

WEB TECHNOLOGY-LAB 9

ASSIGNMENT

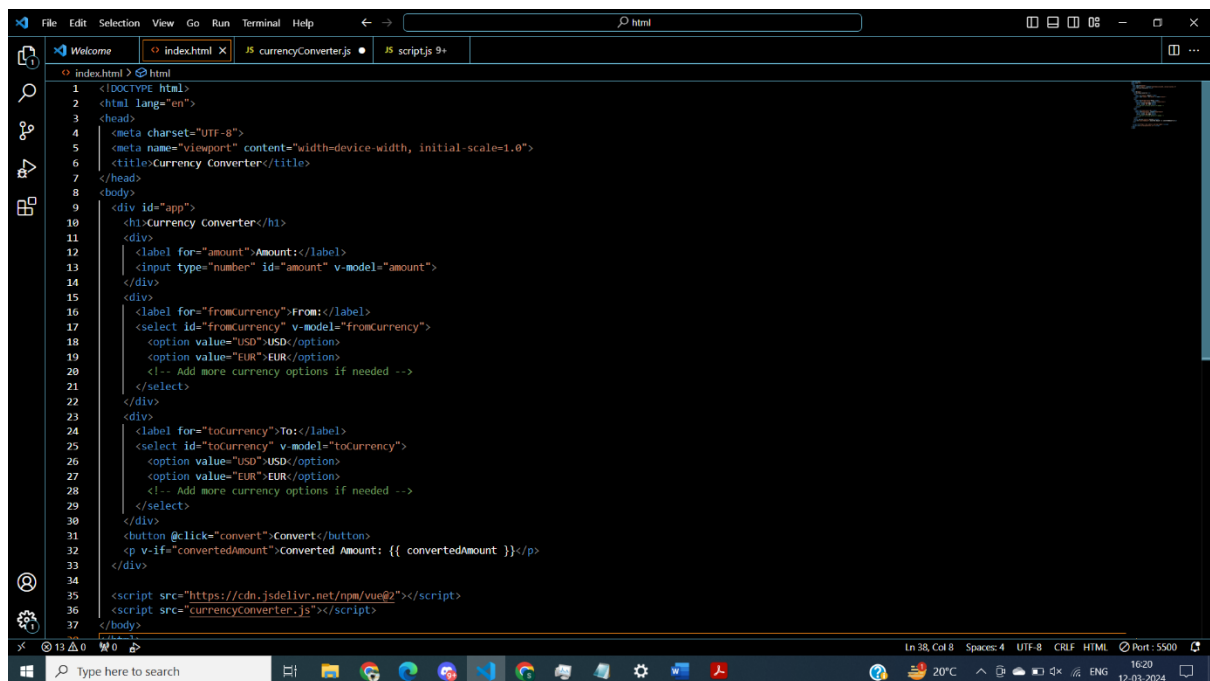
Bharath Gaddam

22Q12013

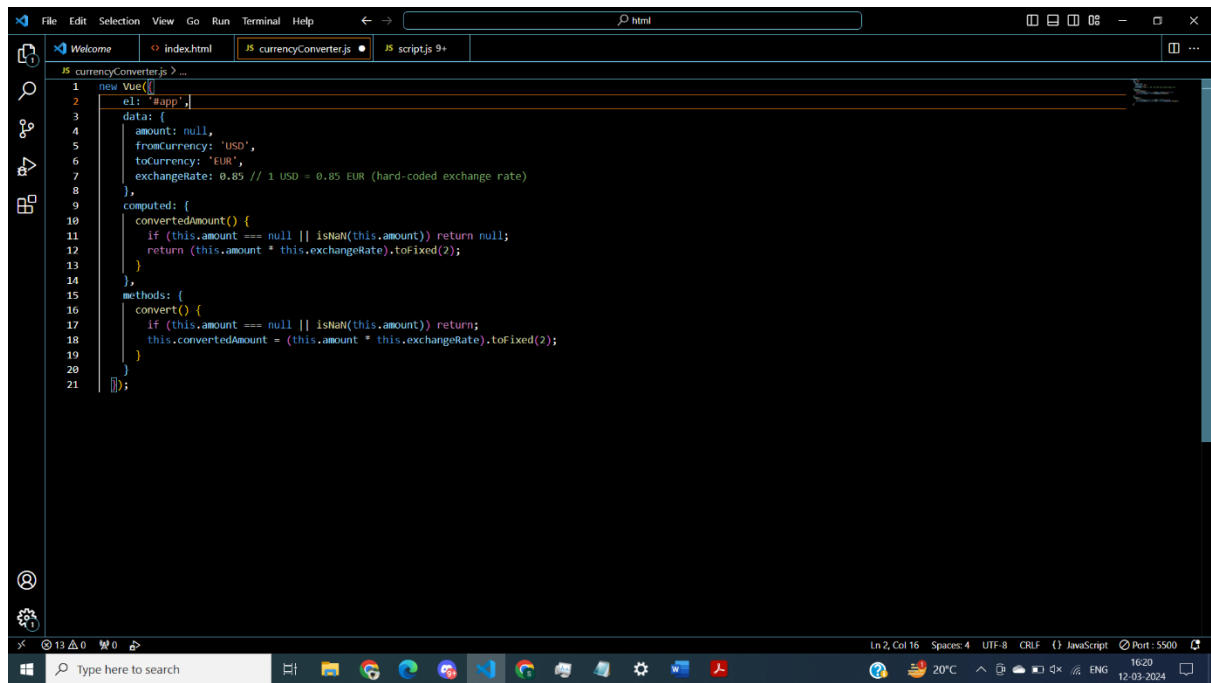
CSE-IDD

Q1 T1. Develop a currency converter application that allows users to input an amount in one currency and convert it to another. For the sake of this challenge, you can use a hard-coded exchange rate. Take advantage of React state and event handlers to manage the input and conversion calculations.

