WEB TECHNOLOGY

LAB 8

ASSIGNMENT

Sahil

22cs3052

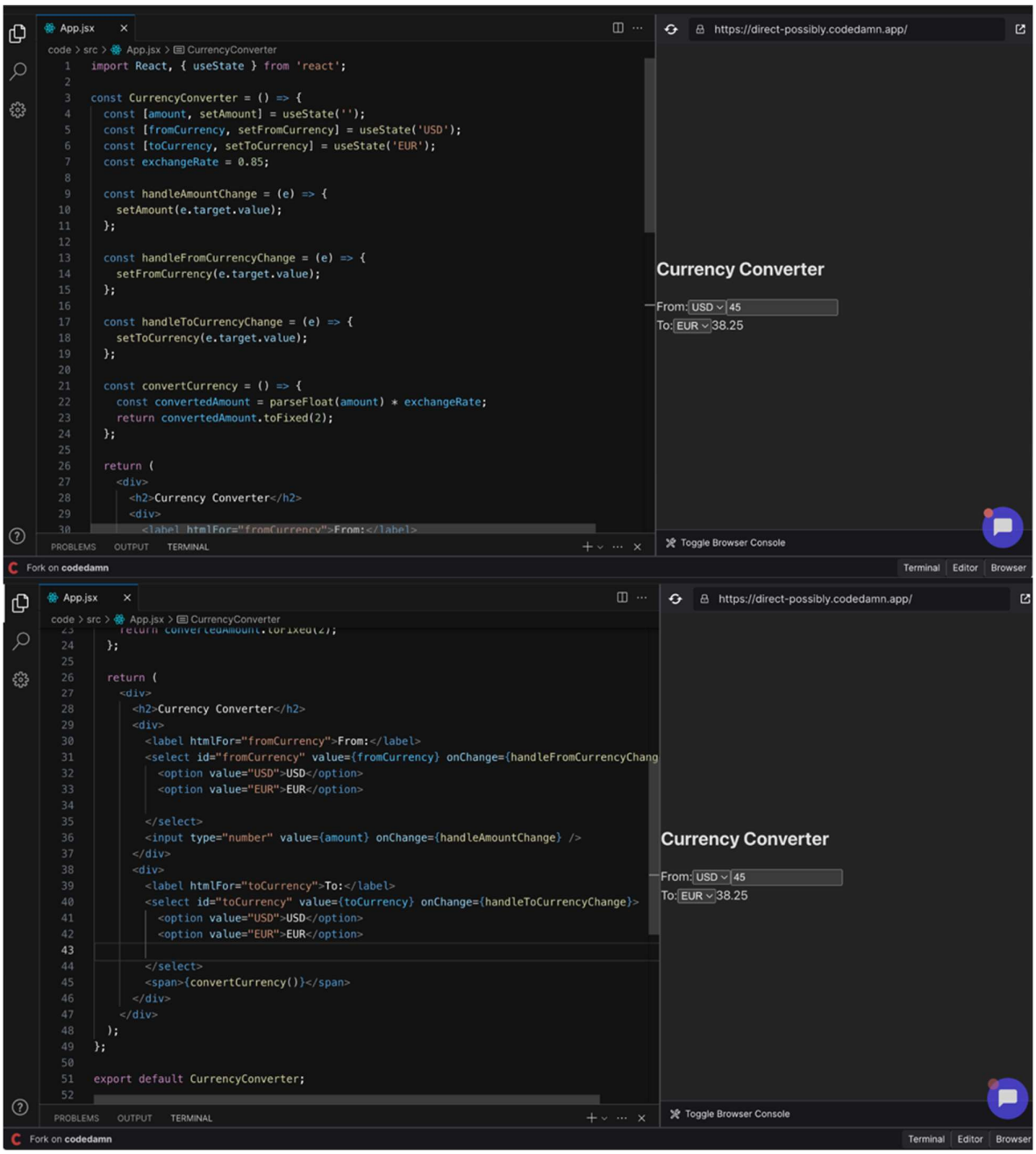
Cse

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

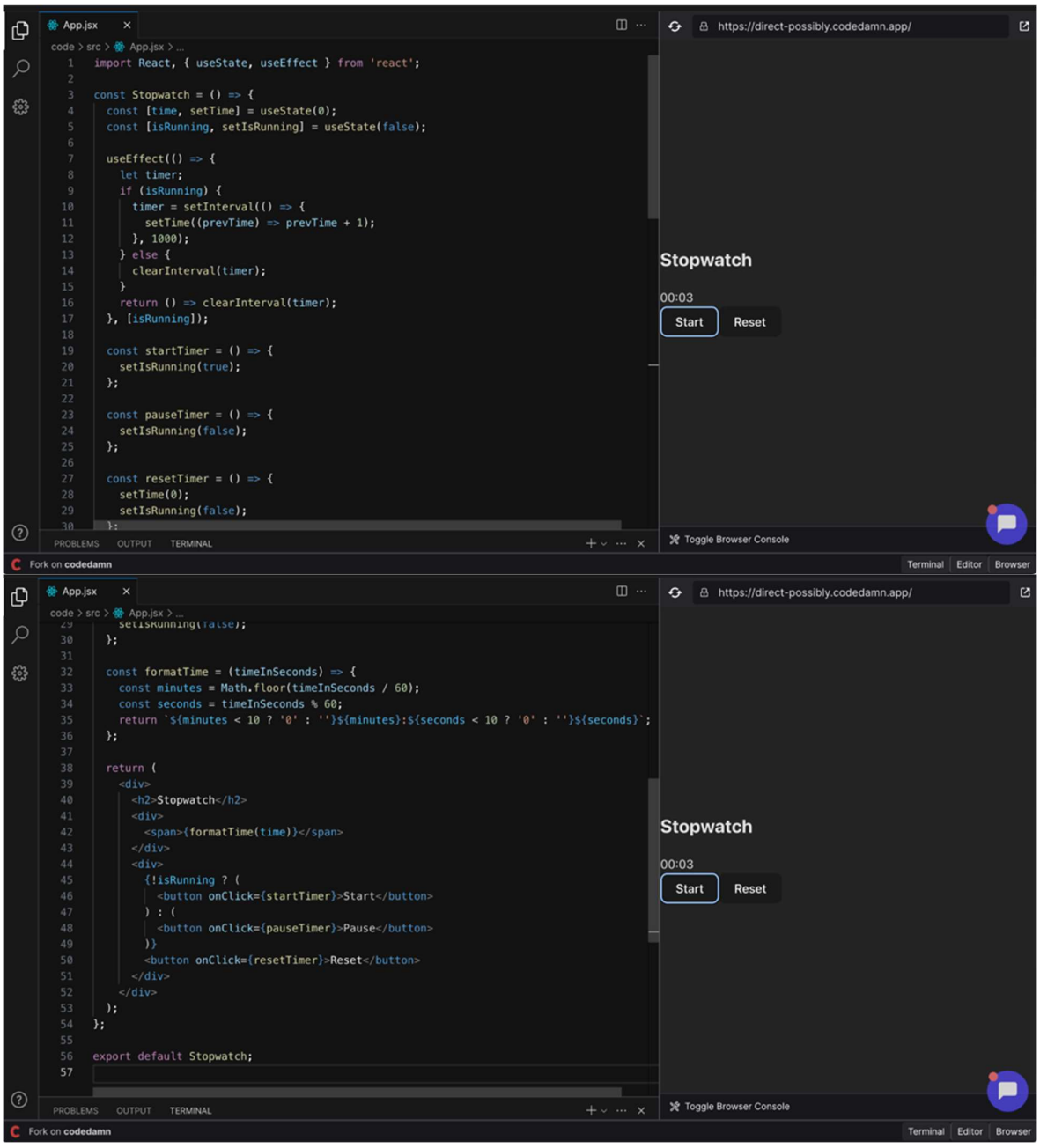
conversion calculations.



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.



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 ===
onversation.id ? 'active' : ''}
            onClick={() => handleConversationClick(conversation)}
          >
            {conversation.name}
          </li>
```

