

# WEB TECHNOLOGY

## LAB 8

### ASSIGNMENT

Sahil

22cs3052

Cse

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

```
code > src > App.jsx > CurrencyConverter
1  import React, { useState } from 'react';
2
3  const CurrencyConverter = () => {
4    const [amount, setAmount] = useState('');
5    const [fromCurrency, setFromCurrency] = useState('USD');
6    const [toCurrency, setToCurrency] = useState('EUR');
7    const exchangeRate = 0.85;
8
9    const handleAmountChange = (e) => {
10     setAmount(e.target.value);
11   };
12
13   const handleFromCurrencyChange = (e) => {
14     setFromCurrency(e.target.value);
15   };
16
17   const handleToCurrencyChange = (e) => {
18     setToCurrency(e.target.value);
19   };
20
21   const convertCurrency = () => {
22     const convertedAmount = parseFloat(amount) * exchangeRate;
23     return convertedAmount.toFixed(2);
24   };
25
26   return (
27     <div>
28       <h2>Currency Converter</h2>
29       <div>
30         <label htmlFor="fromCurrency">From:</label>
31         <select id="fromCurrency" value={fromCurrency} onChange={handleFromCurrencyChange}>
32           <option value="USD">USD</option>
33           <option value="EUR">EUR</option>
34         </select>
35         <input type="number" value={amount} onChange={handleAmountChange} />
36       </div>
37       <div>
38         <label htmlFor="toCurrency">To:</label>
39         <select id="toCurrency" value={toCurrency} onChange={handleToCurrencyChange}>
40           <option value="USD">USD</option>
41           <option value="EUR">EUR</option>
42         </select>
43         <span>{convertCurrency()}</span>
44       </div>
45     </div>
46   );
47 }
48
49 export default CurrencyConverter;
```

```
App.css
1  .container {
2    padding: 20px;
3  }
```

https://direct-possibly.codedamn.app/

Currency Converter

From: USD 45

To: EUR 38.25

Toggle Browser Console

Fork on codedamn

Terminal Editor Browser

Use React state, event handlers and the `setTimeout` or `setInterval` functions to manage the

timer's state and actions.

```
code > src > App.jsx ...
1 import React, { useState, useEffect } from 'react';
2
3 const Stopwatch = () => {
4   const [time, setTime] = useState(0);
5   const [isRunning, setIsRunning] = useState(false);
6
7   useEffect(() => {
8     let timer;
9     if (isRunning) {
10       timer = setInterval(() => {
11         setTime((prevTime) => prevTime + 1);
12       }, 1000);
13     } else {
14       clearInterval(timer);
15     }
16     return () => clearInterval(timer);
17   }, [isRunning]);
18
19   const startTimer = () => {
20     setIsRunning(true);
21   };
22
23   const pauseTimer = () => {
24     setIsRunning(false);
25   };
26
27   const resetTimer = () => {
28     setTime(0);
29     setIsRunning(false);
30   };
31 };
32
33 export default Stopwatch;
```

Stopwatch

00:03

Start Reset

```
code > src > App.jsx ...
29   setIsRunning(false);
30 };
31
32 const formatTime = (timeInSeconds) => {
33   const minutes = Math.floor(timeInSeconds / 60);
34   const seconds = timeInSeconds % 60;
35   return `${minutes < 10 ? '0' : ''}${minutes}:${seconds < 10 ? '0' : ''}${seconds}`;
36 };
37
38 return (
39   <div>
40     <h2>Stopwatch</h2>
41     <div>
42       <span>{formatTime(time)}</span>
43     </div>
44     <div>
45       {!isRunning ? (
46         <button onClick={startTimer}>Start</button>
47       ) : (
48         <button onClick={pauseTimer}>Pause</button>
49       )}
50       <button onClick={resetTimer}>Reset</button>
51     </div>
52   </div>
53 );
54
55 export default Stopwatch;
56
57
```

Stopwatch

00:03

Start Reset

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.