Note: Everything has to be done using Vue.js



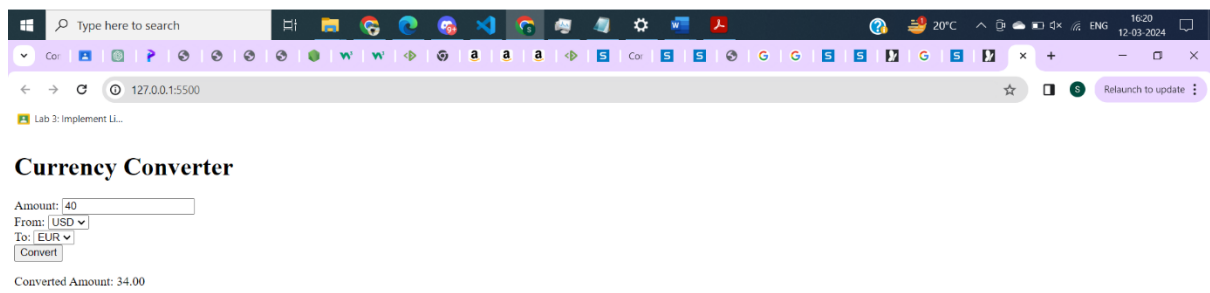
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Currency Converter</title>
7 </head>
8 <body>
9   <div id="app">
10    <h1>Currency Converter</h1>
11    <div>
12      <label for="amount">Amount:</label>
13      <input type="number" id="amount" v-model="amount">
14    </div>
15    <div>
16      <label for="fromCurrency">From:</label>
17      <select id="fromCurrency" v-model="fromCurrency">
18        <option value="USD">USD</option>
19        <option value="EUR">EUR</option>
20        <!-- Add more currency options if needed -->
21      </select>
22    </div>
23    <div>
24      <label for="toCurrency">To:</label>
25      <select id="toCurrency" v-model="toCurrency">
26        <option value="USD">USD</option>
27        <option value="EUR">EUR</option>
28        <!-- Add more currency options if needed -->
29      </select>
30    </div>
31    <button @click="convert">Convert</button>
32    <p v-if="convertedAmount">Converted Amount: {{ convertedAmount }}</p>
33  </div>
34  <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
35  <script src="currencyConverter.js"></script>
36 </body>
```



