WEB TECHNOLOGY LAB

Assignment- 8

Name- Parminder Singh Roll No- 22CS2030 Branch- IDD

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.

App.js

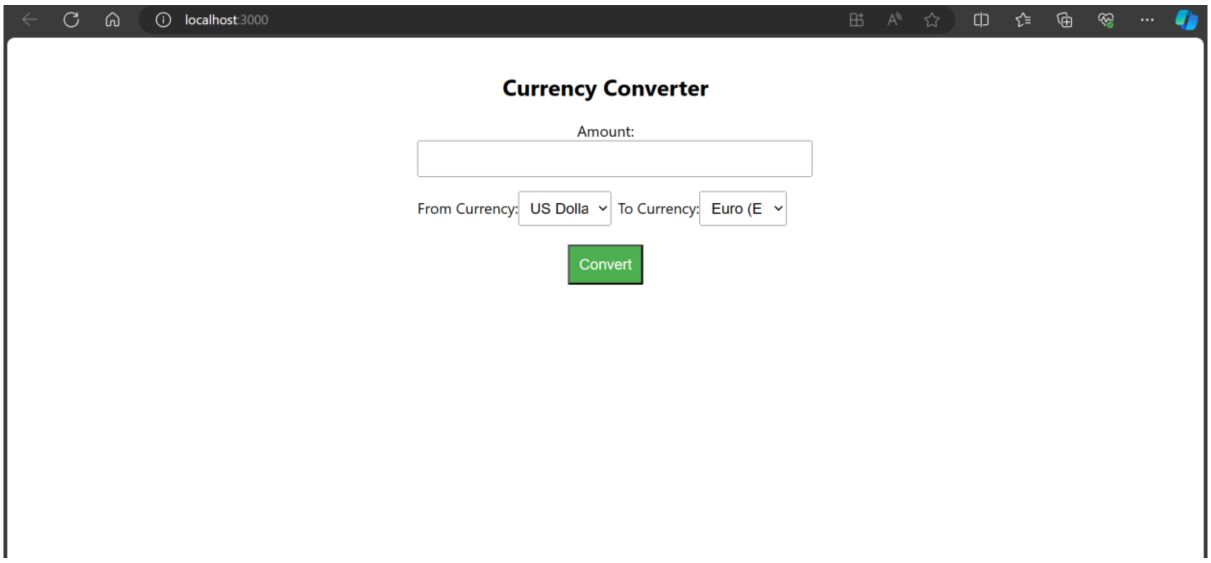
```
import React, { useState } from 'react';
import './App.css';

const currencies = [
  { code: 'USD', name: 'US Dollar' },
  { code: 'EUR', name: 'Euro' },
  { code: 'GBP', name: 'British Pound Sterling' },
  { code: 'JPY', name: 'Japanese Yen' },
  { code: 'INR', name: 'Indian Rupee' },
];

const exchangeRates = {
  USD: {
    EUR: 0.85,
    GBP: 0.75,
    JPY: 110.25,
    INR: 74.8,
  },
  EUR: {
    USD: 1.18,
    GBP: 0.89,
    JPY: 128.89,
    INR: 86.61,
  },
  GBP: {
    USD: 1.33,
    EUR: 1.12,
    JPY: 144.15,
    INR: 99.45,
  },
  JPY: {
    USD: 0.0091,
    EUR: 0.0078,
    GBP: 0.0069,
    INR: 0.74,
  },
  INR: {
    USD: 0.0134,
    EUR: 0.0115,
    GBP: 0.0101,
    JPY: 1.35,
  },
};

const App = () => {
  const [amount, setAmount] = useState('');
  const [sourceCurrency, setSourceCurrency] = useState('USD');
  const [targetCurrency, setTargetCurrency] = useState('EUR');
  const [convertedAmount, setConvertedAmount] = useState(null);

  const handleAmountChange = (e) => {
    setAmount(e.target.value);
  };

  const handleSourceCurrencyChange = (e) => {
    setSourceCurrency(e.target.value);
  };

  const handleTargetCurrencyChange = (e) => {
    setTargetCurrency(e.target.value);
  };

  const handleConvert = () => {
    const exchangeRate = exchangeRates[sourceCurrency][targetCurrency];
    const converted = parseFloat(amount) * exchangeRate;
    setConvertedAmount(isNaN(converted) ? 'Invalid Input' :
converted.toFixed(2));
  };

  return (
    <div className="container">
      <h1 className="heading">Currency Converter</h1>
      <div className="input-container">
        <label className="label">
          Amount:
          <input type="number" value={amount} onChange={handleAmountChange}
className="input" />
        </label>
      </div>
      <div className="select-container">
        <label className="label">
          From Currency:
```

```
          <select value={sourceCurrency} onChange={handleSourceCurrencyChange}
className="select">
            {currencies.map((currency) => (
              <option key={currency.code} value={currency.code}>
                {currency.name} ({currency.code})
              </option>
            ))}
          </select>
        </label>
        <label className="label">
          To Currency:
          <select value={targetCurrency} onChange={handleTargetCurrencyChange}
className="select">
            {currencies.map((currency) => (
              <option key={currency.code} value={currency.code}>
                {currency.name} ({currency.code})
              </option>
            ))}
          </select>
        </label>
      </div>
      <div>
        <button onClick={handleConvert} className="convert-button">
          Convert
        </button>
      </div>
      {convertedAmount !== null && (
        <div className="result">
          <p className="result-text">
            Converted Amount: {convertedAmount} {targetCurrency}
          </p>
        </div>
      )}
    </div>
  );
};

export default App;
```

