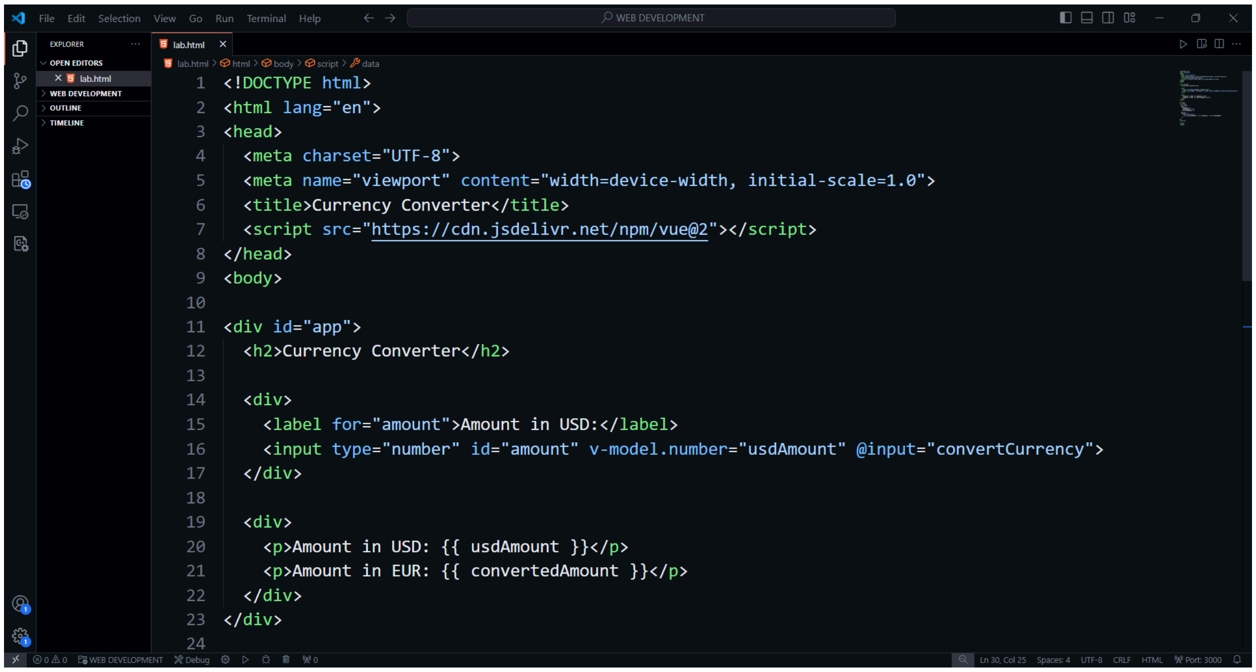
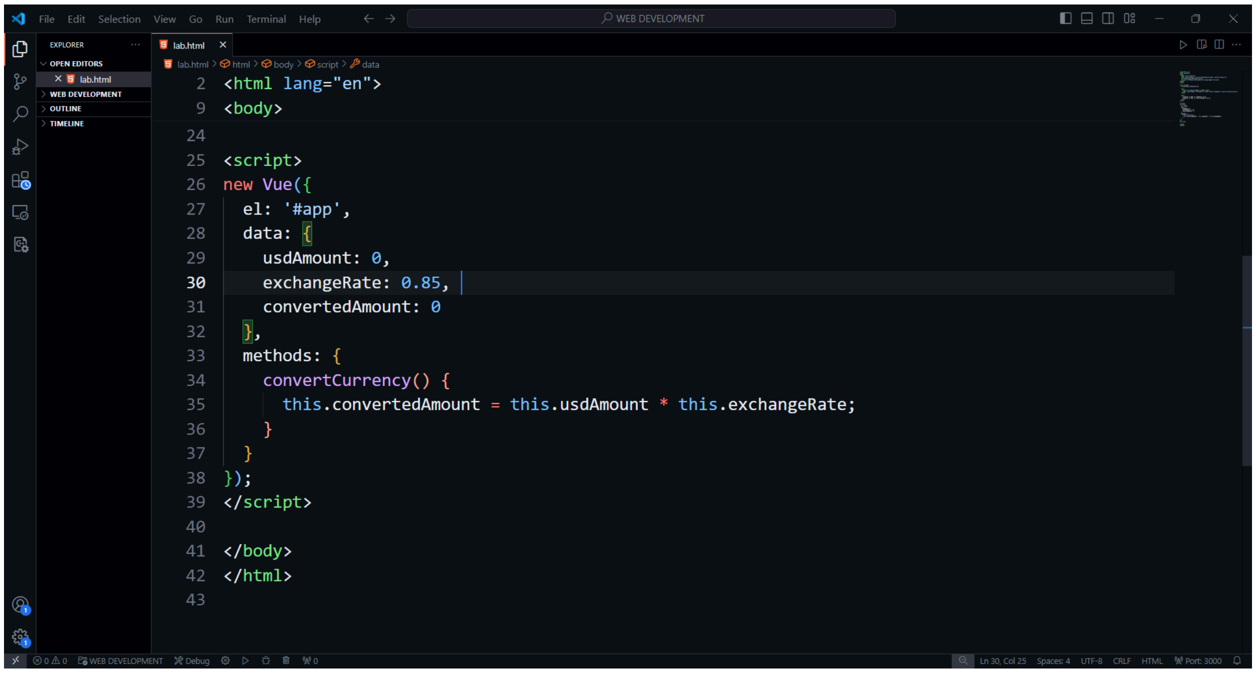
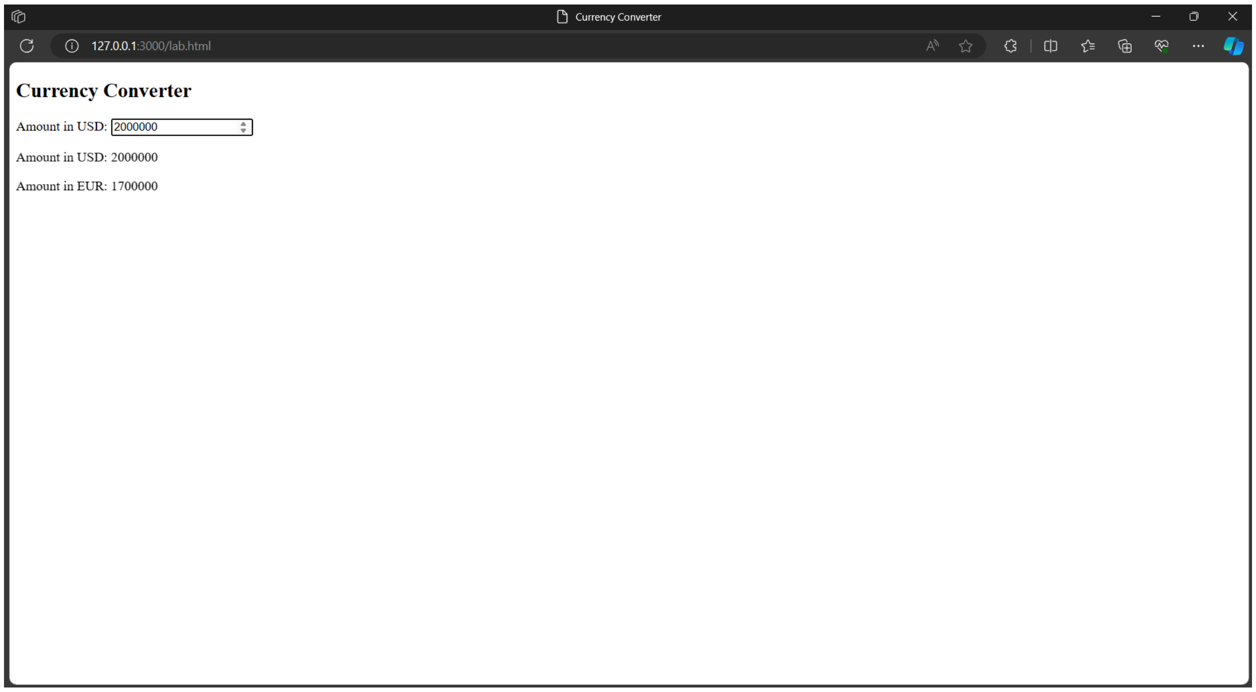
NAME:ARYAN KHAIWAL
ROLL NO.:22CS2029
T1. Develop a currency converter application that allows users to input an amount in one
currency and convert it to another. For the sake of this challenge, you can use a hard-coded
exchange rate. Take advantage of React state and event handlers to manage the input and
conversion calculations.



```html
<!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"></script>
</head>
<body>

<div id="app">
  <h2>Currency Converter</h2>

  <div>
    <label for="amount">Amount in USD:</label>
    <input type="number" id="amount" v-model.number="usdAmount" @input="convertCurrency">
  </div>

  <div>
    <p>Amount in USD: {{ usdAmount }}</p>
    <p>Amount in EUR: {{ convertedAmount }}</p>
  </div>
</div>

```



```
<html lang="en">
<body>

<script>
new Vue({
  el: '#app',
  data: {
    usdAmount: 0,
    exchangeRate: 0.85,
    convertedAmount: 0
  },
  methods: {
    convertCurrency() {
      this.convertedAmount = this.usdAmount * this.exchangeRate;
    }
  }
});
</script>

</body>
</html>
```



**Currency Converter**

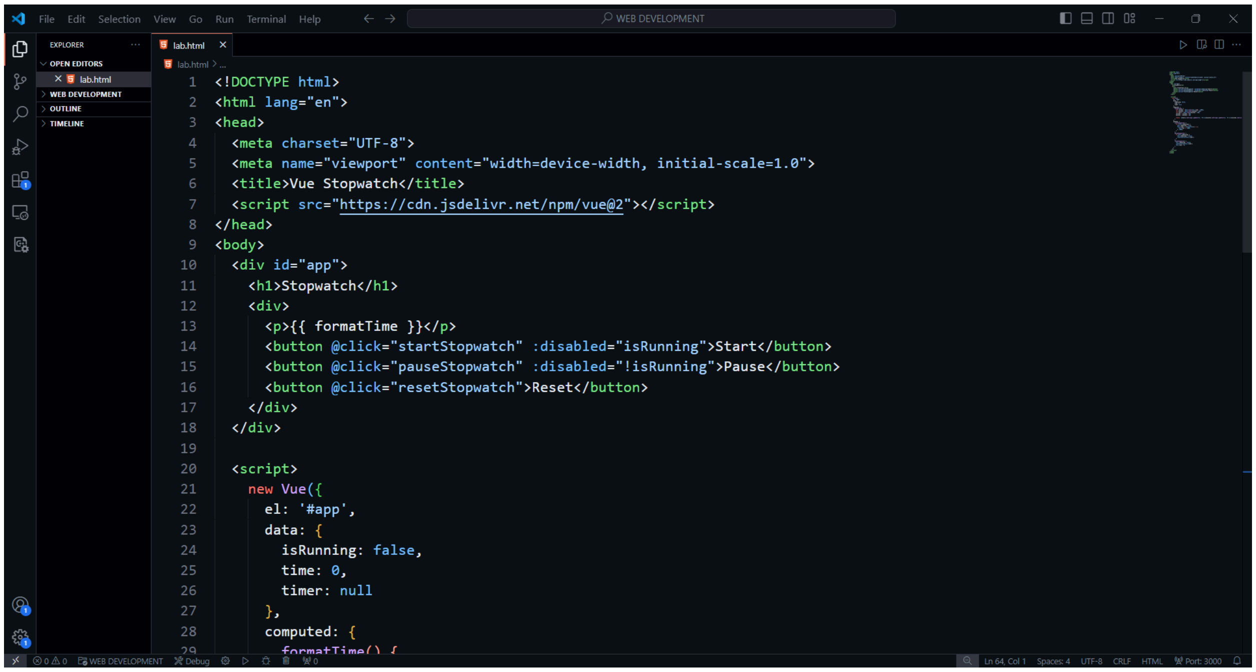
Amount in USD: 2000000

Amount in USD: 2000000

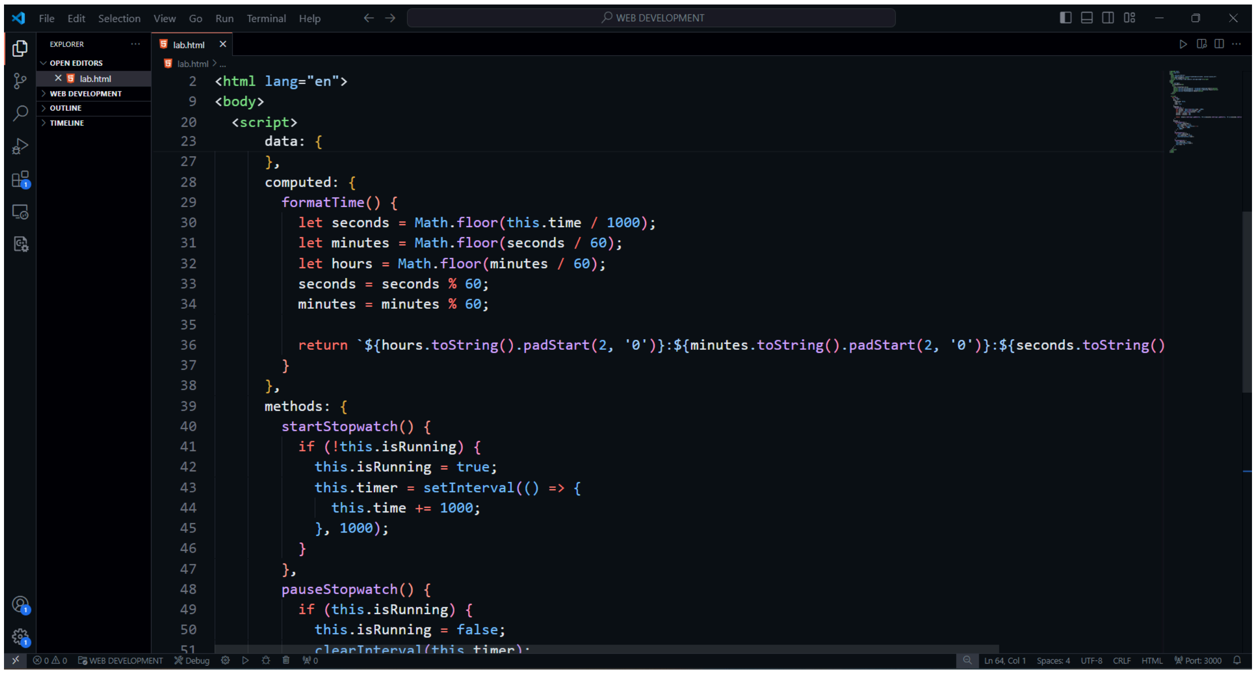
Amount in EUR: 1700000

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

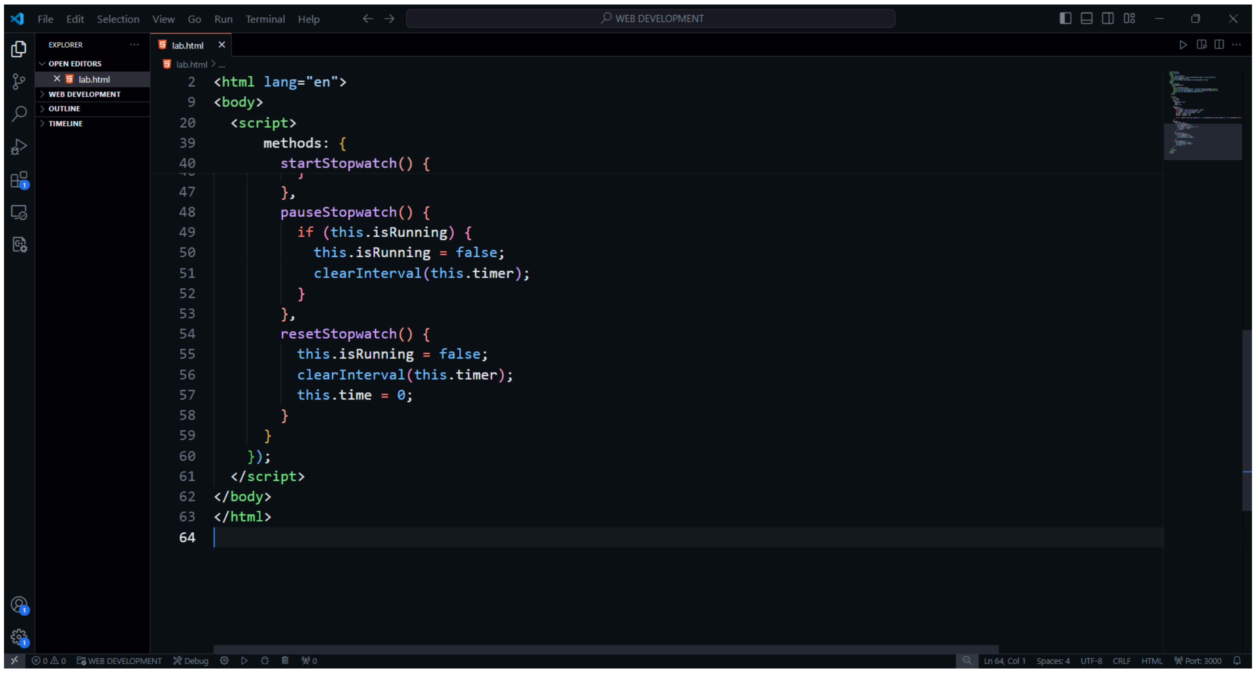
timer’s state and actions.


File Edit Selection View Go Run Terminal Help
WEB DEVELOPMENT
EXPLORER
OPEN EDITORS
lab.html
WEB DEVELOPMENT
OUTLINE
TIMELINE
Ln 64, Col 1 Spaces: 4 UTF-8 CRLF HTML Port: 3000

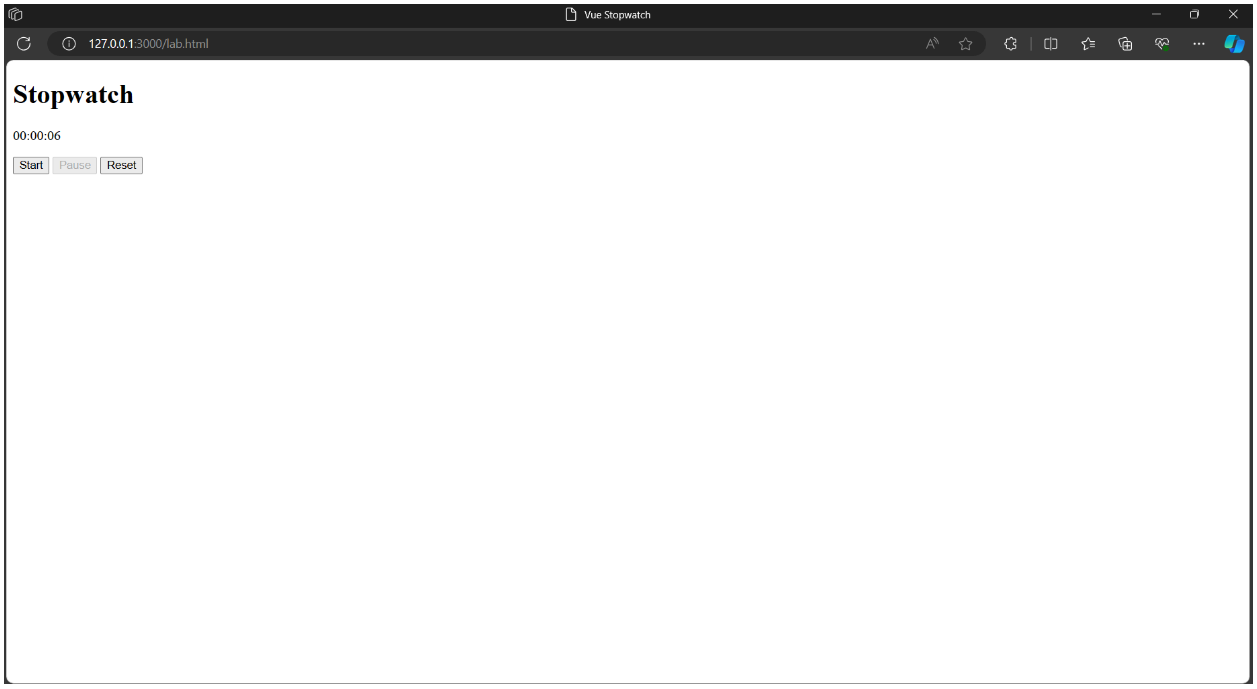
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vue Stopwatch</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
</head>
<body>
  <div id="app">
    <h1>Stopwatch</h1>
    <div>
      <p>{{ formatTime }}</p>
      <button @click="startStopwatch" :disabled="isRunning">Start</button>
      <button @click="pauseStopwatch" :disabled="!isRunning">Pause</button>
      <button @click="resetStopwatch">Reset</button>
    </div>
  </div>

  <script>
    new Vue({
      el: '#app',
      data: {
        isRunning: false,
        time: 0,
        timer: null
      },
      computed:
        formatTime() {
```


File Edit Selection View Go Run Terminal Help
WEB DEVELOPMENT
EXPLORER
OPEN EDITORS
lab.html
WEB DEVELOPMENT
OUTLINE
TIMELINE
Ln 64, Col 1 Spaces: 4 UTF-8 CRLF HTML Port: 3000

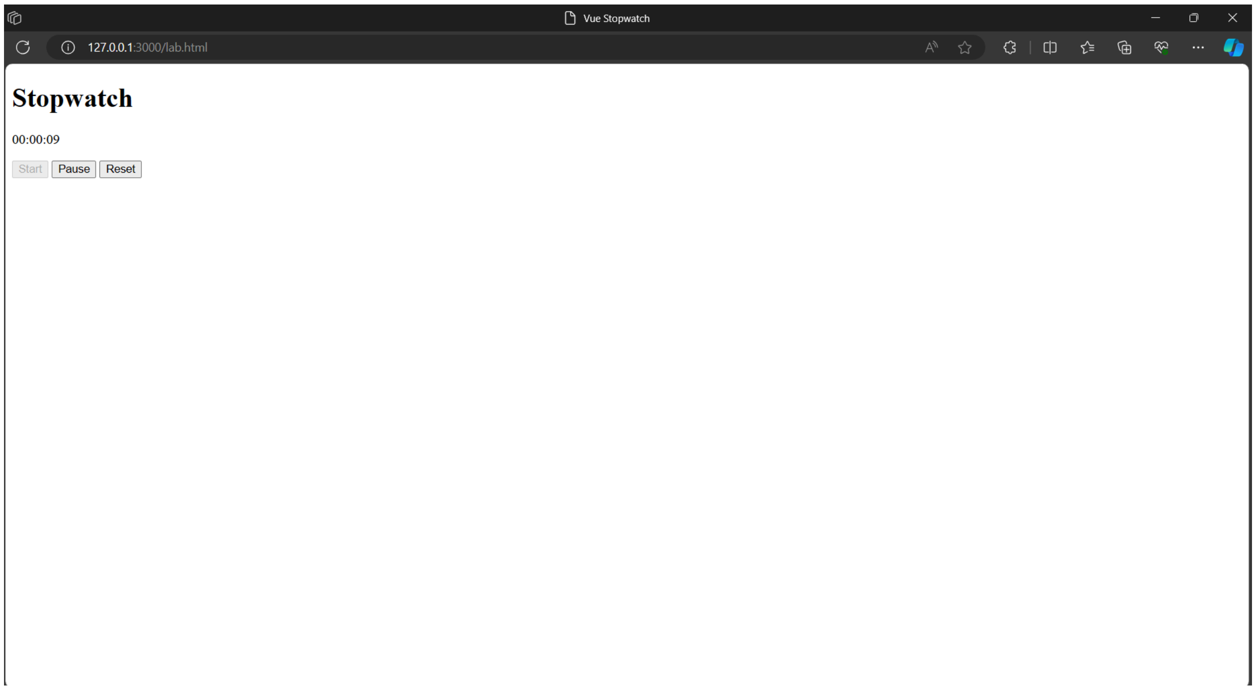
```html
<html lang="en">
<body>
  <script>
      data: {
      },
      computed: {
        formatTime() {
          let seconds = Math.floor(this.time / 1000);
          let minutes = Math.floor(seconds / 60);
          let hours = Math.floor(minutes / 60);
          seconds = seconds % 60;
          minutes = minutes % 60;

          return `${hours.toString().padStart(2, '0')}:${minutes.toString().padStart(2, '0')}:${seconds.toString()
        }
      },
      methods: {
        startStopwatch() {
          if (!this.isRunning) {
            this.isRunning = true;
            this.timer = setInterval(() => {
              this.time += 1000;
            }, 1000);
          }
        },
        pauseStopwatch() {
          if (this.isRunning) {
            this.isRunning = false;
            clearInterval(this.timer);
```



```
<html lang="en">
<body>
  <script>
      methods: {
        startStopwatch() {
        },
        pauseStopwatch() {
          if (this.isRunning) {
            this.isRunning = false;
            clearInterval(this.timer);
          }
        },
        resetStopwatch() {
          this.isRunning = false;
          clearInterval(this.timer);
          this.time = 0;
        }
      }
    });
  </script>
</body>
</html>

```


Stopwatch

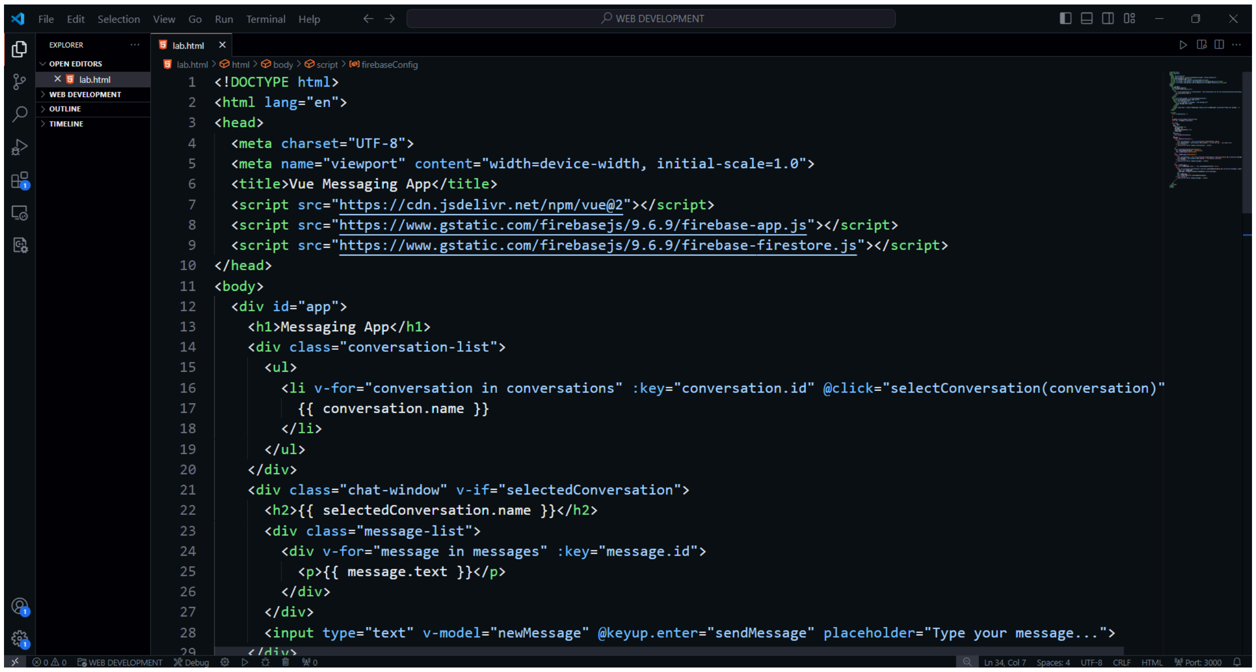
00:00:06


Stopwatch

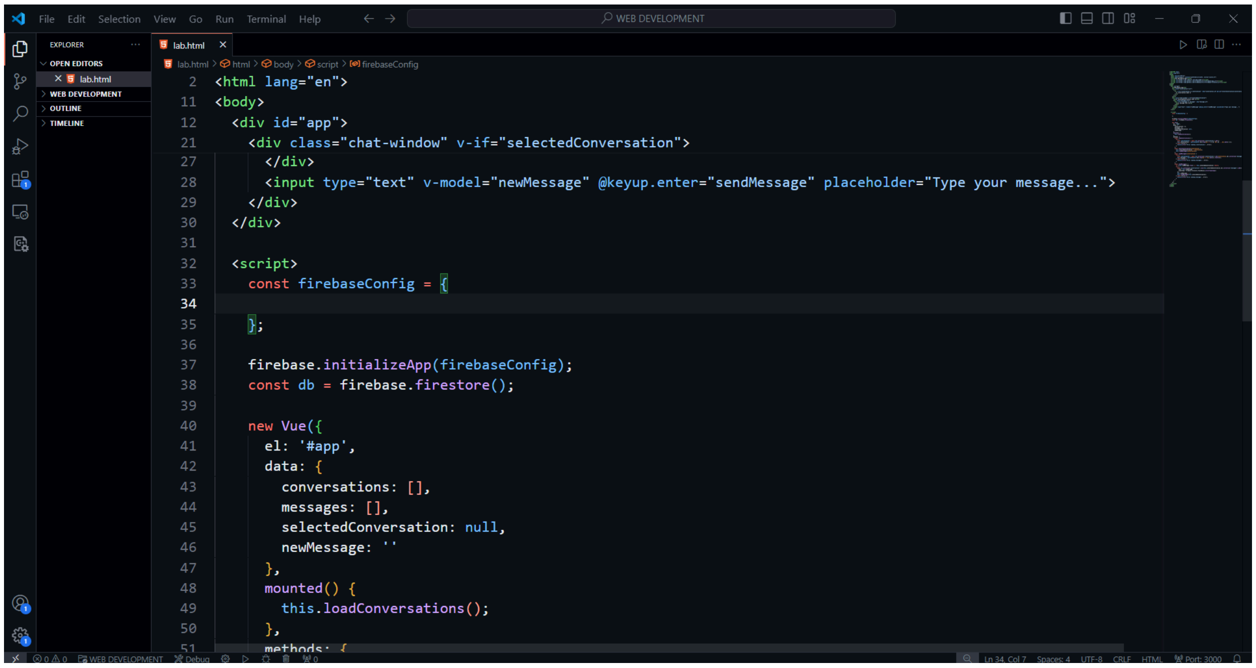
00:00:09

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

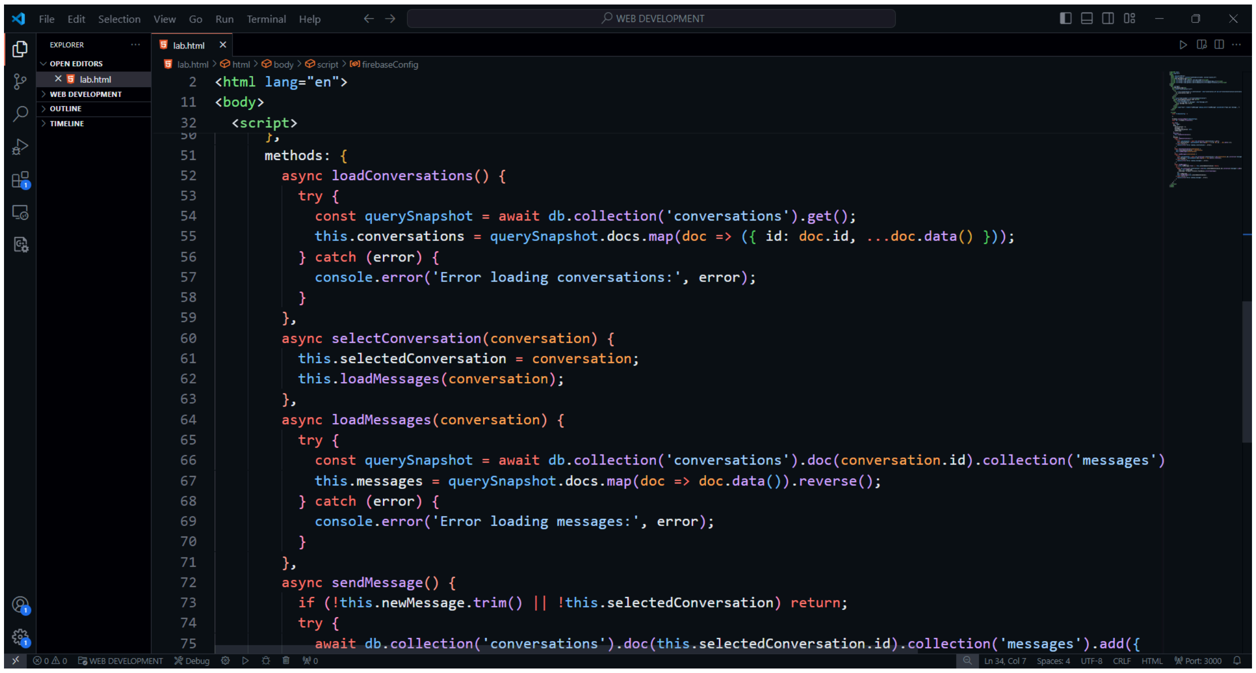
specific conversation to view its messages. The messages should be displayed in a chat
interface with the most recent message at the top. Users should be able to send new
messages and receive push notifications.



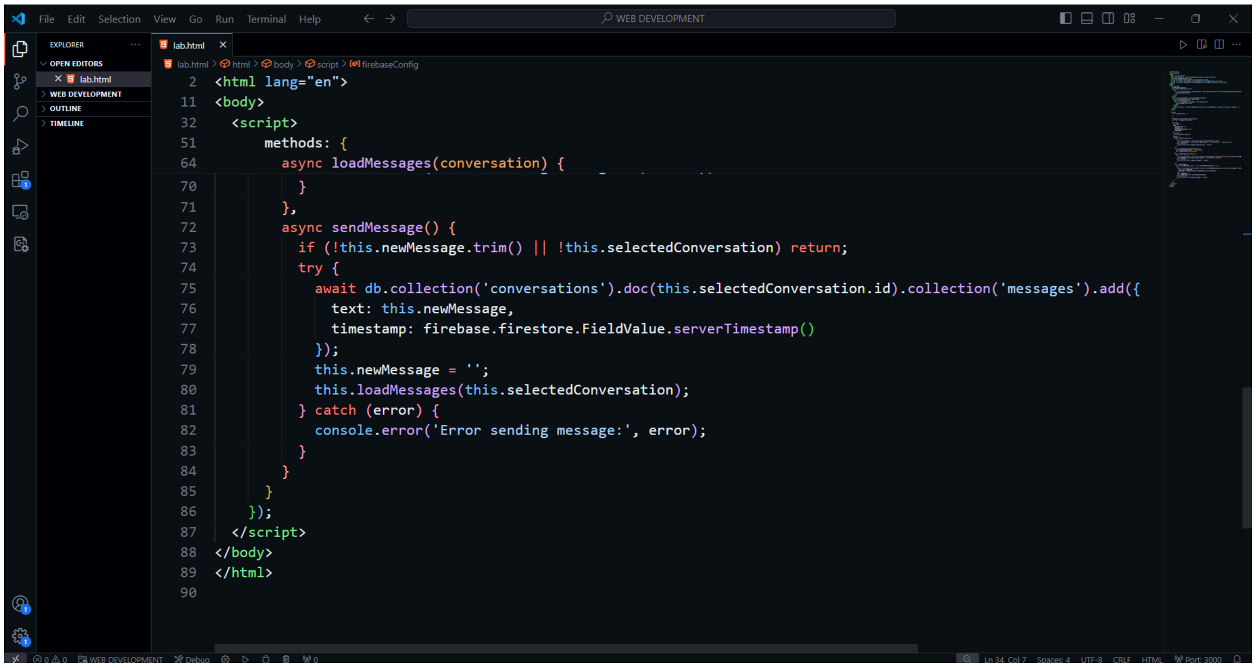
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vue Messaging App</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
  <script src="https://www.gstatic.com/firebasejs/9.6.9/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.6.9/firebase-firestore.js"></script>
</head>
<body>
  <div id="app">
    <h1>Messaging App</h1>
    <div class="conversation-list">
      <ul>
        <li v-for="conversation in conversations" :key="conversation.id" @click="selectConversation(conversation)"
          {{ conversation.name }}
        </li>
      </ul>
    </div>
    <div class="chat-window" v-if="selectedConversation">
      <h2>{{ selectedConversation.name }}</h2>
      <div class="message-list">
        <div v-for="message in messages" :key="message.id">
          <p>{{ message.text }}</p>
        </div>
      </div>
      <input type="text" v-model="newMessage" @keyup.enter="sendMessage" placeholder="Type your message...">
```



