </form>
    <table border = "1">
        <thead>
            <tr>
                <th> order ID </th>
                <th> Date </th>
                <th> status </th>
                <th> Amount </th>
            </tr>
        </thead>
    </body>
    <c:forEach var = "order" items = "${orders}">
        <c:choose>
            <c:when test = "${param.status == null ||
                order.status == param.status}">
        <tr>
```

```
<td>${order.date}</td>
<td>${order.status}</td>
<td>${order.amount}</td>
</tr>
</c:when>
<c:choose>
</c:forchoose>
</c:foreach>
</tbody>
</table>
</body>
</html>
```

Output:-

Order list:

Order ID	Date	status	Amount
10000	24/10/23	Pending	159.00
10001	25/10/23	Pending	307.00

JSTL function library (-fn)

* creating custom functions in JSTL

```
<taglibx mins = "http://java.sun.com/xml/ns/java
                                ee" version = "2.1">
  <tlib=version>1.0</tlib-version>
  <short-name> custom </short-name>
  <uri>http://example.com/custom/uri</uri>
  <function>
      <name> reverse string </name>
```

```
<function>
  <name>reverseString</name>
  <function-class>com.example.CustomFunctions</function-class>
</function>
</taglib>
```

Output:-

Custom function Example

Original: Hello world

Reversed: dlrow olleH

③ Javascript Code snippets:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Stock Market Quotes</title>
  <script>
    function refreshPage(){
      location.reload();
    }
    setTimeout(() => {
      const confirmRefresh = confirm("The
      page will refresh in 20 seconds.");
      if(confirmRefresh){
        refreshPage();
```

```
        alert("Page refresh cancelled");
    }
}, 28000);
</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. - Alphabet Inc (GOOGL): $2800.00.

Confirmation Dialog.

The page will refresh in 20 seconds
OK Cancel

Alert

Page refresh cancelled
OK

Page Refresh.

④ To integrate an external payment gate serv into your e-commerence application

① Generate client code from WSD

wsimport -keep -s src -d bin -p com.exam payment -verbose http://example.com/payment gateway?ws.

② Integrate Generate code into Application

- Include Generated Code
- Configure service endpoint

③ Invoke the payment services

- create service Instance

```
payment service service = new PaymentService();
PaymentPortType Port = service.getPaymentPort();
```

- Invoke Methode

```
paymentResponse response = port.processPayment
                (PaymentRequest);
```

④ Handle Response and ~~Goroos~~.

- check Response

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

- Except

```
try {
    paym
} catch
    //
} catch
    //
}
```

Output:

Succes

Pa

Paymen

SOAP

pay

Con

• Exception Handling

```
try{
    payment response response = (payment request);
} catch (SOAPfault exception e){
    // Handle SOAP faults eg; (Invalid request)
} catch (Web Service exception e){
    // Handle connectivity or configuration errors
}
```

Output:-

Successful Payment

Payment successful Transaction ID: 19873210.

Payment failure (e.g; 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)