App.css

```
.container {
  text-align: center;
  padding: 20px;
  max-width: 400px;
  margin: auto;
}
```

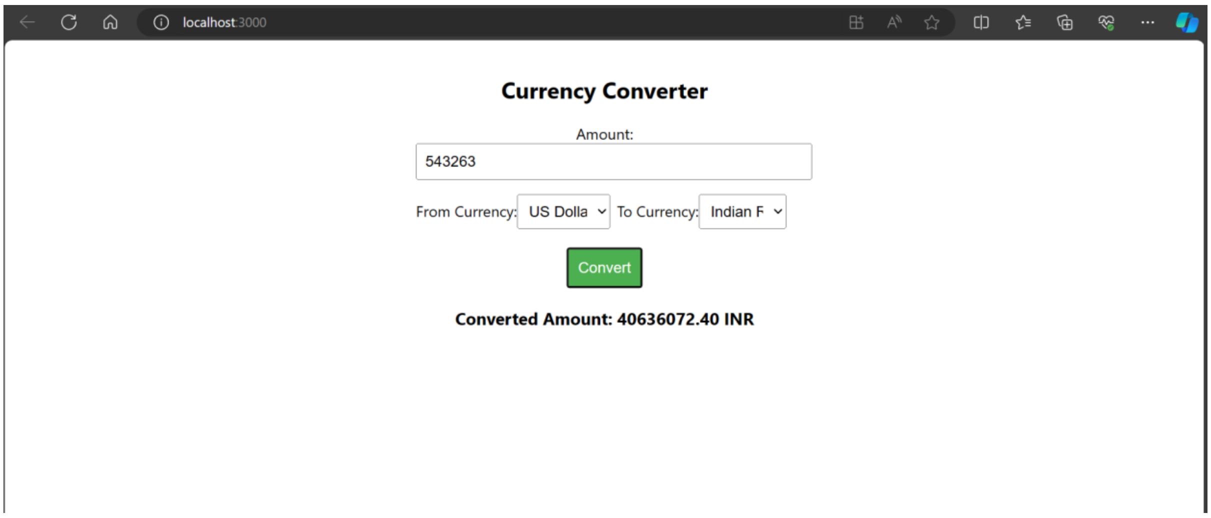
```css
.heading {
  font-size: 24px;
  margin-bottom: 20px;
}

.input-container {
  margin-bottom: 15px;
}

.label {
  display: block;
  margin-bottom: 5px;
}

.input {
  width: 100%;
  padding: 8px;
  font-size: 16px;
}

.select-container {
  display: flex;
  justify-content: space-between;
  margin-bottom: 15px;
}

.select {
  width: 48%;
  padding: 8px;
  font-size: 16px;
}

.convert-button {
  background: #4caf50;
  color: white;
  padding: 10px;
  font-size: 16px;
  cursor: pointer;
}

.result {
  margin-top: 20px;
}

.result-text {
  font-size: 18px;
  font-weight: bold;
}
```



