TASK 02

Stopwatch web Application

To build a stopwatch web application, you can use HTML, CSS, and JavaScript. HTML is used to structure the elements of the application. By implementing functions for starting, pausing, and resetting the stopwatch, as well as tracking and displaying lap times, users can accurately measure and record time intervals. With these technologies and functionalities, you can create an interactive and user-friendly stopwatch web application.

Let's delve into the coding aspect of building a stopwatch web application using HTML, CSS, and JavaScript:

1. Setting Up the Files:

Create three files:
index.html: This will be your main web page containing the HTML structure.
style.css: This will hold the styles for your application.
script.js: This will contain the JavaScript code to handle functionality.

2. Building the HTML Structure (index.html):

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Stopwatch</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Stopwatch</h1>
    <div id="display">00:00:00.000</div>
    <div id="buttons">
        <button id="start">Start</button>
        <button id="stop" disabled>Stop</button>
        <button id="reset" disabled>Reset</button>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

This code creates a basic structure with:

A heading for "Stopwatch"
A div with the ID "display" to show the stopwatch time.
Another div with the ID "buttons" containing three buttons for start, stop, and reset (initially disabled).
A link to the script.js file.

3. Styling the Application (style.css):

```css
body {
    font-family: sans-serif;
    text-align: center;
}

h1 {
    margin-bottom: 20px;
}

#display {
    font-size: 3em;
    margin-bottom: 20px;
}

#buttons {
    display: flex;
    justify-content: center;
}

button {
    padding: 10px 20px;
    margin: 0 10px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

#start {
    background-color: green;
    color: white;
}

#stop, #reset {
    background-color: lightgray;
}

#stop:hover, #reset:hover {
    background-color: gray;
}
```

This code styles the elements for a clean and user-friendly interface.

4. Implementing Functionality with JavaScript (script.js):

```
let startTime = 0;
let elapsedTime = 0;
let intervalId = null;
let isRunning = false;

const display = document.getElementById('display');
const startBtn = document.getElementById('start');
const stopBtn = document.getElementById('stop');
const resetBtn = document.getElementById('reset');

function updateTime() {
  elapsedTime = Date.now() - startTime;
  const milliseconds = Math.floor((elapsedTime / 1000) % 1000);
  const seconds = Math.floor((elapsedTime / 1000 / 60) % 60);
  const minutes = Math.floor(elapsedTime / (1000 * 60 * 60));

  display.innerHTML = `<span class="math-inline">\{minutes\.toString\
(\)\.padStart\(2, '0'\)\}\:</span>{seconds.toString().padStart(2, '0')}:$
{milliseconds.toString().padStart(3, '0')}`;
}

startBtn.addEventListener('click', () => {
  if (!isRunning) {
    isRunning = true;
    startTime = Date.now() - elapsedTime;
    intervalId = setInterval(updateTime, 10);
    stopBtn.disabled = false;
    resetBtn.disabled = false;
    startBtn.disabled = true;
  }
});

stopBtn.addEventListener('click', () => {
  if (isRunning) {
    isRunning = false;
    clearInterval(intervalId);
    stopBtn.disabled = true;
  }
});

resetBtn.addEventListener('click', () => {
  clearInterval(intervalId);
  startTime = 0;
  elapsedTime = 0;
  isRunning = false;
  display.innerHTML = '00:00:00.000';
  stopBtn.
```

Sources
info
medium.com/samsung-internet-dev/making-an-ar-game-with-aframe-529e03ae90cb
github.com/AislanCunha/cronometroProgressivo
github.com/evvoon/stopwatch
github.com/Boycey63/Rich-Web
stackoverflow.com/questions/64408465/taking-a-stab-at-a-javascript-timer-and-failing