Course Code	**:**	**BCS-053**
Course Title	**:**	**Web Programming**
Assignment Number	**:**	**BCA(V)053/Assignment/2025-26**
Maximum Marks	**:**	**100**
Last Date of Submission	**:**	**31stOctober,2025(For July, Session)**
		30thApril, 2026(For January, Session)

This assignment has two questions of 80 marks. Answer all the questions. Rest 20 marks are for viva voce. You may use illustrations and diagrams to enhance explanations. Please go through the guidelines regarding assignments given in the Programme Guide for the format of the presentation. Please give precise answers. The word limit for each part is 300 words.

Question 1: (Covers Block 1)

a) Explain the features of the following technologies: Content Networks, Social Media and Web Services. How are these technologies useful to you? **(6 Marks)**

b) (i) Create an online admission form for an Institute using HTML. The form should ask for the following information: (**3 Marks**)

- The Name of the candidate
- Mother’s Name
- Subject (Choose only one Subject from a drop-down list containing CS-01, CS-02, CS-03 and CS-04)
- Fee of the subject
- Have you registered for a subject earlier? (Yes/No)
- Your Educational Background (to be entered in text area)
- A SUBMIT button

(ii) Create an external CSS file for this form. This CSS file should select the font size of 16-point italics for all the labels; font colour should be Blue for the headings and dark green for the normal text. The background colour of the form should be light yellow.

(2 Marks)

(iii) Write JavaScript code to validate if any of the field of the form is not filled.

(3 Marks)

Submit the HTML code, JavaScript code and screenshot of the form opened in a browser window. You must demonstrate the form and validations at the time of the viva.

c) Using tables, create a webpage displaying the list of items in a departmental store. This webpage should display the item code, item name, unit price, and date of expiry of the item. Create a second page containing the ordered item by a customer, showing the item code, item name, unit price and ordered quantity. You should use <div> tags, wherever needed, and create an internal CSS file, which formats the web pages as given below:

(i) The headings of the table must be in 14-point Bold, and all other content should be in 12-point Times Roman font.

(ii) The table heading should be in a different shade. The data rows of the table should have alternate light yellow and light green colour shades. The background of the table should be light blue.

(iii) The font of the ordered list should be "Arial" and the font size should be 12 points. The background colour of the list should be light blue.

(iv) At the time of the viva, you should demonstrate how changes in CSS can change the display.

You must submit the HTML and CSS code and the screenshots of the pages in a browser window.

(6 Marks)

d) A University maintains the list of Books in its library using XML. Every Book is allotted a unique book code, which should be used as an attribute in the XML document. In addition, the following information is stored about the Books: Title of the book, Author(s) (minimum 1 and maximum 3), year of publication, and publisher. Create an XML document containing information about five Books. Also, create the DTD to verify the XML document created by you.

(8 Marks)

e) Write JavaScript code that displays the text "The Power of JavaScript is Dynamism". When you bring the mouse pointer over this text on the screen, it changes to “This is a Demonstration of Dynamism”. You may use event handling to perform the action as stated above. Make suitable assumptions, if any. Submit the code. You should demonstrate this code at the time of the viva.

(6 Marks)

f) Explain the WAP protocol stack. Also, explain the following WML elements with the help of an example of each:

- WML tables
- WML Images
- WML <anchor> element

(6 Marks)

Question 2: (Covers Block 2) **(10×4=40 Marks)**

a) Explain the following with the help of a diagram/example, if needed:

(i) Features of dynamic web pages

(ii) MVC Architecture

(iii) Tools for client-side scripting

(iv) HTTP methods

(v) Web Containers

b) Explain with the help of an example/diagram or write code for the following using JSP:

(i) *Page* and *include* directives of JSP

(ii) Write a JSP scriptlet to display a list of the first 8 double-digit positive even numbers.

(iii) <jsp:useBean> and <jsp:plugin> action elements of JSP

(iv) *out* and *exception* implicit objects in JSP

(v) Steps of JSP page processing

c) Write JSP programs which can perform the following tasks (you may create a single or multiple webpages for these tasks):

(i) Write a JSP code to create a simple web page that accepts user input for two variables, namely alpha and beta. After the successful input to these variables, the JSP program should display the values entered for the alpha and beta variables, along with the result of their multiplication.

(ii) Create a web page that takes input in two fields, namely the student ID and the Programme of the student. In case the data is correctly entered in both fields, two cookies, one for the student ID and the second for the Programme of the student, are created.

d) Create a database for the Student Examination System consisting of the following two tables:

Student (StudentID, Name, ProgrammeCode, DateOfEnrolment)

FeePaid (StudentID, Semester, dateofPayment, AmountPaid)

Develop and deploy a web-based "Student Examination System" using JSP, having a database backend and a web server (you may select DBMS and web server, as per your choice). Your system should use JDBC for input of information to both tables. The system should also display the StudentID, Name, dateofPayment and Amount paid for all the students.

Submit the JSP program, screens and database of the system. You must demonstrate this system at the time of viva voce.

Make and state suitable assumptions, if any.

BCS-53

Question 1: (Covers Block 1)

a) Explain the features of the following technologies: Content Networks, Social Media and Web Services. How are these technologies useful to you?

Answer **1. Content Delivery Networks (CDNs)**

A Content Delivery Network (CDN) is a geographically distributed network of proxy servers and their data centers. The goal is to provide high availability and performance by distributing the service spatially relative to end-users.