Currency Converter

Amount:

From Currency: US Dolla To Currency: Euro (E



Currency Converter

Amount: 543263

From Currency: US Dolla To Currency: Indian F

Converted Amount: 40636072.40 INR

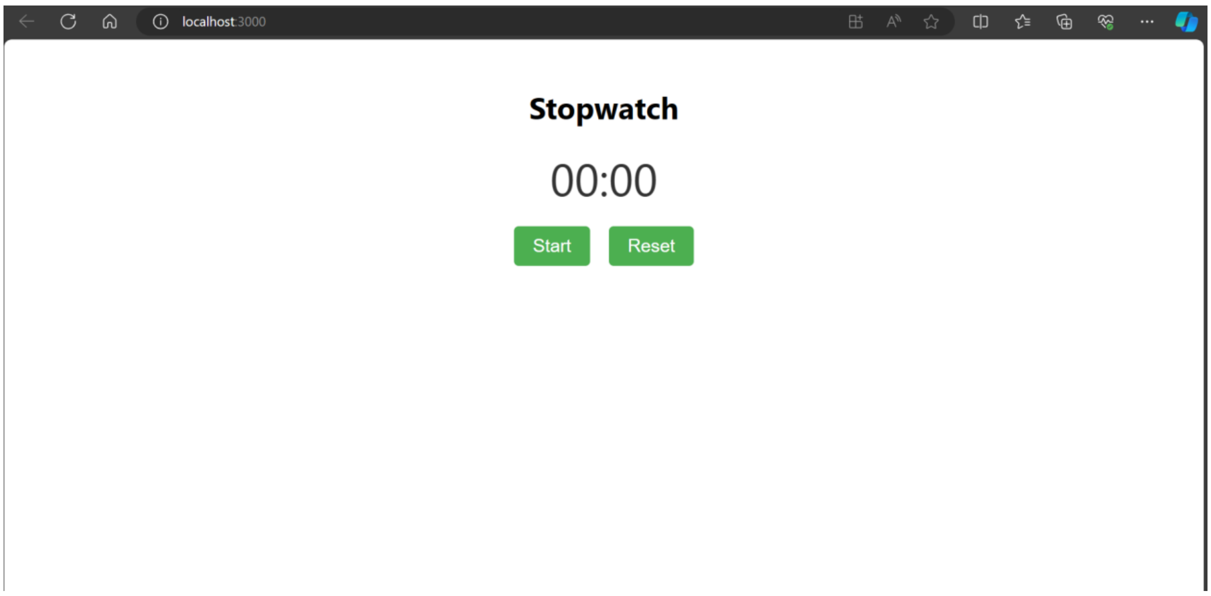
T2. Create a stopwatch application through which users can start, pause and reset the timer. Use React state, event handlers and the setTimeout or setInterval functions to manage the timer's state and actions.

