**Web Application Workshop**

**Session -2**

1. **Introduction**

First, let's clone(copy) the project files from session 1. Go to the Desktop and right-click to select ‘Open with Code’. This will open VSCode in this location.

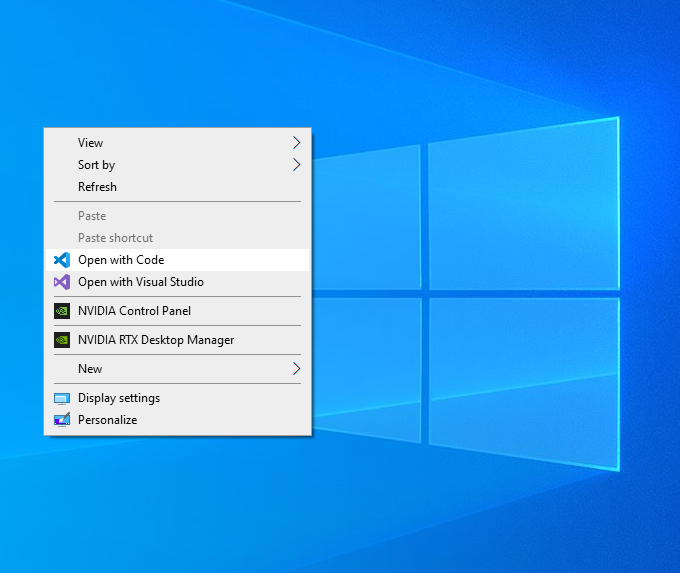


Figure 1: Open VScode in Desktop

In the vscode go to Terminal > New Terminal. This will open a new terminal on the bottom in your Desktop location. Click on the drop-down sign with ‘+’ from the top right corner of the Terminal (Figure 2) and get the Git Bash Terminal. Let’s clone the repository by typing “ git clone <https://github.com/seenaimul/csi-quizapp.git> ” and press enter. GitHub project will be cloned in your local working directory.

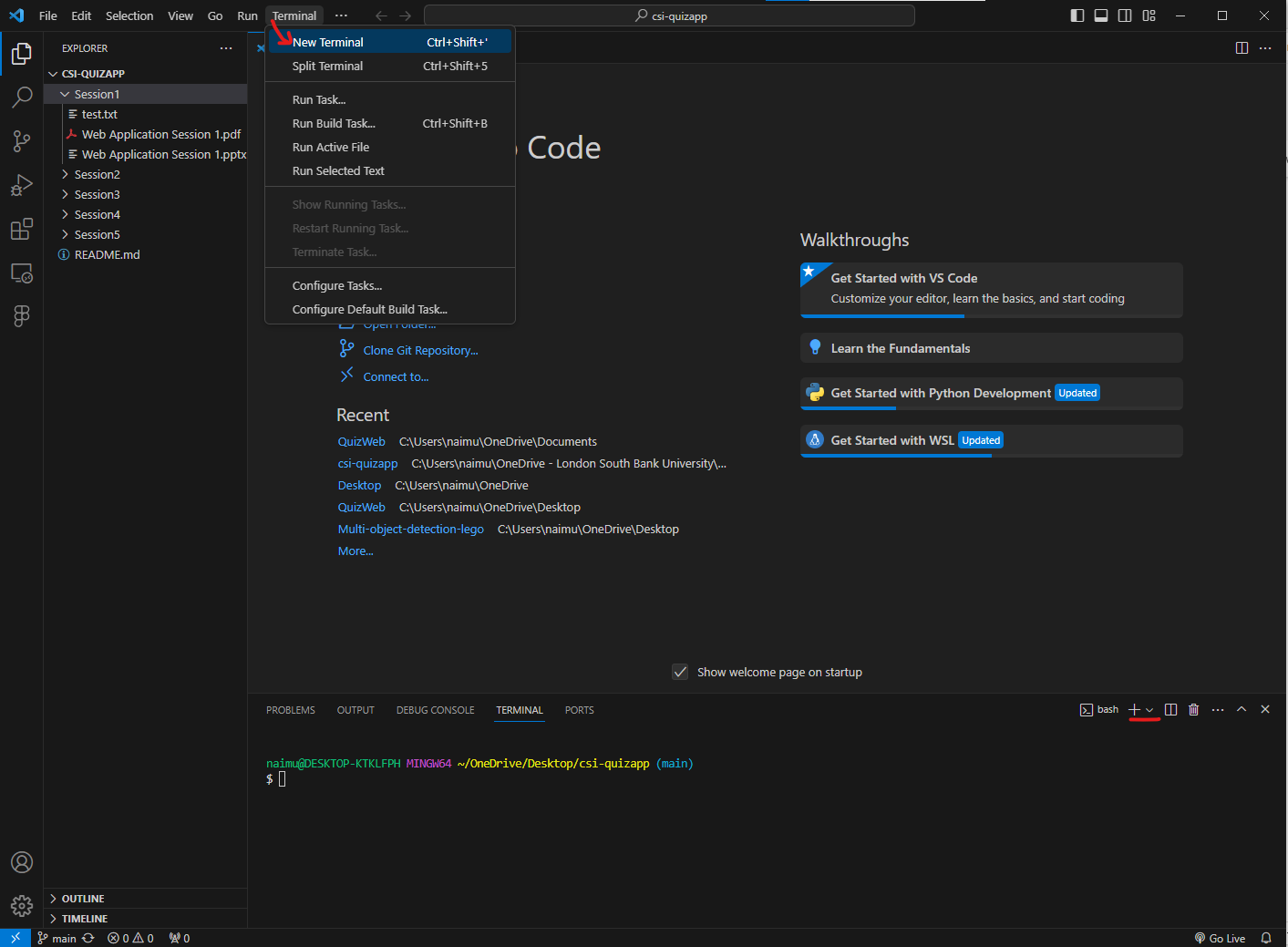


Figure 2: Getting Bash Terminal

Now in the VScode go to File>Open Folder to open the csi-quizapp folder and let’s also get the bash terminal again.

At this point, you should be seeing the Figure 3

A screenshot of a computer

Description automatically generated

Figure 3: Project is Ready for Session 2

Let’s create a folder called ‘images’ and 3 files ‘index.html, style.css and script.js’ under the CSI-QUIZAPP folder.

We will start with index.html file, a typical naming convention of the homepage html file the webservers look for while serving a web service.

Lets add the code below in the index.html file. Right click on the page and click on the ’Open with Live Server’ to load the web page into your default browser.

|  |
| --- |
| <!DOCTYPE html> <!-- This tells the browser that the document type is html -->  <html> <!-- Root element/tag of html page -->  <head> <!-- head contains the meta information; Doesn't get displayed-->  <title>Title of the page</title>  </head>  <body> <!-- contains the visible area of the page-->  <h1> Wel come to CSI </h1>  <p> Well Done. You just loaded your index.html file</p>  </body>  </html> |

A screenshot of a computer

Description automatically generated

Figure 4: Load web page

Save the file and go to the terminal in the IDE and type “ **git add .** “ to add changes in the staging area/index then “ **git commit -m “commit message”**  “to add commit message which allows git to save the snapshot of changes. Now do a “ **git push** “ to send these changes from local to remote GitHub repository.

A screenshot of a computer program

Description automatically generated

Figure 5: First commit of the day.

You can also host static pages from ‘GitHub Pages’. As you have pushed your code in your GitHub repository, lets have a quick look how we can host. From the browser, go to your GitHub repository, select “Settings” then find “Pages”. Once you click pages, you need to select the branch name which you want to deploy. In our case we have only one branch ‘main’, lets select it and ‘Save’. Give it some time and reload the page, you will see the URL created by GitHub for your project. You can share it to anyone, for example see <https://seenaimul.github.io/csi-quizapp/>

A screenshot of a computer

Description automatically generated

Figure 6: Host webpage on GitHub

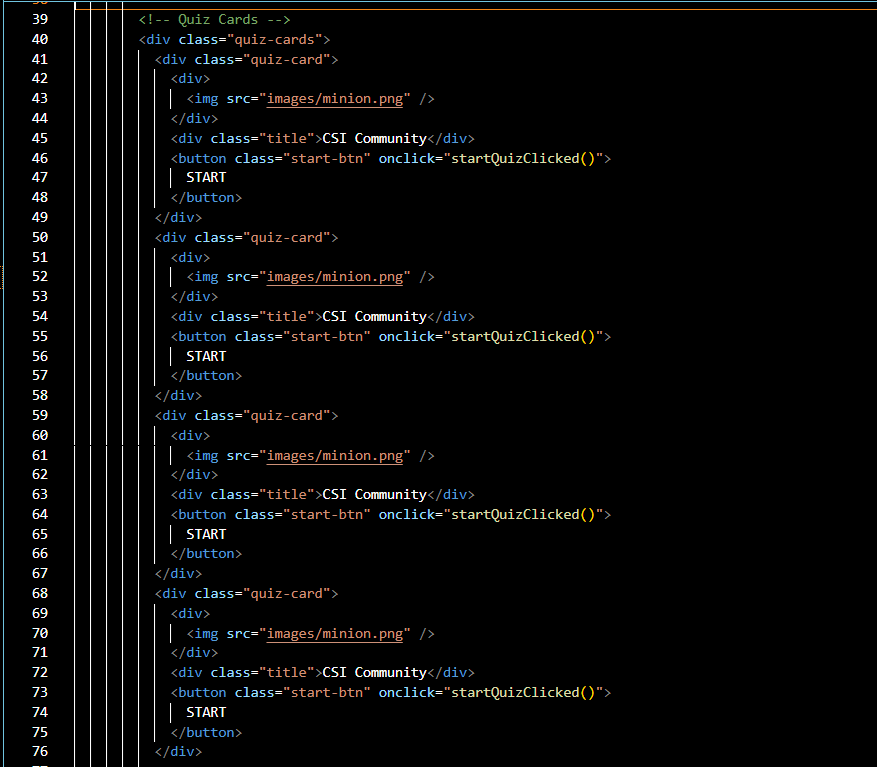
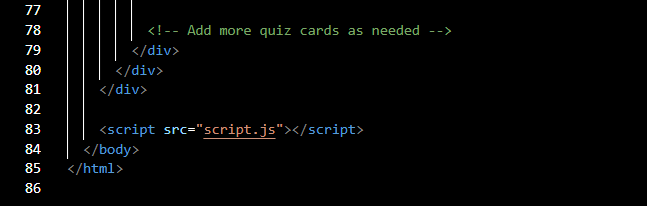
**Don’t forget to explore resources from online to learn more about the tags/elements and get ready build templates. See the README.md file to see few additional resources.**

[W3Schools Online Web Tutorials](https://www.w3schools.com/) -> One of best sources for web development.

1. **Build Landing Page**

**Now we will build our landing page in the index.html file.**



  
  
**Here is a brief explanation of what each part of your code does**

* The <!DOCTYPE html> declaration specifies the HTML version that the web page follows, which is HTML5 in this case.
* The <html> tag encloses the entire HTML document and indicates that it is written in HTML.
* The <head> tag contains information about the web page, such as the title and the link to the external style sheet.
* The <title> tag defines the title of the web page, which is QuizApp in this case. The title is displayed in the browser’s tab or window.
* The <link> tag links the web page to an external style sheet, which is a file that contains the CSS code that defines the appearance and layout of the web page. The rel attribute specifies the relationship between the web page and the linked file, which is “stylesheet” in this case. The href attribute specifies the URL of the linked file, which is “style.css” in this case.
* The <body> tag contains the content of the web page that will be displayed in the browser.
* The <div> tag creates a division or a section in the web page. It is used to group related elements together and apply styles or attributes to them. The class attribute assigns a name to the division, which can be used to refer to it in the style sheet or the script. For example, the <div class="container"> creates a division that contains the entire content of the web page and has a class name of “container”.
* The <div class="header"> creates a division that contains the header section of the web page, which includes the logo, the search box, and the action buttons.
* The <div class="logo"> creates a division that contains the logo of the quiz app, which is a text that says “QuizApp”.
* The <div class="search-box"> creates a division that contains the search box, which is an input field that allows the user to search for a quiz by typing a keyword or a phrase. The <input> tag creates an input field that can accept user input. The type attribute specifies the type of the input field, which is “text” in this case. The placeholder attribute specifies a text that is displayed in the input field when it is empty, which is “Search quiz…” in this case.
* The <div class="action-buttons"> creates a division that contains the action buttons, which are buttons that allow the user to log in, sign up, or join a quiz. The <button> tag creates a button that can perform an action when clicked. The class attribute assigns a style to the button, which can be “login-btn”, “signup-btn”, or “join-quiz-btn” in this case. The onclick attribute specifies a JavaScript function that will be executed when the button is clicked. For example, the <button class="login-btn" onclick="loginClicked()"> creates a button that has a style of “login-btn” and calls the loginClicked function when clicked. The text between the opening and closing <button> tags is the label of the button, which is “LOG IN” in this case.
* The <div class="main"> creates a division that contains the main section of the web page, which includes the welcome section and the quiz cards.
* The <div class="welcome-container"> creates a division that contains the welcome section, which includes a welcome message and a sign up button.
* The <div class="welcome"> creates a division that contains the welcome message, which includes a title and a description. The <div class="title"> creates a division that contains the title of the welcome message, which is a text that says “Welcome to QuizApp”. The <div class="description"> creates a division that contains the description of the welcome message, which is a text that says “Your one-stop source to learn, assess with fun”.
* The <button class="signup-btn-grey" onclick="signupClicked()"> creates a button that has a style of “signup-btn-grey” and calls the signupClicked function when clicked. The text between the opening and closing <button> tags is the label of the button, which is “Sign up now” in this case.
* The <div class="quiz-cards"> creates a division that contains the quiz cards, which are cards that display the quizzes available for the user to take. Each quiz card has an image, a title, and a start button.
* The <div class="quiz-card"> creates a division that contains a single quiz card. It has the following elements:
  + The <div> tag creates a division that contains the image of the quiz card. The <img> tag creates an image that shows a picture related to the quiz. The src attribute specifies the source of the image, which is “images/minion.png” in this case. The alt attribute provides an alternative text for the image, which is “Minion” in this case.
  + The <div class="title"> creates a division that contains the title of the quiz card, which is a text that says “CSI Community” in this case.
  + The <button class="start-btn" onclick="startQuizClicked()"> creates a button that has a style of “start-btn” and calls the startQuizClicked function when clicked. The text between the opening and closing <button> tags is the label of the button, which is “START” in this case.
* The <script> tag contains the JavaScript code that handles the logic and functionality of the web page. The src attribute specifies the URL of the external script file, which is “script.js” in this case.

Lets add each block of the division container (<div> tag/element) into the body section of index.html. Run section by section and if anything is wrong, cross check with the code snippets above.

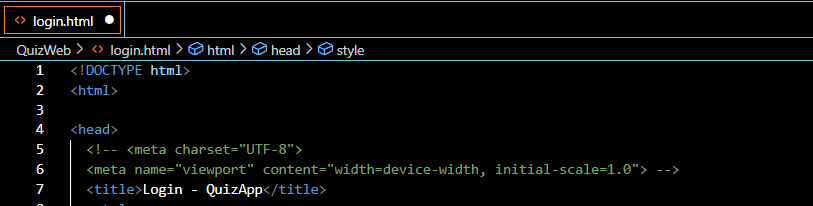
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>QuizApp</title>  <link rel="stylesheet" href="style.css" />  </head>  <body>  <div class="container">  <div class="main">  </div>  </div>  <script src="script.js"></script>  </body>  </html> |

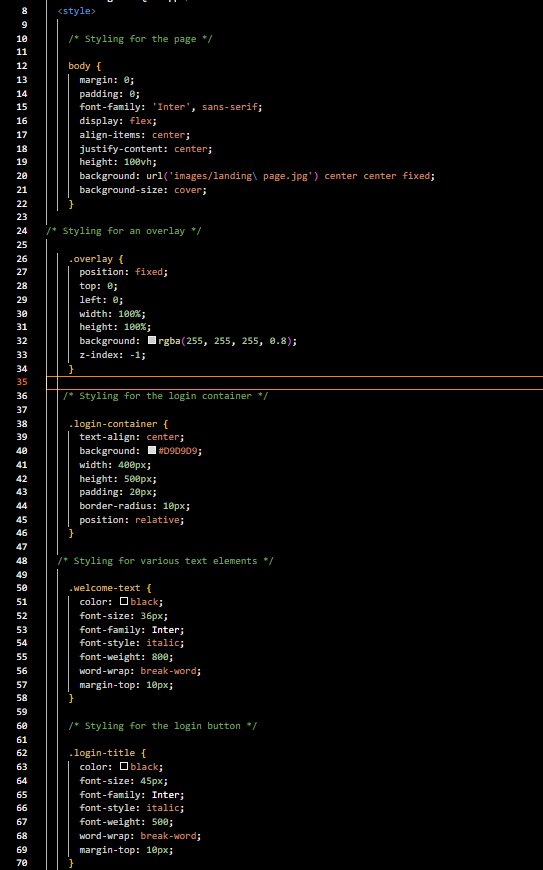
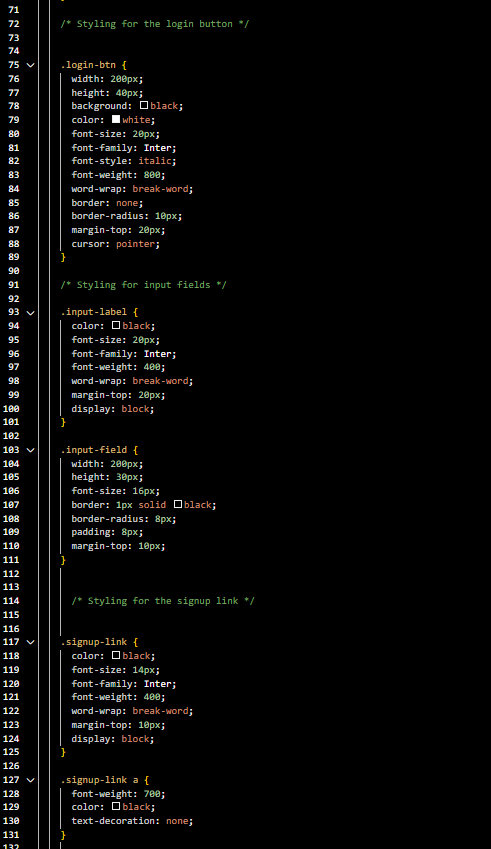
|  |
| --- |
| <!-- Header Section -->    <div class="header">  <div class="logo">QuizApp</div>  <div class="search-box">  <input type="text" placeholder="Search quiz..." />  </div>  <div class="action-buttons">  <button class="login-btn" onclick="loginClicked()">LOG IN</button>  <button class="signup-btn" onclick="signupClicked()">SIGN UP</button>  <button class="join-quiz-btn" onclick="joinQuizClicked()">  Join Quiz  </button>  </div>  </div> |

|  |
| --- |
| <!-- Welcome Section -->  <div class="welcome-container">  <div class="welcome">  <div class="title">Welcome to QuizApp</div>  <div class="description">  Your one-stop source to learn, assess with fun  </div>  </div>  <button class="signup-btn-grey" onclick="signupClicked()">  Sign up now  </button>  </div> |

|  |
| --- |
| <!-- Quiz Cards -->  <div class="quiz-cards">  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <!-- Add more quiz cards as needed -->  </div> |

**3. Build Login Page**

**Now we will build our Login page in the login.html file.**  
  


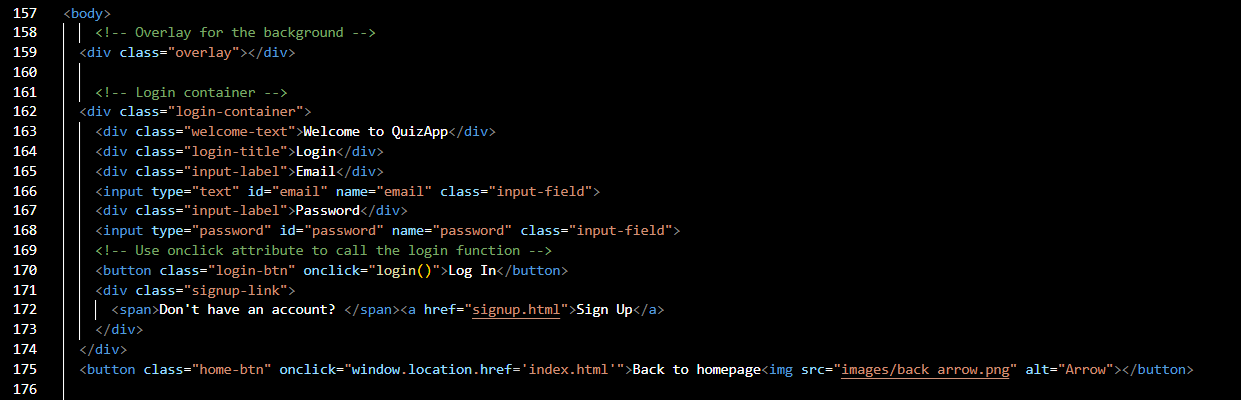
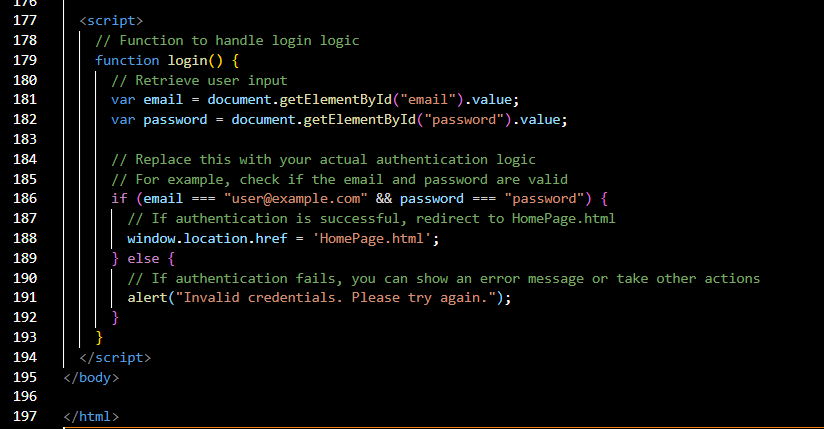
**First, we’ll give title of the web page between <title> </title> tag**  
  
   
  
  
  
  
  
  
  
   
  
**this way of CSS styling is called internal styles, there’s another way of writing CSS styles for example for the login page we can directly write the styles in the HTML tags using inline styles**   
  
Inline styles have the advantage of being directly applied to individual elements, but they can make the HTML code less clean and can be harder to maintain, especially when styling multiple elements. while With internal styles, you can define styles for multiple elements at once using CSS selectors.  
  
In summary:

* **Inline Styles:**
  + Applied directly within HTML tags using the **style** attribute.
  + Styling is specific to individual elements.
  + Can make the HTML code less clean and harder to maintain.
* **Internal Styles:**
  + Defined in the **<style>** tag within the **<head>** section.
  + Allows styling multiple elements using CSS selectors.
  + Promotes better separation of concerns between HTML and CSS.

**In these figures we are styling our web page**   
  
styles that we are performing here are as follows,

* Setting page properties like margin, padding, and font
* Creating a transparent overlay covering the entire page
* Styling the main login container
* Styling the welcome text
* Styling the Login title and button
* Styling the Input fields

etc.  
  