- **Features:**
  - **Geographic Distribution:** Servers are located at multiple points of presence (PoPs) around the world.
  - **Caching:** Static content (like HTML, CSS, JS, images) is cached on these distributed servers.
  - **Load Balancing:** Directs user requests to the server that is geographically closest or has the least network latency, which reduces the load on the origin server.
  - **High Availability:** Provides redundancy. If one server fails, traffic can be rerouted to another, preventing downtime.
- **Usefulness:**
  As a user, CDNs improve my web browsing experience significantly. When I visit a website that uses

a CDN, its content loads much faster because it is delivered from a server near my location. This reduces latency and makes streaming videos, viewing images, and accessing web pages quicker and more reliable.

2. Social Media

Social Media refers to websites and applications that enable users to create and share content or to participate in social networking.

- **Features:**
  - **User-Generated Content:** The core content is created and shared by the users themselves (e.g., posts, photos, videos).
  - **Networking:** Users can connect with friends, family, and communities based on shared interests.
  - **Interactivity:** Features like comments, likes, shares, and direct messaging facilitate real-time interaction.
  - **Personalization:** Algorithms create a personalized feed of content for each user based on their activity and connections.
- **Usefulness:**
  Social media is useful to me for staying connected with friends and family, staying informed about current events, and joining communities related to my hobbies and professional interests. It's a platform for self-expression and learning from others' experiences and knowledge.

3. Web Services

Web Services are a standardized way of integrating web-based applications using open standards over the internet. They allow different applications from different sources to communicate with each other without time-consuming custom coding.

- **Features:**
  - **Interoperability:** Based on open standards like XML, SOAP, and WSDL, they can be accessed by any application on any platform.
  - **Self-Describing:** A web service can describe its own functionality using a WSDL (Web Services Description Language) file.

- **Loosely Coupled:** The service consumer (client) and the service provider are independent. Changes in one do not break the other, as long as the interface contract is maintained.
- **Stateless:** Each request from a client is treated independently, without relying on previous requests.

- **Usefulness:**
  Web services work behind the scenes in many applications I use daily. For example, when a weather app on my phone shows the current temperature, it is likely calling a web service from a weather provider. When an e-commerce site processes a payment, it uses a payment gateway's web service. They enable seamless integration of different functionalities across the web.

b(i) Create an online admission form for an Institute using HTML. The form should ask for the following information: (3 Marks)

- **The Name of the candidate**
- **Mother's Name**
- **Subject (Choose only one Subject from a drop-down list containing CS-01, CS-02, CS-03 and CS04)**
- **Fee of the subject**
- **Have you registered for a subject earlier? (Yes/No)**
- **Your Educational Background (to be entered in text area)**
- **A SUBMIT button**

(ii) Create an external CSS file for this form. This CSS file should select the font size of 16-point italics for all the labels; font colour should be Blue for the headings and dark green for the normal text.
The background colour of the form should be light yellow. (2 Marks)

(iii) Write JavaScript code to validate if any of the field of the form is not filled. (3 Marks)

Submit the HTML code, JavaScript code and screenshot of the form opened in a browser window. You must demonstrate the form and validations at the time of the viva.

Answer

Here is the code for the HTML, CSS, and JavaScript parts. You would create three separate files as described below.

1. HTML File (admission.html)

This file contains the structure of the admission form.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Admission Form</title>
    <!-- Link to the external CSS file -->
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <h1>Online Admission Form</h1>

    <form name="admissionForm" action="#" onsubmit="return validateForm()"
method="post">
        <div class="form-container">
            <div>
                <label for="candidateName">Name of the candidate:</label>
                <input type="text" id="candidateName" name="candidateName">
            </div>
            <div>
                <label for="motherName">Mother’s Name:</label>
                <input type="text" id="motherName" name="motherName">
            </div>
            <div>
                <label for="subject">Subject:</label>
                <select id="subject" name="subject">
                    <option value="">--Choose a Subject--</option>
                    <option value="CS-01">CS-01</option>
                    <option value="CS-02">CS-02</option>
                    <option value="CS-03">CS-03</option>
                    <option value="CS-04">CS-04</option>
                </select>
            </div>
            <div>
                <label for="fee">Fee of the subject:</label>
                <input type="text" id="fee" name="fee">
            </div>
            <div>
                <label>Have you registered for a subject earlier?</label>
                <input type="radio" id="registeredYes" name="registered" value="Yes">
                <label class="radio-label" for="registeredYes">Yes</label>
                <input type="radio" id="registeredNo" name="registered" value="No">
                <label class="radio-label" for="registeredNo">No</label>
            </div>
            <div>
                <label for="background">Your Educational Background:</label>
                <textarea id="background" name="background" rows="4"></textarea>
            </div>
            <div>
                <input type="submit" value="SUBMIT">
            </div>
        </div>
    </form>

    <!-- Link to the JavaScript file -->
    <script src="validation.js"></script>

</body>
</html>
```

2. CSS File (style.css)

This file contains the styling for the form.

