GEETA UNIVERSITY
PANIPAT
(Delhi-NCR Campus)

School of Computer Application
Session:- 2023-2024

Web Technologies Lab

(CS3242)

Lab Record File

Submitted To: -

Mr. Md Shafiqul Sir

Submitted By:-

2301301016

Manav

B.tech 2st Sem (2023-2027)

LIST OF EXPERIMENTS AS PER CURRICULUM

S.No.	LIST OF EXPERIMENT	EXPERIMENT NO.	PAGE NO.	DATE	SIGN.
1.	Book Review Blog, you have to create a simple blog page to display the contents of blog with pictures, you have to complete the required tasks.	**1**		08-02-2024	
2.	Html Table- basic table generation to get into the understanding, you have to complete the task as defined	**2**		15-02-2024	
3.	College Admission Form, create a basic admission form using html forms to capture the admission information from the user.	**3**		20-02-2024	
4.	Share Your Recipe, your task is to display the contents of the Food Recipe with various components like ingredients, preparations time.	**4**		22-02-2024	
5.	Chess Board Write a program in js that creates a chessboard that represent a grid using to separate lines. At each position of the grid there is either a " * " or a "#".	**5**		27-02-2024	
6.	Calculate Bill, You have been given 2 arrays of objects (menu and categories) and another json to have bill, from these various object we need to calculate the Bill and generate a JSON	**6**		05-03-2024	
7.	ToDo App – Basic Application, you have to build a basic todo app to manage the tasks and complete the user story as defined	**7**		07-03-2024	
8.	Discussion App - D1 Basic Discussion app to include questions and responses to all the questions, application will display questions on the left pane and their response on right pane on the click of question, till then displays the new question form to add new question	**8**		12-03-2024	
9.	Build Pomodoro Clock – Pomodoro is a time management technique developed in the 1980s which uses a timer to break down work into intervals, traditionally 25 minutes in length, separated by short breaks. In this assignment,	**9**		21-03-2024	

	you'll know how to create such a timer in the browser with JavaScript.				
10.	Web API, in this assignment, you need Build a basic app to fetch the records using API's with the help of AJAX. This assignment uses the GIThub api to access data using API.	**10**		26-03-2024	
11.	Build a basic app to compile coding questions using API's with the help of AJAX. In this assignment you need to create a compiler app using the codequotient api and follow the tasks.	**11**		02-04-2024	
12.	React functional Component, handle click event in a functional component to handle the change in variable effects ui.	**12**		04-04-2024	
13.	Stopwatch uses react class components and handles events.	**13**		23-04-2024	

EXPERIMENT NO. – 1

AIM: Book Review Blog, you have to create a simple blog page to display the contents of blog with pictures, you have to complete the required tasks.

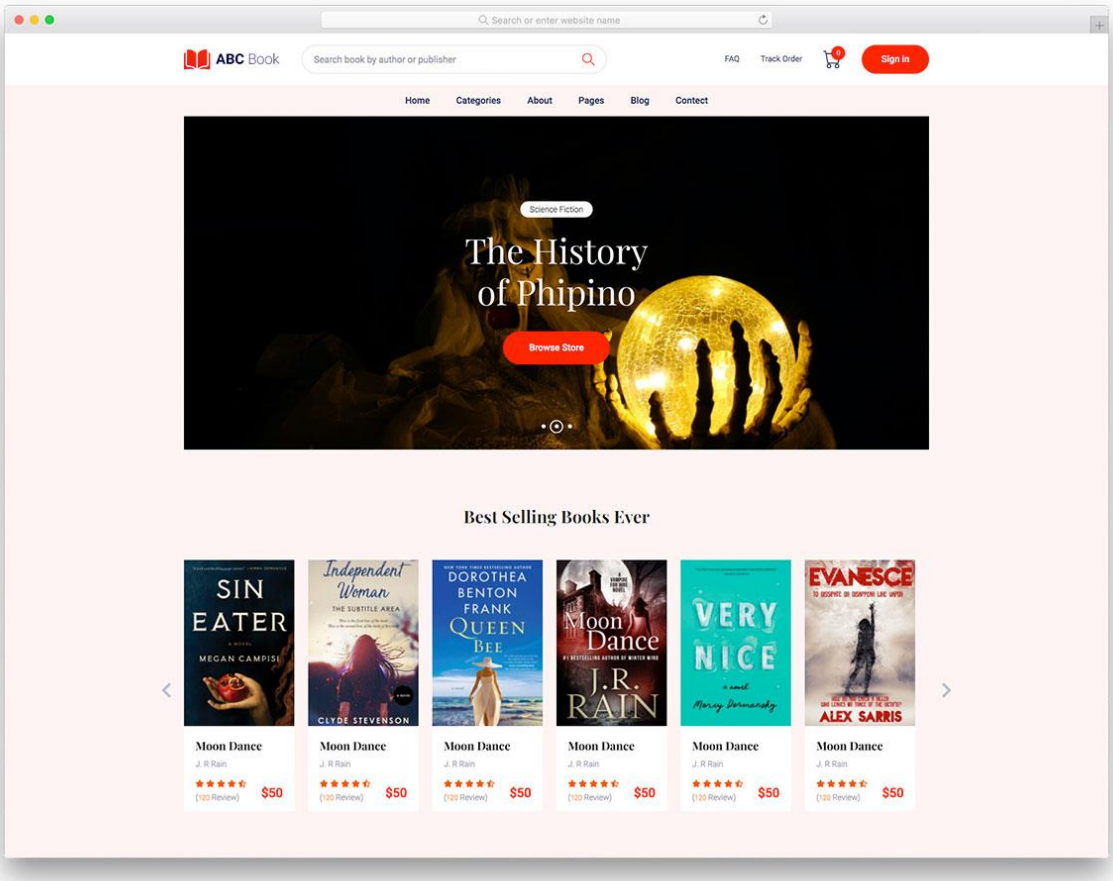
CODE :

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Book Review Blog</title>
```

```
<style>
  body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }
  header {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 20px 0;
  }
  .container {
    max-width: 800px;
    margin: 20px auto;
    padding: 0 20px;
  }
  .post {
    margin-bottom: 40px;
  }
  .post img {
    max-width: 100%;
    height: auto;
  }
  .post h2 {
    color: #333;
  }
  .post p {
    line-height: 1.6;
  }
</style>
```

```
</head>
<body>

<header>
  <h1>Book Review Blog</h1>
</header>

<div class="container">

  <div class="post">
    <img src="image1.jpg" alt="Book Cover">
    <h2>Book Title</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
  </div>

  <div class="post">
    <img src="image2.jpg" alt="Book Cover">
    <h2>Another Book Title</h2>
    <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
  </div>

  <!-- Add more posts as needed -->

</div>

</body>
</html>
```

OUTPUT:


Search or enter website name
ABC Book
Search book by author or publisher
FAQ
Track Order
Sign in
Home
Categories
About
Pages
Blog
Contect
Science Fiction
The History of Phipino
Browse Store
Best Selling Books Ever
SIN EATER
MEGAN CAMPISI
Independent Woman
THE SUBTITLE AREA
CLYDE STEVENSON
DOROTHEA BENTON FRANK
QUEEN BEE
Moon Dance
J.R. RAIN
VERY NICE
EVANESCE
ALEX SARRIS
Moon Dance
J. R Rain
(120 Review)
$50

EXPERIMENT NO. – 2

AIM: Html Table-basic table generation to get into the understanding, you have to complete the task as defined

CODE:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Basic HTML Table</title>

  <style>

    table {

      width: 100%;

      border-collapse: collapse;

    }

    th, td {

      border: 1px solid #ddd;

      padding: 8px;

      text-align: left;

    }

    th {

      background-color: #f2f2f2;

    }

  </style>

</head>

<body>


<h2>Basic HTML Table</h2>


<table>

 <tr>
```

```
    <th>Name</th>
    <th>Age</th>
    <th>City</th>
  </tr>
  <tr>
    <td>John Doe</td>
    <td>30</td>
    <td>New York</td>
  </tr>
  <tr>
    <td>Jane Smith</td>
    <td>25</td>
    <td>Los Angeles</td>
  </tr>
  <tr>
    <td>Michael Johnson</td>
    <td>40</td>
    <td>Chicago</td>
  </tr>
</table>

</body>
</html>
```

OUTPUT:

Basic HTML Table

Name	Age	City
John Doe	30	New York
Jane Smith	25	Los Angeles
Michael Johnson	40	Chicago

EXPERIMENT NO. – 3

AIM: College Admission Form, create a basic admission form using html forms to capture the admission information from the user.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>College Admission Form</title>
   <style>
      label {
         display: block;
         margin-bottom: 10px;
      }
      input[type="text"],
      input[type="email"],
      select {
         width: 100%;
         padding: 8px;
         margin-bottom: 15px;
         border: 1px solid #ccc;
         border-radius: 4px;
         box-sizing: border-box;
      }
      input[type="submit"] {
         background-color: #4CAF50;
         color: white;
         padding: 12px 20px;
         border: none;
         border-radius: 4px;
         cursor: pointer;
      }
      input[type="submit"]:hover {
         background-color: #45a049;
      }
   </style>
</head>
<body>

<h2>College Admission Form</h2>

<form action="/submit_admission" method="post">
   <label for="fullname">Full Name:</label>
   <input type="text" id="fullname" name="fullname" required>

   <label for="email">Email:</label>
```

```
    <input type="email" id="email" name="email" required>

    <label for="phone">Phone Number:</label>
    <input type="text" id="phone" name="phone" required>

    <label for="dob">Date of Birth:</label>
    <input type="date" id="dob" name="dob" required>

    <label for="address">Address:</label>
    <input type="text" id="address" name="address" required>

    <label for="program">Program of Interest:</label>
    <select id="program" name="program">
        <option value="computer_science">Computer Science</option>
        <option value="engineering">Engineering</option>
        <option value="business">Business</option>
        <option value="biology">Biology</option>
        <!-- Add more options as needed -->
    </select>

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

</body>
</html>
```

OUTPUT:

College Admission Form

Full Name:

Email:

Phone Number:

Date of Birth:

mm/dd/yyyy

Address:

Program of Interest:

Computer Science

Submit

EXPERIMENT NO. – 4

AIM: Share Your Recipe, your task is to display the contents of the Food Recipe with various components like ingredients, preparations time.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Delicious Recipe</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 800px;
            margin: 20px auto;
            padding: 0 20px;
        }
        h1, h2 {
            text-align: center;
        }
        .ingredient-list {
            margin-bottom: 20px;
        }
        .ingredient-list ul {
            list-style-type: none;
            padding: 0;
        }
        .ingredient-list ul li {
            margin-bottom: 5px;
        }
        .instructions {
            margin-bottom: 20px;
        }
    </style>
</head>
<body>

<div class="container">

    <h1>Delicious Recipe</h1>

    <h2>Ingredients</h2>
    <div class="ingredient-list">
```

```
        <ul>
            <li>Ingredient 1</li>
            <li>Ingredient 2</li>
            <li>Ingredient 3</li>
            <!-- Add more ingredients as needed -->
        </ul>
    </div>

    <h2>Preparation Time</h2>
    <p>30 minutes</p>

    <h2>Instructions</h2>
    <div class="instructions">
        <ol>
            <li>Step 1: Instructions for step 1.</li>
            <li>Step 2: Instructions for step 2.</li>
            <li>Step 3: Instructions for step 3.</li>
            <!-- Add more steps as needed -->
        </ol>
    </div>

</div>

</body>
</html>
```

OUTPUT:

Delicious Recipe

Ingredients

Ingredient 1
Ingredient 2
Ingredient 3

Preparation Time

30 minutes

Instructions

1. Step 1: Instructions for step 1.
2. Step 2: Instructions for step 2.
3. Step 3: Instructions for step 3.

EXPERIMENT NO. – 5

AIM: Chess Board Write a program in js that creates a chessboard that represent a grid using to separate lines. At each position of the grid there is either a " * " or a "#".

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Chessboard</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div id="chessboard"></div>
  <script src="script.js"></script>
</body>
</html>
```

CSS CODE:

```
#chessboard {
  display: flex;
  flex-wrap: wrap;
  width: 400px;
  height: 400px;
}

.square {
  width: 50px;
  height: 50px;
```

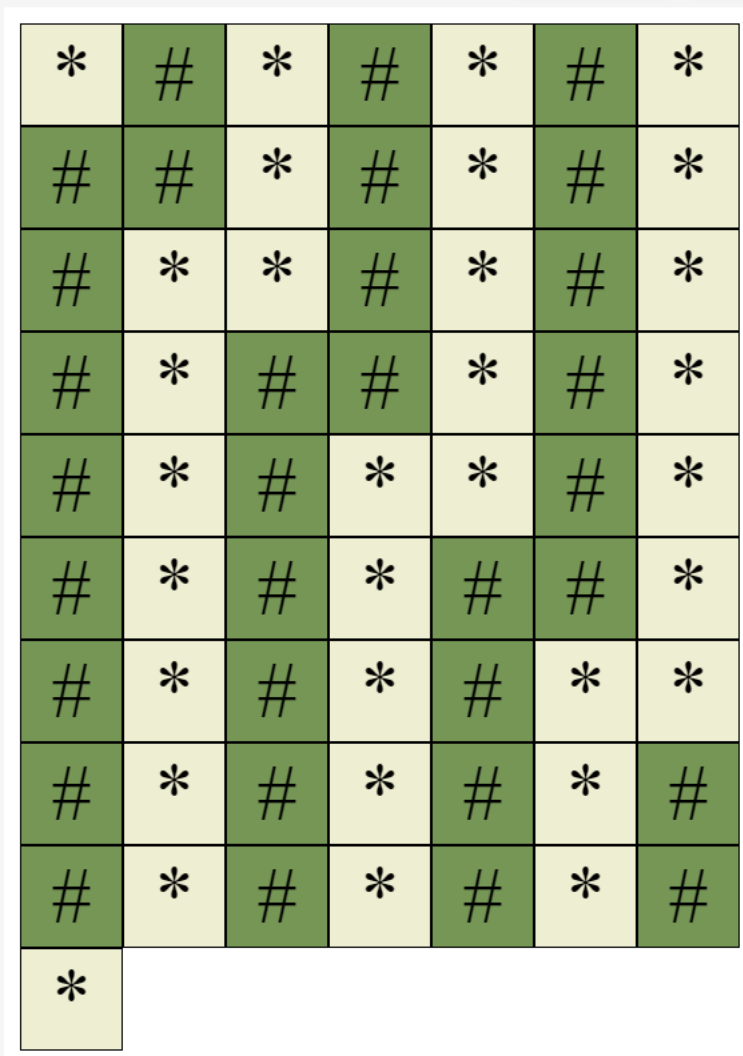
```
  border: 1px solid black;
}

.dark {
  background-color: #769656;
}

.light {
  background-color: #eeeed2;
}

.symbol {
  font-size: 40px;
  text-align: center;
  line-height: 50px;
}
```

JAVASCRIPT CODE:

```
const chessboard = document.getElementById('chessboard');

for (let row = 0; row < 8; row++) {
  for (let col = 0; col < 8; col++) {
    const square = document.createElement('div');
    square.classList.add('square');
    if ((row + col) % 2 === 0) {
      square.classList.add('light');
    } else {
      square.classList.add('dark');
    }
```

```
    const symbol = document.createElement('div');
    symbol.classList.add('symbol');
    if (row % 2 === 0) {
      symbol.textContent = (col % 2 === 0) ? '*' : '#';
    } else {
      symbol.textContent = (col % 2 === 0) ? '#' : '*';
    }

    square.appendChild(symbol);
    chessboard.appendChild(square);
  }
}
```

OUTPUT:



EXPERIMENT NO. – 6

AIM: Calculate Bill, You have been given 2 arrays of objects (menu and categories) and another json to have bill, from these various object we need to calculate the Bill and generate a JSON .

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Restaurant Bill</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div id="bill"></div>
  <script src="script.js"></script>
</body>
</html>
```

CSS CODE:

```
.bill {
  font-family: Arial, sans-serif;
  max-width: 400px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

h2 {
  text-align: center;
}

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

li {
  margin-bottom: 10px;
}

.item {
  display: inline-block;
  width: 60%;
}
```

```
.quantity {
  display: inline-block;
  width: 20%;
  text-align: right;
}

.subtotal {
  display: inline-block;
  width: 20%;
  text-align: right;
}

.total {
  font-weight: bold;
  text-align: right;
  margin-top: 20px;
}
```

JAVASCRIPT CODE:

```
// Sample menu and categories
const menu = [
  { id: 1, name: "Burger", category_id: 1, price: 5.99 },
  { id: 2, name: "Pizza", category_id: 1, price: 8.99 },
  { id: 3, name: "Salad", category_id: 2, price: 6.49 },
  { id: 4, name: "Pasta", category_id: 2, price: 7.99 }
];

const categories = [
  { id: 1, name: "Main Course" },
  { id: 2, name: "Appetizers" }
];

// Sample bill JSON
const bill = {
  items: [
    { id: 1, quantity: 2 },
    { id: 3, quantity: 1 }
  ]
};

// Function to calculate total bill
function calculateBill(menu, bill) {
  let total = 0;
  for (const item of bill.items) {
    const menuItem = menu.find(menuItem => menuItem.id === item.id);
    if (menuItem) {
      total += menuItem.price * item.quantity;
    }
  }
  return total;
```

```
}

// Calculate total bill
const totalBill = calculateBill(menu, bill);

// Generate JSON representation of bill
const billJSON = {
  total: totalBill.toFixed(2),
  items: bill.items.map(item => {
    const menuItem = menu.find(menuItem => menuItem.id === item.id);
    return {
      name: menuItem ? menuItem.name : "Unknown Item",
      quantity: item.quantity,
      subtotal: menuItem ? (menuItem.price * item.quantity).toFixed(2) : 0
    };
  })
};

// Generate HTML for displaying bill
const billHTML = `
<div class="bill">
  <h2>Bill Details</h2>
  <ul>
    ${billJSON.items.map(item => `
      <li>
        <span class="item">${item.name}</span>
        <span class="quantity">${item.quantity}</span>
        <span class="subtotal">$${item.subtotal}</span>
      </li>
    `).join('')}
  </ul>
  <div class="total">Total: $${billJSON.total}</div>
</div>
`;

// Display bill on the webpage
const billContainer = document.getElementById('bill');
billContainer.innerHTML = billHTML;
```