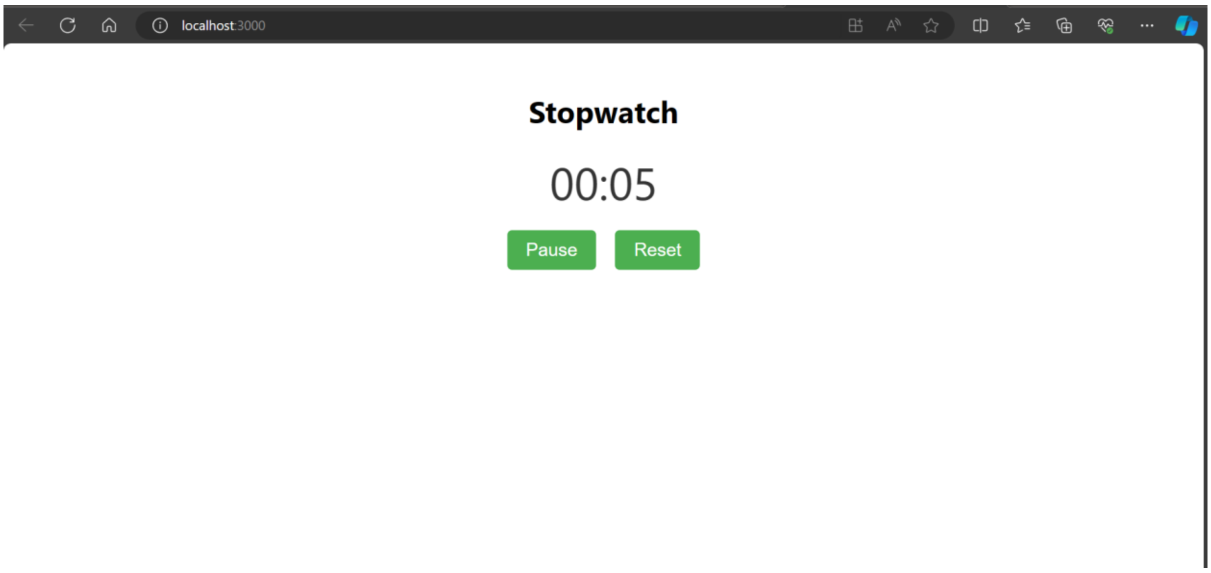
App.js

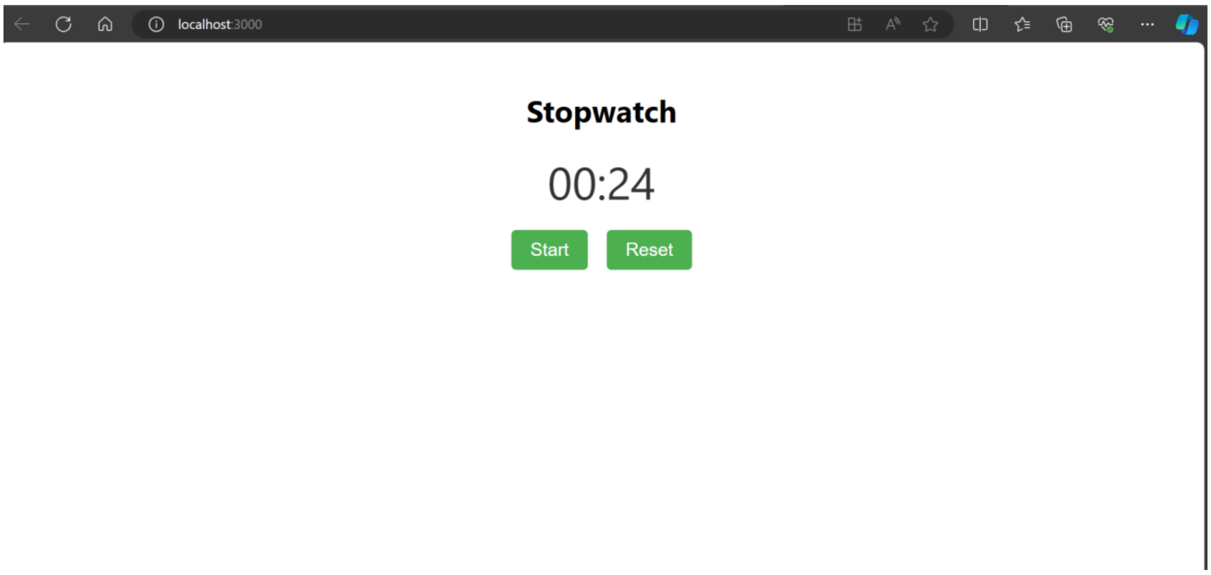
```
import React, { useState, useEffect } from 'react';
import './App.css';

const App = () => {
  const [time, setTime] = useState(0);
  const [isRunning, setIsRunning] = useState(false);

  useEffect(() => {
    let interval;
    if (isRunning) {
      interval = setInterval(() => {
        setTime((prevTime) => prevTime + 1);
      }, 1000);
    } else {
      clearInterval(interval);
    }

    return () => {
      clearInterval(interval);
    };
  }, [isRunning]);

  const handleStartPause = () => {
    setIsRunning((prevIsRunning) => !prevIsRunning);
  };

  const handleReset = () => {
    setTime(0);
    setIsRunning(false);
  };

  const formatTime = (seconds) => {
    const minutes = Math.floor(seconds / 60);
    const remainingSeconds = seconds % 60;
    return `${String(minutes).padStart(2,
'0')}:${String(remainingSeconds).padStart(2, '0')}`;
  };

  return (
    <div className="App">
      <h1>Stopwatch</h1>
      <div className="timer">{formatTime(time)}</div>
      <div className="controls">
        <button onClick={handleStartPause}>{isRunning ? 'Pause' :
'Start'}</button>
        <button onClick={handleReset}>Reset</button>
      </div>
    </div>
  );
};

export default App;
```