**Now let's move onto the body of login.html web page**

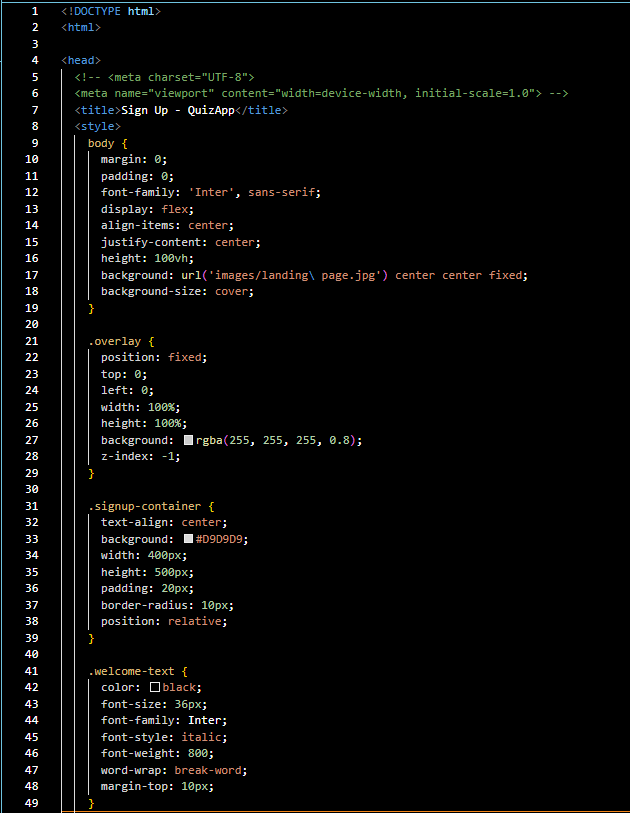
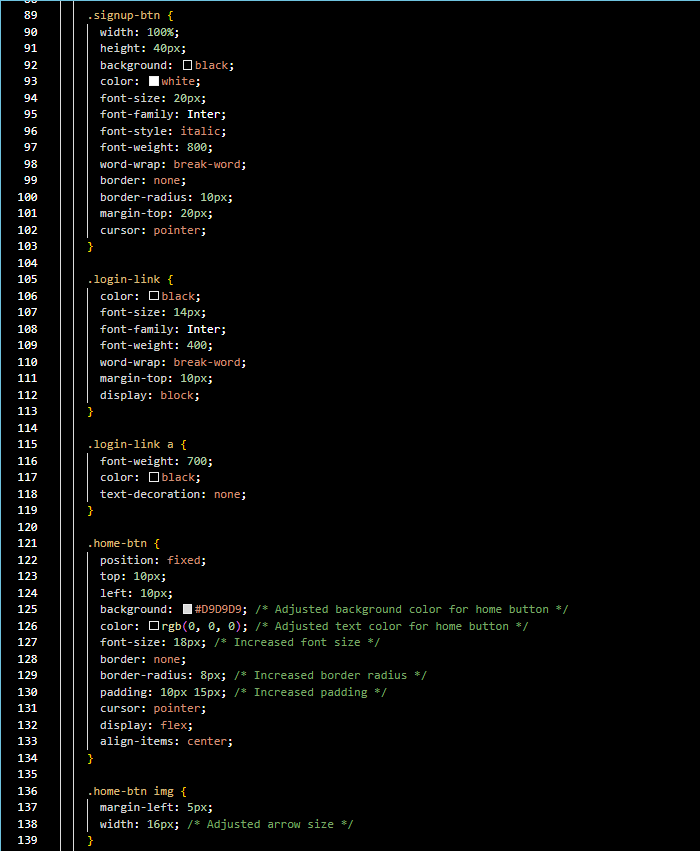
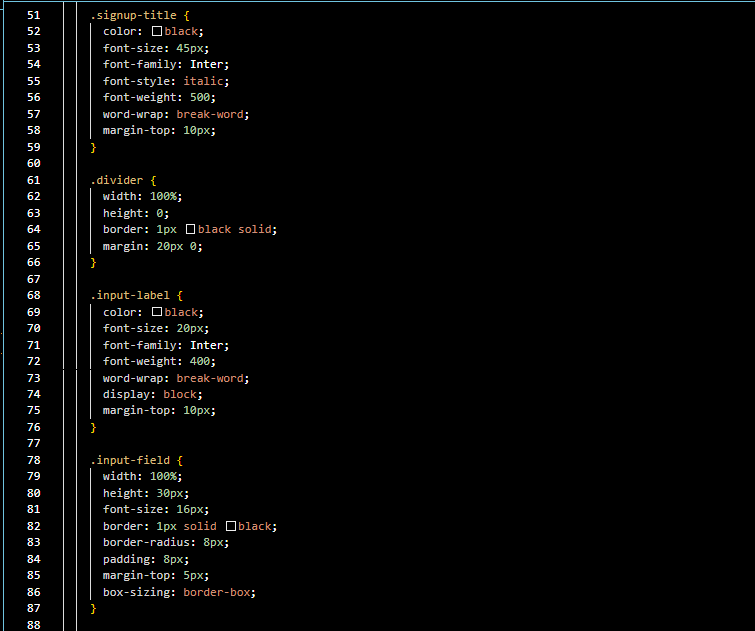
  


**In these figures we are writing HTML and JavaScript code that allows user to log in to the Quiz App.**

* The <body> tag contains the content of the web page that will be displayed in the browser.
* The <div class="overlay"> creates a transparent overlay that covers the entire page, giving it a dark background.
* The <div class="login-container"> creates a container that holds the elements related to the login functionality, such as the welcome text, the input fields, and the buttons.
* The <div class="welcome-text"> displays a text that greets the user and invites them to log in to the quiz app.
* The <div class="login-title"> displays a text that indicates the purpose of the container, which is to log in.
* The <div class="input-label"> displays a text that labels the input field below it, either email or password.
* The <input type="text" id="email" name="email" class="input-field"> creates a text input field that allows the user to enter their email address. It has an id attribute that identifies it as “email”, a name attribute that assigns it a name of “email”, and a class attribute that assigns it a style of “input-field”.
* The <input type="password" id="password" name="password" class="input-field"> creates a password input field that allows the user to enter their password. It has similar attributes as the email input field, except that the id and name are “password” and the type is “password”, which hides the characters entered by the user.
* The <button class="login-btn" onclick="login()"> creates a button that allows the user to submit their login credentials. It has a class attribute that assigns it a style of “login-btn” and an onclick attribute that calls the login function when the button is clicked.
* The <div class="signup-link"> creates a container that holds a text and a link related to the signup functionality, in case the user does not have an account yet.
* The <span> tag creates a text that says “Don’t have an account?”.
* The <a href="signup.html"> creates a link that redirects the user to the signup page when clicked. It has an href attribute that specifies the URL of the signup page, which is “signup.html”.
* The <button class="home-btn" onclick="window.location.href='index.html'"> creates a button that allows the user to go back to the homepage of the quiz app. It has a class attribute that assigns it a style of “home-btn” and an onclick attribute that changes the window location to the homepage URL, which is “index.html”.
* The <img src="images/back arrow.png" alt="Arrow"> creates an image that shows a back arrow icon on the button. It has a src attribute that specifies the source of the image, which is “images/back arrow.png” and an alt attribute that provides an alternative text for the image, which is “Arrow”.
* The <script> tag contains the JavaScript code that handles the login logic of the web page.
* The function login() defines a function that performs the login operation when called.
* The var email = document.getElementById("email").value; declares a variable named email and assigns it the value of the email input field, which is retrieved by using the document.getElementById method and the id attribute of the input field.
* The var password = document.getElementById("password").value; declares a variable named password and assigns it the value of the password input field, which is retrieved in a similar way as the email variable.
* The if (email === "[user@example.com](mailto:user@example.com)" && password === "password") checks if the email and password entered by the user match the predefined values, which are “[user@example.com](mailto:user@example.com)” and “password”. This is a simple example of authentication logic, but in a real application, you would need to use more secure and complex methods, such as hashing, encryption, or database queries.
* The window.location.href = 'HomePage.html'; changes the window location to the homepage URL, which is “HomePage.html”. This means that the user will be redirected to the homepage of the quiz app if the authentication is successful.
* The else clause handles the case when the authentication fails, meaning that the email and password entered by the user do not match the predefined values.
* The alert("Invalid credentials. Please try again."); displays an alert message to the user, informing them that their credentials are invalid and asking them to try again. This is a simple example of error handling, but in a real application, you might want to use more user-friendly and informative methods, such as displaying a message on the web page, highlighting the input fields, or providing hints or suggestions.

**4. Build Sign up Page**

**Now we will build our Login page in the login.html file**

  
‘

**CSS Styling:**

* The **.signup-container** style sets the appearance of the container for the signup form, including background color, width, height, padding, and border radius.
* Various styles such as **.welcome-text**, **.signup-title**, and others are used to define the appearance of specific text elements, adjusting properties like color, font size, style, and weight.
* The **.divider** style creates a horizontal line to visually separate sections.
* Styles for input fields (**input-field**), signup button (**signup-btn**), login link (**login-link**), and home button (**home-btn**) are also defined.
* The **.home-btn img** style adjusts the appearance of the image inside the home button.



**Here we are creating a Sign-Up form to register new users**

Here is a brief explanation of what each part of your code does:

* The <body> tag contains the content of the web page that will be displayed in the browser.
* The <div class="overlay"> creates a transparent overlay that covers the entire page, giving it a dark background.
* The <div class="signup-container"> creates a container that holds the elements related to the sign up functionality, such as the welcome text, the input fields, and the buttons.
* The <div class="welcome-text"> displays a text that greets the user and invites them to sign up to the quiz app.
* The <div class="signup-title"> displays a text that indicates the purpose of the container, which is to sign up.
* The <div class="divider"> creates a horizontal line that separates the title from the input fields.
* The <div class="input-label"> displays a text that labels the input field below it, either first name, last name, email, or password.
* The <input type="text" id="firstName" name="firstName" class="input-field"> creates a text input field that allows the user to enter their first name. It has an id attribute that identifies it as “firstName”, a name attribute that assigns it a name of “firstName”, and a class attribute that assigns it a style of “input-field”.
* The <input type="text" id="lastName" name="lastName" class="input-field"> creates a text input field that allows the user to enter their last name. It has similar attributes as the first name input field, except that the id and name are “lastName”.
* The <input type="text" id="email" name="email" class="input-field"> creates a text input field that allows the user to enter their email address. It has similar attributes as the first name input field, except that the id and name are “email”.
* The <input type="password" id="password" name="password" class="input-field"> creates a password input field that allows the user to enter their password. It has similar attributes as the first name input field, except that the id and name are “password” and the type is “password”, which hides the characters entered by the user.
* The <button class="signup-btn"> creates a button that allows the user to submit their sign up information. It has a class attribute that assigns it a style of “signup-btn”. The text between the opening and closing <button> tags is the label of the button, which is “SIGN UP” in this case.
* The <div class="login-link"> creates a container that holds a text and a link related to the login functionality, in case the user already has an account.
* The <span> tag creates a text that says “Already a member?”.
* The <a href="login.html"> creates a link that redirects the user to the login page when clicked. It has an href attribute that specifies the URL of the login page, which is “login.html”.
* The <button class="home-btn" onclick="window.location.href='index.html'"> creates a button that allows the user to go back to the homepage of the quiz app. It has a class attribute that assigns it a style of “home-btn” and an onclick attribute that changes the window location to the homepage URL, which is “index.html”.
* The <img src="images/back arrow.png" alt="Arrow"> creates an image that shows a back arrow icon on the button. It has a src attribute that specifies the source of the image, which is “images/back arrow.png” and an alt attribute that provides an alternative text for the image, which is “Arrow”.

**5. Build Home Page after user login session**





The homepage.html is almost same as index.htm. The primary difference between the two HTML files lies in the content and functionality displayed to the user, based on their login state. Let's break down the differences:

`index.html` (

1. Action Buttons:

- Three action buttons: "LOG IN," "SIGN UP," and "Join Quiz" are displayed in the header.

2. Welcome Section:

- Contains a welcome message and a "Sign up now" button for users who have not yet signed up.

3. Quiz Cards:

- Displays multiple quiz cards with a common layout (image, title, and "START" button)

4. Script:

- The JavaScript file (`script.js`) likely contains functions like `loginClicked()`, `signupClicked()`, and `joinQuizClicked()` to handle button clicks and initiate corresponding actions.

`homepage.html` (After Login)

1. \*\*Action Buttons:\*\*

- Only two action buttons: "Join Quiz" and "Create Quiz" are displayed in the header.

- The "Create Quiz" button allows users to create a quiz after logging in.

2. Welcome Section:

- Similar welcome message but without the "Sign up now" button since the user is already logged in.

3. Quiz Cards:

- The quiz cards remain the same as on the landing page.

4. \*\*Script:\*\*

- Likely contains functions like `joinQuizClicked()` and `createQuizClicked()` to handle button clicks and initiate corresponding actions.

General Notes:

- Both pages include a search box in the header for users to search for quizzes.

- The CSS styling is likely shared between the two pages as they both reference the same external stylesheet (`style.css`).

- The logic for handling user interactions and navigating between pages is implemented in the JavaScript file (`script.js`), which is shared between the pages.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <!-- <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0"> -->  <title>QuizApp</title>  <link rel="stylesheet" href="style.css" />  </head>  <body>  <div class="container">  <!-- Header Section -->  <div class="header">  <div class="logo">QuizApp</div>  <div class="search-box">  <input type="text" placeholder="Search quiz..." />  </div>  <div class="action-buttons">  <button class="join-quiz-btn" onclick="joinQuizClicked()">  Join Quiz  </button>  <button class="create-quiz-btn" onclick="createQuizClicked()">  Create Quiz  </button>  </div>  </div>  <div class="main">  <!-- Welcome Section -->  <div class="welcome-container">  <div class="welcome">  <div class="title">Welcome to QuizApp</div>  <div class="description">  Your one-stop source to learn, assess with fun  </div>  </div>  </div>  <!-- Quiz Cards -->  <div class="quiz-cards">  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <div class="quiz-card">  <div>  <img src="images/minion.png" />  </div>  <div class="title">CSI Community</div>  <button class="start-btn" onclick="startQuizClicked()">  START  </button>  </div>  <!-- Add more quiz cards as needed -->  </div>  </div>  </div>  <script src="script.js"></script>  </body>  </html> |

In summary, the `index.html` page is designed for users who haven't logged in yet, providing options to log in, sign up, and join quizzes. The `homepage.html` page, on the other hand, is tailored for users who are already logged in, focusing on joining or creating quizzes.