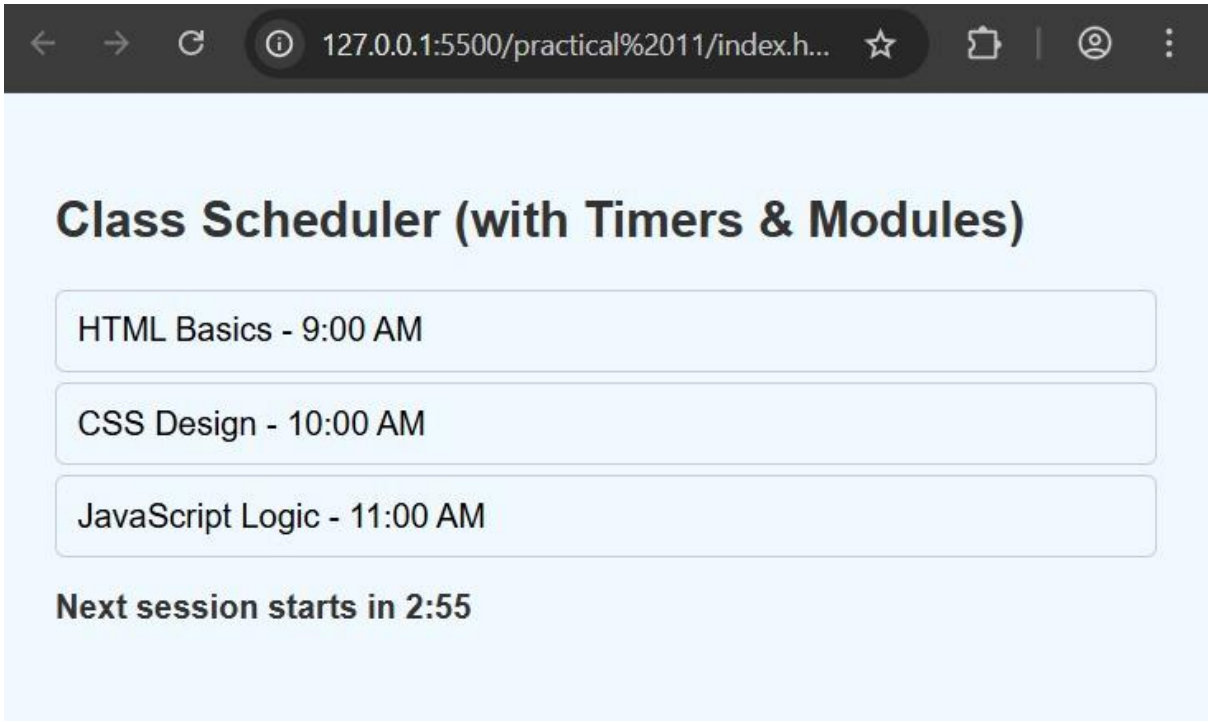
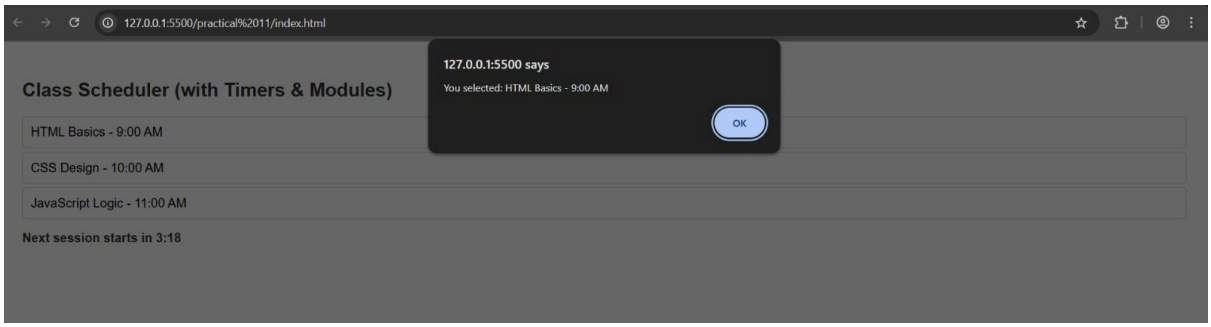
Name: Mukund Kuthe

3rd Year Section B (B1)

Practical 11 (JavaScript)



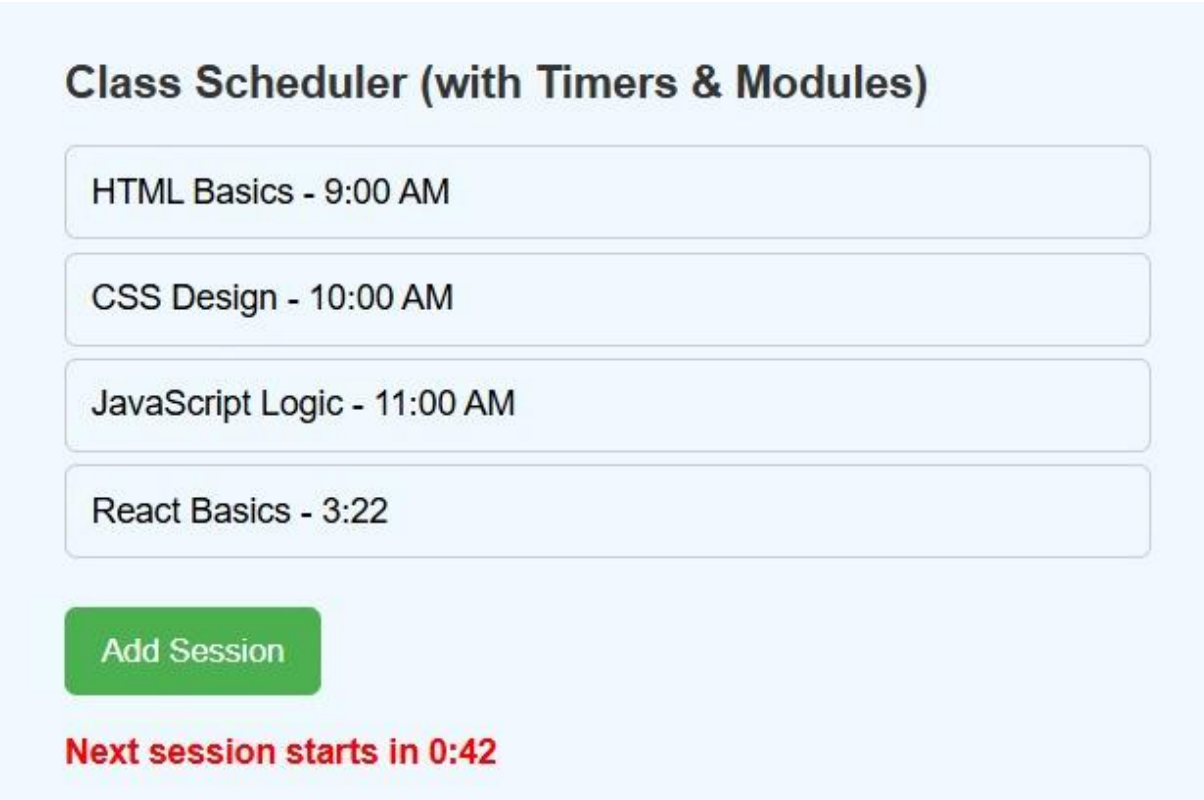

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Timers, Dynamic UI & Modular Design</title>

  <!-- Basic styling for the page -->
  <style>
    body { font-family: Arial; padding: 20px; }

    /* Each class session block */
    .session {
      padding: 10px;
      margin: 5px 0;
      border: 1px solid #ccc;
      border-radius: 5px;
    }

    /* Highlight active session */
    .active {
      background: #c8f7c5;
```

```
  }
 </style>
</head>
<body>
 <h2>Class Scheduler (with Timers & Modules)</h2>

 <!-- Container for all sessions -->
 <div id="sessions">
  <div class="session" data-time="09:00">HTML Basics - 9:00 AM</div>
  <div class="session" data-time="10:00">CSS Design - 10:00 AM</div>
  <div class="session" data-time="11:00">JavaScript Logic - 11:00 AM</div>
 </div>

 <!-- Display countdown timer -->
 <p id="countdown"></p>

 <!-- Main entry JavaScript file (module-based) -->
 <script type="module" src="main.js"></script>
</body>
</html>

 // timer.js — contains countdown and session highlight logic

 // Function to start a countdown timer
 export function startCountdown(duration, display) {
  let time = duration; // time in seconds

  // setInterval() runs the code every 1 second
  const timer = setInterval(() => {
   const minutes = Math.floor(time / 60); // convert seconds to minutes
   const seconds = time % 60;             // remaining seconds

   // Update countdown message on the page
```

```
    display.textContent = `Next session starts in ${minutes}:${seconds < 10 ? '0' + seconds
: seconds}`;

    time--; // decrease the timer each second

    // Stop timer when it reaches 0
    if (time < 0) {
      clearInterval(timer);
      display.textContent = "Session Started!";
    }
  }, 1000); // executes every 1000 milliseconds (1 second)
}


// Function to highlight the currently active session
export function highlightActiveSession() {
  const sessions = document.querySelectorAll('.session'); // get all session elements
  const now = new Date(); // get current date/time
  const currentHour = now.getHours(); // extract current hour (24-hour format)

  // Loop through all session divs
  sessions.forEach(session => {
    const hour = parseInt(session.dataset.time.split(':')[0]); // extract hour from data-time

    // If current hour matches session time → highlight
    if (hour === currentHour) {
      session.classList.add('active');
    } else {
      session.classList.remove('active');
    }
  });
```

```
 // Function to change background color based on screen size
 export function handleViewportChange() {
  // If screen width < 600px, apply a light blue background
  if (window.innerWidth < 600) {
   document.body.style.background = '#f0f8ff';
  } else {
   // For larger screens, use white background
   document.body.style.background = 'white';
  }
 }

// Function to add dynamic behavior (event delegation)
export function addDynamicSessions() {
 const sessionsDiv = document.getElementById('sessions');

 // Add event listener to parent container (event delegation)
 sessionsDiv.addEventListener('click', (e) => {
  // Check if clicked element has class "session"
  if (e.target.classList.contains('session')) {
   alert("You selected: " + e.target.textContent);
  }
 });
}
```

```
// Import functions from timer.js and ui.js modules
import { startCountdown, highlightActiveSession } from './timer.js';
import { handleViewportChange, addDynamicSessions } from './ui.js';

// Select the countdown display element from the DOM
const countdownDisplay = document.getElementById('countdown');

// Start countdown for next class (e.g., 5 minutes = 300 seconds)



 startCountdown(300, countdownDisplay);

 // Highlight the active session every minute automatically
 setInterval(highlightActiveSession, 60000); // recheck every 60 seconds
 highlightActiveSession(); // run once immediately on load

 // Handle screen resizing for responsive behavior
 window.addEventListener('resize', handleViewportChange);
 handleViewportChange(); // run on page load as well

 // Enable event delegation on session elements
 addDynamicSessions();
```




## Class Scheduler (with Timers & Modules)

- HTML Basics - 9:00 AM
- CSS Design - 10:00 AM
- JavaScript Logic - 11:00 AM
- React Basics - 3:22

Add Session

**Next session starts in 1:27**

## Class Scheduler (with Timers & Modules)

- HTML Basics - 9:00 AM
- CSS Design - 10:00 AM
- JavaScript Logic - 11:00 AM
- React Basics - 3:22

Add Session

**Next session starts in 0:58**

C: > Users > mukund kuthe > AppData > Local > Microsoft > Windows > INetCache > IE > JPSIXWFS > index[1].html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Timers, Dynamic UI & Modular Design</title>
  <!-- Basic styling for the page -->
  <style>
    body {
      font-family: Arial;
      padding: 20px;
      transition: background 0.3s ease;
    }

    h2 {
      color: #333;
    }

    /* Each class session block */
    .session {
      padding: 10px;
      margin: 5px 0;
      border: 1px solid #ccc;
      border-radius: 5px;
      cursor: pointer;
      transition: background 0.3s ease;
    }

    /* Highlight active session */
    .active {
      background: #c8f7c5;
      border-color: #8fd694;
    }

    /* Countdown display */
    #countdown {
      font-weight: bold;
      color: #333;
```

```
      #addSessionBtn:hover {
      }

      /* Responsive: change font size for small screens */
      @media (max-width: 500px) {
        body {
          font-size: 14px;
        }
        h2 {
          font-size: 18px;
        }
      }
    </style>
  </head>
  <body>
    <h2>Class Scheduler (with Timers & Modules)</h2>

    <!-- Container for all sessions -->
    <div id="sessions">
      <div class="session" data-time="09:00">HTML Basics - 9:00 AM</div>
      <div class="session" data-time="10:00">CSS Design - 10:00 AM</div>
      <div class="session" data-time="11:00">JavaScript Logic - 11:00 AM</div>
    </div>

    <!-- Add Session Button -->
    <button id="addSessionBtn">Add Session</button>

    <!-- Display countdown timer -->
    <p id="countdown"></p>

    <!-- Main entry JavaScript file (module-based) -->
    <script type="module" src="main.js"></script>
  </body>
  </html>
```

```
import { startCountdown, highlightActiveSession } from './timer.js';
import { handleViewportChange, addDynamicSessions, setupAddSessionButton } from './ui.js';

// Select the countdown display element
const countdownDisplay = document.getElementById('countdown');

// Start countdown for next class (1 minute 30 seconds = 90 seconds)
startCountdown(90, countdownDisplay);

// Highlight active session every minute
setInterval(highlightActiveSession, 60000);
highlightActiveSession();

// Handle viewport resize
window.addEventListener('resize', handleViewportChange);
handleViewportChange();

// Enable event delegation and session addition
addDynamicSessions();
setupAddSessionButton();

```

```
import { formatTime, logEvent } from './helper.js';

// Function to start a countdown timer
export function startCountdown(duration, display) {
  let time = duration; // time in seconds

  const timer = setInterval(() => {
    const minutes = Math.floor(time / 60);
    const seconds = time % 60;

    // Update countdown text
    display.textContent = `Next session starts in ${formatTime(minutes, seconds)}`;

    // Change color when less than 1 minute left
    if (time <= 60) {
      display.style.color = 'red';
    } else {
      display.style.color = '□#333';
    }

    time--;

    // Stop timer when it reaches 0
    if (time < 0) {
      clearInterval(timer);
      display.textContent = "Session Started!";
      display.style.color = 'green';
      logEvent("Countdown completed.");
    }
  }, 1000);
}

// Function to highlight the currently active session
export function highlightActiveSession() {
  const sessions = document.querySelectorAll('.session');
  const now = new Date();
```

```
38      const currentHour = now.getHours();
39
40      sessions.forEach(session => {
41        const hour = parseInt(session.dataset.time.split(':')[0]);
42        if (hour === currentHour) {
43          session.classList.add('active');
44        } else {
45          session.classList.remove('active');
46        }
47      });
48    }
49
```

```
export function formatTime(minutes, seconds) {
  return `${minutes}:${seconds < 10 ? '0' + seconds : seconds}`;
}

// Function to log important events to console (for debugging)
export function logEvent(message) {
  const timestamp = new Date().toLocaleTimeString();
  console.log(`[${timestamp}] ${message}`);
}
```

```
import { logEvent } from './helper.js';

// Function to change background color based on screen size
export function handleViewportChange() {
  if (window.innerWidth < 600) {
    document.body.style.background = '#f0f8ff';
  } else {
    document.body.style.background = 'white';
  }
}

// Function to add dynamic behavior (event delegation)
export function addDynamicSessions() {
  const sessionsDiv = document.getElementById('sessions');
  sessionsDiv.addEventListener('click', (e) => {
    if (e.target.classList.contains('session')) {
      alert("You selected: " + e.target.textContent);
      logEvent(`User clicked on: ${e.target.textContent}`);
    }
  });
}

// Function to dynamically add a new session
export function setupAddSessionButton() {
  const button = document.getElementById('addSessionBtn');
  const sessionsDiv = document.getElementById('sessions');

  button.addEventListener('click', () => {
    const newSessionTime = prompt("Enter session time (HH:MM):", "12:00");
    const newSessionName = prompt("Enter session name:", "React Basics");

    if (newSessionTime && newSessionName) {
      const div = document.createElement('div');
      div.classList.add('session');
      div.dataset.time = newSessionTime;
      div.textContent = `${newSessionName} - ${newSessionTime}`;
```

```
38            sessionsDiv.appendChild(div);
39            logEvent(`New session added: ${newSessionName} (${newSessionTime})`);
40        }
41    });
42  }
43
```