Q1.<!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>

<!-- Include Vue.js -->

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

</head>

<body>

<div id="app">

<h1>Currency Converter</h1>

<div>

<label for="amount">Enter Amount:</label>

<input type="number" v-model="amount" id="amount" @input="convertCurrency">

</div>

<div>

<label for="fromCurrency">From Currency:</label>

<select v-model="fromCurrency" id="fromCurrency" @change="convertCurrency">

<option value="USD">USD</option>

<option value="EUR">EUR</option>

<!-- Add more currencies as needed -->

</select>

</div>

<div>

<label for="toCurrency">To Currency:</label>

<select v-model="toCurrency" id="toCurrency" @change="convertCurrency">

<option value="USD">USD</option>

<option value="EUR">EUR</option>

<!-- Add more currencies as needed -->

</select>

</div>

<div>

<p>Converted Amount: {{ convertedAmount }}</p>

</div>

</div>

<script>

new Vue({

el: '#app',

data: {

amount: 1,

fromCurrency: 'USD',

toCurrency: 'EUR',

exchangeRate: 0.85, // Example exchange rate (USD to EUR)

},

computed: {

convertedAmount: function () {

return (this.amount \* this.exchangeRate).toFixed(2);

},

},

methods: {

convertCurrency: function () {

this.convertedAmount = (this.amount \* this.exchangeRate).toFixed(2);

},

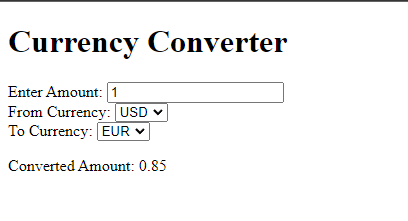
},

});

</script>

</body>

</html>



Q2.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Stopwatch</title>

<!-- Include Vue.js -->

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

</head>

<body>

<div id="app">

<h1>Stopwatch</h1>

<div>

<p>{{ formatTime }}</p>

</div>

<div>

<button @click="startPauseTimer">{{ isRunning ? 'Pause' : 'Start' }}</button>

<button @click="resetTimer">Reset</button>

</div>

</div>

<script>

new Vue({

el: '#app',

data: {

timer: 0,

isRunning: false,

},

computed: {

formatTime: function () {

const minutes = Math.floor(this.timer / 60);

const seconds = this.timer % 60;

return `${this.padNumber(minutes)}:${this.padNumber(seconds)}`;

},

},

methods: {

startPauseTimer: function () {

if (this.isRunning) {

clearInterval(this.timerInterval);

} else {

this.timerInterval = setInterval(() => {

this.timer++;

}, 1000);

}

this.isRunning = !this.isRunning;

},

resetTimer: function () {

clearInterval(this.timerInterval);

this.timer = 0;

this.isRunning = false;

},

padNumber: function (number) {

return number.toString().padStart(2, '0');

},

},

});

</script>

</body>

</html>



Q3.<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Messaging App</title>

<!-- Include Vue.js and Vuex -->

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

<script src="https://cdn.jsdelivr.net/npm/vuex@3"></script>

</head>

<body>

<div id="app">

<h1>Messaging App</h1>

<!-- Conversation List -->

<div class="conversation-list">

<div v-for="(conversation, index) in conversations" :key="index" @click="selectConversation(index)" :class="{ 'selected': selectedConversation === index }">

{{ conversation.name }}

</div>

</div>

<!-- Chat Interface -->

<div class="chat-interface">

<div v-if="selectedConversation !== null">

<h2>{{ conversations[selectedConversation].name }}</h2>

<div class="message-list">

<div v-for="(message, index) in conversations[selectedConversation].messages" :key="index" class="message">

<span :class="{ 'sender': message.sender === 'user' }">{{ message.sender }}:</span> {{ message.text }}

</div>

</div>

<div class="message-input">

<input v-model="newMessage" placeholder="Type your message..." />

<button @click="sendMessage">Send</button>

</div>

</div>

<div v-else>

<p>Select a conversation to start chatting.</p>

</div>

</div>

</div>

<script>

// Mock data

const mockConversations = [

{

name: 'Friend 1',

messages: [

{ sender: 'user', text: 'Hello!' },

{ sender: 'friend', text: 'Hi there!' },

],

},

{

name: 'Friend 2',

messages: [

{ sender: 'friend', text: 'Hey!' },

{ sender: 'user', text: 'How are you?' },

],

},

];

new Vue({

el: '#app',

data: {

conversations: mockConversations,

selectedConversation: null,

newMessage: '',

},

methods: {

selectConversation(index) {

this.selectedConversation = index;

},

sendMessage() {

if (this.selectedConversation !== null && this.newMessage.trim() !== '') {

this.conversations[this.selectedConversation].messages.push({

sender: 'user',

text: this.newMessage.trim(),

});

this.newMessage = '';

// Simulate receiving a message after a delay

setTimeout(() => {

this.conversations[this.selectedConversation].messages.push({

sender: 'friend',

text: 'I got your message!',

});

}, 1000);

}

},

},

});

</script>

<style>

body {

font-family: 'Arial', sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

display: flex;

align-items: center;

justify-content: center;

height: 100vh;

}

#app {

background-color: #fff;

border-radius: 8px;

padding: 20px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

text-align: center;

}

h1 {

color: #333;

}

.conversation-list {

display: flex;

flex-direction: column;

width: 200px;

margin-right: 20px;

padding: 10px;

background-color: #f0f0f0;

border-radius: 8px;

}

.conversation-list div {

cursor: pointer;

padding: 8px;

margin-bottom: 8px;

border-radius: 4px;

transition: background-color 0.3s;

}

.conversation-list div:hover {

background-color: #e0e0e0;

}

.conversation-list .selected {

background-color: #d0d0d0;

}

.chat-interface {

flex: 1;

display: flex;

flex-direction: column;

}

h2 {

margin-top: 0;

color: #333;

}

.message-list {

flex: 1;

overflow-y: auto;

padding: 10px;

}

.message {

margin-bottom: 8px;

}

.sender {

font-weight: bold;

margin-right: 5px;

}

.message-input {

display: flex;

align-items: center;

padding: 10px;

}

input {

flex: 1;

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

background-color: #4caf50;

color: #fff;

padding: 8px 16px;

border: none;

border-radius: 4px;

cursor: pointer;

margin-left: 8px;

transition: background-color 0.3s;

}

button:hover {

background-color: #45a049;

}

</style>

</body>

</html>

