I2A2

Challenge 1

Leonel Henrique Alves dos Santos

leo.cvsm@hotmail.com

2023

Summary

I2A2 – Challenge1 3
Website Prompt – Chat GPT OpenAi 4
Image Prompt – Microsoft Designer Generate 31

I2A2 – Challenge1

Link to access the project: https://leohena.github.io/i2a2_chllg1/

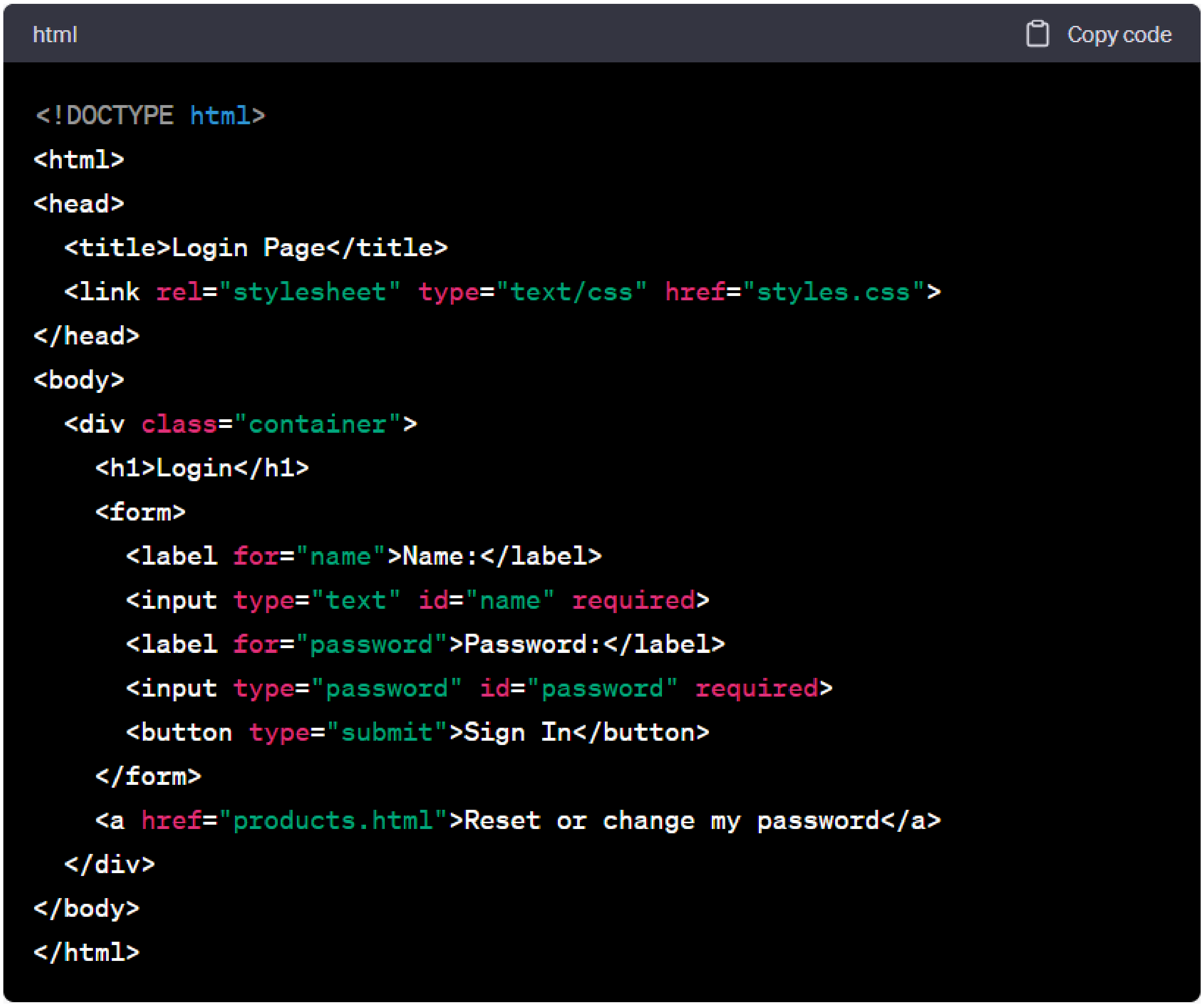
Name: Test

Password: Admin123

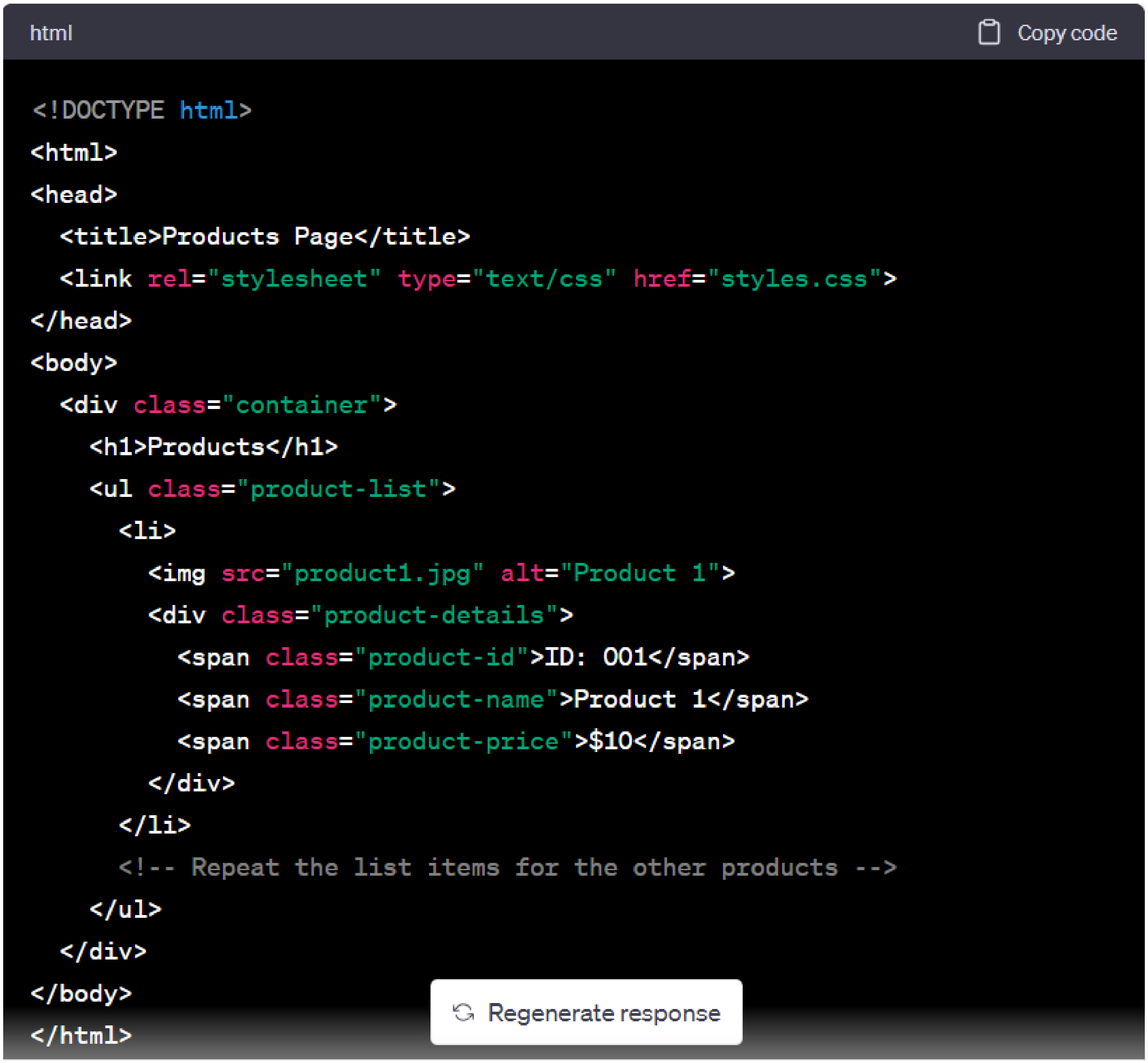
Website Prompt – Chat GPT OpenAi

Develop a website in HTML and CSS with three pages, The first page Login with field name and password that are required to click on button Sign In, the button Sign In guide to the second page Products and a link Reset or change my password that to guide to the third page Forgot my password. The second page Products need to the list of 10 products and the list need to have product id, product name, unit price and product image. The third page Forgot My Password show the field e-mail and button to reset your password, when click to button need to valide the fiel e-mail is filed up, shows up a pop up page with the message "The link to reset your password was sending to informed e-mail address and go back to the first page Login. All page need to space to insert the logo brand, the script be hind the website is Java Script. Develop the website font code and script in Java Script.

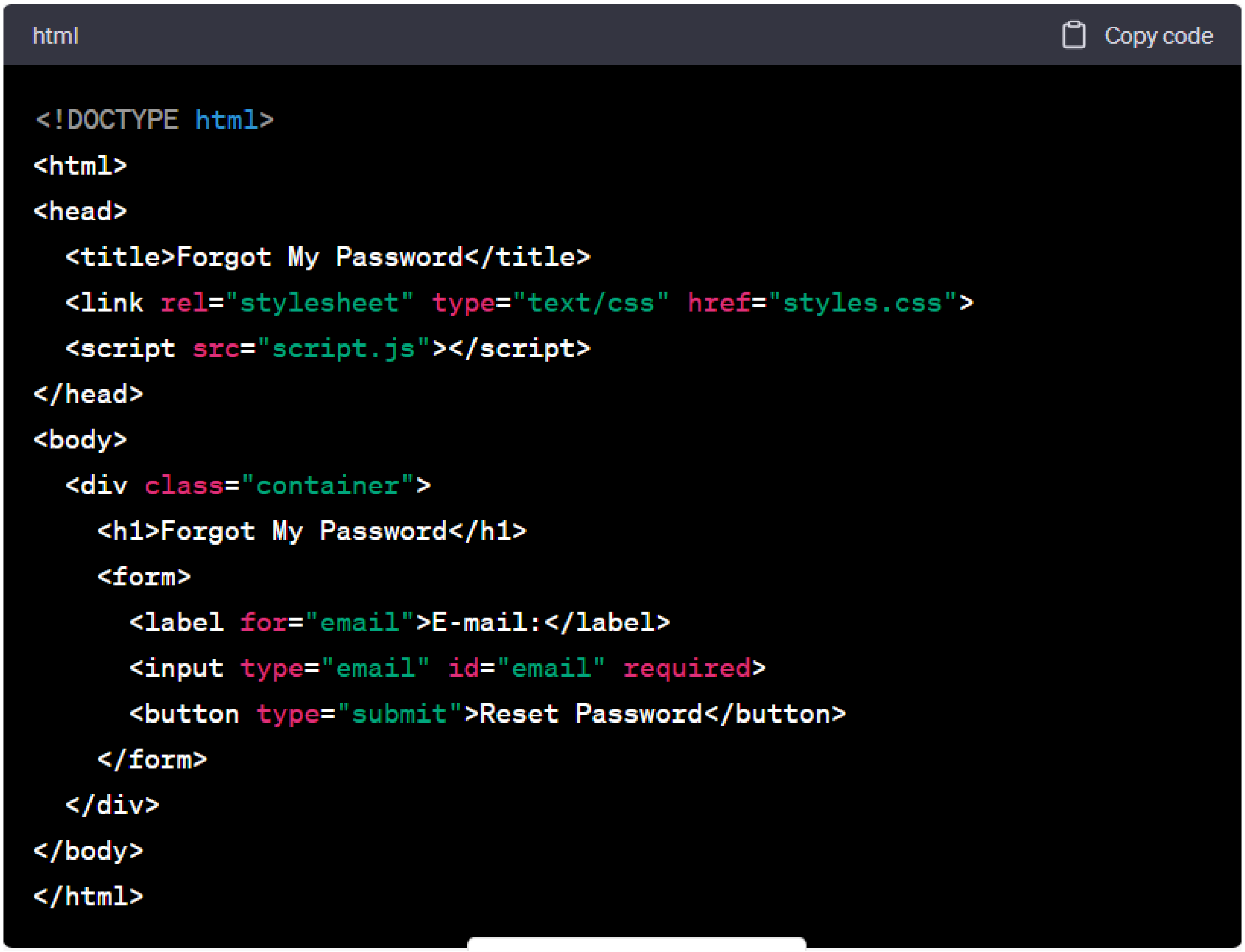
Sure! Here's an example of an HTML and CSS code for the three pages you described. For simplicity, I will provide a basic structure and styling. You can customize it further according to your needs.



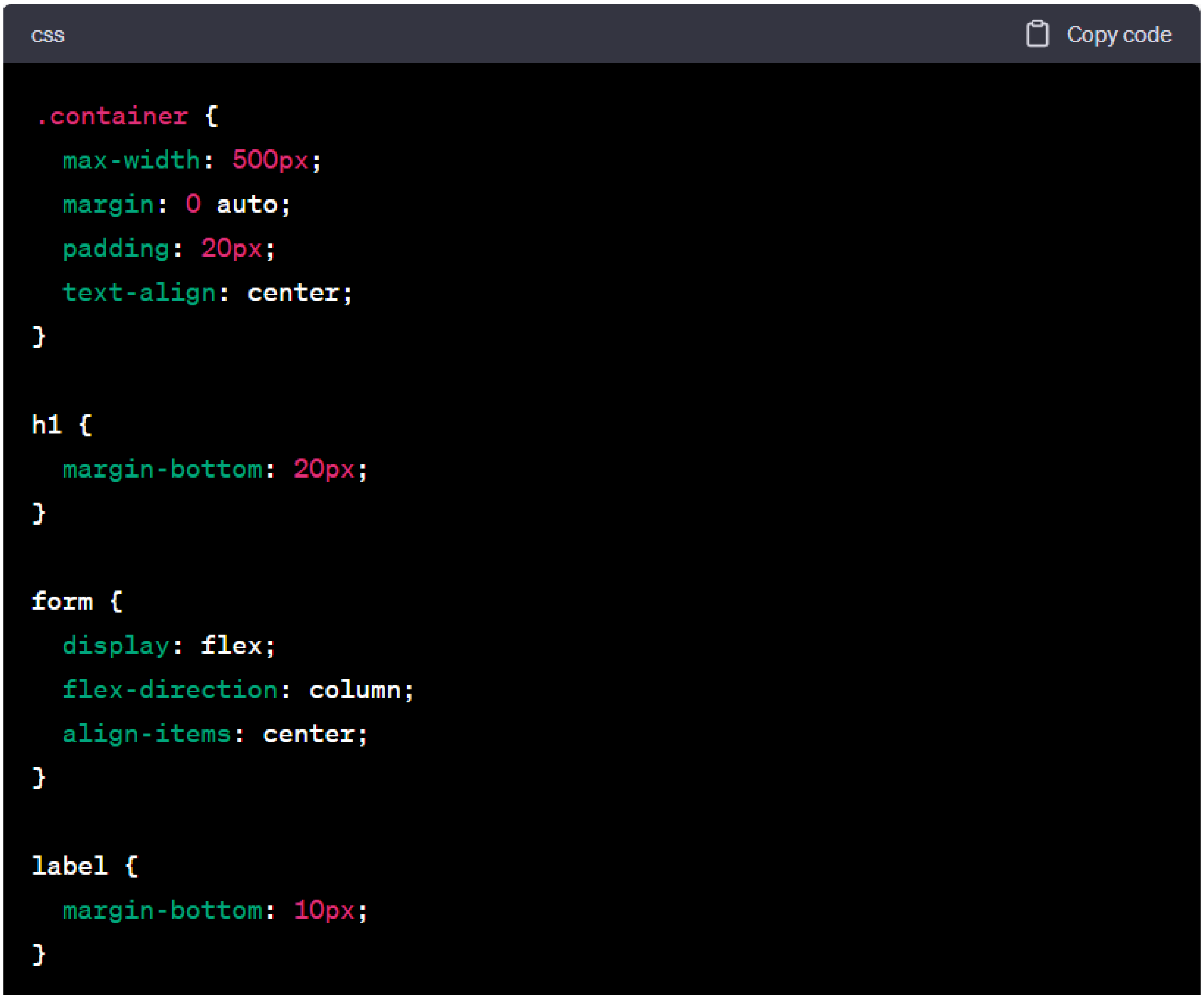
```html
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Login</h1>
    <form>
      <label for="name">Name:</label>
      <input type="text" id="name" required>
      <label for="password">Password:</label>
      <input type="password" id="password" required>
      <button type="submit">Sign In</button>
    </form>
    <a href="products.html">Reset or change my password</a>
  </div>
</body>
</html>
```



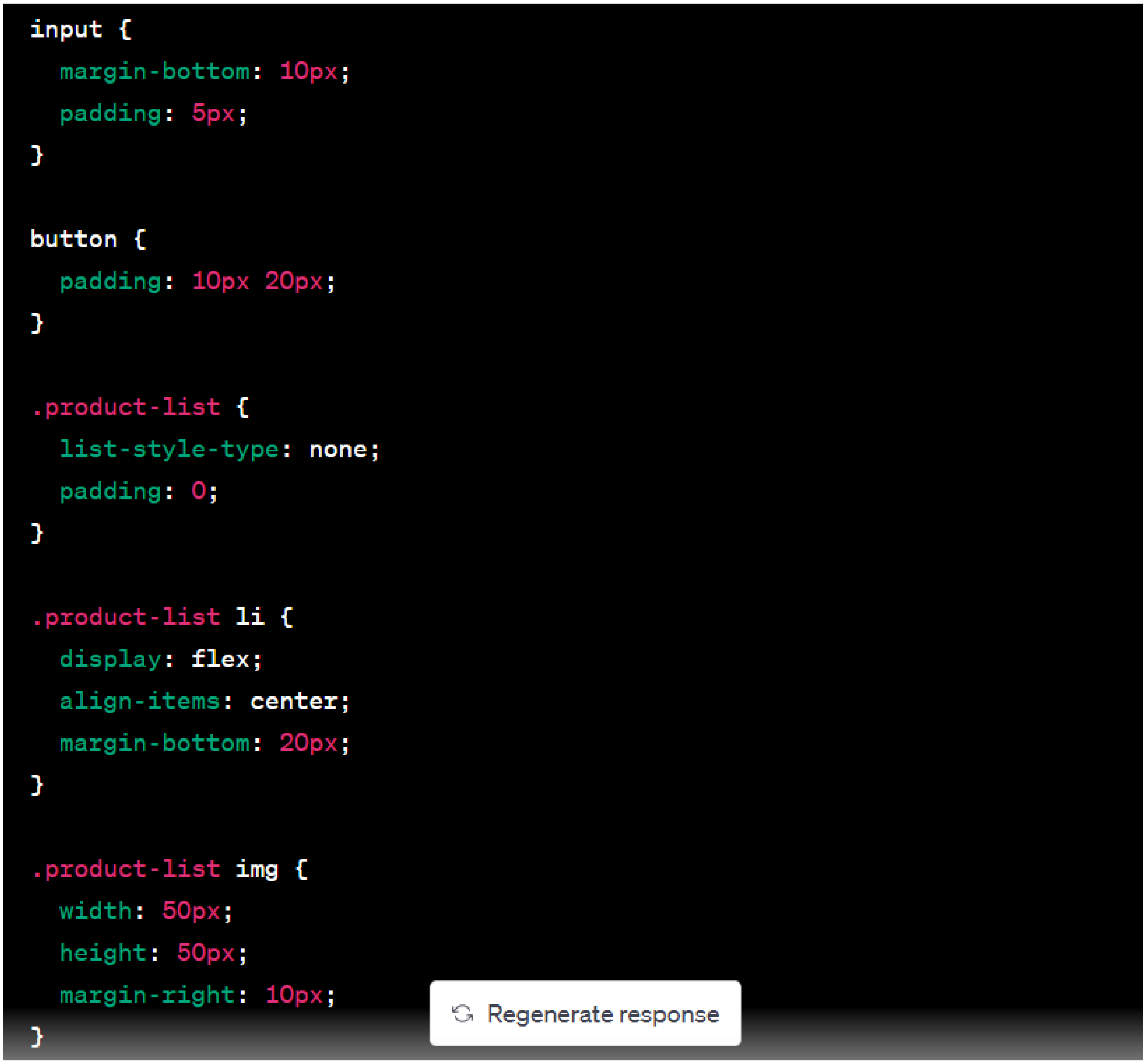
```html
<!DOCTYPE html>
<html>
<head>
  <title>Products Page</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Products</h1>
    <ul class="product-list">
      <li>
        <img src="product1.jpg" alt="Product 1">
        <div class="product-details">
          <span class="product-id">ID: 001</span>
          <span class="product-name">Product 1</span>
          <span class="product-price">$10</span>
        </div>
      </li>
      <!-- Repeat the list items for the other products -->
    </ul>
  </div>
</body>
</html>
```



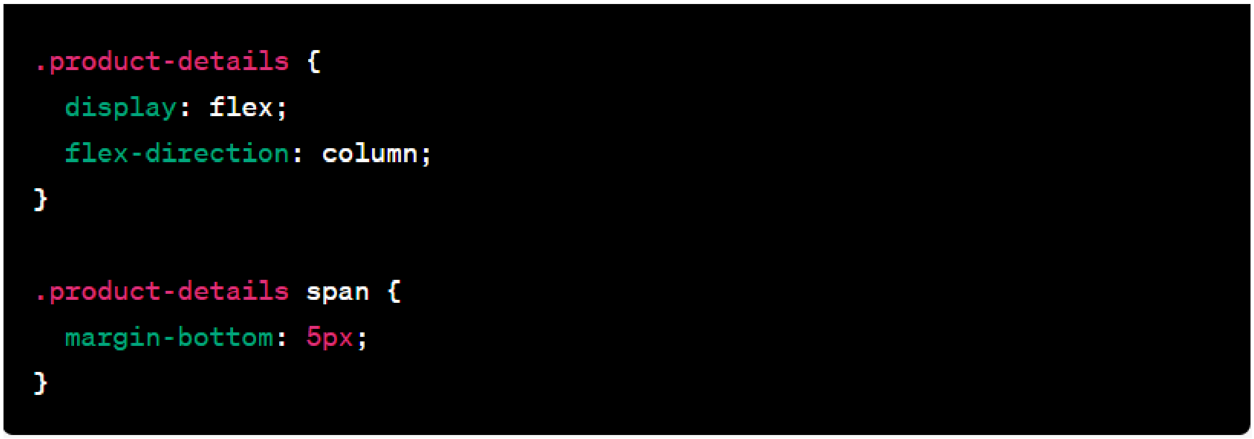
```html
<!DOCTYPE html>
<html>
<head>
  <title>Forgot My Password</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
  <script src="script.js"></script>
</head>
<body>
  <div class="container">
    <h1>Forgot My Password</h1>
    <form>
      <label for="email">E-mail:</label>
      <input type="email" id="email" required>
      <button type="submit">Reset Password</button>
    </form>
  </div>
</body>
</html>
```



```css
.container {
  max-width: 500px;
  margin: 0 auto;
  padding: 20px;
  text-align: center;
}

h1 {
  margin-bottom: 20px;
}

form {
  display: flex;
  flex-direction: column;
  align-items: center;
}

label {
  margin-bottom: 10px;
}
```

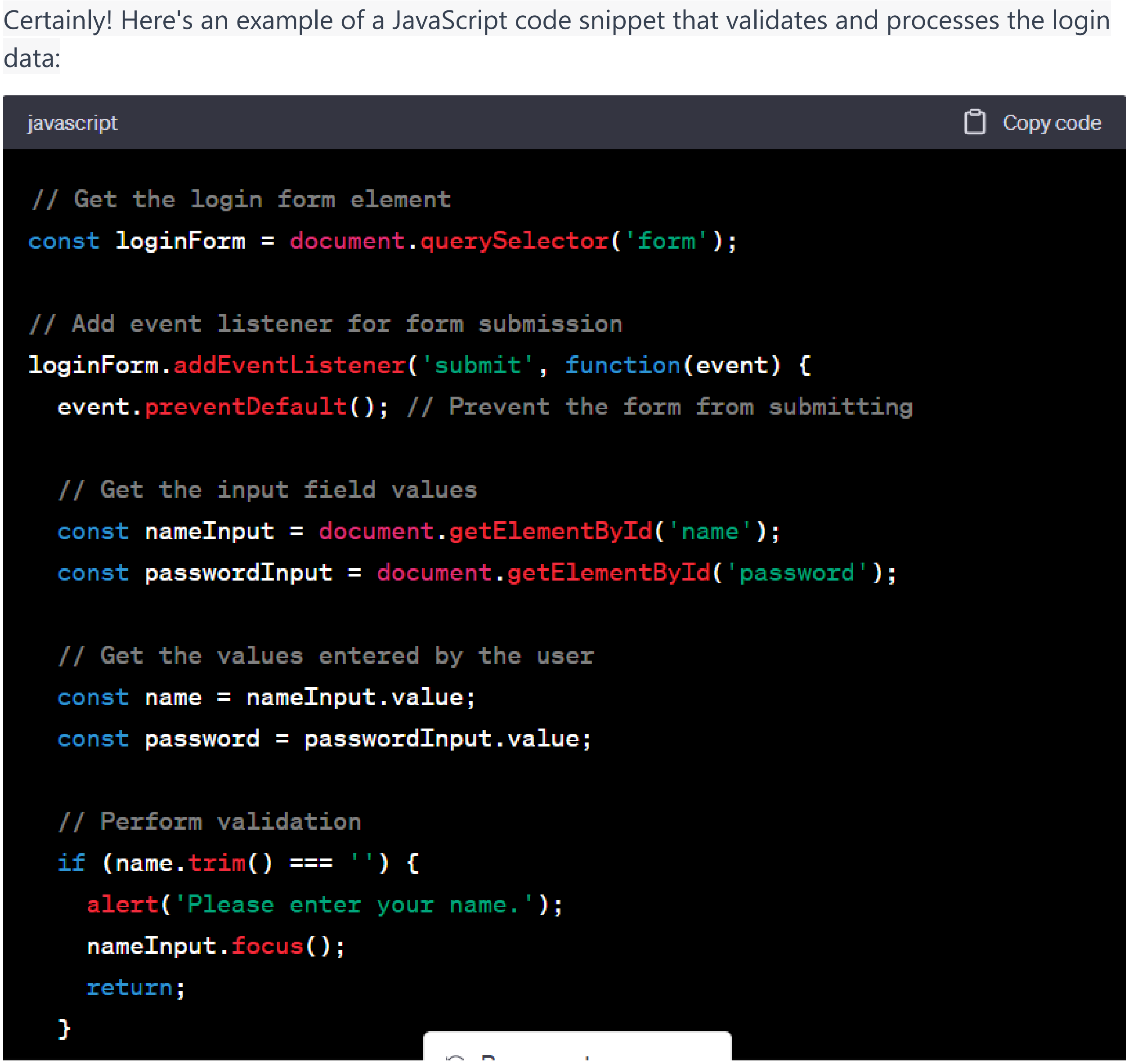


```
input {
  margin-bottom: 10px;
  padding: 5px;
}

button {
  padding: 10px 20px;
}

.product-list {
  list-style-type: none;
  padding: 0;
}

.product-list li {
  display: flex;
  align-items: center;
  margin-bottom: 20px;
}

.product-list img {
  width: 50px;
  height: 50px;
  margin-right: 10px;
}
```



```
.product-details {
  display: flex;
  flex-direction: column;
}

.product-details span {
  margin-bottom: 5px;
}
```

Develop a script in Java Script to validate and processing the login data



Certainly! Here's an example of a JavaScript code snippet that validates and processes the login data:

```javascript
// Get the login form element
const loginForm = document.querySelector('form');

// Add event listener for form submission
loginForm.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent the form from submitting

  // Get the input field values
  const nameInput = document.getElementById('name');
  const passwordInput = document.getElementById('password');

  // Get the values entered by the user
  const name = nameInput.value;
  const password = passwordInput.value;

  // Perform validation
  if (name.trim() === '') {
    alert('Please enter your name.');
    nameInput.focus();
    return;
  }
```

```
    if (password.trim() === '') {
      alert('Please enter your password.');
      passwordInput.focus();
      return;
    }

    // Process the login data (e.g., send it to a server for authentication)
    // You can replace this code with your own logic
    console.log('Name:', name);
    console.log('Password:', password);

    // Reset the form fields
    loginForm.reset();
  });
```

In this code snippet, we first get the login form element using `document.querySelector`. Then, we add an event listener to the form's `submit` event. When the form is submitted, the event listener function is executed.