App.css

```css
.App {
  text-align: center;
  margin: 50px auto;
}

.timer {
  font-size: 3em;
  margin: 20px 0;
  color: #333;
}

.controls {
  display: flex;
  justify-content: center;
  gap: 20px;
}

.controls button {
  font-size: 1.2em;
  padding: 10px 20px;
  cursor: pointer;
  background-color: #4caf50;
  color: white;
  border: none;
  border-radius: 5px;
  outline: none;
}

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







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.

App.js

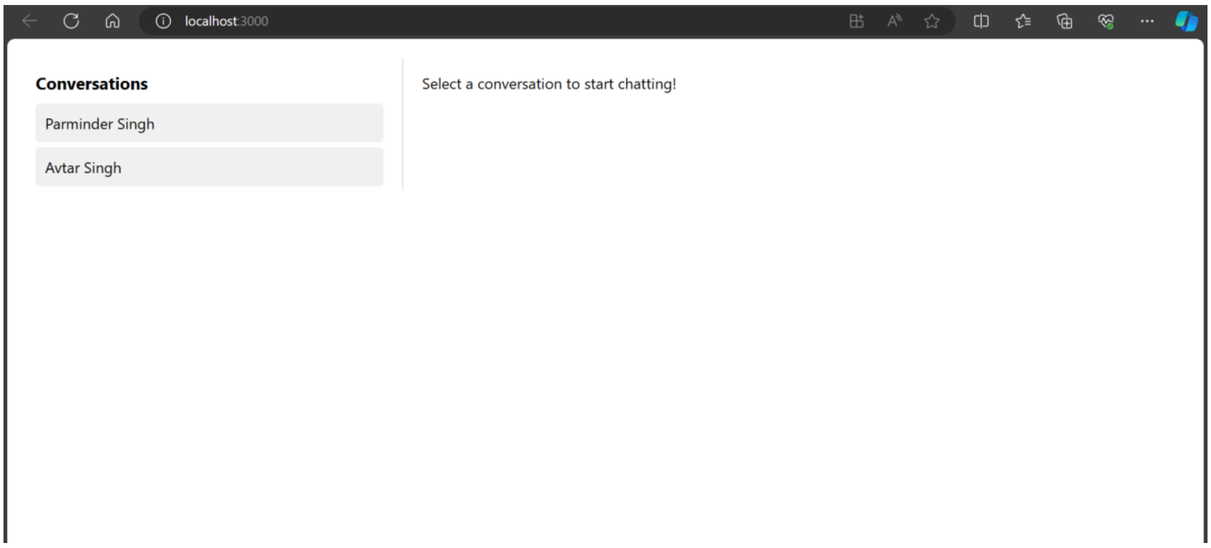
```
import React, { useState, useEffect } from 'react';
import './App.css';

