Name- Anshu Priya
Branch- CSE-IDD
Roll no. 22cs2020

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.

Code:

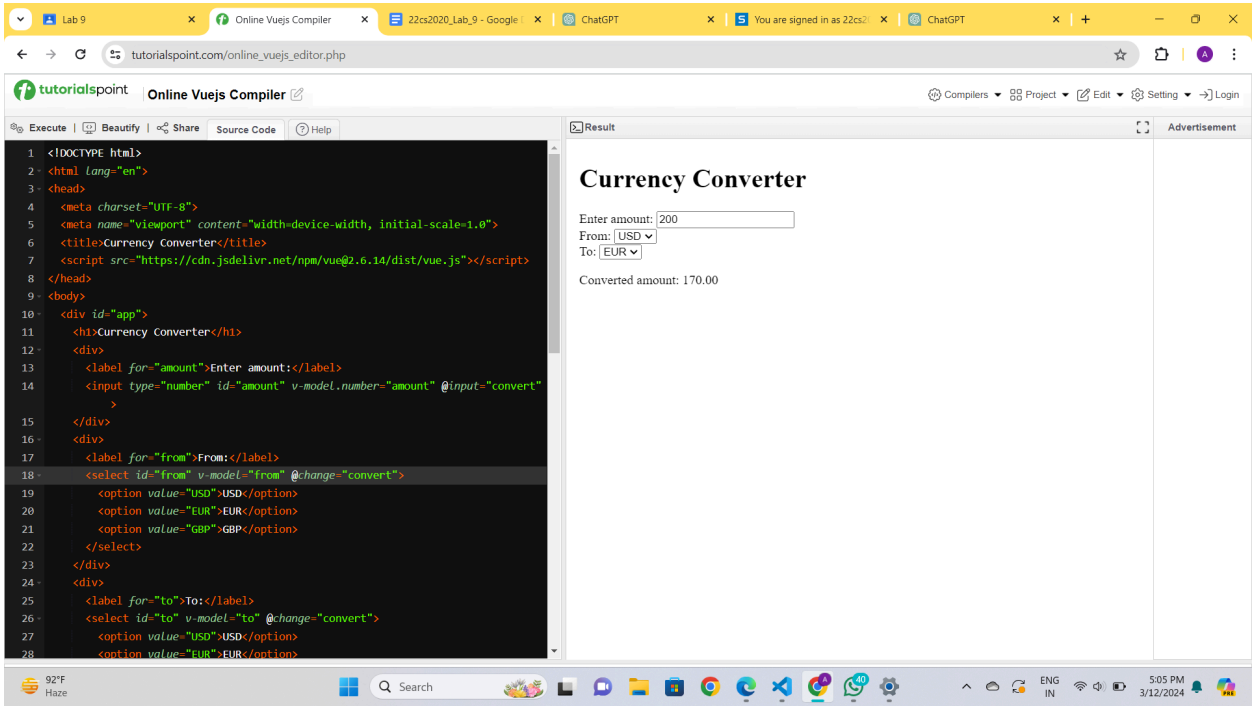
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Currency Converter</title>
  <script
src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
</head>
<body>
  <div id="app">
    <h1>Currency Converter</h1>
    <div>
      <label for="amount">Enter amount:</label>
      <input type="number" id="amount" v-model.number="amount"
@input="convert">
    </div>
    <div>
      <label for="from">From:</label>
      <select id="from" v-model="from" @change="convert">
        <option value="USD">USD</option>
        <option value="EUR">EUR</option>
```

```
      <option value="GBP">GBP</option>
    </select>
  </div>
  <div>
    <label for="to">To:</label>
    <select id="to" v-model="to" @change="convert">
      <option value="USD">USD</option>
      <option value="EUR">EUR</option>
      <option value="GBP">GBP</option>
    </select>
  </div>
  <div>
    <p>Converted amount: {{ convertedAmount }}</p>
  </div>
</div>

<script>
  new Vue({
    el: '#app',
    data: {
      amount: 0,
      from: 'USD',
      to: 'USD',
      exchangeRates: {
        USD: {
          EUR: 0.85,
          GBP: 0.72
        },
        EUR: {
          USD: 1.18,
          GBP: 0.85
        },
        GBP: {
          USD: 1.38,
```