OUTPUT:

Bill Details

Burger $11.98	2
Salad $6.49	1

Total: $18.47

EXPERIMENT NO. – 7

AIM: ToDo App – Basic Application, you have to build a basic todo app to manage the tasks and complete the user story as defined .

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ToDo App</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>ToDo App</h1>
    <div class="input-container">
      <input type="text" id="taskInput" placeholder="Enter your task...">
      <button onclick="addTask()">Add Task</button>
    </div>
    <ul id="taskList"></ul>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

CSS CODE:

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

.container {
  max-width: 600px;
  margin: 0 auto;
  padding: 20px;
}

h1 {
  text-align: center;
}

.input-container {
  margin-bottom: 20px;
}

input[type="text"] {
  width: 70%;
```

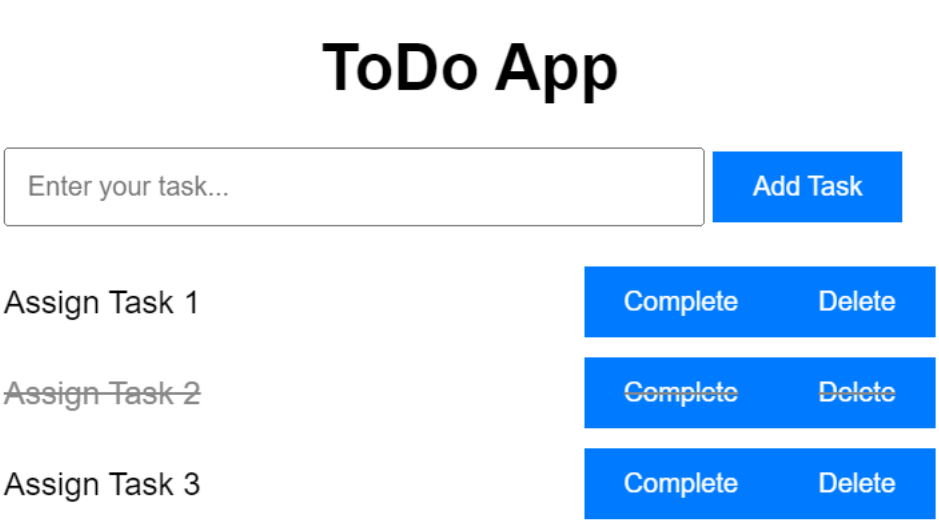
```
  padding: 10px;
}

button {
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  border: none;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

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

li {
  margin-bottom: 10px;
  display: flex;
  align-items: center;
}

li span {
  flex-grow: 1;
}

.completed {
  text-decoration: line-through;
  color: #888;
}
```

JAVASCRIPT CODE:

```
// Function to add a new task
function addTask() {
  const input = document.getElementById('taskInput');
  const taskText = input.value.trim();

  if (taskText === '') {
    alert('Please enter a task.');
    return;
  }

  const taskList = document.getElementById('taskList');
  const li = document.createElement('li');
```

```
  li.innerHTML = `
    <span>${taskText}</span>
    <button onclick="completeTask(this)">Complete</button>
    <button onclick="deleteTask(this)">Delete</button>
  `;
  taskList.appendChild(li);

  input.value = '';
}

// Function to complete a task
function completeTask(button) {
  const li = button.parentElement;
  li.classList.toggle('completed');
}

// Function to delete a task
function deleteTask(button) {
  const li = button.parentElement;
  li.remove();
}
```

OUTPUT:


ToDo App
Enter your task...
Add Task
Assign Task 1
Complete
Delete
Assign Task 2
Complete
Delete
Assign Task 3
Complete
Delete

EXPERIMENT NO. – 8

AIM: Discussion App - D1 Basic Discussion app to include questions and responses to all the questions, application will display questions on the left pane and their response on right pane on the click of question, till then displays the new question form to add new question.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Discussion App</title>
<style>
   body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
   }
   .container {
      display: flex;
      height: 100vh;
   }
   .left-pane, .right-pane {
      flex: 1;
      height: 100%;
      overflow-y: auto;
      border: 1px solid #ccc;
   }
   .question {
      cursor: pointer;
      padding: 10px;
      border-bottom: 1px solid #ccc;
   }
   .question:hover {
      background-color: #f0f0f0;
   }
   .new-question {
      padding: 10px;
   }
</style>
</head>
<body>

<div class="container">
   <div class="left-pane">
      <div class="questions">
         <!-- Questions will be displayed here -->
      </div>
```

```
        <div class="new-question">
            <h2>New Question</h2>
            <input type="text" id="questionInput" placeholder="Enter your question">
            <button onclick="addQuestion()">Submit</button>
        </div>
    </div>
    <div class="right-pane">
        <div class="responses">
            <!-- Responses to questions will be displayed here -->
        </div>
    </div>
</div>

<script>
    // Sample initial questions
    let questions = [
        { id: 1, text: "What is your favorite programming language?" },
        { id: 2, text: "What is your favorite book?" },
        { id: 3, text: "What is your favorite movie?" }
    ];

    // Sample initial responses
    let responses = {
        1: ["JavaScript", "Python", "Java"],
        2: ["To Kill a Mockingbird", "1984", "Harry Potter"],
        3: ["The Shawshank Redemption", "The Godfather", "Inception"]
    };

    // Function to display questions
    function displayQuestions() {
        const questionsDiv = document.querySelector('.questions');
        questionsDiv.innerHTML = '';
        questions.forEach(question => {
            const questionDiv = document.createElement('div');
            questionDiv.classList.add('question');
            questionDiv.textContent = question.text;
            questionDiv.addEventListener('click', () => {
                displayResponses(question.id);
            });
            questionsDiv.appendChild(questionDiv);
        });
    }

    // Function to display responses to a question
    function displayResponses(questionId) {
        const responsesDiv = document.querySelector('.responses');
        responsesDiv.innerHTML = '';
        const responseList = responses[questionId] || [];
        responseList.forEach(response => {
            const responseDiv = document.createElement('div');
```

```
            responseDiv.textContent = response;
            responsesDiv.appendChild(responseDiv);
        });
    }

    // Function to add a new question
    function addQuestion() {
        const questionInput = document.getElementById('questionInput');
        const newQuestionText = questionInput.value.trim();
        if (newQuestionText !== '') {
            const newQuestion = { id: questions.length + 1, text: newQuestionText };
            questions.push(newQuestion);
            // Add an empty array for responses to the new question
            responses[newQuestion.id] = [];
            displayQuestions();
            questionInput.value = '';
        }
    }

    // Initial display of questions
    displayQuestions();
</script>

</body>
</html>
```

OUTPUT:

What is your favorite programming language?

What is your favorite book?

What is your favorite movie?

Full form HTML

New Question

Enter your question

Submit

EXPERIMENT NO. – 9

AIM: Build Pomodoro Clock – Pomodoro is a time management technique developed in the 1980s which uses a timer to break down work into intervals, traditionally 25 minutes in length, separated by short breaks. In this assignment, you'll know how to create such a timer in the browser with JavaScript.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Pomodoro Clock</title>
<style>
   body {
      font-family: Arial, sans-serif;
      text-align: center;
   }
   #clock {
      font-size: 36px;
      margin-top: 50px;
   }
   #controls {
      margin-top: 20px;
   }
</style>
</head>
<body>

<div id="clock">25:00</div>
<div id="controls">
   <label for="workDuration">Work Duration (minutes):</label>
   <input type="number" id="workDuration" value="25">
   <label for="breakDuration">Break Duration (minutes):</label>
   <input type="number" id="breakDuration" value="5">
   <button id="startBtn">Start</button>
   <button id="pauseBtn" style="display:none">Pause</button>
   <button id="resetBtn">Reset</button>
</div>

<script>
   let workDuration = 25; // Default work duration
   let breakDuration = 5; // Default break duration
   let timerInterval;
   let timeLeft;
   let isPaused = false;
   let isWorking = true;
```

