WEB TECHNOLOGY

LAB8

**Name : Kriti Mehrotra**

**Roll no.: 22IT3022**

**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.**

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

function App() {

const [amount, setAmount] = useState('');

const [fromCurrency, setFromCurrency] = useState('USD'); const [toCurrency, setToCurrency] = useState('INR');

const [convertedAmount, setConvertedAmount] = useState(null);

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

};

const handleFromCurrencyChange = (event) => { setFromCurrency(event.target.value);

};

const handleToCurrencyChange = (event) => { setToCurrency(event.target.value);

};

const handleConvert = () => { let exchangeRate;

if (fromCurrency === 'USD' && toCurrency === 'INR') {

exchangeRate = 82.91;

} else if (fromCurrency === 'INR' && toCurrency === 'USD') { exchangeRate = 1 / 82.91;

} else { exchangeRate = 1;

}

const converted = parseFloat(amount) \* exchangeRate; setConvertedAmount(converted.toFixed(2));

};

return (

<div className="App">

<h1>Currency Converter</h1>

<div>

<label> Amount:

<input type="number" value={amount} onChange={handleAmountChange} />

</label>

</div>

<div>

<label> From:

<select value={fromCurrency} onChange={handleFromCurrencyChange}>

<option value="USD">USD</option>

<option value="INR">INR</option>

</select>

</label>

</div>

<div>

<label> To:

<select value={toCurrency} onChange={handleToCurrencyChange}>

<option value="USD">USD</option>

<option value="INR">INR</option>

</select>

</label>

</div>

<button onClick={handleConvert}>Convert</button>

{convertedAmount && (

<p>{amount} {fromCurrency} equals {convertedAmount}

{toCurrency}</p>

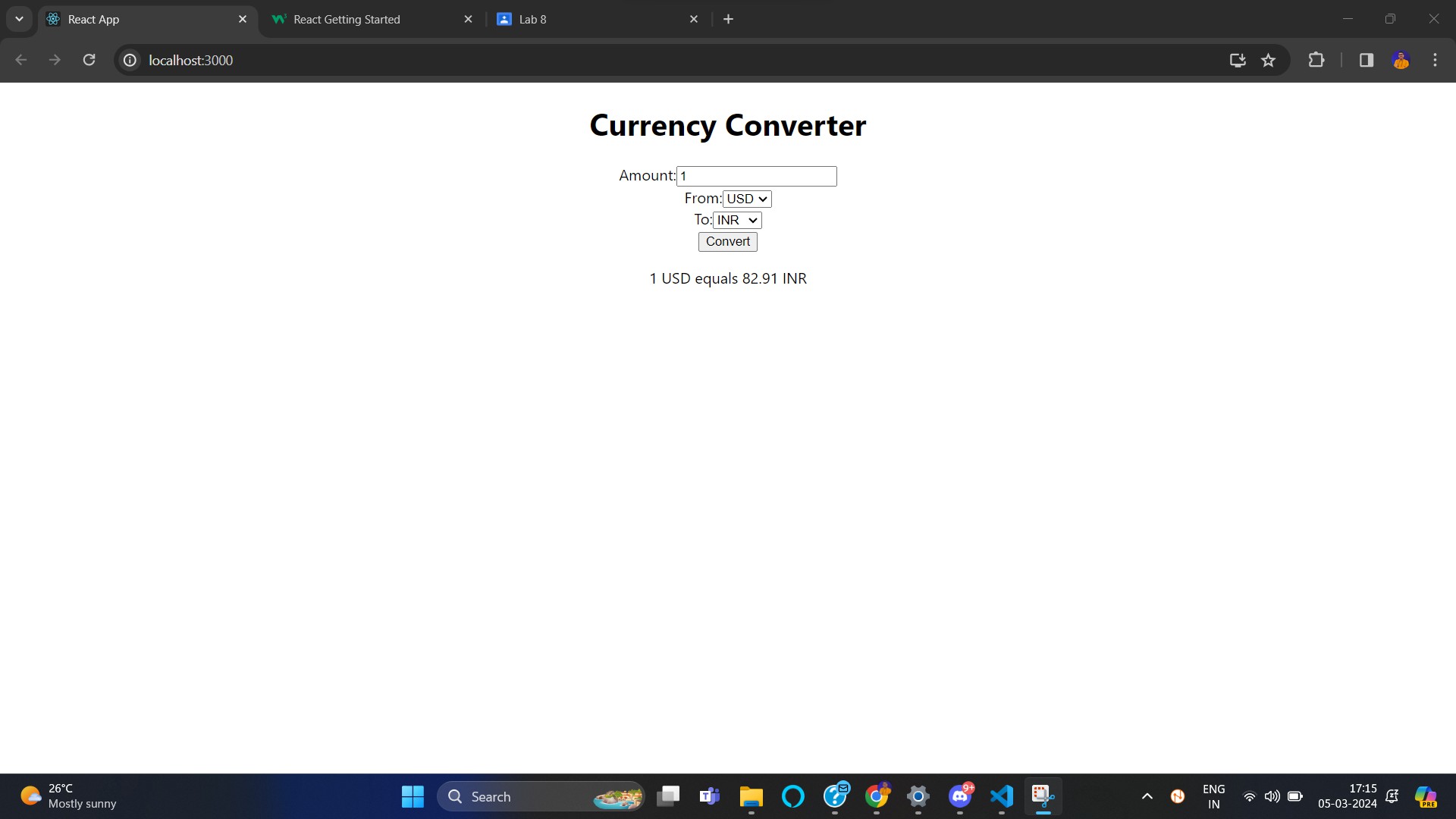
)}

</div>

);

}

export default App;



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, useRef } from 'react'; import './App.css';

function App() {

const [time, setTime] = useState(0);

const [isRunning, setIsRunning] = useState(false); const intervalRef = useRef(null);

const handleStart = () => { setIsRunning(true);

intervalRef.current = setInterval(() => { setTime((prevTime) => prevTime + 1);

}, 1000);

};

const handlePause = () => { clearInterval(intervalRef.current); setIsRunning(false);

};

const handleReset = () => { clearInterval(intervalRef.current); setTime(0);

setIsRunning(false);

};

const formatTime = (timeInSeconds) => {

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

return (

String(minutes).padStart(2, '0') + ':' + String(seconds).padStart(2, '0')

);

};

return (

<div className="App">

<h1>Stopwatch</h1>

<div className="timer">{formatTime(time)}</div>

<div className="controls">

{!isRunning ? (

<button onClick={handleStart}>Start</button>

) : (

<button onClick={handlePause}>Pause</button>

)}

<button onClick={handleReset}>Reset</button>

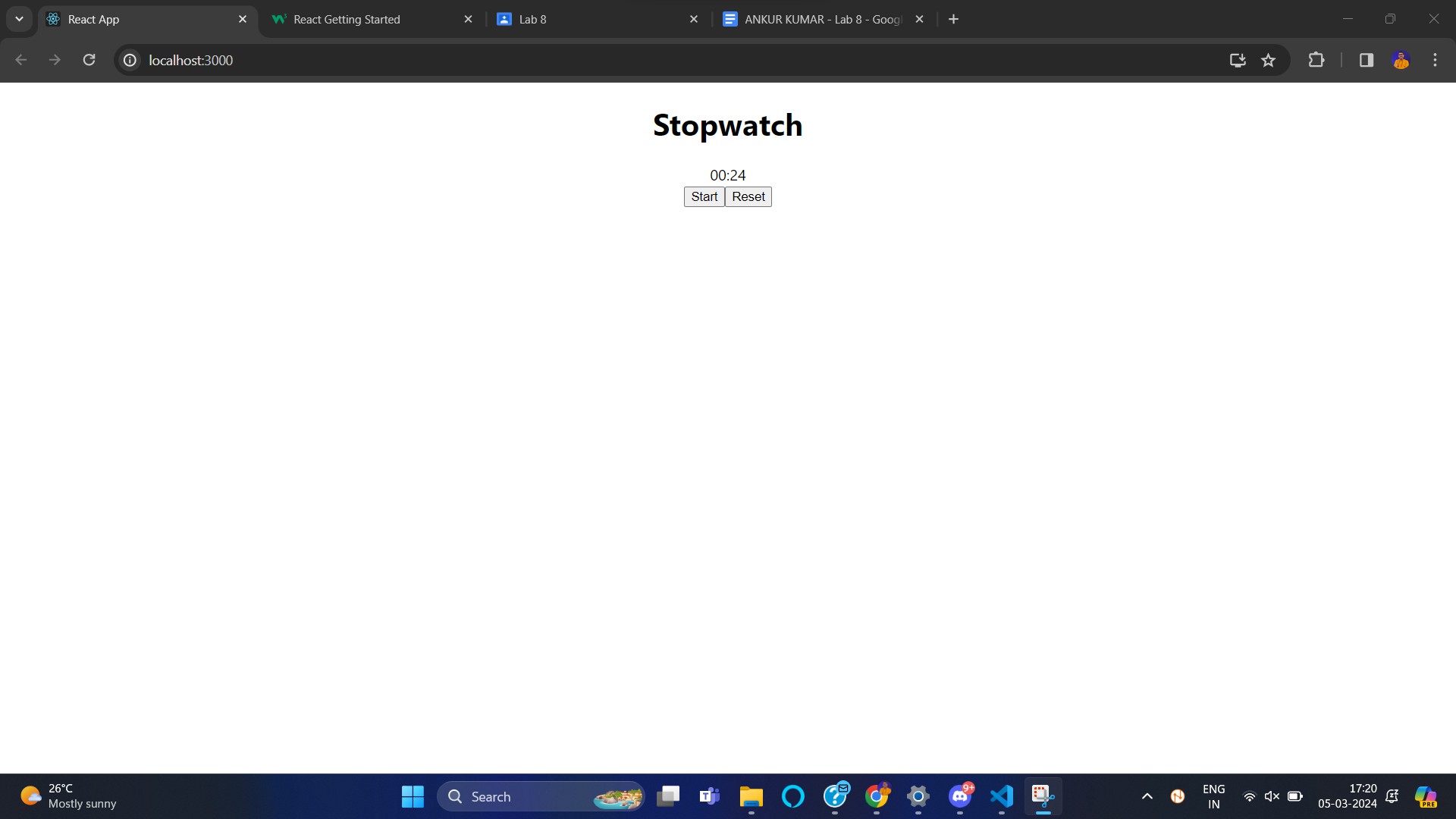
</div>

</div>

);

}

export default App;



**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('');

const [messages, setMessages] = useState([]);

useEffect(() => { fetchConversations();

}, []);

useEffect(() => {

if (selectedConversation) { fetchMessages(selectedConversation.id);

}

}, [selectedConversation]);

const fetchConversations = () => { const mockConversations = [

{ id: 1, name: 'Friend 1' },

{ id: 2, name: 'Friend 2' },

];

setConversations(mockConversations);

};

const fetchMessages = (conversationId) => { const mockMessages = [

{ id: 1, text: 'Hello!', sender: 'Friend 1', timestamp: new Date()

},

{ id: 2, text: 'Hi there!', sender: 'You', timestamp: new Date() },

];

setMessages(mockMessages);

};

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

};

const handleMessageSend = () => {

const message = { id: messages.length + 1, text: newMessage, sender: 'You', timestamp: new Date() };

setMessages([message, ...messages]); setNewMessage('');

};

return (

<div className="App">

<div className="sidebar">

<h2>Conversations</h2>

<ul>

{conversations.map((conversation) => (

<li key={conversation.id} onClick={() => handleConversationClick(conversation)}>

{conversation.name}

</li>

))}

</ul>

</div>

<div className="chat">

<h2>Chat</h2>

{selectedConversation && (

<div>

<h3>{selectedConversation.name}</h3>

<div className="messages">

{messages.map((message) => (

<div key={message.id} className={message.sender === 'You'

? 'sent' : 'received'}>

<p>{message.text}</p>

<span>{message.sender} -

{message.timestamp.toLocaleString()}</span>

</div>

))}

</div>

<div className="message-input">

<input type="text" value={newMessage} onChange={(e) => setNewMessage(e.target.value)} />

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

</div>

</div>

)}

</div>

</div>

);

}

export default App;

