WEB TECHNOLOGY LAB-8

Lavanya Singh

22it3023

T1.

REACT

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


const CurrencyConverter = () => {
  // State variables
  const [amount, setAmount] = useState('');
  const [fromCurrency, setFromCurrency] = useState('USD');
  const [toCurrency, setToCurrency] = useState('EUR');
  const exchangeRate = 0.85; // Hard-coded exchange rate (1 USD = 0.85 EUR)

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

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

  const handleToCurrencyChange = (e) => {
```

```
  setToCurrency(e.target.value);

};
// Conversion calculation

const convertedAmount = parseFloat(amount) * exchangeRate;



return (

  <div>

    <h1>Currency Converter</h1>

    <div>

      <label>Amount:</label>

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

    </div>

    <div>

      <label>From Currency:</label>

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

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

        <option value="EUR">EUR</option>

        {/* Add more currencies as needed */}

      </select>

    </div>

    <div>

      <label>To Currency:</label>

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

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

        <option value="EUR">EUR</option>

        {/* Add more currencies as needed */}

      </select>

    </div>
    <div>
```

```
        <h2>Converted Amount:</h2>
        <p>{convertedAmount.toFixed(2)}</p>
      </div>
    </div>
  );
};

export default CurrencyConverter;
```

VUE:

```
<template>
  <div class="currency-converter">
    <h1>Currency Converter</h1>
    <div>
      <label for="amount">Amount:</label>
      <input type="number" id="amount" v-model="amount" @input="convertCurrency" />
    </div>
    <div>
      <label for="fromCurrency">From Currency:</label>
      <select id="fromCurrency" v-model="fromCurrency" @change="convertCurrency">
```

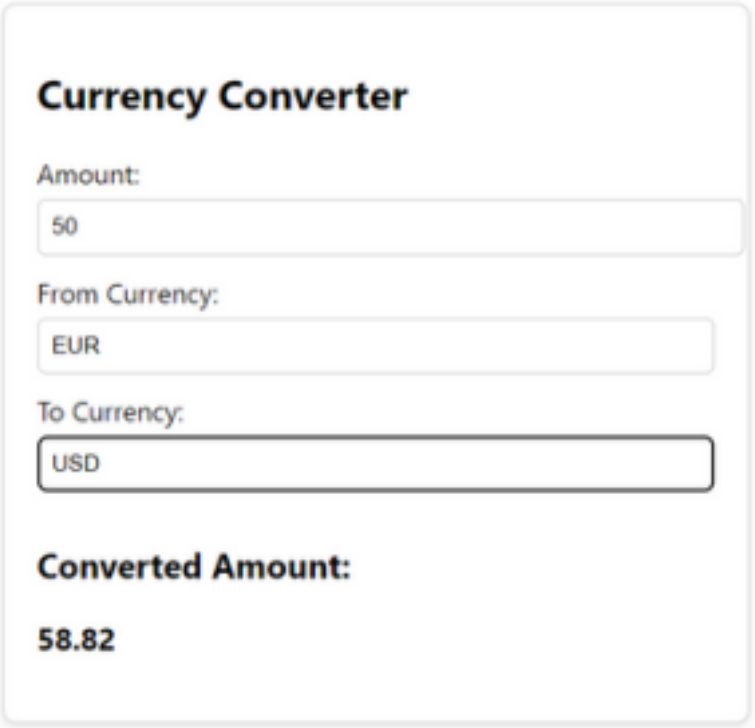
```
        <option value="USD">USD</option>
        <option value="EUR">EUR</option>
        <!-- Add more currencies as needed -->
      </select>
    </div>
    <div>
      <label for="toCurrency">To Currency:</label>
      <select id="toCurrency" v-model="toCurrency" @change="convertCurrency">
        <option value="USD">USD</option>
        <option value="EUR">EUR</option>
        <!-- Add more currencies as needed -->
      </select>
    </div>
    <div>
      <h2>Converted Amount:</h2>
      <p>{{ convertedAmount.toFixed(2) }}</p>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      amount: 0,
      fromCurrency: 'USD',
      toCurrency: 'EUR',
      exchangeRates: {
        USD: 1, // Assuming 1 USD = 1 unit of other currencies for simplicity
```

