LAB ASSIGNMENT

NAME : Raksha Kumari

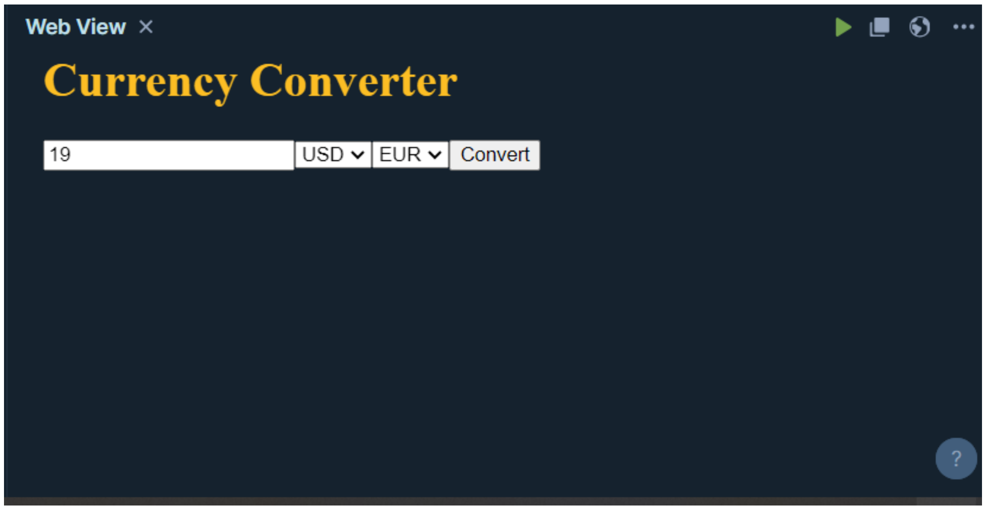
Roll No : 22cs3046

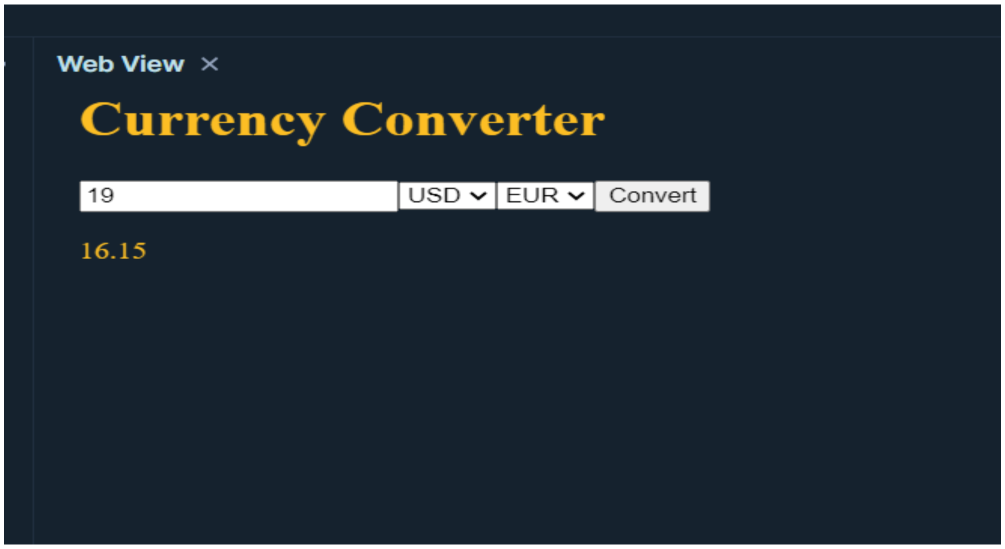
Q1.Develop a currency converter application that allows users to input an amount in one currency and convert it to another. For the sake of this challenge, you can use a hard-coded exchange rate. Take advantage of React state and event handlers to manage the input and conversion calculations.

```
<template>
  <div id="app">
    <h1>Currency Converter</h1>
    <input type="number" v-model="amount" placeholder="Enter amount" />
    <select v-model="fromCurrency">
      <option value="USD">USD</option>
      <option value="EUR">EUR</option>
    </select>
    <select v-model="toCurrency">
      <option value="USD">USD</option>
      <option value="EUR">EUR</option>
    </select>
    <button @click="convertCurrency">Convert</button>
    <p>{{ convertedAmount }}</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      amount: null,
      fromCurrency: 'USD',
      toCurrency: 'EUR',
      exchangeRate: {
        USD: { EUR: 0.85 },
        EUR: { USD: 1.18 },
      },
      convertedAmount: null,
    };
  },
  methods: {
    convertCurrency() {
      this.convertedAmount = this.amount *
this.exchangeRate[this.fromCurrency][this.toCurrency];
    },
  },
};
</script>
```

Output:

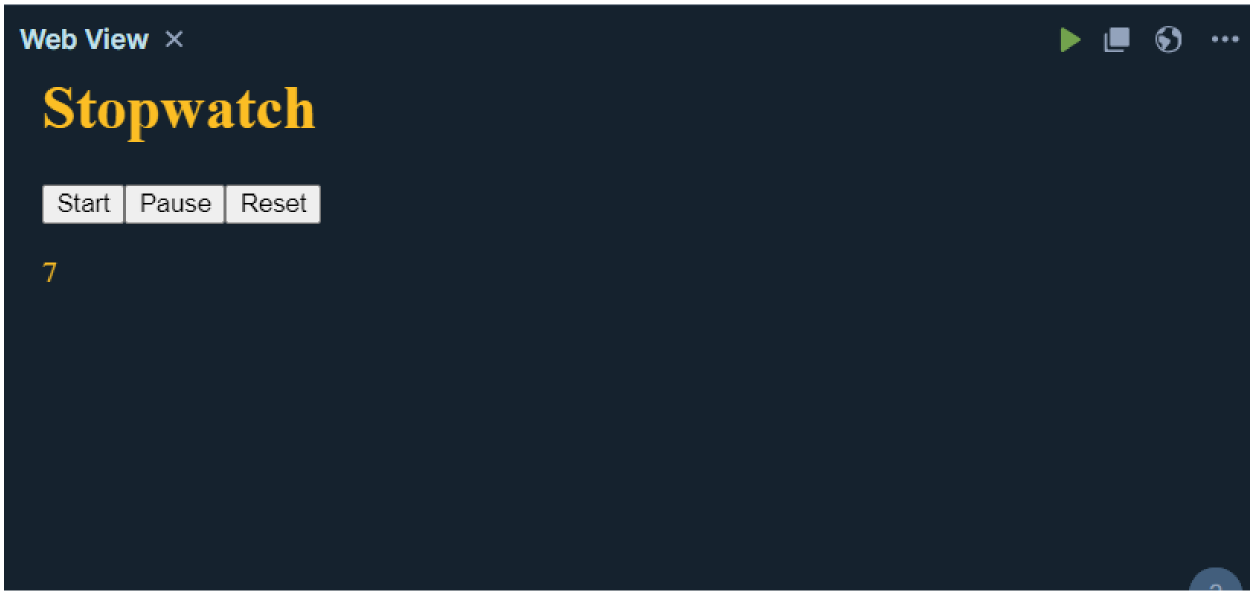




Q2.Create a stopwatch application through which users can start, pause and reset the timer. Use React state, event handlers and the setTimeout or setInterval functions to manage the timer's state and actions.

```
<template>
  <div id="app">
    <h1>Stopwatch</h1>
    <button @click="startStopwatch">Start</button>
    <button @click="pauseStopwatch">Pause</button>
    <button @click="resetStopwatch">Reset</button>
    <p>{{ time }}</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      time: 0,
      intervalId: null,
    };
  },
  methods: {
    startStopwatch() {
      this.intervalId = setInterval(() => {
        this.time++;
      }, 1000);
    },
    pauseStopwatch() {
      clearInterval(this.intervalId);
    },
    resetStopwatch() {
      this.time = 0;
      clearInterval(this.intervalId);
    },
  },
};
</script>
```



Stopwatch

7

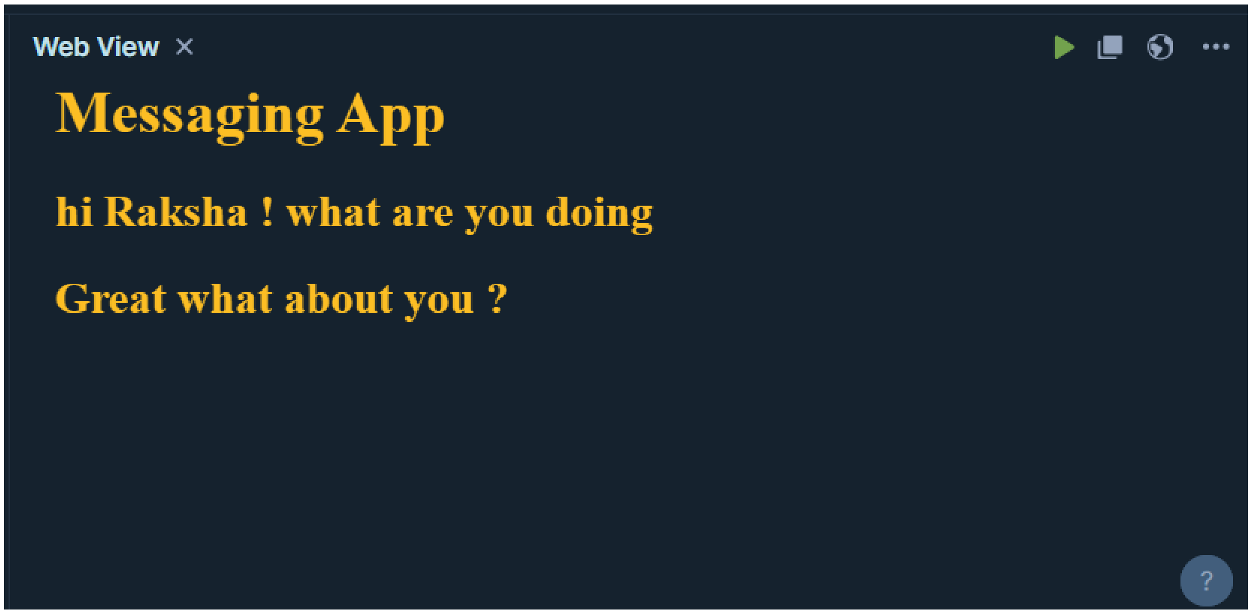
Q3. Develop a messaging application that allows users to send and receive messages in real time. The application should display a list of conversations and allow the user to select a specific conversation to view its messages. The messages should be displayed in a chat interface with the most recent message at the top. Users should be able to send new messages and receive push notifications.

```
<template>
  <div id="app">
    <h1>Messaging App</h1>
    <div v-for="conversation in conversations" :key="conversation.id">
      <h2 @click="selectConversation(conversation.id)">{{ conversation.name }}</h2>
    </div>
    <div v-if="selectedConversation">
```

```
      <div v-for="message in selectedConversation.messages" :key="message.id">
        <p>{{ message.text }}</p>
      </div>
      <input type="text" v-model="newMessage" placeholder="Type a message" />
      <button @click="sendMessage">Send</button>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      conversations: [
        { id: 1, name: 'hi Raksha ! what are you doing ', messages: [] },
        { id: 2, name: 'Great what about you ?', messages: [] },
      ],
      selectedConversation: null,
      newMessage: '',
    };
  },
  methods: {
    selectConversation(id) {
      this.selectedConversation = this.conversations.find(c => c.id === id);
    },
    sendMessage() {
      if (this.newMessage.trim() !== '') {
        this.selectedConversation.messages.push({ id: Date.now(), text: this.newMessage });
        this.newMessage = '';
      }
    },
  },
};
</script>
```

Output:



Messaging App

hi Raksha ! what are you doing

Great what about you ?