```
const clockDisplay = document.getElementById('clock');
const startBtn = document.getElementById('startBtn');
const pauseBtn = document.getElementById('pauseBtn');
const resetBtn = document.getElementById('resetBtn');
const workDurationInput = document.getElementById('workDuration');
const breakDurationInput = document.getElementById('breakDuration');

startBtn.addEventListener('click', startTimer);
pauseBtn.addEventListener('click', pauseTimer);
resetBtn.addEventListener('click', resetTimer);
workDurationInput.addEventListener('change', updateWorkDuration);
breakDurationInput.addEventListener('change', updateBreakDuration);

function startTimer() {
    if (!isPaused) {
        timeLeft = isWorking ? workDuration * 60 : breakDuration * 60;
    }
    timerInterval = setInterval(updateTimer, 1000);
    startBtn.style.display = 'none';
    pauseBtn.style.display = 'inline-block';
}

function pauseTimer() {
    clearInterval(timerInterval);
    isPaused = true;
    startBtn.textContent = 'Resume';
    startBtn.style.display = 'inline-block';
    pauseBtn.style.display = 'none';
}

function resetTimer() {
    clearInterval(timerInterval);
    isPaused = false;
    isWorking = true;
    startBtn.textContent = 'Start';
    startBtn.style.display = 'inline-block';
    pauseBtn.style.display = 'none';
    updateDisplay(workDuration);
}

function updateTimer() {
    timeLeft--;
    if (timeLeft <= 0) {
        if (isWorking) {
            isWorking = false;
            timeLeft = breakDuration * 60;
        } else {
            isWorking = true;
            timeLeft = workDuration * 60;
        }
```

```
        }
        updateDisplay(timeLeft);
    }

    function updateDisplay(seconds) {
        const minutes = Math.floor(seconds / 60);
        const remainderSeconds = seconds % 60;
        const display = `${minutes < 10 ? '0' : ''}${minutes}:${remainderSeconds < 10 ? '0' :
''}${remainderSeconds}`;
        clockDisplay.textContent = display;
    }

    function updateWorkDuration() {
        workDuration = parseInt(workDurationInput.value);
        updateDisplay(workDuration * 60);
    }

    function updateBreakDuration() {
        breakDuration = parseInt(breakDurationInput.value);
        if (!isWorking) {
            updateDisplay(breakDuration * 60);
        }
    }
</script>

</body>
</html>
```

OUTPUT:

EXPERIMENT NO. – 10

AIM: Web API, in this assignment, you need Build a basic app to fetch the records using API's with the help of AJAX. This assignment uses the GIThub api to access data using API.

CODE:

```
 <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>GitHub User Repositories</title>
<style>
   body {
      font-family: Arial, sans-serif;
      text-align: center;
   }
   #repositories {
      margin-top: 20px;
   }
</style>
</head>
<body>

<h1>GitHub User Repositories</h1>
<form id="searchForm">
   <label for="username">Enter GitHub Username:</label>
   <input type="text" id="username" required>
   <button type="submit">Search</button>
</form>

<div id="repositories"></div>

<script>
   const form = document.getElementById('searchForm');
   const repositoriesDiv = document.getElementById('repositories');

   form.addEventListener('submit', function(event) {
      event.preventDefault();
      const username = document.getElementById('username').value;
      if (username.trim() === '') {
         alert('Please enter a GitHub username.');
         return;
      }

      // Fetch user's repositories using GitHub API
      fetch(`https://api.github.com/users/${username}/repos`)
         .then(response => {
            if (!response.ok) {
```

```
                throw new Error('Network response was not ok');
            }
            return response.json();
        })
        .then(data => {
            displayRepositories(data);
        })
        .catch(error => {
            console.error('Error fetching data:', error);
            repositoriesDiv.innerHTML = '<p>Error fetching repositories. Please try again later.</p>';
        });
    });

    function displayRepositories(repositories) {
        repositoriesDiv.innerHTML = ''; // Clear previous results
        if (repositories.length === 0) {
            repositoriesDiv.innerHTML = '<p>No repositories found for this user.</p>';
            return;
        }

        const list = document.createElement('ul');
        repositories.forEach(repo => {
            const listItem = document.createElement('li');
            listItem.innerHTML = `<a href="${repo.html_url}" target="_blank">${repo.full_name}</a>`;
            list.appendChild(listItem);
        });
        repositoriesDiv.appendChild(list);
    }
</script>

