DAYANANDA SAGAR COLLEGE OF ENGINEERING
SHAVIGE MALLESHWARA HILLS, KS LAYOUT, BANGALORE-560078

Department of Computer Science and Engineering

2023-2024

Third Semester

Web Technology Laboratory Manual

Sub Code: 22CSL35

COMPILED BY

Prof. Anitha M

Prof Amith Pradhan

DSCE
DEPARTMENT OF COMPUTER SCIENCE

DAYANANDA SAGAR COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision and Mission of the Department

Vision

To provide a vibrant learning environment in computer science and engineering with focus on industry needs and research, for the students to be successful global professionals contributing to the society.

Mission

* To adopt a contemporary teaching learning process with emphasis on hands on and collaborative learning

* To facilitate skill development through additional training and encourage student forums for enhanced learning.

* To collaborate with industry partners and professional societies and make the students industry ready.

* To encourage innovation through multidisciplinary research and development activities

* To inculcate human values and ethics to groom the students to be responsible citizens.

DSCE
DEPARTMENT OF COMPUTER SCIENCE

DAYANANDA SAGAR COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Code of Conduct in the Lab

Do’s

Students shall

- Come prepared for the program to be developed in the Laboratory.
- Report any broken plugs or exposed electrical wires to your faculty/laboratory technician immediately.
- Turn off the machine once you have finished using it.
- Maintain silence while working in the lab.
- Keep the Computer lab premises clean and tidy.
- Place backpacks under the table or computer counters.
- Treat fellow users of the laboratory, and all equipment within the laboratory, with the appropriate level of care and respect.

Don’ts

Students shall not

- Talk on cell phones in the lab.
- Eat or drink in the laboratory.
- Touch, connect or disconnect any plug or cable without the faculty/laboratory technician’s permission.
- Install or download any software or modify or delete any system files on any lab computers.
- Read or modify other users' files.
- Meddle with other users files.
- Leave their personal belongings unattended. We are not responsible for any theft.

Course Objectives

1. To understand the various elements of a web page
2. To apply Javascript in creating dynamic interactive web pages.
3. To analyze various elements of ReactJS to design the user interface.
4. To track and maintain user information among web pages.

Course Outcomes

At the end of the course, student will be able to:

CO1	Identify web page elements and attributes using HTML and CSS
CO2	Demonstrate reusable multiple components in web pages using ReactJS
CO3	Apply DOM concepts in creating dynamic web pages using JavaScript
CO4	Implement session management using PHP scripts
CO5	Design and develop dynamic and responsive web pages

Contents

1a) Write an HTML code to display your education details in a tabular format.

```
<html>
<head>
        <title>Education details</title>
<style>
table.center{
   margin-left: auto;
   margin-right: auto;
}
</style>
</head>
<body>
<h1 align="center" style="color:blue">EDUCATION DETAILS</h1><br><br>
<table border="1" background="gray" cellspacing="5" cellpadding="5" class="center">
<th>SNo</th>
<th>Course</th>
<th>Board/University</th>
<th>School/College</th>
<th>Year of Passing</th>
<th>Percentage</th>

<tr>
<td>1</td>
<td>10</td>
<td>ICSE</td>
<td>DPS</td>
<td>2008</td>
<td>90</td>
</tr>
```

```
<tr>
<td>2</td>
<td>12</td>
<td>ICSE</td>
<td>DPS</td>
<td>2010</td>
<td>95</td>
</tr>

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

1b) Write an HTML code to create a Registration Form

Create a “registration form” with the following fields

1) Name (Text field)
2) Password (password field)
3) E-mail id (text field)
4) Phone number (text field)
5) Gender (radio button)
6) Date of birth (3 select boxes)
7) Languages known (check boxes – Kannada, English, French, Spanish)
8) Address (text area)

```
<html>
<head>
<title>Registration Page</title>
</head>
<body>
<center>
<form action="/action_page.php">
<h3 align="center"><u>REGISTRATION PAGE</u></h3>

<table border="3">
<tr><td>
<table cellspacing="10" cellpadding="5">
<tr><td>NAME</td><td><input type="text" size="30" name="uname"/></td></tr>
<tr><td>PASSWORD</td><td><input type="password" size="30" name="pass"/></td></tr>

<tr><td>E-MAIL ID</td><td><input type="text" size="30" name="email"/></td></tr>
<tr><td>PHONE NUMBER</td><td><input type="text" size="15"
name="phone"/></td></tr>

<tr><td>GENDER</td><td><input type="radio" name="gen" value="m" />MALE
```

```
<input type="radio" name="gen" value="f" />FEMALE </td></tr>
<tr><td>DATE OF BIRTH</td>
<td><select name="day">
<option value="day">DAY</option>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
```

```
<option value="31">31</option>
</select>

<select name="month">
<option value="month">MONTH</option>
<option value="jan">JANUARY</option>
<option value="feb">FEBRUARY</option>
<option value="mar">MARCH</option>
<option value="apr">APRIL</option>
<option value="may">MAY</option>
<option value="jun">JUNE</option>
<option value="jul">JULY</option>
<option value="aug">AUGUST</option>
<option value="sep">SEPTEMBER</option>
<option value="oct">OCTOBER</option>
<option value="nov">NOVEMBER</option>
<option value="dec">DECEMBER</option>
</select>

<select name="year">
<option value="year">YEAR</option>
<option value="1986">1986</option>
<option value="1987">1987</option>
<option value="1988">1988</option>
<option value="1989">1989</option>
<option value="1990">1990</option>
<option value="1991">1991</option>
<option value="1992">1992</option>
<option value="1993">1993</option>
<option value="1994">1994</option>
<option value="1995">1995</option>
<option value="1996">1996</option>
<option value="1997">1997</option>
<option value="1998">1998</option>
```

```
<option value="1999">1999</option>
<option value="2000">2000</option>
<option value="2001">2001</option>
<option value="2002">2002</option>
<option value="2003">2003</option>
<option value="2004">2004</option>
<option value="2005">2005</option>
<option value="2006">2006</option>
<option value="2007">2007</option>
<option value="2008">2008</option>
<option value="2009">2009</option>
<option value="2010">2010</option>
<option value="2011">2011</option>
<option value="2012">2012</option>
<option value="2013">2013</option>
<option value="2014">2014</option>
<option value="2015">2015</option>
<option value="2016">2016</option>
<option value="2017">2017</option>
<option value="2018">2018</option>
<option value="2019">2019</option>
<option value="2020">2020</option>
<option value="2021">2021</option>
<option value="2022">2022</option>
<option value="2023">2023</option>

</select></td></tr>
<tr><td>LANGUAGES KNOWN</td>
<td>
<input type="checkbox" value="kan" name="lang" />KANNADA
<input type="checkbox" value="eng" name="lang" />ENGLISH
<input type="checkbox" value="fre" name="lang" />FRENCH
<input type="checkbox" value="spa" name="lang" />SPANISH
</td></tr>
```

```html
<tr>
<td>ADDRESS</td>
<td><textarea name="addr" cols="25" rows="5"></textarea></td></tr>
<tr><td colspan="2" align="center"><input type="submit" value="SUBMIT"/>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="reset" value="RESET" /></td>
</tr>
</table>
</form>
</td></tr></table>
</center>
</body>
</html>
```

2a) Develop and demonstrate the usage of inline, embedded and external style sheet using CSS.

Prgm2a.html

```
<!--Inline Style-->
<html>
<head>
<title>Inline Style</title>
</head>
<body>
<p style="text-align:center; color: sienna; margin-left: 60px; margin-top: 60px ;font-size: 40pt"> Welcome to
Web Technologies lab </p>
</body>
</html>
<!--Embedded Style-->
<html>
<head>
<style>

   h1{font-family: serif; text-align:center;}

   p{color: sienna; text-align:center;
   margin-left: 60px;
   margin-top: 60px ;
   font-size: 40pt;}

   .blue{color:blue; text-align:center;}
```

```
</style>
</head>
<body>
<h1 class="blue"> Ability Enhancement </h1>
<p> Web Technology Lab </p>
</body>
</html>
```

<!--External Style-->

```
<head>
    <link rel="stylesheet" href="pgrm2a.css">
</head>
<body>
    <h1 class="blue"> Ability Enhancement </h1>
    <p> Web Technology Lab </p>
</body>
</htm
```

<u>Pgrm2a.css</u>

```
h1{font-family: serif; text-align:center;}
p{color: red; text-align:center;
    margin-left: 60px;
    margin-top: 60px ;
    font-size: 40pt;}
.blue{color:blue; text-align:center;
```

2b) Write an HTML code to display short profile card applying proper CSS style.

Prgm2b.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
    <linkrel="stylesheet"href="https://cdnjs.cloudflare.com/ajax/libs/font- awesome/5.15.3/css/all.min.css"/>
    <link rel="stylesheet" href="pgrm2b.css">
    <title>Profile Card</title>
</head>
<body>
    <div class="card">
        <div class="card-header">
            <img src=" img.jpg " alt="Profile Image" class="profile-img">
        </div>
        <div class="card-body">
            <p class="name">Your Name</p>
            <a href="#" class="mail">yourname@gmail.com</a>
        <p class="job">Developer | Designer</p>
        </div>
        <div class="social-links">
            <a href="#" class="fab fa-github social-icon"></a>
            <a href="#" class="fab fa-twitter social-icon"></a>
            <a href="#" class="fab fa-youtube social-icon"></a>
            <a href="#" class="fab fa-linkedin social-icon"></a>
        </div>
        <div class="card-footer">
        <p class="count"><span>120k</span> Followers | <span>10k</span> Following</p>
        </div>
      </div>
  </body>
</html>
```

Prgm2b.css

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    text-decoration: none;
    transition: 0.3s;
}

body {
    font-family: "Montserrat";
    background-color: #b8b6b6;
    color: #fdfdfd;
}

.card {
    max-width: 250px;
    margin: 150px auto 0;
    background-color: #42515a;
    box-shadow: 0 10px 90px #00000024;
    text-align: center;
    font-size: 20px;
    border-radius: 15px;
}

.card    .card-header    {
    position:      relative;
    height: 48px; }
```

```
.card .card-header .profile-img {
    width: 130px;
    height: 130px;
    border-radius: 1000px;
    position: absolute; left:
    50%;
    transform: translate(-50%, -50%);
    border: 8px solid #c74385;
    box-shadow: 0 0 20px #00000033;
}

.card .card-header .profile-img:hover { width:
    180px;
    height: 180px;
    border: 8px solid #d885af;
}

.card .card-body { padding:
    10px 40px;
}

.card .card-body .name {
    margin-top: 30px;
    font-size: 22px;
    font-weight: bold;
    color: #c74385;
}

.card .card-body .name:hover {
    margin-top: 30px;
    font-size: 24px;
    color: #d885af;
}
```

```
.card .card-body .mail {
    font-size: 14px; color:
    #c2bdbd;
}

.card .card-body .mail:hover {
    font-size: 16px;
    color: #ffffff;
}

.card .card-body .job {
    margin-top: 10px;
    font-size: 14px;
}

.card .social-links {
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: 30px;
}

.card .social-links .social-icon {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    height: 40px;
    width: 40px;
    background-color: #c74385;
    color: #ffffff;
    font-size: 20px;
    border-radius: 100%;
    text-decoration: none;
    margin: 0 13px 30px 0;
}

.card .social-links .social-icon:last-child {
 margin-right: 0;
}

.card .social-links .social-icon:hover {
    background-color: #d885af;
    height: 50px;
    width: 50px;
    text-decoration: none;
}

.card .card-footer {
    background-color: #c74385;
    border-bottom-left-radius: 15px;
    border-bottom-right-radius: 15px;
    padding: 20px 0 20px 0;
}

.card .card-footer .count {
    font-size: 14px;
}

@media screen and (max-width: 575px) {
    .card {
        width: 96%;
    }

    .card .card-body { padding:
        10px 20px;
    }
}
```

3a) Develop and demonstrate the usage of dynamic changes of CSS.

<u>Prgm3a.html</u>

```
<html>
<body>

<div onmousedown="mDown(this)"
onmouseup="mUp(this)"style="background-color:#D94A38;width:90px;height:20px;paddin
g:40px;">Click Me</div>

<script>
function mDown(obj) {
  obj.style.backgroundColor="#1ec5e5"; obj.innerHTML =
  "Release Me";
}

function mUp(obj) {
  obj.style.backgroundColor="#D94A38";
  obj.innerHTML="Thank You";
}
</script>

</body>
</html>
```

3b. Write an HTML page that reads name and USN from the form and displays it

Prgm3b.html

```
<html>
<head>
   <title>Events</title>
</head>

<scripttype="text/javascript"> function msg()
   {varname=document.person.n1; var usn=
       document.person.n2;
  alert("Hello"+name.value+"USN"+usn.value);
 }
</script>

<body>
</br></br>
<formname="person">

EnterThename</br>
<inputtype="text"name="n1"></br></br>

EnterTheUSN</br>
<inputtype="text"name="n2">

<inputtype="submit"onClick="msg();">

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

3c) Write an HTML page that contains a selection box with a list of 5 countries. When the user selects a country, its capital should be printed next in the list. Add CSS to customize the properties of the font of the capital (color, bold and font size).