```
/* Styling for the form background */
.form-container {
    background-color: lightyellow;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 8px;
}

/* Styling for the main heading */
h1 {
    color: Blue;
}

/* Styling for all labels */
label {
    font-size: 16pt;
    font-style: italic;
    color: darkgreen; /* Normal text color */
    display: block;
    margin-bottom: 5px;
}

/* Normal text color for radio button labels which are not block elements */
.radio-label {
    display: inline;
    font-size: 16pt;
    font-style: normal; /* Typically radio labels aren't italicized */
    color: darkgreen;
}



/* General styling for input fields, select, and textarea */
input[type="text"], select, textarea {
    width: 100%;
    padding: 8px;
    margin-bottom: 15px;
    box-sizing: border-box; /* Ensures padding doesn't affect width */
    color: darkgreen;
}

/* SUBMIT button styling */
input[type="submit"] {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

input[type="submit"]:hover {
    background-color: #45a049;
}
```

3. JavaScript File (validation.js)

This file contains the validation logic.

```
function validateForm() {
    // Get form field values
    let candidateName = document.forms["admissionForm"]["candidateName"].value;
    let motherName = document.forms["admissionForm"]["motherName"].value;
    let subject = document.forms["admissionForm"]["subject"].value;
    let fee = document.forms["admissionForm"]["fee"].value;
    let registered = document.forms["admissionForm"]["registered"].value;
    let background = document.forms["admissionForm"]["background"].value;

    // Check if any field is empty
    if (candidateName == "" || candidateName == null) {
        alert("Candidate Name must be filled out");
        return false;
    }
    if (motherName == "" || motherName == null) {
        alert("Mother's Name must be filled out");
        return false;
    }
    if (subject == "") {
        alert("Please choose a subject");
        return false;
    }
    if (fee == "" || fee == null) {
        alert("Fee must be filled out");
        return false;
    }
    if (registered == "") {
        alert("Please select if you have registered earlier");
        return false;
    }
    if (background == "" || background == null) {
        alert("Educational Background must be filled out");
        return false;
    }

    // If all fields are filled, allow form submission
    alert("Form Submitted Successfully!");
    return true;
}
```

Screenshot Description:

When you open `admission.html` in a browser, you will see a heading "Online Admission Form" in blue. Below it, a form with a light yellow background will be displayed. Each field label (like "Name of the candidate:") will be in 16-point, italic, dark green font. If you click the "SUBMIT" button without filling out the form, a series of alert boxes will pop up, asking you to fill each required field.

c).Using tables, create a webpage displaying the list of items in a departmental store. This webpage should display the item code, item name, unit price, and date of expiry of the item. Create a second page containing the ordered item by a customer, showing the item code, item name, unit price and ordered quantity. You should use <div> tags, wherever needed, and create an internal CSS file, which formats the web pages as given below:

- **(i)** **The headings of the table must be in 14-point Bold, and all other content should be in 12-point Times Roman font.**
- **(ii)** **The table heading should be in a different shade. The data rows of the table should have alternate light yellow and light green colour shades. The background of the table should be light blue.**
- **(iii)** **The font of the ordered list should be "Arial" and the font size should be 12 points. The background colour of the list should be light blue.**
- **(iv)** **At the time of the viva, you should demonstrate how changes in CSS can change the display.**

You must submit the HTML and CSS code and the screenshots of the pages in a browser window.

(6 Marks)

Answer Here is the code for the two required pages with internal CSS.

Page 1: Departmental Store Items (store_items.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Departmental Store - Item List</title>
    <style>
        /* Internal CSS */
        body {
            font-family: "Times New Roman", serif;
            font-size: 12pt;
        }

        .container {
            width: 80%;
            margin: auto;
            background-color: lightblue;
            padding: 20px;
        }

        h1 {
            text-align: center;
        }

        table {
            width: 100%;
            border-collapse: collapse;
        }

        th, td {
            border: 1px solid #333;
            padding: 8px;
            text-align: left;
        }

        /* (i) Table headings */
        th {
            font-size: 14pt;
            font-weight: bold;
        }

        /* (ii) Table heading shade and row colors */
        thead {
            background-color: #a0c4ff; /* Different shade for heading */
        }

        tbody tr:nth-child(even) {
            background-color: lightgreen; /* Alternate row color */
        }
        tbody tr:nth-child(odd) {
            background-color: lightyellow; /* Alternate row color */
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Available Items in Store</h1>
        <table>
            <thead>
                <tr>
                    <th>Item Code</th>
                    <th>Item Name</th>
                    <th>Unit Price (INR)</th>
                    <th>Date of Expiry</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>ITM001</td>
                    <td>Whole Wheat Bread</td>
                    <td>45.00</td>
                    <td>20-Aug-2025</td>
                </tr>
                <tr>
                    <td>ITM002</td>
                    <td>Cheddar Cheese (200g)</td>
                    <td>150.00</td>
                    <td>30-Nov-2025</td>
                </tr>
                <tr>
                    <td>ITM003</td>
                    <td>Organic Milk (1L)</td>
                    <td>70.00</td>
                    <td>18-Aug-2025</td>
                </tr>
                <tr>
                    <td>ITM004</td>
                    <td>Basmati Rice (1kg)</td>
                    <td>120.00</td>
                    <td>15-Jun-2026</td>
                </tr>
                <tr>
                    <td>ITM005</td>
                    <td>Green Tea (100 bags)</td>
                    <td>250.00</td>
                    <td>01-Jan-2027</td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

Page 2: Customer Order (customer_order.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Customer Order</title>
    <style>
        /* Internal CSS */
        body {
            font-family: "Arial", sans-serif; /* (iii) Font for ordered list */
            font-size: 12pt;
        }

        .order-container {
            width: 80%;
            margin: auto;
            background-color: lightblue; /* (iii) Background color */
            padding: 20px;
        }

        h1 {
            text-align: center;
        }
        table {
            width: 100%;
            border-collapse: collapse;
        }

        th, td {
            border: 1px solid #333;
            padding: 8px;
            text-align: left;
            font-family: "Times New Roman", serif; /* Revert to Times for table content
as per Q(i) */
            font-size: 12pt;
        }

        /* (i) Table headings */
        th {
            font-size: 14pt;
            font-weight: bold;
        }

        /* (ii) Table heading shade and row colors */
        thead {
            background-color: #a0c4ff; /* Different shade for heading */
        }

        tbody tr:nth-child(even) {
            background-color: lightgreen;
        }

        tbody tr:nth-child(odd) {
            background-color: lightyellow;
        }
    </style>
</head>
<body>
    <div class="order-container">
        <h1>Customer Order Details</h1>
        <div>
            <p style="font-family: Arial; font-size: 12pt;">Customer: John Doe</p>
            <p style="font-family: Arial; font-size: 12pt;">Order ID: ORD5891</p>
        </div>
        <table>
            <thead>
                <tr>
                    <th>Item Code</th>
                    <th>Item Name</th>
                    <th>Unit Price (INR)</th>
                    <th>Ordered Quantity</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>ITM001</td>
                    <td>Whole Wheat Bread</td>
                    <td>45.00</td>
                    <td>2</td>
                </tr>
                <tr>
                    <td>ITM003</td>
                    <td>Organic Milk (1L)</td>
                    <td>70.00</td>
                    <td>3</td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

Note: The question (iii) refers to an "ordered list" but the context implies the second page's content. I have styled the container's font and background as requested and kept the table styling consistent with the first page, as is standard practice.

d). A University maintains the list of Books in its library using XML. Every Book is allotted a unique book code, which should be used as an attribute in the XML document. In addition, the following information is stored about the Books: Title of the book, Author(s) (minimum 1 and maximum 3), year of publication, and publisher. Create an XML document containing information about five Books. Also, create the DTD to verify the XML document created by you.

Answer

1. XML Document (library.xml)

This document contains the data for five books.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Library SYSTEM "library.dtd">
<Library>
    <Book book_code="B001">
        <Title>Web Programming</Title>
        <Author>John Doe</Author>
        <Year>2024</Year>
        <Publisher>Tech Publications</Publisher>
    </Book>
    <Book book_code="B002">
        <Title>Data Structures</Title>
        <Author>Jane Smith</Author>
        <Author>Peter Jones</Author>
        <Year>2023</Year>
        <Publisher>EduWorld Press</Publisher>
    </Book>
    <Book book_code="B003">
        <Title>Introduction to Algorithms</Title>
        <Author>T. Cormen</Author>
        <Author>C. Leiserson</Author>
        <Author>R. Rivest</Author>
        <Year>2022</Year>
        <Publisher>MIT Press</Publisher>
    </Book>
    <Book book_code="B004">
        <Title>Operating System Concepts</Title>
        <Author>Abraham Silberschatz</Author>
        <Year>2021</Year>
        <Publisher>Wiley</Publisher>
    </Book>
    <Book book_code="B005">
        <Title>Database System Concepts</Title>
        <Author>Korth</Author>
        <Author>Sudarshan</Author>
        <Year>2020</Year>
        <Publisher>McGraw Hill</Publisher>
    </Book>
```

```
</Library>
```

2. DTD File (library.dtd)

This file defines the structure and rules for `library.xml`. Both files should be in the same directory for verification.

```
<!-- DTD for Library XML -->
<!ELEMENT Library (Book+)>

<!ELEMENT Book (Title, Author+, Year, Publisher)>
<!-- Attribute book_code is a required unique identifier -->
<!ATTLIST Book book_code ID #REQUIRED>

<!ELEMENT Title (#PCDATA)>
<!-- Element Author allows 1 to 3 authors -->
<!ELEMENT Author (#PCDATA)>
<!-- The '+' in Book's definition (Author+) ensures at least one. We cannot enforce a
maximum of 3 in DTD. -->
<!-- For strict max=3 validation, an XML Schema (XSD) would be needed. DTD is less
expressive. -->
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

Note on DTD limitation: DTDs can specify a minimum of one author (`Author+`) or zero or more authors (`Author*`), but they cannot enforce a maximum limit like "maximum 3". XML Schema (XSD) is required for that level of constraint. The provided DTD fulfills the requirements as closely as possible.

e). Write JavaScript code that displays the text "The Power of JavaScript is Dynamism". When you bring the mouse pointer over this text on the screen, it changes to "This is a Demonstration of Dynamism". You may use event handling to perform the action as stated above. Make suitable assumptions, if any. Submit the code. You should demonstrate this code at the time of the viva.

Answer

This can be done with a simple HTML file containing JavaScript.

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

```
<head>
    <meta charset="UTF-8">
    <title>JavaScript Event Handling</title>
    <style>
        #dynamicText {
            font-size: 24px;
            font-weight: bold;
            padding: 20px;
            cursor: pointer;
            border: 2px solid black;
            display: inline-block;
        }
    </style>
</head>
<body>

    <h1>Mouse Event Demonstration</h1>

    <div id="dynamicText"
         onmouseover="changeText(this)"
         onmouseout="revertText(this)">
        The Power of JavaScript is Dynamism
    </div>

    <script>
        // Function to change the text on mouse over
        function changeText(element) {
            element.innerHTML = "This is a Demonstration of Dynamism";
        }

        // Function to revert the text on mouse out
        function revertText(element) {
            element.innerHTML = "The Power of JavaScript is Dynamism";
        }
    </script>

</body>
</html>
```

How it works:
The `<div>` element has an ID `dynamicText`. We use two event handler attributes:

- `onmouseover`: This event fires when the mouse pointer is moved onto the element. It calls the `changeText()` function.
- `onmouseout`: This event fires when the mouse pointer is moved out of the element. It calls the `revertText()` function to change the text back to the original.

f) Explain the WAP protocol stack. Also, explain WML elements with the help of an example of each:

WAP Protocol Stack

The Wireless Application Protocol (WAP) is a technical standard for accessing information over a mobile wireless network. Its protocol stack is designed to be lightweight and optimized for low-bandwidth, high-latency mobile environments.

The layers are:

1. **Application Layer (WAE - Wireless Application Environment):** This is where WML (Wireless Markup Language) and WMLScript reside. WAE provides the environment for web applications on wireless devices, similar to how HTML and JavaScript work in a standard browser.
2. **Session Layer (WSP - Wireless Session Protocol):** It provides two types of services: a connection-oriented service (like HTTP) and a connectionless service. It manages session suspension and resumption, which is crucial for mobile networks where connections can be intermittent.
3. **Transaction Layer (WTP - Wireless Transaction Protocol):** This layer runs on top of a datagram service like UDP and provides a reliable request-response transaction service. It's more efficient than TCP for mobile networks.
4. **Security Layer (WTLS - Wireless Transport Layer Security):** Based on TLS (formerly SSL), this layer provides privacy, data integrity, and authentication for data transmitted between the device and the WAP gateway.
5. **Transport Layer (WDP - Wireless Datagram Protocol):** This is the adaptation layer that makes the upper layers independent of the underlying network bearer. It presents a consistent datagram service to the upper layers, communicating over various bearers like SMS, USSD, or GPRS.

WML Elements with Examples

1. WML Tables

WML tables are used to arrange text and images in rows and columns, similar to HTML tables.

- **Explanation:** The `<table>` element defines the table. It contains `<tr>` (table row) and `<td>` (table data/cell) elements. The `columns` attribute in the `<table>` tag is mandatory and specifies the number of columns.

- **Example:**

```
<wml>
    <card id="table_card" title="Schedule">
        <p>
            <b>Daily Schedule:</b>
            <table columns="2">
                <tr><td>09:00 AM</td><td>Meeting</td></tr>
                <tr><td>01:00 PM</td><td>Lunch</td></tr>
                <tr><td>03:00 PM</td><td>Project Work</td></tr>
            </table>
        </p>
    </card>
</wml>
```

2. WML Images

WML supports images, but they must be in the Wireless Bitmap (`.wbmp`) format, which is a monochrome, uncompressed image format optimized for mobile devices.

- **Explanation:** The `<img>` tag is used to display an image. Key attributes are `src` (the URL of the image) and `alt` (alternative text if the image cannot be displayed).

- **Example:**

```
<wml>
    <card id="image_card" title="Logo">
        <p align="center">
            Our Company Logo:<br/>
            <img src="logo.wbmp" alt="Company Logo"/>
        </p>
    </card>
```

```
</wml>
```

3. WML <anchor> Element

The `<anchor>` element provides a way to create a link and associate a task with it, combining hyperlink functionality with an action.

- **Explanation:** It's more versatile than a simple `<a>` (anchor) tag. It wraps both the link text and the task to be performed, such as `go`, `prev`, or `refresh`.
- **Example:**

```
<wml>
    <card id="menu" title="Main Menu">
        <p>
            <anchor>
                Go to News
                <go href="#news_card"/>
            </anchor>
            <br/>
            <anchor>
                Back to Previous
                <prev/>
            </anchor>
        </p>
    </card>

    <card id="news_card" title="News">
        <p>Today's headlines...</p>
    </card>
</wml>
```

Question 2: (Covers Block 2)

a) Explain the following with the help of a diagram/example, if needed:

(i) Features of dynamic web pages

Answer A dynamic web page is a page whose content is generated on-the-fly by a server-side script in response to a user's request. This contrasts with static pages, which are pre-built and delivered as-is.

- **Features:**
  - **User Interaction:** Content can change based on user input (e.g., search results, customized profiles).
  - **Database Integration:** Can retrieve and display data from a database (e.g., a product catalog on an e-commerce site).
  - **Real-time Content:** Can display content that changes frequently (e.g., stock prices, news headlines).
  - **Personalization:** Can provide a customized experience for each user (e.g., Amazon's recommended products).
- **Example:** A Facebook news feed. The content is different for every user and changes every time they visit the page, based on their friends' activities.

(ii) MVC Architecture

Answer Model-View-Controller (MVC) is a software design pattern for developing user interfaces that divides the application logic into three interconnected components.

- **Model:** Manages the application's data and business logic. It receives input from the controller and is independent of the user interface.
- **View:** Manages the data display. It renders the model into a suitable user interface. It gets the data to display from the model.
- **Controller:** Handles user input and interactions. It acts as an intermediary between the Model and the View, receiving user requests and calling the appropriate resources in the Model and View to generate a response.
- **Diagram/Flow:**
  a. User interacts with the **View**.
  b. The **Controller** receives the user's request.
  c. The **Controller** interacts with the **Model** to update its state or retrieve data.
  d. The **View** queries the **Model** for the data it needs to display.
  e. The **View** renders the final output for the user.

(iii) Tools for client-side scripting

Answer Client-side scripting refers to code that is executed on the user's computer (in the browser), rather than on the web server.

- **Core Technology:**
  - **JavaScript:** The primary language for client-side scripting.
- **Frameworks and Libraries:** These are built on top of JavaScript to make development faster and easier.
  - **React:** A library for building user interfaces, developed by Facebook.
  - **Angular:** A platform and framework for building single-page client applications, developed by Google.
  - **Vue.js:** A progressive framework for building user interfaces.
  - **jQuery:** A fast, small, and feature-rich JavaScript library that simplifies HTML document traversal, event handling, and animating.
- **Development Tools:**
  - **Code Editors:** VS Code, Sublime Text, Atom.
  - **Browser DevTools:** Built-in tools in Chrome, Firefox, etc., for debugging, inspecting the DOM, and analyzing network activity.

(iv) HTTP methods

Answer HTTP (Hypertext Transfer Protocol) methods are request methods (or "verbs") that indicate the desired action to be performed for a given resource.

- **GET:** Requests a representation of the specified resource. It should only retrieve data and have no other effect.
- **POST:** Submits an entity to the specified resource, often causing a change in state or side effects on the server (e.g., creating a new user).
- **PUT:** Replaces all current representations of the target resource with the request payload. It is idempotent (multiple identical requests have the same effect as one).
- **DELETE:** Deletes the specified resource.
- **HEAD:** Asks for a response identical to a GET request, but without the response body. Useful for checking if a resource exists before downloading it.
- **PATCH:** Applies partial modifications to a resource.

(v) Web Containers

Answer A web container (or servlet container) is a component of a web server that interacts with Java servlets and JSPs. It is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet, and ensuring that the URL requester has the correct access rights.

- **Functionality:**
  - **Lifecycle Management:** Manages `init()`, `service()`, and `destroy()` methods of servlets.
  - **Request Dispatching:** Maps incoming HTTP requests to the appropriate servlet or JSP.
  - **Security:** Manages authentication and authorization.
  - **Session Management:** Creates and manages user sessions.
- **Examples:**
  - **Apache Tomcat:** The most popular open-source web container.
  - **Jetty:** A lightweight, open-source web server and servlet container.
  - **JBoss / WildFly:** A full application server that includes a web container.

b) Explain with the help of an example/diagram or write code for the following using JSP:

(i) Page and include directives of JSP

- **Page Directive (<%@ page ... %>):** Provides instructions to the container that apply to the entire JSP page. It can be used to import classes, define the session model, set the content type, etc.
  - **Example:**

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ page import="java.util.Date" %>
<html><body>
Today's date is: <%= new Date() %>
</body></html>
```

- **Include Directive (<%@ include ... %>):** Includes the content of another file (HTML, JSP, etc.) at translation time (when the JSP is converted to a servlet). The inclusion is static.

- **Example:**

  File: header.jsp

```
<h1>My Website</h1><hr>
```

File: main.jsp

```
<%@ page language="java" %>
<html><body>
<%@ include file="header.jsp" %>
<p>This is the main content.</p>
</body></html>
```

(ii) Write a JSP scriptlet to display a list of the first 8 double-digit positive even numbers.

A scriptlet is a block of Java code embedded in a JSP page, enclosed in `<% ... %>`.

- **Code:**

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head><title>Even Numbers</title></head>
<body>
    <h2>First 8 Double-Digit Positive Even Numbers:</h2>
    <ul>
        <%
            int count = 0;
            int num = 10; // Start with the first double-digit number
            while (count < 8) {
                if (num % 2 == 0) {
                    out.println("<li>" + num + "</li>");
                    count++;
                }
                num++;
            }
        %>
    </ul>
</body>
</html>
```

(iii) <jsp:useBean> and <jsp:plugin> action elements of JSP

- **<jsp:useBean>:** This action tag is used to locate or instantiate a JavaBean component. If an object of the bean already exists in the specified scope, it uses that one; otherwise, it creates a new instance.
    - **Example:** Suppose you have a `User.java` bean.

```
<jsp:useBean id="user" class="com.example.User" scope="session" />
<jsp:setProperty name="user" property="name" value="Admin" />
<p>Welcome, <jsp:getProperty name="user" property="name" /></p>
```

- **<jsp:plugin>:** This action tag is used to embed a Java applet or a JavaBean in a JSP page. It generates the appropriate `OBJECT` or `EMBED` tag depending on the browser, ensuring compatibility.
    - **Example:**

```
<jsp:plugin type="applet" code="MyApplet.class" codebase="applets"
width="300" height="200">
    <jsp:param name="message" value="Hello from JSP"/>
    <jsp:fallback>
        <p>Applet could not be loaded.</p>
    </jsp:fallback>
</jsp:plugin>
```

(iv) out and exception implicit objects in JSP

- **out:** This object is an instance of `javax.servlet.jsp.JspWriter`. It is used to write content to the response stream. It's the most common way to output text in a JSP.
    - **Example:**

```
<% out.println("Hello, World!"); %>
```

- **exception:** This object is an instance of `java.lang.Throwable`. It is only available in JSP pages that are designated as error pages (by setting `<%@ page isErrorPage="true" %>`). It holds the exception that was thrown from another JSP page.
    - **Example:**
      File: errorPage.jsp

```
<%@ page isErrorPage="true" %>
<html><body>
<h2>An error occurred:</h2>
<p>Error Message: <%= exception.getMessage() %></p>
```

```
</body></html>
```

(v) Steps of JSP page processing

When a client requests a JSP page, the web container performs the following steps:

1. **Translation:** The container translates the `.jsp` file into a Java servlet source code (`.java`) file. This happens only the first time the JSP is requested or if the JSP has been modified since the last request.
2. **Compilation:** The container compiles the generated servlet source code into a Java servlet class file (`.class`).
3. **Loading:** The container loads the compiled servlet class into memory.
4. **Instantiation:** The container creates an instance of the servlet class.
5. **Initialization:** The container calls the `jspInit()` method of the servlet instance. This method is called only once during the servlet's lifecycle.
6. **Request Processing:** For each client request, the container calls the `_jspService()` method of the servlet instance, passing the `request` and `response` objects. This method processes the request and generates the HTML response.
7. **Destruction:** When the container shuts down or the application is undeployed, it calls the `jspDestroy()` method to clean up any resources used by the JSP.

c) Write JSP programs which can perform the following tasks:

(i) Write a JSP code... that accepts user input for two variables, alpha and beta... and display the values... along with their multiplication.

This task is best done with two files: an HTML form and a JSP file to process the data.

File 1: input_multiply.html

```
<!DOCTYPE html>
<html>
<head><title>Multiplication Input</title></head>
<body>
    <h2>Enter two numbers (alpha and beta)</h2>
    <form action="calculate.jsp" method="post">
        Alpha: <input type="text" name="alpha"><br><br>
        Beta:  <input type="text" name="beta"><br><br>
        <input type="submit" value="Calculate">
    </form>
</body>
</html>
```

File 2: calculate.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head><title>Multiplication Result</title></head>
<body>
    <h2>Calculation Result</h2>
    <%
        try {
            // Retrieve parameters from the request
            String alphaStr = request.getParameter("alpha");
            String betaStr = request.getParameter("beta");

            // Convert strings to numbers (doubles for flexibility)
            double alpha = Double.parseDouble(alphaStr);
            double beta = Double.parseDouble(betaStr);

            // Perform multiplication
            double result = alpha * beta;

            // Display the values and the result
            out.println("<p>Value of alpha: " + alpha + "</p>");
            out.println("<p>Value of beta: " + beta + "</p>");
            out.println("<h3>Result of alpha * beta: " + result + "</h3>");

        } catch (NumberFormatException e) {
            out.println("<p style='color:red;'>Error: Please enter valid numbers.</p>");
        } catch (Exception e) {
            out.println("<p style='color:red;'>An unexpected error occurred.</p>");
        }
    %>
    <br>
    <a href="input_multiply.html">Go Back</a>
</body>
```

```
</html>
```

(ii) Create a web page that takes input in two fields, namely the student ID and the Programme... two cookies... are created.

File: create_cookies.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head><title>Cookie Management</title></head>
<body>

    <%
        String studentId = request.getParameter("studentId");
        String programme = request.getParameter("programme");
        String message = "";

        // Check if the form has been submitted
        if (studentId != null && programme != null && !studentId.isEmpty() &&
!programme.isEmpty()) {
            // Create a cookie for Student ID
            Cookie studentIdCookie = new Cookie("studentID", studentId);
            studentIdCookie.setMaxAge(60 * 60 * 24); // Set expiry to 1 day
            response.addCookie(studentIdCookie);

            // Create a cookie for Programme
            Cookie programmeCookie = new Cookie("programme", programme);
            programmeCookie.setMaxAge(60 * 60 * 24); // Set expiry to 1 day
            response.addCookie(programmeCookie);

            message = "Cookies for Student ID and Programme have been created
successfully!";
        }
    %>

    <h2>Enter Student Details to Create Cookies</h2>
    <form action="create_cookies.jsp" method="post">
        Student ID: <input type="text" name="studentId"><br><br>
        Programme:  <input type="text" name="programme"><br><br>
        <input type="submit" value="Create Cookies">
    </form>

    <%
        if (!message.isEmpty()) {
            out.println("<h3>" + message + "</h3>");
        }
    %>

</body>
</html>
```

d) Create a database for the Student Examination System... Develop and deploy a web-based “Student Examination System” using JSP...

This is a complete application. It requires a database setup, JDBC driver, and multiple JSP files.

Assumptions:

- **DBMS:** MySQL
- **Web Server:** Apache Tomcat
- **JDBC Driver:** MySQL Connector/J `.jar` file placed in the `WEB-INF/lib` directory of the web application.

Step 1: Database and Table Creation (SQL)

```
-- Create a new database for the system
CREATE DATABASE student_exam_db;

-- Use the new database
USE student_exam_db;

-- Create the Student table
CREATE TABLE Student (
    StudentID VARCHAR(20) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    ProgrammeCode VARCHAR(20),
    DateOfEnrolment DATE
);

-- Create the FeePaid table
CREATE TABLE FeePaid (
    FeeID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID VARCHAR(20),
    Semester INT,
    dateofPayment DATE,
    AmountPaid DECIMAL(10, 2),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);
```

Step 2: Web Application Files

You would structure your web application in a standard directory structure (e.g., `StudentSystem/`).

File 1: add_student.html (Data Entry Form)

This form will be used to input data into both tables.

```
<!DOCTYPE html>
<html>
<head><title>Add Student and Fee Details</title></head>
<body>
    <h2>Student Examination System - Data Entry</h2>
    <form action="process_student.jsp" method="post">
        <h3>Student Information</h3>
        Student ID: <input type="text" name="studentId" required><br>
        Name: <input type="text" name="name" required><br>
        Programme Code: <input type="text" name="programmeCode" required><br>
        Date of Enrolment (YYYY-MM-DD): <input type="date" name="enrolmentDate"
required><br>

        <h3>Fee Payment Information</h3>
        Semester: <input type="number" name="semester" required><br>
        Date of Payment (YYYY-MM-DD): <input type="date" name="paymentDate"
required><br>
        Amount Paid: <input type="text" name="amountPaid" required><br><br>

        <input type="submit" value="Submit Information">
    </form>
    <hr>
    <a href="display_students.jsp">View Student Payment Report</a>
</body>
</html>
```

File 2: process_student.jsp (JDBC Logic to Insert Data)

```
<%@ page import="java.sql.*" %>
<%
    // Get form parameters
    String studentId = request.getParameter("studentId");
    String name = request.getParameter("name");
    String programmeCode = request.getParameter("programmeCode");
    String enrolmentDate = request.getParameter("enrolmentDate");

    String semester = request.getParameter("semester");
    String paymentDate = request.getParameter("paymentDate");
    String amountPaid = request.getParameter("amountPaid");

    Connection conn = null;
    PreparedStatement pstmtStudent = null;
    PreparedStatement pstmtFee = null;
    String message = "";

    try {
        // JDBC connection details
        String url = "jdbc:mysql://localhost:3306/student_exam_db";
        String user = "root"; // change as per your DB user
        String password = "password"; // change as per your DB password
        Class.forName("com.mysql.cj.jdbc.Driver");

        conn = DriverManager.getConnection(url, user, password);
        conn.setAutoCommit(false); // Start transaction

        // Insert into Student table (handle potential duplicates)
        String sqlStudent = "INSERT INTO Student (StudentID, Name, ProgrammeCode,
DateOfEnrolment) VALUES (?, ?, ?, ?) ON DUPLICATE KEY UPDATE Name=Name";
        pstmtStudent = conn.prepareStatement(sqlStudent);
        pstmtStudent.setString(1, studentId);
        pstmtStudent.setString(2, name);
        pstmtStudent.setString(3, programmeCode);
        pstmtStudent.setDate(4, java.sql.Date.valueOf(enrolmentDate));
        pstmtStudent.executeUpdate();

        // Insert into FeePaid table
        String sqlFee = "INSERT INTO FeePaid (StudentID, Semester, dateofPayment,
AmountPaid) VALUES (?, ?, ?, ?)";
        pstmtFee = conn.prepareStatement(sqlFee);
        pstmtFee.setString(1, studentId);
        pstmtFee.setInt(2, Integer.parseInt(semester));
        pstmtFee.setDate(3, java.sql.Date.valueOf(paymentDate));
        pstmtFee.setDouble(4, Double.parseDouble(amountPaid));
        pstmtFee.executeUpdate();

        conn.commit(); // Commit transaction
        message = "Student and Fee information saved successfully!";

    } catch (Exception e) {
        if (conn != null) conn.rollback(); // Rollback on error
        message = "Error saving data: " + e.getMessage();
        e.printStackTrace();
    } finally {
        if (pstmtStudent != null) pstmtStudent.close();
        if (pstmtFee != null) pstmtFee.close();
        if (conn != null) conn.close();
    }
%>
<html><body>
    <h2><%= message %></h2>
    <a href="add_student.html">Add Another Student</a><br>
    <a href="display_students.jsp">View Student Payment Report</a>
</body></html>
```

File 3: display_students.jsp (Display Report)

```
<%@ page import="java.sql.*" %>
<html>
<head><title>Student Payment Report</title></head>
<body>
    <h2>Student Payment Report</h2>
    <table border="1" cellpadding="5" cellspacing="0">
        <tr>
            <th>Student ID</th>
            <th>Name</th>
            <th>Date of Payment</th>
            <th>Amount Paid</th>
        </tr>
        <%
            Connection conn = null;
            Statement stmt = null;
            ResultSet rs = null;
            try {
                String url = "jdbc:mysql://localhost:3306/student_exam_db";
                String user = "root";
                String password = "password";
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection(url, user, password);

                String sql = "SELECT s.StudentID, s.Name, f.dateofPayment, f.AmountPaid
" +
                             "FROM Student s JOIN FeePaid f ON s.StudentID = f.StudentID
" +
                             "ORDER BY s.Name, f.dateofPayment";
                stmt = conn.createStatement();
                rs = stmt.executeQuery(sql);

                while (rs.next()) {
        %>
                    <tr>
                        <td><%= rs.getString("StudentID") %></td>
                        <td><%= rs.getString("Name") %></td>
                        <td><%= rs.getDate("dateofPayment") %></td>
                        <td><%= rs.getDouble("AmountPaid") %></td>
                    </tr>
        <%
                }
            } catch (Exception e) {
                out.println("<tr><td colspan='4'>Error fetching data: " + e.getMessage()
+ "</td></tr>");
                e.printStackTrace();
            } finally {
                if (rs != null) rs.close();
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            }
        %>
    </table>
    <br>
    <a href="add_student.html">Back to Data Entry</a>
</body>
</html>
```