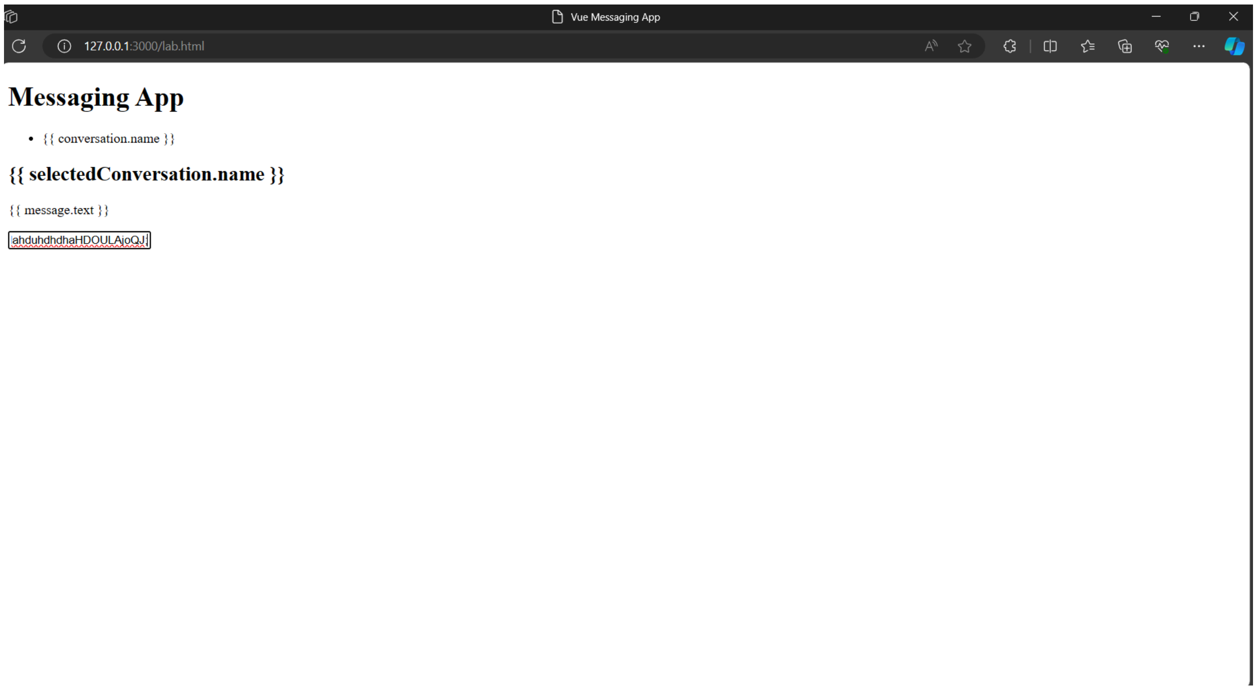
```html
<html lang="en">
<body>
  <div id="app">
    <div class="chat-window" v-if="selectedConversation">
      </div>
      <input type="text" v-model="newMessage" @keyup.enter="sendMessage" placeholder="Type your message...">
    </div>
  </div>

  <script>
    const firebaseConfig = {

    };

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

    new Vue({
      el: '#app',
      data: {
        conversations: [],
        messages: [],
        selectedConversation: null,
        newMessage: ''
      },
      mounted() {
        this.loadConversations();
      },
      methods: {
```