The image shows a Visual Studio Code editor window with a dark theme. The editor is open to a file named `currencyConverter.js`. The code is a Vue.js component definition. The `data` object has `amount` set to `null`, `fromCurrency` set to `'USD'`, `toCurrency` set to `'EUR'`, and `exchangeRate` set to `0.85` with a comment `// 1 USD = 0.85 EUR (hard-coded exchange rate)`. The `computed` object has a `convertedAmount` property with a getter function that checks if `amount` is null or NaN and returns the converted value. The `methods` object has a `convert` function that calls `convertedAmount` and updates the `convertedAmount` property.

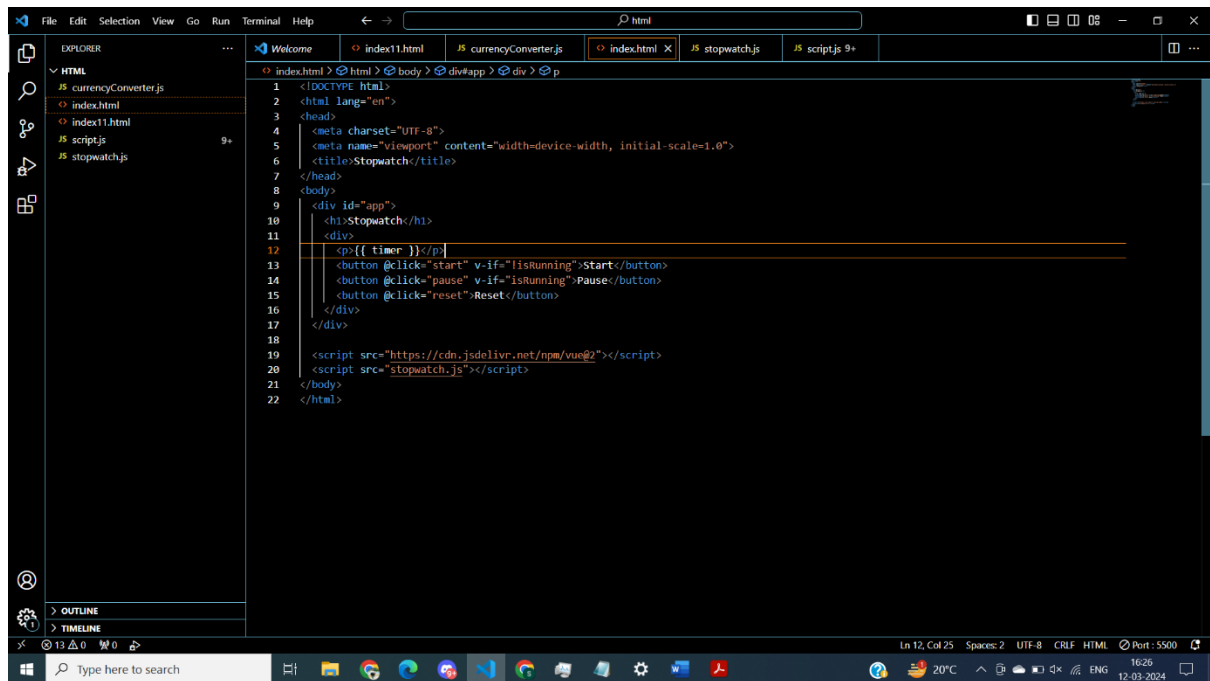
```
1 new Vue({
2   el: '#app',
3   data: {
4     amount: null,
5     fromCurrency: 'USD',
6     toCurrency: 'EUR',
7     exchangeRate: 0.85 // 1 USD = 0.85 EUR (hard-coded exchange rate)
8   },
9   computed: {
10    convertedAmount() {
11      if (this.amount === null || isNaN(this.amount)) return null;
12      return (this.amount * this.exchangeRate).toFixed(2);
13    }
14  },
15  methods: {
16    convert() {
17      if (this.amount === null || isNaN(this.amount)) return;
18      this.convertedAmount = (this.amount * this.exchangeRate).toFixed(2);
19    }
20  }
21 });
```

The status bar at the bottom shows the following information: `Ln 2, Col 16`, `Spaces: 4`, `UTF-8`, `CRLF`, `JavaScript`, `Port: 5500`. The system tray at the bottom right shows the date `12-03-2024` and time `16:20`.



Q2 T2. Create a stopwatch application through which users can start, pause and reset the timer. Use React state, event handlers and the `setTimeout` or `setInterval` functions to manage the timer's state and actions.

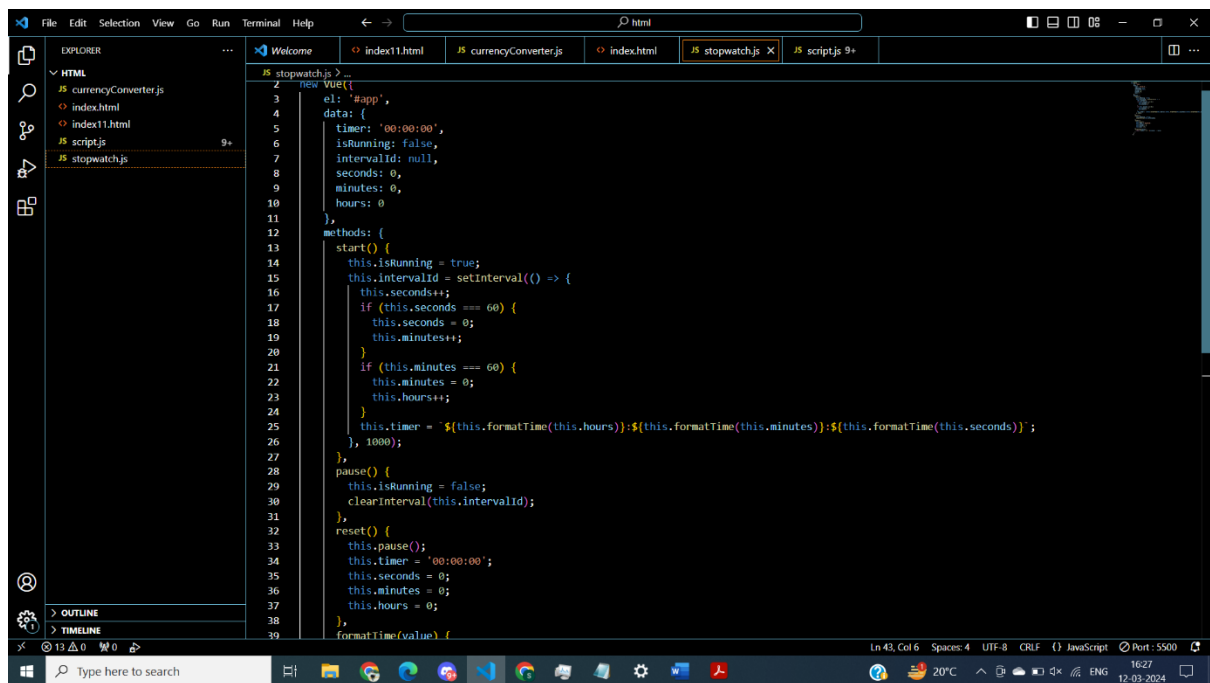
Note: Everything has to be done using Vue.js



This screenshot shows the VS Code editor with the `index.html` file open. The Explorer sidebar on the left shows the project structure: `HTML` (containing `index.html`), `JS` (containing `currencyConverter.js`, `stopwatch.js`, and `script.js`), and `OUTLINE`. The main editor area displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Stopwatch</title>
7 </head>
8 <body>
9   <div id="app">
10    <h1>Stopwatch</h1>
11    <div>
12      <p>{{ timer }}</p>
13      <button @click="start" v-if="!isRunning">Start</button>
14      <button @click="pause" v-if="isRunning">Pause</button>
15      <button @click="reset">Reset</button>
16    </div>
17  </div>
18  <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
19  <script src="stopwatch.js"></script>
20 </body>
21 </html>
```