```
            EUR: 1.18
          }
        },
        convertedAmount: 0

      },
      methods: {
        convert() {
          const rate = this.exchangeRates[this.from][this.to];
          this.convertedAmount = (this.amount * rate).toFixed(2);
        }
      }
    });
  </script>
</body>
```



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.

Code:

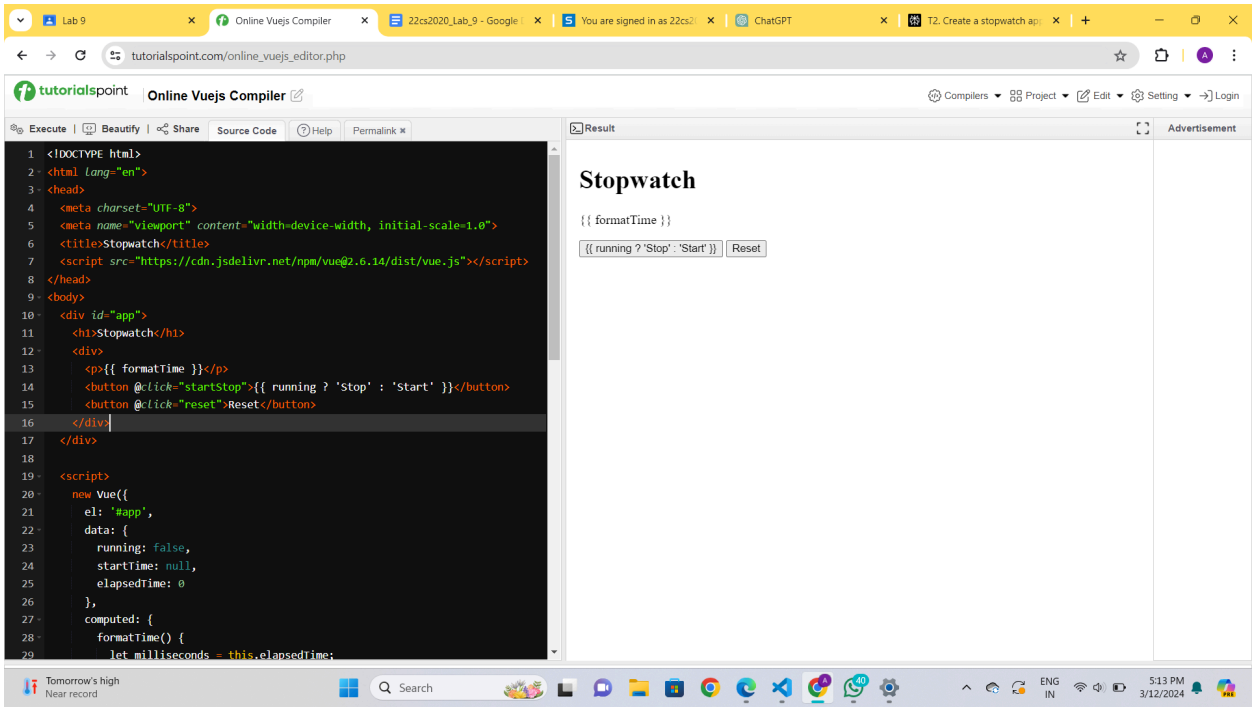
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Stopwatch</title>
  <script
src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
</head>
<body>
  <div id="app">
    <h1>Stopwatch</h1>
    <div>
      <p>{{ formatTime }}</p>
      <button @click="startStop">{{ running ? 'Stop' : 'Start'
}}</button>
      <button @click="reset">Reset</button>
    </div>
  </div>

  <script>
    new Vue({
      el: '#app',
      data: {
        running: false,
        startTime: null,
        elapsedTime: 0
      },
      computed: {
```

```
        formatTime() {
          let milliseconds = this.elapsedTime;
          let seconds = Math.floor(milliseconds / 1000);
          let minutes = Math.floor(seconds / 60);
          seconds = seconds % 60;
          milliseconds = milliseconds % 1000;

          return
${this.pad(minutes)}:${this.pad(seconds)}.${this.pad(milliseconds,
3)};
        }
      },
      methods: {
        startStop() {
          if (this.running) {
            clearInterval(this.timer);
            this.running = false;
          } else {
            this.startTime = Date.now() - this.elapsedTime;
            this.timer = setInterval(() => {
              this.elapsedTime = Date.now() - this.startTime;
            }, 10);
            this.running = true;
          }
        },
        reset() {
          clearInterval(this.timer);
          this.running = false;
          this.elapsedTime = 0;
        },
        pad(num, size = 2) {
          let str = num.toString();
          while (str.length < size) {
            str = '0' + str;
```

```
            }
            return str;
          }
        }
    });
  </script>
</body>
</html>
```



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.

Code:

```
<template>
  <div>
    <div v-for="conversation in conversations" :key="conversation.id"
@click="selectConversation(conversation)">
      {{ conversation.title }}
    </div>

    <div v-if="selectedConversation">
      <h2>{{ selectedConversation.title }}</h2>
      <div v-for="message in selectedConversation.messages"
:key="message.id">
        {{ message.text }}
      </div>

      <input v-model="newMessage" placeholder="Type your message..."
/>
      <button @click="sendMessage">Send</button>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      conversations: [],
      selectedConversation: null,
      newMessage: ''
    };
  },
  methods: {
    selectConversation(conversation) {
      this.selectedConversation = conversation;
```

```
    },
    sendMessage() {
      // Implement logic to send message
    }
  }
};
</script>
```

```
const express = require('express');
const app = express();
const http = require('http').Server(app);
const io = require('socket.io')(http);

let conversations = [
  { id: 1, title: 'Conversation 1', messages: [] },
  { id: 2, title: 'Conversation 2', messages: [] }
];

io.on('connection', (socket) => {
  console.log('A user connected');

  socket.on('sendMessage', (data) => {
    const conversationId = data.conversationId;
    const message = data.message;

    const conversation = conversations.find(conv => conv.id ===
conversationId);
    if (conversation) {
      conversation.messages.push({ text: message });
      io.emit('messageReceived', { conversationId, message });
    }
  });
});
```

```
http.listen(3000, () => {
  console.log('Server running on port 3000');
});
```