Date: 12th March 2024

Web Technology lab

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.

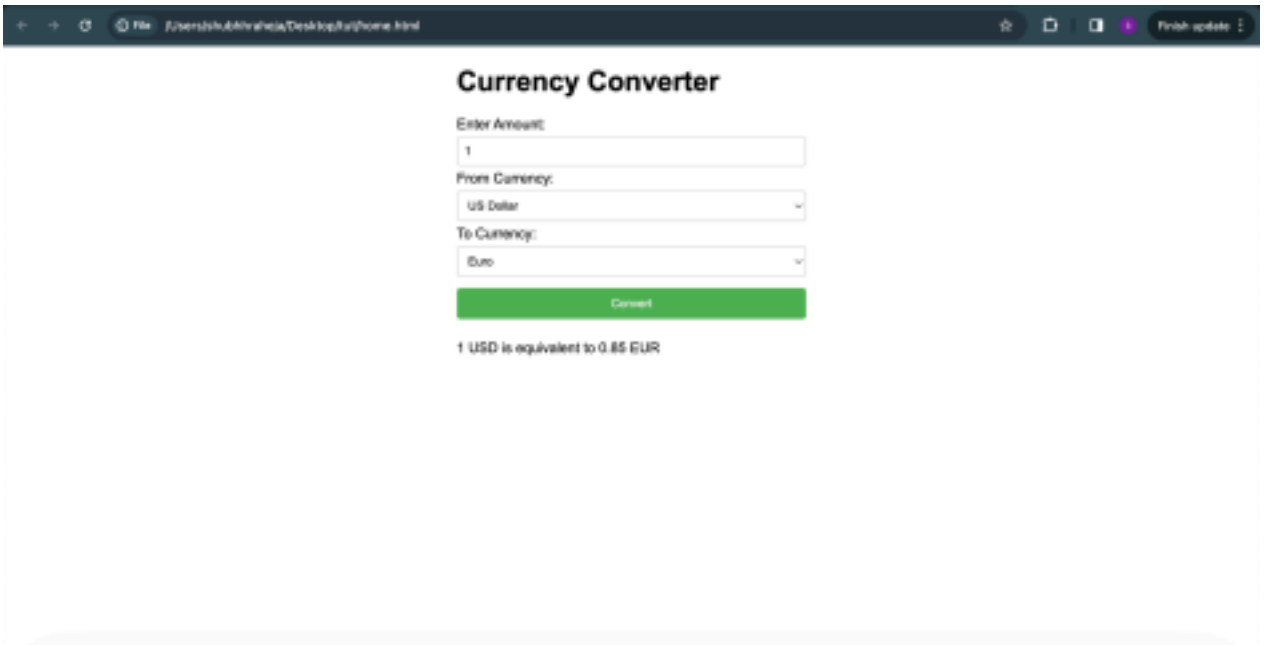
```
<!DOCTYPE html>

<html lang="en">

<head>

 <meta charset="UTF-8">

 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Currency Converter</title>

 <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

 <style>

   body {

     font-family: Arial, sans-serif;

   }

   #app {

     max-width: 400px;

     margin: 0 auto;

   }
```

```
input[type="number"], select {
  width: 100%;
  padding: 8px;
  margin: 5px 0;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}

button {
  width: 100%;
  background-color: #4CAF50;
  color: white;
  padding: 10px 15px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #45a049;
}
```

```
 </style>

</head>

<body>



<div id="app">

 <h1>Currency Converter</h1>

 <div>

   <label for="amount">Enter Amount:</label>

   <input type="number" id="amount" v-model.number="amount">
 </div>

 <div>

   <label for="fromCurrency">From Currency:</label>

   <select id="fromCurrency" v-model="fromCurrency">

               <option v-for="(currency, index) in currencies" :key="index"
:value="currency.code">{{ currency.name }}</option>

   </select>

 </div>

 <div>

   <label for="toCurrency">To Currency:</label>

   <select id="toCurrency" v-model="toCurrency">
               <option v-for="(currency, index) in currencies" :key="index"
:value="currency.code">{{ currency.name }}</option>

   </select>
```

```
  </div>

  <button @click="convert">Convert</button>

  <div v-if="convertedAmount !== null">

      <p>{{ amount }} {{ fromCurrency }} is equivalent to {{ convertedAmount }} {{
toCurrency }}</p>

  </div>

</div>




<script>
new Vue({

  el: '#app',

  data: {

    amount: 0,

    fromCurrency: 'USD',

    toCurrency: 'EUR',

    exchangeRate: {

      USD: 1, // 1 USD = 1 USD (base currency)

      EUR: 0.85, // 1 USD = 0.85 EUR

      GBP: 0.73, // 1 USD = 0.73 GBP
      JPY: 110.21 // 1 USD = 110.21 JPY

    },

    currencies: [
```

```
      { code: 'USD', name: 'US Dollar' },
      { code: 'EUR', name: 'Euro' },
      { code: 'GBP', name: 'British Pound' },
    { code: 'JPY', name: 'Japanese Yen' } ],
    convertedAmount: null
  },
 methods: {
    convert() {
                                      const rate = this.exchangeRate[this.toCurrency] /
this.exchangeRate[this.fromCurrency];
      this.convertedAmount = (this.amount * rate).toFixed(2);
    }
  }
});
</script>


</body>
</html>
```



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.

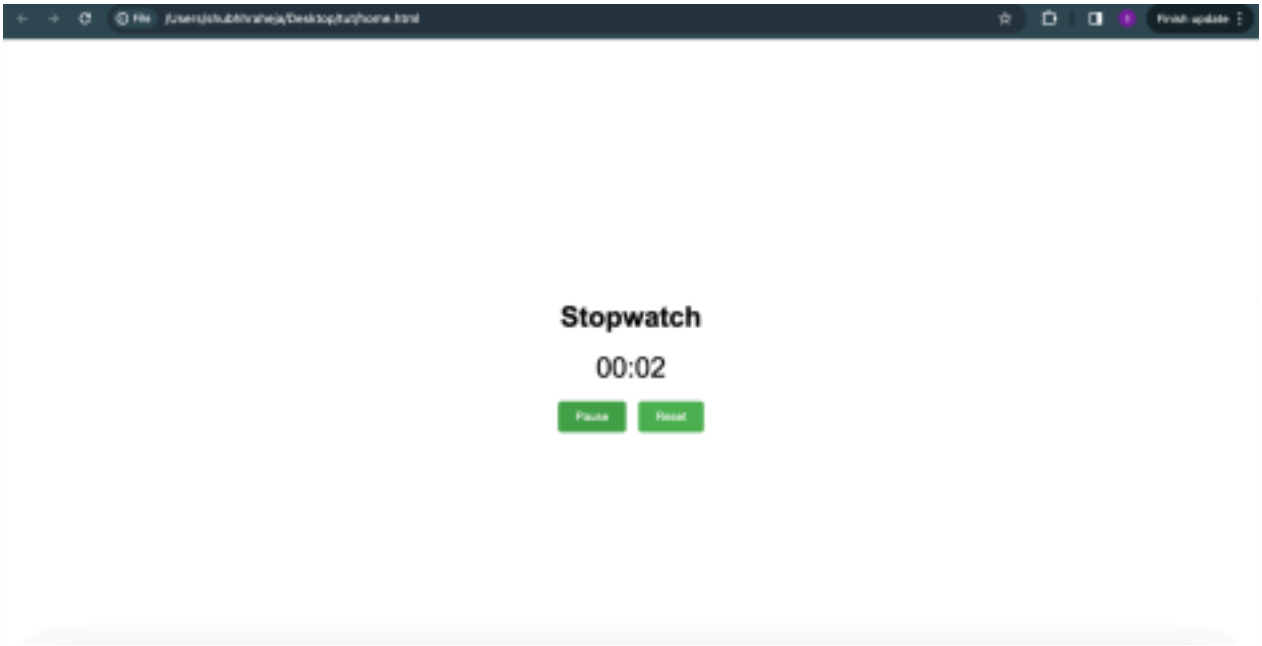
```
<!DOCTYPE html>

<html lang="en">

<head>

 <meta charset="UTF-8">

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

 <title>Stopwatch</title>

 <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

 <style>

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

     display: flex;

     justify-content: center;

     align-items: center;
```

```
      height: 100vh;
    }
    #app {
      text-align: center;
    }
    button {
      background-color: #4CAF50;
      color: white;
      padding: 10px 20px;
      margin: 0 5px;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background-color: #45a049;
    }
    #timer {
      font-size: 2em;
      margin-bottom: 20px;
    }
  </style>
</head>
```

```
<body>



<div id="app">

 <h1>Stopwatch</h1>

 <div id="timer">{{ formatTime }}</div>

 <div>

   <button @click="startStop">{{ running ? 'Pause' : 'Start' }}</button>

 <button @click="reset" :disabled="!running && seconds === 0">Reset</button>

 </div>

</div>



<script>

 new Vue({

   el: '#app',

   data: {

     running: false,

     seconds: 0,
     interval: null

   },

   computed: {

     formatTime() {

       const minutes = Math.floor(this.seconds / 60);

       const seconds = this.seconds % 60;
```

```
            return `${minutes < 10 ? '0' : ''}${minutes}:${seconds < 10 ? '0' :
''}${seconds}`;

      }

    },

    methods: {

      startStop() {
        if (this.running) {

          clearInterval(this.interval);

        } else {

          this.interval = setInterval(() => {

            this.seconds++;

          }, 1000);

        }

        this.running = !this.running;

      },

      reset() {

        clearInterval(this.interval);
        this.running = false;

        this.seconds = 0;

      }

    }

  });

</script>
```

```
</body>

</html>
```



T2. 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.

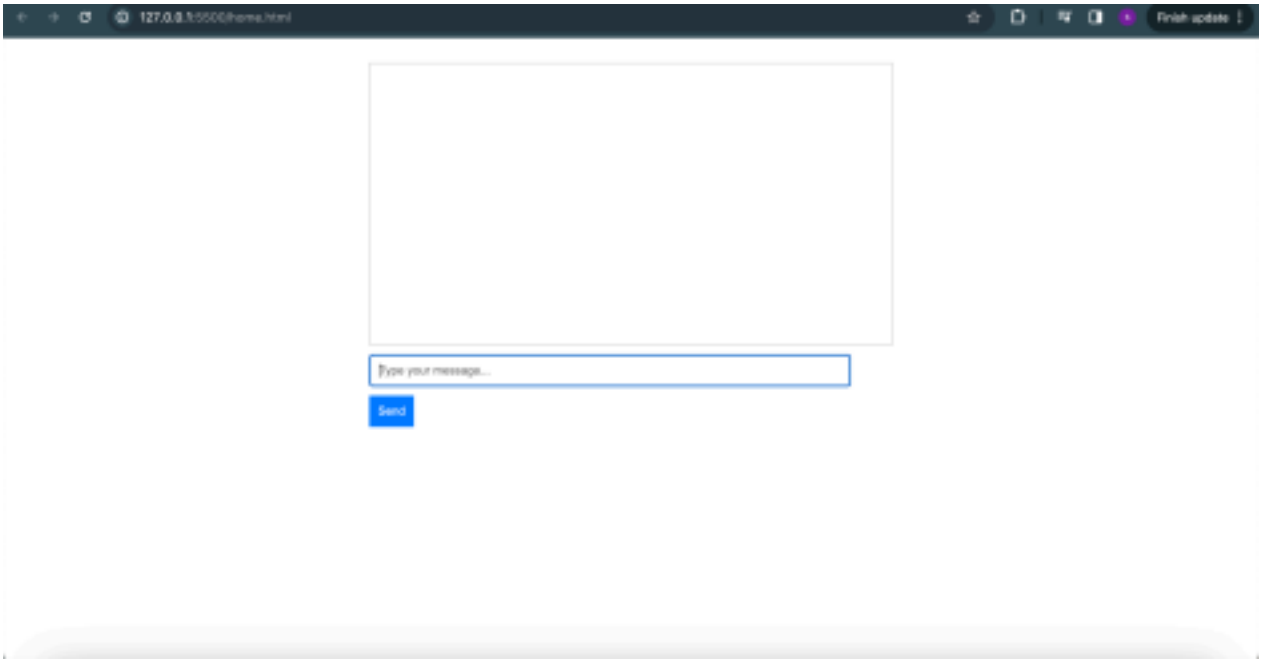
```
<!DOCTYPE html>
<html lang="en">

<head>
```

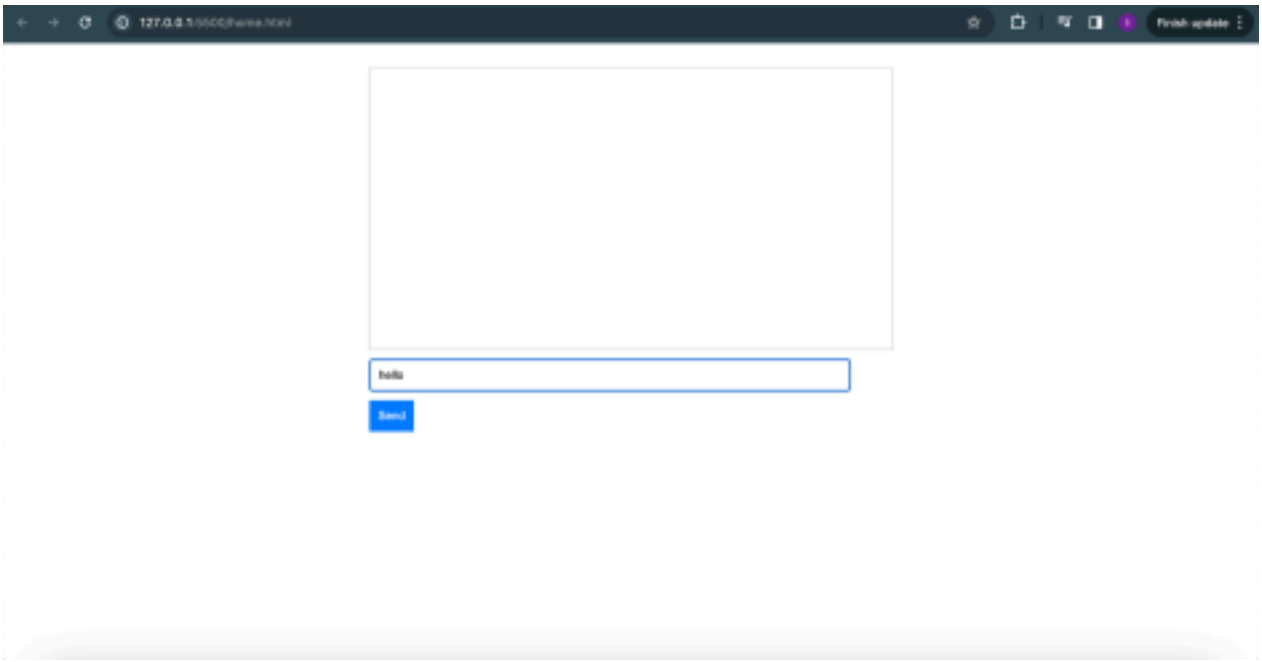
```
<meta charset="UTF-8">

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

<title>Vue.js Chat App</title>

<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>

<style>

  /* CSS styles for the chat app interface */

  #app {

    font-family: Arial, sans-serif;

    max-width: 600px;
    margin: 0 auto;

    padding: 20px;

  }



  .chat-window {

    border: 1px solid #ccc;

    padding: 10px;

    height: 300px;

    overflow-y: scroll;

  }

  .message {

    margin-bottom: 10px;

  }
```

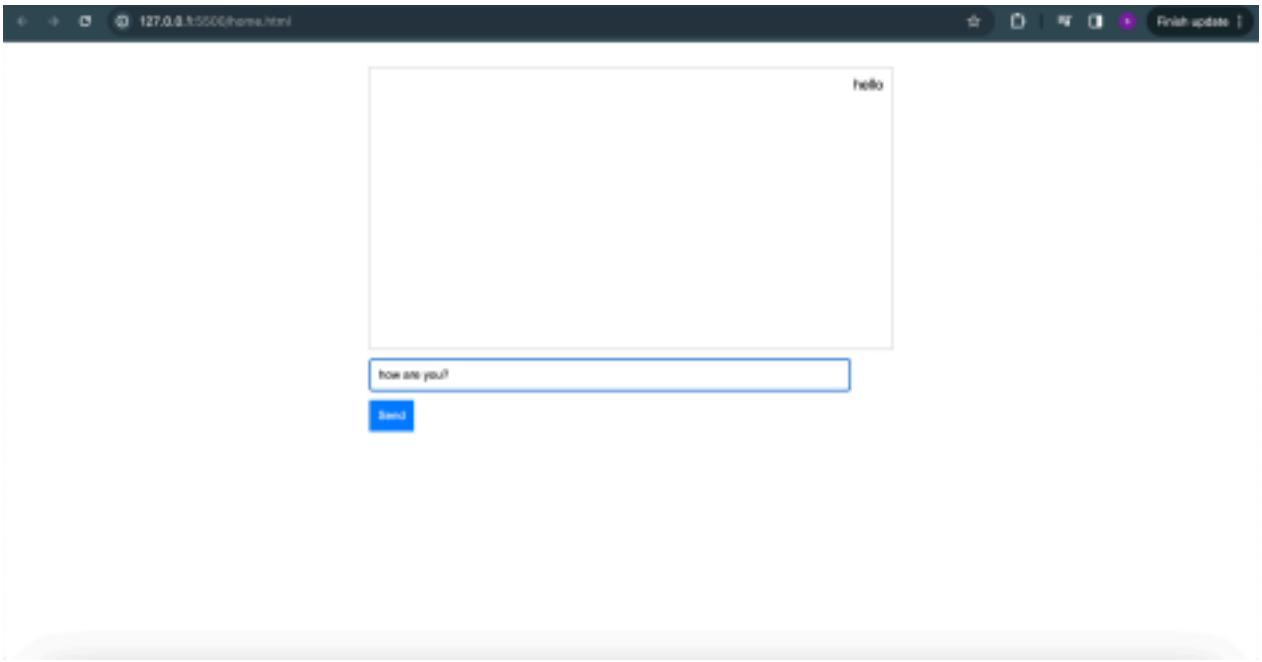
```
    .message.sent {
      text-align: right;
    }

    input[type="text"] {
      width: calc(100% - 70px);
      padding: 10px;
      margin-top: 10px;
      border: 1px solid #ccc;
    }

    button {
      padding: 10px;
      margin-top: 10px;
      background-color: #007bff;
      color: #fff;
      border: none;
      cursor: pointer;
    }
  </style>
</head>
<body>
```

```
<div id="app">
  <div class="chat-window" ref="chatWindow">
    <div class="message" v-for="(message, index) in messages" :key="index" :class="{ 'sent': message.sentByMe }">
      {{ message.text }}
    </div>
  </div>
  <input type="text" v-model="newMessage" @keyup.enter="sendMessage" placeholder="Type your message...">
  <button @click="sendMessage">Send</button>
</div>

<script>
  new Vue({
    el: '#app',
    data: {
      messages: [],
      newMessage: ''
    },
    methods: {
      sendMessage() {
        if (this.newMessage.trim() !== '') {
          this.messages.push({
            text: this.newMessage,
```

```
            sentByMe: true // Assume the message is sent by the user for simplicity
          });
          this.newMessage = '';
          // Scroll to the bottom of the chat window after sending a message
          this.$nextTick(() => {
            this.$refs.chatWindow.scrollTop = this.$refs.chatWindow.scrollHeight;
          });
        }
      }
    });
  </script>
</body>
</html>
```




hello
Send


hello
how are you?
Send