<u>Prgm3c.html</u>

```
<html>
<head>
<title>CapitalsofCountries</title>

<style>p{
color:red;
font-weight:bold;
font-size:50;
}
</style>

<scriptlanguage="javascript">
function capital()
{
var cunt=document.forms["frm1"].country.value; var
capital=" Please select any country ";
if(cunt=="INDIA")
{
capital="NEWDELHI";
}
if(cunt=="USA")
{
capital="WASHINGTONDC";
}
if(cunt=="NETHERLANDS")
{
capital="AMSTERDAM";
}
if(cunt=="RUSSIA")
{
capital="MOSCOW";
}
if(cunt=="JAPAN")
{
capital="TOKYO";
}
if(cunt=="select")
{
capital="PleaseselectanyCountry";
}
document.getElementById("capt").innerHTML=capital;
}
</script>
</head>

<body>
<formname="frm1">
<br/>
<center>
SelectaCountry:<selectname="country"onchange="capital()">
<optionvalue="select">--SELECT--</option>
<optionvalue="INDIA">INDIA</option>
<optionvalue="USA">USA</option>
<optionvalue="NETHERLANDS">NETHERLANDS</option>
<optionvalue="RUSSIA">RUSSIA</option>
<optionvalue="JAPAN">JAPAN</option>
</select>
<br/>
<br/>
```

```
<fontcolor="green"size="6">Capitalis:</font><p id="capt"></p>
</center>
</form>
</body>
</html>
```

WEB TECHNOLOGY LABORATORY (22CSL35))

4. Develop a Progressive Web Application (PWA) using HTML, CSS and JavaScript.

Progressive web apps are a way to bring that native app feeling to a traditional web app. With PWAs we can enhance our website with mobile app features which increase usability and offer a great user experience. It gives you the ability,

- To install it on a mobile home screen
- To access it when offline
- To access the camera
- To get push notifications
- To do background synchronization

PWA has five technical components,

1. **Web App manifest**

   The web app manifest is the first component of the PWA. It is a simple JSON file that controls a user's application. Usually, it is named "manifest.json". It is the most important component for the presence of PWA. When you first connect PWA to a network, a mobile browser reads the "manifest.json" file and stores it locally in cache memory.

2. **Application shell**

   It is specialized to split the static and dynamic content of the application. The minimal HTML, CSS, JavaScript and any other dynamic and static resources offer the structure for your web page. It reduces the actual content that is unique to the webpage. This component ensures a very critical approach to the development of progressive web apps.

3. **Service worker**

   A service worker is a web worker. It is a JavaScript file that runs aside from the mobile browser. In other words, it is another technical component that promotes the functionality of PWA. The service worker retrieves the resources from the cache memory and delivers the messages.

4. **Webpack**

   It is used to design the PWA front-end. It allows the PWA-developers to gather all JavaScript resources and data in one location.

5. **Transport Layer Security (TLS)**

   This component is a standard for all robust and secure data exchange between any two applications. The integrity of the data requires the website's service through the HTTPS and an SSL certificate installed on the server.

Create a new project using the following structure. Create a new folder for your project and set up the following files inside it:

- **pgrm4.html**: The main HTML file for your PWA.
- **pgrm4.css**: The CSS file to style your application.
- **app.js**: The JavaScript file for handling interactivity and service workers.
- **service-worker.js**: The service worker file.

prgm4.html

```
<!DOCTYPE html>
//<link rel="manifest" href="/Users/amithpradhaan/Documents/ABILITY ENHANCEMENT (WEB
TECHNOLOGY)/Pgrm4/manifest.json">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ToDo App</title>
  <link rel="stylesheet" href="pgrm4.css">
</head>

<body>
  <header>
    <h1>ToDo App</h1>
  </header>

  <main>
    <form id="todo-form">
      <input type="text" id="todo-input" placeholder="Enter a task">
      <button type="submit">Add</button>
    </form>
    <ul id="todo-list"></ul>
  </main>

  <script src="app.js"></script>
</body>

</html>
```

prgm4.css

```
body {
   font-family: Arial, sans-serif;
 }

 header {
   background-color: #f2f2f2;
   padding: 20px;
   text-align: center;
 }

 h1 { margin:
   0;
 }

 form { display:
   flex;
   margin-bottom: 20px;
 }

 input[type="text"] {
   flex: 1;
   padding: 10px;
 }

 button {
   padding: 10px 20px;
   background-color: #4CAF50;
   color: #fff;
   border: none;
   cursor: pointer;
 }
```

```
  ul {
    list-style-type: none;
    padding: 0;
  }

  li {
    display: flex;
    align-items: center;
    padding: 10px;
    border-bottom: 1px solid #ccc;
  }

  li:last-child {
    border-bottom: none;
  }

  .completed {
    text-decoration: line-through;
  }
```

app.js

```
// Register Service Worker
if ('serviceWorker' in navigator) {
    navigator.serviceWorker
      .register('service-worker.js')
      .then(function() {
        console.log('Service Worker Registered');
      });
  }

  // DOM elements
  const todoForm = document.getElementById('todo-form'); const
  todoInput = document.getElementById('todo-input');
```

```
const todoList = document.getElementById('todo-list');

// Store tasks in an array let
tasks = [];

// Render tasks in the list function
renderTasks() {
todoList.innerHTML = '';

  tasks.forEach(function(task) {
    const li = document.createElement('li');
    li.textContent = task.title;

    if (task.completed) {
      li.classList.add('completed');
    }

    li.addEventListener('click', function() {
      task.completed = !task.completed;
      renderTasks();
    });

    todoList.appendChild(li);
  });
}

// Handle form submission todoForm.addEventListener('submit',
function(event) { event.preventDefault();

  const taskTitle = todoInput.value.trim();

  if (taskTitle !== '') {
    const task = {
        title: taskTitle,
        completed: false
      };

      tasks.push(task);

      renderTasks();

      todoInput.value = '';
    }
  });
```

service-worker.js

```
// service-worker.js
const CACHE_NAME = 'my-pwa-cache-v1'; const
urlsToCache = [
  '/',
  'pgrm4.html',
  'prgm4.css',
  'app.js'
];

self.addEventListener('install', (event) => {
  event.waitUntil( caches.open(CACHE_NAME)
      .then((cache) => cache.addAll(urlsToCache))
  );
});

self.addEventListener('fetch', (event) => {
  event.respondWith( caches.match(event.request)
      .then((response) => response || fetch(event.request))
  );
});
```

5. Develop a simple task tracker application from scratch by making use of React.js.

TodoItem.js

```
import React, { useState } from "react";

const TodoItem = (props) => {
  const { eachTodo, onDeleteTodo, onUpdateTodo } = props;

  const [editing, setEditing] = useState(false);
  const [title, setUpdatedTitle] = useState(eachTodo.title);
  const [description, setUpdatedDescription] = useState(eachTodo.description);
  const [completed, setCompleted] = useState(eachTodo.completed);

  const handleUpdate = () => {
    onUpdateTodo(eachTodo.id, title, description, completed);
    setEditing(false);
  };

  const handleDelete = () => {      onDeleteTodo(eachTodo.id);    };
  const handleToggle = () => {     setCompleted(!completed);
    onUpdateTodo(eachTodo.id, title, description, completed);
  };

  const handleEdit = () => {      setEditing(true);    };

  return (
    <div className={`arrayofList ${completed ? "completed" : ""}`}>
      {editing ? (
        <>
          <input           type="text"           value={title}
            onChange={(e) => setUpdatedTitle(e.target.value)}          />

          <input           type="text"           value={description}
            onChange={(e) => setUpdatedDescription(e.target.value)}      />
          <button onClick={handleUpdate}>Update</button>
        </>
      ) : (
        <>
          <span className="todo-title">{title}</span>
          <span className="todo-description">{description}</span>
          <span>
            <button onClick={handleToggle}>  {completed ? "Incomplete" : "Complete"}    </button>
            <button onClick={handleEdit}>Edit</button>
            <button onClick={handleDelete}>Delete</button>
          </span>
        </>
      )}
    </div>
  );
};

export default TodoItem;
```

TodoList.js

```
import React, { useState } from "react";
import TodoItem from "./TodoItem";

// Sample todos data
const todosListArray = [
  {
    id: 1,
    title: "ReactJS",
    description: "ReactJS description",
    completed: false,
  },
  {
    id: 2,
    title: "Javascript",
    description: "Javascript description",
    completed: true,
  },
];

const TodoList = () => {
  // storing sample todos data into a state to perform CRUD operations
  const [todosList, setTodosList] = useState(todosListArray);
  const [title, setTitle] = useState("");
  const [description, setDescripion] = useState("");

  // Adding a new todo
  const handleSubmit = (e) => {
    e.preventDefault();
    if (title !== "" && description !== "") {
      const newTodo = {
        id: todosList.length + 1,
        title: title,
        description: description,
      };
      setTodosList((prevTodosList) => [...prevTodosList, newTodo]);
      setTitle("");
      setDescripion("");
    } else {
      alert("Please enter all the fields to proceed...!");
    }
  };

  // updating existing todo
  const onUpdateTodo = (id, title, description, completed) => {
    setTodosList((prevTodosList) =>
      prevTodosList.map((eachTodo) =>
        eachTodo.id === id
          ? {
              ...eachTodo,
              title: title,
              description: description,
              completed: completed,
            }
          : eachTodo
      )
    );
```

```
};

// deleting todo
const onDeleteTodo = (id) => {
  setTodosList((prevTodosList) =>
    prevTodosList.filter((eachTodo) => eachTodo.id !== id)
  );
};



return (
  <div>

    Todo Manager
    {/* Adding Todo */}
    <div>
      <form onSubmit={handleSubmit}>

        <input  placeholder="title"   name="title"      value={title}
          onChange={(e) => setTitle(e.target.value)}            />

        <input    placeholder="description"        value={description}
          onChange={(e) => setDescripion(e.target.value)}            />

        <button onSubmit={handleSubmit}>Add</button>
      </form>
    </div>

    {/* Displaying todos */}
    {  todosList.map((eachTodo) => {
      return (
        <TodoItem
          key={eachTodo.id}
          eachTodo={eachTodo}
          onUpdateTodo={onUpdateTodo}
          onDeleteTodo={onDeleteTodo}
        />
      );
      })}
    </div>
  );
};

export default TodoList;
```

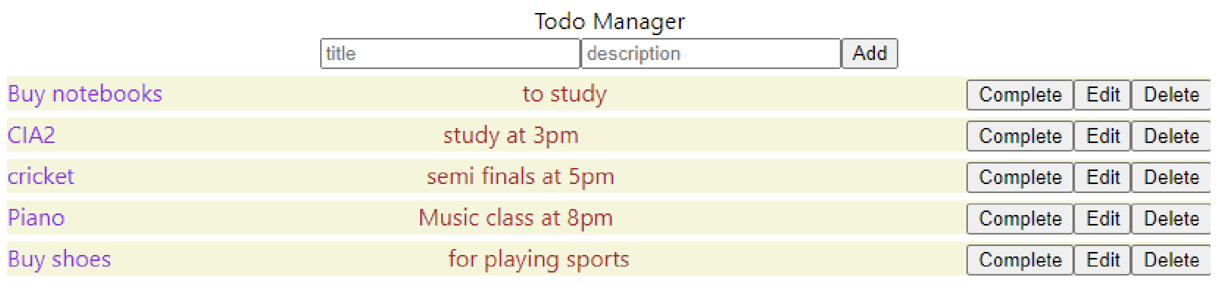
App.js

```
import "./App.css";
import TodoList from "./components/TodoList";

function App() {
  return (
    <div className="App">
      <TodoList />
    </div>
  );
}

export default App;
```

Output



```
                         Todo Manager
          title          description          Add
Buy notebooks                to study                Complete Edit Delete
CIA2                    study at 3pm                 Complete Edit Delete
cricket               semi finals at 5pm             Complete Edit Delete
Piano                 Music class at 8pm             Complete Edit Delete
Buy shoes              for playing sports            Complete Edit Delete
```

6. Develop a WebRTC (Web Real-Time Communication) platform for a video calling website using HTML and CSS.

What is WebRTC?

WebRTC, which stands for Web Real-Time Communication, is an open-source technology and set of APIs (Application Programming Interfaces) that enables real-time communication directly between web browsers or other compatible applications. It enables peer-to-peer communication, such as audio and video conferencing, as well as data sharing, without requiring users to install additional plugins or software.

WebRTC is built into modern web browsers like Google Chrome, Mozilla Firefox, and Microsoft Edge, making it possible for developers to create applications with real-time communication features directly within the browser. It leverages a combination of audio and video codecs, networking protocols, and JavaScript APIs to facilitate secure and efficient communication between devices.

Key features of WebRTC include:

1. **Audio and Video Communication**: WebRTC allows for high-quality audio and video streaming between browsers, facilitating real-time conversations and video conferencing.
2. **Peer-to-Peer Data Channel**: In addition to audio and video, WebRTC provides a data channel that allows browsers to exchange arbitrary data directly, making it suitable for file sharing, gaming, and other interactive applications.
3. **Encryption and Security**: WebRTC incorporates encryption mechanisms to ensure the security and privacy of communications, making it suitable for sensitive conversations.
4. **Cross-Platform Compatibility**: Since WebRTC is supported by major web browsers, developers can create cross-platform applications that work seamlessly on various devices and operating systems.
5. **No Plugins or Downloads**: Unlike traditional communication technologies that often require users to install plugins or software,

WebRTC is natively supported by modern browsers, simplifying the user experience. WebRTC has a wide range of applications, including:

Video conferencing and online meetings

Voice and video calls in web applications

Real-time gaming and collaboration

Live streaming and broadcasting

Remote desktop sharing and support

Internet of Things (IoT) applications

WebRTC has significantly impacted the way real-time communication is implemented on the web, enabling developers to create rich and interactive experiences without the need for third- party plugins or complex setups.

What is an SDK?

SDK stands for software development kit, also known as a devkit, the SDK is a set of software- building tools for a specific platform, including the building blocks, debuggers and often, a framework or group of code libraries such as a set of routines specific to an operating system (OS).

A typical SDK might include some or all of these resources in its set of tools:

Compiler: Translates from one programming language to the one in which you will work

Code samples: Give a concrete example of an application or web page

Code libraries (framework): Provide a shortcut with code sequences that programmers will use repeatedly

Testing and analytics tools: Provide insight into how the application or product performs in testing and production environments

Documentation: Gives developers instructions they can refer to as they go

Debuggers: Help teams spot errors in their code so they can push out code that works as expected

Often, at least one API is also included in the SDK because without the API, applications can't relay information and work together.

Steps for creating WebRTC

Step-1: Setup app on agora.io and get app credentials like Token, App ID and Channel name.

Go to agora.io and create an account (https://agora.io/). Agora gives us 10000 free minutes to use each month.

After login, go to project console dashboard and select the projects tab. Create a new project by filling the basic information needed.

For authentication select "Secured mode: APP ID and Token".

Once your project is created go back to project tab and click on configure. Get your APP ID and Temporary token here by typing your channel name (it can be anything).

The temporary token generated will last only for 24 hours, generate a new token when it gets expired or else there will be error on the code.

In a production environment we will want to generate this token dynamically so users of our app can generate their own channel names to host a call.

Download the agora SDK (https://docs.agora.io/en/All/download...) from the agora dashboard. For the platform choose "Web SDK" then choose the "Video SDK". This will download a zip file to your computer, extract the file we need "AgoraRTC_N- 4.18.2.js" and place it directly into our project. This file should be directly pointed to in the index.html file.

Step-2: Creating a folder containing HTML, CSS and JavaScript along with agora SDK.

Once our credential variables are set in main.js, we use these values to create a client object. The client object is an interface for providing the local client with basic functions for voice and video calls joining a channel, publishing our tracks or subscribing to other tracks.

Pass in the required properties "mode: rtc" which specifies the optimization algorithm that will be used and "codec" which is the codec of the web browser will use for encoding.

Below the client object set some values to represent local video and audio tracks along with the values to hold the remote user's video or audio tracks.

Local tracks will store user's video and audio track in a list while all other users that join our stream will be called remote users and this will simply be an object.

Create a function to toggle local user to join a stream with our camera and audio track and make sure it is asynch function.

In the function call the join method from the client object, this method takes in all our app credentials and adds our local user to the channel while returning back a UID.

The local track variable is now a list which holds the audio track in index 0 and video track in index 1.

Step-3: Display remote users.

Step-4: Adding controls and styling to our website.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>Web Real-Time Communication WebRTC</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
</head>

<body>
    <button id="join-btn">Join Stream</button>
    <div id="stream-wrapper">
        <div id="video-streams"></div>
        <div id="stream-controls">
            <button id="leave-btn">Leave Stream</button>
            <button id="mic-btn">Mic On</button>
            <button id="camera-btn">Camera on</button>
        </div>
    </div>
</body>
<script src= “AgoraRTC_N-4.7.3.js” </script>
<script src='main.js'></script></html>
```

main.css

```
body{
    background:#0F2027;
    background:-webkit-linear-gradient(to right, #2C5364, #203A43, #0F2027) ; background:
    linear-gradient(to right, #2C5364, #203A43, #0F2027);
}
```

```
#join-btn{
    position:fixed;
    top:50%;
    left:50%;
    margin-top:-50px;
    margin-left:-100px;
    font-size:18px;
    padding:20px 40px;
}

#video-streams
{ display:grid;
    grid-template-columns: repeat(auto-fit, minmax(500px, 1fr)); height:
    90vh;
    width: 1400px;
    margin:0 auto;
}

.video-container
{
     max-height: 100%;
    border: 2px solid black;
    background-color: #203A49;
}
.video-player
{ height: 100%;
    width: 100%;
}


button{
    border:none;
    background-color: cadetblue;
    color:#fff;
    padding:10px 20px;
    font-size:16px;
    margin:2px;
    cursor: pointer;
}
```

