WEB TECHNOLOGY

LAB 8

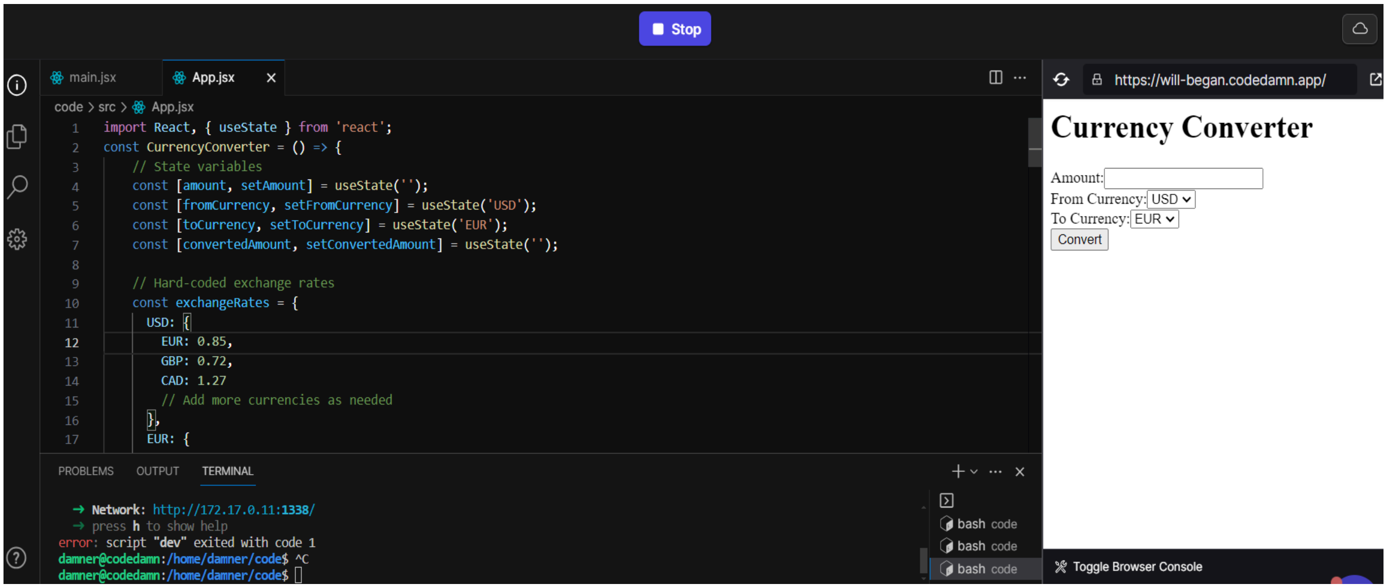
Name : Gourab Chowdhury
Roll No : 22cs3069

Task 1 :

```
import React, { useState } from 'react';
const CurrencyConverter = () => {
   // State variables
   const [amount, setAmount] = useState('');
   const [fromCurrency, setFromCurrency] = useState('USD');
   const [toCurrency, setToCurrency] = useState('EUR');
   const [convertedAmount, setConvertedAmount] = useState('');
    // Hard-coded exchange rates
   const exchangeRates = {
     USD: {
       EUR: 0.85,
       GBP: 0.72,
       CAD: 1.27
       // Add more currencies as needed
     },
     EUR: {
       USD: 1.18,
       GBP: 0.85,
       CAD: 1.48
       // Add more currencies as needed
     },
     GBP: {
       USD: 1.39,
       EUR: 1.18,
       CAD: 1.74
       // Add more currencies as needed
     },
     CAD: {
       USD: 0.79,
       EUR: 0.68,
       GBP: 0.57
       // Add more currencies as needed
     }
   };
    // Function to handle amount change
   const handleAmountChange = (event) => {
     const value = event.target.value;
     setAmount(value);
   };
    // Function to handle from currency change
   const handleFromCurrencyChange = (event) => {
     const value = event.target.value;
     setFromCurrency(value);
   };
    // Function to handle to currency change
   const handleToCurrencyChange = (event) => {
     const value = event.target.value;
     setToCurrency(value);
   };
    // Function to handle conversion
   const handleConvert = () => {
     const exchangeRate = exchangeRates[fromCurrency][toCurrency];
     const result = parseFloat(amount) * exchangeRate;
     setConvertedAmount(result.toFixed(2));
   };
    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>
```

```
          <option value="GBP">GBP</option>
          <option value="CAD">CAD</option>
        </select>
      </div>
      <div>
        <label>To Currency:</label>
        <select value={toCurrency} onChange={handleToCurrencyChange}>
          <option value="USD">USD</option>
          <option value="EUR">EUR</option>
          <option value="GBP">GBP</option>
          <option value="CAD">CAD</option>
        </select>
      </div>
      <button onClick={handleConvert}>Convert</button>
      <div>
        {convertedAmount && (
          <p>
            Converted Amount: {convertedAmount} {toCurrency}
          </p>
        )}
      </div>
    </div>
  );
};
 export default CurrencyConverter;
```

Output:



```jsx
import React, { useState } from 'react';
const CurrencyConverter = () => {
    // State variables
    const [amount, setAmount] = useState('');
    const [fromCurrency, setFromCurrency] = useState('USD');
    const [toCurrency, setToCurrency] = useState('EUR');
    const [convertedAmount, setConvertedAmount] = useState('');

    // Hard-coded exchange rates
    const exchangeRates = {
      USD: {
        EUR: 0.85,
        GBP: 0.72,
        CAD: 1.27
        // Add more currencies as needed
      },
      EUR: {
```

```
  → Network: http://172.17.0.11:1338/
  → press h to show help
error: script "dev" exited with code 1
damner@codedamn:/home/damner/code$ ^C
damner@codedamn:/home/damner/code$ 
```

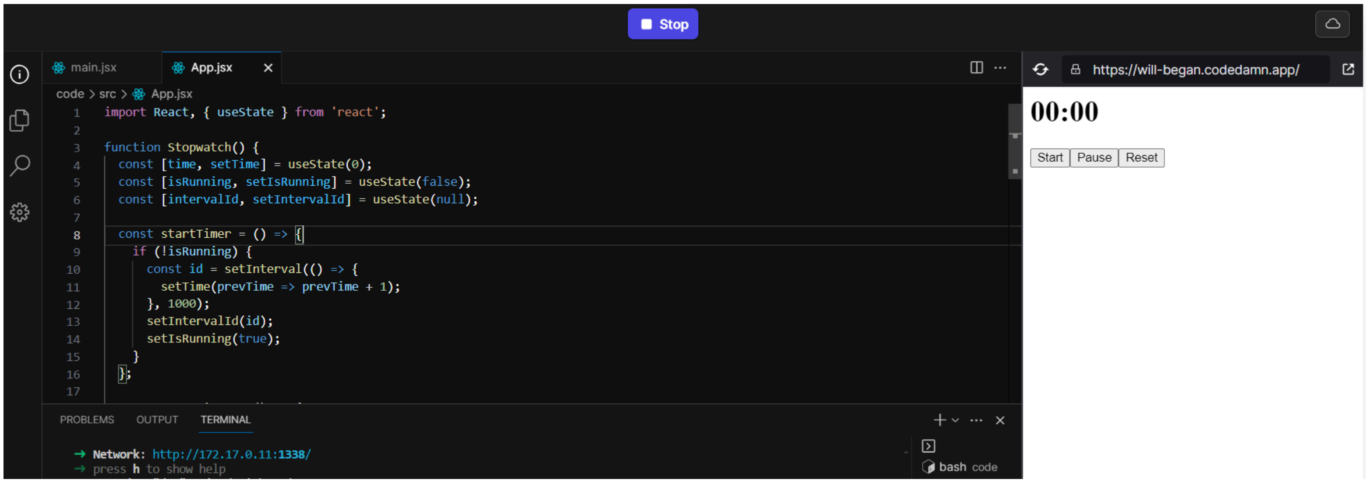
Currency Converter

Amount:
From Currency: USD
To Currency: EUR
Convert

Task2

```
function Stopwatch() {
 const [time, setTime] = useState(0);
 const [isRunning, setIsRunning] = useState(false);
 const [intervalId, setIntervalId] = useState(null);


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


 const pauseTimer = () => {
   clearInterval(intervalId);
   setIsRunning(false);
 };


 const resetTimer = () => {
   clearInterval(intervalId);
   setTime(0);
   setIsRunning(false);
 };


 const formatTime = time => {
   const hours = Math.floor(time / 3600);
   const minutes = Math.floor((time % 3600) / 60);
   const seconds = time % 60;
   return (
     (hours > 0 ? (hours < 10 ? '0' + hours : hours) + ':' : '') +
     (minutes < 10 ? '0' + minutes : minutes) +
     ':' +
     (seconds < 10 ? '0' + seconds : seconds)
   );
 };


 return (
   <div>
     <h1>{formatTime(time)}</h1>
     <button onClick={startTimer}>Start</button>
     <button onClick={pauseTimer}>Pause</button>
     <button onClick={resetTimer}>Reset</button>
   </div>
 );
}


export default Stopwatch;
```

Output



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

function Stopwatch() {
  const [time, setTime] = useState(0);
  const [isRunning, setIsRunning] = useState(false);
  const [intervalId, setIntervalId] = useState(null);

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

```

```
➜ Network: http://172.17.0.11:1338/
➜ press h to show help
```

```
00:00
```

Task 3

```js
import React, { useState, useEffect } from 'react';
import firebase from 'firebase/app';
import 'firebase/auth';
import 'firebase/firestore';


// Initialize Firebase (replace with your own config)
const firebaseConfig = {
 apiKey: 'YOUR_API_KEY',
 authDomain: 'YOUR_AUTH_DOMAIN',
 projectId: 'YOUR_PROJECT_ID',
 storageBucket: 'YOUR_STORAGE_BUCKET',
 messagingSenderId: 'YOUR_MESSAGING_SENDER_ID',
 appId: 'YOUR_APP_ID',
};
```

```
firebase.initializeApp(firebaseConfig);
const db = firebase.firestore();


function App() {
 const [user, setUser] = useState(null);
 const [messages, setMessages] = useState([]);
 const [newMessage, setNewMessage] = useState('');


 useEffect(() => {
   // Listen for authentication state changes
   firebase.auth().onAuthStateChanged((authUser) => {
     if (authUser) {
       setUser(authUser);
     } else {
       setUser(null);
     }
   });


   // Fetch chat messages from Firestore
   const unsubscribe =
db.collection('messages').orderBy('timestamp').onSnapshot((snapshot) => {
     const messageData = snapshot.docs.map((doc) => doc.data());
     setMessages(messageData);
   });


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


 const handleSignIn = async () => {
   try {
     const provider = new firebase.auth.GoogleAuthProvider();
     await firebase.auth().signInWithPopup(provider);
   } catch (error) {
     console.error('Error signing in:', error.message);
   }
```

```
};


const handleSignOut = async () => {
  try {
    await firebase.auth().signOut();
  } catch (error) {
    console.error('Error signing out:', error.message);
  }
};


const handleSendMessage = async () => {
  if (newMessage.trim() !== '') {
    await db.collection('messages').add({
      text: newMessage,
      userId: user.uid,
      timestamp: firebase.firestore.FieldValue.serverTimestamp(),
    });
    setNewMessage('');
  }
};


return (
  <div className="App">
    {user ? (
      <>
        <button onClick={handleSignOut}>Sign Out</button>
        <div>
          {messages.map((message) => (
            <div key={message.timestamp}>
              {message.text}
            </div>
          ))}
        </div>
        <input
          type="text"
          value={newMessage}
          onChange={(e) => setNewMessage(e.target.value)}
```

```
         />
         <button onClick={handleSendMessage}>Send</button>
       </>
     ) : (
       <button onClick={handleSignIn}>Sign In with Google</button>
     )}
   </div>
 );
}


export default App;
```