WEB-TECH LAB-8

Mayank Bharti

Roll no- 22CS3039

Branch-CSE

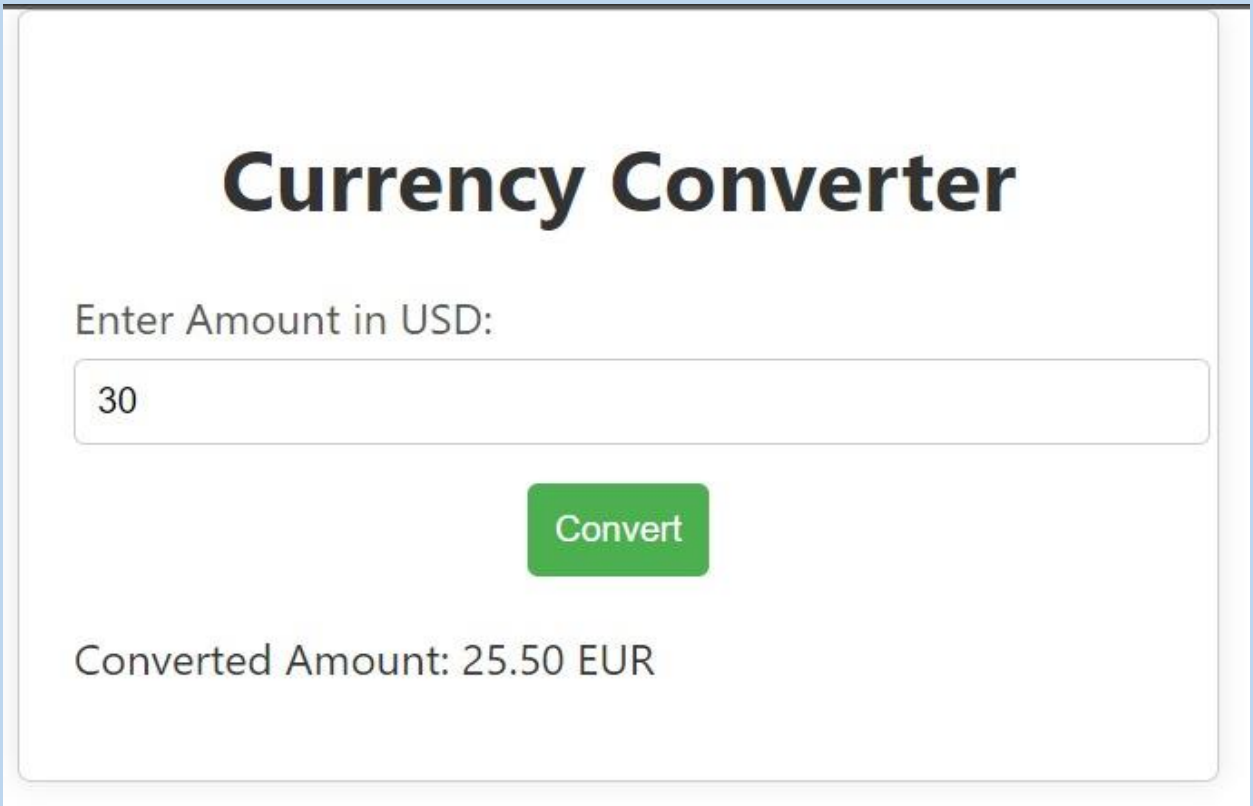
T1. Develop a currency converter application that allows users to input an amount in one

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.

OUTPUT:



JS FILE:

```
import React, { useState } from 'react';
import './App.css'; // Import a separate CSS file for styling (create this
file in the same directory as your component)

const CurrencyConverter = () => {
  const [amount, setAmount] = useState('');
  const [convertedAmount, setConvertedAmount] = useState('');
  const exchangeRate = 0.85; // 1 USD = 0.85 EUR (Replace with actual exchange
rate)

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

  const convertCurrency = () => {
    const result = parseFloat(amount) * exchangeRate;
    setConvertedAmount(result.toFixed(2));
  };

  return (
    <div className="currency-converter-container">
      <h1 className="converter-title">Currency Converter</h1>
      <div className="input-container">
        <label htmlFor="amount" className="label-text">
          Enter Amount in USD:
        </label>
        <input
          type="number"
          id="amount"
          value={amount}
          onChange={handleAmountChange}
          className="input-field"
        />
      </div>
      <div className="button-container">
        <button onClick={convertCurrency} className="convert-button">
          Convert
        </button>
      </div>
      {convertedAmount && (
        <div className="result-container">
          <p className="result-text">Converted Amount: {convertedAmount}
EUR</p>
        </div>
      )}
    </div>
```

```
  );
};

function App() {
  return (
    <div className="app-container">
      <CurrencyConverter />
    </div>
  );
}

export default App;
```

CSS FILE:

```
.currency-converter-container {
    max-width: 400px;
    margin: auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }

  .converter-title {
    text-align: center;
    color: #333;
  }

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

  .label-text {
    display: block;
    margin-bottom: 5px;
    color: #555;
  }

  .input-field {
    width: 100%;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 4px;
  }

  .button-container {
    text-align: center;
```

```
}

.convert-button {
  padding: 10px;
  background-color: #4caf50;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.result-container {
  margin-top: 20px;
}

.result-text {
  color: #333;
}
```

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.

OUTPUT:

CSS FILE:

```
.stopwatch-container {
    max-width: 300px;
    margin: auto;
    text-align: center;
  }

  .timer {
    font-size: 2em;
    margin: 10px 0;
  }

  .controls button {
    font-size: 1em;
    margin: 5px;
```

```
    padding: 10px;
    cursor: pointer;
  }

  .controls button:disabled {
    cursor: not-allowed;
  }
```

JS FILE:

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

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

  const startTimer = () => {
    if (!isRunning) {
      timerRef.current = setInterval(() => {
        setTime((prevTime) => prevTime + 1);
      }, 1000);
      setIsRunning(true);
    }
  };

  const pauseTimer = () => {
    clearInterval(timerRef.current);
    setIsRunning(false);
  };

  const resetTimer = () => {
    clearInterval(timerRef.current);
    setTime(0);
    setIsRunning(false);
  };

  return (
    <div className="stopwatch-container">
      <h1>Stopwatch</h1>
      <div className="timer">{formatTime(time)}</div>
      <div className="controls">
        <button onClick={startTimer} disabled={isRunning}>
          Start
        </button>
```

```
        <button onClick={pauseTimer} disabled={!isRunning}>
          Pause
        </button>
        <button onClick={resetTimer}>Reset</button>
      </div>
    </div>
  );
}

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

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.

JS FILE:

```
// src/App.js
import React, { useState, useEffect } from 'react';
import { auth, firestore } from './firebase';
import Conversations from './components/Conversations';
import Chat from './components/Chat';
import { useCollectionData } from 'react-firebase-hooks/firestore';

const App = () => {
  const [user, setUser] = useState(null);
  const [selectedConversation, setSelectedConversation] = useState(null);

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged((user) => {
```

```
      setUser(user);
    });

    return () => unsubscribe();
  }, []);

  const conversationsRef = firestore.collection('conversations');
  const [conversations] = useCollectionData(conversationsRef, { idField: 'id'
});

  const messagesRef = selectedConversation
    ? conversationsRef.doc(selectedConversation.id).collection('messages')
    : null;
  const [messages] = useCollectionData(messagesRef, { idField: 'id' });

  const sendMessage = () => {
    // Implement message sending logic
  };

  return (
    <div>
      {user ? (
        <>
          <button onClick={() => auth.signOut()}>Sign Out</button>
          <Conversations conversations={conversations}
onSelectConversation={setSelectedConversation} />
          {selectedConversation && <Chat messages={messages}
sendMessage={sendMessage} />}
        </>
      ) : (
        <button onClick={() => auth.signInAnonymously()}>Sign In
Anonymously</button>
      )}
    </div>
  );
};

export default App;
```

```
// src/firebase.js
import firebase from 'firebase/app';
import 'firebase/auth';
import 'firebase/firestore';

const firebaseConfig = {
  // Your Firebase Config Object
};
```

```
firebase.initializeApp(firebaseConfig);

export const auth = firebase.auth();
export const firestore = firebase.firestore();
```

```
// src/components/Chat.js
import React from 'react';

const Chat = ({ messages, sendMessage }) => {
  return (
    <div>
      <h2>Chat</h2>
      <div>
        {messages.map((message) => (
          <div key={message.id}>
            <strong>{message.sender}:</strong> {message.text}
          </div>
        ))}
      </div>
      <div>
        <input type="text" placeholder="Type your message" />
        <button onClick={sendMessage}>Send</button>
      </div>
    </div>
  );
};

export default Chat;
```

```
// src/components/Conversations.js
import React from 'react';

const Conversations = ({ conversations, onSelectConversation }) => {
  return (
    <div>
      <h2>Conversations</h2>
      <ul>
        {conversations.map((conversation) => (
          <li key={conversation.id} onClick={() =>
onSelectConversation(conversation)}>
            {conversation.name}
          </li>
        ))}
      </ul>
    </div>
```

```
  );
};

export default Conversations;
```

OUTPUT: