

WEB ENGINEERING LAB

ETEC- 356

Faculty Name: Dr Sunil Maggu

Student Name : Sarthak Bhatia

Roll No. : 07714803118

Semester : 6th

Batch: 2022



Department Of Information Technology

Maharaja Agrasen Institute of Technology, PSP area,
Sector-22, Rohini, New Delhi -110085

INDEX

S.NO.	Name of the Program	Date	Signature	Remarks
1.	Create a web page that covers your CV using various HTML Tags	18 - 3 - 21		
2.	Create a webpage that display brief Details of various Programming Languages using various types of CSS.	18 - 3 - 21		
3.	Create a web page using java-script and HTML to demonstrate Simple Calculator Application.	18 - 3 - 21		
4.	Create an XML file to store details of students in a class and apply CSS for its presentation on web browser.	25 - 3 - 21		
5.	Create a webpage to demonstrate the use of Date, String, Array and Math object in javascript.	25 - 3 - 21		
6.	Design a User registration form using HTML tags.	25 - 3 - 21		
7.	Create a static web application using HTML, CSS and javascript with navigation links on topic " Career opportunities in Information Technology .	1 - 4 - 21		
8.	Explain DTD, XSL, XSLT and Display Food menu using XML and XSLT.	1 - 4 - 21		
9.	Define Web Engineering? Explain characteristics of Web application and Draw a diagram to categorize Web applications.	1 - 4 - 21		
10.	Explain how search engine works and list out step by step process.	8 - 4 - 21		

11.	What are the main challenges in developing an organizational website.	8 - 4 – 21		
12.	Explain the steps in designing and developing a website and explain characteristics of a good website design	8 - 4 – 21		
13.	What is SEO and Explain steps to improve Website ranking using SEO	29 - 4 – 21		
14.	Write short notes on CGI and perl	29 - 4 – 21		
15.	What are the advantages and disadvantages of javascript	29 - 4 – 21		
16.	Create a webpage to demonstrate the use of alert, prompt and confirm dialog box in javascript.	29 - 4 – 21		
17.	Create a game "Lucky Seven" using JavaScript.	6 - 5 – 21		
18.	Create a jsp page for fetching request data and display with current date as response.	6 - 5 – 21		
19.	Create a servlet for fetching request data and display with current date as response.	6 - 5 – 21		
20.	Create a jsp page Form Validation using java-script(blank text field and email).	13 - 5 – 21		
21.	What are cookies? Explain their uses and Create a web app and use cookies to manage sessions.	13 - 5 – 21		
22.	Create a webpage using HTML 5 tags.	13 - 5 – 21		
23.	What are Web security issues and how they can be controlled.	20 - 5 – 21		

24.	Define the terms Semantic Web and Ontology.	20 - 5 – 21		
25.	What is Digital signature and Digital Certificate. Explain their use.	20 - 5 – 21		
26.	Explain steps of implementing web services in any web application.	27-5-21		
27.	What is E-commerce and explain various ecommerce models and Ecommerce infrastructure.	27-5-21		
28.	What are various online payment methods and explain their Security issues.	27-5-21		
29.	What is Internet Marketing? Explain various type and their relevance.	27-5-21		
30.	Develop a login based application using Ajax, javascript and JSP.	3-6-21		
31.	Create snake game using HTML5(Canvas) and java script.	3-6-21		
32.	Describe how https can be configured on web server to ensure security requirements of Website.	3-6-21		

Experiment – 1

Aim: Create a web page that covers your CV using various HTML Tags

```
<!DOCTYPE html>
<html>
  <title>W3.CSS Template</title>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css" />
  <link
    rel="stylesheet"
    href="https://fonts.googleapis.com/css?family=Roboto"
  />
  <link
    rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
  />
  <style>
    html,
    body,
    h1,
    h2,
    h3,
    h4,
    h5,
    h6 {
      font-family: "Roboto", sans-serif;
    }
  </style>
  <body class="w3-light-grey">
    <div class="w3-content w3-margin-top" style="max-width: 1400px">
      <div class="w3-row-padding">
        <div class="w3-third">
          <div class="w3-white w3-text-grey w3-card-4">
            <div class="w3-display-container">
              <div
                class="w3-display-bottomleft w3-container w3-text-black"
                style="margin: 0px"
              >
                <h2>John Smith</h2>
              </div>
            </div>
          <div class="w3-container" style="margin-top: 50px">
            <p>
              <i
                class="
                  fa fa-briefcase fa-fw
                  w3-margin-right w3-large w3-text-teal
                "
              ></i>
              >Engineer
            </p>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```







```

    </h2>
    <div class="w3-container">
      <h5 class="w3-opacity">
        <b>Maharaja Agrasen Institute of Technology</b>
      </h5>
      <h6 class="w3-text-teal">
        <i class="fa fa-calendar fa-fw w3-margin-right"></i>2018 - 2022
      </h6>
      <p>Bachelor Degree</p>
      <hr />
    </div>
  </div>
</div>
</div>
</div>
</body>
</html>

```

Output:

John Smith

 Engineer
  Delhi, India
  john@it.mait.ac.in
  9876543211

* Skills

Coding

90%

Web Designing


80%

Aptitude


80%

Work Experience

Front End Developer / w3schools.com


 June 2020 - **Current**

Web Developer / tutorialspoint.com

 May 2019 - July 2019

Education

Maharaja Agrasen Institute of Technology

 2018 - 2022

Bachelor Degree

Experiment – 2

Aim: Create a webpage that display brief Details of various Programming Languages using various types of CSS.

```
<!DOCTYPE html>
<html>
  <title>W3.CSS</title>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css" />
  <body>
    <style>
      img {
        width: 80%;
        max-height: 170px;
      }
      body{
        background-color: lightblue;
      }
    </style>
    <div style="background-color: lightblue; font-size:10rem; text-align:center; color:blue">
      <h1>Programming Languages</h1>
    </div>

    <div class="w3-content">
      <div class="w3-row w3-margin">
        <div class="w3-third">
          
        </div>

        <div class="w3-twothird w3-container">
          <h2>Javascript</h2>
          <p>
            JavaScript (which, confusingly, is not at all related to Java) is
            another favorite programming language because it's so ubiquitous on
            the web--it's basically everywhere. JavaScript allows developers to
            add interactive elements to their website, and its presence is felt
            across the internet. At WordStream, we use a JavaScript library
            called JQuery to make our JavaScript work even easier.
          </p>
        </div>
      </div>

      <div class="w3-row w3-margin">
        <div class="w3-third">
          
        </div>
        <div class="w3-twothird w3-container">
          <h2>Python</h2>
          <p>
```

Python is a one-stop shop. There's a Python framework for pretty much anything, from web apps to data analysis. In fact, WordStream is written in Python! You're the best bud. Python is often heralded as the easiest programming language to learn, with its simple and straightforward syntax. Python has risen in popularity due to Google's investment in it over the past decade (in fact, one recent study has shown Python to be the most commonly taught programming language in U.S. schools). Other applications built with Python include Pinterest and Instagram.

</p>

</div>

</div>

<div class="w3-row w3-margin">

<div class="w3-third">

</div>

<div class="w3-twothird w3-container">

<h2>Java</h2>

<p>

The Java language is a multi platform language that's particularly helpful in networking. Of course, mostly this language is used on the web with Java applets. However, this language is used to design cross platform programs, Since it similar to C++ in structure and syntax. For C++ programmers, Java language is very easy to learn and it offers some advantages provided by object oriented programming. Like reusability and it can be difficult to write efficient code in Java. But, nowadays the speed of the Java language has increased and 1.5 version offers some good features for easy program making.

</p>

</div>

</div>

<div class="w3-row w3-margin">

<div class="w3-third">

</div>

<div class="w3-twothird w3-container">

<h2>C++</h2>

<p>

The C++ language has an object oriented structure which is used in large projects. Programmers can collaborate one program into different parts or even one individual work on each part of the program. The structure of object oriented also permit code to be reused many times. This language is an efficient language. But, many programmers will disagree.

</p>

</div>

</div>

</div>

</body>

```
</html>
```

Output:

Programming Languages



Javascript

JavaScript (which, confusingly, is not at all related to Java) is another favorite programming language because it's so ubiquitous on the web--it's basically everywhere. JavaScript allows developers to add interactive elements to their website, and its presence is felt across the internet. At WordStream, we use a JavaScript library called JQuery to make our JavaScript work even easier.



Python

Python is a one-stop shop. There's a Python framework for pretty much anything, from web apps to data analysis. In fact, WordStream is written in Python! You're the best bud. Python is often heralded as the easiest programming language to learn, with its simple and straightforward syntax. Python has risen in popularity due to Google's investment in it over the past decade (in fact, one recent study has shown Python to be the most commonly taught programming language in U.S. schools). Other applications built with Python include Pinterest and Instagram.



Java

The Java language is a multi platform language that's particularly helpful in networking. Of course, mostly this language is used on the web with Java applets. However, this language is used to design cross platform programs, Since it similar to C++ in structure and syntax. For C++ programmers, Java language is very easy to learn and it offers some advantages provided by object oriented programming. Like reusability and it can be difficult to write efficient code in Java. But, nowadays the speed of the Java language has increased and 1.5 version offers some good features for easy program making.



C++

The C++ language has an object oriented structure which is used in large projects. Programmers can collaborate one program into different parts or even one individual work on each part of the program. The structure of object oriented also permit code to be reused many times. This language is an efficient language. But, many programmers will disagree.

Experiment – 3

Aim: Create a webpage using java-script and HTML to demonstrate Simple Calculator Application.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        background-color: #3a3b35;
        color: white;
      }
      table {
        margin: auto;
        background-color: #9dd2ea;
        width: 295px;
        height: 325px;
        text-align: center;
        border-radius: 4px;
      }
      input {
        outline: 0;
        position: relative;
        left: 5px;
        top: 5px;
        border: 0;
        color: #495069;
        background-color: 4px;
        width: 60px;
        height: 50px;
        float: left;
        margin: 5px;
        font-size: 20px;
        box-shadow: 0 4px rgba(0, 0, 0, 0.2);
      }
      .operator {
        background-color: #f1ff92;
        position: relative;
      }
      #clear {
        float: left;
        position: relative;
        display: block;
        background-color: #ff9fa8;
        margin-bottom: 15px;
      }
      #answer {
        width: 270px;
        font-size: 26px;
```

```

        text-align: center;

        background-color: #f1faeb;
        float: left;
    }
    #answer:hover {
        width: 270px;
        font-size: 26px;
        text-align: center;
        box-shadow: 0 4px rgba(0, 0, 0, 0.2);
        background-color: #f1faeb;
    }
    input:hover {
        border: 0 solid #000;
        color: #495069;
        background-color: #8f5fda;
        border-radius: 4px;
        width: 60px;
        height: 50px;
        float: left;
        margin: 5px;
        font-size: 20px;
        box-shadow: 0 4px #644294;
    }
    .operator:hover {
        background-color: #e7f56b;
        box-shadow: 0 4px #b7c259;
    }
    #clear:hover {
        float: left;
        display: block;
        background-color: #f297a0;
        margin-bottom: 15px;
        box-shadow: 0 4px #cc7f86;
    }
</style>
</head>
<body>
    <form name="calc">
        <div style="display: flex">
            <input type="text" name="input" size="16" id="answer" />
        </div>
        <br />
        <div style="display: flex">
            <input
                type="button"
                name="one"
                value="1"
                onclick="calc.input.value += '1'"
            />
            <input
                type="button"
                name="two"

```

```
        value="2"
        onclick="calc.input.value += '2'"
    />
    <input
        type="button"
        name="three"
        value="3"
        onclick="calc.input.value += '3'"
    />
    <input
        type="button"
        class="operator"
        name="plus"
        value="+"
        onclick="calc.input.value += '+'"
    />
</div>
<br />
<div style="display: flex">
    <input
        type="button"
        name="four"
        value="4"
        onclick="calc.input.value += '4'"
    />
    <input
        type="button"
        name="five"
        value="5"
        onclick="calc.input.value += '5'"
    />
    <input
        type="button"
        name="six"
        value="6"
        onclick="calc.input.value += '6'"
    />
    <input
        type="button"
        class="operator"
        name="minus"
        value="-"
        onclick="calc.input.value += '-'"
    />
</div>
<br />
<div style="display: flex">
    <input
        type="button"
        name="seven"
        value="7"
        onclick="calc.input.value += '7'"
    />
```

```

<input
  type="button"
  name="eight"
  value="8"
  onclick="calc.input.value += '8'"
/>
<input
  type="button"
  name="nine"
  value="9"
  onclick="calc.input.value += '9'"
/>
<input
  type="button"
  class="operator"
  name="times"
  value="*"
  onclick="calc.input.value += '*'"
/>
</div>
<br />
<div style="display: flex">
  <input
    type="button"
    id="clear"
    name="clear"
    value=" C "
    onclick="calc.input.value=''"
  />
  <input
    type="button"
    name="zero"
    value="0"
    onclick="clac.input.value += '0'"
  />
  <input
    type="button"
    name="doit"
    value=" = "
    onclick="calc.input.value=eval(calc.input.value)"
  />
  <input
    type="button"
    class="operator"
    name="div"
    value="/"
    onclick="calc.input.value += '/'"
  />
</div>
<br />
</form>
</body>
</html>

```

Output:



Experiment – 4

Aim: Create an XML file to store details of students in a class and apply CSS for its presentation on web browser.

XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="4.css"?>
<students>
  <heading>Student Information- IT department(MAIT)</heading>
  <student>
    <name>Name -: Abhay Sharma</name>
    <RollNo>Roll-No -: 0512612671</RollNo>
    <group>Group -: I-1</group>
    <dob>D.O.B -: 03/10/1999</dob>
  </student>
  <student>
    <name>Name -: Bhavay Jain</name>
    <RollNo>Roll-No -: 8328324263</RollNo>
    <group>Group -: I-4</group>
    <dob>D.O.B -: 06/12/2000</dob>
  </student>
  <student>
    <name>Name -: Saksham Bansal</name>
    <RollNo>Roll-No -: 1771480311</RollNo>
    <group>Group -: I-4</group>
    <dob>D.O.B -: 04/06/1999</dob>
  </student>
  <student>
    <name>Name -: Shweta Sharma</name>
    <RollNo>Roll-No -: 8038939012</RollNo>
    <group>Group -: I-7</group>
    <dob>D.O.B -: 23/01/1999</dob>
  </student>
  <student>
    <name>Name -: Yashika Garg</name>
    <RollNo>Roll-No -: 8737309129</RollNo>
    <group>Group -: I-6</group>
    <dob>D.O.B -: 10/11/1999</dob>
  </student>
</students>
```

CSS file

```
students {  
    color: white;  
    background-color : gray;  
    width: 100%;  
}  
heading {  
    color: green;  
    font-size : 40px;  
    background-color : powderblue;  
}  
heading, name, RollNo, group, dob {  
    display : block;  
}  
name {  
    font-size : 25px;  
    font-weight : bold;  
}
```

Output:

Student Information- IT department(MAIT)

Name -: Abhay Sharma

Roll-No -: 0512612671

Group -: I-1

D.O.B -: 03/10/1999

Name -: Akshay Jain

Roll-No -: 8328324263

Group -: I-4

D.O.B -: 06/12/2000

Name -: Shivam Bansal

Roll-No -: 1771480311

Group -: I-4

D.O.B -: 04/06/1999

Name -: Shweta Sharma

Roll-No -: 8038939012

Group -: I-7

D.O.B -: 23/01/1999

Name -: Yashika Garg

Roll-No -: 8737309129

Group -: I-6

D.O.B -: 10/11/1999

Experiment – 5

Aim: Create a webpage to demonstrate the use of Date, String, Array and Math object in javascript.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      body {
        background-color : beige; width: 100vw;
        display: flex;
      }
      p {
        font-size: medium; margin: 10px;
      }
      .math {
        height: 100%; width: 25vw;
      }
      .dates { height: 100%; width: 25vw;
      }
      .Sting {
        height: 100%; width: 25vw;
      }
      .Array { height: 100%; width: 25vw;
        position: absolute;
      }
    </style>
  </head>
  <body>
    <div class="math">
      <h1>JavaScript Math Object</h1>
      <p id="first"></p>
      <p id="second"></p>
    </div>
    <div class="dates">
      <h1>JavaScript Date Object</h1>
      <p id="date"></p>
    </div>
    <div class="String">
      <h1>JavaScript String Object</h1>
      <p id="str"></p>
    </div>
    <div class="Array">
      <h1>JavaScript Array Object</h1>
      <p id="arr"></p>
    </div>
  </body>
</html>
```

```
</div>
```

```
<script>
```

```
document.getElementById("first").innerHTML =  
  "<p>Math.LN10: " +  
  Math.LN10 +  
  "</p>" +  
  "<p>Math.LOG2E: " +  
  Math.LOG2E +  
  "</p>" +  
  "<p>Math.Log10E: " +  
  Math.LOG10E +  
  "</p>" +  
  "<p>Math.SQRT2: " +  
  Math.SQRT2 +  
  "</p>" +  
  "<p>Math.SQRT1_2: " +  
  Math.SQRT1_2 +  
  "</p>" +  
  "<p>Math.LN2: " +  
  Math.LN2 +  
  "</p>" +  
  "<p>Math.E: " +  
  Math.E +  
  "</p>" +  
  "<p>Math.PI: " +  
  Math.PI +  
  "</p>";
```

```
document.getElementById("second").innerHTML =  
  "<p>Math.abs(-4.7): " +  
  Math.abs(-4.7) +  
  "</p>" +  
  "<p>Math.ceil(4.4): " +  
  Math.ceil(4.4) +  
  "</p>" +  
  "<p>Math.floor(4.7): " +  
  Math.floor(4.7) +  
  "</p>" +  
  "<p>Math.sin(90 * Math.PI / 180): " +  
  Math.sin((90 * Math.PI) / 180) +  
  "</p>" +  
  "<p>Math.min(0,150,30,20,-8,-200):</b>" +  
  Math.min(0, 150, 30, 20, -8, -200) +  
  "</p>" +  
  "<p>Math.random(): " +  
  Math.random() +  
  "</p>";
```

```
const d = new Date();  
document.getElementById("date").innerHTML =  
  "<p> Time:" +
```

```
d.getTime() +
"</p>" +
"<p> Full Year:" +
d.getFullYear() +
"</p>" +
"<p> Month:" +
d.getMonth() +
"</p>" +
"<p> Date:" +
d.getDate() +
"</p>" +
"<p> Hours:" +
d.getHours() +
"</p>" +
"<p> Minutes:" +
d.getMinutes() +
"</p>" +
"<p> Seconds:" +
d.getSeconds() +
"</p>";
```

```
let s = "abcdef";
document.getElementById("str").innerHTML =
"<p><b>String= abcdef</b></p>" +
"<p> Length:" +
s.length +
"</p>" +
"<p> Index of 'e':" +
s.indexOf("e") +
"</p>" +
"<p> Search 'c':" +
s.search("c") +
"</p>" +
"<p> Slice 2nd index:" +
s.slice(2) +
"</p>" +
"<p> Substring from 3 to last:" +
s.substring(3) +
"</p>" +
"<p> Concat 'ghi':" +
s.concat("ghi") +
"</p>" +
"<p> Replace 'a' with 'z':" +
s.replace("a", "z") +
"</p>";
```

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("arr").innerHTML =
"<p><b>Array= ['Banana', 'Orange', 'Apple', 'Mango']</b></p>" +
"<p> Length:" +
fruits.length +
"</p>" +
```

```
"<p> Remove 'Mango' from last:" +
fruits.pop() +
"<br> <b>[" +
fruits +
"]</b>" +
"</p>" +
"<p> Add 'Kiwi' at last:" +
fruits.push("Kiwi") +
"<br> <b>[" +
fruits +
"]</b>" +
"</p>" +
"<p> Add 'Kiwi' at first:" +
fruits.unshift("Kiwi") +
"<br> <b>[" +
fruits +
"]</b>" +
"</p>" +
"<p> Remove 'Kiwi' from first:" +
fruits.shift() +
"<br> <b>[" +
fruits +
"]</b>" +
"</p>" +
"<p> Slice a piece from 3rd element: <b>[" +
fruits.slice(2) +
"]</b>" +
"</p>";
</script>
</body>
</html>
```

Output:

JavaScript Math Object

Math.LN10: 2.302585092994046
Math.LOG2E: 1.4426950408889634
Math.Log10E: 0.4342944819032518
Math.SQRT2: 1.4142135623730951
Math.SQRT1_2: 0.7071067811865476
Math.LN2: 0.6931471805599453
Math.E: 2.718281828459045
Math.PI: 3.141592653589793
Math.abs(-4.7): 4.7
Math.ceil(4.4): 5
Math.floor(4.7): 4
Math.sin(90 * Math.PI / 180): 1
Math.min(0,150,30,20,-8,-200):-200
Math.random(): 0.971642103164378

JavaScript Date Object

Time:1623869510213
Full Year:2021
Month:5
Date:17
Hours:0
Minutes:21
Hours:50

JavaScript String Object

String= abcdef
Length:6
Index of 'e':4
Search 'c':2
Slice 2nd index:cdef
Substring from 3 to last:def
Concat 'ghi':abcdefghi
Replace 'a' with 'z':dbcdef

JavaScript Array Object

Array= ['Banana', 'Orange', 'Apple', 'Mango']
Length:4
Remove 'Mango' from last:Mango
[Banana,Orange,Apple]
Add 'Kiwi' at last:4
[Banana,Orange,Apple,Kiwi]
Add 'Kiwi' at first:5
[Kiwi,Banana,Orange,Apple,Kiwi]
Remove 'Kiwi' from first:Kiwi
[Banana,Orange,Apple,Kiwi]
Slice a piece from 3rd element: [Apple,Kiwi]

Experiment – 6

Aim: Create a registration form using HTML tags.

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

<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    body {
      font-family: Calibri, Helvetica, sans-serif;
      background-color: pink;
    }

    .container {
      padding: 50px;
      background-color: lightblue;
    }

    input[type=text],
    input[type=password],
    textarea {
      width: 100%;
      padding: 15px;
      margin: 5px 0 22px 0;
      display: inline-block;
      border: none;
      background: #f1f1f1;
    }

    input[type=text]:focus,
    input[type=password]:focus {
      outline: none;
    }

    div {
      padding: 10px 0;
    }

    hr {
      border: 1px solid #f1f1f1;
      margin-bottom: 25px;
    }

    .registerbtn {
      background-color: #4CAF50;
      color: white;
      padding: 16px 20px;
      margin: 8px 0;
```



```

        <textarea cols="80" rows="5" placeholder="Current Address" value="address" require
d>
</textarea>
<label for="email"><b>Email</b></label>
<input type="text" placeholder="Enter Email" name="email" required>

<label for="psw"><b>Password</b></label>
<input type="password" placeholder="Enter Password" name="psw" required>

<label for="psw-repeat"><b>Re-type Password</b></label>
<input type="password" placeholder="Retype Password" name="psw-repeat" required>
<button type="submit" class="registerbtn">Register</button>
</form>
</body>

</html>

```

Output:

Student Registration Form

Firstname
adas

Middlename:
adsas

Lastname:
sadsad

Course : Course ▾

Gender :
☒ Male
 ☐ Female
 ☐ Other

Phone :
 +91
 Phone no.

Current Address :

Email
 Enter Email

Password
 Enter Password

Re-type Password
 Retype Password

Register

Experiment – 7

Aim: Create a static web application using HTML, CSS and javascript with navigation links on topic " Career opportunities in Information Technology .

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      body {
        background-color: beige;
      }
      ul {
        margin: 20px;
      }
      li {
      }
      a {
        margin: 10px;
        font-size: 20px;

        color: blue;
      }
    </style>
  </head>
  <body>
    <h1>Career opportunities in Information Technology.</h1>
    <h2>
      Here is a list of resources for career opportunities in Information
      Technology
    </h2>
    <ul>
      <li>
        <a
          href="https://www.jagranjosh.com/careers/information-technology- 1528686112-1"
          >
            Career in Information Technology</a>
        </li>
      <li>
        <a
          href="https://www.indeed.com/career-advice/finding-a-job/types-of-it-jobs"
          >
            21 Different Types of IT Jobs To Explore</a>
        </li>
      </ul>
    </body>
  </html>
```

```

</li>
<li>
  <a href="https://www.careerizma.com/careers/information-technology/">
    Top educational qualifications for IT jobs</a>
  >
</li>

<li>
  <a
    href="https://www.bestcollegereviews.org/faq/jobs-qualified-bachelors-
degree- information-technology/"
  >
    What Jobs am I Qualified for with a Bachelor's Degree in Information
    Technology?</a>
  >
</li>
<li>
  <a
    href="https://thebestschools.org/careers/best-information-technology-jobs/"
  >
    The Best IT Jobs</a>
  >
</li>
</ul>
</body>
</html>

```

Output:

Career opportunities in Information Technology.

Here is a list of resources for career opportunities in Information Technology

- [Career in Information Technology](https://www.careerizma.com/careers/information-technology/)
- [21 Different Types of IT Jobs To Explore](https://www.bestcollegereviews.org/faq/jobs-qualified-bachelors-degree- information-technology/)
- [Top educational qualifications for IT jobs](https://thebestschools.org/careers/best-information-technology-jobs/)
- [What Jobs am I Qualified for with a Bachelor's Degree in Information Technology?](https://www.bestcollegereviews.org/faq/jobs-qualified-bachelors-degree- information-technology/)
- [The Best IT Jobs](https://thebestschools.org/careers/best-information-technology-jobs/)

Experiment - 8

Aim: Explain DTD, XSL, XSLT and Display Food menu using XML and XSLT.

DTD

DTD stands for Document Type Definition. It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes. Its main purpose is to define the structure of an XML document. It contains a list of legal elements and define the structure with the help of them.

Before proceeding with XML DTD, you must check the validation. An XML document is called "well-formed" if it contains the correct syntax. A well-formed and valid XML document is one which has been validated against DTD.

Let's take an example of well-formed and valid XML document. It follows all the rules of DTD.

employee.xml

1. `<?xml version="1.0"?>`
2. `<!DOCTYPE employee SYSTEM "employee.dtd">`
3. `<employee>`
4. `<firstname>vimal</firstname>`
5. `<lastname>jaiswal</lastname>`
6. `<email>vimal@javatpoint.com</email>`
7. `</employee>`

In the above example, the DOCTYPE declaration refers to an external DTD file. The content of the file is shown in the below paragraph.

employee.dtd

1. `<!ELEMENT employee (firstname,lastname,email)>`
2. `<!ELEMENT firstname (#PCDATA)>`
3. `<!ELEMENT lastname (#PCDATA)>`
4. `<!ELEMENT email (#PCDATA)>`

Description of DTD

<!DOCTYPE employee : It defines that the root element of the document is employee.

<!ELEMENT employee: It defines that the employee element contains 3 elements "firstname, lastname and email".

<!ELEMENT firstname: It defines that the firstname element is #PCDATA typed. (parse-able data type).

<!ELEMENT lastname: It defines that the lastname element is #PCDATA typed. (parse-able data type).

<!ELEMENT email: It defines that the email element is #PCDATA typed. (parse-able data type).

XSL

XSL is a language for expressing style sheets. An XSL style sheet is, like with [CSS](#), a file that describes how to display an XML document of a given type. XSL shares the functionality and is compatible with CSS2 (although it uses a different syntax). It also adds:

- A transformation language for XML documents: **XSLT**. Originally intended to perform complex styling operations, like the generation of tables of contents and indexes, it is now used as a general purpose XML processing language. XSLT is thus widely used for purposes other than XSL, like generating HTML web pages from XML data.
- Advanced styling features, expressed by an XML document type which defines a set of elements called **Formatting Objects**, and attributes (in part borrowed from CSS2 properties and adding more complex ones).

Styling requires a source XML documents, containing the information that the style sheet will display and the style sheet itself which describes how to display a document of a given type.

The following shows a sample XML file and how it can be transformed and rendered.

The XML file

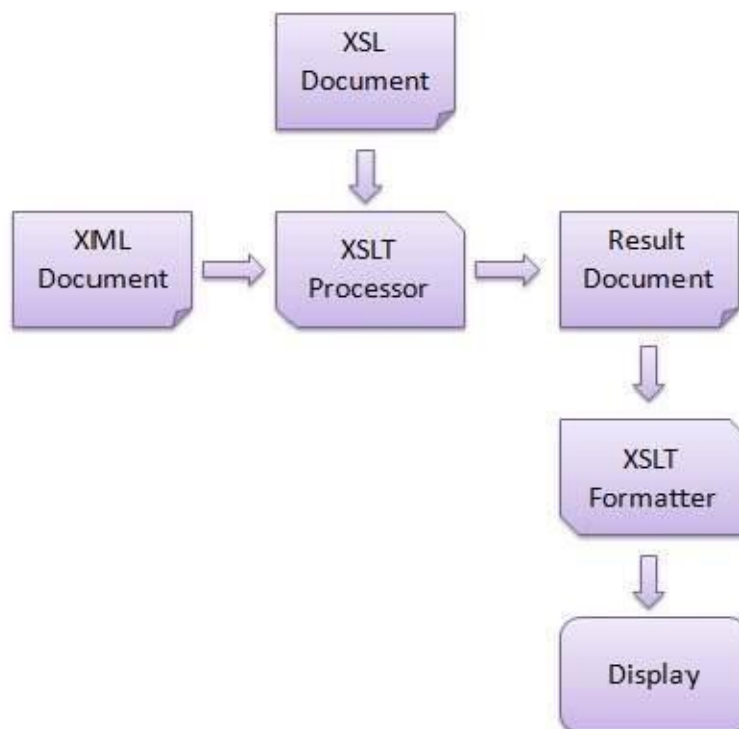
```
<scene>
  <FX>General Road Building noises.</FX>
  <speech speaker="Prosser">
    Come off it Mr Dent, you can't win
    you know. There's no point in lying
    down in the path of progress.
  </speech>
  <speech speaker="Arthur">
    I've gone off the idea of progress.
    It's overrated
  </speech>
</scene>
```

This XML file doesn't contain any presentation information, which is contained in the stylesheet. Separating the document's content and the document's styling information allows displaying the same document on different media (like screen, paper, cell phone), and it also enables users to view the document according to their preferences and abilities, just by modifying the style sheet.

XSLT

Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.



Advantages

Here are the advantages of using XSLT –

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

XML FILE

```
<?xml version = "1.0"?>

<?xml-stylesheet type = "text/xsl" href = "8.xsl"?>

<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>Rs. 250</price>
    <description>two of our famous Belgian Waffles with plenty of real maple syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>Rs. 240</price>
    <description>light Belgian waffles covered with strawberries and whipped cream</description>
    <calories>900</calories>
  </food>

  <food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>Rs. 300</price>
    <description>light Belgian waffles covered with an assortment of fresh berries and whipped cream</description>
    <calories>900</calories>
  </food>

  <food>
    <name>French Toast</name>
    <price>Rs. 200</price>
    <description>thick slices made from our homemade sourdough bread</description>
    <calories>600</calories>
  </food>

  <food>
    <name>Homestyle Breakfast</name>
    <price>Rs. 260</price>
    <description>two eggs, bacon or sausage, toast, and our ever-popular hash browns</description>
    <calories>950</calories>
  </food>
</breakfast_menu>
```


XSLT FILE

```
<?xml version = "1.0"?>

<!-- A simple XSLT transformation -->

<xsl:stylesheet version = "1.0"

    xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
    <xsl:output method = "html" omit-xml-declaration = "no" doctype-system =

        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"

        doctype-public = "-//W3C//DTD XHTML 1.0 Strict//EN"/>
    <xsl:template match = "/">
<style type="text/css">
    .space10 {margin-left:10px}
</style>

<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns="http://www.w3.
org/1999/xhtml">
    <body style="font-family:Arial;font-size:12pt;background-color:#EEEEEE;font-size:x-large">
        <div style="text-align:center;color:black;font-size:80px">Our Breakfast Menu</div>
        <BR></BR>
        <xsl:for-each select="breakfast_menu/food">
            <div style="background-color:lightblue;color:white;padding:4px">
                <span style="font-weight:bold">
                    <xsl:value-of select="name"/></span>
                <span class="space10">
                    The price is <xsl:value-of select="price"/></span>
            </div>

            <div style="margin-left:20px;margin-bottom:1em;font-size:1em">
                <xsl:value-of select="description"/>.
                <span style="font-style:italic; padding:20px; color:red" >
                    <xsl:value-of select="calories"/> (calories per serving)
                </span>
            </div>
        </xsl:for-each>
    </body>
</html>
    </xsl:template>
</xsl:stylesheet>
```

Output:

Our Breakfast Menu		
Belgian Waffles	The price is Rs. 250	
two of our famous Belgian Waffles with plenty of real maple syrup.		650 (calories per serving)
Strawberry Belgian Waffles	The price is Rs. 240	
light Belgian waffles covered with strawberries and whipped cream.		900 (calories per serving)
Berry-Berry Belgian Waffles	The price is Rs. 300	
light Belgian waffles covered with an assortment of fresh berries and whipped cream.		900 (calories per serving)
French Toast	The price is Rs. 200	
thick slices made from our homemade sourdough bread.		600 (calories per serving)
Homestyle Breakfast	The price is Rs. 260	
two eggs, bacon or sausage, toast, and our ever-popular hash browns.		950 (calories per serving)

Experiment – 9

Aim: Define Web Engineering? Explain characteristics of Web application and Draw a diagram to categorize web applications.

Web Engineering

A discipline concerned with the establishment and use of sound scientific, **engineering** and management principles and disciplined and systematic approaches to the successful development, deployment, and maintenance of high-quality **Web** Applications.

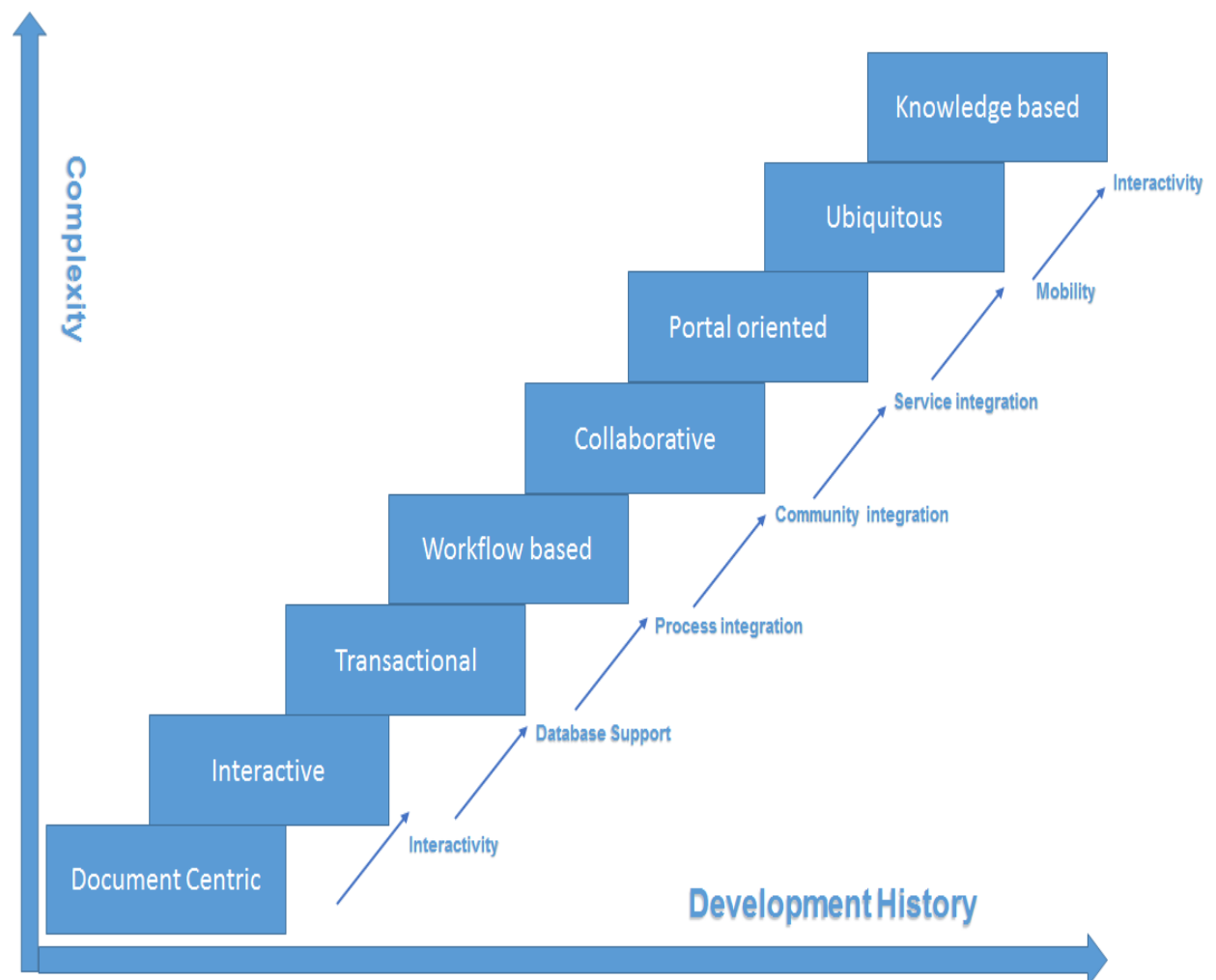
Web applications are software designed to execute on web with web specific resources. Web applications vary from small scale solution to large scale ERP. Web applications have evolved and turned to complex according to time. This complexity may be in terms of dynamic nature, use of multimedia, performance or in many other ways.

Traditional applications were simple, less complex, static content, limited use and level of security was minimal. There were no scope of timely update, had limited functionality and interactivity. Advanced web applications come with dynamic contents, contain large information, easy to integrate, schedule and plan. Perform better with more security.

Characteristics of web applications :

- Product related characteristics
- User related characteristics
- Development related characteristics
- Evolution related characteristics

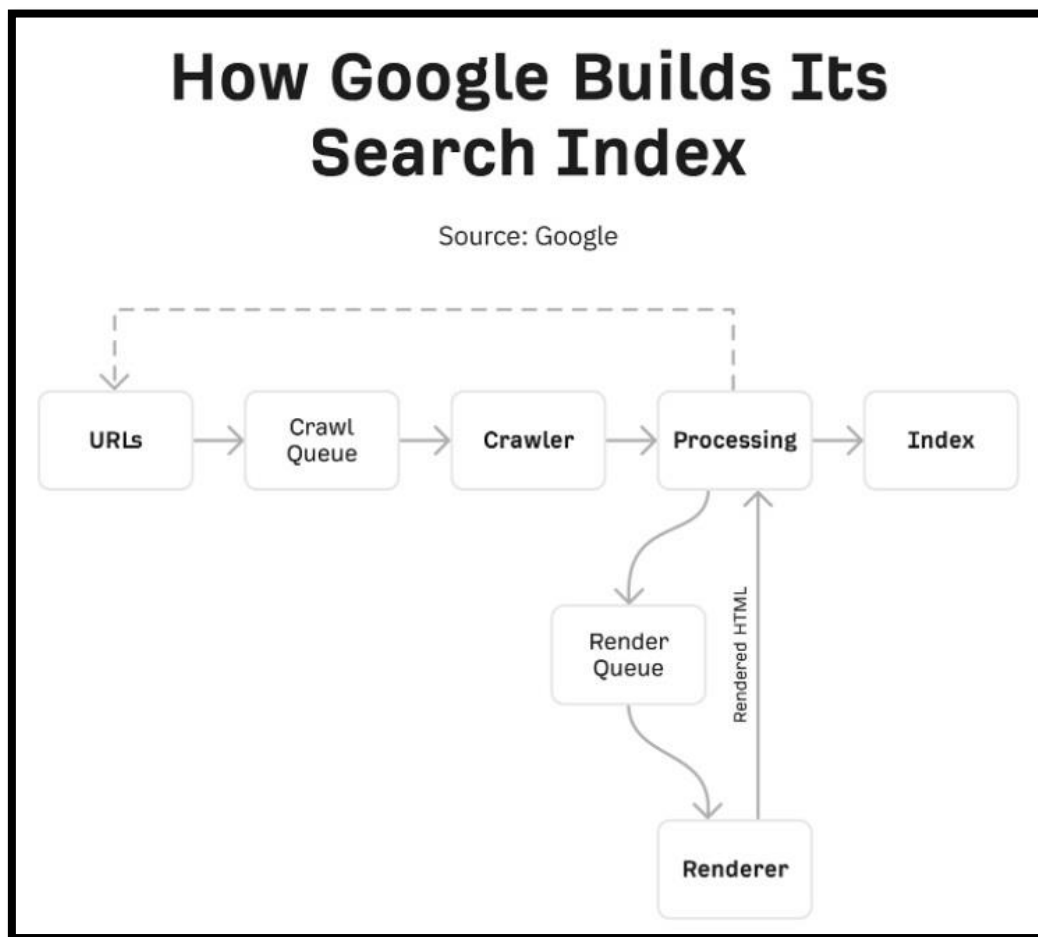
CATEGORIZE OF WEB APPLICATION:



Experiment – 10

Aim: Explain how search engine works and list out step by step process.

All search engines work using an approach to managing, ranking and returning search results. But a lot of people have no idea what is happening behind that search box when they type in their search queries. Search engines work by crawling the web using bots called spiders. These web crawler effectively follow links from page to page to find new content to add to the search index. When you use a search engine, relevant results are extracted from the index and ranked using an algorithm.



Step 1. URLs

Everything begins with a known list of URLs. Google discovers these through various processes, but the three most common ones are:

- **From backlinks**

Google already has an index containing trillions of web pages. If someone adds a link to one of your pages from one of those web pages, they can find it from there. You can see your website's backlinks for free using multiple sites

- **From sitemaps**

Sitemaps list all of the important pages on your website. If you submit your sitemap to Google, it may help them discover your website faster.

- **From URL submissions**

Google also allows submissions of individual URLs via Google Search Console.

Step 2. WebCrawling

Crawling is where a computer bot called a spider (e.g., Googlebot) visits and downloads the discovered pages. This is the means by which search engines can find out what is published out on the World Wide Web. Essentially, crawling is copying what is on web pages and repeatedly checking the multitude of pages. It's important to note that Google doesn't always crawl pages in the order they discover them. Google queues URLs for crawling based on a few factors, including:

- the PageRank of the URL
- how often the URL changes
- whether or not it's new

This is important because it means that search engines might crawl and index some of your pages before others. If you have a large website, it could take a while for search engines to fully crawl it.

Step 3. Processing

Processing is where Google works to understand and extract key information from crawled pages. Nobody outside of Google knows every detail about this process, but the important parts for our understanding are extracting links and storing content for indexing. Google has to render pages to fully process them, which is where Google runs the page's code to understand how it looks for users. That said, some processing occurs before and after rendering.

Step 4. Indexing

Once a spider has crawled a web page, the copy that is made is returned to the search engine and stored in a data center. Data centers are huge, purpose built collections of servers which act as a repository of all the copies of webpages being made by the crawlers. Google owns dozens of them dotted around the world, which it guards very closely and which are among the most hi-tech buildings in the world.

Indexing is the process of organizing the masses of data and pages so they can be searched quickly for relevant results to your search query. That's an important point. When you type a query into a search engine, you're not directly searching the internet for matching results. You're searching a search engine's index of web pages. If a web page isn't in the search index, search engine users won't find it. That's why getting your website indexed in major search engines like Google and Bing is so important.

Step 5. The Algorithm

Finally, we have a huge collection of web page copies which are being constantly updated and organized so we can quickly find what you are looking for. But we need a means by which they can be ranked in order of relevance to your search term – this is where the Algorithm comes into play.

The algorithm is a very complex and lengthy equation which calculates a value for any given site in relation to a search term. We don't know what the algorithm actually is, because search engines tend to keep this a closely guarded secret from competitors and from people looking to game the search engine to get to the top spots. That said, enough about the algorithm has been worked out to let SEOs advise website owners on how to improve their sites and SEO factors to move up in the rankings.

Experiment – 11

Aim: What are the main challenges in developing an organizational website.

Main challenges in developing an organizational website are :

1. Clearly defining your goals.

Clearly defining your goals and requirements will make or break your web app. Many of the other challenges that we'll talk about later, such as application performance and speed, can be almost entirely addressed by making thoughtful choices during the planning stage.

It all starts with your vision for your app. That affects everything that follows. The key requirements for an enterprise-level collaborative app are different than the requirements for a fitness app or a game.

There are a few key considerations in any development process:

- Who are your intended users?
- What experience do you want to give them?
- What are your must-have design features?
- What are your technical requirements?

Once you have defined all of these things, you can start to address them.

2. Choosing the right tech stack.

Throughout this section we're going to talk about tech stacks. They're a combination of programming languages, frameworks, servers, software and other tools used by developers. In essence, they're the set of technologies a web development and design agency uses to build a web or mobile app development.

You should always choose a tech stack that aligns with the problems you are trying to solve. For example, you probably won't need a complex tech stack for a simple web application, or a tech stack that helps you optimize for scalability when your user base will be a consistent size.

You should consider whether your tech stack is widely used in the industry. With industry-standard tech stacks, you will have a large pool of skilled developers to draw from for your initial build and future requirements.

Make sure to choose a tech stack that has thorough documentation. It's almost inevitable that you'll run into a troubleshooting problem at some point during development. Great documentation and support can save you time, money and hassle.

3. UX

User experience (UX) encapsulates the reactions, perceptions, and feelings your users experience while engaged in your application. It's the feeling of ease and simplicity that you get from great design. It's also the frustration that you feel when interacting with poor design.

That's why it's important to think about the overall impression you want to leave on your users before you start making detailed decisions about how to build it.

If you deliver effectively to the deeper needs of all your users – providing pleasure, engagement, and an overall emotional appeal – your application can become a central part of your users' day-to-day lives and deliver to your business goals.

4. UI & simplistic design

User Interface (UI) design includes all the visual elements your users interact with on your web application. It is everything that your users see on their screens and everything they click on to guide them through the experience.

Great UI design certainly makes your application visually appealing, but it goes beyond just simple aesthetics. The goal is to make the actual user experience simple and accessible—and usable. This means using only a targeted, purposeful selection of copy and content, making clear the options your users have throughout the experience and ensuring information is readily available at each step.

Intuitive UI typically involves:

- Clear navigation
- Engaging visuals
- Easy-to-read typography

5. Performance and speed

No user likes slow load times. And they can have real consequences for your business. If your application is slow, users won't wait. They'll leave. This is the reality of web application development issues today. You may only get a single chance to hook a user on your product.

So, if you know that you're building an application with a lot of content (e.g. videos), you should outline that upfront so your developers can build a more robust application to ensure performance. Likewise, be clear about any intentions you have to scale your application rapidly. You don't want your application to slow down if it makes a splash in the market or you see periodic surges in traffic.

This kind of planning for the future will ensure your initial app launch can deliver the kind of speed and performance you want for your early, often-critical, users. It will also mean that your initial build will provide the foundation you need for future business growth.

6. Scalability

The challenge of scalability relates to how you want your application to develop over time. If you want your application built right today, you'll need to know as much as possible about what you need it to do in the future.

Your application might include fairly lean content at launch but, a year or two in, you could be planning a detailed, expansive content-rich experience. This is where extensibility comes in. Extensibility is when an application is initially designed to incorporate new capabilities and functionality in the future. This can be integrated right from the start of the development process. If you can articulate a long-term vision for your app, it can be built to grow and evolve over time.

Planning for scalability helps you manage different user types, handle increased traffic, and an expansion of, for example, an ecommerce shop. Overall, it's valuable to prioritize scalability because it can improve your user experience, fulfill your business goals and extend the lifespan of your application.

7. Web security threats

There are a number of things to prioritize so your application and your users are secure. Choosing the right development infrastructure is one. Make sure that

the infrastructure you build on has enough security services and options so your developers can implement the proper security measures for your application.

SSL certificates are a global standard security technology that enables encrypted communication between your web browser and server. When integrated in your app, they enhance its security and eliminate the chance it is flagged as unsecure by web browsers. SSL certificates also help protect credit-card numbers in ecommerce transactions and other sensitive user information like usernames, passwords and email addresses. In essence, SSL allows for a private “conversation” just between the two intended parties and hides sensitive information from hackers and identity thieves.

Creating robust password requirements and multi-factor authentication for your users are also effective security measures. More complex passwords are less likely to be hacked. Multi-factor authentication, where your users take multiple steps to confirm their identities, also gives your application an added layer of protection.

Experiment – 12

Aim: Explain the steps in designing and developing a website and explain characteristics of a good website design?

Steps in designing and developing a website

1. **Goal identification:** Where I work with the client to determine what goals the new website needs to fulfill. I.e., what its purpose is.
2. **Scope definition:** Once we know the site's goals, we can define the scope of the project. I.e., what web pages and features the site requires to fulfill the goal, and the timeline for building those out.
3. **Sitemap and wireframe creation:** With the scope well-defined, we can start digging into the sitemap, defining how the content and features we defined in scope definition will interrelate.
4. **Content creation:** Now that we have a bigger picture of the site in mind, we can start creating content for the individual pages, always keeping search engine optimization (SEO) in mind to help keep pages focused on a single topic. It's vital that you have real content to work with for our next stage:
5. **Visual elements:** With the site architecture and some content in place, we can start working on the visual brand. Depending on the client, this may already be well-defined, but you might also be defining the visual style from the ground up. Tools like style tiles, moodboards, and element collages can help with this process.
6. **Testing:** By now, you've got all your pages and defined how they display to the site visitor, so it's time to make sure it all works. Combine manual browsing of the site on a variety of devices with automated site crawlers to identify everything from user experience issues to simple broken links.
7. **Launch:** Once everything's working beautifully, it's time to plan and execute your site launch! This should include planning both launch timing and communication strategies — i.e., when will you launch and how will you let the world know?

Characteristics of Good Website:

1. Well Designed and Functional

Your site reflects your company, your products, your services and ultimately your brand. So it's important to be visually appealing, polished and professional. Allow white space, uncluttered layouts with quality photographs and graphics look and let your message shine through.

Equally important, the site must work quickly, correctly and as expected. Build to web standards, proofread rigorously and test regularly for problems with speed or functionality. Every page should always be fast and functional, because any of them could be a potential customer's first or only impression. Broken, slow, or poorly constructed areas will leave your visitors frustrated and encourage them to leave.

2. Easy to Use

Site visitors are always in a hurry. Don't make them work for information. User Experience (UX) plays a key role in helping visitors use, understand and stay on your website. Create obvious, logical navigation with clear hierarchy. Use consistent layouts and visual cues for functionality across the site.

Your site should satisfy both 'searchers'—coming for something specific, and 'browsers' - just looking. Help users accomplish their tasks quickly with onsite search, and keep them engaged by suggesting related content and minimizing dead ends.

3. Optimized for Mobile

Today there are no excuses, your site must look great and work well on any platform. The growth of mobile and tablet devices is not slowing down and you just don't know what your next visitor will be using. Optimizing for mobile will improve both the experience of your visitors and your *SEO* Rankings.

4. Fresh, Quality Content

Be succinct, interesting and new. Use language that makes sense to your audience—avoid jargon, corporate speak and acronyms. Explain your "Why." Visitors have short attention spans: spell correctly, be accurate, be relevant and update regularly. Blogs and social media updates are great ways to add fresh content, which keeps visitors returning and helps *SEO* strategy.

5. Readily accessible contact and location

Your audience won't chase you down. Make it easy to engage, offer multiple points of contact: phone, email, social media and maybe an easy-to-use contact form. A Google map is a bonus. Above all, ensure that this information is readily available on an easy-to-find contact page—if not every page of your site.

6. Clear calls to action

If your site asks nothing of visitors, they will surely do nothing. What is the purpose of your site? Is that purpose clear to visitors? Even informational sites want visitors to read and share articles, follow the company on social media, download toolkits, join mailing lists or learn more about the organization. Include an ask on each page.

7. Optimized for Search and the Social Web

It's not enough to build a nice looking website that's easy to use. It needs to earn traffic. Otherwise, all that effort in design, UX and content development will be for naught. There are hundreds of rules and guidelines for effective search engine optimization, so here are a few to start with:

- Use page titles and meta tags on every page and alt tags on every image.
- Optimize content on your site to align with words real people search for.
- Use keywords appropriately in content and links.
- Use Cascading Style Sheets for layout and keep your HTML code clutter-free.

Experiment – 13

Aim: What is SEO and Explain steps to improve Website ranking using SEO?

SEO stands for “search engine optimization.” In simple terms, it means the process of improving your site to increase its visibility when people search for products or services related to your business in Google, Bing, and other search engines. The better visibility your pages have in search results, the more likely you are to garner attention and attract prospective and existing customers to your business.

SEO is a fundamental part of digital marketing because people conduct trillions of searches every year, often with commercial intent to find information about products and services. Search is often the primary source of digital traffic for brands and complements other marketing channels. Greater visibility and ranking higher in search results than your competition can have a material impact on your bottom line. In sum, SEO is the foundation of a holistic marketing ecosystem.

Ways to improve Website Ranking using SEO

1. Keyword research

Use search data and research your competition to find the best targets for each page on your website. Depending on how big your website is and the resources you have available, start with the pages that have the most content, and are helpful for your customers. Some examples would be your top one or two product/service pages or a well-written blog post that provides detailed information your customers are looking for. Strategically select which phrases you want your pages to rank well for. The pages you choose should have a clear message that is focused on one topic. Remember, don't use the insider terminology you use, focus on what your everyday customer may search.

2. Write informative copy that people seek

You know your industry better than anyone. Write content that is helpful and also includes keywords and synonyms that get your website found by Google and other search engines. Include the keywords throughout the page to improve your search engine ranking, but don't go overboard with keyword stuffing. Make sure it's still readable for your visitors.

3. Title your page to help search engines and users

Search engines read the title tag, or meta title, as an important indicator of what the page is about. Your title tag is also the blue link everyone sees on a search engine results page, so be sure to include the keywords users search for. This title is different from the bold heading everyone sees on your webpage, which is typically the heading tag described in the next step. Therefore, if it helps get more clicks, you can write one title for the search engine results page (SERP) and a different heading for your webpage.

4. Build pages with HTML header tags

The first heading, or H1 tag, is the most important heading on the page. The H1 tag is typically the first thing a visitor reads on your webpage. Header tags are commonly used for styling but should instead be used for content organization. Making use of subheadings is one of the best ways to improve search engine ranking. For search engines, heading tags have more weight than regular copy and make skimming the page easy on a user. If you're having trouble editing your heading tags, you'll likely need a developer to update the page.

5. Give your URLs a purpose

The page URL is often ignored, so don't let that be the case with you! The URLs should also benefit from keyword research. If you use a CMS like WordPress, the name of the page will appear in the URL. Before you launch the page, review what you named it and check the URL. You can update it to include the keywords you found in step 2. If you've already published the page, be sure to redirect visitors from your old page to the new page.

6. Internally link to other pages

As you're writing copy for the page, you'll often find yourself referencing other topics that you've already published. Internal linking is selecting a phrase and adding a hyperlink to the blog post or page you are referencing. These links will get some users to click through and read your other pages, but more importantly, search engines will crawl these links too. The keywords you hyperlink are called anchor text. This gives the user and search engines context about the page you link to so it is important to use meaningful keywords in the hyperlink.

7. Use ALT tags with all your images

Images visually support the message on the page. Users love them, and search engines do too. That's why you should always use descriptive (keyword-rich) ALT tags for all your images. ALT tags also keep your site accessible, which could help you avoid an ADA lawsuit.

8. Mobile-friendly design and page speed

It's no secret that more and more users are coming to your site from mobile devices. Make sure your website design is mobile-friendly. Also, optimize the images, code, and content to load each page quickly. Visitors will leave slow websites and search engines will notice. This is why having a fast website will improve your search engine ranking. You need great content, but website design and development are important when asking how to make your website rank.

9. Limit any duplicate content

Search engines like new and unique content. Find any duplicate content and make it substantially different or remove it altogether. There's no advantage to saying the same thing over and over, and Google will penalize you for it. There are two likely scenarios where you'll have duplicate content.

- You've created a lot of content over the years and forget that you, or someone else, already wrote a page or blog post about the same topic. This probably isn't a complete duplicate, but if the content significantly overlaps, then you are better off combining it into one strong page.
- You don't even realize it but your website has multiple versions of the same page. This could be from someone accidentally copying a page or you've created multiple versions for different languages or tracking purposes.

10. Create an XML Sitemap and monitor crawl errors

Submit Sitemap

Submitting a sitemap will help get pages indexed by search engines faster. If you've verified Google Search Console for your site in the first step, then you can submit your sitemap with Google. Bing also has a webmaster tools dashboard where you can submit a sitemap.

Crawl Errors and Warnings

Search engines also crawl websites through links on pages. Be sure to monitor any errors or broken links and fix them or create redirects to ensure your site has the best chance to rank well. There are free broken link checkers you'll find on Google.

EXPERIMENT-14

AIM: Write short notes on CGI and Perl.

PERL:

Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

Perl is an interpreted language, which means that your code can be run as is, without a compilation stage that creates a non-portable executable program.

Traditional compilers convert programs into machine language. When you run a Perl program, it's first compiled into a byte code, which is then converted (as the program runs) into machine instructions. So it is not quite the same as shells, or Tcl, which are **strictly** interpreted without an intermediate representation.

Perl Features:

- It has a very simple Object-oriented programming syntax.
- It is easily extendible as it supports 25,000 open source modules.
- It supports Unicode.
- It includes powerful tools to process text to make it compatible with mark-up languages like HTML, XML.
- It supports third party database including Oracle, MySQL and many others.

CGI:

Common Gateway Interface (CGI) is nothing more than a protocol which defines the interaction of web servers with some executable programs in order to produce dynamic web pages. Basically, it shows how the web server sends information to the program and the program sends the information back to the web server which in turn can be sent back to the browser. Between web servers and external programs, it is considered as the standard programming interface.

CGI programs can send many types of data or media, like documents, images, audio clips, etc. Most Web sites use CGI with fields for input and deals of dynamic content on the Web is done mainly using CGI. It is a method through which a web server can get/send the data from/to databases, documents, and other programs respectively, and then present that data to viewers through the web.

Features of CGI:

- It is a very well defined and supported standard.
- CGI scripts are generally written in either Perl, C, or maybe just a simple shell script.
- CGI is a technology that interfaces with HTML.
- CGI is the best method to create a counter because it is currently the quickest
- CGI standard is generally the most compatible with today's browsers

Advantages of CGI:

- The advanced tasks are currently a lot easier to perform in CGI than in Java.
- It is always easier to use the code already written than to write your own.
- CGI specifies that the programs can be written in any language, and on any platform, as long as they conform to the specification.
- CGI-based counters and CGI code to perform simple tasks are available in plenty.

EXPERIMENT-15

AIM: Advantages and Disadvantages of Javascript.

Advantages of JavaScript:

- Regardless of where you host JavaScript, it always gets executed on client environment to save lots of bandwidth and make execution process fast.
- In JavaScript, XMLHttpRequest is an important object that was designed by Microsoft. The object call made by XMLHttpRequest as a asynchronous HTTP request to the server to transfer the data to both sides without reloading the page
- The biggest advantage to JavaScript having a ability to support all modern browsers and produce an equivalent result.
- Global companies support community development by creating projects that are important. An example is Google (created Angular framework) or Facebook (created the React.js framework).
- JavaScript is employed everywhere on the web .
- JavaScript plays nicely with other languages and may be utilized in an enormous sort of applications.
- There are many open source projects that provide a useful help at the developer's add JavaScript.
- There are many available courses within the field of JavaScript, because of which you'll quickly and simply expand your knowledge of this programming language.
- It is not difficult to start working in JavaScript. For this reason, many of us prefer to start their adventure with the IT sector from learning this language.
- It gives the power to make rich interfaces.
- There are some ways to use JavaScript through Node.js servers. It is possible to develop a whole JavaScript app from front to back using only JavaScript.

Disadvantages of JavaScript:

- This may be difficult to develop large applications, although you'll also use the TypeScript overlay.
- This applies to larger front-end projects. The configuration is often a tedious task to the amount of tools that require to figure together to make an environment for such a project. This is often directly associated with the library's operation.
- The main problem or disadvantage in JavaScript is that the code is always visible to everyone anyone can view JavaScript code.
- No matter what proportion fast JavaScript interpret, JavaScript DOM (Document Object Model) is slow and can be a never fast rendering with HTML.
- If the error occurs in the JavaScript, it can stop to render the whole website. Browsers are extremely tolerant of JavaScript errors.
- JavaScript is usually interpreted differently by different browsers. This makes it somewhat complex to read and write cross-browser code.
- Though some HTML editors support debugging, it's not as efficient as other editors like C/C++ editors. Hence difficult for the developer to detect the matter.
- This continuous conversions takes longer in conversion of number to an integer. This increases the time needed to run the script and reduces its speed.

EXPERIMENT-16

AIM: Create a webpage to demonstrate Alert, Prompt and confirm dialog box in javascript.

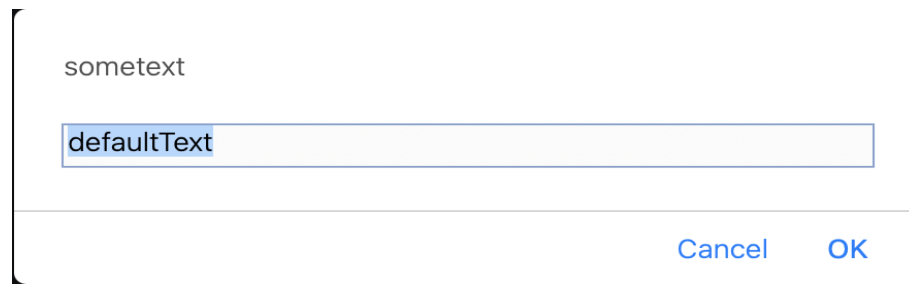
```
<html>
  <head>
    <title>Demo</title>
  </head>
  <body>
    <script>
      function getAlert() {
        alert("Alert!");
      }
      function getPrompt() {
        window.prompt("sometext", "defaultText");
      }
      function getConfirm() {
        window.confirm("Do you Agree?");
      }
    </script>
    <button onclick="{getAlert()}">Alert Box</button>
    <button onclick="{getConfirm()}">Confirm Box</button>
    <button onclick="{getPrompt()}">Window Prompt</button>
  </body>
</html>
```

OUTPUT:

AlertBox:



Prompt Box:



A screenshot of a JavaScript prompt box. The box has a title bar with a close button. Inside, the text "sometext" is displayed above a text input field containing "defaultText". At the bottom right, there are two buttons labeled "Cancel" and "OK".

Confirm Box:



A screenshot of a JavaScript confirm box. The box has a title bar with a close button. Inside, the text "Do you Agree?" is displayed. At the bottom right, there are two buttons labeled "Cancel" and "OK".

Alert Box Confirm Box Window Prompt



EXPERIMENT-17

AIM: Create a game “Lucky seven” using Javascript.

HTML:

```
<!DOCTYPE html>
<html>

<head>
  <title>Lucky Sevens</title>
  <link href="css/LuckySevensStyles.css" rel="stylesheet" type="text/css">
  <script type="text/javascript" src="js/LuckySevensV1.js"></script>
</head>

<body onload="hideResults()">
  <!-- BET STUFFS -->
  <div id="gameDiv">
    <h1>Lucky Sevens</h1>
    <span>Starting Bet:<input type="number" name="Starting Bet" id="betInput
" placeholder="$0.00"></span>
    <br />
    <button onclick="play()" id="playButton">Play</button>
  </div>

  <!-- RESULT STUFFS -->

  <div id="results">
    <table>
      <caption>
        <h2>Results</h2>
      </caption>
      <tr>
        <th>Starting Bet</th>
        <th id="resultsBet"></th>
      </tr>
      <tr>
        <td>Total Rolls Before Going Broke</td>
        <td id="resultsRollCounter"></td>
      </tr>
      <tr>
        <td>Highest Amount Won</td>
        <td id="resultsHighestHeld"></td>
      </tr>
      <tr>
        <td>Roll Count at Highest Amount Won</td>
        <td id="resultsRollsFromHighest"></td>
      </tr>
    </table>
```



```
    </div>
</body>

</html>
```

JavaScript:

```
function hideResults() {
    document.getElementById("results").style.display = "none";
}

function play() {
    var startingBet = document.getElementById("betInput").value;
    var bet = startingBet;
    var dice1 = Math.floor(Math.random() * 6) + 1;
    var dice2 = Math.floor(Math.random() * 6) + 1;
    var diceRoll = dice1 + dice2;
    var betsArray = [];

    while (bet > 0) {
        if (diceRoll != 7) {
            bet -= 1;
        } else {
            bet += 4;
        }
        betsArray.push(bet);
    }

    var rollCounter = betsArray.length;
    var highestAmount = Math.max.apply(Math, betsArray);
    var highestPosition = betsArray.indexOf(highestAmount);
    var rollsFromHighest = rollCounter - highestPosition;

    function showResults() {
        document.getElementById("results").style.display = "inline";
        document.getElementById("playButton").innerHTML = "Play Again";
        document.getElementById("resultsBet").innerHTML = "$" + startingBet + ".00";
        document.getElementById("resultsRollCounter").innerHTML = rollCounter;
        document.getElementById("resultsHighestHeld").innerHTML =
            "$" + highestAmount + ".00";

        document.getElementById("resultsRollsFromHighest").innerHTML =
            rollsFromHighest;
    }

    showResults();
}
```

OUTPUT:

Lucky Sevens

Starting Bet:

[Play Again](#)

Results

Starting Bet	\$10.00
Total Rolls Before Going Broke	10
Highest Amount Won	\$9.00
Roll Count at Highest Amount Won	10

EXPERIMENT-18

AIM: Create a jsp page for fetching request data and display with current date as response.

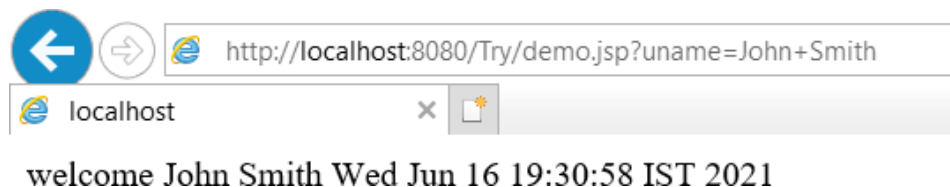
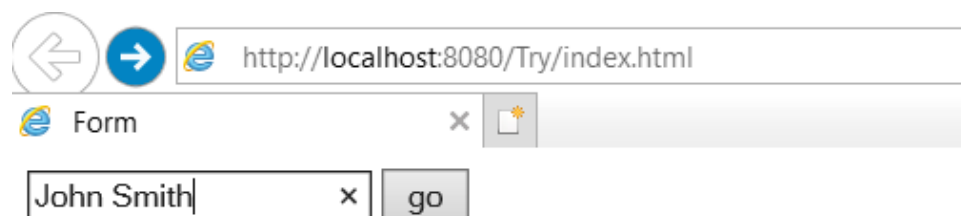
Index.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Guru Form</title>
  </head>
  <body>
    <form action="demo.jsp">
      <input type="text" name="uname" />
      <input type="submit" value="go" /><br />
    </form>
  </body>
</html>
```

Demo.jsp

```
<% String name=request.getParameter("uname"); Date date = new Date();
out.print("welcome "+name + date.toString()); %>
```

OUTPUT:



Experiment –19

Aim: Create a servlet for fetching request data and display with current date as response.

Index.html

```
<html>
  <head>
    <title>Form</title>
  </head>
  <body>
    <form action="try1" method="GET">
      <input type="text" name="uname" />
      <input type="submit" value="go" /><br />
    </form>
  </body>
</html>
```

Try1.java

```
import java.io.*;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;

public class try1 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

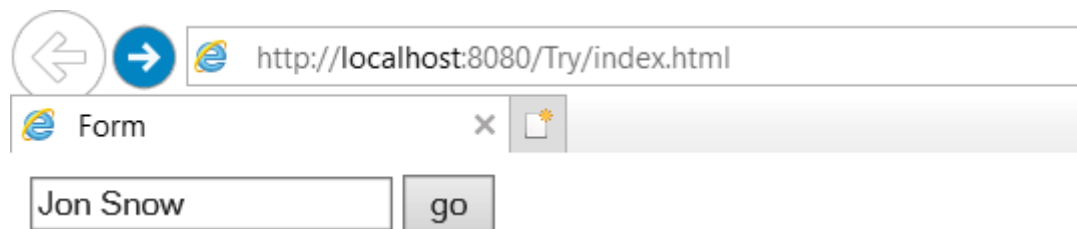
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String name = request.getParameter("uname");

        String title = "Welcome " + name + "\n";
        Date date = new Date();
        String docType = "<!doctype html public \"-
//w3c//dtd html 4.0 \" + \"transitional//en\">\n";

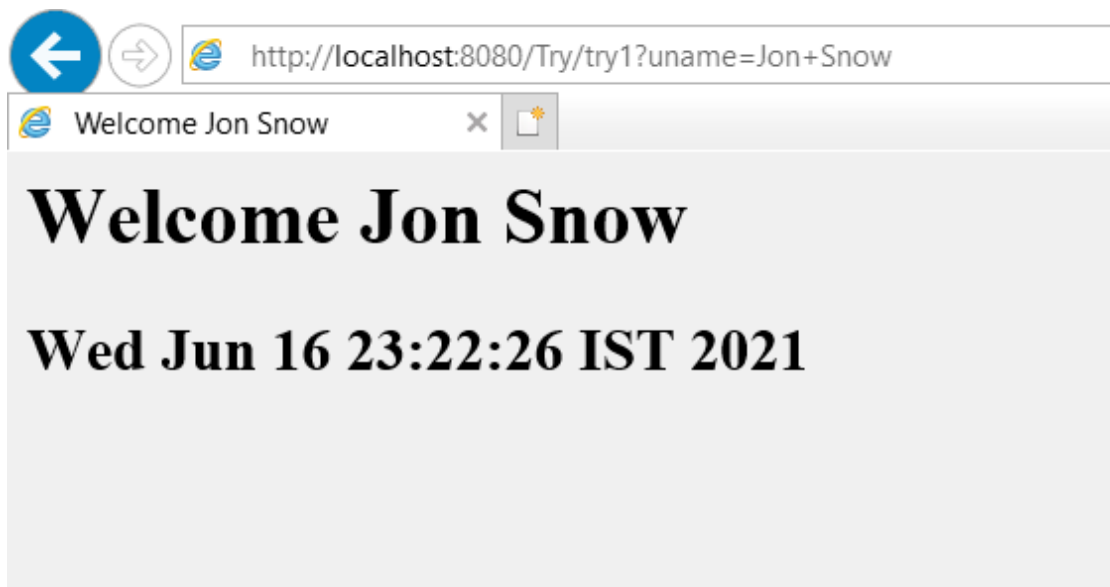
        out.println(docType + "<html>\n" +

            "<head><title>" + title + "</title></head>\n" + "<body bgcolor="
            + "\"#f0f0f0\">\n" + "<h1>" + title
            + "</h1>\n" + "<h2>" + date.toString() + "</h2>\n" + "</body>
</html>");
    }
}
```

Output :



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/Try/index.html`. The browser tab is titled "Form". Below the address bar, there is a text input field containing the text "Jon Snow" and a button labeled "go".



Experiment 20

Aim: Create a jsp page Form Validation using java-script (blank text field and email).

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

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Document</title>
</head>
<style>
  body {
    background-color: black;
    color: white;
    text-align: center;
  }

  form {
    margin-top: 50px;
  }
</style>

<body>
  <script>
    function validateform() {
      var name = document.frm.name.value; name = name.trim();
      var password = document.frm.password.value; if (name == null || name == "") {
        alert("Name can't be blank"); return false;
      } else if (password.length < 6) {
        alert("Password must be at least 6 characters long."); return false;
      }
    }
  </script>

  <body>
    <h1>JSP Form validation using JavaScript</h1>
    <form name="frm" method="post" action="http://www.javatpoint.com/javascriptpages/valid.jsp"
      onsubmit="return validateform()">
      Name : <input type="text" name="name"><br /><br />
      Password : <input type="password" name="password"><br /><br />
      <input type="submit" value="Register">
    </form>
  </body>
</html>
```

JSP Form validation using JavaScript

Name :

Password :

You are valid user

Thanks for [visiting our](#) site

Experiment -21

Aim: What are Cookies? Explain their uses and Create a Web app and use Cookies to manage the sessions.

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

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Document</title>
</head>

<body style="background-color: lightgray;">
  <h1>Web Development</h1>
  <p style="font-size:x-large;">
    Web development is the work involved in developing a Web site for the
    Internet (World Wide Web) or an intranet (a private network).[1] Web
    development can range from developing a simple single static page of plain
    text to complex web applications, electronic businesses, and social
    network services. A more comprehensive list of tasks to which Web
    development commonly refers, may include Web engineering, Web design, Web
    content development, client liaison, client-side/server-side scripting,
    Web server and network security configuration, and e-commerce development.
    Among Web professionals, "Web development" usually refers to the main
    non-design aspects of building Web sites: writing markup and coding.[2]
    Web development may use content management systems (CMS) to make content
    changes easier and available with basic technical skills. For larger
    organizations and businesses, Web development teams can consist of
    hundreds of people (Web developers) and follow standard methods like Agile
    methodologies while developing Web sites. Smaller organizations may only
    require a single permanent or contracting developer, or secondary
    assignment to related job positions such as a graphic designer or
    information systems technician. Web development may be a collaborative
    effort between departments rather than the domain of a designated
    department. There are three kinds of Web developer specialization:
    front-end developer, back-end developer, and full-stack developer.
    Front-end developers are responsible for behavior and visuals that run in
    the user browser, while back-end developers deal with the servers.
  </p>
  <br />
  

  <h1>Practical Web Development</h1>
  <h3 style="font-size: x-large;">Basic</h3>
  <p style="font-size: x-
  large;">In practice, many Web developers will have basic interdisciplinary skills / roles
  ,
```



```

        including:
<ul style="font-size: x-large;">
  <li>Graphic design / Web design</li>
  <li>Information architecture and copywriting/copyediting with Web usability, accessibility and search engine optimization in mind</li>
  <li>Mobile responsiveness</li>
</ul>
</p>
<h3 style="font-size: x-large;">Testing</h3>
<p style="font-size: x-large;">Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements contrary to the actual requirements. The extent of testing varies greatly between organizations, developers, and individual sites or applications.</p>
</body>
</html>

```

CookiesDemo.html

```

<!DOCTYPE html>
<html>
  <head> </head>
  <body>
    <select id="color" onchange="display()">
      <option value="Select Color">Select Color</option>
      <option value="yellow">Yellow</option>
      <option value="green">Green</option>
      <option value="red">Red</option>
    </select>
    <script type="text/javascript">
      function display() {
        var value = document.getElementById("color").value;
        if (value != "Select Color") {
          document.bgColor = value;
          document.cookie = "color=" + value;
        }
      }
      window.onload = function () {
        if (document.cookie.length != 0) {
          var array = document.cookie.split("=");
          document.getElementById("color").value = array[1];
          document.bgColor = array[1];
        }
      };
    </script>
  </body>

```

</html>

Output:

Cookies

127.0.0.1:5500/20.html

What are Cookies?

Cookies are data, stored in small text files, on your computer. When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user. Cookies were invented to solve the problem "how to remember information about the user".

1. When a user visits a web page, his/her name can be stored in a cookie.
2. Next time the user visits the page, the cookie "remembers" his/her name.

Cookies are saved in name-value pairs like:
username = John Smith

When a browser requests a web page from a server, cookies belonging to the page are added to the request. This way the server gets the necessary data to "remember" information about users.

Create a Cookie with JavaScript

JavaScript can create, read, and delete cookies with the document.cookie property. With JavaScript, a cookie can be created like this:
`document.cookie = "username=John Smith";`

You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed:
`document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC";`

With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page:
`document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";`

Read a Cookie with JavaScript

With JavaScript, cookies can be read like this:
`var x = document.cookie;`

Change a Cookie with JavaScript

With JavaScript, you can change a cookie the same way as you create it:
`document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";`

The old cookie is overwritten.

Delete a Cookie with JavaScript

Deleting a cookie is very simple. You don't have to specify a cookie value when you delete a cookie. Just set the expires parameter to a past date:
`document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";`

Live Demo

Here if we click on the button a web page open where user selects a colour and that colour is saved in cookies. Even after refreshing or closing the browser colour remains the same.

Click here to go to live demo

127.0.0.1:5500/cookiesDemo.html

127.0.0.1:5500/cookiesDemo.html

Select Color

Yellow

Green

Red

Experiment -22

Aim : Create a webpage using HTML 5 tags

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

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Document</title>
</head>

<body style="background-color: lightgray;">
  <h1>Web Development</h1>
  <p style="font-size:x-large;">
    Web development is the work involved in developing a Web site for the
    Internet (World Wide Web) or an intranet (a private network).[1] Web
    development can range from developing a simple single static page of plain
    text to complex web applications, electronic businesses, and social
    network services. A more comprehensive list of tasks to which Web
    development commonly refers, may include Web engineering, Web design, Web
    content development, client liaison, client-side/server-side scripting,
    Web server and network security configuration, and e-commerce development.
    Among Web professionals, "Web development" usually refers to the main
    non-design aspects of building Web sites: writing markup and coding.[2]
    Web development may use content management systems (CMS) to make content
    changes easier and available with basic technical skills. For larger
    organizations and businesses, Web development teams can consist of
    hundreds of people (Web developers) and follow standard methods like Agile
    methodologies while developing Web sites. Smaller organizations may only
    require a single permanent or contracting developer, or secondary
    assignment to related job positions such as a graphic designer or
    information systems technician. Web development may be a collaborative
    effort between departments rather than the domain of a designated
    department. There are three kinds of Web developer specialization:
    front-end developer, back-end developer, and full-stack developer.
    Front-end developers are responsible for behavior and visuals that run in
    the user browser, while back-end developers deal with the servers.
  </p>
  <br />
  

  <h1>Practical Web Development</h1>
  <h3 style="font-size: x-large;">Basic</h3>
  <p style="font-size: x-
  large;">In practice, many Web developers will have basic interdisciplinary skills / roles
  ,
  including:
  <ul style="font-size: x-large;">
    <li>Graphic design / Web design</li>
```

```

        <li>Information architecture and copywriting/copyediting with Web usability, accessibility and search engine optimization in mind</li>
        <li>Mobile responsiveness</li>
    </ul>
</p>
<h3 style="font-size: x-large;">Testing</h3>
<p style="font-size: x-large;">Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements contrary to the actual requirements. The extent of testing varies greatly between organizations, developers, and individual sites or applications.</p>
</body>
</html>

```

Output:

Web Development

Web development is the work involved in developing a Web site for the Internet (World Wide Web) or an intranet (a private network).[1] Web development can range from developing a simple single static page of plain text to complex web applications, electronic businesses, and social network services. A more comprehensive list of tasks to which Web development commonly refers, may include Web engineering, Web design, Web content development, client liaison, client-side/server-side scripting, Web server and network security configuration, and e-commerce development. Among Web professionals, "Web development" usually refers to the main non-design aspects of building Web sites: writing markup and coding.[2] Web development may use content management systems (CMS) to make content changes easier and available with basic technical skills. For larger organizations and businesses, Web development teams can consist of hundreds of people (Web developers) and follow standard methods like Agile methodologies while developing Web sites. Smaller organizations may only require a single permanent or contracting developer, or secondary assignment to related job positions such as a graphic designer or information systems technician. Web development may be a collaborative effort between departments rather than the domain of a designated department. There are three kinds of Web developer specialization: front-end developer, back-end developer, and full-stack developer. Front-end developers are responsible for behavior and visuals that run in the user browser, while back-end developers deal with the servers.



Practical Web Development

Basic

In practice, many Web developers will have basic interdisciplinary skills / roles, including:

- Graphic design / Web design
- Information architecture and copywriting/copyediting with Web usability, accessibility and search engine optimization in mind
- Mobile responsiveness

Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements contrary to the actual requirements. The extent of testing varies greatly between organizations, developers, and individual sites or applications.

Experiment 23

Aim: What are Web security issues and how they can be controlled.

The web security vulnerabilities are prioritized depending on exploitability, detectability and impact on software.

❓ **Exploitability –**

What is needed to exploit the security vulnerability? Highest exploitability when the attack needs only web browser and lowest being advanced programming and tools.

❓ **Detectability –**

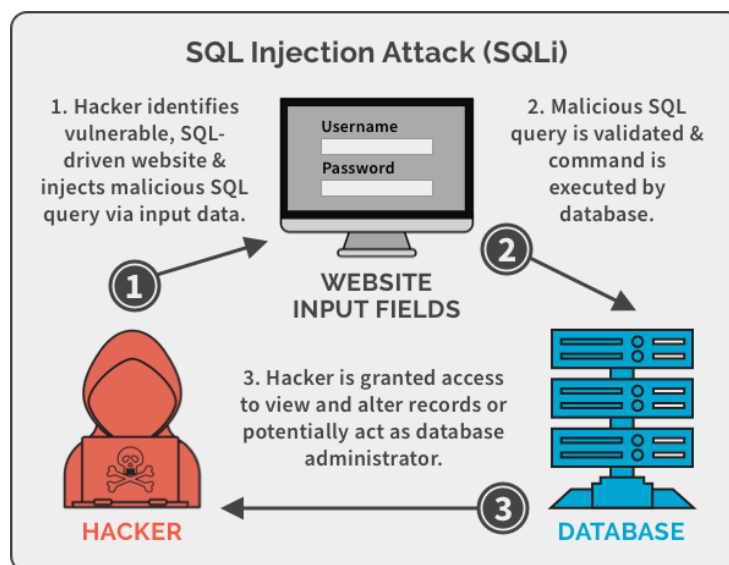
How easy is it to detect the threat? Highest being the information displayed on URL, Form or Error message and lowest being source code.

❓ **Impact or Damage–**

How much damage will be done if the security vulnerability is exposed or attacked? Highest being complete system crash and lowest being nothing at all.

Following is a list of top web security vulnerabilities based on the data from various security organizations.

SQL Injection



Description

Injection is a security vulnerability that allows an attacker to alter backend SQL statements by manipulating the user supplied data.

Injection occurs when the user input is sent to an interpreter as part of command or query and trick the interpreter into executing unintended commands and gives access to unauthorized data. The SQL command which when executed by web application can also expose the back-end database.

Implication

- ❓ An attacker can inject malicious content into the vulnerable fields.
- ❓ Sensitive data like User Names, Passwords, etc. can be read from the database.
- ❓ Database data can be modified (Insert/Update/ Delete).
- ❓ Administration Operations can be executed on the database

Vulnerable Objects

- ❓ Input Fields
- ❓ URLs interacting with the database.

Examples:

- ❓ SQL injection on the Login Page

Logging into an application without having valid credentials. Valid userName is available, and password is not available. Test URL:

<http://demo.testfire.net/default.aspx>

User Name: sjones Password:

1=1' or pass123

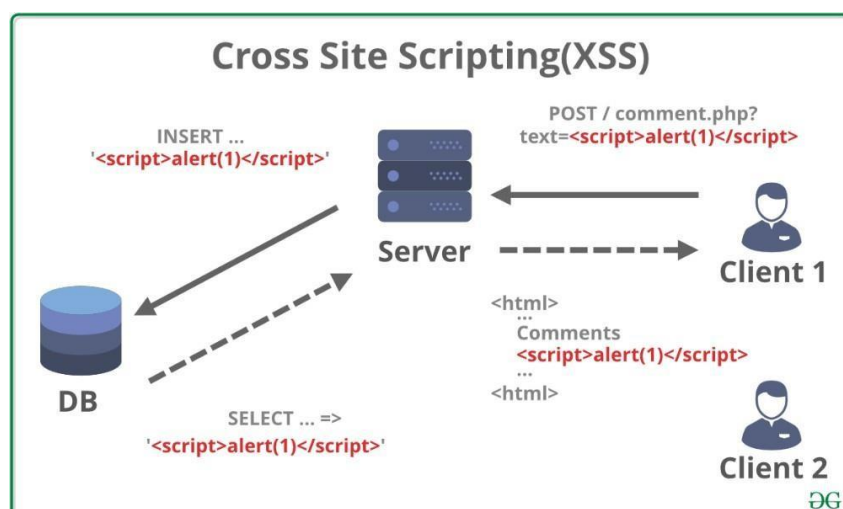
SQL query created and sent to Interpreter as below

`SELECT * FROM Users WHERE User_Name=sjones AND Password=1=1' or pass123;`

Recommendations

- ❓ White listing the input fields
- ❓ Avoid displaying detailed error messages that are useful to an attacker.

Cross Site Scripting



Description

Cross Site Scripting is also shortly known as XSS. XSS vulnerabilities target scripts embedded in a page that are executed on the client side i.e. user browser rather than at the server side. These flaws can occur when the application takes untrusted data and send it to the web browser without proper validation. Attackers can use XSS to execute malicious scripts on the users in this case victim browsers. Since the browser cannot know if the script is trustworthy or not, the script will be executed, and the attacker can hijack session cookies, deface websites, or redirect the user to an unwanted and malicious websites. XSS is an attack which allows the attacker to execute the scripts on the victim's browser.

Implication:

- ❓ Making the use of this security vulnerability, an attacker can inject scripts into the application, can steal session cookies, deface websites, and can run malware on the victim's machines.

Vulnerable Objects

- ❓ Input Fields
- ❓ URLs

Examples

1. `http://www.vulnerablesite.com/home?"<script>alert("xss")</script>`

The above script when run on a browser, a message box will be displayed if the site is vulnerable to XSS. The more serious attack can be done if the attacker wants to display or store session cookie.

2. `http://demo.testfire.net/search.aspx?txtSearch<iframe> <src = http://google.com width = 500 height 500></iframe>`

The above script when run, the browser will load an invisible frame pointing to `http://google.com`.

The attack can be made serious by running a malicious script on the browser.

Recommendations

- White Listing input fields
- Input Output encoding

Broken Authentication and Session Management



Description

The websites usually create a session cookie and session ID for each valid session, and these cookies contain sensitive data like username, password, etc. When the session is ended either by logout or browser closed abruptly, these cookies should be invalidated i.e. for each session there should be a new cookie.

If the cookies are not invalidated, the sensitive data will exist in the system. For example, a user using a public computer (Cyber Cafe), the cookies of the vulnerable site sits on the system and exposed to an attacker. An attacker uses the same public computer after some time, the sensitive data is compromised.

In the same manner, a user using a public computer, instead of logging off, he closes the browser abruptly. An attacker uses the same system, when browses the same vulnerable site, the previous session of the victim will be opened. The attacker can do whatever he wants to do from stealing profile information, credit card information, etc. A check should be done to find the strength of the authentication and session management. Keys, session tokens, cookies should be implemented properly without compromising passwords.

Vulnerable Objects

- Session IDs exposed on URL can lead to session fixation attack.
- Session IDs same before and after logout and login.
- Session Timeouts are not implemented correctly.
- Application is assigning same session ID for each new session.
- Authenticated parts of the application are protected using SSL
- Passwords are stored in hashed or encrypted format.
- The session can be reused by a low privileged user.

Implication

- Making use of this vulnerability, an attacker can hijack a session, gain unauthorized access to the system which allows disclosure and modification of unauthorized information.
- The sessions can be highjacked using stolen cookies or sessions using XSS.

Examples

1. Airline reservation application supports URL rewriting, putting session IDs in the URL:
http://Examples.com/sale/saleitems;jsessionid=2P0OC2oJMODPXSNQPLME34SERTBG/dest=Maldives (Sale of tickets to Maldives)

An authenticated user of the site wants to let his friends know about the sale and sends an email across. The friends receive the session ID and can be used to do unauthorized modifications or misuse the saved credit card details.

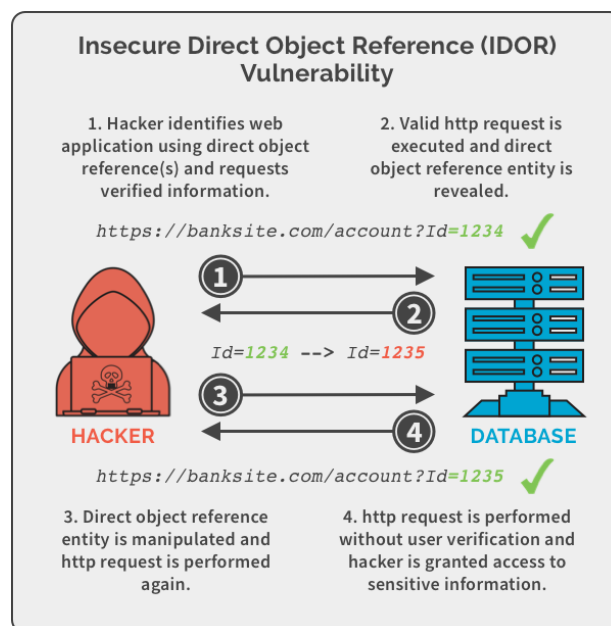
2. An application is vulnerable to XSS, by which an attacker can access the session ID and can be used to hijack the session.

3. Application timeouts are not set properly. The user uses a public computer and closes the browser instead of logging off and walks away. The attacker uses the same browser sometime later, and the session is authenticated.

Recommendations

- All the authentication and session management requirements should be defined as per OWASP Application Security Verification Standard.
- Never expose any credentials in URLs or Logs.
- Strong efforts should be also made to avoid XSS flaws which can be used to steal session IDs.

Insecure Direct Object References



Description

It occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key as in URL or as a FORM parameter. The attacker can use this information to access other objects and can create a future attack to access the unauthorized data.

Implication

Using this vulnerability, an attacker can gain access to unauthorized internal objects, can modify data or compromise the application.

Vulnerable Objects

- In the URL.

Examples:

Changing "userid" in the following URL can make an attacker to view other user's information.

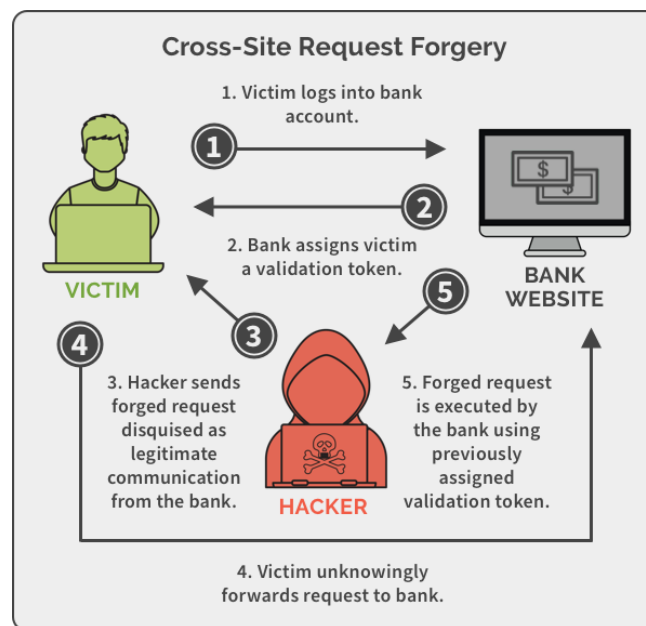
`http://www.vulnerablesite.com/userid=123` Modified to
`http://www.vulnerablesite.com/userid=124`

An attacker can view others information by changing user id value.

Recommendations:

1. Implement access control checks.
2. Avoid exposing object references in URLs.
3. Verify authorization to all reference objects.

Cross Site Request Forgery



Description

Cross Site Request Forgery is a forged request came from the cross site. CSRF attack is an attack that occurs when a malicious website, email, or program causes a user's browser to perform an unwanted action on a trusted site for which the user is currently authenticated. A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application.

A link will be sent by the attacker to the victim when the user clicks on the URL when logged into the original website, the data will be stolen from the website.

Implication

- ❑ Using this vulnerability as an attacker can change user profile information, change status, create a new user on admin behalf, etc.

Vulnerable Objects

- ❑ User Profile page
- ❑ User account forms
- ❑ Business transaction page

Examples

The victim is logged into a bank website using valid credentials. He receives mail from an attacker saying "Please click here to donate \$1 to cause."

When the victim clicks on it, a valid request will be created to donate \$1 to a particular account.

<http://www.vulnerablebank.com/transfer.do?account=cause&amount=1>

The attacker captures this request and creates below request and embeds in a button saying "I Support Cause."

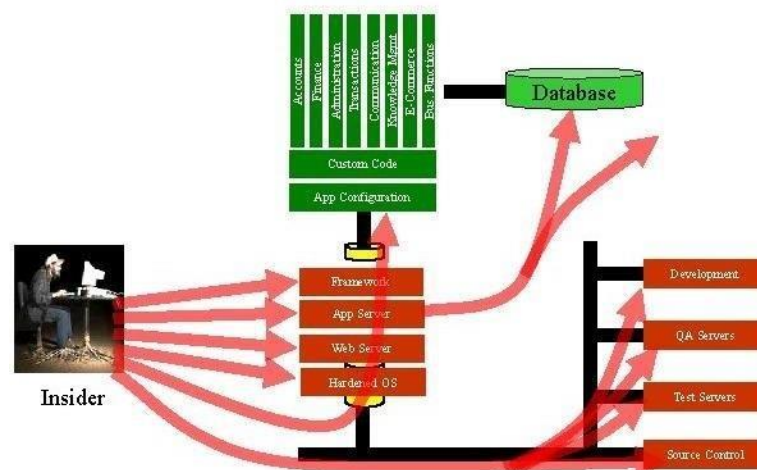
<http://www.vulnerablebank.com/transfer.do?account=Attacker&amount=1000>

Since the session is authenticated and the request is coming through the bank website, the server would transfer \$1000 dollars to the attacker.

Recommendation

- ❑ Mandate user's presence while performing sensitive actions.
- ❑ Implement mechanisms like CAPTCHA, Re-Authentication, and Unique Request Tokens.

Security Misconfiguration



Description

Security Configuration must be defined and deployed for the application, frameworks, application server, web server, database server, and platform. If these are properly configured, an attacker can have unauthorized access to sensitive data or functionality. Sometimes such flaws result in complete system compromise. Keeping the software up to date is also good security.

Implication

- ❑ Making use of this vulnerability, the attacker can enumerate the underlying technology and application server version information, database information and gain information about the application to mount few more attacks.

Vulnerable objects

- ? URL
- ? Form Fields
- ? Input fields

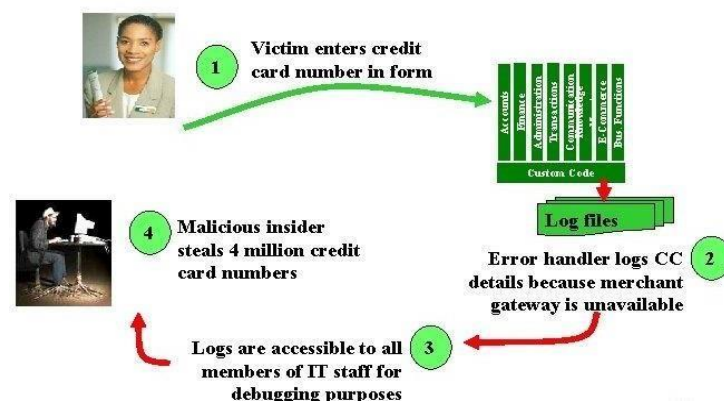
Examples

- ? The application server admin console is automatically installed and not removed. Default accounts are not changed. The attacker can log in with default passwords and can gain unauthorized access.
- ? Directory Listing is not disabled on your server. Attacker discovers and can simply list directories to find any file.

Recommendations

- ? A strong application architecture that provides good separation and security between the components.
- ? Change default usernames and passwords.
- ? Disable directory listings and implement access control checks.

Insecure Cryptographic Storage



Description

Insecure Cryptographic storage is a common vulnerability which exists when the sensitive data is not stored securely. The user credentials, profile information, health details, credit card information, etc. come under sensitive data information on a website. This data will be stored on the application database. When this data is stored improperly by not using encryption or hashing*, it will be vulnerable to the attackers.

(*Hashing is transformation of the string characters into shorter strings of fixed length or a key. To decrypt the string, the algorithm used to form the key should be available)

Implication

By using this vulnerability, an attacker can steal, modify such weakly protected data to conduct identity theft, credit card fraud or other crimes.

Vulnerable objects

Application database.

Examples

In one of the banking application, password database uses unsalted hashes* to store everyone's passwords. An SQL injection flaw allows the attacker to retrieve the password file. All the unsalted hashes can be brute forced in no time whereas, the salted passwords would take thousands of years.

(*Unsalted Hashes – Salt is a random data appended to the original data. Salt is appended to the password before hashing)

Recommendations

- ❑ Ensure appropriate strong standard algorithms. Do not create own cryptographic algorithms. Use only approved public algorithms such as AES, RSA public key cryptography, and SHA-256, etc.
- ❑ Ensure offsite backups are encrypted, but the keys are managed and backed up separately.

Failure to restrict URL Access



- Attacker notices the URL indicates his role
/user/getAccounts
- He modifies it to another directory (role)
/admin/getAccounts, or
/manager/getAccounts
- Attacker views more accounts than just their own

Description

Web applications check URL access rights before rendering protected links and buttons. Applications need to perform similar access control checks each time these pages are accessed. In most of the applications, the privileged pages, locations and resources are not presented to the privileged users. By an intelligent guess, an attacker can access privilege pages. An attacker can access sensitive pages, invoke functions and view confidential information.

Implication

Making use of this vulnerability attacker can gain access to the unauthorized URLs, without logging into the application and exploit the vulnerability. An attacker can access sensitive pages, invoke functions and view confidential information.

Vulnerable objects:

URLs

Examples

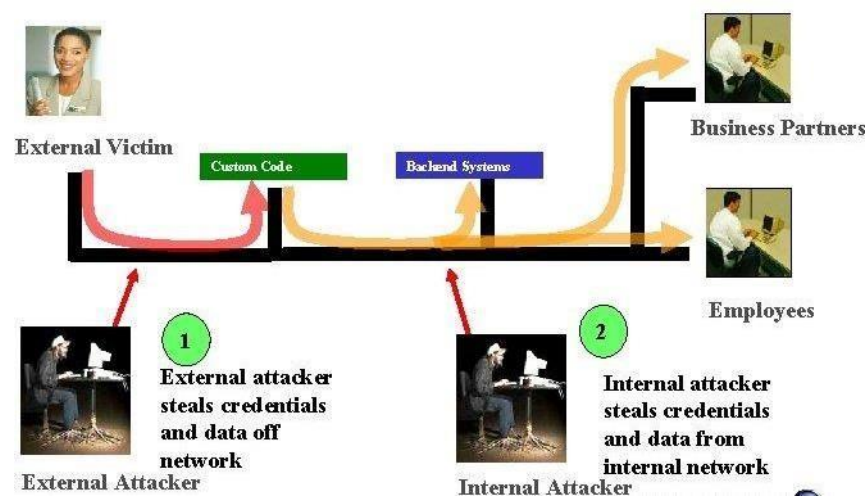
- ❑ Attacker notices the URL indicates the role as "/user/getaccounts." He modifies as "/admin/getaccounts".
- ❑ An attacker can append role to the URL.

<http://www.vulnerabsite.com> can be modified as <http://www.vulnerablesite.com/admin>

Recommendations

- ❑ Implement strong access control checks.
- ❑ Authentication and authorization policies should be role-based.
- ❑ Restrict access to unwanted URLs.

Insufficient Transport Layer Protection



Description

Deals with information exchange between the user (client) and the server (application). Applications frequently transmit sensitive information like authentication details, credit card information, and session tokens over a network.

By using weak algorithms or using expired or invalid certificates or not using SSL can allow the communication to be exposed to untrusted users, which may compromise a web application and or steal sensitive information.

Implication

- ❑ Making use of this web security vulnerability, an attacker can sniff legitimate user's credentials and gaining access to the application.
- ❑ Can steal credit card information.

Vulnerable objects

- ❑ Data sent over the network.

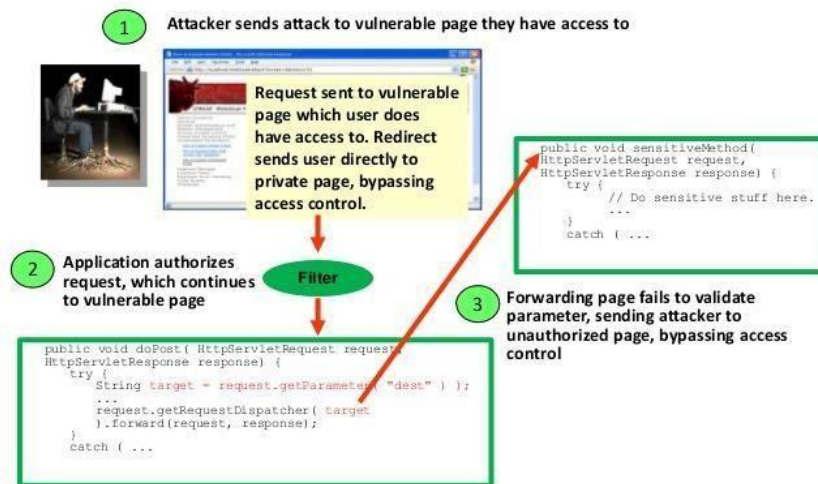
Recommendations

1. Enable secure HTTP and enforce credential transfer over HTTPS only.
2. Ensure your certificate is valid and not expired.

Examples:

1. An application not using SSL, an attacker will simply monitor network traffic and observe an authenticated victim session cookie. An attacker can steal that cookie and perform Man-in-the-Middle attack.

Unvalidated Redirects and Forwards



Description

The web application uses few methods to redirect and forward users to other pages for an intended purpose. If there is no proper validation while redirecting to other pages, attackers can make use of this and can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Implication

An attacker can send a URL to the user that contains a genuine URL appended with encoded malicious URL. A user by just seeing the genuine part of the attacker sent URL can browse it and may become a victim.

Examples

<http://www.vulnerablesite.com/login.aspx?redirectURL=ownsite.com>

Modified to

<http://www.vulnerablesite.com/login.aspx?redirectURL=evilsite.com>

Recommendations

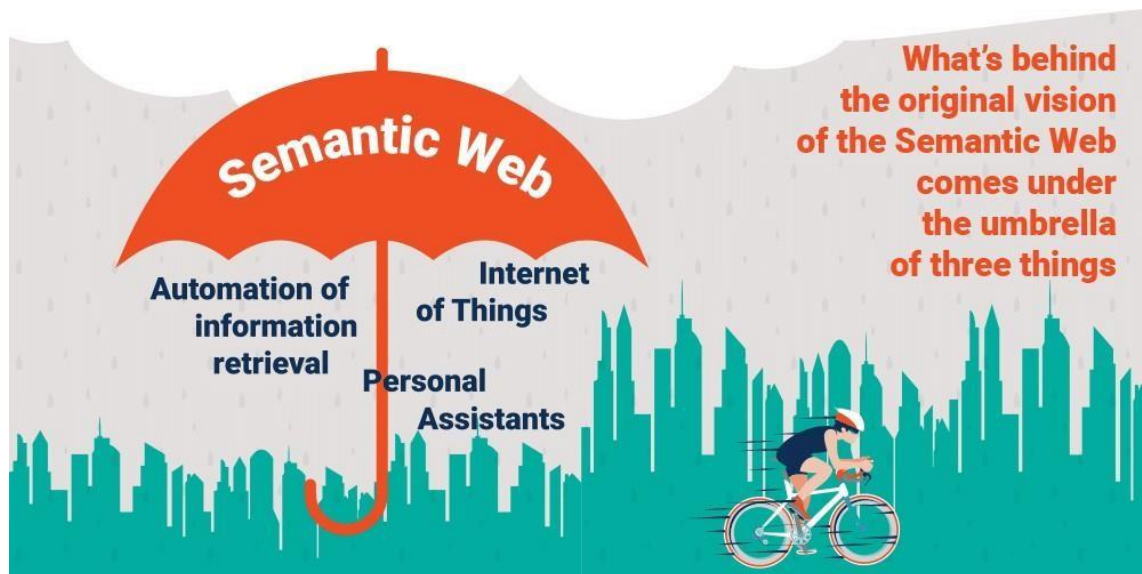
- Simply avoid using redirects and forwards in the application. If used, do not involve using user parameters in calculating the destination.
- If the destination parameters can't be avoided, ensure that the supplied value is valid, and authorized for the user.

Experiment - 24

Aim: Define the terms Semantic Web and Ontology?

Semantic Web

The Semantic Web is a vision about an extension of the existing World Wide Web, which provides software programs with machine-interpretable metadata of the published information and data. In other words, we add further data descriptors to otherwise existing content and data on the Web. As a result, computers are able to make meaningful interpretations similar to the way humans process information to achieve their goals.

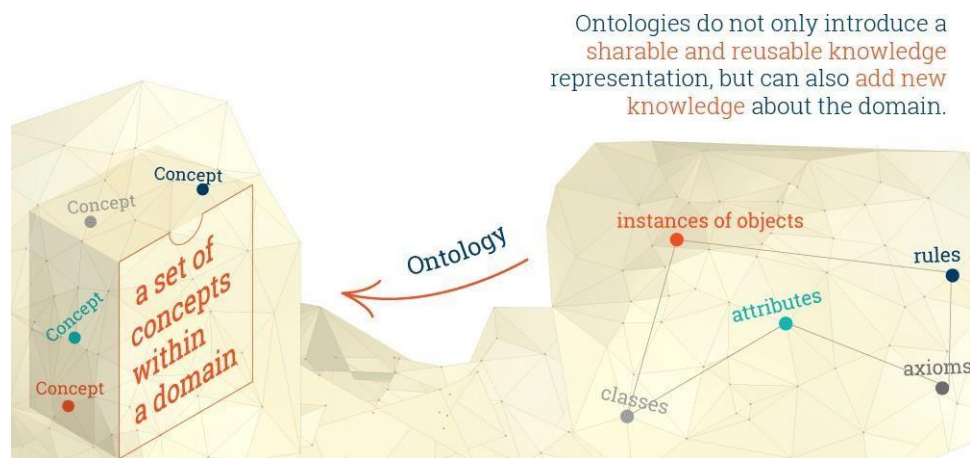


The ultimate ambition of the Semantic Web, as its founder Tim Berners-Lee sees it, is to enable computers to better manipulate information on our behalf. He further explains that, in the context of the Semantic Web, the word "semantic" indicates machine-processable or what a machine is able to do with the data. Whereas "web" conveys the idea of a navigable space of interconnected objects with mappings from URIs to resources.

For the Semantic Web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning.

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. The Semantic Web is the web of connections between different forms of data that allow a machine to do something it wasn't able to do directly.

Ontology



An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them. To enable such a description, we need to formally specify components such as individuals (instances of objects), classes, attributes and relations as well as restrictions, rules and axioms. As a result, ontologies do not only introduce a sharable and reusable knowledge representation but can also add new knowledge about the domain.

The ontology data model can be applied to a set of individual facts to create a knowledge graph – a collection of entities, where the types and the relationships between them are expressed by nodes and edges between these nodes. By describing the structure of the knowledge in a domain, the ontology sets the stage for the knowledge graph to capture the data in it.

As one of the building blocks of Semantic Technology, ontologies are part of the W3C standards stack for the Semantic Web. They provide users with the necessary structure to link one piece of information to other pieces of information on the Web of Linked Data. Because they are used to specify common modelling representations of data from distributed and heterogeneous systems and databases, ontologies enable database interoperability, cross- database search and smooth knowledge management.

The Benefits of Using Ontologies

One of the main features of ontologies is that, by having the essential relationships between concepts built into them, they enable automated reasoning about data. Such reasoning is easy to implement in semantic graph databases that use ontologies as their semantic schemata.

What's more, ontologies function like a 'brain'. They 'work and reason' with concepts and relationships in ways that are close to the way humans perceive interlinked concepts.

In addition to the reasoning feature, ontologies provide more coherent and easy navigation as users move from one concept to another in the ontology structure.

Limitations of Ontologies

While ontologies provide a rich set of tools for modelling data, their usability comes with certain limitations. One such limitation is the available property constructs. For example, while providing powerful class constructs, the most recent version of the Web Ontology Language – OWL2 has a somewhat limited set of property constructs.

Another limitation comes from the way OWL employs constraints. They serve to specify how data should be structured and prevent adding data inconsistent with these constraints. This, however, is not always beneficial. Often, data imported from a new source into the RDF triple store would be structurally inconsistent with the constraints set using OWL. Consequently, this new data would have to be modified before being integrated with what is already loaded in the triple store.

Experiment 25

Aim: What is Digital signature and Digital Certificate. Explain their use.

Digital Signature.

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document.

1. **Key Generation Algorithms** : Digital signature are electronic signatures, which assures that the message was sent by a particular sender. While performing digital transactions authenticity and integrity should be assured, otherwise the data can be altered or someone can also act as if he was the sender and expect a reply.
2. **Signing Algorithms**: To create a digital signature, signing algorithms like email programs create a one-way hash of the electronic data which is to be signed. The signing algorithm then encrypts the hash value using the private key (signature key). This encrypted hash along with other information like the hashing algorithm is the digital signature. This digital signature is appended with the data and sent to the verifier. The reason for encrypting the hash instead of the entire message or document is that a hash function converts any arbitrary input into a much shorter fixed length value. This saves time as now instead of signing a long message a shorter hash value has to be signed and moreover hashing is much faster than signing.
3. **Signature Verification Algorithms** : Verifier receives Digital Signature along with the data. It then uses Verification algorithm to process on the digital signature and the public key (verification key) and generates some value. It also applies the same hash function on the received data and generates a hash value. Then the hash value and the output of the verification algorithm are compared. If they both are equal, then the digital signature is valid else it is invalid.

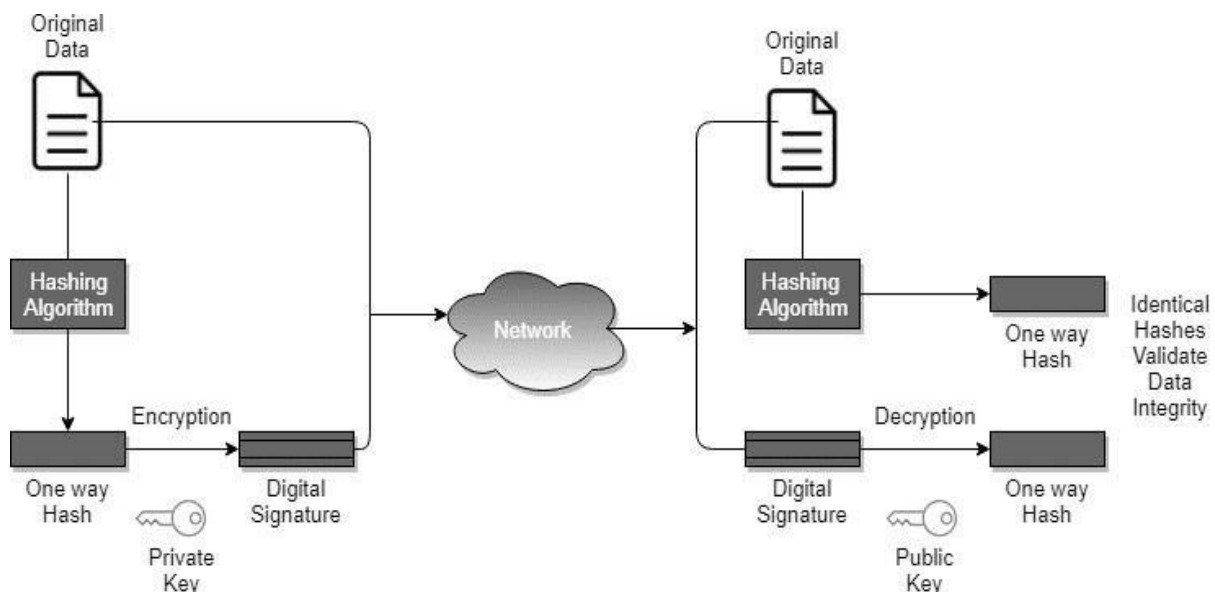
The steps followed in creating digital signature are :

1. Message digest is computed by applying hash function on the message and then message digest is encrypted using private key of sender to form the digital signature. (digital signature = encryption (private key of sender, message digest) and message digest = message digest algorithm(message)).
2. Digital signature is then transmitted with the message.(message + digital signature is transmitted)
3. Receiver decrypts the digital signature using the public key of sender.(This assures authenticity, as only sender has his private key so only sender can

encrypt using his private key which can thus be decrypted by sender's public key).

4. The receiver now has the message digest.
5. The receiver can compute the message digest from the message (actual message is sent with the digital signature).
6. The message digest computed by receiver and the message digest (got by decryption on digital signature) need to be same for ensuring integrity.

Message digest is computed using one-way hash function, i.e. a hash function in which computation of hash value of a message is easy but computation of the message from hash value of the message is very difficult.



Digital Certificate

Digital certificate is issued by a trusted third party which proves sender's identity to the receiver and receiver's identity to the sender. A digital certificate is a certificate issued by a Certificate Authority (CA) to verify the identity of the certificate holder. The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information. Digital certificate is used to attach public key with a particular individual or an entity.

Digital certificate contains:-

1. Name of certificate holder.
2. Serial number which is used to uniquely identify a certificate, the individual or the entity identified by the certificate
3. Expiration dates.

4. Copy of certificate holder's public key.(used for decrypting messages and digital signatures)
5. Digital Signature of the certificate issuing authority.

Digital certificate is also sent with the digital signature and the message.

Digital certificate vs digital signature :

Digital signature is used to verify authenticity, integrity, non-repudiation ,i.e. it is assuring that the message is sent by the known user and not modified, while digital certificate is used to verify the identity of the user, maybe sender or receiver. Thus, digital signature and certificate are different kind of things but both are used for security. Most websites use digital certificate to enhance trust of their users.

Feature	Digital Signature	Digital Certificate
Basics / Definition	Digital signature is like a fingerprint or an attachment to a digital document that ensures its authenticity and integrity.	Digital certificate is a file that ensures holder's identity and provides security.
Process / Steps	Hashed value of original message is encrypted with sender's secret key to generate the digital signature.	It is generated by CA (Certifying Authority) that involves four steps: Key Generation, Registration, Verification, Creation.
Security Services	Authenticity of Sender, integrity of the document and non-repudiation .	It provides security and authenticity of certificate holder.
Standard	It follows Digital Signature Standard (DSS).	It follows X.509 Standard Format

Experiment 26

Aim: Explain steps of implementing web services in any web application.

Web services enable applications to interact with one another over the Web in a platform-neutral, language independent environment. In a typical Web services scenario, a business application sends a request to a service at a given URL by using the HTTP protocol. The service receives the request, processes it, and returns a response. You can incorporate calls to external Web services in applications developed in Oracle Application Express.

Web services in Oracle Application Express are based on SOAP (Simple Object Access Protocol). SOAP is a World Wide Web Consortium (W3C) standard protocol for sending and receiving requests and responses across the Internet. SOAP messages can be sent back and forth between a service provider and a service user in SOAP envelopes.

Web services are called from within an Oracle Application Express application by:

- Using the Universal Description, Discovery, and Integration (UDDI) registry
- Manually providing the WSDL URL

Creating a New Application

First, create a new application. To create an application:

1. On the Workspace home page, click the **Application Builder** icon.
2. On the Application Builder home page, click **Create**.
3. For Method, select **Create Application**, and click **Next**.
4. For Name:
 - a. For Name - Enter **Web Services**.
 - b. Accept the remaining defaults and click **Next**.
5. Add a blank page:
 - a. Under Select Page Type, accept the default, **Blank**.
 - b. In Page Name, enter **Web Services** and then click **Add Page**. The new page appears in the list at the top of the page.
 - c. Click **Next**.
6. For Tabs, accept the default, **One Level of Tabs**, and click **Next**.
7. For Shared Components, accept the default, **No**, and click **Next**.
8. For Attributes, accept the default for Authentication Scheme, Language, and User Language Preference Derived From and click **Next**.
9. For User Interface, select **Theme 2** and click **Next**.
10. Review your selections and click **Create**.

Specifying an Application Proxy Server Address

If your environment requires a proxy server to access the Internet, you must specify a proxy server address on the Application Attributes page before you can create a Web service reference.

To specify a proxy address:

1. On the Application home page, click **Shared Components**.
2. Under Application, click **Definition**.
3. Under Name, enter the proxy server in the Proxy Server field.
4. Click **Apply Changes**.

Creating a Web Service Reference from a WSDL

To create a new Web reference by supplying the WSDL location:

1. On the Application home page, click **Shared Components**. The Shared Components page appears.
2. Under Logic, select **Web Service References**.
3. Click **Create**
4. When prompted whether to search a UDDI registry to find a WSDL, select **No** and click **Next**.
 - a. In the WSDL Location field enter:
`http://www.ignite.com/webservices/ignite.whatsshowing.webservice/moviefunctions.asmx?wsdl`
 - b. Click **Next**.
A summary page appears describing the selected Web service.
5. Click **Create Reference**.

The Create Web Service Reference page appears. The Web service reference for Movie Information is added to the Web Service References Repository.

Create a Form and Report

To create a form and report after creating a Web Service Reference:

1. On the Create Web Service Reference success page, select **Create Form and Report on Web Service**.
2. For Choose Service and Operation:
 - a. Web Service Reference - Select **Movie Information**.
 - b. Operation - Select **GetTheatersAndMovies**.
 - c. Click **Next**.
3. For Page and Region Attributes:
 - a. Form Region Title - Change to **Theater Information**.
 - b. Accept the other defaults and click **Next**.
4. For Input Items:
 - a. For P2_ZIPCODE and P2_RADIUS, accept the default, **Yes**, in the Create column.
 - b. For P2_ZIPCODE, change the Item Label default to **ZIP Code**.
 - c. Click **Next**.
5. For Web Service Results:
 - a. Temporary Result Set Name (Collection) - Accept the default.
 - b. Result Tree to Report On - Select **Theater (tns:Theater)**.
 - c. Click **Next**.

- For Result Parameters, select all the parameters and click **Finish**.
- Click **Run Page**.
- If prompted to login, enter the username and password for your workspace and click **Login**.

Theater Information

ZIP Code

Radius

Results

No data found.

Theater Information Form and Report without Data

- To test the form, enter **43221** in the ZIP Code field and **5** in the Radius field. Then click **Submit**.

Results	
Name	Address
Starplex Movies 12	3773 Ridge Mill Drive, Hilliard, OH
Cinemark Movie 12	2570 Bethel Road, Columbus, OH
AMC Lennox Town Center 24	777 Kinnear Road, Columbus, OH
Starplex Cinemas 10	5275 Westpointe Plaza Drive, Columbus, OH
Studio 35 Cinema	3055 Indianola Avenue, Columbus, OH
Rave Motion Pictures Polaris 18	1071 Gemini Place, Columbus, OH
Dollar Cinemas-Westland	4265 Shoppers Lane, Columbus, OH
Arena Grand Theatre	175 W. Nationwide, Columbus, OH
AMC Dublin Village 18	6700 Village Parkway, Dublin, OH
Marcus Crosswoods Center	200 Hutchinson Avenue, Columbus, OH
Regal Georgesville Square Stadium 16	1800 Georgesville Square Drive, Columbus, OH
AMC Easton Town Center 30 with IMAX	275 Easton Town Ctr, Columbus, OH
Hollywood Studio Theatres-Westerville	5996 Westerville Road, Westerville, OH
Star Cinema 8-Grove City	2384 Stringtown Rd., Grove City, OH
Cinemark Movies 16 Gahanna	323 Stoneridge Lane, Gahanna, OH

1 - 15

Theater Information Report Showing Resulting Data

Creating a Web Service Reference Manually

To create a Web reference manually, you will copy code from the WSDL for a service called MovieInformation.

To create a manual Web reference:

- On the Application home page, click **Shared Components**.
- Under Logic, click **Web Service References**.
- Click **Create**.
- For Search UDDI, select **No** and click **Next**.
- From the Tasks list on the right, click the **Create Web Reference Manually** link. The Create/Edit Web Service page appears.
- In the Name field, enter **Movie Info**.

7. Locate the endpoint of the MovieInformation service:

a. Open the WSDL by going to:

b. <http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx?wsdl>

c. In the WSDL, find the **location** attribute of the **soap:address** element, which is a child of the **port** element. You can search for the following term within the code: **soap:address location**.

At the time of this release, it was this attribute:

<http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx>

8. In the URL field on the Create/Edit Web Service page, enter the endpoint of the MovieInformation service you located. For

example: **<http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx>**

9. Locate the SOAP action for the **GetTheatersAndMovies** operation:

a. If necessary, open the WSDL again. See Step 7a.

b. In the WSDL, find the **soapAction** attribute of the **soap:operation** element, which is a child of the operation element that has a name attribute of **GetTheatersAndMovies**. You can search for the following term within the code: **soap:operation soapAction**.

At the time of this release, it was this attribute:

<http://www.ignyte.com/whatsshowing/GetTheatersAndMovies>

10. In the Action field on the Create/Edit Web Service page, enter the SOAP action you located. For example:

<http://www.ignyte.com/whatsshowing/GetTheatersAndMovies>

11. In the SOAP Envelope field on the Create/Edit Web Reference page, enter the xml code representing the SOAP Request message. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignyte.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <tns:GetTheatersAndMovies>
      <tns:zipCode>#ZIP#</tns:zipCode>
      <tns:radius>#RADIUS#</tns:radius>
    </tns:GetTheatersAndMovies>
  </soap:Body>
</soap:Envelope>
```

You can use a SOAP message generating tool, such as MindReef, to construct a valid SOAP Request for a given Web service.

12. In the `StoreResponseInCollection` field, enter **MOVIE_RESULTS**. This is where the response from the Web service will be stored.

The screenshot shows a web service configuration window with the following sections:

- Web Service**
 - Application: 433 Web Services 3
 - Name: Movie Info
- Service Description**
 - URL: http://www.ignite.com/webservices/ignite.whatsshowing.webservice/moviefun
 - Action: http://www.ignite.com/whatsshowing/GetTheatersAndMovies
 - Proxy: (empty)
 - Basic Authentication: ☐ Yes ☒ No
- SOAP Envelope**
 - SOAP Envelope:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.ignite.com/whatsshowing"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <tns:GetTheatersAndMovies>
      <tns:zipCode>#ZIP#</tns:zipCode>
      <tns:radius>#RADIUS#</tns:radius>
    </tns:GetTheatersAndMovies>
  </soap:Body>
</soap:Envelope>
```
- SOAP Response**
 - Store Response in Collection: MOVIE_RESULTS

13. Click **Create**.

The Web Service References page appears, showing Movie Info in the list.

Test the Web Service

To test the Web service:

1. On the Web Service References page, click the **Test** icon next to the Movie Info Web reference. The Web Services Testing page appears. Note View must be set to Details, otherwise the Test icon is not displayed.
2. In the SOAP Envelope field, replace #ZIP# with **43221** and #RADIUS# with **5**.
3. Click **Test**.
4. Review the Result field and note the following about the response:

- The base node in the return envelope is called:

GetTheatersAndMoviesResponse

- The namespace for the message is:

http://www.ignite.com/whatsshowing

- The XPath to the interesting parts of the response under the result element is:

```
/GetTheatersAndMoviesResponse/GetTheatersAndMoviesResult/Theater/Movies/Movie
```

- The interesting elements in the results are called:

```
Name
Rating
RunningTime
ShowTimes
```

Create a Page to Call the Manual Web Service

Next, you want to create a page to call the manual Web Service. To create a page to call the manual Web service:

1. Click the **Application** breadcrumb link.
2. On the Application home page, click **Create Page**.
3. For Page, select **Blank Page** and click **Next**.
4. Accept the default for the Page Number and click **Next**.
5. In Name, enter **Find Movies** and click **Next**.
6. For Tabs, accept the default, **No**, and click **Next**.
7. Click **Finish**.
8. On the Success page, click **Edit Page**.
9. On the Page Definition, locate the Regions section.
10. Click the **Create** icon.
11. For Region, select **HTML** and click **Next**.
12. Select **HTML** as the HTML region container and click **Next**.
13. In the Title field, enter **Movie Information** and click **Next**.
14. Click **Create Region**.

Create a Submit Button

To create a Submit button:

1. On the Page Definition, click the **Create** icon in the Buttons section.
2. For Button Region, accept the default, **Movie Information**, and click **Next**.
3. For Button Position, accept the default, **Create a button in a region position**, and click **Next**.
4. For Button Attributes, enter **SUBMIT** in the Button Name and click **Next**.
5. For Button Template, accept the default, **Button**, and click **Next**.
6. For Display Properties, select **Region Template Position #CREATE#** from the Position list and click **Next**.
7. In the Branch to Page field, select **Find Movies** from the list. The page number appears in the field.
8. Click **Create Button**.

Create Items for ZIP Code and Radius

To create the ZIP Code item:

1. On the Find Movies Page Definition, click the **Create** icon in the Items section.
2. For Item Type, select **Text** and click **Next**.
3. For Text Control Display Type, accept the default, **Text Field**, and click **Next**.
4. For Display Position and Name, specify the following:
 - a. Item Name - Enter **ZIP**.

The Movie Info Web Service Reference defines the zip codes sent in the SOAP Request as #ZIP#. Therefore, this Item Name must be ZIP in order for its value to be substituted into the SOAP Request sent to the Web Service.

- b. Region - Accept the default, **Movie Information**.
 - c. Click **Next**.
5. In the Label field, replace the existing text with **ZIP Code** and click **Next**.
 6. Click **Create Item**.

To create the Radius item:

1. On the Find Movies Page Definition, click the **Create** icon in the Items section.
2. For Item Type, select **Text** and click **Next**.
3. For Text Control Display Type, accept the default, **Text Field**, and click **Next**.
4. For Display Position and Name, specify the following:
 - a. Item Name - Enter **RADIUS**.

The Movie Info Web Service Reference defines the radius sent in the SOAP Request as #RADIUS#. Therefore, this Item Name must be RADIUS in order for its value to be substituted into the SOAP Request sent to the Web Service.

- b. Region - Accept the default, **Movie Information**.
 - c. Click **Next**.
5. In the Label field, enter **Radius** and click **Next**.
 6. Click **Create Item**.

Create a Process to Call the Manually Created Web Reference

To create a process to call the manually created Web reference:

1. On the Find Movies Page Definition, click the **Create** icon in the Processes section.
2. For Process Type, select **Web Services** and click **Next**.
3. In the Name field, enter **Call Movie Info** and click **Next**.
4. From the Web Service Reference list, select **Movie Info** and click **Next**.
5. In the Success Message area, enter **Called Movie Info**.
6. In the Failure Message area, enter **Error calling Movie Info** and click **Next**.
7. From the When Button Pressed list, select **SUBMIT** and click **Create Process**.

Create a Report on the Web Service Result

To create a report on the Web service result:

1. On the Find Movies Page Definition, click the **Create** icon in the Regions section.
2. Select **Report** and click **Next**.
3. For Region, select **Report on collection containing Webservice result** and click **Next**.

4. In the Title field, enter **Search Results** and click **Next**.
5. For Web Reference Type, select **Manually Created** and click **Next**.
6. For Web Reference Information, specify the following:
 - a. Web Service Reference - Select **Movie Info** from the list.
 - b. SOAP Style - Select **Document**.
 - c. Message Format - Select **Literal**.

Note that these two attributes can be determined by manually inspecting the WSDL document for the service.

- d. Result Node Path - Enter:

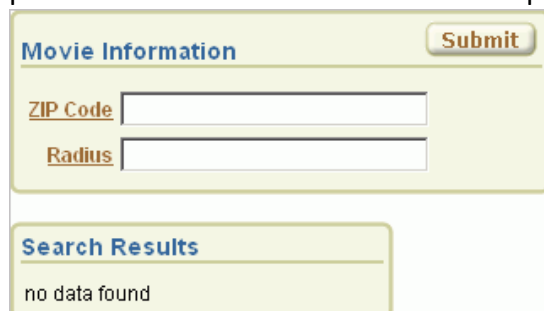
/GetTheatersAndMoviesResponse/GetTheatersAndMoviesResult/Theater/Movies/Movie

- e. Message Namespace - Enter:

http://www.ignyte.com/whatsshowing

Note that you reviewed both the Result Node Path and Message Namespace when testing the service.

- f. Click **Next**.
7. In the first four Parameter Names, enter **Name**, **Rating**, **RunningTime**, and **ShowTimes**, and click **Create SQL Report**.
8. To test the page:
 - a. Click **Run Page**. If prompted to log in, enter your workspace user name and password. The Movie Information form appears, as shown:



The screenshot shows a web form titled "Movie Information" with a "Submit" button. Below the title are two input fields labeled "ZIP Code" and "Radius". Underneath these fields is a section titled "Search Results" which contains the text "no data found".

- b. In the ZIP Code and Radius fields, enter information and click **Submit**. The results appear in the Search Results area.

Experiment - 27

Aim: What is E-commerce and explain various ecommerce models and Ecommerce infrastructure.



E-commerce (electronic commerce) is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the internet. These business transactions occur either as business-to-business (B2B), business-to-consumer (B2C), consumer-to-consumer or consumer-to-business. The terms e-commerce and e-business are often used interchangeably. The term e-tail is also sometimes used in reference to the transactional processes that make up online retail shopping.

In the last decade, widespread use of e-commerce platforms such as Amazon and eBay has contributed to substantial growth in online retail. In 2007, e-commerce accounted for 5.1% of total retail sales; in 2019, e-commerce made up 16.0%.

How does e-commerce work?

E-commerce is powered by the internet, where customers can access an online store to browse through, and place orders for products or services via their own devices.

As the order is placed, the customer's web browser will communicate back and forth with the server hosting the online store website. Data pertaining to the order will then be relayed to a central computer known as the order manager -- then forwarded to databases that manage inventory levels, a merchant system that manages payment information (using applications such as PayPal), and a bank computer -- before circling back to the order manager. This is to make sure that store inventory and customer funds are sufficient for the order to be processed. After the order is validated, the order manager will notify the store's web server, which will then display a message notifying the customer that their order has been successfully processed. The order manager will then send order data to the warehouse or fulfillment department, in order for the product or service to be successfully dispatched to the customer. At this point tangible and/or digital products may be shipped to a customer, or access to a service may be granted.

Platforms that host e-commerce transactions may include online marketplaces that sellers simply sign up for, such as Amazon.com; software as a service (SaaS) tools that allow customers to 'rent' online store infrastructures; or open source tools for companies to use in-house development to manage.

Types of e-commerce



Business-to-business (B2B) e-commerce refers to the electronic exchange of products, services or information between businesses rather than between businesses and consumers. Examples include online directories and product and supply exchange websites that allow businesses to search for products, services and information and to initiate transactions through e-procurement interfaces.

In 2017, Forrester Research predicted that the B2B e-commerce market will top \$1.1 trillion in the U.S. by 2021, accounting for 13% of all B2B sales in the nation.

Business-to-consumer (B2C) is the retail part of e-commerce on the internet. It is when businesses sell products, services or information directly to consumers. The term was popular during the dot-com boom of the late 1990s, when online retailers and sellers of goods were a novelty. Today, there are innumerable virtual stores and malls on the internet selling all types of consumer goods. The most recognized example of these sites is Amazon, which dominates the B2C market.

Consumer-to-consumer (C2C) is a type of e-commerce in which consumers trade products, services and information with each other online. These transactions are generally conducted through a third party that provides an online platform on which the transactions are carried out.

Online auctions and classified advertisements are two examples of C2C platforms, with eBay and Craigslist being two of the most popular of these platforms. Because eBay is a business, this form of e-commerce could also be called C2B2C--consumer-to-business-to-consumer.

Consumer-to-business (C2B) is a type of e-commerce in which consumers make their products and services available online for companies to bid on and purchase. This is the opposite of the traditional commerce model of B2C.

A popular example of a C2B platform is a market that sells royalty-free photographs, images, media and design elements, such as iStock. Another example would be a job board.

Business-to-administration (B2A) refers to transactions conducted online between companies and public administration or government bodies. Many branches of government are dependent on e-services or products in one way or another, especially when it comes to legal documents, registers, social security, fiscals and employment. Businesses can supply these electronically. B2A services have grown considerably in recent years as investments have been made in e-government capabilities.

Consumer-to-administration (C2A) refers to transactions conducted online between individual consumers and public administration or government bodies. The government rarely buys products or services from citizens, but individuals frequently use electronic means in the following areas:

- **Education.** Disseminating information, distance learning/online lectures, etc.
- **Social security.** Distributing information, making payments, etc.
- **Taxes.** filing tax returns, making payments, etc.
- **Health.** Making appointments, providing information about illnesses, making health services payments, etc.

Mobile e-commerce (M-commerce) is a type of e-commerce on the rise that features online sales transactions made using mobile devices, such as smartphones and tablets. M-commerce includes mobile shopping, mobile banking and mobile payments. Mobile chatbots also provide e-commerce opportunities to businesses, allowing consumers to complete transactions with companies via voice or text conversations.

Advantages and disadvantages of e-commerce

Benefits of e-commerce include its around-the-clock availability, the speed of access, the wide availability of goods and services for the consumer, easy accessibility and international reach.

- **Availability.** Aside from outages or scheduled maintenance, e-commerce sites are available 24x7, allowing visitors to browse and shop at anytime. Brick-and-mortar businesses tend to open for a fixed number of hours and may even close entirely on certain days.
- **Speed of access.** While shoppers in a physical store can be slowed by crowds, e-commerce sites run quickly, which is determined by compute and bandwidth considerations on both consumer device and e-commerce site. Product pages and shopping cart pages load in a few seconds or less. An e-commerce transaction can comprise a few clicks and take less than five minutes.

- **Wide availability.** Amazon's first slogan was "Earth's Biggest Bookstore." They could make this claim because they were an e-commerce site and not a physical store that had to stock each book on its shelves. E-commerce enables brands to make a wide array of products available, which are then shipped from a warehouse after a purchase is made. Customers will likely have more success finding what they want.
- **Easy accessibility.** Customers shopping a physical store may have a hard time determining which aisle a particular product is in. In e-commerce, visitors can browse product category pages and use the site search feature to find the product immediately.
- **International reach.** Brick-and-mortar businesses sell to customers who physically visit their stores. With e-commerce, businesses can sell to any customer who can access the web. E-commerce has the potential to extend a business' customer base.
- **Lower cost.** Pure play e-commerce businesses avoid the cost associated with physical stores, such as rent, inventory and cashiers, although they may incur shipping and warehouse costs.
- **Personalization and product recommendations.** E-commerce sites can track visitors' browse, search and purchase history. They can use this data to present useful and personalized product recommendations, and obtain valuable insights about target markets. Examples include the sections of Amazon product pages labeled "Frequently bought together" and "Customers who viewed this item also viewed."

The perceived disadvantages of e-commerce include sometimes limited customer service, consumers not being able to see or touch a product prior to purchase and the wait time for product shipping.

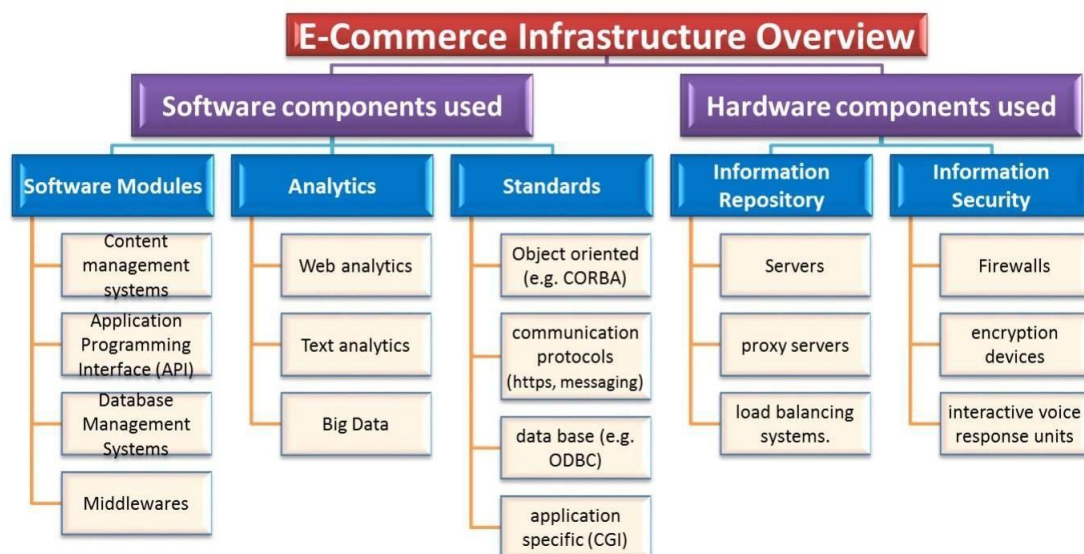
- **Limited customer service.** If a customer has a question or issue in a physical store, he or she can see a clerk, cashier or store manager for help. In an e-commerce store, customer service may be limited: The site may only provide support during certain hours of the day, or a call to a customer service phone number may keep the customer on hold.
- **Not being able to touch or see.** While images on a webpage can provide a good sense about a product, it's different from experiencing it "directly," such as playing music on speakers, assessing the picture quality of a television or trying on a shirt or dress. E-commerce can lead consumers to receive products that differ from their expectations, which leads to returns.
- **Wait time.** If a customer sees an item that he or she likes in a store, the customer pays for it and then goes home with it. With e-commerce, there is a wait time for the product to be shipped to the customer's address. Although shipping windows are decreasing as next day delivery is now quite common, it's not instantaneous.
- **Security.** Skilled hackers can create authentic-looking websites that claim to sell well-known products. Instead, the site sends customers counterfeit or imitation versions of those products -- or, simply collects customers' credit card information. Legitimate e-

commerce sites also carry risk, especially when customers store their credit card information with the retailer to make future purchases easier.

E-Commerce Infrastructure

E-Commerce Infrastructure identifies the functionalities of the Hardware and Software components, specifies the corresponding service level requirements, and describes the management and operations of the whole system. It may comprise briefly of the following components at a very abstract level.

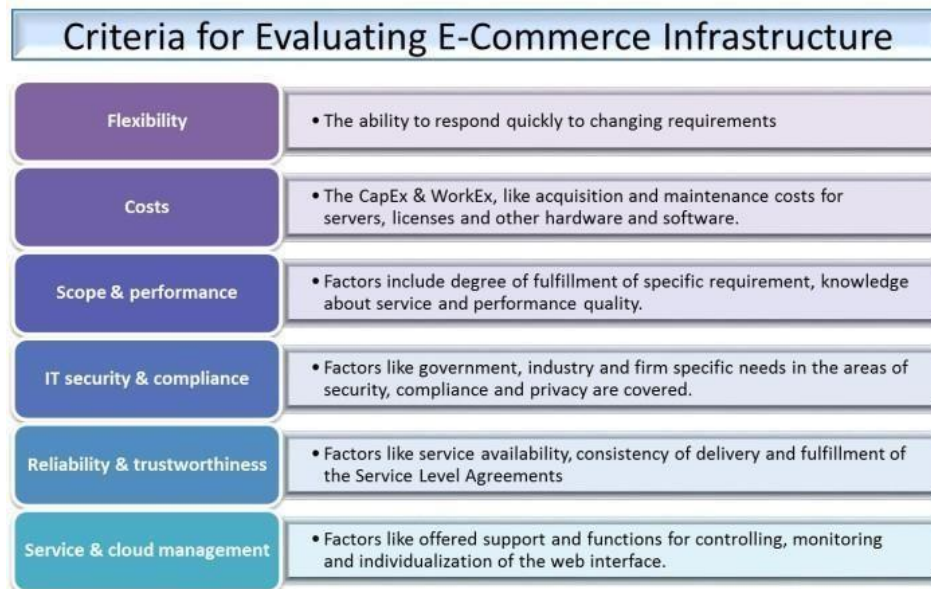
- **Software components used:** Content management systems, Web analytics, Text analytics, Application Programming Interface (API), Database server, Middlewares etc. Object oriented (e.g. CORBA), Transaction processing, communication (https, messaging), data base (e.g. ODBC), application middleware (CGI)
- **Hardware components used:** Servers, proxy servers, load balancing systems. Firewalls, encryption devices and interactive voice response units etc.



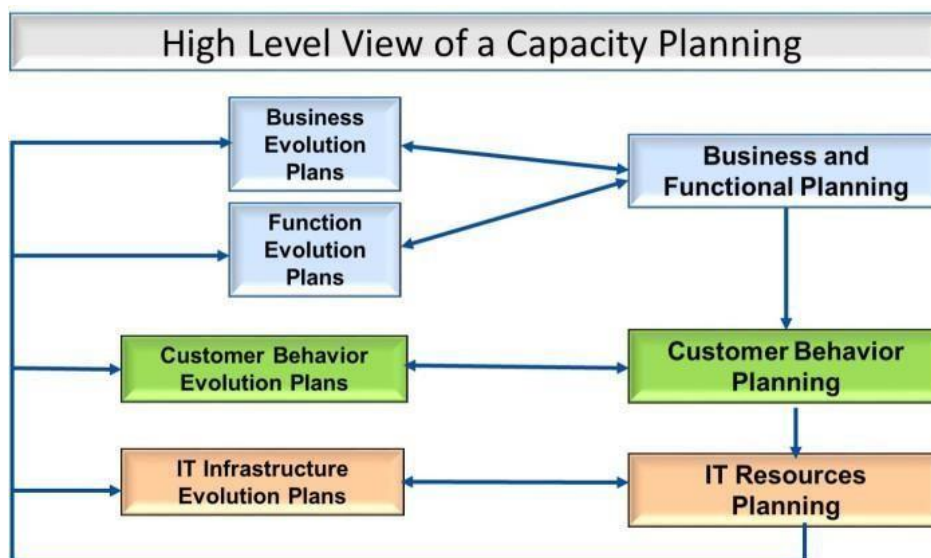
Some of the major components, which a techno-functional professional in the domain is expected to know are as follows:

- **Middleware:** Systems that resides between the client (user) and the server (database and application resources). These could be data access components, communication protocols, specialized servers, or a mix of all.
- **Directory services:** Email Directory Services enables users to locate other users for sending emails. LAN Directory Services facilitates functions like connecting to the web, sharing printers, LAN chats, LAN based KMS
- **Lookup Database:** This is the database that stores information about network resources and user profiles. Enables usage of network resources based on entitlements.
- **Meta-Directories:** Facilitates the flow of data between one or more directory services and databases. Enables synchronization of data across databases or data warehouses
- **Groupware:** Facilitate the automation and streamlining of business processes not implemented in legacy/ERP systems. Group communications and information sharing enabling collaboration between teams and individuals

- **Internet Domain Name Service (DNS):** DNS facilitates the unique identification of an organization or entity on the Internet. DNS maps the domain name of an organization to its IP address



Capacity planning would address these requirements through a cycle of workflows of analysis in a multi-stage approach.



During the Business & Functional Planning, focus would be on the following components:

- **Interaction Model:** It focuses on how a user interacts with the e-Business site to execute the function. Example: two consecutive HTML forms may be needed to implement the function of online application to a course.
- **Web Technology Used:** Different technologies may be used to implement an e-Commerce function. E.g. HTML forms, Java Applets, Active X controls. Suitability of technology chosen to fulfil functional needs would have paramount importance.

- **Use of User Credentials for Authentication:** This information specifies if an authentication protocol such as SSL is used to implement the e-Commerce function. This would have significant importance in terms of security and information assurance.

Experiment -28

Aim: What are various online payment methods and explain their Security issues.

Types of Payment Methods for ECommerce

Credit/Debit card payments:

Payments via cards are one of the most widely used and popular methods not only in India but on the international level. As a global payment solution, by enabling payment acceptance via cards merchants can reach out to an international market.

Credit cards are simple to use and secure. The customer just has to enter the card number, expiry date, and CVV, which has been introduced as a precautionary measure. The CVV helps detect fraud by comparing customer details and the CVV number. Coming to debit cards, they can be considered the next popular method for eCommerce payments.

Debit cards are usually preferred by customers who shop online within their financial limits. The main difference between credit and debit card is with a debit card one can only pay with the money that is already in the bank account, whereas in the case of a credit card, the spent amount is billed, and payments are made at the end of the billing period.

Prepaid card payments:

As an alternative for credit/debit cards, prepaid cards are introduced. They usually come in different stored values and the customer has to choose from them. Prepaid cards have virtual currency stored in them. Though the adoption rate of prepaid cards is low, they are gradually becoming popular for certain niche categories.

Bank transfers:

Though not popular nowadays but still bank transfer is considered as an essential payment method for eCommerce. It is considered as 'if all else fails' kind of payment method. Some of the eCommerce stores are also keen on using bank transfer payment options.

Customers enrolled in internet banking can do bank transfers for their online purchases. Bank transfer is the most secure method as the transactions need to be approved and authenticated by the customers. It is a simple way of paying for online purchases and does not require the customer to have a card for payment purposes.

E-Wallets:

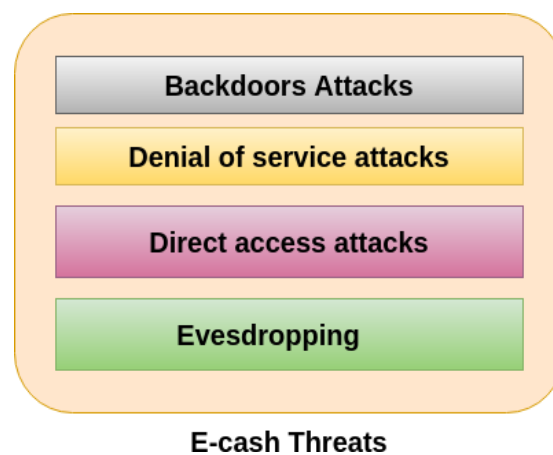
E-wallet is one of the upcoming trends which gives a new shopping experience altogether. The use of e-wallets is becoming popular at an alarming rate. E-Wallets require a sign up from merchants as well as customers. After creating an e-wallet account and linking it to the bank account they can withdraw or deposit funds. The whole procedure with an e-wallet is easy and fast. Considered as an advanced and instant digital payment method, e-wallets can be integrated with mobile wallets using advanced functionalities like NFC. Prepaid e-wallet

accounts store customer information and multiple credit/debit cards and bank accounts. It needs one-time registration and eliminates the need for re-entering information every time while making payments.

E-cash has four major components-

1. **Issuers** - They can be banks or a non-bank institution.
2. **Customers** - They are the users who spend the e-cash.
3. **Merchants or Traders** - They are the vendors who receive e-cash.
4. **Regulators** - They are related to authorities or state tax agencies.

In e-cash, we stored financial information on the computer, electronic device or on the internet which is vulnerable to the hackers. Some of the major threats related to e-cash system are-



Backdoors Attacks

It is a type of attacks which gives an attacker to unauthorized access to a system by bypasses the normal authentication mechanisms. It works in the background and hides itself from the user that makes it difficult to detect and remove.

Denial of service attacks

A denial-of-service attack (DoS attack) is a security attack in which the attacker takes action that prevents the legitimate (correct) users from accessing the electronic devices. It makes a network resource unavailable to its intended users by temporarily disrupting services of a host connected to the Internet.

Direct Access Attacks

Direct access attack is an attack in which an intruder gains physical access to the computer to perform an unauthorized activity and installing various types of software to compromise security. These types of software loaded with worms and download a huge amount of sensitive data from the target victims.

Eavesdropping

This is an unauthorized way of listening to private communication over the network. It does not interfere with the normal operations of the targeting system so that the sender and the recipient of the messages are not aware that their conversation is tracking.

Credit/Debit card fraud

A credit card allows us to borrow money from a recipient bank to make purchases. The issuer of the credit card has the condition that the cardholder will pay back the borrowed money with an additional agreed-upon charge.

A debit card is of a plastic card which is issued by the financial organization to an account holder who has a savings deposit account that can be used instead of cash to make purchases. The debit card can be used only when the fund is available in the account.

Some of the important threats associated with the debit/credit card are-

ATM (Automated Teller Machine)-

It is the favourite place of the fraudster from there they can steal our card details. Some of the important techniques which the criminals opt for getting hold of our card information is:

Skimming-

It is the process of attaching a data-skimming device in the card reader of the ATM. When the customer swipes their card in the ATM card reader, the information is copied from the magnetic strip to the device. By doing this, the criminals get to know the details of the Card number, name, CVV number, expiry date of the card and other details.

Unwanted Presence-

It is a rule that not more than one user should use the ATM at a time. If we find more than one people lurking around together, the intention behind this is to overlook our card details while we were making our transaction.

Vishing/Phishing

Phishing is an activity in which an intruder obtained the sensitive information of a user such as password, usernames, and credit card details, often for malicious reasons, etc.

Vishing is an activity in which an intruder obtained the sensitive information of a user via sending SMS on mobiles. These SMS and Call appears to be from a reliable source, but in real they are fake. The main objective of vishing and phishing is to get the customer's PIN, account details, and passwords.

Online Transaction

Online transaction can be made by the customer to do shopping and pay their bills over the internet. It is as easy as for the customer, also easy for the customer to hack into our system and steal our sensitive information. Some important ways to steal our confidential information during an online transaction are-

- By downloading software which scans our keystroke and steals our password and card details.
- By redirecting a customer to a fake website which looks like original and steals our sensitive information.
- By using public Wi-Fi

POS Theft

It is commonly done at merchant stores at the time of POS transaction. In this, the salesperson takes the customer card for processing payment and illegally copies the card details for later use.

Experiment - 29

Aim: What is Internet Marketing? Explain various types and their relevance.

Internet marketing is an all-inclusive term for marketing products and services online. This includes a variety of methods and platforms for communicating with customers, such as website, email, social media, and online advertising.

Learn more about internet marketing, its role and importance in business, and how to use it to your benefit.

What Is Internet Marketing?

Internet marketing refers to the strategies used to market products and services online and through other digital means. These can include a variety of online platforms, tools, and content delivery systems, such as:

- Website content and design
- Email marketing
- Social media
- Blogging
- Video/podcasting
- Online ads
- Sponsorships and paid promotions

While internet marketing's apparent purpose is to sell goods and services, or advertising over the internet, it's not the only reason a business will do it.

A company may be marketing online to communicate a message about itself (building its brand) or to conduct research. Online marketing can also be an effective way to identify a target market, discover a marketing segment's wants and needs, build long-term relationships with customers, or establish authority and expertise within an industry.

Experiment -30

Aim: Develop a login based application using Ajax, javascript and JSP.

loginValidation.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>Validate login from database table using jQuery JSP & jQuery</title>
    <script src="jquery-1.4.2.js"></script>
    <script src="loginValidation.js" type="text/javascript"></script>
    <style type="text/css">
      .style1 {
        color: #0000ff;
      }
      .style3 {
        color: #cc0066;
      }
    </style>
  </head>

  <body>
    <div>
      <form id="formL" name="formL" method="post" onsubmit="return false;">
        <span class="style3">
          <strong>Enter Username & Password to Login : </strong></span>
          <br />
          <br />
          <span class="style1">Username :</span><input name="user" id="user" type="text" />
          <br />
          <span class="style1">Password :</span><input name="pass" id="pass" type="password" />
          <br />
          <input name="button" id="button" value="Submit" type="submit" />
        </form>
      </div>
      <div id="targetDiv" style="display: none"></div>
    </body>
  </html>
```

loginValidation.js

```
$(document).ready(function () {
    //global vars
    var userName = $("#user"); //user name field var userPass = $("#pass"); //password field
    //function to check name and comment field function checkCommentsForm(){ if(userName.attr("value") && userPass.attr("value"))
    return true; else
    return false;
}
    //When form submitted
    $("#formL").submit(function () {
        if (checkCommentsForm()) {
            $.ajax({
                type: "post"
                , url: "loginValidation.jsp"
                , data: "user=" + userName.val() + "&pass=" + userPass.val(),
                success: function (msg) { $('#targetDiv').hide(); $('#targetDiv').html("<h3>" + msg + "</h3>").fadeIn("slow"); }
            });
        }
        else alert("Please fill UserName & Password!"); return false;
    });
});
```

loginValidation.jsp

```
<%@ page language="java" import="java.sql.*" %> <%
response.setContentType("text/html"); String
userName=request.getParameter("user"); String
userPass=request.getParameter("pass"); %>
<h2>
    <font color="blue">
        <% String connectionURL = "jdbc:mysql://192.168.10.13:3306/ankdb";
        Connection connection=null; ResultSet rs; String userNam=new String("");
        String passwr=new String(""); String name=new String("");
        response.setContentType("text/html"); try { // Load the database driver
        Class.forName("com.mysql.jdbc.Driver"); // Get a Connection to the database
        connection = DriverManager.getConnection(connectionURL, "root", "root");
        //Add the data into the database String sql = "select user,password from
        jqlogin"; Statement s = connection.createStatement(); s.executeQuery (sql);
        rs = s.getResultSet(); while (rs.next ()) { userNam=rs.getString("user");
        passwr=rs.getString("password");
        if(userNam.equals(userName)&&passwr.equals(userPass)){ name=userNam; } }
        rs.close (); s.close (); }catch(Exception e){ System.out.println("Exception
        is ;"+e); } if(name.equals(userName) { %>
        <font color="red">Welcome</font>
        <% out.println(name); } else{ out.println("You are not an authentic
        person"); } %>
    </font>
</h2>
```

Experiment -31

Aim: Create snake game using HTML5 (Canvas) and javascript. (Tutorial for game development).

To start with, you'll want to create a html file with the following markup:

```
<html> <head> <title>Snake</title> <link href='/reset.css' rel='stylesheet'> <link  
href='/master.css' rel='stylesheet'>  
<script src='snake.js' type='text/javascript'></script>  
</head> <body> <canvas id="canvas" width="400"  
height="300"></canvas> </body> </html>
```

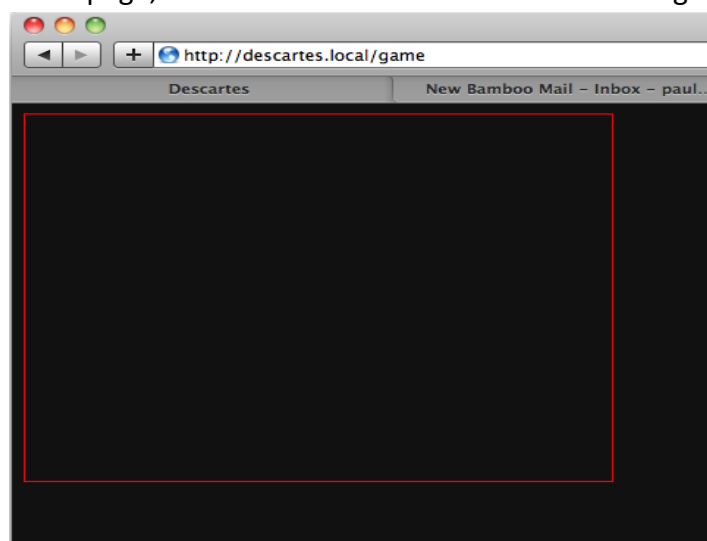
For the reset.css stylesheet:

```
html, body div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a  
abbr, acronym, address, big, cite, code, del, dfn, em, font, img, ins, kdb, q, s, samp, small,  
strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li fieldset, form, label, legend,  
table, caption, tbody, tfoot, thead, tr, th, td { margin: 0; padding: 0; font-size: 100%;  
vertical-align: baseline; border: 0; outline: 0; background: transparent; } ol, ul { list-style:  
none; } blockquote, q { quotes: none; } a, a:hover { text-decoration: none; } table { border-  
collapse: collapse; border-spacing: none; }
```

For the master.css stylesheet:

```
body { background: #111; } canvas { border: solid 1px red; }
```

This sets up a black web page, with a red border around the canvas tag.



Detecting Canvas support

Unfortunately, not all web browsers support canvas yet (*cough* IE), so the first thing we should do is check if the browser supports it. The `snake.js` file is where we will put the code to check for canvas support.

```
function checkSupported() { canvas = document.getElementById('canvas'); if  
(canvas.getContext){ctx  
= canvas.getContext('2d'); // Canvas is supported } else { // Canvas is not supported  
alert("We're sorry, but your browser does not support the canvas tag. Please use any web  
browser other than Internet Explorer."); } }
```

And to make this code execute when the web page loads, adjust the body tag so it reads like this:

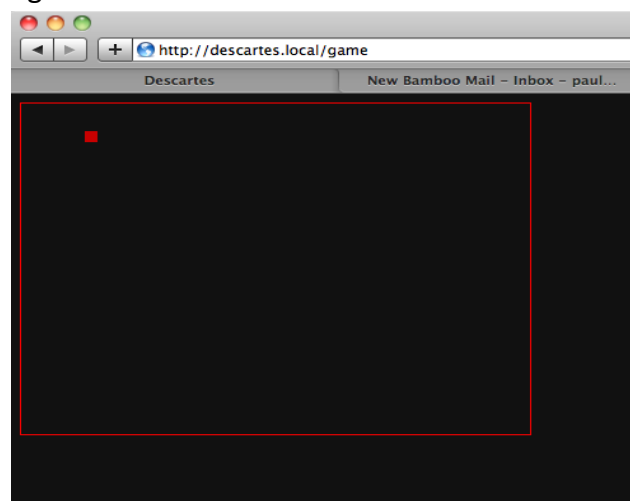
```
<body onload="checkSupported();">
```

Drawing

Now, I'm interested in drawing objects on the canvas, namely the snake. From what I remember from my first mobile phone (a Nokia 3210), the snake was a chain of squares which you could move in four directions, so let's start by drawing a square:

```
// This sets the fill color to red ctx.fillStyle = "rgb(200,0,0)"; // This sets some variables for  
demonstration purposes var x = 50; var y = 50; var width = 10; var height = 10; // This  
draws a square with the parameters from the variables set above ctx.fillRect(x, y, width,  
height);
```

If you put this javascript code in the canvas-supported section of the IF block, then you should see the following result below:



A nice little red square somewhere in the top left corner of the canvas. This shows you how easy it is to draw rectangular objects on the canvas. There are also options to draw the stroke of a rectangle, as well as clear a rectangle: `strokeRect(x,y,w,h); clearRect(x,y,w,h);`

Movement

Now that I know how to draw squares, I want to move that square, or at least give the impression of a moving square. The snake can move up, down, left and right. In javascript, we want to capture keyboard keys being hit, and use those keys to interact with the canvas.

To capture the keystrokes, I use the following javascript code:

```
document.onkeydown = function(event) { var keyCode; if(event  
== null) { keyCode = window.event.keyCode; } else { keyCode = event.keyCode; }  
switch(keyCode) { // left case 37: // action when pressing left key break; // up case 38: //  
action when pressing up key break; // right case 39: // action when pressing right key  
break; // down case 40: // action when pressing down key break; default: break; } }
```

This code sets up the ability to capture the user pressing the up, down, left, and right keys on the keyboard, providing the controls needed to move the Snake around the canvas.

When we press a key, we want to draw a new square going in that direction. To match the direction, we need to know the following:

- What is the current position of the snake's head
- What direction do we want to go, which is given by the key press

So, we need a variable to record the position of the snake's head, and when we press a key, we draw a new square, whose position is the offset of the direction from the current position of the snake's head.

In the canvas-supported part of the IF block, fill in the following code:

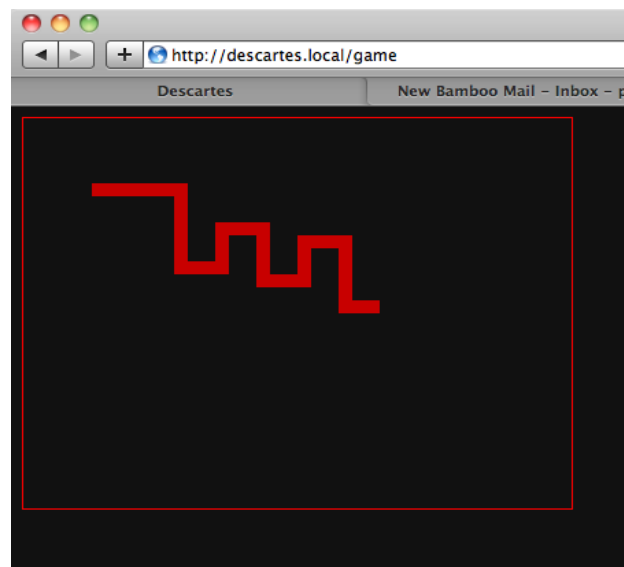
```
// The current position of the Snake's head, as xycoordinates this.currentPosition = [50, 50];  
// Sets the grid dimensions as one value this.gridSize = 10;
```

We're going to use these global variables to set the starting point for the snake, as well as the grid size, which defines both how far they move in a direction, as well as how large each square will be.

Now, we want to adjust the keydown switch statement to generate new xy coordinates, which we will then provide to the fillRect argument to create a new square:

```
switch(keyCode) { // left case 37: // set new position, and draw square at that position.  
  currentPosition['x'] = currentPosition['x'] - gridSize; ctx.fillRect(currentPosition['x'],  
  currentPosition['y'], gridSize, gridSize); break; // up case 38: currentPosition['y'] =  
  currentPosition['y'] - gridSize; ctx.fillRect(currentPosition['x'], currentPosition['y'],  
  gridSize, gridSize); break; // right case 39: currentPosition['x'] = currentPosition['x'] +  
  gridSize; ctx.fillRect(currentPosition['x'], currentPosition['y'], gridSize, gridSize); break;  
  // down case 40: currentPosition['y'] = currentPosition['y'] + gridSize;  
  ctx.fillRect(currentPosition['x'], currentPosition['y'], gridSize, gridSize); break; default:  
  break; }
```

You should see something like this below:



If you were successful, you should be able to draw a line that can go in all four directions. You can also change the code that drew the initial square in the IF block so that it looks like this:

```
ctx.fillRect(currentPosition['x'], currentPosition['y'], gridSize, gridSize);
```


You'll notice that this line is commonly repeated in several places within the code. I'm going to create a function for convenience, which does seem a bit much for a 1 line method, but will make the code easier to read:

```
function drawSnake() { ctx.fillRect(currentPosition['x'], currentPosition['y'], gridSize, gridSize); }
```

You can go through the code and replace those lines as appropriate.

Snake keeps moving

When you play the Snake mobile phone game, you can control the direction of the snake, but not it's speed, that remains a constant, and means that you have to focus on guiding the snake around the area so as to not bump into itself. It's a bit like the guy in Australia who had to steer his car as the cruise control locked.

To provide this functionality, we need to do two things:

- Have a new variable record what direction the snake is going in
- Make a timed loop that will execute a function to move the snake in that direction

So, first things first, we create a new global variable, called direction:

```
direction = 'right';
```

Now, we need to execute an animation that moves the Snake in that direction, put this in the canvas-supported IF block:

```
setInterval(moveSnake,100);
```

Now, add this function somewhere towards the bottom of the code:

```
function moveSnake(){ switch(direction){ case 'up': currentPosition['y'] = currentPosition['y'] - gridSize; drawSnake(); break; case 'down': currentPosition['y'] = currentPosition['y'] + gridSize; drawSnake(); break; case 'left': currentPosition['x'] = currentPosition['x'] - gridSize; drawSnake(); break; case 'right': currentPosition['x'] = currentPosition['x'] + gridSize; drawSnake(); break; } }
```

You should see the Snake moving right, which is good, but it seems that despite our insisting, the Snake is determined to go right, right out of the canvas in fact. This highlights that a) we haven't associated keystrokes with a change in direction, and b) The snake is free to escape the canvas, which it shouldn't be. The latter issue we will deal with later, but first, let's associate the keystrokes with the direction.

Simply set the direction variable with the keystroke:

```
switch(keyCode) { // left case 37: // set new position, and draw square at that position.  
  direction = 'left'; currentPosition['x'] = currentPosition['x'] - gridSize; drawSnake();  
  break; }
```

Now, let's set the boundaries for the Snake to move within.

Setting boundary on movement

We want our Snake's movements to be restricted to within the boundaries of the canvas tag, therefore we need to do the following:

- Find out what the canvas boundaries are
- Detect if the snake's position will go out of the boundary
- and if it will, redirect it to an alternative direction

The first step is simple:

```
canvas.width; canvas.height;
```

The html attributes we defined for the canvas tag are available in javascript, so we will refer to them in that way. When the snake moves down or right, we then want to check whether the new position's x or y value is greater than the canvas' width or height. If it is, then we don't draw the new square. Instead, we choose to move in a different direction, so as to keep the momentum going. The same principle takes effect if the current position's x or y value is less than 0 when the snake moves up or left.

I have to admit, at this point I took the opportunity to refactor my code, so that I kept the logic in one place. I started by turning the calculated positions for up, down, left, and right into functions:

```
function leftPosition(){ return currentPosition['x'] - gridSize; } function  
rightPosition(){ return currentPosition['x'] + gridSize; } function upPosition(){ return  
currentPosition['y'] - gridSize; } function downPosition(){ return currentPosition['y']  
+ gridSize; }
```

The calculated position functions would then be used for the logic to determine if the snake could move in a certain direction, as well as for setting the current position for which to move to.

The next step would be to wrap the boundary logic, direction setter, current position setter and drawShape function call within named functions:

```
function moveUp(){ if (upPosition() >= 0) {executeMove('up', 'y', upPosition()); } } function  
moveDown(){ if (downPosition() < canvas.height) { executeMove('down', 'y',  
downPosition()); } } function moveLeft(){ if (leftPosition()  
>= 0) { executeMove('left', 'x', leftPosition()); } } function moveRight(){ if (rightPosition() <  
canvas.width) { executeMove('right', 'x', rightPosition()); } } function executeMove(dirValue,  
axisType, axisValue) { direction = dirValue; currentPosition[axisType] = axisValue;  
drawSnake();  
}
```

The first four methods are similar in setup. The first line calls an IF statement to check if the movement in that direction will set the current position inside of the canvas area. If it does, then the second line is executed, pass 3 parameters to the executeMove function.

The executeMove function is a refactoring of the same code that was encapsulated in the moveUp/moveDown/moveLeft/moveRight functions. It sets the direction, then it sets the currentPosition according to the axis type and the axis value passed in, then it draws the square representing the head of the snake.

Finally, we can now clean up a lot of common code, so now our keystroke calls and automated snake movements look like this:

```
document.onkeydown = function(event) { var keyCode; if(event  
== null) { keyCode = window.event.keyCode; } else { keyCode = event.keyCode; }  
switch(keyCode) { // left case 37: moveLeft(); break; // up case 38: moveUp(); break; //  
right case 39: moveRight(); break; // down case 40: moveDown(); break; default: break; } }  
function moveSnake(){ switch(direction){ case 'up': moveUp(); break; case 'down':  
moveDown(); break; case 'left': moveLeft(); break; case 'right': moveRight(); break; } }
```

Nice and clean. What you should see now is that the Snake moves by itself, and when it gets to the edge of the canvas, it actually stops at that point. You can test this by waiting a few seconds after the snake has reaches the edge, then press a key going to either side of the snake, and you will notice that the snake then traverses immediately from that point in the given direction, it has not escaped from the canvas.

This is good so far, but what we want to happen is for the snake to automatically move to either side once it hits an edge and cannot go further. We do this by using the following code:

```
function whichWayToGo(axisType){ if (axisType=='x') { a = (currentPosition['x'] > canvas.width / 2) ? moveLeft() : moveRight(); } else { a = (currentPosition['y'] > canvas.height / 2) ? moveUp() : moveDown(); } }
```

This code takes an axis type, and then works out which way to go based on the current position. So, for example if the snake has hit the right edge of the canvas, then the function is passed a y parameter, telling the function it needs to move either up or down. This function will then check if the current position is above or below the middle of the canvas. If it's above, it will move down, and vice versa. The same principles apply for the horizontal case.

So, with that function, adjust the moveUp/moveDown/moveLeft/moveRight functions as shown below:

```
function moveUp(){ if (upPosition() >= 0) { executeMove('up', 'y', upPosition()); } else { whichWayToGo('x'); } } function moveDown(){ if (downPosition() < canvas.height) { executeMove('down', 'y', downPosition()); } else { whichWayToGo('x'); } } function moveLeft(){ if(leftPosition() >= 0) { executeMove('left', 'x', leftPosition()); } else { whichWayToGo('y'); } } function moveRight(){ if (rightPosition() < canvas.width) { executeMove('right', 'x', rightPosition()); } else { whichWayToGo('y'); } }
```

If you now reload the page, you will see that the Snake keeps moving when it hits an edge, and cleverly opts to head in the direction furthest from an edge.

Pulling the Snake's body along

It's starting to get there, the Snake looks great, even a bit intelligent as he, or she, sees an edge. That's all good, but right now the Snake has a backside that doesn't move, or to quote Homer Simpson (and I kid you not, this is in one of the early episodes), "Marge, you got a butt that won't quit". It needs to drag it's body along, not draw an infinite line, so that's what we're going to do.

You'll notice that I thought about this earlier on when I came up with the drawSnake() function name. I wanted to have a placeholder where I would draw the Snake's body. Remember when I referred to the Snake as looking like 'a chain of squares'? That's exactly what the snake will be, a chain of squares.

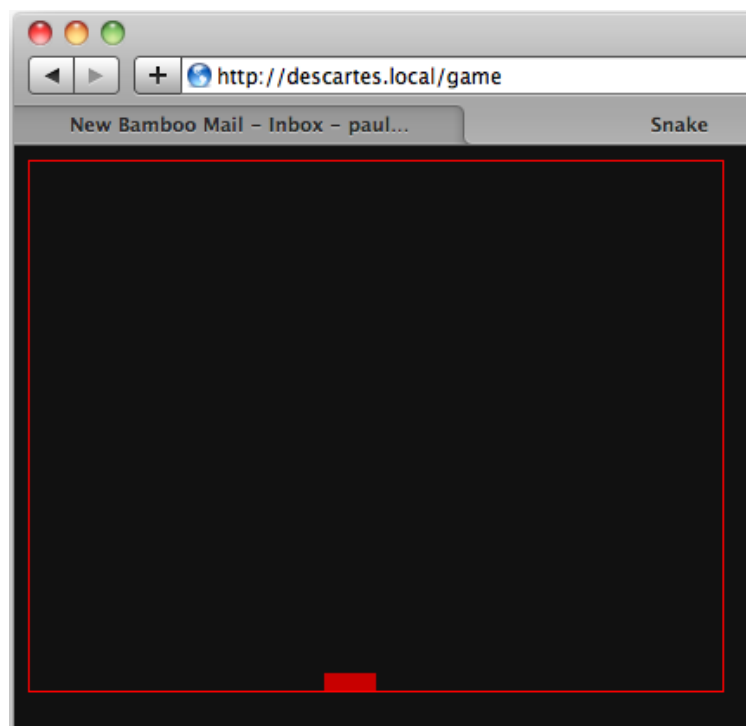
The idea is to have an array of xy coordinates that will represent all the square points which the Snake body occupies:

```
snakeBody = [];
```

and adjust the drawSnake function so it looks like this:

```
function drawSnake() { snakeBody.push([currentPosition['x'], currentPosition['y']]);  
ctx.fillRect(currentPosition['x'], currentPosition['y'], gridSize, gridSize); if (snakeBody.length  
> 3) { var itemToRemove = snakeBody.shift(); ctx.clearRect(itemToRemove[0],  
itemToRemove[1], gridSize, gridSize); } }
```

What happens here is that every time there is a call to drawSnake, the function adds the new position to the snakeBody array, keeping it in memory. Then, when the size of the SnakeBody array grows above 3, the SnakeBody array's first element is removed, and the rectangle is removed. What you should see now is a little snake moving around, with a tiny body rather than an infinitely-long line.



Eat food, grow longer

Now we need to add 1 element to the game which is currently not implemented in any way; we need the food. In the Snake game, little squares of food pop up around the area, and when the Snake passes over it, he grows longer, hence the label 'food'. To add this element to the game, we need to do the following:

- Create a method to draw a food square on the canvas, inserted in a random
- location that is not occupied by the Snake's body When the Snake passes over
- that food item, increase the Snake's body length, and place a food square
- somewhere else

We already know how to draw squares, but how do we place them in a random location, one which is not occupied by the Snake? The former problem is solved by using the Math library, and the latter by detecting whether the snakeBody array contains an array of the random point that is suggested for locating the food square. Here is the code:

```
function makeFoodItem(){ suggestedPoint =
[Math.floor(Math.random()*(canvas.width/gridSize))*gridSize,
Math.floor(Math.random()*(canvas.height/gridSize))*gridSize]; if
(snakeBody.some(hasPoint)) { makeFoodItem(); } else { ctx.fillStyle = "rgb(10,100,0)";
ctx.fillRect(suggestedPoint[0], suggestedPoint[1], gridSize, gridSize); }; } function
hasPoint(element, index, array) { return (element[0] == suggestedPoint[0] &&
element[1]
== suggestedPoint[1]); }
```

It looks ugly, it probably is ugly, so I must confess, I'm not a Javascript guru (that's why I pester Ismael now and then), but I'll explain the logic. First, before we can do anything, we need to generate a random point in the canvas, an array of two numbers well within the range of the canvas' width and height, but also rounded so that the random numbers are divisible by the gridSize, which happens to be 10.

Secondly, we need to check if the suggestedPoint is already occupied by the Snake's body. Unfortunately comparing an array of arrays is not a straightforward procedure in javascript, so I had to create a function called hasPoint, which using Array.some, will return true if any of the snakeBody array's arrays directly match with the suggestedPoint array.

If snakeBody already has the suggestedPoint in its collection, then we run the whole method all over again, and this repeats until the other case occurs, in which case set the fill color to green, and then draw a square of that color, with the suggestedPoint's coordinates. A special note, make sure to move the red fillColor line to the drawSnake function, as demonstrated below:

```
function drawSnake() {snakeBody.push([currentPosition['x'], currentPosition['y']]);  
ctx.fillStyle = "rgb(200,0,0)"; }
```

If you don't, you'll be scratching your head wondering why your beloved snake just turned green. Put `makeFoodItem()`; right in the IF block for the canvas-supported initial startup code, and what you should see is a little green square on the map, awesome. For an added bonus, move your Snake so he eats the food...

It's gone! but wait, we didn't write any code to remove the green square after we ate it? Intentionally, no, unintentionally, yes. The little code which helps to give the impression of the Snake moving by removing the trailing squares also clears the area where the Snake has been, including green squares. How nice is that?

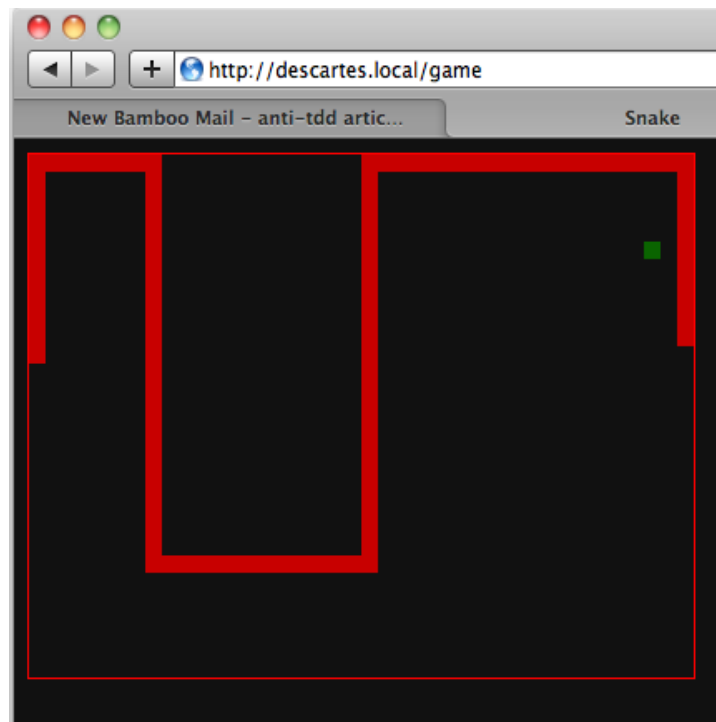
We're close, now we need to make the food reappear in a different place once it has been snacked on, and we need the Snake to grow in length. First, start by putting this code in the canvas-supported IF statement block, precisely in this order:

```
snakeLength = 3; makeFoodItem();drawSnake();
```

The `makeFoodItem()` function must come before the `drawSnake` function, otherwise the code will break because of the `suggestedPoint` variable being missing. You'll also notice that we've defined a `snakeLength` variable, with the value of 3, the same value used to determine when to start bumping the last points from the `snakeBody` array? You can guess where this is heading, and here is the answer:

```
function drawSnake() { snakeBody.push([currentPosition['x'], currentPosition['y']]);  
ctx.fillStyle = "rgb(200,0,0)"; ctx.fillRect(currentPosition['x'], currentPosition['y'], gridSize,  
gridSize); if (snakeBody.length > snakeLength) {var itemToRemove = snakeBody.shift();  
ctx.clearRect(itemToRemove[0], itemToRemove[1], gridSize, gridSize); } if  
(currentPosition['x'] == suggestedPoint[0] && currentPosition['y'] ==  
suggestedPoint[1]) { makeFoodItem(); snakeLength += 1; } }
```

You'll notice 2 changes to the `drawSnake` method. Firstly, it now refers to the `snakeLength` variable instead of just 3, and importantly, the IF statement comparison of the `currentPosition` variable with the `suggestedPoint` variable allows us to capture when the Snake eats the food, so then we generate the next food item on the canvas, and we increment the `snakeLength` variable, which makes the Snake grow longer. Refresh the browser and see it in action.



Game Over

We're very close to finishing the game, except for one small thing, we haven't catered for when the Snake bumps into itself, and thus the end game case. Right now, if the Snake moves over itself, it simply continues its path. What we need to do here is the following:

- Track when the snake traverses over itself
- End the game once that happens

Thankfully, to deal with No. 1, we've already got a solution at hand:

```
function drawSnake() { if (snakeBody.some(hasEatenItself)) { gameOver(); return false; } }  
function hasEatenItself(element, index, array) { return (element[0] == currentPosition['x']  
&& element[1] == currentPosition['y']); }
```

The `hasEatenItself` function checks if the `currentPosition` point is already occupied by the Snake's body (i.e. the `snakeBody` array contains the `currentPosition`), and if it is, it will call a new function called `gameOver`:

```
function gameOver() { var score = (snakeLength - 3)*10; clearInterval(interval);  
snakeBody = []; snakeLength = 3; allowPressKeys = false; alert("Game Over. Your score  
was " + score); ctx.clearRect(0, 0, canvas.width, canvas.height); }
```


The gameOver function does a couple of things:

- Calculates the score, and displays it
- Tells the Snake to stop animating (moving)
- Resets the Snake
- flicks a allowPressKeys variable to false (usage explained later)
- Clears the canvas

The allowPressKeys variable is defined in the IF statement canvas-supported block like this:

```
allowPressKeys = true;
```

and used to disable key press functionality like this:

```
document.onkeydown = function(event) { if(!allowPressKeys){ return null; } }
```

That way, the user cannot move the Snake once the game is over. At this point, we have effectively replicated the Snake game.

A little spit and polish

Not quite. Yes we could ship it and it would work, but how does the user start up a new game? or pause it? and what about displaying the score in real time? We're going to add a few more features before we call this game finished.

Before we add restart and pause functionality, we've got a small usability flaw in the game. If you press a key that is going in the opposite direction to the Snake, the game ends since it's effectively trying to go over itself. Logically this is the correct outcome for the Snake, but when you accidentally go backwards or hit the keys in the wrong order when trying to execute a sharp u-turn, then it becomes annoying.

The solution here is to ignore any keystroke going in the opposite direction to the Snake, like this:

```
switch(keyCode) { case 37: if (direction != "right"){ moveLeft(); } break; case 38: if (direction != "down"){ moveUp(); } break; case 39: if (direction != "left"){ moveRight(); } break; case 40: if (direction != "up"){ moveDown(); } break; }
```

You simply don't execute the move if the Snake is going in the opposite direction. Try it now, you'll notice that the Snake continues in its current direction if you attempt to make it go backwards.

Next, we want to be able to pause the game.

```
function pause(){ clearInterval(interval); allowPressKeys = false; }
```

This stops the animation of the moving snake, and disables keypress movements, effectively pausing the game for the user. To resume the game, the reverse happens: `function play(){ interval = setInterval(moveSnake,100); allowPressKeys = true; }`

With this functionality isolated, you can now DRY up some of the code in other places:

```
function gameOver(){ var score = (snakeLength - 3)*10; pause(); alert("Game Over. Your score was "+ score); ctx.clearRect(0,0, canvas.width, canvas.height); }
```

And we can now create a start function that can be used to initialize a new game.

```
if (canvas.getContext){ ctx = canvas.getContext('2d'); this.gridSize = 10; start(); }  
function start(){ ctx.clearRect(0,0, canvas.width, canvas.height); this.currentPosition = {'x':50, 'y':50}; snakeBody = []; snakeLength = 3; makeFoodItem(); drawSnake(); direction = 'right'; play(); }
```

The start function encapsulates all the methods needed to setup a new game. However, executing a game whilst another game is in progress will cause an error; the game will restart, but the snake will start to travel faster and faster as a result. What we need is a way to remove the existing interval variable, and then start the game:

```
function restart(){ pause(); start(); }
```

You can test this methods using the error console in Safari, or by using Firebug's javascript console for Firefox.

Now would be a good time to add some buttons to the web page to provide the user with access to these functions. In terms of context, it would make sense to have the restart and resume buttons hidden by default, and the pause button displayed by default. Once the pause button is pressed, it is hidden, and the other 2 buttons are shown.

```
<body onload='checkSupported();'> <span id='logo'>SNAKE</span>  
<div id='play_menu'> <button onclick="pause();  
document.getElementById('play_menu').style.display='none';docu  
ment.getElementById('pause_menu').style.display='block';">Pause</button> </div> <div  
id='pause_menu'> <button onclick=
```

```
"restart();document.getElementById('play_menu').style.display=
'block';document.getElementById('pause_menu').style.display='none';">Restart</button>
<button onclick="play(); document.getElementById('play_menu').style.display='block';doc
ument.getElementById('pause_menu').style.display='none';">Resume</button> </div>
```

Also add this css rule for displaying the logo and hiding the pause menu buttons by default:

```
#logo { font-family: helvetica; font-size: 1.4em; } #pause_menu { display:
none; }
```

We're now able to display context-relevant buttons to the user as they play the game, oh, and when the game finishes:

```
<div id='restart_menu'> <button
onclick="restart();document.getElementById('play_menu').style.
display='block';document.getElementById('restart_menu').style.
display='none';">Restart</button> </div>
```

and the css:

```
#pause_menu, #restart_menu { display: none; }
```

and the Snake js:

```
function gameOver(){ var score = (snakeLength - 3)*10; pause(); alert("Game Over. Your
score was "+ score); ctx.clearRect(0,0, canvas.width, canvas.height);
document.getElementById('play_menu').style.display='none';
document.getElementById('restart_menu').style.display='block';
}
```

Now when a game ends, you can restart the game with ease. I won't say that this is the most efficient way of doing this, but it does the job. The final thing to do is to display the score in real time.

```
<div id="score_container"> Score: <span id="score">0</span>
</div>
```

The css:

```
#score_container { float: right; display: inline; }
```

The snake js code:

```
function updateScore(){ var score = (snakeLength - 3)*10  
document.getElementById('score').innerText = score; }
```

We have a new method for updating the score on the page, and we use this in two places, the first being the drawSnake function, when the Snake eats the food: `function drawSnake() { if (snakeBody.some(hasEatenItself)) { gameOver(); return false; } snakeBody.push([currentPosition['x'], currentPosition['y']]); ctx.fillStyle = "rgb(200,0,0)"; ctx.fillRect(currentPosition['x'], currentPosition['y'], gridSize, gridSize); if (snakeBody.length > snakeLength) {var itemToRemove = snakeBody.shift(); ctx.clearRect(itemToRemove[0], itemToRemove[1], gridSize, gridSize); } if (currentPosition['x'] == suggestedPoint[0] && currentPosition['y'] == suggestedPoint[1]) { makeFoodItem(); snakeLength += 1; updateScore(); } }`

and when the game is started/restarted:

```
function start(){ ctx.clearRect(0,0, canvas.width, canvas.height); this.currentPosition =  
{'x':50, 'y':50}; snakeBody = []; snakeLength = 3; updateScore(); makeFoodItem();  
drawSnake(); direction = 'right'; play(); }
```

Now, when you play the game, you should see the score updated when you eat food, and reset whenever you press the restart button:

Experiment - 32

Aim: Describe how https can be configured on web server to ensure security requirements of Website.

HTTP transfers data as plain text between the client and server. Therefore, anyone who has access to any network segment between you and the server -- on your network, on the server's network or any place in between -- is able to view the contents of your web surfing.

Use HTTPS to protect data relating to financial transactions, personally identifiable information or any other sensitive data, as well as to avoid having browsers flag your site as insecure. HTTPS enables website encryption by running HTTP over the Transport Layer Security (TLS) protocol. Even though the SSL protocol was replaced 20 years ago by TLS, these certificates are still often referred to as SSL certificates.

Here's a simplified view of how it works:

1. You start your web browser and request a secure page by using the https:// prefix on the URL.
2. Your web browser contacts the web server on the HTTPS port -- TCP port 443 - and requests a secure connection.
3. The server responds with a copy of its SSL certificate.
4. Your web browser uses the certificate to verify the identity of the remote server and extract the remote server's public key.
5. Your web browser creates a session key, encrypts it with the server's public key and sends the encrypted key to the server.
6. The server uses its private key to decrypt the session key.
7. The client and server use the session key to encrypt all further communications.

While HTTPS sessions can be reliably considered secure from eavesdropping attacks, HTTPS by itself does not protect against any other types of attack. Site administrators must still take an active role in preventing and mitigating cross-site scripting, injection and many other attacks that target application or other website vulnerabilities.