Inside the event listener function, we get the values entered by the user in the name and password fields. We perform validation by checking if the fields are empty after trimming any whitespace. If either field is empty, we display an alert message and return early to prevent further processing.

If the validation passes, you can replace the placeholder code with your own logic for processing the login data. In this example, we simply log the name and password to the console.

Lastly, we reset the form fields using `loginForm.reset()` to clear the entered values.

Remember to include this JavaScript code in the `<script></script>` tags within your HTML file, or link it as an external script using the `src` attribute in the script tag.

Feel free to modify the code according to your specific requirements and integrate it into your login page.

How to connect my HTML page with my Java Script

You can create a separate JavaScript file with your code and link it to your HTML file using the `<script>` tag's `src` attribute. Here's an example:

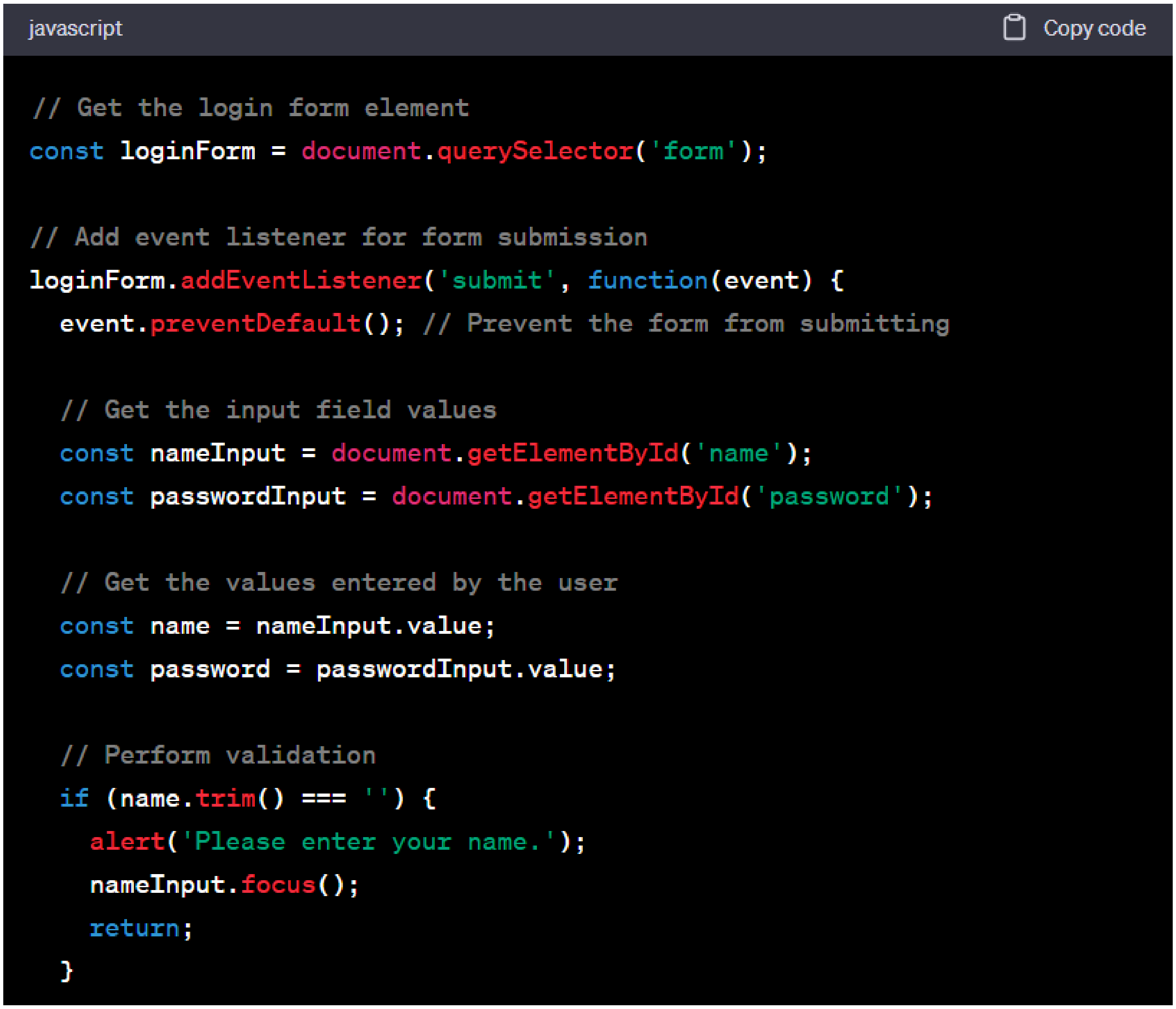
html Copy code

```html
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <!-- Rest of your HTML content -->

  <script src="script.js"></script>
</body>
</html>
```

In this example, the JavaScript code is saved in a separate file named `script.js`. Make sure that the `script.js` file is located in the same directory as your HTML file. Adjust the `src` attribute value accordingly if the file is located in a different directory.

To click on button Sign In process the script.js

To process the JavaScript code when the "Sign In" button is clicked, you can modify the JavaScript code as follows:



```javascript
// Get the login form element
const loginForm = document.querySelector('form');

// Add event listener for form submission
loginForm.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent the form from submitting

  // Get the input field values
  const nameInput = document.getElementById('name');
  const passwordInput = document.getElementById('password');

  // Get the values entered by the user
  const name = nameInput.value;
  const password = passwordInput.value;

  // Perform validation
  if (name.trim() === '') {
    alert('Please enter your name.');
    nameInput.focus();
    return;
  }
  if (password.trim() === '') {
    alert('Please enter your password.');
    passwordInput.focus();
    return;
  }

  // Process the login data (e.g., send it to a server for authentication)
  // You can replace this code with your own logic
  console.log('Name:', name);
  console.log('Password:', password);

  // Reset the form fields
  loginForm.reset();
});

// Function to be called when the "Sign In" button is clicked
function signIn() {
  // Place any additional code you want to run when the "Sign In" button is
  console.log('Sign In button clicked!');
}
```

```
// Add a click event listener to the "Sign In" button
const signInButton = document.getElementById('signInButton');
signInButton.addEventListener('click', signIn);
```

In this updated code, we added a new function `signIn()` that will be called when the "Sign In" button is clicked. You can place any additional code or processing logic inside this function. In the example, I simply logged a message to the console.

To ensure the function is called, we added an `id` attribute to the `<script>` tag (`id="login-script"`) in the HTML file.

We then selected the "Sign In" button element using `document.getElementById('signInButton')` and added a click event listener to it. When the button is clicked, the `signIn()` function will be executed.

Make sure you have an element with the `id` attribute set to `signInButton` in your HTML code, like this:

html Copy code

```html
<button id="signInButton" type="submit">Sign In</button>
```

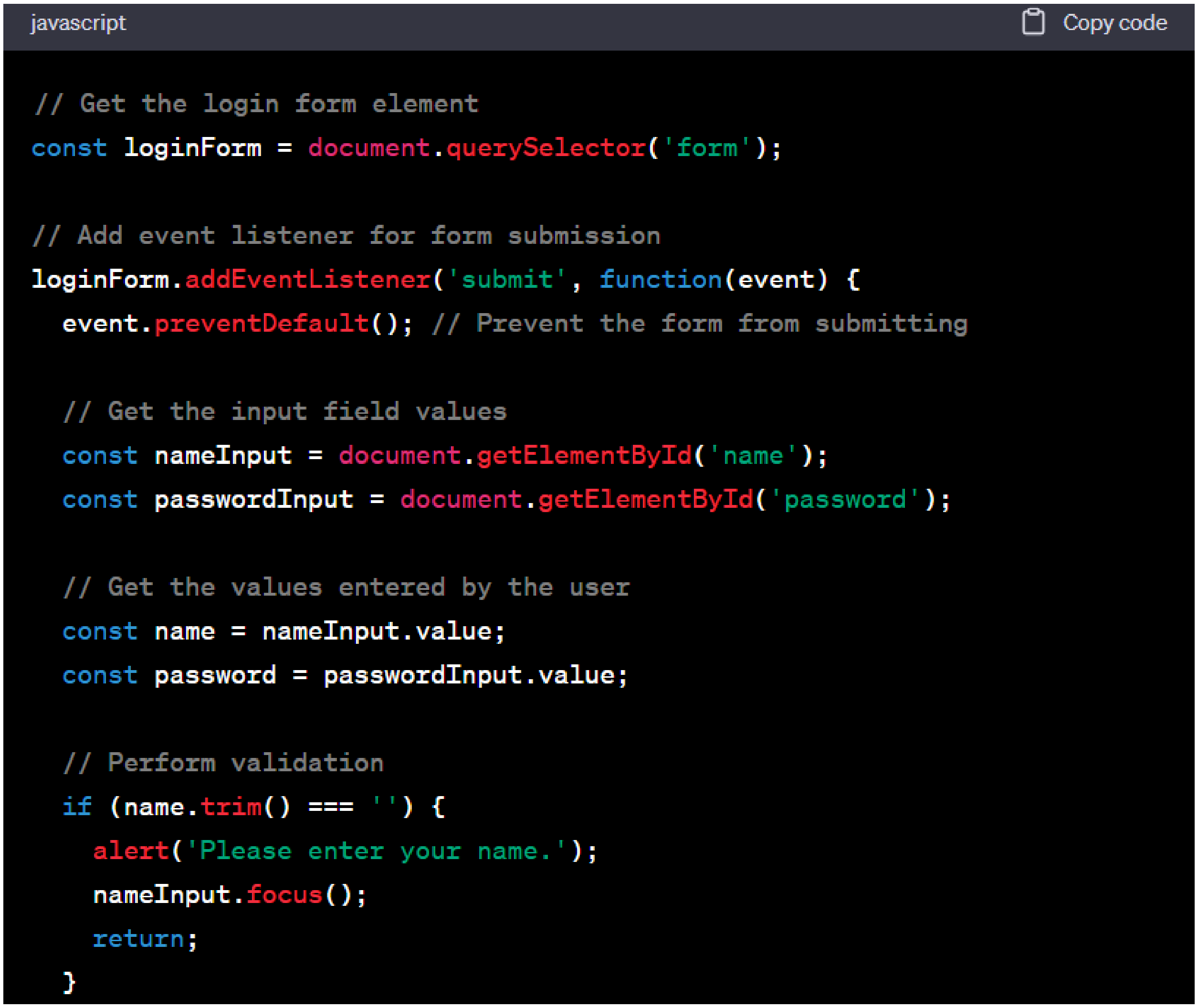
With this modification, when the "Sign In" button is clicked, it will trigger the `signIn()` function along with the existing code for form validation and processing.

Remember to adjust the `id` attributes and HTML elements in your code to match the changes made in the JavaScript code.

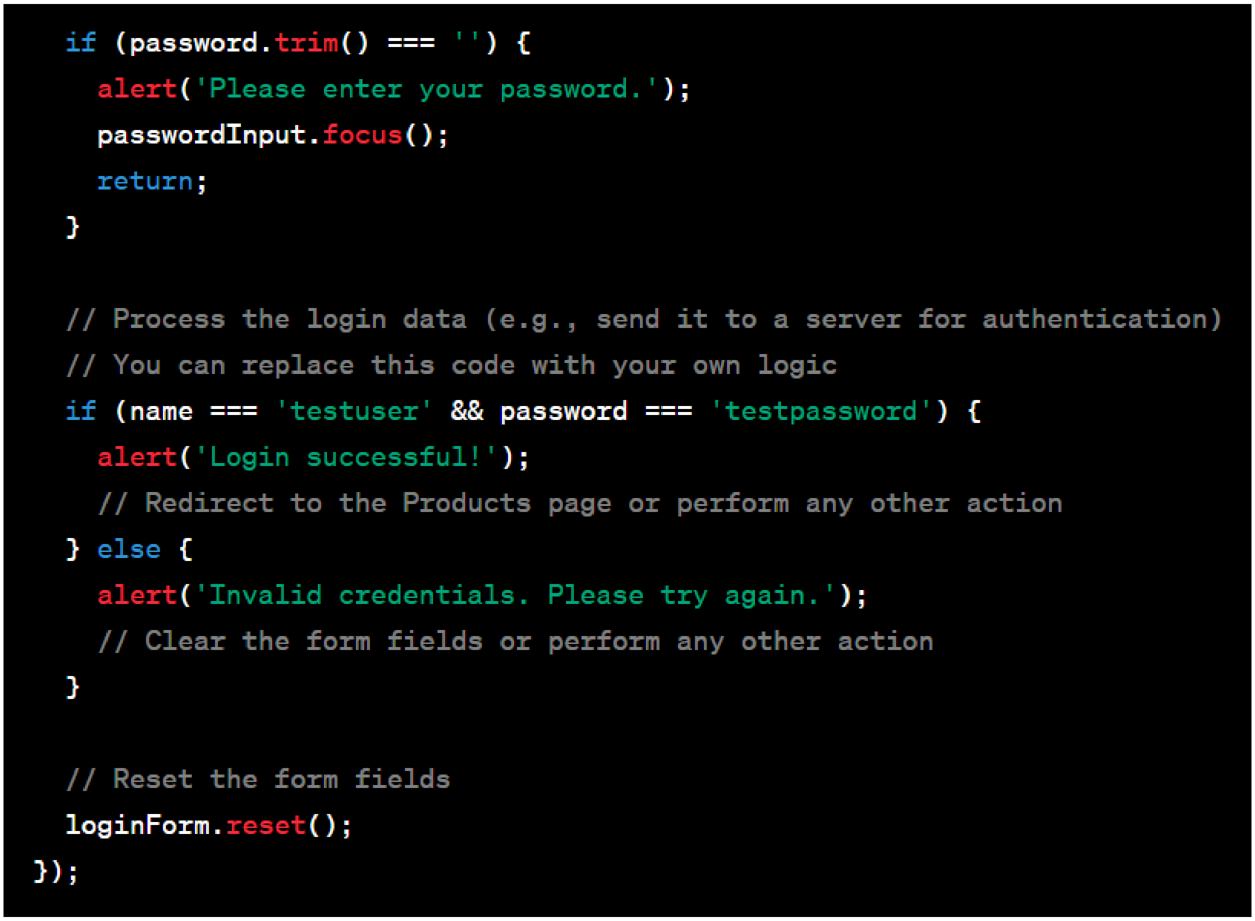
Note: If you prefer to keep the script tag in the head section, make sure to place the JavaScript code after the HTML elements are loaded, either by placing the `<script>` tag at the end of the `<body>` section or by wrapping the JavaScript code in an event listener that waits for the DOMContentLoaded event.

I need to test the login, where I set data on script to do the test?

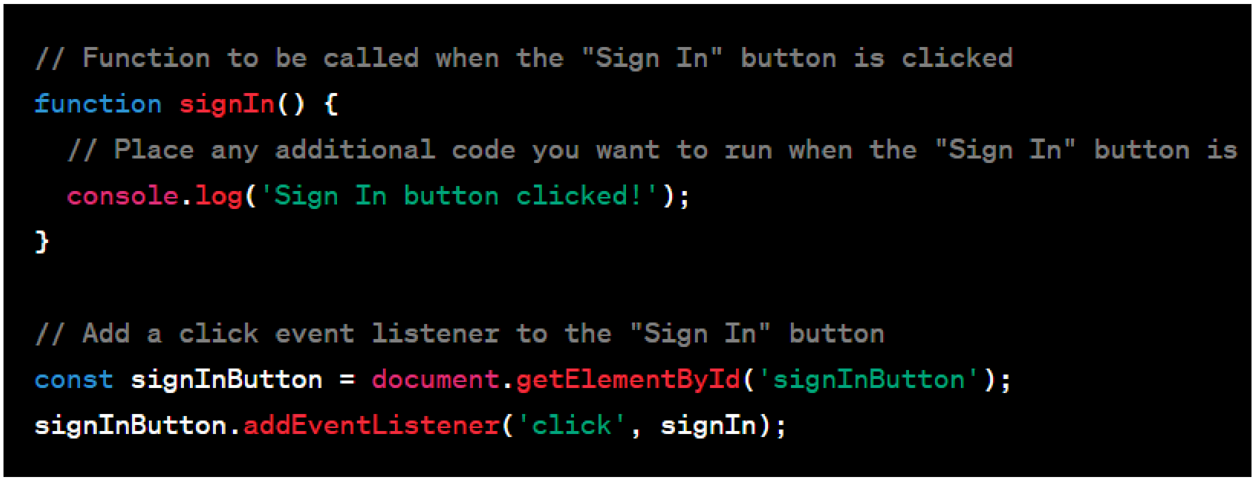
To test the login functionality, you can manually set some test data in your JavaScript code. Here's an example of how you can modify the code to include test credentials:



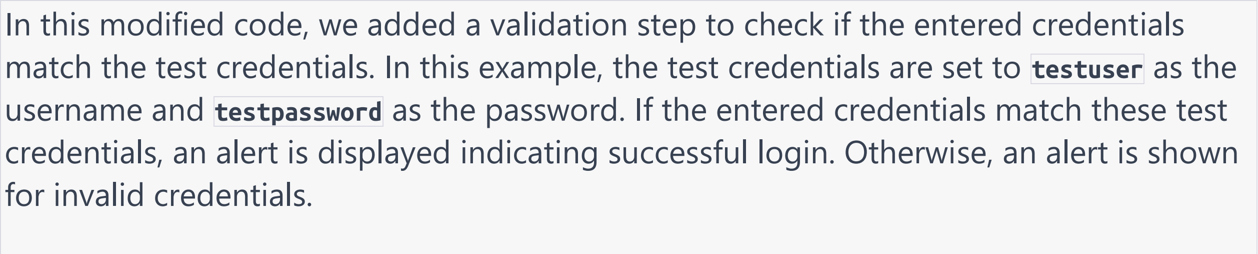
```javascript
// Get the login form element
const loginForm = document.querySelector('form');

// Add event listener for form submission
loginForm.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent the form from submitting

  // Get the input field values
  const nameInput = document.getElementById('name');
  const passwordInput = document.getElementById('password');

  // Get the values entered by the user
  const name = nameInput.value;
  const password = passwordInput.value;

  // Perform validation
  if (name.trim() === '') {
    alert('Please enter your name.');
    nameInput.focus();
    return;
  }
```



```
  if (password.trim() === '') {
    alert('Please enter your password.');
    passwordInput.focus();
    return;
  }

  // Process the login data (e.g., send it to a server for authentication)
  // You can replace this code with your own logic
  if (name === 'testuser' && password === 'testpassword') {
    alert('Login successful!');
    // Redirect to the Products page or perform any other action
  } else {
    alert('Invalid credentials. Please try again.');
    // Clear the form fields or perform any other action
  }

  // Reset the form fields
  loginForm.reset();
});
```



```
// Function to be called when the "Sign In" button is clicked
function signIn() {
  // Place any additional code you want to run when the "Sign In" button is
  console.log('Sign In button clicked!');
}

// Add a click event listener to the "Sign In" button
const signInButton = document.getElementById('signInButton');
signInButton.addEventListener('click', signIn);
```



In this modified code, we added a validation step to check if the entered credentials match the test credentials. In this example, the test credentials are set to `testuser` as the username and `testpassword` as the password. If the entered credentials match these test credentials, an alert is displayed indicating successful login. Otherwise, an alert is shown for invalid credentials.

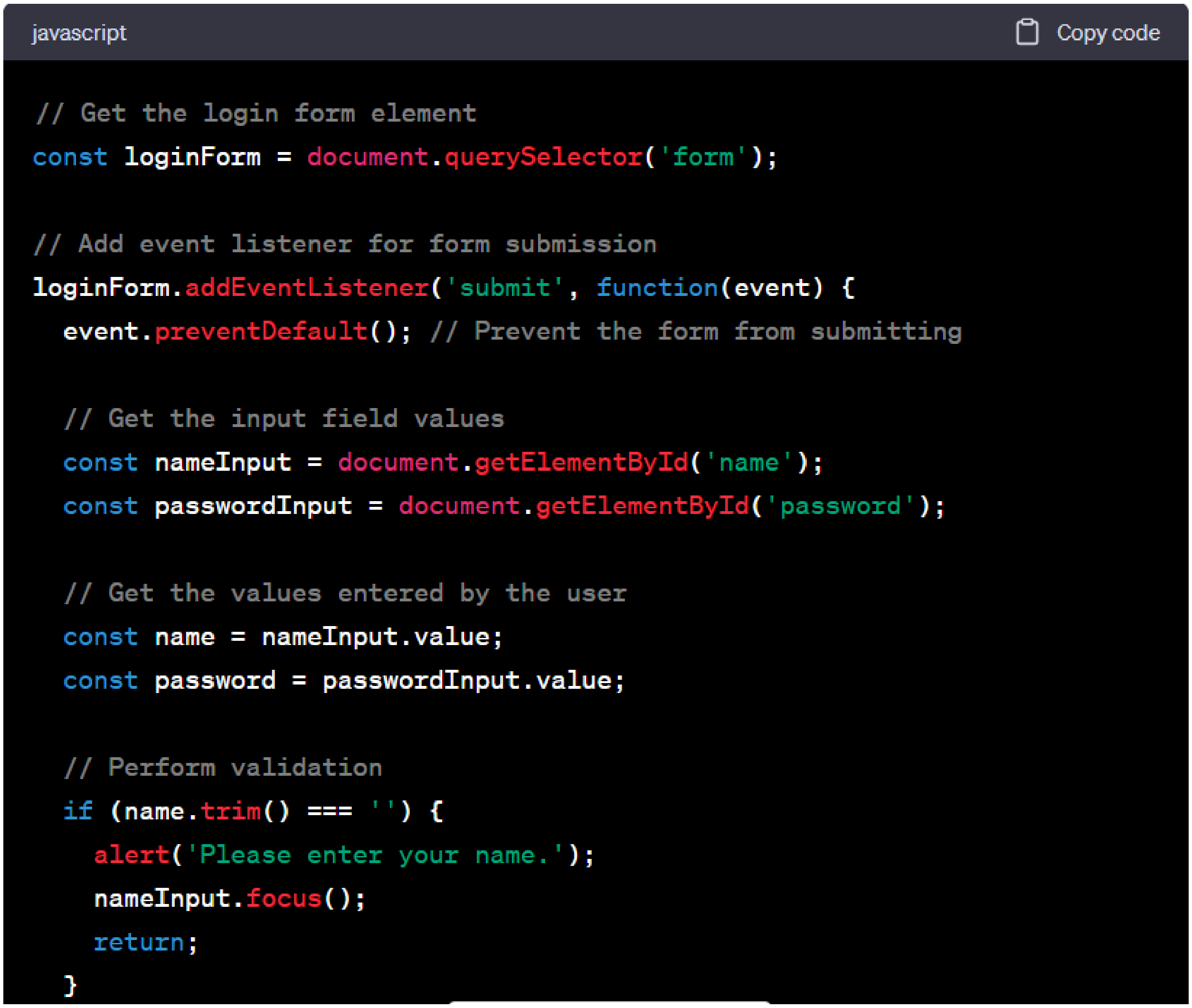
You can customize the test credentials by changing the values in the `if` statement. You can also add more complex logic, such as checking against a database or an API, to handle the actual login authentication.

Remember to remove or update the test credentials and replace the validation logic with your own authentication mechanism when deploying the code in a production environment.

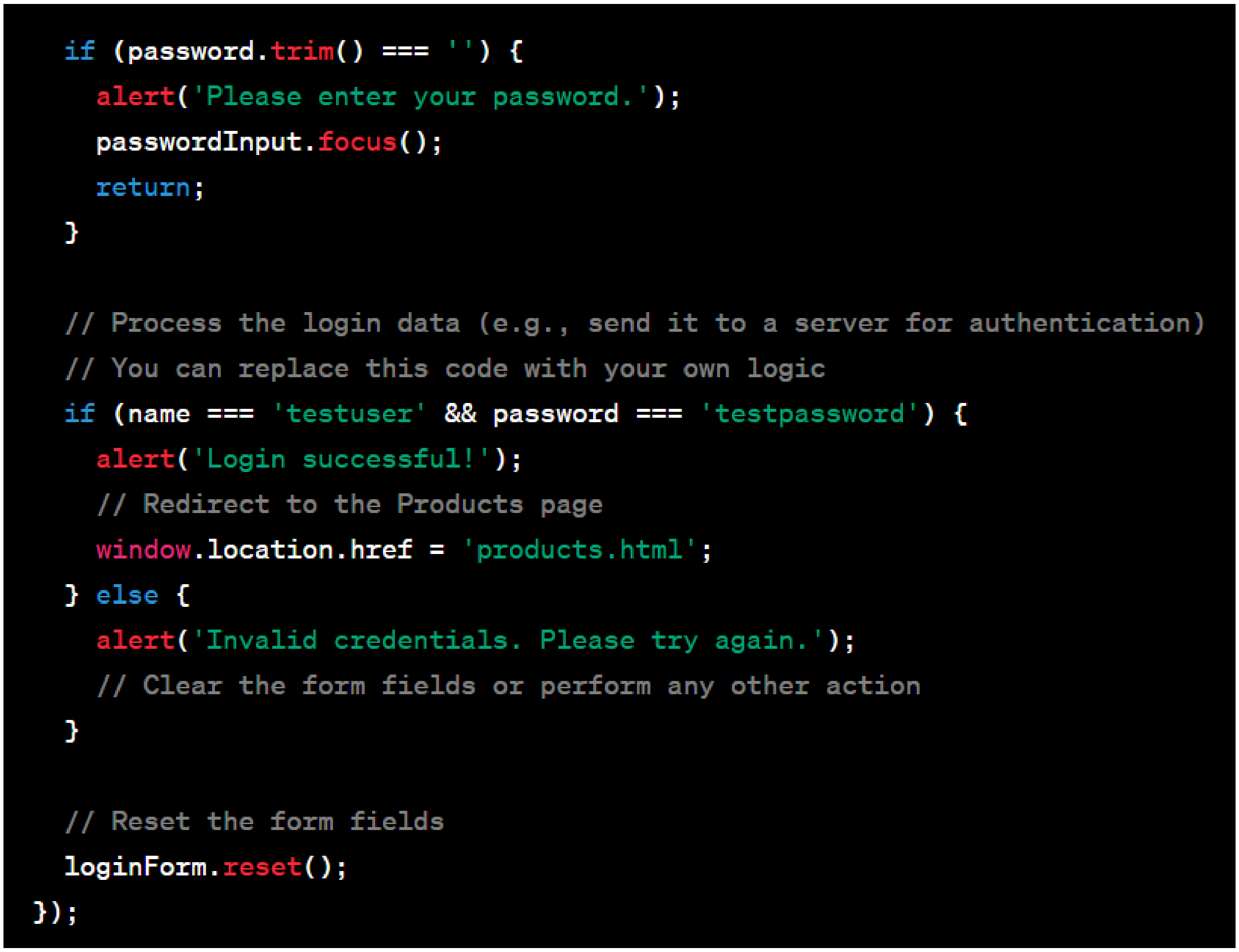
When testing, enter the test credentials (`testuser` and `testpassword`) into the login form and click the "Sign In" button to simulate a login attempt.

I have sucessful on Login, but they don't open the products page.