const App = () => {
  const [conversations, setConversations] = useState([
  { id: 1, name: 'Parminder Singh', messages: [{ id: 1, text: 'Hello!',
sender: 'Parminder Singh', timestamp: Date.now() }] },
  { id: 2, name: 'Avtar Singh', messages: [] },
 ]);

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

 const handleConversationClick = (conversation) => {
  setSelectedConversation(conversation);
 };

 const handleSendMessage = () => {
  if (newMessage.trim() === '') return;

  const updatedConversations = conversations.map((conversation) =>
   conversation.id === selectedConversation.id
    ? {
      ...conversation,
      messages: [
       ...conversation.messages,
       { id: conversation.messages.length + 1, text: newMessage,
sender: 'You', timestamp: Date.now() },
      ],
     }
    : conversation
  );

  setConversations(updatedConversations);
  setNewMessage('');
 };

 useEffect(() => {
  const timeoutId = setTimeout(() => {
   if (selectedConversation) {
    const updatedConversations = conversations.map((conversation) =>
     conversation.id === selectedConversation.id
      ? {
        ...conversation,
        messages: [
         ...conversation.messages,
         { id: conversation.messages.length + 1, text: 'New
Message!', sender: 'Parminder Singh', timestamp: Date.now() },
        ],
       }
      : conversation
    );
```

```
        setConversations(updatedConversations);
      }
    }, 2000);

    return () => clearTimeout(timeoutId);
  }, [selectedConversation, conversations]);

  return (
    <div className="app">
      <div className="conversation-list">
        <h2>Conversations</h2>
        <ul>
          {conversations.map((conversation) => (
            <li key={conversation.id} onClick={() =>
handleConversationClick(conversation)}>
              {conversation.name}
            </li>
          ))}
        </ul>
      </div>
      <div className="chat">
        {selectedConversation ? (
          <>
            <h2>{selectedConversation.name}</h2>
            <div className="message-list">
              {selectedConversation.messages.map((message) => (
                <div key={message.id} className={message.sender === 'You' ?
'sent' : 'received'}>
                  <p>{message.text}</p>
                </div>
              ))}
            </div>
            <div className="message-input">
              <input
                type="text"
                placeholder="Type a message..."
                value={newMessage}
                onChange={(e) => setNewMessage(e.target.value)}
              />
              <button onClick={handleSendMessage}>Send</button>
            </div>
          </>
        ) : (
          <p>Select a conversation to start chatting!</p>
        )}
      </div>
    </div>
  );
```

```
};

export default App;
```

App.css

```css
.app {
  display: flex;
  justify-content: space-around;
  padding: 20px;
}

.conversation-list {
  width: 30%;
  border-right: 1px solid #ccc;
  padding-right: 20px;
}

.conversation-list h2 {
  font-size: 18px;
  margin-bottom: 10px;
}

ul {
  list-style: none;
  padding: 0;
  margin: 0;
}

li {
  cursor: pointer;
  padding: 10px;
  margin-bottom: 5px;
  background-color: #f0f0f0;
  border-radius: 5px;
}

li:hover {
  background-color: #e0e0e0;
}

.chat {
  width: 65%;
}

.message-list {
  max-height: 400px;
  overflow-y: auto;
}

.message-list div {
  margin: 10px;
  padding: 10px;
  border-radius: 5px;
}

.sent {
  background-color: #a3d2ca;
  align-self: flex-end;
}

.received {
  background-color: #ddd;
}

.message-input {
  margin-top: 20px;
  display: flex;
}

.message-input input {
  flex: 1;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin-right: 10px;
}

.message-input button {
  padding: 10px 20px;
  background-color: #4caf50;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

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



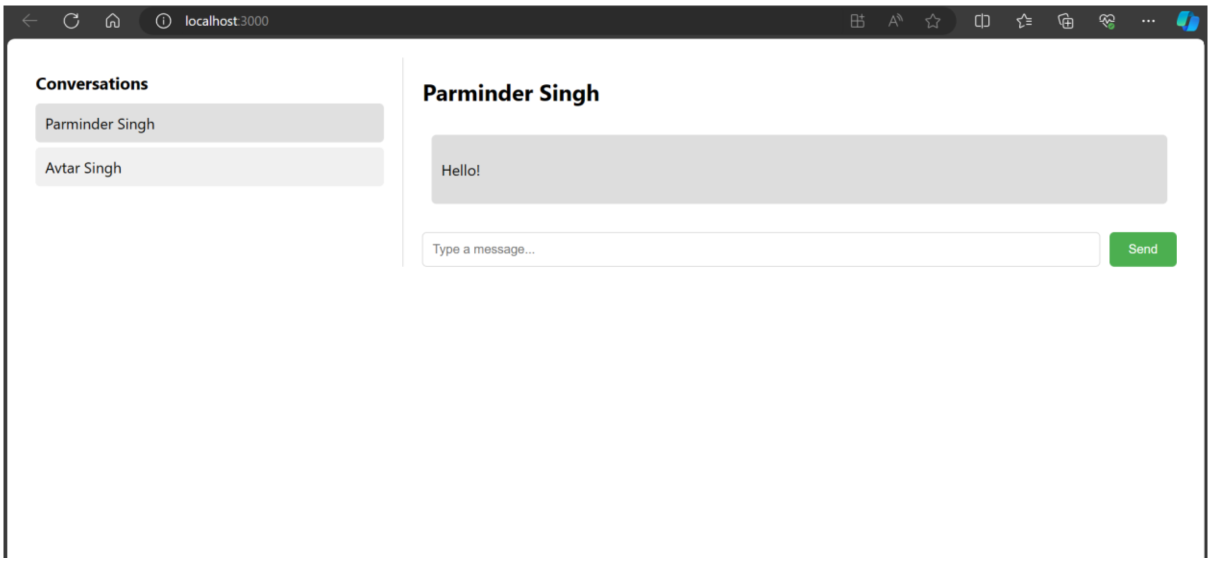
localhost:3000

Conversations

Parminder Singh

Avtar Singh

Select a conversation to start chatting!



localhost:3000

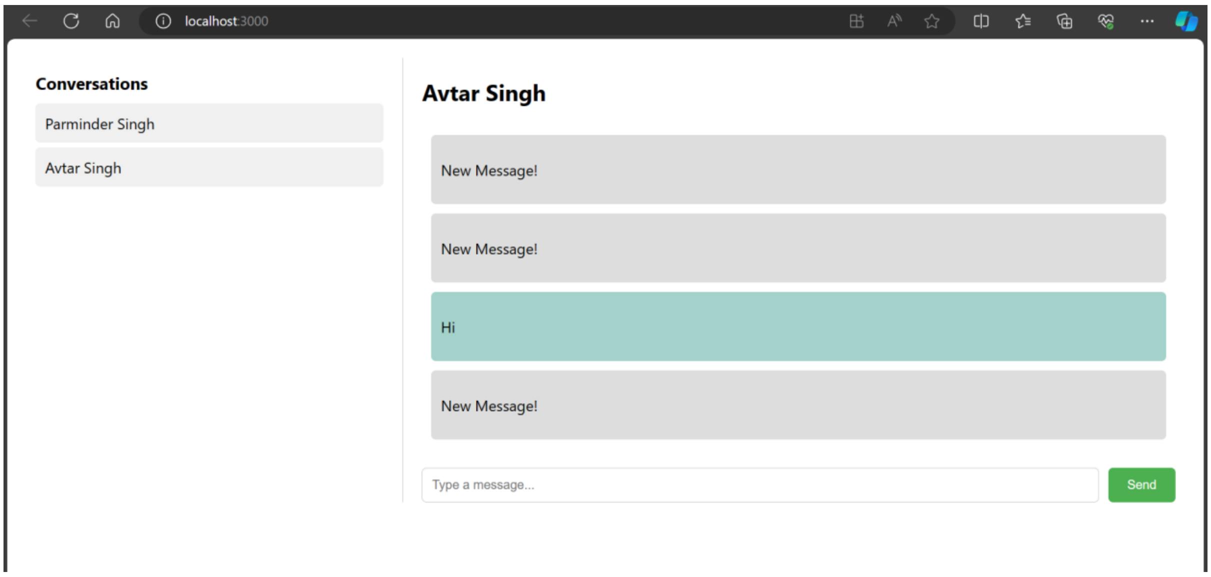
Conversations

Parminder Singh

Avtar Singh

Parminder Singh

Hello!



localhost:3000

Conversations

Parminder Singh

Avtar Singh

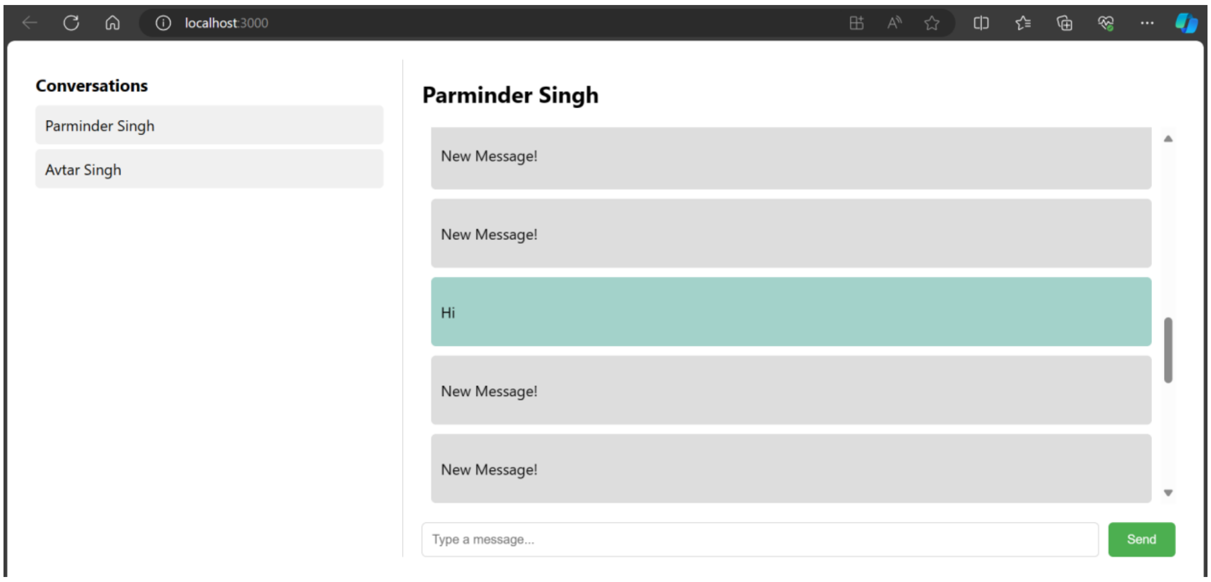
Avtar Singh

New Message!

New Message!

Hi

New Message!


localhost:3000

Conversations

Parminder Singh

Avtar Singh

Parminder Singh

New Message!

New Message!

Hi

New Message!

New Message!

Type a message...

Send