05 September 2024 11:16 AM

Question 1:

a) Features of Blogging, Mashups, and Rich Internet Applications

Blogging:

- **Content Creation:** Allows users to create and publish their own content, such as articles, blog posts, and multimedia.
- **Community Building:** Enables users to connect with others who share similar interests and engage in discussions.
- **SEO Benefits:** Can improve search engine rankings through effective content creation and optimization.

Mashups:

- **Integration:** Combines data from multiple sources to create new applications or services.
- **Customization:** Allows users to personalize their online experiences by combining different elements.
- **Innovation:** Facilitates the development of innovative solutions by leveraging existing technologies and data.

Rich Internet Applications (RIAs):

- **Interactivity:** Offers a more interactive and engaging user experience compared to traditional web applications.
- **Offline Functionality:** Enables some applications to function without a constant internet connection.
- **Desktop-like Features:** Provides features similar to desktop applications, such as drag-and-drop, real-time updates, and offline data storage.

How these technologies are useful: These technologies can be used for various purposes, including:

- **Personal Expression:** Blogging allows individuals to share their thoughts, experiences, and ideas with the world.
- **Business:** Mashups can be used to create innovative business applications, while RIAs can enhance customer experiences.
- **Education:** Blogging and RIAs can be used to create educational content and interactive learning experiences.
- **Social Networking:** Mashups and RIAs can be used to create social networking platforms and tools.

b) HTML, CSS, and JavaScript Code for Online Library Membership Form

HTML Code:

HTML

```
<!DOCTYPE html><html><head><title>Online Library Membership Form</title>
<linkrel="stylesheet"href="styles.css"></head><body><h1>Online Library Membership Form</h1>
<form><labelfor="name">Name:</label><inputtype="text"id="name"name="name"><br><br>
<labelfor="aadhar">Aadhar Number:</label><inputtype="number"id="aadhar"name="aadhar"><br>
<br><label for="membershipType">Membership Type:</label>
    <select id="membershipType" name="membershipType">
      <option value="Student">Student</option>
      <option value="Faculty">Faculty</option>
      <option value="Staff">Staff</option>

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

```
      </select><br><br>

<labelfor="yearOfMembership">Year of Membership:</label>
<inputtype="number"id="yearOfMembership"name="yearOfMembership"><br><br>
<labelfor="previousMember">Were you a member earlier?</label>
<inputtype="radio"id="previousMemberYes"name="previousMember"value="Yes">Yes
      <inputtype="radio"id="previousMemberNo"name="previousMember"value="No">No<br><br>
<labelfor="description">Description of services expected:</label><br>
<textareaid="description"name="description"></textarea><br><br>
<inputtype="submit"value="Submit"></form></body></html>
```

CSS Code (styles.css):

CSS

```
h1{
   font-size: 14pt;
   font-weight: bold;
   color: red;
}
label{
   font-size: 14pt;
   font-style: italic;
}
form{
   background-color: lightgreen;
   padding: 20px;
}
```

JavaScript Code:

JavaScript

```
functionvalidateForm() {
   varname = document.getElementById("name").value;
   varaadhar = document.getElementById("aadhar").value;
   varmembershipType = document.getElementById("membershipType").value;
   varyearOfMembership = document.getElementById("yearOfMembership").value;
   vardescription = document.getElementById("description").value;
if(name == ""|| aadhar == ""|| membershipType == ""|| yearOfMembership == ""|| description == "") {
      alert("Please fill in all the required fields.");
      returnfalse;
   }
}
```

Screenshot:

online library membership form

c) Creating a Webpage with Tables and CSS

HTML Code:

HTML

```
<!DOCTYPE html><html><head><title>BCA Course List</title><style>
      table{
         border-collapse: collapse;
         width: 100%;
         background-color: lightgreen;
```

```
    }
th{
        text-align: left;
        padding: 8px;
        background-color: #4CAF50;
        color: white;
        font-size: 12pt;
        font-weight: bold;
    }
td{
        text-align: left;
        padding: 8px;
        font-size: 11pt;
        font-family: Arial;
    }
tr:nth-child(even) {
        background-color: #f2f2f2;
    }
tr:nth-child(odd) {
        background-color: #d3d3d3;
    }
div.theory-courses, div.practical-courses{
        background-color: lightyellow;
        padding: 10px;
    }
ol{
        font-family: Times New Roman;
        font-size: 11pt;
    }
  </style></head><body><h2>BCA Course List</h2><table><tr><th>Serial Number</th><th>Course
Code</th><th>Course Title</th><th>Course Credits</th><th>Course Type</th></tr><tr><td>1</td>
<td>BCA101</td><td>Introduction to Computer Science</td><td>3</td><td>Theory</td></tr></table>
<divclass="theory-courses"><h2>Theory Courses</h2><ol><li>Introduction to Computer Science</li>
</ol></div><divclass="practical-courses"><h2>Practical Courses</h2><ol><li>Programming
Fundamentals Lab</li></ol></div></body></html>
```

Screenshot:

webpage displaying the BCA course list

d) XML Document and DTD for Student Information

XML Document:

XML

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE studentsSYSTEM"students.dtd"><students>
<studentenrolmentNumber="12345"><name>John Doe</name><programme>BCA</programme>
<duration>3</duration><courses><course>BCA101</course><course>BCA102</course><course>
BCA103</course></courses></student></students>
```

DTD (students.dtd):

```
<!ELEMENT students (student+)>
<!ELEMENT student (name, programme, duration, courses)>
<!ATTLIST student enrolmentNumber CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
```

```
<!ELEMENT programme (#PCDATA)>
<!ELEMENT duration (#PCDATA)>
<!ELEMENT courses (course+)>
<!ELEMENT course (#PCDATA)>
```

e) JavaScript Code for Event Handling

```
JavaScript
functiondisplayText() {
  varwelcomeText = document.getElementById("welcomeText");
  if(welcomeText.innerHTML === "Welcome to JavaScript Event Demonstration") {
    welcomeText.innerHTML = "We just demonstrated the click Event";
  } else{
    welcomeText.innerHTML = "Welcome to JavaScript Event Demonstration";
  }
}
// Add event listener to the text elementvarwelcomeText = document.getElementById("welcomeText");
welcomeText.addEventListener("click", displayText);
```

f) WAP Model, Benefits, Limitations, and WML Elements

WAP Model: The Wireless Application Protocol (WAP) is a set of communication protocols designed for mobile devices. It allows users to access the internet and web-based services on their mobile phones. The WAP model consists of three main components:

- **WAP Gateway:** A server that acts as a bridge between the mobile device and the internet.
- **WAP Browser:** A specialized browser on the mobile device that interprets WAP content.
- **WAP Content:** Web content specifically designed for mobile devices, using the Wireless Markup Language (WML).

Benefits of WAP:

- **Accessibility:** Enables access to the internet on mobile devices.
- **Efficiency:** Optimizes content for mobile devices, reducing data usage and improving performance.
- **Standardization:** Provides a standardized platform for developers to create mobile web applications.

Limitations of WAP:

- **Limited Functionality:** Compared to traditional web browsers, WAP browsers have limited capabilities and may not support all web technologies.
- **Slower Performance:** WAP can be slower than traditional web browsing due to its reliance on wireless networks and optimized content.
- **Reduced User Experience:** WAP's limitations can result in a less rich and interactive user experience compared to desktop or modern mobile web browsers.

WML Elements:

- **Preformatted text:** tag
  - Example: This is preformatted text.
- **WML Navigational elements:** <card>, <anchor>, <go>
  - Example:
    Code snippet
    ```
    <card id="card1">
      <p>This is the first card.</p>
      <anchor href="card2.wml">Go to card 2</anchor>
    </card>
    ```
- **WML <select> element:**
  - Example:
    Code snippet

```
<select name="choice">
  <option value="option1">Option 1</option>
  <option value="option2">Option 2</option>
</select>
```

Question 2:

a) Explanations

i) Static web pages and Dynamic web pages

- **Static web pages:** These pages are created once and remain unchanged unless manually edited. They are typically HTML files that are served directly to the client's browser.
- **Dynamic web pages:** These pages are generated on-the-fly by a web server, based on user input, database queries, or other factors. They can provide a more personalized and interactive experience for users.

static vs. dynamic web pages

ii) N-Tier Architecture

N-tier architecture is a software architecture model that divides an application into multiple layers or tiers. Each tier has specific responsibilities and can be scaled independently. Common tiers include:

- **Presentation tier:** Handles user interaction and displays content.
- **Business logic tier:** Implements the application's core functionality and business rules.
- **Data access tier:** Interacts with databases and other data storage systems.

Ntier architecture

iii) Tools for server-side scripting

- **PHP:** A popular general-purpose scripting language.
- **ASP.NET:** Microsoft's web development framework.
- **Java Server Pages (JSP):** A technology for creating dynamic web content using Java.
- **Python:** A versatile programming language often used for web development.
- **Ruby on Rails:** A full-stack web development framework.

iv) HTTP primitives

HTTP primitives are the basic operations that can be performed using the HTTP protocol. They include:

- **GET:** Retrieves data from a server.
- **POST:** Sends data to a server.
- **PUT:** Updates data on a server.
- **DELETE:** Deletes data from a server.
- **HEAD:** Retrieves the header information of a resource without the body.
- **OPTIONS:** Retrieves the allowed HTTP methods for a resource.
- **TRACE:** Echoes back the request to the client.
- **CONNECT:** Establishes a tunnel to a server.

v) Web Container

A web container is a software component that provides a runtime environment for executing web applications. It handles tasks such as servlet lifecycle management, HTTP request and response processing, and session management. Examples of web containers include Tomcat, Jetty, and GlassFish.

b) JSP Examples

i) include and taglib directives

- **include directive:** Includes the content of another file into the current JSP page.
  Java
  ```
  <%@ include file="header.jsp"%>
  ```
- **taglib directive:** Declares a custom tag library to be used in the JSP page.
  Java

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

ii) JSP scriptlet to display a list of first 10 positive odd numbers

```
Java
<%
for(inti = 1; i <= 10; i++) {
   if(i % 2!= 0) {
      out.println(i + "<br>");
   }
}
%>
```

iii) <jsp:setProperty> and <jsp:getProperty> action elements

- **<jsp:setProperty>:** Sets the value of a JavaBean property.
  ```
  Java
  <jsp:setProperty name="myBean"property="name"value="John Doe"/>
  ```
- **<jsp:getProperty>:** Retrieves the value of a JavaBean property.
  ```
  Java
  <jsp:getProperty name="myBean"property="name"/>
  ```

iv) session and application implicit objects

- **session object:** Represents a session between a client and a web application.
- **application object:** Represents the web application itself.

v) JSP Life cycle

1. **Compilation:** The JSP page is translated into a servlet class.
2. **Initialization:** The servlet class is loaded into memory and its init() method is called.
3. **Service:** The servlet's service() method is called for each HTTP request.
4. **Destruction:** The servlet's destroy() method is called when the web application is stopped.

c) JSP Programs

i) Finding the smallest of three variables

```
Java
<%
int x = Integer.parseInt(request.getParameter("x"));
int y = Integer.parseInt(request.getParameter("y"));
int z = Integer.parseInt(request.getParameter("z"));

intsmallest = Math.min(Math.min(x, y), z);
out.println("The smallest value is: "+ smallest);
%>
```

ii) Issuing a book from a library

```
Java
<%
String membershipID = request.getParameter("membershipID");
String bookID = request.getParameter("bookID");
String dateOfReturn = request.getParameter("dateOfReturn");
if(membershipID != null&& bookID != null&& dateOfReturn != null) {
   Cookie membershipCookie = newCookie("membershipID", membershipID);
   Cookie bookCookie = newCookie("bookID", bookID);
   response.addCookie(membershipCookie);
   response.addCookie(bookCookie);
}
%>
```

d) Book Sales System

Database Schema:

SQL

```
CREATETABLEBook (
   ISBNnumber VARCHAR(13) PRIMARYKEY,
   Title VARCHAR(100),
   FirstAuthor VARCHAR(50),
   YearOfPublication INT,
   CopiesAcquired INT);
CREATETABLESales (
   ISBNnumber VARCHAR(13),
   PersonName VARCHAR(50),
   NumberofCopiesSold INT,
   FOREIGNKEY (ISBNnumber) REFERENCESBook(ISBNnumber)
);
```

JSP Program:

Java

```
<%
// Get ISBNnumber from requestString isbn = request.getParameter("isbn");
// Retrieve book information from database// ... (JDBC code to query the Book table)// Display book
information and sales history%>
```

Webpage:

book sales system webpage

```
<%
// Get ISBNnumber from requestString isbn = request.getParameter("isbn");
// Retrieve book information from database
try {
   Class.forName("com.mysql.jdbc.Driver");
   Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/booksales",
"username", "password");
   PreparedStatement stmt = conn.prepareStatement("SELECT * FROM Book WHERE ISBNnumber = ?");
   stmt.setString(1, isbn);
   ResultSet rs = stmt.executeQuery();
if(rs.next()) {
      String title = rs.getString("Title");
      String author = rs.getString("FirstAuthor");
      intyearOfPublication = rs.getInt("YearOfPublication");
      intcopiesAcquired = rs.getInt("CopiesAcquired");
// Display book informationout.println("Book Title: "+ title);
      out.println("Author: "+ author);
      out.println("Year of Publication: "+ yearOfPublication);
      out.println("Copies Acquired: "+ copiesAcquired);
// Retrieve sales historystmt = conn.prepareStatement("SELECT * FROM Sales WHERE ISBNnumber = ?");
      stmt.setString(1, isbn);
      rs = stmt.executeQuery();
out.println("Sales History:");
      while(rs.next()) {
         String personName = rs.getString("PersonName");
         intnumberOfCopiesSold = rs.getInt("NumberofCopiesSold");
```

```
out.println("Person Name: "+ personName);
          out.println("Number of Copies Sold: "+ numberOfCopiesSold);
          out.println("<br>");
       }
   } else{
       out.println("Book not found.");
   }
conn.close();
} catch(Exception e) {
   e.printStackTrace();
}
%>
```

Webpage:

HTML

```
<formaction="bookSales.jsp"method="post">ISBN Number: <inputtype="text"name="isbn"><br>
<inputtype="submit"value="Search"></form>
```

Database: