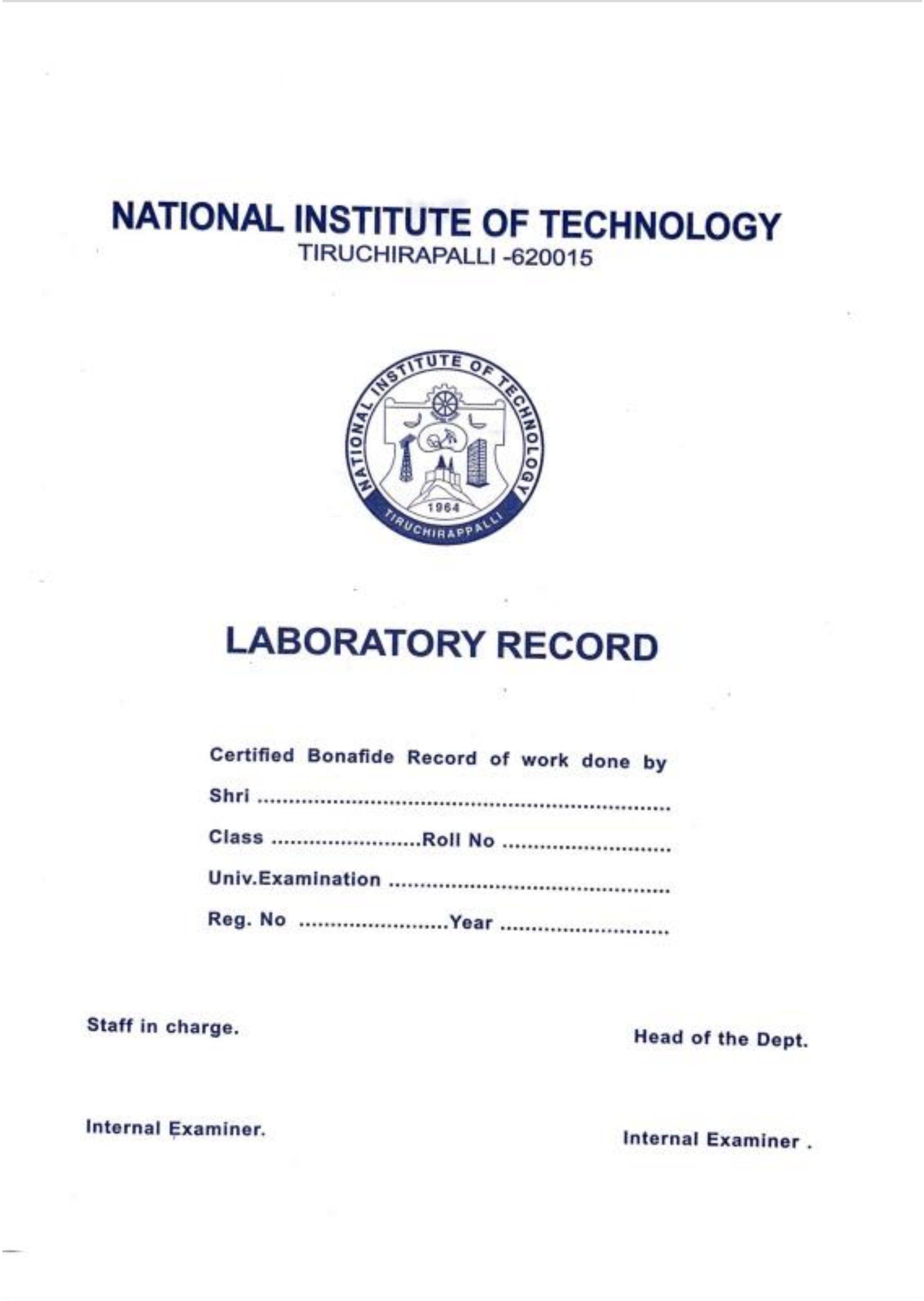
NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPALLI

Application Development Lab

CSLR62

Name: Haneel Kumar Nagineni
Roll Number: 106120073



NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPALLI -620015

LABORATORY RECORD

Certified Bonafide Record of work done by

Shri ....

Class ....Roll No ....

Univ.Examination ....

Reg. No ....Year ....

Staff in charge.

Head of the Dept.

Internal Examiner.

Internal Examiner .

Page of Contents

Creation of Static Webpages using HTML

Ex. No: 1

Date:

Problem Statement: To design static webpages using HTML and utilise the concept of navigation frames, nested lists, and image map.

Algorithm:

1. Designing a static webpage:
    - Create an HTML document.
    - Add a "body" element.
    - Add an "h1" heading element with the text "Table example page".
    - Add a "p" paragraph element with the text "Here is a more complex sample table" and centre it.
    - Create a table with a border attribute set to 1.
    - Create table rows and data cells for the table content, including an image, table headers, and table data.
    - End the table, body, and HTML elements.
2. Designing a webpage to display profile:
    - Create an HTML document.
    - Add a "head" element with a "title" element inside, and a "link" element to reference a CSS stylesheet.
    - Add a "body" element.
    - Add an "h1" element with your name and a brief introduction.
    - Add an "img" element with a link to your digital image.
    - Create an "a" element around the image, with the "href" attribute set to the URL of the new page that opens when the image is clicked.
    - Create a new HTML document for the linked page.
    - Add links to academic profile, list of mini projects done, and personal profile on the new page.
    - Create an "a" element for each link, with the "href" attribute set to the URL of the respective page.
    - Create a "HOME" link on each of the linked pages, with the "href" attribute set to the URL of the first page.
    - End the body and HTML elements for both pages.
3. Designing a webpage using the concept of navigation frames:
    - Create an HTML document with the "frameset" element.

- Add a "frame" element with the "src" attribute set to a new HTML document for the left navigation frame.
- Add a "frame" element with the "src" attribute set to a new HTML document for the right details frame.
- In the left navigation frame HTML document:
    - Add a "ul" element with "li" elements for each flower link.
    - Create an "a" element for each flower link, with the "href" attribute set to the URL of the respective flower details page.
- In the flower details HTML documents:
    - Add an "h1" element with the name of the flower.
    - Add a "p" element with details about the flower.
- End the HTML documents with the "body" and "html" elements.

4. Designing a webpage using the concept of nested lists:

- Create an HTML document.
- Add a "body" element.
- Add an "ul" element with the "type" attribute set to "disc".
- Add an "li" element with the text "Coffee" inside.
- Add another "li" element with the text "Tea" inside.
- Create another "ul" element inside the previous "li" element, with the "type" attribute set to "circle".
- Add an "li" element with the text "Black tea" inside the nested "ul" element.
- Add another "li" element with the text "Green tea" inside the nested "ul" element.
- Create another "ul" element inside the previous "li" element, with the "type" attribute set to "square".
- Add an "li" element with the text "China" inside the doubly nested "ul" element.
- Add another "li" element with the text "Africa" inside the doubly nested "ul" element.
- Add another "li" element with the text "Milk" outside the nested "ul" element.
- End the "body" and "html" elements.

5. Designing a webpage to embed an image map:

- Create an HTML document.
- Add a "head" and "body" element.
- Inside the "head" element, add a "title" element with the title of the page.
- Inside the "body" element, add an "img" element with the "src" attribute set to the image file path and the "usemap" attribute set to the name of the image map.
- Add a "map" element with the "name" attribute set to the name of the image map.
- Inside the "map" element, add an "area" element for each hotspot in the image.

- For each "area" element, set the "shape" attribute to the shape of the hotspot (e.g. "rect" for rectangular hotspots, "circle" for circular hotspots, or "poly" for polygonal hotspots).
- Set the "coords" attribute to define the coordinates of the hotspot.
- Set the "href" attribute to the URL of the page or section to display when the hotspot is clicked.
- Optionally, set the "alt" attribute to provide alternative text for the hotspot.
- End the "map" element.
- Add the related information in the web page for each hotspot.
- Add styles to the webpage if needed.
- End the "body" and "html" elements.

Code:

1. Designing a static webpage:

```
<html>
  <body>
    <h1>Table example page</h1>
    <p align="center">Here is a more complex sample table</p>
    <table border="1">
      <tr>
        <td rowspan="2" align="center">
          <img src="Forest.jpg" width="40%" height="40%" />
        </td>
        <td colspan="4" align="center">
          <h2>Camelid comparison</h2>
          Approximate as of 9/2002
        </td>
      </tr>
      <tr>
        <td># of Humps</td>
        <td>Indigenious region</td>
        <td>Spits?</td>
        <td>Produces wool?</td>
      </tr>
      <tr>
        <td align="right">Camels (bactrian)</td>
        <td>2</td>
        <td>Africa/Asia</td>
        <td rowspan="2">Llama</td>
        <td rowspan="2">Llama</td>
      </tr>
      <tr>
        <td align="right">Llamas</td>
        <td>1</td>
        <td>Andes Mountains</td>
    </tr>
   </table>
  </body>
</html>
```

2. Designing a webpage to display profile:

- Image.html:

```
<html>
  <body>
   <a href="Profiles.html">
    <img src="Pic.jpg" width="40%" height="60%" />
   </a>
  </body>
</html>
```

- Profiles.html:

```
<html>
  <body>
   <a href="acad.html"> Academic profile </a> <br />
   <a href="projects.html"> List of projects done </a> <br />
   <a href="personal.html"> Personal profile </a> <br />
   <a href="Image.html"> Home </a>
  </body>
</html>
```

3. Designing a webpage using the concept of navigation frames:

- Frames.html:

```
<html>
  <frameset cols="25%,*">
   <frame src="Left.html" />
   <frame src="Right.html" name="abc" />
  </frameset>
</html>
```

- Left.html:

```
<html>
  <body>
   MENU
   <ul>
    <li><a href="Rose.html" target="abc">Rose </a></li>
    <li><a href="Jasmine.html" target="abc">Jasmine </a></li>
    <li><a href="Lotus.html" target="abc">Lotus </a></li>
   </ul>
  </body>
</html>
```

- Right.html:

```
<html>
```

```
 <body>
  Click on the choice of your favorite flower links in the left frame in order
  to view the details of your favorite flower in the right frame.
 </body>
</html>
```

- Rose.html:

```
<html>
 <body>
  Rose is a beautiful flower which denotes the symbol of love.
 </body>
</html>
```

- Jasmine.html:

```
<html>
 <body>
  Jasmine is a white colored flower.
 </body>
</html>
```

- Lotus.html:

```
<html>
 <body>
  Lotus is a white colored flower. It is the national flower of India.
 </body>
</html>
```

4. Designing a webpage using the concept of nested lists:

```
<html>
 <body>
  <ul type="disc">
   <li>Coffee</li>
   <li>Tea</li>
   <ul type="circle">
    <li>Black tea</li>
    <li>Green tea</li>
    <ul type="square">
     <li>China</li>
     <li>Africa</li>
    </ul>
   </ul>
   <li>Milk</li>
  </ul>
 </body>
</html>
```

5. Designing a webpage to embed an image map:

- Forest.html:

```
<html>
 <body>
  <img src="Forest.jpg" width="80%" height="80%" usemap="#nam" />
  <map name="nam">
   <area shape="rect" coords="0,0,200,200" href="Red.html" />
   <area shape="circle" coords="500,500,100" href="Green.html" />
  </map>
 </body>
</html>
```

- Green.html:

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

- Red.html:

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

Sample Output:

1. Designing a static webpage:

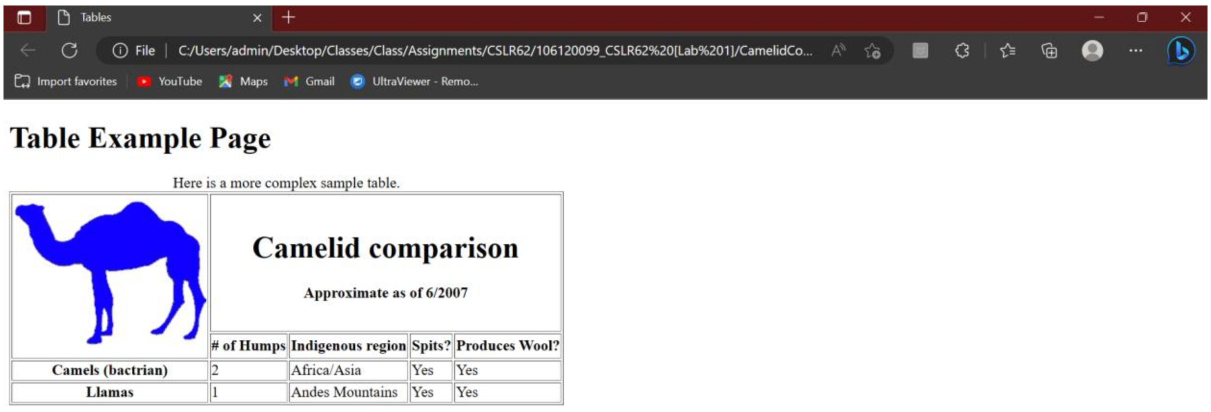

Tables
File
C:/Users/admin/Desktop/Classes/Class/Assignments/CSLR62/106120099_CSLR62%20[Lab%201]/CamelidCo...
Import favorites
YouTube
Maps
Gmail
UltraViewer - Remo...

Table Example Page

Here is a more complex sample table.

Camelid comparison

Approximate as of 6/2007

	# of Humps	Indigenous region	Spits?	Produces Wool?
Camels (bactrian)	2	Africa/Asia	Yes	Yes
Llamas	1	Andes Mountains	Yes	Yes

2. Designing a webpage to display profile:

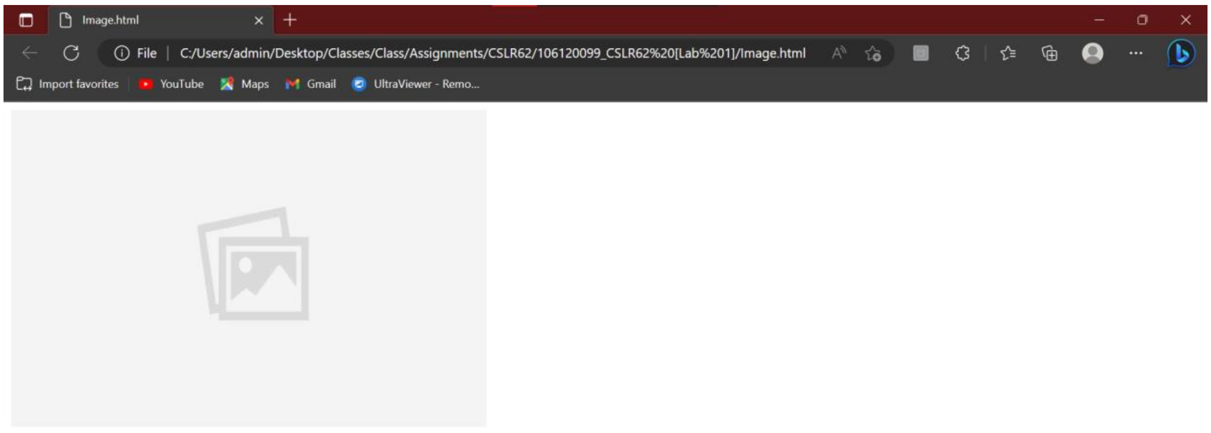
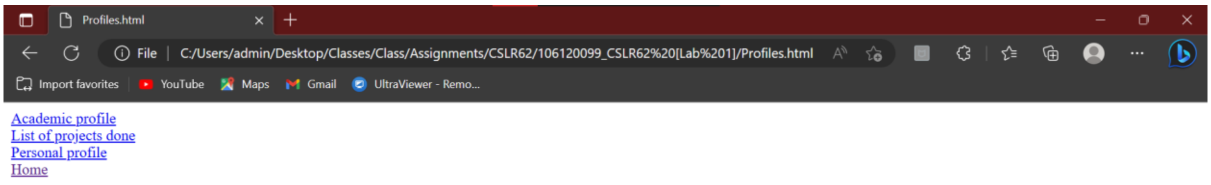
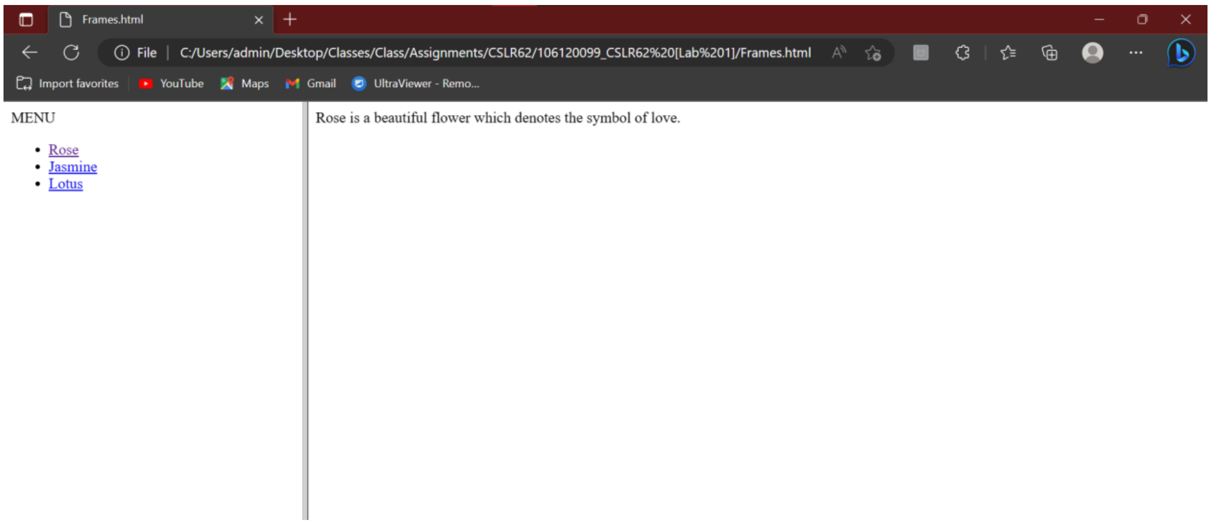


Image.html
C:/Users/admin/Desktop/Classes/Class/Assignments/CSLR62/106120099_CSLR62%20[Lab%201]/Image.html


Profiles.html
C:/Users/admin/Desktop/Classes/Class/Assignments/CSLR62/106120099_CSLR62%20[Lab%201]/Profiles.html

Academic profile
List of projects done
Personal profile
Home

3. Designing a webpage using the concept of navigation frames:

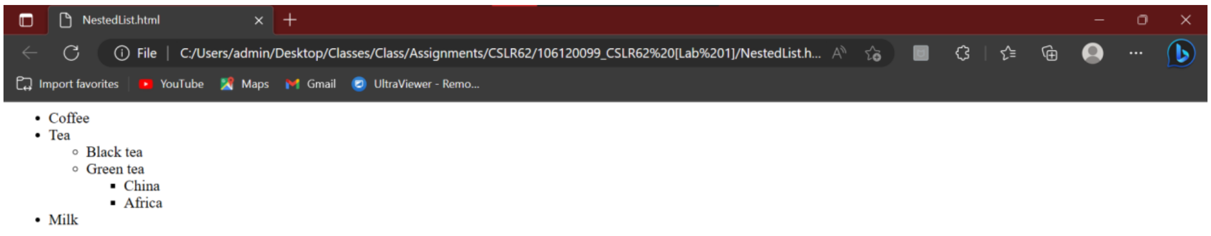

Frames.html
C:/Users/admin/Desktop/Classes/Class/Assignments/CSLR62/106120099_CSLR62%20[Lab%201]/Frames.html

MENU

- Rose
- Jasmine
- Lotus

Rose is a beautiful flower which denotes the symbol of love.

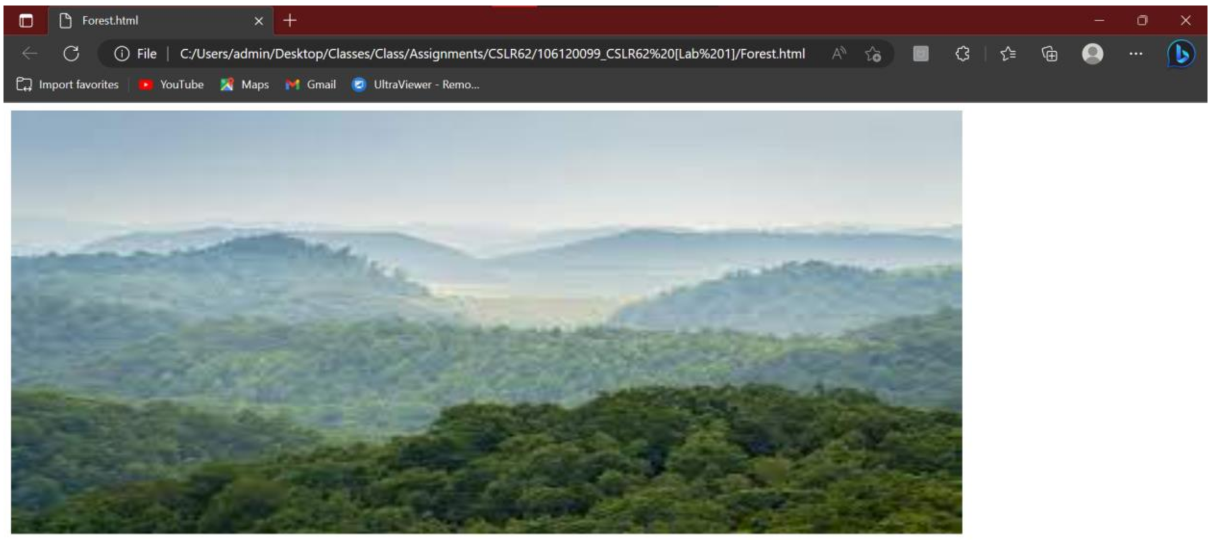
4. Designing a webpage using the concept of nested lists:


NestedList.html

C:/Users/admin/Desktop/Classes/Class/Assignments/CSLR62/106120099_CSLR62%20[Lab%201]/NestedList.h...

- Coffee
- Tea
  - Black tea
  - Green tea
    - China
    - Africa
- Milk

5. Designing a webpage to embed an image map:


Forest.html

C:/Users/admin/Desktop/Classes/Class/Assignments/CSLR62/106120099_CSLR62%20[Lab%201]/Forest.html

Result: Static webpages were successfully created using HTML.

Creation of Webpages using CSS

Ex. No: 2

Date:

Problem Statement: To design webpages using CSS .

Algorithm:

1. Write a CSS rule that makes all text 1.5 times larger than the base font of the system and colours the text red.
   - Create an HTML document.
   - Set the language of the document to "en" (English).
   - Add a <head> element inside the HTML document.
   - Inside the <head> element:
   - Add a <meta> element with the attribute charset="UTF-8" to specify the character encoding.
   - Add a <meta> element with the attribute http-equiv="X-UA-Compatible" and set its content to "IE=edge" for compatibility.
   - Add a <meta> element with the attributes name="viewport" and content="width=device-width, initial-scale=1.0" for viewport control.
   - Set the title of the document to "Document" using a <title> element.
   - Add a <style> element to define CSS styles.
   - Set the font size to 1.5em and color to red for <p> elements.
   - Add a <body> element inside the HTML document.
   - Inside the <body> element, add a <p> element.
   - Set the text content of the <p> element.
   - Close the <p> element.
   - Close the <body> element.
   - Close the HTML document.
2. Write a CSS rule that removes the underlines from all links inside list items (li) and shifts all list items left by 3 ems.
   - - Create an HTML document.
   - - Add a \`<head>\` element inside the HTML document.
   - - Inside the \`<head>\` element:
   - - Set the title of the document to "lists" using a \`<title>\` element.
   - - Add a \`<style>\` element to define CSS styles.
   - - Set the text decoration for \`<a>\` elements with the pseudo-class \`:link\` to none.

- - Add a `<body>` element inside the HTML document.
- - Inside the `<body>` element:
-   - Add an `<h1>` element with the attribute `align="left"` and set its text content to "LISTS".
-   - Add an ordered list `<ol>` element.
-     - Add a list item `<li>` element with an `<a>` element inside.
-       - Set the `href` attribute of the `<a>` element to "#".
-       - Set the text content of the `<a>` element to "html".
-     - Add another list item `<li>` element with an `<a>` element inside.
-       - Set the `href` attribute of the `<a>` element to "#".
-       - Set the text content of the `<a>` element to "css".
-     - Add another list item `<li>` element with an `<a>` element inside.
-       - Set the `href` attribute of the `<a>` element to "#".
-       - Set the text content of the `<a>` element to "js".
- - Close the `<ol>` element.
- - Close the `<body>` element.
- - Close the HTML document.

3. Write a CSS rule that places a background image halfway down the page, tiling it horizontally. The image should remain in place when the user scrolls up or down.
    - Algorithm for the provided HTML code:
    - - Create an HTML document.
    - - Add a `<head>` element inside the HTML document.
    - - Inside the `<head>` element:
    -   - Set the title of the document to "lists" using a `<title>` element.
    -   - Add a `<style>` element to define CSS styles.
    -     - Set the background image of the `<body>` element to '9.jpg' using the `background-image` property.
    -     - Set the background position to 50% horizontally and 0 vertically using the `background-position` property.
    -     - Set the background repeat to repeat horizontally using the `background-repeat` property.
    - - Add a `<body>` element inside the HTML document.
    - - Inside the `<body>` element, add a `<p>` element.
    -   - Set the text content of the `<p>` element.
    - - Close the `<p>` element.
    - - Close the `<body>` element.
    - - Close the HTML document.
4. Write a CSS rule that displays h1elements in blue. In addition, create a rule that displays all links in blue without underlining

them. When the mouse hovers over a link, change the link’s background color to yellow.

- Create an HTML document.
- Set the language of the document to "en" (English).
- Add a <head> element inside the HTML document.
- Inside the <head> element:
- Set the character encoding to UTF-8 using a <meta> element.
- Set the compatibility mode to IE=edge using a <meta> element.
- Set the viewport to control the width and initial scale using a <meta> element.
- Set the title of the document to "Document" using a <title> element.
- Add a <style> element to define CSS styles.
- Set the color of <h1> elements to blue.
- Remove the text decoration for <a> elements with the pseudo-class :link.
- Change the background color to yellow for <a> elements with the pseudo-class :hover.
- Add a <body> element inside the HTML document.
- Inside the <body> element:
- Add an <h1> element with the text content "heyyy u".
- Add another <h1> element with the text content "how are u".
- Add an <a> element with the text content "link 1" and the href attribute set to "#".
- Add another <a> element with the text content "link 2" and the href attribute set to "#".
- Close the <body> element.
- Close the HTML document.

5. Write a CSS rule that gives all h1 elements a padding of 0.8 ems, a dashed border style and a margin of 0.8 ems.

- Create an HTML document.
- Set the language of the document to "en" (English).
- Add a <head> element inside the HTML document.
- Inside the <head> element:
- Set the character encoding to UTF-8 using a <meta> element.
- Set the compatibility mode to IE=edge using a <meta> element.
- Set the viewport to control the width and initial scale using a <meta> element.
- Set the title of the document to "Document" using a <title> element.
- Add a <style> element to define CSS styles.
- Set the padding of <h1> elements to 0.8em.
- Set the margin of <h1> elements to 0.8em.
- Set the border style of <h1> elements to dashed.

- Add a <body> element inside the HTML document.
- Inside the <body> element, add an <h1> element with the text content "heyy you".
- Add another <h1> element inside the <body> element with the text content "how are you".
- Close the <body> element.
- Close the HTML document.

Code:

1. Write a CSS rule that makes all text 1.5 times larger than the base font of the system and colours the text red.

```
<!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>

    p{

      font-size: 1.5em;

      color:red;

    }

  </style>

</head>

<body>

  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolorum aperiam quidem
quibusdam recusandae vitae voluptatem cum quaerat quos at atque adipisci, quas nobis dicta
fugiat, nesciunt consequuntur facere distinctio quis. Lorem, ipsum dolor sit amet consectetur
adipisicing elit. Voluptatibus fugiat beatae tempora at adipisci repellat qui quo quod tempore
reiciendis et enim sunt totam numquam consectetur quia dignissimos, velit omnis!</p>

</body>

</html>
```

2.Write a CSS rule that removes the underlines from all links inside list items (li) and shifts all list items left by 3 ems.

```
<!DOCTYPE html>
<html>
  <head>
    <title>lists</title>
    <style>
      a:link{
        text-decoration: none;

      }
    </style>

  </head>
  <body>
    <h1 align="left">LISTS</h1>
    <ol>
      <li><a href="#">html</a></li>
      <li><a href="#">css</a></li>
      <li><a href="#">js</a></li>



    </ol>

  </body>
</html>
```

3. CSS rule that places a background image halfway down the page, tiling it horizontally. The image should remain in place when the user scrolls up or down.

```
<!DOCTYPE html>
<html>
  <head>
    <title>lists</title>
    <style>
      body {
      background-image: url('9.jpg');
      background-position: 50% 0;
      background-repeat: repeat-x;
    }
    </style>

  </head>
  <body>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Animi, laborum ea consequatur aut velit aspernatur quod, temporibus incidunt nobis debitis officiis, quas aliquid modi obcaecati adipisci vitae quibusdam dolores soluta. Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ipsa, eligendi ex porro esse, dolores sapiente saepe pariatur amet reiciendis, vero consequuntur! Excepturi voluptate placeat odio impedit? Iure inventore quo ad. Lorem, ipsum dolor sit amet consectetur adipisicing elit. Harum tenetur deserunt a omnis eum id, architecto itaque aperiam numquam blanditiis quibusdam iusto accusantium possimus corporis eveniet, ab fuga in. Quas?</p>



  </body>
</html>
```

4. Write a CSS rule that displays h1elements in blue. In addition, create a rule that displays all links in blue without underlining them. When the mouse hovers over a link, change the link’s background color to yellow.

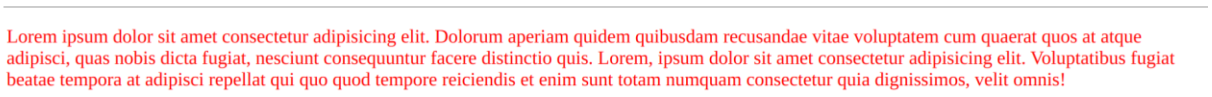
```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    h1{
            color: blue;
        }
        a:link{
            text-decoration: none;
        }
        a:hover{
            background-color: yellow;
        }
    </style>
</head>
<body>
    <h1>heyyy u</h1>
    <h1> how are u</h1>
    <a href="#">link 1</a>
    <a href="#">link 2</a>
</body>
</html>
```

5. Write a CSS rule that gives all h1 elements a padding of 0.8 ems, a dashed border style and a margin of 0.8 ems.

```
<!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>
        h1{
            padding: 0.8em;
            margin: 0.8em;
            border-style: dashed;
    }
  </style>
</head>
<body>
  <h1>heyy you</h1>
  <h1>how are you</h1>
</body>
</html>
```

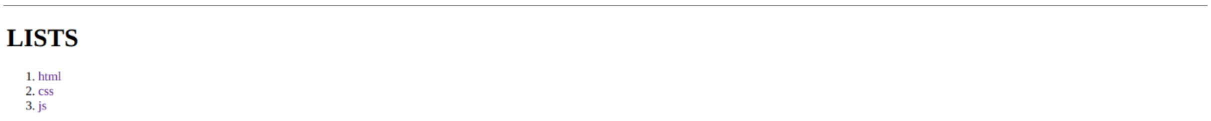
Sample Output:

1.Write a CSS rule that makes all text 1.5 times larger than the base font of the system and colours the text red.



Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolorum aperiam quidem quibusdam recusandae vitae voluptatem cum quaerat quos at atque adipisci, quas nobis dicta fugiat, nesciunt consequuntur facere distinctio quis. Lorem, ipsum dolor sit amet consectetur adipisicing elit. Voluptatibus fugiat beatae tempora at adipisci repellat qui quo quod tempore reiciendis et enim sunt totam numquam consectetur quia dignissimos, velit omnis!

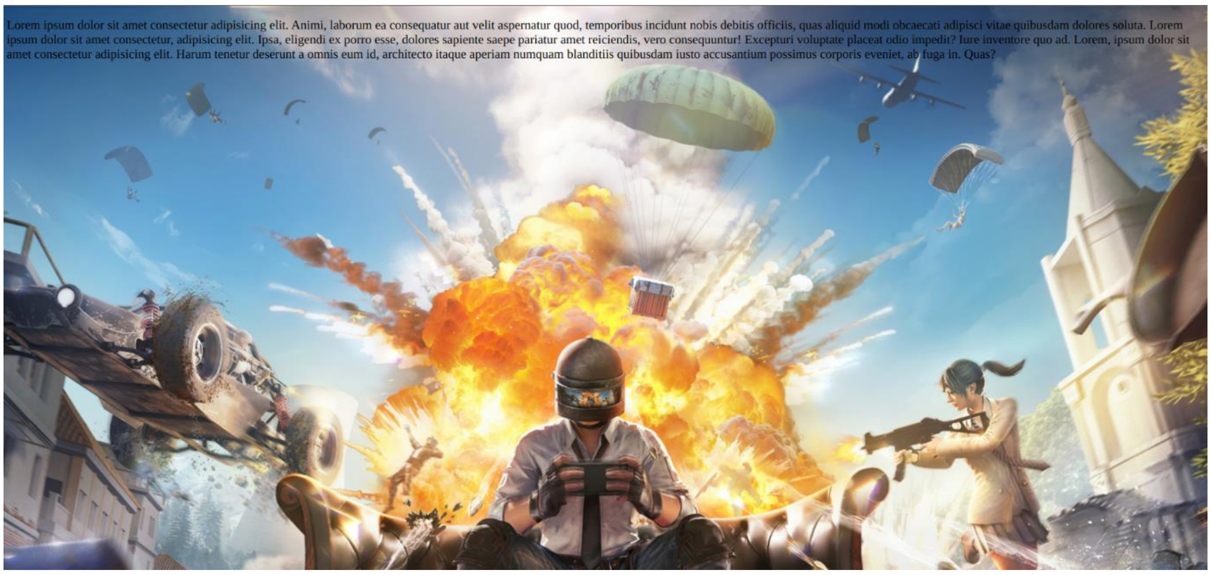
2. Write a CSS rule that removes the underlines from all links inside list items (li) and shifts all list items left by 3 ems.



LISTS

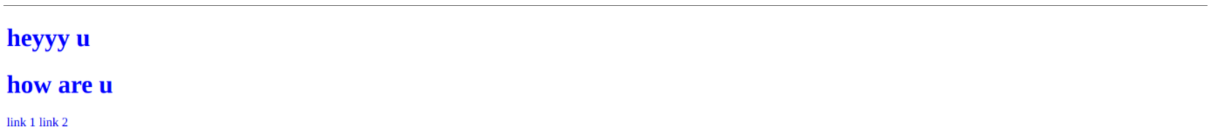
1. html
2. css
3. js

3. CSS rule that places a background image halfway down the page, tiling it horizontally. The image should remain in place when the user scrolls up or down.



Lorem ipsum dolor sit amet consectetur adipisicing elit. Animi, laborum ea consequatur aut velit aspernatur quod, temporibus incidunt nobis debitis officiis, quas aliquid modi obcaecati adipisci vitae quibusdam dolores soluta. Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ipsa, eligendi ex porro esse, dolores sapiente saepe pariatur amet reiciendis, vero consequuntur! Excepturi voluptate placeat odio impedit? Iure inventore quo ad. Lorem, ipsum dolor sit amet consectetur adipisicing elit. Harum tenetur deserunt a omnis eum id, architecto itaque aperiam numquam blanditiis quibusdam iusto accusantium possimus corporis eveniet, ab fuga in. Quas?

4. Write a CSS rule that displays h1elements in blue. In addition, create a rule that displays all links in blue without underlining them. When the mouse hovers over a link, change the link’s background color to yellow.

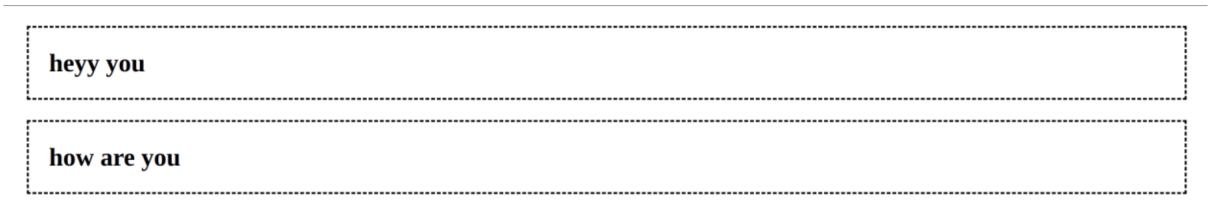


heyyy u

how are u

link 1 link 2

5. Write a CSS rule that gives all h1 elements a padding of 0.8 ems, a dashed border style and a margin of 0.8 ems.



heyy you

how are you

Programs using JavaScript.

Ex. No: 3

Date:

Problem Statement: To implement the given programs using javascript .

Algorithm:

1. Write a JavaScript program to perform the following String Manipulation Operations with and without using the methods of String Object.

   I.To search for the first occurrence of a substring in a given string

   II.To return the character at a specified index

   III.To concatenate two string objects

- - Create an HTML document.
- - Add a `<head>` element inside the HTML document.
- - Inside the `<head>` element:
-   - Set the character encoding to UTF-8 using a `<meta>` element.
-   - Set the title of the document to "4A" using a `<title>` element.
- - Add a `<body>` element inside the HTML document.
- - Inside the `<body>` element, add a `<script>` element.
-   - Declare a constant variable `string` and set it to "Ashrith".
-   - Declare a constant variable `substring` and set it to " from AP".
-   - Define a function `searchSubstring` that takes in `string` and `substring` as parameters.
-     - Find the index of the `substring` in the `string` using the `indexOf` method of the String object.
-     - Write the result to the document using the `document.write` method.
-     - Iterate through the `string` using a for loop.
-       - Check if the current slice of the `string` matches the `substring`.
-       - If a match is found, write the index to the document and break the loop.
-   - Call the `searchSubstring` function with the `string` and "shr" as arguments.

- - Define a function `getCharacterAtIndex` that takes in `string` and `index` as parameters.
- - Get the character at the specified `index` in the `string` using the `charAt` method of the String object.
- - Write the result to the document using the `document.write` method.
- - Write the result again without using the String object method.
- - Call the `getCharacterAtIndex` function with the `string` and 4 as arguments.
- - Define a function `concatenateStrings` that takes in `string1` and `string2` as parameters.
- - Concatenate `string1` and `string2` using the `concat` method of the String object.
- - Write the result to the document using the `document.write` method.
- - Write the result again without using the String object method.
- - Call the `concatenateStrings` function with the `string` and `substring` as arguments.
- - Close the `<script>` element.
- - Close the `<body>` element.
- - Close the HTML document.

2. Write a JavaScript that inputs text from an XHTML form and outputs it in Upper case and Lower case letters.
    - - Create an HTML document.
    - - Add a `<head>` element inside the HTML document.
    - - Inside the `<head>` element:
    - - Set the character encoding to UTF-8 using a `<meta>` element.
    - - Set the title of the document to "Document" using a `<title>` element.
    - - Add a `<body>` element inside the HTML document.
    - - Inside the `<body>` element, add a `<form>` element.
    - - Add an `<input>` element of type "text" with a placeholder "Enter text".
    - - Add a `<button>` element with the text content "click me" and an `onclick` attribute that calls the `getVal()` function.
    - - Add a `<script>` element inside the `<body>` element.
    - - Define a function `getVal`.
    - - Get the value of the `<input>` element using the `querySelector` method and the CSS selector `'input'`.
    - - Convert the value to uppercase using the `toUpperCase` method and store it in the variable `f`.
    - - Convert the value to lowercase using the `toLowerCase` method and store it in the variable `z`.
    - - Write the uppercase value to the document using the `document.write` method.
    - - Write a line break to the document using the `document.write` method.

- - Write the lowercase value to the document using the `document.write` method.
- - Close the `<script>` element.
- - Close the `<body>` element.
- - Close the HTML document.

3. Write a JavaScript that reads a Five-Letter word from the user and produces all possible Three-Letter words that can be derived from the letters of the Five-Letter word. For example, the three-letter words produced from the word “bathe” include the commonly used words : ate bat bet tab hat the tea
   - Add a <head> element inside the HTML document.
   - Inside the <head> element:
   - Set the character encoding to UTF-8 using a <meta> element.
   - Set the title of the document to "Document" using a <title> element.
   - Add a <body> element inside the HTML document.
   - Inside the <body> element, add a <script> element.
   - Use the prompt function to prompt the user to enter a string and store it in the variable s.
   - Get the length of the string using the length property of the string and store it in the variable n.
   - Check if the length of the string is less than 3.
   - If it is less than 3, write "length is less than 3" to the document using the document.write method.
   - If it is not less than 3, proceed with the nested loops.
   - Use a for loop to iterate from i = 0 to n - 2.
   - Use another for loop to iterate from j = i + 1 to n - 1.
   - Use a third for loop to iterate from k = j + 1 to n.
   - Inside the nested loops, create a new string ff by concatenating s[i], s[j], and s[k].
   - Write the value of ff to the document using the document.write method.
   - Write a line break to the document using the document.write method.
   - Close the <script> element.
   - Close the <body> element.
   - Close the HTML document.
4. Write a JavaScript to display the current time and print the message according to the time. For example, Good Morning in Morning, Good After Noon in Noon, Good Evening in evening and Good Night in Night.

- - Create an HTML document.
- - Add a `<head>` element inside the HTML document.
- - Inside the `<head>` element:
-   - Set the character encoding to UTF-8 using a `<meta>` element.
-   - Set the title of the document to "Document" using a `<title>` element.
- - Add a `<body>` element inside the HTML document.
- - Inside the `<body>` element, add a `<script>` element.
-   - Create a new `Date` object and store it in the variable `obj`.
-   - Get the current hour from the `Date` object using the `getHours` method and store it in the variable `h`.
-   - Get the current minute from the `Date` object using the `getMinutes` method and store it in the variable `m`.
-   - Get the current second from the `Date` object using the `getSeconds` method and store it in the variable `s`.
-   - Write the current time in the format "current time is hh:mm:ss" to the document using the `document.write` method.
-   - Write a line break to the document using the `document.write` method.
-   - Initialize the variable `greetings` as an empty string.
-   - Check the value of `h` to determine the appropriate greeting.
-     - If `h` is less than or equal to 12, set `greetings` to "good_mrng".
-     - If `h` is greater than 12 and less than or equal to 17, set `greetings` to "good_afternoon".
-     - If `h` is greater than 17 and less than or equal to 21, set `greetings` to "good_eveng".
-     - If none of the above conditions are met, set `greetings` to "good_night".
-   - Write the value of `greetings` to the document using the `document.write` method.
- - Close the `<script>` element.
- - Close the `<body>` element.
- - Close the HTML document.

5. Write a JavaScript program to perform Merge Sort for a given set of N numbers dynamically prompted from the user. (Use user defined function to perform Merge Sort)

- - Create an HTML document.
- - Add a `<head>` element inside the HTML document.
- - Inside the `<head>` element:
-   - Set the title of the document to "Merge Sort" using a `<title>` element.
- - Add a `<script>` element inside the `<head>` element.

- - Define a function named `merge_Arrays` that takes two parameters: `left_sub_array` and `right_sub_array`.
- - Create an empty array called `array`.
- - Use a `while` loop to compare elements from `left_sub_array` and `right_sub_array`.
- - If the first element of `left_sub_array` is smaller than the first element of `right_sub_array`, remove it from `left_sub_array` using `shift()` and push it to `array`.
- - Otherwise, remove the first element from `right_sub_array` using `shift()` and push it to `array`.
- - Return the merged array by concatenating `array`, `left_sub_array`, and `right_sub_array` using the spread syntax (`...`).
- - Define a function named `merge_sort` that takes one parameter: `unsorted_Array`.
- - Check if the length of `unsorted_Array` is less than 2.
- - If true, return `unsorted_Array` as it is.
- - Calculate the middle index of `unsorted_Array` and store it in the variable `middle_index`.
- - Create a `left_sub_array` by splicing `unsorted_Array` from index 0 to `middle_index`.
- - Recursively call `merge_sort` on `left_sub_array` and store the result in `left_sorted_array`.
- - Recursively call `merge_sort` on the remaining part of `unsorted_Array` and store the result in `right_sorted_array`.
- - Return the result of calling `merge_Arrays` with `left_sorted_array` and `right_sorted_array`.
- - Create an empty array `a` and `inputArray`.
- - Prompt the user to enter the size of the array and store it in the variable `size`.
- - Use a `for` loop to prompt the user to enter elements and store them in `a`.
- - Create an example `unsorted_Array` with fixed values [39, 28, 44, 4, 10, 83, 11].
- - Write the sorted array obtained by calling `merge_sort` on `a` to the document using `document.write`.
- - Close the `<script>` element.
- - Add a `<body>` element inside the HTML document.
- - Close the `<body>` element.
- - Close the HTML document.

Code:

1. Write a JavaScript program to perform the following String Manipulation Operations with and without using the methods of String Object.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8" />
  <title>4A</title>
 </head>
 <body>
  <script>

   const string = "Ashrith";
   const substring = " from AP";

   const searchSubstring = (string, substring) => {

    const index = string.indexOf(substring);
    document.write(`Using String object method: ${index}`);
    document.write("<br>")



    for (let i = 0; i < string.length - substring.length + 1; i++) {
     if (string.slice(i, i + substring.length) === substring) {
      document.write(`Without using String object method: ${i}`);
      document.write("<br>")
      break;
     }
    }
   };

   searchSubstring(string, "shr");
```

```
    const getCharacterAtIndex = (string, index) => {


      const char = string.charAt(index);
      document.write(`Using String object method: ${char}`);
      document.write("<br>")



      document.write(`Without using String object method: ${string[index]}`);
      document.write("<br>")
    };

    getCharacterAtIndex(string, 4);

    // III. To concatenate two string objects
    const concatenateStrings = (string1, string2) => {
      // Using String object method
      const result = string1.concat(string2);
      document.write(`Using String object method: ${result}`);
      document.write("<br>")

      // Without using String object method
      document.write(`Without using String object method: ${string1}${string2}`);
      document.write("<br>")
    };

    concatenateStrings(string, substring);
  </script>
 </body>
</html>
```

2.Write a JavaScript that inputs text from an XHTML form and outputs it in Upper case and Lower case letters.

```
<!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>
  <form>
    <input type="text" placeholder="Enter text" >
    <button onclick="getVal()">click me</button>
  </form>
  <script>
    function getVal() {
      const val = document.querySelector('input').value;
      var f=val.toUpperCase();
      var z=val.toLowerCase();
      document.write(f);
      document.write("<br>");
      document.write(z);
    }

  </script>

</body>
```

```
</html>
```

3.Write a JavaScript that reads a Five-Letter word from the user and produces all possible Three-Letter words that can be derived from the letters of the Five-Letter word

```
<!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>
  <script>
    var s=prompt("enter string");
    var n=s.length;
    if(n<3) document.write("length is less than 3");
    else{
      for(var i=0;i<n-2;i++){
        for(var j=i+1;j<n-1;j++){
          for(var k=j+1;k<n;k++){
            var ff=s[i]+s[j]+s[k];
            document.write(ff);
            document.write("<br>");


          }
        }
      }
    }
  </script>
```

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

4.Write a JavaScript to display the current time and print the message according to the time. For example, Good Morning in Morning, Good After Noon in Noon, Good Evening in evening and Good Night in Night.

```
<!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>
    <script>
        const obj = new Date()
        const h = obj.getHours()
        const m=obj.getMinutes();
        const s=obj.getSeconds();
        document.write("current time is "+h+":"+m+":"+s);
        document.write("<br>")
        let greetings = ""
        if (h <= 12) {
            greetings = "good_mrng"

        }
        else if(h>12 && h<=17){
            greetings="good _afternoon"
          }
          else if(h>17 && h<=21) {
            greetings = "good_eveng"


          }
          else{
            greetings="good_night"
          }
          document.write(greetings);
        </script>
      </body>

      </html>
```

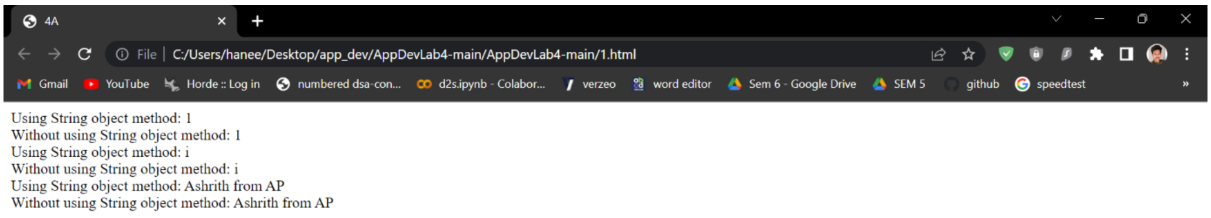
5.Write a JavaScript program to perform Merge Sort for a given set of N numbers dynamically prompted from the user. (Use user defined function to perform Merge Sort)

```
<!DOCTYPE html>
<html>
<title>Merge Sort</title>

<head>
  <script>
    function merge_Arrays(left_sub_array, right_sub_array) {
      let array = []
      while (left_sub_array.length && right_sub_array.length) {
        if (left_sub_array[0] < right_sub_array[0]) {
          array.push(left_sub_array.shift())
        } else {
          array.push(right_sub_array.shift())
```

```
            }
         }
         return [...array, ...left_sub_array, ...right_sub_array]
      }
      function merge_sort(unsorted_Array) {
         const middle_index = unsorted_Array.length / 2
         if (unsorted_Array.length < 2) {
            return unsorted_Array
         }
         const left_sub_array = unsorted_Array.splice(0, middle_index)
         return merge_Arrays(merge_sort(left_sub_array), merge_sort(unsorted_Array))
      }
      const a = [];
      var inputArray = [];
      var size =  prompt("enter array size");

      for (var i = 0; i < size; i++) {
         a[i] = prompt('Enter Element ' + (i + 1));
      }



      unsorted_Array = [39, 28, 44, 4, 10, 83, 11];
      document.write("The sorted array will be: ", merge_sort(a));
   </script>
</head>

<body>
</body>

</html>
```

Sample Output:

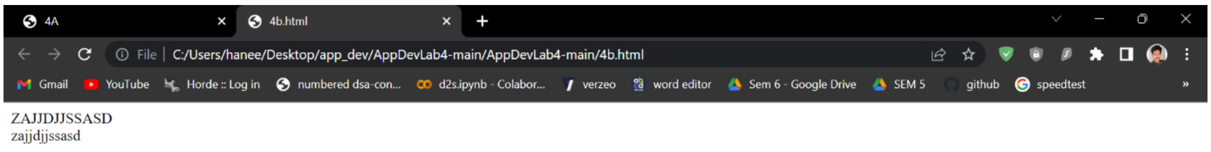
1. Write a JavaScript program to perform the following String Manipulation Operations with and without using the methods of String Object.



4A
File | C:/Users/hanee/Desktop/app_dev/AppDevLab4-main/AppDevLab4-main/1.html
Gmail YouTube Horde :: Log in numbered dsa-con... d2s.ipynb - Colabor... verzeo word editor Sem 6 - Google Drive SEM 5 github speedtest

```
Using String object method: 1
Without using String object method: 1
Using String object method: i
Without using String object method: i
Using String object method: Ashrith from AP
Without using String object method: Ashrith from AP
```

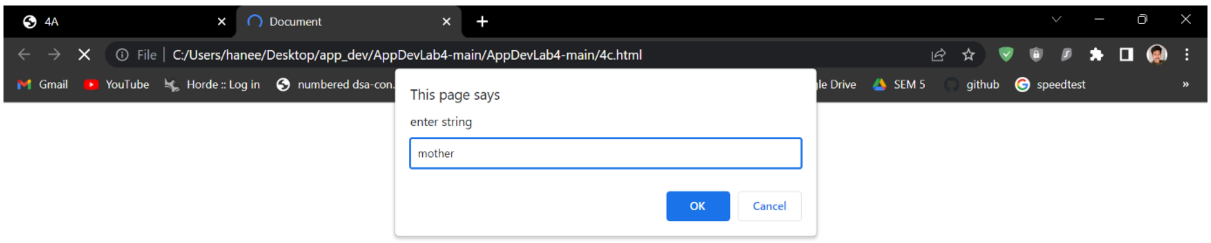
2. Write a JavaScript that inputs text from an XHTML form and outputs it in Upper case and Lower case letters.



4A
4b.html
File | C:/Users/hanee/Desktop/app_dev/AppDevLab4-main/AppDevLab4-main/4b.html
Gmail YouTube Horde :: Log in numbered dsa-con... d2s.ipynb - Colabor... verzeo word editor Sem 6 - Google Drive SEM 5 github speedtest

```
ZAJJDJJSSASD
zajjdjjssasd
```

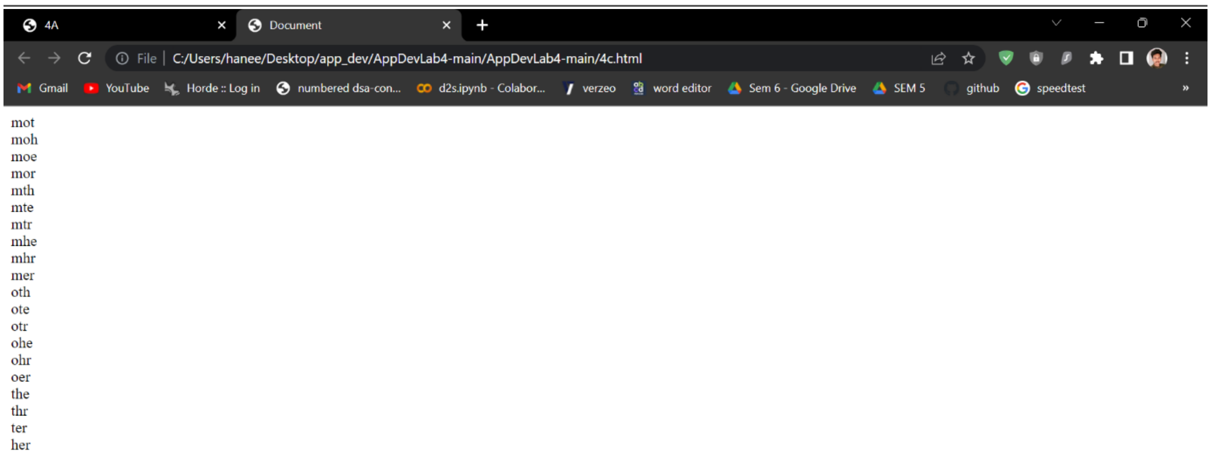
3. Write a JavaScript that reads a Five-Letter word from the user and produces all possible Three-Letter words that can be derived from the letters of the Five-Letter word


This page says

enter string

mother

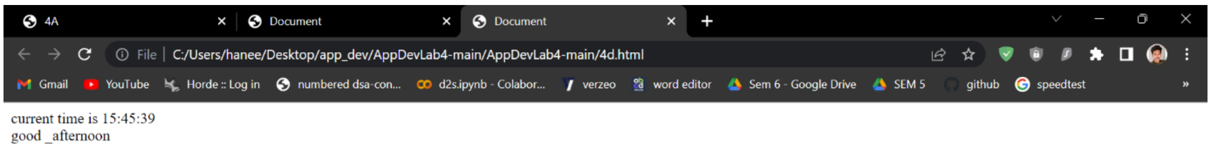
OK Cancel



```
mot
moh
moe
mor
mth
mte
mtr
mhe
mhr
mer
oth
ote
otr
ohe
ohr
oer
the
thr
ter
her
```

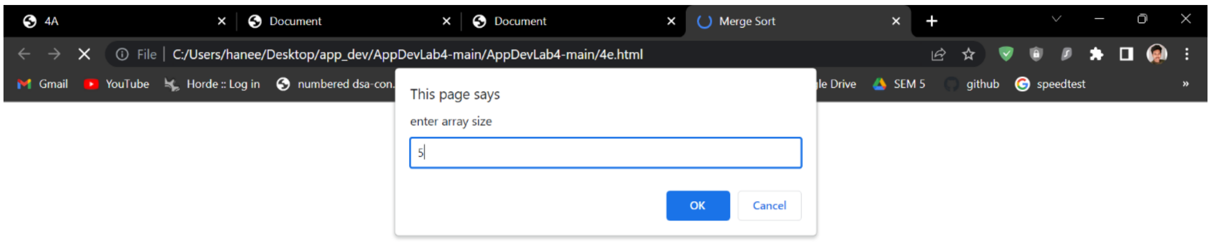
4.Write a JavaScript to display the current time and print the message according to the time. For example, Good Morning in

Morning, Good After Noon in Noon, Good Evening in evening and Good Night in Night.


File | C:/Users/hanee/Desktop/app_dev/AppDevLab4-main/AppDevLab4-main/4d.html

```
current time is 15:45:39
good _afternoon
```

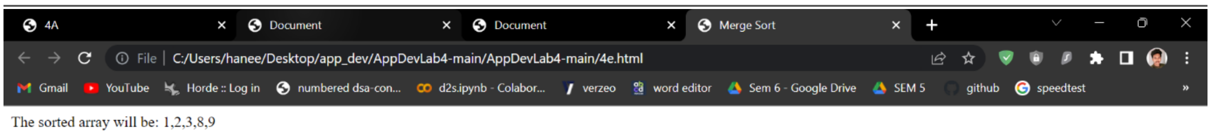
5. Write a JavaScript program to perform Merge Sort for a given set of N numbers dynamically prompted from the user. (Use user defined function to perform Merge Sort)



This page says

enter array size

5



```
The sorted array will be: 1,2,3,8,9
```

Validation of XML Documents using DTD, Schema and Conversion of XML to XHTML using XSLT.

Ex. No: 4

Date:

Problem Statement: Convert the given files into DTD,XML formats

Algorithm:

1. Given the following DTD, Generate the corresponding XML document. Also write the XML schema for that document.
   - The XML document represents a bibliography.
   - There are two books listed in the bibliography.
   - Each book has a year attribute indicating the year of publication.
   - The first book has the following details:
   - Title: "Book Title"
   - Author: First name - "First Name," Last name - "Last Name"
   - Publisher: "Publisher Name"
   - Price: "Price"
   - The second book has the following details:
   - Title: "Book Title"
   - Editor: First name - "First Name," Last name - "Last Name," Affiliation - "Affiliation"
   - Publisher: "Publisher Name"
   - Price: "Price"
   - This XML code provides a structured representation of bibliographic information for two books, including their titles, authors/editors, publishers, and prices.
2. Write the DTD for the following XML document

- The root element of the "pets" XML document is "pets", which can contain zero or more "pet" elements.
- Each "pet" element consists of the following child elements: "name", "age", "type", and "color".
- The "name" element contains character data (PCDATA).
- The "age" element also contains character data (PCDATA).
- The "type" element contains character data (PCDATA).
- The "color" element contains character data (PCDATA).

3. Create an XML document for the following letter as per the element names – [letter, date, addressee, name, address_one, address_two, greeting, paragraph, italics, list, item, closing]. Also validate the XML file generated by writing the appropriate DTD.

- The XML document begins with the XML declaration specifying the version and encoding.
- The document type declaration (DTD) defines the structure and elements of the "letter" XML document.
- The root element of the XML document is "letter".
- The "letter" element contains the following child elements: "date", "addressee", "greeting", "paragraph", "list", and "closing".
- The "date" element contains the date of the letter.
- The "addressee" element contains the recipient's information, including "name", "address_one", and "address_two" elements.
- The "greeting" element contains the salutation of the letter.
- The "paragraph" element represents a paragraph of the letter and can contain both character data (PCDATA) and "italics" elements.
- The "italics" element represents text in italics within a paragraph.
- The "list" element represents a list and contains one or more "item" elements.
- The "item" element represents an item in the list and contains character data (PCDATA).
- The "closing" element contains the closing of the letter.

4. Write the DTD and XML Schema for the following XML document

- The root element of the "bank" XML document is "bank".
- The "bank" element can contain one or more "account", "customer", and "depositor" elements.
- The "account" element is empty (contains no child elements) and represents a bank account.
- The "account" element has the following attributes:
- "account-number" (CDATA type, required)
- "branch-name" (CDATA type, required)
- "balance" (CDATA type, required)
- The "customer" element is empty and represents a bank customer.
- The "customer" element has the following attributes:
- "customer-name" (CDATA type, required)
- "customer-street" (CDATA type, required)
- "customer-city" (CDATA type, required)

- The "depositor" element is empty and represents a relationship between a bank account and a customer.
- The "depositor" element has the following attributes:
- "account-number" (CDATA type, required)
- "customer-name" (CDATA type, required)

5.Convert the XML document given in Ex.b into an XHTML document for tabular presentation using XLST

- The template "match="/" indicates that the transformation should apply to the root node of the XML document.
- The transformed output will be an HTML document.
- The HTML document includes a title element with the text "Pets".
- The body of the HTML document contains a table element.
- The table has a border attribute set to "1".
- The first table row (tr) is colored with a background color of "#9acd32".
- The first row contains table header cells (th) with the labels "Name", "Age", "Type", and "Color".
- The xsl:for-each loop selects all "pet" elements within the "pets" element of the XML document.
- For each "pet" element, a new table row (tr) is created.
- Inside each row, there are four table data cells (td) that display the values of the "name", "age", "type", and "color" elements from the XML document.

Code:

1. Given the following DTD, Generate the corresponding XML document. Also write the XML schema for that document.

```
!ELEMENT bib (book*)>

<!ELEMENT book (title, (author+ | editor+ ), publisher, price)>

<!ATTLIST book year CDATA #REQUIRED >

<!ELEMENT author (last, first)>

<!ELEMENT editor (last, first, affiliation)>

<!ELEMENT title (#PCDATA)> <!ELEMENT last (#PCDATA)>

<!ELEMENT first (#PCDATA)> <!ELEMENT affiliation (#PCDATA)>

<!ELEMENT publisher (#PCDATA)> <!ELEMENT price (#PCDATA)>
```

2. Write the DTD for the following XML document

```
<pets>
<pet>
<name>Tilly</name>
<age>14</age>
<type>cat</type>
<color>silver</color>
</pet>
<pet>
<name>Amanda</name>
<age>10</age>
<type>dog</type>
<color>brown</color>
</pet>
<pet>
<name>Jack</name>
<age>3</age>
<type>cat</type>
<color>black</color>
</pet>
<pet>
<name>Blake</name>
<age>12</age>
<type>dog</type>
<color>blue</color>
</pet>
<pet>
<name>Loosey</name>
<age>1</age>
<type>cat</type>
```

```
<color>brown</color>
</pet>
<pet>
<name>Stop</name>
<age>5</age>
<type>pig</type>
<color>brown</color>
</pet>
</pets>
```

3.Create an XML document for the following letter as per the element names – [letter, date, addressee, name, address_one, address_two, greeting, paragraph, italics, list, item, closing]. Also validate the XML file generated by writing the appropriate DTD

December 11, 2002

Melvile Dewey

Columbia University

New York, NY

Dear Melvile,

I have been reading your ideas concering the nature of
librarianship, and I find them very intriguing. I would love the
opportunity to discuss with you the role of the card catalog in today's
libraries considering the advent to World Wide Web. Specifically, how
are things like Google and Amazon.com changing our patrons' expectations
of library services? Mr. Cutter and I will be discussing these ideas at
the next Annual Meeting, and we are available at the follow dates/times:

* Monday, 2-4

* Tuesday, 3-5

* Thursday, 1-3

We hope you can join us.

4. Write the DTD and XML Schema for the following XML document.

```
<bank>
<account account-number=“A-101” branch-name=“Downtown”
 balance=“500”>
</account>
<account account-number=“A-102” branch-name=“Perryridge”
 balance=“400”>
</account>
<account account-number=“A-201” branch-name=“Brighton”
 balance=“900”>
</account>
<customer customer-name=“Johnson” customer-street=“Alma”
 customer-city=“Palo Alto”>
</customer>
<customer customer-name=“Hayes” customer-street=“Main”
 customer-city=“Harrison”>
</customer>
<depositor account-number=“A-101” customer-name=“Johnson”>
</depositor>
<depositor account-number=“A-201” customer-name=“Johnson”>
</depositor>
<depositor account-number=“A-102” customer-name=“Hayes”>
</depositor>
</bank>
```

5. e. Convert the XML document given in Ex.b into an XHTML document for tabular presentation using XLST

```
<pets>

<pet>

<name>Tilly</name>

<age>14</age>

<type>cat</type>

<color>silver</color>

</pet>

<pet>

<name>Amanda</name>

<age>10</age>

<type>dog</type>

<color>brown</color>

</pet>

<pet>

<name>Jack</name>

<age>3</age>

<type>cat</type>

<color>black</color>

</pet>

<pet>

<name>Blake</name>

<age>12</age>

<type>dog</type>

<color>blue</color>

</pet>

<pet>
```

```
<name>Loosey</name>

<age>1</age>

<type>cat</type>

<color>brown</color>

</pet>

<pet>

<name>Stop</name>

<age>5</age>

<type>pig</type>

<color>brown</color>

</pet>

</pets>
```

Sample Output:

1. Given the following DTD, Generate the corresponding XML document. Also write the XML schema for that document.

Schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="bib">

    <xs:complexType>

      <xs:sequence>

        <xs:element name="book" minOccurs="0" maxOccurs="unbounded">

          <xs:complexType>

            <xs:sequence>

              <xs:element name="title" type="xs:string"/>

              <xs:choice>
```

```
<xs:sequence maxOccurs="unbounded">
  <xs:element name="author">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="last" type="xs:string"/>
        <xs:element name="first" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="editor">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="last" type="xs:string"/>
        <xs:element name="first" type="xs:string"/>
        <xs:element name="affiliation" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
<xs:element name="author" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="last" type="xs:string"/>
      <xs:element name="first" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="editor" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
```

```
                    <xs:sequence>
                      <xs:element name="last" type="xs:string"/>
                      <xs:element name="first" type="xs:string"/>
                      <xs:element name="affiliation" type="xs:string"/>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:choice>
              <xs:element name="publisher" type="xs:string"/>
              <xs:element name="price" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="year" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bib>
  <book year="2023">
    <title>Book Title</title>
    <author>
      <last>Last Name</last>
      <first>First Name</first>
    </author>
    <publisher>Publisher Name</publisher>
```

```
        <price>Price</price>
    </book>
    <book year="2023">
        <title>Book Title</title>
        <editor>
            <last>Last Name</last>
            <first>First Name</first>
            <affiliation>Affiliation</affiliation>
        </editor>
        <publisher>Publisher Name</publisher>
        <price>Price</price>
    </book>
</bib>
```

2. Write the DTD for the following XML document

```
<!DOCTYPE pets [
<!ELEMENT pets (pet*)>
<!ELEMENT pet (name, age, type, color)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT color (#PCDATA)>
]>
```

3. Create an XML document for the following letter as per the element names – [letter, date, addressee, name, address_one, address_two, greeting, paragraph, italics, list, item, closing]. Also validate the XML file generated by writing the appropriate DTD.

.DTD

```
<!DOCTYPE letter [
<!ELEMENT letter (date, addressee, greeting, paragraph+, list, closing)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT addressee (name, address_one, address_two)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address_one (#PCDATA)>
<!ELEMENT address_two (#PCDATA)>
<!ELEMENT greeting (#PCDATA)>
<!ELEMENT paragraph (#PCDATA | italics)*>
<!ELEMENT italics (#PCDATA)>
<!ELEMENT list (item+)>
<!ELEMENT item (#PCDATA)>
<!ELEMENT closing (#PCDATA)>
]>
```

.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE letter [
<!ELEMENT letter (date, addressee, greeting, paragraph+, list, closing)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT addressee (name, address_one, address_two)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address_one (#PCDATA)>
<!ELEMENT address_two (#PCDATA)>
<!ELEMENT greeting (#PCDATA)>
<!ELEMENT paragraph (#PCDATA | italics)*>

<!ELEMENT italics (#PCDATA)>

<!ELEMENT list (item+)>

<!ELEMENT item (#PCDATA)>

<!ELEMENT closing (#PCDATA)>

]>

<letter>

  <date>December 11, 2002</date>

  <addressee>

    <name>Melvile Dewey</name>

    <address_one>Columbia University</address_one>

    <address_two>New York, NY</address_two>

  </addressee>

  <greeting>Dear Melvile,</greeting>

  <paragraph>I have been reading your ideas concering the nature of librarianship, and I find
them very intriguing. I would love the opportunity to discuss with you the role of the card
catalog in today's libraries considering the advent to World Wide Web. Specifically, how are
things like Google and Amazon.com changing our patrons' expectations of library services?
Mr. Cutter and I will be discussing these ideas at the next Annual Meeting, and we are
available at the follow dates/times:</paragraph>

  <list>

    <item>Monday, 2-4</item>

    <item>Tuesday, 3-5</item>

    <item>Thursday, 1-3</item>

  </list>

  <closing>Sincerely, James k Robert</closing>

</letter>
```

4. Write the DTD and XML Schema for the following XML document

.DTD

```
<!DOCTYPE bank [
<!ELEMENT bank (account+, customer+, depositor+)>
<!ELEMENT account EMPTY>
<!ATTLIST account
        account-number CDATA #REQUIRED
        branch-name CDATA #REQUIRED
        balance CDATA #REQUIRED>
<!ELEMENT customer EMPTY>
<!ATTLIST customer
        customer-name CDATA #REQUIRED
        customer-street CDATA #REQUIRED
        customer-city CDATA #REQUIRED>
<!ELEMENT depositor EMPTY>
<!ATTLIST depositor
        account-number CDATA #REQUIRED
        customer-name CDATA #REQUIRED>
]>
```

.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="bank">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="account" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="account-number" type="xs:string" use="required"/>
      <xs:attribute name="branch-name" type="xs:string" use="required"/>
      <xs:attribute name="balance" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="customer" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="customer-name" type="xs:string" use="required"/>
      <xs:attribute name="customer-street" type="xs:string" use="required"/>
      <xs:attribute name="customer-city" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="depositor" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="account-number" type="xs:string" use="required"/>
      <xs:attribute name="customer-name" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

5. Convert the XML document given in Ex.b into an XHTML document for tabular presentation Using XSLT.

.xhtml

```
<html>
 <head>
 <title>Pets</title>
</head>
<body>
 <table border="1">
  <tr bgcolor="#9acd32">
   <th>Name</th>
   <th>Age</th>
   <th>Type</th>
   <th>Color</th>
  </tr>
  <tr>
   <td>Tilly</td>
   <td>14</td>
   <td>cat</td>
   <td>silver</td>
  </tr>
  <tr>
   <td>Amanda</td>
   <td>10</td>
   <td>dog</td>
   <td>brown</td>
  </tr>
  <tr>
   <td>Jack</td>
   <td>3</td>
   <td>cat</td>
   <td>black</td>
  </tr>
  <tr>
```

```
        <td>Blake</td>
        <td>12</td>
        <td>dog</td>
        <td>blue</td>
      </tr>
      <tr>
        <td>Loosey</td>
        <td>1</td>
        <td>cat</td>
        <td>brown</td>
      </tr>
      <tr>
        <td>Stop</td>
        <td>5</td>
        <td>pig</td>
        <td>brown</td>
      </tr>
    </table>
  </body>
</html>
```

.XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Pets</title>
    </head>
    <body>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Name</th>
          <th>Age</th>
          <th>Type</th>
          <th>Color</th>
        </tr>
        <xsl:for-each select="pets/pet">
          <tr>
            <td><xsl:value-of select="name"/></td>
            <td><xsl:value-of select="age"/></td>
            <td><xsl:value-of select="type"/></td>
            <td><xsl:value-of select="color"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Creation of Three-tier applications using Java Servlets and JSP.

Ex. No: 5

Date:

Problem Statement: Write a Generic Servlet program using Java Servlet to print the message -“HELLO WORLD” in the Servlet window.

Algorithm:

1. Write a Generic Servlet program using Java Servlet to print the message -“HELLO WORLD” in the Servlet window.

- Create a new Java class called HelloWorldServlet that extends the HttpServlet class.
- Override the doGet() method to handle HTTP GET requests.
- Set the response content type to "text/html" using the setContentType() method of the response object.
- Get a PrintWriter object from the response object using the getWriter() method.
- Write the HTML code to the PrintWriter object using methods such as println().
- Close the PrintWriter object.
- The Servlet container automatically sends the response back to the client.

2. Design an HTML Form that consists of the following fields and a SUBMIT button.

Emp Name Emp ID DOB

Department Salary Email ID

Write a Generic Servlet Program to retrieve all the Form Parameter Names and their corresponding values and display them in the Servlet window.

- Create a Servlet that extends HttpServlet and override the doPost() method.
- Set the content type of the response to "text/html".
- Create a PrintWriter object to write the response.
- Retrieve all the form parameter names using the getParameterNames() method of the HttpServletRequest object.
- Loop through all the parameter names and their corresponding values using a while loop and the getParameterValues() method.
- Within the loop, print the parameter name and its values to the response using the println() method of the PrintWriter object.
- Close the PrintWriter object and deploy the Servlet program.

3. Write a program using JSP to print the message -"HELLO WORLD" in the browser window.

- Create a new JSP file with the .jsp extension.
- Define the HTML tags for a basic webpage, including a <title> tag and a <body> tag.
- Within the <body> tag, add an <h1> tag with the message "Hello World".
- Use JSP scriptlet tags (<% %>), and the = expression to print the message "HELLO WORLD".
- Save and run the JSP file in a web browser to display the message "HELLO WORLD" in the browser window.

4. Create a web application using Java Server Pages to retrieve the Employee table (Eid, Ename, Dept, Doj, Salary)

from SQL database via JDBC and display the retrieved table entries in the browser window.

Also include options to insert, update and delete the database entries.

- Set up the database connection using JDBC by importing the necessary JDBC libraries and creating a connection to the database.
- Create a JSP file with an HTML form to handle the CRUD (Create, Read, Update, Delete) operations for the Employee table.

- Write JSP code to retrieve the Employee table entries from the database via JDBC and display them in an HTML table on the JSP page.
- Write JSP code to handle the "Create" operation by inserting new entries into the Employee table via JDBC.
- Write JSP code to handle the "Update" operation by updating existing entries in the Employee table via JDBC.
- Write JSP code to handle the "Delete" operation by deleting entries from the Employee table via JDBC.
- Save the JSP file and deploy it on a web server, such as Apache Tomcat.
- Open the JSP file in a web browser to see the Employee table entries displayed in an HTML table.
- Use the HTML form to insert, update and delete entries in the Employee table, and refresh the page to see the updated table.

Code:

1. Write a Generic Servlet program using Java Servlet to print the message -“HELLO WORLD” in the Servlet window.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorldServlet extends HttpServlet {
 public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   out.println("<html>");
   out.println("<head>");
   out.println("<title>Hello World</title>");
   out.println("</head>");
   out.println("<body>");
   out.println("<h1>Hello World</h1>");
   out.println("</body>");
   out.println("</html>");
  }
}
```

2. Design an HTML Form that consists of the following fields and a SUBMIT button.

Emp Name Emp ID DOB

Department Salary Email ID

Write a Generic Servlet Program to retrieve all the Form Parameter Names and their corresponding values and display them in the Servlet window.

2.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletName extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html><body>");

    // Retrieve all form parameter names
    Enumeration<String> parameterNames = request.getParameterNames();

    // Loop through all parameter names and their corresponding values
    while (parameterNames.hasMoreElements()) {
      String paramName = parameterNames.nextElement();
      String[] paramValues = request.getParameterValues(paramName);
      // Print parameter name and its values
      out.println("<p>" + paramName + ": ");
      for (int i = 0; i < paramValues.length; i++) {
        out.println(paramValues[i] + "</p>");
      }
    }
    out.println("</body></html>");
  }
}
```

2.html

```
<form method="POST" action="ServletName">
    <label for="empName">Emp Name:</label>
    <input type="text" id="empName" name="empName"><br>
    <label for="empID">Emp ID:</label>
    <input type="text" id="empID" name="empID"><br>
    <label for="dob">DOB:</label>
    <input type="date" id="dob" name="dob"><br>
    <label for="department">Department:</label>
    <input type="text" id="department" name="department"><br>
    <label for="salary">Salary:</label>
    <input type="number" id="salary" name="salary"><br>
    <label for="email">Email ID:</label>
    <input type="email" id="email" name="email"><br>
    <input type="submit" value="Submit">
  </form>
```

3. Write a program using JSP to print the message -“HELLO WORLD” in the browser window.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <%-- JSP code that outputs "HELLO WORLD" --%>
    <%= "HELLO WORLD" %>
  </body>
</html>
```

4. Create a web application using Java Server Pages to retrieve the Employee table (Eid, Ename, Dept, Doj, Salary)

from SQL database via JDBC and display the retrieved table entries in the browser window.

Also include options to insert, update and delete the database entries.

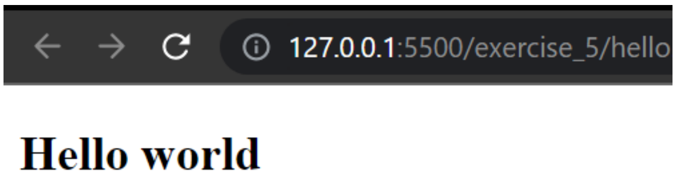
```
<%@ page import="java.sql.*" %>
<%
    // Database connection details
    String dbUrl = "jdbc:mysql://localhost:3306/employee";
    String dbUser = "root";
    String dbPassword = "password";


    // Create the database connection
    Connection conn = DriverManager.getConnection(dbUrl, dbUser, dbPassword);

    // Retrieve the Employee table entries from the database
    String sqlQuery = "SELECT * FROM employee";
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(sqlQuery);
%>
<!DOCTYPE html>
<html>
<head>
    <title>Employee Table</title>
</head>
<body>
    <h1>Employee Table</h1>
    <table border="1">
        <tr>
            <th>Eid</th>
            <th>Ename</th>
            <th>Dept</th>
            <th>Doj</th>
            <th>Salary</th>
        </tr>
        <% while (rs.next()) { %>
        <tr>
            <td><%= rs.getInt("Eid") %></td>
            <td><%= rs.getString("Ename") %></td>
            <td><%= rs.getString("Dept") %></td>
            <td><%= rs.getString("Doj") %></td>
            <td><%= rs.getDouble("Salary") %></td>
```

```
  </tr>
  <% } %>
</table>
<hr>
<h2>Insert Employee</h2>
<form action="insert.jsp" method="post">
  Eid: <input type="text" name="eid"><br>
  Ename: <input type="text" name="ename"><br>
  Dept: <input type="text" name="dept"><br>
  Doj: <input type="text" name="doj"><br>
  Salary: <input type="text" name="salary"><br>
  <input type="submit" value="Insert">
</form>
<hr>
<h2>Update Employee</h2>
<form action="update.jsp" method="post">
  Eid: <input type="text" name="eid"><br>
  Ename: <input type="text" name="ename"><br>
  Dept: <input type="text" name="dept"><br>
  Doj: <input type="text" name="doj"><br>
  Salary: <input type="text" name="salary"><br>
  <input type="submit" value="Update">
</form>
<hr>
<h2>Delete Employee</h2>
<form action="delete.jsp" method="post">
  Eid: <input type="text" name="eid"><br>
  <input type="submit" value="Delete">
</form>
```

```
  <%
    // Close the database connection
    rs.close();
    stmt.close();
    conn.close();
  %>
</body>
</html>
```

Sample Output:

1. Write a Generic Servlet program using Java Servlet to print the message -“HELLO WORLD” in the Servlet window.



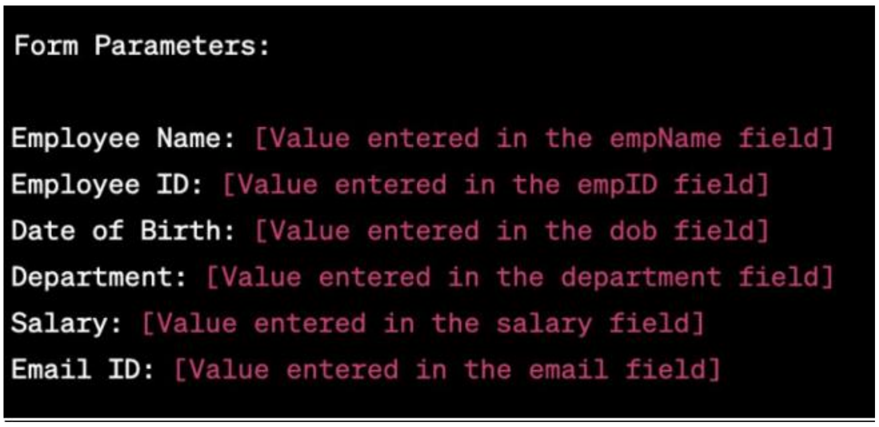
Hello world

2. Design an HTML Form that consists of the following fields and a SUBMIT button.

Emp Name Emp ID DOB

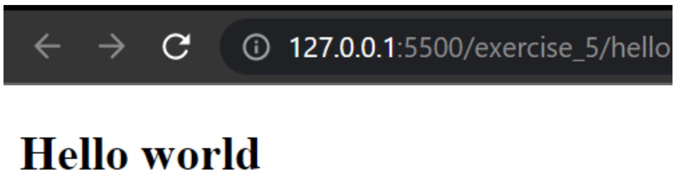
Department Salary Email ID

Write a Generic Servlet Program to retrieve all the Form Parameter Names and their corresponding values and display them in the Servlet window.



```
Form Parameters:

Employee Name: [Value entered in the empName field]
Employee ID: [Value entered in the empID field]
Date of Birth: [Value entered in the dob field]
Department: [Value entered in the department field]
Salary: [Value entered in the salary field]
Email ID: [Value entered in the email field]
```

3. Write a program using JSP to print the message -“HELLO WORLD” in the browser window.

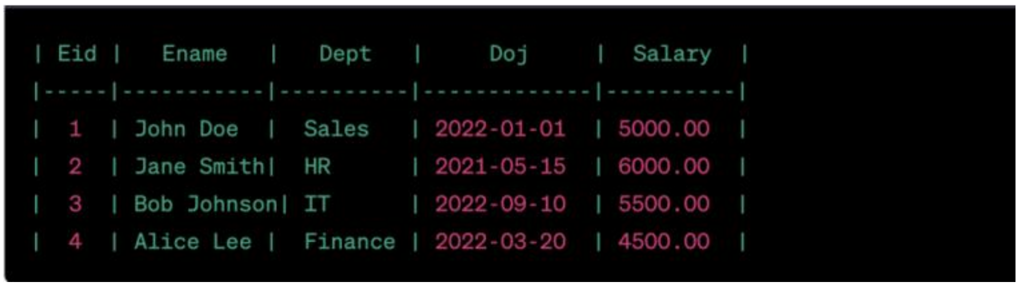


```
127.0.0.1:5500/exercise_5/hello

Hello world
```

4. Create a web application using Java Server Pages to retrieve the Employee table (Eid, Ename, Dept, Doj, Salary)

from SQL database via JDBC and display the retrieved table entries in the browser window.

Also include options to insert, update and delete the database entries.



```
| Eid |   Ename    |   Dept   |     Doj     |  Salary  |
|-----|------------|----------|-------------|----------|
|  1  | John Doe   |  Sales   | 2022-01-01  | 5000.00  |
|  2  | Jane Smith|  HR       | 2021-05-15  | 6000.00  |
|  3  | Bob Johnson| IT       | 2022-09-10  | 5500.00  |
|  4  | Alice Lee  |  Finance | 2022-03-20  | 4500.00  |
```

Creation of Three-tier real-time applications using PHP.

Ex. No: 6

Date:

Problem Statement:

Create a Web application using PHP. The application should obtain the information from the SQL database that consists of the following tables.

PATIENT (Pid, Pname, DOB, ContactNo, Address) DIAGNOSIS (Did, Dname, Medication, Department) TREATMENT (Pid, Type, Did, DoctorName)

Perform the following operations in PHP using Database Connectivity in the above tables.

i. Add a Patient Information
v. Add a Diagnosis
v. Assign a Treatment to a Patient

iv. Update a Patient Entry

v. Delete a Patient Entry

ALGORITHM:

- Create a MySQL database and tables for Patient, Diagnosis, and Treatment.
- Establish a connection to the MySQL database using PHP.
- For adding a patient, obtain the patient information from the HTML form using the $_POST superglobal variable.
- Write an SQL query to insert the patient information into the patient table.
- Execute the SQL query using the $conn object and check if the query is successful.
- For adding a diagnosis, obtain the diagnosis information from the HTML form using the $_POST superglobal variable.
- Write an SQL query to insert the diagnosis information into the diagnosis table.
- Execute the SQL query using the $conn object and check if the query is successful.

- For assigning a treatment, obtain the treatment information from the HTML form using the $_POST superglobal variable.
- Write an SQL query to insert the treatment information into the treatment table, execute the query using the $conn object and check if the query is successful.

Code:

Tables.sql

```
CREATE TABLE patient (
 Pid INT PRIMARY KEY AUTO_INCREMENT,
 Pname VARCHAR(50) NOT NULL,
 DOB DATE NOT NULL,
 ContactNo VARCHAR(15) NOT NULL,
 Address VARCHAR(100) NOT NULL
);

CREATE TABLE diagnosis (
 Did INT PRIMARY KEY AUTO_INCREMENT,
 Dname VARCHAR(50) NOT NULL,
 Medication VARCHAR(50) NOT NULL,
 Department VARCHAR(50) NOT NULL
);

CREATE TABLE treatment (
 Pid INT,
 Type VARCHAR(50) NOT NULL,
 Did INT,
 DoctorName VARCHAR(50) NOT NULL,
 FOREIGN KEY (Pid) REFERENCES patient(Pid),
 FOREIGN KEY (Did) REFERENCES diagnosis(Did)
);
```

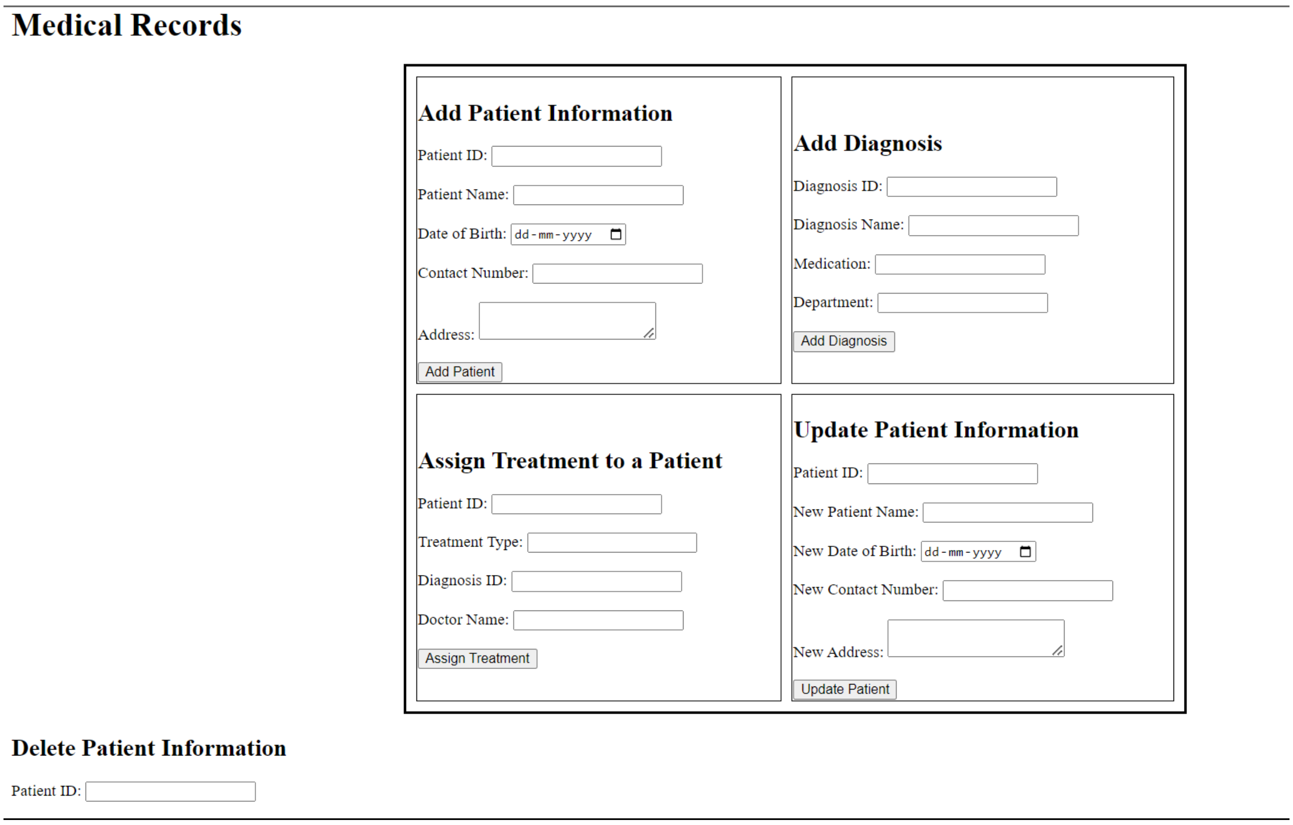
Connection.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "clinic";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
?>
```

Diagonisis.php

```
<?php
// get values from the form
$pname = $_POST["pname"];
$dob = $_POST["dob"];
$contactno = $_POST["contactno"];
$address = $_POST["address"];

// insert into patient table
$sql = "INSERT INTO patient (Pname, DOB, ContactNo, Address)
       VALUES ('$pname', '$dob', '$contactno', '$address')";

if ($conn->query($sql) === TRUE) {
   echo "New patient added successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}


$conn->close();
?>
```

Sample output:



Medical Records

Add Patient Information

Patient ID:

Patient Name:

Date of Birth: dd-mm-yyyy

Contact Number:

Address:

Add Patient

Add Diagnosis

Diagnosis ID:

Diagnosis Name:

Medication:

Department:

Add Diagnosis

Assign Treatment to a Patient

Patient ID:

Treatment Type:

Diagnosis ID:

Doctor Name:

Assign Treatment

Update Patient Information

Patient ID:

New Patient Name:

New Date of Birth: dd-mm-yyyy

New Contact Number:

New Address:

Update Patient

Delete Patient Information

Patient ID:

Creation of webpage and database with user information using AJAX.

Ex. No: 7

Date:

Problem Statement:

We have to create the webpage and database with user information using AJAX.

Algorithm:

- HTML:
    - Create a form to allow users to input their information
    - Create a table to display the user information
- CSS:
    - Style the form and table to match the design of the webpage
- JavaScript:
    - Use AJAX to retrieve data from the server-side code
    - Use AJAX to send user input to the server-side code
    - Use JavaScript to validate user input
    - Use JavaScript to update the table with the retrieved user information
- Server-side code:
    - Connect to the database
    - Retrieve data from the database using SQL statements
    - Insert data into the database using SQL statements
    - Send data back to the client-side using JSON

Code:

Server-side code

```
<?php

// Connect to the database

$servername = "localhost";
```

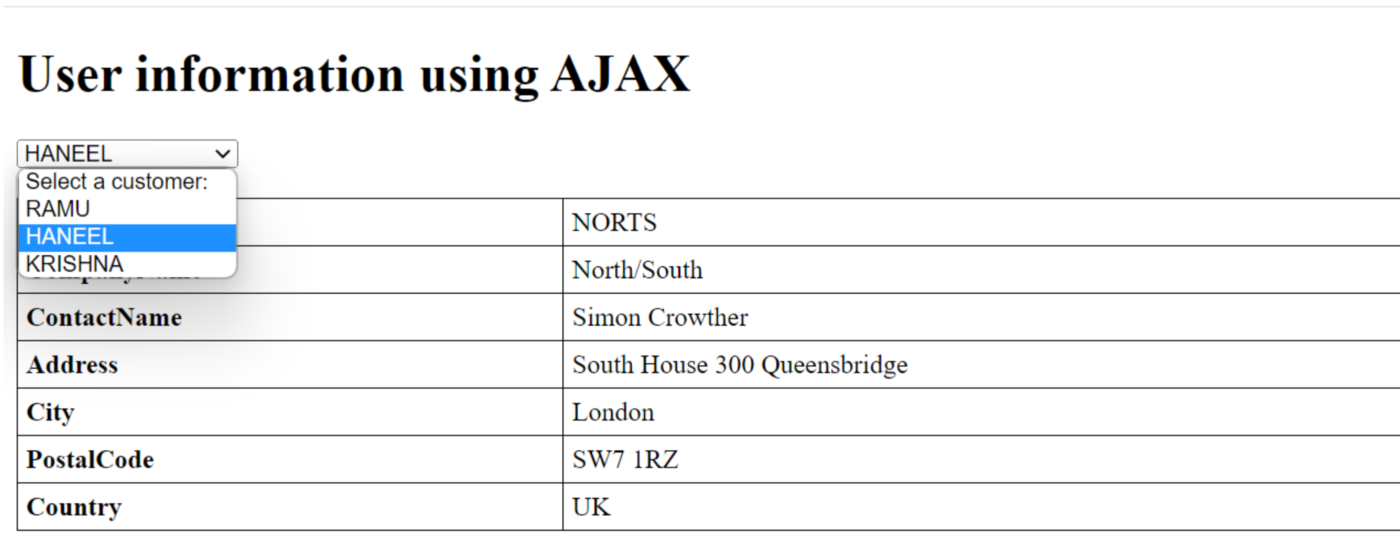
```
$username = "username";

$password = "password";

$dbname = "myDB";


$conn = new mysqli($servername, $username, $password, $dbname);


// Check connection

if ($conn->connect_error) {

  die("Connection failed: " . $conn->connect_error);

}


// Retrieve data from the database

$sql = "SELECT * FROM users";

$result = $conn->query($sql);


if ($result->num_rows > 0) {

  // Output data of each row

  while($row = $result->fetch_assoc()) {

    echo "Name: " . $row["name"]. " - Email: " . $row["email"]. "<br>";

  }

} else {

  echo "0 results";

}


// Insert data into the database

$name = $_POST["name"];

$email = $_POST["email"];


$sql = "INSERT INTO users (name, email)
```

```
VALUES ('$name', '$email')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Client-side code

```
// Retrieve data from the server-side code
$.ajax({
  url: "get_users.php",
  type: "GET",
  success: function(data) {
    // Update the table with the retrieved data
    $("#user-table tbody").html(data);
  }
});

// Send user input to the server-side code
$("#user-form").submit(function(event) {
  event.preventDefault();

  $.ajax({
    url: "add_user.php",
    type: "POST",
    data: $("#user-form").serialize(),
  success: function(data) {
    alert("User added successfully");
  }
 });
});
```

Sample Output:



User information using AJAX

	NORTS
[illegible]	North/South
ContactName	Simon Crowther
Address	South House 300 Queensbridge
City	London
PostalCode	SW7 1RZ
Country	UK

Creation of Mobile App using Table Layout and Activity Class

Ex. No: 8

Date:

Problem Statement: To design mobile applications using table layout and activity class.

Algorithm:

1. Designing a restaurant data entry form:
    - Create an XML layout file for the restaurant data entry form using TableLayout.
    - Define the necessary UI elements in the XML file, including EditText fields and a Save button.
    - Create a Java activity class for the restaurant data entry form that extends AppCompatActivity.
    - Define the necessary class variables in the Java activity class, including the TableLayout and EditText fields.
    - In the onCreate() method, set the activity content view to the XML layout file and initialize the UI elements using findViewById().
    - Define an onClickListener for the Save button that calls a saveData() method.
    - In the saveData() method, retrieve the user input from the EditText fields and validate that all fields are filled.
    - If all fields are filled, create a new TableRow and add EditText views for each field.
    - Add the TableRow to the TableLayout, clear the input fields, and display a success message using Toast.
    - If any fields are missing, display an error message using Toast.

Code:

1. Designing a restaurant data entry form:
    - Restaurant.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/tableLayout"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TableRow>
    <TextView
      android:text="Restaurant Name:"
      android:textSize="18sp" />
    <EditText
      android:id="@+id/restaurantNameEditText"
      android:layout_span="2"
      android:hint="Enter restaurant name"
      android:textSize="18sp" />
  </TableRow>
  <TableRow>
    <TextView
      android:text="Location:"
      android:textSize="18sp" />
    <EditText
      android:id="@+id/locationEditText"
      android:layout_span="2"
      android:hint="Enter restaurant location"
      android:textSize="18sp" />
  </TableRow>
  <TableRow>
    <TextView
      android:text="Cuisine Type:"
      android:textSize="18sp" />
    <EditText
      android:id="@+id/cuisineTypeEditText"
      android:layout_span="2"
      android:hint="Enter restaurant cuisine type"
      android:textSize="18sp" />
  </TableRow>
  <TableRow>
    <Button
      android:id="@+id/saveButton"
      android:text="Save"
      android:textSize="18sp" />
    <Button
      android:id="@+id/resetButton"
      android:text="Reset"
      android:textSize="18sp" />
  <Button
      android:id="@+id/cancelButton"
      android:text="Cancel"
      android:textSize="18sp" />
  </TableRow>
</TableLayout>
```

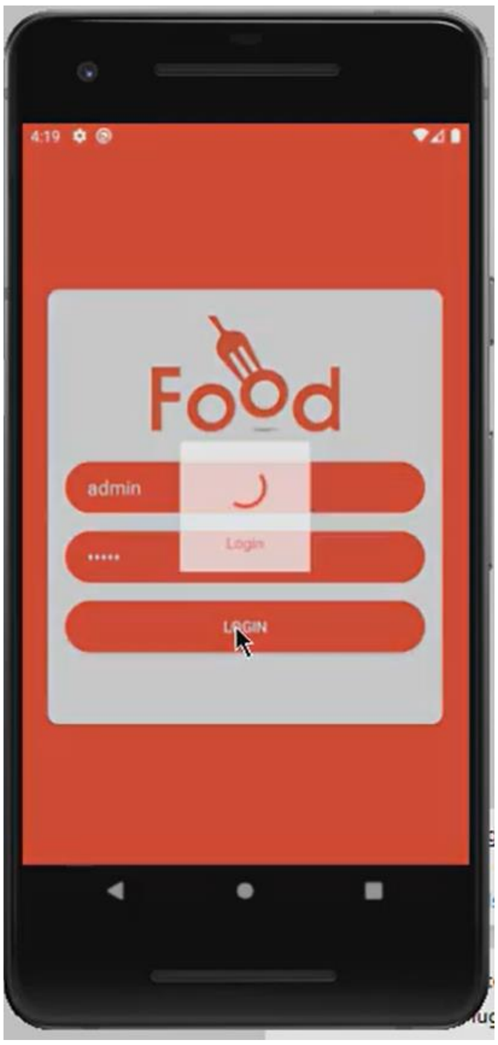
- Restaurant.java:

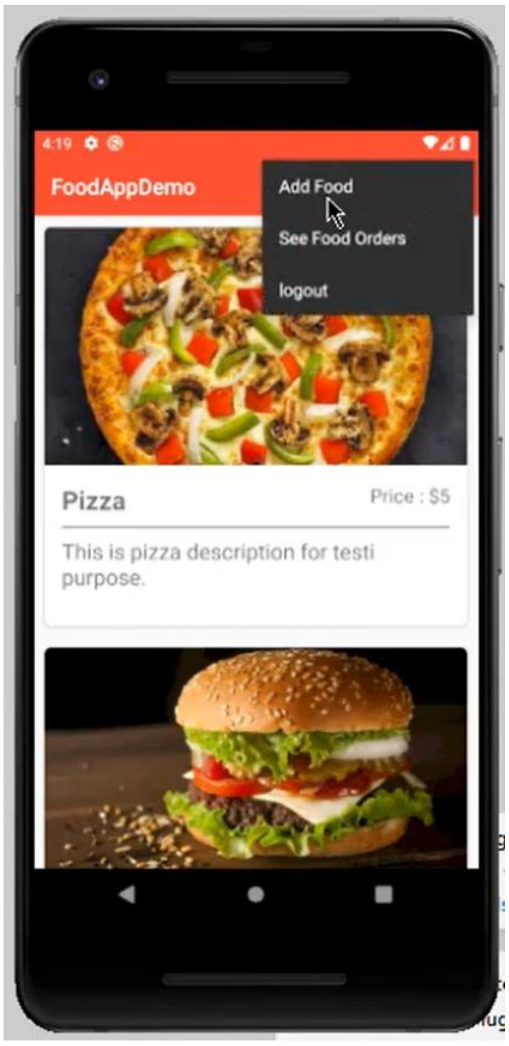
```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class RestaurantDataEntryActivity extends AppCompatActivity {
  private TableLayout tableLayout;
  private EditText nameEditText, addressEditText, phoneEditText;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_restaurant_data_entry);
    tableLayout = findViewById(R.id.table_layout);
    nameEditText = findViewById(R.id.name_edit_text);
    addressEditText = findViewById(R.id.address_edit_text);
    phoneEditText = findViewById(R.id.phone_edit_text);
    Button saveButton = findViewById(R.id.save_button);
    saveButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        saveData();
      }
    });
  }
  private void saveData() {
    String name = nameEditText.getText().toString();
    String address = addressEditText.getText().toString();
    String phone = phoneEditText.getText().toString();
    if (name.isEmpty() || address.isEmpty() || phone.isEmpty()) {
      Toast.makeText(this, "Please fill all fields", Toast.LENGTH_SHORT).show();
      return;
    }
    TableRow tableRow = new TableRow(this);
```

```
        tableRow.setLayoutParams(new TableLayout.LayoutParams(
                TableLayout.LayoutParams.MATCH_PARENT,
                TableLayout.LayoutParams.WRAP_CONTENT));
        EditText nameView = new EditText(this);
        nameView.setText(name);
        tableRow.addView(nameView);
        EditText addressView = new EditText(this);
        addressView.setText(address);
        tableRow.addView(addressView);
        EditText phoneView = new EditText(this);
        phoneView.setText(phone);
        tableRow.addView(phoneView);
        tableLayout.addView(tableRow);
        nameEditText.setText("");
        addressEditText.setText("");
        phoneEditText.setText("");
        Toast.makeText(this, "Data saved successfully", Toast.LENGTH_SHORT).show();
    }
}
```

Sample Output:

1. Designing a restaurant data entry form:





Result: Mobile applications were created successfully using table layout and activity class.

Creation of Mobile App for Image Capturing and Storage

Ex. No: 9

Date:

Problem Statement: To design a mobile application to capture and store an image in a database.

Algorithm:

1. Designing an application to capture and store an image:
   - Add the necessary permissions to your AndroidManifest.xml file for camera and storage access.
   - Create an XML layout file with a button to trigger the camera capture.
   - Create a Java activity class that handles the camera capture and database storage.
   - In the Java activity class, create an ImageView and Button object by finding them with their IDs.
   - Implement the camera capture functionality using an intent to launch the built-in camera app.
   - Implement the onActivityResult() method to retrieve the captured image and convert it to a byte array.
   - Save the image bytes to a SQLite database using a database helper class.

Code:

1. Designing an application to capture and store an image:
   - Camera.xml:

```
<Button
  android:id="@+id/capture_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Capture" />
```

   - Camera.java:

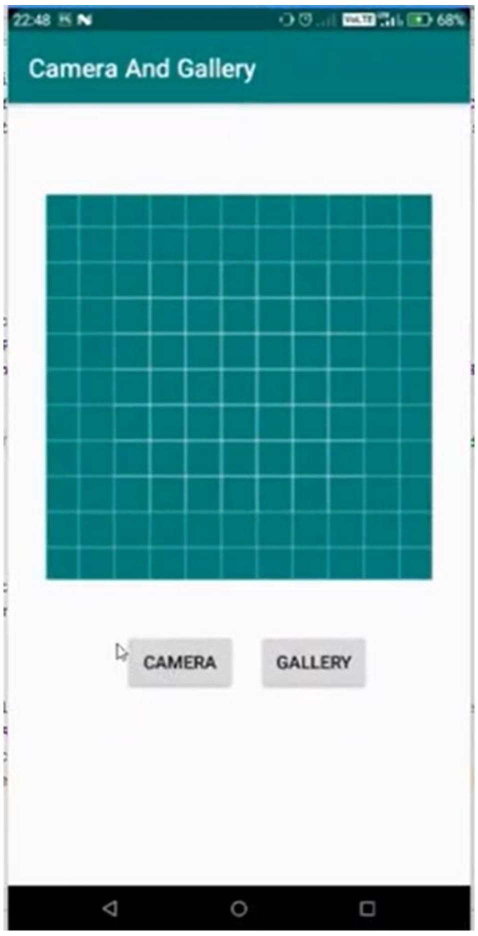
```
import android.content.ContentValues;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import java.io.ByteArrayOutputStream;

public class MainActivity extends AppCompatActivity {
  private static final String TAG = "MainActivity";
  private static final int REQUEST_IMAGE_CAPTURE = 1;
  private ImageView imageView;
  private Button captureButton;
  private byte[] imageBytes;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    imageView = findViewById(R.id.image_view);
    captureButton = findViewById(R.id.capture_button);
    captureButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        dispatchTakePictureIntent();
      }
    });
  }
  private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
      startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
  }
  @Override
  protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
      Bundle extras = data.getExtras();
      Bitmap imageBitmap = (Bitmap) extras.get("data");
      imageView.setImageBitmap(imageBitmap);
      ByteArrayOutputStream stream = new ByteArrayOutputStream();
      imageBitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);
      imageBytes = stream.toByteArray();
    }
    }
    private void saveImageToDatabase() {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(ImageContract.ImageEntry.COLUMN_NAME_IMAGE_DATA, imageBytes);
        long newRowId = db.insert(ImageContract.ImageEntry.TABLE_NAME, null, values);
        if (newRowId == -1) {
            Log.e(TAG, "Error inserting image data into database");
        } else {
            Log.i(TAG, "Image data inserted into database with row ID " + newRowId);
        }
        db.close();
    }
}
```

- Database.java:

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.provider.BaseColumns;

public class ImageDbHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "Image.db";
    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + ImageContract.ImageEntry.TABLE_NAME + " (" +
                    ImageContract.ImageEntry._ID + " INTEGER PRIMARY KEY," +
                    ImageContract.ImageEntry.COLUMN_NAME_IMAGE_DATA + " BLOB)";
    private static final String SQL_DELETE_ENTRIES =
            "DROP TABLE IF EXISTS " + ImageContract.ImageEntry.TABLE_NAME;
    public ImageDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }
    public static class ImageContract {
        private ImageContract() {}
        public static class ImageEntry implements BaseColumns {
            public static final String TABLE_NAME = "image";
            public static final String COLUMN_NAME_IMAGE_DATA = "data";
        }
    }
}
```

Sample Output:

1. Designing an application to capture and store an image:



Result: Mobile application was created for capturing and storing an image in a database.

Creation of Mobile App for OTP Verification

Ex. No: 10

Date:

Problem Statement: To design a mobile application that registers the users by verifying an OTP.

Algorithm:

1. Designing an application for OTP verification:
   - Create an XML layout file with fields for user information such as name, email, phone number, and password, along with a button for registration.
   - Create a Java activity class for the registration screen and bind the XML layout to the activity class.
   - In the activity class, initialize the EditText views and the Register button using their IDs.
   - Set an OnClickListener on the Register button to initiate the OTP verification process.
   - Use a third-party library or service to send the OTP to the user's phone number.
   - Prompt the user to enter the received OTP code.
   - Verify the OTP code against the generated code to authenticate the user.
   - If the OTP code is valid, proceed with registering the user and storing their information in a database or server.

Code:

1. Designing an application for OTP verification:
   - OTP.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:padding="16dp">
  <EditText
    android:id="@+id/edit_text_name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Name"
    android:inputType="text" />
  <EditText
```

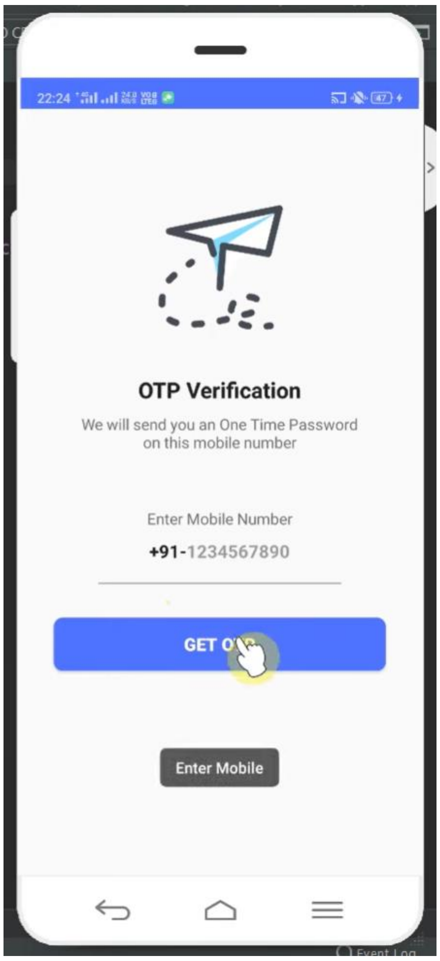
```
        android:id="@+id/edit_text_email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress" />
    <EditText
        android:id="@+id/edit_text_phone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Phone Number"
        android:inputType="phone" />
    <EditText
        android:id="@+id/edit_text_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword" />
    <Button
        android:id="@+id/button_register"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Register" />
</LinearLayout>
```

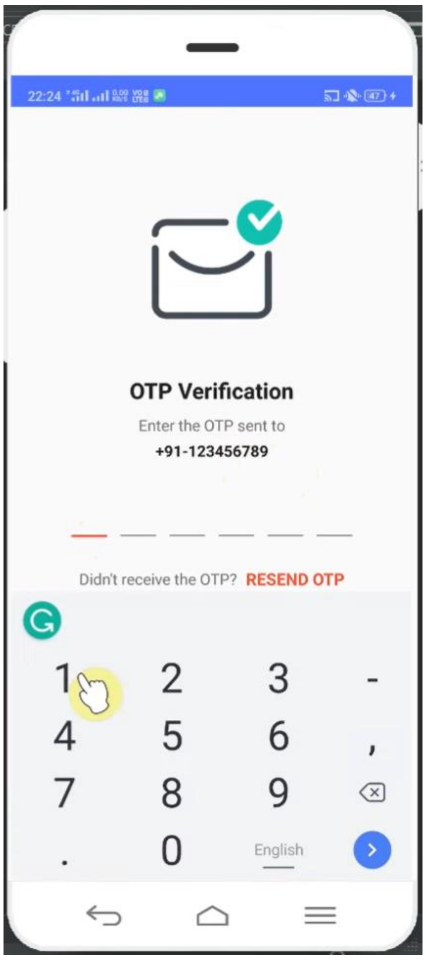
- OTP.java:

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class RegisterActivity extends AppCompatActivity {
    private EditText mNameEditText;
    private EditText mEmailEditText;
    private EditText mPhoneEditText;
    private EditText mPasswordEditText;
    private Button mRegisterButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        mNameEditText = findViewById(R.id.edit_text_name);
        mEmailEditText = findViewById(R.id.edit_text_email);
        mPhoneEditText = findViewById(R.id.edit_text_phone);
        mPasswordEditText = findViewById(R.id.edit_text_password);
        mRegisterButton = findViewById(R.id.button_register);
        mRegisterButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(RegisterActivity.this, "Registration successful!",
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }
}
```

Sample Output:

1. Designing an application for OTP verification:





Result: Mobile application was created that registers the users by verifying an OTP.

Creation of Mobile App to Convert Text to Speech

Ex. No: 11

Date:

Problem Statement: To design a mobile application to convert text to speech.

Algorithm:

1. Designing an application to convert text to speech:
    - Create a new Android project with an activity named MainActivity.
    - Add a layout file named activity_main.xml with an EditText and a Button.
    - Initialize a TextToSpeech object in MainActivity and implement the OnInitListener interface.
    - In the onInit() method, set the default language for text-to-speech using the setLanguage() method.
    - In the onDestroy() method, shutdown the TextToSpeech engine.
    - In the onSpeakButtonClick() method, get the text from the EditText and call the speak() method on the TextToSpeech object.
    - Add the necessary permissions in the AndroidManifest.xml file, such as INTERNET, ACCESS_NETWORK_STATE, RECORD_AUDIO, MODIFY_AUDIO_SETTINGS, and WAKE_LOCK.

Code:

1. Designing an application to convert text to speech:
    - Text.xml:

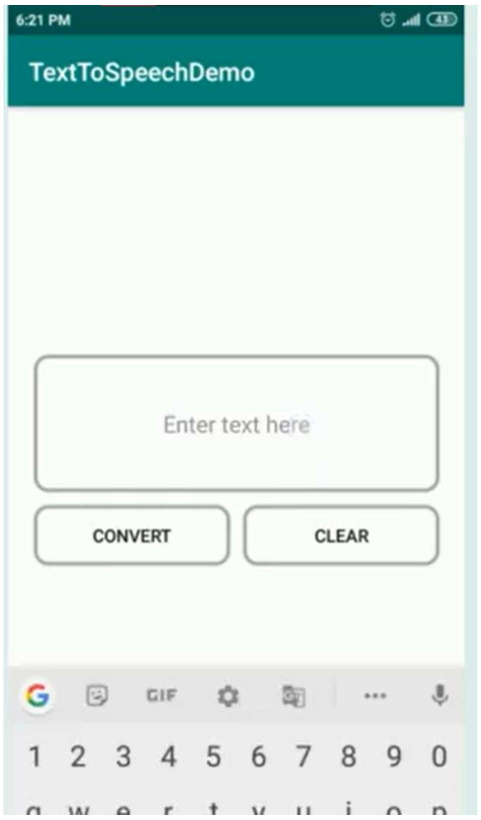
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <EditText
    android:id="@+id/edit_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter text to speak" />
  <Button
    android:id="@+id/speak_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Speak"
    android:onClick="onSpeakButtonClick" />
</LinearLayout>
```

- Text.java:

```
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.Locale;

public class MainActivity extends AppCompatActivity implements
TextToSpeech.OnInitListener {
  private TextToSpeech textToSpeech;
  private EditText editText;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textToSpeech = new TextToSpeech(this, this);
    editText = findViewById(R.id.edit_text);
  }
  @Override
  public void onInit(int status) {
    if (status == TextToSpeech.SUCCESS) {
      int result = textToSpeech.setLanguage(Locale.getDefault());
      if (result == TextToSpeech.LANG_MISSING_DATA || result ==
TextToSpeech.LANG_NOT_SUPPORTED) {
        Toast.makeText(this, "Text-to-speech language not supported",
Toast.LENGTH_SHORT).show();
      }
    } else {
      Toast.makeText(this, "Text-to-speech initialization failed",
Toast.LENGTH_SHORT).show();
    }
  }
  @Override
  protected void onDestroy() {
    super.onDestroy();
    if (textToSpeech != null) {
      textToSpeech.stop();
      textToSpeech.shutdown();
    }
    }
    public void onSpeakButtonClick(View view) {
        String text = editText.getText().toString();
        textToSpeech.speak(text, TextToSpeech.QUEUE_FLUSH, null);
    }
}
```

Sample Output:

1. Designing an application to convert text to speech:



Result: Mobile application was created to convert text to speech.

Creation of Mobile App to Convert Speech to Text

Ex. No: 12

Date:

Problem Statement: To design a mobile application to convert speech to text.

Algorithm:

1. Designing an application to convert speech to text:
    - Create a new Android project in Android Studio and set up the necessary files, including the MainActivity Java class and the activity_main XML layout file.
    - Add a button and an EditText view to the activity_main XML layout file.
    - In the MainActivity Java class, define a SpeechRecognizer object and a constant integer variable REQ_CODE_SPEECH_INPUT.
    - In the onCreate() method, initialize the EditText view and the button, and set an OnClickListener for the button that calls a new method called promptSpeechInput().
    - In the promptSpeechInput() method, create a new Intent with the RecognizerIntent.ACTION_RECOGNIZE_SPEECH action.
    - Add extras to the Intent to specify the language model, language, and prompt message.
    - Start the activity for result with the Intent and the REQ_CODE_SPEECH_INPUT constant.
    - Override the onActivityResult() method to handle the result of the speech recognition activity.
    - In the onActivityResult() method, check the result code and retrieve the text from the recognized speech using the data intent. Set the text in the EditText view.

Code:

1. Designing an application to convert speech to text:
    - Speech.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:padding="16dp">
  <EditText
```

```
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:hint="Enter text here" />
  <Button
    android:id="@+id/btnSpeak"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Speak" />
</LinearLayout>
```

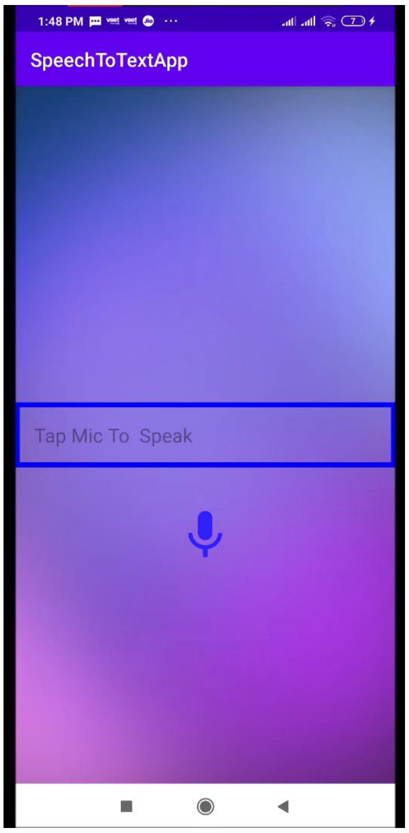
- Speech.java:

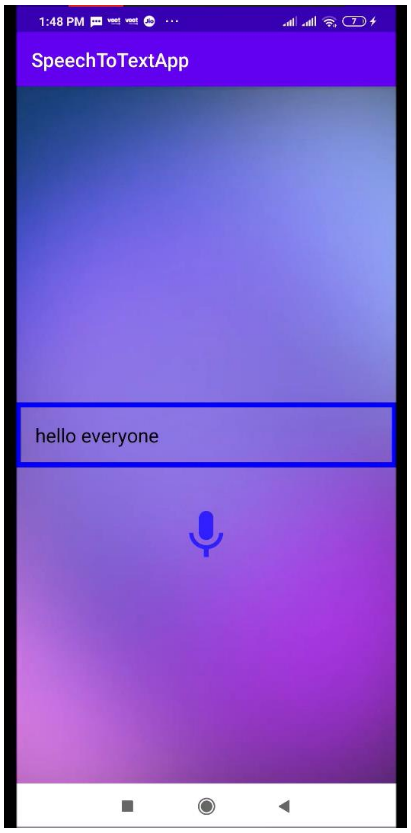
```
import android.Manifest;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import java.util.ArrayList;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {
  private final int REQ_CODE_SPEECH_INPUT = 100;
  private EditText editText;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    editText = findViewById(R.id.editText);
    Button btnSpeak = findViewById(R.id.btnSpeak);
    btnSpeak.setOnClickListener(view -> promptSpeechInput());
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO)
        != PackageManager.PERMISSION_GRANTED) {
      ActivityCompat.requestPermissions(this,
          new String[]{Manifest.permission.RECORD_AUDIO}, 1);
    }
  }
```

```
    private void promptSpeechInput() {
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
                getString(R.string.speech_prompt));
        try {
            startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
        } catch (ActivityNotFoundException a) {
            Toast.makeText(getApplicationContext(),
                    getString(R.string.speech_not_supported),
                    Toast.LENGTH_SHORT).show();
        }
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
            case REQ_CODE_SPEECH_INPUT: {
                if (resultCode == RESULT_OK && null != data) {
                    ArrayList<String> result = data
                            .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                    editText.setText(result.get(0));
                }
                break;
            }

        }
    }
}
```

Sample Output:

1. Designing an application to convert speech to text:





Result: Mobile application was created to convert speech to text.