```
#stream-controls{
    display: none;
    justify-content: center;
    margin-top:0.5em;
}

@media screen and (max-width:1400px){
    #video-streams{
        grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); width:
        95%;
    }
}
```

main.js

```
const APP_ID = “YOUR APP ID”
const TOKEN = “YOUR TEMP TOKEN”
const CHANNEL = “YOUR CHANNEL NAME”
const client = AgoraRTC.createClient({mode:'rtc', codec:'vp8'})

let localTracks = [] let
remoteUsers = {}

let   joinAndDisplayLocalStream   =   async   ()   =>   {
    client.on('user-published',              handleUserJoined)
    client.on('user-left', handleUserLeft)
    let UID = await client.join(APP_ID, CHANNEL, TOKEN, null) localTracks = await
    AgoraRTC.createMicrophoneAndCameraTracks()

    let player = `<div class="video-container" id="user-container-${UID}">
                    <div class="video-player" id="user-${UID}"></div>
                </div>`
    document.getElementById('video-streams').insertAdjacentHTML('beforeend', player)
    localTracks[1].play(`user-${UID}`)
    await client.publish([localTracks[0], localTracks[1]])
}

let joinStream = async () => {
    await joinAndDisplayLocalStream()
    document.getElementById('join-btn').style.display = 'none'
    document.getElementById('stream-controls').style.display = 'flex'
```

```
}

let handleUserJoined = async (user, mediaType) => {
    remoteUsers[user.uid] = user
    await client.subscribe(user, mediaType)
    if (mediaType === 'video'){
        let player = document.getElementById(`user-container-${user.uid}`) if (player !=
        null){
            player.remove()
        }

        player = `<div class="video-container" id="user-container-${user.uid}">
                        <div class="video-player" id="user-${user.uid}"></div>
                  </div>`
        document.getElementById('video-streams').insertAdjacentHTML('beforeend', player)
        user.videoTrack.play(`user-${user.uid}`)
    }

    if (mediaType === 'audio'){
        user.audioTrack.play()
    }
}

let handleUserLeft = async (user) => { delete
    remoteUsers[user.uid]
    document.getElementById(`user-container-${user.uid}`).remove()
}
let leaveAndRemoveLocalStream = async () => { for(let i
    = 0; localTracks.length > i; i++){
        localTracks[i].stop()
        localTracks[i].close()
    }

    await client.leave()
    document.getElementById('join-btn').style.display = 'block'
    document.getElementById('stream-controls').style.display = 'none'
    document.getElementById('video-streams').innerHTML = ' '
}

let toggleMic = async (e) => { if
    (localTracks[0].muted){
        await localTracks[0].setMuted(false)
        e.target.innerText = 'Mic on'
        e.target.style.backgroundColor = 'cadetblue'
    }else{
        await localTracks[0].setMuted(true)
        e.target.innerText = 'Mic off'
        e.target.style.backgroundColor = '#EE4B2B'
    }
}

let toggleCamera = async (e) => {
    if(localTracks[1].muted){
        await localTracks[1].setMuted(false)
        e.target.innerText = 'Camera on'
        e.target.style.backgroundColor = 'cadetblue'
    }else{
        await localTracks[1].setMuted(true)
        e.target.innerText = 'Camera off'
        e.target.style.backgroundColor = '#EE4B2B'
    }
}

document.getElementById('join-btn').addEventListener('click', joinStream)
document.getElementById('leave-btn').addEventListener('click',
leaveAndRemoveLocalStream)
document.getElementById('mic-btn').addEventListener('click', toggleMic)
document.getElementById('camera-btn').addEventListener('click', toggleCamera)
```

7a) Write a HTML page that uses JavaScript to display a message when the mouse button is pressed and hide it once it is lifted.

<u>Prgm7a.html</u>

```
<!DOCTYPE html>
<html>
<head>
    <title>Mouse Button Event</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
        }

        h1 {
            text-align: center;
                color: #333;
            }

        p {
          margin-bottom: 10px;
        }

        .container {
            max-width: 400px; margin:
            0 auto; padding: 20px;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }

        #message {
             display: none;
            text-align:       center;
            margin-top:       20px;
            padding: 10px;
            background-color: #ffd700;
            color: #333;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Mouse Button Event</h1>
        <p>Press and hold the mouse button to show the message.</p>
        <p>Release the mouse button to hide the message.</p>

        <div id="message">Mouse button is pressed!</div>
    </div>

    <script>
        var message = document.getElementById('message');
        document.addEventListener('mousedown', function() {
            message.style.display = 'block';
        });
        document.addEventListener('mouseup', function() {
            message.style.display = 'none';
        });
    </script>
</body>
</html>
```

7b) Write a JavaScript code that displays text "text-growing" with increasing font size in the interval of 100ms in red colour, when the font size reaches 50pt it displays "text-shrinking" in blue colour, then the font size decreases to 5pt.

<u>Prgm7b.html</u>

```
<!DOCTYPE HTML>
<html> <head><style>
p {position: absolute;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
 }</style>  </head>

<body>
<p id="demo"></p>

<script>
var var1 = setInterval(inTimer, 1000);
var fs = 5;
var ids = document.getElementById("demo");

function inTimer()
{  ids.innerHTML = 'TEXT GROWING';
ids.setAttribute('style', "font-size: " + fs+ "px; color: red");    fs += 5;

 if(fs >= 50 )       {   clearInterval(var1);
                          var2 = setInterval(deTimer, 1000);  }
}
 function deTimer()
 {  fs -= 5;
ids.innerHTML = 'TEXT SHRINKING';
ids.setAttribute('style', "font-size: " + fs + "px; color: blue");
if(fs === 5 )   {  clearInterval(var2);  }
}
</script>
</body> </html>
```

8. Write a program using PHP and HTML to create a form and display the details entered by the user.

What is PHP?

PHP (Hypertext Preprocessor) is a widely used server-side scripting language that is designed for web development. It is embedded in HTML and executed on the server, generating dynamic content for web pages. PHP is a versatile language with many more features for handling databases, file manipulation, error handling, and more. It is widely used in conjunction with web servers like Apache and databases like MySQL for building dynamic web applications.

Here are some basics of PHP

- **Syntax**: PHP code is enclosed within **<?php ... ?>** tags. For example:

```
<?php
        // PHP code goes here
?>
```

- **Variables**: PHP variables start with a dollar sign **$** followed by the variable name. PHP is loosely typed, so you don't need to declare the variable type explicitly.