File Edit Selection View Go Run Terminal Help
WEB DEVELOPMENT
EXPLORER
OPEN EDITORS
lab.html
WEB DEVELOPMENT
OUTLINE
TIMELINE
lab.html > html > body > script > firebaseConfig
Ln 34, Col 7 Spaces: 4 UTF-8 CRLF HTML Port: 3000

```
<html lang="en">
<body>
  <script>
      },
      methods: {
        async loadConversations() {
          try {
            const querySnapshot = await db.collection('conversations').get();
            this.conversations = querySnapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
          } catch (error) {
            console.error('Error loading conversations:', error);
          }
        },
        async selectConversation(conversation) {
          this.selectedConversation = conversation;
          this.loadMessages(conversation);
        },
        async loadMessages(conversation) {
          try {
            const querySnapshot = await db.collection('conversations').doc(conversation.id).collection('messages')
            this.messages = querySnapshot.docs.map(doc => doc.data()).reverse();
          } catch (error) {
            console.error('Error loading messages:', error);
          }
        },
        async sendMessage() {
          if (!this.newMessage.trim() || !this.selectedConversation) return;
          try {
            await db.collection('conversations').doc(this.selectedConversation.id).collection('messages').add({
```


File Edit Selection View Go Run Terminal Help
WEB DEVELOPMENT
EXPLORER
OPEN EDITORS
lab.html
WEB DEVELOPMENT
OUTLINE
TIMELINE
lab.html > html > body > script > firebaseConfig

```
<html lang="en">
<body>
  <script>
      methods: {
        async loadMessages(conversation) {
          }
        },
        async sendMessage() {
          if (!this.newMessage.trim() || !this.selectedConversation) return;
          try {
            await db.collection('conversations').doc(this.selectedConversation.id).collection('messages').add({
              text: this.newMessage,
              timestamp: firebase.firestore.FieldValue.serverTimestamp()
            });
            this.newMessage = '';
            this.loadMessages(this.selectedConversation);
          } catch (error) {
            console.error('Error sending message:', error);
          }
        }
      }
    });
  </script>
</body>
</html>
```



# Messaging App

- {{ conversation.name }}

## {{ selectedConversation.name }}

{{ message.text }}

ahduhdhdhaHDOULAjoQJ