</body>
</html>
```

OUTPUT:

GitHub User Repositories

Enter GitHub Username: shafikcmt Search

- shafikcmt/ajax-with-php
- shafikcmt/anwargen
- shafikcmt/autobloggingsystem
- shafikcmt/Awesome_project
- shafikcmt/Basic-Bootstrap
- shafikcmt/Basic-HTML-CSS-Project
- shafikcmt/Basic-Web-Template-With-HTML-and-CSS
- shafikcmt/BasicWebTemplate
- shafikcmt/BlogSite-With-PHP-OOP
- shafikcmt/Bootcamp
- shafikcmt/Bootcamp_Day3
- shafikcmt/Bootstrap-Fundamental
- shafikcmt/Bootstrap4thcse
- shafikcmt/bootstrapbtech3rd
- shafikcmt/btech2nd-template
- shafikcmt/btechcse3rd
- shafikcmt/bwt
- shafikcmt/codetechbd
- shafikcmt/CRUD-with-PHP-OOP-and-MySQLi
- shafikcmt/eCom-Laravel-Project
- shafikcmt/geeta
- shafikcmt/Image-Upload-System-PHP

EXPERIMENT NO. – 11

AIM: Build a basic app to compile coding questions using API's with the help of AJAX. In this assignment you need to create a compiler app using the codequotient api and follow the tasks.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Code Compiler</title>
<style>
   body {
      font-family: Arial, sans-serif;
      text-align: center;
   }
   #output {
      margin-top: 20px;
   }
</style>
</head>
<body>

<h1>Code Compiler</h1>
<form id="compilerForm">
   <label for="code">Enter Your Code:</label><br>
   <textarea id="code" rows="10" cols="50" required></textarea><br>
   <label for="language">Select Programming Language:</label>
   <select id="language" required>
      <option value="cpp">C++</option>
      <option value="java">Java</option>
      <option value="python">Python</option>
   </select><br>
   <button type="submit">Compile</button>
</form>

<div id="output"></div>

<script>
   const form = document.getElementById('compilerForm');
   const outputDiv = document.getElementById('output');

   form.addEventListener('submit', function(event) {
      event.preventDefault();
      const code = document.getElementById('code').value;
      const language = document.getElementById('language').value;
```

```
        // Send code and language to the CodeQuotient API for compilation
        fetch('https://codequotient.com/api/executeCode', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                code: code,
                lang: language
            })
        })
        .then(response => {
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            return response.json();
        })
        .then(data => {
            displayOutput(data);
        })
        .catch(error => {
            console.error('Error compiling code:', error);
            outputDiv.innerHTML = '<p>Error compiling code. Please try again later.</p>';
        });
    });

    function displayOutput(data) {
        outputDiv.innerHTML = ''; // Clear previous results
        if (data.errors) {
            outputDiv.innerHTML = `<p>Error: ${data.errors}</p>`;
        } else {
            outputDiv.innerHTML = `<p>Output:</p><pre>${data.output}</pre>`;
        }
    }
</script>

</body>
</html>
```

OUTPUT:

Code Compiler

Enter Your Code:

```
        // but if n2 is less than n3
        // but n2 is greater than n1
        // then n3 is the largest number
        else
            cout << "Largest number: " << n3;
    }

    return 0;
}
```

Select Programming Language: C++

Compile

Error compiling code. Please try again later.

EXPERIMENT NO. – 12

AIM: React functional Component, handle click event in a functional component to handle the change in variable effects ui.

CODE:

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

function ClickCounter() {
  // Define a state variable to hold the count
  const [count, setCount] = useState(0);

  // Function to handle click event
  const handleClick = () => {
    // Increment the count by 1 when the button is clicked
    setCount(count + 1);
  };

  return (
    <div>
      <h1>Click Counter</h1>
      <p>Count: {count}</p>
      <button onClick={handleClick}>Click Me</button>
    </div>
  );
}

export default ClickCounter;
```

EXPERIMENT NO. – 13

AIM: Stopwatch uses react class components and handles events.

CODE:

```
import React, { Component } from 'react';

class Stopwatch extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isRunning: false,
      elapsedTime: 0
    };
    this.timer = null;
  }

  handleStart = () => {
    if (!this.state.isRunning) {
      this.setState({ isRunning: true });
      this.timer = setInterval(() => {
        this.setState(prevState => ({
          elapsedTime: prevState.elapsedTime + 1
        }));
      }, 1000);
    }
  };

  handleStop = () => {
    clearInterval(this.timer);
    this.setState({ isRunning: false });
  };

  handleReset = () => {
    clearInterval(this.timer);
    this.setState({ isRunning: false, elapsedTime: 0 });
  };

  formatTime = (time) => {
    const hours = Math.floor(time / 3600);
    const minutes = Math.floor((time % 3600) / 60);
    const seconds = time % 60;
    return `${hours}:${minutes < 10 ? '0' : ''}${minutes}:${seconds < 10 ? '0' : ''}${seconds}`;
  };

  render() {
    const { isRunning, elapsedTime } = this.state;

    return (
```

```
      <div>
        <h1>Stopwatch</h1>
        <p>Time: {this.formatTime(elapsedTime)}</p>
        <button onClick={this.handleStart} disabled={isRunning}>Start</button>
        <button onClick={this.handleStop} disabled={!isRunning}>Stop</button>
        <button onClick={this.handleReset}>Reset</button>
      </div>
    );
  }
}

export default Stopwatch;
```