```
$name = "John";
$age = 25;
```

- **Data Types**: PHP supports various data types, including strings, integers, floats, booleans, arrays, objects and more.

- **Comments**: PHP comments can be added using // for single-line comments and **/* ... */** for multi-line comments.

- **Output**: You can display output using **echo** or **print** functions.

```
echo "Hello, World!";
```

- **Conditional Statements**: PHP supports if-else and switch statements for conditional branching.

```
if ($age >= 18) {
   echo "You are an adult.";
} else {
   echo "You are a minor.";
}
```

- **Loops**: PHP provides various types of loops, such as **for**, **while**, and **foreach**.

```
 for ($i = 1; $i <= 5; $i++)
 {   echo $i;   }
```

- **Functions**: You can define your custom functions in PHP. function

```
greet($name) {
    echo "Hello, " . $name . "!";
}
greet("Alice"); // Output: Hello, Alice!
```

- **Arrays**: PHP supports both indexed and associative arrays.
  ```
  // Indexed Array
  $fruits = array("Apple", "Banana", "Orange");

  // Associative Array
  $person = array("name" => "John", "age" => 25);
  ```

- **Super Global Variables**: PHP provides super global variables like **$_GET**, **$_POST**, **$_SESSION**, **$_COOKIE**, **$_FILES**, etc., to access data from different sources.

- **Include and Require**: PHP allows you to include external files using **include** or **require** statements.
  ```
  // Include a file (non-fatal error if file not found) include
  "header.php";
  // Require a file (fatal error if file not found) require
  "config.php";
  ```

- **Form Handling**: PHP can be used to process form data submitted from HTML forms.

<u>What is XAMPP?</u>

XAMPP is a free and open-source software package that provides a local server environment for web development and testing.
The name "XAMPP" stands for:

- **X**: Cross-platform (available for different operating systems)
- **A**: Apache HTTP Server
- **M**: MariaDB (formerly MySQL) database
- **P**: PHP
- **P**: Perl

It was originally created by Apache Friends and is now maintained by the Apache Friends community.
XAMPP bundles together all the essential components required to set up a web server environment, making it easy for developers to create and test web applications on their local machines. It is available for Windows, macOS, Linux, and other operating systems.

The core components of XAMPP are:

1. **Apache HTTP Server**: A popular web server software that handles HTTP requests from clients

(browsers) and serves web pages in response.

2. **MariaDB (MySQL) Database**: A relational database management system used for storing and managing data, commonly used with web applications.
3. **PHP**: The server-side scripting language used for creating dynamic web pages and interacting with databases.
4. **Perl**: A programming language mainly used for server-side scripting and command- line processing.

In addition to these core components, XAMPP also includes other useful tools and libraries like phpMyAdmin (for managing databases through a web interface), FileZilla FTP server (for handling file transfers), and more. Using XAMPP, developers can set up a local web server environment quickly without the need for individual installation and configuration of each component. This local server environment mimics the functionalities of a live web server, enabling developers to work on their web projects locally before deploying them to a remote server for public access. It is widely used for web development, testing and learning purposes.

XAMPP installation

To install XAMPP on Windows, follow these steps:

1. **Download XAMPP**: Visit the official Apache Friends website (https://www.apachefriends.org/download.html) and download the latest version of XAMPP for Windows.
2. **Run the Installer**: Once the download is complete, locate the installer file and double- click on it to run it. You may see a security warning; click "Yes" or "Run" to proceed.
3. **Select Components**: The installer will prompt you to select which components you want to install. These components typically include Apache, MySQL, PHP, and phpMyAdmin. You can keep the default selection or customize it based on your requirements. Then, click "Next" to proceed.
4. **Choose Installation Folder**: Choose the folder where you want to install XAMPP. The default location is usually "C:\xampp," but you can change it if needed. Click "Next."
5. **Start Menu Folder**: You can choose whether to create a shortcut in the Start Menu for XAMPP. This is optional, so choose according to your preference. Click "Next."
6. **Bitnami for XAMPP (Optional)**: The installer might ask if you want to install Bitnami for XAMPP, which provides additional software applications like WordPress, Joomla, etc. You can decide whether to install this or not. Click "Next."
7. **Ready to Install**: Review the installation settings and click "Next" to begin the installation process.
8. **Wait for Installation**: The installer will now proceed to install XAMPP along with the selected components. This may take a few minutes.

9. **Firewall Warning**: During the installation, your firewall might prompt you to allow Apache HTTP Server to access the network. Choose to allow access, as it's required for XAMPP to work properly.
10. **Installation Complete**: Once the installation is finished, you'll see a confirmation message. Make sure the "Start the control panel now" option is checked, and click "Finish."
11. **Start XAMPP Control Panel**: The XAMPP Control Panel will open. From here, you can start or stop the Apache and MySQL services. To start using XAMPP, click the "Start" buttons next to Apache and MySQL.
12. **Testing XAMPP**: To verify that XAMPP is running correctly, open your web browser and type "http://localhost" (without quotes) in the address bar. If everything is set up correctly, you should see the XAMPP dashboard.
13. **Using XAMPP**: Now that XAMPP is installed and running, you can place your web files in the "htdocs" folder located inside the XAMPP installation directory. For example, "C:\xampp\htdocs". This is where you can develop and test your web applications locally.

Remember to keep XAMPP updated and secure, especially if you plan to use it on a public network. Additionally, avoid using XAMPP as a production server, as it is primarily designed for development and testing purposes. For production, consider using a dedicated web hosting service.

form.php

```
<!DOCTYPE html>
<html>

<head>
    <title>User Details Form</title>
</head>

<body>
    <h2>User Details Form</h2>

    <form method="post" action="display.php">
        <label for="name">Name:</label>
        <input type="text" name="name" required><br><br>

        <label for="email">Email:</label>
        <input type="email" name="email" required><br><br>

        <label for="age">Age:</label>
        <input type="number" name="age" required><br><br>

        <label for="gender">Gender:</label>
        <input type="radio" name="gender" value="Male" required> Male
        <input type="radio" name="gender" value="Female" required> Female<br><br>
        <br>

        <input type="submit" value="Submit">
    </form>

</body>
</html>
```

display.php

```
<!DOCTYPE html>
<html>

<head>
   <title>User Details</title>
</head>

<body>
   <h2>Submitted User Details</h2>
   <?php
   if ($_SERVER["REQUEST_METHOD"] === "POST")
   {
       $name = $_POST["name"];
       $email = $_POST["email"];
       $age = $_POST["age"];
       $gender = $_POST["gender"];

       echo "Name: " . $name . "<br><br>"; echo
       "Email: " . $email . "<br><br>"; echo "Age:
       " . $age . "<br><br>";
       echo "Gender: " . $gender . "<br><br>";
   }
   ?>
 </body>
 </html>
```

9a) Write a PHP program to display a chess board.

chessboard.php

```
<!DOCTYPE html>

<html>

<head>
    <title>Chessboard</title>
    <style>
        .chessboard { width:
            240px; height:
            240px; display:
            grid;
            grid-template-columns: repeat(8, 30px);
            grid-template-rows: repeat(8, 30px);
        }

        .cell {
            width:      30px;
            height: 30px;
            text-align:       center;
            line-height:       30px;
            font-weight: bold;
        }

        .white {
            background-color: #f0d9b5;
        }

        .black {
            background-color: #b58863;
            color: white;
        }
    </style>
</head>

<body>
    <h2>Chessboard</h2>
    <div class="chessboard">
        <?php
        $isWhite = true;
        for ($row = 1; $row <= 8; $row++) { for
            ($col = 1; $col <= 8; $col++) {
                $cellClass = $isWhite ? "white" : "black";
                echo "<div class='cell $cellClass'>$row, $col</div>";
                $isWhite = !$isWhite;
            }
            $isWhite = !$isWhite;
        }
        ?>
    </div>
</body>

</html>
```

This program creates an 8x8 chessboard using an HTML grid layout. It uses PHP to loop through the rows and columns and assigns appropriate classes for white and black cells. The CSS styles define the appearance of the chessboard with alternating colours for the cells.

9b) Write a PHP program to store page views count in SESSION, to increment the count on each refresh, and to show the count on web page.

session.php

```
<?php
session_start();
// Check if the 'views' session variable exists if
(!isset($_SESSION['views'])) {
    $_SESSION['views'] = 1; // Initialize the 'views' session variable to 1 if it doesn't exist
} else {
    $_SESSION['views']++; // Increment the 'views' session variable if it exists
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Page Views Counter</title>
</head>
<body>
    <h2>Page Views Counter</h2>
    <p>This page has been viewed <?php echo $_SESSION['views']; ?> times.</p>
    <p>Refresh the page to see the count increase.</p>
</body>
</html>
```

- In this program, we start the session using **session_start()** at the beginning. The page views count is stored in the **$_SESSION['views']** variable. If the session variable 'views' doesn't exist (first visit), it is initialized to 1. On subsequent visits, the count is incremented by 1.
- The page will display the current page views count and on each refresh, the count will increase. The count will persist as long as the session is active, typically until the browser is closed.

9c) Write a PHP program to store current date-time in a COOKIE and display the 'Last visited on' date-time on the web page upon reopening of the same page.

cookie.php

```
<?php
// Set the cookie with the current date-time
$expiration = time() + (60 * 60 * 24 * 30); // Cookie will expire in 30 days if
(!isset($_COOKIE['last_visited'])) {
    setcookie('last_visited', date('Y-m-d H:i:s'), $expiration);
}

// Get the last visited date-time from the cookie
$lastVisitedDateTime = $_COOKIE['last_visited'] ?? 'Unknown';
?>

<!DOCTYPE html>
<html>

<head>
    <title>Last Visited Date-Time</title>
</head>

<body>
    <h2>Last Visited Date-Time</h2>
    <p>
        <?php
        if ($lastVisitedDateTime !== 'Unknown') {
            echo 'Last visited on: ' . $lastVisitedDateTime;
        } else {
```

```
        echo 'Welcome! This is your first visit.';
      }
      ?>
    </p>
  </body>

  </html>
```

- In this program, we use the **setcookie()** function to store the current date-time in a cookie named 'last_visited'. We also set an expiration time for the cookie (in this case, 30 days from the current time).
- On subsequent visits to the page, the program retrieves the 'last_visited' cookie using **$_COOKIE['last_visited']**. If the cookie exists, it will display the 'Last visited on' date-time. If the cookie does not exist (i.e., the user is visiting the page for the first time or after clearing cookies), it will display a welcome message.

Note: Keep in mind that cookies are stored on the user's device and can be manipulated by the user. Therefore, they should not be used to store sensitive data or critical information. For more secure scenarios, consider using server-side session management.

10. Implement the web application with NoSQL Database

```
from mongoengine import connect

class Mongodb:

    def __init__(self, database, collection):
        self.connection = connect(host='localhost', port=27017)
        self.db = self.connection[database]
        self.collection = self.db[collection]

    def insert(self,doc):
        if doc:
            try:
                self.collection.insert_one(doc)
                return True
            except Exception as e:
                return False
        else:
            return False

    def update(self,filter, parameter):
        if filter and parameter:
            try:
                print(self.collection.update_one(filter,{"$set":parameter}))
                return True
            except Exception as e:
                return False
        else:
            return False

    def find_doc(self, filter):
        return self.collection.find(filter)

    def delete(self, filter):
        self.collection.delete_many(filter)

import datetime
class Todo:
```

```
def __init__(self, title, description, is_completed, due_date=None):
    self.title = title
    self.description = description
    self.is_completed = is_completed
    self.due_date = due_date if due_date else datetime.datetime.now()
    self.mongo_obj = Mongodb('SampleTodo', 'Todo')

def insert_task(self):
    doc = {
        "title": self.title,
        "description": self.description,
        "is_completed": self.is_completed,
        "self.due_date": self.due_date
    }
    self.mongo_obj.insert(doc)
    return True

def list_all_uncomplted_tasks(self):
    res =  self.mongo_obj.find_doc({"is_completed":0})
    for r in res:
        print(r)

def update_task(self, filter, parameter):
    self.mongo_obj.update(filter, parameter)
    return True

def delete_all_completed_task(self):
    self.mongo_obj.delete({"is_completed":1})
```