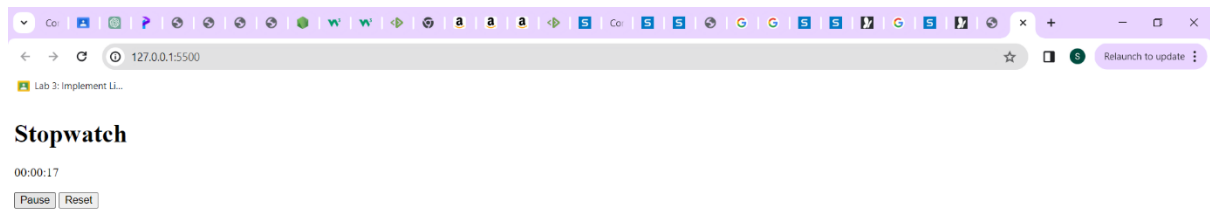
The status bar at the bottom indicates the cursor is at line 12, column 25, with 2 spaces, in UTF-8 encoding, CRLF line endings, and HTML language mode. The system tray shows the date and time as 16:26 on 12-03-2024.



This screenshot shows the VS Code editor with the `stopwatch.js` file open. The Explorer sidebar on the left shows the project structure: `HTML` (containing `index.html`), `JS` (containing `currencyConverter.js`, `stopwatch.js`, and `script.js`), and `OUTLINE`. The main editor area displays the following JavaScript code:

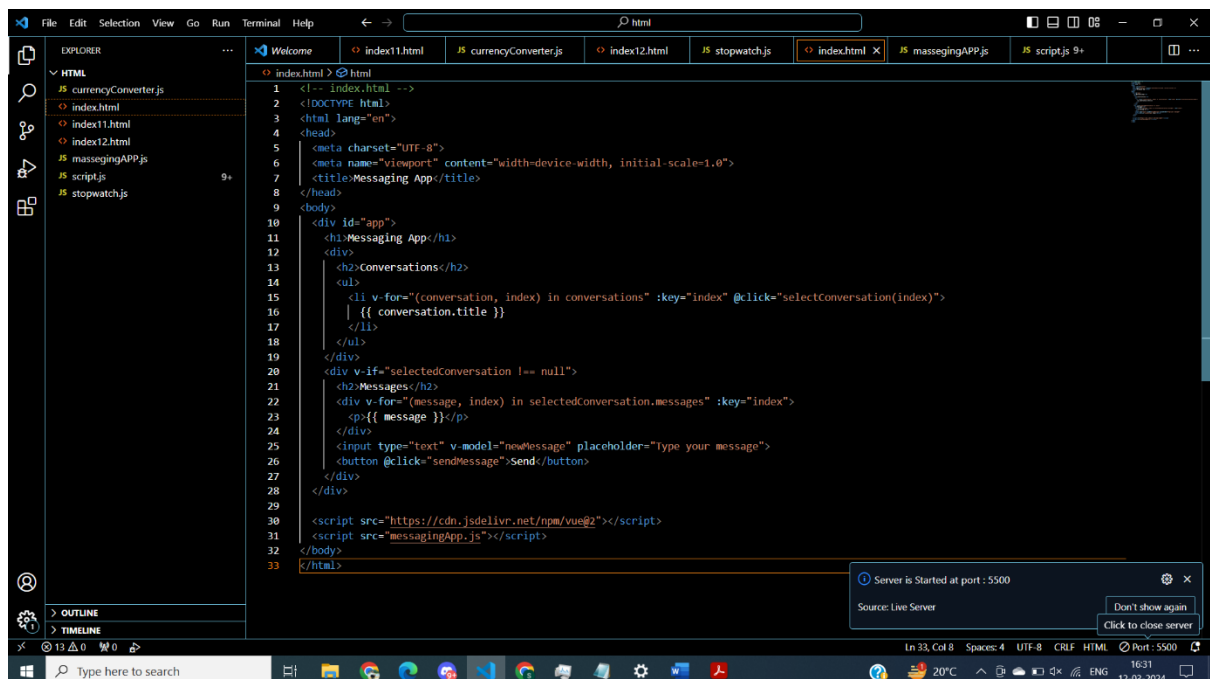
```
1 new Vue({
2   el: '#app',
3   data: {
4     timer: '00:00:00',
5     isRunning: false,
6     intervalId: null,
7     seconds: 0,
8     minutes: 0,
9     hours: 0,
10  },
11  methods: {
12    start() {
13      this.isRunning = true;
14      this.intervalId = setInterval(() => {
15        this.seconds++;
16        if (this.seconds === 60) {
17          this.seconds = 0;
18          this.minutes++;
19        }
20        if (this.minutes === 60) {
21          this.minutes = 0;
22          this.hours++;
23        }
24        this.timer = `${this.formatTime(this.hours)}:${this.formatTime(this.minutes)}:${this.formatTime(this.seconds)}`;
25      }, 1000);
26    },
27    pause() {
28      this.isRunning = false;
29      clearInterval(this.intervalId);
30    },
31    reset() {
32      this.pause();
33      this.timer = '00:00:00';
34      this.seconds = 0;
35      this.minutes = 0;
36      this.hours = 0;
37    },
38    formatTime(value) {
39      return value < 10 ? `0${value}` : value;
```

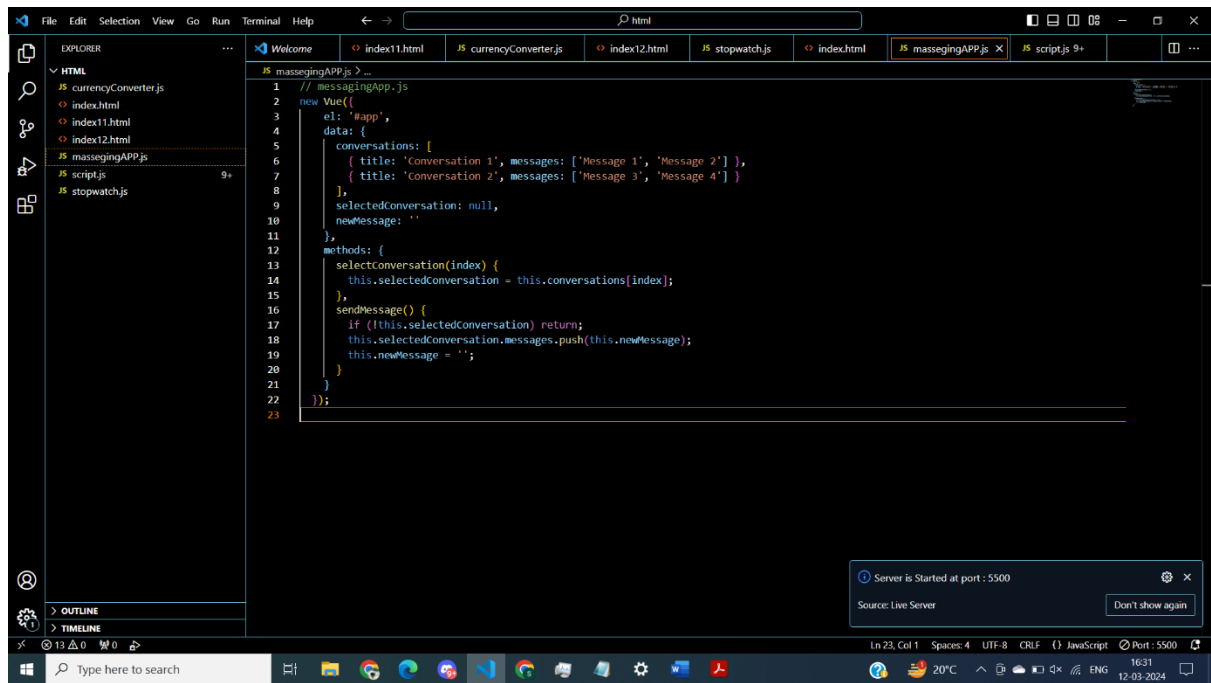
The status bar at the bottom indicates the cursor is at line 43, column 6, with 4 spaces, in UTF-8 encoding, CRLF line endings, and JavaScript language mode. The system tray shows the date and time as 16:27 on 12-03-2024.



Q2 T3. Develop a messaging application that allows users to send and receive messages in real time. The application should display a list of conversations and allow the user to select a specific conversation to view its messages. The messages should be displayed in a chat interface with the most recent message at the top. Users should be able to send new messages and receive push notifications.

Note: Everything has to be done using Vue.js





Messaging App

Conversations

- {{ conversation.title }}

Messages

{{ message }}