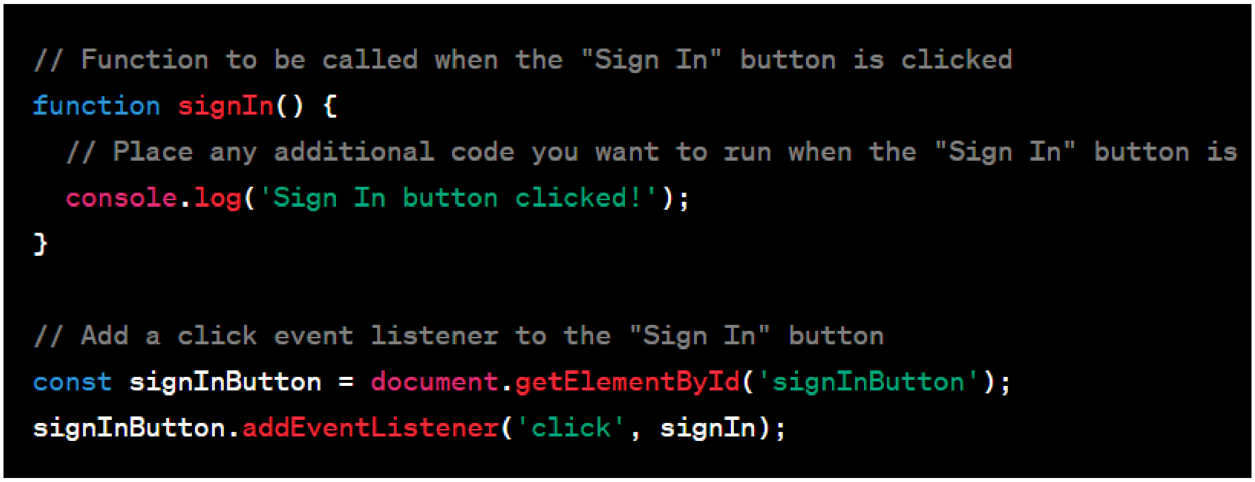
To redirect the user to the products page after a successful login, you can use the `window.location` object to set the URL of the products page. Here's an updated version of the JavaScript code that includes the redirection:



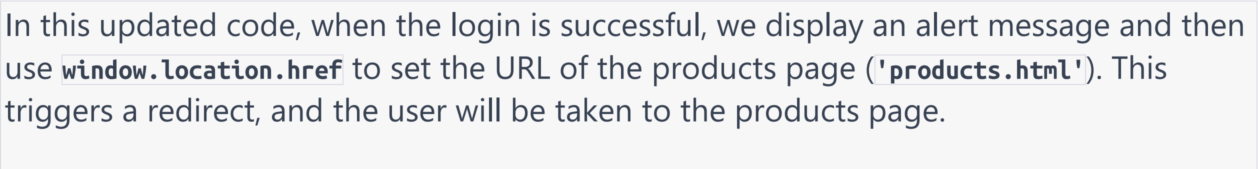
```javascript
// Get the login form element
const loginForm = document.querySelector('form');

// Add event listener for form submission
loginForm.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent the form from submitting

  // Get the input field values
  const nameInput = document.getElementById('name');
  const passwordInput = document.getElementById('password');

  // Get the values entered by the user
  const name = nameInput.value;
  const password = passwordInput.value;

  // Perform validation
  if (name.trim() === '') {
    alert('Please enter your name.');
    nameInput.focus();
    return;
  }
```



```
  if (password.trim() === '') {
    alert('Please enter your password.');
    passwordInput.focus();
    return;
  }

  // Process the login data (e.g., send it to a server for authentication)
  // You can replace this code with your own logic
  if (name === 'testuser' && password === 'testpassword') {
    alert('Login successful!');
    // Redirect to the Products page
    window.location.href = 'products.html';
  } else {
    alert('Invalid credentials. Please try again.');
    // Clear the form fields or perform any other action
  }

  // Reset the form fields
  loginForm.reset();
});
```



```
// Function to be called when the "Sign In" button is clicked
function signIn() {
  // Place any additional code you want to run when the "Sign In" button is
  console.log('Sign In button clicked!');
}

// Add a click event listener to the "Sign In" button
const signInButton = document.getElementById('signInButton');
signInButton.addEventListener('click', signIn);
```



In this updated code, when the login is successful, we display an alert message and then use `window.location.href` to set the URL of the products page (`'products.html'`). This triggers a redirect, and the user will be taken to the products page.

Make sure that you have a separate HTML file named `products.html` containing the products page content.

If the redirect is not working, please ensure that the file name and path are correct, and that the products page (`products.html`) is located in the same directory as the login page. Adjust the URL path as needed to match the location of your products page.

Once the redirection is set up correctly, a successful login will trigger the redirect to the products page.

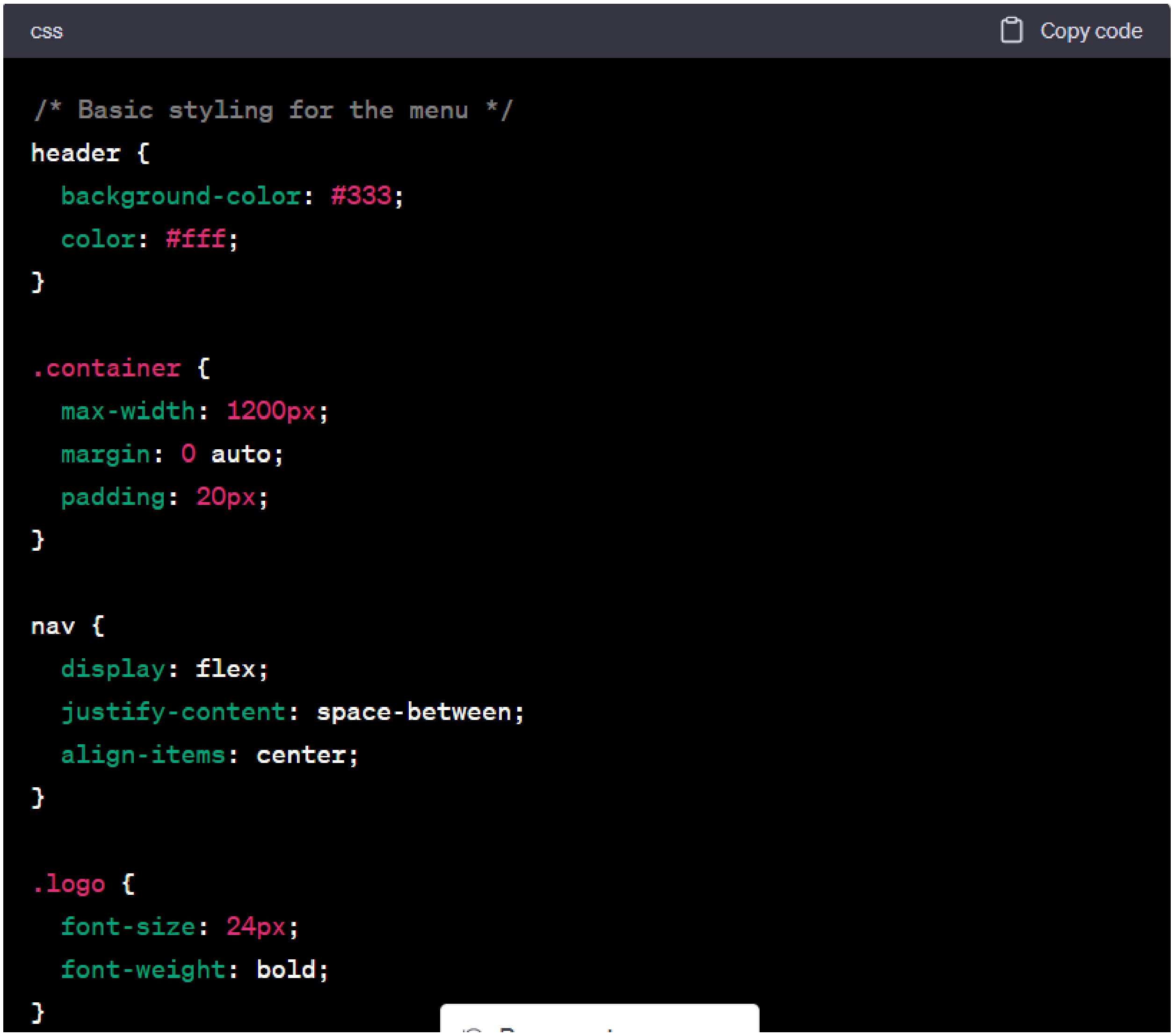
I forgot to insert a menu on my website, I would like to insert a responsive menu in all pages

To add a responsive menu to your website, you can modify the HTML code and CSS to include a menu structure and style it accordingly. Here's an example of how you can add a responsive menu using HTML and CSS:

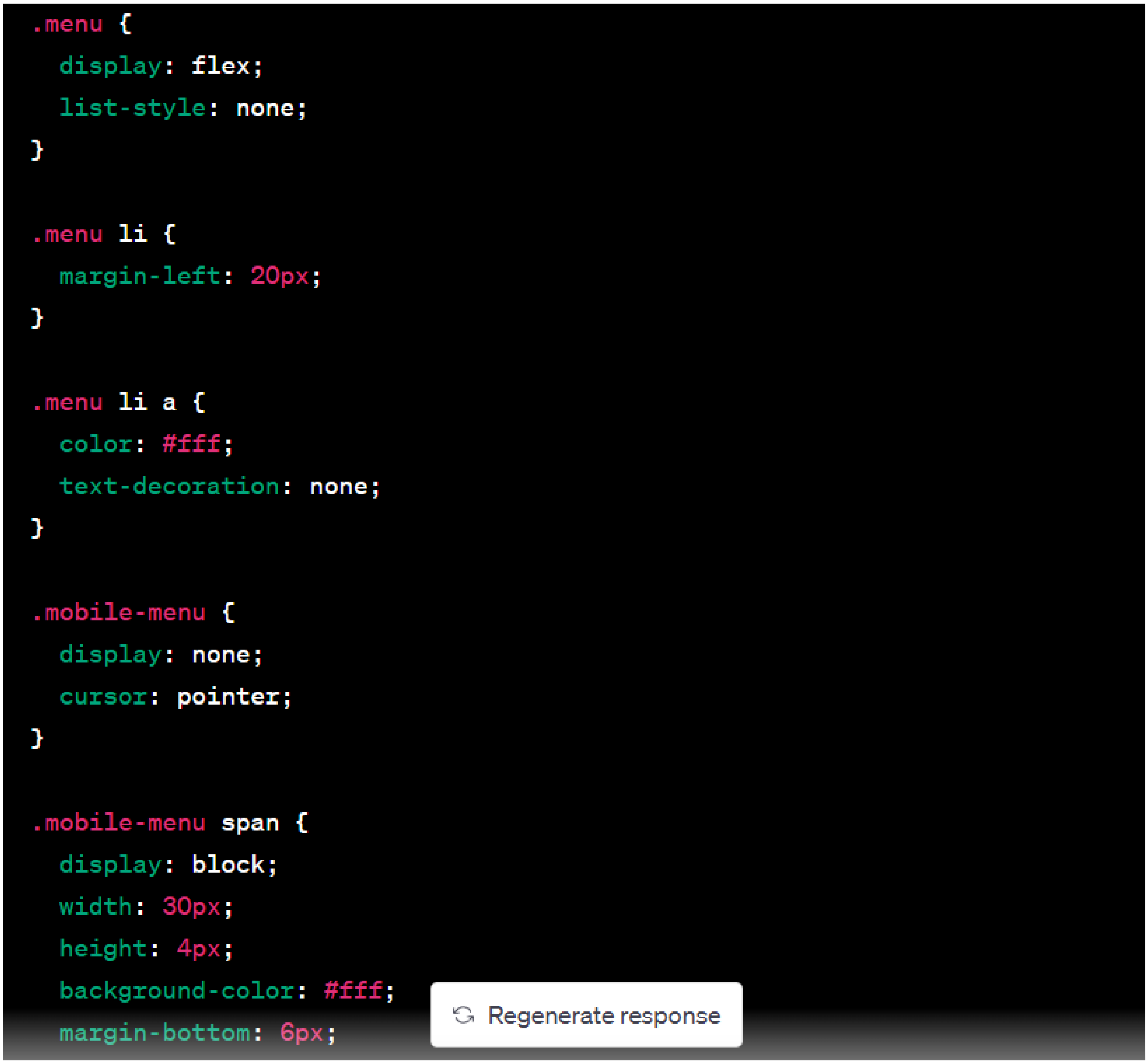
HTML (header section of each page):

html Copy code

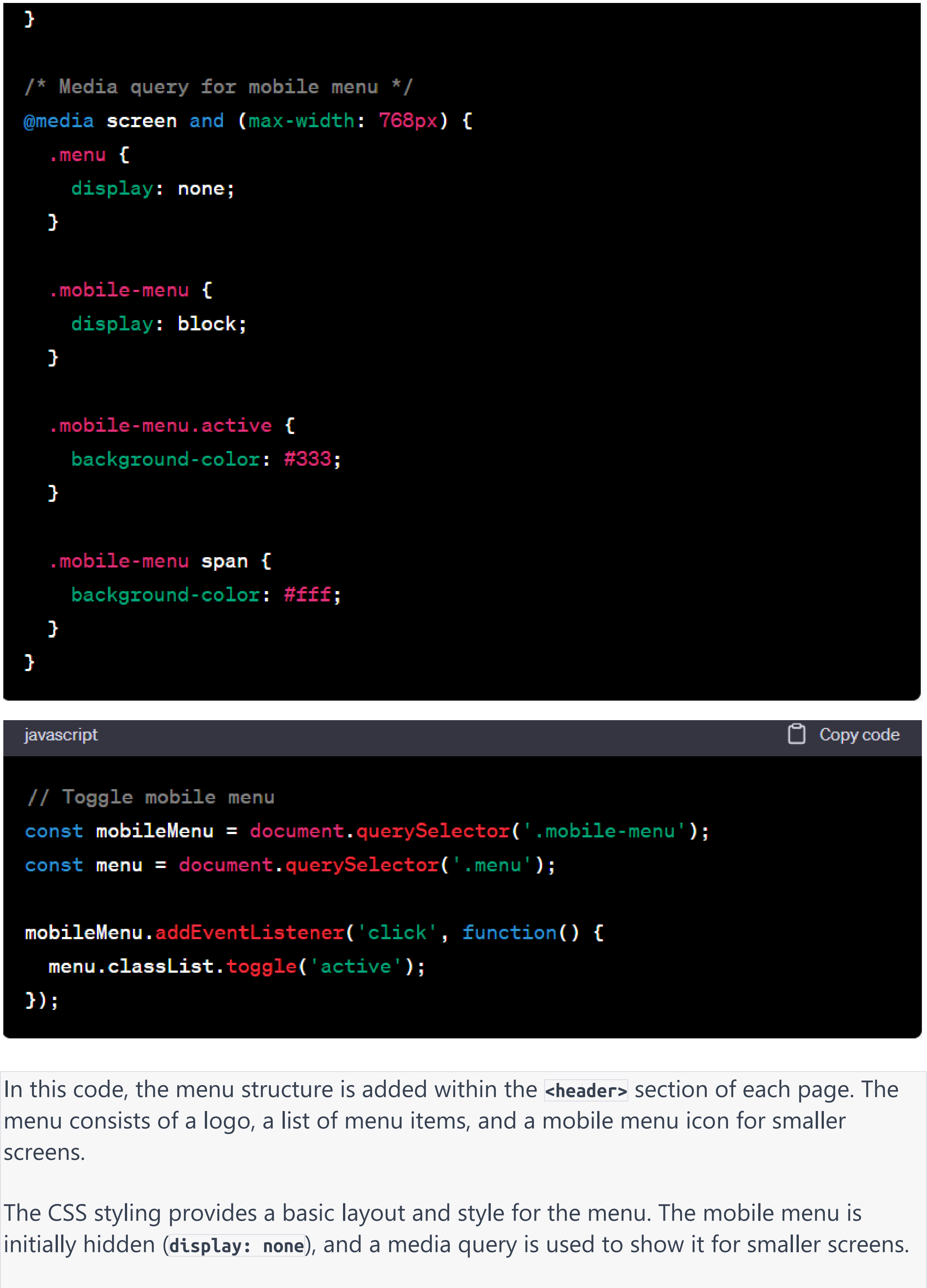
```html
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
  <script src="script.js" id="login-script"></script>
</head>
<body>
  <header>
    <div class="container">
      <nav>
        <div class="logo">
          <!-- Insert your logo here -->
        </div>
        <ul class="menu">
          <li><a href="login.html">Login</a></li>
          <li><a href="products.html">Products</a></li>
          <li><a href="forgot-password.html">Forgot Password</a></li>
        </ul>
        <div class="mobile-menu">
          <span></span>
          <span></span>
          <span></span>
        </div>
      </nav>
    </div>
  </header>

  <!-- Rest of your HTML content -->

</body>
</html>
```



```css
/* Basic styling for the menu */
header {
  background-color: #333;
  color: #fff;
}

.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}

nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.logo {
  font-size: 24px;
  font-weight: bold;
}
```



```
.menu {
  display: flex;
  list-style: none;
}

.menu li {
  margin-left: 20px;
}

.menu li a {
  color: #fff;
  text-decoration: none;
}

.mobile-menu {
  display: none;
  cursor: pointer;
}

.mobile-menu span {
  display: block;
  width: 30px;
  height: 4px;
  background-color: #fff;
  margin-bottom: 6px;
```



```
}

/* Media query for mobile menu */
@media screen and (max-width: 768px) {
  .menu {
    display: none;
  }

  .mobile-menu {
    display: block;
  }

  .mobile-menu.active {
    background-color: #333;
  }

  .mobile-menu span {
    background-color: #fff;
  }
}
```

```javascript
// Toggle mobile menu
const mobileMenu = document.querySelector('.mobile-menu');
const menu = document.querySelector('.menu');

mobileMenu.addEventListener('click', function() {
  menu.classList.toggle('active');
});
```

In this code, the menu structure is added within the `<header>` section of each page. The menu consists of a logo, a list of menu items, and a mobile menu icon for smaller screens.

The CSS styling provides a basic layout and style for the menu. The mobile menu is initially hidden (`display: none`), and a media query is used to show it for smaller screens.

The JavaScript code adds functionality to toggle the mobile menu when the menu icon is clicked. When the icon is clicked, the `active` class is toggled on the menu, and the menu is shown/hidden accordingly.

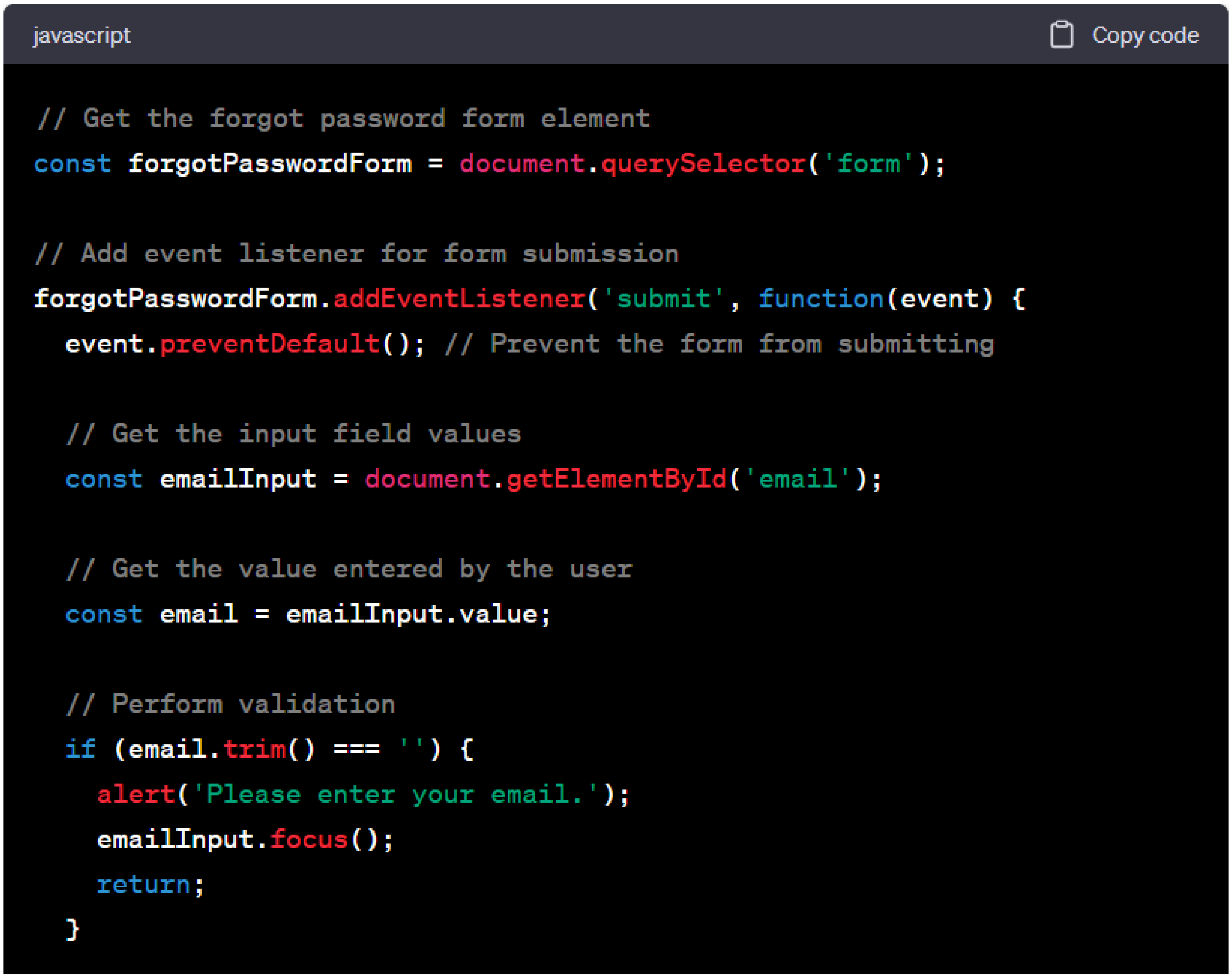
You can customize the menu's styling and structure based on your needs. Additionally, you can extend the JavaScript code to handle further menu interactions or animations as desired.

Remember to adjust the file paths for the CSS and JavaScript files according to your directory structure.

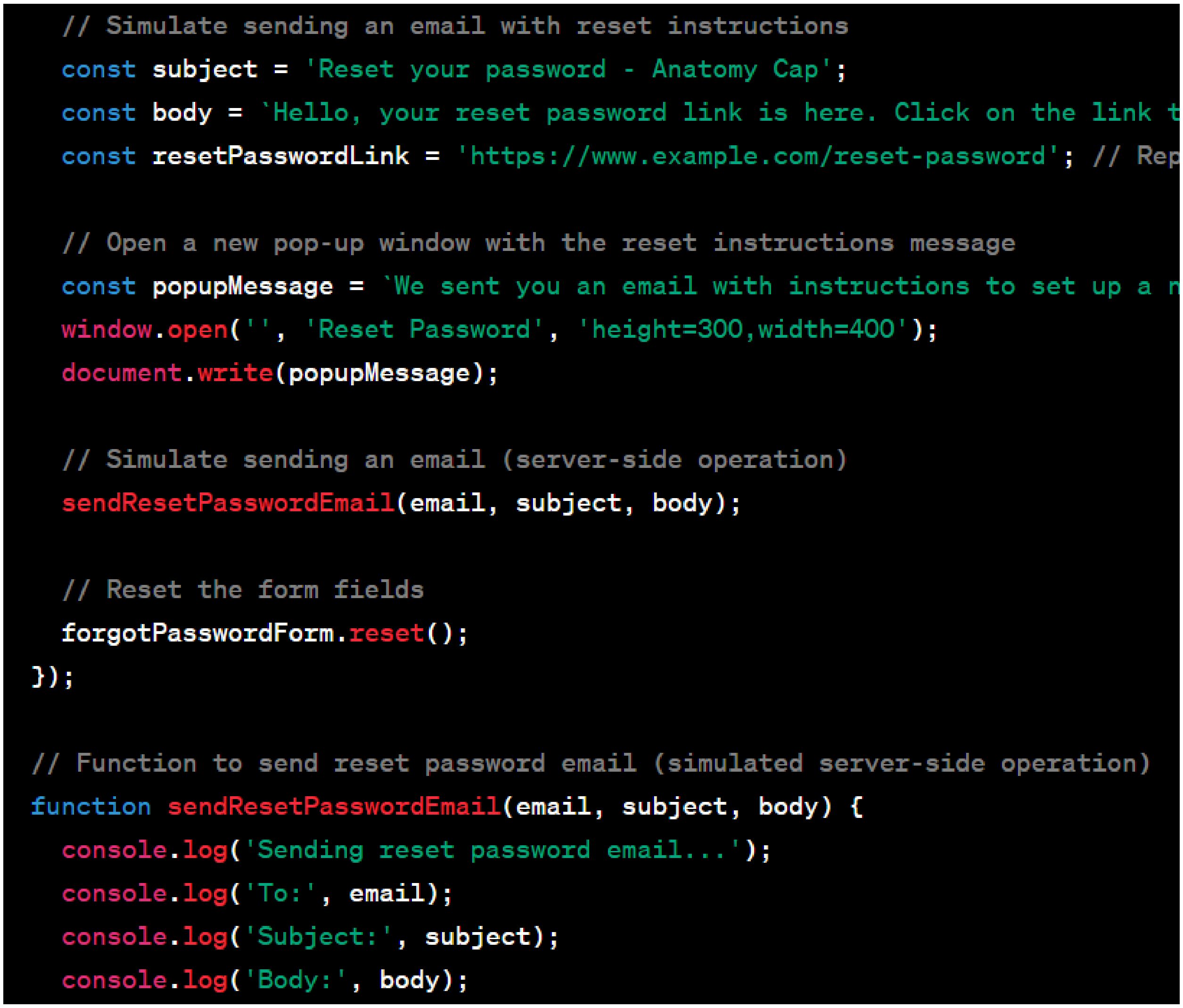
By incorporating this menu code, you will have a responsive menu that can be displayed on all pages of your website.

I want to click on Reset Password button, and they shows up a new pop window with this message "We sent you a e-mail with the instructions to set up a new password" and they send e-mail to informed e-mail on field where on the subject e-mail is Reset your password - Anatomy Cap and body e-mail need to have this message "Hello your reset password link is here, click on link and reset a new password" and this link Reset your Password when you click on it they open a new pop up window with two password fields New Password and Confirmation New Password and they need to validate both field where the values need to have the same information.

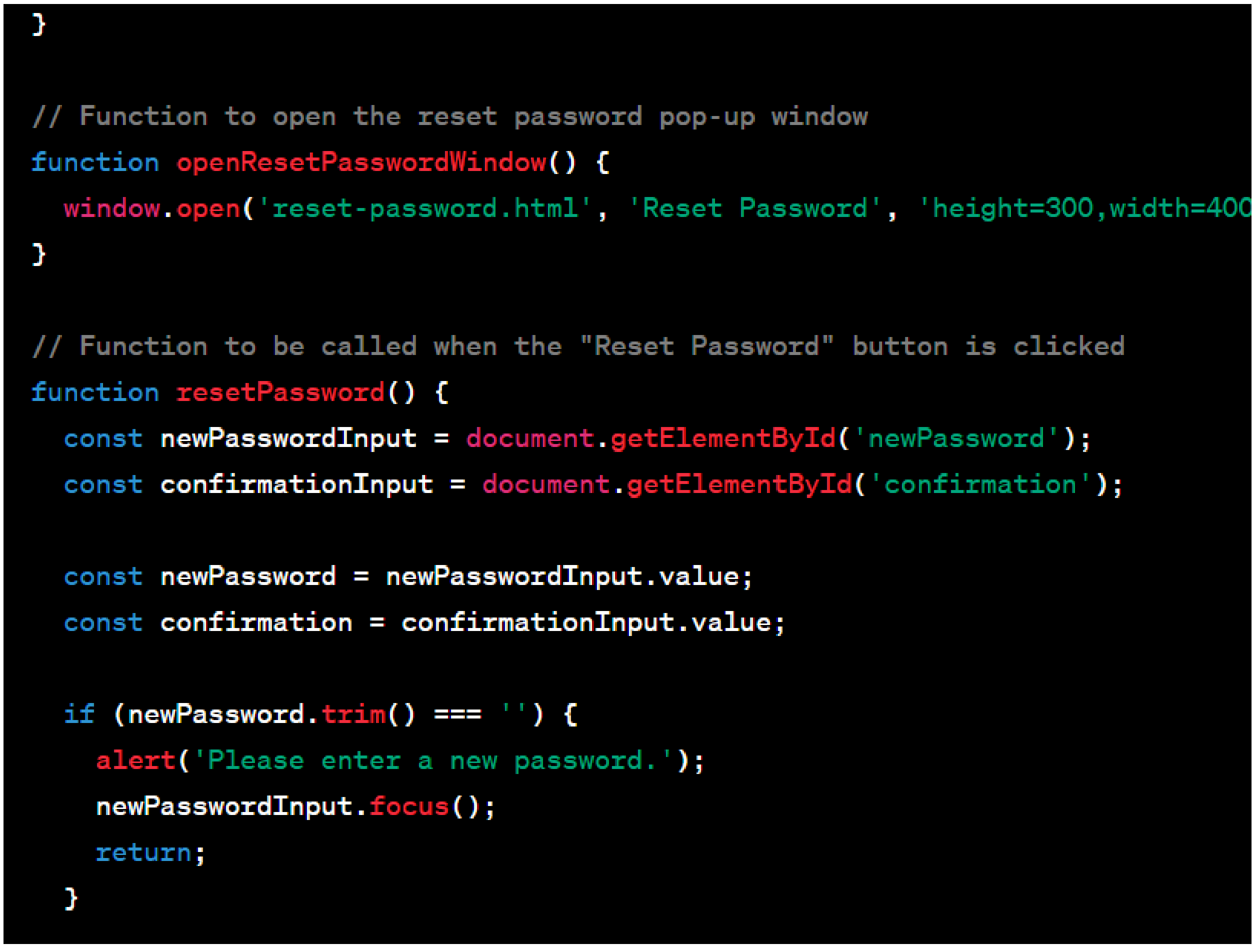
To implement the password reset functionality with email sending and password confirmation, you will need to incorporate additional code and utilize server-side operations. The given response can guide you on how to structure the client-side code for the user interface. However, keep in mind that the server-side implementation for sending emails and processing password resets requires a backend server with appropriate configurations. Below is an updated version of the JavaScript code with the described functionality:



```javascript
// Get the forgot password form element
const forgotPasswordForm = document.querySelector('form');

// Add event listener for form submission
forgotPasswordForm.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent the form from submitting

  // Get the input field values
  const emailInput = document.getElementById('email');

  // Get the value entered by the user
  const email = emailInput.value;

  // Perform validation
  if (email.trim() === '') {
    alert('Please enter your email.');
    emailInput.focus();
    return;
  }
```



```
  // Simulate sending an email with reset instructions
  const subject = 'Reset your password - Anatomy Cap';
  const body = `Hello, your reset password link is here. Click on the link t
  const resetPasswordLink = 'https://www.example.com/reset-password'; // Rep

  // Open a new pop-up window with the reset instructions message
  const popupMessage = `We sent you an email with instructions to set up a n
  window.open('', 'Reset Password', 'height=300,width=400');
  document.write(popupMessage);

  // Simulate sending an email (server-side operation)
  sendResetPasswordEmail(email, subject, body);

  // Reset the form fields
  forgotPasswordForm.reset();
});

// Function to send reset password email (simulated server-side operation)
function sendResetPasswordEmail(email, subject, body) {
  console.log('Sending reset password email...');
  console.log('To:', email);
  console.log('Subject:', subject);
  console.log('Body:', body);
```



```
}

// Function to open the reset password pop-up window
function openResetPasswordWindow() {
  window.open('reset-password.html', 'Reset Password', 'height=300,width=400
}

// Function to be called when the "Reset Password" button is clicked
function resetPassword() {
  const newPasswordInput = document.getElementById('newPassword');
  const confirmationInput = document.getElementById('confirmation');

  const newPassword = newPasswordInput.value;
  const confirmation = confirmationInput.value;

  if (newPassword.trim() === '') {
    alert('Please enter a new password.');
    newPasswordInput.focus();
    return;
  }
  if (confirmation.trim() === '') {
    alert('Please confirm the new password.');
    confirmationInput.focus();
    return;
  }

  if (newPassword !== confirmation) {
    alert('New password and confirmation password do not match.');
    newPasswordInput.value = '';
    confirmationInput.value = '';
    newPasswordInput.focus();
    return;
  }

  alert('Password reset successful!');
  // Perform any necessary actions (e.g., update password in the backend)

  // Close the reset password pop-up window
  window.close();
}
```

```
// Add a click event listener to the "Reset Password" button
const resetPasswordButton = document.getElementById('resetPasswordButton');
resetPasswordButton.addEventListener('click', resetPassword);
```

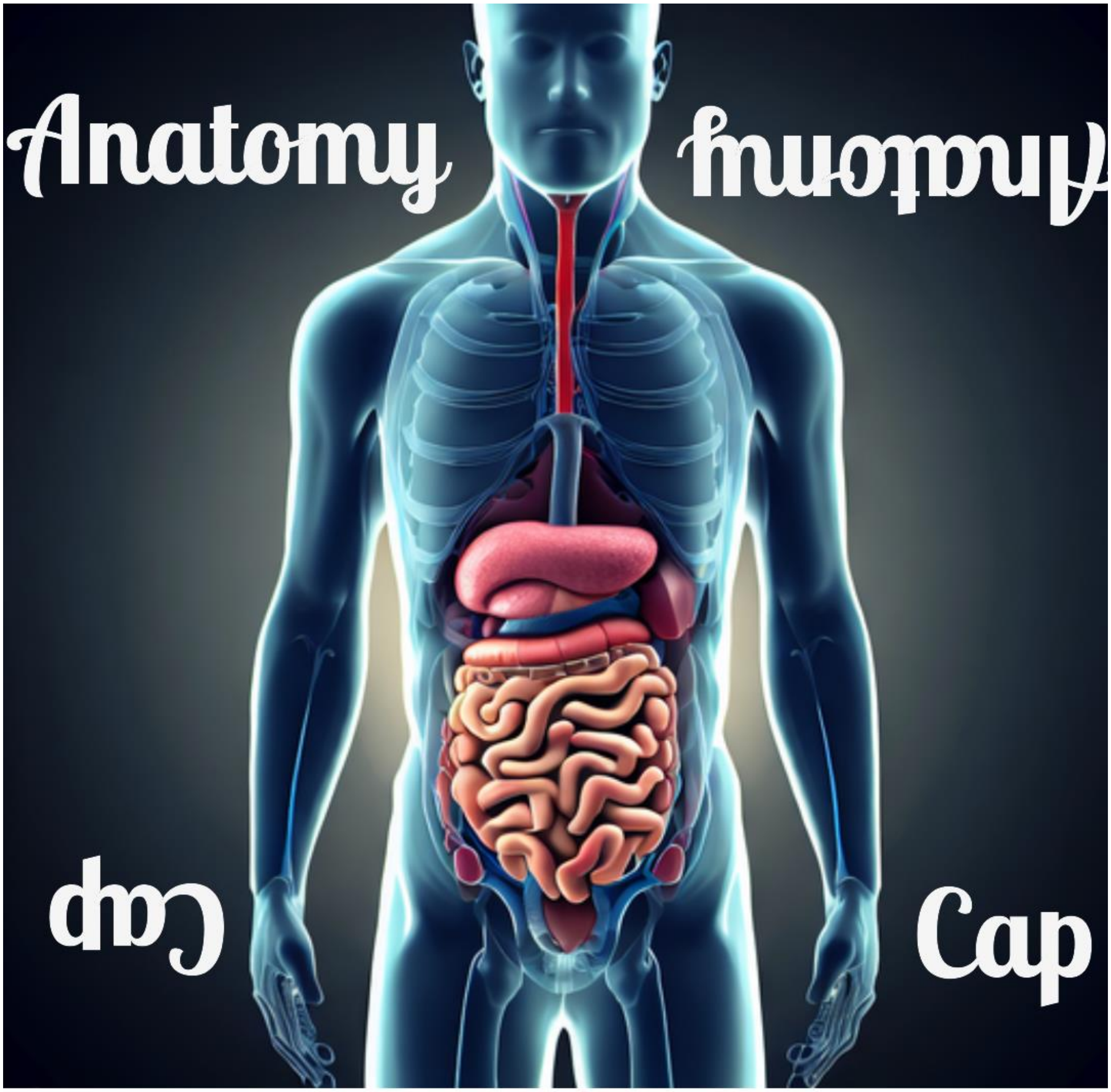
Please note that the above code only demonstrates the client-side functionality, and the server-side implementation for sending emails and processing password resets is not included. You will need to implement server-side code using a server-side language (e.g., Node.js, PHP, Python) and configure email services to handle sending emails.

Additionally, you need to create a separate HTML file for the reset password pop-up window (`reset-password.html`) and include the necessary password fields (`newPassword` and `confirmation`) and the "Reset Password" button (`resetPasswordButton`). Make sure to link this HTML file appropriately in your code.

Remember to customize the code according to your specific requirements and integrate it with your existing HTML and CSS.

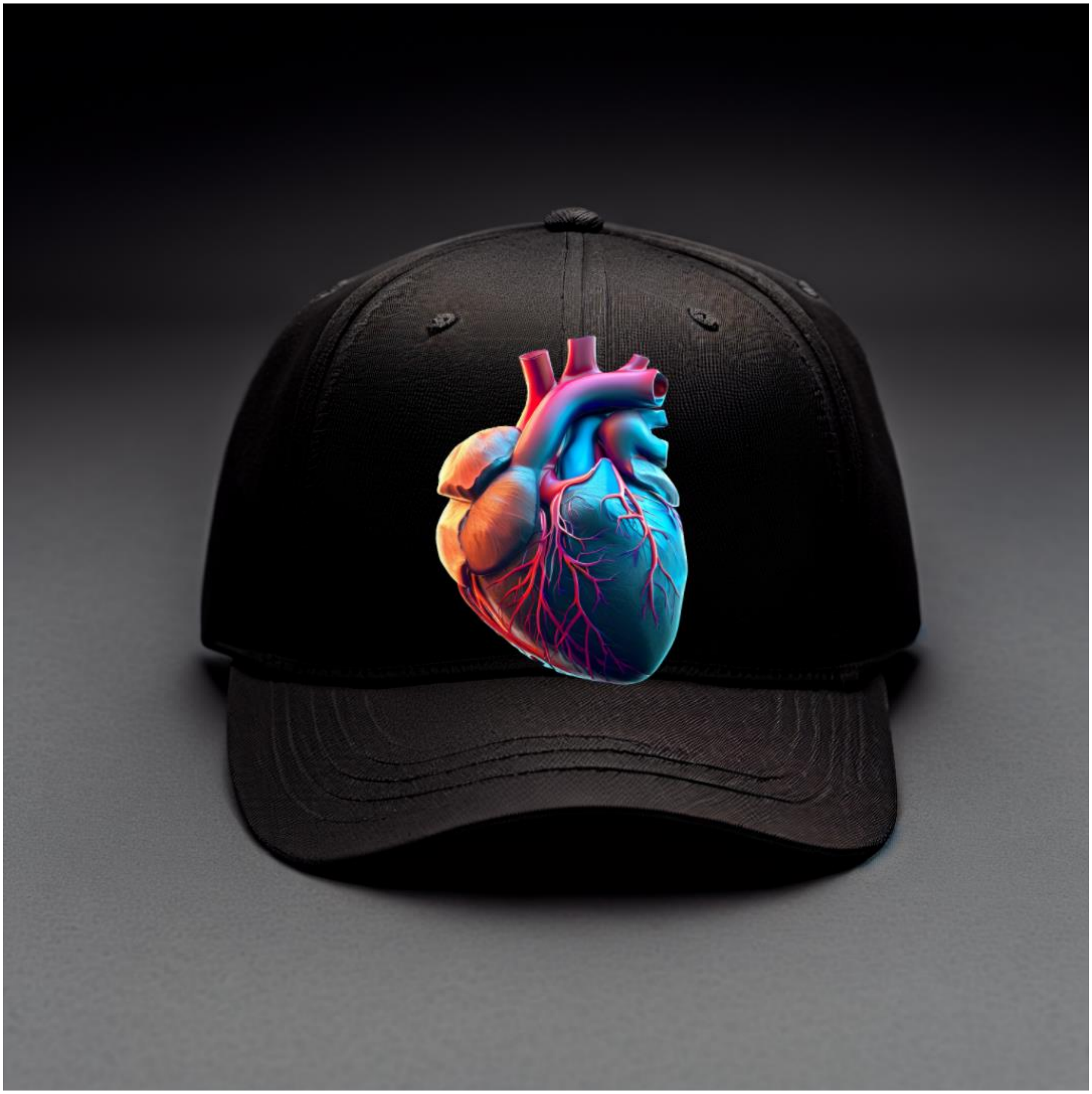
Image Prompt – Microsoft Designer Generate

Create a Logo brand with Realistic Full Human Body 3D image and variance of Anatomy Cap Text;

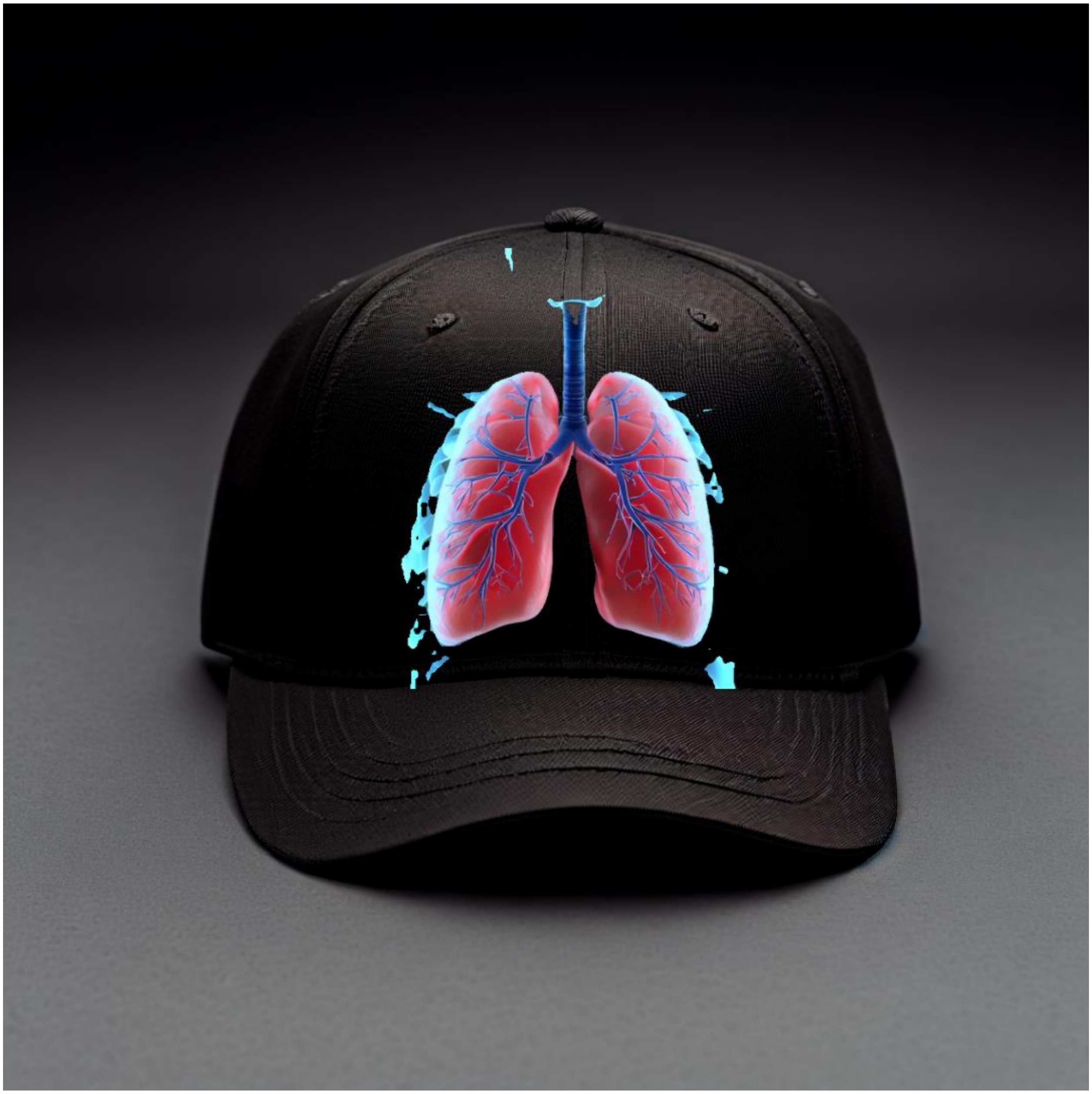


Black cap in high resolution;

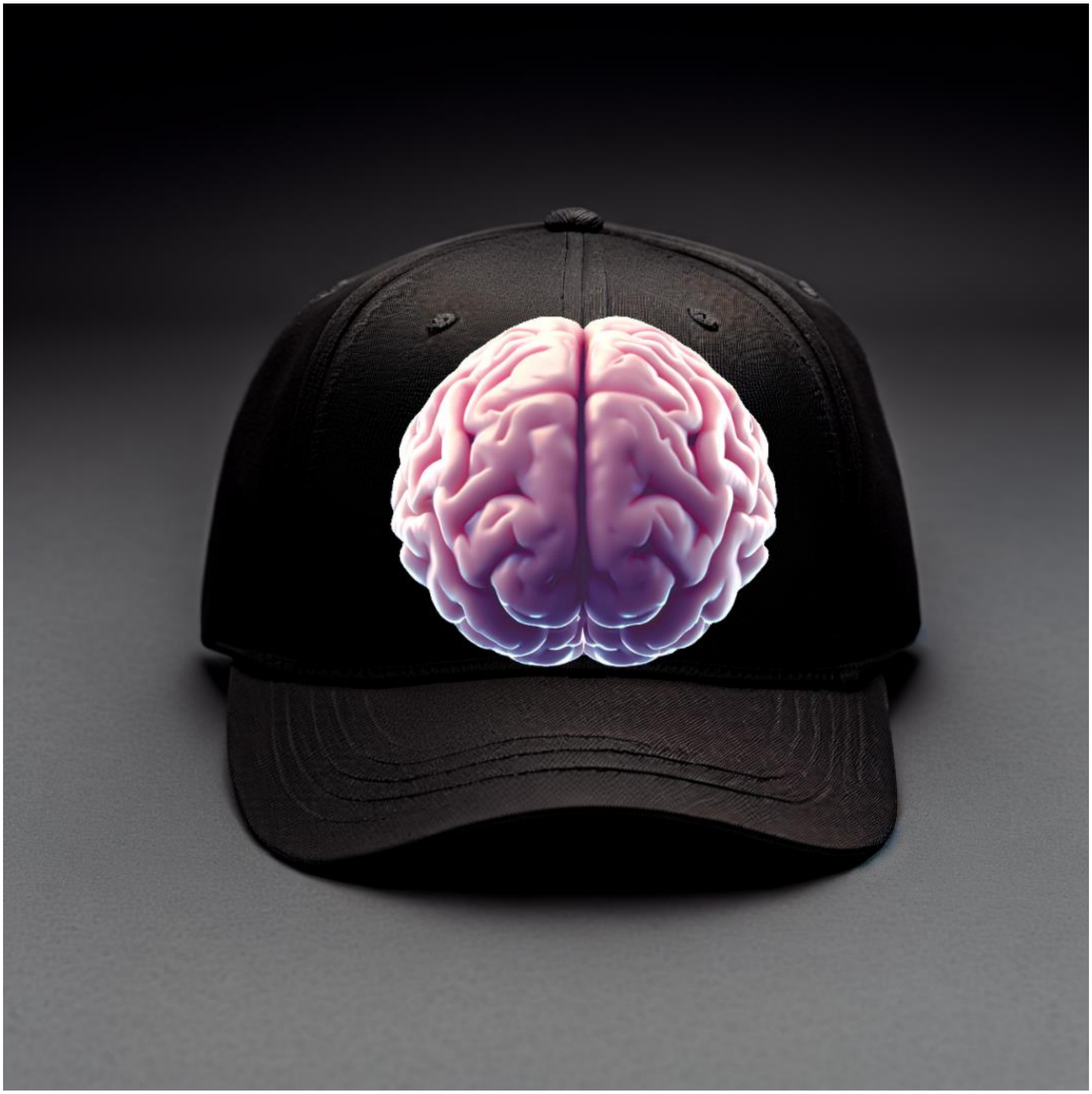
Realistic Human Heart 3D image front;



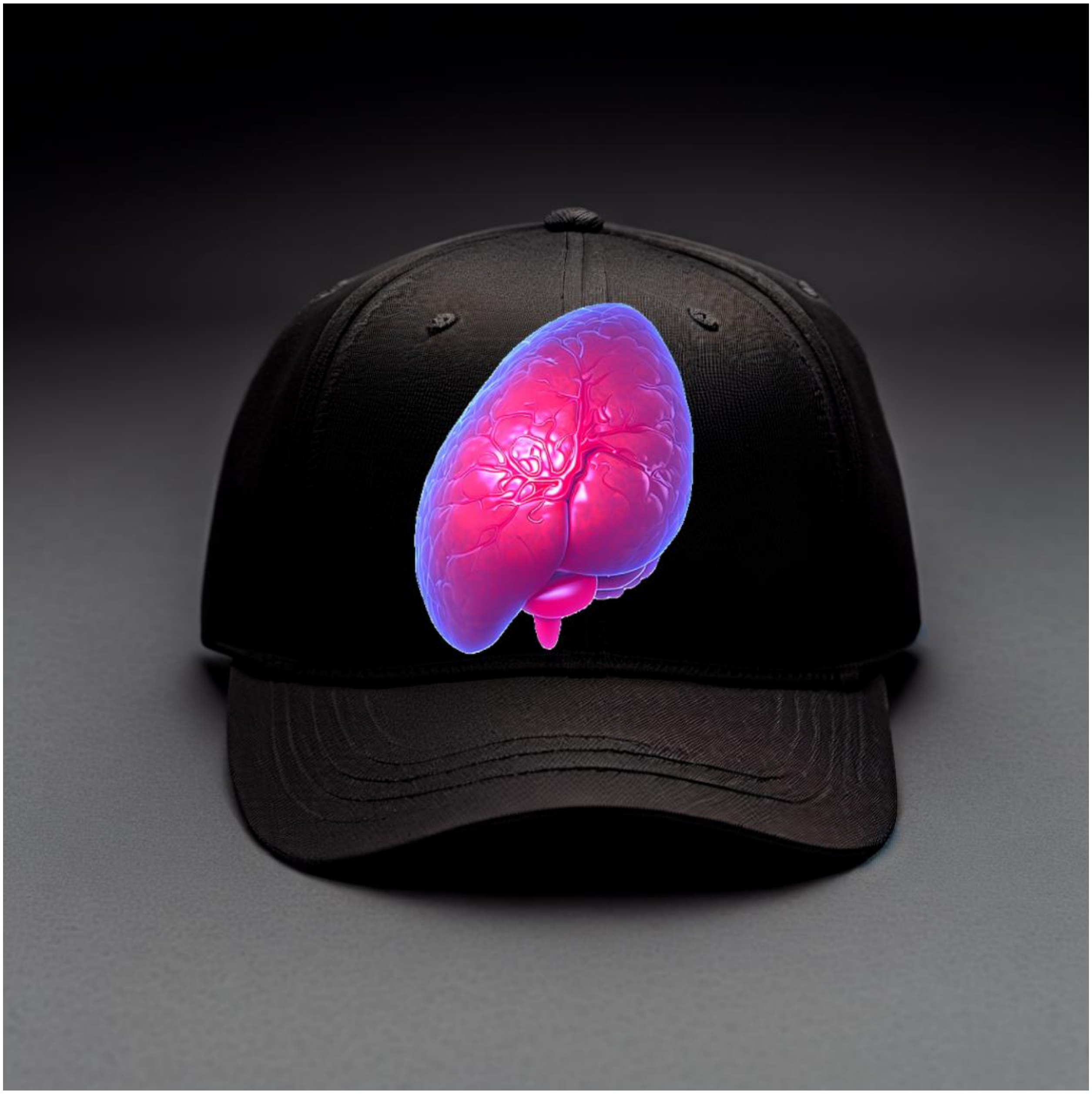
Realistic Human Lungs 3D image front;



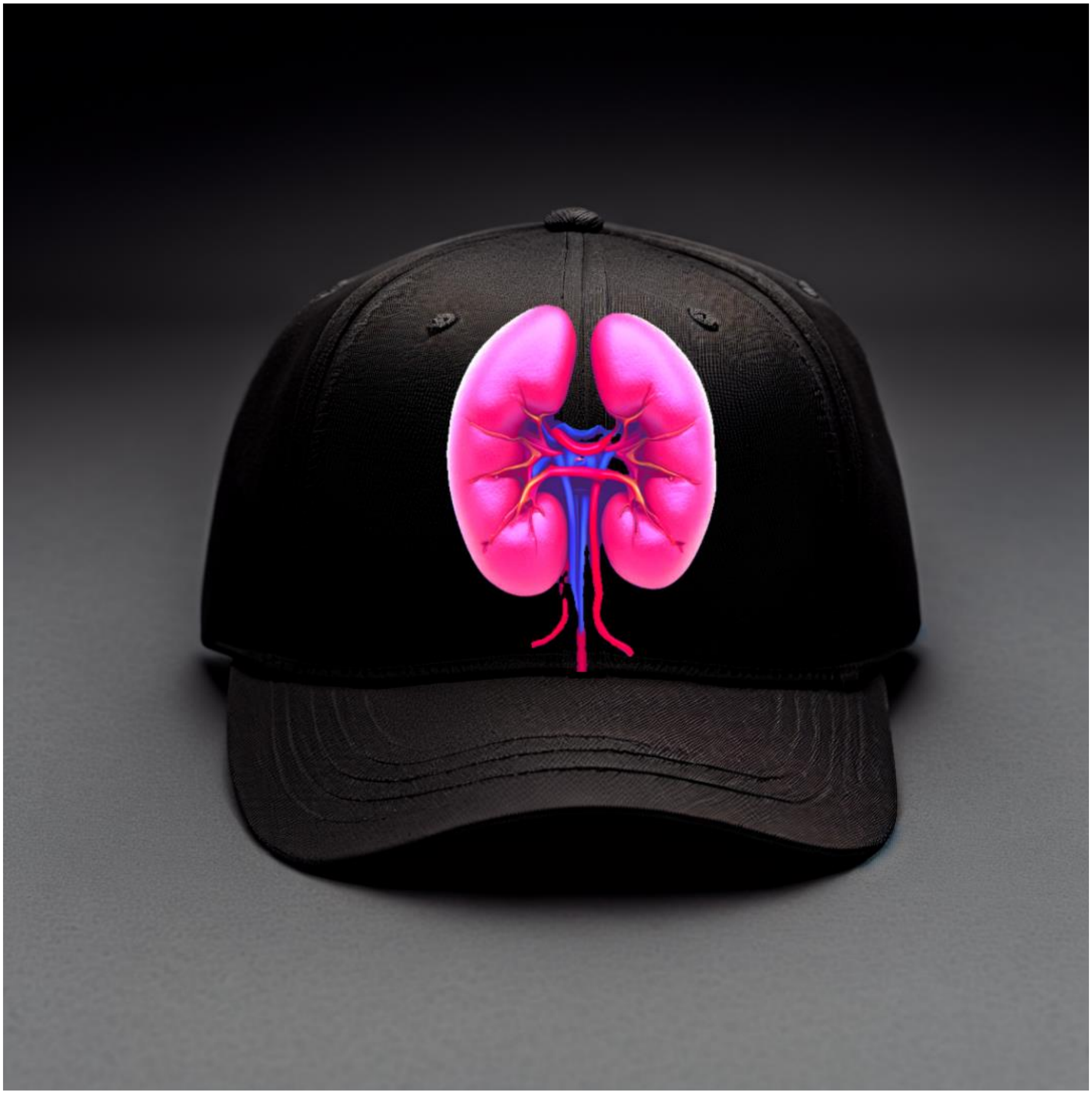
Realistic Human Brain 3D image front;



Realistic Human Liver 3D image front;



Realistic Human Kidneys 3D image front;



Realistic Human Stomach 3D image front;

Realistic Human Spleen 3D image front;

Realistic Human Large Intestine 3D image front;

Realistic Human Small Intestine 3D image front;

Realistic Human Pancreas 3D image front;