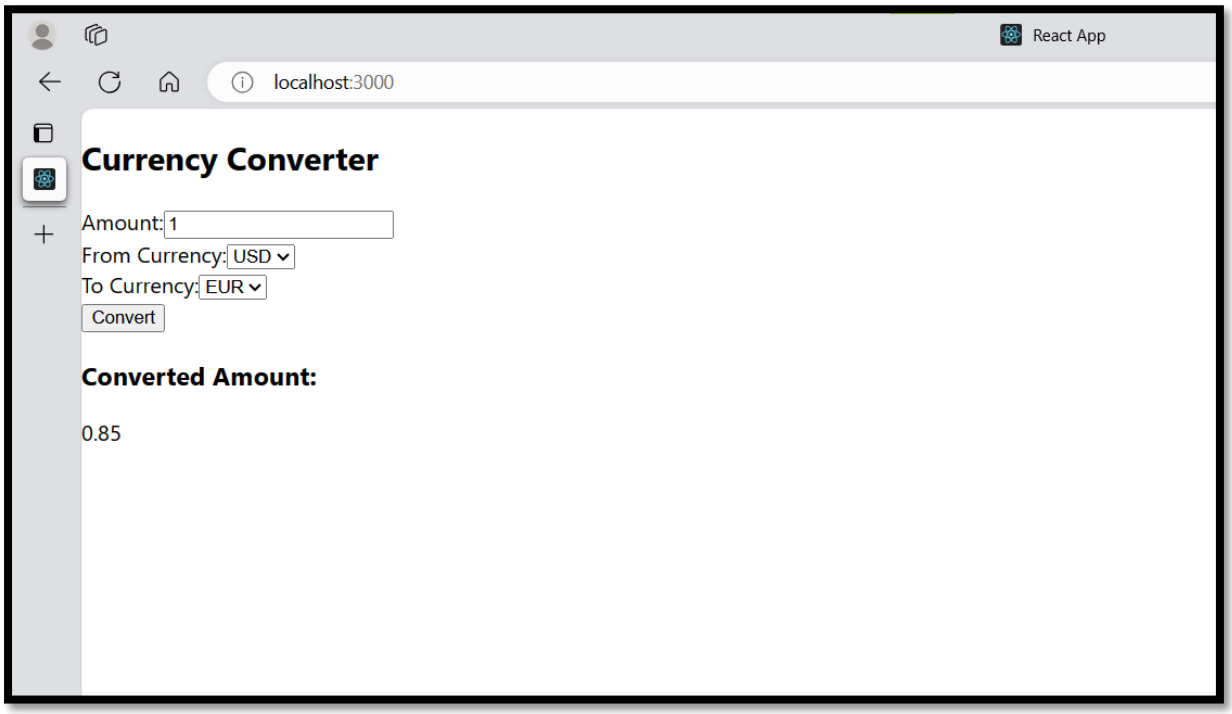
Raj Pathak

Roll no – 22cs3045

T1. Currency converter

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

const CurrencyConverter = () => {
  const [amount, setAmount] = useState('');
  const [fromCurrency, setFromCurrency] = useState('USD');
  const [toCurrency, setToCurrency] = useState('EUR');
  const [convertedAmount, setConvertedAmount] = useState('');

  const exchangeRate = 0.85;

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

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

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

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

  return (
    <div>
      <h2>Currency Converter</h2>
      <div>
        <label htmlFor="amount">Amount:</label>
        <input type="number" id="amount" value={amount} onChange={handleAmountChange} />
      </div>
      <div>
        <label htmlFor="fromCurrency">From Currency:</label>
        <select id="fromCurrency" value={fromCurrency} onChange={handleFromCurrencyChange}>
          <option value="USD">USD</option>
        </select>
      </div>
      <div>
        <label htmlFor="toCurrency">To Currency:</label>
```

```
          <select id="toCurrency" value={toCurrency} onChange={handleToCurrencyChange}>
            <option value="EUR">EUR</option>
          </select>
        </div>
        <button onClick={convertCurrency}>Convert</button>
        <div>
          <h3>Converted Amount:</h3>
          <p>{convertedAmount}</p>
        </div>
      </div>
    );
};

export default CurrencyConverter;
```



T2. Stopwatch application

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

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

  useEffect(() => {
    let intervalId;
    if (isRunning) {
      intervalId = setInterval(() => {
        setTime((prevTime) => prevTime + 1);
```

```
    }, 1000);
  } else {
    clearInterval(intervalId);
  }

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

const startStopwatch = () => {
  setIsRunning(true);
};

const pauseStopwatch = () => {
  setIsRunning(false);
};

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

const formatTime = (seconds) => {
  const hours = Math.floor(seconds / 3600);
  const minutes = Math.floor((seconds % 3600) / 60);
  const remainingSeconds = seconds % 60;

  return `${hours.toString().padStart(2, '0')}:${minutes
    .toString()
    .padStart(2, '0')}:${remainingSeconds.toString().padStart(2, '0')}`;
};

return (
  <div>
    <h1>Stopwatch</h1>
    <div>
      <p>{formatTime(time)}</p>
    </div>
    <div>
      {!isRunning ? (
        <button onClick={startStopwatch}>Start</button>
      ) : (
        <button onClick={pauseStopwatch}>Pause</button>
      )}
      <button onClick={resetStopwatch}>Reset</button>
    </div>
  </div>
);
};
```

```
export default Stopwatch;
```