

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Chatbot</title>
  <style>
    body {
      font-family: Roboto, sans-serif;
      display: flex;
      flex-direction: column;
      height: 100vh;
      margin: 0;
      padding: 0;
    }
    #banner {
      background-color: #c0c0c0;
      color: black;
      font-size: 20pt;
      padding: 12px;
      text-align: left;
    }
    #chat-container {
      display: flex;
      flex-direction: column;
      flex-grow: 1;
      padding: 20px;
      box-sizing: border-box;
    }
    #spacing {
      float = 'none';
      margin = "10px";
      padding = "10px";
      display = 'flex';
      flex-direction = 'column';
      overflow = 'auto';
    }
    #messages {
      flex-grow: 1;
      overflow-y: auto;
      border: 1px solid #ccc;
      padding: 10px;
    }
    #input-container {
      display: flex;
    }
    #user-input {
      flex-grow: 1;
      padding: 10px;
      border: 1px solid #ccc;

```

```

        border-top: none;
    }
    #send-button {
        padding: 10px;
        background-color: #007bff;
        color: white;
        border: none;
        cursor: pointer;
    }
    #correctness-dialog-container {
        display: flex;
        flex-direction: column;
        padding: 10px;
        border: 12px solid #ddd;
        background-color: #f9f9f9;
    }
    #correctness-dialog-container button {
        margin: 5px 0;
        padding: 10px;
        background-color: #fff;
        color: #518eca;
        border: 1px solid #ddd;
        cursor: pointer;
    }
    #dialog-container {
        display: flex;
        flex-direction: column;
        padding: 10px;
        border: 12px solid #ddd;
        background-color: #f9f9f9;
    }
    #dialog-container button {
        margin: 5px 0;
        padding: 10px;
        background-color: #fff;
        color: #518eca;
        border: 1px solid #ddd;
        cursor: pointer;
    }
    #message-time {
        white-space: pre-wrap;
    }
</style>
</head>
<body>
    <div id="banner">
        <div style="float: left; line-height: 63px; height: 63px;">The
chatbot collects all the information</div>
        <div style="float: right;"></div>
    </div>

```

```

<div id="chat-container">
  <div id="messages"></div>
  <div id="input-container">
    <input type="text" id="user-input" placeholder="Type a message...">
    <button id="send-button">Send</button>
  </div>
</div>
<script>
  const messages = document.getElementById('messages');
  const userInput = document.getElementById('user-input');
  const sendButton = document.getElementById('send-button');
  const correctnessDialogContent =
document.getElementById('correctness-dialog-container');
  const dialogContainer = document.getElementById('dialog-container');
  const changeNameButton = document.getElementById('change-name-button');
  const changeRoleButton = document.getElementById('change-role-button');
  const changeEmailButton = document.getElementById('change-email-button');
  const changePhoneButton = document.getElementById('change-phone-button');

  let state = 'initial';
  let userName = '';
  let userRole = '';
  let userEmail = '';
  let userPhone = '';

  sendButton.addEventListener('click', () => {
    const userMessage = userInput.value.trim();
    if (userMessage) {
      appendMessage(userMessage, 'user');
      const botResponse = getBotResponse(userMessage);
      setTimeout(() => appendMessage(botResponse, 'bot'), 500);
      userInput.value = '';
    }
  });

  function appendMessage(message, sender) {
    const messageSpacing = document.createElement('div');
    const messageElement = document.createElement('div');
    const messageNode = document.createTextNode(message);
    const messageUserData = document.createElement('div');
    const messageUser = document.createElement('div');
    const messageDate = document.createElement('div');
    const messageStack = document.createElement('div');

    messageSpacing.id = 'spacing';

    messageElement.style.cssFloat = sender === 'user' ? 'right'
: 'left';
    messageElement.style.color = sender === 'user' ? 'white' :
'black';

```

```

messageElement.appendChild(messageNode);

if (state === 'change_question' && sender === 'bot') {
    messageElement.id='correctness-dialog-container';
    const diaButtonYes =
document.createElement('button');
    const diaButtonNo =
document.createElement('button');
    diaButtonYes.id="affirmative";
    diaButtonYes.textContent="Yes";
    diaButtonNo.id="negative";
    diaButtonNo.textContent="No";
    messageElement.appendChild(diaButtonYes);
    messageElement.appendChild(diaButtonNo);
    messageElement.style.width = '400px';
    diaButtonYes.addEventListener('click', () => {
        state = 'change_question';
        appendMessage('Yes', 'user');
        const botResponse = getBotResponse("Yes");
        setTimeout(() => appendMessage(botResponse,
'bot'), 500);

        userInput.value = '';
    });
    diaButtonNo.addEventListener('click', () => {
        state = 'done';
        appendMessage('No', 'user');
        const botResponse = getBotResponse("No");
        setTimeout(() => appendMessage(botResponse,
'bot'), 500);

        userInput.value = '';
    });
} else if (state === 'change_info' && sender === 'bot') {
    messageElement.id='dialog-container';
    const diaButtonName =
document.createElement('button');
    const diaButtonRole =
document.createElement('button');
    const diaButtonEmail =
document.createElement('button');
    const diaButtonPhone =
document.createElement('button');
    diaButtonName.addEventListener('click', () => {
        state = 'change_name';
        appendMessage('Please enter the new name:',
'bot');
    });
    diaButtonRole.addEventListener('click', () => {
        state = 'change_role';
        appendMessage('Please enter the new role:',
'bot');

```

```

    });
    diaButtonEmail.addEventListener('click', () => {
        state = 'change_email';
        appendMessage('Please enter the new email
address:', 'bot');
    });
    diaButtonPhone.addEventListener('click', () => {
        state = 'change_phone';
        appendMessage('Please enter the new phone
number:', 'bot');
    });
    diaButtonName.textContent="Change Name";
    diaButtonRole.textContent="Change Role";
    diaButtonEmail.textContent="Change Email";
    diaButtonPhone.textContent="Change Phone";
    messageElement.appendChild(diaButtonName);
    messageElement.appendChild(diaButtonRole);
    messageElement.appendChild(diaButtonEmail);
    messageElement.appendChild(diaButtonPhone);
    messageElement.style.width = '400px';
    } else {
        messageElement.style.backgroundColor = sender ===
'user' ? '#3a95db' : '#fff3ba';
        messageElement.style.padding = "12px";
    }

    messageElement.style.overflow = 'auto';
    var mySend = sender === 'user' ? 'User' : 'Bot';
    var myDate = new
Date().toLocaleTimeString().replace(/.*(\d{2}:\d{2}:\d{2}).*/, "$1");
    var myTime = ` at ${myDate}`;
    messageUser.textContent = mySend;
    messageDate.textContent = myTime;
    messageDate.id = 'message-time';
    messageUserDate.style.cssFloat = sender === 'user' ?
'right' : 'left';

    messageUserDate.id = 'message-stamp';
    messageUserDate.style.display = 'flex';
    messageUserDate.style.flexDirection = 'row';
    messageUserDate.style.overflow = 'auto';
    messageUserDate.style.alignItems = sender === 'user' ?
'flex-end' : 'flex-start';
    const msgDate = document.getElementById('message-time');
    <!-- remove the previous -->
    if (msgDate) {
        msgDate.remove();
    }
    messageUserDate.appendChild(messageUser);
    messageUserDate.appendChild(messageDate);

```

```

        messageStack.id = 'message-stack';
        messageStack.style.display = 'flex';
        messageStack.style.flexDirection = 'column';
        messageStack.style.overflow = 'auto';
        messageStack.style.alignItems = sender === 'user' ?
'flex-end' : 'flex-start';

```

```

        messages.scrollTop = messages.scrollHeight;
        messageStack.appendChild(messageElement);
        messageStack.appendChild(messageUserData);
        messageSpacing.appendChild(messageStack);
        messages.appendChild(messageSpacing);
    }

```

```

function getBotResponse(message) {
    switch (state) {
        case 'initial':
            state = 'new_signature';
            return 'Hi, at this moment you can only use
me to create a new Cronos signature';
        case 'new_signature':
            state = 'get_name';
            return 'What is your name?';
        case 'get_name':
            userName = message;
            state = 'get_role';
            return 'What is your role?';
        case 'get_role':
            userRole = message;
            state = 'get_email';
            return 'What is your email address?';
        case 'get_email':
            userEmail = message;
            state = 'get_phone';
            return 'What is your cell phone number?';
        case 'get_phone':
            userPhone = message;
            state = 'change_question';
            return `Thank you! You have entered this
information.\nName: ${userName}\nRole: ${userRole}\nEmail: ${userEmail}\nPhone:
${userPhone}\nDo you want to change anything? (yes/no)`;
        case 'change_info':
            state = 'change_info';
            return 'What would you like to change?';
        case 'change_question':
            state = 'change_info';
            return 'What would you like to change?';
        case 'change_name':
            userName = message;
            state = 'change_question';

```

```

        return `Name updated to ${userName}. Do you want to change
anything else? (yes/no)`;
        case 'change_role':
            userRole = message;
            state = 'change_question';
            return `Role updated to ${userRole}. Do you want to change
anything else? (yes/no)`;
        case 'change_email':
            userEmail = message;
            state = 'change_question';
            return `Email address updated to ${userEmail}. Do you want to
change anything else? (yes/no)`;
        case 'change_phone':
            userPhone = message;
            state = 'change_question';
            return `Phone number updated to ${userPhone}. Do you want to
change anything else? (yes/no)`;
        case 'done':
            state = 'initial';
            sendToServer();
            return `Thank you! Your signature is being
created.\n\nFinal Information:\nName: ${userName}\nRole: ${userRole}\nPhone:
${userPhone}\nEmail: ${userEmail}`;
        default:
            state = 'initial';
            return 'Say Again, over?';
    }
}

function someWebsiteFunction(UserName, userRole, userPhone, userEmail) {
    const url =
'https://url.us.m.mimecastprotect.com/s/xC5XC0YZ02Uplg1Jtk_DAZ?domain=rpaserver.com
';
    const data = { name, role, phone: phoneNumber };

    fetch(url, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(responseData => {
        console.log('Success:', responseData);
    })
    .catch(error => {
        console.error('Error:', error);
    });
}

function callSomeWebsiteFunction(name, role, phoneNumber, Email) {
    someWebsiteFunction(name, role, phoneNumber, Email);
}

```

```
function sendToServer() {  
    // Uncomment the line below to enable sending data to the server  
    appendMessage(`Sending your information to Server:\nName:  
${userName}\nRole: ${userRole}\nPhone: ${userPhone}\nEmail: ${userEmail}`, 'bot');  
    callSomeWebsiteFunction(userName, userRole, userPhone, userEmail);  
}  
</script>  
</body>  
</html>
```