```
        </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;
}
```

```css
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;
}
```

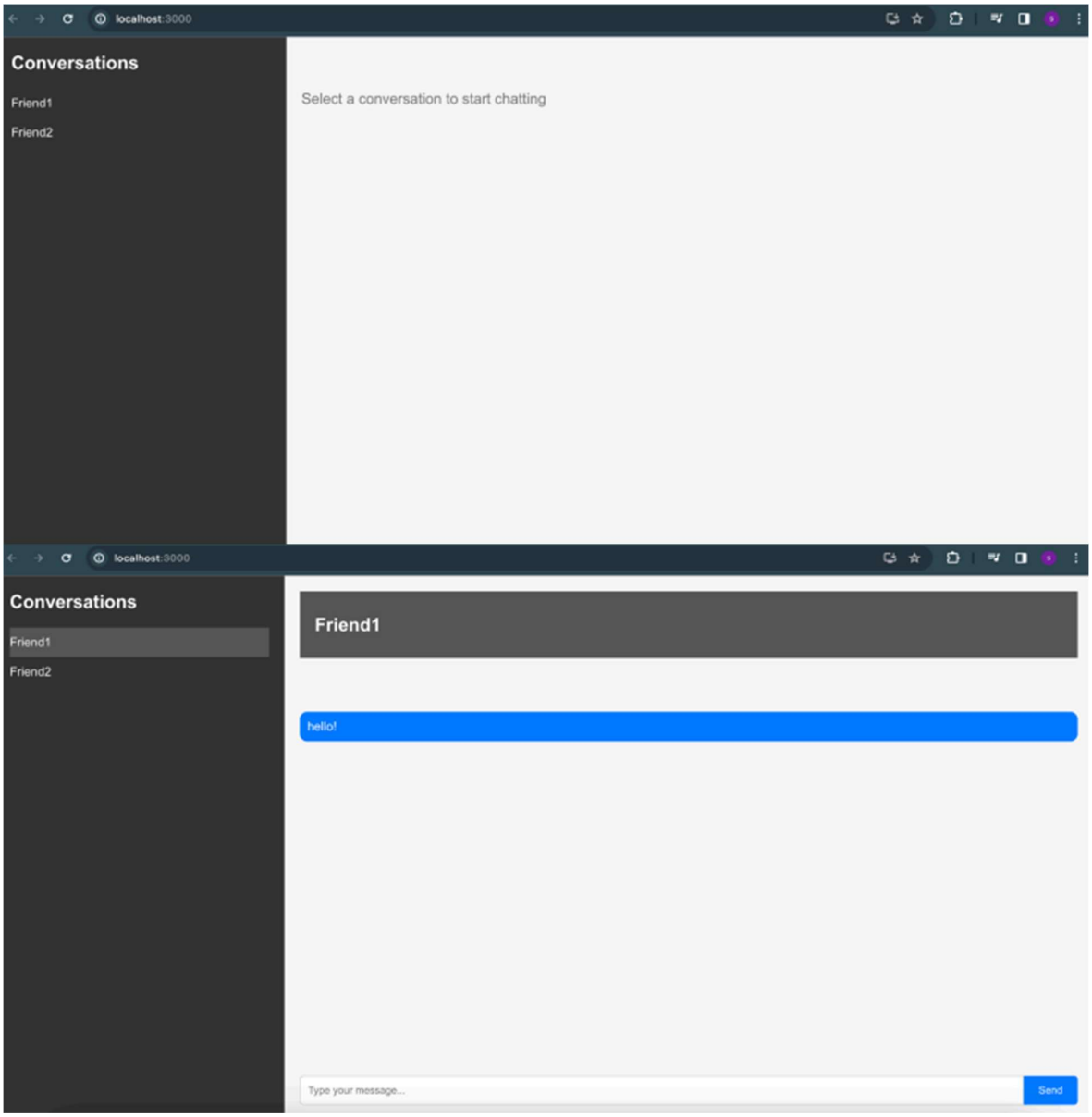
```
.messages {
  flex: 1;
  overflow-y: scroll;
  margin-top: 10px;
}


.message {
```

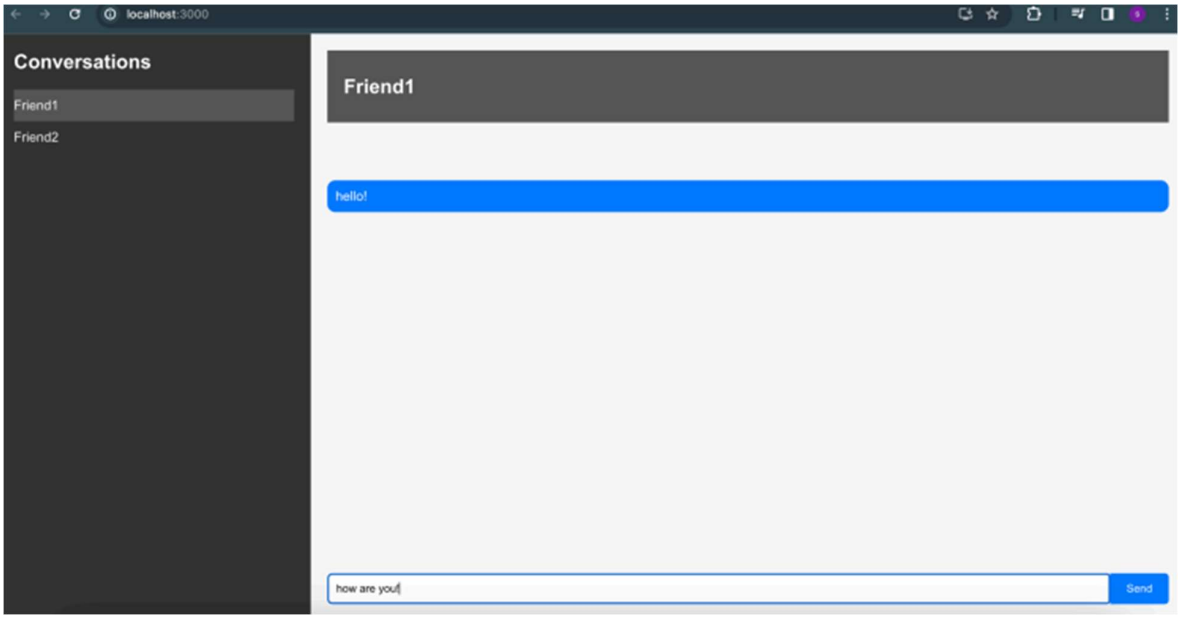
```
background-color: #f9f9f9;

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;
```

```css
.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;
}
```


localhost:3000
Conversations
Friend1
Friend2
Select a conversation to start chatting
localhost:3000
Conversations
Friend1
Friend2
Friend1
hello!
Type your message...
Send


localhost:3000
Conversations
Friend1
Friend2
Friend1
hello!
how are you!
Send