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

import './App.css';

function App() {

  const [conversations, setConversations] = useState([]);

  const [selectedConversation, setSelectedConversation] = useState(null);

  const [newMessage, setNewMessage] = useState('');

  useEffect(() => {

    // Simulating fetching conversations from backend
    // In a real scenario, you would make an API call to fetch conversations

    const sampleConversations = [

      { id: 1, name: 'Friend1', messages: ['Hello', 'Hi there!'] },

      { id: 2, name: 'Friend2', messages: ['Hey', 'How are you?'] },

      // Add more sample conversations if needed

    ];

    setConversations(sampleConversations);

  }, []);

  const handleConversationClick = (conversation) => {

    setSelectedConversation(conversation);

  };

}
```

```
const handleSendMessage = () => {  
  
  if (newMessage.trim() === '') return;  
  
  const updatedConversations = conversations.map((conv) => {  
  
    if (conv.id === selectedConversation.id) {  
  
      return {  
  
        ...conv,
```

```

        messages: [...conv.messages, { text: newMessage, sender: 'user' }],

    };

}

return conv;

});

setConversations(updatedConversations);

setNewMessage('');

});

return (

    <div className="App">

        <div className="sidebar">

            <h2>Conversations</h2>

            <ul>

                {conversations.map((conversation) => (

                    <li

                        key={conversation.id}

                        className={selectedConversation && selectedConversation.id ===
conversation.id ? 'active' : ''}

                        onClick={() => handleConversationClick(conversation)}

                    >

                        {conversation.name}

                    </li>

                )}

            </ul>

        </div>

    </div>

);

```

```
        </li>

      )})

    </ul>

  </div>

  <div className="chatbox">

    {selectedConversation ? (

      <>

        <div className="chat-header">
```



```

        <h2>{selectedConversation.name}</h2>

      </div>

      <div className="messages">

        {selectedConversation.messages.map((message, index) => (

          <div key={index} className={`message ${message.sender}`}>

            {message.text}

          </div>

        ))}

      </div>

      <div className="input">

        <input

          type="text"

          placeholder="Type your message..."

          value={newMessage}

          onChange={(e) => setNewMessage(e.target.value)}

        />

        <button onClick={handleSendMessage}>Send</button>

      </div>

    </>

  ) : (

    <p className="no-conversation">Select a conversation to start chatting</p>

  )}

</div>

</div>

);

}

export default App;

```

```
body {  
  
  font-family: Arial, sans-serif;  
  
  margin: 0;  
  
  padding: 0;  
  
  background-color: #f5f5f5;  
  
}
```

```
.App {  
  
  display: flex;  
  
  height: 100vh;  
  
}
```

```
.sidebar {  
  
  flex: 1;  
  
  background-color: #333;  
  
  color: white;  
  
  padding: 20px;  
  
}
```

```
.sidebar h2 {  
  
  margin-top: 0;  
  
  font-size: 1.5rem;  
  
}
```

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

```
ul li {  
  
  cursor: pointer;  
  
  padding: 10px 0;  
  
}  
  
ul li.active {  
  
  background-color: #555;  
  
}  
  
.chatbox {  
  
  flex: 3;  
  
  padding: 20px;  
  
  display: flex;  
  
  flex-direction: column;  
  
}  
  
.chat-header {  
  
  background-color: #555;  
  
  color: white;  
  
  padding: 10px 20px;  
  
}
```

```
.messages {  
  flex: 1;  
  overflow-y: scroll;  
  margin-top: 10px;  
}  
  
.message {
```

```
background-color: #f9f9f9;

padding: 10px;

border-radius: 10px;

margin-bottom: 10px;
}

.message.user {

align-self: flex-end;

background-color: #007bff;

color: white;
}

.message.user::after {

content: '';

position: absolute;

width: 0;

height: 0;

border-top: 8px solid transparent;

border-bottom: 8px solid transparent;

border-right: 8px solid #007bff;

left: -8px;

top: 50%;
```

```
margin-top: -8px;
```

```
}
```

```
.message:not(.user)::after {
```

```
content: '';
```

```
position: absolute;
```

```
width: 0;
```

```
height: 0;
```

```
border-top: 8px solid transparent;

border-bottom: 8px solid transparent;

border-left: 8px solid #f9f9f9;

right: -8px;

top: 50%;

margin-top: -8px;

}
```

```
.input {

display: flex;

margin-top: 10px;

}
```

```
.input input {

flex: 1;

padding: 10px;

border-radius: 5px 0 0 5px;

border: 1px solid #ccc;

}
```

```
.input button {

padding: 10px 20px;
```

```
.input button {  
  
  padding: 10px 20px;  
  
  border: none;  
  
  border-radius: 0 5px 5px 0;  
  
  background-color: #007bff;  
  
  color: white;  
  
  cursor: pointer;  
}  
  
.input button:hover {  
  
  background-color: #0056b3;  
}  
  
.no-conversation {  
  
  margin-top: 50px;  
  
  font-size: 1.2rem;  
  
  color: #777;  
}
```