```
        EUR: 0.85,
        // Add more exchange rates as needed
      },
      convertedAmount: 0,
    };
  },
 methods: {
  convertCurrency() {
    const fromRate = this.exchangeRates[this.fromCurrency];
    const toRate = this.exchangeRates[this.toCurrency];
    const convertedValue = this.amount * (toRate / fromRate);
    this.convertedAmount = Number(convertedValue.toFixed(2)); // Convert to number and round to 2 decimal places
  },
},
};
</script>

<style scoped>
.currency-converter {
  max-width: 400px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 8px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

h1 {
```

```
  font-size: 24px;

  margin-bottom: 20px;

}


label {

  display: block;

  margin-bottom: 5px;

}


input[type='number'],

select {

  width: 100%;

  padding: 8px;

  margin-bottom: 10px;

  border: 1px solid #ccc;
  border-radius: 5px;

}


select {

  appearance: none;

  -webkit-appearance: none;

  background-image: url('data:image/svg+xml;utf8,<svg fill="#000000"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24px" height="24px"><path d="M7 10l5
5 5-5z" /></svg>');

  background-repeat: no-repeat;

  background-position: right 8px center;

}


h2 {

  font-size: 20px;
```

```
  margin-top: 20px;
}

p {
  font-size: 18px;
  font-weight: bold;
}
</style>
```

OUTPUT:



T2.

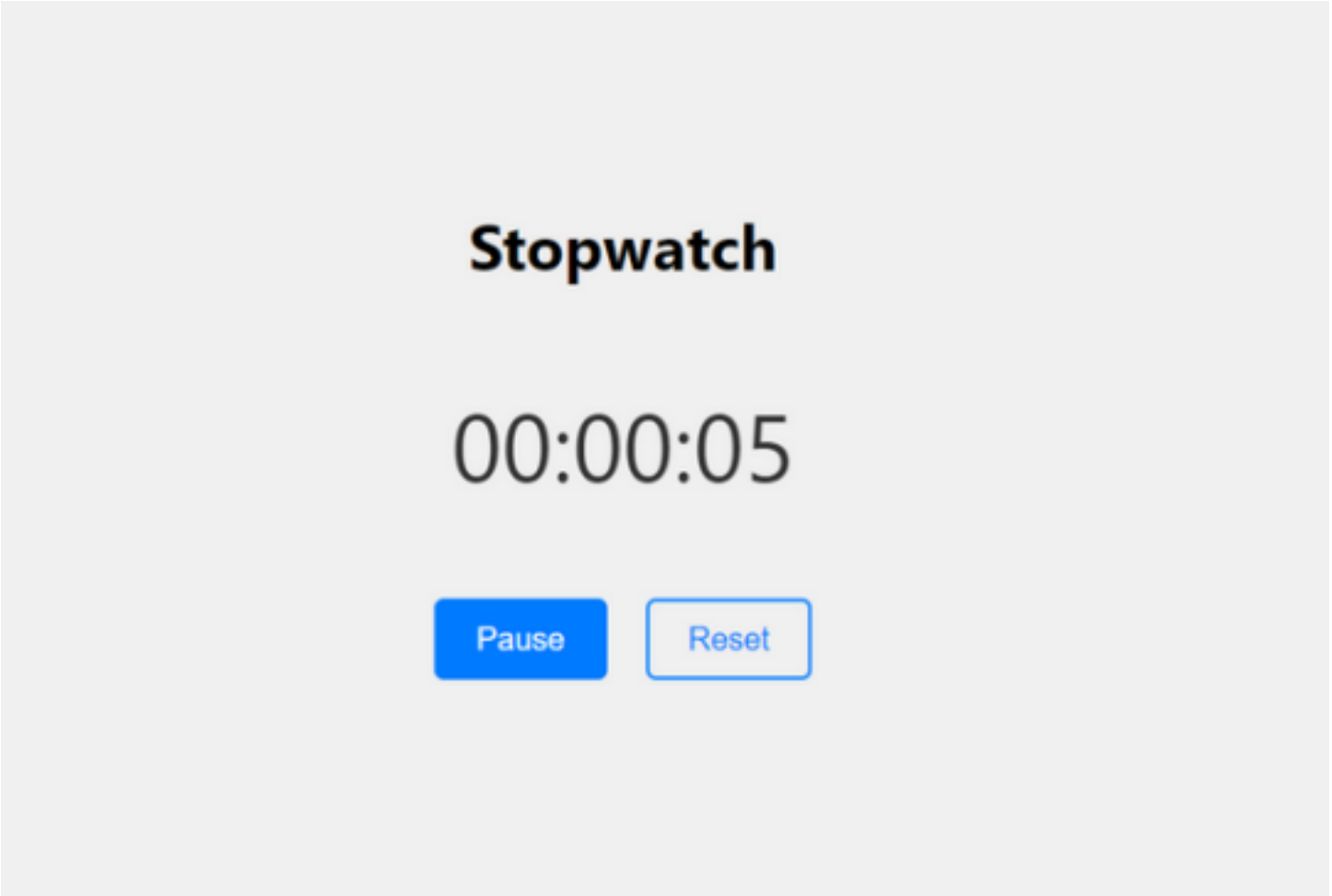
React:

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

const Stopwatch = () => {
```

```
const [isRunning, setIsRunning] = useState(false);

const [timer, setTimer] = useState(0);


useEffect(() => {

  let intervalId;


  if (isRunning) {

    intervalId = setInterval(() => {
      setTimer((prevTimer) => prevTimer + 1);

    }, 1000);

  }


  return () => clearInterval(intervalId);

}, [isRunning]);


const handleStart = () => {

  setIsRunning(true);

};


const handlePause = () => {

  setIsRunning(false);

};


const handleReset = () => {

  setIsRunning(false);

  setTimer(0);

};


const formatTime = (timeInSeconds) => {
```

```
    const hours = Math.floor(timeInSeconds / 3600); const
    minutes = Math.floor((timeInSeconds % 3600) / 60); const
    seconds = timeInSeconds % 60;

    const formattedHours = String(hours).padStart(2, '0');
    const formattedMinutes = String(minutes).padStart(2, '0');
    const formattedSeconds = String(seconds).padStart(2, '0');
    return `${formattedHours}:${formattedMinutes}:${formattedSeconds}`;
 };

  return (
    <div className="container">
  <h1>Stopwatch</h1>
  <div className="timer">
    <p>{formatTime(timer)}</p>
  </div>
  <div className="buttons">
    {!isRunning ? (
      <button onClick={handleStart}>Start</button>
    ) : (
      <button onClick={handlePause}>Pause</button>
    )}
    <button onClick={handleReset}>Reset</button>
  </div>
</div>

  );
};
```

```
export default Stopwatch;
```



Vue:

```
<template>

  <div class="stopwatch">

    <h1>Stopwatch</h1>

    <div class="timer">{{ formatTime }}</div>

    <div class="buttons">

      <button @click="startStopwatch" v-if="!isRunning">Start</button>

      <button @click="pauseStopwatch" v-if="isRunning">Pause</button>

      <button @click="resetStopwatch">Reset</button>

    </div>

  </div>

</template>
<script>

export default {
```

```
data() {
  return {
    isRunning: false,
    startTime: null,
    currentTime: 0,
    timerInterval: null,
  };
},
computed: {
  formatTime() {
    const minutes = Math.floor(this.currentTime / 60);
    const seconds = this.currentTime % 60;
    return `${String(minutes).padStart(2, '0')}:${String(seconds).padStart(2, '0')}`;
  },
},
methods: {
  startStopwatch() {
    this.isRunning = true;
    this.startTime = Date.now() - this.currentTime * 1000;
    this.timerInterval = setInterval(() => {
      this.currentTime = Math.floor((Date.now() - this.startTime) / 1000);
    }, 1000);
  },
  pauseStopwatch() {
    this.isRunning = false;
    clearInterval(this.timerInterval);
  },
  resetStopwatch() {
    this.isRunning = false;
```

```
      clearInterval(this.timerInterval);
      this.currentTime = 0;
    },
  },
  destroyed() {
    clearInterval(this.timerInterval); // Cleanup interval when component is destroyed
  },
};
</script>

<style>
.stopwatch {
  text-align: center;
  max-width: 300px;
  margin: 0 auto;
}

.timer {
  font-size: 36px;
  margin-bottom: 20px;
}

.buttons button {
  margin-right: 10px;
  padding: 8px 16px;
  font-size: 16px;
  border: 1px solid #333;
  border-radius: 5px;
  background-color: #fff;
```

```
  cursor: pointer;
  transition: background-color 0.3s ease, color 0.3s ease, border-color 0.3s ease;
}

.buttons button:hover {
  background-color: #333;
  color: #fff;
  border-color: #333;
}
</style>
```

OUTPUT: